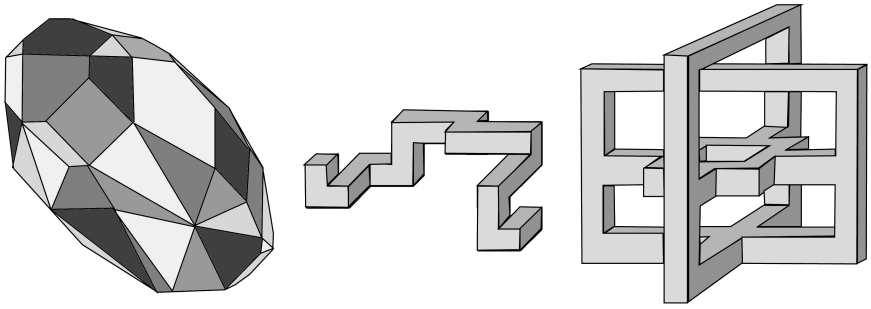

Fig. 1.13 *Left:* a simple polyhedron with randomly-rendered faces. Middle: a simple polyhedron defined in a regular orthogonal 3D grid. *Right:* a non-simple polyhedron (which could be, e.g., the layout for a "world" of a computer game).

The non-simple polyhedron on the right[13] can be generated as a union of five tori. A polyhedral torus cannot be topologically transformed into a sphere.

A point p on the surface of a polyhedron is *visible from the outside* iff there is a ray starting at p that intersects the polyhedron in no other point than at p. A point p in a polyhedron *sees* a point q in this polyhedron iff the straight segment pq is contained in the polyhedron.

Fig. 1.14 *Left:* a tree (at Tzintzuntzan, Mexico) can be modelled as a simple polyhedron, assuming that nature was not producing any torus when growing this tree. *Right:* a sponge can only be modelled as a non-simple polyhedron.

Objects in our real world (see Figure 1.14) can be modelled as polyhedrons by approximating curved surface patches at some selected scale by polygons. The two objects shown are of low shape complexity compared with the dimensions and variations of shapes in the whole universe.

We assume that some integer $n > 0$ characterizes the complexity of input data, such as the number of points in a set, the number of vertices of a polygon, or the

[13] Used by *Johann Benedict Listing* (1808–1882) in 1861 when illustrating the skeletonization of shapes in $\mathbb{R}^3$.

number of faces of a simple polyhedron. The upper limit for n is unknown because progress in technology may shift feasible values further up right now, or in a few years from now. Those values of n are so 'little' compared with the universe, or even to the infinity of the set of all integers. It would make sense to define an upper limit for n, say $n < 10^{80}$, which is the estimated number of protons in the observable universe.[14] We could conclude that all our algorithms only need to work for inputs with $n < 10^{80}$; larger inputs are out of scope.

This would not simplify considerations such as "this algorithm is correct for any $n > 0$", thus also, for example, a proof of correctness for

$$n = 10^{10^{10^{10^{10}}}}$$

We prove results for numbers n which will never be experienced by humankind, just for the sake of mathematical simplicity.

The complexity of input data (e.g., sets of points, polygons, or polyhedrons) will be characterized by integers, without taking into account any limitation for those integers.

1.9 Euclidean Shortest Paths

Euclidean shortest paths (ESP) have been specified in Definition 1.7. In this section we illustrate only a few examples of ESP problems, for demonstrating subjects to be considered in this book. Polygons or polyhedrons are simple in what follows unless otherwise stated.

Obstacles or *attractions* are bounded or unbounded subsets in Euclidean space. When calculating a path, it must not enter any of the obstacles, but it has to visit all the given attractions in a specified order. A path ρ *visits* a set S iff ρ has a non-empty intersection with S.

We also assume a *search domain* that is the space of possible moves: source and destination for the path are in the search domain, all the attractions need to have at least a non-empty intersection with the search domain, and obstacles can possibly be outside of the search domain. See Figure 1.16 for an example.

We consider *polygonal or polyhedral search domains*, defined by a finite number $n \geq 0$ of lines or line segments or polygonal faces, respectively. For example, a half plane or a 3D half space are also possible; they are defined by one straight line in 2D space, or one plane in 3D space (i.e., $n = 1$). Finally, all $\mathbb{R}^2$ or $\mathbb{R}^3$ are also possible, they are defined by having *no* limiting straight line or segment or polygon (i.e., $n = 0$).

[14] See, e.g., `www.madsci.org/posts/archives/oct98/905633072.As.r.html`.

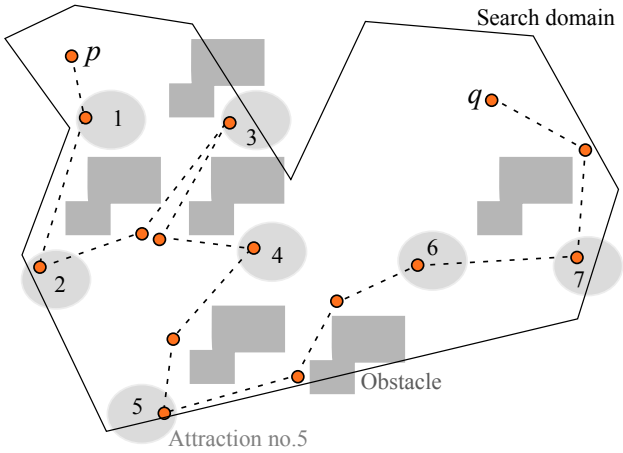

Fig. 1.15 Sketch of an ESP problem in the plane defined by a polygonal search domain, source p and destination q, an ordered set of attractions (shown by shaded ellipses), and a set of obstacles, shown as shaded polygons of constant shape. The figure shows a possible (not yet length minimized) path, connecting p with q via the given sequence of attractions, avoiding all the obstacles, and staying in the given search domain.

Definition 1.14. Assume that the Euclidean space $[\mathbb{R}^2, d_e]$ or $[\mathbb{R}^3, d_e]$ contains a finite set of polygonal or polyhedral obstacles and also an ordered set of polygonal or polyhedral attractions. We consider two points p (the *source*) and q (the final *destination*) within the given polygonal or polyhedral search domain. The *ESP problem*

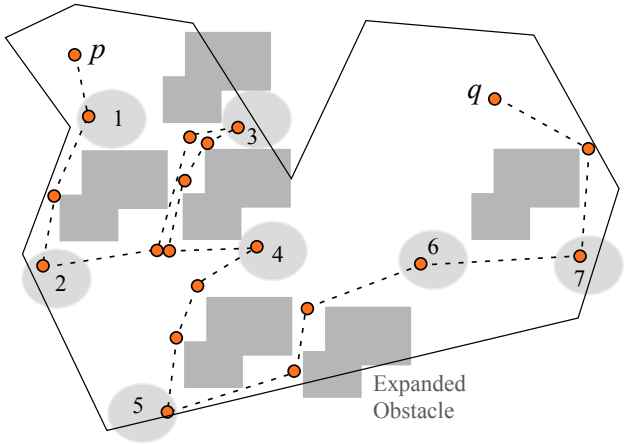

Fig. 1.16 The same ESP problem as in Figure 1.15 but after expanding all the obstacles in x- and y-directions, thus 'giving the moving object less space' . The figure shows a possible path from p to q.

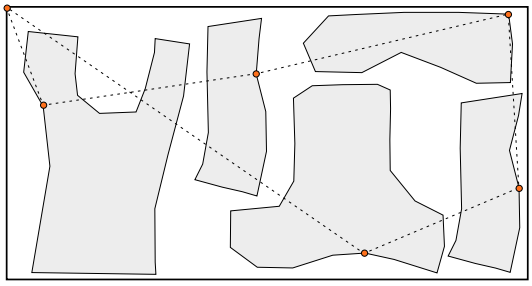

Fig. 1.17 A rectangular sheet with five embedded polygonal shapes (i.e., being the attractions) and a non-optimized path of a cutting head. At each of the five points on frontiers of the shapes, the cutting head would start (and also end) when cutting out a shape from the sheet.

is to compute a path ρ between p and q in such a way that the path ρ does not intersect the interior of any obstacle, visits all the attractions in the specified order, does not leave the search domain, and is of minimum Euclidean length.

An *exact solution* is a path ρ that solves such an ESP problem. See Figure 1.15. Due to a constraint about the size of the object that is expected to move along a calculated shortest path, we may consider expanded obstacles rather than the original obstacles; see Figure 1.16. Vertices of a calculated path, connecting p and q, can move freely in the search domain, not only in the given attractions.

We provide five examples of ESP problems. Those and others are discussed in this book:

(1) Find an ESP between points p and q in a simple polygon (the polygon defines the search domain); there are no obstacles or attractions. This is the well-known problem of *finding a shortest path in a simple polygon*.

(2) Attractions are pairwise-disjoint polygons (see Figure 1.17), all within a rectangular 'sheet' of material (e.g., textile, metal), $p = q$, and there are no obstacles. This is known as a *parts-cutting problem*, where a 'cutting head' needs to travel from one polygonal shape to the other for cutting them 'out' from the rectangular 'sheet'; this rectangle is the search domain.

(3) The search domain is the interior or the surface of a polyhedron, and there are no obstacles or attractions. This defines either the problem of *finding a shortest path*

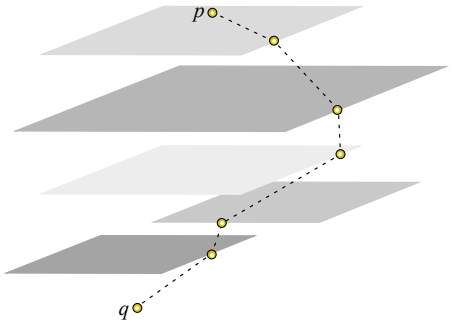

Fig. 1.18 Five stacked rectangles (i.e. being the obstacles) in parallel planes in 3D space. The figure shows a possible path from p to q.

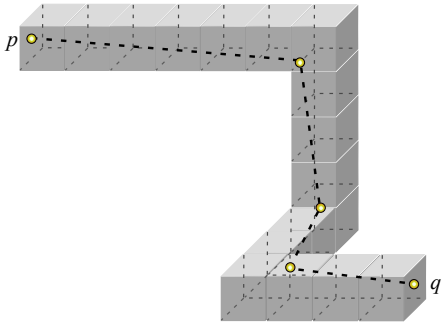

Fig. 1.19 A simple path of cubes and start and end points p and q. The union of the cubes in the path defines a *tube*, and the ESP problem is to find a shortest connection from p to q in this tube.

in a simple polyhedron, or the problem of *finding a shortest path on the surface of a simple polyhedron*.

(4) The whole $\mathbb{R}^3$ is our search domain, and the obstacles are a finite set of *stacked rectangles* (i.e., rectangles in parallel planes), without any attractions. We need to *find a shortest path which avoids the stacked rectangles*. See Figure 1.18.

(5) The 3D space is subdivided into a regular orthogonal grid forming cubes (e.g. voxels in 3D medical imaging, or geometric units in gene modelling). These cubes form a *simple path* iff each cube in this path is face-adjacent to exactly two other cubes in the path, except the two end cubes which are only face-adjacent to one other cube in the path. The two end cubes contain start and end vertex p and q. See Figure 1.19.

A given source p and destination q specify a *fixed ESP problem*. If there is no fixed start or end point, then this defines a *floating ESP problem* with a higher degree of uncertainty, thus a larger computational challenge.

Problems

1.1. Let $g(n) = n \log_{10} n$ and $f(n) = n \log_2 n$. Show that $g(n) \in \mathcal{O}(f(n))$ and $f(n) \in \mathcal{O}(g(n))$ (i.e., both functions are *asymptotically equivalent*).

1.2. Show that the Minkowski distance measure d_1 satisfies all the three axioms **M1**, **M2**, and **M3** of a metric.

1.3. A shortest d_1-path is defined by minimizing the total distance

$$\mathcal{L}(\rho) = \sum_{i=0}^{n} d_1(p_i, p_{i+1}), \quad \text{with} \quad p_0 = p \quad \text{and} \quad p_{n+1} = q$$

between source p and destination q. Figure 1.20 illustrates three (of many) options for shortest d_1-paths between p and q. Specify the area defined by the union of all shortest d_1-paths from grid point p to grid point q. How is the analogous area for Minkowski metric d_∞?

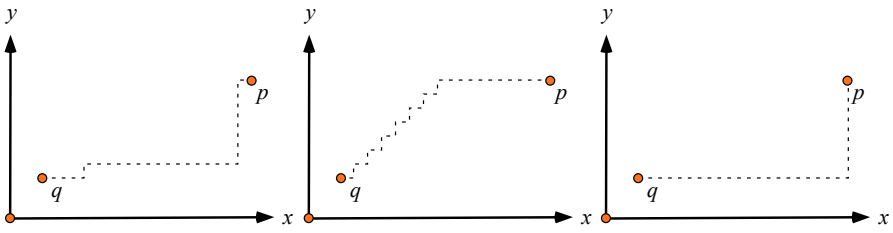

Fig. 1.20 Three shortest d_1-paths between p and q. The coordinate axes define the only two possible directions of moves (also called *isothetic moves*).

1.4. How to define the length of a shortest path between two points p and q in $\mathbb{R}^3$ when applying the forest distance d_f rather than d_e? Generalize the given 2D distance d_f at first to a metric in $\mathbb{R}^3$.

1.5. Do some experiments with the Dijkstra algorithm. Download a source from the net, run it on weighted graphs with different values of $|E|$, and measure the actual run time (e.g., by running it on the same input 1,000 times and divide measured time by 1,000). Generate a diagram which plots values of $|E|$ together with the measured time. After a sufficient number of runs you should have a diagram showing a 'dotted curve'. Discuss the curve in relation to the upper time bounds specified above for the Dijkstra algorithm.

1.6. (Programming exercise)
Implement a program which generates 'randomly' a simple polygon with $n > 0$ vertices, with only specifying the value of parameter n at the start of the program. The program should also contain the option that only the vertices are generated (i.e., a set of points) as a set of n randomly generated points.

Aim at generating a large diversity of shapes of polygons (e.g., more than just star-shaped polygons; see Figure 1.21). For controlling the output of your program,

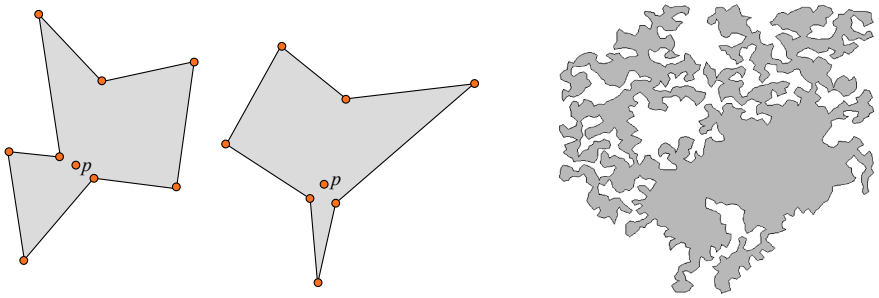

Fig. 1.21 *Left and middle:* two star-shaped polygons (note: in a star-shaped polygon there is at least one point p 'such that a 'guard' at p sees' all the frontier of this polygon). *Right:* a randomly generated simple polygon (courtesy of Partha Bhowmick, Indian Institute of Technology Kharagpur, India).

draw the resulting polygons on screen (e.g., by drawing with OpenGL in an OpenCV window). The generated polygons will be useful for testing ESP programs later on, for programming exercises listed in subsequent chapters.

References

For algorithm design in general, see, for example, the books [5, 10, 19]. Finding a general solution to the general ESP problem (starting with dimension $m = 3$, without predefined areas of destination, and with having the whole $\mathbb{R}^m$ as search space) is known to be NP-hard [3]. The survey article [16] informs about ESP algorithms for 3D space. There are arithmetic δ-approximation algorithms solving the Euclidean shortest path problem in 3D in polynomial time, see [4].

Shortest paths or path planning in 3D robotics (see, for example, [12, 13], or the annual ICRA conferences in general) is dominated by decision-theoretic planning, stochastic algorithms, or heuristics. Describing a robot with its degrees of freedom (in moving) requires a high-dimensional space; the robot is a point of parameters in this space. ESP algorithms need then to be applied to those high-dimensional spaces. The problem is that there is typically no prior complete knowledge about the space; when going from p to q it might be just possible to notice that a straight path is not possible because there are some obstacles in the way (without having complete geometric knowledge about the scene).

Path planning is the subject of the book [12]. The book contains illustrations of shortest paths depending upon the chosen distance measure, and provides, in general, a very good connection between geometric problems and various robotics applications. The calculation of Euclidean shortest paths is not a subject in [12].

The pioneering paper for breadth-first search is [17]. The calculation of shortest paths is also a subject in graph theory [5] (Chapters 24 and 25); here, a shortest path connects vertices in a given graph, where edges of the graph are labelled by weights; this situation differs from the Euclidean shortest path problem, where possible vertices are not within a predefined finite set, and thus we also do not have a finite set of predefined weights (representing distances). Dijkstra's algorithm was published in [6]. The heuristic $A^\star$- search algorithm was published in [7].

For approximation algorithms, see the books [1, 8, 9, 14, 20] or the website [18]; so-called "absolute" or "relative approximation" schemes are basically not much different from the discussed concept of δ-approximation. Approximation algorithms for ESP calculations are specified in [15].

For random curve generation, as addressed in Problem 1.6, see [2].

1. Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi (1999). *Complexity and Approximation*. Springer, New York.
2. Bhowmick, P., O. Pal, and R. Klette (2010). A linear-time algorithm for the generation of random digital curves. In Proc. *PSIVT*, IEEE CS Press, pages 168–173.
3. Canny, J., and J. H. Reif (1987). New lower bound techniques for robot motion planning problems. In Proc. *IEEE Conf. Foundations Computer Science*, pages 49–60.

4. Choi, J., J. Sellen, and C.-K. Yap (1994). Approximate Euclidean shortest path in 3-space. *Proc. ACM Conf. Computational Geometry*, pages 41–48.
5. Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2001). *Introduction to Algorithms*, 2nd edition. The MIT Press, Cambridge, Massachusetts.
6. Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, **1**:269–271.
7. Hart, P. E., N.J. Nilsson, and B. Raphael (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science Cybernetics*, **4**:100–107.
8. Hromkovič, J. (2001). *Algorithms for Hard Problems*. Springer, Berlin, New York.
9. Hochbaum, D.S., editor (1997). *Approximation Algorithms for NP-Hard Problems*. PWS Pub. Co., Boston.
10. Kleinberg, J. and E. Tardos (2005). *Algorithm Design*. Pearson Education, Toronto.
11. Latombe, J.-C. (1991). *Robot Motion Planning.* Kluwer, Boston.
12. LaValle, S. M. (2006). *Planning Algorithms*. Cambridge Univ. Press, Cambridge, UK.
13. Li, T.-Y., P.-F. Chen, and P.-Z. Huang (2003). Motion for humanoid walking in a layered environment. In Proc. *Conf. Robotics Automation*, volume 3, pages 3421–3427.
14. Mayr, E. W., H. J. Prömel, and A. Steger, editors (1998). *Lectures on Proof Verification and Approximation Algorithms*. Springer, Berlin, New York.
15. Mitchell, J. S. B. (2000). Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701, Elsevier.
16. Mitchell, J. S. B., and M. Sharir (2004). New results on shortest paths in three dimensions. In Proc. *SCG*, pages 124–133.
17. Moore, E. F. (1959). The shortest path through a maze. In Proc. *Int. Symp. Switching Theory*, volume 2, pages 285–292, Harvard University Press, Cambridge, Massachusetts.
18. Rabani, Y. (2004). Approximation algorithms. See `http://www.cs.technion.ac.il/~rabani/236521.04.wi.html`, version: October 28, 2004.
19. Skiena, S. S. (1998). *The Algorithm Design Manual*. Springer, New York.
20. Vazirani, V. V. (2001). *Approximation Algorithms*. Springer, Berlin, New York.