# Curriculum Vitae

Arnoldus Franciscus van der Stappen

| | |
|---|---|
| 4 oktober 1964 | geboren te Eindhoven |
| 1977-1983 | VWO aan het Bisschop Bekkerscollege te Eindhoven |
| 1983-1988 | studie Informatica aan de Technische Universiteit Eindhoven<br>afstudeerverslag: *'Distributed data structures for sparse linear algebra'* |
| 1988-1990 | assistent-in-opleiding in de post-doctorale ontwerpersopleiding Technische Informatica aan het Instituut Vervolgopleidingen van de Technische Universiteit Eindhoven<br>projectverslag: *'Planning and scheduling in crude oil refineries'* (ISBN 90-5282-083-X) |
| 1990-1994 | onderzoeker-in-opleiding aan de Vakgroep Informatica van de Universiteit Utrecht, in dienst van de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (N.W.O.) |
| 1994 | toegevoegd onderzoeker aan de Vakgroep Wiskunde van de Universiteit Utrecht |

Veel praktische motion planning problemen combineren een niet te grote robot met een zekere vetheid van de aanwezige obstakels, zodat vetheid ook in motion planning een waardevol begrip is.

De bovengenoemde omstandigheden leiden tot een verlaging van de rekentijd van een aantal bestaande cel-decompositie- en retractiemethoden. De kern van de efficiëntiewinst voor deze methoden ligt in een lage obstakeldichtheid in de werkruimte, die een direct gevolg is van de vetheid van de obstakels. De eigenschap vormt de basis voor een nieuwe algemene aanpak, of *paradigma*, voor motion planning problemen tussen vette obstakels. Het paradigma volgt de cel-decompositie-aanpak.

Het gepresenteerde paradigma voor motion planning tussen vette obstakels is toepasbaar op problemen waarvoor de werkruimte een projectieve deelruimte is van de configuratieruimte. Wanneer de robot vrij beweegt in de werkruimte, en dus niet is verankerd, dan zal in het algemeen aan deze voorwaarde voldaan zijn. De aanpak reduceert het probleem van het vinden van een cel-decompositie van de vrije ruimte tot het probleem van het vinden van een verdeling van de werkruimte met de vette obstakels die aan bepaalde voorwaarden voldoet. Een aantal uniforme stappen berekenen vervolgens een cel-decompositie van de vrije ruimte uit de verdeling van de werkruimte. Het aantal subcellen is direct afhankelijk van de grootte van de werkruimte-verdeling. Optimale verdelingen blijken te bestaan voor motion planning problemen in het vlak, voor motion planning in een drie-dimensionale ruimte met obstakels van vergelijkbare grootte, en voor motion planning op een werkvloer in een drie-dimensionale werkruimte met obstakels. De inpassing van de verdeling en de berekening ervan in het paradigma resulteert in zeer efficiënte algoritmen voor de betreffende problemen. Goede verdelingen en dus efficiënte algoritmen bestaan ook voor motion planning problemen in een drie-dimensionale ruimte met polyhedrale en willekeurige obstakels van onbeperkte afmetingen. Verbeteringen van deze laatste resultaten lijken echter mogelijk. De efficiëntie van elk algoritme dat volgt uit het paradigma is onafhankelijk van het aantal vrijheidsgraden van de robot (ofwel de dimensie van de configuratieruimte), in tegenstelling tot de meeste andere exacte motion planning algoritmen.

Naast motion planning besteedt het proefschrift ook aandacht aan de rol van vetheid in twee kernproblemen in de computationele geometrie: *point location* en *range searching*. Het point location probleem vraagt om, gegeven een aantal niet-snijdende objecten, voor een willekeurig punt te rapporteren welk object het punt bevat of om te concluderen dat geen enkel object het punt bevat. Het range searching probleem vraagt om voor een willekeurige regio de verzameling doorsneden objecten te rapporteren. Het proefschrift laat zien dat, wanneer de objecten vet zijn, één enkele datastructuur volstaat om beide problemen efficiënt op te lossen. De oplossing werkt in willekeurige dimensie.

In de huidige praktijk worden vrijwel altijd *benaderende methoden* gebruikt om het motion planning probleem op te lossen. Benaderende methoden verwerken de vrije ruimte tot een structuur die de vrije ruimte benadert. Ze vinden in veel gevallen snel een pad voor de robot, doch in sommige lastige gevallen zullen ze er niet in slagen om een oplossing te vinden. Sinds een aantal jaren bestaan er ook een aantal *exacte methoden*. Deze methoden, die hun oorsprong voornamelijk in de computationele geometrie gemeenschap hebben, vinden gegarandeerd een pad wanneer er een pad bestaat. Dit proefschrift concentreert zich op exacte methoden.

Exacte methoden kunnen op grond van hun aanpak grofweg in twee categorieën worden ingedeeld: *cel-decompositiemethoden* en *retractiemethoden*. Cel-decompositiemethoden verdelen de vrije ruimte in eenvoudige *subcellen*. Deze subcellen vormen de knopen van een graaf. Twee knopen van de graaf zijn met elkaar verbonden wanneer de overeenkomstige subcellen aan elkaar grenzen. De aanpak reduceert het motion planning probleem tot het vinden van een pad in de graaf. Door de vereiste eenvoud van de subcellen is het aantal benodigde subcellen om de vrije ruimte te verdelen in hoge mate afhankelijk van de complexiteit van de ruimte. Retractiemethoden trachten de structuur van de vrije ruimte vast te leggen in een een-dimensional netwerk van curves in diezelfde ruimte: het *wegennet*. Dit houdt in dat elke vrije configuratie via een eenvoudig pad verbonden dient te zijn met het wegennet, en dat alle curves in een samenhangende component van de vrije ruimte met elkaar een samenhangende component van het wegennet vormen. Deze voorwaarden reduceren het motion planning probleem wederom tot het vinden van een pad in een graaf, namelijk het wegennet. De beide eisen aan het wegennet maken de grootte van het wegennet sterk afhankelijk van de complexiteit van de vrije ruimte. Uiteraard beïnvloedt de grootte van de berekende structuur (graaf) de rekentijd van de exacte algoritmen.

De conclusie uit het voorgaande is dat de efficiëntie van exacte algoritmen sterk afhangt van de complexiteit van de vrije ruimte. Aangezien die complexiteit in theorie erg hoog kan zijn, lijken exacte methoden ongeschikt voor toepassing in praktijksituaties. De literatuur toont echter dat de omstandigheden (dat wil zeggen vorm en posities van de obstakels en vorm van de robot) die leiden tot de hoge complexiteiten vaak een kunstmatig karakter hebben en vrijwel nooit voorkomen in praktische motion planning problemen. In praktijkgevallen zal de complexiteit van de vrije ruimte ver beneden de theoretische grenzen blijven. Voor dergelijke gevallen zal de toepassing van exacte algoritmen dan wellicht realistisch worden. Een studie naar milde voorwaarden die een bewijsbaar lage complexiteit van de vrije ruimte tot gevolg hebben is daarom van groot belang voor de toepasbaarheid van exacte methoden.

Dit proefschrift toont dat de combinatie van *vette* obstakels en een niet al te grote robot leidt tot een drastische reductie van de complexiteit van de vrije ruimte. Een vet obstakel is een obstakel dat niet lang en dun is en ook niet dergelijke delen heeft. Vetheid vormt in een aantal problemen uit de computationele geometrie een realistische aanname die resulteert in lage complexiteiten en efficiënte algoritmen.

# Samenvatting

Het groeiende aantal toepassingsgebieden voor robots en de toenemende verscheidenheid in hun taken vereist steeds meer autonomie van de robots. Een autonome robot accepteert complexe taken en voert die uit zonder hulp van zijn omgeving. Een voor de hand liggende opdracht voor zo'n autonome robot is om van een beginpositie naar een eindpositie te bewegen, waarbij botsing met de aanwezige obstakels vermeden dient te worden. Het vinden van een dergelijk pad wordt het *motion planning* probleem genoemd.

Het motion planning probleem wordt in het algemeen opgelost in de *configuratieruimte*. Dit is de ruimte van de representaties van alle mogelijke robotposities, ofwel *configuraties*. Het aantal *vrijheidsgraden* van de robot bepaalt de dimensie van de configuratieruimte. Een configuratie is *vrij* wanneer de robot in de overeenkomstige positie geen enkel obstakel doorsnijdt. Indien de robot in een positie een of meerdere obstakels doorsnijdt, dan is de betreffende configuratie *verboden*. De *vrije ruimte* (FP) is de deelruimte van de configuratieruimte die bestaat uit alle vrije configuraties. Het oplossen van het motion planning probleem in de configuratieruimte komt neer op het vinden van een continue curve die de beginconfiguratie met de eindconfiguratie verbindt en bovendien volledig is bevat in de vrije ruimte. De continue curve in de configuratieruimte komt overeen met een botsingsvrij pad voor de robot in zijn werkruimte. Het is tamelijk eenvoudig om in te zien dat de moeilijkheid van het vinden van een continue curve in de vrije ruimte (ofwel, het oplossen van het motion planning probleem) sterk afhankelijk is van de complexiteit (beschrijvingsgrootte) van die ruimte. Op haar beurt hangt de complexiteit van de vrije ruimte in hoge mate af van het aantal meervoudige contacten van de robot met de obstakels. Helaas kan in theorie het aantal meervoudige contacten, en dus de complexiteit van de vrije ruimte, erg hoog zijn.

Motion planning algoritmen trachten een continue curve in de vrije ruimte te vinden. De vrije ruimte is daarbij indirect gegeven door middel van de robot en de obstakels. Aangezien zo'n continue curve door het complexe karakter van de vrije ruimte (zelfs voor eenvoudige motion planning problemen) onmogelijk direct te bepalen is, is verdere verwerking van de vrije ruimte noodzakelijk. De verschillende motion planning algoritmen onderscheiden zich door de manier waarop zij de vrije ruimte verwerken tot een structuur waarmee men in staat is om op efficiënte wijze een pad te vinden tussen twee configuraties.

# Acknowledgements

I thank all people that have contributed, directly or indirectly, in making this thesis into what it is. In particular, I wish to thank

my supervisor Mark Overmars; he has been available for inspiring discussions at almost any moment; it has been a pleasure working for *and* with him;

the present and former members of the Vakgroep Informatica at Utrecht University and particularly those in the field of computational geometry for creating a pleasant research environment;

the people at the School of Mathematical Sciences at Tel Aviv University, and especially Dan Halperin (currently at Stanford University), for making my stay in Tel Aviv in March and April 1992 both instructive and enjoyable;

Marko de Groot and Maarten Pennings for being pleasant roommates during different periods of my stay at the Vakgroep Informatica;

the members of the reviewing committee: Prof. J. van Leeuwen (Utrecht University), Prof. F. Groen (University of Amsterdam), Prof. C. Erkelens (Utrecht University), and Dr. J.-D. Boissonnat (INRIA - Sophia Antipolis, France) for reviewing this thesis;

my parents for support and for their interest in my work;

and, finally, Petra, for support and care; spending almost 24 hours a day together turned out to be so easy!

# Index

# Bibliography

Chapter 2 introduces a universal notion of fatness. A preliminary abstract on the complexity of the free space for motion planning amidst fat obstacles [94] used a different notion of fatness, which is introduced as 'thickness' in Chapter 2. The failure to prove a low free space complexity for motion planning problems involving *non-convex* 'thick' obstacles has led to the current, slightly more restrictive, fatness notion.

Chapter 3 generalizes earlier reported results [74] on point location and range searching among sets of fat objects in 2- and 3-space to spaces of arbitrary dimension $d$, by applying different tools. The general results are reported in [75].

The results of Chapter 4 on the complexity of the free space for a robot moving amidst fat obstacles appeared in [95].

The $O(n \log n)$ time bound on the running time of a fat version of the Piano Movers' algorithm [84] in Section 5.2 improves the earlier reported bound of $O(n \log^2 n)$ in both [96] and [97]. The basis of the gain lies in a more efficient way of searching for pairs of neighboring obstacle features: through a plane sweep of 'enveloped' features [93] rather than by orthogonal range searching (or windowing) among obstacle edges [72].

An extended abstract of the contents of Chapters 6 and 7 can be found in [98]. A full version of the paper is in preparation.

[99] W. Walter, *Analysis II (Grundlagen Mathematik Bd. 4)*, Springer-Verlag, Berlin (1990) (in German).

[100] C. Wentink and O. Schwarzkopf, Motion planning for vacuum cleaner robots, *Proc. 6th Canadian Conf. on Computational Geometry* (1994), pp. 51-56.

[101] A. Wiernik and M. Sharir, Planar realizations of nonlinear Davenport-Schinzel sequences by segments, *Discrete & Computational Geometry* **3** (1988), pp. 15-47.

[102] C.-K. Yap, Algorithmic motion planning, in *Advances in Robotics, Volume I: Algorithmic and Geometric Aspects of Robotics* (J.T. Schwarz and C.-K. Yap Eds.), Lawrence Erlbaum Associates, Hillsdale NJ (1987), pp. 95-143.

[103] D. Zhu and J.-C. Latombe, New heuristic algorithms for efficient hierarchical path planning, *IEEE Transactions on Robotics and Automation* **7** (1991), pp. 9-20.

[87] J.T.Schwartz and M. Sharir, On the piano movers' problem: V. The case of a rod moving in three-dimensional space amidst polyhedral obstacles, *Communications on Pure and Applied Mathematics* **37** (1984), pp. 815-848.

[88] J.T.Schwartz and M. Sharir, Efficient motion planning algorithms in environments of bounded local complexity, Report 164, Department of Computer Science, Courant Inst. Math. Sci., New York NY (1985).

[89] M. Sharir, Efficient algorithms for planning purely translational collision-free motion in two and three dimensions, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Raleigh NC (1987), pp. 1326-1331.

[90] M. Sharir, Davenport-Schinzel sequences and their geometric applications, *Theoretical Foundations of Computer Graphics and CAD* (R.A. Earnshaw Ed.), NATO ASI Series, F40, Springer-Verlag, Berlin (1988), pp. 253-278.

[91] M. Sharir and E. Ariel-Sheffi, On the piano movers' problem: IV. Various decomposable two-dimensional motion planning problems, *Communications on Pure and Applied Mathematics* **37** (1984), pp. 479-493.

[92] M. Sharir and M.H. Overmars, A simple output-sensitive algorithm for hidden surface removal, *ACM Transactions on Graphics* **11** (1992), pp. 1-11.

[93] S. Sifrony and M. Sharir, A new efficient motion planning algorithm for a rod in two-dimensional polygonal space, *Algorithmica* **2** (1987), pp. 367-402.

[94] A.F. van der Stappen, M.H. Overmars, On the complexity of the free space for motion planning problems, *Proc. of the 13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, July 1991, pp. 140-141.

[95] A.F. van der Stappen, D. Halperin, M.H. Overmars, The complexity of the free space for a robot moving amidst fat obstacles, *Computational Geometry: Theory and Applications* **3** (1993), pp. 353-373.

[96] A.F. van der Stappen, D. Halperin, M.H. Overmars, Efficient algorithms for exact motion planning amidst fat obstacles, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Atlanta GA (1993), Vol. 1, pp. 297-304.

[97] A.F. van der Stappen, The complexity of the free space for motion planning amidst fat obstacles, *Journal of Intelligent & Robotic Systems*, in press (1994) (also: Technical Report RUU-CS-92-31, Dept. of Computer Science, Utrecht University).

[98] A.F. van der Stappen and M.H. Overmars, Motion planning amidst fat obstacles, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 31-40.

[75] M.H. Overmars and A.F. van der Stappen, Range searching and point location among fat objects, Technical Report UU-CS-1994-30, Dept. of Computer Science, Utrecht University (1994).

[76] M.H. Overmars and P. Švestka, A probabilistic learning approach to motion planning, in *The First Workshop on the Algorithmic Foundations of Robotics*, A.K. Peters, Boston (1994).

[77] *The Concise Oxford dictionary*, Oxford University Press, Oxford.

[78] Ph. Pignon, *Structuration de l' espace pour une planification hiérarchisée des trajectoires de robots mobiles*, Ph.D. Thesis, LAAS-CNRS and Université Paul Sabatier de Toulouse, Rapport LAAS N$^o$ 93395 (1993) (in French).

[79] F.P. Preparata and M.I. Shamos, *Computational geometry: an introduction*, Springer Verlag, New York (1985).

[80] F.P. Preparata and R. Tamassia, Efficient spatial point location, *Proc. 1st Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science **382** (1989), pp. 3-11.

[81] J.H. Reif and M. Sharir, Motion planning in the presence of moving obstacles, *Proc. 26th IEEE Symp. on Foundations of Computer Science* (1985), pp. 144-154.

[82] E. Rimon and D.E. Koditschek, The construction of analytic diffeomorphisms for exact robot navigation on star worlds, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Scottsdale AZ (1989), pp. 21-26.

[83] E. Rimon and D.E. Koditschek, Exact robot navigation in geometrically complicated but topologically simple spaces, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Cincinnati OH (1990), pp. 1937-1942.

[84] J.T.Schwartz and M. Sharir, On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal boundaries, *Communications on Pure and Applied Mathematics* **36** (1983), pp. 345-398.

[85] J.T. Schwartz and M. Sharir, On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds, *Advances in Applied Mathematics* **4** (1983), pp. 298-351.

[86] J.T.Schwartz and M. Sharir, On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers, *International Journal of Robotics Research* **2** (1983), pp. 46-75.

[62] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, *Discrete & Computational Geometry* **2** (1987), pp. 9-31.

[63] D. Leven and M. Sharir, On the number of critical free contacts of a convex polygonal object moving in two-dimensional polygonal space, *Discrete & Computational Geometry* **2** (1987), pp. 255-270.

[64] D. Leven and M. Sharir, An efficient and simple motion planning algorithm for a ladder amidst polygonal barriers, *Journal of Algorithms* **8** (1987), pp. 192-215.

[65] J. Matoušek, Efficient partition trees, *Discrete & Computational Geometry* **8** (1992), pp. 315-334.

[66] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete & Computational Geometry* **10** (1993), pp. 157-182.

[67] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl, Fat triangles determine linearly many holes, *SIAM Journal on Computing* **23** (1994), pp. 154-169.

[68] K. Mehlhorn and S. Näher, Dynamic fractional cascading, *Algorithmica* **5** (1990), pp. 215-241.

[69] K. Mulmuley, Randomized multidimensional search trees: further results in dynamic sampling, *Proc. 32nd Symp. on Foundations of Computer Science* (1991), pp. 216-227.

[70] C. Ó'Dúnlaing, M. Sharir, and C.-K. Yap, Retraction: A new approach to motion planning, *Proc. 15th Ann. ACM Symp. on the Theory of Computing* (1983), pp. 207-220.

[71] C. Ó'Dúnlaing and C.-K. Yap, A retraction method for planning the motion of a disc, *Journal of Algorithms* **6** (1985), pp. 104-111.

[72] M.H. Overmars, Range searching in a set of line segments, *Proc. 1st Ann. ACM Symp. on Computational Geometry* (1985), pp. 177-185.

[73] M.H. Overmars, Point location in fat subdivisions, *Information Processing Letters* **44** (1992), pp. 261-265.

[74] M.H. Overmars and A.F. van der Stappen, Range searching and point location among fat objects, *Proc. 2nd Annual European Symp. on Algorithms (ESA'94)* (J. van Leeuwen Ed.), Lecture Notes in Computer Science (1994).

[49] M.J. Katz, M.H. Overmars, and M. Sharir, Efficient hidden surface removal for objects with small union size, *Computational Geometry: Theory and Applications* **2** (1992), pp. 223-234.

[50] Y. Ke and J. O'Rourke, Moving a ladder in three dimensions: upper and lower bounds, *Proc. 3rd Ann. ACM Symp. on Computational Geometry* (1987), pp. 136-145.

[51] K. Kedem and M. Sharir An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space, *Discrete & Computational Geometry* **5** (1990), pp. 43-75.

[52] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research* **5** (1986), pp. 90-98.

[53] P. Khosla and R. Volpe, Superquadric artificial potentials for obstacle avoidance and approach, *Proc. IEEE Int. Conf. on Robotics and Automation*, Philadelphia PA (1988), pp. 1778-1784.

[54] D.G. Kirkpatrick, Optimal search in planar subdivisions, *SIAM Journal on Computing* **12** (1983), pp. 28-35.

[55] D.E. Koditschek, Exact robot navigation by means of potential functions: some topological considerations, *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh NC (1987), pp. 1-6.

[56] D.E. Koditschek, Robot planning and control via potential functions, in *The Robotics Review 1* (O. Khatib, J.J. Craig, and T. Lozano-Pérez Eds.), MIT Press, Cambridge MA (1989).

[57] M. van Kreveld, *New results on data structures in computational geometry*, Ph.D. Thesis, Dept. of Computer Science, Utrecht University (1992).

[58] M. van Kreveld, On fat partitioning, fat covering and the union size of polygons, Technical Report RUU-CS-93-36, Dept. of Computer Science, Utrecht University (1993).

[59] J.-C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston (1991).

[60] C. Laugier and F. Germain, An adaptative collision-free trajectory planner, *Proc. Int. Conf. on Advanced Robotics*, Tokyo, Japan (1985).

[61] K. Leichtweiss, Über die affine Exzentrizität konvexer Körper, *Archiv der Mathematik* **10** (1959), pp. 187-199 (in German).

[36] G.M. Fikhtengol'ts, *The Fundamentals of Mathematical Analysis, Vol. II*, English edition, Pergamon Press (1965).

[37] M.T. Goodrich and R. Tamassia, Dynamic trees and dynamic point location, *Proc. 23rd Ann. ACM Symp. on Theory of Computing* (1991), pp. 523-533.

[38] L.J. Guibas, M. Sharir, and S. Sifrony, On the general motion planning problem with two degrees of freedom, *Discrete & Computational Geometry* **4** (1989), pp. 491-521.

[39] L. Guibas and M. Sharir, Combinatorics and algorithms of arrangements, in *New trends in discrete and computational geometry* (J. Pach Ed.), Springer-Verlag, Berlin (1993), pp. 9-36.

[40] L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Transactions on Graphics* **4** (1985), pp. 74-123.

[41] D. Halperin, *Algorithmic motion planning via arrangements of curves and surfaces*, Ph.D. Thesis, Dept. of Computer Science, Tel Aviv University (1992).

[42] D. Halperin and M.H. Overmars, Spheres, molecules, and hidden surface removal, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 113-122.

[43] D. Halperin, M.H. Overmars, and M. Sharir, Efficient motion planning for an L-shaped object, *SIAM Journal on Computing* **21** (1992), pp. 1-23.

[44] D. Halperin and M. Sharir, Near-quadratic bounds for the motion planning problem for a polygon in a polygonal environment, *Proc. 34th IEEE Symp. on Foundations of Computer Science* (1993), pp. 382-391.

[45] D. Halperin and M. Sharir, Almost tight upper bounds for the single cell and zone problems in three dimensions, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 11-20.

[46] D. Halperin and C.-K. Yap, Combinatorial complexity of translating a box in polyhedral 3-space, *Proc. 9th Ann. ACM Symp. on Computational Geometry* (1993), pp. 29-37.

[47] J. Hershberger, Finding the upper envelope of $n$ line segments in $O(n \log n)$ time, *Information Processing Letters* **33** (1989), pp. 169-174.

[48] S. Kambhampati and L.S. Davis, Multiresolution path planning for mobile robots, *IEEE Transactions on Robotics and Automation* **2** (1986), pp. 135-145.

[22] B. Chazelle, How to search in history, *Information and Control* **64** (1985), pp. 77-99.

[23] B. Chazelle, H. Edelsbrunner, L.J. Guibas, and M. Sharir, A singly-exponential stratification scheme for real semi-algebraic varieties and its applications, *Theoretical Computer Science* **84** (1991), pp. 77-105.

[24] B. Chazelle and J. Friedman, Point location among hyperplanes and unidirectional ray shooting, *Computational Geometry: Theory and Applications* **4** (1994), pp. 53-62.

[25] B. Chazelle and L. Guibas, Fractional cascading I: A data structuring technique, *Algorithmica* **1** (1986), pp. 133-162.

[26] B. Chazelle, M. Sharir, and E. Welzl, Quasi-optimal upper bounds for simplex range searching and new zone theorems, *Algorithmica* **8** (1992), pp. 407-429.

[27] B. Chazelle and E. Welzl, Quasi-optimal range searching in spaces of finite VC-dimension, *Discrete & Computational Geometry* **4** (1989), pp. 467-489.

[28] K.L. Clarkson, New applications of random sampling in computational geometry, *Discrete & Computational Geometry* **2** (1987), pp. 195-222.

[29] R. Cole, Searching and storing similar lists, *Journal of Algorithms* **7** (1986), pp. 202-220.

[30] G.E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Lecture Notes in Computer Science* **33**, Springer-Verlag, New York (1975), pp. 134-183.

[31] T.H. Cormen, C.E. Leierson, and R.L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge MA (1990).

[32] H. Edelsbrunner, *Algorithms in combinatorial geometry*, Springer-Verlag, Berlin (1987).

[33] H. Edelsbrunner, L.J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, *SIAM Journal on Computing* **15** (1986), pp. 317-340.

[34] A. Efrat, G. Rote, and M. Sharir, On the union of fat wedges and separating a collection of segments by a line, *Computational Geometry: Theory and Applications* **3** (1993), pp. 277-288.

[35] B. Faverjon, Object level programming of industrial robots, *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco CA (1986), pp. 1406-1412.

[10] F. Avnaim, J.-D. Boissonnat, and B. Faverjon, A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles, *Proc. Geometry and Robotics Workshop* (J.-D. Boissonnat and J.-P. Laumond Eds.), Lecture Notes in Computer Science **391** (1988), pp. 67-86.

[11] J.M. Bañon, Implementation and extension of the ladder algorithm, *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati OH (1990), pp. 1548-1553.

[12] J. Barraquand and J.-C. Latombe, Robot motion planning: a distributed representation appoach, Report No. STAN-CS-89-1257, Dept. of Computer Science, Stanford University (1989).

[13] J. Barraquand and J.-C. Latombe, A Monte-Carlo algorithm for path planning with many degrees of freedom, *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati OH (1990), pp. 1712-1717.

[14] J.L. Bentley and T.A. Ottman, Algorithms for reporting and counting geometric intersections, *IEEE Transactions on Computers* **28** (1979), pp. 643-647.

[15] M. de Berg, M. de Groot, and M. Overmars, New results on binary space partitions in the plane, *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)* (E.M. Schmidt and S. Skyum Eds.) Lecture Notes in Computer Science **824** (1994), pp. 61-72.

[16] M. de Berg, L.J. Guibas, and D. Halperin, Vertical decompositions for triangles in 3-space, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 1-10.

[17] M. de Berg, M.H. Overmars, and O. Schwarzkopf, Computing and verifying depth orders, *SIAM Journal on Computing* **23** (1994), pp. 437-446.

[18] R.A. Brooks, Solving the find-path problem by good representation of free space, *IEEE Transactions on Systems, Man, and Cybernetics* **13**, pp. 190-197.

[19] R.A. Brooks and T. Lozano-Pérez, A subdivision algorithm in configuration space for findpath with rotation, *Proc. 8th Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany (1983), pp. 799-806.

[20] J.F. Canny, *The complexity of robot motion planning*, MIT Press, Cambridge MA (1988).

[21] B. Chazelle, Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, *SIAM Journal on Computing* **13** (1984), pp. 488-507.

# References

[1] P.K. Agarwal, M.J. Katz, and M. Sharir, Computing depth orders and related problems, *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)* (E.M. Schmidt and S. Skyum Eds.) Lecture Notes in Computer Science **824** (1994), pp. 1-12.

[2] P.K. Agarwal and J. Matoušek, On range searching with semialgebraic sets, *Discrete & Computational Geometry* **11** (1994), pp. 393-418.

[3] P.K. Agarwal and M. Sharir, Applications of a new space partitioning technique, *Proc. 2nd Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science **519** (1991), pp. 379-391.

[4] P.K. Agarwal, M. Sharir, and P.W. Shor, Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences, *Journal on Combinatorial Theory, Series A* **52** (1989), pp. 228-274.

[5] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig, Approximate motion planning and the complexity of the boundary of the union of simple geometric figures, *Algorithmica* **8** (1992), pp. 391-406.

[6] B. Aronov and M. Sharir, Triangles in space or building (and analyzing) castles in the air, *Combinatorica* **10** (1990), pp. 137-173.

[7] B. Aronov and M. Sharir, On translational motion planning in 3-space, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 21-30.

[8] F. Aurenhammer, Voronoi diagrams: a survey of a fundamental geometric data structure, *ACM Computing Surveys* **23** (1991), pp. 345-405.

[9] F. Avnaim and J.-D. Boissonnat, Polygon placement under translation and rotation, *Proc. 5th Annual Symp. on Theoretical Aspects of Computer Science (STACS'88)*, (R. Cori and M. Wirsing Eds.), Lecture Notes in Computer Science **294** (1988), pp. 322-333.

paragraph of this chapter to an anchored robot arm does not give rise to an interesting motion planning problem. Even though the number of obstacles can be high, the total number of obstacles touched by the robot in any placement is at most constant, due to its bounded size. The free space of the anchored robot has constant complexity and a rigorous cell decomposition method [85] suffices to compute a cell decomposition of the free space in constant time.

A more interesting problem formulation follows when we take a closer look at industrial robot arms. Typically, the links close to the base of the arm are long (*major axes*) whereas the links close to the tip, or hand, are short (*minor axes*). Figure 8.2 shows a robot arm with two major axes $L_1$ and $L_2$ and two minor axes $L_3$ and $L_4$. Now consider an $f$-link robot arm $\mathcal{B}$ of which the $m$ links closest to the

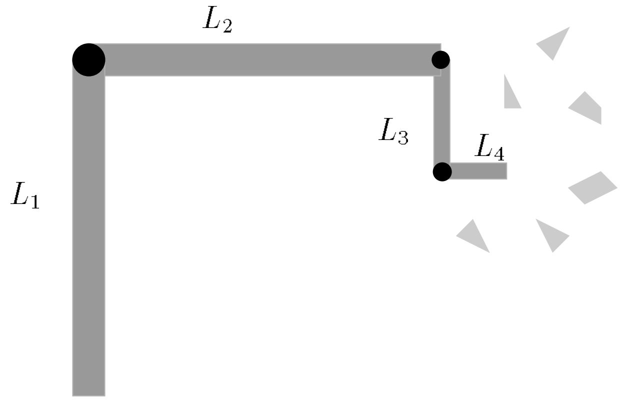


Figure 8.2: A robot arm with two major axes $L_1$ and $L_2$ and two minor axes $L_3$ and $L_4$.

tip are not too large compared to the obstacles. The sizes of the $f-m$ major axes are not bounded. Assume that $C_1$ is the $(f-m)$-dimensional configuration space corresponding to the major axes and that $C_2$ is the $m$-dimensional configuration space corresponding to the minor axes. Hence, $C = C_1 \times C_2$ is the $f$-dimensional configuration space of $\mathcal{B}$. A point $p \in C_1$ fixes the placements of all major axes. If $m$ is a constant, then the $m$ minor axes can only touch a constant number of obstacles while the major axes are fixed, due to the low obstacle density. As a result, the lifting of the point $p \in C_1$ into $C$ will be intersected by only a constant number of constraint hypersurfaces. So, the configuration space $C$ is cylindrifiable and $C_1$ is a valid base space. Again, however, the existence of a small and efficiently computable base partition in $C_1$ remains uncertain. Nevertheless, we expect the complexity of the free space to be closer to $O(n^{f-m})$ then to the obvious upperbound of $O(n^f)$.

We can conclude that the paradigm presented in this thesis might lead to many more efficient motion planning algorithms for a variety of instances of the problem.

at a particular time $t$ has linear complexity.

The usual approach to the exact solution of a motion planning problem with $c$ robots $\mathcal{B}_1, \ldots, \mathcal{B}_c$ with configuration spaces $C_1, \ldots, C_c$ of dimensions $f_1, \ldots, f_c$ is *centralized planning*. In centralized planning, the $c$ robots are regarded as one multi-body robot $\mathcal{B} = \mathcal{B}_1 \cdots \mathcal{B}_c$. Planning the motion of the multi-body robot $\mathcal{B}$ takes place in the composite configuration space $C = C_1 \times \ldots \times C_c$. There, collisions of the robots $\mathcal{B}_i$ ($1 \leq i \leq c$) turn into collisions of the multi-body robot $\mathcal{B}$. (The alternative to centralized planning, *decoupled planning*, plans the motions of each of the robots independently, and then considers the interactions of the resulting paths. Decoupled planning is not guaranteed to find a solution to the problem.) The complexity of the free part of the composite configuration space $C$ can, for fat obstacles, easily be as high as $\Omega(n^c)$, even when the reaches $\rho_{\mathcal{B}_i}$ of the individual bodies $\mathcal{B}_i$ ($1 \leq i \leq c$) are bounded by $\rho_{\mathcal{B}_i} \leq b \cdot \rho$, for some constant $b \geq 0$. The key observation here is that the reach $\rho_{\mathcal{B}}$ of the composite robot is in no way bounded: two bodies $\mathcal{B}_i$ and $\mathcal{B}_j$ can be infinitely far apart. Figure 8.1 shows a single $c$-fold contact of $\mathcal{B}$. Clearly, there are $\Omega(n^c)$ such contacts. Still, there is a considerable gap between the $\Omega(n^c)$

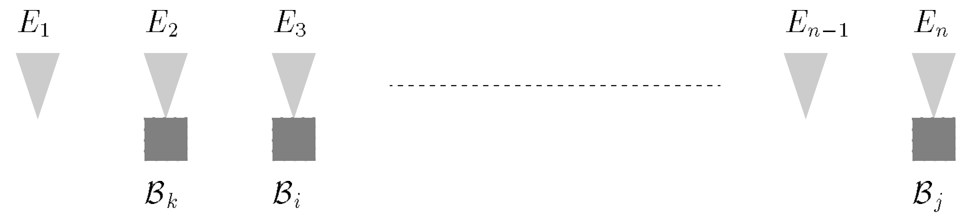


Figure 8.1: Each of the $c$ robots $\mathcal{B}_1, \ldots, \mathcal{B}_c$ touches one of the $n$ obstacles $E_1, \ldots, E_n$. This corresponds to a single $c$-fold contact of the composite robot $\mathcal{B} = \mathcal{B}_1 \cdots \mathcal{B}_c$. The total number of such contacts is $\Omega(n^c)$.

lowerbound construction and the obvious upperbound of $O(n^f)$, with $f = \sum_{1 \leq i \leq c} f_i$, on the complexity of the free space. We believe the complexity of the free space to be close to the lowerbound. The ideas of cylindrical decomposition of the free space seem applicable to some extent if the workspace $\mathrm{W} = \mathbb{R}^d$ is a projective subspace of each of the configuration spaces $C_i$ ($1 \leq i \leq c$). Then the 'composite workspace' $\mathrm{W}^c$ is a projective subspace of the composite configuration space $C$. A point $p \in \mathrm{W}^c$ fixes the positions of the reference points of all bodies $\mathcal{B}_i$. The low obstacle density of the workspace and the bounds on the sizes of the individual bodies yield that each body can touch only a constant number of obstacles while its reference point is fixed. Provided that $c$ is a constant, the lifting of $p$ into $C$ is intersected by $O(1)$ constraint hypersurfaces. Hence, $C$ is a cylindrifiable configuration space and W is a valid base space. The existence of a small and efficiently computable base partition remains an open question.

The straightforward application of the framework of assumptions in the second

The cc-partitions that are obtained for the two remaining instances, motion planning in 3-space amidst polyhedral obstacles and amidst arbitrarily-shaped obstacles, have sizes $O(n^2)$ and $O(n^3)$ and are computable in time $O(n^2 \log n)$ and $\Theta(n^3)$ respectively. The partitions give rise to motion planning algorithms with running times $O(n^2 \log n)$ and $\Theta(n^3)$ that compute cell decompositions of sizes $O(n^2)$ and $O(n^3)$ for the respective problems, regardless of the number of degrees of freedom of the robot. These results might be improvable. The challenge is to find subquadratic and subcubic partitions of the workspace W such that each region has constant complexity and intersects no more than a constant number of cells of the arrangement of grown obstacle boundaries.

Besides attempting to improve the latter two results, it is also interesting to see if the paradigm for motion planning amidst fat obstacles applies to other classes of motion planning problems involving fat obstacles. Let us give some thoughts on some possible extensions, like motion planning with moving obstacles, multiple robots, and anchored robot arms.

Motion planning problems involving moving obstacles are normally solved in configuration-time space. The configuration-time space $CT$ is the Cartesian product of the configuration space $C$ of the stationary version of the problem and the time dimension $T$. The fact that motion back in time is impossible is reflected by the additional requirement that any solution curve between a pair of query placements in the configuration-time space must be strictly monotone in time. The requirement imposes restrictions on the search of the connectivity graph of a cell decomposition of the free part of $CT$ and on the simple motions within each subcell of the decomposition. The complexity of the free part of the configuration-time space $CT$ can increase rapidly when many obstacles are non-stationary and travel along complicated trajectories. The ideas of the preceding two chapters will definitely not be applicable in such cases. Things seem different when we assume that only a constant number $c$ out of the $n$ obstacles move along simple trajectories, that is, algebraic curves of bounded degree. Let us assume furthermore that a cc-partition of the $d$-dimensional workspace with the $n - c$ stationary obstacles is given. Now, the cylinders obtained by lifting the $d$-dimensional cc-partition into the $(f+1)$-dimensional configuration-time space are intersected by only a constant number of constraint hypersurfaces defined by contacts of the robot and the stationary obstacles. The overall number of constraint hypersurfaces due to contacts of the robot and the moving obstacles is constant. The simplicity of the motion supposedly implies that the intersection of each such constraint hypersurface with a cylinder consists of a constant number of constant-complexity connected components. Hence, the constraint hypersurfaces define constant-complexity arrangements in every cylinder. Therefore, the results of the previous chapter seem to generalize directly to environments in which a constant number of the obstacles are non-stationary. When the number of moving obstacles is not constant, the preceding arguments no longer hold. It is though to be expected that, still, a large complexity reduction is achievable: from the fatness of the (moving and stationary) obstacles it follows that any cross-section of the free part of $CT$

they allow for simple path-finding in their interiors due to their constant complexity and connectedness. The validity of a projective subspace as a base space is verified by a proof that the lifting back into configuration space of every point in the base space is intersected by at most a constant number of constraint hypersurfaces, which are sets of contact placements of a robot and an obstacle feature. The preceding considerations reduce the problem of finding a cell decomposition of the free space to the problem of finding some constrained decomposition of the lower-dimensional base space, in which the regions are appropriate cylinder bases. A uniform sequence of operations then suffices to transform the base partition into a cell decomposition of the free space of asymptotically equal size. The running time of the entire paradigm is determined by the time to compute the base partition. A small and efficiently computable base partition is therefore of obvious importance to the success of the approach. Finding a small and efficiently computable base partition may seem a hard problem at first sight, because the regions are constrained by a restriction on their liftings, hence in configuration space.

The extensive and interesting class of motion planning problems with configuration spaces $C = \mathrm{W} \times D$, which includes for example all problems that involve a free-flying robot, allows for choosing the robot's workspace W as a base space. Moreover, any so-called cc-partition of W, which is subject to constraints that are formulated exclusively in W, turns out to be a valid base partition of the base space W. A cc-partition decomposes the workspace into constant-complexity regions that intersect at most a constant number of cells of the arrangement of grown obstacle boundaries[1]. A grown obstacle is the collection of points within a distance $\rho_{\mathcal{B}}$ from the original obstacle. The arrangement of grown obstacle boundaries has complexity $O(n)$ and each point $p \in \mathrm{W}$ lies in no more than a constant number of grown obstacles simultaneously.

Optimal $O(n)$ size cc-partitions exist for three out of five practical instances of the motion planning problem amidst fat obstacles studied in this thesis. These instances are motion planning in the plane amidst arbitrarily-shaped obstacles, motion planning in 3-space amidst arbitrarily-shaped obstacles of comparable sizes, and motion planning on a workfloor in 3-space amidst arbitrarily-shaped obstacles. The reason for treating restricted instances of the motion planning problem in 3-space lies in the failure to find an optimal partition for the general version of the problem and in the frequent occurrence of these specific instances in real-life motion planning problems. All three cc-partitions are computable in nearly-optimal $O(n \log n)$ time. In conclusion, we have obtained $O(n \log n)$ motion planning algorithms for each of the three classes of problems. The algorithms yield a cell decomposition of optimal size $O(n)$. Notice that these results do not depend on the number of degrees of freedom of the robot.

[1] Note that this description of a cc-partition seems somewhat different from the formal definition in Chapter 6. Nevertheless, it is essentially equivalent, due to the constant-size coverage of the arrangement cells.

# Chapter 8

# Concluding remarks

This final chapter recaptures some of the main results in this thesis and gives some reflections on possible extensions and improvements. The reader is warned that most reflections are based on intuitive feelings and not supported by indisputable proofs.

We have studied the motion planning problem for a constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom amidst $n$ constant-complexity $k$-fat obstacles $E \subseteq \mathbb{R}^d$, for some constants $d, f \geq 0$ and $k \geq 1$. In addition, the reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is assumed to be bounded from above by a constant multiple $b \geq 0$ of $\rho$, where $\rho$ is a lower bound on the minimal enclosing hypersphere radius of any obstacle $E$. The mild assumptions are considered to provide a realistic framework for many practical motion planning problems. The complexity of the free space for problems that satisfy the assumptions was proven to be $O(n)$, whereas the complexity can easily be as high as $\Omega(n^f)$ when both the fatness assumption on the obstacles and the bounded-size assumption on the robot are dropped. This remarkable gap makes it interesting to study the individual influence on the free space complexity of each of the two assumptions in more detail, which we leave as an open question.

The basis of the linear complexity result is a low obstacle density property (relative to the robot size) of the workspace $\mathrm{W} = \mathbb{R}^d$, which is implied by the combination of the two assumptions. As such, the low obstacle density property imposes a weaker restriction for obtaining linear complexity free spaces. (To see that the latter condition is weaker, imagine a workspace with non-fat obstacles that are far apart.) All algorithmic motion planning results in this thesis, however, apply equally well to problems that satisfy the relaxed condition.

Besides having a low combinatorial complexity, the free space for a motion planning problem that fits in our framework also has a beneficial structure. The structure allows for a decomposition of the configuration space into cylinders, with bases in some projective subspace, the so-called base space, such that the free space part of every cylinder has constant-complexity. In other words, the cylinder walls partition the free space into constant-complexity parts. The maximal connected components of the free parts make perfect subcells in a cell decomposition of the free space, as

the appropriate hypersurfaces is possible when the function $Cov_{\mathrm{F}} : V_{\mathrm{F}} \to \mathcal{P}(\mathcal{E})$ is given. Therefore, we decide to compute the function $Cov_{\mathrm{F}}$ along with the vertical decomposition. The similarity of the present triple $(V_{\mathrm{F}}, E_{\mathrm{F}}, Cov_{\mathrm{F}})$ and the triple $(V_{\mathrm{W}}, E_{\mathrm{W}}, Cov)$ in Section 7.1 suggests the use of the plane sweep from that section for the simultaneous computation of $V_{\mathrm{F}}$, $E_{\mathrm{F}}$, and $Cov_{\mathrm{F}}$ in $O(n \log n)$ time. The input to the sweep are the $O(n)$ (labeled) maximal connected $x$-monotone arcs of the boundaries $\partial G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$ having no vertices in their interiors. The arcs of a single boundary $\partial G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ are obtained in constant time by intersecting the constant-complexity grown obstacle $G(E, \rho_{\mathcal{B}})$ with the plane $\mathrm{F} \times \{0\}$, and subsequently cutting the projection of the intersection into the appropriate $x$-monotone arcs. The generation of all input arcs takes $O(n)$ time, so the entire computation of the base partition $(V_{\mathrm{F}}, E_{\mathrm{F}})$ and the function $Cov_{\mathrm{F}}$ from the grown obstacles takes $O(n \log n)$ time.

**Lemma 7.31** *The computation of the base partition $V_{\mathrm{F}}$, the set $E_{\mathrm{F}}$, and the function $Cov_{\mathrm{F}}$ takes $O(n \log n)$ time.*

The computation of a cell decomposition consists of two steps: the first step which computes a valid base partition, and the second step which transforms the base partition into a decomposition of the free space. The second step takes time $O(|V_{\mathrm{F}}| + |E_{\mathrm{F}}|) = O(n)$, because the function $Cov_{\mathrm{F}}$ supports the constant time computation of $\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D)\}$ for any region $R$. The combination with the $O(n \log n)$ running time for the first step justifies the following theorem, which formulates the main result of this section.

**Theorem 7.32** *Let $k \geq 1$ and $b \geq 0$ be constants and let $\mathcal{E}$ be a collection of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W} = \mathbb{R}^3$ with minimal enclosing sphere radii at least $\rho$. Algorithm FatMot solves the motion planning problem for any constant-complexity robot $\mathcal{B}$ with $f \geq 2$ degrees of freedom and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ amidst $\mathcal{E}$ whose reference point $O \in \mathcal{B}$ is confined to the planar workfloor $\mathrm{F} \times \{0\}$ in time $O(n \log n)$. The connectivity graph $\mathrm{CG} = (V_C, E_C)$ of the resulting decomposition of $\mathrm{FP}$ into simple subcells has optimal size $O(n)$.*

complexities of the obstacles and the robot. Hence, any partition of F into constant-complexity regions $R$ with $|Cov_{\mathrm{F}}(R)| = O(1)$ is a valid base partition for a cylindrical cell decomposition of $\mathrm{FP} \subseteq C$, and as such, a valid input to the transformation algorithm. Let us now focus on the arrangement of regions $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ in the plane F.

The planar arrangement $\mathcal{A}(G_{\mathrm{F}})$ of all boundaries $\partial G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$ subdivides F into maximal connected sets of points $p \in \mathrm{F}$ with equivalent collections $Cov_{\mathrm{F}}(p)$. Lemma 7.29 states that the arrangement has complexity $O(n)$. In addition, it states that each 2-cell $A$ in $\mathcal{A}(G_{\mathrm{F}})$ satisfies $|Cov_{\mathrm{F}}(A)| = O(1)$. $\mathrm{F} \times \{0\}$.

**Lemma 7.29** *Let $\mathcal{A}(G_{\mathrm{F}})$ be the planar arrangement of all boundaries $\partial G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$. Then*

- **(a)** *$\mathcal{A}(G_{\mathrm{F}})$ has complexity $O(n)$,*
- **(b)** *$|Cov_{\mathrm{F}}(A)| = O(1)$ for all 2-faces $A \in \mathcal{A}(G_{\mathrm{F}})$.*

**Proof:** Theorem 2.12 yields for the spatial arrangement $\mathcal{A}(G)$ of all grown obstacles $G(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$: (i) the complexity of $\mathcal{A}(G)$ is $O(n)$, and (ii) every point $p \in \mathrm{W} = \mathbb{R}^3$ lies in at most $O(1)$ regions $G(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$. The intersection of the linear complexity arrangement $\mathcal{A}(G)$ with the plane $\mathrm{F} \times \{0\}$ results in a planar arrangement in $\mathrm{F} \times \{0\}$ with complexity $O(n)$. Hence, $\mathcal{A}(G_{\mathrm{F}})$ has complexity $O(n)$. To prove (b) it suffices to pick a point $p \in A$ and prove that it lies in at most $O(1)$ regions $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$, because all points in a single face $A$ lie in exactly the same regions. By expression (ii), every point $p \in \mathrm{F} \times \{0\} \subset \mathrm{W}$ lies in at most $O(1)$ regions $G(E, \rho_{\mathcal{B}}) \cap (\mathrm{F} \times \{0\})$, yielding (b). $\square$

The arrangement $\mathcal{A}(G_{\mathrm{F}})$ partitions the base space F into regions whose corresponding configuration space cylinders are intersected by only $O(1)$ constraint hypersurfaces. The regions, however, do not have constant complexity, but thanks to the planarity of the subdivision, such is easily remedied by vertical decomposition of the arrangement. The resulting vertical decomposition regions are the regions of $V_{\mathrm{F}}$. The set $E_{\mathrm{F}} = \{(R, R') \in V_{\mathrm{F}} \times V_{\mathrm{F}} | \partial R \cap \partial R' \neq \emptyset\}$ consists of all adjcencies of pairs of regions. Section 7.1 explains why the vertical decomposition of an arrangement of complexity $O(n)$ consists of $O(n)$ regions and region adjacencies.

**Lemma 7.30** *$V_{\mathrm{F}}$ consists of $O(n)$ constant-complexity regions $R$ that partition* F *and satisfy $|Cov_{\mathrm{F}}(R)| = O(1)$; $E_{\mathrm{F}}$ has size $O(n)$.*

The transformation algorithm in Section 6.1 transforms the base partition represented by the sets $V_{\mathrm{F}}$ and $E_{\mathrm{F}}$ into a decomposition of the free space in time $O(|V_{\mathrm{F}}| + |E_{\mathrm{F}}|)$ provided that for every region $R$ the collection of constraint hypersurfaces $\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D)\}$ is computable in constant time. Fortunately, we have found that $\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D)\} \subseteq \{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f Cov_{\mathrm{F}}(R)\} = O(1)$, which shows that a constant time computation of

**Definition 7.26** $G_{\mathrm{F}}(E, \rho_{\mathcal{B}}) = \{(x,y)|(x,y,0) \in G(E, \rho_{\mathcal{B}})\}$.

The restriction on the position of the reference point of the robot $\mathcal{B}$ for being able to touch an obstacle $E$ has implications for the location in configuration space of the constraint hypersurfaces defined by a contact of a feature of $\mathcal{B}$ with a feature of $E$.

**Lemma 7.27** *Let $\phi \in_f \mathcal{B}$ and $\Phi \in_f E$. Then:*

$$f_{\phi,\Phi} \subseteq G_{\mathrm{F}}(E, \rho_{\mathcal{B}}) \times D.$$

**Proof:** Let $p = (p_{\mathrm{F}}, p_D) = ((p_x, p_y), p_D) \in f_{\phi,\Phi}$, such that $p_{\mathrm{F}} \in \mathrm{F}$ and $p_D \in D$. We must prove that $p = ((p_x, p_y), p_D) \in G_{\mathrm{F}}(E, \rho_{\mathcal{B}}) \times D$, which may be reduced to proving that $(p_x, p_y) \in G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ as $p_D \in D$ is trivially true. Proving $(p_x, p_y) \in G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ is, in turn, equivalent to proving $(p_x, p_y, 0) \in G(E, \rho_{\mathcal{B}})$. This means that $\mathcal{B}$'s reference point must be placed inside $G(E, \rho_{\mathcal{B}})$ when $\mathcal{B}$'s feature $\phi$ touches $\Phi$.

Assume, for a contradiction, that $(p_x, p_y, 0) \notin G(E, \rho_{\mathcal{B}})$. Then, by the definition of grown obstacles, the distance from $(p_x, p_y, 0)$ to $E$ exceeds $\rho_{\mathcal{B}}$. But then, it is impossible for $\mathcal{B}$ to reach and touch the obstacle $E$, by the definition of the reach of a robot. In other words, no feature $\phi' \in_f \mathcal{B}$ can touch a feature $\Phi' \in_f E$. So, the point $p = (p_{\mathrm{F}}, p_D) = ((p_x, p_y), p_D)$ with $(p_x, p_y, 0) \notin G(E, \rho_{\mathcal{B}})$ cannot lie on $f_{\phi,\Phi}$ contradicting the assumption of the lemma. □

Lemma 7.27 provides, similar to Lemma 6.3 in Section 6.2 for free-flying robots, some simple outer approximation on the location of any constraint hypersurface $f_{\phi,\Phi}$ in the configuration space $C = \mathrm{F} \times D$. If $R \subseteq \mathrm{F}$ does not intersect $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$, then no constraint hypersurface $f_{\phi,\Phi}$ with $\Phi \in_f E$ intersects the configuration space cylinder $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$. The following definition simplifies a more general formulation of the consequences of Lemma 7.27.

**Definition 7.28** *Let $R \subseteq \mathrm{F} = \mathbb{R}^2$.*

$$Cov_{\mathrm{F}}(R) = \{E \in \mathcal{E} | R \cap G_{\mathrm{F}}(E, \rho_{\mathcal{B}}) \neq \emptyset\}.$$

$Cov_{\mathrm{F}}(R)$ is the collection of obstacles $E$ whose corresponding regions $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ intersect $R$. Now, if a region $R \subseteq \mathrm{F}$ is intersected by a collection of regions $G_{\mathrm{F}}(E, \rho_{\mathcal{B}})$ then the configuration space cylinder $R \times D$ can only be intersected by constraint hypersurfaces induced by the corresponding obstacles $E$, or, more formally:

$$\begin{aligned} & \{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\} \\ \subseteq \; & \{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f Cov_{\mathrm{F}}(R)\} \end{aligned}$$

The set inclusion directly shows that if $R$ is chosen such that $|Cov_{\mathrm{F}}(R)| = O(1)$, then $|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1)$, due to the constant

shown that this arrangement partitions the plane F into regions whose corresponding configuration space cylinders are intersected by a constant number of constraint hypersurfaces. The $O(n)$ curves resulting from the intersection of grown obstacle faces with $\mathrm{F} \times \{0\}$ have constant complexity, due to the constant complexity of the grown obstacles. The arrangement of curves has complexity $O(n)$. To obtain a valid base partition in the base space F we compute, inspired by resemblance with Section 7.1, the vertical decomposition of the planar arrangement of intersection curves in time $O(n \log n)$ time. The resulting base partition graph has complexity $O(n)$. The computation of the constraint hypersurfaces that intersect the configuration space cylinders is supported by sets that resemble the region coverages, and are computed along with the vertical decomposition in a way that strongly resembles the computation of the coverages. Below, we settle the details of the informally described approach.

A restriction on the applicability of the general paradigm is that the configuration space $C$ is cylindrifiable, that is, decomposable in some $B$ and $D$ such that for all $p \in B$:

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

Lemma 6.1 states that the condition is satisfied for $B = \mathrm{W}$, but W is not a subspace of $C$. Fortunately, the property is inherited by the subset $\mathrm{F} \times \{0\} \subset \mathrm{W}$ leading to the following property.

**Property 7.25** *For all $p \in \mathrm{F}$:*

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

As a result, the two-dimensional Euclidean space F is a feasible base space for a cylindrical decomposition of $\mathrm{FP} \subseteq C = \mathrm{F} \times D$. The algorithm in Section 6.1 transforms the graph $(V_\mathrm{F}, E_\mathrm{F})$ corresponding to a base partition in F into a cell decomposition of the free part of $C$, provided that all regions $R \in V_\mathrm{F}$ have constant complexity and satisfy:

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1).$$

In the previous chapter we have formalized the informal observation that a robot can only touch an obstacle lying within its reach using the notion of grown obstacles. A robot with its reference point at some point $p \in \mathrm{W}$ is only able to touch an obstacle $E$ if $p \in G(E, \rho_\mathcal{B})$. In our constrained case, the position $p$ of $\mathcal{B}$'s reference point for potential collision with $E \in \mathcal{E}$ is further restricted to $p \in G(E, \rho_\mathcal{B}) \wedge p \in \mathrm{F} \times \{0\}$. In other words, the reference point of $\mathcal{B}$ must be positioned at some point in the intersection of the grown obstacle $G(E, \rho_\mathcal{B})$ and the plane $\mathrm{F} \times \{0\} = \mathbb{R}^2 \times \{0\}$. Notice that the emptiness of the intersection implies that $\mathcal{B}$ cannot collide with $E$ during its constrained motion. We define $G_\mathrm{F}(E, \rho_\mathcal{B})$ to be the intersection $G(E, \rho_\mathcal{B}) \cap (\mathrm{F} \times \{0\})$. (Actually, it is the intersection restricted to the first two coordinates, to make it lie in F.)

and fat obstacles, the free space is shown below to be decomposable into $O(n)$ simple subcells in time $O(n \log n)$, using the general ideas of Section 6.1, yielding an $O(n \log n)$ motion planning algorithm.

Choosing the position of the robot's reference point as a part of the specification of the robot placement, similar to the preceding sections, seems a bad idea. The fact that the position $(x, y, z)$ of the robot's reference point is restricted by $z = 0$, would have unclear implications for path-finding in the resulting configuration space, as not all free placements are valid placements according to the constraint on the position of the robot's reference point. Finding a free path for the robot would require some constrained path-finding in such a configuration space. Instead, we choose the configuration space to be

$$C = \mathrm{F} \times D = \mathbb{R}^2 \times D,$$

where $D$ is again some rest-space. Any point $Z = (Z_{\mathrm{F}}, Z_D) = ((x, y), Z_D)$, with $Z_{\mathrm{F}} \in \mathrm{F}, Z_D \in D$, in configuration space corresponds to a placement of $\mathcal{B}$ in which its reference point is positioned at $(x, y, 0)$ in the workspace W. A possible rest-space for a vacuum cleaner would be $D = [0, 2\pi)$. The configuration space $C$ formulated above makes the application of the tailored paradigm from Section 6.2 for solving the motion planning problem impossible, as the configuration space is not a superspace of the robot's workspace.

A possible strategy for computing a cell decomposition of the free space would be to temporarily discard the restriction on the position of the reference point and act as if the robot is free-flying, and, hence, has configuration space $C' = \mathrm{W} \times D = \mathrm{F} \times \mathbb{R} \times D$. We may borrow the ideas of Section 7.2 to decompose the free part $\mathrm{FP}'$ of $C'$ into $O(n^2)$ simple subcells in $O(n^2 \log n)$ time. The free part FP of the configuration space $C = \mathrm{F} \times D$ of the constrained problem can be regarded as the projection onto the space $\mathrm{F} \times D$ of the subset $\mathrm{FP}' \cap (\mathrm{F} \times \{0\} \times D)$ of the free part $\mathrm{FP}' \subseteq C' = (\mathrm{W} \times D)$. To obtain a cell decomposition of the free part $\mathrm{FP} \subseteq C = \mathrm{F} \times D$, we simply intersect all subcells and common boundaries in the decomposition of $\mathrm{FP}' \subseteq C' = \mathrm{W} \times D$ with the space $\mathrm{F} \times \{0\} \times D$ and subsequently project the intersection onto the subspace $\mathrm{F} \times D$. Regardless of the (possibly unnecessarily large) complexity of the resulting decomposition, the computation takes $O(n^2 \log n)$ time, which is inferior to the approach that we follow below leading to a decomposition of size $O(n)$ in time $O(n \log n)$. The approach exploits the general paradigm for cylindrifiable configuration spaces, which is given in Section 6.1 as an algorithm for transforming a base partition of some appropriate base space into a cell decomposition of the free space.

The projected workfloor F turns out to be a good choice for a base space for the cylindrical decomposition of the free space. The subdivision that we propose strongly resembles the partition of the workspace $\mathrm{W} = \mathbb{R}^2$ by the grown obstacles discussed in Section 7.1. Consider the planar arrangement in $\mathrm{F} \times \{0\}$ defined by the intersection of the grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}}) \subseteq \mathrm{W}$ and the workfloor $\mathrm{F} \times \{0\}$. It will be

## 7.5 Planar motion amidst arbitrary obstacles in 3-space

With the present technological state-of-the-art, one rarely encounters free-flying three-dimensional robots in industrial environments. Instead, many robots move in three-dimensional workspaces amidst spatial obstacles while their motion is confined to a (planar) workfloor. A realistic example of such a setting is a vacuum cleaner moving in a room in which objects hang from the ceiling and stand on the floor [100]. Sometimes, the nature and positions of the obstacles does not allow to reduce such problem to purely planar motion planning. The vacuum cleaner, for example, can easily pass under a table. An approach to solve the problem by projecting the vacuum cleaner and the obstacles onto the floor and then finding a (planar) path for the projected vacuum cleaner amidst the projected obstacles would forbid such paths. In this section we study a general formulation of the type of problem outlined above in the context of fat obstacles.

We consider a workspace $\mathrm{W} = \mathbb{R}^3$ with $k$-fat constant-complexity obstacles. The motion of the robot $\mathcal{B}$ in this workspace is constrained by the assumption that a specific point $p$ in $\mathcal{B}$ is restricted to a workfloor $\mathrm{F} \times \{0\} = \mathbb{R}^2 \times \{0\} \subset \mathbb{R}^3$; the workspace W is the Cartesian product of the projection F of the workfloor and the real line: $\mathrm{W} = \mathrm{F} \times \mathbb{R}$. For convenience, we choose the robot's reference point $O \in \mathcal{B}$ to be equal to the point $p$ that is restricted to the workfloor. Hence,

$$O[Z] \in \mathrm{F} \times \{0\}$$

for all placements $Z \in C$ of the robot $\mathcal{B}$. The following problem statement fixes the setting of this section.

> *A constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom ($f \geq 2$) and reach $\rho_{\mathcal{B}}$ moves with some point $O \in \mathcal{B}$ restricted to a plane* F *in the workspace* $\mathrm{W} = \mathrm{F} \times \mathbb{R} = \mathbb{R}^3$ *amidst a collection $\mathcal{E}$ of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W}$ with minimal enclosing sphere radii at least $\rho$, for some constant $k \geq 1$. The system is constrained by the inequality $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some fixed constant $b \geq 0$.*

Note that the problem statement does not restrict the robot to rotate around an axis perpendicular to the workfloor only (as for the vacuum cleaner). The robot is allowed to rotate arbitrarily and can have more degrees of freedom. (An example of such a robot is a moon vehicle, equipped with several arms to grasp stones etc.)

Planar motion planning in 3-space is sometimes referred to as two-and-a-half-dimensional motion planning, for understandable reasons. A solution to this type of path-finding for a polyhedral robot amidst polyhedral obstacles is given by Wentink and Schwarzkopf in [100]. Their algorithm, which is a generalization of the boundary cell decomposition algorithm by Avnaim, Boissonnat, and Faverjon [10], runs in time $O(n^3 \log n)$. Under the realistic assumptions of a bounded-size robot

tion, both pointers are forwarded to the start of the next level. As the number of $y$-adjacencies is bounded by $O(n)$ and both pointers make a single traversal of the sorted sequence of regions, the total time for computing the $y$-adjacencies equals $O(n)$.

The computation of the $x$-adjacencies, finally, resembles the computation of the $y$-adjacencies. Recall that $x$-adjacencies are restricted to pairs of regions in subsequent slices. Again we traverse the sorted sequence of regions with two pointers which now invariantly point to the start of 'comparable-height' levels in subsequent slices. (Comparable-height levels in subsequent slices share a common face parallel to the $(y, z)$-plane.) As a result, pairs of regions in the comparable levels are $x$-adjacent. The $x$-adjacencies induced by the two levels can be computed by the basic strategy outlined above in time proportional to the number of adjacencies. After the computation, the pointer corresponding to the level with the lowest upper boundary with respect to the $y$-coordinate is forwarded to the next level. As the number of $x$-adjacencies is bounded by $O(n)$ and both pointers make a single traversal of the sorted sequence of regions, the total time for computing the $x$-adjacencies equals $O(n)$.

To compute the coverage of the regions of $V_{\mathrm{W}}$, we borrow the ideas from Section 7.3. Thus, the approach is to consider obstacle by obstacle and compute all regions $R \in V_{\mathrm{W}}$ (in fact $R \in V_1$) intersected by the corresponding grown obstacle. One arbitrary region $R \in V_{\mathrm{W}}$ intersecting the grown obstacle can be determined in $O(\log n)$ time, using the ordered sequence of regions. Starting from that region, the other $O(1)$ regions intersected by the grown obstacle are identified in $O(1)$ time, using $E_{\mathrm{W}}$. (See Section 7.3 for the details.) The approach amounts to $O(n \log n)$ for computing $Cov$.

**Lemma 7.23** *The computation of $E_{\mathrm{W}}$ and $Cov$ takes $O(n \log n)$ time.*

If we substitute the above computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function $Cov$ for the abstract first step of algorithm FatMot, then we obtain an efficient algorithm for computing a cell decomposition of the free space. The algorithm yields a decomposition into $O(n)$ constant-complexity cells in time $T(n) = O(n \log n)$ by Theorem 6.8, as $T(n)$ stands for the time required to compute $V_{\mathrm{W}}$, $E_{\mathrm{W}}$, and $Cov$.

**Theorem 7.24** *Let $k \geq 1$, $b \geq 0$, and $u \geq 1$ be constants and let $\mathcal{E}$ be a collection of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W} = \mathbb{R}^3$ with minimal enclosing sphere radii in the range $[\rho, u\rho]$. Algorithm FatMot solves the motion planning problem for any constant-complexity robot $\mathcal{B}$ with $f \geq 3$ degrees of freedom and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ amidst $\mathcal{E}$ in time $O(n \log n)$. The connectivity graph $\mathrm{CG} = (V_C, E_C)$ of the resulting decomposition of $\mathrm{FP}$ into simple subcells has optimal size $O(n)$.*

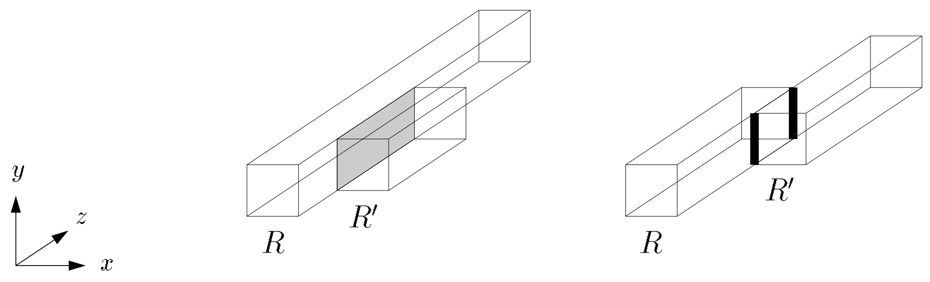


Figure 7.7: Two types of adjacencies $(R, R') \in V_2 \times V_2$.

involved side of $R'$ and vice versa. We can charge the adjacency to either edge (the bold edges in Figure 7.7). Each edge can only be charged once, because it can lie in the interior of only one face. The number of edges of regions of $V_2$ is $O(n)$, so the number of adjacencies of two regions from $V_2$ of the second kind is $O(n)$ as well.

Combination of all linear bounds yields that $|E_{\mathrm{W}}| = O(n)$. □

For the computation of the set $E_{\mathrm{W}}$ of adjacencies, it is convenient to have the $O(n)$ regions of $V_{\mathrm{W}}$ ordered lexicographically. Such can be achieved in time $O(n \log n)$. The computations of the $z$-adjacencies, the $y$-adjacencies, and the $x$-adjacencies proceed as outlined below.

The $z$-adjacencies can be extracted from the sequence of regions in a straightforward single scan, taking $O(n)$ time. Two regions that are $z$-adjacent appear consecutively in a $z$-column, and also in the lexicographical order. Any pair of consecutive regions in the sequence lying in the same $z$-column should be reported as an adjacency.

At the heart of the computation of the $x$- and $y$-adjacencies lies a basic operation that reports pairs of adjacent regions in two adjacent levels (either in a single slice or in two subsequent slices). Assume that the adjacent levels are divided into $m$ and $n$ regions respectively. Then, it is easily verified that the number of adjacencies involving one region from either level is $\Theta(m+n)$. Moreover, the adjacencies can be reported in time $\Theta(m+n)$ by a simultaneous scan of the two levels from $z = -\infty$ to $z = \infty$. In conclusion, the region adjacencies in two adjacent levels can be reported in time proportional to the number of adjacencies.

The $y$-adjacencies are restricted to pairs of regions in subsequent levels of a single slice. To identify all pairs of subsequent levels (and to compute all adjacencies induced by their regions) we traverse the lexicographical order of regions with two pointers. The pointers invariantly point to the start of the subsequences corresponding to two subsequent levels. At any combination of pointer positions, we apply the techniques of the previous paragraph to compute the adjacencies induced by the two levels in time proportional to the number of adjacencies. After this computa-

is a $z$-column of width and height $\rho$. Recall that a $z$-column contains no regions from $V_3$. The recursive structure of the subdivision shows that all regions in $V_{\mathrm{W}}$ can be stored lexicographically, that is, slice by slice by increasing $x$, level by level by increasing $y$ within a single slice, and by increasing $z$ within a single level. The above structuring makes it easy to distinguish three different types of adjacencies. Two adjacent regions are $x$-adjacent if they lie in different (subsequent) slices; two adjacent regions are $y$-adjacent if they lie in different (subsequent) levels of a single slice; two adjacent regions in a single level (or $z$-column) are $z$-adjacent.

We are now ready to prove an upper bound on the number of region adjacencies in the cc-partition.

**Lemma 7.22** *$E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $O(n)$.*

**Proof:** The proof uses case analysis with respect to the types of the regions involved in the adjacency. The number of adjacencies in each case is bounded by $O(n)$.

Let us first count the adjacencies involving a region, say $R$, from $V_1$. A region $R'$ adjacent to the cube $R$ entirely covers one of the six sides of $R$, regardless of the type of $R'$. Charging the adjcency to the covered side leads to at most $O(n)$ chargings, and, hence at most $O(n)$ adjacencies involving a region from $V_1$.

Now consider an adjacency $(R, R') \in V_2 \times V_3$; $R$ is a (possibly semi-infinite) part of a $z$-column, and $R'$ is unbounded in the $z$-direction. $R$ and $R'$ must be either $x$-adjacent or $y$-adjacent. The restriction on the type of adjacency and the unboundedness of $R'$ in the $z$-direction establish that $R'$ completely covers one of the four sides of $R$ parallel to the $(x, z)$- or $(y, z)$-plane. Charging the adjacency to the entirely covered side of $R$ implies that the number of adjacencies $(R, R') \in V_2 \times V_3$ is $O(n)$.

The regions involved in an adjacency $(R, R') \in V_3 \times V_3$ are both obtained by lifting a rectangular region in the $(x, y)$-plane orthogonally into the $z$-dimension. Hence, the rectangular intersections of $R$ and $R'$ with the plane $z = 0$ must be adjacent in the planar subdivision of $z = 0$. The planar arrangement in $z = 0$ is a planar graph with $O(n)$ edges and vertices, dividing the plane into $O(n)$ regions with a total of $O(n)$ adjacencies. Hence, the number of adjacencies $(R, R') \in V_3 \times V_3$ is $O(n)$.

The number of the remaining adjacencies $(R, R') \in V_2 \times V_2$ is more difficult to bound. Both $R$ and $R'$ are (possibly semi-infinite) parts of different $z$-columns. $R$ and $R'$ must be either $x$-adjacent or $y$-adjacent. It is easily seen that $R$ and $R'$ are in one of the two relative positions depicted in in Figure 7.7. In the left case, one side involved in the adjacency is contained in the other involved one. We charge the adjacency to the covered side (the dashed one in Figure 7.7). Each side can only be charged once. The number of sides of regions of $V_2$ is $O(n)$, so the number of adjacencies $(R, R') \in V_2 \times V_2$ of the first kind is $O(n)$. In the complimentary case, neither one of the sides is completely contained in the other one. In this case, however, one edge bounding the involved side of $R$ is contained in the interior of the

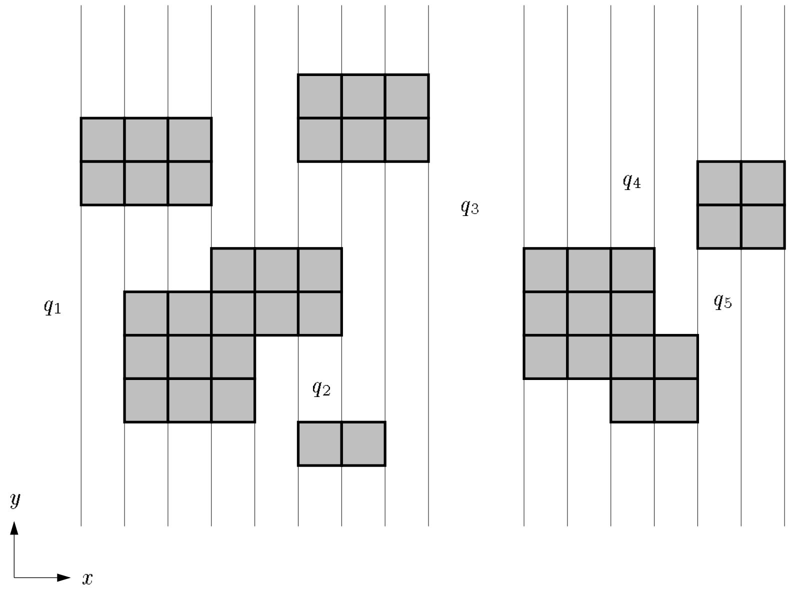


Figure 7.6: The regions $q_1, \ldots, q_5$ are regions in the vertical decomposition of the complement of the column cross-sections (the grey squares). The liftings of these regions are elements of $V_3$.

every consecutive pair of squares in non-adjacent slabs, adding semi-infinite regions (like $q_5$ and $q_4$) for all first and last squares in a slab, and adding a bounded region (like $q_2$) for every consecutive pair of non-adjacent squares in a single slab. The sketched computation of $V_3$ takes $O(n)$ time, provided that the ordered sequence of cubes from $V_1$ is given.

**Lemma 7.20** *$V_3$ consists of $O(n)$ constant-complexity regions $R$ with $Cov(R) = \emptyset$; the computation of $V_3$ takes $O(n \log n)$ time.*

The results obtained so far show that the regions of $V_1 \cup V_2 \cup V_3$ have constant-complexity and constant-size coverage. In addition, the regions collectively partition $\mathbb{R}^3$, which makes $V_1 \cup V_2 \cup V_3$ an adequate choice for a cc-partition of the workspace $\mathrm{W} = \mathbb{R}^3$, so we choose $V_{\mathrm{W}} = V_1 \cup V_2 \cup V_3$.

**Lemma 7.21** *$V_{\mathrm{W}} = V_1 \cup V_2 \cup V_3$ is a cc-partition of size $O(n)$ of $\mathrm{W}$ with the obstacles $\mathcal{E}$; the computation of $V_{\mathrm{W}}$ takes $O(n \log n)$ time.*

The cc-partition of $\mathrm{W} = \mathbb{R}^3$ by the regions of $V_{\mathrm{W}}$ has a recursive structure which turns out to be useful in the sequel. At the upper level, the workspace W is divided into slices, separated by planes $x = h\rho$ ($h \in \mathbf{Z}$). A slice is either a region from $V_3$ (like $q_1, q_3$ in Figure 7.6) or it is divided into levels by planes $y = i\rho$ ($i \in \mathbf{Z}$) and has width $\rho$. A level is either a region from $V_3$ (like $q_2, q_4, q_5$) or it

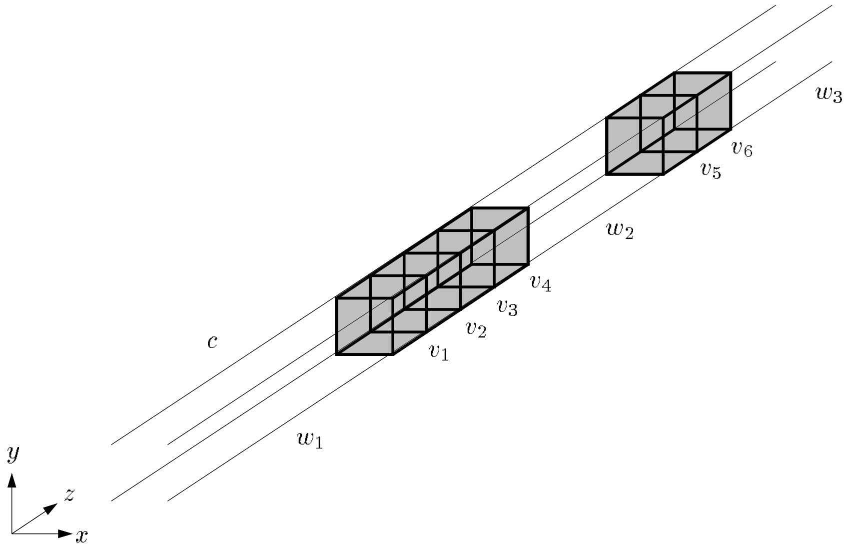


Figure 7.5: The column $c$ contains cubes $v_1, \ldots, v_6$, all being elements of $V_1$; the regions $w_1$, $w_2$, and $w_3$ are members of $V_2$.

The problem of finding a partition of the complement $\mathbb{R}^3 \setminus (\cup_{R \in V_1 \cup V_2} R)$ of the $z$-columns is a problem that can be solved in the plane $z = 0$, as such a partition can be obtained by orthogonally lifting a planar subdivision of the complement of the square intersections of the columns with $z = 0$. The lifting preserves the asymptotic complexity of the regions in the partition, so it suffices to find a partition of the plane into constant-complexity regions.

Consider the plane $z = 0$. The column cross-sections are squares of the form $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho]$, with $h, i \in \mathbf{Z}$. Figure 7.6 shows an example of a plane $z = 0$ with column cross-sections. To partition the complement of the squares into constant-complexity regions, we simply extend vertical walls, parallel to the $y$-axis, through the vertical edges of the squares. Note that no more than $\mathcal{O}(n)$ walls are extended, partitioning the complement of the squares into at most $\mathcal{O}(n)$ constant complexity regions. The orthogonal liftings of these regions into 3-space are constant-complexity regions that collectively partition the three-dimensional complement of the columns.

The ordered sequence of cubes of $V_1$ again turns out a useful tool in the computation. The restriction of the cubes to the first two coordinates turns the sequence into the lexicographical ordering of the squares resulting from the intersection of the $z$-columns with the plane $z = 0$. Hence, the squares appear from left to right (see Figure 7.6) and, within a vertical slab of the plane, from bottom to top. Using this sequence of squares, it is not hard to compute the decomposition of the plane by a single scan of the sequence of squares: adding an unbounded region (like $q_3$) for

**Lemma 7.18** *$V_1$ consists of $O(n)$ constant-complexity regions $R$ with $|Cov(R)| = O(1)$; the computation of $V_1$ takes $O(n \log n)$ time.*

At this stage, the remaining task is to find a way of efficiently partitioning the closure of the complement $\mathbb{R}^3 \setminus (\cup_{R \in V_1} R)$ into a small number of constant-complexity regions. The efficient subdivision of the complement outlined below proceeds in two phases. The first phase will be such that, upon its completion, all workspace columns of the form $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times \mathbb{R}$ that contain a cube $R \in V_1$ are partitioned into regions with constant complexity and constant-size coverage. This approach reduces the remaining step, that is, the subdivision of the complement of the columns, to the essentially two-dimensional problem of subdividing the planar complement of the intersections of the columns with the plane $z = 0$. Both phases result in a collection of $O(n)$ additional regions, which are computable in time $O(n)$ time using the ordered sequence of cubes from $V_1$.

The first of the remaining two steps completes the partition of the union of the $z$-columns through the cubes of $V_1$. A $z$-column through a cube $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times [j\rho, j\rho + \rho]$ is the infinite extension $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times \mathbb{R}$ of that cube in the $z$-direction. The complement $c \setminus (\cup_{R \in V_1} R)$ of the cubes in each $z$-column $c$ through a cube of $V_1$ consists of a collection of (possibly semi-infinite) maximal connected components of height and width $\rho$ (see Figure 7.5). The regions that constitute the set $V_2$ are the closures of all such maximal connected components in the $O(n)$ $z$-columns through cubes of $V_1$. To understand the contribution to $V_2$ of a single $z$-column $c$, consider the example of Figure 7.5. The grey cubes $v_1, \ldots, v_6$ in the $z$-column $c$ all belong to $V_1$. The complement of the cubes, that is, the white space between the cubes, consists of three connected components $w_1, w_2, w_3$ of height and width $\rho$. The closures of these regions form the contribution of the $z$-column $c$ to $V_2$.

From the construction of the regions of $V_2$ it is clear that the number of regions $R' \in V_2$ contributed by a $z$-column $c$ can exceed the number of cubes $R \in V_1$ in $c$ by at most one, as each pair of subsequent regions from $V_2$ in $c$ must be separated by at least one cube from $V_1$. As a consequence, the total number of regions in $V_2$ is of the same order of magnitude as the number of cubes in $V_1$. The computation of the regions of $V_2$ contributed by a single $z$-column $c$ is simple if the ordered sequence of cubes (from $V_1$) in $c$ is given. Fortunately, this cube sequence appears as a subsequence in the lexicographical ordering of all cubes of $V_1$. As a result, the regions of $V_2$ contributed by all $z$-columns can be computed by a single scan of the full $O(n)$-length sequence.

**Lemma 7.19** *$V_2$ consists of $O(n)$ constant-complexity regions $R$ with $Cov(R) = \emptyset$; the computation of $V_2$ takes $O(n \log n)$ time.*

Note that the computation time of $O(n \log n)$ incorporates the ordering of the cubes from $V_1$.

subdivision is such that the regions can be ordered lexicographically. This ordering on the regions simplifies many of the computations.

Let $V_1$ be the set of cubic cc-partition regions; $V_1$ is defined by

$$V_1 = \bigcup_{E \in \mathcal{E}} V_1(E),$$

where

$$\begin{aligned} V_1(E) = \{ & [h\rho, h\rho+\rho] \times [i\rho, i\rho+\rho] \times [j\rho, j\rho+\rho] \\ & | h, i, j \in \mathbf{Z} \wedge [h\rho, h\rho+\rho] \times [i\rho, i\rho+\rho] \times [j\rho, j\rho+\rho] \cap B(G(E, \rho_{\mathcal{B}})) \neq \emptyset \}. \end{aligned}$$

Lemma 7.16 proves the linear bound on the number of regions in $V_1$

**Lemma 7.16** $|V_1| = O(n)$.

**Proof:** Let us bound the number of cubes in $V_1(E)$, for some $E \in \mathcal{E}$. The minimal enclosing sphere radius of $E$ lies in the range $[\rho, u\rho]$. As a result, no two points in $E$ are more than $2u\rho$ apart, and, hence, no two points in $G(E, \rho_{\mathcal{B}})$ are more than $2u\rho + 2\rho_{\mathcal{B}} \leq 2(u+b)\rho$ apart, which in turn implies that the length, width and height of the bounding box $B(G(E, \rho_{\mathcal{B}}))$ do not exceed $2(u+b)\rho$. Such a box is certainly embedded in an orthogonal cluster of $(2(u+b)+1)^3$ cubes of side length $\rho$, which is a constant number. The number of elements in $V_1(E)$ is bounded by this constant. Summing over all sizes of set $V_1(E)$, yields a bound of $O(n)$ on the size of the set $V_1$. □

Lemma 7.17 confirms the intuition that the cubes of $V_1$ are so small that they can only be intersected by a constant number of grown obstacles.

**Lemma 7.17** *For all* $R \in V_1$: $|Cov(R)| = O(1)$.

**Proof:** Any obstacle $E \in \mathcal{E}$, for which $G(E, \rho_{\mathcal{B}})$ intersects $R \in V_1$, must itself intersect the enclosing cube $R'$ of $R$ obtained by growing $R$ by $\rho_{\mathcal{B}}$ in all three axis-parallel directions. The resulting $R'$ is a cube with side length $\rho + 2\rho_{\mathcal{B}} \leq (2b+1) \cdot \rho$, and, hence, with diameter at most $(2b+1)\sqrt{3} \cdot \rho$. By Theorem 2.9, the number of obstacles in $\mathcal{E}$, which all have minimal enclosing sphere radi at least $\rho$, intersecting the cube $R'$ is constant. So, $|Cov(R)| = O(1)$. □

The definition of $V_1$ as the union of all sets $V_1(E)$ ($E \in \mathcal{E}$) gives a clue on a straightforward but efficient computation of the set $V_1$. A first step computes all constant-cardinality sets $V_1(E)$. Each set $V_1(E)$ of cubes intersecting the grown obstacle $G(E, \rho_{\mathcal{B}})$ is trivially computable in constant time from $E$ and $\rho_{\mathcal{B}}$. The resulting sets of cubes are not disjoint; a cube $R$ may occur in more than one set $V_1(E)$. To eliminate multiple copies of a single cube, we simply sort the cubes of all sets $V_1(E)$ lexicographically. Multiple copies of a single cube appear consecutively in the ordered sequence, and are easily filtered out. In conclusion, the $O(n \log n)$ time to sort the $O(n)$ cubes of the sets $V_1(E)$ determines the time bound for the computation of $V_1$.

## 7.4 Similarly-sized arbitrary obstacles in 3-space

The motion planning algorithm in the previous section for a robot amidst arbitrary obstacles has a relatively high running time compared to the free space complexity. It is therefore interesting to see what realistic additional assumptions may lead to a relevant improvement of the performance. This section shows an interesting example of such a realistic assumption, namely that the obstacles have comparable sizes. The assumption is realistic because in many practical situations, the largest obstacle in the workspace will not be more than a constant factor bigger than the smallest obstacle. The addition of the assumption to the mildly constrained setting of the previous section leads to a surprising improvement of the performance. The resulting motion planning algorithm computes an optimal $O(n)$ cell decomposition in nearly-optimal $O(n \log n)$ time. The following problem statement fixes the setting of the results in this section.

> *A constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom ($f \geq 3$) and reach $\rho_{\mathcal{B}}$ moves freely in the workspace $\mathrm{W} = \mathbb{R}^3$ amidst a collection $\mathcal{E}$ of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W}$ with minimal enclosing sphere radii in the range $[\rho, u\rho]$, for some constants $k \geq 1$ and $u \geq 1$. The system is constrained by the inequality $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some fixed constant $b \geq 0$.*

Again, the goal is to compute a (small) cc-partition of the workspace. The bounded ratio between the size of the smallest and largest obstacle in $\mathcal{E}$ provides the opportunity of a simple and structured cc-partition, consisting of axis-parallel rectangloid regions. The corners of the rectangloid regions are restricted to the points of the regular orthogonal grid $\mathcal{G}(\rho)$ (with resolution $\rho$). More specifically, the rectangloid regions of the subdivision are either cubes with side length $\rho$, or (possibly semi-infinite) rectangloids of width and height $\rho$, or rectangloids that are unbounded in both $z$-directions. All regions of the latter two types have empty coverage. The number of each of the three types of regions in the subdivision is bounded by $O(n)$. Moreover, the number of adjacencies is equally low: $O(n)$. The rectangloid subdivision is as such an optimal partition of the workspace.

The basic idea behind the rectangloid subdivision is to embed the bounding boxes $B(G(E, \rho_{\mathcal{B}}))$ of all grown obstacles $G(E, \rho_{\mathcal{B}})$ in cubes of the form $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times [j\rho, j\rho + \rho]$, where $h, i, j \in \mathbf{Z}$. The idea seems promising for two different reasons. On the one hand, the cubes are small enough to certify constant-size coverage, while, on the other hand, the cubes are large enough to be able to embed each of the (bounded-size) grown obstacles in a constant number of cubes with pair-wise disjoint interiors. As a result, the total number of cubes is linear in the number obstacles. The complement of the cubes clearly has empty intersection with the grown obstacles. The regions of the two non-cubic types serve as a means of efficiently subdividing this complement. All these regions have empty coverage. The

To find an arbitrary region $R$ intersected by $G(E, \rho_{\mathcal{B}})$, we take a point $p \in G(E, \rho_{\mathcal{B}})$. Next, we perform a point location query with $p = (p_x, p_y, p_z)$ in the partition $V_{\mathrm{W}}$, which, by the orthogonality of the partition can be decomposed into three binary searches to find $X_h \ni p_x$, $Y_i \ni p_y$, and $Z_j \ni p_z$ in time $O(\log n)$. The Cartesian product $X_h \times Y_i \times Z_j$ contains the point $p$, and, hence, intersects $G(E, \rho_{\mathcal{B}})$, so we add $E$ to $Cov(R)$.

In preparation for a second phase, we create a set $Ev$ yet consisting of just one element: $R$. We repeatedly extract an element $R$ from $Ev$ for further processing, until $Ev$ is empty. We compute the $O(1)$ neighbors of $R$ and verify for each neighbor $R'$ if $R'$ intersects $G(E, \rho_{\mathcal{B}})$. If this is the case and the neighbor has not been treated yet (which can be tested by marking the regions visited), then we add $E$ to $Cov(R')$ and the neighbor $R'$ itself to $Ev$.

The above search through the regions considers a superset of the collection of $m_E$ regions $R \in V_{\mathrm{W}}$ satisfying $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. More precisely, it also considers all regions adjacent to regions $R$ with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. Still, the total number of regions that are considered is $O(m_E)$. As the amount of work per region is constant, the total time spent in the search is $O(m_E)$.

It remains to bound the number $m_E$ of regions $R \in V_{\mathrm{W}}$ with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. For each region $R$ with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$, we add $E$ to $Cov(R)$. Hence, the sum of all $m_E$ over all grown obstacles $G(E, \rho_{\mathcal{B}})$ equals the sum of all $|Cov(R)|$ over all regions $R$. As $|Cov(R)| = O(1)$ for all $R \in V_{\mathrm{W}}$, the latter sum amounts to $O(n^3)$. As a result, the sum of all $m_E$ equals $O(n^3)$, and, hence, all searches together take $O(n^3)$ time. The $n$ point location queries (in the orthogonal cc-partition) to identify a starting region for each of the $n$ searches require additional $O(n \log n)$ time in total. As a result, the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function $Cov$ can be computed in $\Theta(n^3)$ time.

**Lemma 7.14** *The computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$ takes $\Theta(n^3)$ time.*

The substitution of the computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the function $Cov$ in the first step of the algorithm FatMot leads, by Theorem 6.8, to a motion planning algorithm that decomposes FP into constant-complexity subcells in time $T(n) = \Theta(n^3)$ time. The number of subcells and subcell adjacencies is in the worst case of the same order of magnitude as the number of regions and adjacencies in the cc-partition.

**Theorem 7.15** *Let $k \geq 1$ and $b \geq 0$ be constants and let $\mathcal{E}$ be a collection of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W} = \mathbb{R}^3$ with minimal enclosing sphere radii at least $\rho$. Algorithm FatMot solves the motion planning problem for any constant-complexity robot $\mathcal{B}$ with $f \geq 3$ degrees of freedom and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ amidst $\mathcal{E}$ in time $\Theta(n^3)$. The connectivity graph $\mathrm{CG} = (V_C, E_C)$ of the resulting decomposition of* FP *into simple subcells has size $O(n^3)$.*

**Proof:** $V_W$ and $E_W$ are easily seen to have size $\Theta(n^3)$. The remaining task is to prove that the regions of $V_W$ partition the workspace W into regions with constant complexity and constant-size coverage. The first part is trivial as a rectangloid has constant complexity; the second part is less obvious.

The structure of the partition by the supporting planes of the bounding boxes of the grown obstacles is such that an arbitrary rectangloid region $R \in V_W$ either lies in the exterior of all bounding boxes, in which case it has empty coverage, or it lies entirely in the interior of a number of bounding boxes of grown obstacles. Let, in the latter case, $D$ be the set of all obstacles $E$ for which $R \subseteq B(G(E, \rho_{\mathcal{B}}))$. $D$ may have more than a constant number of elements.

Let $E^-$ be the obstacle in $D$ with the smallest minimal enclosing sphere radius, say, $\rho^-$. We first prove that no obstacles with minimal sphere radii smaller than $\rho^-$ belong to $Cov(R)$. Assume, for a contradiction, that $E^*$ has minimal enclosing sphere radius $\rho^* < \rho^-$ and satisfies $E^* \in Cov(R)$. By the definition of coverage, this means that, $R \cap G(E^*, \rho_{\mathcal{B}}) \neq \emptyset$. But then, since $G(E^*, \rho_{\mathcal{B}}) \subseteq B(G(E^*, \rho_{\mathcal{B}}))$, also $R \cap B(G(E^*, \rho^{\mathcal{B}})) \neq \emptyset$. So, $E^*$ must belong to $D$, violating the assumption that $E^-$ is the obstacle in $D$ with the smallest minimal enclosing sphere radius.

From $E^- \in D$ it follows that $R \subseteq B(G(E^-, \rho_{\mathcal{B}}))$. The minimal enclosing hypersphere radius $\rho^-$ of $E^-$ implies that the length of none of the sides of $B(G(E^-, \rho_{\mathcal{B}}))$ exceeds $2\rho^- + 2\rho_{\mathcal{B}} \leq 2\rho^- + 2b\rho \leq (2b+2)\rho^-$. As a result, the length of none of the sides of $R \subseteq B(G(E^-, \rho_{\mathcal{B}}))$ exceeds $(2b+2)\rho^-$ as well. An obstacle $E'$ with minimal enclosing sphere radius $\rho' \geq \rho^-$ whose corresponding grown obstacle $G(E', \rho_{\mathcal{B}})$ intersects $R$, must itself intersect the enclosing rectangloid $R^G \supseteq R$, obtained by growing $R$ by $\rho_{\mathcal{B}} \leq b\rho \leq b\rho^-$ in all (six) axis-parallel directions, so $R^G = R \ominus C_{O,\rho_{\mathcal{B}}}$. The edges of $R^G$ have length at most $(4b+2)\rho^-$. By Theorem 2.9, the number of obstacles $E'$ with mes-radius $\rho' \geq \rho^-$ intersecting the rectangloid region $R^G$ is bounded by a constant, and hence the number of grown obstacles $G(E', \rho_{\mathcal{B}})$ intersecting $R$ is bounded by a constant, meaning that $|Cov(R)| = O(1)$. □

The single algorithmic issue that is to be solved concerns the computation of the coverage $Cov(R) \subset \mathcal{E}$ of each region $R \in V_W$, because the regions of $V_W$ and the adjacencies of $E_W$ can be trivially extracted in time $\Theta(n^3)$ from the the three ordered sequences of planes. Instead of taking a single region $R \in V_W$ and computing all grown obstacles $G(E, \rho_{\mathcal{B}})$ that intersect it, we choose a more or less inverse approach here: we take a grown obstacle region $G(E, \rho_{\mathcal{B}})$ and compute all regions $R \in V_W$ that are intersected by it, and add $E$ to all corresponding sets $Cov(R)$ under construction. In other words, we want to determine all regions $R$ with $Cov(R) \ni E$. The approach is to identify a single region $R$ intersected by $G(E, \rho_{\mathcal{B}})$ and then use this region as a basis for searching the adjacency graph $E_W$ to find the entire connected set of regions intersected by $G(E, \rho_{\mathcal{B}})$. The correctness of the approach relies on the connectedness of $G(E, \rho_{\mathcal{B}})$, which is implied by the connectedness of $E$.

The main implication of the fact that the obstacles in the workspace have arbitrary shape is that they cannot be tightly wrapped by some polyhedron of constant complexity. A consequence is that we are no longer able to construct a low complexity arrangement in a first decomposition step which partitions the workspace into regions with constant-size coverage. Instead, we propose a simple direct cc-partition of the workspace leading to a cubic number of regions.

The axis-parallel bounding boxes $B(G(E, \rho_{\mathcal{B}}))$ of the grown obstacles $G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$) partition the workspace with the obstacles $\mathcal{E}$ into regions with constant coverage (which is not completely obvious). The arrangement of boxes has complexity $O(n^3)$. Unfortunately, the 3-cells of the arrangement may very well have more than constant complexity. If, however, we replace the arrangement of bounding boxes by the arrangement of the supporting planes of all bounding box faces, then we obtain rectangloid (constant-complexity) 3-cells without increasing the asymptotic worst-case complexity of the arrangement: the cc-partition of the workspace by the supporting planes has complexity $\Theta(n^3)$.

Each obstacle $E \in \mathcal{E}$ contributes six planes to the arrangement defining the cc-partition. The constant complexity of the obstacle $E$ allows us to compute the supporting planes $x = x'$, $x = x''$, $y = y'$, $y = y''$, $z = z'$, and $z = z''$ of the grown obstacle $G(E, \rho)$ in constant time. For simplicity, we assume that none of the supporting planes is tangent to any other grown obstacle $G(E', \rho_{\mathcal{B}})$ ($E' \neq E$). This extra assumption, however, can be avoided quite easily. After having computed all $2n$ planes parallel to the $(y, z)$-plane, we sort the resulting planes by increasing $x$-coordinates, yielding a sequence $x_1 < \ldots < x_{2n}$. The sequence partitions the real line into $2n + 1$ intervals $X_h$ ($0 \leq h \leq 2n$) with $X_0 = (-\infty, x_1]$, $X_h = [x_h, x_{h+1}]$ for all $1 \leq h < 2n$, and $X_{2n} = [x_{2n}, +\infty)$. A similar treatment of the planes parallel to the $(x, z)$-plane and $(x, y)$-plane results in sequences $y_1 < \ldots < y_{2n}$ and $z_1 < \ldots < z_{2n}$ and two partitions of the real line into intervals $Y_i$ ($0 \leq i \leq 2n$) and $Z_j$ ($0 \leq j \leq 2n$) respectively. The strictly increasing nature of the sequences is due to the assumption that no plane is tangent to two grown obstacles. The three ordered sequences define a cc-partition graph consisting of a set $V_{\mathrm{W}}$ of regions

$$V_{\mathrm{W}} = \{\, X_h \times Y_i \times Z_j \mid 0 \leq h, i, j \leq 2n \,\},$$

and a set $E_{\mathrm{W}}$ of adjacencies:

$$\begin{aligned} E_{\mathrm{W}} = \quad & \{\, (X_h \times Y_i \times Z_j \,,\, X_{h+1} \times Y_i \times Z_j) \mid 0 \leq h < 2n \wedge 0 \leq i, j \leq 2n \,\} \\ \cup \quad & \{\, (X_h \times Y_i \times Z_j \,,\, X_h \times Y_{i+1} \times Z_j) \mid 0 \leq i < 2n \wedge 0 \leq h, j \leq 2n \,\} \\ \cup \quad & \{\, (X_h \times Y_i \times Z_j \,,\, X_h \times Y_i \times Z_{j+1}) \mid 0 \leq j < 2n \wedge 0 \leq h, i \leq 2n \,\} \end{aligned}$$

**Lemma 7.13** *$V_{\mathrm{W}}$ is a cc-partition of size $\Theta(n^3)$ of* W *with the obstacles $\mathcal{E}$; $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $\Theta(n^3)$.*

proportional to the number of regions. As a consequence, the time to compute the cc-partition graph $(V_W, E_W)$ and the function $Cov$ is dominated by the running time of the vertical decomposition algorithm: $O(n^2 \log n)$.

**Lemma 7.11** *The computation of the cc-partition graph* $(V_W, E_W)$ *and the corresponding function* $Cov : V_W \to \mathcal{P}(\mathcal{E})$ *takes* $T(n) = O(n^2 \log n)$ *time.*

Substitution of the computation of $(V_W, E_W)$ and $Cov$ outlined above for the first (abstract) step of the algorithm FatMot yields, by Theorem 6.8 and Lemmas 7.10 and 7.11, a motion planning algorithm with running time $O(n^2 \log n)$. The algorithm decomposes the free space of a robot amidst fat polyhedral obstacles into $O(n^2)$ subcells of constant-complexity, defining $O(n^2)$ pairwise adjacencies.

**Theorem 7.12** *Let* $k \geq 1$ *and* $b \geq 0$ *be constants and let* $\mathcal{E}$ *be a collection of* $k$*-fat constant-complexity obstacles* $E \subseteq \mathrm{W} = \mathbb{R}^3$ *with minimal enclosing sphere radii at least* $\rho$*. Algorithm FatMot solves the motion planning problem for any constant-complexity robot* $\mathcal{B}$ *with* $f \geq 3$ *degrees of freedom and reach* $\rho_{\mathcal{B}} \leq b \cdot \rho$ *amidst* $\mathcal{E}$ *in time* $O(n^2 \log n)$*. The connectivity graph* $\mathrm{CG} = (V_C, E_C)$ *of the resulting decomposition of* FP *into simple subcells has size* $O(n^2)$*.*

The gap between the (linear) complexity of the free space and the (quadratic) size of the connectivity graph of the FP decomposition shows that the cell decomposition is not optimal. An interesting open problem is therefore to attempt to bridge the gap by exploring alternative cc-partitions of the workspace. A smaller cc-partition would probably require a partitioning strategy that differs completely from the two-step approach of first subdividing the workspace with the obstacles into regions with constant-size coverage and then refining the regions in the subdivision to constant-complexity regions. The next two sections show examples of cc-partitions that are not obtained via the two-step approach.

## 7.3 Arbitrary obstacles in 3-space

The setting that is considered in this section differs from the setting of the previous section in that the obstacles are not required to be polyhedral, but instead only assumed to be of constant complexity. The only general methods that could solve such a problem are those by Schwartz and Sharir [84] and Canny [20]. Here, it is shown that an instance of the algorithm FatMot for a bounded-size robot among fat obstacles exists with running time $\Theta(n^3)$, independent of the actual number $f$ of degrees of freedom of the robot. The setting is fixed by the following description.

> *A constant-complexity robot* $\mathcal{B}$ *with* $f$ *degrees of freedom* ($f \geq 3$) *with reach* $\rho_{\mathcal{B}}$ *moves freely in the workspace* $\mathrm{W} = \mathbb{R}^3$ *amidst a collection* $\mathcal{E}$ *of* $k$*-fat constant-complexity obstacles* $E \subseteq \mathrm{W}$ *with minimal enclosing sphere radii at least* $\rho$*, for some constant* $k \geq 1$*. The system is constrained by the inequality* $\rho_{\mathcal{B}} \leq b \cdot \rho$*, for some fixed constant* $b \geq 0$*.*

magnitude as the complexity of the arrangement (because a triangle arrangement is a planar graph). Hence, the numbers of adjacencies on the inward- and outward-facing sides of a triangle $t$ are $O(m_t)$ and $O(n_t)$ respectively.

We recall that two adjacent regions share a common boundary that is either embedded in a triangle or embedded in a vertical wall. Each non-empty intersection of two 2-faces in arrangements on either side of a triangle represents an adjacency of the first type. The number of non-empty intersections on opposite sides of a triangle $t$ is $O(m_t \cdot n_t) = O(n_t)$ due to $m_t = O(1)$. Hence, the number of adjacencies in $E_\mathrm{W}$ of the first type equals $\sum_t O(n_t) = O(n^2)$. To find the number of adjacencies of the second type, note that two adjacent regions whose common boundary is part of a vertical wall have adjacent floors or adjacent ceilings in a triangle arrangement. Hence, the total number of adjacencies of 2-faces in all triangle arrangements supplies an upper bound on the number of pairs of regions in $V_\mathrm{W}$ that share a vertical face. The total number of adjacencies of 2-faces on a triangle $t$ is $O(m_t + n_t) = O(n_t)$. Hence, the number of adjacencies in $E_\mathrm{W}$ of the second (and last) type equals $\sum_t O(n_t) = O(n^2)$ as well. Lemma 7.10 summarizes the bounds on the sizes of $V_\mathrm{W}$ and $E_\mathrm{W}$.

**Lemma 7.10** *$V_\mathrm{W}$ is a cc-partition of size $O(n^2)$ of $\mathrm{W} = \mathbb{R}^3$ with the polyhedral obstacles $\mathcal{E}$; $E_\mathrm{W} = \{(R, R') \in V_\mathrm{W} \times V_\mathrm{W} | \partial R \cap \partial R' \neq \emptyset\}$ has size $O(n^2)$.*

After applying the $O(n^2 \log n)$ vertical decomposition algorithm, we traverse the $O(n^2)$ constant-complexity regions of the decomposition using the quad-edge structure storing the triangle arrangements and the walls, starting from an arbitrarily chosen region. The aim of the traversal is to extract explicit descriptions of the regions in $V_\mathrm{W}$, report the region adjacencies of $E_\mathrm{W}$, and compute the coverages $Cov(R)$ of the regions $R \in V_\mathrm{W}$. The latter part of the computation deserves some additional explanation. Instead of attempting to compute the constant-size coverages $Cov(R) = \{E \in \mathcal{E} | G(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$ directly, we first compute the constant cardinality sets $\{E \in \mathcal{E} | H(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$. The constant cardinality of these set is due to the property that each region $R \in V_\mathrm{W}$ is a subset of a 3-face $A$ of the arrangement $\mathcal{A}(H)$, which satisfy $|\{E \in \mathcal{E} | H(E, \rho_\mathcal{B}) \cap A \neq \emptyset\}| = O(1)$, by Lemma 7.6. The necessary data for the computation of the sets $\{E \in \mathcal{E} | H(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$ are available from the decomposition, contrary to the data for the computation of $Cov(R)$. Throughout the traversal of the decomposition we use the fact that adjacent regions are intersected (or actually enclosed) by the same set of Minkowski differences $H(E, \rho_\mathcal{B})$ unless their common boundary is contained in a triangle $t$ that is part of some Minkowski difference boundary $\partial H(E, \rho_\mathcal{B})$, in which case the sets of intersecting Minkowski differences differ by exactly $\{E\}$: the label of the triangle $t$. The set $\{E \in \mathcal{E} | H(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$ is a superset of $Cov(R) = \{E \in \mathcal{E} | G(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$, since $G(E, \rho_\mathcal{B}) \subseteq H(E, \rho_\mathcal{B})$. The latter set is obtained in constant time from the former set by elimination of the $O(1)$ obstacles $E \in \{E \in \mathcal{E} | H(E, \rho_\mathcal{B}) \cap R \neq \emptyset\}$ that satisfy $E \cap G(E, \rho_\mathcal{B}) = \emptyset$. The traversal of the $O(n^2)$ regions in the decomposition requires, taking into account the limited amount of work per traversed region, time

and an edge of another triangle $t'$ incident to a vertex of $t$. We define $\gamma = \min(\gamma_1, \gamma_2)$. We construct a set $F(T_{\mathcal{A}(H)})$ by applying the following procedure to every $t \in T_{\mathcal{A}(H)}$. Let $v_1, v_2, v_3$ be the vertices of $t$.

1. The planes $\pi_1, \pi_2, \pi_3$ through $v_1, v_2, v_3$ that make a positive angle $\gamma/2$ with the top side (facing $z = \infty$) of $t$ intersect in a point $v$. If the distance from $v$ to $t$ is strictly less than $\epsilon$, then the tetrahedron is defined by the vertices $v, v_1, v_2, v_3$.

2. If the distance from $v$ to $t$ is at least $\epsilon$, then we take the half-line $h$ through $v$ and perpendicular to and ending on $t$. The tetrahedron is defined by $v_1, v_2, v_3$ and the unique point $v'$ on the half-line $h$ with distance $\epsilon/2$ to $t$.

The application of the above two-step process to the triangles of $T_{\mathcal{A}(H)}$ results in a collection of tetrahedra with disjoint interiors. Property 7.9 is a compact statement of the result. The property contains a minor abuse of the definition of the set $F(T_{\mathcal{A}(H)})$ as it is interpreted to be the set of flat tetrahedra instead of the triangular faces of the tetrahedra. Let the (open) interior of the closed set $\tau$ be denoted by $int(\tau)$.

**Property 7.9** $\forall\, \tau, \tau' \in F(T_{\mathcal{A}(H)}) :\ int(\tau) \cap int(\tau') = \emptyset$.

Before applying the vertical decomposition algorithm to the triangles of $F(T_{\mathcal{A}(H)})$, we must convince ourselves that these triangles partition the workspace W with the obstacles $\mathcal{E}$ into regions with constant-size coverage. Fortunately, this follows directly from the construction of $F(T_{\mathcal{A}(H)})$ from the triangles of $T_{\mathcal{A}(H)}$, which already define a partition of into regions with constant-size coverage. The addition of disjoint triangles to the partition can only lead to a refinement of the regions into smaller regions, with smaller or equally-sized coverages.

Application of the decomposition algorithm to the $O(n)$ disjoint triangles of $F(T_{\mathcal{A}(H)})$ yields a full vertical decomposition of complexity $O(n^2)$ and therefore consisting of $O(n^2)$ regions with constant complexity. The decomposition regions are appointed to be the regions of $V_{\mathrm{W}}$. The coverage of each region $R \in V_{\mathrm{W}}$ has constant size as $R$ is a subset of a region in the partition by the triangles of $F(T_{\mathcal{A}(H)})$, which were shown to have constant-size coverage in the previous paragraph.

Let us now bound the size of the set $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$. The complexities of the triangle arrangements are crucial to the analysis of the adjacencies, so we first study these complexities in more detail. The complexity of the entire full vertical decomposition is $O(n^2)$. As a consequence, the cumulative complexity of all triangle arrangements is $O(n^2)$ as well. Each triangle $t \in F(T_{\mathcal{A}(H)})$ has a side facing the interior of the tetrahedron $\tau$ it belongs to and a side facing outward. The complexity $m_t$ of the arrangement on the inward-facing side of $t$ is constant, because we have seen that only a constant number of walls define this arrangement: $m_t = O(1)$, for all $t$. The complexity $n_t$ of the arrangement on the outward-facing side of a single triangle $t$, however, can be as high as $O(n^2)$. The number of adjacencies of 2-faces in a triangle arrangement is of the same order of

collectively partition W with the polygonal obstacles $\mathcal{E}$ into $O(n^2)$ regions with constant-complexity and constant-size coverage. Hence, the collection of these regions would make an appropriate choice for a set $V_{\mathrm{W}}$. Unfortunately, the decomposition does not give us a low number of adjacencies as well.

The common boundary of two adjacent regions in the full vertical decomposition is either embedded in a vertical wall or embedded in a triangle. Although the number of adjacencies of the first type can be bounded by $O(n^2)$, it seems impossible to obtain a similar bound on the number of adjacencies of the latter type. Walls extended from other triangles' edges end on both sides of a triangle $t \in T_{\mathcal{A}(H)}$ and define arrangements of line segments on these sides. The complexity of a single arrangement and its planar vertical decomposition can be as high as $O(n^2)$, and, hence, the number of 2-faces in the vertically decomposed arrangement is bounded by $O(n^2)$ only. These 2-faces are the floors (or ceilings) of the full vertical decomposition regions. Each non-empty intersection of 2-faces on either side of a single triangle corresponds to an adjacency of two regions. In general, two subdivisions of a single triangle $t$ into $m_t$ and $n_t$ 2-faces could easily give rise to $O(m_t \cdot n_t)$ non-empty intersections of pairs of 2-faces. At present, it is unclear if the specific properties of the full vertical decomposition make it possible to bound the cumulative number of intersections on all $O(n)$ triangles, and, hence, the number of region adjacencies, by anything close to $O(n^2)$.

An elegant way to overcome the problem outlined above is by replacing all triangles of $T_{\mathcal{A}(H)}$ by tetrahedra that are sufficiently flat to prevent them from intersecting. The tetrahedra have the initial triangles of $T_{\mathcal{A}(H)}$ as one of their faces. The triangular faces of the tetrahedra are collected in the set $F(T_{\mathcal{A}(H)})$; $F(T_{\mathcal{A}(H)})$ satisfies $F(T_{\mathcal{A}(H)}) \supseteq T_{\mathcal{A}(H)}$ and $|F(T_{\mathcal{A}(H)})| = 4 \cdot |T_{\mathcal{A}(H)}|$. Hence, the size of $F(T_{\mathcal{A}(H)})$ is still $O(n)$. Moreover, $F(T_{\mathcal{A}(H)})$ is again a set of non-intersecting though touching triangles, which now have the simple but beneficial property that one of their sides faces the interior of a tetrahedron $\tau$ (consisting of four triangles from $F(T_{\mathcal{A}(H)})$). The benefit of this property lies in the fact that walls extended from triangles outside $\tau$ are unable to penetrate $\tau$ (and, hence, to end on the inward-facing sides of its triangular faces) as they end upon hitting the outside of $\tau$. The walls inside $\tau$ must therefore either be extended from one of the six edges of $\tau$ itself, or added during the subsequent refinement of the $O(1)$ regions in the vertical decomposition inside $\tau$. Clearly, the refinement introduces no more than a constant number of walls as well. As a result, the $O(1)$ ending walls on the inward-facing side of a triangle define a constant-complexity arrangement on that side.

We first discuss how to replace each triangle $t \in T_{\mathcal{A}(H)}$ by a tetrahedron $\tau$ having $t$ as one of its faces, such that the resulting tetrahedra are non-intersecting. Recall that $T_{\mathcal{A}(H)}$ is a set of triangles that may share a vertex or an edge but do not intersect each others' interiors. Let $\epsilon$ be the minimum distance between any pair of disjoint (non-touching) triangles. Furthermore, let $\gamma_1$ be the minimum over all dihedral angles between pairs of (touching) triangles that share an edge and $\gamma_2$ be the minimum over all angles between the supporting plane of a triangle $t \in T_{\mathcal{A}(H)}$

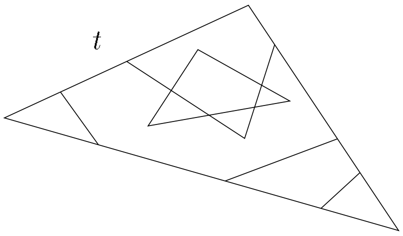


Figure 7.4: The arrangement of walls ending on a side of a triangle, and the planar vertical decomposition of the arrangement.

the arrangement. As the cumulative complexity of all arrangements is $O(n^2)$, the computation of all arrangements takes $O(n^2 \log n)$ time. The floors and ceilings, that is, the bounding faces in vertical direction, of the resulting regions are parts of triangles. More precisely, they are 2-faces in an arrangement of ending walls on a triangle side. Note that the floor and ceiling of a vertical decomposition region have equivalent projections onto the $(x, y)$-plane.

The second step refines the vertical decomposition into a full vertical decomposition, consisting of regions bounded by six (possibly degenerate) planar faces, by adding walls parallel to the $(x, z)$-plane. The refining is obtained through a planar vertical decomposition of all triangle arrangements, in which segments are extended within the triangles parallel to the $y$-axis from every vertex of the arrangement (see Figure 7.4). The additional walls connect corresponding extended segments (that is, with equivalent projections on the $(x, y)$-plane) in the upper and lower bounding triangles of a vertical decomposition region. The entire refining takes $O(n^2 \log n)$ using a sweep of all triangle arrangements. Every region in the full vertical decomposition has a trapezoidal floor and ceiling with equivalent projections onto the $(x, y)$-plane. The remaining four (possibly degenerate) bounding faces are vertical walls: two resulting from the vertical decomposition and two added during the refinement. The representation of the full vertical decomposition computed in the refinement step consists of: (i) all walls $W(e)$ extended from triangle edges $e$ plus all additional (paralleloid) walls parallel to the $(x, y)$-plane, and (ii) for each triangle $t \in T$, the two arrangements of ending walls, both extended from edges and added during the refinement, on either side of $t$.

**Lemma 7.8** *Let $T$ be a set of $n$ non-intersecting triangles in $\mathbb{R}^3$. The full vertical decomposition of $T$ decomposes the arrangement of triangles in $T$ into constant-complexity regions. The decomposition has complexity $O(n^2)$ and can be computed in time $O(n^2 \log n)$.*

Application of the decomposition algorithm to the set of triangles $T_{\mathcal{A}(H)}$ yields a subdivision of the constant-size coverage 3-faces of the arrangement $\mathcal{A}(H)$ into constant-complexity regions. As a result, the regions of the refined subdivision

are alternating polygonal chains of parts of intersection edges and of vertical segments. Both envelopes have complexity $O(n)$, since the upper/lower envelope of a collection of disjoint line segments in the plane has complexity $O(n)$. Each halfwall (between an envelope and the edge) is divided into slabs by vertical line segments connecting the vertices of the envelope to the edge (see Figure 7.3). The vertical segments correspond to intersections with other walls. Clearly, the vertical segments do not increase the complexity of the wall. The envelopes (and the vertical line segments) can be computed in $O(n \log n)$ time, using divide-and-conquer [47]. The computation of all $O(n)$ walls requires $O(n^2 \log n)$ time; the cumulative complexity of the walls is $O(n^2)$.

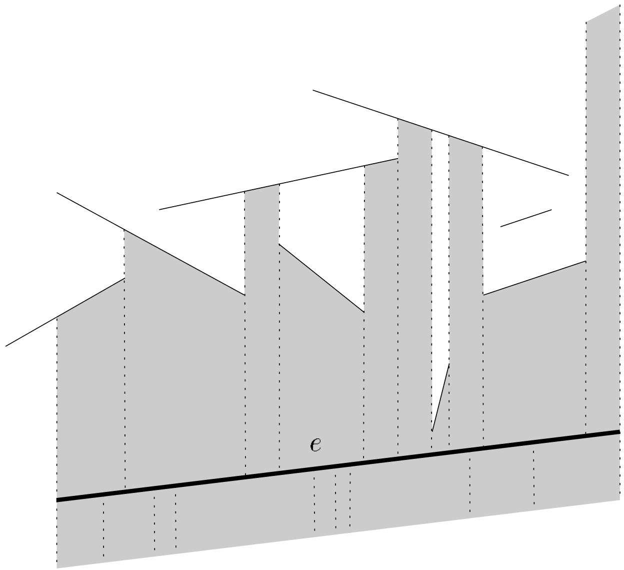


Figure 7.3: The grey face is the (upper half of the) wall extended in vertical direction from the edge $e$. The line segments above $e$ are the intersections of triangles with the supporting plane of the vertical surface $H(e)$.

Walls end as line segments on both sides of the triangles of $T$. The ending walls on the upward-facing side of the triangle are (non-vertical) portions of the lower boundaries of walls $W(e)$. Similarly, the ending walls on the downward-facing side are portions of upper boundaries of walls $W(e)$. Figure 7.4 shows an example of a triangle side and the arrangement of walls ending on it. By charging the complexity of the arrangement of ending walls to the corresponding walls and by noticing that the complexity of $T$ is $O(n)$, we find that the asymptotic cumulative complexity of all triangle arrangements equals the cumulative complexity of all walls: $O(n^2)$. A single scan of all walls suffices to find for all triangle sides the segments that define the arrangement on that specific side. Next, a single arrangement can be computed by a single sweep of the segments in time $O(m \log m)$, where $m$ is the complexity of

De Berg, Guibas, and Halperin [16] give a rather simple algorithm for computing a full vertical decomposition of an arrangement of triangles in 3-space. The general position of the obstacles in $\mathcal{E}$ and the rotated cube $\Lambda \cdot C_{O,\rho_{\mathcal{B}}}$ establish that the triangles in $T_{\mathcal{A}(H)}$ are in general position in the sense that no triangle is vertical, no edge is parallel to a coordinate axis, and no two edges lie in a vertical plane unless they coincide. The fact that we deal with sets of touching triangles requires some additional bookkeeping to prevent multiple extensions of equivalent walls. The bookkeeping is simple and does not affect the efficiency of the algorithm. We neglect the bookkeeping in the description of the algorithm. In the restricted case of non-intersecting triangles, the vertical decomposition algorithm leads to a decomposition of the arrangement of the triangles into $O(n^2)$ constant-complexity regions [69]. The computation takes $O(n^2 \log n)$ time [16]. Below we first briefly discuss the algorithm and the structure of the full vertical decomposition. After that, we show that application of the algorithm to the set of triangles $T_{\mathcal{A}(H)}$ leads to a cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$, with $|V_{\mathrm{W}}| = O(n^2)$ but, unfortunately, a larger set $E_{\mathrm{W}}$. To remedy this, we will replace each triangle by a flat tetrahedron and then show that the resulting set $F(T_{\mathcal{A}(H)})$ of triangular tetrahedron faces solves the problem as its full vertical decomposition leads to $|V_{\mathrm{W}}| = |E_{\mathrm{W}}| = O(n^2)$ and $T(n) = O(n^2 \log n)$.

The computation of the full vertical decomposition proceeds in two steps. The first step results in the vertical decomposition of the arrangement of non-intersecting triangles. The second step uses the specific shape of the resulting regions to subdivide them further to obtain constant-complexity regions, which together constitute the full vertical decomposition. We discuss each step in more detail.

The vertical decomposition step partitions the arrangement of non-intersecting triangles from a given set $T$ into maximal connected collections of points with equal vertical visibility with respect to the triangles of $T$, both in upward and downward $z$-direction. More precisely, a region in the vertical decomposition is a maximal connected set $\{x \in \mathbb{R}^3 | up(x) = t_1 \wedge down(x) = t_2\}$ where $t_1, t_2 \in T$ and $up(x)/down(x)$ denotes the first triangle in $T$ that is hit by the vertical ray emanating from $x$ in upward/downward $z$-direction. The decomposition is achieved by the extension of vertical walls from all triangle edges, which end upon hitting other triangles. The representation of the vertical decomposition computed by the vertical decomposition algorithm consists of: (i) for each triangle edge $e$, the wall $W(e)$ extended from it, and (ii) for each triangle $t \in T$, the two arrangements of ending walls on either side of $t$. The algorithm stores the walls and the triangle arrangements in a quad-edge structure [40] to facilitate future navigating through the decomposition and explicit reporting of the regions and the region adjacencies.

The wall $W(e)$ extended from the edge $e$ is obtained by intersecting the vertical surface $H(e)$ through $e$, that is, the union of all vertical lines through $e$, with all non-intersecting triangles, resulting in $O(n)$ disjoint intersection segments in $H(e)$. The upper boundary of $W(e)$ is defined by the lower envelope of all intersection segments in $H(e)$ lying above $e$. Similarly, the lower boundary of $W(e)$ is defined by the upper envelope of all intersection segments in $H(e)$ lying below $e$. The envelopes

$(b\sqrt{3} \cdot \rho)$-wrapping of $E$. The additional lower bound of $\rho$ on the minimal enclosing sphere radii of all obstacles in $\mathcal{E}$ makes Theorem 2.12 applicable to the arrangement $\mathcal{A}(H)$ of all Minkowski difference boundaries $\partial H(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$). Thus, we obtain Lemma 7.6.

**Lemma 7.6** *Let $\mathcal{A}(H)$ be the three-dimensional arrangement of all boundaries $\partial H(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$). Then*

- **(a)** *$\mathcal{A}(H)$ has complexity $O(n)$,*
- **(b)** *$|\{E \in \mathcal{E} | H(E, \rho_{\mathcal{B}}) \cap A \neq \emptyset\}| = O(1)$ for all 3-faces $A \in \mathcal{A}(H)$.*

The set of obstacles $E$ whose Minkowski differences $H(E, \rho_{\mathcal{B}})$ intersect a given region $R$ is a superset of $Cov(R)$, the set of obstacles $E$ whose grown obstacles $G(E, \rho_{\mathcal{B}})$ intersect $R$, due to the inclusion $G(E, \rho_{\mathcal{B}}) \subseteq H(E, \rho_{\mathcal{B}})$. With the observation we deduce the following interesting corollary from Lemma 7.6(b).

**Corollary 7.7** *$|Cov(A)| = |\{E \in \mathcal{E} | G(E, \rho_{\mathcal{B}}) \cap A \neq \emptyset\}| = O(1)$ for all 3-faces $A \in \mathcal{A}(H)$.*

Hence, the $O(n)$ complexity polyhedral arrangement $\mathcal{A}(H)$ subdivides $\mathrm{W} = \mathbb{R}^3$ into regions with constant-size coverage.

The range searching results in Chapter 3 facilitate a computation of the linear-complexity arrangement $\mathcal{A}(H)$ in time $O(n \log^2 n \log\log n)$. A naive and simpler, but in the light of steps to come sufficiently efficient, computation of $\mathcal{A}(H)$ takes $O(n^2)$ time and simply intersects all pairs of constant-complexity faces of Minkowski difference boundaries $\partial H(E, \rho_{\mathcal{B}})$ and stores the (potential) intersection segment with both faces. After that, each face and the segments defined by its intersection with other faces undergo a constrained triangulation. The triangulation is constrained in the sense that it incorporates all intersection edges as edges of triangles in the triangulation. The triangulation introduces no new vertices. The constrained triangulation can be done by a single sweep (comparable to the sweep in Section 7.1) of each face $f$ in time $O(m_f \log m_f)$, where $m_f$ is the complexity of the respective face and the corresponding intersection segments. As the cumulative complexity $\sum_f m_f$ equals (asymptotically) the complexity $O(n)$ of the arrangement $\mathcal{A}(H)$, the triangulation of all faces of the arrangement takes $O(n \log n)$ time. The result is a collection $T_{\mathcal{A}(H)}$ of non-intersecting triangles. Although the triangles are non-intersecting they do touch each other, that is, they share edges and vertices. For future purposes, we take care to label each triangle $t \in T_{\mathcal{A}(H)}$ with the appropriate obstacle $E$ to indicate that $t$ belongs to the Minkowski difference $H(E, \rho_{\mathcal{B}})$. The decomposition of the workspace by the arrangement $\mathcal{A}(H)$ is not a cc-partition, because the 3-cells of the arrangement may have more than constant complexity. To refine the 3-cells into constant-complexity regions, we apply a full vertical decomposition algorithm to the triangulated arrangement.

FatMot for a bounded-size robot among fat polyhedral obstacles with running time $O(n^2 \log n)$, regardless of the number $f$ of degrees of freedom of the robot. The following description fixes the setting of the results.

> *A constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom ($f \geq 3$) with reach $\rho_{\mathcal{B}}$ moves freely in the workspace $\mathrm{W} = \mathbb{R}^3$ amidst a collection $\mathcal{E}$ of $k$-fat constant-complexity polyhedral obstacles $E \subseteq \mathrm{W}$ with minimal enclosing sphere radii at least $\rho$, for some constant $k \geq 1$. The system is constrained by the inequality $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some fixed constant $b \geq 0$.*

The problem of finding cc-partitions for three-dimensional Euclidean workspaces is much harder than its two-dimensional equivalent, which is illustrated by the relatively small number of results on partitioning 3D arrangements into constant-complexity subcells like tetrahedra or prisms. Moreover, the existing results (see, for example, papers by Aronov and Sharir [6], Chazelle [21], and De Berg, Guibas, and Halperin [16]) do not apply to arbitrary arrangements but instead only hold for arrangements of planar faces, which makes their application to the arrangement of (arbitrarily-shaped) grown obstacles impossible. The two-step approach of first decomposing the workspace into constant-size coverage regions and then refining the regions to constant-complexity regions is likely to lead to $O(n^2)$ regions, as the application of any of the above methods gives $O(n^2)$ subcells when applied to an $O(n)$ complexity arrangement of planar faces. Although a smaller decomposition might be achievable by either this approach or a completely different strategy, we are currently unaware of such a decomposition and therefore choose to settle for a cc-partition of size $O(n^2)$. The partition is obtained by following the two-step approach.

Instead of using the grown obstacle boundaries to achieve the decomposition into constant-size coverage regions, we now use the boundary of a polyhedral outer approximation of these grown obstacles. The polyhedral approximations still achieve a decomposition of W into regions of constant-size coverage, but additionally allow for subsequent application of a vertical decomposition algorithm (to the triangulated polyhedral arrangement) to obtain $O(n^2)$ constant-complexity regions. A tight outer approximation of the grown obstacle $G(E, \rho_{\mathcal{B}})$ is the Minkowski difference $H(E, \rho_{\mathcal{B}})$ of $E$ and a cube with side length $2\rho_{\mathcal{B}}$.

**Definition 7.5** *Let $\Lambda \in SO(3)$ be some arbitrary rotation matrix, establishing that none of the faces of the cube $\Lambda \cdot C_{O,\rho_{\mathcal{B}}}$ is vertical. Then*

$$H(E, \rho_{\mathcal{B}}) = E \ominus (\Lambda \cdot C_{O,\rho_{\mathcal{B}}}).$$

The computation of a Minkowski difference $H(E, \rho_{\mathcal{B}})$ from the constant-complexity obstacle $E$ takes constant time. The Minkowski difference $H(E, \rho_{\mathcal{B}})$ encloses $E$ and, by its definition, no point in $H(E, \rho_{\mathcal{B}})$ has a distance larger than $\sqrt{3} \cdot \rho_{\mathcal{B}}$ to $E$. Hence, $H(E, \rho_{\mathcal{B}})$ is a $(\sqrt{3} \cdot \rho_{\mathcal{B}})$-wrapping of $E$, and because $\rho_{\mathcal{B}} \leq b \cdot \rho$, also a

above and $L$ bounding the old and new lower regions from below. The latter two regions are found by two searches of the binary tree storing the sweep-line status. In summary, the processing of an event requires a constant number of searches, deletions, and insertions in a balanced binary tree. The data structure supports each of these operations in time $O(\log n)$. The remaining computations in a single step a constant-complexity data set, and therefore require constant time.

The $O(1)$ newly obtained pairs of consecutive arcs in the sweep-line status, involving the new separating arcs and $u$ and $l$, may necessitate an update of the event point schedule with the potential intersections of the new consecutive arc pairs. The insertion of an element in the priority queue storing the schedule takes $O(\log n)$ time. As a result, the entire update of the event point schedule takes $O(\log n)$ time.

Throughout the plane sweep a total of $O(n)$ events are encountered, each requiring $O(\log n)$ processing time. The entire sweep, and, hence, the computation of the sets $V_{\mathrm{W}}$ and $E_{\mathrm{W}}$ and the function $Cov$, therefore takes $O(n \log n)$ time.

**Lemma 7.3** *The computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$ takes $O(n \log n)$ time.*

Lemma 7.3 shows that the time $T(n)$ in Theorem 6.8 to compute the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the function $Cov$ is $O(n \log n)$. In addition, Lemma 7.2 bounds the sizes of the sets $V_{\mathrm{W}}$ and $E_{\mathrm{W}}$ by $O(n)$. Hence, the algorithm FatMot computes a decomposition of FP of size $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}|) = O(n)$ in time $T(n) = O(n \log n)$. (Note that the outlined vertical decomposition must be substituted for the first step of the algorithm in order to actually achieve this performance.)

**Theorem 7.4** *Let $k \geq 1$ and $b \geq 0$ be constants and let $\mathcal{E}$ be a collection of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W} = \mathbb{R}^2$ with minimal enclosing circle radii at least $\rho$. Algorithm FatMot solves the motion planning problem for any constant-complexity robot $\mathcal{B}$ with $f \geq 2$ degrees of freedom and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ amidst $\mathcal{E}$ in time $O(n \log n)$. The connectivity graph $\mathrm{CG} = (V_C, E_C)$ of the resulting decomposition of* FP *into simple subcells has optimal size $O(n)$.*

## 7.2 Polyhedral obstacles in 3-space

In this section, we move on to three-dimensional workspaces where we study a setting of a free-flying robot amidst polyhedral obstacles. The number of algorithms for motion planning problems in a three-dimensional workspace with polyhedral obstacles is limited. The two methods that apply to robots with an arbitrary number $f \geq 3$ of degrees of freedom are the general $O(n^{2^{f+6}})$ cell decomposition algorithm by Schwartz and Sharir [85] and the $O(n^f \log n)$ roadmap method by Canny [20]. More specific results include $O(n^{11})$ [87] and $O(n^6 \log n)$ [50] algorithms for a (5-DOF) ladder among polyhedral obstacles, and an $O(n^{15})$ algorithm [87] for a polyhedral robot in the same environment. This section presents an instance of the algorithm

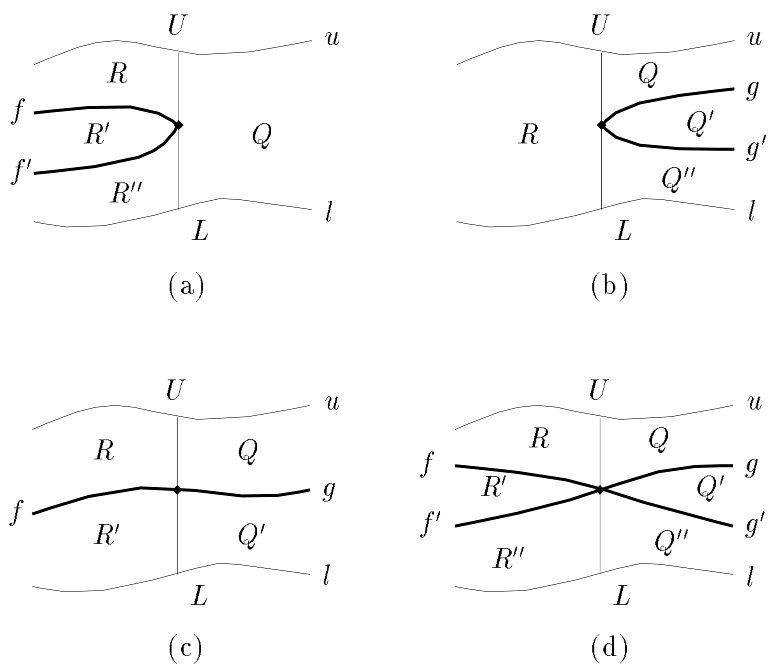


Figure 7.2: The different types of event points.

boundary (and, hence, carrying the same label). The dotted lines are the vertical decomposition walls extended from the event points.

The upcoming event can be extracted in constant time from the event point schedule. The event points, and more particularly the walls extended from it, mark the end of at most three consecutive regions in the sweep line status (named $R$, $R'$, $R''$ in Figure 7.2). The ending regions and their descriptions, which are completed by the addition of the right boundary (a wall or event point), are deleted from the balanced binary tree storing the sweep-line status and transferred to $V_{\mathrm{W}}$. The corresponding coverages are assigned to the appropriate entries of $Cov$. The separating arcs ($f$, $f'$) are deleted along with the ending regions. The deleted regions are replaced by at most three new regions (named $Q$, $Q'$, $Q''$ in Figure 7.2), and their separating arcs. The regions are accompanied by their partial representations, which involve the walls, the new separating arcs ($g$, $g'$), and the two arcs $u$ and $l$ bounding the upper new region from above and the lower new region from below. The identification of the latter two arcs requires two searches of the sweep-line status. The coverages of the new regions are easily computable from the coverages of the old regions, using the simple observations that the coverages of regions on opposite sides of a wall are equal, and the set difference of the coverages on opposite sides of an arc with label $E$ is exactly $\{E\}$. Finally, we must report the adjacencies of the new regions. These adjacencies only involve the $O(1)$ new regions, the $O(1)$ old regions, and the regions $U$ bounding the old and new upper regions from

Moreover, they inherit the constant-size coverage from the original (enclosing) 2-cell of $\mathcal{A}(G)$. The arcs and their endpoints and pairwise intersections and the walls and their endpoints collectively form a planar graph consisting of $O(n)$ edges and vertices, subdividing the plane into $O(n)$ constant complexity regions: the regions of $V_{\mathrm{W}}$. The set $E_{\mathrm{W}}$ of pairs of adjacent vertical decomposition regions has size $O(n)$ as well, as each adjacency can be charged to one of the $O(n)$ edges in the planar graph. Moreover, it is clear that all possible sets $E_{\mathrm{W}}$ and $V_{\mathrm{W}}$ also have size $\Omega(n)$.

**Lemma 7.2** *$V_{\mathrm{W}}$ is a cc-partition of size $\Theta(n)$ of* W *with the obstacles $\mathcal{E}$; $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $\Theta(n)$.*

The plane-sweep algorithm must not only report the regions of $V_{\mathrm{W}}$, but also the coverages $Cov(R)$ of all regions $R \in V_{\mathrm{W}}$, and the region adjacencies of $E_{\mathrm{W}}$. To achieve this, the following invariant regarding the available data is maintained throughout the entire sweep.

- $V_{\mathrm{W}}$ contains all regions strictly left of the sweep-line and their descriptions; $E_{\mathrm{W}}$ contains all adjacencies of regions left of the sweep-line; $Cov(R)$ is assigned for all regions $R$ currently in $V_{\mathrm{W}}$;

- The *sweep-line status* is a top-to-bottom ordered cross-section of the vertical decomposition of $\mathcal{A}(G)$ at $x = X$. As such, it is an alternating sequence of intersected regions and intersected arcs. The elements of the sequence are stored in the leaves of a balanced binary tree. The regions in the data structure are accompanied by their coverages, and partial descriptions, i.e., their (possibly degenerate) left vertical bounding wall and upper and lower bounding arc.

- The *event point schedule* is the sequence of upcoming events, consisting of all statically computable arc endpoints, and potential intersections of pairs of consecutive arcs (separated by a single region) in the sweep-line status. The summarized events are stored in a priority queue, ordered by increasing $x$-coordinate. The upcoming event is always either an arc endpoint of an intersection of two consecutive arcs, so the first event in the queue is indeed the upcoming event.

The creation of an artificial event point at $x = +\infty$ and the proper initialization of the event point schedule and the sweep-line status causes the consistent maintainance of the invariant to eventually lead to the situation where the event point schedule is empty, the set $V_{\mathrm{W}}$ consists of all regions in the vertical decomposition of $\mathcal{A}(G)$, the set $E_{\mathrm{W}}$ contains all pairs of adjacent region from $V_{\mathrm{W}}$, and the function $Cov$ is assigned appropriately for all arguments $R \in V_{\mathrm{W}}$.

Figure 7.2 shows the different kinds of events that are encountered: (a)-(c) show all possible endpoint events, (d) shows the intersection event. Notice that the endpoint events are vertices joining two arcs originating from a single grown obstacle

present application. The (b)-part follows directly from Lemma 6.7 if one realizes that all points $p$ in a single 2-cell $A \in \mathcal{A}(G)$ lie in exactly the same collection of grown obstacles.

**Lemma 7.1** *Let $\mathcal{A}(G)$ be the planar arrangement of all boundaries $\partial G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$). Then*

**(a)** *$\mathcal{A}(G)$ has complexity $O(n)$,*

**(b)** *$|Cov(A)| = O(1)$ for all 2-faces $A \in \mathcal{A}(G)$.*

By Lemma 7.1, the resulting $O(n)$ arcs define only $O(n)$ (yet unknown) arc intersections, and additionally subdivide W into regions with constant-size coverage.

In a second step we compute the vertical decomposition of the arrangement $\mathcal{A}(G)$ of grown obstacle boundaries, by sweeping the plane [14] with the arcs with a vertical line, meanwhile extending walls in upward and downward vertical direction from all $O(n)$ arc endpoints (known in advance) and all $O(n)$ arc intersections (to be determined during the sweep). Figure 7.1 shows an example of the vertical decomposition of an arrangement of $x$-monotone arcs. For simplicity, we assume

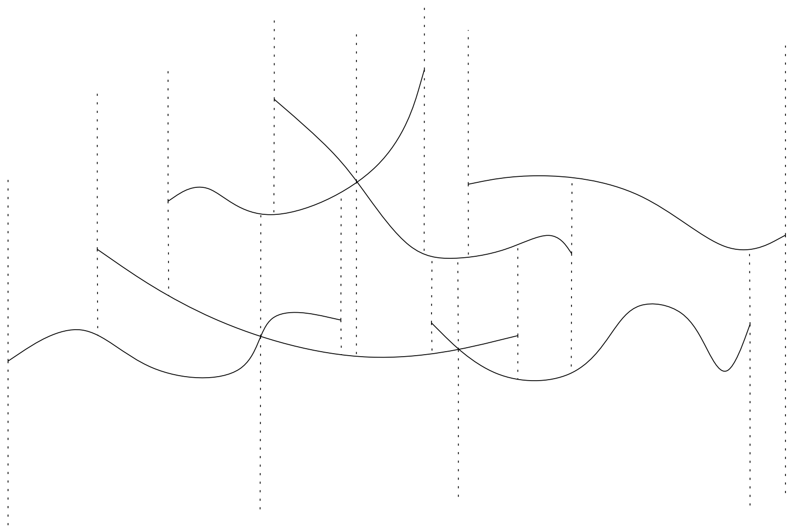

Figure 7.1: The vertical decomposition of an arrangement of $x$-monotone arcs: walls are extended in vertical direction from all arc endpoints and arc intersections.

that no two events points, that is, arc intersections or arc endpoints, lie on a vertical line $x = X$. The extended walls end on the first arc that is hit in the direction of the extension. The walls subdivide the 2-cells of the arrangement into regions bounded by two (possibly degenerate) vertical walls and two arc sections. Hence, the regions in the vertical decomposition of the arrangement $\mathcal{A}(G)$ have constant complexity.

lifting the regions into $C$ has constant complexity. The algorithm in Section 6.1 transforms the decomposition into an actual cell decomposition of the free space. The base decomposition in the plane very strongly resembles the decomposition outlined below in Section 7.1.

## 7.1 Arbitrary obstacles in 2-space

In order to get a feeling for the different aspects of computing an appropriate base partition, we first focus on planar motion planning before we move on to three-dimensional workspaces and obstacles. The aim in this section is to find a small cell decomposition of the free space for the class of motion planning problems with the following characteristics.

> *A constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom ($f \geq 2$) with reach $\rho_{\mathcal{B}}$ moves freely in the workspace $\mathrm{W} = \mathbb{R}^2$ amidst a collection $\mathcal{E}$ of $k$-fat constant-complexity obstacles $E \subseteq \mathrm{W}$ with minimal enclosing circle radii at least $\rho$, for some constant $k \geq 1$. The system is constrained by the inequality $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some fixed constant $b \geq 0$.*

A substantial part of the specification of a placement of the free-flying robot $\mathcal{B}$ is the position of its reference point $O \in \mathcal{B}$ in the workspace $\mathcal{B}$. As a result, the configuration space $C$ of the problem is the Cartesian product of the 2-dimensional Euclidean workspace W and some $(f-2)$-dimensional space $D$, hence $C = \mathrm{W} \times D = \mathbb{R}^2 \times D$. For a rigid robot ($f = 3$), the space $D$ equals the one-dimensional rotational interval $[0, 2\pi)$; for free-flying articulated robots ($f \geq 4$), the space $D$ models the relative placements of the robot's links.

The partition that is proposed below, a vertical decomposition of the arrangement of grown obstacle boundaries, is a simple example of a conceptually two-layer approach that we will use more often in this chapter. This two-layer approach finds a partition of W by first dividing W into regions with constant-size coverage, and subsequently refines the regions to obtain constant-complexity regions. Notice that the instance of algorithm FatMot presented in this subsection provides a generalization of the algorithm presented in Section 5.2, which is a modified version of Schwartz and Sharir's algorithm [84] and as such dedicated to a polygonal robot amidst polygonal obstacles.

Our first step towards a cc-partition comprises the computation of the grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}})$ for all obstacles $E \in \mathcal{E}$. Each boundary is obtained in $O(1)$ time leading to $O(n)$ time for computing all boundaries. As a preparation for the next step, each grown obstacle boundary $\partial G(E, \rho_{\mathcal{B}})$ is cut up into $O(1)$ arcs which are maximal connected, $x$-monotone boundary parts having no vertices in their interiors. Note that the arc endpoints are generally incident to two arcs. For future purposes we label each arc from $\partial G(E, \rho_{\mathcal{B}})$ with $E$. Lemma 7.1 repeats earlier results on the arrangement $\mathcal{A}(G)$ in a formulation that better suits their

# Chapter 7

# Efficiently computable base partitions

This objective in this chapter is to find instances of the general paradigm presented in Chapter 6 for a handful of different settings of the motion planning problem. Besides a universal and nearly-optimal solution for planning in two-dimensional workspaces, we shall consider four different problems in three-dimensional workspaces.

Appropriate workspace decompositions for application of the algorithm FatMot are shown to exist for problems involving a free-flying robot moving among polyhedral and arbitrary obstacles. The decompositions have sizes $O(n^2)$ and $O(n^3)$ and are computable in $O(n^2 \log n)$ and $\Theta(n^3)$ time respectively. FatMot transforms the workspace partitions into cell decompositions of asymptotically equivalent size. Sections 7.2 and 7.3 discuss the details of the respective partitions and their construction.

Nearly-optimal results are obtained for two classes of motion planning problems in 3-space with regularly encountered additional properties. The first class consists of problems involving a free-flying robot and arbitrary obstacles from a bounded range of sizes. More precisely, the ratio of the minimal enclosing hypersphere radii of any pair of obstacles is bounded by a constant. The workspace with the obstacles of this type of problem allow for a simple cc-partition of size $O(n)$. Section 7.4 reports the details of the partition and its computation. The second class of problems, discussed in Section 7.5 concerns a further constrained robot. The robot's reference point is confined to a plane in the workspace W. The class contains the realistic problems where the robot moves on a workfloor. Such problems are often encountered in industrial environments. One example is the vacuum cleaner robot studied in [100]. Contrary to all other problem types dealt with so far, the workspace W is not a projective subspace of the robot's configuration space $C$. (Clearly, the robot is not free-flying.) Thus, the paradigm of Section 6.2 does not apply directly to this class of problems. The plane to which the robot's reference point is confined, however, is a subspace of $C$, and it turns out that this plane is decomposable into regions such that the free part of the configuration space cylinders obtained after

method to be rather efficient since the paradigm reduces the problem of finding a decomposition of certain $f$-cells in an arrangement in $f$-dimensional configuration space to the problem of finding some constrained partition of the $d$-dimensional workspace ($d \leq f$). Besides the dimensional reduction, there is also the feeling that the hypersurfaces in configuration space have more complex shapes than the obstacles in the workspace that are responsible for the partition constraints. A major part of the next chapter is devoted to providing convincing and less intuitive arguments for the validity of our approach, simply by deducing small and efficiently computable cc-partitions for workspaces with various kinds of obstacles. Another part of that chapter exploits the more general ideas of this chapter - on cylindrifiable configuration spaces - to obtain an efficient algorithm for planning the motion of a non-free flying robot amidst fat obstacles.

In Chapter 4, we have seen that, within our framework, self-collisions have no major implications for the complexity of the free space. The $O(1)$ self-collision constraint hypersurfaces do not increase the asymptotic complexity of the arrangement of constraint hypersurfaces, and, hence, of the free space. Let us now reveal the consequences of self-collisions for the motion planning paradigm given as the algorithm FatMot. We have seen that self-collisions are independent of the position of the robot's reference point, as they can occur anywhere in the workspace. The corresponding constraint hypersurfaces in $C = \mathrm{W} \times D$ are therefore of the form $\mathrm{W} \times s$, where $s \subseteq D$. As a result, any self-collision hypersurface $f_s$ intersects *all* configuration space cylinders $R \times D$. The constant number of such hypersurfaces and their individual constant complexity, however, guarantee that the combinatorial complexity of the arrangement $\mathcal{A}$ and the set $\mathrm{FP} \cap (R \times D)$ in steps 1.3 and 2 of the algorithm FatMot *and* the running time of FatMot are not affected. The correctness of the paradigm after the incorporation of self-collisions is established by replacing the initialization of $F := \emptyset$ in step 1.1 by the initialization $F := F_s$, where $F_s$ is the set of all self-collision constraint hypersurfaces.

ALGORITHM FATMOT

Find a cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and compute $Cov$;
$V_C := \emptyset$;
$E_C := \emptyset$;
**for all** $R \in V_{\mathrm{W}}$ **do**
  1.1. $F := \emptyset$;
  1.2. **for all** $\phi \in_f \mathcal{B} \wedge \Phi \in_f Cov(R)$ **do**
    1.2.1. compute $f_{\phi,\Phi}$;
    1.2.2. $F := F \cup \{f_{\phi,\Phi}\}$;
  1.3. compute the arrangement $\mathcal{A}$ of all $f \in F$;
  2. use $\mathcal{A}$ to compute a decomposition of
    $\mathrm{FP} \cap (R \times D)$ into connected subcells;
  3. $Desc(R) := \emptyset$;
  4. **for all** constant-complexity subcells $A$ of $\mathrm{FP} \cap (R \times D)$ **do**
    4.1. $V_C := V_C \cup \{A\}$;
    4.2. $Desc(R) := Desc(R) \cup \{A\}$;
**for all** $(R_1, R_2) \in E_{\mathrm{W}}$ **do**
  **for all** $A_1 \in Desc(R_1) \wedge A_2 \in Desc(R_2)$ **do**
    **if** $\partial A_1 \cap \partial A_2 \neq \emptyset$ **then** $E_C := E_C \cup \{(A_1, A_2)\}$.

The refinement of step 1 of the first **for**-loop verifies the running time of $O(|V_{\mathrm{W}}|)$ for the first **for**-loop of the transformation. The running time of the entire algorithm FatMot becomes $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}| + T(n))$, because of the running time of $T(n)$ for finding the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and computing $Cov$, and the $O(|E_{\mathrm{W}}|)$ time bound on the execution of the second **for**-loop (see Section 6.1). The $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}| + T(n))$ time bound emphasizes once again that the efficiency of FatMot is fully determined by the size of the graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the time to compute the graph and the function $Cov : V_{\mathrm{W}} \rightarrow \mathcal{P}(\mathcal{E})$. Since the time $T(n)$ to compute the graph and the function dominates the time $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}|)$ to simply report both, we may conclude that the $T(n)$-factor dominates the running time of the algorithm FatMot, which may therefore be said to equal $T(n)$.

**Theorem 6.8** *Let $b$ and $k$ be positive constants. In addition, let $\mathcal{E}$ be a set of $k$-fat constant-complexity obstacles $E$ in the robot's workspace* W *with minimal enclosing hypersphere radii at least $\rho$, and let $\mathcal{B}$ be a constant-complexity robot with reach $\rho_{\mathcal{B}} \leq b \cdot \rho$. Furthermore, let $C = \mathrm{W} \times D$ be the configuration space of $\mathcal{B}$. Then, algorithm FatMot computes a decomposition of the free space $\mathrm{FP} \subseteq C$ into simple subcells with a connectivity graph $\mathrm{CG} = (V_C, E_C)$ of size $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}|)$ in time $O(T(n))$, where $T(n)$ is the time to compute a cc-partition of the workspace and the corresponding function $Cov : V_{\mathrm{W}} \rightarrow \mathcal{P}(\mathcal{E})$.*

Although the exact performance of the algorithm depends on the ability to find small and efficiently computable cc-partitions, one may, at this stage, expect the

**(b)** *every point $p \in \mathrm{W} = \mathbb{R}^d$ lies in at most $O(1)$ grown obstacles $G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$).*

Lemma 6.7(b) shows that it is possible to partition the workspace W with the $k$-fat obstacles of $\mathcal{E}$ into (constant-complexity) regions with constant-size coverage. Notice that the arrangement $\mathcal{A}(G)$ even partitions $\mathrm{W} = \mathbb{R}^d$ into $O(n)$ regions $R$ with $|Cov(R)| = O(1)$, as each $d$-cell of the arrangement is a subset of the intersection of $O(1)$ grown obstacles (by Lemma 6.7(b)). Unfortunately, the partition does not suit our purposes, because the $d$-cells themselves may have more than constant complexity. Hence, it is not a cc-partition (although it can be further refined into one).

In summary, we have found (Lemma 6.7) that a cc-partition of a workspace with non-intersecting $k$-fat obstacles always exists. The cc-partition in the workspace corresponds, by Lemma 6.5, to a decomposition of the configuration space into constant-complexity cylinders that are intersected by no more than a constant number of constraint hypersurfaces. As a result, the cc-partition is a valid partition of the base space W allowing for application of the transformation algorithm from Section 6.1.

The algorithm FatMot given below combines the search for a small cc-partition with the transformation of that cc-partition into a cell decomposition of the free space based on the transformation steps from the previous section. Besides the cc-partition regions, gathered in a set $V_{\mathrm{W}}$, the first step is to report the adjacencies of the cc-partition regions in a set $E_{\mathrm{W}}$, and the function $Cov : V_{\mathrm{W}} \rightarrow \mathcal{P}(\mathcal{E})$ mapping each region $R \in V_{\mathrm{W}}$ onto the (constant-cardinality) set of obstacles $E \in \mathcal{E}$ with $G(E, \rho_{\mathcal{B}}) \cap R \neq \emptyset$. (Occasionally, the pair $(V_{\mathrm{W}}, E_{\mathrm{W}})$ will be referred to as a cc-partition graph.) We denote the time required to compute the pair $(V_{\mathrm{W}}, E_{\mathrm{W}})$ as well as the coverage function $Cov$ by $T(n)$, where the argument $n$ represents the number of obstacles in $\mathcal{E}$.

The remainder of the algorithm FatMot is a copy of the transformation algorithm from Section 6.1 with the exception of the refinement in step 1. The refinement shows how the precomputed sets $Cov(R)$ aid in computing in constant time the arrangement $\mathcal{A}$ of all constraint hypersurfaces that intersect the cylinder $R \times D$. A closer look at the refinement learns that $\mathcal{A}$ is the arrangement of all constraint hypersurfaces in a set $F = \{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f Cov(R)\}$, which is a superset of the set of hypersurfaces $f_{\phi,\Phi}$ that satisfy $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. Fortunately, the easily computable set $F$ contains only a constant number of hypersurfaces, due to the constant cardinality of $Cov(R)$ and the constant complexity of $\mathcal{B}$ and the individual obstacles $E$. Crucial to the validity of the approach of computing a somewhat larger arrangement is the simple observation that $\mathcal{A} \cap (R \times D)$, i.e., the restriction of the arrangement $\mathcal{A}$ to the cylinder $R \times D$, is equivalent to the restriction to $R \times D$ of the arrangement of hypersurfaces $f_{\phi,\Phi}$ with $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. The techniques by Schwartz and Sharir from [85] may be useful to compute a decomposition of the free part $\mathrm{FP} \cap (R \times D)$ of a cylinder $R \times D$.

**Lemma 6.5** *Let $R \subseteq \mathrm{W} = \mathbb{R}^d$ be such that $|Cov(R)| = O(1)$. Then*

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1).$$

**Proof:** Take a constraint hypersurface $f_{\phi,\Phi}$ with $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. Now let $E$ be such that $\Phi \in_f E$. By Lemma 6.3, $f_{\phi,\Phi} \subseteq G(E, \rho_{\mathcal{B}}) \times D$. Hence, necessarily $(R \times D) \cap (G(E, \rho_{\mathcal{B}}) \times D) \neq \emptyset$ and thus $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. By the definition of $Cov(R)$ and the assumption $|Cov(R)| = O(1)$, it follows that there are only $O(1)$ obstacles $E$ such that $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. Due to the constant complexity of these obstacles and the robot, there is only a constant number of hypersurfaces $f_{\phi,\Phi}$ with $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. □

The lemma states that any region $R$ with $|Cov(R)| = O(1)$ is guaranteed to satisfy the constraint on the regions of the base partition requiring that the corresponding cylinder is intersected by $O(1)$ constraint hypersurfaces. As a consequence, a decomposition of the workspace W into regions $R$ with both $|Cov(R)| = O(1)$ and constant complexity is a valid base partition of the base space $B =$ W. We shall refer to workspace partitions of this kind as cc-partitions (<u>c</u>onstant-size coverage, <u>c</u>onstant-complexity).

**Definition 6.6 [cc-partition]**
*A cc-partition $V$ of a workspace $W$ with obstacles $\mathcal{E}$ is a partition of $W$ into regions $R$ satisfying the following additional constraints:*

- $|Cov(R)| = O(1)$,
- *$R$ has constant complexity.*

The constant-size coverage constraint $|Cov(R)| = O(1)$ replaces the constraint $|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1)$; the new constraint is simpler because it is truly a constraint in the workspace. The result in Lemma 6.5 and the definition of cc-partitions, however, would be completely useless if a partition of W into regions $R$ with $|Cov(R)| = O(1)$ does not exist. Note that the existence of such a partition solely depends on the absence of points $p \in$ W that are contained in $\omega(1)$ grown obstacles. Fortunately, such points do indeed not exist by Lemma 6.7. The result follows immediately from Theorem 2.12, noting that each grown obstacle $G(E, \rho_{\mathcal{B}})$ is a constant-complexity $\rho_{\mathcal{B}}$-wrapping and, by $\rho_{\mathcal{B}} \leq b \cdot \rho$, also a constant-complexity $(b \cdot \rho)$-wrapping of the obstacle $E$ itself.

**Lemma 6.7** *Let $\mathcal{E}$ be a set of $n$ non-intersecting $k$-fat obstacles in $\mathbb{R}^d$ with minimal enclosing hypersphere radii at least $\rho$. Furthermore, let $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some positive constant $b$. Then*

**(a)** *the complexity of the arrangement $\mathcal{A}(G)$ of all grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$) is $O(n)$,*

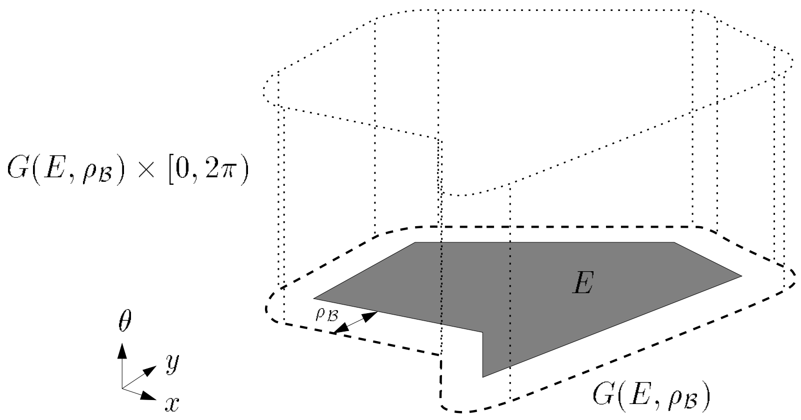


Figure 6.3: A grown obstacle and the corresponding configuration space cylinder for a robot with $\mathrm{W} = \mathbb{R}^2$ and $C = \mathbb{R}^2 \times [0, 2\pi)$.

Assume, for a contradiction, that $p_{\mathrm{W}} \notin G(E, \rho_{\mathcal{B}})$. Then, by the definition of a grown obstacle, the distance from $p_{\mathrm{W}}$ to $E$ exceeds $\rho_{\mathcal{B}}$. But then, it is impossible for $\mathcal{B}$ to reach (and touch) the obstacle $E$, by the definition of the reach of a robot. In other words, no feature $\phi' \in_f \mathcal{B}$ can touch a feature $\Phi' \in_f E$. So, the point $p = (p_{\mathrm{W}}, p_D)$ with $p_{\mathrm{W}} \notin G(E, \rho_{\mathcal{B}})$ cannot lie on $f_{\phi,\Phi}$ contradicting the assumption of the lemma. □

The lemma supplies some kind of a simple outer approximation of the location of a constraint hypersurface in configuration space. If a workspace region $R$ does not intersect a grown obstacle $G(E, \rho_{\mathcal{B}})$ then certainly none of the constraint hypersurfaces $f_{\phi,\Phi}$ with $\Phi \in_f E$ intersects the configuration space cylinder $R \times D$. If on the other hand, $R$ intersects $G(E, \rho_{\mathcal{B}})$, then one or more constraint hypersurfaces $f_{\phi,\Phi}$ with $\Phi \in_f$ may (but not necessarily must) intersect $R \times D$. As a result, the configuration space cylinder $R \times D$ corresponding to a region $R$ that is intersected by $O(1)$ grown obstacles is itself intersected by at most $O(1)$ constraint hypersurfaces. The following definition of the coverage of a workspace region facilitates a compact statement of this interesting result.

**Definition 6.4** **[coverage** $Cov(R)$**]**
*Let* $R \subseteq \mathrm{W} = \mathbb{R}^d$.

$$Cov(R) = \{\, E \in \mathcal{E} \mid R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset \,\}.$$

Hence, $Cov(R)$ is the set of obstacles $E$ whose corresponding grown obstacles $G(E, \rho_{\mathcal{B}})$ intersect $R$. The definition allows for a compact formulation of the preceding observations regarding the relation between the grown obstacles in the workspace and the constraint hypersurfaces in the configuration space.

**Proof:** The subspace $p \times D$ of the configuration space is intersected by constraint hypersurfaces $f_{\phi,\Phi}$. A point in $f_{\phi,\Phi} \cap (p \times D)$ corresponds to a placement of the robot $\mathcal{B}$ in which its reference point is positioned at $p$ and its feature $\phi$ touches an obstacle feature $\Phi$. This feature $\Phi$ must necessarily belong to one of the $O(1)$ constant-complexity obstacles that can be touched by $\mathcal{B}$ while its reference point is fixed at $p$. Combined with the constant complexity of $\mathcal{B}$ itself, this implies that there exist only a constant number of pairs $(\phi, \Phi)$ for which $f_{\phi,\Phi}$ intersects $p \times D$.
$\square$

In the sequel we define a partition of the workspace that is subject to constraints that are formulated exclusively in the workspace. The partition subsequently turns out to be a valid base partition for a cylindrical decomposition of the configuration space.

We define the notion of grown obstacles to formalize the observation that the robot $\mathcal{B}$ is unable to touch an obstacle $E$ if the distance from the location of $\mathcal{B}$'s reference point to the obstacle $E$ exceeds $\rho_{\mathcal{B}}$.

**Definition 6.2 [grown obstacle** $G(E,\rho)$**]**
*Let $E$ be an obstacle in $\mathbb{R}^d$ and let $\rho \in \mathbb{R}^+$. The $\rho$-grown obstacle $E$ is defined as:*

$$G(E,\rho) = \{\, p \in \mathbb{R}^d \mid d(p,E) \leq \rho \,\}.$$

Note that, as an alternative definition, the $\rho$-grown obstacle $G(E,\rho)$ equals the Minkowski difference of $E$ and the hypersphere with radius $\rho$ centered at the origin, so

$$G(E,\rho) = E \ominus S_{O,\rho}.$$

Clearly, the robot's reference point must lie inside $G(E,\rho_{\mathcal{B}})$ in order for the robot $\mathcal{B}$ to be in contact with $E$; if the reference point lies outside $G(E,\rho_{\mathcal{B}})$ there is no danger for $\mathcal{B}$ of colliding with $E$. A formalization of these informal observations leads to a very interesting property on the 'location' of a constraint hypersurface in configuration space.

**Lemma 6.3** *Let $\phi \in_f \mathcal{B}$ and $\Phi \in_f E$. Then:*

$$f_{\phi,\Phi} \subseteq G(E,\rho_{\mathcal{B}}) \times D.$$

**Proof:** Figure 6.3 illustrates the construction by means of a two-dimensional grown obstacle $G(E,\rho_{\mathcal{B}}) \subseteq \mathrm{W} = \mathbb{R}^2$ and a one-dimensional rest-space $D = [0, 2\pi)$. The arguments of the proof are given in the workspace.

Let $p = (p_{\mathrm{W}}, p_D) \in f_{\phi,\Phi}$, such that $p_{\mathrm{W}} \in \mathrm{W}$ and $p_D \in D$. We must prove that $p = (p_{\mathrm{W}}, p_D) \in G(E,\rho_{\mathcal{B}}) \times D$, which may be reduced to proving that $p_{\mathrm{W}} \in G(E,\rho_{\mathcal{B}})$, since $p_D \in D$ is trivially true. This means that it should be proven that the reference point of the robot $\mathcal{B}$ must be placed inside $G(E,\rho_{\mathcal{B}})$ when $\mathcal{B}$'s feature $\phi$ touches $E$'s feature $\Phi$.

$O(1)$ time to find $A_1 \ni Z_1$ using $Desc(R_1)$ and $A_2 \ni Z_2$ using $Desc(R_2)$, followed by a search in the graph $(V_C, E_C)$ for a sequence of subcells connecting $A_1$ to $A_2$. The constant complexities of the subcells and of the common boundaries of pairs of adjacent subcells facilitate the transformation of the subcell sequence into an actual free path for $\mathcal{B}$.

## 6.2 A tailored paradigm for free-flying robots

We now focus on a special instance of the class of motion planning problems with cylindrifiable configuration spaces, namely the problem of planning the motion of a not too large constant-complexity robot $\mathcal{B}$ with $f$ degrees of freedom moving amidst $n$ $k$-fat constant-complexity obstacles $E \in \mathcal{E}$, where $f$ and $k$ are constants. The restriction on the size of the robot is expressed by a bound on its reach: $\rho_{\mathcal{B}} \leq b \cdot \rho$, where $b$ is some positive constant and $\rho$ is a lower bound on the minimal enclosing hypersphere radii of the obstacles in $\mathcal{E}$. For the moment, we assume that the robot $\mathcal{B}$ does not self-collide, that is, no part of $\mathcal{B}$ can collide with any other part of $\mathcal{B}$. Let $O \in \mathcal{B}$ be the reference point of the robot. The tailored paradigm presented below suits robots with configuration spaces that can be written as the Cartesian product of the $d$-dimensional Euclidean workspace W and some other (rest-)space $D$ (of dimension $f - d$),

$$C = \mathrm{W} \times D,$$

such that the position of the robot's reference point in the robot's workspace is part of the specification of its placement. A placement $Z$ of the robot can thus be written as $Z = (Z_{\mathrm{W}}, Z_D)$, where $Z_{\mathrm{W}} \in \mathrm{W} = \mathbb{R}^d$ and $Z_D \in D$. Free-flying robots fit very naturally in this framework. Examples for the rest-space $D$ are $D = [0, 2\pi)$ for a free-flying unsymmetric rigid robot in the plane, and $D = [0, 2\pi)^2 \times [0, \pi]$ for a similar robot in three-dimensional space.

If either the obstacles are non-fat or the robot is arbitrarily large, the robot $\mathcal{B}$ with its reference point $O$ fixed at some point $p \in \mathrm{W}$ may be able to touch all obstacles $E \in \mathcal{E}$. The circumstances summarized above, however, make this impossible: the robot with its reference point fixed at $p$ can only touch obstacles within a distance $\rho_{\mathcal{B}}$ from the point $p$; such obstacles clearly intersect the hypersphere $S_{p,\rho_{\mathcal{B}}}$. Theorem 2.9 implies that the number of obstacles with minimal enclosing hypersphere radii at least $\rho$ intersecting any region with diameter $2\rho_{\mathcal{B}} \leq 2b \cdot \rho$ is bounded by a constant. As all obstacles in $\mathcal{E}$ have minimal enclosing hypersphere radius at least $\rho$, the robot $\mathcal{B}$ can touch no more than $O(1)$ obstacles while its reference point remains fixed at $p$. This fact leads to the following lemma, which provides the theoretical feasibility of choosing W as a basis of the cylindrical cell decomposition.

**Lemma 6.1** *For all $p \in \mathrm{W}$:*

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

The first **for**-loop computes the $O(1)$ (constant-complexity) maximal connected components of $\mathrm{FP} \cap (R \times D)$. A possible way to compute this free part of the cylinder $R \times D$ in constant time could be to apply the techniques by Schwartz and Sharir [85] to the constant number of constraint hypersurfaces intersecting the cylinder $R \times D$. Each of the four steps in the loop is easily verified to run in constant time, provided that the constraint hypersurfaces $f_{\phi,\Phi}$ intersecting $R \times D$ can be determined in constant time. In future applications of the transformation algorithm, we shall take care that this precondition is fulfilled. If the requirement is indeed settled, the entire loop runs in time $O(|V_B|)$. Upon termination of the first loop, each set $Desc(R)$ stores all nodes in $V_C$ that correspond to free subcells in $R \times D$. Note that each set $Desc(R)$ has constant cardinality.

Two free subcells $A_1$ and $A_2$ are adjacent if they share a common boundary (which allows for collision-free crossing from one subcell into the other). Such subcells $A_1$ and $A_2$ can only be adjacent if their containing cylinders $R_1 \times D \supseteq A_1$ and $R_2 \times D \supseteq A_2$ are adjacent in $C$ and, hence, $R_1$ and $R_2$ are adjacent in $B$. An adjacency $(R_1, R_2)$ gives rise to only a constant number of adjacencies of nodes $A_1$ and $A_2$ in $Desc(R_1)$ and $Desc(R_2)$ respectively due to the constant cardinality of $Desc(R_1)$ and $Desc(R_2)$. Two free subcells $A_1$ and $A_2$ in adjacent cylinders are adjacent if they share a common boundary. Such a common boundary has constant complexity since both involved free subcells have constant complexity. The nested **for**-loop in the second **for**-loop takes constant time by the above considerations, implying a running time of $O(|E_B|)$ for the latter loop.

If we combine the time-bounds of the three steps in the paradigm, then we find that the running time of the entire paradigm depends solely on the size of the base partition in a lower-dimensional subspace and on the time to compute the partition. A small and efficiently computable partition is therefore crucial to the success of the paradigm. We notice that the base partition is implicitly subject to constraints in configuration space (not more than a constant number of constraint hypersurfaces may intersect the configuration space cylinder corresponding to the base partition region). We have, however, already suggested the possibility of using hypersurface properties to translate the constraints into simpler, lower-dimensional constraints. In the next section we focus on the large class of motion planning planning problems for free-flying robots (amidst $k$-fat obstacles). We will see that these problems allow for a unique choice of base space. Efficient partitions are likely to be achievable in this subspace due to the possiblity to translate the implicit configuration space constraints into (simpler) constraints in the lower-dimensional subspace. In Chapter 7, we shall see that the resulting tailored paradigm really leads to efficient algorithms for planning motions for free-flying robots.

Finally, we mention that the problem of solving a motion planning query 'find a free path from a placement $Z_1 = (Z_{1B}, Z_{1D})$ to another placement $Z_2 = (Z_{2B}, Z_{2D})$' basically reduces to a point location query with $Z_{1B}$ and $Z_{2B}$ in $V_B$ to find $R_1 \ni Z_{1B}$ and $R_2 \ni Z_{2B}$. So, we need a structure for point location in the base space rather than in the full configuration space $C$. After having found $R_1$ and $R_2$, it takes

> 1. compute the arrangement $\mathcal{A}$ of surfaces $f_{\phi,\Phi}$ intersecting $R \times D$;
> 2. use $\mathcal{A}$ to compute $\mathrm{FP} \cap (R \times D)$;
> 3. $Desc(R) := \emptyset$;
> 4. **for all** maximal connected components $A$ of $\mathrm{FP} \cap (R \times D)$ **do**
>    - 4.1. $V_C := V_C \cup \{A\}$;
>    - 4.2. $Desc(R) := Desc(R) \cup \{A\}$;
>
> **for all** $(R_1, R_2) \in E_B$ **do**
>
>     **for all** $A_1 \in Desc(R_1) \wedge A_2 \in Desc(R_2)$ **do**
>
>         **if** $\partial A_1 \cap \partial A_2 \neq \emptyset$ **then** $E_C := E_C \cup \{(A_1, A_2)\}$.

Figure 6.2 gives a pictorial explanation of the transformation.

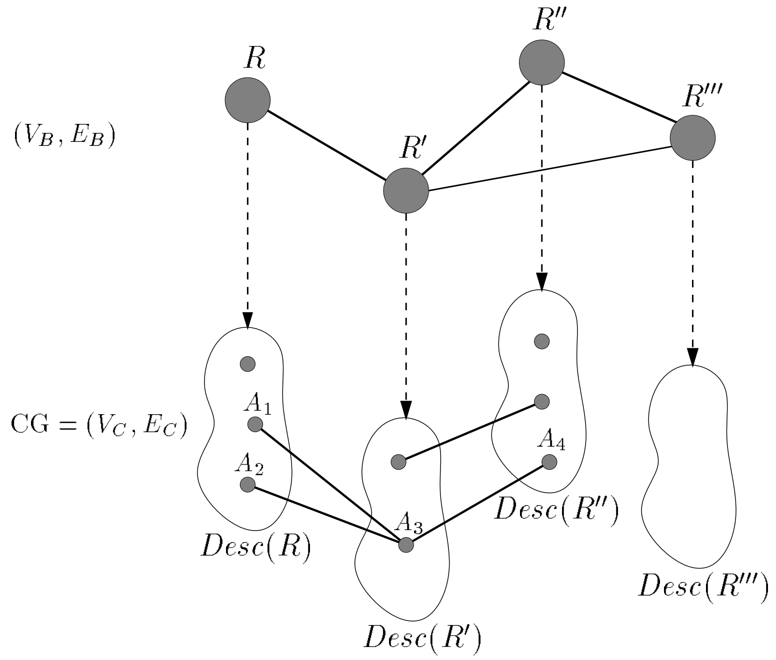


Figure 6.2: The relation between the base partition graph $(V_B, E_B)$ in the subspace $B$ of $C$ at the top, and the connectivity graph $\mathrm{CG} = (V_C, E_C)$ in the configuration space at the bottom. Each node/region $R \in V_B$ defines at most $O(1)$ nodes $A \in V_C$, collected in a set $Desc(R)$. Two nodes $A$ and $A'$ in $V_C$ can only be connected if the corresponding nodes $R$ and $R'$ in $V_B$ are connected, so, for example, $A_1$ may be connected to all nodes in $Desc(R')$, but $A_1$ and $A_4$ can never be connected.

We review the different steps of the transformation in more detail to verify their validity and to determine the efficiency. Recall that the definition of the set $V_B$ and the constant complexity of the regions $R \in V_B$ together imply the constant complexity of all subcells $A \in V_C$.

shapes and relative positions of the obstacles to simplify the constraints on the partition of the base space $B = \mathrm{W}$ for motion planning problems involving free-flying robots. The new and simpler constraints combined with the transformation steps result in a tailored paradigm for motion planning for free-flying robots amidst fat obstacles. In the next chapter, this paradigm is shown to lead to efficient algorithms for planning motions for free-flying robots amidst several types of obstacles in different workspaces. Moreover, the ideas presented in this chapter prove useful for motion planning problems that do not fit neatly in the sketched framework: they lead to an efficient algorithm for planning the motion of a vacuum cleaning robot, which is definitely not free-flying.

## 6.1 Transforming a base partition into a cell decomposition

We consider a motion planning problem for a constant-complexity robot $\mathcal{B}$ amidst constant-complexity obstacles $E \in \mathcal{E}$. Pairs of a feature $\phi \in_f \mathcal{B}$ and a feature $\Phi \in_f \mathcal{E}$ of matching dimension define constraint hypersurfaces $f_{\phi,\Phi}$ in the cylindrifiable configuration space $C = B \times D$. Furthermore, we assume that we are given a graph $(V_B, E_B)$, where $V_B$ is a set of constant-complexity closed regions $R$ that partition $B$ and individually satisfy

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1),$$

and $E_B$ contains the adjacencies of $V_B$'s regions: $E_B = \{(R, R') \in V_B \times V_B | \partial R \cap \partial R' \neq \emptyset\}$.

The algorithm outlined below transforms the graph $(V_B, E_B)$ into a connectivity graph $\mathrm{CG} = (V_C, E_C)$, consisting of a set $V_C$ of constant-complexity subcells that collectively partition the set of free placements FP, and a set $E_C = \{(A, A') \in V_C \times V_C | \partial A \cap \partial A' \neq \emptyset\}$ of subcell adjacencies. The sizes of the sets $V_C$ and $E_C$ are of the same order of magnitude as the sizes of $V_B$ and $E_B$ respectively: $|V_C| = O(|V_B|)$, $|E_C| = O(|E_B|)$. Note that the graph $(V_C, E_C)$ supports simple path-finding between two placements in subcells $A \in V_C$ and $A' \in V_C$: the constant complexity of the individual subcells guarantee easy path-finding within a subcell, and the constant complexity of the shared boundary of two adjacent subcells - following from the constant complexity of the involved subcells - caters for simple boundary crossing rules. The transformation steps are, contrary to the computation of the base partition, independent of the actual motion planning problem under consideration.

Transform Base Partition into Cell Decomposition

$V_C := \emptyset$;
$E_C := \emptyset$;
**for all** $R \in V_B$ **do**

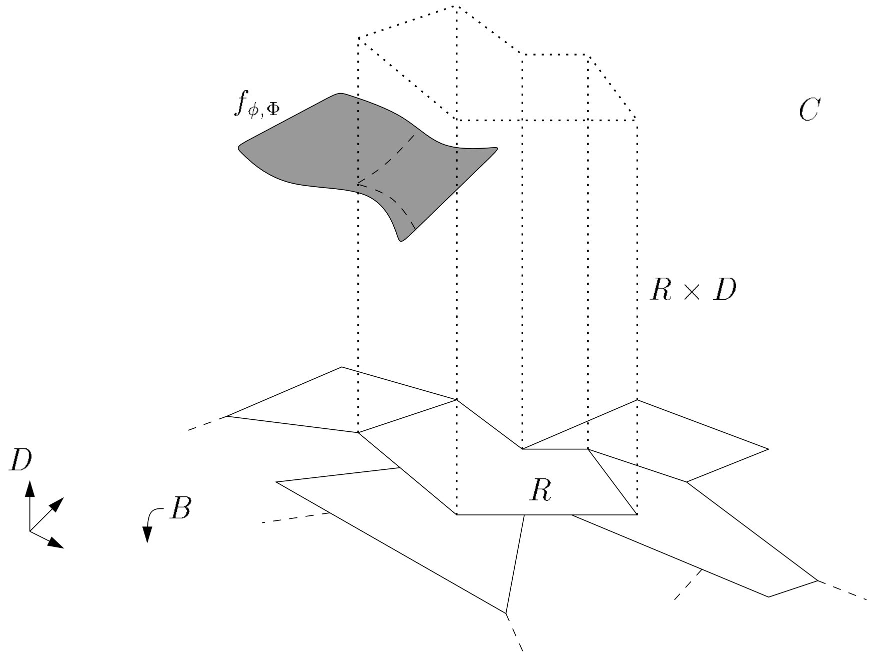


Figure 6.1: A three-dimensional example of a cylindrifiable configuration space $C$ with a base space $B$, and a fragment of the base partition in $B$. The configuration space cylinder $R \times D$, obtained by lifting the base partition region $R$ into $C$, is intersected by at most a constant number of constraint hypersurfaces $f_{\phi,\Phi}$.

The preceding arguments suggest a two-step approach for computing a cell decomposition for a motion planning problem with a cylindrifiable configuration space: first, find a base partition in some appropriate base space $B$ of $C$, and then tranform the partition into a cell decomposition of the free space FP $\subseteq C$, by computing a decomposition of the free part of every cylinder. We shall see that the resulting decomposition consists of cells that allow for simple motion planning within their interiors, and, moreover, that the rules for crossing from one cell into another are simple. The proposed approach follows a projection-like approach to cell decomposition that is encountered in several other algorithms (see e.g. [84, 85]). Basically, these methods (recursively) decompose a lower-dimensional subspace of the configuration space $C$ and lift the decomposition regions into $C$. The free part of the resulting cylinders is subsequently partitioned into a number of simple subcells.

In Section 6.1, it is shown how the latter part of the two-step approach outlined above transforms a base partition into a cell decomposition of comparable size in time proportional to the size of the base partition. Noting this, the problem of finding a (small) cell decomposition of the free space FP $\subseteq C$ reduces to the problem of finding a (small-sized) base partition in an appropriate base space $B \subseteq C$. Section 6.2 exploits specific properties of the constraint hypersurfaces that follow from the

for each point $p \in \mathrm{W}$:

$$|\{f_{\phi,\Phi}|\phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

In words, the $(f-d)$-dimensional subspace $p \times D$ obtained by lifting the workspace point $p$ into configuration space is intersected by only a constant number of constraint hypersurfaces. An immediate consequence of this result is that the hypersurfaces define a constant-complexity arrangement in each cross-section $p \times D$ of the configuration space $C$.

At a more abstract level, motion planning problems for free-flying robots amidst fat obstacles can be regarded as a subclass of the larger class of motion planning problems with configuration spaces $C = B \times D$ that satisfy for each point $p \in B$:

$$|\{f_{\phi,\Phi}|\phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

In general, a configuration space $C$ that satisfies this constraint will be said to be *cylindrifiable*. Furthermore, we call the subspace $B$ of $C$ a *base space*. Hence, motion planning problems involving a free-flying robot among fat obstacles have cylindrifiable configuration spaces in which the workspace constitutes a valid base space. As a result of the cylindrifiability of $C$, it is possible to partition the subspace $B$ into closed regions $R$ (or $C$ into cylinders $R \times D$) such that

$$|\{f_{\phi,\Phi}|\phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1).$$

We refer to such a decomposition of the configuration space $C$ into cylinders as a *constrained cylindrification*. The partition of $B$ that leads to the cylinders will be called the *base partition* corresponding to the cylindrification. Figure 6.1 illustrates the terminology. The figure shows a three-dimensional cylindrifiable configuration space $C$ with a two-dimensional base space $B$. (Hence, the rest-space $D$ is one-dimensional.) In addition, the figure reveals a fragment of the base partition in the subspace $B$ and shows the configuration space cylinder $R \times D$ corresponding to one of the regions $R$ in the partition. The cylinder in this specific example is bounded although in many cases (e.g. $D = \mathbb{R}$, $D = \mathbb{R}^+$) the cylinder will be unbounded. The constraints on the base partition guarantee that the cylinder $R \times D$ is intersected by at most a constant number of surfaces like $f_{\phi,\Phi}$.

Let us now consider the configuration space cylinder $R \times D$ corresponding to a region $R$ in a base partition in $B$. By the definition of a base partition, the cylinder $R \times D$ is intersected by $O(1)$ constraint hypersurfaces. These hypersurfaces subdivide the cylinder $R \times D$ into a constant number of cells, due to their constant complexity. If we furthermore assume that the cylinders themselves have constant descriptional complexity (achievable by establishing that $R$ has constant complexity) then each of the $O(1)$ (free or forbidden) cells in $R \times D$ has constant complexity as well. In conclusion, the constraint hypersurfaces and the cylinder boundaries divide the free space into constant-complexity, and thus simple, subcells.

# Chapter 6

# A paradigm for motion planning amidst fat obstacles

The aim of this chapter is to determine a general approach to planning the motion of a not too large, constant-complexity robot moving amidst $k$-fat constant-complexity obstacles. In Section 5.5, we have seen that the existing planar motion planning algorithms are not easily extendible towards other problems. Moreover, the existing general approaches to motion planning (like those by Schwartz and Sharir [85] and Canny [20]) are computationally expensive, even for problems involving fat objects.

Motion planning problems in Euclidean workspaces of dimension three normally imply at least three-dimensional configuration spaces. A configuration space contains constraint hypersurfaces of the form $f_{\phi,\Phi}$, consisting of placements of the robot $\mathcal{B}$ in which a robot feature $\phi$ is in contact with an obstacle feature $\Phi$. We shall denote the fact that $\xi$ is a feature of some object or object set $X$ by $\xi \in_f X$. The arrangement of all (constant-complexity) constraint hypersurfaces $f_{\phi,\Phi}$ ($\phi \in_f \mathcal{B}, \Phi \in_f \mathcal{E}$) divides the higher-dimensional configuration space into free cells and forbidden cells. Even in the case of fat motion planning, the complexity of a single free cell can be $O(n)$, which illustrates that some additional processing is necessary to facilitate efficient motion planning. Naturally, the structure of a higher-dimensional arrangement like the arrangement of constraint hypersurfaces is complex to understand, let alone to subdivide the free arrangement cells into simple subcells or to catch their structure in some one-dimensional roadmap. At this point, however, fatness comes to our help to provide us with a very useful property of an $f$-dimensional configuration space $C$ of the form $C = \mathrm{W} \times D$, where W is the $d$-dimensional workspace and $D$ is some $(f-d)$-dimensional (rest-)space. (Free-flying rigid robots, for example, fit well in this framework. For a free-flying rigid robot in $\mathrm{W} = \mathbb{R}^3$, $D$ is the space defined by the three rotational degrees of freedom of the robot.) The low object density property of the workspace implied by the fatness of the obstacles can be shown to result in a very interesting property of configuration space, namely that

The Piano Movers' algorithm outlined in Section 5.2 deviates from the general $O(n^{2^{f+6}})$ cell decomposition approach in that it takes a decomposition of the two-dimensional subspace $\mathrm{W} = \mathbb{R}^2$ of $C = \mathbb{R}^2 \times [0, 2\pi)$ as the basis for a cylindrical decomposition, while the general approach would take a decomposition of a one-dimensional subspace of $C$ as a starting point. The alternative of a cylindrical decomposition based in a higher-dimensional subspace $B$ of $C$ offers the opportunity to use properties of the space $B$ to obtain efficient decompositions. The projections of $B$ in the general approach affect such beneficial properties, so that they no longer hold in the projective subspaces of $B$.

A closer look at the details of Section 5.2 learns that the $O(n)$ workspace regions $R$ defined by the critical curves are such that the intersection of the free space with their liftings $R \times [0, 2\pi)$ has constant complexity. This property led to Lemma 5.6, stating that each region $R$ induces only $O(1)$ subcells in the free space decomposition, and, hence, only $O(n)$ subcells in the entire decomposition of the free space. A recursive decomposition of W into similar regions $R$ could easily lead to $\Omega(n^2)$ regions, and thus to $\Omega(n^2)$ subcells. The ability to define a decomposition like the one in the first sentences of this paragraph is rooted in the relative low obstacle density in the workspace. While the robot's reference point is confined to some sufficiently small region $R \in \mathrm{W}$, the robot is able to touch only a constant number of obstacle features. This fact causes the free part of the configuration space cylinder $R \times [0, 2\pi)$ to have constant complexity.

The validity of the low obstacle density property for workspaces of arbitrary dimensions suggests that the ideas in the preceding paragraph are extendible to other motion planning problems. Chapter 6 formalizes and exploits the ideas to obtain a strategy for motion planning that reduces the problem of partitioning the free space to the intuitively simpler problem of computing some constrained decomposition of the (lower-dimensional) workspace, provided that the workspace is a projective subspace of the configuration space. The efficiency of the approach depends on the availability of small constrained workspace decompositions, which is the topic of Chapter 7. By demonstrating the existence of small decompositions, the chapter verifies the validity of the approach.

same free cell. The paper by Avnaim, Boissonnat, and Faverjon [10] includes clues on transforming the sequence of faces into an actual semi-free path for the robot $\mathcal{B}$. Notice that, contrary to most other exact algorithms, part of the work is dedicated to the specific query with the points $Z_0 = (x_0, y_0, \theta_0)$ and $Z_1 = (x_1, y_1, \theta_1)$: another query requires re-doing the second part of the construction.

The most expensive step from a computational point of view of the algorithm sketched above is the computation of the close feature pairs, taking $O(n \log n)$ time. The following theorem summarizes the result obtained in this section.

**Theorem 5.11** *Avnaim, Boissonnat and Faverjon's boundary cell decomposition algorithm* [10] *can be adapted to plan the motion of a polygonal robot $\mathcal{B}$ with diameter $\ell \leq b \cdot \rho$ amidst the fat obstacles of $\mathcal{E}$ in time $O(n \log n)$, for any constant $b \geq 0$.*

## 5.5 Towards a general method

The preceding sections include a variety of algorithms for the solution of the planar motion planning problem amidst fat obstacles, all running in $O(n \log n)$ time. The main goal of the final chapters of this thesis, however, is to find a more general solution to the motion planning problem amidst fat obstacles. Unfortunately, the algorithms presented here are dedicated to planar problems.

Algorithms for efficient motion planning in three-dimensional workspaces are scarce. Approaches in contact space, like the algorithms by Sifrony and Sharir in Section 5.1 and by Avnaim, Boissonnat, and Faverjon in Section 5.4 were never shown to generalize to 3D workspaces. The problem in generalizing such methods lies in the difficulty of establishing (sufficient) connectivity among the nodes corresponding to vertices or faces in a single free cell to guarantee the exact solution of the planning problem.

The general approaches to motion planning for robots with $f$ degrees of freedom are the cell decomposition method by Schwartz and Sharir [85] running in time $O(n^{2^{f+6}})$ and the roadmap method by Canny [20] running in $O(n^f \log n)$ time. The general and recursive nature of these approaches makes it unlikely that they take advantage of any special structure of FP if present, like in our framework. The repeated projection of the free space in the first algorithm is likely to destroy any structure of the free space and may lead to high complexities of the free space projection, regardless of the complexity of the original free space. The number of subcells in the resulting cylindrical decomposition can therefore be high, despite a possible low complexity of FP. The recursive manner of introducing curves to guarantee the connectivity of the roadmap in Canny's method seems insensitive to a special structure or low complexity of the free space. The number of such curves relates to some extent to the number of local extrema of the free space and in certain lower-dimensional subspaces of the free space, and the fatness of the obstacles in the workspace does not seem to reduce the latter quantity.

simultaneous subdivision of the difference via the surface sweep takes $O(m \log m)$ time. Repeating the arguments for all contact surfaces $f$ leads to at most two chargings of every close pair $(\phi, e')$, so it follows that $\sum_f m_f = O(n)$. The cumulative running time of all sweeps, and, hence, of the entire computation of all faces of BFP equals $\sum_f O(m_f \log m_f) = O(n \log n)$. Extraction of the adjacency information for the faces takes $O(n \log n)$ time. The resulting graph is denoted by $G_B$.

Assume, for the second part of Avnaim, Boissonnat, and Faverjon's algorithm, that the goal is to find a path between the free placements $Z_0 = (x_0, y_0, \theta_0)$ and $Z_1 = (x_1, y_1, \theta_1)$. Let $K_0$ and $K_1$ be the free cells containing $Z_0$ and $Z_1$ respectively. The boundedness of $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$ implies that all free cells are bounded as well. The boundaries $\partial K_0$ and $\partial K_1$ may well be non-connected. The boundedness of the cells guarantees that one connected component of a cell boundary, the so-called external boundary, encloses all other connected components. Let $\partial K_0^*$ and $\partial K_1^*$ be the external boundaries of $K_0$ and $K_1$ respectively. Furthermore, let $K_{\theta_0}$ be the intersection of the plane $\theta = \theta_0$ and the free cell $K_0$ and let $K_{\theta_1}$ be the intersection of $\theta = \theta_1$ and $K_1$. The placement $Z_0$ is connected to any placement in $\partial K_0^*$ by a (semi-free) path that is entirely contained in $K_{\theta_0} \cup \partial K_0$. Similarly, $Z_1$ is connected to any placement $Z' \in \partial K_1^*$ by a path in $K_{\theta_1} \cup \partial K_1$. The key observation is that $Z_0$ and $Z_1$ are connected by a semi-free path if and only if both placements belong to the same free cell: $K_0 = K_1$. In that case, $Z_0$ and $Z_1$ are connected by a path in $K_{\theta_0} \cup \partial K_0 \cup K_{\theta_1} = K_{\theta_0} \cup \partial K_1 \cup K_{\theta_1}$. A decomposition of $K_{\theta_0}$ and $K_{\theta_1}$ into simple faces facilitates path-finding in these two subsets of FP.

The cross-section $K_{\theta_0}$ of the free cell $K_0$ is a polygonal region. The complexity of the polygonal region $K_{\theta_0}$ is bounded by the complexity of the intersection of the FP and the plane $\theta = \theta_0$, which is $O(n)$ by the results from Chapter 4. A vertical decomposition subdivides $K_{\theta_0}$ into $O(n)$ faces bounded by at most four edges. The computation of the decomposition and the adjacencies of the faces via a sweep of $K_{\theta_0}$ takes $O(n \log n)$ time. (For details on such a sweep, the reader is referred to Section 7.1.) Let $G_0$ be the adjacency graph on the faces in the vertical decomposition. A similar treatment of $K_{\theta_1}$ results in an $O(n)$ decomposition of $K_{\theta_1}$ and a corresponding graph $G_1$. The faces containing $Z_0$ and $Z_1$ are easily determined during the sweeps. Notice that no face corresponding to a node in $G_0$ is adjacent to a face corresponding to a node in $G_1$, unless $\theta_0 = \theta_1$.

The final task is to merge the graphs $G_B$, $G_0$, and $G_1$ into a single graph $G$ on the faces in all three decompositions. Merging the graph $G_0$ into $G_B$ requires a single simultaneous scan of the nodes in $G_B$ corresponding to faces that are intersected by $\theta = \theta_0$ and the nodes in $G_0$ corresponding to faces on the boundary of $K_{\theta_0}$. A node in $G_B$ and a node in $G_0$ are connected by an edge if the corresponding faces share a curve of non-zero length. A careful implementation of the merge requires $O(n)$ time. Finally, the graph $G_1$ is merged into $G_B \cup G_0$ using the same ideas and, hence, within the same time bound.

A search of the graph $G$ with size $O(n)$ returns a sequence of faces connecting the face containing $Z_0$ and the face containing $Z_1$ if and only if $Z_0$ and $Z_1$ lie in the

problem between two specific placements. The initial and final placements determine the faces that are added in this step. The preceding decomposition of BFP, on the other hand, is independent of the query. The boundary faces and the additional faces constitute the nodes of a graph in which two nodes are connected if their corresponding faces share a common boundary. The graph is subsequently searched for a path connecting the initial and final robot placements. The motion planning algorithm has worst-case running time $O(n^3 \log n)$, but the authors claim that the running time improves to $O(n \log n)$ in workspaces of bounded local complexity [88]. Below, we see that the running time is $O(n \log n)$ in the case of a polygonal robot $\mathcal{B}$ with reach $\rho_{\mathcal{B}} \leq b \cdot \rho$, and, hence, diameter at most $2\rho_{\mathcal{B}}$, moving among the $k$-fat polygonal obstacles of $\mathcal{E}$.

The free space boundary decomposition is based on ideas borrowed from a paper by Avnaim and Boissonnat [9]. The authors decompose BFP into faces bounded by two straight edges parallel to the plane $\theta = 0$ and by two curved arcs. They essentially compute the $O(n)$ contact surfaces consisting either of all placements in which a robot vertex $v$ touches an obstacle edge $e'$ or of all placements in which a robot edge $e$ touches an obstacle vertex $v'$, and subsequently subtract the collection of placements in which a robot edge intersects an obstacle edge from each of these contact surfaces. A sweep of each contact surface computes the difference of the initial surface and the collection of placements corresponding to intersecting robot and obstacle edges[2]. The sweep simultaneously subdivides the resulting set difference into faces bounded by two arcs and two straight edges. A sweep of a single contact surface takes worst-case $O(n^2 \log n)$ time, resulting in a total time of $O(n^3 \log n)$ time for handling all surfaces. The faces form a decomposition of the free space boundary into simple subcells. Determining the connectivity of the faces takes time proportional to the cumulative complexity of the faces, provided that the bounding curves of the faces are labeled with a characterization of the double contact that they represent.

Let us now consider the consequences of fatness on the decomposition sketched above. Assume that $f_{\phi,\Phi}$ is the (constant-complexity) surface consisting of the placements in which the robot feature $\phi$ touches the obstacle feature $\Phi$. The objective is to subtract from $f_{\phi,\Phi}$ all placements $Z$ in which an edge $e$ of $\mathcal{B}$ intersects some obstacle edge $e'$. Such an obstacle edge $e'$ must lie within a distance $2\rho_{\mathcal{B}} \leq 2b \cdot \rho$ from the obstacle feature $\phi$, because $\mathcal{B}$ simultaneously touches $\Phi$ and intersects $e'$. By Lemma 5.1, the overall number of such pairs is $O(n)$, and they can be computed in time $O(n \log n)$ using the technique by Sifrony and Sharir outlined in Subsection 5.2.2. Provided that these $O(n)$ close feature pairs are computed in advance, it is possible to determine the $m$ edges $e'$ within a distance $2\rho_{\mathcal{B}}$ from $\phi$ in time $O(m)$. We charge each edge $e'$ to the close pair $(\phi, e')$. The $m$ edges $e'$ and the $O(1)$ robot edges $e$ define $O(m)$ (constant-complexity) collections of points that are to be subtracted from $f_{\phi,\Phi}$. The computation of the difference of $f_{\phi,\Phi}$ and these $O(m)$ sets and the

[2] A suitable parametrization of the contact surface facilitates an efficient sweep.

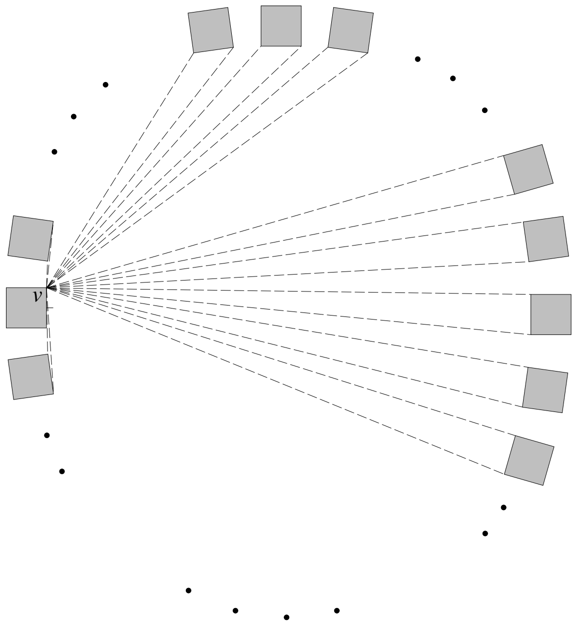


Figure 5.3: The number of critical orientations is not reduced by the fatness of the obstacles.

wholly contained in the closure of $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$. If we do the same for each of the remaining $2n-1$ obstacle corners facing the interior of the circle, we obtain a total number of $\frac{1}{2}(2n-1)(2n-2) = (2n-1)(n-1)$ different line segments, each of them corresponding to an occurrence of condition (C4). By small perturbations of the obstacles we can establish that each of the line segments has a different orientation, which results in at least $(2n-1)(n-1)$ critical orientations induced by occurrences of condition (C4). As a consequence, the number of critical orientations is $O(n^2)$. Thus, the fatness of the obstacles does not lead to a reduction of the complexity of the cell decomposition in this case.

## 5.4 Boundary cell decomposition

Avnaim, Boissonnat, and Faverjon [10] describe a variant of the cell decomposition approach that, rather than decomposing the free space itself, decomposes the free space boundary $\mathrm{BFP} = cl(\mathrm{FP}) \setminus \mathrm{FP}$ into simple subcells or faces. The results in Chapter 4 imply that the complexity of BFP in our realistic setting is $O(n)$. The algorithm requires that the complement $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$ of the obstacles is bounded. This requirement is easily met by enclosing the workspace obstacles in arbitrarily large box. After the decomposition of the free space boundary, additional faces are created to establish sufficient connectivity among the faces to solve the path-finding

The basis of the algorithm by Leven and Sharir is the fact that a simple $O(n)$ cell decomposition exists for the strictly translational version of the problem. This cell decomposition is, off course, a decomposition of the projective subspace $\mathbb{R}^2$ of the configuration space $\mathbb{R}^2 \times [0, 2\pi)$ of the original problem. The decomposition is such that a small change in the orientation of the robot leads to an only slightly different (and often topologically equivalent) cell decomposition. In configuration space we conceptually obtain an infinitely small-grain stack of such continuously varying planar cell decompositions. If one would descend the stack, then, at certain orientations, the planar cell decomposition changes topologically. These so-called critical orientations divide the angular dimension $[0, 2\pi)$ of the configuration space into intervals of similar planar cell decompositions. The one-dimensional intervals $I \subseteq [0, 2\pi)$ define slices $\mathbb{R}^2 \times I$ in configuration space, that cut the stack into substacks. Corresponding regions in different layers of the stack form a subcell in the decomposition of the configuration space. All three-dimensional subcells in a slice span the entire slice from its lower boundary $\theta = \theta_0$ to its upper boundary $\theta = \theta_1$. Subcells appear or disappear only at slice boundaries. Moreover, assuming general position of the obstacles, exactly one subcell appears or disappears at any slice boundary.

The complexity of the resulting cell decomposition is determined by the number of critical orientations, which form the interval endpoints, and, hence, the slice boundaries. A critical orientation $\theta$ is an orientation for which one of the three conditions listed below is true ($\ell$ is the length of the ladder). The identification of the conditions is adopted from the paper by Leven and Sharir [64].

(C4) There exist two obstacle corners such that the open line segment connecting them is entirely contained in the closure of $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$ and has orientation $\theta$.

(C5) There exist an obstacle corner and a point on some obstacle edge such that the open line segment connecting the corner and the point is entirely contained in $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$ and has orientation $\theta$ and length $\ell$.

(C6) There exist two points on two obstacle edges and an obstacle corner $c$ such that the open line segment connecting the two points has orientation $\theta$ and length $\ell$, passes through $c$, and is entirely contained in $\mathrm{W} \setminus (\cup_{E \in \mathcal{E}} E)$ except at $c$.

Clearly, the number of critical orientations in the case of arbitrary polygonal obstacles is $O(n^2)$. The resulting free space decomposition consists of $O(n^2)$ cells. In contrast to the algorithm in Section 5.2, the number of critical events is not influenced by the possible fatness of the obstacles. This is most easily seen from a situation where we have $n$ square $2\pi$-fat obstacles placed in circular fashion (see Figure 5.3). In this example, the obstacle corner $v$ can be connected to any of the $2n - 2$ obstacle corners facing the interior of the circle by a line segment that is

The above lemma shows that the total number $K$ of intersections encountered during the sweep of the arrangement of the $O(n)$ (properly chosen) circles and the $O(n)$ (non-intersecting) obstacle edges equals $O(n)$. The running time of the sweep therefore amounts to $O((n+K)\log n) = O(n \log n)$. The output of the sweep are the constant size sets $V_2(S)$ of obstacles $E \in \mathcal{E}$. From the constant size unions $V_1(S_{m,\ell}) \cup V_2(S_{m,\ell})$, the sets $\sigma(R)$ with $R \ni m$ can be computed in constant time.

The modified version of Schwartz and Sharir's algorithm, which is tailored to fat obstacles, now consists of three steps: the first two are plane sweeps and the third combines a sequence of range search queries with a plane sweep. Each of the three steps runs in time $O(n \log n)$, so that the running time of the entire algorithm amounts to $O(n \log n)$ as well. The result is summarized in the following theorem.

**Theorem 5.10** *Schwartz and Sharir's cell decomposition algorithm* [84] *can be adapted to plan the motion of a ladder robot $\mathcal{B}$ with length $\ell \leq b \cdot \rho$ amidst the fat obstacles of $\mathcal{E}$ in time $O(n \log n)$, for any constant $b \geq 0$.*

### 5.2.3 A polygonal robot

In the preceding two subsections we have restricted our attention to a ladder moving amidst polygonal obstacles. Schwartz and Sharir's paper [84], however, also gives an algorithm for a polygonal robot.

The algorithm for a polygonal robot is similar to the algorithm for a ladder. The only difference concerns the definition of the critical curves. There are more and different types of critical curves in the polygonal case. Although the critical curves are different, the basic properties of these curves remain valid: features that are involved in the definition of a single critical curve are less than the diameter of the robot apart, and each point on a critical curve is less than the diameter of the robot away from the features that define it. The validity of these properties allows us to use a similar proof strategy and a similar approach for an algorithm in the case of a polygonal robot, resulting in the same $O(n \log n)$ complexity for the cell decomposition and for the motion planning algorithm.

## 5.3 A fatness-insensitive cell decomposition

A different example of the cell decomposition approach is the algorithm of Leven and Sharir presented in [64], which also applies to a ladder moving in a two-dimensional workspace amidst polygonal obstacles. Although the worst-case cell decomposition size and running time of the algorithm for general obstacles, $O(n^2)$ and $O(n^2 \log n)$ respectively, are superior to the $O(n^5)$ worst-case bounds for the Schwartz-Sharir algorithm, a simple example shows that the Leven-Sharir algorithm is inferior when the obstacles in the workspace are fat. More precisely, the example shows that motion planning problems with fat obstacles can still give rise to a decomposition of the free space into $\Omega(n^2)$ subcells.

enclosing squares and subsequently filtering out the obstacles that do not have a vertex inside $S$.

The nature of the second subproblem, reporting intersections of circles and obstacle edges, suggests the use of the plane sweep for reporting intersections of constant-complexity curves. By the proof of Lemma 5.6 or Theorem 2.9, each circle $S = S_{(x,y),\ell}$ is intersected by only a constant number of edges. This observation not only implies that each set $V_2(S)$ has constant cardinality but also that the total number of circle-edge intersections is $O(n)$. Unfortunately, we do not only encounter circle-edge intersections throughout the sweep: the circles also intersect each other. These intersections are non-interesting events with respect to finding the subcells induced by the non-critical regions, but they do affect the efficiency of the plane sweep. The number of such (irrelevant) circle-circle intersections seems impossible to bound without further provisions.

Before analyzing the number of circles intersecting a given circle, we recall that each critical curve section $\beta$ defines two circles, centered in *left*$(\beta)$ and *right*$(\beta)$, respectively. If we allow these centers to be anywhere in *left*$(\beta)$ and *right*$(\beta)$, then it seems impossible to bound the number of circle-circle intersections. However, if the centers of both circles are restricted to the vicinity of their implying critical curve sections, then proving an $O(n)$ bound on the number of intersections becomes feasible. In the modified version of Schwartz and Sharir's algorithm, we therefore take care to choose the circle centers within a distance, say, $\ell$ from the originating critical curve section $\beta$, thus allowing for an efficient sweep of the arrangement of curve and circle parts.

**Lemma 5.9** *If the centers of the circles implied by the $O(n)$ critical curve sections $\beta$ are chosen within a distance $\ell$ from $\beta$, then the number of circle-circle intersection is $O(n)$.*

**Proof:** Let us consider a circle $S_{m,\ell}$ and count the number of circles intersecting $S_{m,\ell}$. Any circle $S_{m',\ell}$ intersecting $S_{m,\ell}$ must clearly have its center $m'$ lying in the circle $S_{m,2\ell}$. As $m'$ was chosen within a distance $\ell$ from the critical curve section $\beta'$ defining $S_{m',\ell}$, this defining section $\beta$ must intersect the again larger circle $S_{m,3\ell}$. By Property 5.3 in turn, any point on $\beta$ lies within a distance $\ell$ from all its defining features, which must therefore intersect the circle $S_{m,4\ell}$. By Theorem 2.9, the number of obstacles, and, hence, obstacle features, intersecting $S_{m,4\ell}$ is bounded by a constant. This constant number of features can define at most a constant number of critical curves, that may possibly intersect $S_{m,3\ell}$. The (constant size) subset of critical curves that intersect $S_{m,3\ell}$ define only a constant number of critical curve sections $\beta'$. As each of the $O(1)$ sections defines two circles the total number of circles defined by such curve sections is constant. The circles $S_{m',\ell}$ that intersect $S_{m,\ell}$ must belong to this constant-size set of circles. Hence, any circle $S_{m,\ell}$ is intersected by $O(1)$ other circles, leading to the $O(n)$ bound on the total number of intersections. $\square$

After having computed the critical curves in $O(n \log n)$ time, we encounter a similar problem when computing the curve intersections (and the resulting critical curve sections). We could, naively, spend $O(n^2)$ by intersecting each curve with any other curve, although we know that the number of intersection is only $O(n)$ by Lemma 5.5. An obviously better idea is to cut the (constant-complexity) critical curves into a constant number of $x$-monotone subcurves and feed these to the plane sweep. The sweep reports the $K = O(n)$ intersections in time $O((n + K) \log n) = O(n \log n)$. The intersection points cut the critical curves into the desired critical curve sections.

For each of the $O(n)$ critical curve sections $\beta$, we have to compute $\sigma(\mathit{left}(\beta))$ and $\sigma(\mathit{right}(\beta))$. Computing $\sigma(\mathit{left}(\beta))$ requires choosing a point $(x, y) \in \mathit{left}(\beta)$, fixing the robot's endpoint $P$ at $(x, y)$, and reporting all features that can be touched by $\mathcal{B}$ and the orientations in which they are touched, which seems rather difficult to do efficiently. Fortunately, the set of features that are to be reported forms a subset of the set of features of the $O(1)$ obstacles (Theorem 2.9) that intersect the circular region $S_{(x,y),\ell}$ (centered at $(x, y)$ and with radius $\ell$). Checking all $O(n)$ obstacles for intersection with $S_{(x,y),\ell}$ is surely not the most efficient way of determining the $O(1)$ obstacles that intersect the circle. A better idea is to use the results on bounded size range searching from Chapter 3. The construction of the data structure storing all obstacles of $\mathcal{E}$ takes $O(n \log n \log \log n)$ time. The $O(n)$ queries with circles $S$ (with radius $\ell$) induced by all critical curve section then take $O(n \log n)$ in total. The entire computation takes $O(n \log n \log \log n)$. The following two-step approach, however, avoids the '$\log \log n$'-factor: first find for all circles $S$ the set $V_1(S)$ of obstacles that have a vertex inside $S$, and then report for all circles $S$ the set $V_2(S)$ of obstacles $E$ with edges that intersect the boundary $\partial S$ of $S$. Notice that the union of possibly overlapping sets $V_1(S)$ and $V_2(S)$ is clearly the set of all obstacles intersecting $S$. Below we solve the subproblems one by one.

The computation of the (constant-cardinality) sets $V_1(S)$ can be further simplified by realizing that $V_1(S)$ is a subset of the set of obstacles having a vertex inside the axis-parallel minimal enclosing square $C$ of $S$. The enclosing square $C$ of the circle $S$ with radius $\ell$ has side length $2\ell \leq 2b \cdot \rho$. By Theorem 2.9, the number of obstacles from $\mathcal{E}$ intersecting $C$ is constant, so definitely the number of obstacles having a vertex inside $C$ is constant. To solve the reduced problem we store all obstacle vertices in a data structure that supports efficient axis-parallel (or orthogonal) range searching queries. Preparata and Shamos' text book [79] on computational geometry gives an appropriate data structure, based on the layered range tree. The characteristics of the (layered range tree) structure are given as Lemma 5.8.

**Lemma 5.8** *There exists a data structure of size $O(n \log n)$ that answers planar range search queries among points in time $O(\log n + K)$, where $K$ is the number of answers to the query. Building the structure requires $O(n \log n)$ time.*

After having spent $O(n \log n)$ time to build the range searching structure, we can easily report all sets $V_1(S)$ in time $O(n \log n)$ by querying the structure with the

Instead we should use the knowledge that only two features that lie less than $\ell$ apart can define a curve of type 4 or 5. Let us first determine all pairs of features that lie less than $\ell$ apart. By Lemma 5.1, there exist $O(n)$ such pairs. After determining the close feature pairs $O(n)$ time suffices to find which of the corner-corner and corner-edge pairs indeed define curves of types 4 and 5.

To determine the pairs of features with distance at most $\ell$, we use the ideas of Sifrony and Sharir [93], which in turn rely on the techniques from [14]. Sifrony and Sharir wrap each obstacle edge and its two endpoints by a so-called *envelope*. The envelope of an edge $e$ is the set $\partial(e \ominus S_{O,\ell/2})$: the boundary of the Minkowski difference of $e$ and the circle with radius $\ell/2$ centered at the origin. Hence, the envelope of an edge $e$ equals the set of points with distance $\ell/2$ to $e$. As such, it consists of two straight segments parallel to $e$, and two circular arcs of arc length $\pi$ (half-circles) about $e$'s endpoints (see Figure 5.2). It is not too hard to see that

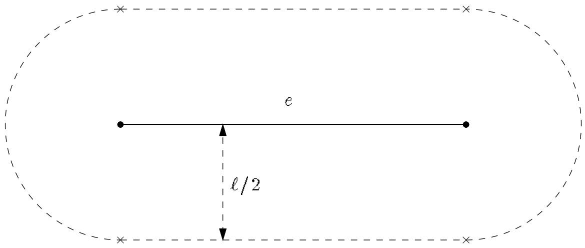


Figure 5.2: The envelope of an edge $e$ and its two endpoints.

the envelopes of the edges $e = v_1v_2$ and $e' = v_1'v_2'$ intersect if and only if features from $\{e, v_1, v_1\}$ and $\{e', v_1', v_2'\}$ lie less than $\ell$ apart. As a result, we can find the close pairs of features by sweeping the (constant-complexity) envelope curves in the plane. To satisfy the input requirement that the curves are $x$-monotone, we simply cut the circular arcs into two subarcs at $x$-extremal points. The envelope curves are labeled with the corresponding edge and endpoints[1].

Let us now consider the efficiency of the above sweep for envelope intersections, and, hence, for close feature pairs. By the convexity of the envelopes, two envelopes can only intersect in a constant number of points. Moreover, each reported intersection of two envelopes leads to a non-zero number of close feature pairs. As a result, the number of envelope intersections is of the same order of magnitude as the number of close feature pairs: $O(n)$ by Lemma 5.1. Reporting the $K = O(n)$ envelope curve intersections with the plane sweep takes $O((n + K)\log n) = O(n \log n)$ time. From the envelope intersections, the close feature pairs, and subsequently, the critical curves of type 4 and 5 can be computed in $O(n)$ time.

[1]Note that envelopes of consecutive edges on the boundary of a single obstacle may partially coincide. Potential problems with these coinciding parts are avoided if we merge these parts into a single curve, labeled with all corresponding edges and endpoints.

**Proof:** The number of subcells induced by a noncritical region $R$ depends on the number of obstacle features that can be touched by $\mathcal{B} = PQ$ while its endpoint $P$ is fixed at some arbitrary point $(x, y) \in R$. Obviously, such a feature intersects the circular region $S$ obtained by rotating $\mathcal{B}$ while keeping $P$ at $(x, y)$. The number of objects $E_i$ $(i \geq 1)$ intersecting the circle $S$ with radius $\ell \leq b \cdot \rho$ is constant by Theorem 2.9. As a consequence, the number of features $f$ intersecting $S$, and potentially touched by $\mathcal{B}$, is $O(1)$. So, the number of subcells induced by a noncritical region $R$ is bounded by a constant. □

Lemma 5.6 shows that each critical curve section $\beta$ adds at most twice a constant number of nodes to the connectivity graph. By Lemma 5.5, we conclude that the total number of nodes is $O(n)$. Let us count the adjacencies of a single node $N$ added by some section $\beta$. Assume without loss of generality that $N$ is induced by the noncritical region *left*$(\beta)$. The nodes that are adjacent to $N$ either correspond to the same subcell, or are induced by the noncritical region *right*$(\beta)$ and added by the section $\beta$. As all nodes corresponding to a single subcell are circularly connected, the number of adjacencies of the first type cannot exceed two. The number of adjacent nodes of the second type is constant by Lemma 5.6. Hence, each subcell is adjacent to a constant number of other subcells, resulting in a total of $O(n)$ graph edges.

**Theorem 5.7** *The connectivity graph corresponding to the cell decomposition of the free space of a ladder moving amidst k-fat obstacles has $O(n)$ nodes and edges.*

### 5.2.2 Computing the cell decomposition

Although the complexity of the connectivity graph corresponding to the cell decomposition is $O(n)$, a straightforward application of Schwartz and Sharir's method would result in $O(n^2)$ time to compute the decomposition. This bound turns up in each of the three steps in the algorithm: the first step where all critical curves are computed, the second step where all critical curve (inter)sections are computed, and the third step where all subcells induced by a single noncritical region are determined. If we incorporate the plane sweep ideas by Bentley and Wood [14] for reporting geometric intersections in each of the three steps, then the efficiency of each individual step is enhanced to $O(n \log n)$. For a discussion of the main ingredients of a plane sweep, we refer to Section 7.1. Here, we confine ourselves to mentioning that the $K$ intersections of $n$ line segments in the plane can be reported in time $O((n+K)\log n)$. The ideas are straightforwardly generalized to $x$-monotone constant-complexity curves. Below we redefine each of the three above steps as a problem of reporting constant-complexity curve intersections.

We have shown in the previous subsection that the number of type 4 or 5 curves is linear in the case of fat obstacles. Each of these curves is determined by two features. We could naively try all possible pairs of features to find out which pairs generate a curve. This strategy would require $\Omega(n^2)$ time to find the $O(n)$ curves.

**Lemma 5.5** *The number of critical curve sections in the workspace is $\mathcal{O}(n)$.*

**Proof:** Let us bound the number of critical curve intersections $p$ implied by $f$ and a set $F$ of lexicographically larger features ($1 \leq |F| \leq 3$); the features in $\{f\} \cup F$ define the two curves $\beta$ and $\beta'$ intersecting in $p$. By Property 5.3, the distances from the intersection point $p$ to the defining features $\{f\} \cup F$ of $\beta$ and $\beta'$ do not exceed $\ell$. As a result, the distance from any feature $f' \in F$ to $f$ is at most $2\ell$. Moreover, the distance from the object $E_i$ containing $f$ to any feature $f' \in F$ is bounded by $2\ell \leq 2b \cdot \rho \leq 2b \cdot \rho_i$. Each lexicographically larger feature $f' \in F$ belongs either to $E_i$ or to an obstacle $E_j$ with $j > i$. By Corollary 2.10, the number of such obstacles $E_j$ within a distance $b \cdot \delta_i$ from $E_i$ is bounded by a constant. Combined with the constant complexity of these obstacles and of $E_i$, we find that there is also at most a constant number of candidates for inclusion in $F$. Hence, there exist only a constant number of sets $F$ of features that, together with $f$, can imply a pair of intersecting curves $\beta$ and $\beta'$. The low degree of the curves implies that the number of intersections of a pair of curves $\beta$ and $\beta'$ is bounded by a constant, so any choice for $F$ can contribute no more than $O(1)$ intersections. Adding up the contributions of all features $f$ yields a total of $O(n)$ intersections and, hence, critical curve sections. □

We have seen that the number of connectivity graph nodes added by the critical curve section $\beta$ equals the number of subcells induced by the region *left*$(\beta)$ plus the number of subcells induced by the region *right*$(\beta)$. If $\beta$ is a type 0 curve, only one of the two regions is noncritical, in all other cases both regions will be noncritical. A region that is *not* noncritical will induce no graph nodes.

Now, we analyze the number of subcells induced by a single noncritical region $R$. In Schwartz and Sharir's method, each pair $[s_1, s_2] \in \sigma(R)$ defines a subcell. Hence, the number of subcells induced by a noncritical region $R$ is determined by the number of pairs in $\sigma(R)$, each pair in $\sigma(R)$ consisting of two different contact placements for $\mathcal{B}$ with $P$ fixed at some point in $R$. The number of subcells induced by $R$ is therefore determined by the number of different contact placements for $\mathcal{B}$ when we fix its endpoint $P$ at some point $(x, y)$ in $R$ and vary its orientation $\theta$. Note that, due to the shape of the features, each feature can be touched by $\mathcal{B}$ in at most two different orientations while its endpoint $P$ is fixed at $(x, y)$. In the case of arbitrary polygonal obstacles, we can easily construct examples where the robot can touch any of the $O(n)$ obstacle features, each at a different orientation $\theta$. A noncritical region $R$ can therefore induce $O(n)$ subcells. Since the number of noncritical regions is $O(n^4)$, we obtain a total number of $O(n^5)$ subcells. As before, things are different in a fat setting. It is easy to see that an obstacle feature $f$ touched by $\mathcal{B}$ with $P$ fixed at $(x, y)$ in some contact position must lie close to $(x, y)$. Using the low density property of spaces with fat objects, we can bound the number of such features $f$.

**Lemma 5.6** *Each noncritical region in the workspace induces only $O(1)$ subcells in the configuration space.*

### 5.2.1 Complexity of the cell decomposition

We recall that the number of obstacle edges and corners is $O(n)$. First, we observe that the number of type 0-3 curves is not influenced by the fatness of the obstacles and is $O(n)$ in both the arbitrary and the fat case.

In the general case of arbitrary polygonal obstacles, the number of type 4 curves is $O(n^2)$ since each pair of corners may define such a curve. The same number applies to type 5 curves since each pair of one edge and one (convex) corner may induce such a curve. As a result, the total number of critical curves is $O(n^2)$. Each pair of curves may intersect, so that there can be up to $O(n^4)$ intersections, and, hence, $O(n^4)$ critical curve sections.

Things change if we assume the obstacles to be $k$-fat. Let us first observe an important property of all critical curves, following from the definitions of the curves.

**Property 5.3** *Each point on a critical curve is less than the length $\ell$ of the ladder away from the obstacle features (corners, edges) that define this curve.*

The other important tools in the analysis of the 'fat case' are the low object density results Theorem 2.9, Corollary 2.10, and Lemma 5.1. Like in the previous section, the length $\ell$ of the robot $\mathcal{B} = PQ$ is bounded by $\ell \leq b \cdot \rho \leq b \cdot \rho_i$, for all $1 \leq i \leq n$.

Lemma 5.4 bounds the number of critical curves of type 4 and 5. The lemma follows more or less directly from the definition of the curves and Lemma 5.1, which bounds the overall number of feature pairs, and, hence, the number of feature pairs defining critical curves of type 4 and 5.

**Lemma 5.4** *The number of critical curves of type 4 and 5 in the workspace is $\mathcal{O}(n)$.*

**Proof:** Each pair of obstacle corners with mutual distance at most $\ell$ may, under some additional conditions, like relative angles of incoming edges, define one critical curve of type 4. The number of corner pairs lying less than $\ell$ apart is bounded by $O(n)$ by Lemma 5.1. Thus, the number of type 4 curves is bounded by $O(n)$. Each pair of an obstacle corner and an obstacle edge with mutual distance at most $\ell$ may, again under some additional conditions, define one curve of type 5. Lemma 5.1 bounds the number of such pairs and, hence, the number of type 5 curves, by $O(n)$. □

Each of the of $O(n)$ critical curves may be intersected by other critical curves and, as a result of that, be cut into a number of critical curve section. Since each critical curve $\beta$ is defined by one or two obstacle features, an intersection point $p = \beta \cap \beta'$ can be regarded as being implied by the union of the two sets of defining features. Each intersection is as such implied by a collection of two, three, or four obstacle features. If we now charge each intersection $p$ to the lexicographically smallest of these features, then we find again that each feature is charged at most a constant number of times. Adding up all contributions of features $f$ leads to a total of $O(n)$ intersections and, hence, critical curve sections.

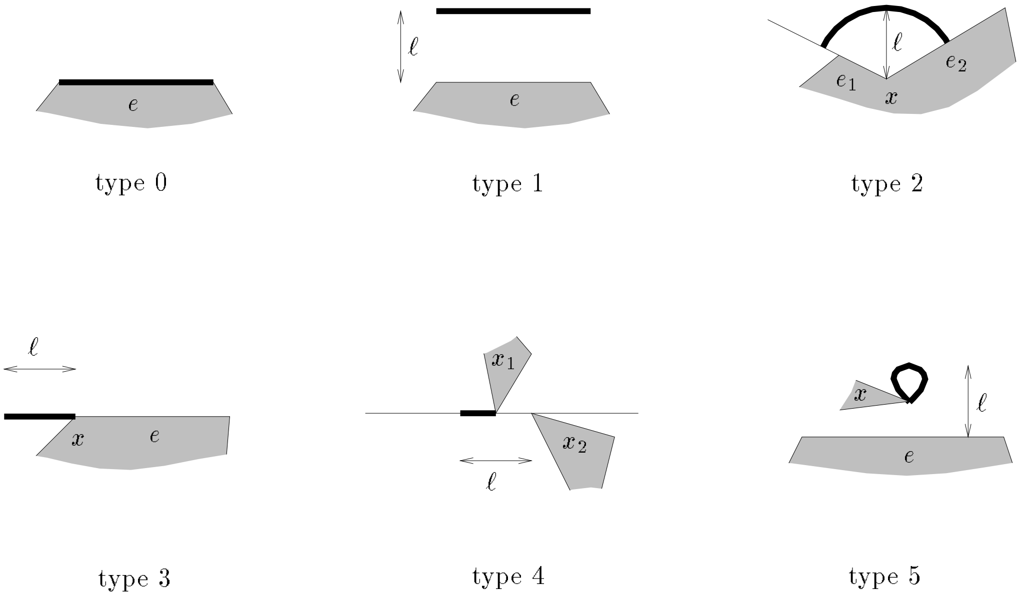


Figure 5.1: The six types of critical curves in Schwartz and Sharir's solution to the Piano Movers' problem.

single noncritical region $R$, the sets $\sigma(x,y)$ and $\sigma(x',y')$ are equal [84]. In the sequel, we use the abbreviation $\sigma(R) = \sigma(x,y)$, where $(x,y)$ is any point in $R$. Each pair of contact positions $[s_1, s_2] \in \sigma(R)$ defines a cell in the cell decomposition of FP.

Schwartz and Sharir's method first computes all critical curves, and then all intersections of the curves, resulting in a collection of intersection points and a collection of critical curve sections. With each intersection point, we store the critical curve sections that are incident at this intersection point. Each critical curve section $\beta$ separates two regions; we arbitrarily call one of the regions *left*$(\beta)$ and the other one *right*$(\beta)$. Next, we compute $\sigma($*left*$(\beta))$ and $\sigma($*right*$(\beta))$, define a connectivity graph node for each subcell $[s_1, s_2]$ induced by the regions *left*$(\beta)$ and *right*$(\beta)$, and build the adjacency relation (based on the adjacency of the corresponding subcells) between the nodes induced by both regions. (Note that each node is generated by a single critical curve section.) If we repeat this procedure for every critical curve section, each subcell is represented in the connectivity graph as many times as there are critical curve sections bordering the region that induced the subcell. All nodes that correspond to the same subcell are circularly connected.

ourselves to summarizing the different types of critical curves. We adopt the classification of the critical curves used in Latombe's book [59]. Let $P$ and $Q$ be the endpoints of the ladder $\mathcal{B}$ and let $\ell = |PQ|$ be its length. Choose $P$ as the robot's reference point.

- An obstacle edge is a critical curve of **type 0**.
- Let $e$ be an obstacle edge. The line segment at a distance $\ell$ from $e$ is a critical curve of **type 1**. The length of the critical curve equals the length of $e$.
- Let $x$ be an obstacle corner and let $e_1$ and $e_2$ be the edges emerging from $x$. The circular arc with radius $\ell$, centered at $x$, and running between the half-lines starting at $x$ and containing the edges $e_1$ and $e_2$ respectively, is a critical curve of **type 2**.
- Let $x$ be a convex obstacle corner and let $e$ be one of the edges emerging from $x$. The line segment traced out by $P$ while $\mathcal{B}$ slides along $e$, so that $Q$ touches $e$ and $x$ touches $\mathcal{B}$, is a critical curve of **type 3**. The curve is the extension with length $\ell$ of the edge $e$ at $x$.
- Let $x_1$ and $x_2$ be convex obstacle corners such that the line passing through $x_1$ and $x_2$ is tangent to the obstacle set $\mathcal{E}$ in both $x_1$ and $x_2$. The line segment traced out by endpoint P, while $\mathcal{B}$ slides along $x_1$ and $x_2$, is a critical curve of **type 4**. Note that the distance from $x_1$ to $x_2$ must be less than $\ell$.
- Let $x$ be a convex obstacle corner and let $e$ be an obstacle edge such that $x$ is not an endpoint of $e$. The curve traced out by $P$ while $Q$ slides along $e$ and while $\mathcal{B}$ remains in contact with $x$, is a (fourth degree) critical curve of **type 5**. Note that again the distance from $x$ to $e$ must be less than $\ell$.

Figure 5.1 illustrates the various critical curve types. The (intersecting) critical curves partition $W = \mathbb{R}^2$. The part of a critical curve between two points of intersection with other critical curves is called a *critical curve section*. A position $(x, y)$ of the robot $\mathcal{B}$ is *admissible* if there exists an orientation $\theta$, such that $(x, y, \theta) \in$ FP. A noncritical region is a maximal subset of admissible robot positions intersecting no critical curves. Hence, the critical curves determine a set of noncritical regions in W.

Let $\Theta_{x,y} = \{\, \theta \mid (x, y, \theta) \in \mathrm{FP} \,\}$ be the set of free orientations of $\mathcal{B}$ with $P$ fixed at a point $(x, y)$ in a noncritical region $R$. The set $\Theta_{x,y}$ consists of a finite number of open maximum connected intervals. For each such interval $(\theta_1, \theta_2) \in \Theta_{x,y}$, both the robot placement $(x, y, \theta_1)$ and the robot placement $(x, y, \theta_2)$ are placements in which the robot touches the obstacle set. The unique stop touched by $\mathcal{B}$ in the contact placement $(x, y, \theta_1)$ (resp. $(x, y, \theta_2)$) is denoted by $s(x, y, \theta_1)$ (resp. $s(x, y, \theta_2)$). The set of all pairs $[s(x, y, \theta_1), s(x, y, \theta_2)]$ such that $(\theta_1, \theta_2) \in \Theta_{x,y}$ is referred to as $\sigma(x, y)$. The critical curves are defined so that for each pair of points $(x, y)$ and $(x', y')$ in a

features then, first of all, each pair is counted, but, more importantly, it turns out that each feature gets charged only a constant number of times. As a result the total number of chargings, and, hence, close feature pairs, adds up to $O(n)$.

**Lemma 5.1** *Let $k$ and $b$ be positive constants and let $\mathcal{E}$ be a set of polygonal $k$-fat constant-complexity obstacles with minimal enclosing circle radii at least $\rho$. Then the number of feature pairs (edges, corners) that lie less than $b \cdot \rho$ apart is $O(n)$.*

**Proof:** Let us count all lexicographically larger features $f'$ that lie within a distance $b \cdot \rho$ from a feature $f$. Clearly, the distance from the obstacle $E_i$ containing $f$ to the feature $f'$ is bounded by $b \cdot \rho \leq b \cdot \rho_i$. The lexicographically larger feature $f'$ belongs, by definition, either to $E_i$ or to an obstacle $E_j$ with $j > i$. By Corollary 2.10, the number of such obstacles $E_j$ within a distance $b \cdot \rho_i$ from $E_i$ is bounded by a constant. Combined with the constant complexity of these obstacles and of $E_i$, it follows that there is at most a constant number of choices for $f'$. The $O(n)$ bound follows after summing over all $f$. $\square$

Lemma 5.1 proves that $K = O(n)$. We obtain the following final result.

**Theorem 5.2** *Sifrony and Sharir's boundary-vertices retraction algorithm* [93] *plans the motion of a ladder robot $\mathcal{B}$ with length $\ell \leq b \cdot \rho$ amidst the fat obstacles of $\mathcal{E}$ in time $O(n \log n)$, for any constant $b \geq 0$.*

## 5.2 Fatness-sensitive cell decomposition

This section considers the combinatorial and algorithmic consequences of fatness for the famous Piano Movers' algorithm by Schwartz and Sharir [84]. Subsection 5.2.1 shows that the complexity of the cell decomposition of FP computed by the method is $O(n)$, under the assumption of fat obstacles and a bounded-size robot, whereas the bound is $O(n^5)$ in the general case. The algorithmic part in Subsection 5.2.2 deals with the efficient computation of the decomposition; the subsection improves the direct bound of $O(n^2)$ to $O(n \log n)$.

Schwartz and Sharir [84] apply the cell decomposition technique to obtain an $O(n^5)$ algorithm for planning the motion of a ladder $\mathcal{B}$ moving amidst polygonal obstacles $\mathcal{E}$ in the plane. Their method decomposes $W = \mathbb{R}^2$ into so-called *noncritical* regions, lifts these regions into three-dimensional cylinders (in $C = \mathbb{R}^2 \times [0, 2\pi)$), decomposes the free part of the cylinders into subcells, and finally captures the adjacency of the subcells in a connectivity graph. We will go into more detail on each of these steps and, while doing so, focus on the consequences of fatness for each of these steps. We emphasize that we will not give an extensive explanation of the ladder algorithm. The reader is referred to the original paper [84] or Latombe's book [59] for a detailed description.

The noncritical regions in the robot's workspace $W = \mathbb{R}^2$ are defined by *critical curves*. The meaning of these curves is not important in our analysis, so we restrict

polygons by Schwartz and Sharir [84] appears to be surprisingly sensitive to the complexity of the free space. The algorithm outputs a decomposition of the free space into $O(n)$ subcells for problems involving a bounded-size robot amidst fat obstacles, whereas the worst-case number of subcells for general settings can be as high as $O(n^5)$. Without modifications, the algorithm computes such a cell decomposition in time $O(n^2)$. A number of adaptations enhance the running time to $O(n \log n)$. We discuss these results quite thoroughly as they provide the main ideas for the general paradigm of Chapter 6. The algorithm by Leven and Sharir [64] for a ladder among polygons does not benefit from the fatness of the obstacles, although its worst-case behavior is superior to that of Schwartz and Sharir. We give an example with fat obstacles that leads to a cell decomposition consisting of $\Omega(n^2)$ subcells, which equals the worst-case number of subcells for general obstacles.

In Section 5.4, the claim of Avnaim, Boissonnat, and Faverjon [10] that their boundary cell decomposition algorithm performs considerably better than the worst-case $O(n^3 \log n)$ running time if the workspace has a low obstacle density is confirmed for workspaces that satisfy our assumptions. We find that the running time of the algorithm indeed reduces to $O(n \log n)$.

Throughout the entire chapter, it is assumed that a constant-complexity rigid robot $\mathcal{B}$ moves in a two-dimensional Euclidean workspace amidst a collection $\mathcal{E}$ of $n$ polygonal $k$-fat obstacles, for some positive constant $k$. Each individual obstacle $E \in \mathcal{E}$ has constant complexity and a minimal enclosing hypersphere radius of at least $\rho$. As an exception, the bounds on the sizes of ladder robots are expressed in terms of their lengths instead of their reaches, to conform to the original papers. Needless to say is that the length of a ladder is closely related to its reach. Note that the robot itself need not be fat. The assumptions are generally omitted in the formulation of the results of this chapter.

## 5.1 Boundary-vertices retraction

Let us first consider the boundary-vertices retraction method of Sifrony and Sharir [93] for planning the motion of a ladder amidst polygonal obstacles, which runs in time $O(K \log n)$, where $K$ is the number of pairs of obstacle corners (vertices) that lie less than the length of the ladder apart. We will prove that in the indicated setting $K = O(n)$, yielding an efficient $O(n \log n)$ algorithm.

For simplicity, we assume that the polygonal obstacles in $E_1, \ldots, E_n \in \mathcal{E}$ are ordered by increasing minimal enclosing circle radii $\rho_1, \ldots, \rho_n$, and furthermore that the features of each object $E_i$ are ordered in some way $f_{i1}, \ldots, f_{ic}$ ($c = O(1)$ by the constant complexity of $E_i$). Note that a feature that appears after feature $f$ in the lexicographical ordering of features either belongs to the same obstacle as $f$ or to a larger obstacle.

Lemma 5.1 bounds the number of feature pairs with small mutual distance. If we charge each such close pair to the lexicographically smallest of the two involved

# Chapter 5

# Existing algorithms and fat obstacles

Exact motion planning algorithms process the free space FP for answering path finding queries. The linear complexity result for FP does not directly imply that the outcome of the processing, a representation of FP, has linear complexity as well. Moreover, if an algorithm succeeds in supplying a linear complexity representation, then it may still take far more time to compute this representation.

Before we focus on a general paradigm for motion planning amidst fat obstacles in the next chapter, we study the influence of fatness on a number of existing algorithms for moving a translating and rotating rigid robot among polygonal obstacles. This specific problem has been studied extensively in the mid-80's which has resulted in a number of algorithms with varying efficiency. (In fact, planar motion amidst polygonal obstacles is the most extensively studied motion planning problem.) The algorithms that are discussed below constitute an interesting cross-section of the available algorithms and illustrate as such the differences between the various approaches to solving the problem. We consider examples of both major exact approaches to motion planning: cell decomposition and retraction. The aim of studying the algorithms is to learn the specific combinatorial and algorithmic properties that lead to efficient motion planning algorithms, so that we can use the results in finding an efficient paradigm for general fat motion planning.

The running time of the boundary-retraction algorithm for a ladder among polygons by Sifrony and Sharir [93] is sensitive to the number $K$ of pairs of obstacle corners that lie less than the length of the ladder apart. Section 5.1 confirms the intuitive feeling that the low obstacle density implied by the fatness and the bound on the relative sizes of the robot and the obstacles cause $K$ to be only $O(n)$ instead of $O(n^2)$. The performance of the Voronoi-based retraction algorithms [70, 71] though is not enhanced by our assumptions as the worst-case size of the respective Voronoi diagrams does not benefit from them.

Sections 5.2 and 5.3 consider two cell decomposition algorithms. The famous Piano Movers' algorithm for planning the motion of a ladder or polygon among

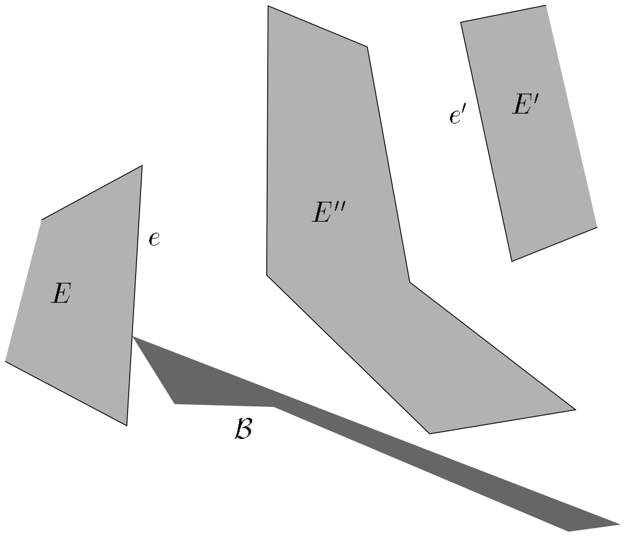


Figure 4.8: The robot $\mathcal{B}$ touching the edge $e$ of obstacle $E$ is long enough to touch $e$ and the edge $e'$ of $E'$ simultaneously. Nevertheless, it is unable to do so, because the obstacle $E''$ is in its way.

of larger obstacles $E'$, however, seems in no way relevant in obtaining a constant bound. The general position assumption, although common in motion planning, is therefore not very essential to the validity of our result.

A second glance at the proof of Theorem 4.3 learns that most of the assumptions are not used explicitly. Instead, the combination of the assumptions leads to a low obstacle density in the workspace, which basically means that any workspace region with size comparable to the reach of the robot intersects only a constant number of obstacle features. This implied workspace property, rather than the individual assumptions, is essential to the proof of Theorem 4.3. The linear bounds on the numbers of $j$-fold contacts can therefore be extended to motion planning problems for constant-complexity robots in workspaces that satisfy the low density property. If, in addition, the constraint hypersurfaces defined by the robot-obstacle contacts are algebraic of bounded degree, then the linear free space complexity result of Corollary 4.4 extends to such motion planning problems as well.

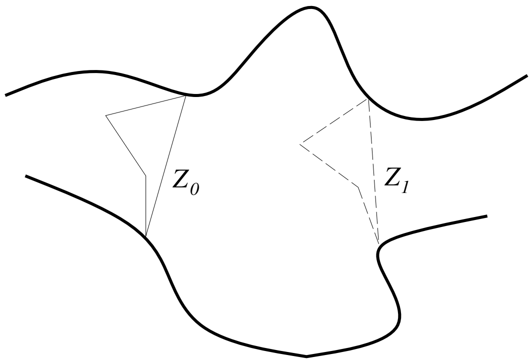


Figure 4.7: There is no continuous motion of the robot from $Z_0$ to $Z_1$ during which it remains in contact with both features.

only have $f-1$ different values, so the total number of multiple contacts is linear and, hence, the free space has linear complexity.

**Corollary 4.4** *Let $d, k \geq 1$ and $b \geq 0$ be positive constants and let $\mathcal{E}$ be a set of $n$ $k$-fat obstacles in $\mathbb{R}^d$ of constant complexity each and with minimal enclosing hypersphere radii at least $\rho$. The robot $\mathcal{B}$ with constant complexity, $f$ degrees of freedom, and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moves in $\mathrm{W} = \mathbb{R}^d$ amidst the obstacles of $\mathcal{E}$. Then, the free space for the robot $\mathcal{B}$ moving amidst the $k$-fat obstacles of set $\mathcal{E}$ has linear complexity.*

The linear upper bound on the free space complexity obviously imposes an equal bound on the complexity of a single free cell.

The constant that we obtained in Theorem 4.3 can be quite high: the first contact for the robot is a feature of $E$, but each of the other $j-1$ contacts are chosen from all features in the proximity of $E$. In practice, this approach yields a bound that is far from tight because many of the features in $E$'s proximity cannot be touched by $\mathcal{B}$ while it touches some feature of $E$. Figure 4.8 shows an example of such a situation. Even though the distance between two features alone may allow the robot to touch both of them simultaneously, the positions of the obstacles in the workspace may prevent the robot from actually doing so. So, only a subset of all theoretical combinations of features really implies a multiple contact in the configuration space. Clearly, the number of actual $j$-fold contacts for $\mathcal{B}$ will remain far below the upper bound of Theorem 4.3.

The framework of assumptions that leads to the linear free space complexity includes a general position assumption for the obstacles. The assumption basically simplifies the analysis by allowing us to neglect multiple contacts involving more than $f$ pairs of features. The upper bound of $f$ on the value of $j$ in counting the number of $j$-fold contacts involving a feature of some obstacle $E$ and features

**Theorem 4.3** *Let $d, k \geq 1$ and $b \geq 0$ be constants and let $\mathcal{E}$ be a set of $n$ $k$-fat obstacles in $\mathbb{R}^d$ of constant complexity each and with minimal enclosing hypersphere radii at least $\rho$. The robot $\mathcal{B}$ with constant complexity, $f$ degrees of freedom, and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moves in $\mathrm{W} = \mathbb{R}^d$ amidst the obstacles of $\mathcal{E}$. Then, for each $2 \leq j \leq f$, the number of $j$-fold contacts of the robot $\mathcal{B}$ is linear in the number of obstacles: $\mathcal{O}(n)$.*

**Proof:** Consider some obstacle $E \in \mathcal{E}$ and let $\rho_E \geq \rho$ be its minimal enclosing hypersphere radius. Let us count the number of $j$-fold contacts of $\mathcal{B}$ that involve $E$ and obstacles $E'$ with larger minimal enclosing hypersphere radii. Such an obstacle $E'$ must lie within a distance $2\rho_{\mathcal{B}}$ from $E$ in order to allow $\mathcal{B}$ to touch $E$ and $E'$ simultaneously (because the reach $\rho_{\mathcal{B}}$ of $\mathcal{B}$ bounds $\mathcal{B}$'s maximum diameter by $2\rho_{\mathcal{B}}$). Let $p$ be the number of obstacles $E'$ that lie within a distance $2\rho_{\mathcal{B}}$ from $E$. Since $2\rho_{\mathcal{B}} \leq 2b \cdot \rho \leq 2b \cdot \rho_E$, we know by Property 4.2 that $p \leq k \cdot (2b+2)^d = O(1)$.

A single $j$-fold contact is determined by $j$ different pairs, each pair consisting of a robot feature and an obstacle feature. Let us assume that the robot has $x_{\mathcal{B}}$ different features and that the number of features of each obstacle $E$ is bounded by $x_{\mathcal{E}}$. The first contact is a contact between a robot feature and a feature of the obstacle $E$. We have at most $x_{\mathcal{B}} \cdot x_{\mathcal{E}}$ choices for this contact. For each of the $j-1$ remaining contacts we can choose the obstacle feature on each of the $p$ obstacles in the proximity of $E$, which gives a total number of $x_{\mathcal{E}} \cdot p$ possibly involved obstacle features. For each contact we can again choose from all $x_{\mathcal{B}}$ robot features. Hence, the total number of $j$-fold contacts involving $E$ is bounded by $(x_{\mathcal{B}} \cdot x_{\mathcal{E}} \cdot p)^{j-1} \times x_{\mathcal{B}} \cdot x_{\mathcal{E}}$, which is a constant.

Adding all the $n$ constant upper bounds results in an overall upper bound on the number of $j$-fold contacts of $n \cdot ((x_{\mathcal{B}} \cdot x_{\mathcal{E}} \cdot p)^{j-1} \times x_{\mathcal{B}} \cdot x_{\mathcal{E}})$, which is $\mathcal{O}(n)$, since $x_{\mathcal{B}}$, $x_{\mathcal{E}}$, $p$, and $j$ are constants. $\Box$

Note that the value of $j$ in Theorem 4.3 ranges from 2 to $f$. The number of single contacts is of course also linear, because the number of pairs of a robot feature and an obstacle feature is linear. The case $j = 1$ is deliberately excluded from Theorem 4.3 to emphasize that fatness 'only' reduces the number of *multiple* contacts, and not the number of single contacts.

The $(f-j)$-dimensional subspace defined by a single $j$-fold contact is not necessarily connected. Figure 4.7 shows an example for $f = 3$ and $j = 2$, where it is impossible for the robot to move from $Z_0$ to $Z_1$ without losing contact with either the upper or the lower obstacle feature. The 1-dimensional subspace induced by the contact with both features is therefore non-connected. Our assumption that all contact hypersurfaces are of bounded degree, however, implies that the number of different connected subspaces induced by a single multiple contact is bounded by some small constant. The complexity of the free space is now solely determined by the number of multiple contacts, since the contribution of a single multiple contact to the free space apparently has constant complexity. Variable $j$ in Theorem 4.3 can

can be discarded without affecting the complexity of the free space. We see that a robot $\mathcal{B}$ with $f$ degrees of freedom can have up to $f$ simultaneous contacts with the boundaries of the obstacles in $\mathcal{E}$.

We consider the situation where a robot $\mathcal{B}$ moves amidst $n$ $k$-fat obstacles $E \in \mathcal{E}$ in general position, where $k \geq 1$ is a constant. The robot as well as the individual obstacles are assumed to have constant complexity, so the number of robot features is $O(1)$ and the number of boundary features in the obstacle set $\mathcal{E}$ is $O(n)$. As a consequence, the total number of hypersurfaces is $\mathcal{O}(n)$. The hypersurfaces are assumed to be algebraic of bounded degree, so that the intersection of any $j$ hypersurfaces consists of at most a constant number of connected components. To successfully apply Property 4.2, the robot $\mathcal{B}$ is assumed to be not too big compared to the obstacles. Let $\rho$ be a lower bound on the minimal enclosing hypersphere radii of all obstacles. The reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is constrained by $\rho_{\mathcal{B}} \leq b \cdot \rho$, where $b$ is some positive constant. This assumption regarding the size of the robot is not very restrictive: it basically rules out the situation where the robot $\mathcal{B}$ is so large that it would make the obstacles into point obstacles relative to its own size. The assumption will be satisfied in most practical cases. (In the previous subsection we already saw that such a restriction is required to obtain low free space complexities.) We summarize the assumptions below.

- The workspace W of the robot $\mathcal{B}$ is the $d$-dimensional Euclidean space $\mathbb{R}^d$.
- The workspace W of the robot $\mathcal{B}$ contains a collection $\mathcal{E}$ of $n$ $k$-fat obstacles $E \subseteq \mathbb{R}^d$ in general position, for some constant $k \geq 1$.
- The reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is bounded by $\rho_{\mathcal{B}} \leq b \cdot \rho$, where $b \geq 0$ is a constant and $\rho$ is a lower bound on the minimal enclosing hypersphere radii of all obstacles $E \in \mathcal{E}$.
- The robot $\mathcal{B}$ has constant complexity.
- Each obstacle $E \in \mathcal{E}$ has constant complexity.
- The hypersurface in the configuration space corresponding to the set of robot placements in which a certain robot feature is in contact with a certain obstacle feature is algebraic of bounded degree.

The assumptions remain valid throughout the remaining chapters.

The proximity result given in Property 4.2 is the key to successful application of a proof strategy that repeatedly considers an obstacle $E$ and counts the number of multiple contacts for the robot $\mathcal{B}$ involving $E$ and obstacles with larger minimal enclosing hypersphere radii. Property 4.2 guarantees that we find a constant upper bound on this number for each obstacle $E$. The resulting overall number of multiple contacts will be linear, which is stated in Theorem 4.3.

involves a 'sup' instead of a 'max' because the rest-space $D$ might be open, like e.g. $D = [0, 2\pi)$ in the case of the planar rotating and translating rigid robot. Notice that the definition of the reach causes any robot with reach $\rho_{\mathcal{B}}$ and its reference point $O$ placed at $p \in \mathrm{W}$ to be completely contained in the hypersphere $S_{p,\rho_{\mathcal{B}}}$ with radius $\rho_{\mathcal{B}}$ centered at $p$, regardless of the actual placement $Z$ of the robot. Note furthermore that the maximum diameter of a robot with reach $\rho_{\mathcal{B}}$ is $2\rho_{\mathcal{B}}$. In the remainder of the thesis, the reach of the robot will be used as the main means of expressing the robot size.

A convenient strategy for bounding the number of multiple contacts is by charging each multiple contact to the smallest obstacle involved in the contact, and subsequently bounding the number of chargings to any obstacle $E$. The observation in the previous paragraph learns that all features involved in a contact of $\mathcal{B}$ with $E$ and larger obstacles must lie in the proximity of $E$. Corollary 2.10 supports this strategy by supplying a valuable bound on the number of obstacles (and, hence, on the number of features of obstacles) larger than $E$ that lie in $E$'s proximity. The corollary, which we recall below as Property 4.2 in a form that is tailored to our current needs, also gives clear indications on what additional assumptions on the workspace W and obstacles $\mathcal{E}$ are required for obtaining a linear number of multiple contacts.

**Property 4.2** *Let $k \geq 1$ and $b \geq 0$ be constants and let $\mathcal{E}$ be a set of non-intersecting $k$-fat objects in $\mathbb{R}^d$. Let $E \in \mathcal{E}$ be an object with minimal enclosing hypersphere radius $\rho$. Then the number of objects $E' \in \mathcal{E}$ with larger minimal enclosing hypersphere radii within a distance $2b \cdot \rho$ from $E$ is bounded by the constant $k \cdot (2b+2)^d$.*

The importance of the property becomes clear if we realize that a robot with reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ can only simultaneously touch obstacles that are less then $2\rho_{\mathcal{B}} \leq 2b \cdot \rho$ apart. The features involved in a multiple contact of $\mathcal{B}$ (with minimal enclosing hypersphere radius $\rho_{\mathcal{B}} \leq b \cdot \rho$) with $E$ and larger must clearly be among the features of the at most $k \cdot (2b+2)^d = O(1)$ obstacles in the proximity of $E$.

Before we focus on the problem of finding an upper bound on the number of multiple contacts, we briefly reconsider the notion of multiple contact itself. What kind of subspaces of the configuration space are defined by multiple contacts and how many obstacles can participate in a multiple contact?

The set of placements of the robot $\mathcal{B}$ in which a certain feature of $\mathcal{B}$ is in contact with a boundary feature (of an obstacle) in $\mathcal{E}$ of appropriate dimension forms an $(f-1)$-dimensional subspace (or hypersurface) in the $f$-dimensional configuration space. An intersection of $j$ of these hypersurfaces corresponds to a simultaneous contact of the robot with $j$ obstacle boundary features in $\mathcal{E}$. Such an intersection is an $(f-j)$-dimensional subspace of the configuration space. Consequently, the $f$-fold contacts appear at isolated points in the configuration space, and, hence, fix the position of the robot. Contacts that involve more than $f$ obstacle features do not appear if we assume that the obstacles are in general position. Such contacts

the complexity of a single cell in an arrangement of $n$ low-degree algebraic surface patches in 3-space of $O(n^{2+\epsilon})$, for any $\epsilon > 0$, where the constant of proportionality depends on $\epsilon$. The result bounds the complexity of a single connected component of the free space for motion planning problems with three degrees of freedom involving a robot and obstacles of constant complexity.

Ke and O'Rourke [50] report a lower bound for a more complicated spatial motion planning problem, namely that of a translating and rotating ladder ($f = 5$) among polyhedral obstacles. They show that $\Omega(n^4)$ distinct moves, or constant degree curves in configuration space, may be necessary to connect two placements of the ladder. It is not hard to understand that $\Omega(n^4)$ moves can only be necessary within a free cell with at least the same complexity $\Omega(n^4)$, and, hence, also in a free space with complexity $\Omega(n^4)$.

## 4.3 Fat obstacles and the free space complexity

In this section, we deduce a linear bound on the complexity of the free space for a robot moving amidst fat obstacles, and formulate the necessary additional assumptions that lead to the result. The considerations in the first section of this chapter show that the number of multiple contacts is of the same order of magnitude as the complexity of the free space, provided that the constraint hypersurfaces are algebraic of bounded degree. Under this assumption, we may therefore settle for a bound on the number of multiple contacts to bound the complexity of the free space. A very useful observation now is that two obstacle features that are far apart (more than the maximum diameter of the robot) cannot be involved in any multiple contact for $\mathcal{B}$, simply because $\mathcal{B}$ is unable to touch both simultaneously. Hence, obstacles that do induce such a contact must lie in each other's proximity.

As the notion of proximity depends on the size of the robot, we must first determine a convenient way of expressing this size. The fact that the robot may be articulated implies that its diameter is variable, so a more general notion is needed. Let $O \in \mathcal{B}$ be the robot's reference point. The reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$ is defined to be the maximum distance from the reference point $O \in \mathcal{B}$ to any point in $\mathcal{B}$ in any placement $Z$ of $\mathcal{B}$. More formally:

**Definition 4.1 [reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$]**
*Let $Z_{\mathrm{W}}$ be some arbitrary position of the reference point $O$ of the robot $\mathcal{B}$. Then the reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is defined as*

$$\rho_{\mathcal{B}} = \sup_{Z_D \in D} \max_{p \in \mathcal{B}[(Z_{\mathrm{W}}, Z_D)]} d(p, Z_{\mathrm{W}}).$$

In words, the reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$ is the maximum distance in the workspace that any point in the robot $\mathcal{B}$ can ever have to the reference point, which is also equal to how far the robot can reach, measured from its reference point. Naturally, the reach is independent of the actual position of the reference point. The definition

the obstacles probably makes the construction impossible; the worst setting that is then achievable seems to be the setting of Figure 4.3, yielding quadratic size free space. Halperin and Sharir [44] present an upper bound of $O(n^2 2^{\mathcal{O}(\log^{2/3} n)})$ on the complexity of a single connected component of the free space. This bound is almost tight as a quadratic lower bound construction is given by Figure 4.4 if we apply a minor modification to the robot to turn it into a non-convex shape (see also [41]).

Finally, we briefly consider a translating polyhedral robot ($f = 3$) among polyhedral obstacles in a three-dimensional workspace, with a trivial upper bound of $O(n^3)$ on the complexity of the free space. A rather straightforward generalization ([89]) of the construction of Figure 4.3 shows that the cubic complexity can indeed by achieved if the robot is non-convex. Figure 4.6 shows the three-dimensional con-

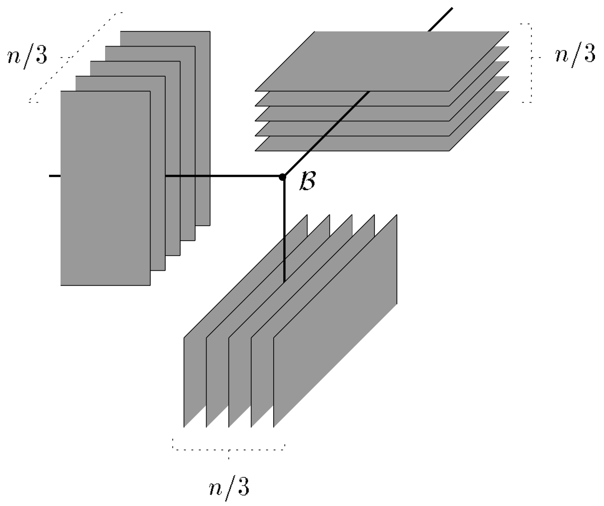


Figure 4.6: A spatial translational motion planning problem for a non-convex robot with free space complexity $\Omega(n^3)$.

struction; the orientations of the three sets of parallel planes restrict the position of the meeting point of the three bars of $\mathcal{B}$ to a small cube, similar to the planar example. The construction becomes impossible upon addition of a minimal fatness requirement for the obstacles, for reasons similar to those of the corresponding planar case. Other results on the complexity of the free space are restricted to convex robots. Results by Wiernik and Sharir [101] imply the existence of a lower bound construction of size $\Omega(n^2\alpha(n))$ for a translating convex polyhedral robot among polyhedra, based on Davenport-Schinzel sequences. Halperin and Yap [46] prove an upper bound of $O(n^2\alpha(n))$ for the free space complexity of a translating box, confirming the more general conjecture of Sharir [89] that the same upper bound holds for any convex polyhedral robot. Recently, Aronov and Sharir [7] have shown that the worst-case complexity of FP for a convex polyhedral robot amidst polyhedral obstacles is $O(n^2 \log^2 n)$. Halperin and Sharir [45], finally, prove an upper bound on

are connected by (semi-)free paths and belong therefore to the boundary of a single connected component of the free space. Thus, the complexity of this free cell, and obviously also of the entire free space, is $\Omega(n^2)$. Like in the purely translational case, it is possible to build the construction if the obstacles have a certain minimal fatness, simply by making the robot sufficiently large. A restriction on the relative sizes of the robot and the obstacles, however, seems to make the construction impossible. Once more, the combination of both assumptions implies a linear upper bound on the complexity of FP, as will be shown in Section 4.3.

Contrary to the case of a convex robot, the trivial upper bound of $O(n^3)$ on 3-DOF motion planning can be achieved for some construction involving a non-convex robot. Figure 4.5 shows an L-shaped robot moving among two horizontal rows of $n/3$ obstacles and one vertical row of $n/3$ obstacles. The example is taken from a paper by Halperin, Overmars, and Sharir [43] on motion planning for an L-shaped robot. For appropriately chosen distances between the three rows and between consecutive

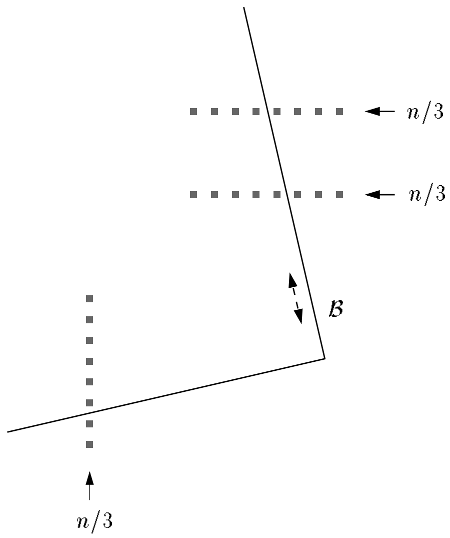


Figure 4.5: A planar motion planning problem for a non-convex robot with free space complexity $\Omega(n^3)$.

obstacles within a single row, it can easily be verified that it is possible to place one bar between any combination of pairs of consecutive obstacles in the upper *and* lower horizontal rows and the other bar between any pair of consecutive obstacles in the vertical row. Once the bars are placed between consecutive pairs of obstacles in each of the three rows, the robot is 'stuck' between these pairs: it is not connected by a free path to a placement of the bars between different obstacle pairs. As the number of combinations is about $n^3/27$, the free space consists of $\Omega(n^3)$ free cells and has complexity $\Omega(n^3)$. Like in the previous examples, the construction can be built for fat obstacles as well. A restriction on the relative sizes of the robot and

the above construction can be obtained by choosing the robot sufficiently large. More obstacles though necessitate a larger robot, so the size of the robot is proportional to the number of obstacles. This observation indicates that the combination of a minimal fatness requirement and a bound on the relative sizes of the robot and the obstacles make the construction impossible. The results in the next section confirm the supposition. For the sake of completeness, we mention that the complexity of a single free cell for a polygonal robot amidst polygonal obstacles is $O(n\alpha(n))$ [38], where $\alpha(n)$ is the extremely slowly growing inverse of the Ackermann function.

The complexity of the free space of a translating and rotating polygonal robot ($f = 3$) amidst polygonal obstacles is trivially bounded by $O(n^3)$. Leven and Sharir [63] report an $O(n\lambda_6(n))$ bound on the complexity if the robot is a convex polygon, where $\lambda_6(n)$ is a near-linear function depending on the length of certain so-called Davenport-Schinzel sequences. (For a discussion of Davenport-Schinzel sequences and more detailed bounds on $\lambda_s(n)$ for various values of $s$, the reader is referred to [4].) Ke and O'Rourke [50] show that $\Omega(n^2)$ moves, that is, constant complexity curves, may be necessary to connect two placements in a single free cell. This result gives an indication of the potential complexity of finding a path in a single cell; the cell complexity alone does not give full insight in this matter (finding a path between two points in a convex cell, for example, is simple, regardless of its complexity). Figure 4.4 gives an $\Omega(n^2)$ lower bound construction for the single cell complexity, and, hence, for the complexity of the entire free space that approximates the near-quadratic upper bound of $O(n\lambda_6(n))$, thus leaving a relatively small gap between the theoretical upper bound and achievable lower bound construction. The

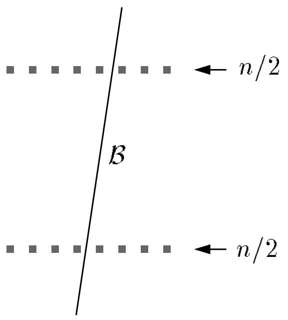


Figure 4.4: A planar motion planning problem for a convex robot with free space and single cell complexity $\Omega(n^2)$.

robot $\mathcal{B}$, which is a simple line segment moving among small (point or square) obstacles, can be in simultaneous contact with any combination of obstacles from the top and bottom row, yielding $n^2/4$ obstacle pairs. Any simultaneous contact with features of such a pair defines a one-dimensional face on the boundary of the free space, resulting in $\Omega(n^2)$ one-dimensional faces. All double contact placements

proximated by artificial settings. Below we review some of the known lower bound constructions and upper bounds on free space and single cell complexities for motion planning problems. The currently available results are restricted to problems with at most three-dimensional configuration spaces. We consider motion planning problems involving a constant-complexity robot amidst $n$ constant-complexity obstacles.

The trivial upper bound on the complexity of the free space of a translating polygonal robot ($f = 2$) amidst polygonal obstacles is $O(n^2)$. If the robot is convex, then the worst-case complexity of the free space remains an order of magnitude below the trivial bound (see e.g. [89]). The linear bound is clearly optimal. Things change if the robot is allowed to be non-convex. Figure 4.3 shows an L-shaped robot amidst $n/2$ vertical line segments close to each other and $n/2$ horizontal line segments arranged similarly on a horizontal line. If the robot's horizontal bar

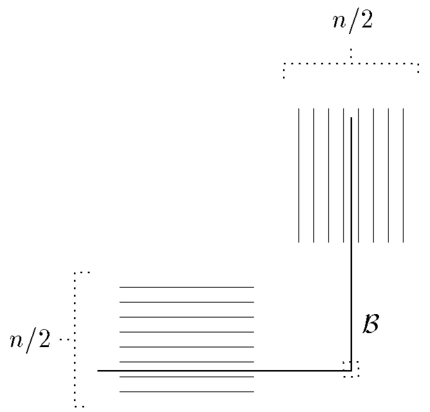


Figure 4.3: A planar translational motion planning problem with free space complexity $\Omega(n^2)$.

is placed between two consecutive horizontal line segments and its vertical bar is placed between two vertical segments, then both bars of the 'L' are stuck between these two pairs of segments and the reachable positions of the vertex incident to the two bars are restricted to a small (dotted) square of points. These position constitute a separate connected component of the free space. As there are $n^2/4$ combinations of a vertical and horizontal segment, the free space consists of $\Omega(n^2)$ free cells. Hence, the complexity of FP is $\Omega(n^2)$. Let us try to find out what specific properties of the construction in Figure 4.3 lead to the quadratic free space complexity. First of all, it turns out that a similar constructions can be obtained for any combination of sizes of the robot and the segments, simply by appropriately choosing the distance between any pair of consecutive parallel segments. Thus, a restriction on the relative sizes alone does not make the construction impossible. On the other hand, a minimal fatness restriction on the obstacles alone is also insufficient. In the case that the line segments are replaced by squares of equal size,

multiple contacts than (fat) obstacles that do not have such parts. Before we formalize these intuitions in the Section 4.3, we first review some known bounds on the complexities of the free space for motion planning problems.

## 4.2 Results on free space complexities

Research in motion planning throughout the last few years has concentrated on studying the structure and complexity of the free space rather than on developing new motion planning algorithms. This trend can be explained from the fact that a thorough insight in the structure and complexity of the free space is evidently very important for the design of algorithms that efficiently preprocess FP into a small structure capable of providing fast answers to motion planning queries.

Besides studying the entire free space, several papers also focus on the complexity and computation of a single connected component, or single cell, of the free space. The motivation for this direction of research lies in the simple observation that a robot can only reach placements that lie in the free space component containing the initial robot placement. The complexity of a single free cell is sometimes an order of magnitude smaller than the complexity of the entire free space.

In the preceding section, we have seen that the complexity of an arrangement of $O(n)$ constraint hypersurfaces in $f$-dimensional configuration space is bounded by $O(n^f)$, by standard arguments on arrangements of algebraic hypersurfaces of bounded degree. For certain motion planning problems, that is, for certain robots and obstacle types, the shapes of the constraint hypersurfaces are such that the arrangement of hypersurfaces, and, hence, the complexity of the free space, has a worst-case complexity smaller than $O(n^f)$. Given the upper bounds obtained by combinatorial arguments, it is interesting to see if examples of motion planning environments, that is, a robot in a workspace with obstacles, can be constructed that do indeed achieve these worst-case free space complexities. When trying to do so, it turns out that it is often difficult to construct settings of robots in workspaces with obstacles that establish or even approximate the upper bound on the free space complexity (or single cell complexity), thus leaving a gap between theoretical worst-case bounds obtained by combinatorial arguments and complexities obtained by constructed difficult (complex) motion planning environments. The existence of such a gap clearly raises uncertainty on the tightness of the upper bound.

The lower bound constructions of difficult motion planning environments found in literature are often very artificial and as such rarely encountered in real-life situations: they often involve robots and obstacles that are extremely thin and/or have exorbitant relative sizes. If the extreme properties of the robot and the obstacles that are necessary to construct these difficult settings do not occur, then most of the artificial lower bound constructions become impossible. This illustrates that the complexity of the free space for practical motion planning problems is likely to stay far below the worst-case complexities obtained by combinatorial arguments and ap-

self-intersections of the robot. Robot-robot collisions can be dealt with in exactly the same way as the robot-obstacle collisions, so that we end up with a number of self-collision constraint hypersurfaces that separate placements in which parts of the robot intersect each other from placements in which the robot does not self-intersect. A self-collision constraint hypersurface is induced by the contact of a robot feature with another robot feature of appropriate dimension. The self-collision constraint hypersurfaces in the linked 4-DOF robot example are the two three-dimensional surfaces $\alpha = \beta$ and $\alpha = \beta + \Delta$. The arrangement of all 'regular' and self-collision constraint hypersurfaces subdivides the $f$-dimensional configuration space in free and forbidden cells w.r.t. the obstacles *and* the robot itself. Below we search for conditions for the self-collisions that prevent an increase of the complexity of the arrangement and, hence, of the complexity of the free space.

The shape of the self-collision constraint hypersurfaces differs somewhat from the shape of the regular constraint hypersurfaces in the sense that they are 'larger'. The bounded range of reference point positions in the workspace in which a specific robot feature touches a specific obstacle feature is reflected in a certain compactness of the corresponding constraint hypersurface in configuration space. On the contrary, a collision of two robot features is independent of the position of the robot's reference point so that the corresponding constraint hypersurface can be unboundedly large (see for example the surfaces of the above example). The size of the self-collision constraint hypersurfaces causes such surfaces to (possibly) intersect all other surfaces. If, however, the number of self-collision constraint hypersurfaces is constant and each hypersurface is algebraic of bounded degree, then certainly these additional surfaces will not increase the asymptotic complexity of the arrangement of constraint hypersurfaces. Because of our general assumption that the robot is of constant complexity, this is always the case. Hence, in bounding the free space complexity, we may neglect self-collisions. We briefly revisit self-collisions in Chapter 6 to examine their algorithmic consequences.

Unfortunately, the worst-case number of multiple contacts, and, hence, the complexity of the free space, can be high. If $n$ is the number of obstacle features and $f$ is the number of degrees of freedom of the robot (i.e., the dimension of the configuration space) and the number of robot features is bounded by some constant, then this complexity can be $\Omega(n^f)$. So, theoretically, motion planning techniques whose performances depend on the size of the free space are expensive. Fortunately, in many practical situations the complexity of the free space FP tends to be much smaller and, as a result, such methods might become feasible. A study of properties that limit the number of multiple contacts for the robot (and consequently the complexity of FP) is therefore of obvious importance.

In many practical cases the relative positions and the shapes of the obstacles are such that the number of multiple contacts for the robot $\mathcal{B}$ is very low. Obstacles that lie far apart clearly result in less multiple contacts for $\mathcal{B}$ than obstacles that are cluttered. Similarly, obstacles that have long and skinny parts will induce more

two different non-parallel edges now defines a curve in the configuration space: the intersection of the two constraint surfaces corresponding to both contacts of the endpoints. The fact that the double contact defines a curve can be understood by the observation that it is possible to slide both robot endpoints along the respective obstacle edges that they touch, thus maintaining the double contact. The continuously changing robot placements lie on a curve in configuration space. An additional third contact for the ladder robot, for example when its interior touches an obstacle vertex fixes the position of the robot in the sense that is unable to move without losing at least one of the three contacts. Triple contacts therefore occur at isolated points in configuration space; they correspond to intersections of three constraint surfaces.

In the preceding paragraphs we have implicitly assumed that the robot can only collide with the obstacles and not with itself. In other words, we have assumed that no part of the robot can collide with another part of the robot. Although the absence of such so-called self-collisions (or self-intersections) is a common assumption in motion planning, we choose to give some thought to the possible consequences when self-collisions are not neglected. Self-collisions clearly only occur when the robot under consideration is not a single rigid body but instead consists of a number of such bodies linked together by revolute or prismatic joints. A specific property of self-collisions is that they depend solely on the relative positions of the robot parts; the location of the robot in the workspace is irrelevant for determining if a certain robot placement causes self-collision. Figure 4.2 illustrates the observation. Consider the

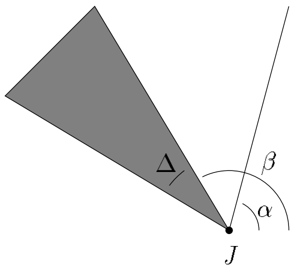


Figure 4.2: An example of a self-colliding (4-DOF) robot.

robot consisting of a triangle and a line segment attached to each other by a revolute joint $J$; the triangle edges incident to $J$ define an angle $\Delta$. The joint $J$ serves also as the robot's reference point. A placement of this 4-DOF robot is specified by the position $(x, y)$ of the reference point in the workspace $\mathrm{W} = \mathbb{R}^2$, and by the angles $\alpha$ and $\beta$ between the positive $x$-axis and the line segment and the triangle respectively. Clearly, the two linked parts of the robot intersect in any placement $(x, y, \alpha, \beta)$ satisfying $\beta \leq \alpha \leq \beta + \Delta \pmod{2\pi}$. Hence, the entire subspace $\mathbb{R}^2 \times \{(\alpha, \beta) \in [0, 2\pi) \times [0, 2\pi) | \beta \leq \alpha \leq \beta + \Delta \pmod{2\pi}\}$ consists of forbidden placements due to

shaped. A direct consequence of the assumption is that the intersection of any multiple of hypersurfaces consists of only a constant number of connected components. Moreover, it implies a first simple upper bound of $O(n^f)$ on the complexity of the entire arrangement of constraint hypersurfaces.

The complexity of a cell in an arrangement of algebraic hypersurfaces of bounded degree is defined to be the number of faces of various dimensions on the cell's boundary. A $j$-dimensional face, or $j$-face is a maximal connected $j$-dimensional part of the arrangement containing no lower-dimensional faces in its interior. A $j$-dimensional face of the specific arrangement of constraint hypersurfaces is a maximal connected component of the intersection (or common boundary) of $f-j$ constraint hypersurfaces. For example, the complexity of a two-dimensional cell in an arrangement of line segments in the plane is the number of edges (1-faces) and vertices (0-faces) on the cell's boundary, where a vertex is either an endpoint of a line segment or the intersection point of segments, and an edge is a maximal portion of a line segment meeting no vertex of the arrangement.

The complexity of the free space is the sum of the complexities of the free cells and, hence, bounded by the complexity of the entire arrangement of constraint hypersurfaces, that is, the total number of faces of any dimension in the arrangement. As each constraint hypersurface is induced by a contact of a robot feature and an obstacle feature, the intersection of $j$ such surfaces corresponds to the simultaneous occurrence of $j$ contacts for the robot. Because each intersection of $j$ hypersurfaces consists of a constant number of connected components by the assumption on the shape and complexity of the hypersurfaces, the number of $j$-fold contacts is of the same order of magnitude as the number of $j$-dimensional faces in the arrangement. Thus, the complexity of the free space is determined by the total number of different single and multiple contacts for the robot, since they determine the complexity of the arrangement of constraint hypersurfaces (which bounds the complexity of FP).

To get a feeling of what a multiple contact is, consider the case of a ladder (line segment) translating among polygonal obstacles in the plane. This is a motion planning problem with two degrees of freedom and the constraint curves that it induces in the configuration space $C = \mathbb{R}^2$ are straight line segments. Each of these constraint segments is induced either by the contact of a ladder endpoint with an obstacle edge, or by the contact of the interior of the ladder with an obstacle vertex. Consider now the case where each ladder endpoint touches a distinct obstacle edge (and assume further that these two edges have different directions). The contact of each ladder endpoint with an obstacle edge is expressed as a segment in the configuration space, and this double contact will manifest itself as the meeting point of these two segments, namely as a vertex in the configuration space. The fact that the double contact occurs at an isolated point in configuration space can be easily understood by observing that it is impossible to maintain the double contact while slightly moving the ladder. If the ladder is also allowed to rotate then the single contacts mentioned above define constraint surfaces in the configuration space $\mathbb{R}^2 \times [0, 2\pi)$. The double contact of the robot in which its two endpoints touch

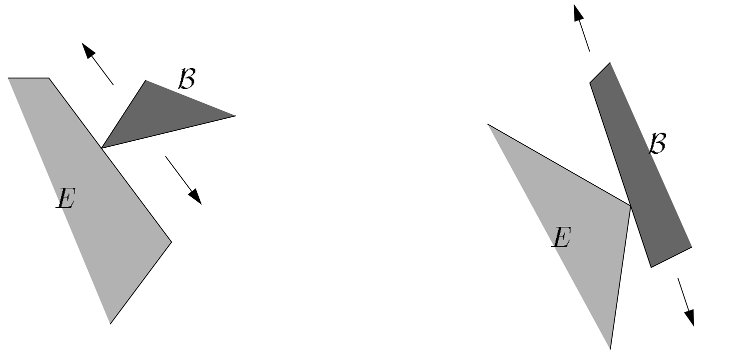


Figure 4.1: The two types of contacts inducing a constraint curve in the configuration space of a polygonal robot translating amidst polygonal obstacles: a robot vertex sliding along an obstacle edge, and a robot edge sliding along an obstacle vertex.

obstacles, each constraint surface is induced by the contact of either a robot vertex and an obstacle face, or a robot edge and an obstacle edge, or a robot face and an obstacle vertex.

The constraint hypersurfaces in configuration space allow for an interpretation of the free space with a more computational-geometry-like flavor. The $(f-1)$-dimensional constraint hypersurfaces partition the $f$-dimensional configuration space into $f$-dimensional cells. A cell is a maximal connected $f$-dimensional subset of the configuration space containing no part of a constraint hypersurface. The cells consist either exclusively of free placements of exclusively of forbidden placements. The cells are referred to as free cells and forbidden cells respectively. The free cells in the arrangement of constraint hypersurfaces collectively constitute the free space FP. We are therefore interested in studying a collection of cells, namely, the free cells, in the partitioning of $f$-dimensional space by a collection of $(f-1)$-dimensional hypersurfaces. Note that, by the definition of a cell, no two points in two different free cells are linked by a free path, that is, a path that is entirely contained in the free space. The definition of the free space via the arrangement of constraint hypersurfaces links the study of the motion planning problem to a basic study in computational geometry, namely, the study of arrangements of hypersurfaces.

Before we define the complexity of a cell in an arrangement, we first formulate an assumption regarding the constraint hypersurfaces in configuration space. The assumption stands throughout all of the remaining chapters.

> The hypersurface in configuration space corresponding to the set of placements in which a certain robot feature is in contact with a certain obstacle feature of appropriate dimension is algebraic of bounded degree.

The assumption on the shape and complexity of the constraint hypersurfaces mainly means that the boundaries of the robot and the obstacles are not too irregularly

$Z$. Furthermore, we denote the interior of a closed set $X$ by $int(X)$. Assume that $Z \in C$, then

- $Z$ is a *free placement* when $\forall_{E \in \mathcal{E}} \mathcal{B}[Z] \cap E = \emptyset$,
- $Z$ is a *contact placement* when $\exists_{E \in \mathcal{E}} \mathcal{B}[Z] \cap E \neq \emptyset$ and $\forall_{E \in \mathcal{E}} \mathcal{B}[Z] \cap int(E) = \emptyset$,
- $Z$ is a *forbidden placement* when $\exists_{E \in \mathcal{E}} \mathcal{B}[Z] \cap int(E) \neq \emptyset$.

Informally, a free placement is a placement in which $\mathcal{B}$ does not intersect any obstacle, a contact placement is a placement in which $\mathcal{B}$ is in contact with the boundary of some obstacle but does not intersect the interior of any obstacle, and a forbidden placement is a placement in which $\mathcal{B}$ intersects the interior of some obstacle. Clearly, any point $Z \in C$ satisfies exactly one of the three expressions and corresponds therefore to either a free placement, or a contact placement, or a forbidden placement.

The subset of configuration space of all free placements can, according to the above classification, be obtained by subtracting the union of all sets $C_E = \{Z \in C | \mathcal{B}[Z] \cap E \neq \emptyset\}$ with $E \in \mathcal{E}$ from $C$: FP $= C \setminus \cup_{E \in \mathcal{E}} C_E$. A set $C_E$ is sometimes (see e.g. [59]) referred to as a configuration space obstacle (of $E$), as it consists of all placements of $\mathcal{B}$ in which it intersects $E$. The boundary $\partial C_E$ consists of robot placements $Z$ such that $\mathcal{B}[Z] \cap E \neq \emptyset$ and $\mathcal{B}[Z] \cap int(E) = \emptyset$, or, in other words, of all placements in which $\mathcal{B}$ touches $E$. The set of placements $\partial C_E$ in which $\mathcal{B}$ touches $E$ separates the placements of $C_E$ in which $\mathcal{B}$ intersects the interior of $E$ from the placements of $C \setminus C_E$ in which $\mathcal{B}$ does not intersect $E$. On a more global level, we find that the union boundary $\partial(\cup_{E \in \mathcal{E}} C_E)$ separates the forbidden placements from the free placements. The union boundary equals exactly the set of all contact placements. (Notice that the semi-free space SFP defined in Chapter 1 consists of all free placements *and* all contact placements.)

The boundary $\partial C_E$ of a configuration space obstacle $C_E$ in the $f$-dimensional configuration space can be regarded as a collection of $(f-1)$-dimensional hypersurfaces consisting of contact placements of a single robot feature and a single obstacle feature of appropriate dimension. We use the term feature to describe a basic part of the boundary of a geometric object whether an obstacle or the robot. An $(f-1)$-dimensional hypersurface of contact placements is called a constraint hypersurface. The lower-dimensional features on the boundary of a configuration space obstacle are common boundaries or intersections of two or more constraint hypersurfaces. For example, in the two-dimensional configuration space $C = \mathbb{R}^2$ of a translating polygonal robot amidst polygonal obstacles, each constraint curve is induced either by the contact of a robot vertex with an obstacle edge or by the contact of a robot edge with an obstacle vertex. Figure 4.1 shows both types of contact. If the robot is allowed to rotate as well, then both combinations define constraint surfaces in the yet three-dimensional configuration space $C = \mathbb{R}^2 \times [0, 2\pi)$. In the three-dimensional configuration space $C = \mathbb{R}^3$ of a translating polyhedral robot amidst polyhedral

# Chapter 4

# The complexity of the free space

In this chapter, we return to the motion planning problem and start the investigation on how fatness influences the problem and its solution. The problem of finding a collision-free motion for a robot in a workspace with obstacles is commonly transformed into the problem of finding a continuous curve in the free space. In Chapter 1, we have argued that the complexity of finding such a curve highly depends on the complexity of the free space. This chapter studies the influence of fatness on the free space complexity. More specifically, it shows that the complexity of FP for a constant-complexity robot moving amidst constant-complexity fat obstacles is linear in the number of obstacles, provided that the size of the robot is proportional to the size of the smallest obstacle and provided that the constraint hypersurfaces defined by the robot-obstacle contacts are algebraic of bounded degree.

Section 4.1 studies the structure of the free space in detail and establishes the relation between the complexity of the free space and the number of multiple contacts for the robot with the obstacles in the workspace. Section 4.2 gives an overview of known results on free space complexities. The overview gives an idea of the conditions that typically lead to high complexities. The observations are used in Section 4.3 to formulate a realistic framework of motion planning problems with a linear complexity free space.

## 4.1 The structure of the free space

The configuration space $C$ is the space of parametric representations of all robot placements. The number of degrees of freedom $f$ of the robot $\mathcal{B}$ determines the dimension of the configuration space. We classify the points in the configuration space according to the robot placements that they represent, resulting in three different types of points[1]. Let $Z \in C$ be a placement of the robot $\mathcal{B}$, and let $\mathcal{B}[Z] \subset \mathrm{W}$ be the collection of points in the workspace covered by $\mathcal{B}$ when placed at

[1]In the sequel, we generally do not distinguish between the point $Z \in C$ and the placement that $Z$ represents.

In the generalized setting of intersecting objects, a query point may be contained in more than one object. The answer to a point location, or point stabbing, query should therefore be the collection of objects containing the query point. Note that the new restriction on the data set $\mathcal{E}$ prevents more than a constant number of simultaneous containments. An interesting example of such a set is a collection $\mathcal{E}_w$ of $(c \cdot \rho)$-wrappings of non-intersecting $k$-fat objects. If the wrappings $E \in \mathcal{E}_w$ are $k$-fat for some constant $k$ and convex or polytopes, then Theorem 3.18 applies to the set $\mathcal{E}_w$ of intersecting objects.

The results of this chapter have an important application in the motion planning part of this thesis. The running time of the general paradigm for motion planning amidst fat obstacles in Chapter 6 depends on the time spent in computing the pairs of obstacles within a distance $b \cdot \rho$ from each other, where $b$ is a constant and $\rho$ is a lower bound on the minimal enclosing hypersphere radii of the obstacles in the workspace. By Corollary 2.10 there are only $O(n)$ such pairs. The result in this chapter allow for the computation of all pairs in time $O(n \log^{d-1} \log \log n)$ time instead of the trivial $O(n^2)$ time. As a related application, it is possible to compute the linear complexity arrangement (by Theorem 2.12) of tight wrappings of non-intersecting fat objects in time $O(n \log^{d-1} \log \log n)$.

**Lemma 3.16** *Let, for all $n-m < j \leq n$, $V_j$ be a set of convex objects or polytopes. Then the computation of the set $V_{n-m}$ from $T_m$ and $\{V_j | n-m < j \leq n\}$ takes $O(\log^{d-1} n \log\log n)$ time.*

Lemmas 3.15 and 3.16 show that each of the $O(n)$ steps in the incremental construction of the $d$-level data structure for point location and range searching among convex objects or polytopes takes $O(\log^{d-1} n \log\log n)$ time, resulting in a time bound of $O(n \log^{d-1} n \log\log n)$ for the computation of $T_n$ from the skeleton $T_0$. Adding to this bound the $O(n \log^{d-1} n)$ time bound for building the $(d-1)$-level segment tree $T_0$, we obtain the desired result.

**Theorem 3.17** *Let $\mathcal{E}$ be a set of non-intersecting constant-complexity k-fat convex objects or arbitrary polytopes. Then the d-level point stabbing structure can be built in time $O(n \log^{d-1} n \log\log n)$.*

After the construction of the data structure, the query time can be improved back to $O(\log^{d-1} n)$. To this end, it suffices to rebuild the structure using static fractional cascading. As all the sets $V_i$ are now known, this can easily be achieved in time $O(n \log^{d-1} n)$.

## 3.4 Summary of results and extensions

In this chapter we have presented a data structure for both point location and range searching with bounded-size ranges in certain scenes of fat objects. Theorem 3.18 summarizes the results by combining Theorems 3.3, 3.10, 3.13, and 3.17.

**Theorem 3.18** *Let $k > 0$ and $h \geq 0$ be constants and let $\mathcal{E}$ be a set of non-intersecting constant-complexity k-fat arbitrary convex objects and/or non-convex polytopes $E \subseteq \mathbb{R}^d$ with minimal enclosing hypersphere radii at least $\rho$. Then the set $\mathcal{E}$ can be stored in time $O(n \log^{d-1} n \log\log n)$ in a data structure of size $O(n \log^{d-1} n)$ which supports point location queries and range searching queries with ranges $R \subseteq \mathbb{R}^d$ of diameter at most $h \cdot \rho$ among the objects of $\mathcal{E}$ in time $O(\log^{d-1} n)$.*

The theorem does not apply to scenes of arbitrarily-shaped non-convex objects. It is though believed that a similar result holds in that case as well. Preliminary results in that direction with two-dimensional objects bounded by algebraic polygonal curves of bounded degree are promising.

Throughout the entire chapter, the assumption that the objects in $\mathcal{E}$ are non-intersecting only plays a role in showing that the number of larger objects $E'$ intersecting the enclosing hypercube $C$ of some object $E$ is bounded by a constant. No other lemma or theorem relies on the disjointness of the objects. As a consequence, all results in this chapter remain valid if we drop the requirement of disjointness and instead impose the weaker restriction upon $\mathcal{E}$ that each enclosing hypercube $C$ of $E \in \mathcal{E}$ is intersected by at most a constant number of objects $E'$ larger than $E$.

### Computation of $V_{n-m}$

The computation of $V_{n-m} = \{E_j \in \mathcal{E} | E_j \cap C_{n-m} \neq \emptyset \wedge j \geq n-m\}$ is based on a sequence of point location queries. In the static point location structure, the query time was found to be $O(\log^{d-1} n)$ due to the incorporation of fractional cascading. Throughout the incremental construction of the point location structure, however, we use dynamic fractional cascading instead of fractional cascading to achieve efficient insertions, which leads to a query time of $O(\log^{d-1} n \log\log n)$. Let us analyze a point stabbing query in the intermediate structure $T_m$.

Like in the static case described in Section 3.1, a query with a point $p$ in the intermediate structure $T_m$ proceeds recursively in the substructure associated to the $O(\log n)$ nodes on the search path from root to leaf. This eventually leads to a search of $O(\log^{d-2} n)$ substructures at level $d-1$, the level where dynamic fractional cascading is incorporated. The upper level of such a substructure $T$ is a segment tree on the projections onto the second coordinate axis of the locally stored hypercubes. The query point $p$ will therefore again be contained in the slabs corresponding to the nodes $\nu$ on the search path from the root to the leaf of the elementary interval containing $p$'s projection. Hence, we must search all lists $L(\nu)$ of nodes $\nu$ on the search path. Fortunately, the nodes on the path form a connected subtree $T'$ of $T$ with $O(\log n)$ nodes, so, by Theorem 3.14(a) these nodes can be searched simultaneously in worst-case time $O(\log n \log\log n)$. The entire point stabbing query time amounts to $O(\log^{d-1} n \log\log n)$. Note that a single search with $p$ yields $O(\log^{d-1} n)$ candidate answers: one for each list $L(\nu)$ that is searched. The ultimate answer to the query is clearly the minimum among all hypercube indices found.

The computation of the set $V_{n-m}$ benefits from the fact that the hypercubes are inserted into the data structure from large to small in the sense that at the time of the set's computation, the intermediate data structure only stores hypercubes and objects from the appropriate index range $[n-m+1,\ldots,n]$. Therefore, we may restrict ourselves to finding the hypercubes in the data structure that intersect the query hypercube $C_{n-m}$ without having to bother about the sizes of these hypercubes. Moreover, note that future additions of hypercubes and their corresponding objects do not affect the earlier computed sets $V_j$. To apply the range searching results from Section 3.2, we must verify the validity of the constant ratio between the diameter of the search region (the hypercube $C_{n-m}$) and the lower bound on the radii of the minimal enclosing hyperspheres of the stored objects. The radii of the minimal enclosing hyperspheres of the objects in $\{E_{n-m+1},\ldots,E_n\}$ are bounded from below by $\rho_{n-m+1}$, and, by the ordering on the radii, also by $\rho_{n-m}$. The query region $C_{n-m}$ is the axis-parallel enclosing hypercube of the minimal enclosing hypersphere of $E_{n-m}$ with radius $\rho_{n-m}$. As a result, the diameter of $C_{n-m}$ is $2\sqrt{d}\cdot\rho_{n-m}$. The application of Theorems 3.10 and 3.13, yields, taking into account the modified point location query time due to dynamic fractional cascading, a worst-case time bound of $O(\log^{d-1} n \log\log n)$ for the computation of $V_{n-m}$.

intervals that have non-empty intersection with $[i_1^L, i_1^H]$ and are present in $L(\nu)$ upon insertion of the latter interval. Using the pointer into $L(\nu)$ we can identify the subsequence of intervals that have non-empty intersection with the interval $[i_1^L, i_1^H]$ in time proportional to its length. Let $[\alpha_1, \beta_1], \ldots, [\alpha_g, \beta_g]$ be this sequence and note that only $[\alpha_1, \beta_1]$ may contain $i_1^L$ and only $[\alpha_g, \beta_g]$ may contain $i_1^H$. The update of $L(\nu)$ proceeds in four simple steps, in which we scan all intervals intersected by $[i_1^L, i_1^H]$:

1. **if** $i_1^L \in (\alpha_1, \beta_1]$ **then** replace $[\alpha_1, \beta_1]$ by $[\alpha_1, i_1^L]$
   **else delete** $[\alpha_1, \beta_1]$;
2. **for all** $2 \leq h \leq g - 1$ **do delete** $[\alpha_h, \beta_h]$;
3. **if** $i_1^H \in [\alpha_g, \beta_g)$ **then** replace $[\alpha_g, \beta_g]$ by $[i_1^H, \beta_g]$
   **else delete** $[\alpha_g, \beta_g]$;
4. **insert** $[i_1^L, i_1^H]$.

As $l = O(n \log n)$ and $t = O(n)$, the amortized time spent on each of the above deletions or insertions is, by Theorem 3.14(b),(c), $O(\log \log n)$. The four steps include one insertion and $g$ deletions in $L(\nu)$. The fact that some varying number of $g$ deletions take place during a single update of a list $L(\nu)$ is not a problem, due to the observation that each deletion must follow an earlier insertion of the same interval. Hence, at any time during the preprocessing, the number of insertions so far exceeds the number of deletions. So, the amortized number of deletions per list update is one as well, which implies that the amortized time spent in updating a single list with a new hypercube is $O(\log \log n)$. Within each level-$(d-1)$ substructure $T$, the required associated list updates are restricted to a subset of the $O(\log n)$ nodes of the subtree $T'$, so the time spent on all necessary list updates in $T$ is $O(\log n \log \log n)$. Combined with the $O(\log n \log \log n)$ time bound for the simultaneous search in all lists $L(\nu')$ with $\nu'$ in $T'$ for the locations of the value $i_1^L$, this implies that the total time spent on the update of a single level-$(d-1)$ substructure is $O(\log n \log \log n)$. Since the total number of level-$(d-1)$ substructures that have to be updated is bounded by $O(\log^{d-2} n)$, the insertion of $C_{n-m}$ into $T_m$ to obtain $T_{m+1}$ takes time $O(\log^{d-1} n \log \log n)$.

**Lemma 3.15** *The insertion of the hypercube $C_{n-m}$ into $T_m$ to obtain $T_{m+1}$ requires $O(\log^{d-1} n \log \log n)$ amortized time.*

The computation of the structure $T_{m+1}$ by inserting the hypercube $C_{n-m}$ into the intermediate structure $T_m$ is independent of the actual shape of the objects under consideration. The efficiency of this part is therefore guaranteed, irrespective of the object shape. The efficiency of the computation of $V_{n-m}$, however, relies on the fact that the objects under consideration are convex or polytopes. The dependence follows from the use of the results from Section 3.2.

We move on to study the reduced problem of inserting the planar projection $[i_1^L, i_1^H] \times [i_2^L, i_2^H]$ of $C_{n-m}$ into a level-$(d-1)$ substructure $T$ of $T_m$. The upper level of $T$ is a segment tree on the projections onto the second coordinate axis of all, at most $n$, planar hypercube projections stored in $T$. The associated (ordered list) structure $L(\nu)$ of a node $\nu$ of $T$ stores the sequence (from $x_1 = -\infty$ to $x_1 = +\infty$) of disjoint slab intervals $[\alpha, \beta] \times I_\nu$ within which all points share the same smallest containing hypercube. The intervals are labeled with the respective hypercube index.

The hypercube $C_{n-m}$ must be stored in the ordered list substructures $L(\nu)$ at nodes $\nu$ of $T$ corresponding to slabs $\mathbb{R} \times I_\nu$ that are entirely spanned by the projection $[i_1^L, i_1^H] \times [i_2^L, i_2^H]$ of $C_{n-m}$ onto the plane spanned by the first two coordinate axes, but with parents $parent(\nu)$ corresponding to slabs $\mathbb{R} \times I_{parent(\nu)}$ that are not spanned by $[i_1^L, i_1^H] \times [i_2^L, i_2^H]$. Alternatively phrased, the hypercube $C_{n-m}$ must be stored in the ordered lists $L(\nu)$ at nodes $\nu$ corresponding to intervals $I_\nu$ that are entirely spanned by $[i_2^L, i_2^H]$ and have parents $parent(\nu)$ corresponding to intervals $I_{parent(\nu)}$ that are not spanned by $[i_2^L, i_2^H]$. Although the resulting nodes do not form a connected subtree of $T$ they are in fact never more than one node off the search path from root to leaf in $T$ for either the endpoint $i_2^L$ or the endpoint $i_2^R$. Hence, we can apply Theorem 3.14(a) to the connected subtree $T'$ of $T$ consisting of all nodes on and just off both search paths and therefore efficiently search all $L(\nu')$ for $\nu'$ in $T'$ simultaneously for the location of the left endpoint $i_1^L$ of the projection of $C_{n-m}$ onto the first coordinate axis. The search time depends (see Theorem 3.14) on the number of nodes $(s)$ in the connected subtree $T'$, the cumulative length $(l)$ of all associated ordered lists in $T$, and the number of nodes $(t)$ in $T$. First of all, the tree $T$ is a priorly built tree on a subset of the projections of all hypercubes $C_1, \ldots, C_n$, so $t = O(n)$. The subtree $T'$ consists of two root-leaf paths in $T$ plus all nodes that are only one node off these search paths, thus $s = O(\log n)$. Moreover, an ordered list $L(\nu)$ (before insertion of $C_{n-m}$) stores only projections of the $m$ hypercubes $C_{n-m+1}, \ldots, C_n$, so $|L(\nu)| = O(m)$. Because (a part of) each projection appears at no more than two nodes at a single height-level in $T$, we have $l = \sum_\nu |L(\nu)| = O(m \log m) = O(n \log n)$. Application of Theorem 3.14(a) to the subtree $T'$ of $T$ and the query value $i_1^L$ yields that the location of the query value is identified in all lists $L(\nu')$ for $\nu'$ in $T'$ in time $O(\log n \log\log n)$ worst-case. Note that the set of nodes in $T'$ is a superset of the set of nodes in whose associated substructures $C_{n-m}$ must be inserted.

The problem that remains is to, given the interval $[i_1^L, i_1^H]$ and pointers to the location of $i_1^L$ in all lists $L(\nu)$ with $\nu$ in $T'$, insert the interval $[i_1^L, i_1^H]$ only into the lists $L(\nu)$ of nodes in which $C_{n-m}$ must be inserted, i.e., $[i_2^L, i_2^H]$ spans $I_\nu$ but not $I_{parent(\nu)}$. Let us consider a node $\nu$ in $T'$. Verifying whether $[i_1^L, i_1^H]$ must be inserted into $L(\nu)$ is easily done in constant time by comparing $[i_2^L, i_2^H]$ with $I_\nu$ and $I_{parent(\nu)}$. Assume that $[i_1^L, i_1^H]$ must indeed be inserted into $L(\nu)$, labeled with the index '$n - m$'. After insertion of the interval, a query with a point $p \in [i_1^L, i_1^H] \times I_\nu$ for the smallest, or lowest indexed, covering hypercube (projection) must obviously yield the answer '$n - m$'. Hence, the interval $[i_1^L, i_1^H]$ must overwrite all parts of

the preprocessing time to $O(n \log^{d-1} n \log\log n)$. The alternative requires only minor modifications. We give the main result from [68] in a formulation that is tailored to our applications.

**Theorem 3.14** *Let $T$ be a tree with $t$ nodes and let an ordered list $L(\nu)$ of elements from a given domain $D$ be associated to each node $\nu$. Furthermore, define $l$ to be the total length of all lists $L(\nu)$, so $l = \sum_\nu |L(\nu)|$.*

- **(a)** *Let $T'$ be a connected subtree of $T$ with $s$ nodes. Location of a query value $x$ in $L(\nu')$ for every $\nu' \in T'$, that is, finding the position in $L(\nu')$ of the smallest value larger or equal than $x$, takes $O(\log(l+t)+s\log\log(l+t))$ time worst-case.*
- **(b)** *The deletion of a value $x$ from a list $L(\nu)$ takes, given $x$'s position in $L(\nu)$, amortized time $O(\log\log(l+t))$.*
- **(c)** *The insertion of a value $x$ from a list $L(\nu)$ takes, given the position in $L(\nu)$ of the smallest value larger than $x$, amortized time $O(\log\log(l+t))$.*

To simplify the process of incrementally building the objective structure $T_n$, we observe that all hypercubes that are to be added throughout the construction process can be computed in advance. This observation facilitates a less complex semi-dynamic (instead of dynamic) preprocessing. The prior knowledge of all $n$ hypercubes means that the endpoints of the projections of the hypercubes on the $i$-th coordinate axis are from a fixed finite universe $U_i$ of size $O(n)$. We act, however, as if we only know the projections of the hypercubes on the last $d-1$ coordinate axes; hence each corner $(x_1, x_2, \ldots, x_d)$ is assumed to be a point from $\mathbb{R} \times U_2 \times \ldots \times U_d$. The static nature of the hypercubes with respect to the last $d-1$ axes is used to compute the major part of the $d$-level data structure in advance by recursively building a segment tree on the projections of the hypercubes onto the last $d-1$ coordinate axes. The resulting $(d-1)$-level segment tree, which can be regarded as the initial tree $T_0$ in our incremental construction, differs from our objective $d$-level data structure, $T_n$, only in that the one-dimensional ordered lists in the nodes of the substructures at level $d-1$ are missing. These lists are built incrementally by 'inserting' the hypercubes from large to small into the skeleton provided by the $(d-1)$-level segment tree. Note that the substructures at level $d-1$ represent decompositions into vertical slabs of the plane spanned by the second and first coordinate axes.

Let us now consider the intermediate structure $T_m$, obtained after inserting the largest $m$ hypercubes $C_{n-m+1}, \ldots, C_n$ into the skeleton $T_0$. Following the standard insertion procedure for multi-level segment trees, the insertion of the hypercube $C_{n-m}$ into $T_m$ boils down to the insertion of a square, that is, the projection of $C_{n-m}$ onto the plane spanned by the first two coordinate axes, into $O(\log^{d-2} n)$ substructures $T$ at level $d-1$. Note that the prior computation of the skeleton $T_0$ guarantees that no new nodes have to be created in any of the higher-level structures during the insertion of a hypercube.

If, besides $c$, the parameters $k$ and $h$ are also constant, then we get $|\Pi| = O(1)$, which induces polylogarithmic query time for range searching with bounded-size ranges among fat objects.

**Theorem 3.13** *Let $k > 0$ and $h \geq 0$ be constants, and let $\mathcal{E}$ be a set of $k$-fat constant-complexity polytopes $E \subseteq \mathbb{R}^d$ with minimal enclosing hyperspheres with radii at least $\rho$. A range searching query with a region $R \subseteq \mathbb{R}^d$ of diameter at most $h \cdot \rho$ among $\mathcal{E}$ takes $O(\log^{d-1} n)$ time.*

## 3.3 Building the data structure

The results of the previous section can be used for the incremental construction of the point location (and range searching) data structure. Let us assume we are given the $d$-level data structure for priority point stabbing queries among hypercubes from Section 3.1, storing the $m$ largest hypercubes $C_{n-m+1}, \ldots, C_n$, and the corresponding constant cardinality sets $V_{n-m+1}, \ldots, V_n$. We refer to this partial priority point stabbing structure as $T_m$. Hence, the objective is to eventually compute $T_n$ from some initial structure $T_0$. The outline of the incremental construction is as follows.

> compute $T_0$;
> $m := 0$;
> **while** $m < n$ **do**
> 1. compute $V_{n-m}$ by a range query with $C_{n-m}$
> (using $T_m$ and the sets $V_j$, $n - m < j \leq n$);
> 2. compute $T_{m+1}$ by inserting $C_{n-m}$ into $T_m$.

We study both steps in the loop in more detail, starting with the second step, as the implications of its solution influence the first step as well.

### Computation of $T_{m+1}$

The problem with the insertion of a hypercube into the $d$-level priority point stabbing structure lies in the use of fractional cascading, which was incorporated to improve the point location and range search query time from $O(\log^d n)$ to $O(\log^{d-1} n)$. Unfortunately, insertions into the multi-level data structure do not benefit from fractional cascading, so an insertion into the structure $T_m$ would require $O(\log^d n)$ time instead of $O(\log^{d-1} n)$. Moreover, a sequence of insertions into the multi-level data structure with the static fractional cascading part is likely to increase the time for a query back to $O(\log^d n)$ as the fractional cascading part no longer 'suits' the updated multi-level data structure. Building the data structure would, even with fractional cascading, require $O(n \log^d n)$ time. Fortunately, Mehlhorn and Näher describe in [68] a dynamic version of fractional cascading. Incorporation of dynamic fractional cascading in the data structure during the construction phase improves

us that the convex shape $E'$ contains an ellipsoid $L$ with

$$volume(L) \geq \frac{\omega_d \rho^d}{2^{c+2d-1} k d^d}. \tag{3.5}$$

As the ellipsoid $L$ lies entirely inside the hypercube $C_{m,\rho}$, the diameter of $L$ is bounded by $2\rho\sqrt{d}$. Lemma 3.6 now implies that $L$ contains a hypersphere $S$ with radius at least $2^{-(c+2d-1)}k^{-1}d^{-\frac{1}{2}(3d-1)}\rho$. Property 3.5, finally, shows that any such hypersphere $S$ is hit by at least one point from the regular orthogonal grid with resolution $2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho$. Hence, at least one point from $\mathcal{G}(2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho)$ hits $S \subseteq L \subseteq E' \subseteq E \cap C_{m,\rho}$. Notice that in the case of polytopes, unlike for convex objects, the required grid resolution depends on the complexity $c$ of the objects of $\mathcal{E}$.

By the considerations of the previous paragraphs, one of the grid points that hit any object $E \in Q(R)$ lies inside a hypercube $C_{m,\rho}$ with $m \in E \cap R$. This hypercube must therefore lie completely inside the Minkowski difference $R \ominus C_{O,\rho}$. As a consequence, the set $\Pi$ may be restricted to

$$\Pi = \mathcal{G}(2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho) \cap R \ominus C_{O,\rho}.$$

Lemma 3.12 summarizes the results obtained so far in this subsection.

**Lemma 3.12** *Let $\mathcal{E}$ be a set of $k$-fat polytopes $E \subseteq \mathbb{R}^d$ bounded by $c$ hyperplanar faces and with minimal enclosing hyperspheres with radii at least $\rho$. Furthermore, let $R \subseteq \mathbb{R}^d$ be a region with diameter $h \cdot \rho$, for some constant $h$. The set $Q(R)$ of objects $E \in \mathcal{E}$ intersecting $R$ can be found by point location queries with the points from $\mathcal{G}(2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho) \cap R \ominus S_{O,\rho}$.*

Similar to the convex case, the sequence of point location queries with all points in $\Pi$ takes $O(|\Pi| \log^{d-1} n)$ time and results in the set $\{E \in \mathcal{E} | \Pi \cap E \neq \emptyset\}$. The extraction of $Q(R)$ from this set takes $O(|\Pi|)$ time under the additional assumption that $R$ and all $E \in \mathcal{E}$ have constant complexity, so $c$ must be constant. To bound the number of elements in $\Pi$, we notice that $R \ominus C_{O,\rho}$ also fits completely in the hypercube $[x_0^R - \rho, x_0^R + (h+1)\rho] \times \ldots \times [x_d^R - \rho, x_d^R + (h+1)\rho]$, where $x_i^R$ $(1 \leq i \leq d)$ is once again the minimal $x_i$-coordinate in $R$. The number of grid points in the hypercube bounds the number of elements in $\Pi$:

$$\begin{aligned}
|\Pi| &= |\mathcal{G}(2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho) \cap (R \ominus C_{O,\rho})| \\
&\leq |\mathcal{G}(2^{-(c+2d-2)}k^{-1}d^{-\frac{3}{2}d}\rho) \\
&\quad \cap [x_0^R - \rho, x_0^R + (h+1)\rho] \times \ldots \times [y_d^R - \rho, y_d^R + (h+1)\rho]| \\
&= (2^{c+2d-2} k d^{\frac{3}{2}d}(\lfloor h \rfloor + 2))^d \\
&= O((2^c k d^d h)^d)
\end{aligned}$$

$volume(C_{m,\rho})/volume(S_{m,\rho}) = 2^d/\omega_d$. The ratio of the volumes, the inequality (3.1), and the obvious inequality $volume(E \cap C_{m,\rho}) \geq volume(E \cap S_{m,\rho})$, together yield

$$\frac{2^d k}{\omega_d} \cdot volume(E \cap C_{m,\rho}) \geq volume(C_{m,\rho}) = 2^d \rho^d. \tag{3.3}$$

The intersection $E \cap C_{m,\rho}$ is a collection of polytopes. The arrangement of the $c+2d$ supporting hyperplanes of the $c$ faces of $E$ and the $2d$ faces of the hypercube $C_{m,\rho}$ subdivide $\mathbb{R}^d$ and, more importantly, $E \cap C_{m,\rho}$ into convex regions: the $d$-faces (or cells) of the arrangement. Edelsbrunner's book on combinatorial geometry [32] supplies bounds on the numbers of faces of various dimensions in arrangements of hyperplanes. Lemma 3.11 reproduces the bounds.

**Lemma 3.11** *The maximum number $f_k^{(d)}(n)$ of $k$-faces in an arrangement of $n$ hyperplanes in $\mathbb{R}^d$ is given by*

$$f_k^{(d)}(n) = \sum_{i \in \{0,\ldots,k\}} \binom{d-i}{k-i} \binom{n}{d-i}.$$

We are interested in the maximum number of $d$-faces in an arrangement of $c+2d$ hyperplanes in $\mathbb{R}^d$, or $f_d^{(d)}(c+2d)$ for short. By Lemma 3.11, we have that

$$\begin{aligned} f_d^{(d)}(c+2d) &= \sum_{i \in \{0,\ldots,d\}} \binom{d-i}{d-i} \binom{c+2d}{d-i} \\ &= \sum_{j \in \{0,\ldots,d\}} \binom{c+2d}{j} \\ &\leq \frac{1}{2} \cdot \sum_{j \in \{0,\ldots,c+2d\}} \binom{c+2d}{j} \\ &\leq 2^{c+2d-1}. \end{aligned}$$

The basis of the first of the above inequalities lies in the simple observation that $d \leq \frac{1}{2}(c+2d)$. Note that the bound of $2^{c+2d-1}$ on the number of $d$-faces is probably not very tight, as the $c+2d$ hyperplanes include many parallel pairs of hyperplanes. The $c+2d$ hyperplanes subdivide the collection of polytopes $E \cap C_{m,\rho}$ into $g \leq 2^{c+2d-1}$ convex regions. The largest region $E' \subseteq E \cap C_{m,\rho}$ of these $g$ convex regions clearly satisfies

$$\begin{aligned} volume(E') &\geq \frac{1}{g} \cdot volume(E \cap C_{m,\rho}) \geq \frac{1}{2^{c+2d-1}} \cdot volume(E \cap C_{m,\rho}) \\ &\geq \frac{\omega_d \rho^d}{2^{c+2d-1} k}. \end{aligned} \tag{3.4}$$

The convexity of the subshape $E' \subseteq E \cap C_{m,\rho}$ allows for the subsequent application of Corollary 3.8, Lemma 3.6, and Lemma 3.5. First of all, Corollary 3.8 tells

The data structure presented in the previous section allows us to perform point location queries among the objects of $\mathcal{E}$ with all points in $\Pi$ in time $O(|\Pi|\log^{d-1} n)$. The resulting set $\{E \in \mathcal{E} | \Pi \cap E \neq \emptyset\}$ of query answers, which clearly has at most $|\Pi|$ elements, is a superset of the answer $Q(R)$ to the range query with $R$. For each of the objects $E \in \{E \in \mathcal{E} | \Pi \cap E \neq \emptyset\}$, additional constant time suffices to verify the membership $E \in Q(R)$ by a simple test for the non-emptiness of $E \cap R$, provided that $R$ and all $E \in \mathcal{E}$ have constant complexity. Hence, the computation of $Q(R)$ takes $O(|\Pi|\log^{d-1} n)$ time.

It remains to bound the size of $\Pi$. Clearly, the Minkowski difference $R \ominus S_{O,\rho}$ fits entirely in the hypercube $[x_0^R - \rho, x_0^R + (h+1)\rho] \times \ldots \times [x_d^R - \rho, x_d^R + (h+1)\rho]$, where $x_i^R$ $(1 \leq i \leq d)$ is the minimal $x_i$-coordinate occurring in $R$. As a result, the number of elements in $\Pi$ is bounded by the number of grid points in the enclosing hypercube, leading to

$$\begin{aligned}
|\Pi| &= |\mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho) \cap (R \ominus S_{O,\rho})| \\
&\leq |\mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho) \\
&\quad \cap [x_0^R - \rho, x_0^R + (h+1)\rho] \times \ldots \times [y_d^R - \rho, y_d^R + (h+1)\rho]| \\
&= (\frac{1}{2}kd^{d+\frac{1}{2}}(\lfloor h \rfloor + 2))^d \\
&= O((kd^d h)^d)
\end{aligned}$$

In a setting where all objects are $k$-fat for some constant $k$ and the diameter of the query region $R$ does not exceed a constant multiple $h$ of $\rho$ it follows that $|\Pi| = O(1)$, which implies an $O(\log^{d-1} n)$ time bound for range searching with a bounded-size region $R$.

**Theorem 3.10** *Let $k > 0$ and $h \geq 0$ be constants, and let $\mathcal{E}$ be a set of convex $k$-fat constant-complexity objects $E \subseteq \mathbb{R}^d$ with minimal enclosing hyperspheres with radii at least $\rho$. A range searching query with a region $R \subseteq \mathbb{R}^d$ of diameter at most $h \cdot \rho$ among $\mathcal{E}$ takes $O(\log^{d-1} n)$ time.*

## 3.2.2 Searching among polytopes

Having solved the range searching problem among convex objects we now turn our attention to (non-convex) polytopes. We assume that all polytopes $E \subseteq \mathbb{R}^d$ in $\mathcal{E}$ are bounded by $c$ hyperplanar faces, that is, each face is part of a $(d-1)$-dimensional hyperplane. Let $E$ be a $k$-fat polytope intersecting $R$, and let $m \in E \cap R$. Inequality (3.1) applies to the intersection of the polytope $E$ and the hypersphere $S_{m,\rho} \in U_E$, on exactly the same grounds as in the convex case. Unfortunately, the intersection $E \cap S_{m,\rho}$ is not a polytope as its boundary contains portions of the hyperspherical boundary of $S_{m,\rho}$. This can be remedied by replacing $S_{m,\rho}$ by its (axis-parallel) enclosing hypercube $C_{m,\rho}$ (with 'center' $m$ and side length $2\rho$) with

grid resolution: the resolution is chosen such that any object $E \in Q(R)$ is hit inside a hypersphere $S_{m,\rho}$ with $m \in R$. This hypersphere lies entirely inside the region $R \ominus S_{O,\rho}$, where $O$ is the origin of the Euclidean coordinate frame and $\ominus$ denotes the Minkowski difference operator. The Minkowski difference of two sets $A$ and $B$ is defined by $A \ominus B = \{a - b | a \in A \wedge b \in B\}$. Hence, at least one of the grid points hitting $E$ lies in $R \ominus S_{O,\rho}$. As a result, it suffices to restrict the set of sample points $\Pi$ to be the set of grid points in the grown query region:

$$\Pi = \mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho) \cap (R \ominus S_{O,\rho}).$$

The result is summarized in the following lemma.

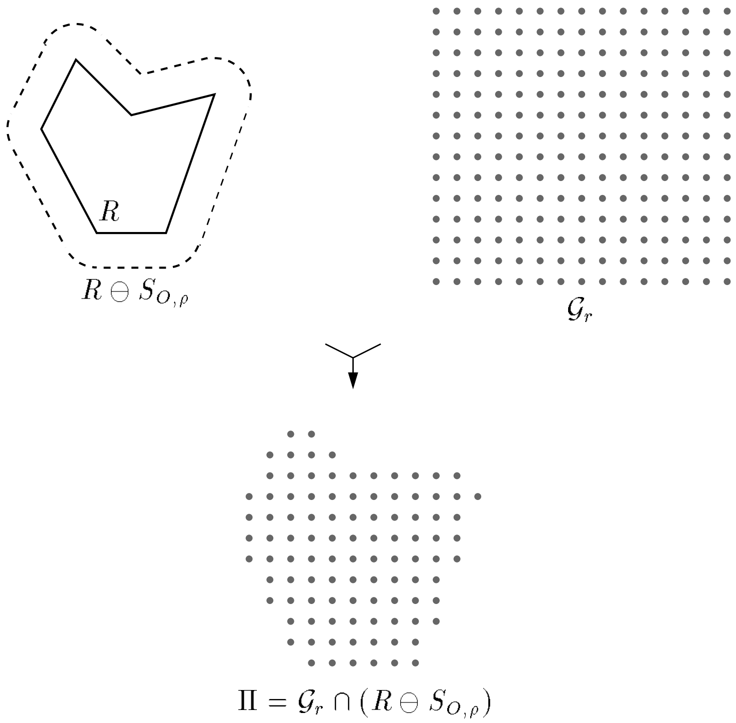


Figure 3.4: A two-dimensional example of the construction of the set of sample points $\Pi$ for a query with a region $R$ with diameter $h \cdot \rho$ among a set $\mathcal{E}$ of objects with minimal enclosing circle radius $\rho$. The resolution $r$ of the orthogonal grid is determined by the type of the objects in $\mathcal{E}$.

**Lemma 3.9** *Let $\mathcal{E}$ be a set of convex $k$-fat objects $E \subseteq \mathbb{R}^d$ with minimal enclosing hyperspheres with radii at least $\rho$ and let $R \subseteq \mathbb{R}^d$ be a region with diameter $h \cdot \rho$, for some constant $h$. The set $Q(R)$ of objects $E \in \mathcal{E}$ intersecting $R$ can be found by point location queries with the points from $\mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho) \cap R \ominus S_{O,\rho}$.*

In the two subsections below, we use the property and lemma to find a valid grid resolution, and identify a small subset of that grid that suffices to hit all objects intersecting the query region $R \subseteq \mathbb{R}^d$.

### 3.2.1 Searching among convex objects

Let $E \subseteq \mathbb{R}^d$ be a convex $k$-fat object intersecting the query region $R \subseteq \mathbb{R}^d$, and let $m \in E \cap R$. The hypersphere $S_{m,\rho}$ belongs to $U_E$ as it is centered inside $E$ and can impossibly have $E$ entirely in its interior. The membership $S_{m,\rho} \in U_E$ and the $k$-fatness of $E$ yield:

$$k \cdot volume(E \cap S_{m,\rho}) \geq volume(S_{m,\rho}) = \omega_d \cdot \rho^d. \tag{3.1}$$

The shape $E \cap S_{m,\rho}$ is convex as it is the intersection of the convex objects $E$ and $S_{m,\rho}$. The following result due to Leichtweiß[61] holds for any convex shape.

**Lemma 3.7** *Let $E \subset \mathbb{R}^d$ be a convex object. There exist ellipsoids $L^I, L^O \subseteq \mathbb{R}^d$ such that $L^I \subseteq E \subseteq L^O$ and*

$$d^d \cdot volume(L^I) \geq volume(L^O).$$

Corollary 3.8 is a trivial consequence of Lemma 3.7.

**Corollary 3.8** *Any convex object $E \subseteq \mathbb{R}^d$ contains an ellipsoid $L$ with*

$$d^d \cdot volume(L) \geq volume(E).$$

Application of Corollary 3.8 to the shape $E \cap S_{m,\rho}$, satisfying (3.1), implies the containment of an ellipsoid $L \subseteq E \cap S_{m,\rho}$ such that

$$volume(L) \geq \frac{\omega_d \rho^d}{k d^d}. \tag{3.2}$$

The diameter of $L$ is bounded by $2\rho$, because $L$ is contained in the hypersphere $S_{m,\rho}$ with diameter $2\rho$. The application of Lemma 3.6 to the ellipsoid $L$ with diameter at most $2\rho$ and volume bounded by (3.2) yields that $L$ contains a hypersphere $S$ with radius at least $k^{-1}d^{-d}\rho$. Property 3.5 subsequently implies that such a hypersphere is hit by at least one point from the regular orthogonal grid with resolution $2k^{-1}d^{-(d+\frac{1}{2})}\rho$, or $\mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho)$. Hence, at least one point from $\mathcal{G}(2k^{-1}d^{-(d+\frac{1}{2})}\rho)$ hits $S \subseteq L \subseteq E \cap S_{m,\rho}$.

So far, we have only bothered about finding a sufficiently high resolution for a grid to hit all objects $E \in Q(R)$. Clearly, it is unnecessary and even undesirable to perform point location queries with a too large subset of the grid, both because it increases the query time and because it would lead to many accidental hits of objects $E \notin Q(R)$. Fortunately, the size of the sample set (and the number of accidental hits) can be adequately limited by a quick glance at the deduction of the

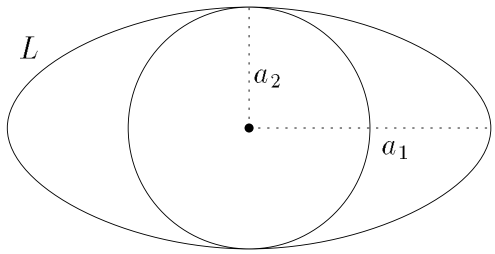


Figure 3.3: The ellipse $L$ with half-axes $a_1 \geq a_2$ contains a circle with radius $a_2$.

where $\omega_d$ is again the dimension-dependent constant multiplier from the volume formulae for hyperspheres (see Chapter 2).

A lower bound $V$ on the volume of an ellipsoid $L$ alone does not suffice to prove that $L$ encloses a large hypersphere, as it is possible to construct a very 'long and thin' ellipsoid. An additional upper bound on the diameter of the ellipsoid, and, hence, on the $a_i$'s ($1 \leq i \leq d$), however, makes such a construction impossible. This follows easily from the volume formula for ellipsoids.

**Lemma 3.6** *Let $L \subseteq \mathbb{R}^d$ be an ellipsoid with $volume(L) \geq V$ and let $\delta$ be an upper bound on its diameter. Then $L$ contains a hypersphere with radius at least $\frac{V}{\omega_d} \cdot (2/\delta)^{d-1}$.*

**Proof:** Assume without loss of generality that $L$ is in standard position, and, hence, of the form $\sum_{i\in\{1,\ldots,d\}} x_i^2/a_i^2 = 1$. Then, its volume is $volume(L) = \omega_d \cdot \prod_{i\in\{1,\ldots,d\}} a_i$. The upper bound of $\delta$ on the diameter of $L$ implies that none of the axes of $L$ is longer than $\delta$, and, thus, for all $1 \leq i \leq d$:

$$a_i \leq \delta/2.$$

Now assume, for a contradiction, that $a_j < \frac{V}{\omega_d} \cdot (2/\delta)^{d-1}$. Subsequent application of this inequality and the upper bound $a_i \leq \delta/2$ ($1 \leq i \leq d$) yields

$$\begin{aligned} volume(L) &= \omega_d \cdot a_j \cdot \prod_{i\in\{1,\ldots,d\}, i\neq j} a_i \\ &< \omega_d \cdot \frac{V}{\omega_d} \cdot (2/\delta)^{d-1} \cdot \prod_{i\in\{1,\ldots,d\}, i\neq j} a_i \\ &\leq \omega_d \cdot \frac{V}{\omega_d} \cdot (2/\delta)^{d-1} \cdot (\delta/2)^{d-1} \\ &= V, \end{aligned}$$

contradicting the assumption $volume(L) \geq V$.

The ellipsoid $L$ entirely contains the hypersphere with radius $\frac{V}{\omega_d} \cdot (2/\delta)^{d-1}$ centered at the origin. $\square$

This section focuses on the problem of finding a grid resolution such that a small subset of the corresponding regular orthogonal grid is guaranteed to hit any object $E$ having non-empty intersection with the query region $R$. We shall first determine a suitable grid resolution for convex objects, and subsequently use the results obtained there to find an appropriate resolution for general polytopes.

The main implication of these results is that the range query with the bounded-size range $R \subseteq \mathbb{R}^d$ can be solved by a sequence of point location queries, each taking $O(\log^{d-1} n)$ time, using the data structure for point location among fat objects. Under the assumption that the diameter of the query region does not exceed $h \cdot \rho$, the sequence of point location queries will have constant length.

The aim is to find a grid resolution $r$ establishing that each object $E \in Q(R)$ is hit by at least one point in some subset $\Pi \subset \mathcal{G}(r)$, where, preferably, the size of $\Pi$ depends on $k$ and $h$ (and the complexity of the individual objects) only. Before we focus on the different types of objects, we first give some basic results that ease the task to find a grid resolution.

To simplify the approach, we will define for each object a large hypersphere that is contained in the object $E \in Q(R)$. For the hypersphere, it is easy to determine the grid resolution $r$ such that the hypersphere is always hit by at least one of the grid points.

**Property 3.5** *Any hypersphere with radius at least $\frac{1}{2}r\sqrt{d}$ contains at least one point of the orthogonal grid $\mathcal{G}(r)$.*

The hypersphere itself is determined in two steps. First, a result by Leichtweiß[61] makes it possible to identify a large ellipsoid inside a convex part of the object $E$. Due to the fatness of the object $E$, the ellipsoid, in turn, indeed contains a large hypersphere.

Ellipsoids play an important role throughout this section. Let us therefore briefly review some relevant properties of these shapes. Any ellipsoid $L \subseteq \mathbb{R}^d$ can be regarded as a translated and rotated copy of an ellipsoid in so-called standard position. An ellipsoid $L_s$ in standard position has the following form

$$L_s : \sum_{i \in \{1,\ldots,d\}} \frac{x_i^2}{a_i^2} \leq 1,$$

where $a_1, \ldots, a_d$ are constants. The segment connecting the points $0^{i-1} \times -a_i \times 0^{d-i} \in L_s$ and $0^{i-1} \times a_i \times 0^{d-i} \in L_s$, which has length $2a_i$, is referred to as an axis of $L_s$; $L_s$ has $d$ such axes. If $w \leq a_i$ for all $1 \leq i \leq d$, then the hypersphere with radius $w$ centered at the origin is entirely contained in $L_s$ (see Figure 3.3). The volume of an ellipsoid can be given as a function of the lengths of its axes; the volume of $L_s$ (and of its translated and rotated copies $L$) is given by (see e.g. [99])

$$volume(L_s) = \omega_d \cdot \prod_{i \in \{1,\ldots,d\}} a_i,$$

*complexity query region $R$ of arbitrary shape with diameter at most $h \cdot \rho$ for some positive constant $h$, report all objects $E \in \mathcal{E}$ that intersect $R$.*

Figure 3.2 shows a bounded-size range query in a set of five non-intersecting fat objects. The query with the range $R$ must yield the answer $\{E_2, E_3, E_5\}$. Using

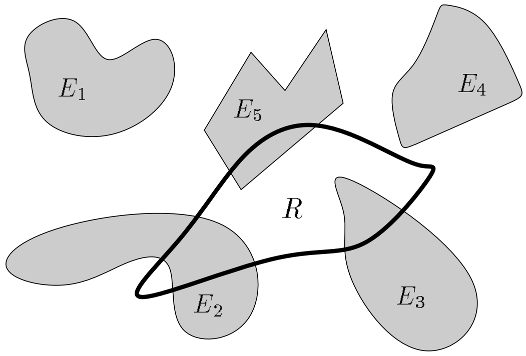


Figure 3.2: A range query in the set $\{E_1, \ldots, E_5\}$; the query with $R$ yields the answer $\{E_2, E_3, E_5\}$.

Theorem 2.9 it is easy to verify that the answer to the query with a region $R$, satisfying the diameter bound, is a set of objects of constant cardinality. Let us define the set $Q(R)$ of objects intersecting the region $R$:

$$Q(R) = \{E \in \mathcal{E} | E \cap R \neq \emptyset\}.$$

It is shown how the point location structure can be used to solve the range searching problem in time $O(\log^{d-1} n)$ in the case that $\mathcal{E}$ is a set of arbitrary convex objects and/or non-convex polytopes. The solution relies on local properties of fat objects. The definition of fatness requires a $k$-fat object $E$ to have a large 'density' in the vicinity of any point $p$ in the object: $1/k$-th of a hypersphere centered at $p$ is covered by $E$. It turns out that this property makes it possible to hit any object or object part with a certain minimum size, regardless of its exact location, with at least one point from a sufficiently dense, but not too large, pattern of sample points, whereas this would clearly be impossible if the obstacle is non-fat: the chance to hit a line segment by an extremely dense pattern of sample points is practically zero. To structure the problem and the shape of its solution, we restrict the sample points to be arranged as a regular orthogonal grid.

**Definition 3.4** *A regular orthogonal grid $\mathcal{G}(r)$ with resolution $r$ is defined by:*

$$\mathcal{G}(r) = \{(z_1 r, \ldots, z_d r) | z_1, \ldots, z_d \in \mathbf{Z}\}.$$

The structure uses $O(n \log^{d-1} n)$ storage. The result is summarized in the following theorem.

**Theorem 3.3** *A set $\mathcal{E}$ of non-intersecting constant-complexity k-fat objects in $\mathbb{R}^d$ can be stored in a data structure of size $O(n \log^{d-1} n)$, such that, for a query point $p \in \mathbb{R}^d$, it takes $O(\log^{d-1} n)$ time to report the object $E \in \mathcal{E}$ that contains p, or to conclude that no object contains p.*

The remaining open problem concerns the preprocessing phase, that is, the computation of the data structure: given a set of non-intersecting $k$-fat constant-complexity objects $E_1, \ldots, E_n$ ordered by increasing radii of their minimal enclosing hyperspheres, compute the multi-level data structure storing the enclosing hypercubes $C_1, \ldots, C_n$ of the minimal enclosing hyperspheres of these objects plus the sets $V_1, \ldots, V_n$ of larger objects intersecting the respective hypercubes $C_1, \ldots, C_n$. Building the multi-level data structure can be accomplished in time $O(n \log^{d-1} n)$ using standard techniques. Another option, that is exploited in Section 3.3, is to build the structure in an incremental way. It is well-known that a hypercube can be inserted in a $d$-level segment tree in time $O(\log^d n)$. Moreover, if we use dynamic fractional cascading [68] instead of 'regular' fractional cascading, the insertion time can be further reduced to $O(\log^{d-1} n \log \log n)$ (see Section 3.3).

The computation of the sets $V_i$, on the other hand, seems to pose more problems. Finding the objects $E_j$ with $j \geq i$ intersecting the hypercube $C_i$ requires a range search query with $C_i$. The query is not an ordinary range search query, since we are only interested in objects with a certain minimal size (or index). Performing a range search query among all objects and subsequently filtering out the smaller objects is not a good idea, as the answer to the range query might by orders of magnitude larger than $V_i$. A better idea would be to perform a range search query with $V_i$ only among objects that are larger than $E_i$. Surprisingly, we show in the next section that it is possible, in most interesting cases, to use the point location structure itself for solving the range search query. This suggests an approach where we add the hypercubes from large to small, meanwhile computing the sets $V_i$ in the following (incremental) way: use, before insertion of the hypercube $C_{n-m}$, the sets $V_{n-m+1}, \ldots, V_{n-m}$ and the multi-level data structure storing the hypercubes $C_{n-m+1}, \ldots, C_n$ to compute $V_{n-m}$ by a range query with $C_{n-m}$ among the objects $E_{n-m+1}, \ldots, E_n$. Next, insert $C_{n-m}$ into the structure and continue with $C_{n-m-1}$. Section 3.3 contains the details of the approach.

## 3.2 Range searching by point location

In this section we use the point location data structure to tackle the following general version of the range searching problem:

> *Given a set $\mathcal{E}$ of non-intersecting constant-complexity $k$-fat objects in $\mathbb{R}^d$ with minimal enclosing hyperspheres with radii at least $\rho$ and a constant-*

$d = 1$ The hypercubes $C_1, \ldots, C_n$ are intervals on the real line. The interval endpoints partition the real line into $2n + 1$ so-called elementary intervals. All points within a single elementary interval are covered by exactly the same one-dimensional hypercubes. We organize the intervals as an ordered list and label each elementary interval with the index of the smallest of all one-dimensional hypercubes covering it. The list structure requires $O(n)$ storage. The answer to a point stabbing query is provided by the label of the elementary interval containing the query point. The interval can be identified in time $O(\log n)$.

$d > 1$ The $d$-dimensional hypercubes are stored in a segment tree $T$ on their projections onto the $d$-th coordinate axis. The endpoints of the hypercube projections partition the $d$-th coordinate axis into a number of elementary intervals. An interval $I_\nu$ is associated with each node $\nu$ in $T$: $I_\nu$ is the union of all (consecutive) elementary intervals associated with the leaves of the subtree rooted at $\nu$. With $\nu$ we store in an associated structure the intersections $[-\infty, \infty]^{d-1} \times I_\nu \cap C_i$ for hypercubes $C_i$ that entirely span the slab $[-\infty, \infty]^{d-1} \times I_\nu$ but do not entirely span the slab $[-\infty, \infty]^{d-1} \times I_{\nu'}$ corresponding to the parent $\nu'$ of $\nu$ in $T$. The projection of such an intersection $[-\infty, \infty]^{d-1} \times I_\nu \cap C_i$ onto the subspace spanned by the first $d - 1$ coordinate axes is a $(d - 1)$-dimensional hypercube. We store these hypercubes in a (recursively defined) similar $(d - 1)$-level data structure on the first $d - 1$ coordinates, that is, if $d - 1 > 1$. If $d - 1 = 1$ we use the one-dimensional construction outlined above to store the intersection of the squares and the planar slab. The structure suffices because the squares intersect the vertical slab perpendicularly, and can therefore be represented as intervals. So, the bottom-level structure is a list or ordinary balanced binary tree instead of a segment tree.

Searching the multi-level data structure with a query point $p$ proceeds in the following recursive manner: start at the root and repeatedly continue towards the child corresponding to the slab containing $p$. (Testing only the last coordinate is sufficient.) The search ends at the leaf corresponding to the elementary interval containing the last coordinate. We have now obtained $O(\log n)$ nodes on the search path, each corresponding to a slab containing $p$. The search is continued recursively in the substructures associated to each of these nodes. The entire search from top to bottom in the multi-level data structure takes therefore $O(\log^d n)$ time, resulting in $O(\log^{d-1} n)$ candidate answers. The minimum among these candidates is the final answer to the query. The query time can be improved by applying fractional cascading [25] to the two lower levels of the data structure. This is possible because the bottom-level structures are ordered lists (a sequence of intervals). Fractional cascading improves the query time in a 2-level data structure consisting of a segment tree with the one-dimensional ordered lists as substructures from $O(\log^2 n)$ to $O(\log n)$. Hence, a priority point stabbing query among hypercubes takes $O(\log^{d-1} n)$ time.

Let furthermore $\rho_i$ be the radius of $E_i$'s minimal enclosing hypersphere. Let us denote the axis-parallel enclosing hypercube of the minimal enclosing hypersphere of an object $E_i$ by $C_i$. By construction, the hypercubes are ordered by increasing size. Note that the side length of the hypercube $C_i$ is $2\rho_i$. Notice that the hypercubes $C_i$ may be overlapping, although the objects $E_i$ are disjoint. Furthermore, let $V_i$ be defined as follows:

$$V_i = \{E_j \in \mathcal{E} | E_j \cap C_i \neq \emptyset \wedge j \geq i\}.$$

Hence, $V_i$ is the set of objects that are larger than $E_i$, i.e., with larger minimal enclosing hypersphere, intersecting the box $C_i$. Theorem 2.9 immediately supplies a useful property of the sets $V_i$, $1 \leq i \leq n$.

**Lemma 3.1** *For all* $i : 1 \leq i \leq n : \; |V_i| = O(1)$.

A second crucial lemma from [73] is the following.

**Lemma 3.2** *Let* $p \in E_j$ *and* $i = \min\{h | p \in C_h\}$. *Then* $E_j \in C_i$.

In words, the lemma states that if the query point $p$ lies in an object $E_j$, then $E_j$ is an element of the set of objects $V_i$ associated to the smallest hypercube $C_i$ containing $p$. This suggest the approach outlined below.

Assuming that the hypercubes $C_i$ and the sets $V_i$ for all $1 \leq i \leq n$ are available, we proceed as follows to find the answer to a point location query with a point $p \in \mathbb{R}^d$. Determine the smallest hypercube, if any, containing $p$. If no hypercube contains $p$ than $p$ lies in no object; if, on the contrary, $C_i$ is the smallest hypercube containing $p$, then the set $V_i$ must contain the answer to the query. To this end, we check the objects in $V_i$ for containment of $p$. Note that the check for containment of a point in an object $E_j \in V_i$ takes constant time due to the constant complexity of the objects. The constant cardinality of $V_i$ yields that the entire inspection of all objects in $V_i$ takes $O(1)$ time. If no object in $V_i$ contains $p$ then no object in $\mathcal{E}$ contains $p$; otherwise the unique object $E_j \in V_i$ containing $p$ obviously is the answer to the query. As the inspection of $V_i$ takes constant time, the point location query time is dominated by the time to find the smallest hypercube $C_i$ containing the query point $p$. An appropriate data structure that solves this problem is given below.

The point location problem is now essentially reduced to the following priority point stabbing problem among (intersecting) hypercubes:

> *Given a set of hypercubes* $\mathcal{C}$ *in* $\mathbb{R}^d$ *and a query point* $p \in \mathbb{R}^d$, *report the smallest hypercube* $C \in \mathcal{C}$ *containing* $p$, *or report that no hypercube in* $\mathcal{C}$ *contains* $p$.

To solve a priority point stabbing query among hypercubes, Overmars proposes a $d$-level data structure in which the upper $d-1$ levels are based on the segment tree and the lowest level is a list or a balanced binary tree.

## 3.1 Point location among fat objects

This section discusses a data structure for point location among disjoint fat objects by Overmars [73]. The author presents the data structure as a structure for solving the problem of point location in subdivisions of $d$-dimensional space into fat cells. The answer to the query is the specific cell containing the query point. The point location problem in fat subdivisions can be seen as an instance of the following, more general, formulation of the point location problem:

> *Given a set $\mathcal{E}$ of non-intersecting constant-complexity $k$-fat objects in $\mathbb{R}^d$ and a query point $p \in \mathbb{R}^d$, report the object $E \in \mathcal{E}$ that contains $p$, or report that no such object exists.*

Figure 3.1 shows two point location queries in a set of five non-intersecting fat objects. The query with the point $p$ should yield the answer $E_1$, whereas the query with $q$ must result in the answer that no object contains $q$. If the objects in $\mathcal{E}$

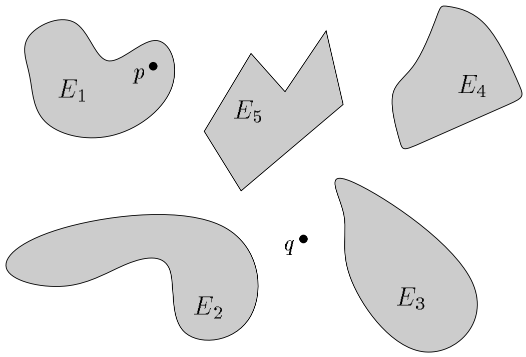


Figure 3.1: Two point location queries in the set $\{E_1, \ldots, E_5\}$; the query with $p$ yields the answer $E_1$, while $q$ is reported to lie in no object.

entirely cover $\mathbb{R}^d$ then we obtain a fat subdivision like in Overmars' paper. In the more general setting, the complement of the objects need not be fat, nor does it have constant complexity. The ideas from [73] are discussed below in the context of this more general problem formulation.

Overmars' paper only presents a data structure for efficiently answering point location queries; the issue of building the structure remains untouched. Later in this chapter, a solution for this problem is given for arbitrary convex objects and non-convex polytopes. The solution relies on the ability to do efficient range searching queries among these objects. Before discussing range searching and its application to building the point location structure, this section simply summarizes the results of Overmars presented in [73].

Let us assume in this chapter that the constant-complexity $k$-fat objects in $E_1, \ldots, E_n \in \mathcal{E}$ are ordered by radius of their minimal object enclosing hyperspheres.

precise value depends on $\Gamma$.

Overmars [73] discusses a data structure for efficient and simple point location in fat subdivisions or sets of fat objects with total complexity $n$. The structure supports point location queries in time $O(\log^{d-1} n)$ and uses $O(n \log^{d-1} n)$ storage. In his paper, Overmars does not touch the issue of efficiently computing the data structure, that is, in time comparable to the storage requirement. It is shown in this chapter that for arbitrary convex objects and for non-convex polytopes, the structure can be built incrementally in time $O(n \log^{d-1} n \log\log n)$. Besides supporting efficient point location queries, the data structures storing the arbitrary convex objects and/or polytopes can, surprisingly, also be used for range searching with arbitrarily-shaped but bounded-size ranges. In fact, we show that each bounded-size range query can be implemented by a constant number of point location queries, thus leading to a time bound of $O(\log^{d-1} n)$ for range search queries. Chapter 6 gives an important application of bounded-size range searching. The motion planning paradigm presented there requires the a priori knowledge of all pairs of neighboring obstacles, where the notion of neighboring is related to the size of the smallest obstacle. The results presented in this chapter facilitate the computation of all pairs in time $O(n \log^{d-1} n \log\log n)$.

The main contribution of our results with respect to the point location problem lies in their dimensional generality, where previous results in higher dimensions are restricted to arrangements of hyperplanes or hypersurfaces of bounded degree, and to severely restricted subdivisions and scenes of non-intersecting convex polyhedra in 3-space. We find that point location queries in scenes of non-intersecting (or mildly intersecting) fat convex objects and/or fat polytopes can be performed in polylogarithmic time at the cost of near-linear storage and preprocessing. To see the contribution for range searching we recall the results by Van Kreveld [57] quoted earlier. A query with a simplex in $d$-space for all contained simplices takes logarithmic query time at the cost of roughly $O(n^d)$ storage or linear storage at the cost of polynomial query time. The solutions to range searching among points have similar performance. Although the range searching results apply only to fat objects and small query ranges, they succeed in combining polylogarthmic query time with near-linear storage and query time for simplicial range searching among arbitrary convex shapes and non-convex polytopes. Moreover, the data structure caters for arbitrary query ranges.

The first section below discusses Overmars' data structure for efficient point location among fat objects, while the next section shows how the structure can be used for simple and efficient range searching among classes of fat objects. In the third section, the range searching results are used to support the incremental construction of the multi-purpose data structure, starting from the largest object and repeatedly adding the next largest object. Finally, we summarize the results and point out the various potential generalizations.

arbitrary dimension $d$ are further restricted, applying only to arrangements of hyperplanes or hypersurfaces of bounded degree. Clarkson [28] presents a data structure for point location in an arrangement of $n$ hyperplanes in $d$-dimensional space. The structure supports queries in time $O(\log n)$ and requires roughly[1] $O(n^d)$ storage and preprocessing. (Chazelle and Friedman [24] improve the storage to exactly $O(n^d)$ at the cost of an increase of the preprocessing time to $O(n^{2d+2})$). Chazelle et al. [23] achieve the same $O(\log n)$ query time for point location in arrangements of hypersurfaces of constant degree with a data structure of size roughly $O(n^{2d-3})$, which is computable in roughly $O(n^{2d-2})$ time. Apparently, efficient solutions for sets of non-convex objects or non-polyhedra in 3-space and for scenes other than arrangements of hyperplanes or hypersurfaces of bounded degree in higher-dimensional spaces are lacking.

Nearly all papers on range searching discuss how to preprocess a very elementary class of geometric objects, namely points, for efficiently answering range search queries with specific range types. The most extensively studied type of query range is the orthogonal range, and a long-established result says (see e.g. [79]) that a set consisting of $n$ points can be preprocessed in time $O(n \log^{d-1} n)$ into a data structure of size $O(n \log^{d-1} n)$, which is capable of answering an orthogonal range query in time $O(\log^{d-1} n)$[2]. (Some small improvements are possible.) Different range types give rise to more complicated solutions. In general, the solutions to e.g. simplicial range searching only provide low query time at the cost of a relatively high storage requirement and preprocessing time or vice versa. At the one end, one finds solutions providing polylogarithmic query time and roughly $O(n^d)$ storage and preprocessing (see [26, 66]), whereas, at the other end of the spectrum, solutions require only $O(n)$ storage, but guarantee only a larger query time of $O(n^{1-1/d})$ (see [65, 66]). Van Kreveld [57] gives similar bounds for the problem of reporting all simplices that are entirely contained in a query simplex. In between the previous results are the trade-off solutions which allow for exchanging storage for query time. An example of such a solution is given by Matoušek [66]: the presented structure supports queries in time $O((n/m^{1/d}) \log^{d+1}(m/n))$ at the cost of an $O(m)$ storage requirement, where $n \leq m \leq n^d$. Alternative trade-off solutions provide similar bounds. More specific results (with respect to dimension) are reported by Chazelle and Welzl [27] who give a solution with $O(\sqrt{n} \log n)$ query time and $O(n)$ storage for triangular range searching in 2-space and a solution with $O(n^{2/3} \log^2 n)$ query time and a storage requirement of $O(n \log n)$ storage for tetrahedral range searching in 3-space. Semi-algebraic query ranges are considered by Agarwal and Matoušek [2], resulting in a data structure of size $O(n)$ with preprocessing time $O(n \log n)$ which supports range queries with a region $\Gamma$ in $d$-space in time $O(n^{1-1/b+\delta})$, where $\delta$ is some arbitrarily small value and $b$ is bounded by $d \leq b \leq 2d-3$, although its

[1] All quoted rough bounds are adequate up to a factor $n^\epsilon \log^c n$, for some arbitrarily small $\epsilon$ and some constant $c$.

[2] In all quoted time bounds we neglect the dependence of the query time on the size of the answer to the query. All results actually have the form $O(f(n) + h)$, where $h$ is the output size.

# Chapter 3

# Range searching and point location among fat objects

In this chapter we study two fundamental problems in computational geometry in a context of fat objects: point location and range searching. The point location problem aims at preprocessing a set of disjoint geometric objects for efficiently reporting the specific object containing a query point. The objective of the (general) version of the range searching problem is to preprocess a set of geometric objects for quickly reporting all objects intersecting some query range (e.g. rectangloid, simplex, hypersphere). It is shown that arbitrary convex objects and/or non-convex polytopes in $d$-dimensional space can be preprocessed in time $O(n \log^{d-1} n \log \log n)$ into a data structure of size $O(n \log^{d-1} n)$ which supports point location queries and range searching queries with arbitrarily-shaped but bounded-size regions in time $O(\log^{d-1} n)$. The data structure is based on Overmars' structure for point location in fat subdivisions [73]. Let us briefly review some relevant results in both point location and range searching to place our result in a broader perspective.

Point location in 2-space has been studied extensively and solved in a satisfactory way for many types of scenes, as several solutions achieve logarithmic query time and (near-)linear storage, after (near-)linear preprocessing time [29, 33, 54]. In higher-dimensional spaces, on the contrary, efficient solutions are available only for restricted problem instances. In 3-space, Chazelle [22] obtains $O(\log^2 n)$ query time and an $O(n)$ storage requirement for the case where the stored geometric objects are the 3-cells of a spatial subdivision, consisting of a total of $n$ facets and satisfying the restrictive constraint that the vertical dominance relation on its cells is acyclic. Preparata and Tamassia [80] consider point location in a set of disjoint convex polyhedra with total complexity $n$. (The polyhedra subdivide $\mathbb{R}^3$ into a number of convex cells - the polyhedra - and a single non-convex cell - the complement of the polyhedra.) Their data structure uses $O(n \log^2 n)$ storage and is capable of answering point location queries in time $O(\log^2 n)$ after $O(n \log^2 n)$ preprocessing time. Goodrich and Tamassia [37] improve the storage requirement for sets of disjoint convex polyhedra to $O(n \log n)$ without affecting the query time. The results for

to determine what part of any axis-parallel square with the intersection point of its diagonals inside $E$ and not fully containing $E$ in its interior is covered by $E$. In other words, we wish to know $E$'s fatness with respect to the square $C = \{(x, y) \in \mathbb{R}^2 | -1 \leq x, y \leq 1\}$. Now, with the theorem available, the only thing that remains to be done is to determine the fatness of the square (w.r.t. the circle). With the additional knowledge that the square is $2\pi$-fat, Theorem 2.18 yields that $E$ is $(2\pi k)$-fat$_C$, and, hence, also that any axis-parallel square with its diagonals intersecting in $E$ and not fully containing $E$ is covered for at least $1/(2\pi k)$-th by $E$.

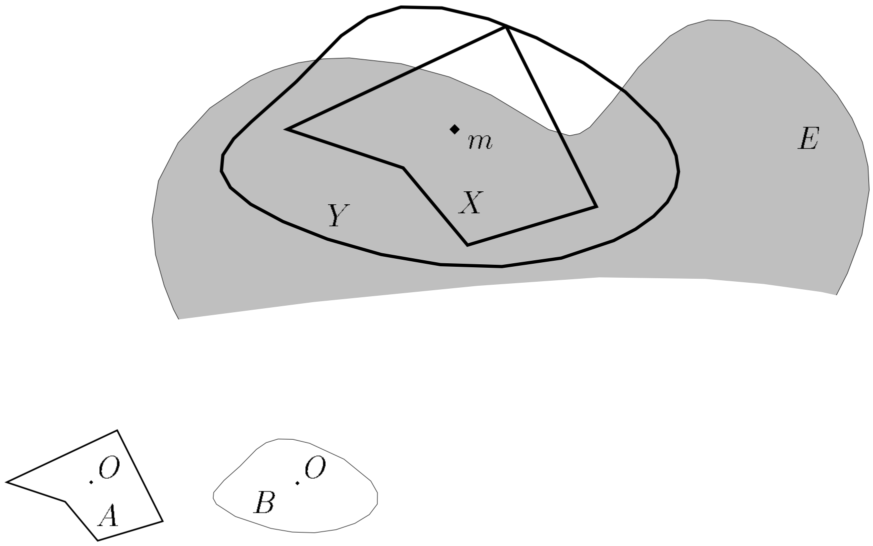


Figure 2.8: Construction of $Y = \lambda B + m$ and $X = \lambda' A + m$, for some arbitrary $m \in E$ and some arbitrary $\lambda$ establishing $\partial Y \cap E \neq \emptyset$.

the assumption that $\lambda'$ is the largest positive real such that $X = \lambda' + m$ fits in $Y$. Hence, we obtain that $Y = \lambda B + m \in U_X^B$.

The object $A$ is $k'$-fat$_B$. By the property that the generalized fatness of an object is not affected by translation and scaling of the object, the object $X = \lambda' A + m$ is also $k'$-fat$_B$. By the definition of fatness, this means that for each $S \in U_X^B$: $k' \cdot \mathit{volume}(X \cap S) \geq \mathit{volume}(S)$. Since $Y \in U_X^B$ and $X \subseteq Y$, we get

$$k' \cdot \mathit{volume}(X) = k' \cdot \mathit{volume}(X \cap Y) \geq \mathit{volume}(Y).$$

Combining both inequalities with the straightforward inequality $\mathit{volume}(E \cap Y) \geq \mathit{volume}(E \cap X)$ induced by $Y \supseteq X$, results in the following lower bound on the part of $Y$ covered by $E$

$$k \cdot k' \cdot \mathit{volume}(E \cap Y) \geq k \cdot k' \cdot \mathit{volume}(E \cap X) \geq k' \cdot \mathit{volume}(X) \geq \mathit{volume}(Y).$$

Recalling the fact that $Y$ was chosen randomly from all members of $U_E^B$, we may conclude that in fact

$$\forall S \in U_E^B \qquad k \cdot k' \cdot \mathit{volume}(E \cap S) \geq \mathit{volume}(S),$$

holds. So, the object $E$ is $(k \cdot k')$-fat$_B$. $\square$

A two-dimensional example illustrates the use of the above theorem. Assume we are given an object $E$ that is $k$-fat, i.e., $E$ is $k$-fat with respect to the (unit)-circle centered at $O$. For some reason (see for example Chapter 3), our interest is

**Definition 2.17** **[$k$-fatness with respect to shape $A$]**
*Let $E \subseteq \mathbb{R}^d$ be an object and let $k$ be a positive constant. The object $E$ is $k$-fat$_A$ if:*

$$\forall S \in U_E^A \qquad k \cdot volume(E \cap S) \geq volume(S).$$

Note that fatness with respect to some arbitrary shape $A$ is not invariant under rotation like the regular fatness (with respect to hyperspheres). Property 2.3(a),(c),(d) hold for the generalized notion of fatness.

Theorem 2.18 establishes a relation between the fatness of an object with respect to a shape $A$ and with respect to a shape $B$. The fatness of the object $E$ with respect to $B$ depends on its fatness with respect to $A$ and the relative fatness of $B$ with respect to $A$.

**Theorem 2.18** *Let $A \subseteq \mathbb{R}^d$ and $B \subseteq \mathbb{R}^d$ be shapes with $O \in A$ and $O \in B$. Let $E \subseteq \mathbb{R}^d$ be a closed connected object. If the object $E$ is $k$-fat$_A$ and $A$ itself is $k'$-fat$_B$, then the object $E$ is $(k \cdot k')$-fat$_B$.*

**Proof:** We must prove that for each $S \in U_E^B$, the inequality $k \cdot k' \cdot volume(E \cap S) \geq volume(S)$ holds. We choose some arbitrary $m \in E$. In addition, we choose some arbitrary $\lambda$, such that $\partial(\lambda B + m) \cap E \neq \emptyset$. The set $\lambda B + m$ will be denoted by $Y$. We define $X = \lambda' A + m$ such that $\lambda'$ is the largest positive real for which $X \cap (\mathbb{R}^d - Y) = \emptyset$. In words, $\lambda'$ is the largest positive real for which $X$ fits in $Y$. Note that point $m$ not only lies in $E$, but, because $O \in A$ and $O \in B$, also lies in $X$ and $Y$. The intersection $E \cap X \cap Y$ is therefore non-empty. (See Figure 2.8 for a two-dimensional example of the construction.)

Our first step is to prove that $X = \lambda' A + m \in U_E^A$. The intersection of $E$ and $X$ is non-empty. Object $E$ will therefore either lie completely in the interior of $X$ or the boundary $\partial X$ of $X$ is intersected by $E$. The assumption that $Y \in U_E^B$ implies that the boundary $\partial Y$ of $Y$ is intersected by $E$. As $\partial Y$ lies completely outside the interior $(X - \partial X)$ of $X$, this means that the object $E$ must lie partly outside the interior of $X$. Combining the facts that the intersection of $E$ and $X$ is non-empty and that $E$ lies partly outside the interior of $X$ with the assumption that $E$ is connected implies that $\partial X \cap E \neq \emptyset$. Together with the fact that $m \in E$ we obtain that $X = \lambda' A + m \in U_E^A$.

The object $E$ is $k$-fat$_A$, which, by definition of fatness, means that for each $S \in U_E^A$: $k \cdot volume(E \cap S) \geq volume(S)$. Since $X \in U_E^A$, we yield

$$k \cdot volume(E \cap X) \geq volume(X).$$

In a second step we prove that $Y = \lambda B + m \in U_X^B$. We know that $m \in X$, so the only thing that remains to be proven is that $\partial Y \cap X \neq \emptyset$. Assume for a contradiction that $\partial Y \cap X = \emptyset$. Since $m \in X \cap Y$, this would imply that $X$ lies completely in the interior $(Y - \partial Y)$ of $Y$. But then we can grow $X = \lambda' A + m$, by increasing $\lambda'$, while it remains fitting in $Y$. This ability to grow $\lambda'$ contradicts

**Theorem 2.15** *Let $E \in \mathbb{R}^d$ be a closed connected object that is not $k$-fat, and let $k' \leq k$. Any covering of $E$ by $k'$-fat parts consists of $\Omega(\log(k/k'))$ parts.*

**Proof:** Let $\mathcal{P}$ be such that each $P \in \mathcal{P}$ is $k'$-fat and $\cup_{P\in\mathcal{P}} P = E$, so $\mathcal{P}$ is a covering of $E$ by $k'$-fat parts. The union $\cup_{P\in\mathcal{P}} P$ is not $k$-fat. Now divide $\mathcal{P}$ into two (non-empty) subsets $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\cup_{P\in\mathcal{P}_1} P$ and $\cup_{P\in\mathcal{P}_2} P$ are connected sets. By Corollary 2.14, at least one of $\cup_{P\in\mathcal{P}_1} P$ and $\cup_{P\in\mathcal{P}_2} P$ is not $(\zeta_d \cdot k)$-fat. We recursively apply the above procedure to the subsets that contain more than one part. As a result, it takes at least ${}^{\zeta_d}\log(k/k') + 1$ recursive set divisions to end up with subsets $\mathcal{P}' \subset \mathcal{P}$ such that $\cup_{P\in\mathcal{P}'} P$ is $k'$-fat. Hence $\mathcal{P}$ must contain $\Omega(\log(k/k'))$ parts. □

Theorem 2.15 gives a lower bound on the number of parts that is involved in any covering of a thin object by fat parts. The existence of an upper bound on the same number, on the other hand, seems unlikely. Especially for small values of $k'$, a covering of the object $E$ (which is not $k$-fat) by $k'$-fat parts may require many small parts.

## 2.5 Fatness defined with respect to other shapes

This section presents a proof of the supposition from Section 2.1 concerning the close relation between fatness definitions with respect to different compact shapes, that is, if an object is $k$-fat with respect to a compact shape $A$ then it is $k'$-fat with respect to some other shape $B$ for some $k'$ that is only a constant multiple of $k$. This supposition sounds rather vague, as it is yet unclear what exactly 'fatness with respect to a shape $A$' means. We define fatness with respect to $A$ by rather straightforward generalization of Definition 2.2.

Let $A \subseteq \mathbb{R}^d$ be a closed connected subset containing the origin $O$ ($O \in A$). Any scaled translate $X$ of $A$ can be described by $X = \lambda A + m$, where $\lambda \in \mathbb{R}^+$ is a scaling parameter and $m = (m_1, \ldots, m_d) \in \mathbb{R}^d$ a translation vector. (So $X = \{(\lambda a_1 + m_1, \ldots, \lambda a_d + m_d) | (a_1, \ldots, a_d) \in A\}$.)

**Definition 2.16** $[U^A_{m,E}, U^A_E]$
*Let $m \in \mathbb{R}^d$ and let $E \subseteq \mathbb{R}^d$ be an object. The set $U^A_{m,E}$ is defined as:*

$$U^A_{m,E} = \{\lambda A + m \mid \partial(\lambda A + m) \cap E \neq \emptyset\}$$

*The set $U_E$ is defined as:*

$$U^A_E = \bigcup_{m\in E} U^A_{m,E}$$

The following definition is a generalization of the notion of fatness given in Definition 2.2.

application of the definition of $\zeta_d$ yield:

$$\begin{aligned}
& \frac{1}{\max(k_1,k_2)} \cdot ((\frac{r_{ES}}{r})^d \cdot volume(S) + (\frac{(r-r_{ES})/2}{r})^d \cdot volume(S)) \\
= & \frac{1}{\max(k_1,k_2)} \cdot ((\frac{1}{\rho})^d \cdot volume(S) + (\frac{(\rho-1)/2}{\rho})^d \cdot volume(S)) \\
= & \frac{1}{\max(k_1,k_2)} \cdot (\frac{(\rho-1)^d + 2^d}{(2\rho)^d}) \cdot volume(S) \\
\geq & \frac{1}{\max(k_1,k_2)} \cdot \zeta_d \cdot volume(S).
\end{aligned}$$

Combination of all inequalities gives

$$volume((E_1 \cup E_2) \cap S) \;\; \geq \;\; \frac{\zeta_d}{\max(k_1,k_2)} \cdot volume(S),$$

proving the inequality for all hyperspheres in $U_{E_1 \cup E_2}$ that completely contain $E_1$.

The fact that $S$ was randomly chosen from all hyperspheres in $U_{E_1 \cup E_2}$ centered in $E_1$ and the symmetry of the construction with respect to $E_1$ and $E_2$ imply that the inequality

$$volume((E_1 \cup E_2) \cap S) \geq \frac{\zeta_d}{\max(k_1,k_2)} \cdot volume(S)$$

holds for all hyperspheres $S \in U_{E_1 \cup E_2}$. □

Informally, Lemma 2.13 states that if we place two fat objects in overlapping positions, then the resulting union is not more than a constant factor less fat than the least fat of the two united objects.

Corollary 2.14 is a generalized version of the result mentioned earlier and saying that a thin object cannot be split into two (relatively) fat parts. The result of the corrolary is more general because the object is not decomposed into two subobjects but covered by two subobjects. Note that a decomposition is a restricted type of covering in which the subobjects only overlap at their boundaries. Corollary 2.14 is essentially a reformulation of Lemma 2.13 where $k_1 = k_2 = k$.

**Corollary 2.14** *Let $E \in \mathbb{R}^d$ be a closed connected object that is not $k$-fat, Any covering of $E$ by two closed connected parts results in at least one part that is not $(\zeta_d \cdot k)$-fat.*

Corollary 2.14 supplies a crucial tool for proving the main result of this section. Assume we are given an object $E$ that is not $k$-fat. Our aim is to partition it into, or cover it by, a preferably small number of $k'$-fat parts, for some $k' \leq k$, provided that it is possible to do so. Theorem 2.15 repeatedly applies Corollary 2.14 to eventually end up with $k'$-fat parts.

$r \leq r_{ES}$ In combination with the connectedness of $E_1$, the assumptions $r \leq r_{ES}$ and $m \in E_1$ yield $S \in U_{E_1}$. Using the $k_1$-fatness of $E_1$ and $0 < \zeta_d \leq 1$, we then get

$$\begin{aligned} & volume((E_1 \cup E_2) \cap S) \\ \geq \ & volume(E_1 \cap S) \\ \geq \ & \frac{1}{k_1} \cdot volume(S) \\ \geq \ & \frac{\zeta_d}{\max(k_1, k_2)} \cdot volume(S), \end{aligned}$$

proving the inequality for all hyperspheres in $U_{E_1 \cup E_2}$ that do not fully contain $E_1$ in their interior.

$r \geq r_{ES}$ In this case the hypersphere $S$ has $E_1$ completely in its interior. The assumption that $S \in U_{E_1 \cup E_2}$ then implies that the boundary of $S$ must intersect $E_2$. On the other hand, the non-emptiness of the intersection $E_1 \cap E_2$ and the connectedness of $E_2$ yield that $E_2$ must also intersect the boundary of $ES_{m,E_1}$. For the same pair of reasons, finally, $E_2$ must also intersect the boundary of the hypersphere $S_{m,(r+r_{ES})/2}$ which lies precisely halfway $\partial ES_{m,E_1}$ and $\partial S$. Now let $m' \in E_2 \cap \partial S_{m,(r+r_{ES})/2}$ and define the hypersphere $L = S_{m',(r-r_{ES})/2}$. Note that this hypersphere touches $\partial ES_{m,E_1}$ from the outside and $\partial S$ from the inside. Because $L$ and $ES_{m,E_1}$ are disjoint and both contained in $S$, we get that

$$\begin{aligned} & volume((E_1 \cup E_2) \cap S) \\ \geq \ & volume((E_1 \cup E_2) \cap ES_{m,E_1}) + volume((E_1 \cup E_2) \cap L) \\ \geq \ & volume(E_1 \cap ES_{m,E_1}) + volume(E_2 \cap L). \end{aligned}$$

$ES_{m,E_1}$ is the largest hypersphere centered at $m$ whose boundary still intersects $E_1$, so $ES_{m,E_1} \in U_{E_1}$. The hypersphere $L$ has its center $m'$ in $E_2$ and, in addition, its boundary $\partial L$ is intersected by $E_2$ because $E_2$ intersects the boundaries $\partial ES_{m,E_1}$ and $\partial S$ which, in turn, both have non-empty intersection with the interior of $L$. Hence, $L \in U_{E_2}$. The memberships $ES_{m,E_1} \in U_{E_1}$ and $L \in U_{E_2}$ combined with the $k_1$-fatness of $E_1$ and the $k_2$-fatness of $E_2$ result in

$$\begin{aligned} & volume(E_1 \cap ES_{m,E_1}) + volume(E_2 \cap L) \\ \geq \ & \frac{1}{k_1} \cdot volume(ES_{m,E_1}) + \frac{1}{k_2} \cdot volume(L) \\ \geq \ & \frac{1}{\max(k_1, k_2)} \cdot (volume(ES_{m,E_1}) + volume(L)) \\ \geq \ & \frac{1}{\max(k_1, k_2)} \cdot ((\frac{r_{ES}}{r})^d \cdot volume(S) + (\frac{(r - r_{ES})/2}{r})^d \cdot volume(S)). \end{aligned}$$

The latter expression is simplified considerably by the definition $\rho := r/r_{ES}$. Note that $\rho \geq 1$ by the assumption $r \geq r_{ES}$. Substitution of $\rho$ and subsequent

Partitioning a thin object into two fat objects, however, is impossible. We shall prove that any split of an object $E$ that is not $n$-fat (for some large $n$) results in two parts of which one is not $(cn)$-fat, for some (dimension-dependent) constant $c$ between 0 and 1. In fact, we even give a more general result stating that the union of two intersecting objects cannot be more than a constant factor less fat, or thinner, than the least fat of its two constituents. This statement does not appear strange at all: if we superimpose two objects, the result does not seem to be a lot thinner than the original objects. Before we prove the corresponding lemma, we first define a dimension-dependent constant $\zeta_d$. The constant plays a role in the upcoming lemma:

$$\zeta_d = \min\{f(\rho) | \rho \geq 1\},$$

where

$$\begin{aligned} f(\rho) &= \frac{1}{\rho^d} + \frac{(\frac{\rho-1}{2})^d}{\rho^d} \\ &= \frac{(\rho-1)^d + 2^d}{(2\rho)^d}. \end{aligned}$$

An analysis of the function $f$ learns that it has a single minimum for $\rho \in [1, \infty)$; this minimum is reached at $\rho = 2^{\frac{d}{d-1}} + 1$, hence:

$$\zeta_d = f(2^{\frac{d}{d-1}} + 1) = (2^{\frac{d}{d-1}} + 1)^{1-d}.$$

Example values are $\zeta_2 = 1/5$ and $\zeta_3 = (9 - 4\sqrt{2})/49 \approx 0.068$.

Now we are ready to formulate the lemma concerning the fatness of the union of two intersecting fat objects.

**Lemma 2.13** *Let $E_1 \subseteq \mathbb{R}^d$ be a closed connected $k_1$-fat object and let $E_2 \subseteq \mathbb{R}^d$ be a closed connected $k_2$-fat object such that $E_1 \cap E_2 \neq \emptyset$. Then the union $E_1 \cup E_2$ is $(\zeta_d^{-1} \cdot \max(k_1, k_2))$-fat.*

**Proof:** The proof obligation is that

$$\forall S \in U_{E_1 \cup E_2} \qquad \frac{\max(k_1, k_2)}{\zeta_d} \cdot volume((E_1 \cup E_2) \cap S) \geq volume(S),$$

or

$$\forall S \in U_{E_1 \cup E_2} \qquad volume((E_1 \cup E_2) \cap S) \geq \frac{\zeta_d}{\max(k_1, k_2)} \cdot volume(S).$$

Let us consider a randomly chosen $S = S_{m,r} \in U_{E_1 \cup E_2}$. By definition, the center $m$ must lie in at least one of $E_1$ and $E_2$. Assume without loss of generality, that $m \in E_1$ and recall that $ES_{m,E_1}$ is the $E_1$-enclosing hypersphere centered at $m$. Define $r_{ES}$ to be the radius of $ES_{m,E_1}$, so $ES_{m,E_1} = S_{m,r_{ES}}$. We distinguish two different cases: $r \leq r_{ES}$ or $r \geq r_{ES}$ and analyze them separately, starting with the first, and easiest, one.

mentioned in the first paragraph.

Before we move on to examine the effect of splitting, we must first find an appropriate way of expressing the bound on the lack of fatness, or 'thinness', of an object, as the observations include thin objects as well as fat objects. A problem lies in the fact that the remark that an object is $k$-fat does not say exactly how fat the object is, but only that the object is not less fat, or thinner, than $k$-fat. On the opposite, we would now also like to have a means of expressing that an object is not less thin, or fatter, than a certain amount. Similarly, saying that an object is $1,000,000$-fat does not necessarily mean that it is very thin, because the object might still be 10-fat as well. Fortunately, the negation does supply a bound on the extent to which an object is fat. The fact that an object is *not* $k$-fat tells us that it is definitely not less thin or fatter than $k$-fat. Hence, the negation of fatness supplies an appropriate means of expressing a certain guaranteed amount of thinness, or lack of fatness.

A simple example shows that it is possible to partition a very fat object into two unboundedly thin subobjects. Take the (4-fat) circle $E$ of Figure 2.7. The circle is

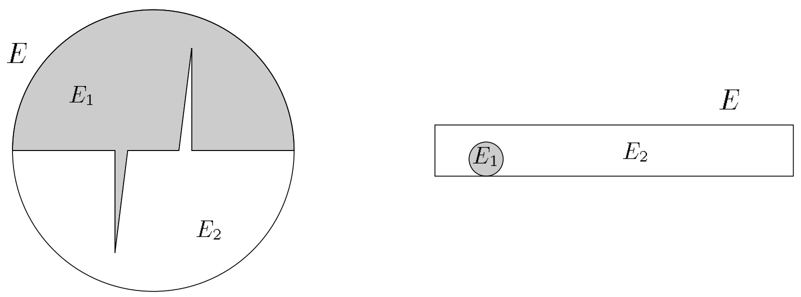


Figure 2.7: The left 4-fat circle $E$ is split into two arbitrarily thin objects $E_1$ and $E_2$. The right relatively thin object $E$ is split into a 4-fat object $E_1$ and another object $E_2$.

split equivalent parts, each one being half a circle with a thin needle sticking into the opposite half. The needle can be made as thin as one likes, resulting in extremely thin subobjects of the circle $S$. The example straightforwardly generalizes to higher dimensions. Obviously, it is also possible to partition a very fat objects into two parts of which exactly one is unboundedly thin.

A second observation is that a very thin object can be split into two parts of which one is extremely fat. Consider the second example in Figure 2.7 of a rectangle $E$ in 2D with very large aspect ratio, i.e., the ratio of its side lengths. The partitioning into two parts of which one is extremely fat can be obtained by simply cutting an (arbitrarily) small 4-fat circle $E_1$ out of the rectangle. Again, the 2D example is straightforwardly generalizable.

smallest object for which the point $p \in \mathbb{R}^d$ lies inside the wrapping $\Delta(E_i)$. Since there is only a constant number of wrappings of larger objects intersecting $E_i$'s wrapping, the point $p$ can be in no more than a constant number of additional wrappings. □

Besides applications in motion planning that become clear later in this thesis, Theorem 2.12 has interesting implications for complexities of union boundaries of certain geometric figures. The relation between the complexity of an arrangement of wrapping boundaries and the complexity of the boundary of the union of the wrappings becomes clear if one realizes that the faces of the union boundary form a subset of the faces of the arrangement of wrapping boundaries. So, under the circumstances sketched in Theorem 2.12, the boundary complexity of $\cup_{E \in \mathcal{E}} \Delta(E)$ is $O(n)$. An alternative, more or less inverse, informal formulation of the result is the following. The boundary complexity of the union of (intersecting) constant-complexity objects is linear in the number of objects if $k$-fat sub-objects can be identified in all objects, such that the sub-objects are mutually non-intersecting and not more than some bounded amount smaller than the original objects. The bounded amount must be proportional to the size of the smallest object. As an example, the ideas are applicable to the molecule model in the paper by Halperin and Overmars [42]. The atoms that constitute a molecule are assumed to satisfy the hard sphere model. The hard sphere model describes atoms by spheres and forbids any sphere center to penetrate another sphere too far. This property makes it possible to regard the atoms as wrappings of certain non-intersecting smaller spheres, which are only a bounded amount smaller than the original atoms. The construction provides an alternative proof for the linear (in the number of atoms) descriptional complexity of the molecule surface.

## 2.4 Assembling and disassembling fat objects

The objective in this section is to show that it takes more than a constant number of cuts to partition a thin object into fat parts. In fact, we show that the minimum number of parts that is needed to cut up a thin object into fat subobjects is at least logarithmically dependent on a measure of the lack of fatness of the object. The main impact of this result is that it is impossible to extend the results in the remainder of this thesis for fat objects to thin objects by simply partitioning the thin object into fat objects without increasing the total number of objects.

To get to the above result we study the implications for fatness of splitting an object into two subobjects. We subsequently observe that a very fat object can be split into two extremely thin objects, that a thin object can be split into two object of which one can be very fat, and finally, that at least one of the parts resulting from splitting a thin object cannot be more than a constant factor fatter (or less thin) than the original object. The latter result forms the basis of the main result

**Proof:** To prove the (a)-part, let us assume that the objects in $\mathcal{E}$ are ordered by increasing size: $E_1, \ldots, E_n$, and that $\rho_1, \ldots, \rho_n$ are the corresponding minimal enclosing hypersphere radii, so $\rho \leq \rho_1 \leq \ldots \leq \rho_n$. We intend to count for each object $E_i$ the subspaces of dimensions 0 to $d-1$ that are defined by the intersection of $\partial\Delta(E_i)$ and wrapping boundaries $\partial\Delta(E_j)$ with $j > i$. A $(c \cdot \rho)$-wrapping boundary $\partial\Delta(E_i)$ can only be intersected by $(c \cdot \rho)$-wrapping boundaries $\partial\Delta(E_j)$ $(i < j)$ if the distance from $E_i$ to $E_j$ does not exceed $2c \cdot \rho \leq 2c \cdot \rho_i$. (See Figure 2.6 for a 2D example of intersecting wrappings.) Application of Corollary 2.10 yields that

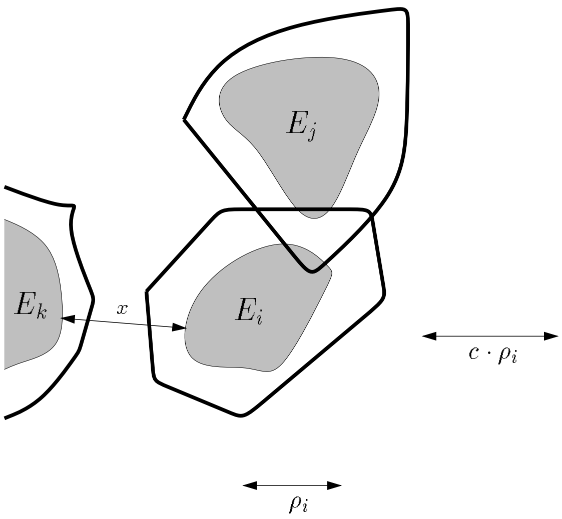


Figure 2.6: The bold lines are the boundaries of $(c \cdot \rho)$-wrappings, and (because $\rho \leq \rho_i$) also $(c \cdot \rho_i)$-wrappings, of the objects $E_i$, $E_j$, and $E_k$. The set of objects with wrapping boundaries that intersect the wrapping boundary of $E_i$ is a subset of the object within a distance $2c \cdot \rho \leq 2c \cdot \rho_i$ from $E_i$. Although $E_k$ lies a distance $x \leq 2c \cdot \rho_i$ from $E_i$, $E_k$'s wrapping (boundary) does not intersect $E_i$'s wrapping (boundary).

there can only be a constant number of such $E_j$'s within a distance $2c \cdot \rho_i$ from $E_i$, so there is at most a constant number of wrapping boundaries $\partial\Delta(E_j)$ $(j > i)$ that intersect $\partial\Delta(E_i)$. By the additional assumption that all wrappings have constant complexity, there is only a constant number of subspaces of dimension between 0 and $d-1$ defined by the intersection of $\partial\Delta(E_i)$ and wrapping boundaries $\partial\Delta(E_j)$ $(j > i)$. Adding the contributions of all wrappings amounts to a total of $O(n)$ subspaces of dimensions 0 to $d-1$ in the arrangement $\mathcal{A}(\Delta)$. The linear bounds on the number of these subspaces imply the same bound of $O(n)$ on the number of $d$-faces in $\mathcal{A}(\Delta)$, making the total combinatorial complexity of the arrangement $O(n)$.

The (b)-part follows immediately from the proof of the (a)-part. Let $E_i$ be the

### 2.3.2 Arrangements of fat object wrappings

This subsection studies the complexity of the arrangement of boundaries of expanded fat objects. Clearly, the arrangement of the boundaries of $n$ non-intersecting constant-complexity objects has $O(n)$ complexity. Let us see what happens if the objects are expanded. While expanding the fat objects, each of the boundaries will eventually start intersecting other boundaries. Intuitively, the first boundaries that are to be intersected belong to neighboring objects. As there is only a constant number of objects closeby, the contribution of each boundary to the complexity of the arrangement of boundaries does, again intuitively, not increase asymptotically as it starts intersecting the boundaries of these objects. Below, these informal ideas are made concrete by giving accurate bounds on the expansion of the $k$-fat objects such that the combinatorial complexity of the arrangement of the boundaries of these (intersecting) expansions equals $O(n)$. The so-called $\epsilon$-wrappings that are introduced first provide a convenient means of expressing the expansion of an object.

Sufficiently tight wrappings of fat objects play a crucial role in providing the justification that the paradigm for motion planning amidst fat obstacles presented in Chapter 6 indeed works. Besides that, the wrappings also help in finding efficient instances of the paradigm, for specific classes of motion planning problems. Moreover, the theorem on wrappings that we prove below is interesting in its own right, as it implies nice complexity bounds for certain arrangements and for the boundary of the union of specific families of shapes.

**Definition 2.11** [**$\epsilon$-wrapping**]
*Let $E \subseteq \mathbb{R}^d$ and let $\epsilon \in \mathbb{R}^+$. Any object $\Delta \supseteq E$ satisfying $d(p, E) \leq \epsilon$ for all $p \in \Delta$ is an $\epsilon$-wrapping of $E$.*

An $\epsilon$-wrapping of an object $E$ is an enclosing shape of $E$, with the property that the distance from the wrapping to $E$ never exceeds $\epsilon$.

Theorem 2.12 states the circumstances that lead to a linear complexity arrangement of expanded fat object boundaries. An obvious way to express a bound on the expansion of an object $E$ is to state that the expanded object is some $\epsilon$-wrapping of the object $E$ itself, for some bounded positive $\epsilon$. Note that the object expansions need not necessarily be fat.

**Theorem 2.12** *Let $k \geq 1$ and $c \geq 0$ be constants and let $\mathcal{E}$ be a set of $n$ non-intersecting $k$-fat objects in $\mathbb{R}^d$ with minimal enclosing hypersphere radii at least $\rho$. Assume that a constant-complexity $(c \cdot \rho)$-wrapping $\Delta(E)$ is given for every object $E \in \mathcal{E}$. Then:*

- **(a)** *the complexity of the arrangement $\mathcal{A}(\Delta)$ of all wrapping boundaries $\partial\Delta(E)$ is $O(n)$,*
- **(b)** *every point $p \in \mathbb{R}^d$ lies inside at most $O(1)$ wrappings $\Delta(E)$.*

The combination of $volume(T) = \omega_d \cdot (c+1)^d \cdot \rho^d$ and $volume(E \cap T) \geq k^{-1} \cdot \omega_d \cdot \rho^d$ for any $E$ intersecting $R$, results in an upper bound of $k \cdot (c+1)^d$ on the number of objects $E$ intersecting $R$. $\square$

Informally, the theorem states that the number of $k$-fat objects intersecting a region, that is not too large compared to the objects, is constant. The weakness of the notion of thickness, as defined in the previous section, lies in the impossibility to deduce a similar property for scenes of such objects. This impossibility is illustrated by the two-dimensional example of Figure 2.5, where an extremely small rectangular region is intersected by $n$ very thick, namely 4-thick, objects. In a motion planning context, this would imply that even an extremely small robot is able to touch many obstacles simultaneously, which potentially leads to a high complexity free space.

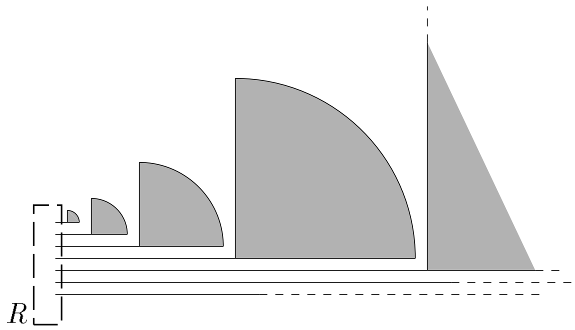


Figure 2.5: The small rectangular region $R$ intersects $n$ 4-thick objects.

The following alternative formulation of the low density property in terms of distances can be given. It bounds the number of larger $k$-fat objects that can lie close to a given $k$-fat object.

**Corollary 2.10** *Let $k \geq 1$ and $c \geq 0$ be constants and let $\mathcal{E}$ be a set of non-intersecting $k$-fat objects in $\mathbb{R}^d$. Let $E \in \mathcal{E}$ be an object with minimal enclosing hypersphere radius $\rho$. Then the number of object $E' \in \mathcal{E}$ with larger minimal enclosing hypersphere radii within a distance $c \cdot \rho$ from $E$ is bounded by the constant $k \cdot (c+2)^d$.*

**Proof:** Any object $E'$ within a distance $c \cdot \rho$ from $E$ must also lie within the same distance $c \cdot \rho$ from the minimal enclosing hypersphere $MES_E$ of $E$, which has radius $\rho$. Necessarily, such an object $E'$ must then intersect the region bounded by the hypersphere concentric to $MES_E$ but at a distance $c \cdot \rho$ from $MES_E$, hence with radius $(c+1) \cdot \rho$. So, application of Theorem 2.9 with $R$ chosen to be the region bounded by the concentric hypersphere, with mes-radius $(c+1) \cdot \rho$ yields the claimed result. $\square$

The choice for the minimal enclosing hypersphere as a measure of size, implies that if we say that 'an object $X$ is larger than another object $E$' we implicitly mean to say that 'the (radius of the) minimal enclosing hypersphere $MES_X$ of $X$ is larger than the (radius of the) minimal enclosing hypersphere $MES_E$ of $E$'. Similarly, 'the smallest object $E$' means 'the object with the smallest (radius) minimal enclosing hyperspere $MES_E$'.

**Definition 2.8 [minimal enclosing hypersphere (or mes-) radius]** *The minimal enclosing hypersphere radius, or mes-radius, of an object $X$ is the radius of the minimal enclosing hypersphere of the object $X$.*

Theorem 2.9 states the low object density property for scenes of non-intersecting $k$-fat objects.

**Theorem 2.9** *Let $k \geq 1$ and $c \geq 0$ be constants and let $\mathcal{E}$ be a set of non-intersecting $k$-fat objects in $\mathbb{R}^d$ with minimal enclosing hypersphere radii at least $\rho$. Then the number of objects $E \in \mathcal{E}$ intersecting any region $R$ with minimal enclosing hypersphere radius $c \cdot \rho$ is bounded by the constant $k \cdot (c+1)^d$.*

**Proof:** The approach is to identify a region $T$ with bounded volume such that each object $E$ intersecting $R$ has a certain minimum volume inside the region $T$. The combination of the volume of $T$ and the lower bound on the volume of $E \cap T$ results in a bound on the number of objects $E$ that intersect $R$.

Define $T = MES_R \ominus S_{O,\rho}$, the Minkowski difference of the minimal enclosing hypersphere $MES_R$ of $R$ and the hypersphere with radius $\rho$, centered at the origin $O$. The radius of the hypersphere $T$ equals $c \cdot \rho + \rho = (c+1) \cdot \rho$, which implies that the volume of $T$ is

$$volume(T) = \omega_d \cdot ((c+1) \cdot \rho)^d = \omega_d \cdot (c+1)^d \cdot \rho^d,$$

where $\omega_d$ is the dimension-dependent multiplier for hypersphere volumes.

Now consider an object $E$ intersecting the region $R$. Let $m$ be a point in the non-empty intersection $E \cap R$. The point $m$ lies inside $E$ so it definitely lies in the minimal enclosing hypersphere $MES_E$ as well. As a consequence, the hypersphere $S_{m,\rho}$ is completely contained in $T = MES_R \ominus S_{O,\rho}$. In order to be able to give a lower bound on the volume of $E$ lying in $S_{m,\rho}$ and hence in $T$, we show that $S_{m,\rho} \in U_E$. The object $E$ has non-empty intersection with $S_{m,\rho}$ because $m \in E$. Moreover, the object $E$ cannot lie entirely in the interior of $S_{m,\rho}$ as this would contradict the assumption that the minimal enclosing hypersphere of $E$ has radius at least $\rho$. So, the boundary of $S_{m,\rho}$ is intersected by $E$ and therefore $S_{m,\rho} \in U_E$. From $S_{m,\rho} \in U_E$ and the containment of $S_{m,\rho}$ in $T$ it follows that

$$volume(E \cap T) \geq volume(E \cap S_{m,\rho}) \geq \frac{1}{k} \cdot volume(S_{m,\rho}) = k^{-1} \cdot \omega_d \cdot \rho^d.$$

The problem of computing the fatness for any class of objects beyond convex shapes remains open. A relaxed version of the problem, aimed at finding an upper bound on the fatness that is not more than a constant multiple of the 'real' fatness (that is, the maximum ratio $volume(S)/volume(E \cap S)$), is also largely unsolved.

## 2.3 Properties of scenes of fat objects

In this section we prove two important results for scenes of fat objects. The results form the basis of many proofs throughout this thesis. In the first subsection we show that a scene of non-intersecting $k$-fat objects satisfies a certain low density property, saying that the number of objects within a 'neighborhood' is at most constant. The exact interpretation of 'neighborhood' is shown to be dependent on the sizes of the objects that are involved. The low density property resembles the notion of *bounded local complexity* introduced by Schwartz and Sharir in [88].

The complexity of the arrangement of the boundaries of $n$ disjoint constant-complexity objects is clearly $O(n)$. If we expand the objects in some way then they will start intersecting, and eventually the asymptotic combinatorial complexity of the arrangement will increase. In the case of general objects, a drastic increase of the complexity can occur soon after the expansion has started. We may expect, on the grounds of the low density property of the original disjoint objects, that such a sudden increase does not take place if the objects are fat: one gets the feeling that an expansion by some bounded amount does not increase the asymptotic complexity of the arrangement. The second subsection provides the circumstances that do indeed lead to this result. The results have immediate consequences for complexities of union boundaries, like those in [58] and [67].

### 2.3.1 Fatness implies low density

This subsection discusses a certain low object density property implied by the fatness of the objects under consideration. The property has a large impact in the rest of this thesis, as many proofs apply it in some form. The result can be paraphrased in many different, but essentially similar, ways. We decide to give two alternative formulations, to save ourselves from repeatedly deducing either one of the two from the other one in the future. As remarked earlier, the result, like many others in this thesis, includes a notion of neighborhood depending on the size of the objects under consideration. Therefore, we shall first introduce a convenient way to express the size of an object.

Clearly, there are many ways to express a bound on the size of an object. The size, that is, the radius, of the minimal enclosing hypersphere of an object is felt to be the most convenient measure for our purposes. The size of the minimal enclosing hypersphere can be seen to relate closely to other measures of the size of the fat object, like the diameter of the object, and, because of the fatness, also the volume.

Informally, the inequality says that the largest of the hyperspheres from $U_E$ centered at $m$ is the emptiest among all such hyperspheres. As a result, the relatively emptiest hypersphere belongs to the set $ES_E \subseteq U_E$ of enclosing hyperspheres, so

$$\begin{aligned}\max\{f_E(S)|S \in U_E\} &= \max\{f_E(S)|S \in ES_E\} \\ &= \max\{\frac{volume(S)}{volume(E \cap S)}|S \in ES_E\}.\end{aligned}$$

Each hypersphere in $ES_E$ fully encloses the convex object $E$, hence $E \cap S = E$ for all $S \in ES_E$. This identity and the assumption $volume(E) = V$ allow for the following reformulation.

$$\max\{f_E(S)|S \in U_E\} = \max\{\frac{volume(S)}{V}|S \in ES_E\}$$

The constant denominator $V$ of the fraction in the right-hand side of the equality justifies the conclusion that the maximum fraction is obtained when the numerator $volume(S)$ is chosen as large as possible. Therefore, the largest hypersphere in $ES_E$ is the relatively emptiest hypersphere in $U_E$. Since the diameter of $E$ equals $\delta$, the maximum distance between any pair of points in $E$ is $\delta$. Let $p, q \in E$ be such that the distance from $p$ to $q$ is $\delta$. Then $S_{p,\delta} \in ES_E$ and $S_{q,\delta} \in ES_E$ because $q \in \partial S_{p,\delta}$ and $p \in \partial S_{q,\delta}$ respectively. For obvious reasons, the set $ES_E$ contains no hyperspheres with radii larger than $\delta$. Combining these considerations with $volume(S_{p,\delta}) = volume(S_{q,\delta}) = \omega_d \cdot \delta^d$ yields $\max\{volume(S)|S \in ES_E\} = \omega_d \cdot \delta^d$, and thus

$$F(E) = \max\{f_E(S)|S \in U_E\} = \frac{\omega_d \cdot \delta^d}{V}.$$

This leads to the following theorem on the fatness of convex objects.

**Theorem 2.7** *Let $E \subseteq \mathbb{R}^d$ be a closed convex object with volume $V$ and diameter $\delta$. Then $E$ is $(\omega_d \cdot V^{-1} \cdot \delta^d)$-fat.*

The problem of maximizing the ratio $f_E(S) = volume(S)/volume(E \cap S)$ for general $E$ or, less ambitious, for different classes of $E$ (like polytopes) is very hard. The difficulty lies both in the shape and (implicit) dimension of the domain of $f_E$ and in the analytic form of $f_E$. The continuous domain $U_E$ of hyperspheres of the function $f_E$ can be seen as a subset of the $(d+1)$-dimensional Cartesian product of the $d$-dimensional space of hypersphere centers $m = (m_1, \ldots, m_d) \in \mathbb{R}^d$ and the one-dimensional space of radii $r \in \mathbb{R}^+$. The complex shape of the domain, constrained by the two dependent expressions $m \in E$ and $\partial S_{m,r} \cap E \neq \emptyset$, contributes to the difficulty of the problem. Another complicating factor is that the (analytical) description of $f_E(S_{m,r})$ in terms of $m = (m_1, \ldots, m_d)$ and $r$ is not unique throughout the entire domain of hyperspheres, due to the changing topology of the intersection of $S_{m,r}$ and $E$.

## 2.2 Computing the fatness of an object

Section 2.1 provides a definition of $k$-fatness, stating when an object is $k$-fat. The fact that an object is $k$-fat however does not give a clue on *how* fat the object really is, due to the property that a $k$-fat object is also $k'$-fat, for all $k' \geq k$. The minimum $k$ for which $E$ is $k$-fat provides a realistic qualitative measure for $E$'s fatness. Let $F(E)$ be this minimum, so

$$F(E) = \min\{k | E \text{ is } k - \text{fat}\}.$$

We will occasionally refer to $F(E)$ as 'the fatness of $E$'. Substitution of the definition of $k$-fatness in the definition given above yields the following formulation for $F(E)$:

$$\begin{aligned} F(E) &= \min\{k | \forall_{S \in U_E} k \cdot \mathit{volume}(E \cap S) \geq \mathit{volume}(S)\} \\ &= \min\{k | \forall_{S \in U_E} \frac{\mathit{volume}(S)}{\mathit{volume}(E \cap S)} \leq k\} \\ &= \min\{k | \max\{\frac{\mathit{volume}(S)}{\mathit{volume}(E \cap S)} | S \in U_E\} \leq k\} \\ &= \max\{\frac{\mathit{volume}(S)}{\mathit{volume}(E \cap S)} | S \in U_E\}. \end{aligned}$$

The equation shows that the minimum $k$ for which $E$ is $k$-fat is achieved by the hypersphere $S \in U_E$ that maximizes the ratio $f_E(S) = \mathit{volume}(S)/\mathit{volume}(E \cap S)$. Informally, this hypersphere $S$ is the relatively emptiest among the hyperspheres of $U_E$. Figure 2.3 shows that the relatively emptiest hyperspheres for the objects $E_1, \ldots, E_6$ are very different.

We will now derive an explicit formula for $F(E)$ in the case that $E \subseteq \mathbb{R}^d$ is an arbitrary convex shape with volume $V$ and diameter $\delta$. We were unable to express the fatness $F(E)$ of an object $E$ as a function of parameters that are related to the shape of $E$, and to characterize the corresponding relatively emptiest hypersphere(s) in $U_E$, for non-convex objects.

The basis for an explicit fatness formula for convex objects $E$ lies in Lemma 2.5, which, after minor manipulations, states that the inequality

$$f_E(S_{m,R}) = \frac{\mathit{volume}(S_{m,R})}{\mathit{volume}(E \cap S_{m,R})} \geq \frac{\mathit{volume}(S_{m,r})}{\mathit{volume}(E \cap S_{m,r})} = f_E(S_{m,r})$$

holds for any pair $S_{m,r}, S_{m,R} \in U_E$ with $r \leq R$. Like in Section 2.1, we abbreviate the largest member of $U_E$ centered at $m$ to $ES_{m,E}$. Notice that $ES_{m,E}$ is the enclosing hypersphere of $E$ centered at $m$. All enclosing hyperspheres $ES_{m,E}$ centered at some $m \in E$ are collected in a set $ES_E$. A consequence of the above inequality is that for all $S_{m,r} \in U_E$:

$$f_E(ES_{m,E}) \geq f_E(S_{m,r}).$$

$E$ is $l$-thick $\Rightarrow$ $E$ is $l'$-fat:

The convex object $E$ is $l$-thick, so $l \cdot volume(E) \geq volume(MES_E)$. By Lemma 2.5 and the convexity of $E$ we know that it suffices to prove that $\forall S \in ES_E : \quad l' \cdot volume(E) \geq volume(S)$, for some constant $l'$. Let $\delta$ be the diameter of $MES_E$ and let $\epsilon$ be the diameter of the object $E$. The obstacle $E$ fits inside $MES_E$ so trivially $\epsilon \leq \delta$. The diameter of the object $E$ is determined by two points $m$ and $m'$ on its boundary. The radius of a hyperspherical region in $ES_E$ is at most $\epsilon$. This is the radius of the largest regions $ES_{m,E}$ and $ES_{m',E}$.

We have $volume(MES_E) = \omega_d \cdot (\delta/2)^d$ and for all $S \in ES_E$: $volume(S) \leq \omega_d \cdot \epsilon^d$, where $\omega_d$ is the dimension-dependent multiplication factor mentioned earlier in this section. Combination of all equalities and inequalities yields for all $S \in ES_E$:

$$\begin{aligned} & 2^d \cdot l \cdot volume(E) \\ \geq\ & 2^d \cdot volume(MES_E) \\ =\ & \omega_d \cdot \delta^d \\ \geq\ & \omega_d \cdot \epsilon^d \\ \geq\ & volume(S), \end{aligned}$$

proving $l'$-fatness of the convex object $E$, with $l' = 2^d \cdot l$. □

Fatness and thickness are definitely not equivalent for non-convex objects as can be concluded from the object $E_1$ in Figure 2.3 which is not fat but very thick.

A consequence of Theorem 2.6 is that the complexity results that we prove for convex objects that are $k$-fat also hold for convex objects that are $k'$-thick. In the sequel, we will only consider fatness, not thickness.

The notion of fatness proposed in this section also relates to most of the other notions summarized in the introductory part of this thesis, for the specific classes of objects to which these other notions apply. A $\delta$-fat triangle [5, 67] is also fat according to our definition. Assume that we are given a $\delta$-fat triangle with a longest edge $e$. The triangle has minimum area if the other two angles have magnitudes $\delta$ and $\pi - 2\delta$. This minimal area is $(|e|^2 \tan \delta)/4$. Using a result from Chapter 2.2 on the fatness of convex objects, we find that this triangle is $4\pi/(\tan \delta)$-fat according to our fatness definition. Maximum fatness is achieved for equilateral triangles and there is no fatness if $\delta = 0$. The latter triangle will also be non-fat in [67]. Furthermore, it is easily verified that a $\delta$-fat wedge is $(2\pi/\delta)$-fat and a $\delta$-fat double wedge is $(\pi/\delta)$-fat. Van Kreveld's notion of wideness for polygons [58] is related for $c$-gons only, where $c$ is some constant. It is, however, rather difficult to determine a relation between the fatness and wideness of a $c$-gon.

3. If $R \leq \rho_E(\phi, \theta_1, \ldots, \theta_{d-2})$ then
   $f(\phi, \theta_1, \ldots, \theta_{d-2}) = 1 = F(\phi, \theta_1, \ldots, \theta_{d-2})$.

Figure 2.4 shows a two-dimensional example of each of the three cases given above. Combining the three different ranges, we obtain

$$f(\phi, \theta_1, \ldots, \theta_{d-2}) \geq F(\phi, \theta_1, \ldots, \theta_{d-2}),$$

for all $0 \leq \phi < 2\pi$ and $0 \leq \theta_1, \ldots, \theta_{d-2} \leq \pi$. □

Lemma 2.5 shows that in each set $U_{m,E}$ the portion of a hyperspherical region that is covered by the object $E$ does not increase as the radius of the hyperspherical region increases. The ratio is therefore minimal for the region $ES_{m,E} \in U_{m,E}$ with maximal volume, which is the enclosing hypersphere of $E$ centered at $m$. The region $ES_{m,E}$ is uniquely defined by following expression

$$ES_{m,E} \in U_{m,E} \quad \wedge \quad \forall_{S \in U_{m,E}} S \subseteq ES_{m,E}.$$

A consequence of Lemma 2.5 is that if $k \cdot volume(E \cap ES_{m,E}) \geq volume(ES_{m,E})$ holds then we can conclude that $k \cdot volume(E \cap S) \geq volume(S)$ for all $S \in U_{m,E}$. Define the set $ES_E$ of all enclosing hyperspherical regions centered at some point in the object:

$$ES_E = \{ES_{m,E} | m \in E\}.$$

It is clear that $ES_E \subseteq U_E$. Lemma 2.5 makes a simplification of the condition in Definition 2.2 possible for convex objects. Note that for all $S \in ES_E$, the obvious equality $E \cap S = E$ holds. Hence, a convex object $E$ is $k$-fat if:

$$\forall S \in ES_E \qquad k \cdot volume(E) \geq volume(S).$$

The preceding lemma and considerations provide useful tools in the proof of the equivalence of thickness and fatness for convex objects.

**Theorem 2.6** *Let $E \subseteq \mathbb{R}^d$ be a convex object. Then*

$$E \text{ is } k\text{-fat} \Rightarrow E \text{ is } k'\text{-thick} \qquad \wedge \qquad E \text{ is } l\text{-thick} \Rightarrow E \text{ is } l'\text{-fat},$$

*with $k' = c \cdot k$ and $l' = c' \cdot l$, for some constants $c$ and $c'$.*

**Proof:**
$E$ is $k$-fat $\Rightarrow$ $E$ is $k'$-thick:

> Choose some hyperspherical region $S \in ES_E$. The object $E$ is $k$-fat and $ES_E \subseteq U_E$, so $k \cdot volume(E) = k \cdot volume(E \cap S) \geq volume(S)$. Region $S$ is some enclosing hyperspherical region of $E$ and $MES_E$ is defined as the minimal volume enclosing hyperspherical region of $E$, so obviously $volume(MES_E) \leq volume(S)$ holds. Combining both inequalities results in $k \cdot volume(E) \geq volume(MES_E)$, proving $k'$-thickness of $E$, with $k' = k$.

the full angular domain. The right-hand side $volume(E \cap S_{m,R})/volume(S_{m,R})$ is obtained by integrating the product of the same $\Phi$ and the function $F$ to the power $p$ over the same domain. The determinant $\Phi$ is a product of $(\sin\theta_i)^j$-terms. Since $0 \le \theta_1, \dots, \theta_{d-2} \le \pi$, function $\Phi$'s range is restricted to $[0,1]$. Functions $f$ and $F$ have the same range. If we can prove that $f(\phi, \theta_1, \dots, \theta_{d-2}) \ge F(\phi, \theta_1, \dots, \theta_{d-2})$, for all $0 \le \phi < 2\pi$ and $0 \le \theta_1, \dots, \theta_{d-2} \le \pi$, then, because $\Phi$, $f$, and $F$ only have non-negative function values, the integral containing $f$ will yield a larger value than the one containing $F$, and hence the inequality involving the volumes will be proved.

Relevant changes in the values of $f$ and $F$ appear at $\rho_E(\phi, \theta_1, \dots, \theta_{d-2}) = r$ and $\rho_E(\phi, \theta_1, \dots, \theta_{d-2}) = R$. Therefore, we consider three different ranges for the value

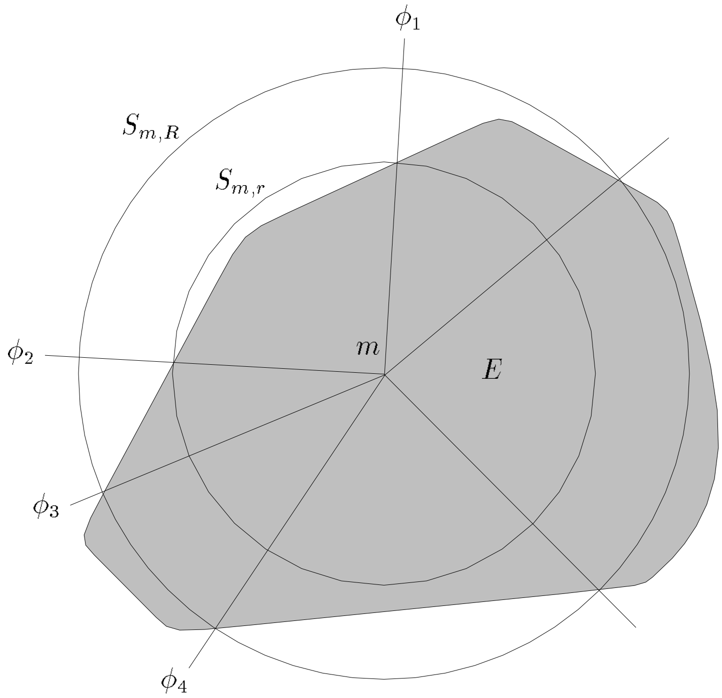


Figure 2.4: The angular interval $[\phi_1, \phi_2]$ is an example of case (1), interval $[\phi_2, \phi_3]$ is an example of case (2), and the angular interval $[\phi_3, \phi_4]$ is an example of case (3).

of $\rho_E(\phi, \theta_1, \dots, \theta_{d-2})$.

1. If $\rho_E(\phi, \theta_1, \dots, \theta_{d-2}) \le r$ then
   $f(\phi, \theta_1, \dots, \theta_{d-2}) = \rho_E(\phi, \theta_1, \dots, \theta_{d-2})/r$
   $\ge \rho_E(\phi, \theta_1, \dots, \theta_{d-2})/R = F(\phi, \theta_1, \dots, \theta_{d-2})$.

2. If $r \le \rho_E(\phi, \theta_1, \dots, \theta_{d-2}) \le R$ then
   $f(\phi, \theta_1, \dots, \theta_{d-2}) = 1 \ge \rho_E(\phi, \theta_1, \dots, \theta_{d-2})/R = F(\phi, \theta_1, \dots, \theta_{d-2})$.

**Definition 2.4** [$k$-**thick**]
*Let $E \subseteq \mathbb{R}^d$ be an object and let $k \geq 1$ be a constant. The object $E$ is $k$-thick if:*

$$k \cdot volume(E) \geq volume(MES_E).$$

The definition of $k$-thickness involves just one hypersphere instead of infinitely many. Note that not necessarily $MES_E \in U_E$: the minimal enclosing hypersphere of an object can have its center outside the object. Again we have the straightforward property that an object that is $k$-thick is also $k'$-thick for $k' \geq k$. Spherical objects are 1-thick, because the minimal enclosing hyperspheres of such objects are the objects themselves.

Even though the notion of $k$-thickness seems more natural, it is not very useful for our purposes because it does not result in low complexities of the free space, due to the impossibility to prove a low object density property for scenes of such objects (similar to Theorem 2.9), which turns out to be the basis of the low free space complexity result presented in Chapter 4. We could restrict ourselves to convex objects but, as we will see below, in that case thickness is equivalent to fatness. Therefore, we have chosen to use the definition of fatness stated as Definition 2.2 because it also allows for non-convex objects. The property of the set $U_{m,E}$ for a convex region $E$ given in the next lemma is a useful tool in the proof of the equivalence of thickness and fatness for convex objects.

**Lemma 2.5** *Let $E \subseteq \mathbb{R}^d$ be a convex object and $m \in E$. Let $S_{m,r} \in U_{m,E}$ and $S_{m,R} \in U_{m,E}$ with $r \leq R$. Now the following inequality holds:*

$$\frac{volume(E \cap S_{m,r})}{volume(S_{m,r})} \geq \frac{volume(E \cap S_{m,R})}{volume(S_{m,R})}.$$

**Proof:** We use a polar coordinate frame with origin $m$ and angles $\phi, \theta_1, \ldots, \theta_{d-2}$, with $0 \leq \phi < 2\pi$, $0 \leq \theta_1, \ldots, \theta_{d-2} \leq \pi$. Each $(d-1)$-tuple of angles $(\phi, \theta_1, \ldots, \theta_{d-2})$ specifies a viewing direction from $m$. Since the object $E$ is convex, each point on the boundary of $E$ can be seen from $m$. Therefore, the relation between the viewing direction and the distance to the boundary of $E$ is a function. The same obviously holds for both spheres. So, there are three functions $\rho_E, \rho_{S_{m,r}}, \rho_{S_{m,R}} : [0, 2\pi) \times [0, \pi]^{d-2} \rightarrow \mathbb{R}^+ \cup \{0\}$, that give the distance from $m$ to the boundary of $E$, $S_{m,r}$, $S_{m,R}$ respectively. The latter two functions are constant: $\rho_{S_{m,r}}(\phi, \theta_1, \ldots, \theta_{d-2}) = r$ and $\rho_{S_{m,R}}(\phi, \theta_1, \ldots, \theta_{d-2}) = R$. Let $f, F : [0, 2\pi) \times [0, \pi]^{d-2} \rightarrow [0, 1]$ be defined as:

$$f(\phi, \theta_1, \ldots, \theta_{d-2}) = \min(\frac{\rho_E(\phi, \theta_1, \ldots, \theta_{d-2})}{r}, 1),$$

$$F(\phi, \theta_1, \ldots, \theta_{d-2}) = \min(\frac{\rho_E(\phi, \theta_1, \ldots, \theta_{d-2})}{R}, 1).$$

The left-hand side $volume(E \cap S_{m,r})/volume(S_{m,r})$ is obtained by integrating the product of some determinant function $\Phi$ and the function $f$ to some power $p$ over

The choice for hyperspherical regions in the definition of fatness is rather arbitrary. In fact we could have used any compact region (with non-zero volume), like hypercubic regions, regions bounded by simplices etc. Section 2.5 examines the relation between definitions of fatness with respect to different shapes. From the results presented there, it is clear that if an object is $k$-fat with respect to a given compact shape $A$, it is $k'$-fat with respect to another compact shape $B$ for some $k'$ that is only a constant multiple of $k$.

The lower bound on $k$ in the definition of fatness equals 1 in any dimension $d$. The maximal 1-fatness in dimension $d$ is achieved by the 'object' $\mathbb{R}^d$ as it covers 100 % of any hypersphere. A half-space in any dimension $d$ is 2-fat as is covers at least half of any hypersphere centered inside the half-space. Determining a lower bound on the value of $k$ is a lot more interesting if we restrict ourselves to bounded objects. Then, the lower bound differs from dimension to dimension. There are for example no (bounded) 1-fat objects at all; there can be 5-fat objects in a two-dimensional workspace but 5-fat objects in a three-dimensional workspace do not exist. Suppose we have a $k$-fat object $E$ with diameter $\delta$. The volume of this object is bounded from above by the volume of a hypersphere with diameter $\delta$ (or radius $\delta/2$). The diameter of $E$ is $\delta$, so there is a pair of points on the boundary of $E$ that are a distance $\delta$ apart; let $m, m' \in E$ be these two points. The hyperspherical region $S_{m,\delta}$ is an element of $U_E$ since $m' \in \partial S_{m,\delta}$ and $m' \in E$. (Similarly, the hyperspherical region $S_{m',\delta}$ is an element of $U_E$.) Hence, the set $U_E$ contains an element $S$ with radius $\delta$. We know that $volume(E \cap S) \leq volume(E) \leq \omega_d \cdot (\delta/2)^d$ and $volume(S) = \omega_d \cdot \delta^d$, where $\omega_d$ is the dimension-dependent multiplier in the volume formulae for hyperspheres[1]. Combination with Definition 2.2 ($E$ is $k$-fat and $S \in U_E$) yields $k \geq 2^d$. The boundary value $2^d$-fatness is only obtained for hyperspherical objects; hyperspherical objects have maximal fatness among the bounded objects.

The definition of $k$-fatness has a 'local' character: a certain portion of the proximity of every point in the object must be covered by the object as well. As stated before, this locality prohibits objects with infinitesimally thin protuberances, even if these protuberances are extremely short. A huge spherical object with a very short line segment sticking out of its boundary will not be $k$-fat for any value of $k$. This might contradict with our intuitive idea of fatness. An alternative is the more 'global' type of fatness given in Definition 2.4. For convenience, we will refer to it as *thickness*[2]. Here, we only compare the volume of the entire object to the volume of its minimal (volume) enclosing hypersphere: the volume of the object should be at least a certain portion of the minimal enclosing hypersphere of the object. This more liberal definition allows objects with small protuberances. If $E$ is an object then we denote the minimal enclosing hypersphere of $E$ by $MES_E$.

[1] For even dimension $\omega_d = \omega_{2m} = \pi^m/m!$. For odd dimension $\omega_d = \omega_{2m+1} = 2(2\pi)^m/(2m+1)!!$. See, e.g., [36, Section 394].

[2] Thickness is equivalent to our initial notion of fatness, as presented in [94]. Its shortcomings with respect to the ability to obtain a low free space complexity led to the present definition of fatness, given as Definition 2.2.

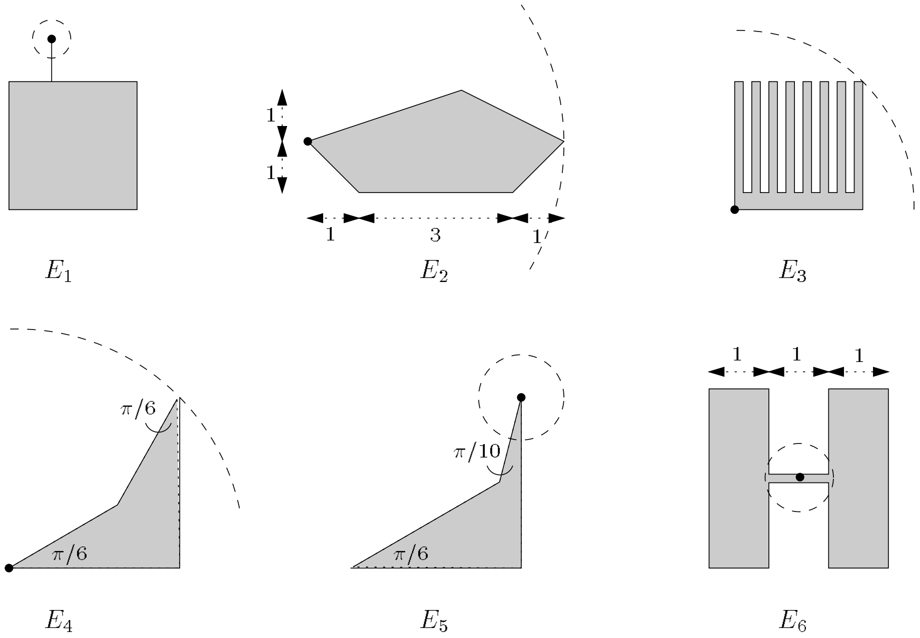


Figure 2.3: A single non-fat object $E_1$ and five objects $E_2, \ldots, E_6$ of varying fatness. The width of a tooth of the 'comb' $E_3$ equals the distance between two successive teeth. The width of the narrow bar of the 'H'-shaped object $E_6$ is 100 times smaller than the width of both wide bars. The length of the narrow bar equals the width of the wide bars. The dashed circles are the relatively emptiest circles in $U_{E_i}$, $(1 \leq i \leq 6)$; the black dots are the circle centers.

the sharpest angle determines the fatness. The object $E_5$ is 20-fat. The object $E_6$, finally, is not very fat, due to the narrow bar; $E_6$ is $(25\pi)$-fat. Here, the narrowest corridor determines the fatness of the object. Note that the emptiest circle slightly penetrates the wide bars. (See the next section for some information on computing the fatness of objects.)

We list a few straightforward properties of fat objects without proof, as the validity of each of the properties is easily verified.

**Property 2.3** *Let $E \subseteq \mathbb{R}^d$ be a $k$-fat closed connected object. Then*

- **(a)** *$E$ is $k'$-fat, for any $k' \geq k$;*
- **(b)** *$R \cdot E$ is $k$-fat, for any rotation matrix $R \in SO(d)$;*
- **(c)** *$E + t$ is $k$-fat, for any translation vector $t \in \mathbb{R}^d$;*
- **(d)** *$\lambda E$ is $k$-fat, for any scaling factor $\lambda \in \mathbb{R}^+$.*

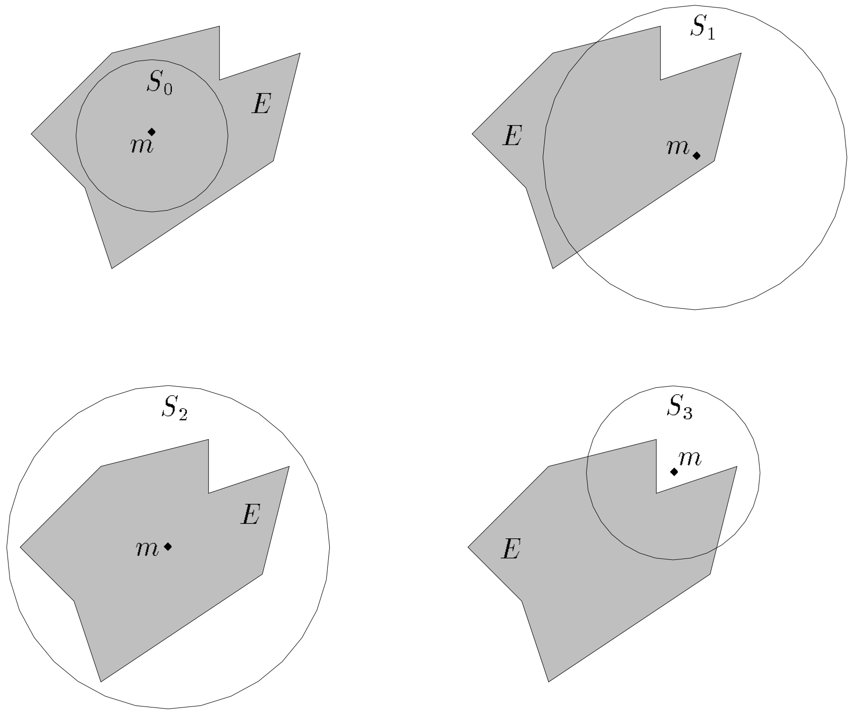


Figure 2.2: Illustration of the definition of $U_E$: $S_0, S_1 \in U_E$, $S_2 \notin U_E$ because $\partial S_2 \cap E = \emptyset$, $S_3 \notin U_E$ because its center $m \notin E$.

Informally, an object $E$ is $k$-fat if the part of any hyperspherical region $S$ with a boundary that intersects $E$ and its center inside $E$ covered by the object $E$ is at least a $1/k$-th of $S$. Hence, the relatively emptiest hypersphere among all hyperspheres centered inside $E$ and with a boundary intersecting $E$ determines the fatness of $E$. Figure 2.3 gives a collection of two-dimensional objects. Below, an indication of the fatness of the these objects is given, along with an indication of the relatively emptiest circle. The diverse character of the various emptiest circles gives a first indication that the fatness of an object may be hard to compute. The object $E_1$ is not fat due to the infinitely thin part on the upper edge. No finite bound exists on the ratio of the area inside the dashed circle and the area of $E_1$ inside the circle. The fatness of a convex object like $E_2$ is computable from its area and its diameter (see Section 2.2); $E_2$ is $(50\pi/13)$-fat. The object $E_3$ may seem quite thin (i.e. not very fat) at first sight. The closeness of the teeth of the 'comb' though makes it hard to draw relatively empty circles centered inside $E_3$. $E_3$ is $(4\pi)$-fat. The object $E_4$ is $(2\pi + 2\pi\sqrt{3})$-fat. Like in many other cases, the relatively emptiest circle is centered on the object boundary and enclosing the object entirely. If one of the angles is chosen smaller, then the emptiest sphere is centered at the corresponding vertex and passes through the edges incident to the vertex. In such a case (see e.g. $E_5$),

## 2.1 Fatness

Our definition of fatness in a $d$-dimensional Euclidean space involves $d$-dimensional closed hyperspherical regions centered at some arbitrary point in an object $E$. The closed hyperspherical region with radius $r$ centered at $m$ will be denoted by $S_{m,r}$, so

$$S_{m,r} = \{x \in \mathbb{R}^d | d(x,m) \leq r\};$$

the boundary of $S_{m,r}$ will be denoted by $\partial S_{m,r}$, so

$$\partial S_{m,r} = \{x \in \mathbb{R}^d | d(x,m) = r\}.$$

Hyperspherical regions with boundaries that have non-empty intersection with an object $E$ play a central role in our notion of fatness. Therefore, the following definition is useful.

**Definition 2.1** $[U_{m,E}, U_E]$
*Let $E \subseteq \mathbb{R}^d$ be an object. The set $U_E$ is defined as*

$$U_E = \bigcup_{m \in E} U_{m,E},$$

*where*

$$U_{m,E} = \{S_{m,r} \subseteq \mathbb{R}^d \mid \partial S_{m,r} \cap E \neq \emptyset\}.$$

So, $U_E$ is the set of all hyperspherical regions with center inside $E$ that do not fully contain $E$. Figure 2.2 gives two-dimensional examples showing two circular regions $S_0$ and $S_1$ that belong to $U_E$ and two circular regions $S_2$ and $S_3$ that do not belong to $U_E$. The region $S_0$ lies completely inside the object $E$ and is therefore easily seen to be an element of $U_E$. The region $S_1$ is only partly covered by $E$ but, since its center lies inside the object $E$ and its boundary has non-empty intersection with $E$, the region $S_1$ is a member of $U_E$. The circular region $S_2$ does not belong to $U_E$ because its boundary has empty intersection with $E$, whereas $S_3$ is not a member of $U_E$ because it has its center outside $E$.

We define fatness in a way such that objects are not only 'compact' but also do not have extremely thin protuberances. The definition of fatness involves some positive number $k$. This number is a measure for the actual fatness of the object. If the value of $k$ is increased then the object is allowed to be less fat. For objects with a boundary with infinitesimally thin protuberances (e.g. line segments) it is impossible to find such a $k$, so these objects can never be fat.

**Definition 2.2** [$k$-**fat**]
*Let $E \subseteq \mathbb{R}^d$ be an object and let $k$ be a positive constant. The object $E$ is $k$-fat if:*

$$\forall S \in U_E \qquad k \cdot volume(E \cap S) \geq volume(S).$$

$n$ triangles with fat $xy$-projections can be computed in time $O(n \log^6 n)$. The computation takes $O(n^{4/3+\epsilon})$, for any $\epsilon > 0$, if the triangles are non-fat [17]. In addition, they prove that the depth order of $n$ convex objects with fat $xy$-projections and sizes within a constant ratio from one another is computable in time $O(n\lambda_s^{1/2}(n) \log^4 n)$, where $\lambda_s(n)$ is related to the length of so-called Davenport-Schinzel sequences [4]; the parameter $s$ equals the maximum number of intersections of the boundaries of any two $xy$-projections of the convex objects. The fatness of the object projections boils down to a constant ratio between the size of the smallest enclosing square and the largest inscribed square of any projection. Halperin and Overmars [42], finally, use ideas from the study of fatness to obtain efficient algorithms for manipulating a molecule model of loosely inter-penetrating spheres, representing the atoms that constitute the molecule.

The notions of fatness encountered so far mostly apply to a limited set of objects in two-dimensional space. For our aim, a study of the role of fatness in motion planning, we need a more general notion that at least applies to planar and spatial objects. The notion of $k$-fatness that is introduced in Section 2.1 applies to arbitrary objects in any dimension and forbids objects to be long and thin or to have long and thin parts. This type of fatness imposes a sufficient requirement on the obstacles to obtain a low free space complexity result. The parameter $k$ gives a qualitative indication of the fatness: the lower the value of $k$, the fatter the object. In the sequel, a fat object is an object that is $k$-fat for a constant $k$. Section 2.1 furthermore compares our notion of fatness with alternative notions.

The remainder of the chapter is mainly devoted to deducing properties that serve as tools elsewhere in the thesis. Nevertheless, some of these results are also interesting in their own right as they have applications outside motion planning.

Section 2.3 contains the low object density result for scenes of non-intersecting fat objects. The property plays a crucial role throughout the thesis. Within the same section, it forms the basis of a linear complexity result for the arrangement obtained by growing the fat object boundaries by an amount proportional to the size of the smallest object. Besides its applications in motion planning, the latter result is also interesting in relation to the complexity of the union boundary of figures in arbitrary dimension.

In Section 2.4, it is shown that the union of two (intersecting) fat objects is at most a constant factor less fat than the least fat of its two constituents. As a result, it takes at least $\Omega(\log(k/k'))$ pieces to partition an object that is *not* $k$-fat (so 'thinner' than $k$-fat) into $k'$-fat pieces for any $k' \leq k$. The failure to subdivide a thin object into a constant number of fat objects makes it impossible to extend the results in this thesis to thin objects by partitioning the objects into fat objects, as such an approach would increase the asymptotic size of the object set. The last section of this chapter presents a generalization of the notion of fatness, and states the relations between the different types of fatness that fit in the generalization.

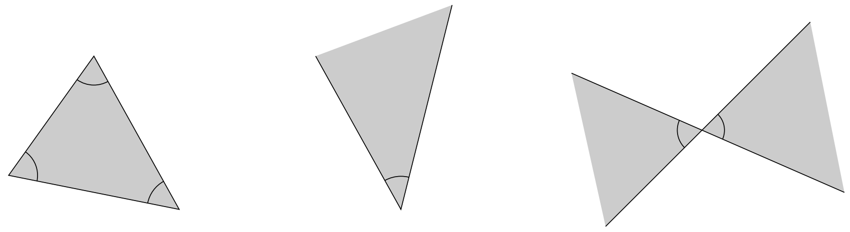

Figure 2.1: The triangle, wedge, and double wedge are $\delta$-fat for some constant $\delta > 0$ if the indicated angles are at least $\delta$.

quadratic instead of inverse cubic dependence on the constant $\delta$. Van Kreveld [58] extends the $O(n \log \log n)$ boundary complexity result to so-called $\delta$-wide polygons, where $\delta$ $(0 < \delta \leq 1)$ gives the minimal ratio of the width and length of any corridor (narrow passage) in the polygon. Finally, Alt et al. [5] report an $O(n)$ bound for the complexity of the union boundary of $n$ translated copies of a bowtie. A bowtie is the rotation figure of a rectangle. The linear bound holds if the rotation angle does not exceed $2 \arctan(b/a)$, where $a$, $b$: $a \geq b$, are the lengths of the rectangle's sides. Note that the aspect ratio of a rectangle, that is, the ratio of the length of its sides, intuitively provides a good qualitative measure of its compactness or fatness.

Papers by Katz, Overmars, and Sharir [49], Overmars [73], De Berg, De Groot, and Overmars [15], and Agarwal, Katz, and Sharir [1] report algorithmic consequences of the fatness of the objects under consideration. Katz, Overmars, and Sharir [49] present an algorithm for efficient hidden surface removal for scenes of objects with small union size. The algorithm computes the visibility map from $z = \infty$ of a set of $n$ $\delta$-fat triangles in 3-space, each of which is contained in a plane parallel to the $(x, y)$-plane, in time $O((n \log \log n + k) \log^2 n)$, where $k$ is the complexity of the output. Results for comparable scenes of non-fat triangles are, among others, an $O(n\sqrt{k} \log n)$ algorithm by Sharir and Overmars [92] and an $O(n^{1+\epsilon} + n^{2/3+\epsilon} k^{2/3})$ algorithm, for any $\epsilon > 0$, by Agarwal and Sharir [3]. Overmars [73] gives a simple data structure for point location in $d$-dimensional subdivisions consisting of fat cells with query time $O(\log^{d-1} n)$ and storage requirement $O(n \log^{d-1} n)$. For a detailed discussion of Overmars' results and an overview of results on point location for non-fat objects, the reader is referred to Chapter 3. There, it is also shown how the point location structure can be used for range searching and how the data structure is built efficiently. Overmars uses the notion of fatness that is introduced in this chapter. De Berg, De Groot, and Overmars [15] show that an $O(n)$ size orthogonal subdivision of a set of $n$ planar non-intersecting fat objects exists in which each region is intersected by a constant number of objects. The subdivision can be computed in time $O(n \log^2 n)$ using $O(n \log n)$ storage. The result leads to efficient binary space partitions for scenes of fat objects. Agarwal, Katz, and Sharir [1] study depth orders of non-intersecting fat objects in 3-space. They show that the depth order of

# Chapter 2

# Fatness in computational geometry

Many combinatorial and algorithmic worst-case complexity bounds in computational geometry follow from rather artificial constructions that are not very likely to occur in practice. Often, such constructions include extremely small, large, or thin objects, like lines, line segments, infinitely long simplices, and points. In many cases, the artificial worst-case constructions become impossible if the objects under consideration are not allowed to have 'extreme' shapes, but are assumed to have some fatness property. Over the past few years, researchers in computational geometry have not only noted that certain constructions become impossible for objects with a certain fatness but, more surprisingly, also that combinatorial and algorithmic complexities of certain problems are provably lower if the objects satisfy specific fatness constraints.

A sequence of recent papers considers the influence of fatness in computational geometry. Alt et al. [5], Efrat, Rote, and Sharir [34], and Matoušek et al. [67] study the complexity of the union of $\delta$-fat triangles, wedges (that is, regions bounded by two half-lines emanating from a single point), and double wedges (that is, regions bounded by two intersecting lines), for a fixed constant $\delta$. Either one of the regions is $\delta$-fat if all its internal angles are at least $\delta$ (see Figure 2.1). Note that quadratic-complexity constructions exist for all union sizes if the objects in the union are non-fat (see for example Figure 1.4). Alt et al. [5] show that the complement of the union of $n$ $\delta$-fat double wedges consists of $O(n)$ components. In addition, they prove an $O(n)$ bound on the boundary complexity of $n$ homothetic (that is, scaled and translated) or reflected (with respect to a vertical line) homothetic copies of a single $\delta$-fat triangle. Matoušek et al. [67] generalize the results by Alt et al. [5]. The authors study the union of $n$ $\delta$-fat triangles and prove that its boundary has complexity $O(n \log\log n)$. The complement of the union consists of $O(n)$ components. If the triangles have roughly the same size or if they are replaced by $\delta$-fat wedges, then the complexity of the union boundary becomes $O(n)$. Efrat, Rote, and Sharir [34] prove a similar result for the union boundary of $\delta$-fat wedges with a nearly inverse

problem left open in [73]) in time $O(n \log^{d-1} n \log\log n)$.

Chapters 4-7 focus on the role of fatness in motion planning. A quick glance of the chapters learns that Chapter 4 concentrates on the combinatorial aspects of motion planning. Besides giving an overview of combinatorial complexities of various motion planning problems (in terms of worst-case free space complexities), it formulates the mild assumptions that, along with the fatness of the obstacles, yield a linear free space complexity. The remaining chapters deal with the algorithmic aspects of motion planning. In Chapter 5, the impact of fatness on a representative selection of existing (planar) motion planning algorithms is considered. The algorithms show varying sensitivity to the low free space complexity induced by the fatness of the obstacles. Chapter 6 presents an efficient general paradigm for motion planning amidst fat obstacles that exploits the specific structure of the free space (of motion planning problems amidst fat obstacles) to reduce the problem of finding a cell decomposition of the free space to the problem of finding some constrained partition of a lower-dimensional subspace. In Chapter 7, the value of the paradigm is demonstrated, as it leads to efficient algorithms for a number of realistic motion planning problems.

$O(n^2 \log n)$ algorithm for motion planning among polyhedral obstacles in 3-space. The algorithm computes a cell decomposition of size $O(n^2)$. Non-polyhedral obstacles require a totally different algorithm. The simple algorithm presented in Chapter 7 for motion planning amidst arbitrarily-shaped obstacles in 3-space runs in $O(n^3)$ time and yields a decomposition of size $O(n^3)$. The results are not restricted to the specific circumstances of fat obstacles and a bounded-size robot but hold in all workspaces with relative low obstacle densities. Note that all bounds are independent of the number of degrees of freedom of the robot, which can easily be as high as six or more.

## 1.3 Fatness in geometry and thesis outline

The first chapters of this thesis introduce a new notion of fatness and discuss its role in a broader geometric context than motion planning. From Chapter 4 onward, the emphasis is on the influence of fatness on different aspects of the motion planning problem.

Chapter 2 introduces a new and general notion of fatness. The new notion is subsequently compared with previous and less general notions. The chapter furthermore reports two properties of scenes of fat objects in $\mathbb{R}^d$ that are both key tools in many proofs throughout the thesis. The first property applies to scenes of $n$ non-intersecting fat objects. It is shown that any region of size proportional to the smallest among the objects intersects at most a constant number of objects in the scene. This property is for obvious reasons repeatedly referred to as the low object density property. If $n$ non-intersecting objects are grown then they eventually start intersecting. The second property states that if the growth is again proportional to the smallest object, then the arrangement of intersecting boundaries of the (not necessarily fat) grown objects has complexity $O(n)$. Besides its role as a tool, the latter property has interesting consequences for complexities of union boundaries of geometric figures. In addition to these results, Chapter 2 studies the relation between the (lack of) fatness of an object $E$ and the (lack of) fatness of objects $E_1, \ldots, E_m$ with $\cup_{1 \leq i \leq m} E_i = E$. The main conclusion from the obtained results is that an object with low fatness cannot be split into (or covered by) a constant number of objects with high fatness.

In [73], Overmars discusses a data structure for efficient and simple point location in fat subdivisions or sets of disjoint fat objects with total complexity $n$. The structure supports point location queries in time $O(\log^{d-1} n)$ and uses $O(n \log^{d-1} n)$ storage. Chapter 3 shows that the data structure can be used to answer range queries with arbitrarily-shaped but bounded-size ranges. To this end, it is proven that, under the condition of fatness of the stored objects, each bounded-size range query can be solved by a constant number of point location queries with carefully chosen points, leading to a range query time of $O(\log^{d-1} n)$. It is furthermore shown that such range queries facilitate the efficient construction of the data structure (a

Algorithms for efficient motion planning in 3D workspaces are scarce: approaches in contact space, like the algorithms mentioned above by Sifrony and Sharir, and by Avnaim, Boissonnat, and Faverjon, were never shown to generalize to higher dimensions. General approaches to motion planning (for example by Canny [20] and Schwartz and Sharir [85]) are computationally expensive, particularly for the low free space complexity motion planning problems from the realistic framework. Three-dimensional workspaces imply at least three-dimensional configuration spaces with arrangements defining the free portions. Naturally, the structure of such higher-dimensional arrangements is considerably more complex to understand, let alone to subdivide the free arrangement cells into simple subcells or catch their structure in some one-dimensional roadmap. At this point, however, fatness comes to our help to provide us with a very beneficial property of the workspace, which in fact also led to the enhanced performance of Schwartz and Sharir's algorithm mentioned in the previous paragraph: the bounded local complexity of the workspace implied by the fatness of the objects makes it possible to partition (a subspace of) the workspace W rather than the configuration space into regions $R$ such that the free part of the configuration space cylinder obtained by lifting $R$ into configuration space has constant complexity. Moreover, the bounded local complexity also establishes the existence of small partitions into such regions.

We formalize and exploit the workspace properties outlined in the preceding paragraph and obtain a paradigm in Chapter 6 for planning the motion of a not too large constant-complexity robot moving amidst constant-complexity fat obstacles. The paradigm follows the cell decomposition approach to motion planning and reduces the problem of finding a decomposition of the free space to the problem of finding a partition of an appropriate lower-dimensional subspace of the configuration space, subject to some constraints. The robot's workspace turns out to be a valid choice for the subspace under the general circumstances of a free-flying robot. The size of the free space decomposition into simple subcells is determined by the size of the partition in the lower-dimensional subspace of the configuration space. The running time of algorithms based on the paradigm depends on the time to find such a partition.

In Chapter 7, the paradigm is shown to lead to efficient algorithms for many motion planning problems among constant-complexity fat obstacles both in $\mathbb{R}^2$ and $\mathbb{R}^3$. We briefly review the results. Unless stated otherwise, the bounds apply to free-flying robots. The algorithm for solving the planar problem among arbitrarily-shaped obstacles in the plane runs in $O(n \log n)$ and outputs an optimal (linear) size decomposition of the free space. The same optimal bounds are obtained for two practical instances of spatial motion planning. The first case concerns settings in which the obstacles have roughly the same size, that is, where the ratios of the obstacle sizes are bounded by a constant. In the other, often encountered case, the obstacles are unconstrained but the motion of the robot is confined to a plane in the spatial workspace: the robot's workfloor. Many examples of such constrained robots can be found in industrial environments. Chapter 7 furthermore reports an

$\lambda_6(n)$ is a nearly-linear function related to Davenport-Schinzel sequences [4, 90]. The resulting motions are once more semi-free rather than free. A general roadmap-based algorithm is due to Canny [20]. The algorithm computes a roadmap in the free part of an $f$-dimensional configuration space in roughly $O(n^f \log n)$ time, assuming a robot and obstacles with bounded complexity. The time-bound is close to worst-case optimal.

The description of the ideas behind both exact planning approaches exhibits how the complexity of the free space influences the efficiencies of the algorithms. As it is impossible to decompose the free space into subcells with a cumulative complexity that is lower than the free space complexity, or to capture the combinatorial structure of FP in a roadmap with complexity below the complexity of FP, the complexity of the free space clearly provides a lower bound on the complexity (and computation time) of any of the motion planning algorithms.

A question that immediately comes to mind when considering the linear free space complexity result is whether it opens the way to efficient motion planning algorithms for realistic environments that fit in the framework sketched in the previous section. The sensitivity to the actual free space complexity of many existing algorithm is unclear: algorithms may e.g. construct wasteful decompositions or roadmaps in cases of low FP complexity, or may construct small decompositions and roadmaps but at relatively high computational cost. Two algorithms, however, yield a more or less immediate result, namely the $O(K \log n)$ boundary-vertices retraction algorithm by Sifrony and Sharir [93] and the $O(n^3 \log n)$ boundary cell decomposition by Avnaim, Boissonnat, and Faverjon [10]. The relative low obstacle density causes the number $K$ of close corner pairs to be only $O(n)$ resulting in a running time of $O(n \log n)$ for a not too large ladder among fat obstacles. Avnaim, Boissonnat, and Faverjon claim that the running time of their algorithm decreases considerably if the obstacle density is low. No other papers claim enhanced performance of algorithms under certain circumstances.

Most of the exact motion planning algorithms discussed in this section have never been implemented. One of the few exceptions is Schwartz and Sharir's $O(n^5)$ algorithm. Bañon [11] discusses an implementation for a ladder robot that is reported to perform surprisingly well, contrary to expectations based on the theoretical complexity analysis. This observation may be due to a hidden sensitivity of the algorithm's running time to the complexity of the free space, which is far below $O(n^5)$. Schwartz and Sharir do not give any clues in this direction. Nevertheless, the surprising performance of the exact algorithm motivates a more precise theoretical analysis of its performance under the realistic assumptions sketched in the previous section. In Chapter 5, it is proven that the algorithm by Schwartz and Sharir runs, unmodified, in time $O(n^2)$ if the obstacles are fat and the robot is not too large, whereas a minor modification even enhances the efficiency to a running time of $O(n \log n)$. The same chapter also shows examples of algorithms that do *not* benefit from the fatness of the obstacles.

the obstacles. When increasing the configuration space dimension beyond three, the results deteriorate rapidly. Schwartz and Sharir [87] decompose the free space of a robot moving amidst polyhedral obstacles in 3-space. The algorithms for a (5-DOF) ladder robot and for a (6-DOF) polyhedral robot yield connectivity graphs with $O(n^{11})$ and $O(n^{15})$ nodes/subcells respectively and have at least corresponding running times, where $n$ is the total complexity of the obstacles. Ke and O'Rourke [50] give a cell decomposition algorithm that improves the $O(n^{11})$ bound for a ladder in 3-space to $O(n^6 \log n)$. In a different paper [85] in the Piano Movers' series, Schwartz and Sharir give a general cell decomposition algorithm, based on algebraic decomposition techniques by Collins [30]. The running time of the algorithm for a robot with $f$ degrees of freedom and constant complexity amidst obstacles with cumulative complexity $n$ is $O(n^{2^{f+6}})$, which amounts e.g. to $O(n^{4096})$ for a free-flying rigid robot in 3-space. Needless to say is that the known results for motion planning problems with more than three degrees of freedom are far from practical due to their performance. Further examples of cell decompositions are found in the two other papers in the Piano Movers' series [86, 91].

An alternative exact approach to motion planning is the *retraction method* or *roadmap method*. The approach recursively 'retracts' the free space FP into a lower-dimensional subspace FP′. The crucial aspect of the approach is the retraction function $\mathrm{Im} : \mathrm{FP} \to \mathrm{FP}'$, mapping each placement in FP onto a placement in the subspace FP′. A simple collision-free motion must exist between every point $Z \in \mathrm{FP}$ and its mapping $\mathrm{Im}(Z) \in \mathrm{FP}'$. Provided that such simple motions exist, the problem of planning a motion between $Z_0$ and $Z_1$ in FP is reduced to the problem of finding a motion between their retractions $\mathrm{Im}(Z_0)$ and $\mathrm{Im}(Z_1)$ in the lower-dimensional space FP′. Hence, motion planning in FP is reduced to lower-dimensional motion planning in FP′. The objective is to obtain, after repeated retractions, a one-dimensional network, or roadmap, $\mathrm{N} \subseteq \mathrm{FP}$. There, motion planning is reduced to graph searching if we represent the one-dimensional network N as a graph. Ó'Dúnlaing and Yap [71] and Ó'Dúnlaing, Sharir, and Yap [70] present algorithms for planning the motion of a disc and a ladder, based on retractions onto curves in two- and three-dimensional Voronoi diagrams. The algorithms run in time $O(n \log n)$ and $O(n^2 \log n \log^* n)$ respectively[3]. Leven and Sharir [64] use generalized Voronoi diagrams to extend the former $O(n \log n)$ result to a translating convex robot. Sifrony and Sharir [93] apply a variant of the retraction technique to a translating and rotating ladder robot among polygonal obstacles. They use a retraction that maps placements in FP onto particular vertices on the boundary of FP. The resulting algorithm runs in $O(K \log n)$, where $K$ is the number of feature pairs that are less than the length of the ladder apart. Kedem and Sharir [51] present a variant of the retraction approach for a convex polygonal robot in which they construct a graph on the edges of the boundary BFP of the free space. The algorithm runs in time $O(n\lambda_6(n) \log n)$, where

[3] $\log^* n = \min\{\, i \geq 0 \mid \log^{(i)} n \leq 1 \,\}$, where $\log^{(i)}$ stands for the logarithm function applied $i$ times in succession [31].

plexity assumption for the robot and the individual obstacles, provide a practical framework for many real-life motion planning problems. The free space for all problems that fit in this framework is shown to have only linear complexity in Chapter 4. The linear complexity result opens the way to devising efficient algorithms for solving many motion planning problems in realistic environments.

## 1.2 Exact motion planning algorithms

Exact algorithms process the free space into a representation that captures all the necessary details of (the structure of) the free space to guarantee completeness. The efficiency of such methods is usually expressible in terms of the complexity of the motion planning environment (see below). Judging purely on the worst-case complexities for exact motion planning, exact methods do not seem to be practical alternatives for approximate methods in real-life situations. The constructions that lead to these complexity bounds, however, are hardly encountered in practice. Inspired by this observation, this thesis shows that under certain realistic assumptions (on fatness and size ratios), some exact algorithms do become feasible as their running times are reduced considerably. Moreover, these realistic assumptions result in a very beneficial structure of the free space that allows for an efficient general paradigm for the exact solution of the motion planning problem.

Let us now briefly review the two main classes of exact algorithms: cell decomposition algorithms and retraction or roadmap algorithms. In general, the existing exact motion planning algorithms process the free space into a structure that is capable of efficiently handling multiple (arbitrary) path-finding queries. The running time of an exact motion planning algorithm is actually the time to process the free space into such a query structure. Both exact approaches reduce the motion planning problem to a graph searching problem. Exact cell decomposition methods partition the free space FP into a finite number of simple connected *subcells*, such that planning a motion between any two placements within a single subcell is straightforward and such that uniform crossing rules can be defined for $\mathcal{B}$ crossing from one subcell into another. Applications of the cell decomposition technique include the famous $O(n^5)$ Piano Movers' algorithm by Schwartz and Sharir [84] for planning the motion of a polygonal robot $\mathcal{B}$ moving amidst polygonal obstacles $\mathcal{E}$ in the plane (with a total number of $n$ edges). This early result has been improved to $O(n^2 \log n)$ for a ladder (line segment) robot by Leven and Sharir [64]. Halperin, Overmars, and Sharir [43] decide in time $O(n^2 \log^2 n)$ on the existence of a collision-free path for an L-shaped (non-convex) robot among polygonal obstacles. Avnaim, Boissonnat, and Faverjon [10] apply a variant of the cell decomposition technique to a translating and rotating polygon among polygonal obstacles. Instead of decomposing the free space, they decompose the free space boundary $\mathrm{BFP} = cl(\mathrm{FP}) \setminus \mathrm{FP}$ in time $O(n^3 \log n)$. The motion obtained with this algorithm is semi-free rather than free: except from the first and last portion, the robot moves in contact with

robot and the obstacles is necessary to obtain the low obstacle density, given that the obstacles may lie arbitrarily close to each other. Comparable robot and obstacle sizes alone, however, are insufficient to achieve really low free space complexities. A very interesting additional assumption for the obstacles is *fatness*. The fatness assumption forbids the obstacles to be long and thin themselves or to have long or thin parts.

Fatness is an interesting phenomenon in computational geometry. It has received quite some attention over the past few years. Several papers study the surprising influence of fatness of the objects under consideration on combinatorial and algorithmic complexities. Examples of combinatorial complexity reductions include papers by Alt et al. [5], Matoušek et al. [67], Efrat, Rote, and Sharir [34], and Van Kreveld [58] which all show that the complexity of the union of certain geometric figures is low if the objects are fat. Overmars [73] presents an efficient algorithm for point location in subdivisions consisting of fat cells. For a discussion of these and some other results, the reader is referred to Chapter 2. For the moment, the impact of fatness is illustrated by a single, though very attractive, example: the complexity of the union of $n$ triangles in the plane. If the triangles are unconstrained then a quadratic union size can be obtained by arranging the triangles in a grid-like fashion as shown in Figure 1.4. Matoušek et al. [67] show that the complexity of the union of $n$ triangles is only $O(n \log\log n)$ if the angles of all triangles are at least $\delta$, for some fixed constant $\delta > 0$.

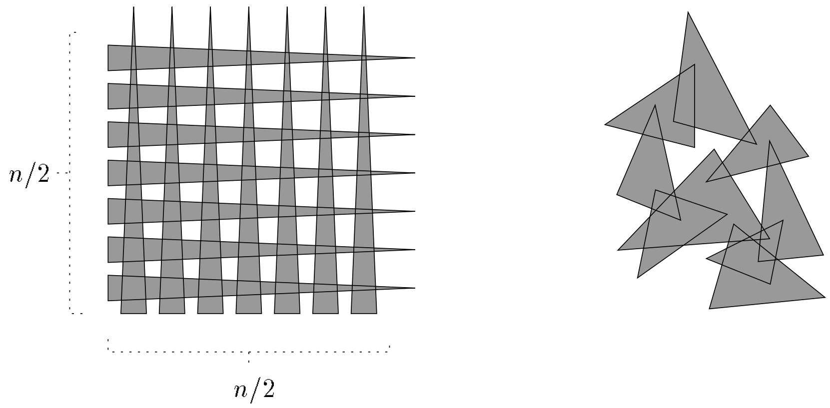


Figure 1.4: The union boundary of $n$ arbitrary triangles can have complexity $\Omega(n^2)$; if the triangles are fat (see right), then the complexity is nearly linear.

Chapter 2 proposes a new notion of fatness that, contrary to previous notions like the $\delta$-fatness for triangles, deals with arbitrary shapes in any dimension. The definition involves a parameter $k$ that gives a qualitative indication of the fatness of an object. The fatness of the obstacles of $\mathcal{E}$ according to this definition, along with a bound on the relative sizes of the obstacles and the robot $\mathcal{B}$, and a bounded com-

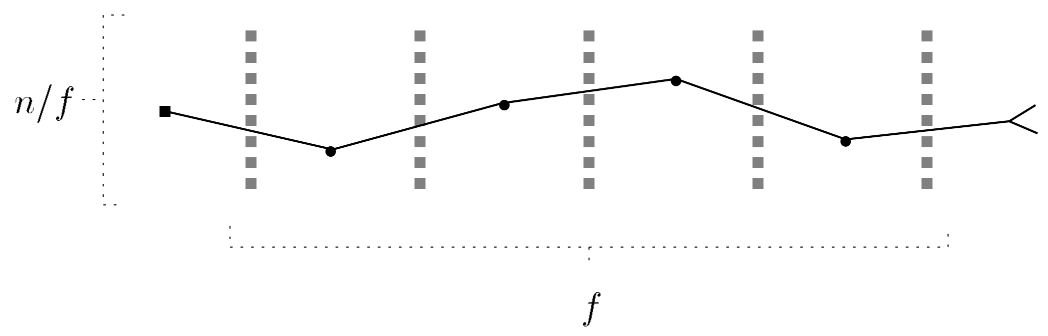


Figure 1.2: An ($f$-DOF) robot arm consisting with $f$ links and $f$ revolute joints with $\Omega(n^f)$ $f$-fold contacts, and, hence, with free space complexity $\Omega(n^f)$.

Figure 1.3 where the 'spider' robot and the obstacles have constant complexity and roughly the same sizes. The robot has six degrees of freedom: two for its position in the workspace, and four for each of the legs that are free to rotate around the central joint. While being in contact with a certain obstacle, the robot is unable to touch more than a constant number of other obstacles (on the average). Then, the number of multiple contacts can impossibly exceed $O(n)$. Hence, the free space for this robot has complexity $O(n)$ and thus remains far below the free space complexity obtained with the construction of Figure 1.2. The impressive gap between the $\Omega(n^f)$

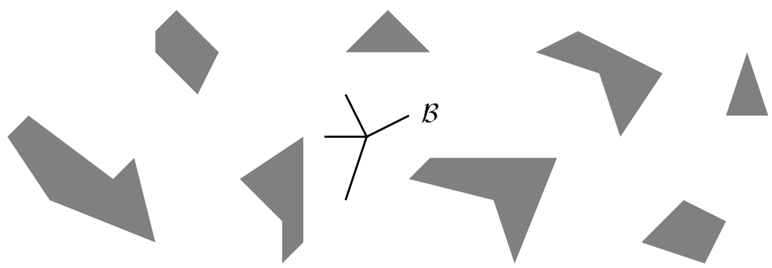


Figure 1.3: A (6-DOF) robot with few multiple contacts, and, hence, with low free space complexity.

construction and the realistic $O(n)$ example immediately raises the question what specific properties of the robot and the obstacles lead to low free space complexities. What natural mild assumptions would for example lead to the relative low obstacle density of the above example, in which the robot is unable to touch more than a constant number of obstacles simultaneously? (Circumstances that resemble the relative low obstacle density have been studied by Schwartz and Sharir [88] who refer to it as *bounded local complexity* and by Pignon [78] who calls it *sparsity*.) The case of the 6-DOF robot strongly suggests that a bound on the relative sizes of the

Non-holonomic constraints are relations between the degrees of freedom of the robot. The relations impose restrictions on the shape of the collision-free robot motions.

A *collision-free path* or *collision-free motion*, or path or motion for short, for a robot $\mathcal{B}$ from an initial placement $Z_0 \in \mathrm{FP}$ to a final placement $Z_1 \in \mathrm{FP}$ is a continuous map:

$$\tau : [0,1] \rightarrow \mathrm{FP},$$

with

$$\tau(0) = Z_0 \quad \text{and} \quad \tau(1) = Z_1.$$

Semi-free motion can be allowed by changing the range of the map $\tau$ into SFP. Hence, the problem of motion planning is equal to the problem of finding a continuous curve between two query points, completely lying inside the free portion FP of the configuration space. No quality restrictions with respect to length, curvature etc. are imposed upon the reported path. The effort that is to be invested in finding such a curve obviously highly depends on the complexity of the free space FP. The discussion of motion planning algorithms below confirms this statement.

The complexity of the free space, as we will see in Chapter 4, is determined by the number of multiple contacts of the robot $\mathcal{B}$. A multiple contact of the robot $\mathcal{B}$ is a placement in which it touches more than one obstacle feature, that is, a basic part of the obstacle boundary like a vertex, edge, or face. Besides the collisions of the robot with the obstacles, parts of the robot can also collide with other robot parts. Although these so-called *self-collisions* are often ignored in our considerations, we shall return to them at appropriate moments to demonstrate the validity of the results when self-collisions are taken into account. Unfortunately, the number of multiple contacts, and, hence, the complexity of the free space, can be very high. If $n$ is the number of obstacle features and $f$ is the constant number of degrees of freedom of the robot (that is, the dimension of the configuration space) and the number of robot features is bounded by some constant, then this complexity can be $\Omega(n^f)$. As a generic example, consider the robot arm in Figure 1.2. If the lengths of the links and the distances between the obstacles are appropriately chosen, then each of the $f$ links can be placed against any of the $n/f$ obstacles in the vertical row that it cuts through, yielding $(n/f)^f$ combinations of obstacles and therefore leading to $\Omega(n^f)$ multiple contacts. As a consequence, the complexity of the free space for the robot arm is $\Omega(n^f)$. Slightly lower worst-case free space complexities have been obtained for specific free-flying rigid robots among certain classes of obstacles. The reader is referred to Chapter 4 for an overview of some relevant results. These bounds generally remain close to an order of magnitude, i.e., a factor $n$, below the $\Omega(n^f)$ bound. Hence, even in such beneficial cases, the theoretical worst-case bounds are high. Fortunately, in many practical situations the complexity of the free space is much smaller, as artificially constructed workspaces with e.g. a very large robot and small obstacles are hardly encountered in real life. When extreme shapes and sizes of the robot and the obstacles do not occur, high free space complexities tend to be harder to obtain. Consider for example the realistic motion planning environment of

Another substantial ingredient of the motion planning problem is the set $\mathcal{E}$ of obstacles in the workspace W. Each *obstacle* $E \in \mathcal{E}$ is a closed connected, possibly unbounded, subset of W. The obstacles are stationary, that is, they do not move or change shape in time. Moreover, the obstacles of $\mathcal{E}$ are assumed to be known, so that the robot $\mathcal{B}$ does not have to *explore* the workspace W and detect the obstacles through certain sensing devices. The presence of the obstacles in the workspace causes some placements to be inaccessible. A point $Z$ in the configuration space $C$ can correspond either to a placement of the robot $\mathcal{B}$ in the workspace W in which it intersects no obstacle, $\mathcal{B}[Z] \cap (\cup_{E \in \mathcal{E}} E) = \emptyset$, or to a placement of $\mathcal{B}$ in W in which is has non-empty intersection with the obstacle set $\mathcal{E}$. The first type of placement is called a *free placement*. The *free space* FP is the open set of all free placements of the robot $\mathcal{B}$, hence

$$\mathrm{FP} \quad = \quad \{\, Z \in C \mid \mathcal{B}[Z] \cap (\cup_{E \in \mathcal{E}} E) = \emptyset \,\}.$$

If the placements of the robot $\mathcal{B}$ are restricted to FP then $\mathcal{B}$ is not allowed to move in contact with the obstacles of $\mathcal{E}$. Sometimes, however, allowing motion in contact, or *compliant* motion, results in more efficient motion planning algorithms [10, 93]. A *semi-free placement* of the robot is either a free placement or a placement in which it touches one or more obstacles but intersects the interior of no obstacle. More formally, a semi-free placement $Z$ satisfies $\mathcal{B}[Z] \cap (\cup_{E \in \mathcal{E}} int(E)) = \emptyset$, where $int(E)$ stands for the interior of the closed set $E$, i.e., $E$ without its boundary $\partial E$. The *semi-free space* SFP is the set of all semi-free placements of the robot $\mathcal{B}$, hence

$$\mathrm{SFP} \quad = \quad \{\, Z \in C \mid \mathcal{B}[Z] \cap (\cup_{E \in \mathcal{E}} int(E)) = \emptyset \,\}.$$

Actually, the quoted results [10, 93] solve the motion planning problem in the closure *cl*(FP) of the free space which is formally a subset of SFP. Except for some very specific circumstances (see [59]), the closure *cl*(FP) of the free space equals the semi-free space SFP.

The presented problem formulation is extendible in many directions. Most of the generalizations are hardly studied in exact motion planning. One extension is to have non-stationary obstacles, either moving autonomously (see e.g. [81]) or movable by the robot. The dynamic behavior of the collection of free placements in the case of autonomously moving obstacles is adequately modeled by adding a time axis to the $f$-dimensional configuration space resulting in an $(f+1)$-dimensional so-called configuration-time space. (The intersection of this configuration-time space at some time $t = T$ shows the free and non-free placements at $t = T$.) The case of movable obstacles raises additional problematic issues like how to grasp objects. Another generalization would be to allow multiple robots. An appropriate choice for the configuration space of such a system of robots is the Cartesian product of the configuration spaces of the individual robots (see e.g. [86]). Although the results of this thesis are generalizable towards multiple robots, we restrict ourselves to a single robot. Other extensions are unknown obstacles and non-holonomic constraints.

Alt et al. [5] introduce the tightness of a motion planning problem for a rectangle among polygonal obstacles as a measure for its complexity. The tightness of a problem is closely related to the scaling factor for the rectangular robot to make the problem unsolvable if the original problem is solvable, or to make the problem solvable if the original problem is unsolvable. The authors present an approximate motion planning algorithm for the rectangular robot with a tightness-dependent running time.

## 1.1 The general motion planning problem

This thesis focusses on the following version of the general motion planning problem.

> *Given a robot $\mathcal{B}$ moving amidst a collection of obstacles $\mathcal{E}$, and an initial placement $Z_0$ and a desired final placement $Z_1$ for $\mathcal{B}$, find a continuous motion for $\mathcal{B}$ from $Z_0$ to $Z_1$ during which the robot avoids collision with the obstacles, or report that no such motion exists.*

A single robot $\mathcal{B}$ moves around in a *workspace*, or *physical space*, W. The robot's workspace W usually equals the Euclidean space of dimension two or three ($\mathbb{R}^2$ or $\mathbb{R}^3$), since these are the most interesting cases from a practical point of view. Throughout the thesis, the *robot* $\mathcal{B}$ is assumed to be a collection of *closed rigid* links (attached to each other by joints) of constant total complexity. A rigid body is a non-deformable compact connected set. A closed set incorporates the set boundary as part of the set (contrary to an open set which excludes the set boundary). The assumption that the robot is a collection of rigid bodies is liberal, as many papers require the robot to be a single rigid body.

The motion planning problem is commonly modeled and solved in the so-called configuration space of the problem. The *configuration space* $C$ is the space of parametric representations of robot placements. A *configuration* $Z \in C$ is a unique (compact) specification of the position of every point of the robot $\mathcal{B}$ at a certain placement in the workspace W[2]. Each placement of the robot $\mathcal{B}$ in its workspace W corresponds to exactly one point $Z$ in the configuration space $C$. In the sequel, the subtle difference between the configuration $Z$ and the represented robot placement is generally ignored. The parameters that are required for a unique specification of a robot placement fix the dimensions of the configuration space. These parameters are referred to as the *degrees of freedom* of the robot. The number of degrees of freedom determines the dimension of the configuration space $C$. At each placement $Z \in C$ the robot $\mathcal{B}$ covers a set of points in the workspace W which is denoted by $\mathcal{B}[Z]$.

[2] For a more concise formulation of the notions of configuration space and configuration in terms of rigid transformations and relative positions of reference frames, the reader is referred to Latombe's book [59] on the state-of-the-art in robot motion planning.

sized polyhedral and general obstacles respectively. The results are independent of the number of degrees of freedom and extend towards any environment with low obstacle density.

We are aware of only few (related) results on exact motion planning methods with provable efficiency or free space complexity-sensitive behavior for realistic motion planning problems (with low complexity workspaces or free spaces). Sifrony and Sharir [93] present a motion planning algorithm for a line segment in a planar workspace with polygonal obstacles. The reported running time of the algorithm depends (nearly exclusively) on the number of pairs of obstacle corners that lie less than the length of the ladder apart. This number gives some idea of how cluttered the obstacles in the workspace are and is furthermore closely related to the complexity of the free space. Sifrony and Sharir's algorithm is the only algorithm with a running time that is reported to depend on complexity-related variables. A few other algorithms (see Chapter 5) have some hidden dependency on the complexity of the free space.

Schwartz and Sharir [88] consider workspaces with obstacles of so-called *bounded local complexity*. Any (imaginary) ball with radius $r$ in such a workspace intersects no more than a constant number of obstacles. The property resembles a workspace property that follows from the fatness of the obstacles (see Chapter 2). The bounded local complexity is shown to have implications for the free space complexity. The authors give directions on how to solve the motion planning problem in such workspaces.

Pignon [78] structures workspaces with polygonal obstacles for a polygonal robot to easily detect certain simple and impossible path-finding queries. The author uses the maximal inscribed circle and minimal enclosing circle of the robot to define the so-called safe and impossible spaces, which are both efficiently computable subspaces of the workspace. The safe space consists of all workspace positions that the robot can occupy at any orientation without intersecting the obstacles. More precisely, the safe space is the collection of center points of the enclosing circle in which that circle does not intersect any obstacle. The impossible space consists of all positions in which the robot always intersects some obstacle, regardless of its orientation. Hence, the impossible space is the collection of centers of the inscribed circle in which that circle intersects some obstacle. The possible space is the complement of the impossible space. Now, two types of simple queries can be easily detected. If the workspace positions of the robot in the initial and final placements belong to a single connected component of the safe space, then both positions are connected by a path for the enclosing circle of the robot. As a result, it suffices to find a motion for the circle, which is a simpler motion planning problem (with two instead of three degrees of freedom). If the positions lie in different components of the possible space, then no motion for the inscribed circle of the robot exists between the query placements, and therefore certainly no motion exists for the robot itself between these placements. In all other cases, the exact solution of the problem requires the application of an exact method to the original problem.

at the exact solution of the problem so that a path for the robot is returned if one exists. It is obvious that the theoretical complexity of finding such a path depends highly on the complexity of the free space. The high bounds on the cumulative complexity of a collection of cells in an arrangement of hypersurfaces established in computational geometry demonstrate the potentially high complexity of the motion planning problem. Even though the hypersurfaces that define the arrangement in the, say, $f$-dimensional configuration space are not arbitrary (as they represent sets of contact placements), they still allow for the construction of worst-case arrangements of, roughly, at least $\Omega(n^{f-1})$ complexity, where $n$ is the number of obstacles. Exact motion planning methods process the appropriate arrangement cells into a structure capable of providing an exact answer to a path-finding query. The cumulative complexity of the cells is reflected in the size and computation time of the query structure. Since the number of degrees of freedom $f$ of practical robots is often as large as five or six, exact methods suffer from impractically high computational costs and are therefore not feasible for practical motion planning problems.

On the other hand, the worst-case arrangements mentioned in the previous paragraph involve artificial constructions with a robot and obstacles with extreme and often uncommon shapes. The complexity of the free space for many real-life motion planning problems tends to remain far below the theoretical worst-case bounds. Exact motion planning methods might become feasible for such realistic problems, provided that their performance is positively affected by reductions of the free space complexity. Unfortunately, only few of the existing exact motion planning exhibit such a dependency. The preceding observations show that it is interesting to seek for mild constraints on the robot and the obstacles that lead to a provable low free space complexity. To make the outcome practically useful, it is necessary to find motion planning methods that benefit from low free space complexities in the sense that they process the free space in time comparable to its complexity into a path-finding query structure of size comparable to the free space complexity.

A bound on the relative sizes of the robot and the obstacles and a certain 'fatness' of the obstacles are shown to be sufficient to get a free space with a complexity that is only linear in the number of obstacles. Fatness has been studied in the context of several problems in computational geometry, but, so far, not in the context of motion planning. Under the sketched circumstances, it will be shown that certain existing exact motion planning algorithms show a considerable performance enhancement. Moreover, the realistic assumptions cause the linear complexity free space to have a structure that allows for a new and simple motion planning paradigm based on the so-called cell decomposition approach. The paradigm basically reduces the planning problem to a partitioning problem in a lower-dimensional subspace. Instances of the paradigm lead to almost linear-time (in the number of obstacles) algorithms for general planar motion planning and for restricted cases of three-dimensional motion planning, namely where the robot is confined to a workfloor or where the sizes of the obstacles differ by at most a constant factor. Quadratic and cubic time algorithms are obtained for three-dimensional motion planning among arbitrarily-

a sequence of curves). Section 1.2 reveals some details of certain exact roadmap methods. Brooks [18] presents an approximate roadmap method for a translating and rotating polygonal robot among polygonal obstacles. The basis of the roadmap is an approximation of the Voronoi diagram [8] on the obstacles in the workspace. Conservative assumptions used in its construction may cause disconnected roadmap components in a single connected component of the free space, leading to potential failure to determine a path between two placements within a connected component of the free space. The method works well if the obstacles are not too much cluttered. Another approximate roadmap method, due to Overmars and Švestka [76], constructs a graph on randomly chosen configurations in free space. Two configurations are connected by an edge if a simple collision-free motion exists between them.

*Potential field* methods [52, 53, 56] direct the motion of the robot through an artificial potential field set up by the goal placement and the obstacles. The goal configuration pulls the robot towards it by generating a strong attractive (negative) potential, while the obstacles push the robot away through a repulsive (positive) potential. The search is guided by trusting the intuitive feeling that the direction of the steepest descent of the potential is the best direction towards the goal: the search proceeds to a neighboring placement that achieves the maximum decrease of the potential. The success of the method clearly depends on adequate choices for the attractive and recursive potential functions. Unfortunately, the search might get stuck in a local minimum of the potential. Considerable efforts are devoted to finding ways to deal with these minima. One direction of research attempts to specify potential functions that cause no or few local minima [55, 82, 83, 53]. Another approach is to develop techniques to escape from local minima [12, 13], for example by random motions. Despite the observed complications due to local minima, potential field methods are efficient in many practical situations. All (known) variants of the potential field approach are approximate, due to fact that steps of a certain minimum size are taken.

Over the past decade, the motion planning problem has attracted the interest of researchers in the field of computational geometry [10, 38, 43, 50, 51, 62, 64, 70, 71, 84, 85, 87, 91, 93]. The explanation for this interest lies in the geometric flavor of the problem which is not only inherent in its statement, but also present in the space in which the motion planning problem is most conveniently solved, the configuration space. The number of degrees of freedom of the robot determines the dimension of this space. The configuration space formulation transforms the motion planning problem into the problem of finding a curve within a subspace, the free space, of the configuration space. The subspace is the union of specific cells in an arrangement of hypersurfaces which are defined by robot-obstacle contacts [39]. The study of arrangements and arrangement cells [32] is one of the main subjects in computational geometry.

The research efforts in motion planning in computational geometry are aimed

The *cell decomposition* approach subdivides (a conservative approximation of) the free space FP into a finite number of simple connected *subcells*, such that planning a motion between any two placements within a single subcell is straightforward and such that uniform crossing rules can be defined for the robot crossing from one subcell into another. Each cell defines a vertex in the *connectivity graph* CG. Two vertices in CG are connected by an edge if their corresponding subcells share a common boundary allowing direct crossing of the robot. Given the connectivity graph CG, the problem of finding a motion between the placements $Z_0$ and $Z_1$ is reduced to a graph problem: find the subcells $C_0$ and $C_1$ in which $Z_0$ and $Z_1$ lie and determine a path in CG between the vertices corresponding to the subcells $C_0$ and $C_1$, or report that no such path exists. Next, the resulting sequence of subcells and the crossing rules for each pair of subsequent subcells are used to transform the sequence into a path for the robot $\mathcal{B}$ from $Z_0$ to $Z_1$. To this end, a point is chosen on the common boundary of each pair of consecutive subcells in the sequence. (The points correspond to unique placements of the robot.) As a result, two points are given in every subcell of the sequence. The imposed simplicity of the subcells facilitates the identification of a continuous curve between the two points that is entirely contained in the subcell. The concatenation of all such curves is a continuous curve between the $Z_0$ and $Z_1$, representing a continuous collision-free motion for the robot between the corresponding placements.

*Exact cell decomposition* methods partition the free space into simple subcells, so that the union of the subcells equals exactly the free space. Examples of exact cell decomposition applied to varying instances of the motion planning problem are found in [43, 50, 64, 84, 85, 86, 87, 91]. Section 1.2 discusses the examples in more detail. *Approximate cell decomposition* methods [19, 35, 48, 60, 103] approximate the free space by a collection of subcells with uniform shapes, for example rectangloids. The union of the subcells is a subset of the free space. Occasional failure to return a path is evident from the difference between the free space and the subcell union. Most approximate methods decompose the free space in a recursive manner, stopping when a subcell is entirely free or entirely non-free and further refining when a subcell contains both types of placements. Physical limitations, like the amount of storage that is available, require the recursive process to stop at a certain level.

The *roadmap* approach to motion planning aims at capturing the structure and connectivity of the free space in some one-dimensional network of curves, the *roadmap*. The availability of the roadmap reduces the planning problem to determining motions between the initial and final robot placements $Z_0$ and $Z_1$ and two placements on the roadmap, and subsequently searching the roadmap for a sequence of curves connecting these two placements. The latter problem is again a graph searching problem if the network of curves is represented as a graph. The sequence of curves resulting from the graph search corresponds directly to a continuous path for the robot in its workspace. Nearly all known roadmap algorithms are exact [20, 62, 70, 71, 93]. They share the property that all roadmap curves in a single connected component of the free space are connected in the roadmap (through

parameter $\beta \in [0, 2\pi)$, being the angle between both links $L_4$ and $L_5$ completes a unique characterization of the placement of $\mathcal{B}_2$. So, the constrained robot $\mathcal{B}_2$ has three degrees of freedom. Any triple $(x, y, \beta) \in \mathbb{R}^2 \times [0, 2\pi)$ represents exactly one placement of the $\mathcal{B}_2$. The triple $(x, y, \beta)$ no longer suffices to uniquely specify a placement of $\mathcal{B}_2$ if the link $L_4$ is allowed to rotate as well. Then, the robot can take infinitely many placements while its joint is placed at $(x, y)$ and the angle between its links $L_4$ and $L_5$ equals $\beta$. The addition of an extra parameter $\gamma \in [0, 2\pi)$, giving the angle between, for example, the link $L_4$ and the positive $x$-axis, solves the problem. Any quadruple $(x, y, \beta, \gamma) \in \mathbb{R}^2 \times [0, 2\pi)^2$ specifies exactly one placement of this unconstrained version of $\mathcal{B}_2$. The robot $\mathcal{B}_2$ has four degrees of freedom. The robot $\mathcal{B}_3$ moving in a three-dimensional workspace is a so-called *rigid* robot consisting of one solid non-deformable link. A triple $(x, y, z) \in \mathbb{R}^3$ fixes the position of some point $p \in \mathcal{B}_3$. While $p$ is placed at $(x, y, z)$, the point $q$ can be chosen to lie anywhere on the sphere with radius $|pq|$ centered at $(x, y, z)$. A pair $(\theta, \phi) \in [0, 2\pi) \times [0, \pi]$ suffices to identify a point on a sphere. Even though the quintuple $(x, y, z, \theta, \phi)$ fixes both $p$ and $q$, the robot $\mathcal{B}_3$ can still be in infinitely many different placements as it is free to rotate around the supporting line of the segment $pq$. One additional parameter $\phi \in [0, 2\pi)$ is enough to model this rotational freedom. Hence, the robot $\mathcal{B}_3$ has six degrees of freedom. Any tuple $(x, y, z, \theta, \phi, \psi) \in \mathbb{R}^3 \times [0, 2\pi) \times [0, \pi] \times [0, 2\pi)$ is a parametric representation of exactly one placement of $\mathcal{B}_3$. We refer to the tuple as a *configuration* of the robot.

The motion planning problem is commonly tackled in the space of these parametric representations of robot placements, or *configuration space* for short. As we will see, the configuration space formulation transforms the motion planning problem into the problem of finding a continuous curve within a subspace, the *free space*, of the configuration space. The free space consists of all placements of the robot in which it intersects no obstacle. The continuous curve in the free space corresponds to a continuous free motion of the robot in the workspace.

Motion planning methods process the free space for the efficient solution of one or more path-finding queries. The methods can be classified according to two, more or less orthogonal, criteria. First al all, a method is either *exact* or *approximate*. Approximate methods, which originate mainly from the robotics community, are often fast and simple to implement. On the other hand, they may occasionally spend a lot of time and storage in finding a path or, worse, fail to find a path, even if one exists. Exact methods, which originate mainly from the computational geometry community, are guaranteed to find a path if one exists. The price to pay for this completeness is generally a considerable increase in computation time. A second subdivision classifies the methods by the type of technique that is used to find a path. Latombe [59] distinguishes three different motion planning approaches: *cell decomposition* methods, *roadmap* methods, and *potential field* methods. The next few paragraphs briefly discuss the essential features of each of the three approaches. Exact and approximate examples of each of the approaches are mentioned, if available.

end is the *tip*, or *hand* [102]. Robots at assembly lines are, in general, robot arms. (Typical assembly robots have approximately six links.) The robots in the difficult environments sketched in the previous paragraph are often not fixed. If, except for possible collisions with the obstacles in the workspace or with itself, the motion of the robot in the workspace is unconstrained, then the robot is *free-flying*. In this thesis, we will mainly deal with free-flying robots.

The unique characterization of any placement of a robot in its workspace involves a certain minimum number of parameters. These parameters are the *degrees of freedom (DOF)* of the robot. Let us consider the examples of robots in Figure 1.1 to get a feeling of the various degrees of freedom of robots. The robot arm $\mathcal{B}_1$ moves

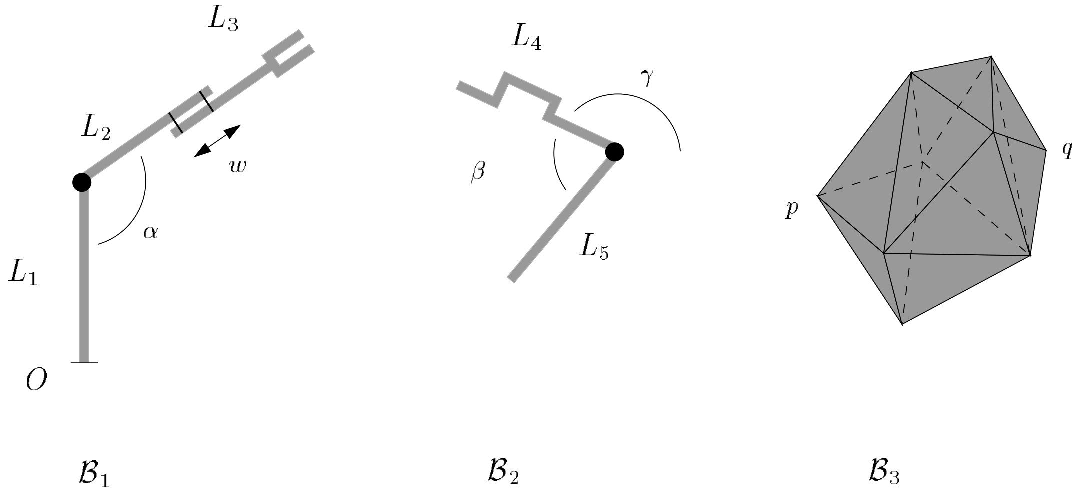


Figure 1.1: Three examples of robots: $\mathcal{B}_1$ is a robot arm in the plane consisting of three links, $\mathcal{B}_2$ is a free-flying articulated robot in the plane consisting of two links, and $\mathcal{B}_3$ is a free-flying rigid robot in three-dimensional space.

in a two-dimensional workspace and consists of three links $L_1$, $L_2$, and $L_3$; the lower end of $L_1$ is fixed at the origin $O$, $L_1$ and $L_2$ are attached to each other by a revolute joint, and $L_2$ and $L_3$ are connected by a prismatic joint; the 'overlap' of the links $L_2$ and $L_3$ at the prismatic joint varies between 0 and $\ell$. The angle between the links $L_1$ and $L_2$, and the length of the overlap of $L_2$ and $L_3$ uniquely define any placement of $\mathcal{B}_1$, so $\mathcal{B}_1$ has two degrees of freedom. Any pair $(\alpha, w) \in [0, 2\pi) \times [0, \ell]$ represents exactly one placement of $\mathcal{B}_1$. As a result, the set of points in the workspace covered by $\mathcal{B}_1$ can be calculated from $(\alpha, w)$, provided that the shapes of the individual links are known. The articulated robot $\mathcal{B}_2$ with links $L_4$ and $L_5$ which are joined by a revolute joint moves in a two-dimensional workspace. Assume, for the moment, that the link $L_4$ is constrained to move at a fixed orientation. In that case, the coordinates $(x, y) \in \mathbb{R}^2$ of, for example, the joint uniquely specify the points covered by the link $L_4$. The orientation of the link $L_5$, however, is still variable. An additional

# Chapter 1

# Introduction

A *robot* is a machine capable of carrying out a complex series of actions automatically (the Concise Oxford dictionary [77]). Over the past years, the use of robots has become common in an increasing number of areas. With the wider range of applications comes a growing need for autonomy of the robots. The earlier generations of robots, encountered for example in assembly lines, mostly execute prescribed (repeating) sequences of uniform actions. As such, they often effectively replace human-beings in routine tasks. More recent and advanced application domains for robots include operation in environments that are dangerous or inaccessible to humans. Among such domains are space exploration, (nuclear) waste handling, and medical surgery. The nature of the robot tasks in these environments requires a high degree of autonomy of the operational robot. The series of actions performed by the robot tends to become less uniform and the descriptions of the tasks will be formulated at a higher level. An ultimate goal in the field of *robotics*[1] inspired by this growing need for autonomy is the development of robots that accept high-level descriptions of tasks and execute these tasks with as little intervention as possible, and ideally without further intervention at all. A fundamental task for such an autonomous robot would be to move from a current placement to another placement while avoiding collision with the obstacles on its way. The *motion planning problem*, that is, the problem of finding such a collision-free path, is the subject of this thesis.

A robot is a movable mechanical device operating in a physical world, the robot's *workspace*. Robots generally consist of one or more bodies, or *links*, that are, in most practical situations, in some way attached to each other. These couplings of the links, which are referred to as *joints*, constrain the relative placements and motions of the attached links. Typical joints are the *revolute* (or rotating) joint and the *prismatic* (or sliding) joint. An *articulated* robot consists of several links that are all connected by joints. If the links of an articulated robot are arranged in a chain and one of the two ends of the chain is fixed at some position, then the robot is an *arm*. The fixed end of an arm is referred to as the *base* of the arm; the other

[1] Robotics is the study of robots or the art or science of their design and operation [77].

# Contents

Promotor: Prof. dr. M.H. Overmars
Faculteit Wiskunde en Informatica

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Stappen, Arnoldus Franciscus van der

Motion planning amidst fat obstacles / Arnoldus Franciscus
van der Stappen. - Utrecht : Universiteit Utrecht,
Faculteit Wiskunde & Informatica
Proefschrift Universiteit Utrecht. - Met index, lit. opg.
- Met samenvatting in het Nederlands.
ISBN 90-393-0654-0
Trefw.: geometrie / robotica / algoritmen.

The research in this thesis was supported by the Netherlands Organization for Scientific Research (N.W.O.), and partially supported by the ESPRIT II BRA Project 3075 (ALCOM) and the ESPRIT III BRA Project 6546 (PROMotion).

# Motion Planning amidst Fat Obstacles

Motion Planning tussen Vette Obstakels

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van
doctor aan de Universiteit Utrecht
op gezag van de Rector Magnificus, Prof. dr. J.A. van Ginkel,
ingevolge het besluit van het College van Decanen
in het openbaar te verdedigen
op vrijdag 7 oktober 1994 des middags te 12.15 uur

door


Arnoldus Franciscus van der Stappen

geboren op 4 oktober 1964
te Eindhoven