Where am I? Indoor Localization Based on Range Measurements

Dissertation
zur Erlangung
des naturwissenschaftlichen
Doktorgrades
der Bayerischen
Julius-Maximilians-Universität
Würzburg

vorgelegt
von **Oliver Karch**
aus Karlstadt am Main

Würzburg
im Dezember 2002

Dissertation zur Erlangung
des naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

Eingereicht am 2. Dezember 2002
bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr. Hartmut Noltemeier
2. Gutachter: Prof. Dr. Rolf Klein

Tag der mündlichen Prüfung: 1. August 2003

Für Karin, Lena und Hannes

Abstract

Nowadays, robotics plays an important role in increasing fields of application. There exist many environments or situations where mobile robots instead of human beings are used, since the tasks are too hazardous, uncomfortable, repetitive, or costly for humans to perform. The autonomy and the mobility of the robot are often essential for a good solution of these problems. Thus, such a robot should at least be able to answer the question "Where am I?".

This thesis investigates the problem of self-localizing a robot in an indoor environment using range measurements. That is, a robot equipped with a range sensor wakes up inside a building and has to determine its position using only its sensor data and a map of its environment. We examine this problem from an idealizing point of view (reducing it into a pure geometric one) and further investigate a method of Guibas, Motwani, and Raghavan from the field of computational geometry to solving it. Here, so-called visibility skeletons, which can be seen as coarsened representations of visibility polygons, play a decisive role.

In the major part of this thesis we analyze the structures and the occurring complexities in the framework of this scheme. It turns out that the main source of complication are so-called overlapping embeddings of skeletons into the map polygon, for which we derive some restrictive visibility constraints. Based on these results we are able to improve one of the occurring complexity bounds in the sense that we can formulate it with respect to the number of reflex vertices instead of the total number of map vertices. This also affects the worst-case bound on the preprocessing complexity of the method.

The second part of this thesis compares the previous idealizing assumptions with the properties of real-world environments and discusses the occurring problems. In order to circumvent these problems, we use the concept of distance functions, which model the resemblance between the sensor data and the map, and appropriately adapt the above method to the needs of realistic scenarios. In particular, we introduce a distance function, namely the polar coordinate metric, which seems to be well suited to the localization problem. Finally, we present the RoLoPro software where most of the discussed algorithms are implemented (including the polar coordinate metric).

Contents

1 Introduction

"What's the meaning of it?
Where am I?"
she said in complete bewilderment,
as though still unable to recover herself.

Sonia in "Crime and Punishment"
F. DOSTOYEVSKY (1821–1881)

NOWADAYS, ROBOTICS PLAYS AN IMPORTANT ROLE in increasing fields of application. Mobile robots are typically used in environments or situations where humans cannot operate or are not willing to operate, since these tasks are too hazardous, uncomfortable, repetitive, or costly for humans to perform. Some exemplary applications in different fields are listed below.

Services sector Here we have in mind cleaning tasks or the guidance, assistance, and entertainment of people.

For example, Siemens' experimental ROAMER robot shown in Figure 1.1 on the following page is used for cleaning supermarkets or offices, see [Ren94, RFZ99, FBL94].

Another realized application in this field is the interactive museum tour-guide robot RHINO of the University of Bonn and the Carnegie Mellon University (depicted in Figure 1.2 on the next page), which acts as a tour guide for visitors of the Deutsches Museum Bonn, see [BC^{+}99, Fox98, FBT99, Rhi].

Both examples have in common that the robots have to operate in crowded environments full of people, trollies, etc.

Office work Possible jobs are the delivery of mail, tedious cleaning tasks, or maintenance work.

The Mobile Post Distribution System MOPS of the ETH Zürich [VTG96, TGVS99] may serve as an example of mail delivery in office environments.

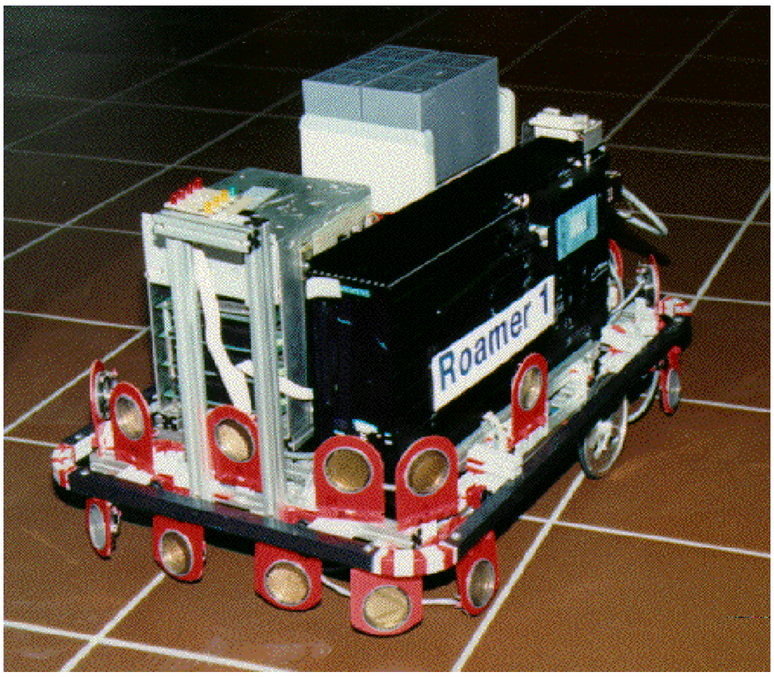


FIGURE 1.1: Siemens' experimental cleaning robot ROAMER (photo taken from [BEF96])

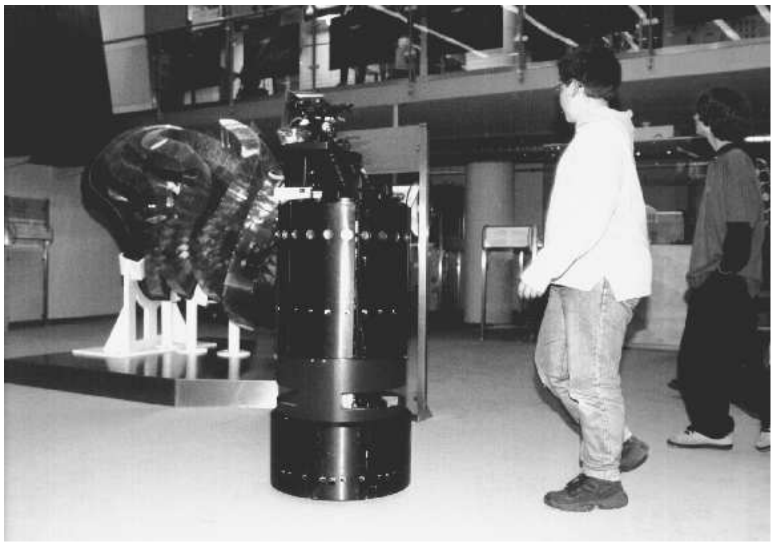


FIGURE 1.2: The interactive museum tour-guide RHINO (photo taken from [Fox98])

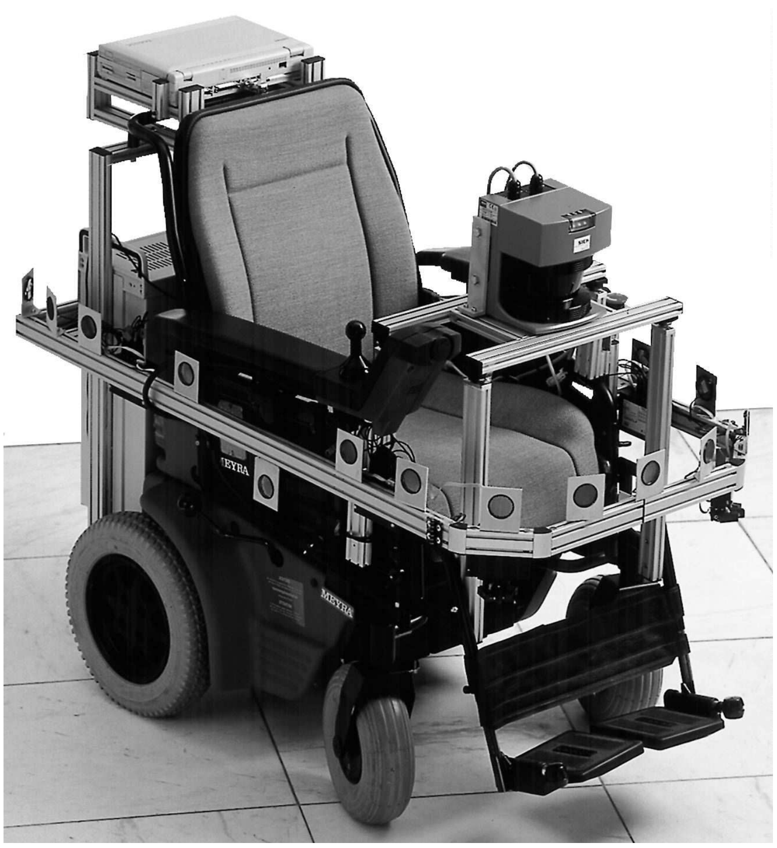


FIGURE 1.3: The robot wheelchair MAID (photo taken from [PS+99])

Medical sector Problems occurring in this field of application are the transportation of patients, the assistance for disabled or elderly people, or surgery assistance tasks.

For instance, the robotic wheelchair MAID of the FAW Ulm shown in Figure 1.3 serves as a transportation aid for people with severely impaired gross motor skills, see [PS+99, PSS98].

Moreover, Yairi and Igi [YI00] propose a moving support system in a broader context, which assists aged and disabled persons not only in transportation tasks, but also compensates for their impaired recognition, actuation, and information access.

Outer space The maintenance of space crafts or exploration of foreign planets are two key problems in this sector.

The mobile robot SOJOURNER shown in Figure 1.4 on the next page, which explored the vicinity of the Mars lander and took about 500 photos within the

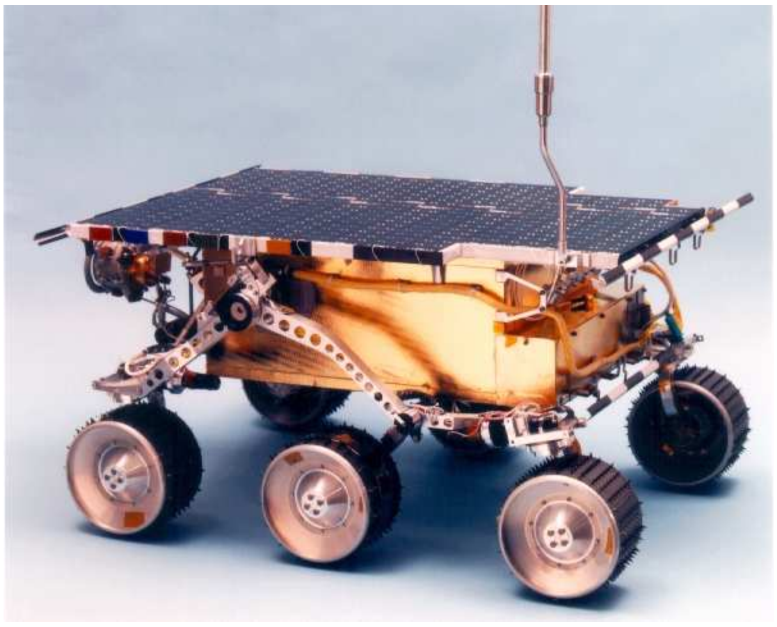


FIGURE 1.4: The Mars rover SOJOURNER (photo taken from [MRF])

Pathfinder mission of the NASA (see [VO+97, Sto96, MRF]) may serve as an example.

Resource industry Here, possible fields of applications of mobile robots are forestry [FM97], farming [OS97], mining, exploration and surveillance [SR+00].

Underwater operations Probable applications are the usage of mobile robots for mining, exploration, search and rescue, as well as maintenance problems.

Power plants and factories Typical jobs are operating, maintenance, and decommissioning work in hazardous or uncomfortable environments.

Civil security Here we have in mind dangerous operations like police work, mines and bomb disposal [Cas00], or firefighting.

Although some of these tasks can also be accomplished by using a telecontrolled robot (see [DM95, p. 304 ff.] for a survey on telerobotics), the autonomy and the mobility of the robot are often essential for a good solution of the respective problem. Moreover, in many applications the communication with remotely controlled manipulators or vehicles is long-winded and error-prone. For instance, according to [MFS] the distance between the planet earth and a remotely controlled robot on Mars varies between $5.5 \cdot 10^7$ km and $4.0 \cdot 10^8$ km, which results in a signal propagation delay between 182 s and 1339 s. This means that the worst-case delay between a command to the robot and its response is about 44 minutes, which emphasizes the benefits of an autonomously operating vehicle.

Furthermore, there also exist applications of mobile robots, where the capability of performing low-level actions (e. g., *"Follow the wall"*, *"Pass the doorway"*, etc.) as well as autonomous high-level actions (e. g., *"Bring me to platform 5 at the railway station"*) is absolutely necessary. An example may be the autonomous wheelchair for disabled persons described above.

Since an autonomous mobile robot has to navigate in its environment, it must at least be able to answer the three following questions, summarized by Leonard and Durrant-Whyte [LDW91]:

"Where am I?",
"Where am I going?", and
"How should I get there?".

At first, these questions appear to be relatively simple to answer, but in fact finding good solutions is very hard. Concerning the first question, Borenstein et al. [BEF96] claim that

> "perhaps the most important result from surveying the vast body of literature on mobile robot positioning is that to date there is no truly elegant solution for the problem."

In the following we concentrate on the first of the three questions above, "Where am I?", that is, on the problem of *localizing a robot*. Some authors also use the term "self localization" instead of "localization" to express that the task is performed by the robot itself, in contrast to problems where someone else has to localize the robot from an outside point of view. In the sequel we only use the shorter term "localization", meaning the robot's task of determining its position and orientation.

1.1 Sensors Used for Localization

A typical robot vehicle may be equipped with a variety of different actuators and sensors, depending on the tasks it has to perform, its size (usually ranging from a few centimeters to one or two meters), and its power supply. Common actuators are, for example, robot arms with gripping pliers, vacuum cleaners, specifically tailored instruments, and devices for certain tasks (confer the examples given above). In the following we describe some common types of sensors, autonomous mobile robots are equipped with.

Odometry is a method to measure the vehicle displacement along the path of travel. For a wheeled locomotion this can directly be accomplished by measuring the wheel rotation and the steering orientation of the robot. This way we can record the path that the robot has travelled so far and provide the robot with an estimate of its position and orientation. Unfortunately, the position and orientation estimate from odometry is corrupted by inevitable drifts if the robot travels long distances without recalibrating.

This method of estimating the present position by advancing a previous known position using course, speed, time, and distance to be travelled is called *dead reckoning* (derived from either "deduced reckoning" or "reckoning a position relative to something stationary or dead in the water" from maritime navigation) and the majority of todays land-based mobile robots rely on it. For surveys on odometry and other dead-reckoning methods see the books of Borenstein et al. [BEF96, p. 13 ff., 130 ff.] and Everett [Eve95, p. 35 ff.].

Gyroscopes and compasses measure the rotation (or orientation, respectively) much more exactly than odometry does, and therefore may be used to augment the dead-reckoning information of mobile robots. But as well as odometry gyroscope data also drifts (with time), even if the robot does not move at all. Therefore, gyroscopes must also be periodically recalibrated using an independent reference [BEF96, p. 30 ff.].

In contrast to gyroscopes, geomagnetic sensors (i. e., instruments that measure the magnetic field of the earth) have no drift, but they cause other problems, for example, if the magnetic field is distorted near power lines [BEF96, p. 45 ff.].

Ultrasonic sensors determine distances to objects in front of the sensor by emitting and receiving a ultrasonic signal. Basically, there are two different approaches to estimate the distance to the surrounding objects (for details see [BEF96, p. 95 ff.] and [Ada99, p. 28 ff.]).

- The time-of-flight technique uses pulses of energy, for which the time between emission and reception is measured.
- The (amplitude modulated) phase-shift measurement technique involves a continuous wave transmission. The reflected portion of the wave is compared to a reference signal and the relative phase shift of the two signals is used to determine the distance.

Since ultrasonic waves are likely to be reflected at smooth surfaces, the distance measurements are not very accurate due to multiple reflections. Consequently, ultrasonic sensors are often used only to avoid collisions with obstacles [FBL94, LD97, Ren94, Soi97a, Soi97b].

Laser scanners work similar to ultrasonic sensors, but use a laser beam instead of ultrasonic waves. In particular, the two distance measuring techniques for ultrasonic sensors described above also exist for laser scanners: the time-of-flight technique and the phase-shift measurement technique. Furthermore, as an alternative also a frequency modulated technique exists, where the frequency of the transmitted continuous wave linearly varies with time. The three methods are described in detail by Adams [Ada99, p. 40 ff.].

Due to the much shorter wave lengths the distance measurements using a laser scanner normally are much more precise than using ultrasonic sensors. Therefore, laser scanners are extremely suitable for localization applications, which are based on range sensing, in particular with regard to their low price. Many authors also use synonyms for the term "laser scanner", either ladar (from "Laser Detection and Ranging") or lidar (from "Light Detection and Ranging").

A typical laser scanner takes a 180°-scan of its environment with an angle increment of 0.5°, that is, one scan consists of 360 individual range measurements (or *scan points*, synonymously). The scan points typically have a resolution of about 10 mm and objects are detected within a range of about 80 m.

Cameras provide digitized pictures of the robot's environment, from which relevant features (e. g., walls, doorways, etc.) can be extracted and used for localization and navigation purposes [LBG97]. Furthermore, cameras are commonly used for tracking moving objects and people [ASV97, DHS96].

Additionally, the robot may make use of systems that allow an absolute position measurement. This could be, for example, active beacons that usually transmit light or radio signals, whose direction of incidence is measured by the robot. From three or more such measurements the robot can compute its absolute position, provided that the transmitters are located at known sites in the environment. Similarly, artificial landmarks that are placed at known positions and which can be recognized by the robot's sensors, can be used. For example, reflecting stripes (for the usage with laser scanners) or signs on the walls with special shapes on them (for vision sensors) could be placed in the environment. Another way of performing an absolute localization is to use the Navstar Global Positioning System (GPS), where 24 satellites, which orbit the earth, transmit radio signals that allow a position measurement with an error of about 3 m; for details see [BEF96, p. 70 ff.]. But for indoor-applications (like the ones that we have in mind) GPS is less applicable, since the GPS signals are likely to be absorbed by the walls of buildings.

In the following we do not consider localization methods using this kind of absolute position measurement, since often it is not possible or too expensive to modify the environment this way. Furthermore, for the long-term objective of an autonomous robot it is not reasonable to rely on modifications to the environment.

Typically, an autonomous robot is equipped with odometry and gyroscopes for maintaining a rough position estimate, ultrasonic sensors for collision avoidance, and laser scanners for the localization task. Of course, also the ultrasonic sensors and possibly additional sensors can be used for localization, for example, by integrating their data using sensor fusion techniques [SW99, KZK97, XvB$^+$95]. A very common method for integrating several measurements of possibly different sensors is the Kalman filtering technique [May90, Kal60]. The Kalman filter combines several estimates of a variable, for which a Gaussian distributed error is assumed, into a new estimate (also with a Gaussian error distribution), such that the variance is minimized.

1.2 Absolute versus Relative Localization

Generally one must distinguish between two different types of localization problems: the *relative* localization problem, where the robot already has an estimate of its position and orientation (e. g., using its odometry), and the *absolute* localization problem, where the robot has no knowledge about previous configurations. For the former problem many approaches exist, which can be classified into scan-based and feature-based approaches, techniques using artificial landmarks and a few other methods, see for example [PM95, LM97, LD97, EW95, Cox91].

The latter problem can be seen as a kind of wake-up situation (e. g., after a power failure or maintenance works), where the robot is placed somewhere in its environment, switched on, and then "wants" to know where it is located. Since here the search space usually is much greater than for a relative localization query, we cannot expect to solve an absolute localization query as efficiently as a relative one. The methods for answering such queries are similar to the ones mentioned above for the relative localization and can be classified the same way: scan-based and feature-based approaches and methods using artificial landmarks. See [RFZ99, Klu99, BK$^+$99, KNS00, WJvP00] for examples of feature-based techniques.

1.3 Thesis Outline and Summary of Results

This thesis investigates the problem of localizing a robot in an indoor environment using range measurements. We examine this problem from an idealizing point of view and reduce it to a pure geometric one, which can then be tackled using methods from the field of computational geometry.

The organization of this thesis is as follows: In Chapter 2 we give a short overview of the notational framework and the basic terminology used throughout this thesis.

Chapter 3 introduces a geometric version of the robot localization problem. We specify a number of idealizing assumptions that allow us to formulate it as the following geometric problem: For a given map polygon and a star-shaped visibility polygon determine all points in the map whose visibility polygon equals the given one. This problem was already investigated by Guibas et al. and we briefly recall their scheme, since it is the basis for our own considerations in the succeeding chapters.

In Chapter 4 we further investigate a central element of the previously described method, namely *embeddings of visibility skeletons*. In particular, we examine the case of so-called *overlapping embeddings*, that is, two embeddings of the same skeleton that overlap. We prove some properties of their intersection region and analyze the different types of intersecting skeleton edges. The main contribution of this chapter describes the connection between two overlapping embeddings and the visibility of their witnesses. We investigate this connection first for the special case where also the kernels of the embeddings overlap and then derive the visibility constraints also for the general case of arbitrary overlapping embeddings.

This property of overlapping embeddings (besides that it represents an interesting problem by itself) also plays a crucial role in Chapter 5, where we further investigate the complexity of a certain structure, namely the *equivalence class* of a skeleton. We refine the bound on this complexity in the sense that we formulate it with respect to the number of reflex vertices instead of the total number of map vertices. This also affects the worst-case bound on the preprocessing complexity of the above method.

The objective of Chapter 6 is to adapt the solution of the idealized problem (described in Chapter 3) to the properties of real-world environments such that the restrictive assumptions can be loosened. To this end, we first discuss the problems that occur if we try to use the computational geometry solution also for realistic scenarios. In order to circumvent these problems we introduce an approach that uses distance functions, which model the resemblance between the sensor data and the map. In particular, we introduce the polar coordinate metric for star-shaped polygons and show how this polygon distance can be used in our setting. Finally, we present the implementation ROLOPRO where our approach was implemented.

1.4 Acknowledgements and Credits

I am greatly indebted to all the persons who supported my work and contributed in various ways to this thesis. First of all, I would like to thank my advisor, Professor Hartmut Noltemeier, who set the course of my line of research. Without his inspiring suggestions and his patient guidance during the last years this work would not have been possible. I want to thank explicitly Hans-Christoph Wirth for his extensive review of my work, the numerous suggestions he made on all parts of it, and the inspiring discussions about computer science and beyond. Riko Jacob and Fiona Jeffery also did detailed reviews and gave competent advice, which I greatly appreciate. My sincere thanks go to Thomas Wahl for our fruitful discussions and the pleasure to work with him. He also did the major part of the implementation of ROLOPRO and the approach presented in Chapter 6 is based on a joint work [Wah97, KW99]. Special thanks go to my former colleagues Sven Oliver Krumke and Knut Verbarg for their support and advice at the beginning of my scientific profession. Further I am thankful to the German research foundation "Deutsche Forschungsgemeinschaft" (DFG) for supporting a large part of my work.

2

Preliminaries

"In the name of heaven, sir," cried she,
"what means all that is passing? Put an end to my doubts;
I have courage enough for any danger I can foresee,
for every misfortune which I understand.
***Where am I**, and why am I here?*
If I am free, why these bars and these doors?
If I am a prisoner, what crime have I committed?"

Milady de Winter in "The Three Musketeers"
A. DUMAS (1802–1870)

GENERALLY WE ASSUME that the reader is acquainted with the standard notations and definitions from computational geometry and related fields. However, in the following sections we give a brief overview of the basic terminology used in the succeeding sections.

2.1 List of Basic Notations

Numbering Conventions

In order not to get confused with the numbers of figures, definitions, theorems, etc., note that each of the following groups is separately and consecutively numbered throughout each chapter:

- Figures and algorithms;
- definitions, assumptions, and observations;
- theorems, lemmas, and corollaries;
- examples and comments.

Furthermore, the former three of these groups are listed on page 171 ff.

Symbols, Sets, and Abbreviations

By $\mathbb{N}$ ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, respectively) we denominate the set of natural numbers (integers, rational numbers, real numbers, respectively), whereas $\mathbb{N} := \{1, 2, \ldots\}$ and $\mathbb{N}_0 := \{0, 1, \ldots\}$. With an expression like $A_{\geq \alpha}$ we denote the subset $\{ a \in A : a \geq \alpha \}$ of the set A. For example, $\mathbb{R}_{\geq 0}$ stands for the nonnegative real numbers.

The closed real interval $\{x \in \mathbb{R} : a \leq x \leq b\}$ is written as $[a, b]$. Likewise, the open interval $]a, b[$ and the half-open intervals $]a, b] := \{x \in \mathbb{R} : a < x \leq b\}$ and $[a, b[$ are defined.

For two sets A and B we write $A \subseteq B$ if A is a subset of B, and $A \subsetneq B$ if A is a proper subset of B. By A^k we denote the k-fold Cartesian product $A \times \cdots \times A$. The set of subsets of A is depicted by 2^A and $\varnothing$ stands for the empty set.

The universal quantor »$\forall_{a \in A}$ …« and the existence quantor »$\exists_{a \in A}$ …« mean »For all $a \in A$ it holds that …« and »There exists at least one $a \in A$ with …«. Likewise, »$\nexists_{a \in A}$ …« stands for »There does not exist any $a \in A$ with …«.

The abbreviation »w. l. o. g.« stands for »without loss of generality«.

Growth of Functions

To describe and compare the asymptotic behavior of two given functions $f, g \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ we use the so-called *Big-Oh notation*: We say that $f \in O(g)$ if there exist constants $c \in \mathbb{R}_{>0}$ and $n_0 \in \mathbb{N}$ such that $\forall_{n \geq n_0}\ f(n) \leq c \cdot g(n)$. That is, the function g is an asymptotic upper bound for the function f.

Conversely, g is a lower bound for f, denoted by $f \in \Omega(g)$, if and only if $g \in O(f)$. We say that the functions *f and g are of the same order*, denoted by $f \in \Theta(g)$, if $f \in O(g)$ and $f \in \Omega(g)$. Moreover, we write $f \in o(g)$ and mean that f is of asymptotically smaller order than g if for any constant $c \in \mathbb{R}_{>0}$, there exists a constant $n_0 \in \mathbb{N}$ such that $\forall_{n \geq n_0}\ f(n) < c \cdot g(n)$ holds *(Little-Oh notation)*. And finally, we say that f is of asymptotically higher order than g and write $f \in \omega(g)$ if and only if $g \in o(f)$.

Topology

For any set A of objects, a *metric* on A is a function $d \colon A \times A \to \mathbb{R}$, which satisfies the following three conditions for all $x, y, z \in A$, cf. [Cop68, VH01]:

(i)	$d(x, x) = 0$	(identity)
(ii)	$d(x, y) = 0 \implies x = y$	(uniqueness)
(iii)	$d(x, y) + d(x, z) \geq d(y, z)$	(triangle inequality)

The set A with metric d is then called a *metric space*. From (i) and (iii) follows the symmetry of d, that is, $d(x,y) = d(y,x)$, by means of $d(y,z) \leq d(z,y) + d(z,z) = d(z,y) \leq d(y,z) + d(y,y) = d(y,z)$. If the function d only satisfies (i) and (iii), it is called a *semimetric*. Note that (i) and the alternative triangle inequality

$$\text{(iii')} \qquad d(x,y) + d(y,z) \geq d(x,z) \qquad \text{(weak triangle inequality)}$$

do not imply symmetry, which then has to be satisfied separately:

$$\text{(iv')} \qquad d(x,y) = d(y,x) \qquad \text{(symmetry)}$$

From (i), (iii'), and (iv') follows that any (semi)metric is nonnegative: $d(x,y) + d(y,x) \geq d(x,x) = 0$, thus $d(x,y) \geq 0$.

The set $\mathbb{E}^d$ stands for the d-dimensional Euclidean vector space, that is, $\mathbb{R}^d$ equipped with the Euclidean norm

$$\|x - y\|_2 := \sqrt{\sum_{i=1}^{d} (y_i - x_i)^2}$$

and the Euclidean metric $d_2(x,y) := \|x - y\|_2$, respectively.

For a set $A \subseteq \mathbb{E}^d$ we denote with A° ($\overline{A}$, $\partial\, A$, respectively) its interior (closure, boundary, respectively). See [Cop68] for an extensive survey of metric spaces.

2.2 Computational Geometry

Geometry is a quite old branch of mathematics. In ancient times famous scientists like Archimedes, Euclid, Pythagoras, and Thales already dealt with geometric problems, see for example Euclid's »*Elements*« [Joy96], which was the basis for geometry for about 2000 years. Not until Descartes introduced the (Cartesian) coordinates, could geometric problems be expressed in an algebraic form, which offered new ways to resolve geometric problems by solving the associated algebraic equations. In recent times using powerful computers it became possible not only to prove theorems about properties of geometric objects, but also to actually compute configurations of many of such objects in reasonable time. More and more fields of application rely on the extensive use of computers together with geometric algorithms, for example, computer graphics, robotics, CAD/CAM applications (computer aided design and manufacturing), and many more.

Therefore, it was inevitable that in the late 1970s a new branch *computational geometry* emerged from the fields of algorithm design and geometry, which concentrates on

- developing *efficient* and *practicable* algorithms for solving geometric problems and on
- determining the inherent complexity of such problems.

Some basic textbooks on computational geometry are those of de Berg et al. [dBvK+97], Klein [Kle97], O'Rourke [O'R98], and Preparata and Shamos [PS85]. A textbook about the strongly related field of *combinatorial geometry* is that of Edelsbrunner [Ede87]. See also the survey of Yao [Yao90], which outlines both fields.

In the following we recall some important definitions concerning planar geometry, which are used in the succeeding sections and with which the reader should be familiar.

Basic Planar Geometry

For two points $p, q \in \mathbb{E}^2$ in the Euclidean plane, the *straight line segment* $\overline{p\,q}$ is defined as $\overline{p\,q} := \{p + a(q-p) : a \in [0,1]\}$, the *ray* $\overset{\longmapsto}{p\,q}$ from p in direction q is defined as $\overset{\longmapsto}{p\,q} := \{p + a(q-p) : a \in \mathbb{R}_{\geq 0}\}$, and the *straight line* $\overleftrightarrow{p\,q}$ through p and q is defined as $\overleftrightarrow{p\,q} := \{p + a(q-p) : a \in \mathbb{R}\}$.

Every (oriented) straight line $l = \overleftrightarrow{p\,q}$ (analogously ray or segment) induces two half planes, the (closed) *positive half plane* of l, denoted by

$$H^+(l) := \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{E}^2 : x_1(p_2 - q_2) + x_2(q_1 - p_1) + (p_1 q_2 - p_2 q_1) \geq 0 \right\},$$

which consists of all points to the left of l, and the (closed) *negative half plane* of l, denoted by

$$H^-(l) := \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{E}^2 : x_1(p_2 - q_2) + x_2(q_1 - p_1) + (p_1 q_2 - p_2 q_1) \leq 0 \right\}$$

consisting of all points to the right of l. Likewise, we define the two open half planes $H^+(l)^\circ$ and $H^-(l)^\circ$ by replacing the inequalities above by strong inequalities »$\ldots > 0$« and »$\ldots < 0$«, respectively.

For two intersecting (oriented) straight lines, rays, or segments a and b we denote the *angle from a to b* by $\sphericalangle(a, b)$, with $0 \leq \sphericalangle(a, b) < 2\pi$ and counterclockwise counting.

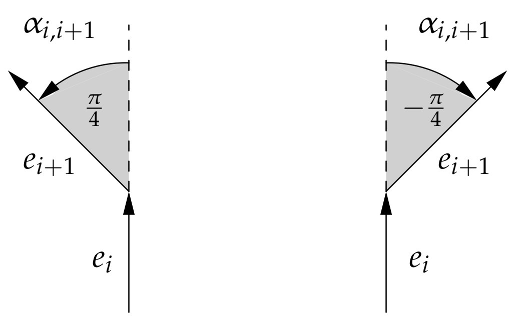


FIGURE 2.1: The angle of rotation between e_i and e_{i+1}

For a given point $p \in \mathbb{E}^2$ we write $\gg\!\prec_p\!\ll$ to describe the (counterclockwise) *angular order* of points $x, y \in \mathbb{E}^2 \setminus \{p\}$ with respect to p. Analogously, we write $x =_p y$ if and only if the rays $\overrightarrow{p\,x}$ and $\overrightarrow{p\,y}$ are identical, and $x \preccurlyeq_p y$ if and only if either $x \prec_p y$ or $x =_p y$. If not stated otherwise, we use the positive axis of abscissa as starting direction, that is, the points in $\{p + \binom{a}{0} : a \in \mathbb{R}_{>0}\}$ are minimal with respect to $\gg\!\prec_p\!\ll$. Thus, using the shortcut $p_{\mathrm{R}} := p + \binom{1}{0}$ we have

$$x \prec_p y :\Leftrightarrow 0 \leq \sphericalangle(\overrightarrow{p\,p_{\mathrm{R}}}, \overrightarrow{p\,x}) < \sphericalangle(\overrightarrow{p\,p_{\mathrm{R}}}, \overrightarrow{p\,y}) \quad \text{and}$$
$$x \preccurlyeq_p y :\Leftrightarrow 0 \leq \sphericalangle(\overrightarrow{p\,p_{\mathrm{R}}}, \overrightarrow{p\,x}) \leq \sphericalangle(\overrightarrow{p\,p_{\mathrm{R}}}, \overrightarrow{p\,y}) \,.$$

If we are only interested in the cyclic sequence of points around p, we ignore the starting direction and call this the *cyclic angular order*.

Polygons

We call a finite sequence c of line segments, $c := \overline{p\,q}, \overline{q\,r}, \ldots, \overline{u\,v}, \overline{v\,w}$, a *polygonal chain* (or *polygonal path*, synonymously) and a *closed* polygonal chain $P := \overline{p\,q}, \overline{q\,r}, \ldots, \overline{v\,w}, \overline{w\,p}$ a *polygon* P. The points $p, q, \ldots, w$ are the *vertices* and the line segments $\overline{p\,q}, \overline{q\,r}, \ldots, \overline{w\,p}$ the *edges* of P.

For two consecutive edges e_i and e_{i+1} of a polygonal chain $e_1, \ldots, e_n$ we denote by $\alpha_{i,i+1}$ the *angle of rotation* of the transition from e_i to e_{i+1}. Thereby, left rotations count positive and right rotations count negative, such that we get

$$\alpha_{i,i+1} := \begin{cases} \sphericalangle(e_i, e_{i+1}) & \text{if } e_{i+1} \subset H^+(e_i)\,, \\ -\sphericalangle(e_{i+1}, e_i) & \text{otherwise}\,, \end{cases}$$

as illustrated in Figure 2.1.

Note that the angle of rotation of two collinear edges e_i and e_{i+1} is zero, that is, $\alpha_{i,i+1} = 0$. Thus, for all $i < n$ the inequality $-\pi < \alpha_{i,i+1} < \pi$ holds. The angle of rotation of two non-consecutive edges e_i and e_k (with $i < k$) is defined as

$$\alpha_{i,k} := \alpha_{i,i+1} + \alpha_{i+1,i+2} + \cdots + \alpha_{k-1,k}\,.$$

Since a polygon $P = e_1, \dots, e_n$ is a *closed* polygon chain, we can define the angle of rotation also for segments e_i and e_k with $i > k$ in the obvious way by considering the chain $e_k, \dots, e_n, e_1, \dots, e_i$ instead of the original one. The total sum of angles of rotation of a complete tour through P is called its *total angle of rotation* and denoted by

$$\alpha_{1,1} := \alpha_{1,2} + \alpha_{2,3} + \cdots + \alpha_{n-1,n} + \alpha_{n,1}\,.$$

The *maximum angle of rotation* between two different edges of a polygon is defined as

$$\alpha_{\max} := \max_{i \neq j} \alpha_{i,j}\,.$$

A polygonal chain or a polygon is called *simple* if the only intersections that occur are between endpoints of consecutive segments. A simple polygon P divides the plane into two distinct regions, its bounded *interior* P° and the unbounded *exterior*.[1] For a simple polygon it can be shown that its total angle of rotation $\alpha_{1,1}$ is either -2π or 2π depending on whether the polygon's edges are traversed clockwise or counterclockwise. In the following we always assume that the polygons are oriented counterclockwise (i. e., $\alpha_{1,1} = 2\pi$). Furthermore, it should always be clear from the respective context, whether we mean with the term "polygon P" the boundary of P along with its interior or only its boundary ∂P.

Two consecutive segments e_i and e_{i+1} of a simple polygon have an *interior angle* of $\pi - \alpha_{i,i+1}$. A vertex of a simple polygon with an interior angle greater than π (i. e., with a negative angle of rotation between the incident edges) is called a *reflex vertex*.

A subset S of $\mathbb{E}^2$ is called *convex* if for every two points $p, q \in S$ the set S also contains the straight line segment $\overline{p\,q}$. The *convex hull* $\mathrm{CH}(S)$ of a subset S of $\mathbb{E}^2$ is the smallest convex set that contains S, that is,

$$\mathrm{CH}(S) := \bigcap_{\substack{C \supseteq S \\ C \text{ convex}}} C\,.$$

It can be shown that the interior of a simple polygon without any reflex vertex is a convex set, such that we call it a *convex polygon*. Moreover, a convex

1 This is a non-trivial consequence of the *Jordan Curve theorem*, see [Mun75].

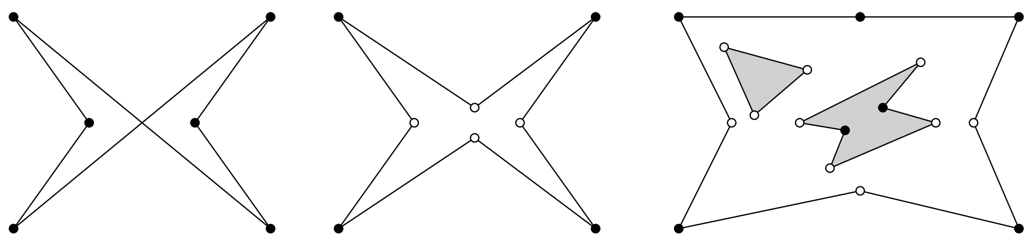


FIGURE 2.2: From left to right: a non-simple polygon with six vertices, a simple polygon with eight vertices (four of it reflex, marked by circles »○«), and a polygon with two holes

polygon $P = e_1, \ldots, e_n$ can also be described as the intersection of the positive half planes that correspond to its edges, that is,

$$P = \bigcap_{i=1}^{n} H^+(e_i)\,.$$

Furthermore, the convex hull of finitely many points and straight line segments always is a convex polygon.

A simple polygon P is called *monotone with respect to a line l* if for any line l' perpendicular to l the intersection of P with l' is connected. That is, the intersection is either a straight line segment, a point, or empty.

A simple polygon P together with simple polygons $H_1, \ldots, H_k \subseteq P^\circ$, which lie in the interior of P and do neither pairwise intersect nor contain each other (i. e., $\overline{H_i} \cap \overline{H_j} = \varnothing$ for $i \neq j$), is called a *polygon with k holes* $H_1, \ldots, H_k$. Figure 2.2 illustrates the above definitions. Note that the role of reflex and non-reflex vertices is reversed in the case of holes. For example, the right hole of the polygon in Figure 2.2 has four reflex and two non-reflex vertices. In the following we use the term "polygon" as a short hand for a simple polygon without holes, if not stated otherwise.

Visibility

We say that two points $p, q \in P$ inside a polygon P *see* each other (or are *visible* to each other, synonymously) if the line segment between p and q has no point in common with the boundary of P, except for its endpoints, that is, $\overline{pq} \cap \partial P \subseteq \{p, q\}$. Figure 2.3 (a) on the following page shows an example: The two polygon vertices v and w see each other, whereas the line of sight between p and w is blocked by the vertex v.

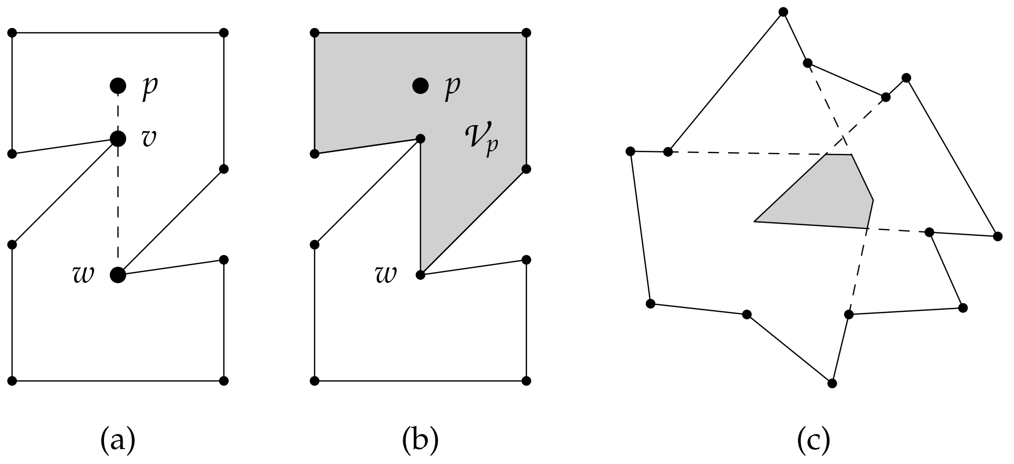


FIGURE 2.3: (a) The two polygon vertices v and w see each other, but the line of sight between p and w is blocked by v. (b) However, w is a vertex of the visibility polygon $\mathcal{V}_p$. (c) A star-shaped polygon and its kernel, drawn in gray

For every point $p \in P$ inside a polygon P it can be shown that the set of all points that are visible to p has a polygonal shape and its boundary is called the *visibility polygon* $\mathcal{V}_p$ of p. See Figure 2.3 (b) for an example and observe that the visibility polygon $\mathcal{V}_p$ may contain points on its boundary that are not visible to p.

A point $p \in P$ that sees the whole polygon P (i. e., $\mathcal{V}_p = P$) is called a *kernel point* of P and all points having this property are called the *kernel of* P, denoted by $\ker P$. If at least one kernel point exists for a polygon P (i. e., $\ker P \neq \varnothing$), we call P a *star-shaped polygon*. It can easily be shown that the kernel of a polygon itself is a convex polygon, provided that it consists not only of a single point or line segment. Moreover, the kernel of a polygon $P = e_1, \dots, e_n$ can be determined by intersecting all positive half planes that correspond to P's edges, that is

$$\ker P := \bigcap_{i=1}^{n} H^+(e_i) \,.$$

These definitions are illustrated in Figure 2.3 (c), which shows a star-shaped polygon together with its kernel polygon, drawn in gray. Note that every kernel point $p \in \ker P$ induces the same cyclic angular order of the vertices of P (with respect to p).

For a given point p and a line segment $\overset{\longmapsto}{s\,t}$ with $p \notin \overset{\longmapsto}{s\,t}$ the *view cone* of p with respect to $\overset{\longmapsto}{s\,t}$ consists of all points q such that the ray from p directed to q intersects $\overset{\longmapsto}{s\,t}$. That is, the view cone of a point p is an infinite triangular region with apex p.

2.3 Robotics

The primary objective in robotics is to design *autonomous robots*, which are capable of performing tasks such as *"Carry this container to gate 3"* without any further human interaction. For this, a robot must be able to move around in its environment, to recognize certain objects, possibly to grab them, to plan a motion to the destination using some internal knowledge of the environment, and finally, to control the execution of the task. Therefore, robotics can be viewed as the union of several component disciplines, such as mechatronics, measurement and control engineering, programming, and computer science. Although each one of these fields is essential to build a working robot, many researchers share the opinion that the term "robotics" primarily addresses the algorithmic side of the problem, cf. [Lat94].

But even if we concentrate only on the algorithmic aspects, many subproblems in different fields arise, which have to be solved. Some of the core problems are, for example, motion planning (i. e., finding a collision-free path from a source to a destination position), assembly planning (i. e., determining, whether and in which order several parts may be joined together), and sensing (i. e., using sensor inputs to gain information about the environment). The state of the art of research in these areas is presented, for example, in the workshops "Algorithmic Foundations of Robotics" [DLR00, AKM98, LO96, GH^{+}94] and "Sensor Based Intelligent Robots" [CBN99, BBN97, BKN95]. The textbook of Zhao [Zha97] gives an overview of most of the robot's tasks described above. For the wide area of robot motion planning we refer the reader, besides the literature given above, also to the survey of Schwartz and Sharir [SS90] and to the classical textbooks of Canny [Can88] and Latombe [Lat91]. The collection of Gupta and del Pobil [GdP98] deals with more practical aspects of motion planning problems.

Basic Terminology of Motion Planning

In the following we introduce some basic notions from the field of motion planning to describe configurations of robots. To this end, we presume a robot that lives in a physical world, which is modeled by an Euclidean smooth manifold (usually the $\mathbb{E}^2$ or $\mathbb{E}^3$ for real-world robots) called the *workspace* of the robot. The robot consists of one or more rigid bodies (or *links*) that are attached to each other. A configuration of the robot in its workspace (i. e., the position and orientation of the robot and of all of its links) is uniquely represented by a vector of parameters, that is, by a point in the so-called *configuration space* $\mathcal{C}$ of the robot. Consequently, the number of degrees of freedom of the robot determines the dimension of its configuration space. The subset $\mathcal{F} \subset \mathcal{C}$ of the configuration space that represents robot configurations not colliding with any obstacles in the workspace is called the *free space* of the robot.

In our localization setting we assume a robot consisting of one rigid body[2] that moves in a plane environment. That is, the configuration space of the robot is either $\mathbb{R}^2$ for cases where its orientation is fixed (or it has a compass, respectively) or $\mathbb{R}^3$ for a robot with one rotational and two translational degrees of freedom. If we furthermore suppose a point-like robot, also the robot's free space can be expressed very easily. Simply, each obstacle of the two-dimensional workspace transforms into an orthogonal prism in the three-dimensional configuration space, having the shape of the original obstacle.

But note that in contrast to this straightforward free space construction, the general motion planning problem may be much harder to solve due to a very complicated free space: Reif [Rei79] has already shown that this general problem (where the number f of degrees of freedom is part of the input) is PSPACE-hard and therefore in particular NP-hard. Furthermore, the complexity of the free space of the robot was shown to be in $\Omega(n^f)$ in the worst case [HS94a], where n describes the complexity of the workspace, that is, the total complexity of the robot and all obstacles. The consequence is that *exact* motion planning algorithms relying on the computation of the free space can be quite expensive, in particular for robots with many degrees of freedom. See the literature given above for alternative methods like approximate or probabilistic motion planning.

2.4 Graph Theory

We assume that the reader is familiar with the basic notions of graph theory (like vertices, edges, directed and undirected graphs, paths and circles in graphs, trees, etc.). Albeit, these terms are used only very sparsely throughout this thesis, such that the ideas and concepts also should be understood without a deeper knowledge of this field. As an introduction to algorithmic graph theory see, for example, the textbooks of Diestel [Die00], Noltemeier [Nol76], or Cormen et al. [CLR92, Part VI].

2 The existence of links is not relevant here, since we are only interested in the robot's position.

2.5 Models of Computation

To compare the running time and space requirements of different algorithms we have to choose an appropriate model of computation, that is, an idealized machine with a set of primitive operations [HU79, vL90a, vL90b] and their respective costs. Although the *Turing machine* [GJ79] is the classical model of computation, for the practical purposes in the field of computational geometry a different model is better suited, the *random access machine (RAM) model* [AHU74, Pap94, PS85], which we present briefly below.

A random access machine as described by Aho et al. [AHU74] primarily consists of an infinite sequence of registers, each capable of containing an *arbitrarily large* integer. The machine is able to perform the following primitive operations: input-output operations, memory-register transfers, the four basic arithmetic operations $(+, -, \cdot, /)$, indirect addressing, comparison of numbers, and branching.

Basically, there are two types of measuring the time (and likewise space) complexity of a RAM program: In the *unit-cost RAM model* each instruction can be executed in one time step, independently of the size of the operands, whereas in the *log-cost RAM model* the costs of an instruction are proportional to the encoding length of its operand, that is, to the logarithm of their size (assuming a binary representation). Note that on a unit-cost RAM integers as large as 2^{2^n} can be computed very quickly in only $O(n)$ time steps by iteratively multiplying. Consequently, a unit-cost RAM is equivalent to a *Turing machine* under a polynomial time simulation only if multiplication and division are excluded from the set of primitive operations [AHU74]. In contrast, every program on a log-cost RAM can be polynomially simulated on a Turing machine.

For most of the computational geometry algorithms the restriction on only integers is quite impractical. Therefore, we use an extension of the classical RAM model, the *real RAM* [PS85, Kle97], where each register may hold a *real number* and where in addition to the basic arithmetic operations also analytic functions may be evaluated. As long as we restrict ourselves to calculations on bounded numbers with a limited precision (i. e., rational numbers from a fixed interval), the real RAM firstly is a suitable model for a realistic computer and secondly is equivalent to the (integer) log-cost RAM described above.

But it should be noted that many computational geometry algorithms rely on calculations with real numbers. The use of finite precision arithmetic (i. e., the classical floating-point numbers with a bounded mantissa) in such algorithms probably causes overflow or round-off errors, or even inconsistent results (e. g., non-convex convex hulls, non-planar Delaunay graphs, etc.), which may lead programs to crash. For a discussion see the "CG impact task force report" of Chazelle et al. [C$^+$96, Chapt. 10] and [Cha95].

Lower Bounds

To obtain lower bound results for the time and space complexity of the problems under consideration we use the *algebraic decision tree model* [PS85], which is closely related to the RAM model in the sense that each computation executed by a RAM can be viewed as a path in the decision tree that corresponds to the RAM program. Each node in such a decision tree represents the evaluation of an algebraic function (of fixed degree) on the set of input variables, a comparison of the outcome with zero, and a branching depending on that comparison.

This way, the leaves of a decision tree represent the possible terminations (i. e., the different answers) of the corresponding RAM program, and the height of the tree gives the maximum number of operations necessary. Counting the maximum number of different answers to a problem we can find upper bounds on the number of operations and branches in the decision tree model. For example, using the *linear* decision tree model (where only functions may be evaluated that are linear in the input variables) it can easily be shown that sorting n numbers requires at least $\Omega(n \log n)$ comparisons.

Applying profound techniques from the field of algebraic complexity the results for linear decision trees can be extended to decision trees of arbitrarily large (but fixed) degree [BO83]. Therefore, also in the algebraic decision tree model sorting n numbers requires at least $\Omega(n \log n)$ operations. But note that only slightly modifying the machine model may dramatically change the time and space bounds even of "simple" problems like sorting. For example, a unit-cost (integer) RAM, which additionally is able to perform bitwise boolean operations, can sort n numbers in $O(n)$ time [Meh84a] by exploiting the unit cost assumption and the inherent parallelism of the bitwise boolean operations. Even if we use a more realistic RAM model, for which the register size of the RAM is bounded to a fixed number of bits and the operations are restricted to addition, subtraction, bitwise boolean operations, and unrestricted bit shift, n numbers can be sorted in $o(n \log n)$ time, as shown by Fredman and Willard [FW93] and Andersson et al. [AH$^+$98]. Of course, these results also have impacts on other fields like, for example, computational geometry [Wil92].

3

A Geometric Version of the Problem

"Now tell me, Mr. Hilton,
something about what has happened.
Where am I?*"*

Harold in "The Man"
B. STOKER (1846–1912)

IN THIS CHAPTER we consider an idealized version of the robot localization problem and describe the method of Guibas, Motwani, and Raghavan from the field of *computational geometry* to solving it. To this end, we make some assumptions about the robot and its environment (e. g., that all sensors are exact). On the one hand most of these assumptions can never be fulfilled in reality (e. g., the signal of every realistic sensor is noisy because of the laws of physics), but on the other hand they make it easier to analyze certain structural aspects of the problem. Moreover, if there exists an algorithm that solves an idealized version of our problem, we can hope to modify this algorithm in a way that some of its restrictions can be loosened or even abandoned.

3.1 An Idealized Scenario

For the rest of this chapter we expect the following about the robot's environment and its sensors, respectively: We assume that the robot is in an environment with flat vertical walls and a flat floor. Thus, the robot's workspace can be modeled by a two-dimensional simple polygon $\mathcal{M}$, the *map polygon*. Furthermore, we assume that the map polygon exactly represents the robot's workspace and that there are no free-standing obstacles (e. g., pillars) in the environment, that is, the map polygon has no holes.

The range sensor of the robot is expected to have an infinitely high precision, that is, each distance measurement has to be exact without any noise. Moreover, we expect the sensor to have also an infinitely high resolution (with respect to the angle increment of consecutive scan points), that means, we assume an

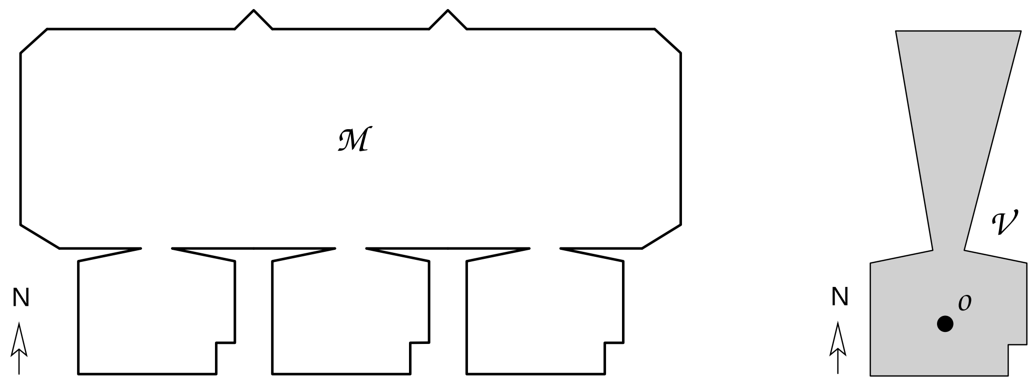


FIGURE 3.1: A polygonal map $\mathcal{M}$ and a star-shaped visibility polygon $\mathcal{V}$

infinite number of range measurements. Consequently, the robot's view can be modeled by a star-shaped *visibility polygon*, provided that the robot takes a full 360°-scan of its environment.

Furthermore, we assume that the robot has an exact compass, that means, the map polygon $\mathcal{M}$ and the visibility polygon $\mathcal{V}$ have a common reference direction (e. g., north). Therefore, the robot already knows its orientation with respect to the map. Using the terminology of Section 2.3, we have a two-dimensional configuration space representing the robot's positions and (by assuming a point-like robot) a free space that exactly corresponds to the map polygon $\mathcal{M}$.

Example 3.1 The left side of Figure 3.1 depicts a typical example of a polygonal map $\mathcal{M}$ describing a very simply structured environment, which consists of a room with two small niches in the northern direction and three larger niches in the southern direction. The right side of the figure shows a visibility polygon $\mathcal{V}$ (with the robot's viewpoint represented by a bullet »•«) of some point inside $\mathcal{M}$. Note that the robot may be located in more than one position, since the map contains some self-similarities. ◁

In the following we assume w. l. o. g. that the robot's viewpoint always lies in the origin o of its coordinate system and that the two coordinate systems of the map and of the robot, respectively, have identical reference directions (as shown, for example, in Figure 3.1). We summarize the above assumptions as follows.

Assumption 3.1 *For the robot's environment and its sensors the following holds:*

1. *The environment is exactly represented by a simple polygon $\mathcal{M}$ without holes, the map polygon.*

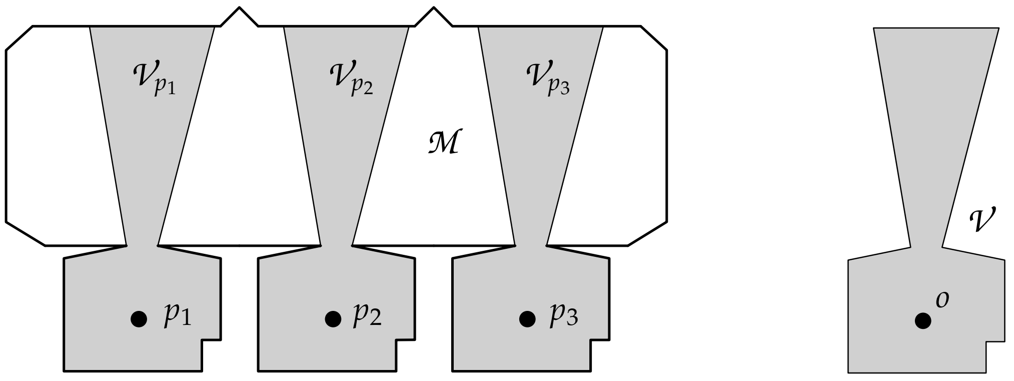


FIGURE 3.2: Three possible placements of $\mathcal{V}$ in $\mathcal{M}$

2. *The robot has a 360°-view, which is exactly represented by a star-shaped visibility polygon $\mathcal{V}$, with the robot placed in the origin o of the corresponding coordinate system.*

3. *The robot's coordinate system and the map coordinate system are oriented with respect to a common reference direction.*

3.2 The Resulting Geometric Problem

Taking advantage of the assumptions made in the previous section, the original problem of localizing a robot using a map, a compass, and a range sensor turns into a pure geometric problem, stated as follows.

Definition 3.2 (Robot Localization Problem)
Given a simple polygon $\mathcal{M}$ without holes (the map polygon) and a star-shaped polygon $\mathcal{V}$ (the robot's visibility polygon), determine all the points $p \in \mathcal{M}$ (or at least one, respectively) such that the visibility polygon $\mathcal{V}_p$ of p is translationally congruent to $\mathcal{V}$.[1]

Figure 3.2 shows as an example the three possible map positions p_1, p_2, p_3 of a robot with visibility polygon $\mathcal{V}$. Note that the visibility polygons $\mathcal{V}_{p_i}$ of all robot placements need not be disjoint like in the figure; they can also overlap and even share complete map edges. In the example of Figure 3.2 such an overlapping would occur between $\mathcal{V}_{p_1}$ and $\mathcal{V}_{p_2}$ if we moved the robot's viewpoint a little bit in the northern direction.

1 More exactly, the equality $\mathcal{V} + p = \mathcal{V}_p$ must hold. Note that the robot's position is assumed to be the origin o of the coordinate system of $\mathcal{V}$.

The main difficulties that can arise are the following:

- Except for trivial cases, every visibility polygon contains edges and vertices that have no counterparts in the map polygon. These edges and vertices are called *spurious*, since they cannot be certified to be edges or vertices of the *map*, and arise only in connection with occlusions caused by reflex vertices (cf. Definition 3.4 on page 28).

- Likewise, it may happen that only an interior portion of a map edge is visible to the robot. And in the case of collinear map edges (like in Figure 3.2), we potentially cannot even easily decide, which one of these edges is partially visible to the robot (cf. Definition 3.5).

Remark 3.2 The existence of parallel and especially collinear map edges is the major source of the problem's complexity. Note that if there are neither collinear nor parallel edges in the map, the localization problem as described above would be trivial: Since each map edge would have a unique slope, we just could pick one non-spurious edge of the visibility polygon and quickly determine the corresponding map edge. $\triangleleft$

In the following let m be the number of vertices of the robot's visibility polygon $\mathcal{V}$, n be the number of vertices of the map polygon $\mathcal{M}$, and $r < n$ be the number of reflex vertices of the map. The number of possible robot placements in $\mathcal{M}$ is denoted by A. Later on we see that this number is at most r.

Given a polygonal map and a visibility polygon, it can be shown that all solutions to the robot localization problem can be determined in time $O(mn)$ by using a suitably computed trapezoidalization of the map (depending on the spurious edges of the visibility polygon), which can be computed in time $O(mn)$, cf. [GMR97] for details. This trapezoidalization then allows us to verify each of the $O(n)$ robot placements in time $O(m)$ such that we get a total time complexity of $O(mn)$.

However, in real applications it is likely that the map does not change too often and it seems to be useful to spend some effort in preprocessing the map such that multiple subsequent localization queries can be answered quickly. Therefore, we concentrate on this kind of *multiple-shot query* and describe in the following sections the scheme of Guibas et al. [GMR97] for preprocessing the map polygon so that a localization query can be answered in time $O(m + \log n + A)$.

An obvious observation, which directly follows from Assumption 3.1 (1) or Definition 3.2, respectively, turns out to be our most powerful tool in the succeeding chapters such that we explicitly state it here:

Observation 3.3 *There cannot exist a closed path inside the map polygon $\mathcal{M}$, which encircles one or more map vertices.*

It will turn out in the sequel that many of the claims are proved by contradiction, namely by constructing a circle inside the map polygon and showing that it contains at least one vertex, which then contradicts Observation 3.3.

3.3 Solving the Geometric Problem

In the following sections we briefly recall the method of Guibas et al. [GMR97]. Their method also is the basis for our own approach described in Chapter 6, which tries to avoid (or at least relax) some of the idealizing assumptions that we made in the previous sections. The time and space complexity of the preprocessing costs (see Section 3.9) is further explored in Chapter 5. For the full details of the described scheme we refer the reader to the original work.

The definitions, lemmas, and theorems that are adopted from their work (or only slightly modified) are marked with a star ⋙⋆⋘ in the sequel. But note that the structure of our work differs at several points from the original one, since we have omitted some of the less important details (e. g., the *vertex chain decomposition*) such that there is no one-to-one correspondence of definitions and claims. Instead we included some illustrative examples and comments together with some definitions that we need in the subsequent chapters (e. g., the explicit definition of *candidate edges* or the notion of the *kernel of a skeleton*), where we investigate some properties and structural aspects of the localization problem and its solution in detail.

The described approach mainly bases on partitioning the map into finitely many regions (so-called *visibility cells*) such that within each region the visibility polygon of any point is roughly the same. This rough view is called a *(visibility) skeleton*; an intuitive definition of the skeleton of $\mathcal{V}$ is that it is a contraction of $\mathcal{V}$ that exactly contains those vertices of $\mathcal{V}$ on its boundary that can be certified to be map vertices.

The main idea is then to provide a data structure that quickly reports all regions that have the same skeleton as the query polygon $\mathcal{V}$. Then, it suffices to check only these candidate regions whether they contain points that have *exactly* the same view as $\mathcal{V}$.

3.4 Basic Definitions

For the following considerations, we assume w. l. o. g. that the map polygon $\mathcal{M}$ contains at least one reflex vertex (i. e., $r \geq 1$), thus excluding the trivial case where both polygons, the map and the visibility polygon, are convex and therefore identical. Furthermore, no two consecutive map edges are allowed to be collinear. This can easily be accomplished by substituting consecutive collinear map edges by one single edge in a preprocessing step.

Let $\mathcal{V}_p$ be a visibility polygon of a viewpoint $p \in \mathcal{M}$, which w. l. o. g. does not lie on the boundary of $\mathcal{M}$. In general, when $p \notin \ker \mathcal{M}$, it is not possible to see the complete map polygon from p and thus $\mathcal{V}_p$ contains vertices and edges, which do not coincide with vertices and edges in $\mathcal{M}$. For this reason, we introduce the notion of spurious edges and vertices that may be caused by obstructions due to reflex vertices of the map.

Definition 3.4* *An edge of a visibility polygon $\mathcal{V}_p$ is* ***spurious*** *if it is collinear with p. A vertex v of $\mathcal{V}_p$ is called* ***spurious*** *if it lies on a spurious edge (u, v) of $\mathcal{V}_p$ such that u is closer to p than v is.*

These notions are illustrated in Figure 3.3 on the facing page: Both visibility polygons $\mathcal{V}_p$ and $\mathcal{V}_q$ have some spurious edges (drawn as dotted lines), each of which consists of a spurious vertex and a non-spurious reflex vertex; the non-spurious vertices are represented by bullets »•« in the figure. Informally, the non-spurious edges and vertices of a visibility polygon $\mathcal{V}_p$ are those ones that can be certified by a robot standing at point p to be map edges and vertices.

Note that there may exist spurious edges that actually coincide with map edges (e. g., the leftmost spurious edge of $\mathcal{V}_p$ in Figure 3.3). But this situation can only occur if such a map edge (u, v) is collinear with the viewpoint p. Then, that one of the two endpoints closer to p, say u, is visible from p, but blocks the view to the other endpoint v. Hence, standing in p we cannot decide whether (u, v) is actually an existing map edge or not.

For spurious edges and vertices the following can easily be shown.

Lemma 3.1* *Let $\mathcal{V}_p$ be a visibility polygon of a point $p \in \mathcal{M}$. Then, the following holds:*

1. *A reflex vertex of $\mathcal{V}_p$ can never be spurious.*

2. *No two spurious edges can be adjacent in $\mathcal{V}_p$.*

3. *Let e be a non-spurious edge of $\mathcal{V}_p$ and e' the supporting edge of $\mathcal{M}$ on which e lies. Then, e is the only portion of e' visible from p.*

4. *An edge e (or a vertex v, respectively) of $\mathcal{V}_p$ is non-spurious if and only if for each choice of a map polygon $\mathcal{M}$ and a viewpoint $q \in \mathcal{M}$ such that $\mathcal{V}_p = \mathcal{V}_q$, the edge e (or the vertex v, respectively) lies on an edge (or on a vertex, respectively) of $\mathcal{M}$.*

The non-spurious edges of a visibility polygon can be classified into three groups:

Definition 3.5⋆ *Let e be a non-spurious edge of a visibility polygon. Then, e is of exactly one of the following types:*

- ***full edge**: both endpoints of e are non-spurious;*
- ***half edge**: one endpoint of e is spurious, the other is non-spurious;*
- ***partial edge**: both endpoints of e are spurious vertices.*

Figure 3.4 on the next page demonstrates the classification of the edges of a visibility polygon into spurious edges and into the three groups of non-spurious edges of the previous definition. Note that the different edge types can occur only in certain patterns. Particularly, every spurious vertex of a half edge must be incident with a corresponding spurious edge. We use this observation to introduce the notion of *blocking vertices*, which are responsible for the occlusions in a visibility polygon (see Figure 3.4).

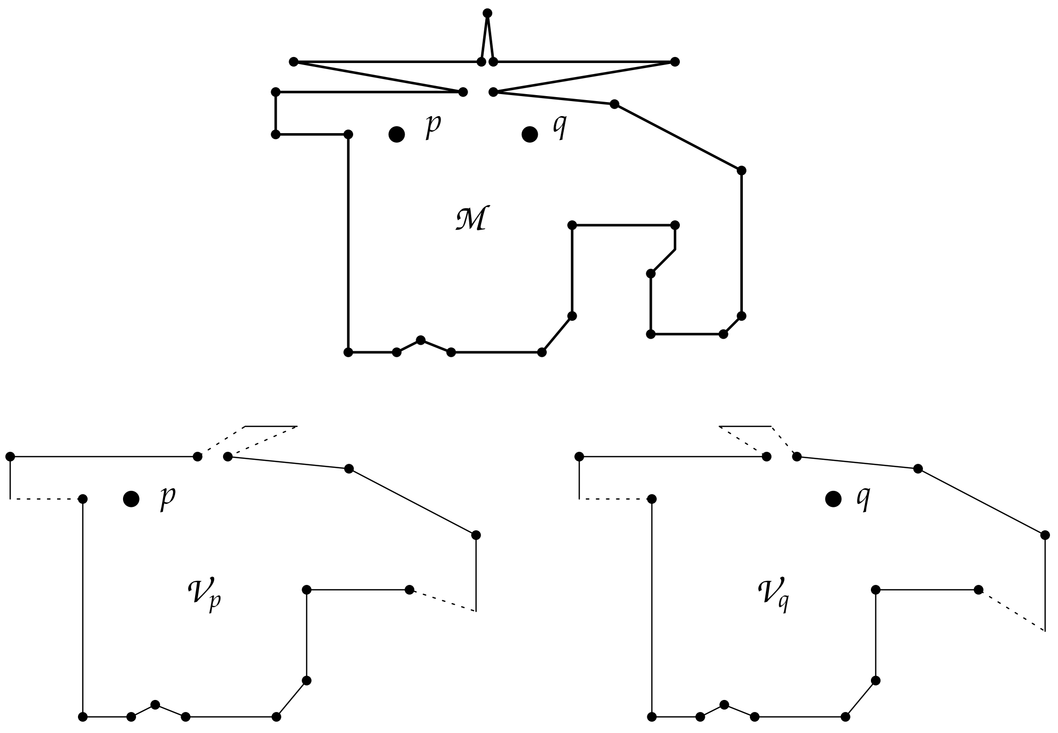


FIGURE 3.3: Two visibility polygons with almost identical shapes

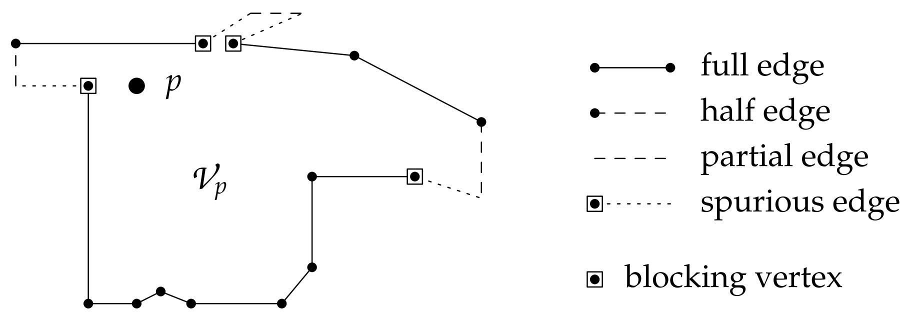


FIGURE 3.4: Classification of the visibility polygon edges and vertices

Definition 3.6* *Let (v, w) be a half edge with a spurious vertex v and let (u, v) be the corresponding spurious edge that is adjacent to (v, w). Then, u is called the **corresponding blocking vertex** to the half edge (v, w).*
Analogously, a partial edge has two corresponding blocking vertices.

Note that a blocking vertex must always be a reflex vertex, that is, the total number of blocking vertices is in $O(r)$.

3.5 The Visibility Skeleton

Now we have all prerequisites at hand to define the visibility skeleton as a coarsened representation of a visibility polygon. On the one hand such a skeleton gives only a rough view onto a visibility polygon, but on the other hand it turns out to be fine enough to fully reconstruct the visibility polygon using only the viewpoint as an additional information.

Definition 3.7* (Visibility Skeleton)
*For a visibility polygon $\mathcal{V}_p$ its corresponding **visibility skeleton V_p^*** is a labeled polygon, defined as follows:*

1. *the vertices of V_p^* are exactly the non-spurious vertices of $\mathcal{V}_p$;*
2. *each full edge of $\mathcal{V}_p$ becomes an edge of V_p^*;*
3. *each half edge e_i of $\mathcal{V}_p$ is replaced by a labeled **artificial edge** a_i between the non-spurious vertex of the half edge and the corresponding blocking vertex;*
4. *each partial edge e_i of $\mathcal{V}_p$ is replaced by a labeled artificial edge a_i between the two corresponding blocking vertices;*

5. *the label of an artificial edge a_i consists of a characterization of the line l_i on which the original half or partial edge e_i lies (i. e., the coefficients of the linear equation that defines l_i);*

6. *all coordinates of V_p^* as well as the coefficients of the artificial edge labels l_i are given with respect to a coordinate system with a canonically chosen reflex vertex v_0 of $\mathcal{V}_p$ as origin (e. g., the leftmost reflex vertex of $\mathcal{V}_p$).*

The polygonal part of a skeleton V_p^*, that is, the skeleton without all edge labels, is called the ***skeleton polygon $V_p^\diamond$*** *of the skeleton V_p^*.*

Figure 3.5 illustrates this definition: It shows the common skeleton of the two visibility polygons of Figure 3.3. Note that the skeletons V_p^* and V_q^* are *identical*, although the *partial edges* in the two polygons not only lie on *different parts* of the same map edge, but even on *completely different* map edges. This property is a consequence of the definition of the artificial edge labels in Definition 3.7 (5). Further note that a visibility skeleton is invariant under translations of the underlying visibility polygon due to Definition 3.7 (6). Consequently, the three visibility polygons $\mathcal{V}_{p_1}$, $\mathcal{V}_{p_2}$, $\mathcal{V}_{p_3}$ of Figure 3.2 on page 25 all have identical skeletons.

In the following we only differentiate between half edges and partial edges where it is necessary, since both result in an artificial edge and can be distinguished through their labels: A half edge can be seen as a special kind of partial edge with a corresponding line that goes through the non-spurious vertex of the half edge.

Although the skeleton is only a coarse representation of a visibility polygon, the following theorem shows that it has enough details to fully reconstruct the visibility polygon if we also know the viewpoint.

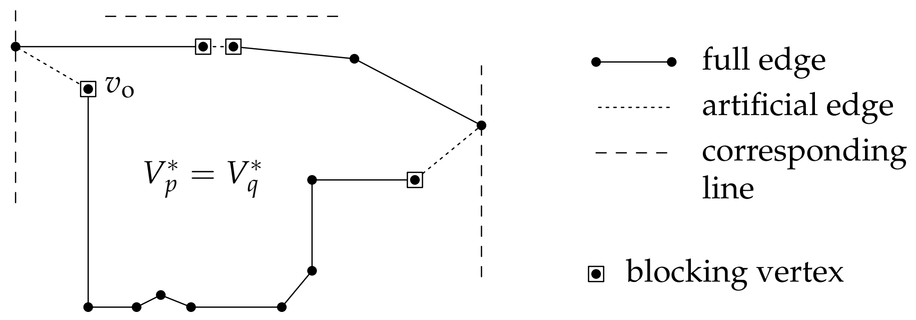


FIGURE 3.5: The common visibility skeleton of $\mathcal{V}_p$ and $\mathcal{V}_q$ of Figure 3.3

Theorem 3.2* *Let $\mathcal{V}_p$ and $\mathcal{V}_q$ be two visibility polygons of points $p, q \in \mathcal{M}$ such that p has the same position relative to the coordinate origin v_o of V_p^* as does q to the origin v_o of V_q^*. Then, $\mathcal{V}_p = \mathcal{V}_q$ if and only if $V_p^* = V_q^*$.*

Proof. Clearly, if p and q have the same visibility polygon, then their skeletons are also identical. For the remaining proof, we assume that p and q have the same skeleton and consequently show that their visibility polygons are identical. Since the non-spurious vertices remain untouched at the transition from $\mathcal{V}$ to V^*, the non-spurious vertices as well as the full edges are identically located in the two visibility polygons. Hence, the only difference between $\mathcal{V}_p$ and $\mathcal{V}_q$ could be in different positions of the spurious vertices and edges.

Recall that any spurious vertex is an endpoint of either a half or a partial edge. Consider now any fixed artificial edge a in the common skeleton of p and q. Then, in the visibility polygon there are two spurious edges (or only one, respectively, if we have the special case of a half edge) that are adjacent to the corresponding partial edge (cf. Figure 3.4 on page 30). Each spurious edge starts at the same blocking vertex of a in both visibility polygons, is collinear with the viewpoint, and ends at the line l that is defined by the label of a. Since the relative positions of the viewpoints p and q as well as the lines l_i in the two skeletons are identical, the location of the spurious vertex must also be identical in both cases. This completes the proof. □

Since we always know the robot's position with respect to its visibility polygon (cf. Assumption 3.1 (2) on page 25), Theorem 3.2 shows that for localizing a given visibility polygon $\mathcal{V}_p$ of a robot it suffices to consider only the corresponding skeleton V_p^*. Later on we see that for any map only a finite number of different skeletons exist (cf. Theorem 3.3 on page 34). This means, the continuous problem[2] of fitting a visibility polygon into the map is discretized and solved in a natural way by using the skeleton instead of the visibility polygon.

3.6 The Visibility Cell Decomposition

As we have seen in the examples before, several points in the map polygon (with different visibility polygons) may result in identical skeletons. In the following we describe a subdivision of the map polygon into convex visibility cells such that in each cell all points have identical visibility skeletons. This subdivision is constructed by introducing lines into the interior of $\mathcal{M}$. Each line partitions $\mathcal{M}$ into two regions, one where some vertex v is not visible due

2 "Continuous" in the sense that for any point $p \in \mathcal{M}$ we cannot find an $\varepsilon > 0$ such that the visibility polygon $\mathcal{V}_p$ of a point p moving by at most ε does not change.

to the obstruction created by some reflex vertex v_r and another region where the view of v is not blocked by v_r.

Definition 3.8* (Visibility Cell Decomposition)
*Let v be a vertex and let v_r be a reflex vertex of $\mathcal{M}$ such that v is either visible from v_r or adjacent to it in $\mathcal{M}$. The **visibility ray emanating** from v and **anchored** at v_r is the (directed) segment inside $\mathcal{M}$ that lies on the line through v and v_r, starts at v_r, ends at the boundary of $\mathcal{M}$, and is oriented as to move away from v. This ray is called a **left** or **right** ray according to whether the map edges incident to the anchor v_r are to the left or to the right of the ray.*

*The **visibility cell decomposition** of a map polygon $\mathcal{M}$ is the subdivision that is generated by introducing all possible visibility rays into $\mathcal{M}$.*

Figure 3.6 shows the visibility cell decomposition of the map of Figure 3.3 together with an example of a left ray emanating from the vertex v and anchored at the reflex vertex v_r.

Since in the sequel we want all points in a visibility cell to have identical skeletons, we have to take care of correctly assigning the points on the interior cell boundaries (i. e., on the visibility rays) to the incident cells. To this end, consider a ray emanating from a vertex v and anchored at a reflex vertex v_r. As already stated above, the cells incident to this ray are divided into two classes, those which can see v and those which cannot. The visibility polygon of some viewpoint on this ray contains v as a spurious vertex. Consequently, standing on the visibility ray the vertex v cannot be certified to be a map vertex. Therefore, we should assign the points on the visibility ray to those cells from which the vertex v also cannot be certified to be a map vertex, that is, those cells that cannot see v at all. Thus, we assign the points on a visibility ray to the cells on that side of the ray where the obstruction determined by the anchor vertex

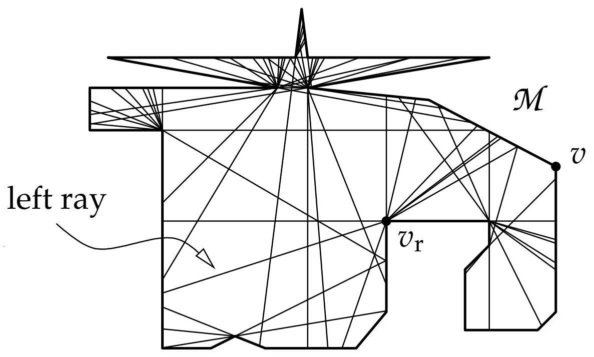


Figure 3.6: The visibility cell decomposition of the map polygon $\mathcal{M}$

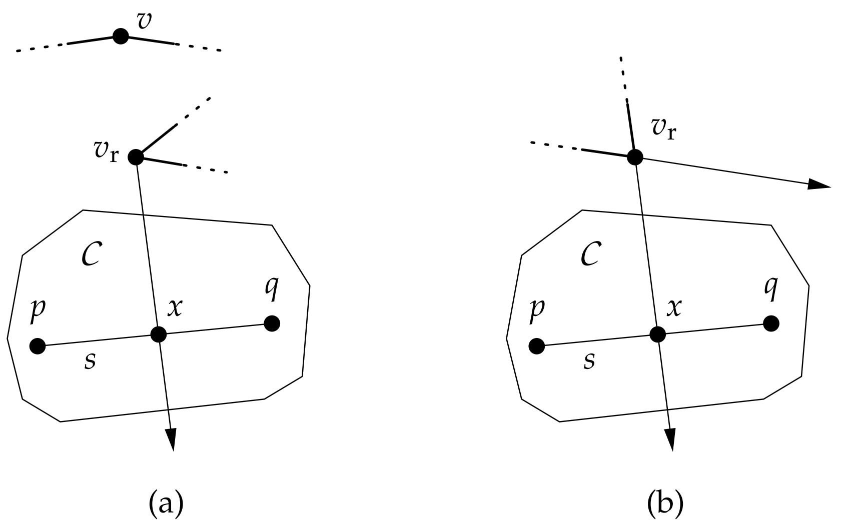


FIGURE 3.7: (a) When walking from p to q, vertex v is obstructed by reflex vertex v_r for the first time at point x. (b) The reflex vertex v_r is a blocking vertex for p and a non-blocking vertex for q

lies. Using this rule, each point of the map polygon gets assigned to a unique visibility cell. Now we are able to show the connection between the visibility cell decomposition and the visibility skeleton.

Theorem 3.3* *Let $\mathcal{C}$ be any cell in the visibility cell decomposition of $\mathcal{M}$.*

1. *The cell $\mathcal{C}$ is a convex polygonal region inside $\mathcal{M}$.*
2. *For any two points $p, q \in \mathcal{C}$, $V_p^* = V_q^*$.*

Proof. For the first part, let $\mathcal{C}$ be a non-convex visibility cell and let v be a reflex vertex from the boundary of $\mathcal{C}$. Then it is clear that v cannot be a vertex from the boundary of $\mathcal{M}$. Otherwise, the edges incident to v would determine two visibility rays that subdivide $\mathcal{C}$. On the other hand, all interior vertices of the visibility cell decomposition are created by the intersection of two visibility rays, which start and end at the map boundary, and therefore cannot be reflex. Thus, v cannot be an interior vertex and this leads to a contradiction.

For the second part, assume that $V_p^* \neq V_q^*$. Consider the straight line segment $s = \overline{pq}$ and recall that s is totally contained in $\mathcal{C}$ (even if p and q are boundary points of $\mathcal{C}$) and no interior line intersects s, because $\mathcal{C}$ is convex.

We consider first the case where V_p^* and V_q^* do not have the same underlying polygon, that is, $V_p^\diamond \neq V_q^\diamond$. Without loss of generality we assume that the

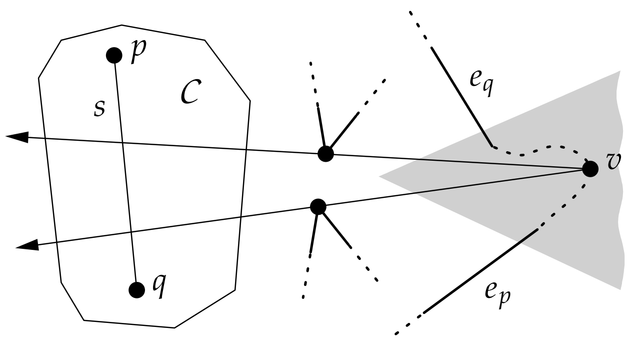


FIGURE 3.8: At least one map vertex v lies in the shaded area and induces two additional visibility rays

difference between the two skeleton polygons is that a vertex v of $\mathcal{M}$ is visible from p but is not visible from q. Let now $x \in l$ be the point at which the vertex v is obstructed by some reflex vertex v_r for the first time, when we walk from p to q, see Figure 3.7 (a) on the facing page. Then the visibility ray emanating from v and anchored at v_r intersects s at point x, giving a contradiction.

Any difference in the two skeletons must then be either in the type of the edges (i. e., full edge or artificial edge) or in the labels of corresponding artificial edges in the two skeletons. It is easy to see that the type of the edges must be identical in both skeletons. Otherwise, there must exist a reflex vertex v_r that serves as a blocking vertex in one skeleton, say V_p^*, and as a non-blocking vertex in the other skeleton V_q^*. But then, p and q lie on different sides of one of the two visibility rays that are induced by the edges incident to v_r, see Figure 3.7 (b). In particular, this ray intersects s in a point x, which again leads to a contradiction.

The last possibility for a difference between V_p^* and V_q^* is then the labeling of the artificial edges, that is, the line characterizations of the corresponding half and partial edges of the original visibility polygons $\mathcal{V}_p$ and $\mathcal{V}_q$. In that case, there must exist two map edges e_p and e_q such that p can see some portion of e_p but cannot see e_q at all, and q can see some portion of e_q but cannot see e_p at all (see Figure 3.8). But then, there must exist at least one map vertex v "between" e_p and e_q (in the shaded area in Figure 3.8), which is neither visible from p nor from q. Thus, the vertex v induces two visibility rays that intersect s in its interior, again giving a contradiction. □

The preceding theorem enables us to define the visibility skeleton of a cell as follows.

Definition 3.9* *For any visibility cell $\mathcal{C}$ its* ***visibility skeleton*** $\boldsymbol{V^*_{\mathcal{C}}}$ *is the common visibility skeleton of all points in $\mathcal{C}$.*

In particular, it follows from Theorem 3.3 and from Definition 3.8 that for any polygonal map only a finite number of different visibility skeletons exist. But note that Theorem 3.3 (2) does not imply that points with identical skeletons lie in the same visibility cell. This need not be true, since the map polygon may contain identical parts at different positions (cf. Example 3.1 on page 24).

Using the fact that Definition 3.9 associates each visibility cell with exactly one visibility skeleton, an equivalence relation over the cells is a straightforward result.

Definition 3.10* (Equivalence Class of a Skeleton)
Two visibility cells $\mathcal{C}_1$ and $\mathcal{C}_2$ are said to be ***equivalent****, $\mathcal{C}_1 \equiv \mathcal{C}_2$, if and only if $V^*_{\mathcal{C}_1} = V^*_{\mathcal{C}_2}$. The* ***equivalence class*** *of all visibility cells with skeleton V^* is denoted by $\boldsymbol{\mathcal{EC}_{V^*}}$.*

The ***complexity*** $\boldsymbol{|\mathcal{EC}_{V^*}|}$ *of an equivalence class $\mathcal{EC}_{V^*}$ is defined as the total number of vertices and edges of all visibility cells with skeleton V^*.*

The maximum worst-case complexity of an equivalence class $\mathcal{EC}_{V^*}$ plays a crucial role in estimating the preprocessing costs of a localization query, see Section 3.9 and Equation (3.2) on page 49.

3.7 Embeddings of Skeletons

Just as a star-shaped polygon $\mathcal{V}$ can fit into the map at several positions, the same holds for its skeleton V^*. Two examples are depicted in Figure 3.9 on the facing page: The skeleton V_1^* fits at exactly one position into the map polygon $\mathcal{M}_1$, whereas skeleton V_2^* has three different positions, where it fits into $\mathcal{M}_2$. But note that even in the relatively simple first case, there are two separated visibility cells with skeleton V_1^* in $\mathcal{M}_1$. The concept of fitting a skeleton into the map is now formalized in the following definition introducing the notion of an *embedding* of a skeleton.

Definition 3.11* (Embedding of a Skeleton)
An ***embedding of a visibility skeleton*** *V^* (with origin v_o) is an injective function h from the vertices of V^* into the vertices of the map polygon $\mathcal{M}$ such that*

1. *for each vertex v of V^* the location of $h(v)$ relative to $h(v_o)$ in the map is identical to the location of v relative to v_o in the skeleton;*

2. *there is a full edge between the vertices v and w in V^* if and only if there exists an edge in $\mathcal{M}$ with endpoints $h(v)$ and $h(w)$;*

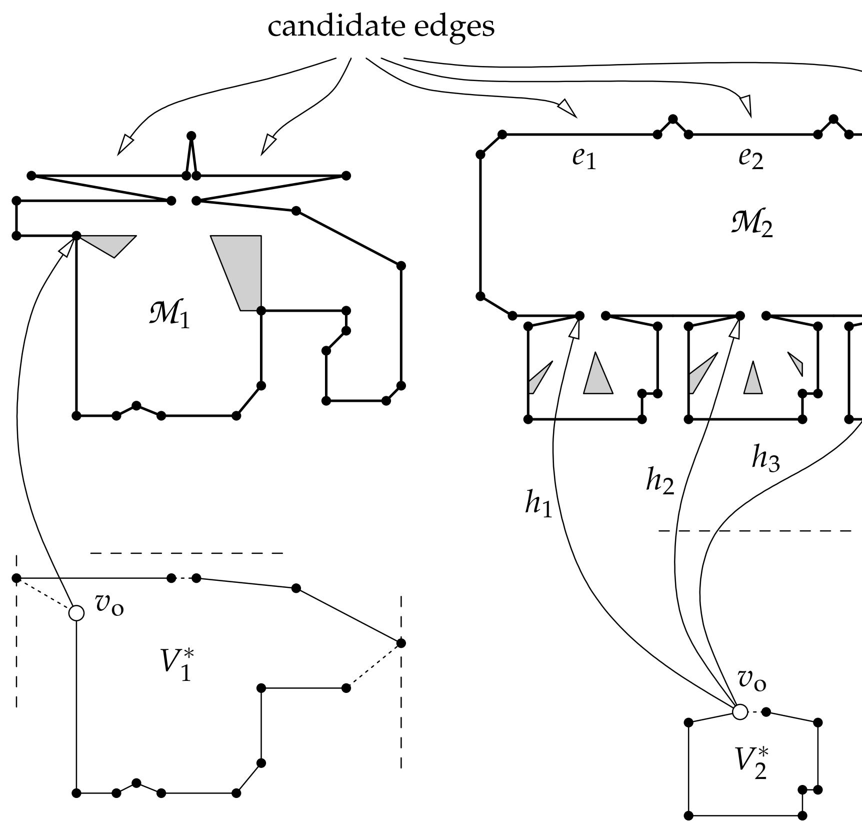


FIGURE 3.9: Embeddings of V_1^* and V_2^*; the visibility cells with skeleton V_1^* or V_2^*, respectively, are drawn in gray

3. *let a be an artificial edge between vertices v and w of V^* and let l be the line labeling a. Then there is an edge e of $\mathcal{M}$ lying on a line l' whose equation (with $h(v_o)$ as origin) is that of l (with v_o as origin), and a point of e is visible from $h(v)$ and $h(w)$.*

*Such an edge e from above is called a **candidate edge** (of the artificial edge a in embedding h of V^*).*

For the sake of a more intuitive notation, let for *any point p* in the coordinate system of V^* (not only for the vertices of V^*) the point $h(p)$ be the corresponding embedded point in the coordinate system of $\mathcal{M}$. We extend this notation also to point sets (like lines and segments) and denote, for example, the embedded line l' in the previous definition by $h(l)$. Furthermore, we use »$h(V^*)$« as a shortcut for the term »embedding h of the skeleton V^*«. The following example illustrates the concept of an embedding of a skeleton.

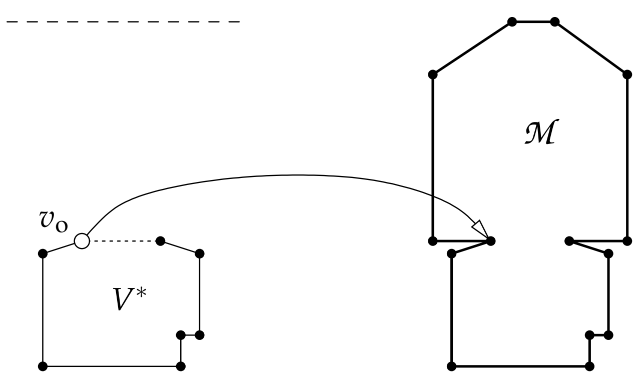


FIGURE 3.10: An embedding of V^*, where no point exists with skeleton V^*

Example 3.3 Look at the embedding of V_1^* in the left part of Figure 3.9 on the page before. Two of the three artificial edges of V_1^*, namely the left one and the right one, have only one corresponding candidate edge in the embedding, but for the middle one of the artificial edges there exist two individual candidate edges on the embedded line, separated by a small peak. Viewing from the left one of the two visibility cells, we only see points of the right candidate edge, and vice versa. Likewise, the three visibility cells in embedding $h_2(V_2^*)$ in the right example correspond to the three candidate edges e_1, e_2, e_3 of the embedded artificial edge of V_2^*. Also notice that a single map edge may serve as a candidate edge in more than one embedding of a skeleton. For example, the edge e_2 is a candidate edge in all embeddings h_1, h_2, h_3. The relation between the (number of) candidate edges and the (number of) visibility cells is described in detail in Chapter 5. $\triangleleft$

Remark 3.4 Note that even if an embedding exists for a given skeleton V^*, there need not exist any point $p \in \mathcal{M}$ with that skeleton V^*. Figure 3.10 shows an example: Apparently, an embedding of V^* into the map $\mathcal{M}$ exists, since all requirements of Definition 3.11 are satisfied. But the candidate edge of the artificial edge of V^* has a smaller length than the artificial edge itself; thus, every point in the map that sees some part of the candidate edge also sees its endpoints and therefore cannot have V^* as skeleton. Consequently, there exists no point in $\mathcal{M}$ with skeleton V^*. $\triangleleft$

From the previous comment follows that the existence of an embedding for a given skeleton is no *sufficient* condition for the existence of a point with that skeleton and in the subsequent chapter we introduce the notion of *valid embeddings* to circumvent such cases. But note that using the following definition of the *kernel of a skeleton*, we can easily show that for any point $p \in \mathcal{M}$ a *necessary*

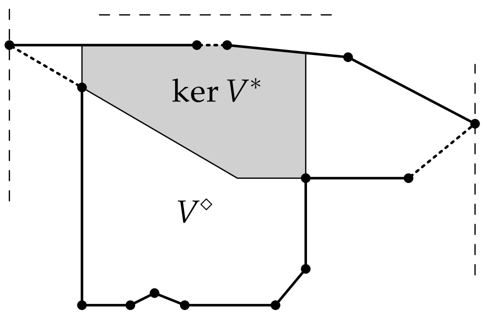


Figure 3.11: The kernel of a skeleton V^*

condition for having a given skeleton V^* is that p lies in an embedding of the *kernel* of V^*.

Definition 3.12 (Kernel of a Skeleton)
For a given skeleton V^* its **kernel ker V^*** *is defined as the kernel of the corresponding skeleton polygon $V^\diamond$.*

Figure 3.11 shows an example of the kernel of a skeleton V^*. The succeeding lemma proves the simple connection between "having a certain skeleton" and "lying in an embedding of that skeleton".

Lemma 3.4 *For any point $p \in \mathcal{M}$ with skeleton V_p^* there exists an embedding $h(V_p^*)$ such that $p \in h(\ker V_p^*)$, that is, p lies in the (embedded) kernel of the skeleton.*

Proof. We first specify an embedding $h(V_p^*)$ and then show that the point p lies in $h(\ker V_p^*)$. To this end, let v_o' be the canonically chosen *map* vertex that corresponds to the origin v_o of V_p^* (see Definition 3.7 (6) on page 31). To define h we map v_o to v_o' and the remaining vertices of V_p^* to the corresponding vertices of $\mathcal{V}_p$. Then, the three requirements of Definition 3.11 are trivially fulfilled.

It remains to show that the point p lies in the kernel $h(\ker V_p^*)$ of the so-defined embedding. But this is clear, because all points of $\mathcal{V}_p$ are visible from p and each artificial edge of V_p^* also lies in $\mathcal{V}_p$. □

In the following we use the slightly incorrect term "kernel of an embedding (of a skeleton)"[3] as an easier-to-understand synonym for the correct term "embedded kernel of a skeleton". From the preceding lemma it particularly follows

3 Incorrect, because we have defined the kernel only for skeletons and *not* for embeddings of skeletons.

that each visibility cell with skeleton V^* lies in the kernel of an embedding of V^*. See for example the right part of Figure 3.9 on page 37, which shows the seven visibility cells of $\mathcal{EC}_{V_2^*}$.

Furthermore, it directly follows from the succeeding lemma that a visibility cell with skeleton V^* always lies in (the kernel of) exactly one embedding of V^*.

Lemma 3.5 *For any point $p \in \mathcal{M}$ with skeleton V_p^* there exists exactly one embedding $h(V_p^*)$ such that $p \in h(\ker V_p^*)$.*

Proof. Due to Lemma 3.4 we only have to show that there cannot be two different embeddings h and h' with $p \in h(\ker V_p^*) \cap h'(\ker V_p^*)$. Assume that this would be the case and recall that an embedding of V^* is totally determined by the position of its origin vertex v_o. From $h \neq h'$ it follows that $h(v_o) \neq h'(v_o)$, which directly leads to a contradiction, because both vertices must be visible to p, but at most one of them could be identical with the canonically chosen reflex vertex v_o of $\mathcal{V}_p$, cf. Definition 3.7 on page 30. □

Note that from the preceding lemma it does not follow that two kernels of different embeddings $h(V^*)$ and $h'(V^*)$ of the same skeleton never intersect (we will discover this situation in detail in Example 4.1 on page 54). The lemma only states that no point with skeleton V^* lies in such an intersection.

3.8 Localizing a Visibility Polygon

In this section we briefly describe how the concepts introduced in the preceding sections can be used to efficiently localize a given visibility polygon $\mathcal{V}$, that is, to determine all points $p \in \mathcal{M}$ such that $\mathcal{V} + p = \mathcal{V}_p$. As already stated, we assume a multiple-shot query such that it is profitable to spend some effort in preprocessing. The basic strategy is as follows.

1. Identify the spurious vertices and edges of the visibility polygon $\mathcal{V}$ using the robot's viewpoint (cf. Assumption 3.1 on page 24) and compute the corresponding visibility skeleton V^*.

2. Determine the equivalence class $\mathcal{EC}_{V^*}$, that is, find all visibility cells with skeleton V^* .

3. Test for each visibility cell if it contains a point p with $\mathcal{V} + p = \mathcal{V}_p$.

In order to perform the Steps 2 and 3 efficiently, we make the following observation, which directly follows from Lemma 3.5:

Observation 3.13 *For any skeleton V^* each visibility cell $\mathcal{C}$ with $V^*_{\mathcal{C}} = V^*$ lies in exactly one embedded kernel of V^*.*

Furthermore, using Theorem 3.2 we notice:

Observation 3.14 *For each embedding h of V^* there is at most one point $p \in h(\ker V^*)$ with visibility polygon $\mathcal{V}_p = \mathcal{V}$.*

This is due to the fact that all points in a fixed embedding have different positions relative to V^* and therefore also different visibility polygons. The consequence is, that we need not inspect each visibility cell individually, but only have to check for each embedding h of V^* whether the embedded viewpoint $h(o)$ of the robot lies in a visibility cell with skeleton V^*. We state this observation as a corollary to Theorem 3.2.

Corollary 3.6* *Let $\mathcal{V}_p$ be a visibility polygon of a point $p \in \mathcal{M}$ and let h be an embedding of its corresponding skeleton V^*_p. Then, $\mathcal{V}_{h(p)} = \mathcal{V}_p$ if and only if $V^*_{h(p)} = V^*_p$.*

It remains to specify how to quickly identify for each embedding the visibility cell that contains the robot's viewpoint. To this end, we perform the following preprocessing step for every equivalence class of visibility cells (i. e., for every occurring skeleton): For each skeleton V^* we determine all embeddings of V^*, overlay the corresponding visibility cells, and compute the arrangement of all of these cells. Each region of the arrangement gets a label, which indicates which of the original visibility cells include this region.

This way, we can determine all visibility cells that contain the robot's viewpoint by a single point location query into the above precomputed arrangement. Then, the actual potential robot positions can be computed from these cells by using the corresponding embeddings. To clarify this localization strategy we perform a localization query for the map introduced in Example 3.1 on page 24 in the succeeding example. The entire strategy is then presented in Algorithm 3.13 on page 44. The respective preprocessing steps in order to compute for each existing skeleton the point location structure are shown in Algorithm 3.14.

Example 3.5 We assume that a robot operates in the map $\mathcal{M}$ shown in Figure 3.12 (d) on page 43 and sees the polygon $\mathcal{V}$ depicted in Figure 3.12 (a). Two edges of $\mathcal{V}$ cannot be certified to be map edges, since they are collinear with the robot's viewpoint o and therefore spurious edges (Step 1 of Algorithm 3.13). To obtain the corresponding skeleton V^* (Step 2) we replace the partial edge of $\mathcal{V}$ by an artificial edge, which gets a label that represents the original partial edge, see Figure 3.12 (b). Note that all coordinates of V^* are given relative to the origin v_o of the skeleton (e. g., the leftmost reflex vertex of $\mathcal{V}$).

In Step 3 of Algorithm 3.13 the skeleton V^* gets encoded into a vector $[V^*]$ of d real numbers in a canonically way (including the label of its artificial edge) and in Step 4 the leaf of the precomputed d-dimensional search tree T_d that contains $[V^*]$ is retrieved. Now we have determined the equivalence class $\mathcal{EC}_{V^*}$.

In order to quickly determine all visibility cells of $\mathcal{EC}_{V^*}$ that contain the robot's viewpoint o, we use the precomputed point location structure PL_{V^*} that is linked to the leaf. Figure 3.12 (c) illustrates this structure by showing a magnified version of the overlay arrangement of the visibility cells $\mathcal{C} \in \mathcal{EC}_{V^*}$. Compare this arrangement to the three embeddings of V^* and the seven visibility cells shown in Figure 3.9 on page 37. Note that the numbers, with which the regions of the arrangement are labeled, stand for the embeddings corresponding to those cells that contain the respective region. Therefore, in order to report the possible robot placements, we perform a point location query in PL_{V^*} (in Step 5), which yields a region with label $\gg 1, 2 \ll$.

Therefore, the only embeddings of V^* that contain a potential robot position are $h_1(V^*)$ and $h_2(V^*)$, see Figure 3.12 (d). In the remaining steps of Algorithm 3.13 the actual coordinates of these positions are computed by applying the corresponding embeddings (i. e., translations) to the robot's viewpoint o (relative to the skeleton origin v_o). $\lhd$

3.9 Time and Space Complexity

In this section we briefly investigate the time and space requirements of Algorithm 3.13 and 3.14. To this end, we first have to study the structures defined in the previous sections and give some upper and lower bounds on the complexity of visibility cells, the number of embeddings and cells, etc. For more extensive proofs of some of the following bounds we refer the reader to the original work of Guibas et al. [GMR97]. Recall for the following investigations that n denotes the total number of map vertices and r stands for the number of reflex vertices of $\mathcal{M}$.

Lemma 3.7* *A skeleton V^* has at most r embeddings in $\mathcal{M}$ and this bound is tight up to a constant factor.*

Proof. We observe that in any embedding $h(V^*)$ of the skeleton V^* the relative position of any vertex v with reference to the origin v_o of the skeleton, that is, $h(v) - h(v_o)$, is identical to the relative position $v - v_o$ in the original skeleton.

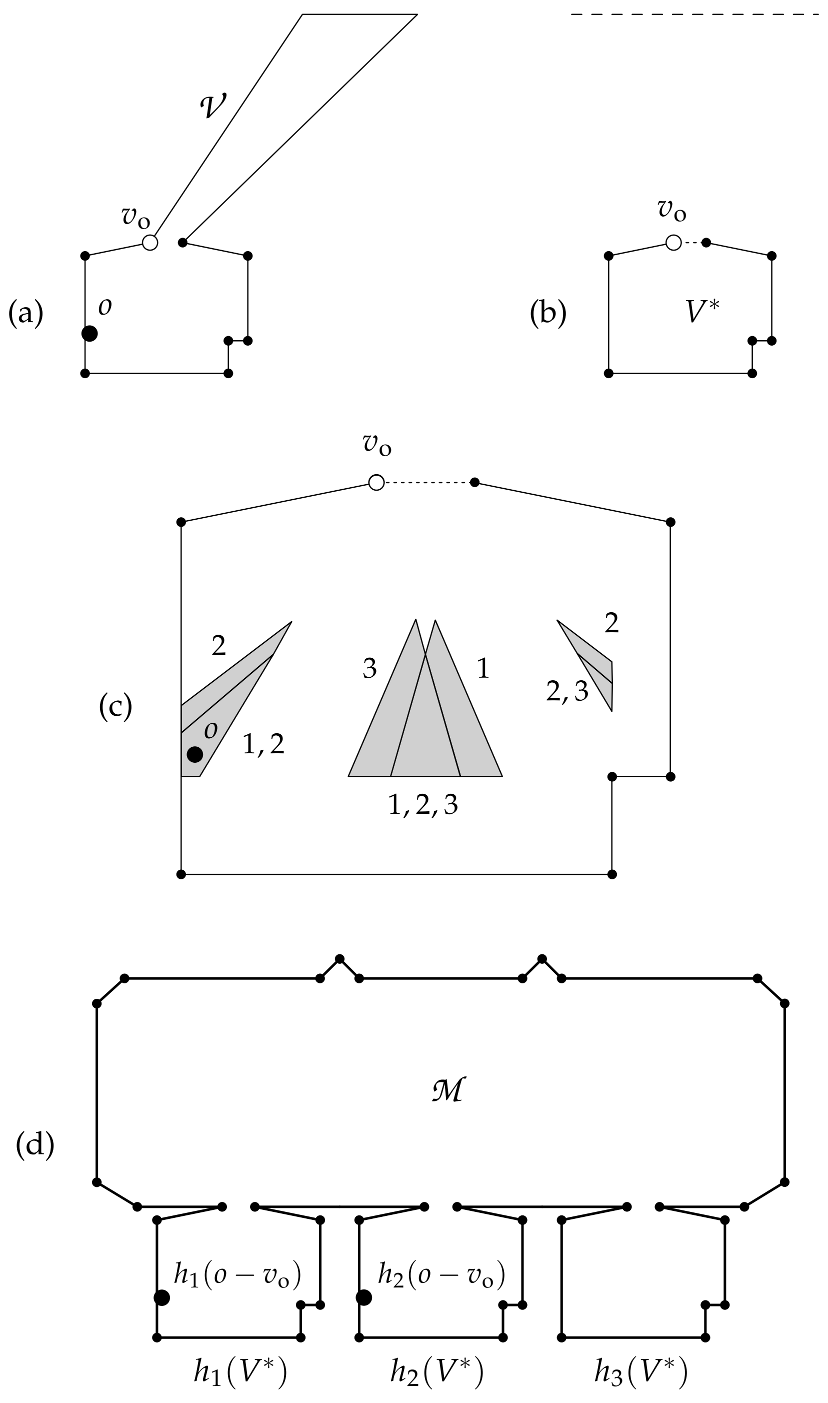


FIGURE 3.12: A localization query illustrating Example 3.5. (a) The robot's visibility polygon $\mathcal{V}$. (b) The corresponding skeleton V^*. (c) The overlay arrangement of V^*. (d) The map polygon $\mathcal{M}$

```
Input:  Visibility polygon $\mathcal{V}$ with $m$ vertices in total (and the viewpoint
        lying in the origin $o$ of its coordinate system)
        Search trees $T_d$ and point location structures $PL_{V^*}$ computed in the
        preprocessing step (Algorithm 3.14)
 1 Detect the spurious edges (collinear with $o$) and vertices of $\mathcal{V}$
 2 Compute the visibility skeleton $V^*$ (with origin $v_o$)
 3 Encode $V^*$ into a vector $[V^*]$ of dimension $d \in O(n)$
 4 Find the leaf in $T_d$ that contains $[V^*]$ (i. e., the equivalence class $\mathcal{EC}_{V^*}$) and
   the linked point location structure $PL_{V^*}$
 5 Perform a point location query in $PL_{V^*}$ at position $o - v_o$, reporting a re-
   gion $R$ in the subdivision that corresponds to $\mathcal{EC}_{V^*}$
 6 $L \leftarrow$ label of $R$                          { L is a set of visibility cells }
 7 $POS \leftarrow \varnothing$                          { The set of potential robot positions }
 8 for each visibility cell $\mathcal{C} \in L$ do
 9     Let $h$ be the embedding whose kernel $h(\ker V^*)$ contains $\mathcal{C}$
10     $POS \leftarrow POS \cup \{h(o - v_o)\}$           { One potential robot position }
11 end for
Output:  The set $POS$ of potential robot positions
```

ALGORITHM 3.13: Localization query for a visibility polygon

This means that the position $h(v_o)$ of the origin in the map uniquely determines the entire embedding h. Since v_o was defined to be a reflex vertex, there cannot be more embeddings than the number r of reflex vertices of the map.

To confirm the tightness of this bound, consider the map polygon of Example 3.5 in Figure 3.12 on the page before, which can be looked as a corridor in the east-western direction with large niches in the southern direction and small niches in the northern direction. If we extend the corridor to the east or west by inserting additional niches on both sides of the corridor, we obtain for each niche one new embedding of the skeleton V^* from Example 3.5. (Note that the small niches in the northern direction do not play any role here. They could also be replaced by one long edge.) Since the number of introduced reflex vertices per niche is a constant, we get $\Omega(r)$ niches. □

Since the location of the robot's viewpoint is fixed with respect to the origin v_o of its skeleton V^*, the number of robot positions with visibility polygon $\mathcal{V}$ can be at most the number of embeddings of V^*. Thus, we get the following corollary to Lemma 3.7.

Corollary 3.8* *The number A of solutions to a localization query is at most r and this bound is tight up to a constant factor.*

```
Input:  Polygonal map $\mathcal{M}$ (without holes) with $n$ vertices in total, hereof
        $r \geq 1$ reflex vertices
 1 Remove all vertices with an inner angle of exactly $\pi$
   { Determine the Visibility Cell Decomposition }
 2 Initialize $S \leftarrow$ set of edges of $\mathcal{M}$
 3 for each reflex vertex $v_r$ do
 4    for each vertex $v$ do
 5       if a visibility ray $x$ exists, emanating from $v$, anchored at $v_r$ then
 6          $b \leftarrow$ the point where $x$ hits the boundary of $\mathcal{M}$
 7          $S \leftarrow S \cup \{v_r b\}$
 8       end if
 9    end for
10 end for
11 Compute the arrangement $VCD$ of the segments in $S$
   { Determine the equivalence classes }
12 for each visibility cell $\mathcal{C} \in VCD$ do
13    Compute $V^*_{\mathcal{C}}$          { Note that the representation of $V^*_{\mathcal{C}}$ is independent
                                             of the actual position of $\mathcal{C}$ in the map }
14    Encode $V^*_{\mathcal{C}}$ into a vector $[V^*_{\mathcal{C}}]$ of dimension $d \in O(n)$
15    Insert $[V^*_{\mathcal{C}}]$ into a $d$-dimensional search tree $T_d$
16 end for              { Every leaf of a tree $T_d$ represents one equivalence class }
   { Overlay the visibility cells }
17 for each equivalence class $\mathcal{EC}_{V^*}$ do
18    Let $v_o$ be the origin of the skeleton $V^*$
19    for each cell $\mathcal{C} \in \mathcal{EC}_{V^*}$ do
20       Let $h$ be the embedding whose kernel $h(\ker V^*)$ contains $\mathcal{C}$
21       Place $(\mathcal{C} - h(v_o))$          { That is, $\mathcal{C}$ is placed at the position relative
                                                  to its corresponding embedding }
22    end for
23    Compute the resulting planar subdivision together with a point location
      structure $PL_{V^*}$
24    Link $PL_{V^*}$ to that leaf of one of the search trees that represents $\mathcal{EC}_{V^*}$
25    Label each region $R$ of $PL_{V^*}$ with the corresponding embeddings of the
      cells whose intersection create $R$
26 end for
Output:  Set of search trees $T_d$ and point location structures $PL_{V^*}$
```

ALGORITHM 3.14: Preprocessing of a polygonal map

Now we investigate the structure of the visibility cell decomposition, that is, the complexity and the maximum number of visibility cells for a given map. To this end, recall that the visibility cell decomposition is developed by an arrangement of $\Omega(nr)$ visibility rays in the worst-case, giving a trivial upper bound of $O(n^2r^2)$ on the number of cells. However, taking advantage of the special structure of the problem (i. e., the points that define the visibility rays are vertices of a simple polygon without holes) we obtain the following tighter bound.

Theorem 3.9* *The number of visibility cells (as well as their total complexity) is in $O(n^2r)$ and this bound is tight.*

Proof. We show that the number of vertices in the arrangement of the $O(nr)$ visibility rays is in $O(n^2r)$. Since each ray gives rise to exactly one vertex on the boundary of $\mathcal{M}$ (i. e., $O(nr)$ vertices in total), it suffices to count only the subdivision vertices in the interior of $\mathcal{M}$, in order to show the upper bound.

We now consider a single visibility ray r. Each subdivision vertex v that lies on r rises from a change of the visibility of some other map vertex u. This means, when we walk on r and pass the subdivision vertex v, either the map vertex u comes into sight or disappears from our view. Now is the crucial observation, that when we walk on a straight line in a simple polygon *without holes*, once a vertex disappears from view, it will never become visible again. Thus, there can lie at most $2n$ subdivision vertices on the visibility ray r. Summing up over all rays gives the desired bound of $O(n^2r)$.

To prove that this bound is tight, consider the polygon $\mathcal{M}$ in Figure 3.15 on the facing page, which is similar to the polygon with which we proved the bound of Lemma 3.7. Again, there are $\Omega(r)$ niches on the southern side of a corridor, constructed using $O(r)$ vertices in total. On the opposite side of the corridor there is a convex chain consisting of $\Omega(n)$ vertices, which are placed (inside the shaded area) such that every vertex induces a visibility ray in each niche. Thus, each niche is divided by $\Omega(n)$ rays. By choosing the diameter D of the corridor and the diameter d of the niches sufficiently large, we can ensure that all intersections between visibility rays take place inside the niches. Hence, we get $\Omega(n^2)$ intersections per niche, which gives a total of $\Omega(n^2r)$ visibility cells. □

The following result on the complexity $|\mathcal{EC}_{V^*}|$ of an equivalence class of visibility cells (that is, the total number of vertices and edges of all cells with skeleton V^*) is cited from [GMR97] without the relatively complicated proof. Note that in Chapter 5 we further investigate this complexity in detail and give a sharper bound, where the dependence on the number r of reflex vertices is revealed.

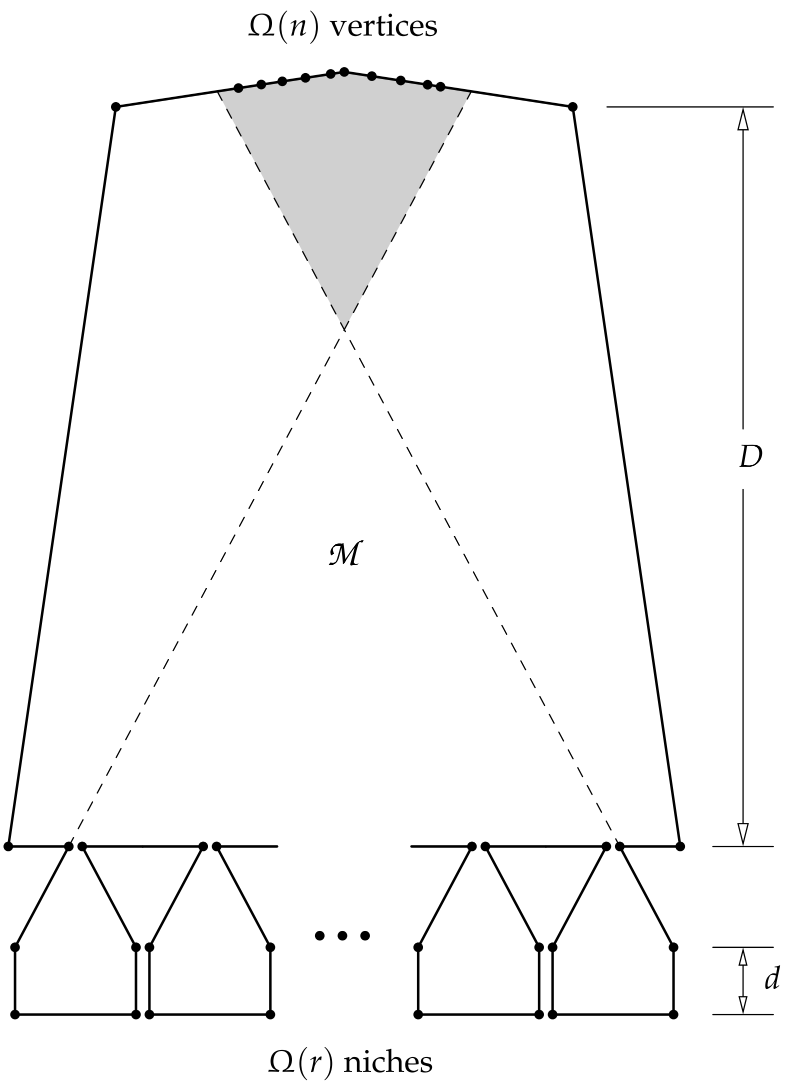


FIGURE 3.15: A map with $\Omega(n^2r)$ visibility cells

Theorem 3.10* *The total number of cells in any equivalence class $\mathcal{EC}_{V^*}$ (as well as their total complexity) is in $O(n^2)$.*

In the following we firstly investigate the steps of Algorithm 3.14 and determine the total running time and space for preprocessing a polygonal map. To this end, let N denote the total complexity of all visibility cells in $\mathcal{M}$ and recall from Theorem 3.9 that $N \in O(n^2r)$.

3.9.1 Computing the Visibility Cell Decomposition

In order to determine all visibility rays of $\mathcal{M}$, we have to identify for each reflex vertex v_r the vertices of $\mathcal{M}$ that are visible from v_r. This can basically be achieved by testing for each vertex v whether the ray $\overleftrightarrow{v_\mathrm{r}\, v}$ is blocked by some portion of the map in Step 5 of Algorithm 3.14. Using standard methods of ray shooting [CE$^+$94], this can be accomplished in time $O(\log n)$ per ray with preprocessing time $O(n \log n)$ and preprocessing space $O(n)$.

Thus, the whole computation of the $O(nr)$ visibility rays requires $O(nr \log n)$ time and $O(nr)$ space. The arrangement of all visibility rays can then be computed in Step 11 by a sweep line algorithm [CE92, dBvK$^+$97] in time $O(nr \log n + N)$ and space $O(N)$.

3.9.2 Determining the Equivalence Classes

In Steps 12 to 16 of Algorithm 3.14 for each visibility cell C the corresponding skeleton V_C^* has to be computed, encoded into a vector $[V_C^*]$ of d real numbers, and inserted into a d-dimensional search tree T_d. Since equivalent visibility cells have identical skeletons, each leaf of one of these search trees represents one equivalence class. The computation of the skeletons can be done in an incremental fashion: First, the visibility polygons (and skeletons, respectively) of the r reflex vertices are computed in time $O(nr)$, see [GH$^+$87, Kle97], and then the remaining skeletons are determined incrementally by walking along the visibility rays, starting at the reflex vertices, where they are anchored. This gives a total of $O(nN)$ time and space for determining and writing down all visibility skeletons, and their encodings, respectively.

The time and space complexity to construct the search tree T_d for an encoding length d is $O(d\,|T_d| + |T_d| \log |T_d|)$, see [Meh84b, Yao90]. Here, $|T_d|$ denotes the size of the tree, that is, the number of cells, whose skeletons have an encoding length of d. This gives a total time and space complexity of $O(nN)$.

3.9.3 Computing the Overlay Arrangement

In order to compute for a single equivalence class $\mathcal{EC}_{V^*}$ the arrangement of all of its visibility cells, we have to place the cells at positions such that the origin vertices of the corresponding embedded skeletons coincide (Steps 18 to 22 of Algorithm 3.14), see Figure 3.12 on page 43. Using the notion $|\mathcal{EC}_{V^*}|$ for the complexity of the equivalence class (that is, the total number of vertices and edges of all cells in $\mathcal{EC}_{V^*}$), the computation of the overlay arrangement in Step 23 can be performed in time

$$O\left(|\mathcal{EC}_{V^*}| \cdot \log |\mathcal{EC}_{V^*}| + |\mathcal{EC}_{V^*}|^2\right) = O\left(|\mathcal{EC}_{V^*}|^2\right) \qquad (3.1)$$

and space $O(|\mathcal{EC}_{V^*}|^2)$, again with a sweep line method [CE92, dBvK+97]. The point location structure PL_{V^*} can then be computed with a time and space complexity, which is linear in the subdivision size, see [Kir83, dBvK+97], that is $O(|\mathcal{EC}_{V^*}|^2)$.

It remains to describe how the labeling in Step 25 is managed, since the straightforward idea of adjoining to each region R of PL_{V^*} a list of the embeddings of the cells whose intersection create R would increase the space requirements by a factor of $O(r)$. The idea described in [GMR97] avoids this problem by using interval trees and fractional cascading without any additional blow-up in time or space.

Summing up over all equivalence classes, the total time and space costs are

$$O\left(\sum_{V^*} |\mathcal{EC}_{V^*}|^2\right) = O\left(n^2 \sum_{V^*} |\mathcal{EC}_{V^*}|\right) = O\left(n^2 N\right) , \qquad (3.2)$$

using $|\mathcal{EC}_{V^*}| \in O(n^2)$ from Theorem 3.10 and $N = \sum_{V^*} |\mathcal{EC}_{V^*}|$.

This finishes the analysis of Algorithm 3.14 and we summarize the preceding discussion in the following theorem.

Theorem 3.11* *The preprocessing of a polygonal map $\mathcal{M}$ with n vertices in total, hereof r reflex vertices, using Algorithm 3.14 has a time and space complexity of $O(n^2N)$, where $N \in O(n^2 r)$ describes the number of visibility cells of $\mathcal{M}$.*

3.9.4 Costs of a Localization Query

In the previous sections we already have described the data structures and algorithms for maintaining the search trees T_d and the point location structures PL_{V^*}. Thus, we can now easily analyze the time of a localization query.

Theorem 3.12* *For a polygonal map $\mathcal{M}$ with n vertices in total, which is already preprocessed by Algorithm 3.14, and a visibility polygon $\mathcal{V}$ with m vertices in total, using Algorithm 3.13 the localization problem can be answered in time $O(m + \log n + A)$, where $A \in O(r)$ denotes the number of possible placements of $\mathcal{V}$ in $\mathcal{M}$.*

Proof. Clearly, Steps 1 to 3 of Algorithm 3.13, where the skeleton V^* and its encoding vector $[V^*]$ are computed, can be accomplished in $O(m)$ time.

An exact match query to the search tree T_d (Step 4), which yields the equivalence class $\mathcal{EC}_{V^*}$, costs $O(d + \log |T_d|) = O(d + \log n)$ due to [Meh84b, Yao90].

Equally, due to [Kir83, dBvK+97] the point location query in PL_{V^*} is logarithmic in the size of PL_{V^*}, that is, in $O(\log |\mathcal{EC}_{V^*}|^2) = O(\log n)$, using the result of Theorem 3.10.

The remaining steps of the algorithm, where the label of the reported region is retrieved and for each visibility cell (of the label) a coordinate transformation is performed, can be accomplished in time $O(\log n + A)$. Summing all up yields the desired outcome. The upper bound of $O(r)$ on the maximum number A of solutions is already known from Corollary 3.8. □

Combining the two theorems from above we get the main theorem of Guibas et al.

Theorem 3.13* *The localization problem can be solved with preprocessing time and space of $O(n^2N)$, and a query time of $O(m + \log n + A)$.*

The following results, which trade off the query time with the preprocessing costs, we only cite from [GMR97]:

Theorem 3.14* *Let $1 \leq f(n) \leq n$. Then, the localization problem can be solved with preprocessing time and space of $O(n^2N/f(n))$, and a query time of $O(m + f(n)\log n + A)$.*

Theorem 3.15* *The localization problem can be solved with preprocessing time and space of $O(nN)$, and a query time of $O(m + r\log n + A)$.*

3.10 Summary

In this chapter we introduced the robot localization problem in its idealized geometric version and described the approach of Guibas et al. to solving it, which will be the basis for our succeeding investigations. Even though many of the definitions, lemmas, and theorems are adopted from their work [GMR97] with only slight modifications (the respective parts are marked with a star »⋆«), recall that there is no one-to-one correspondence of definitions and claims between our work and the original one, since we modified the structure at several points anticipating the needs of the following chapters.

In the following we briefly recall the basic ideas of the described approach: From a given visibility polygon $\mathcal{V}$ we derived a coarsened representation, its visibility skeleton V^*, which (at first sight) contains only the full edges and vertices of $\mathcal{V}$ that can be verified to have counterparts in the map. The spurious edges and their incident partial edges in the visibility polygon are caused by blocking vertices, which are responsible for the occlusions in $\mathcal{V}$. In the corresponding skeleton V^* they are replaced by artificial edges between the blocking vertices, labeled with a characterization of the line on which the original partial edge lies. We proved that a skeleton V^* contains enough details to fully reconstruct the original visibility polygon $\mathcal{V}$ if we also know the viewpoint of $\mathcal{V}$ (Theorem 3.2).

We divided the map polygon into a number of $O(n^2 r)$ convex visibility cells such that for any two points $p, q \in \mathcal{C}$ of a cell their visibility skeletons V_p^* and V_q^* are identical (Theorem 3.3 and 3.9), although their visibility polygon $\mathcal{V}_p$ and $\mathcal{V}_q$ may differ. As a consequence, each cell $\mathcal{C}$ has an uniquely defined skeleton $V_{\mathcal{C}}^*$. We denoted by $\mathcal{EC}_{V^*}$ the equivalence class of all cells with skeleton V^* and defined their complexity $|\mathcal{EC}_{V^*}|$ as the total number of vertices and edges of all visibility cells with skeleton V^*. Guibas et al. showed that this complexity is in $O(n^2)$ and we will refine it to $O(n + r^2)$ in Chapter 5.

In order to localize a given visibility polygon $\mathcal{V}$ in the map, we used its coarsened representation V^* to determine a number of placements where the translated V^* matches into the map. Such a matching is called an embedding of V^*. In the localization process we then have to check for each embedding whether the also translated viewpoint (that is, the origin of the visibility polygon) induces the correct visibility polygon $\mathcal{V}$. By using a suitably preprocessed map this could be performed by a single point location query.

These considerations resulted in a localization algorithm with a query time of $O(m + \log n + A)$, where $A \in O(r)$ denotes the number of possible placements of the visibility polygon (Theorem 3.12). The preprocessing costs were shown to be in $O(N\,|\mathcal{EC}_{V^*}|)$, where $N \in O(n^2 r)$ describes the total number of visibility cells of the map (Theorem 3.11).

4

Overlapping Embeddings

*“**Where am I?** Tell the truth!*
Fear not to tell! Oh, spare me not!
Where? Where?
Have I fallen like a shooting star?”

Cyrano in “Cyrano de Bergerac”
E. ROSTAND (1868–1918)

THE NOTION OF AN EMBEDDING of a skeleton V^* as a place in the map polygon $\mathcal{M}$, where V^* “fits” into $\mathcal{M}$, is very intuitive and easy to understand. But in the case of embeddings, our intuition may also lead us very easily to completely wrong implications, for example, like the conjecture

> “For each embedding $h(V^*)$ there exists a point $p \in \mathcal{M}$ such that $V^*_p = V^*$ holds.”,

which was already disproved in the previous chapter by Figure 3.10 on page 38. This chapter deals with another of these misleading implications, namely with overlapping embeddings, that is, embeddings of the same skeleton, where the embedded skeleton polygons or even the embedded kernels overlap. Such cases, which at first sight seem to be impossible, are the major source of complication if we investigate some of the occurring complexities of the localization problem, and consequently are our main subject of investigation in the succeeding sections.

In the following we first give some examples of embeddings that overlap and show that even the (trivial) bound of at most $O(r)$ overlapping embeddings is tight. We then state the main result of this chapter, which describes the connection between overlapping embeddings and the visibility of their witnesses and claims that these witnesses are not visible to each other. This property (besides that it represents an interesting problem by itself) plays a crucial role in the next chapter, where we further investigate the complexity of an equivalence class, and its proof is the main topic of the following sections.

4.1 Some Examples

In the sequel we give two examples, each consisting of two different embeddings of a skeleton V^*, which contradict our intuition in the sense that the two skeleton polygons overlap. In order not to get confused with "strange" embeddings (like the one of Figure 3.10), we want to consider in the following only embeddings of a skeleton V^*, for which at least one point p with skeleton $V_p^* = V^*$ exists. We call such an embedding a *valid embedding* and define it as follows.

Definition 4.1 (Valid Embedding and Valid Candidate Edge)
*An embedding $h(V^*)$ is called a **valid embedding** of the skeleton V^* if there exists a point $w \in h(\ker V^*)$ with $V_w^* = V^*$. Such a point w is called a **witness for the embedding** $h(V^*)$.*

*A candidate edge c of an embedded artificial edge $h(a)$ is called a **valid candidate edge** if there exists a witness w such that c is part of $\mathcal{V}_w$.*

Now recall Lemma 3.5 on page 40, which states that for any point $p \in \mathcal{M}$ there exists exactly one embedding $h(V_p^*)$ with $p \in h(\ker V_p^*)$. One might think that from this argument it follows that two embeddings of the same skeleton can never overlap (or at least, their kernels can not overlap), since in that case, there would exist such a point p, which lies in the kernels of two different embeddings of the same skeleton, contradicting Lemma 3.5.

Another (fallacious) reason against overlapping embeddings may be that there seems to exist no way of placing two identical skeletons V^* into a map polygon such that they overlap, without intersecting the full edges of the corresponding embedded skeleton polygons $V^\diamond$.

But both arguments ignore placements of the skeletons where all intersection points are either vertices of the two skeleton polygons or intersections between two *artificial* edges. The following examples, where this is the case and where the skeletons as well as their kernels overlap, disprove both conjectures.

Example 4.1 (A map with overlapping embeddings and kernels)
Figure 4.1 (a) on the facing page shows a map polygon $\mathcal{M}$ with two viewpoints p and q that have the same visibility polygon $\mathcal{V} = \mathcal{V}_p = \mathcal{V}_q$, depicted in Figure 4.1 (b). Figure 4.1 (c) shows the corresponding skeleton V^* and its kernel. As we can see, there exist two (valid) embeddings of V^* into the map such that different artificial edges of the skeleton are either mapped onto the same pair of vertices in the map (the three artificial edges of V^* in the northern direction) or share a common vertex in the map (the two remaining artificial edges). Moreover, the two embedded kernels overlap in the shaded area, as can be seen in the upper part of the figure. $\triangleleft$

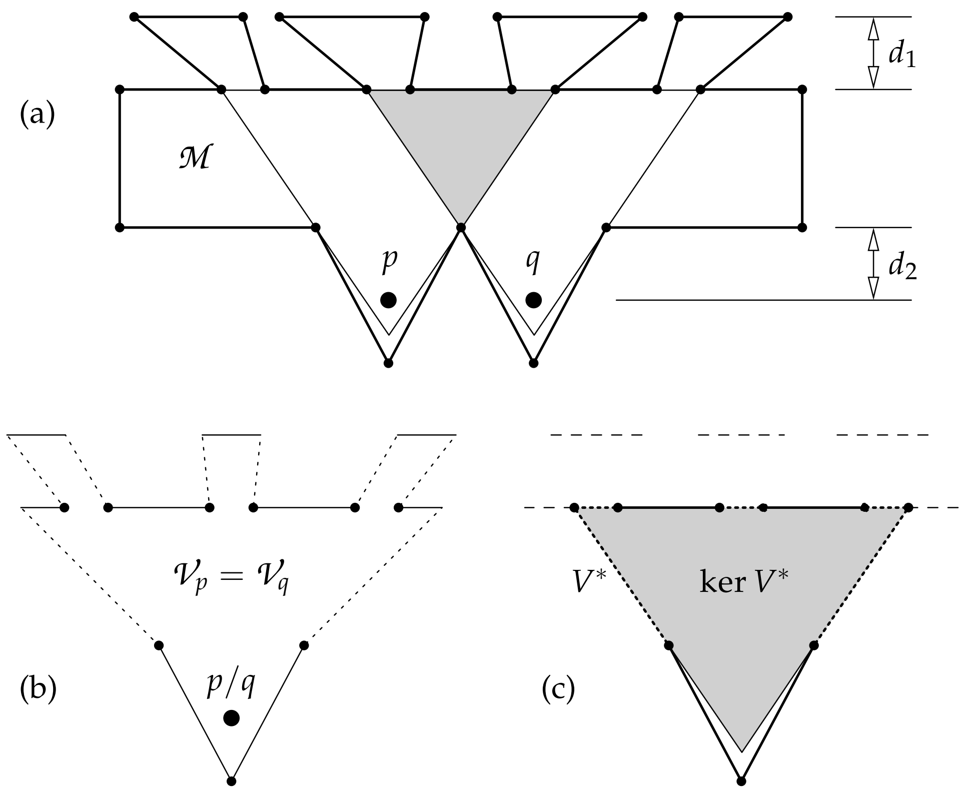


FIGURE 4.1: An example of overlapping embeddings and kernels

Example 4.2 Figure 4.2 on the next page shows a very similar example of two (valid) embeddings with overlapping kernels. The major difference is that in Figure 4.2 the embedded artificial edges even do *strictly* intersect (due to the different construction of the niches), whereas in Figure 4.1 the embedded artificial edges always had a vertex in common. $\lhd$

The following lemma shows that the actual number of overlapping embeddings or kernels may even be as large as the total number of embeddings of V^*, that is, in $\Omega(r)$, cf. Lemma 3.7 on page 42.

Lemma 4.1 *The number of embeddings of a given skeleton, which all overlap in one point, is in $O(r)$ and this bound is tight.*

Proof. In Lemma 3.7 we have already stated that the total number of (not necessarily overlapping) embeddings of a given skeleton is in $O(r)$, so we only have to show the tightness of the bound.

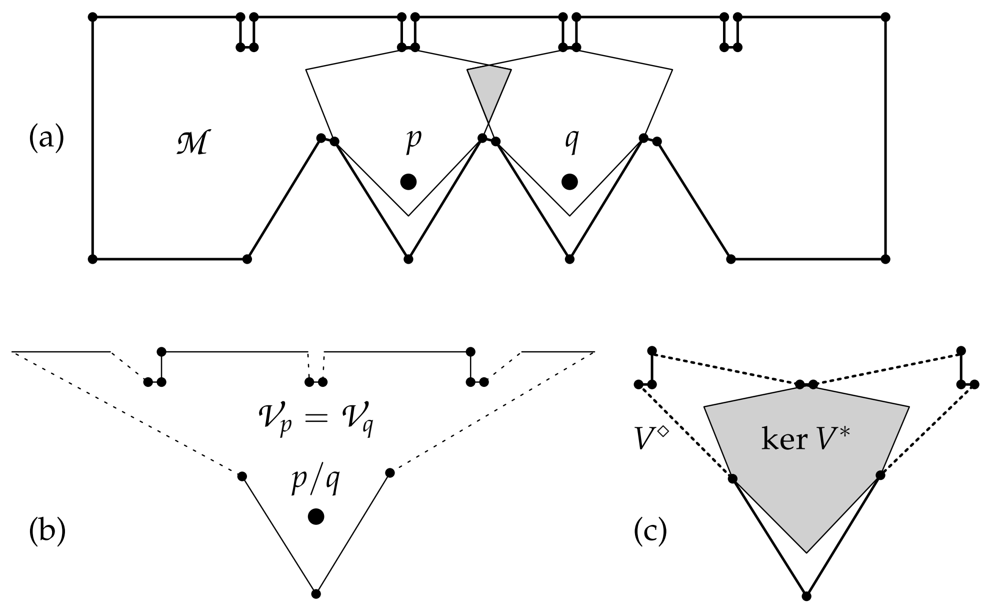


FIGURE 4.2: Another example of overlapping embeddings with strict intersections

To this end, consider the map $\mathcal{M}$ of Figure 4.1 (a). For any given natural numbers $x, y \in \mathbb{N}$, we can extend the map in the east-western direction such that there are x equidistant niches in the southern direction and y equidistant niches in the northern direction, by introducing a total of $O(x + y)$ reflex vertices. Let $z \in \mathbb{N}$ be any natural number. If we choose $x := 2z + 1$, $y := 2x - 1 = 4z + 1$, and appropriately reduce the distance d_2, which describes how deep the viewpoints stand in the southern niches, we can achieve that from each viewpoint exactly $2z + 1$ niches in the northern direction are visible. To assure that no other vertices in any of the niches are visible from any of the viewpoints, the width of the niches has to be chosen large enough, which could have the consequence that the niches overlap each other. But this could easily be avoided by reducing the depth d_1 of the niches.

To recapitulize, the skeletons of the $2z + 1$ viewpoints are identical, each containing $2z + 1$ horizontal artificial edges. Moreover, the most eastern embedding and the most western embedding of this skeleton share exactly one artificial edge in the map. Thus, these embeddings (as well as their kernels) overlap in a triangular region to the south of that artificial edge. Of course, the kernels of the remaining $2z - 1$ embeddings also contain this triangular region. Since the number r of reflex vertices of the thus generated map is in $O(z)$, we have shown that the number of overlapping embeddings is in $\Omega(r)$. □

4.2 The Connection between Overlapping Embeddings and their Witnesses

If we review the examples of Figure 4.1 and 4.2, we notice that in both cases the witnesses p and q of the two overlapping (valid) embeddings cannot see each other. And in fact, it seems impossible to construct an example of overlapping embeddings, so that the witnesses are visible to each other. Actually, we can show the following main theorem of this chapter, which points out the connection between overlapping embeddings and the visibility of their witnesses.

Theorem 4.2 *Let $h(V^*)$ and $h'(V^*)$ be two different valid embeddings of the same skeleton V^* with witnesses w and w'. If the embedded skeleton polygons $h(V^\diamond)$ and $h'(V^\diamond)$ overlap, that is,*

$$\big(h(V^\diamond) \cap h'(V^\diamond)\big)^\circ \neq \varnothing\,,$$

the witnesses w and w' cannot see each other.

Since the proof of this theorem is quite complicated, we first prove a similar theorem with stronger prerequisites. Namely, we assume that not only the embedded skeleton polygons overlap, but also their kernels. Then, the proposition can be proved much easier, since the existence of a common kernel point of the two skeletons is a very valuable property, of which we can take advantage in many ways. Then, we show some basic properties of overlapping embeddings in general (for example, that the intersection of two arbitrary valid embeddings always consists of a single polygonal region), which will be helpful in the sequel. Finally, we prove in the case of overlapping embeddings of *identical* skeletons that a certain structure consists, which allows us to adapt the proof of the less general theorem in order to prove Theorem 4.2.

4.3 A Result on Overlapping Kernels

In the following we consider a special case of Theorem 4.2, where also the kernels of the embedded skeletons overlap. Thereby, the existence of a point that sees *both* skeleton polygons at the same time is very useful for the proof.

Theorem 4.3 *Let $h(V^*)$ and $h'(V^*)$ be two different valid embeddings of the same skeleton V^* with witnesses w and w'. If the embedded kernels $h(\ker V^\diamond)$ and $h'(\ker V^\diamond)$ overlap, that is,*

$$\big(h(\ker V^*) \cap h'(\ker V^*)\big)^\circ \neq \varnothing\,,$$

the witnesses w and w' cannot see each other.

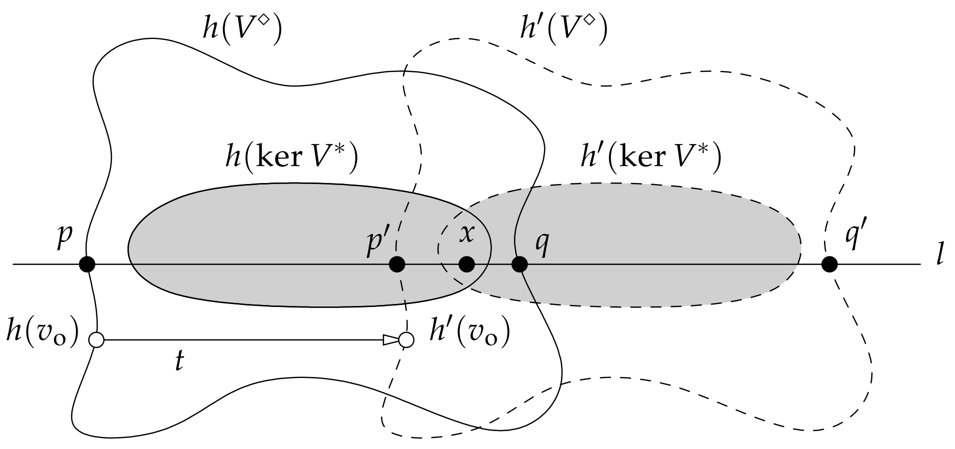


FIGURE 4.3: Two embeddings of V^* with overlapping kernels

Proof. We prove this claim by contradiction. Since $h(V^*)$ and $h'(V^*)$ are different embeddings of V^*, there must exist a vector $t \neq 0$ that translates the embedded origin $h(v_o)$ onto the embedded origin $h'(v_o)$, that is, $h(v_o) + t = h'(v_o)$. Without loss of generality we assume that t is horizontal and directed from left to right, see Figure 4.3.

Let $x \in (h(\ker V^*) \cap h'(\ker V^*))^\circ$ be a point lying in the interior of both embedded kernels and let l be the straight line through x and parallel to the vector t. The line l intersects the boundary of the skeleton polygon $h(V^\diamond)$ at exactly two points, $p \in \partial\, h(V^\diamond)$ and $q \in \partial\, h(V^\diamond)$. There cannot exist less than two intersection points, because x lies in the interior of $h(V^\diamond)$, and there cannot be more than two intersection points, since x is a kernel point of $h(V^\diamond)$. Analogously, we get intersection points p' and q' of $\partial\, h'(V^\diamond)$ with l, see Figure 4.3. Furthermore, it is also clear that $p' \neq q$ and that p' lies strictly to the left of q because of $x \in (h(\ker V^*) \cap h'(\ker V^*))^\circ$.

Obviously, the point q' does not lie in $h(V^\diamond)$ due to $q' = q + t$. And since q' is visible from x, the point q must lie on an artificial edge $a = \overline{a_1\, a_2}$ of $h(V^*)$. Otherwise, q' would be occluded by the full edge on which q lies (see Figure 4.4 on the next page). Since $h'(V^*) = h(V^*) + t$, also an artificial edge $a' = \overline{a_1'\, a_2'}$ of $h'(V^*)$ exists. Using the same argumentation with the points p and p' yields a second pair of artificial edges in direction $-t$: $b' = \overline{b_1'\, b_2'} \in \partial\, h'(V^\diamond)$ and $b = \overline{b_1\, b_2} = \overline{b_1'\, b_2'} - t \in \partial\, h(V^\diamond)$, see Figure 4.4. Note that the vertices a_1, a_1', b_1, and b_1' (and also the vertices a_2, a_2', b_2, and b_2', respectively) need not necessarily lie on the same straight line.

Now we first investigate the possible positions of the two witnesses w and w'.

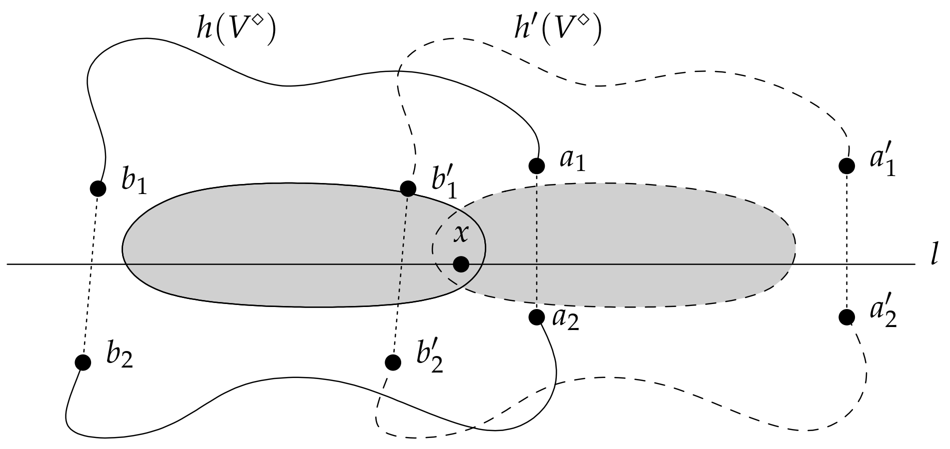


FIGURE 4.4: The line l intersects four artificial edges of the two embedded skeletons

Lemma 4.4 *For the witnesses w and w' the following holds:*

- *The witness w either lies above the straight line $\overleftrightarrow{a_1\, a'_1}$ (parallel to l) or below the straight line $\overleftrightarrow{a_2\, a'_2}$ (parallel to l) and*
- *the witness w' either lies above the straight line $\overleftrightarrow{b_1\, b'_1}$ (parallel to l) or below the straight line $\overleftrightarrow{b_2\, b'_2}$ (parallel to l).*

Proof of Lemma 4.4. Due to the symmetry of the situation it suffices to prove the claim only for the witness w. To this end, we assume that w lies *between* the two described straight lines and show that this leads to a contradiction (independent from whether w and w' see each other). Furthermore, we assume w. l. o. g. that the second witness w' lies below the line l, again exploiting the symmetry of the situation.

Since w is a witness for $h(V^*)$, it follows $V^*_w = V^*$. Thus, the artificial edge a must be visible to w and no other vertex may lie in the view cone of w looking through a. Particularly, neither one of the vertices a'_1 and a'_2 may be visible to w. Due to the placement of w this can only be accomplished in one way, namely by a segment $\overline{s_1\, s_2}$ that lies between w and the vertices a'_1 and a'_2. The endpoints of $\overline{s_1\, s_2}$ must not be visible to w, so that one endpoint must lie above the view cone of w looking through a and the other endpoint must lie below the view cone (see Figure 4.5 on the following page). In particular, s_1 lies above $\overleftrightarrow{a_1\, a'_1}$ and s_2 lies below $\overleftrightarrow{a_2\, a'_2}$.

Then, the lower endpoint s_2 of this segment must be somehow connected to the upper endpoint a'_1 of the artificial edge $\overline{a'_1\, a'_2}$ in $h'(V^*)$ rather than to its

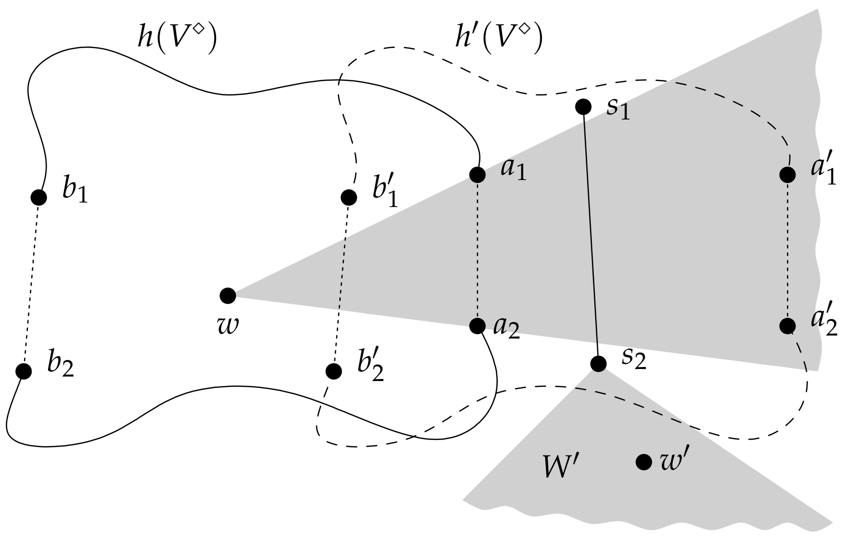


FIGURE 4.5: A segment between w and $\overleftrightarrow{a'_1 a'_2}$ implies a contradiction

lower endpoint a'_2, since otherwise a'_1 would not be visible to the witness w' of $h'(V^*)$, which we assumed to lie below l. This has the consequence that all kernel points of $h'(V^*)$ must lie (in order to see a'_1 and b'_1) in the region W' (shaded gray in the figure), which is determined by the straight lines $\overleftrightarrow{s_2\, a'_1}$ and $\overleftrightarrow{s_2\, b'_1}$, respectively. In particular, the kernel $h'(\ker V^*)$ must lie *below* s_2, which itself lies below $\overleftrightarrow{a_2\, a'_2}$.

Now recall that $h'(V^*)$ is a translated copy of $h(V^*)$ with a translation vector t that is parallel to $\overleftrightarrow{a_2\, a'_2}$. Thus, also the kernel $h(\ker V^*)$ must lie below $\overleftrightarrow{a_2\, a'_2}$, which contradicts the placement of w. $\square$

In the following we study the placements of the eight endpoints of the artificial edges introduced above. To this end, let $\gg\prec_x\ll$ and $\gg\preccurlyeq_x\ll$ describe the counterclockwise angular order with respect to the point x, starting at direction t. The placement of the eight points is then restricted as follows:

- $a'_1 \prec_x a_1$, since $a'_1 = a_1 + t$ and both points lie above the line l.
- $a'_1 \prec_x b'_1$. As both points lie above l and are from the same embedding $h'(V^*)$, at least $a'_1 \preccurlyeq_x b'_1$ must hold. Furthermore, from $a'_1 =_x b'_1$ would follow $a'_1 = b'_1$ (and also $a_1 = b_1$), since x is an interior point of $h'(\ker V^*)$. Consequently, neither one of the two witnesses could lie above the straight line through a_1/b_1 and $a'1/b'_1$. Thus, we would conclude from Lemma 4.4 that w must lie below $\overleftrightarrow{a_2\, a'_2}$ and that w' must lie

below $\overleftrightarrow{b_2 b_2'}$. This would imply that for at least one of the two witnesses the view on its corresponding artificial edges would be blocked by vertices of the other embedding (cf. Figure 4.6 (a) on the next page, which illustrates the case »The points w and w' lie on the same side of l«).

Thus, $a_1' =_x b_1'$ is impossible and $a_1' \prec_x b_1'$ must hold.

- $a_1 \prec_x b_1$ and $b_1' \prec_x b_1$ with the same arguments as in the two cases above.
- $b_1 \prec_x b_2$, because the two vertices lie on different sides of l.
- $b_2 \prec_x b_2'$, $b_2 \prec_x a_2$, $b_2' \prec_x a_2'$ and $a_2 \prec_x a_2'$ again with the same arguments as before.

Note that the angular order with respect to x of these points also describes the order in which the vertices occur in the map polygon, that is, in which order they are connected by map edges. This plays a crucial role in the following. As a result of the above considerations, we must distinguish between four cases of angular orders or four different sequences of vertices in the map polygon, respectively:

1. $a_1' \prec_x a_1 \prec_x b_1' \prec_x b_1 \prec_x b_2 \prec_x b_2' \prec_x a_2 \prec_x a_2'$, as depicted in Figure 4.4;
2. $a_1' \prec_x \boldsymbol{b_1' \preccurlyeq_x a_1} \prec_x b_1 \prec_x b_2 \prec_x b_2' \prec_x a_2 \prec_x a_2'$, where the two middle vertices above l are swapped (or identical, respectively);
3. $a_1' \prec_x a_1 \prec_x b_1' \prec_x b_1 \prec_x b_2 \prec_x \boldsymbol{a_2 \preccurlyeq_x b_2'} \prec_x a_2'$, where the two middle vertices below l are swapped (or identical, respectively);
4. $a_1' \prec_x \boldsymbol{b_1' \preccurlyeq_x a_1} \prec_x b_1 \prec_x b_2 \prec_x \boldsymbol{a_2 \preccurlyeq_x b_2'} \prec_x a_2'$, where both pairs of middle vertices are swapped or identical. This case immediately would contradict the fact that p' lies strictly to the left of q and would result in $x \notin (h(\ker V^*) \cap h'(\ker V^*))^\circ$. Thus, this case is not possible, because the kernels would not overlap. But note that examples exist where the kernels only intersect in a straight line segment (i. e., they do not *overlap*) and w and w' are visible to each other, see Figure 4.28 on page 93.

In the remaining part of the proof we want to study the placements of the two witnesses $w \in h(\ker V^*)$ and $w' \in h'(\ker V^*)$ and the eight points introduced above. To this end, we distinguish two main cases: The witnesses w and w' lie either both on the same side of the straight line l or on different sides. For the first case we show that w and w' cannot see each other and the second case is shown to be impossible, independent of the visibility between w and w'.

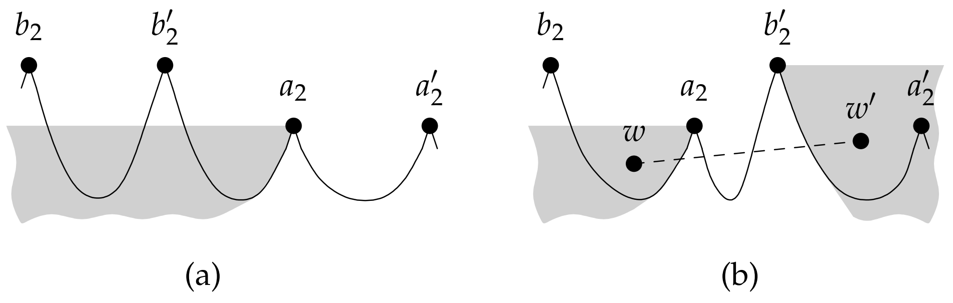


FIGURE 4.6: The placement of the two witnesses if they lie on the same side of l: (a) If $b_2' \prec_x a_2$, no suitable placement for w exists. (b) If $a_2 \preccurlyeq_x b_2'$, w and w' cannot see each other

The points w and w' lie on the same side of l

Without loss of generality we assume that both witnesses lie below l. From Lemma 4.4 then particularly follows that w lies below $\overleftrightarrow{a_2\, a_2'}$ and that w' lies below $\overleftrightarrow{b_2\, b_2'}$. Now we assume w. l. o. g. that the vertices a_2 and a_2' lie below $\overleftrightarrow{b_2\, b_2'}$, since in the other case we could use the same argumentation by interchanging the roles of b_2 and a_2', b_2' and a_2, and w and w'. Then, we get one of the situations depicted in Figure 4.6, depending on the angular order of the vertices b_2' and a_2:

If $b_2' \prec_x a_2$ like in Figure 4.6 (a), the witness w, which must lie in the shaded area, cannot be placed such that it sees both vertices a_2 as well as b_2, because the vertex b_2' blocks the view of w on either a_2 or b_2. Thus, this case results in a contradiction and is not possible.

In the other case, if $a_2 \preccurlyeq_x b_2'$, there are suitable placements for the two witnesses, see Figure 4.6 (b), such that w sees a_2 and b_2 and w' sees a_2' and b_2'. But in this case, the two witnesses obviously cannot see each other, because the vertex b_2' blocks the line of sight between them. This completes the first case, since "w and w' cannot see each other" is the claim that had to be proven. Observe that the examples shown in Figure 4.1 and 4.2 are both from this type.

The points w and w' lie on different sides of l

In the following we show that this case leads to a contradiction, independent from whether w and w' see each other. Without loss of generality we assume that w lies below l and w' lies above l. We again conclude from Lemma 4.4 that w lies below $\overleftrightarrow{a_2\, a_2'}$ and w' lies above $\overleftrightarrow{b_1\, b_1'}$. From the above distinction of cases of possible angular orders we conclude that at least one of the following properties must hold: $a_1 \prec_x b_1'$ or $b_2' \prec_x a_2$. Therefore, we assume w. l. o. g. that $b_2' \prec_x a_2$, since in the other case we could use an almost identical argumentation

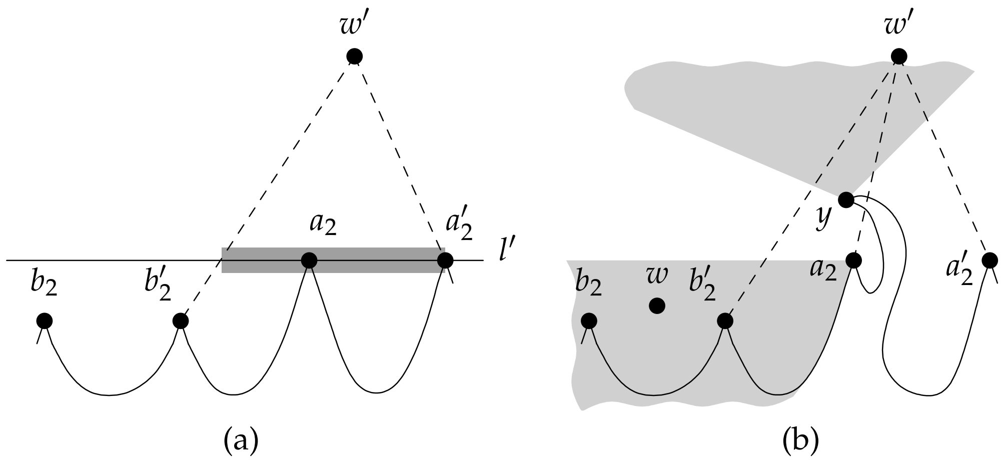


FIGURE 4.7: A counterexample if the two witnesses lie on different sides of l: (a) The vertex a_2 must be visible to w'. (b) Vertex y occludes the sight from w' to a_2

by interchanging the roles of w and w', a_1 and b_2', a_1' and b_2, and so on.

We then conclude that the vertices b_2 and b_2' must lie below $\overleftrightarrow{a_2 a_2'}$, since otherwise there is no suitable placement for the witness w. Confer the situation in Figure 4.6 (a) and recall that w lies below the line through a_2 and a_2'. Now we show that at least one additional vertex y, which is visible to w', must lie between a_2' and b_2' in the map polygon. To this end, we first show that a_2 lies in the view cone of w' looking through $\overline{a_2' b_2'}$: Due to the construction of a_2 it must lie on a straight line l' parallel to l through a_2'. Since $b_2' \prec_x a_2 \prec_x a_2'$, these points must be connected by polygon edges in the same order, as shown in Figure 4.7 (a).

Because w' sees a_2' as well as b_2' (since both vertices are from h') and the line of sight between w' and b_2' intersects l' (since b_2' lies below l'), the vertex a_2 must lie on the shaded part of line l' in Figure 4.7 (a). Thus, a_2 must be either visible to w' (in this case we use $y = a_2$) or it must be occluded by some other vertices and edges of the map polygon, see Figure 4.7 (b). In the latter case we choose y to be one of these occluding vertices, which is visible to w'. We observe that the so-defined vertex y always lies on or above the line l'. Furthermore, since the vertices a_2' and b_2' are both visible to w', y must lie between them in the map polygon.

From this constellation follows that the kernel $h'(\ker V^*)$ must lie in the upper gray shaded region in Figure 4.7 (b), since otherwise the vertices a_2' and b_2' would not be visible from within the kernel. In particular, $h'(\ker V^*)$ must lie *above* l'. Since the kernel $h(\ker V^*)$ results from a (horizontal) translation of $h'(\ker V^*)$ by $-t$, also $h(\ker V^*)$ lies above the line l', which is parallel to t. The placement of the kernel contradicts the fact that the witness w lies *below* l' and proves that w and w' cannot lie on different sides of l.

This completes the proof, since we have shown for all possible cases that either w and w' cannot see each other, or that the respective case implies a contradiction. □

Remark 4.3 Most parts of the previous proof do not rely on the existence of the common kernel point x of the two embeddings $h(V^*)$ and $h'(V^*)$, but instead use only the most important property of valid embeddings, namely the existence of a witness that sees the whole skeleton polygon. Basically, the only point in the proof where we make use of the common kernel point x is the construction of the two pairs of artificial edges a/a' and b/b', which are intersected by a straight line l that is parallel to the translation vector t. Also the distinction of cases of possible angular orders (with respect to x) can be accomplished without a kernel point, since we actually are not interested in the *angular* order of the points, but only in the order in which they occur on the boundary of the map polygon. (Note that in the case of star-shaped polygons these two orders are identical and that skeleton polygons are star-shaped.) ◁

Consequently, if we are able to construct the four artificial edges of the previous proof without using a common kernel point of the two embeddings, we should be able to prove also the more general Theorem 4.2 in almost the same way. Particularly note that it is not necessary that x actually sees the endpoints of the four artificial edges. The only property that is required for the proof is that a straight line l exists, which intersects artificial edges a and b such that the intersection point of l with $h(a)$ lies to the right of the intersection point of l with $b' = h'(b) = h(b) + t$. Showing that these kinds of artificial edges always exist is the aim of the subsequent sections.

4.4 Properties of Overlapping Embeddings

In the following we first investigate the overlapping of two embedded skeletons and prove some properties, which are useful in the remaining part of this chapter.

Firstly, we show that in the case of overlapping embeddings an embedded artificial edge cannot be arbitrarily often intersected by artificial edges of the other embedding. See Figure 4.2 on page 56 for an example, where each embedded artificial edge of the first embedding intersects at most one artificial edge of the second embedding. Note that for the succeeding lemmas we do

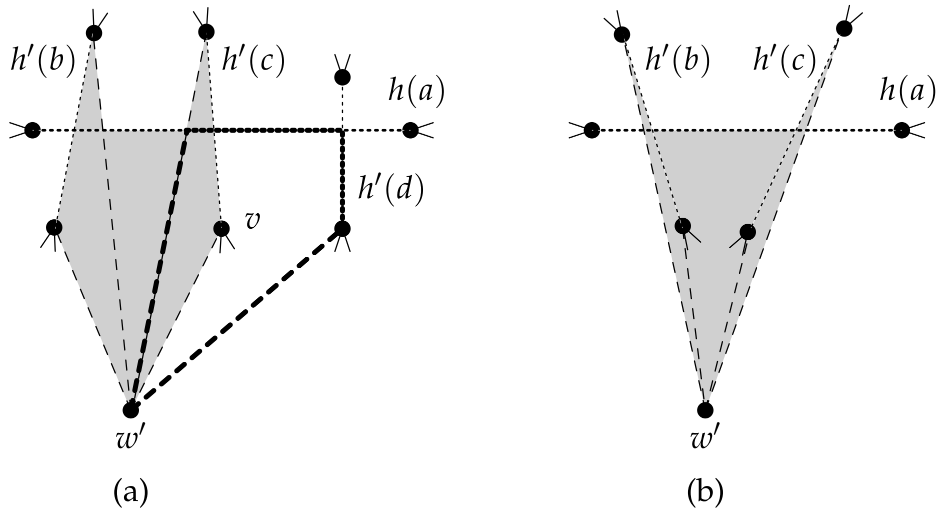


FIGURE 4.8: Intersecting artificial edges of two embeddings

not necessarily assume that the two skeletons, whose embeddings overlap, are *identical*. The propositions remain true also for different overlapping skeletons.

Lemma 4.5 *Let $h(V_1^*)$ and $h'(V_2^*)$ be two valid embeddings. For each embedded artificial edge $h(a)$ of V_1^* there exist at most two embedded artificial edges of V_2^* that intersect $h(a)$. Furthermore, the fraction of $h(a)$ that possibly lies on the boundary of the intersection of $h(V_1^\diamond)$ and $h(V_2^\diamond)$ is a single straight line segment.*

Proof. Let the embedded artificial edge $h(a)$ of skeleton V_1^* intersect the embedded artificial edges $h'(b)$ and $h'(c)$ of skeleton V_2^* as depicted in Figure 4.8 (a).

Since $h(V_1^*)$ and $h'(V_2^*)$ are valid embeddings, there must exist witnesses w and w' such that $V_w^* = V_1^*$ and $V_{w'}^* = V_2^*$. In particular, there must exist lines of sight (drawn dashed in the figure) between w' and the four endpoints of the embedded artificial edges (drawn dotted) and, furthermore, also between the two map vertices of each embedded artificial edge. Thus, in the shaded area of Figure 4.8 (a) there cannot lie any further map vertex, since otherwise Observation 3.3 on page 27 would be contradicted. Now suppose that a third artificial edge $h'(d)$ of V_2^* intersects $h(a)$. Then, independently of the placement of its two endpoints, at least one vertex v of the remaining vertices would be surrounded by a circle of lines of sight (drawn bold in the figure), which again would lead to a contradiction. It can easily be seen that this proposition remains true, even if the endpoints of the intersecting artificial edges are allowed

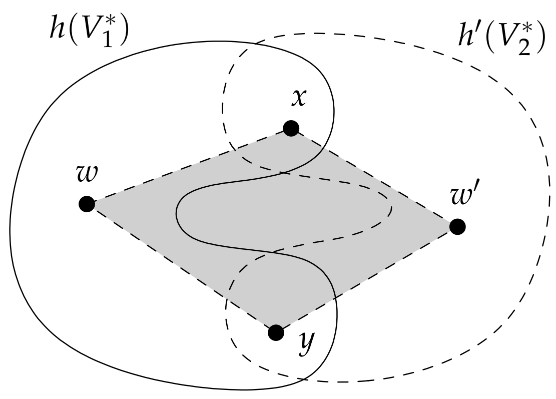


FIGURE 4.9: Overlapping embeddings with two intersection regions

to coincide with the endpoints of $h(a)$. Therefore, at most two artificial edges of $h'(V_2^*)$ may intersect $h(a)$.

With the same argumentation we also conclude that the part of $h(a)$ that lies on the boundary of the intersection of $h(V_1^\diamond)$ and $h(V_2^\diamond)$ is a single straight line segment. Otherwise, the intersection must obviously consist of two straight line segments. Thus, the embedded artificial edges of V_2^*, which intersect $h(a)$, must be oriented like those in Figure 4.8 (b), which again would contradict Observation 3.3, since at least two map vertices would be surrounded by a circle of lines of sight. □

The succeeding lemmas investigate the structure of the intersection of overlapping embeddings.

Lemma 4.6 *Let $h(V_1^*)$ and $h'(V_2^*)$ be two overlapping valid embeddings. Then, their intersection consists of a single polygonal region.*

Proof. We first show that the existence of more than one intersection region leads to a contradiction. This can easily be seen looking at Figure 4.9, which shows two overlapping embeddings with two intersection regions. Since $h(V_1^*)$ and $h'(V_2^*)$ are valid embeddings, there must exist witnesses w and w' such that $V_w^* = V_1^*$ and $V_{w'}^* = V_2^*$. In particular, there must exist lines of sight between the two witnesses and the points x and y, which lie in the two intersection regions. Note that the lines of sight between a witness and its corresponding skeleton polygon are not allowed to cross the boundary of the skeleton polygon. Thus, the shaded region in Figure 4.9 is encircled by four lines of sight and contains at least one map vertex, which contradicts Observation 3.3.

It remains to show that the intersection of the two embeddings is in fact a polygon. But this is clear in this case, since the intersection contains at least one point in its interior and therefore cannot consist of a single point or a single straight line segment. □

Lemma 4.7 *Let $h(V_1^*)$ and $h'(V_2^*)$ be two overlapping valid embeddings. Then, each edge of the intersection polygon is of one of the following types:*

Type 1 *A full edge in both embeddings, which is visible from both witnesses;*

Type 2 *A common artificial edge, that is, a pair of points, which simultaneously matches an artificial edge a in embedding $h(V^*)$ and also matches an artificial edge a' in embedding $h'(V^*)$, not necessarily with identical corresponding lines;*

Type 3 *An artificial edge in only one of the two embeddings, which lies in the interior of the other embedding (except for the common endpoints);*

Type 4 *A part of an artificial edge that intersects exactly one artificial edge of the other embedding;*

Type 5 *A part of an artificial edge that intersects exactly two artificial edges of the other embedding.*

Proof. Firstly, Lemma 4.6 guarantees that there exists exactly one intersection polygon. Recall that each point of the intersection polygon lies in both valid embeddings and thus must be visible to both witnesses. Particularly, each vertex of the intersection polygon is either a vertex in *both* embedded skeletons (if it is a map vertex) or in *neither* of them (if it is a new vertex created by the intersection of two artificial edges).

Then, we can basically distinguish between three cases for a given edge e of the intersection polygon:

Both of e's endpoints are map vertices If there is a map edge between the two endpoints of e, it follows from the arguments above that this edge is visible in both embeddings (i. e., e is a Type 1 edge). See, for example, the middle one of the five collinear horizontal edges in Figure 4.1 (a) on page 55.

If there is no map edge between the two vertices, the edge e must match an artificial edge in at least one of the two embeddings. Depending on whether there exist vertices "between" e's endpoints in the other embedding, the edge is either a Type 2 edge or a Type 3 edge. An example of a Type 2 situation are the two collinear horizontal artificial edges in Figure 4.1 (a) and a Type 3 situation is depicted by the artificial edges in Figure 4.1 (a), which connect the northern and the southern niches.

Exactly one of *e*'s endpoints is a map vertex An edge of the intersection polygon, where exactly one endpoint is a map vertex, can only arise in situations, where an artificial edge of one embedding is intersected by exactly one artificial edge of the other embedding. Thus, the situation must be similar to the example in Figure 4.2 (a) and e is a Type 4 edge.

None of *e*'s endpoints is a map vertex In this case, an artificial edge of the first embedding must be intersected by at least two artificial edges of the second embedding and from Lemma 4.5 it follows that it must be exactly two artificial edges, which intersect e. Thus, e is a Type 5 edge.

This completes the proof. □

Now we know that in the case of overlapping embeddings, their intersection region is a single polygonal region, consisting of only a few types of edges. In the sequel we further investigate the different types of intersections between the artificial edges of the overlapping embeddings and show that they can occur only in certain patterns and certain quantities (e. g., we show in Corollary 4.10 on page 75 that any intersection polygon contains at most four Type 5 edges). To this end, we first introduce a simple notation for classifying intersecting artificial edges, which allows us to visualize the types of intersections and which is helpful in the subsequent proofs, too.

Definition 4.2 (Left-Right- and Right-Left-Edge)
Let $h(V_1^*)$ and $h'(V_2^*)$ be two overlapping valid embeddings with witnesses w and w' and two intersecting artificial edges $h(a)$ and $h'(b)$. Let l be the left endpoint of $h(a)$, viewed from w, and r be the right endpoint.

*If the witness w and the left endpoint l of $h(a)$ lie on the same side of $h'(b)$, we say that $h(a)$ is a **Left-Right-edge** (or **LR-edge** for short) with respect to the intersection with $h'(b)$.*

*If w and the right endpoint r of $h(a)$ lie on the same side of $h'(b)$, we say that $h(a)$ is a **Right-Left-edge** (or **RL-edge** for short) with respect to the intersection with $h'(b)$.*

Figure 4.10 on the next page illustrates the previous definition and one should keep the following mnemonic in mind:

> An (intersecting) artificial edge $h(a)$ is called a *Left-Right*-edge if, sweeping a line parallel to the second intersecting edge $h'(b)$ and starting at the corresponding witness w, we first pass its *left* endpoint, then $h'(b)$, and finally its *right* endpoint.

The next lemma investigates the LR-/RL-property of an artificial edge that intersects *two* different artificial edges.

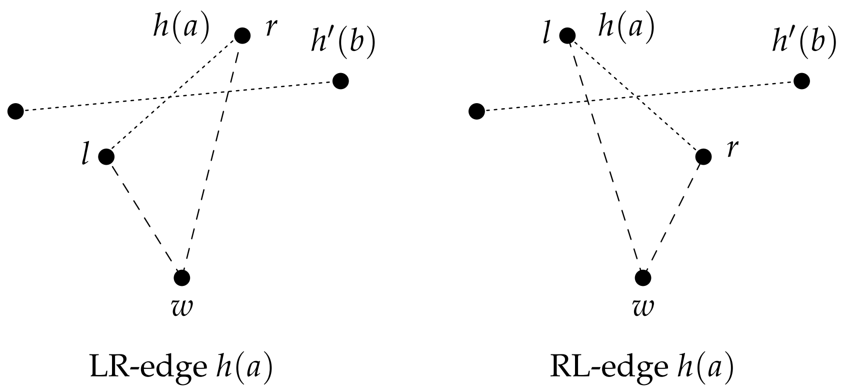


FIGURE 4.10: Illustration of Definition 4.2: LR- and RL-edges

Lemma 4.8 *Let $h(a)$ be an artificial edge of $h(V_1^*)$ that intersects two artificial edges $h'(b)$ and $h'(c)$ of $h'(V_2^*)$. If $h(a)$ is a LR-edge (RL-edge, respectively) with respect to $h'(b)$, it is also a LR-edge (RL-edge, respectively) with respect to $h'(c)$.*

Proof. Without loss of generality we assume that $h(a)$ is a LR-edge with respect to $h'(b)$ and a RL-edge with respect to $h'(c)$; the proof of the other case is identical. Figure 4.11 on the following page illustrates the resulting contradiction: In order to accomplish that $h(a)$ is a LR-edge with respect to $h'(b)$, the endpoints of $h'(b)$ must lie in the light-gray areas.

Analogously, the endpoints of $h'(c)$ have to lie in the dark-gray areas to guarantee that $h(a)$ is a RL-edge with respect to $h'(c)$. But then, the artificial edges $h'(b)$ and $h'(c)$ of *the same embedding h'* would intersect, which is impossible. □

Therefore, for two fixed embeddings $h(V_1^*)$ and $h'(V_2^*)$ the classification of an intersecting artificial edge as a LR-edge or RL-edge is unique, independent of the second involved edge.

In order to concentrate only on the topological aspects of intersections between artificial edges, we can simplify the visualization of such situations by virtually moving the witnesses of the embeddings collinear to the respective artificial edges. This way, the situation of a witness w, which sees a *LR-edge $h(a)$*, like in the left part of Figure 4.10, is represented by a straight line segment starting at w, which first passes the *left* endpoint of $h(a)$, then passes the intersection point of $h(a)$ with the second artificial edge, and finally reaches the *right* endpoint of $h(a)$, see Figure 4.12 on the next page.

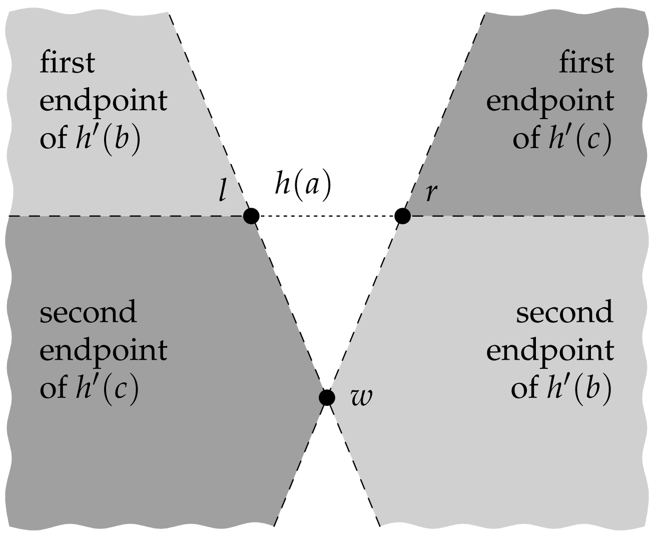

first endpoint of $h'(b)$
first endpoint of $h'(c)$
l
$h(a)$
r
second endpoint of $h'(c)$
second endpoint of $h'(b)$
w

FIGURE 4.11: An artificial edge $h(a)$, which is a LR-edge with respect to $h'(b)$ and a RL-edge with respect to $h'(c)$, is impossible

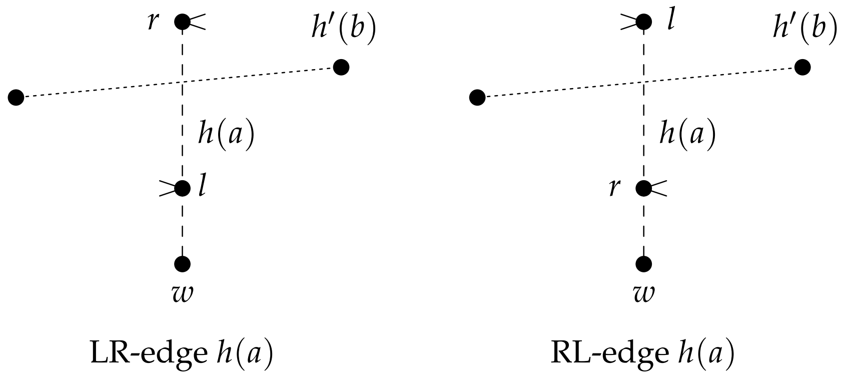


FIGURE 4.12: A simplified illustration of LR-edges and RL-edges

In other words, in the above visualization the view cone of w with respect to $h(a)$ is reduced to a single ray starting at w. Using this notation, we are able to depict more complex situations by concentrating only on the topological aspects of the scene, without getting confused by the view cones and lines of sight. Particularly note that no other vertex can lie in the view cone of a witness with respect to a corresponding artificial edge, since this would be a contradiction to the definition of "artificial edge". This means, such a view cone may safely be shrunk to a ray without changing the relative position of any map vertex with respect to the view cone.

We take advantage of the above considerations in the proof of the following lemma, which investigates the number of possible intersections between the different types of artificial edges.

Lemma 4.9 *Let $h(V_1^*)$ and $h'(V_2^*)$ be two overlapping valid embeddings. Then, the following holds:*

- *There exists at most one intersection between any two LR-edges.*
- *There exists at most one intersection between any two RL-edges.*
- *Let x be the number of intersections between a LR-edge of $h(V_1^*)$ and a RL-edge of $h'(V_2^*)$ and let y be the number of intersections between a RL-edge of $h(V_1^*)$ and a LR-edge of $h'(V_2^*)$.*

  Then, at most one of these numbers is greater than one, that is, $\min\{x, y\} \leq 1$.

Proof. In the following we investigate the different types of intersections between LR- and RL-edges and count them in a canonical way, using the visualization described above. To this end, let w and w' be the witnesses of the two overlapping embeddings and let $h(a)$ and $h'(b)$ be two intersecting artificial edges. Without loss of generality we can assume that the witness w' lies on the right side of the view cone of w with respect to $h(a)$ due to the following arguments: Firstly, it is clear that w' cannot lie *in* the view cone of w, because then one of the endpoints of $h(a)$ would be visible to w', when looking "through" the artificial edge $h'(b)$, which would contradict the fact that $h'(b)$ is an *artificial* edge. Secondly, if w' lies on the left side of the view cone, we can simply exchange the roles of $h(V_1^*)$ and $h'(V_2^*)$ and the roles of the numbers x and y by exploiting the symmetry of the situation. For example, the intersection between a LR-edge $h(a)$ and a RL-edge $h'(b)$ with w' lying on the left side of the view cone of w is identical to an intersection between a RL-edge $h(a)$ and a LR-edge $h'(b)$ with w' lying on the right side of the view cone of w.

Now assume that there exists another pair of intersecting artificial edges $h(c)$ and $h'(d)$ with p being their intersection point. Using our above visualization,

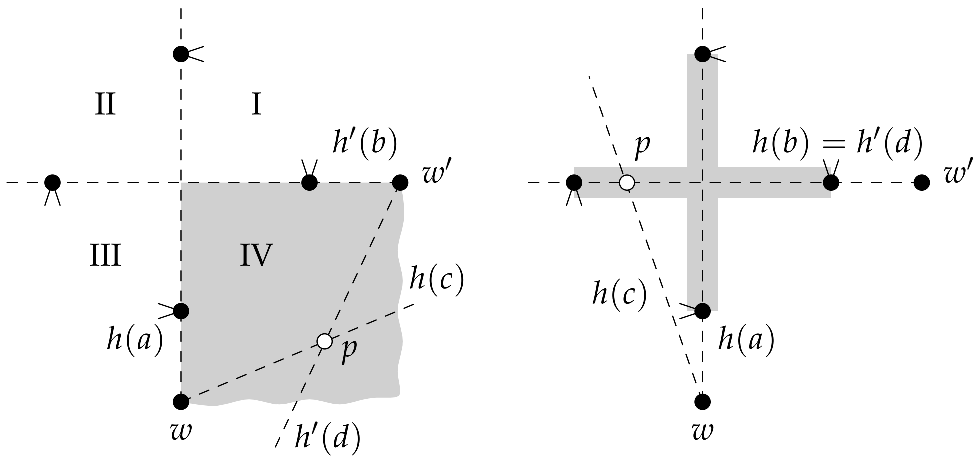


FIGURE 4.13: Possible placements for a second intersection point p

there are six topologically different possibilities, where the point p can lie (see Figure 4.13):

- If neither $a = c$ nor $b = d$, the intersection point p lies in one of the four quadrants formed by the (shrunken) view cones of w and w', as depicted in the left part of Figure 4.13.
- In the other case, where either $a = c$ or $b = d$, the point p lies in the view cone of one of the two witnesses, that is, using our visualization from above, p lies on the straight line segment that represents the respective artificial edge (see the right part of Figure 4.13).

Note that also for the second intersection point p, lines of sight to the two witnesses w and w' must exist. In the remaining part of the proof these lines of sight are the crucial instrument to decide whether a second intersection is possible or not, and to decide which type of intersection may occur. For example, in the right part of Figure 4.13 the placement of the second intersection point p is not possible, since one of the endpoints of $h(a)$ is encircled by lines of sight, which would imply a hole in the map polygon and therefore contradict Observation 3.3 on page 27. Similarly, in the left part of Figure 4.13 the artificial edge $h(c)$ must be a RL-edge with respect to $h'(d)$, since otherwise the endpoint of $h(c)$, which is placed "between" w and p, would lie to the left of the respective view cone, which would also result in a hole.

In the following we investigate the four different types of intersections between LR- and RL-edges (LR/LR, LR/RL, RL/LR, and RL/RL) and decide for each one of the six possibilities of placing a second intersection point, whether

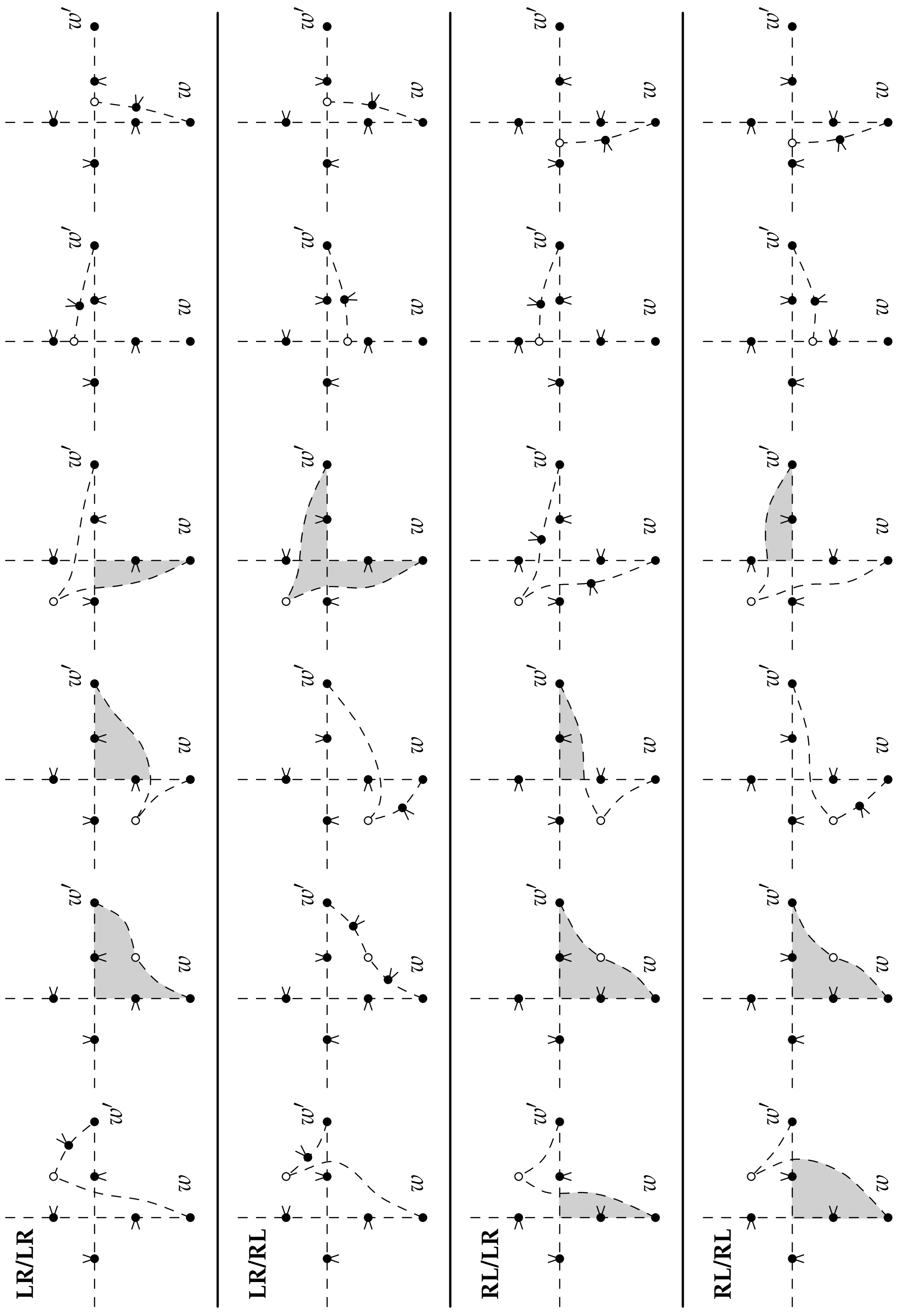


FIGURE 4.14: The 24 cases of placing a second intersection point

this intersection is possible and which type of intersection would result. To this end, see Figure 4.14 on the preceding page, which depicts all 24 cases that can occur if a second intersection point has to be placed for two artificial edges. The four rows of the figure represent the four different types of intersection. For each row we distinguish between the six topologically different positions of the second intersection point. For each of the 24 cases the second intersection point p (marked with a circle »○«) is placed at its respective position in the figure and connected with lines of sight to the witnesses w and w'. If the placement of p implies a hole in the map polygon, the hole is drawn in gray. In the remaining cases of legal placements of p, we have additionally introduced the endpoints of the second pair of intersecting artificial edges into the figure, whose positions are clearly determined by the topological situation.

For each type of intersection between $h(a)$ and $h'(b)$ we can then determine the possible types of intersections for a potential second pair $h(c)$ and $h'(d)$ of intersecting artificial edges (with intersection point p):

LR/LR-intersection From the first row of Figure 4.14 follows that if neither $a = c$ nor $b = d$, the point p can only be placed in the first of the four quadrants, since a second intersection point in one of the three reminding quadrants would cause at least one map vertex to be encircled by lines of sight and therefore be a contradiction. If p lies in the first quadrant, we additionally conclude that the artificial edge $h'(d)$ must be a RL-edge, since otherwise its left endpoint would cause a hole in the map polygon.

With the same arguments we conclude (using the two rightmost columns of the first row of Figure 4.14) that if $a = c$ or $b = d$, either the artificial edge $h(c)$ or the artificial edge $h'(d)$ must be a RL-edge.

Summarizing, we see that for a LR/LR-intersection a potential second intersection point either causes a RL/$*$ or a $*$/RL-intersection, but never a LR/LR-intersection. Therefore, the number of LR/LR-intersections is at most one.

LR/RL-intersection A similar argumentation as before using the second row of Figure 4.14 gives the following results: Placing p in the second quadrant leads to a RL/LR-intersection and placing p onto one of the two artificial edges results either in a LR/LR-intersection (if $a = c$) or in a RL/RL-intersection (if $b = d$). The two remaining cases of placing the point p into the first quadrant (or third quadrant, respectively) are symmetric and result in a $*$/RL-intersection (or a LR/$*$-intersection, respectively).

RL/LR-intersection Analogously, it follows from the third row of Figure 4.14 that a second intersection is either of type LR/RL (if p is placed in the forth quadrant), of type RL/RL (if $a = c$), or of type LR/LR (if $b = d$). That is, the number of RL/LR-intersections is also at most one as in the first case.

RL/RL-intersection Using the bottom row of Figure 4.14 and exactly the same argumentation as in the LR/LR-case, it follows that for a RL/RL-intersection a potential second intersection point either causes a LR/$*$ or a $*$/LR-intersection, but never a RL/RL-intersection. Thus, the number of RL/RL-intersections again is at most one.

Summing all up, we obtain that the number of LR/LR-, RL/LR-, and RL/RL-intersections is at most one, provided that the witness w' lies on the right side of the view cone of w. Finally, the symmetry argument from the beginning of the proof shows that in the other case the number of LR/LR-, LR/RL-, and RL/RL-intersections is at most one, which completes the proof. □

In this section we studied the structure of general overlapping embeddings and proved some properties of their intersection region, namely that it is a single polygonal region consisting of only a few types of edges. We further investigated the different types of intersecting skeleton edges and by using a detailed distinction of cases we proved that certain patterns may occur only a constant number of times.

This allows us to prove the following claim about the maximum number of Type 5 edges:

Corollary 4.10 *The intersection polygon of two overlapping valid embeddings contains at most four Type 5 edges (cf. Lemma 4.7 on page 67), that is, edges that are part of an artificial edge in one of the two embeddings, which intersects exactly two artificial edges of the other embedding.*

Proof. In order to create a Type 5 edge, a single artificial edge of the first embedding must be intersected by exactly two artificial edges of the second embedding. From Lemma 4.5 it follows that these two edges must be a LR-edge and a RL-edge. Thus, depending on the type of the first artificial edge, either a LR/LR-intersection or a RL/RL-intersection is created. Then, it directly follows from the preceding Lemma 4.9 that at most two such intersection situations can occur. The maximal number of four Type 5 edges is then achieved, for example, by two intersecting pairs of LR/RL edges, as sketched in Figure 4.15 on the following page with witnesses w and w'. (The Type 5 edges are underlayed gray in the figure.) □

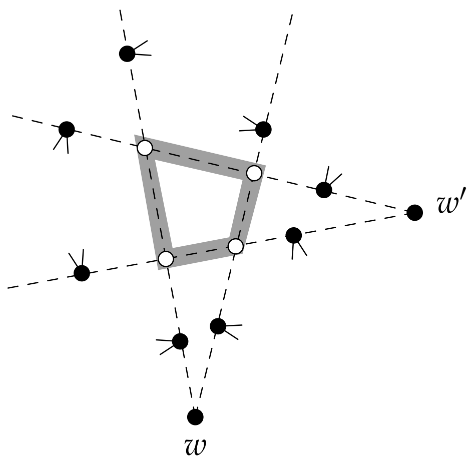


FIGURE 4.15: Two pairs of intersecting LR/RL edges, which create the maximum number of four Type 5 edges

Figure 4.16 on the next page demonstrates that such a map actually exists. It shows two overlapping valid embeddings $h(V^*)$ and $h'(V^*)$ (even of *the same* skeleton V^*), drawn in light-gray, together with their witnesses w and w'. The two embeddings overlap in a way such that their intersection polygon (drawn in dark-gray) consists of four Type 5 edges. In particular, this example illustrates that the intersection polygon need not contain any map vertex.

In the following sections we consider overlapping embeddings $h(V^*)$ and $h'(V^*)$ of *the same* skeleton V^*, in contrast to the previous section, where our main focus of investigation lay on overlappings of arbitrary skeletons.

4.5 Overlapping Embeddings of Identical Skeletons

The aim of the succeeding lemmas is to guarantee that (simply speaking) also in the setting of Theorem 4.2 the four artificial edges a, $a' = a + t$, b, and $b' = b + t$, which we have used for the proof of Theorem 4.3 on page 57, always exist. Recall that we denoted the translation vector that maps the origin of $h(V^*)$ onto the origin of $h'(V^*)$ by t and that the key ingredient of this proof was a common kernel point x, from which we concluded that an artificial edge a of h must exist in direction t and an artificial edge b' of h' must exist in direction $-t$.

That is, in order to guarantee the above artificial edges also in the setting of Theorem 4.2, we have to prove the existence of a point $x \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ such that

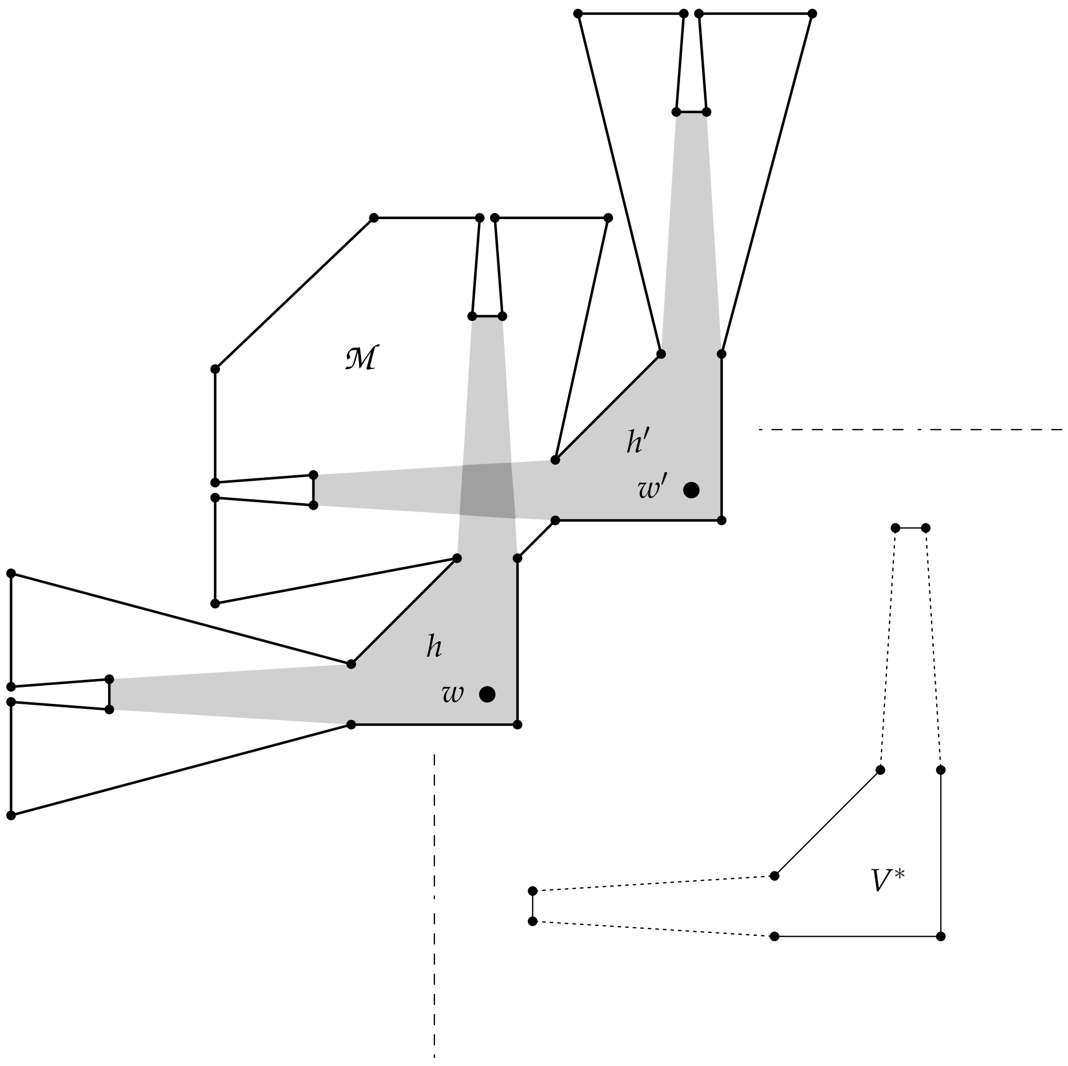


FIGURE 4.16: A map with two overlapping embeddings h and h', which intersect in such a way as sketched in Figure 4.15

- x sees an artificial edge of $h(V^*)$, when looking in direction t, and
- x sees an artificial edge of $h'(V^*)$, when looking in direction $-t$.

We do this step by step in the following way:

Claim 1 We first show that a point x exists that sees a (not necessarily artificial) edge of $h(V^*)$ in direction t.

Claim 2 Then, we conclude that this also holds for an *artificial* edge of $h(V^*)$.

Claim 3 We further deduce that besides the artificial edge of $h(V^*)$ in direction t, x must also see an edge of $h'(V^*)$ in direction $-t$.

Claim 4 Finally, we show that there must exist a point x that sees artificial edges in *both* directions.

4.5.1 A Property of Intersecting Star-Shaped Polygons

At first, we consider a special case of intersecting polygons, namely the intersection of a star-shaped polygon P with a translated copy $P + t$ of itself. We show that (roughly speaking) each component of the intersection contains at least one edge of the first polygon P, which can be seen when looking in the translation direction t. Using this result, Claim 1 from above is a straightforward consequence.

Lemma 4.11 *Let P be a star-shaped polygon and let $P' = P + t$ be a translated copy of P (with $t \neq 0$). Each connected component C of the intersection $P \cap P'$ that has a non-empty interior contains an edge e such that*

- *e belongs to P;*
- *e can be seen from inside C, when looking in direction t. That is, there exists a point $x \in C^\circ$ such that the edge e is the first edge that is hit by the ray that starts in x and emanates in direction t.*

Proof. In the following we assume that no such edge exists and show that this leads to a contradiction. To this end, we assume w. l. o. g. that the translation vector t is horizontal and directed from left to right. For a component C of the intersection $P \cap P'$, let p be a rightmost intersection point (of C) between an edge f of P and an edge f' of P'. Note that such a point must always exist, since otherwise one polygon would totally contain the other one, which is impossible in our setting. For the following recall that the polygons are traversed counterclockwise (cf. Section 2.2).

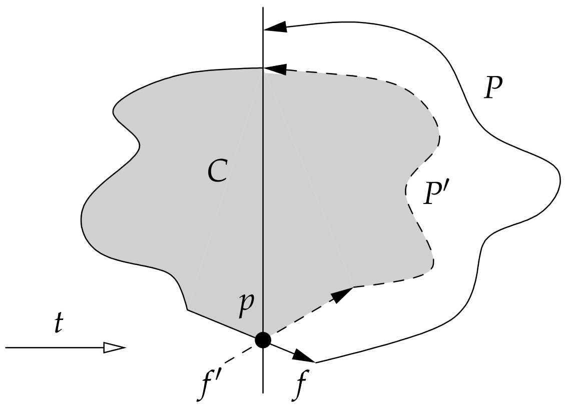


FIGURE 4.17: The intersection of a star-shaped polygon and a translated copy of itself

Without loss of generality we assume that on our traversal of C we first pass the edge f, then the vertex p, and finally the edge f' as depicted in Figure 4.17. (Edges of P are drawn solid, edges of P' dashed.) If this is not the case, we could simply mirror the whole scene at a horizontal line, reverse the direction of the polygons, and get a situation that fits our assumption. Note that from our main assumption ("no such edge e exists") it follows for the angle between the translation vector t and the edge f that $\sphericalangle(t, f) \geq \pi$, since otherwise a part of f could be seen from inside C, when looking in direction t.

Due to the construction of p, the two polygons cannot have additional intersection points to the right of p. Furthermore, some part of P', namely $f' \cap P$, must lie to the left of f. Now recall that P' is a (horizontally) translated copy of P. This means that a copy of the polygonal chain of P to the right of p (translated by t) must also occur in the polygon P'. Then, there are basically two possibilities how the two parts of P' are connected by polygonal chains, depending on the angle of rotation $\alpha_{f+t,f'}$ of the edges $f + t$ and f' in the polygon P'. One possibility (with $\alpha_{f+t,f'} < 2\pi$) is depicted in Figure 4.18 on the following page, the other one (with $\alpha_{f+t,f'} > 2\pi$) is illustrated by Figure 4.19.

In the following we show that the kernel of the so-constructed polygon P' is empty, which contradicts the assumption of a star-shaped polygon P. To this end, we distinguish between the two possibilities mentioned above and handle them separately:

$\alpha_{f+t,f'} < 2\pi$ Consider the horizontal line l through the point p (drawn gray in Figure 4.18). From $\sphericalangle(t, f) \geq \pi$ follows that the edge f (and also $f + t$) must be directed from somewhere above l to a point below l. Consequently, the polygonal chain P_1 that connects the left part of P' with $f + t$

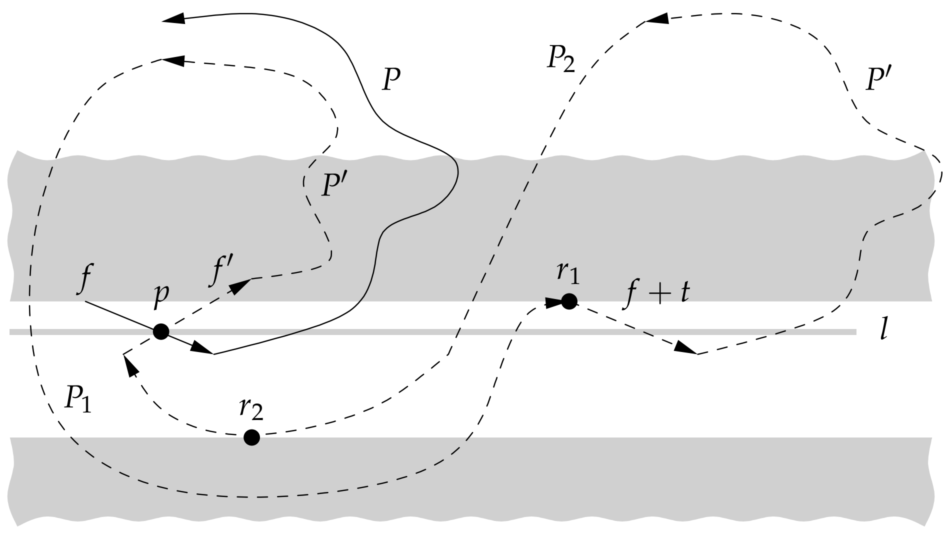


FIGURE 4.18: One possibility of connecting the two polygonal chains of P'

must cross l from below, resulting in at least one reflex vertex r_1 above l. Thus, the kernel of P' must lie above l (in the upper gray shaded area of the figure). On the other hand, it follows with a similar argument that there must also exist a reflex vertex r_2 below l on the polygonal chain P_2 that connects the right part of P' with f'. Therefore, $\ker P'$ must also lie below l (in the lower gray shaded area of the figure), which together results in an empty kernel of P'. This is a contradiction to our assumption of a star-shaped polygon P.

$\alpha_{f+t,f'} > 2\pi$ In order to complete the proof we also have to consider the second possibility of connecting the two parts of P', where the polygonal chain P_2 that connects the right part of P' with f' takes the other way around C. We observe that in this case the two polygons P and P' must have another intersection point p' (see Figure 4.19 on the next page), and with similar arguments as above (using p' instead of p) it follows that the kernel of P' is empty. □

4.5.2 The Existence of an Artificial Edge in One Direction

In this section we prove Claim 2 from page 78 using the preceding Lemma 4.11.

Lemma 4.12 *Under the assumptions of Theorem 4.2, let $t = h'(v_o) - h(v_o)$ be the translation vector that maps the embedded origin $h(v_o)$ onto the embedded origin $h'(v_o)$. Then, there always exists an edge of the intersection polygon that*

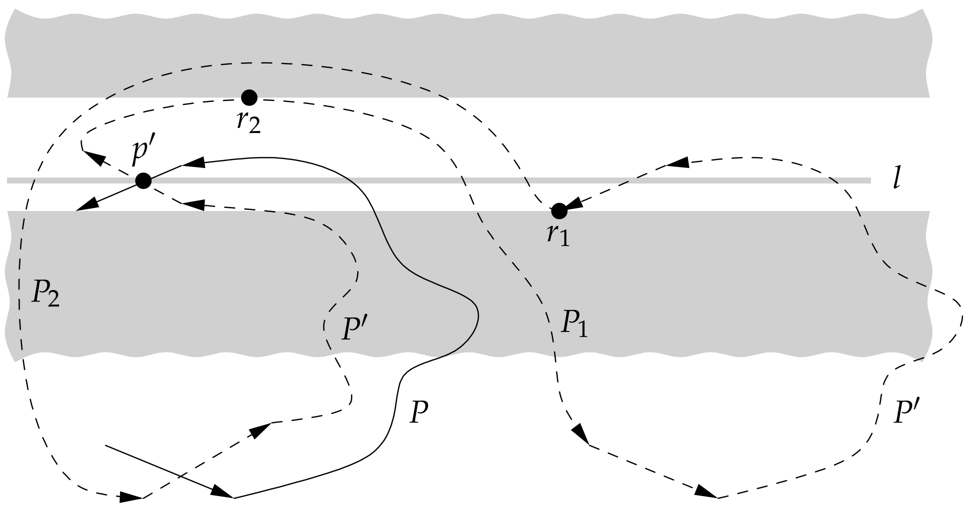


FIGURE 4.19: The other possibility of connecting the two polygonal chains

- *belongs to an artificial edge of $h(V^*)$ and*
- *can be seen from inside the intersection polygon, when looking in direction t. In other words, there exists a point $x \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ such that the respective edge is the first edge being hit by the ray that starts in x and emanates in direction t.*

Proof. Without loss of generality we assume that the translation vector $t \neq 0$, which maps $h(v_o)$ onto $h'(v_o)$, is horizontal and directed from left to right. That is, $h(V^*)$ is the left one and $h'(V^*)$ the right one of the two embeddings of V^*. From Lemma 4.6 follows that the intersection of the two embedded skeleton polygons consists of a single polygonal region I. Since a skeleton polygon always is star-shaped, we conclude from Lemma 4.11 on page 78 that there exists at least one edge of I that belongs to an edge e of $h(V^*)$ and fulfills all of the above requirements, except that it is not guaranteed that e is an *artificial* edge of $h(V^*)$. In the following we assume that such an edge, which belongs to an *artificial* edge, does not exist and show that this leads to a contradiction.

Firstly, it follows from our assumption and Lemma 4.7 that every edge f of I, which belongs to $h(V^*)$, either cannot be seen from inside I (i. e., $\sphericalangle(t, f) \geq \pi$) or is a common full edge of $h(V^*)$ and $h'(V^*)$. Furthermore, Lemma 4.11 guarantees that there must exist at least one such common full edge, which we denote by e. Now let l be a lowermost and u be an uppermost intersection point between $h(V^*)$ and $h'(V^*)$. Then, the edge e must be contained in a polygonal chain of $h(V^\diamond)$ between l and u, as well as in a similar chain of $h'(V^\diamond)$. Since

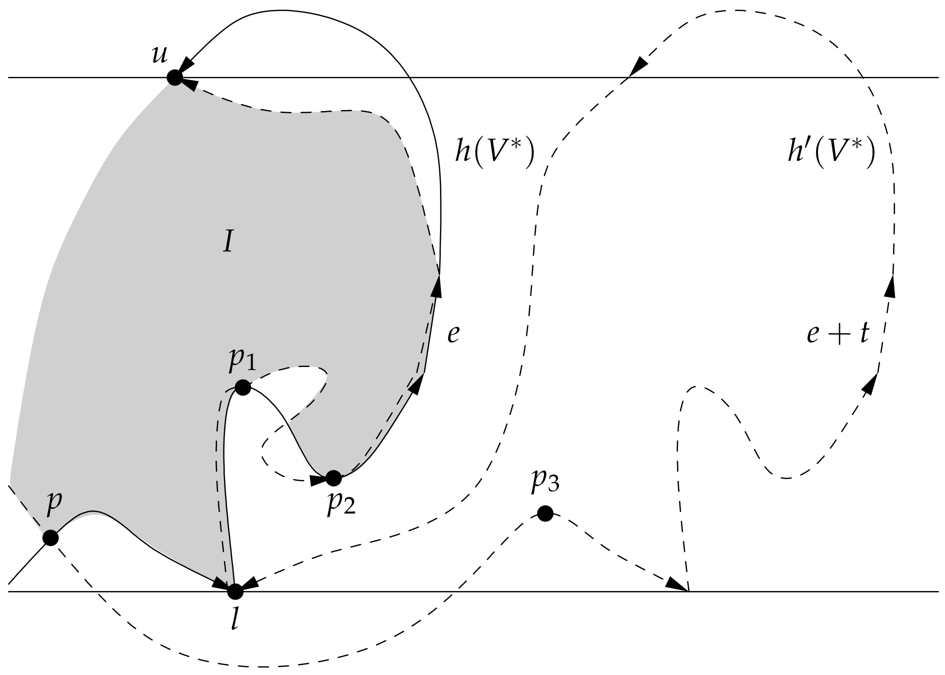


FIGURE 4.20: The first possibility of connecting the polygonal chains of $h'(V^*)$

the intersection of the two skeletons is a single polygonal region, both chains must have the same direction (either from l to u or vice versa), and w. l. o. g. we assume that both chains are directed from l to u. The resulting situation is depicted in Figure 4.20. (Edges of $h(V^*)$ are drawn solid, edges of $h'(V^*)$ dashed.) Note that the two polygonal chains from l to u need not, of course, be identical. Edges of $h'(V^\diamond)$ may lie strictly to the left of $h(V^\diamond)$ like those in the upper part of the figure (between e and u), or (as in the middle part of the figure between p_1 and p_2) even cross $h(V^\diamond)$, as long as it is guaranteed that no artificial edge of $h(V^\diamond)$, which belongs to I, can be seen from the left.

The remaining part of the proof is very similar to the proof of Lemma 4.11 on page 78 in the sense that we use two *identical* polygons, namely $h(V^\diamond)$ and $h'(V^\diamond)$, in order to show that $V^\diamond$ must have an empty kernel, which clearly is a contradiction. To this end, recall that a translated copy of the polygonal chain of $h(V^*)$, which connects l and u, must also occur in $h'(V^*)$. Then, like in the proof of Lemma 4.11 there are several possibilities to build the polygonal chain of $h'(V^*)$: By construction it is clear that l, e, and u must be traversed in this order and it remains to investigate the basically two different possibilities of inserting the right chain, which contains $e+t$, into this order:

$e + t$ is inserted between u and l As in the proof of Lemma 4.11 we have to distinguish between the only two reasonable possibilities of connecting the right part of $h'(V^*)$ to the left part, either with a right turn around I (i. e., $\alpha_{e+t,e} = 0$) or with a left turn (i. e., $\alpha_{e+t,e} = 2\pi$). Since for both possibilities we can use almost identical arguments, we concentrate w. l. o. g. on the situation shown in Figure 4.20, where the right part of $h'(V^*)$ is connected by a right turn to the left part of $h'(V^*)$.

Then, $h(V^*)$ and $h'(V^*)$ must intersect in another point p. From our assumption that l is a lowermost intersection point of the two embedded skeletons follows that p must lie above the horizontal line through l. Thus, $h'(V^*)$ must contain two reflex vertices (p_3 and l in the figure), from which an empty kernel follows.

$e + t$ is inserted between l and u If this is the case, the polygonal chain of $h'(V^\diamond)$ between l and u must cross the chain of $h(V^\diamond)$ either between l and e or between e and u. Without loss of generality we assume that $h'(V^\diamond)$ crosses $h(V^\diamond)$ between l and e as illustrated in Figure 4.21 on the next page. Due to our main assumption this is only possible if $h(V^\diamond)$ contains an edge f with $\sphericalangle(t, f) \geq \pi$, which directly implies that $h'(V^*)$ must contain two reflex vertices r_1 and r_2 (see the figure), from which an empty kernel follows.

Since in all cases $V^\diamond$ must have an empty kernel (that is, $V^\diamond$ cannot be star-shaped), the assumption that no artificial edge of $h(V^*)$ can be seen from inside the intersection leads to a contradiction. □

Apparently, a straightforward consequence of the previous lemma is that in the setting of Theorem 4.2 we can guarantee the existence of a ray r, which intersects artificial edges $\overline{a_1\, a_2}$ and $\overline{a'_1\, a'_2}$ in direction t, and the existence of a ray r', which intersects artificial edges $\overline{b_1\, b_2}$ and $\overline{b'_1\, b'_2}$ in direction $-t$.

4.5.3 The Existence of Artificial Edges in Both Directions

The following two lemmas show that there also exists a straight line that fulfills *both* properties at the same time. According to our strategy described above, we first prove Claim 3.

Lemma 4.13 *Under the assumptions of Theorem 4.2, let $t = h'(v_o) - h(v_o)$ be the translation vector that maps the embedded origin $h(v_o)$ onto the embedded origin $h'(v_o)$. Then, there always exists a point $x \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ such that*

- *x sees an artificial edge of $h(V^*)$, when looking in direction t, and*

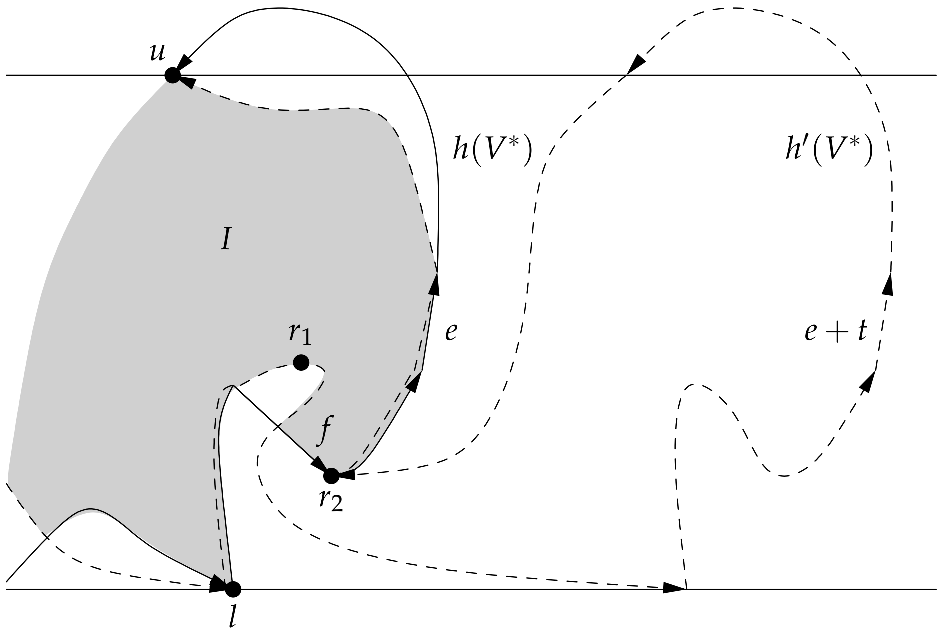


FIGURE 4.21: The second possibility of connecting the polygonal chains of $h'(V^*)$

- *x sees an edge of $h'(V^*)$, when looking in direction $-t$.*

Proof. Without loss of generality we assume that the translation vector $t \neq 0$, which maps $h(v_o)$ onto $h'(v_o)$, is horizontal and directed from left to right. We prove the proposition by assuming that no such point x exists and showing that this leads to a contradiction using an induction argument. The basic idea is to repeatedly remove the concavities of V^* (which V^* must have if such a point x does not exist), but preserving the property that the two embeddings of V^* overlap.

From Lemma 4.12 follows that a point $x \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ exists that sees an edge e, which belongs to an artificial edge of $h(V^*)$, when looking in direction t. If we assume that x does not see an edge of $h'(V^*)$, when looking in the opposite direction $-t$, x must see an edge f, which belongs to an edge of $h(V^*)$. Furthermore, it also follows from our assumption that the edge $f+t$ of $h'(V^*)$ must lie to the right of e. Then, we get the situation depicted in Figure 4.22 on the facing page: The edges e and f of $h(V^*)$ are connected by two polygonal chains c_u (the upper one) and c_l (the lower one). Since e and f are part of the intersection of the two embeddings, they must be surrounded by exactly one of the respective translated chains of $h'(V^*)$, w. l. o. g. by the lower chain $c_l + t$, as illustrated in the figure.

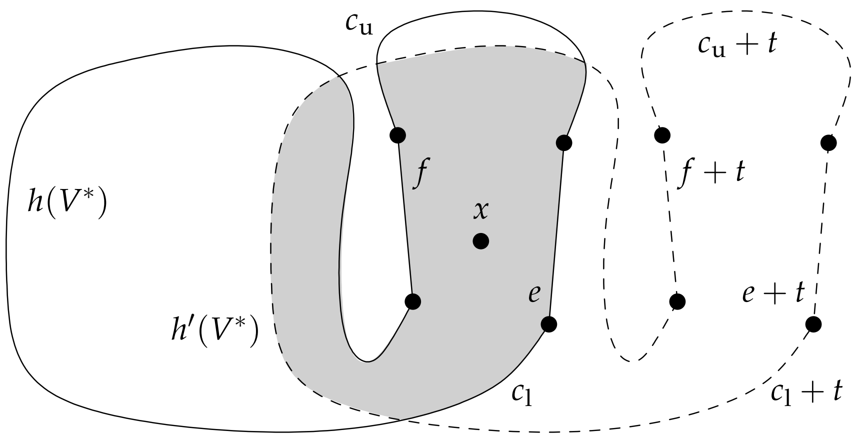


FIGURE 4.22: The skeleton polygon V^* must have a concavity, . . .

Since the polygonal chain of $h'(V^*)$ that consists of $e + t$, $c_\mathrm{u} + t$, and $f + t$ cannot intersect the skeleton polygon $h(V^\diamond)$ at all (because this would be a contradiction to Lemma 4.6), we may delete this chain from V^* and insert a full edge d between the two lower endpoints of e and f. This way, we obtain a modified skeleton V^*_m, which consists of a strictly smaller number of edges than V^*, as depicted in Figure 4.23. It is clear from the above reasoning that the two embeddings $h(V^*_\mathrm{m})$ and $h'(V^*_\mathrm{m})$ still intersect in such a way that we can apply the same argumentation another time. Since the skeletons consist of only finitely many edges, this leads to a contradiction, which shows that our original assumption must be wrong. □

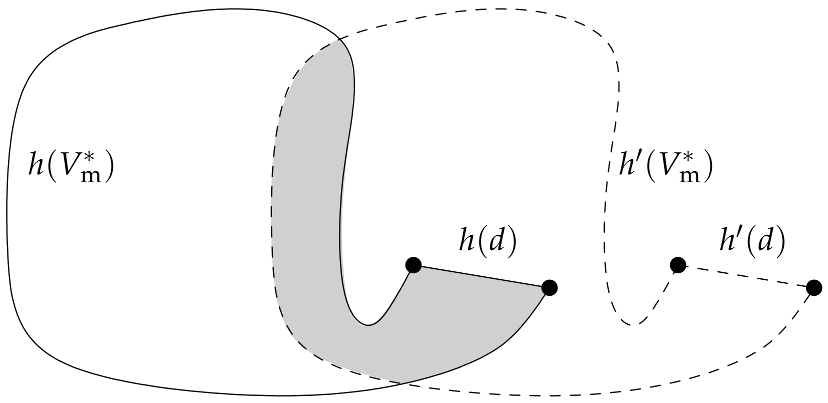


FIGURE 4.23: . . . which can be removed such that the embeddings still intersect

Using the results from the previous considerations, we are now able to prove Claim 4 from page 78, that is, the existence of a point x that sees artificial edges in *both* directions.

Lemma 4.14 *Under the assumptions of Theorem 4.2, let $t = h'(v_o) - h(v_o)$ be the translation vector that maps the embedded origin $h(v_o)$ onto the embedded origin $h'(v_o)$. Then, there always exists a point $x \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ such that*

- *x sees an artificial edge of $h(V^*)$, when looking in direction t, and*
- *x sees an artificial edge of $h'(V^*)$, when looking in direction $-t$.*

Proof. Without loss of generality we assume that the translation vector $t \neq 0$, which maps $h(v_o)$ onto $h'(v_o)$, is horizontal and directed from left to right. The preceding Lemma 4.13 already guarantees a point $x_1 \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ that sees an artificial edge e_1 of $h(V^*)$, when looking in direction t, and an edge f_1 of $h'(V^*)$, when looking in the opposite direction $-t$. Exploiting the symmetry of the setting, also a point $x_2 \in (h(V^\diamond) \cap h'(V^\diamond))^\circ$ exists that sees an artificial edge e_2 of $h'(V^*)$, when looking in direction $-t$, and an edge f_2 of $h(V^*)$, when looking in the opposite direction t. It remains to show that there is also a point x that simultaneously has both properties, that is, x sees artificial edges in *both* directions.

As in the previous proofs, we assume in the following that no such point exists and show that this leads to a contradiction. Using a similar argumentation as in the proof of Lemma 4.13, we may assume w. l. o. g. that the intersection polygon is monotone with respect to a line perpendicular to t. Otherwise, the skeleton V^* must have a concavity, which could safely be removed, whereby ensuring that the two embeddings of V^* still overlap. In particular, exploiting the monotony of the intersection polygon we may assume that the two artificial edges e_1 and e_2 (as well as the edges f_1 and f_2) are separated by a horizontal line, since otherwise the intersection polygon would not be monotone as assumed above. Thus, we get a situation as shown in Figure 4.24 on the facing page. (Edges of $h(V^*)$ are drawn solid, edges of $h'(V^*)$ dashed.)

From our main assumption follows that the edges f_1 and f_2 must be full edges of $h(V^*)$ and $h'(V^*)$ and therefore belong to both embeddings (using Lemma 4.7). Thus, the embedding $h(V^*)$ contains the artificial edges e_1 and $e_2 - t$, as well as the full edges f_1, $f_1 - t$, f_2, and $f_2 - t$. In the following we investigate the different possibilities of connecting these six edges in order to obtain the skeleton V^*. To this end, recall that neither an edge of $h(V^\diamond)$ nor an edge of $h'(V^\diamond)$ is allowed to cross the intersection polygon (drawn gray in the figure) and that additionally the intersection polygon must be contained in both embeddings.

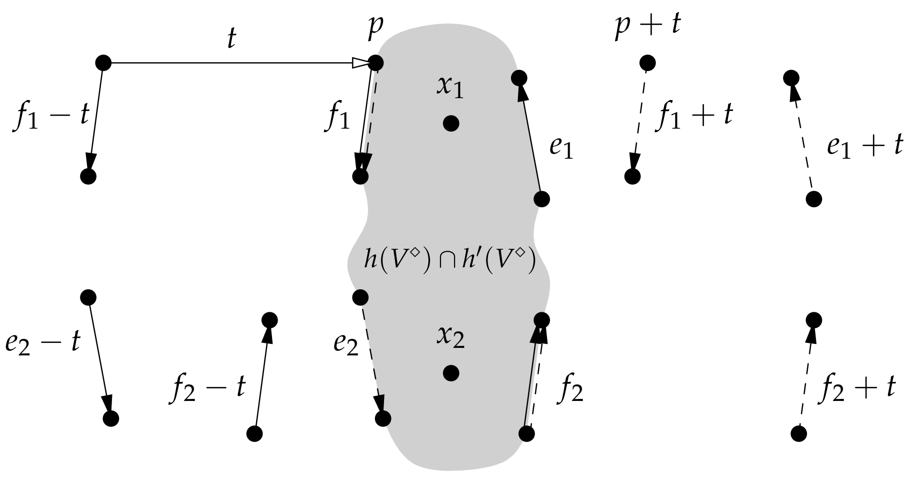


FIGURE 4.24: The relevant edges of $h(V^*)$ and $h'(V^*)$

First of all, we claim that in embedding $h(V^*)$ the only possible vertex, to which the upper endpoint of f_2 may be connected, is the lower endpoint of e_1, since for each of the other possibilities either the intersection polygon is crossed by an edge (a connection to f_1 or $f_1 - t$ implies that e_1 is isolated), or the skeleton polygon is caused to have an empty kernel (a connection to $e_2 - t$ or $f_2 - t$ must be accomplished by a right turn), which both results in a contradiction. Now we distinguish between the four cases of connecting one of the remaining edges to the upper endpoint p of f_1 in embedding $h(V^*)$ and show that we always get a contradiction:

»Edge $f_1 - t$ is connected to p« The only reasonable way of connecting the lower endpoint of $f_1 - t$ to p is a direct connection with $\alpha_{f_1 - t, f_1} = 0$, since the other possibilities imply an empty kernel of the skeleton polygon. However, this connection immediately causes a contradiction, because the polygonal chain from the lower endpoint of $f_1 - t$ to p in embedding $h(V^*)$ translates to a polygonal chain from the lower endpoint of f_1 to $p + t$ in embedding $h'(V^*)$, which crosses the intersection polygon (cf. Figure 4.24).

»Edge $e_2 - t$ is connected to p« If the lower endpoint of $e_2 - t$ is connected to the upper endpoint p of f_1 in embedding $h(V^*)$, it follows for the angle of rotation $|\alpha_{e_2 - t, f_1}| < \pi$, since a right turn would imply an empty kernel and a left turn around the intersection polygon would isolate edge e_1. Thus, the lower endpoint of $e_2 - t$ must be connected to p as depicted in Figure 4.25 on the next page. Note that $f_2 - t$ must lie to the left of

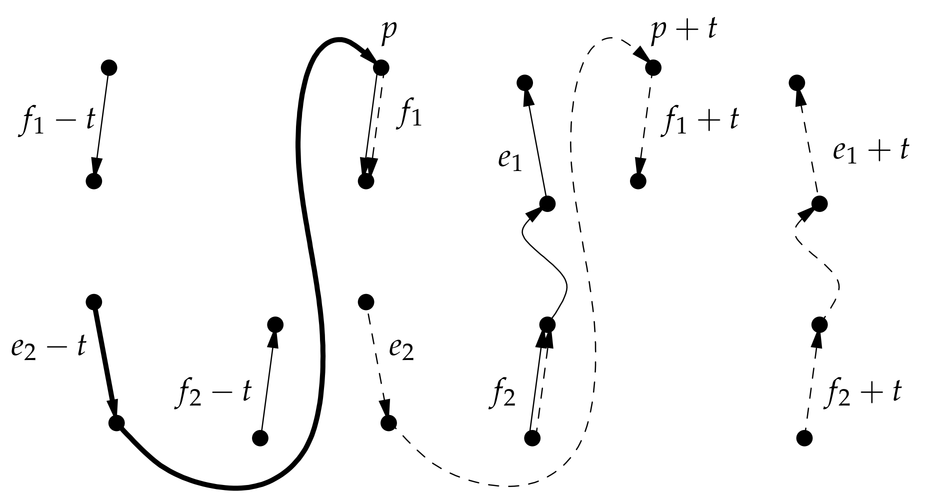


FIGURE 4.25: The case »Edge $e_2 - t$ is connected to p« in the proof of Lemma 4.14

the polygonal chain, since otherwise the respective translated chain between e_2 and $p+t$ in embedding $h'(V^*)$ would cross the intersection polygon. Then, it is easy to see that it is not possible to connect $f_2 - t$ to the remaining skeleton edges without creating two reflex vertices such that the resulting kernel of $h(V^\diamond)$ is empty.

»**Edge $f_2 - t$ is connected to p**« From this connection follows that the upper endpoints of $f_1 - t$ and e_1 must be connected in embedding $h(V^*)$ by a polygonal chain (otherwise either e_1 or $e_2 - t$ would be isolated) and we get the situation depicted in Figure 4.26 on the facing page, where that chain and its translated copy in embedding $h'(V^*)$ introduce an intersection point y. Then, we observe that the edge e of embedding $h'(V^*)$ between p and y must be an artificial edge and from our main assumption follows that the corresponding edge f in direction t must be a full edge in both embeddings. Furthermore, the lower endpoint q of f must lie *below* p due to our assumption (since e_1 is an artificial edge in h). Consequently, the skeleton polygon $h(V^\diamond)$ must have an empty kernel due to its two (reflex) vertices p and $q - t$.

»**Edge e_1 is connected to p**« Here we additionally have to distinguish between the following three subcases of connecting one of the remaining edges in embedding $h(V^*)$ to the lower endpoint p' of f_2:

»**Edge $f_1 - t$ is connected to p'**« Connecting the lower endpoint of $f_1 - t$ to p' in embedding $h(V^*)$ would result in a polygonal chain between f_1 and $f_2 + t$ in embedding $h'(V^*)$. Thus, the edge e_2 would be isolated in embedding $h'(V^*)$ resulting in an empty kernel of $h'(V^*)$.

»**Edge $e_2 - t$ is connected to p'**« In this case, we obtain the situation of the preceding case »Edge $f_2 - t$ is connected to p« (cf. Figure 4.26) rotated by π.

»**Edge $f_2 - t$ is connected to p'**« If the upper endpoint of $f_2 - t$ is connected to p', we get the situation depicted in Figure 4.27 on the next page.

Now recall that the edge f_2 is a full edge in embedding $h(V^*)$, whereas the edge e_1, to which f_2 is connected, is an artificial edge. Therefore, a vertex q on the polygonal chain from f_2 to e_1 must exist, where the chain of full edges changes to artificial edges. Analogously, a vertex q' exists in embedding $h'(V^*)$, where the chain of full edges between f_1 and e_2 changes. From our our main assumption then follows that q' must lie *below* q, since otherwise a point x would exist that sees artificial edges in both directions. Consequently, the skeleton polygon $h(V^\diamond)$ must have an empty kernel due to its two (reflex) vertices q' and $q - t$.

This completes the proof, since we have shown that the assumption that no point x exists that has the described properties leads to a contradiction in all

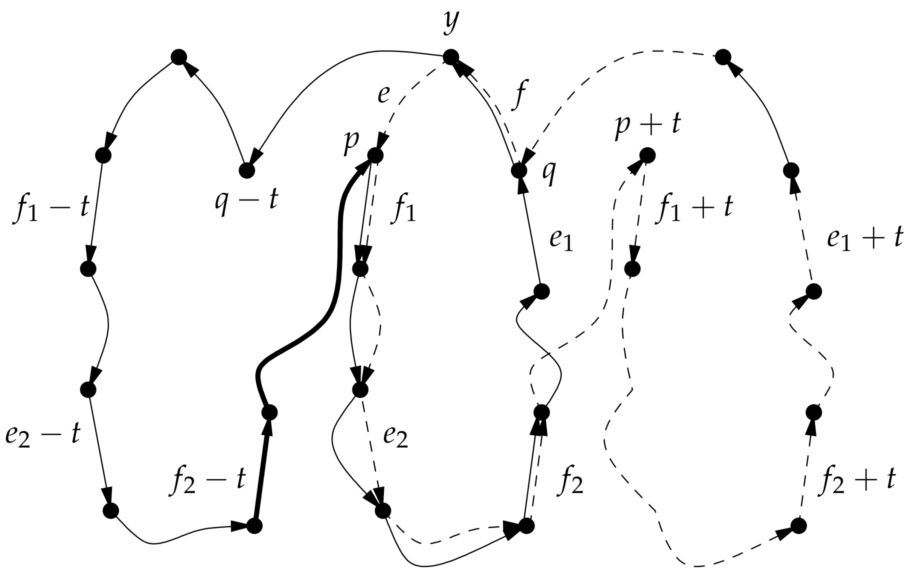


FIGURE 4.26: The case »Edge $f_2 - t$ is connected to p« in the proof of Lemma 4.14

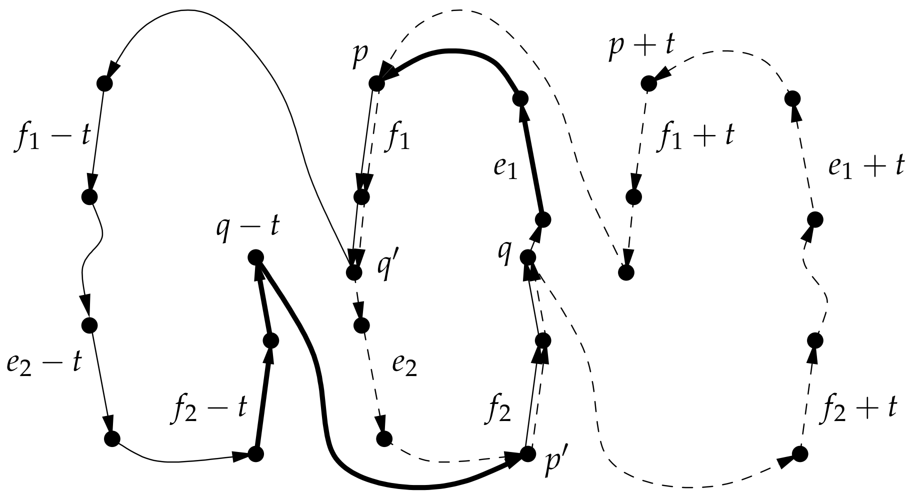


FIGURE 4.27: The case »Edge e_1 is connected to p« in the proof of Lemma 4.14

possible cases. □

4.5.4 The Main Result on Overlapping Embeddings

Using the point x from the previous lemma, we are now able to construct the four artificial edges, which we have used in the proof of Theorem 4.3 on page 57, *without* presuming the existence of overlapping kernels. This way, we can adapt the proof of Theorem 4.3 in order to show the stronger and more general Theorem 4.2.

Proof of Theorem 4.2. As already stated before, the proof of this theorem is almost identical to the proof of the less general Theorem 4.3, which requires overlapping kernels, except for the construction of the artificial edges $\overline{a_1\, a_2}$ and $\overline{b_1\, b_2}$ of embedding $h(V^*)$ and their translated counterparts $\overline{a_1'\, a_2'}$ and $\overline{b_1'\, b_2'}$ in embedding $h'(V^*)$.

Lemma 4.14 guarantees the existence of a point x in the intersection of the two embeddings that sees an artificial edge $\overline{a_1\, a_2}$ of $h(V^*)$, when looking in direction t, and an artificial edge $\overline{b_1'\, b_2'}$ of $h'(V^*)$, when looking in the opposite direction $-t$. Thus, we can assure that a line l through x and parallel to t intersects the two artificial edges in a way such that the intersection point of l with $\overline{a_1\, a_2}$ lies to the right of the intersection point of l with $\overline{b_1'\, b_2'}$. Furthermore, l also intersects the artificial edges $\overline{b_1\, b_2} = \overline{b_1'\, b_2'} - t$ of $h(V^*)$ and $\overline{a_1'\, a_2'} = \overline{a_1\, a_2} + t$ of $h'(V^*)$. It should be noted that we have not proven whether x actually *sees* these so-constructed artificial edges, but this plays no role in the proof, since for the remaining arguments only the visibility relationships with respect to the *witnesses* w and w' are of interest.

In the proof of Theorem 4.3 we have used the angular order with respect to the kernel point x only to express in which order the eight endpoints of the artificial edges occur on the boundary of the map polygon. Thus, if we

- let the vertex sequence of the map polygon start at the vertex a'_1 (cf. Figure 4.4 on page 59),
- replace each occurrence of »$a \prec_x b$« by the term »a is before b in the vertex sequence of the map polygon«, and
- replace each occurrence of »$a \preccurlyeq_x b$« by the term »a and b are identical or a is before b in the vertex sequence of the map polygon«,

we can safely use the argumentation of the proof of Theorem 4.3 to study the placements of the eight endpoints of the artificial edges. This way, we are able to show that the assumption of witnesses w and w', which can see each other, in all cases yields a contradiction. □

The following corollaries follow from Theorem 4.2 and from the proof of Theorem 4.3, respectively.

Corollary 4.15 *Let $h(V^*)$ and $h'(V^*)$ be two different valid embeddings of the same skeleton V^* with witnesses w and w'. If the embedded skeleton polygons $h(V^\diamond)$ and $h'(V^\diamond)$ overlap, their intersection region either lies totally to the left or totally to the right of the straight line through w and w'.*

Proof. Otherwise, the part of the map polygon that must lie between w and w' and intersect the straight line segment $\overline{w\,w'}$ (due to Theorem 4.2), would be encircled by four lines of sight (cf. Figure 4.9 on page 66), which contradicts Observation 3.3. Note that the whole intersection polygon must be visible to both witnesses. □

From the previous corollary it particularly follows that no witness of the two embeddings may lie in the intersection region, that is, w must lie outside of $h'(V^\diamond)$ and w' must lie outside of $h(V^\diamond)$.

Corollary 4.16 *Let $h(V^*)$ and $h'(V^*)$ be two different overlapping valid embeddings of the same skeleton V^* with witnesses w and w' and let $t = h'(v_o) - h(v_o)$ be the translation vector that maps the embedded origin $h(v_o)$ onto the embedded origin $h'(v_o)$.*

Then, for any straight line that is parallel to t and crosses the intersection region of $h(V^*)$ and $h'(V^*)$ follows that both witnesses w and w' lie on the same side of that line.

Proof. Using the notations of the proof of Theorem 4.2/4.3 on page 57, let l' be an arbitrary straight line parallel to the translation vector t, which crosses the intersection region. As in the previous proofs, we assume w. l. o. g. that t is horizontal and directed from left to right. Furthermore, we already know from the proof of Theorem 4.3 that the witnesses w and w' must lie on the same side of the straight line l (since the other case was shown to be impossible on page 62) and w. l. o. g. we assume that they lie below l. Thus, it suffices to revisit the case »w and w' lie on the same side of l« (from page 62) and show that in this case the witnesses also lie on the same side of l'.

To this end, recall that from Lemma 4.4 follows that w lies below the straight line $\overleftrightarrow{a_2 a_2'}$ and that w' lies below $\overleftrightarrow{b_2 b_2'}$, that is, we get a situation as in Figure 4.6 (b) on page 62 with $a_2 \preccurlyeq_x b_2'$. From this situation it directly follows that the intersection polygon of $h(V^*)$ and $h'(V^*)$ must lie above the line $\overleftrightarrow{b_2 b_2'}$. Thus, also l' (which crosses that polygon) must lie above b_2 and b_2', from which the primary claim follows. □

4.6 Intersecting versus Overlapping Embeddings

In the preceding sections we have always investigated valid embeddings that *overlap*, that is, embeddings of skeleton polygons whose intersection again is a polygon. For those cases we have shown that the corresponding witnesses of the embeddings cannot see each other. In the following we shortly take a look on embeddings that only *intersect* but do not overlap, that is, embeddings of (identical) skeleton polygons whose intersection consists of a single point or a straight line segment. Here, the question arises, whether also in these cases the witnesses cannot see each other? And indeed, one can construct valid embeddings that intersect in exactly one artificial edge such that the *witnesses are visible* to each other. An example of such a situation is depicted in Figure 4.28 on the next page where the witnesses w and w' for the two valid embeddings of $V^*_{w/w'}$ see each other, whereas the two embedded kernels (which are identical to $V^\diamond_{w/w'}$) intersect in a straight line segment.

If we revisit the proof of Theorem 4.3 on page 57, we recognize the situation of Figure 4.28 as a special case of »$a_1' \prec_x b_1' \preccurlyeq_x a_1 \prec_x b_1 \prec_x b_2 \prec_x a_2 \preccurlyeq_x b_2' \prec_x a_2'$« in the distinction of cases, where the two pairs of middle vertices are identical, that is, $a_1 = b_1'$ and $a_2 = b_2'$. Apparently, the existence of a point in the *interior* of the intersection of the two embeddings plays a crucial role in the proof of Theorem 4.2/4.3.

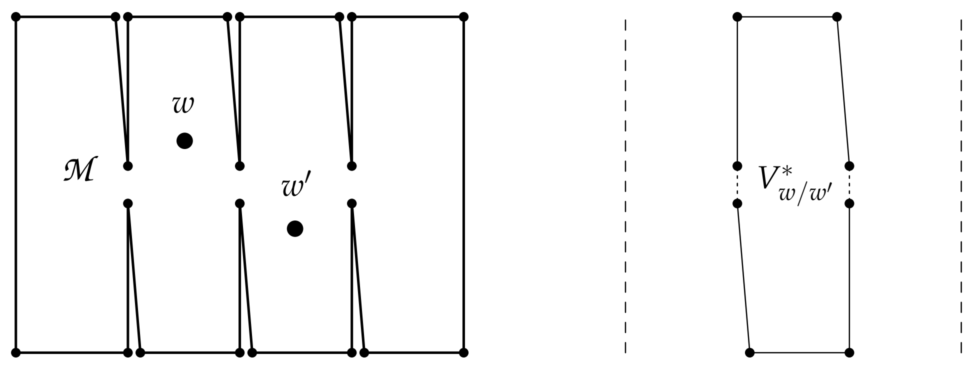


FIGURE 4.28: A map with two valid embeddings, where the kernels *intersect* and the witnesses are visible to each other

4.7 Crossing Candidate Edges

Hitherto, we have concentrated on overlapping valid embeddings and discussed the implications that arise for the visibility of the corresponding witnesses. In the following we extend these investigations to include also the corresponding candidate edges into our considerations. We prove some useful properties of candidate edges of overlapping embeddings, since they will be particularly helpful in order to determine their total number in Section 5.3.

At first, we state the following property, which shows that candidate edges of a *single* embedding cannot be arbitrarily positioned "around" the skeleton polygon. Instead, they belong to their respective artificial edges.

Lemma 4.17 *Let a and a' be two different artificial edges of a skeleton V^* with an embedding h. Furthermore, let e be a candidate edge of $h(a)$ and e' be a candidate edge of $h(a')$. Then, it is not possible that a line of sight from inside $h(V^\diamond)$ to e and a line of sight from inside $h(V^\diamond)$ to e' intersect outside of $h(V^\diamond)$.*

Thus, it particularly follows that for a single embedding a given edge may be a candidate edge of *at most one* artificial edge. That is, the edges e and e' in the above lemma actually must be different.

Proof. We assume that there exist (embedded) artificial edges $h(a) \neq h(a')$ with candidate edges e and e', such that the corresponding lines of sight intersect *outside* of $h(V^\diamond)$, and show that this leads to a contradiction by constructing a circle in the map polygon that must contain at least one vertex, such that Observation 3.3 is contradicted.

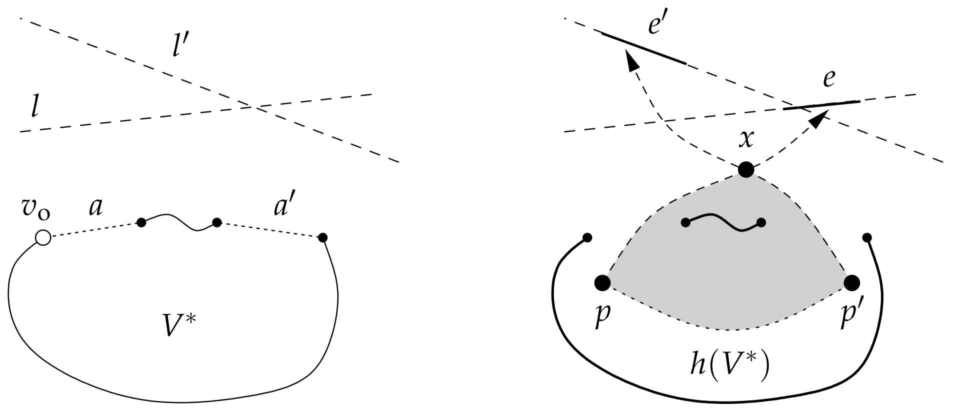


FIGURE 4.29: Lines of sight to candidate edges that cross outside of $h(V^\diamond)$ are impossible

See Figure 4.29 that depicts the assumed situation: In embedding $h(V^*)$ a point $p \in h(V^\diamond)$ sees the candidate edge e "through" the embedded artificial edge $h(a)$, represented by the dotted arrow. Analogously, the point p' sees e' "through" $h(a')$ such that the lines of sight intersect in x outside of $h(V^\diamond)$. Then, some vertices of $h(V^*)$ must be encircled by a closed path inside $\mathcal{M}$, hence, a contradiction to Observation 3.3. □

The following observation compares the angular order of edges that are visible through an artificial edge with the angular order of the corresponding region, from which these edges are visible (confer also the forthcoming definition of visibility wedges on page 108).

Observation 4.3 *Let a_1 and a_2 be two map vertices and e_i possible candidate edges for the (directed) segment $\overline{a_1\,a_2}$ lying in the half plane $H^-(\overleftrightarrow{a_1\,a_2})$. That is, for each e_i, there is an embedding h_{j_i} and an artificial edge a_{k_i} such that e_i is a candidate edge for $h_{j_i}(a_{k_i}) = \overline{a_1\,a_2}$.*[1]

Then, for any point $x \in \overline{a_1\,a_2}$ the angular order of the edges e_i with respect to x (with starting direction $\overrightarrow{a_2\,a_1}$) is identical to the order in which the edges e_i appear on the polygon boundary of the map polygon $\mathcal{M}$. Furthermore, also the angular orders of the regions in the half plane $H^+(\overleftrightarrow{a_1\,a_2})$, which see the edge e_i "through" $\overline{a_1\,a_2}$, are identical according to the illustration in Figure 4.30 on the facing page.

1 To be precise, we additionally should require that the *complete edge e_i* is visible to a_1 and a_2.

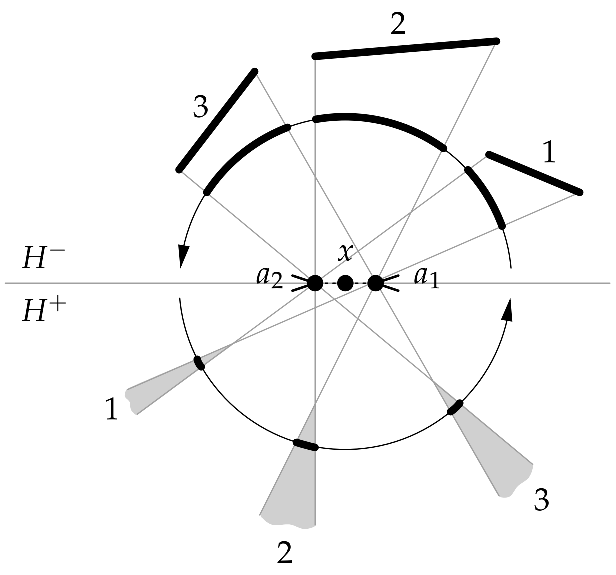


FIGURE 4.30: Candidate edges and their angular order around the corresponding artificial edge

In Section 5.3 it turns out that for two *different* embeddings the most interesting cases are those where the lines of sight between the embeddings and the candidate edges cross, that is, where it seems that the candidate edges occur "in the wrong order" on the polygon boundary. To substantiate this concept, we first introduce the notion of *crossing candidate edges* in the following.

Definition 4.4 (Crossing Candidate Edges)
Let c be a valid candidate edge of an artificial edge a in embedding $h(V^*)$ and let $c' \neq c$ be another valid candidate edge of an artificial edge a' (not necessarily different from a) in another embedding $h' \neq h$ of V^*.

We say that the candidate edge c (of h) ***crosses*** *the candidate edge c' (of h') if there exists a witness $w \in h(\ker V^*)$ that sees a point $p \in c$ through $h(a)$ and a witness $w' \in h'(\ker V^*)$ that sees a point $p' \in c'$ through $h'(a')$, such that the lines of sight from w to p and from w' to p' intersect.*

Example 4.4 There are various situations, in which candidate edges may cross, both with overlapping or intersecting embeddings as well as with completely separated embeddings:

1. Consider the three (non-overlapping) embeddings of the skeleton V_2^* in Figure 3.9 on page 37. There, the edge e_2 is a candidate edge of the embedded artificial edge in $h_1(V_2^*)$ that crosses e_1, which is a candidate edge in embedding $h_2(V_2^*)$.

2. As an example of crossing candidate edges in overlapping embeddings look at Figure 4.1 (a) on page 55. Here, the four horizontal edges in the four northern niches are candidate edges of the two embeddings of V^*, of which the two middle ones cross each other.

3. Also in the case of the two intersecting embeddings in Figure 4.28 on page 93 there are pairs of crossing candidate edges, for example the ones that are visible from the two witnesses w and w'. ◁

In the following we concentrate on *overlapping* embeddings and show that crossing candidate edges may occur only in certain patterns. To this end, the next lemma enables us to roughly divide overlapping embeddings with crossing candidate edges into two categories depending on the number of corresponding artificial edges that intersect the other respective skeleton polygon. Either

- *exactly one artificial edge*, $h(a)$ or $h'(a')$, intersects the other respective skeleton polygon, or

- *both artificial edges* intersect the other respective skeleton polygon, that is, $h(a)$ intersects $h'(V^\diamond)$ and $h'(a')$ intersects $h(V^\diamond)$.

Lemma 4.18 *Let c and c' be two crossing candidate edges. Let a and a' be the corresponding artificial edges and $h \neq h'$ the corresponding embeddings.*

If the two embeddings $h(V^*)$ and $h'(V^*)$ overlap, at least one of the two following conditions must hold:

- *$h(a)$ intersects $h'(V^\diamond)$*
- *$h'(a')$ intersects $h(V^\diamond)$*

Proof. Assume that neither $h(a)$ intersects $h'(V^\diamond)$ nor $h'(a')$ intersects $h(V^\diamond)$. This means, that both embedded artificial edges lie outside of the intersection polygon. Since c crosses c', there exist lines of sight from witness w (inside of the embedding h) through $h(a)$ to c and from witness w' (inside of the embedding h') through $h(a')$ to c', which intersect (outside of the embeddings). Thus, we get a situation similar to the one of Figure 4.29, where some map vertices are encircled by a closed path, which contradicts Observation 3.3. □

In the remaining part of this section we prove the following theorem (and a direct corollary thereof) about pairs of mutually crossing candidate edges in overlapping embeddings. We show that for such situations there basically is only one pattern, in which the candidate edges may occur in the map polygon. This helps us in Section 5.3 in order to prove a tight bound on the total number of candidate edges.

Theorem 4.19 *Let c_1 and c_2 be valid candidate edges of an artificial edge a in embedding $h(V^*)$ such that c_1 appears before c_2 on their (directed) common line. Furthermore, let c' be a valid candidate edge of an artificial edge a' in embedding $h' \neq h$ of V^* such that c' crosses c_1 as well as c_2.*

If the two embeddings $h(V^*)$ and $h'(V^*)$ overlap, the edge c' cannot lie between c_1 and c_2 on the boundary of the map polygon.

This theorem directly implies the following corollary about pairs of mutually crossing candidate edges.

Corollary 4.20 *Let c_1 and c_2 be valid candidate edges of an artificial edge a in embedding $h(V^*)$ such that c_1 appears before c_2 on their (directed) common line. Analogously, let c'_1 and c'_2 be valid candidate edges of an artificial edge a' in embedding $h' \neq h$ of V^* such that c'_1 appears before c'_2 on their (directed) common line. Further assume that each c_i $(i = 1, 2)$ crosses each c'_j $(j = 1, 2)$.*

If the two embeddings $h(V^*)$ and $h'(V^*)$ overlap, the order in which the four candidate edges appear on the boundary of the map polygon is either c_1, c_2, c'_1, c'_2 or c'_1, c'_2, c_1, c_2.

In order to clarify this claim, view the sketched example of Figure 4.31 on the following page. It shows two overlapping embeddings h and h' of a skeleton V^*, each with a pair of candidate edges (c_1, c_2 and c'_1, c'_2), which mutually cross each other. Confer Lemma 4.18 and note that both corresponding artificial edges $h(a)$ and $h'(a')$ intersect the other respective skeleton polygon. Further note that the corresponding witnesses w_1 and w_2 in embedding $h(V^*)$ see each other (since they both lie in the same embedded kernel) as well as the witnesses w'_1 and w'_2 do, but that no witness w_i is visible to a witness w'_j according to Theorem 4.2. These visibility relationships between the witnesses are the key to prove Theorem 4.19 in the following. As can be further seen in the figure, the order of the four candidate edges on the map polygon is c_1, c_2, c'_1, c'_2 as claimed by the above corollary. In particular, no candidate edge of $h(a)$ lies between the candidate edges of $h'(a')$ and vice versa.

Proof of Theorem 4.19. We assume that the claim of the theorem is false and show that this leads to a contradiction (namely, as in many proofs before, a contradiction to Observation 3.3 by constructing a hole in the map polygon). To this end, let c_1 and c_2 be valid candidate edges of an artificial edge a in embedding $h(V^*)$ such that c_1 appears before c_2 on their (directed) common line. Furthermore, let c' be a valid candidate edge of an artificial edge a' in embedding $h' \neq h$ of V^*, which overlaps h, such that

- c' lies between c_1 and c_2 on the boundary of $\mathcal{M}$,

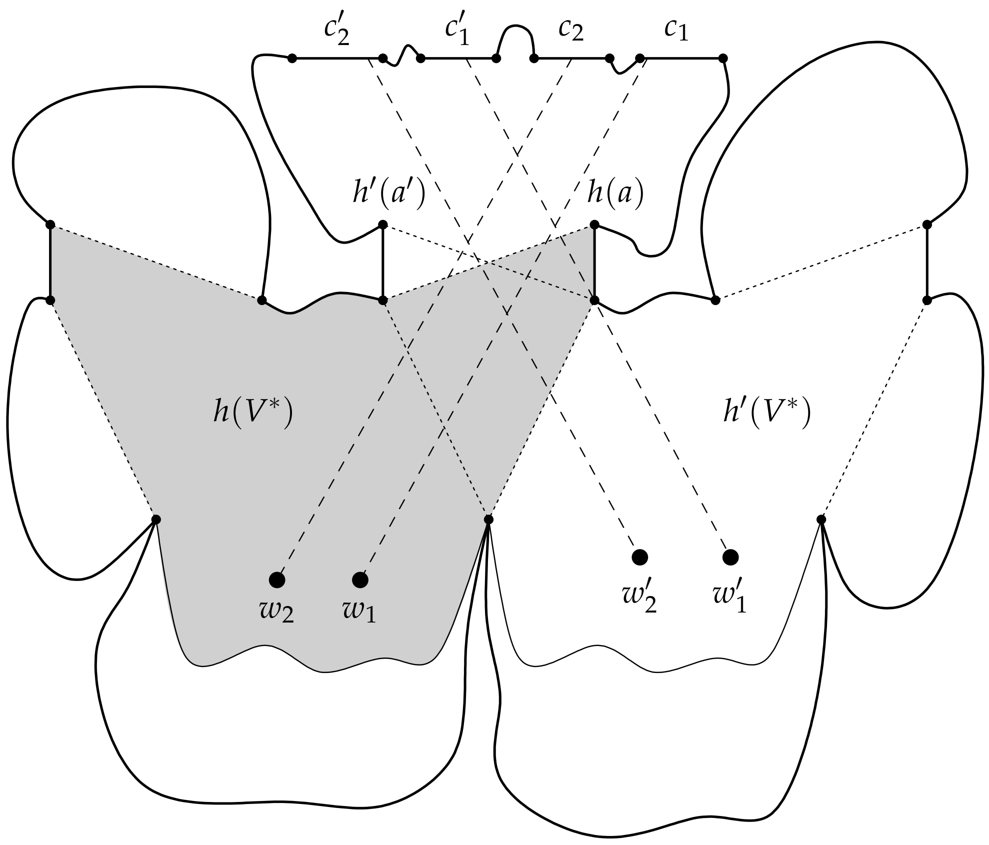


FIGURE 4.31: A sketched example of two pairs of mutually crossing candidate edges and the situation claimed by Corollary 4.20

- c' crosses both candidate edges, c_1 as well as c_2.

For the following, let w_1, w_2, and w' be the witnesses for the valid candidate edges c_1, c_2, and c', respectively. We first claim that w' sees c' not only through the artificial edge $h'(a')$, but also through $h(a)$.

Claim. The line of sight from w' to c' intersects the embedded artificial edge $h(a)$, that is, w' and c' lie on different sides of $h(a)$.

Proof. Again, we prove this claim by contradiction. To this end, let $\tilde{w}_1' \in h'(\ker V^*)$ and $\tilde{w}_1 \in h(\ker V^*)$ be the witnesses of »c' crosses c_1« (cf. Definition 4.4). Analogously, let $\tilde{w}_2'$ and $\tilde{w}_2$ be the witnesses of »c' crosses c_2«. Note that $\tilde{w}_1'$ and $\tilde{w}_2'$ need not be identical and that these two witnesses do not even have to lie in the same visibility cell. Then, we get a situation as depicted in Figure 4.32 on the next page. Remember that in the "forbidden region" (drawn in

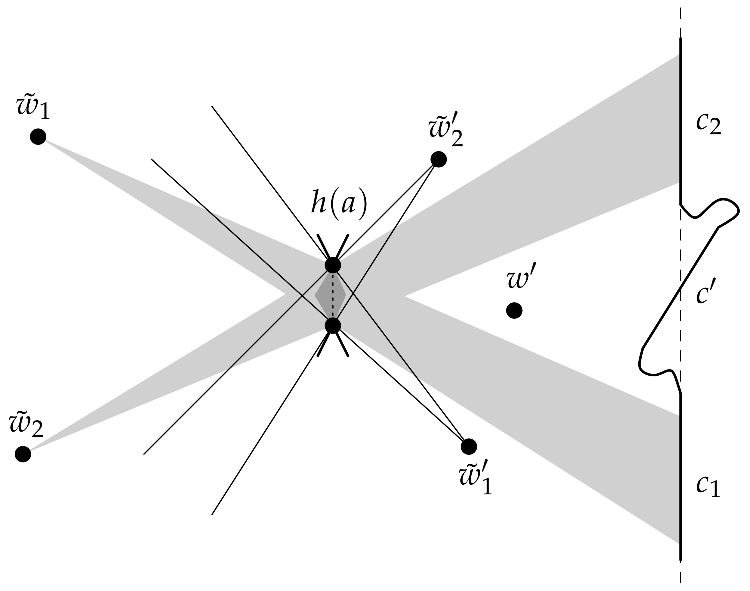


FIGURE 4.32: »w' and c' lie on the same side of $h(a)$« implies a contradiction

light-gray), which consists of the view cones of $\tilde{w}_1$ and $\tilde{w}_2$ through $h(a)$, there cannot lie any map vertices, since otherwise $\tilde{w}_1$ and $\tilde{w}_2$ could not be witnesses of embedding $h(V^*)$. Nor may any witnesses of $h'(V^*)$ lie in these view cones, since otherwise we immediately get a contradiction to Theorem 4.2.

In the following we assume that w' and c' lie on the same side of $h(a)$, as depicted in the figure, and show that for this case the two embeddings $h(V^*)$ and $h'(V^*)$ cannot overlap, which contradicts our prerequisites. To this end, firstly note that if w' lies on the right side of $h(a)$, also the witnesses $\tilde{w}'_1$ and $\tilde{w}'_2$ must lie on the right side of $h(a)$ as depicted in Figure 4.32. This is, because as soon as a viewpoint enters or leaves the region around $h(a)$ drawn in dark-gray, which consists of the intersection of the four positive half planes incident to the endpoints of $h(a)$, its skeleton changes. In particular, any skeleton of a viewpoint that lies to the left of this region is different from any skeleton of a viewpoint that lies to the right of it. And since the region around $h(a)$ is totally contained in the "forbidden region" (drawn in light-gray), which consists of the two view cones of $\tilde{w}_1$ and $\tilde{w}_2$, the witness w' must lie outside of it and therefore on the right side of $h(a)$, as well as the witnesses $\tilde{w}'_1$ and $\tilde{w}'_2$.

We further conclude that $\tilde{w}'_1$ must lie below the view cone of $\tilde{w}_1$, since the respective lines of sight (from $\tilde{w}'_1$ to c' and from $\tilde{w}_1$ to c_1) must intersect, according to Definition 4.4. Analogously, $\tilde{w}'_2$ must lie above the view cone of $\tilde{w}_2$ as depicted in the figure. Now recall that $\tilde{w}'_1$ and $\tilde{w}'_2$ are witnesses of the same embedding, that is, both witnesses must see the same non-spurious vertices and full edges. In particular, the vertices of $h'(V^*)$ that lie on the left side of $h(a)$ must lie in *both* view cones. But as Figure 4.32 shows, the intersection of these view cones is totally contained in the “forbidden region” described above. Thus, there cannot lie any vertex to the left of $h(a)$, which is seen from the witnesses of $h'(V^*)$. In other words, no vertex of $h'(V^*)$ lies to the left of $h(a)$. Therefore, the two embeddings cannot overlap, which contradicts our assumptions and proves the claim that w' and c' must lie on different sides of $h(a)$. □

From the previous claim follows that the witnesses w_1, w_2, and w' lie on the same side of $h(a)$ and using Observation 4.3 we conclude that their angular order “w. r. t. $h(a)$” is identical to the angular order of the candidate edges c_1, c_2, and c'. Thus, we get the situation depicted in Figure 4.33 on the facing page: Without loss of generality we assume that the edges c_1, c', and c_2 are placed from right to left above the horizontal artificial edge $h(a)$. Thus, the witnesses w_1, w', and w_2 lie below $h(a)$ from left to right. In particular, the witness w' must lie in the triangular wedge (drawn with dotted lines) induced by the end-points of the two candidate edges c_1 and c_2 and the two reflex vertices of $h(a)$, as shown in the figure.

The dashed lines in the figure indicate lines of sight between the witnesses and their respective candidate edges. Therefore, no vertex may lie in the interior of the light-gray “forbidden region” that is created by these lines of sight. Due to Theorem 4.2 at least one map vertex must lie between w' and each of the witnesses w_1 and w_2. As a consequence, w' must lie in the dark-gray region below the straight line through w_1 and w_2. Otherwise, these map vertices would lie in the “forbidden region” and cause a contradiction.

Since the line of sight from w' to c' crosses $h(V^\diamond)$, it must enter the skeleton polygon through an artificial edge of $h(V^*)$. Let x and y be the reflex vertices of this embedded artificial edge. Furthermore, let x' and y' be the reflex vertices of the embedded artificial edge of $h'(V^*)$ that belongs to c'. Then we claim the following about the position of the vertices x' and y'.

Claim. At least one of the vertices x' and y' must either be identical to a vertex of $h(a)$ or lie on the opposite side of $h(a)$ as w'.

Note that such a vertex x' (or y', respectively), which lies on the opposite side of $h(a)$ as w', must be placed “above” $h(a)$ in the triangular region, drawn in dark-gray in Figure 4.34 on page 102.

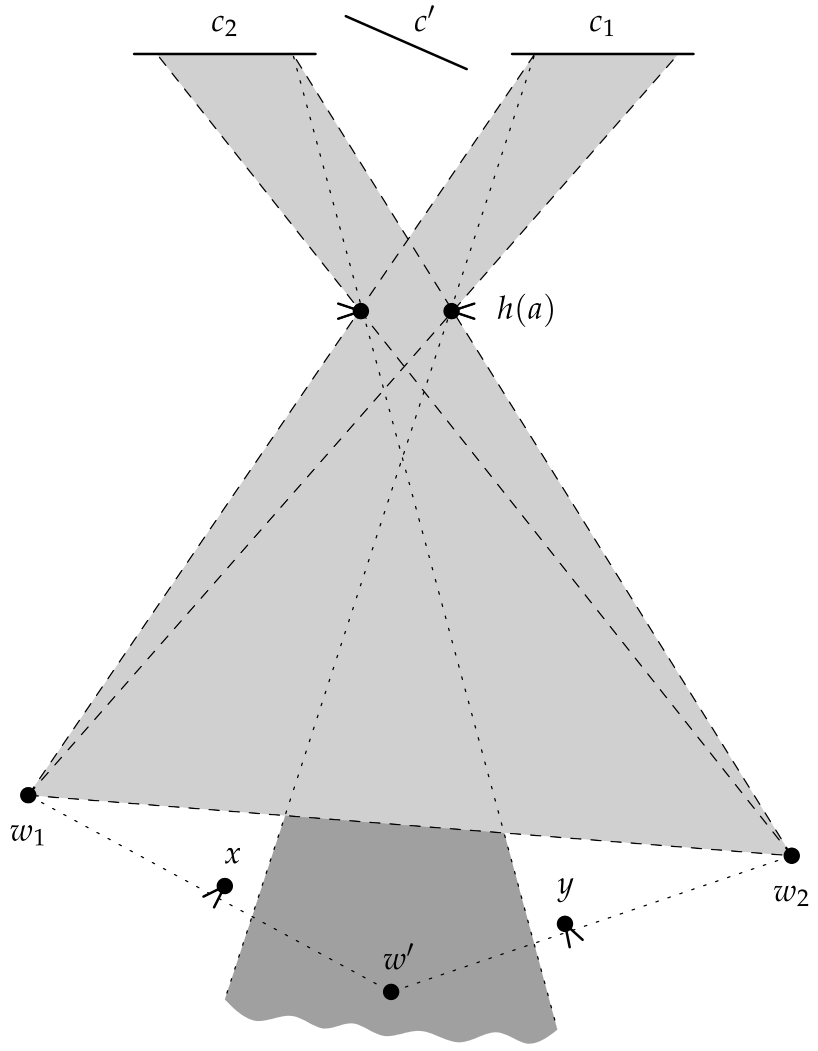


FIGURE 4.33: The placement of w' in the proof of Theorem 4.19

Proof. We assume that this is not the case. Thus, *both* vertices x' and y' must lie on the same side of $h(a)$ as w', that is, below $h(a)$ in Figure 4.34 on the following page. From the preceding considerations follows that neither x' nor y' may lie in the "forbidden region" drawn light-gray in the figure.

In addition, they also cannot lie in the two regions to the left of the line of sight from w_1 to the left vertex of $h(a)$ and to the right of the line of sight from w_2 to the right vertex of $h(a)$, drawn dark-gray in the figure. This is, because in such a case at least one of the vertices of $h(a)$ would be visible to w' through the artificial edge $\overline{x'\,y'}$, which is impossible. (Remember that w' sees c' through $h(a)$.) Therefore, both vertices must lie below $\overline{x\,y}$, from which follows

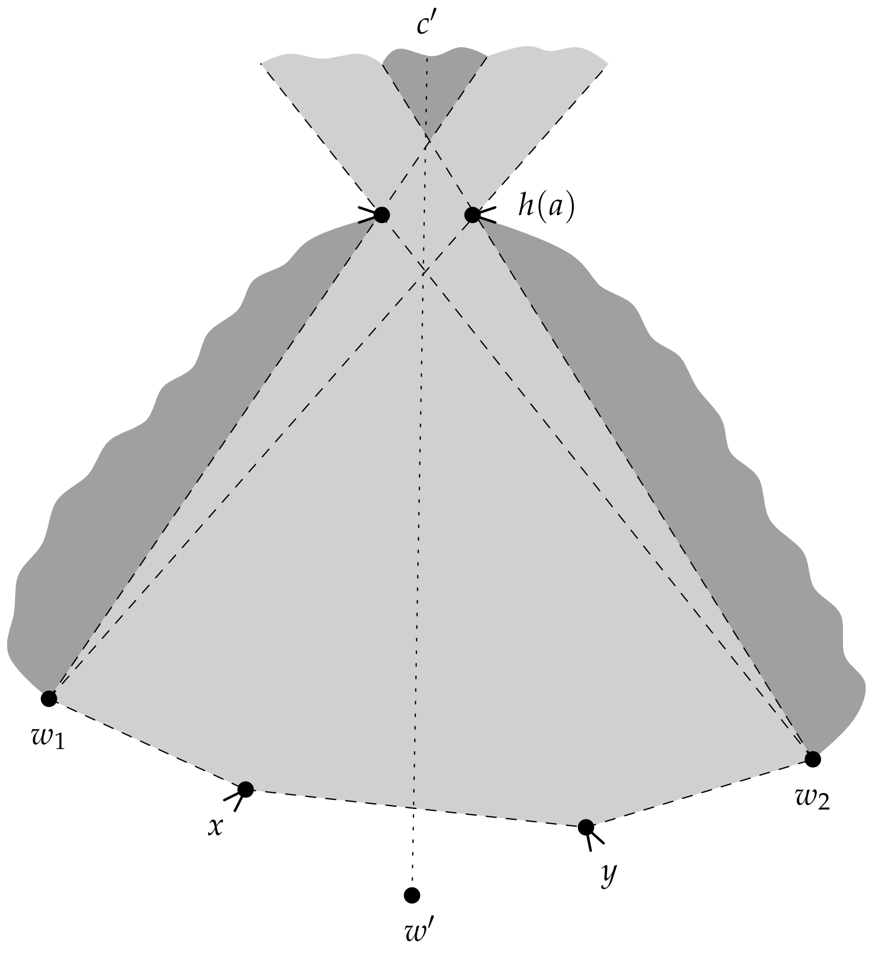


FIGURE 4.34: At least one vertex of $h'(V^\diamond)$ lies above $h(a)$

that $\overleftrightarrow{x\,y}$ and $\overleftrightarrow{x'\,y'}$ cannot strictly intersect.

But from this fact directly follows that the two embeddings $h(V^*)$ and $h'(V^*)$ cannot overlap at all, since otherwise a circle would exist, which encloses at least one of the vertices $\{x, y, x', y'\}$, that is, a contradiction to Observation 3.3. □

Let z' be the vertex of the artificial edge corresponding to c', which we have proved to lie on or *above* $h(a)$ in the previous claim. Note that z' is a vertex of $h'(V^*)$ and that $w' \in h'(V^*)$ lies *below* $\overleftrightarrow{x\,y}$. Thus, we conclude that the embedded kernel $h'(\ker V^*)$ must lie in the view cone of z' with respect to $\overleftrightarrow{x\,y}$. Moreover, if z' is different from the vertices of $h(a)$, that is, z' lies strictly above $h(a)$, we analogously conclude that the kernel $h'(\ker V^*)$ also must lie in the view cone of z' with respect to $h(a)$. Furthermore, it directly follows from the properties of w_1, w_2, and z' that the witnesses w_1 and w_2 lie *outside* of these view cones (cf. Figure 4.34).

In order to complete the proof, we use the fact that h and h' are embeddings of the same skeleton V^*, that is, they have identical skeleton polygons and identical kernels. Thus, the witnesses w_1 and w_2 of $h(V^*)$ must have counterparts[2] $h'(h^{-1}(w_1)) =: w_1^{\mathrm{t}}$ and $h'(h^{-1}(w_2)) =: w_2^{\mathrm{t}}$ that must lie in the embedded kernel of $h'(V^*)$. Of course, the same holds for the vertices of the embedded artificial edges $h(a)$ and $\overleftrightarrow{x\,y}$. In other words: The polygonal region that is bounded by w_1, x, y, w_2, and the endpoints of $h(a)$ must be translated in a way such that w_1^{t} and w_2^{t} lie inside the kernel $h'(\ker V^*)$ and no one of the visibility constraints is violated. We show in the following that this leads to contradiction to Observation 3.3.

First of all, it should be clear that w' and the witness counterparts w_1^{t} and w_2^{t} must lie on the same side of $\overleftrightarrow{x\,y}$, that is, w_1^{t} and w_2^{t} must lie *below* $\overleftrightarrow{x\,y}$ (cf. Figure 4.34). Otherwise the lines of sight from w' to w_1^{t} and w_2^{t} would cross $\overleftrightarrow{x\,y}$. As a consequence, the triangle that consists of w', w_1^{t}, and w_2^{t} would encircle at least one of the also-translated vertices x^{t} and y^{t}, which would contradict Observation 3.3.

Therefore, we yield a situation similar to the one sketched in Figure 4.35 on the next page. From the argument above follows that the witnesses w_1^{t} and w_2^{t} must be located in the gray shaded region such that they can see the vertex z'. Observe further that the angular order of the points $w_1, w_2, x, y, w_1^{\mathrm{t}}, w_2^{\mathrm{t}}$ with respect to z' must be $w_1 \prec_{z'} x \prec_{z'} w_1^{\mathrm{t}} \prec_{z'} w_2^{\mathrm{t}} \prec_{z'} y \prec_{z'} w_2$, as depicted in the figure.

Now recall that the vertex z' must either be identical to a vertex of $h(a)$ or it must lie "above" $h(a)$ in the triangular region, drawn in dark-gray in Figure 4.34. In the first case, z' is also a vertex of embedding $h(V^*)$ and in the second case, we conclude that both vertices of $h(a)$ lie in the triangle that consists of z', w_1, and w_2. Thus, in both cases at least one vertex of $h(a)$ lies in this triangle and w. l. o. g. we may assume that this vertex is z'. That is, the vertex z' is a non-spurious vertex in *both* embeddings and therefore must have a translated counterpart $(z')^{\mathrm{t}}$.

Finally, observe that the translated point $(z')^{\mathrm{t}}$ is encircled by the points w_1^{t}, w_2^{t}, and z' as shown in Figure 4.35 (with dashed lines of sight). As a consequence, in both cases at least one map vertex is encircled by the points w_1^{t}, w_2^{t}, and z', which see each other. This contradicts Observation 3.3 and shows that our initial assumption (that the claim of the proof is false) must be wrong. □

2 For points and vertices in embedding $h(V^*)$, which are translated this way, we write w^{t} for short instead of $h'(h^{-1}(w))$ in the following.

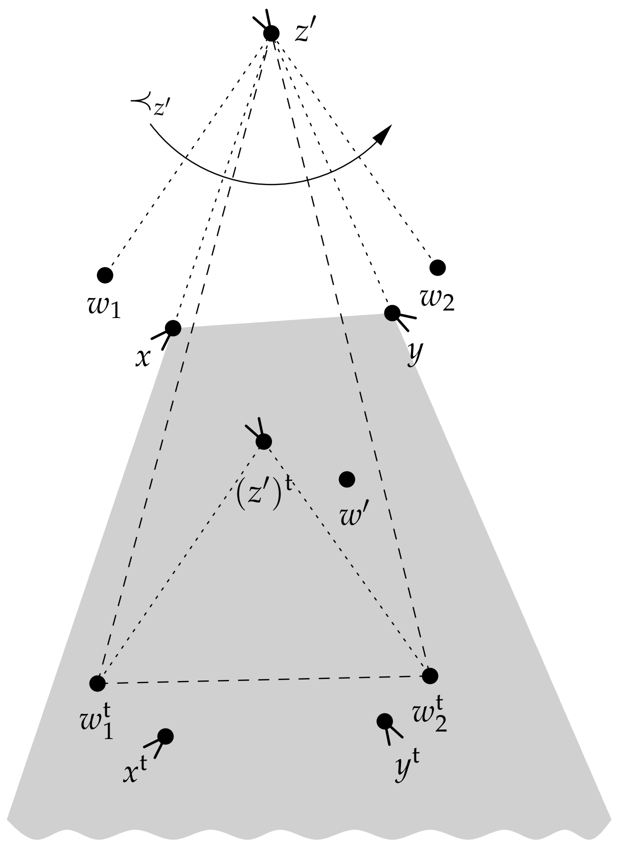


FIGURE 4.35: The points w_1^{t}, w_2^{t}, and z' encircle at least one map vertex

Corollary 4.20 investigates pairs of mutually crossing candidate edges and shows that there basically is only one pattern how the candidate edges appear on the polygon boundary, provided that the two corresponding embeddings overlap. The following corollary also investigates pairs of candidate edges, but relaxes the crossing prerequisites. It states that if the candidate edges appear on the polygon boundary in a certain pattern, the two embeddings cannot overlap.

Corollary 4.21 *Let c_1 and c_2 be valid candidate edges of an artificial edge a in embedding $h(V^*)$ and let c'_1 and c'_2 be valid candidate edges of an artificial edge a' in embedding $h' \neq h$ of V^* such that the four candidate edges are pairwise different.*

If the order in which the candidate edges appear on the boundary of the map polygon is c_1, c'_1, c_2, c'_2, the two embeddings h and h' cannot overlap.

Note that from the order of the candidate edges it directly follows that at

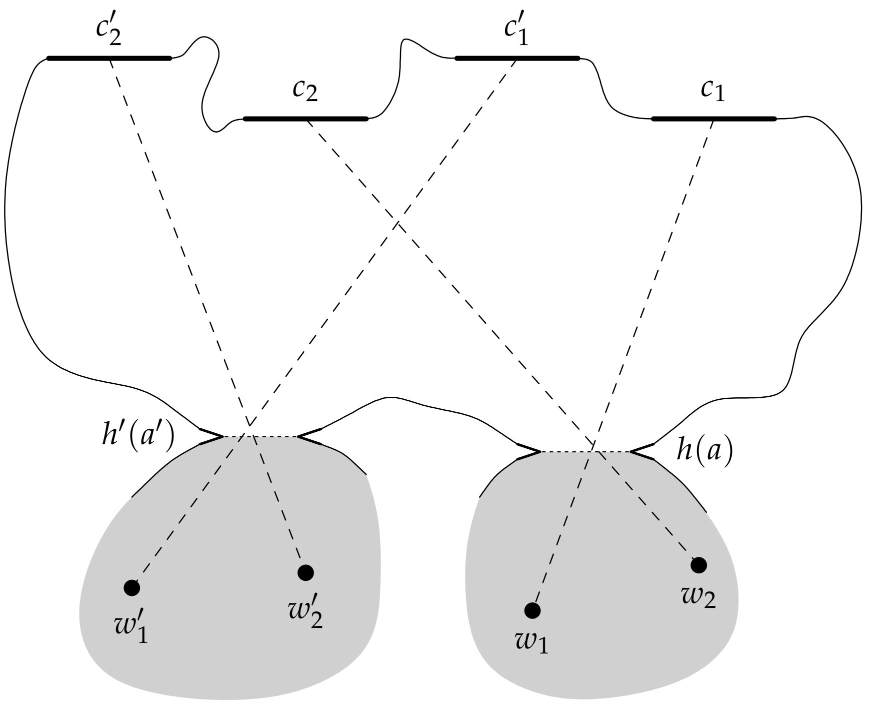


Figure 4.36: The artificial edges $h(a)$ and $h'(a')$ cannot intersect

least one c_i crosses one c_j'.

Proof. We assume that the two embeddings overlap and show that this leads to a contradiction. To this end, let w_1, w_2, w_1', w_2' be witnesses for the candidate edges c_1, c_2, c_1', c_2' and note that w_1' (and therefore also w_2') must lie on the other side of $h(a)$ than w_1 and w_2. Otherwise, the line of sight from w_1' to c_1' would cross *both* lines of sight from w_1 to c_1 *and* from w_2 to c_2. Thus, Theorem 4.19 would be applicable and result in a contradiction, since c_1' lies between c_1 and c_2 on the boundary of the map polygon. Analogously, the witnesses w_1 and w_2 must lie on the other side of $h'(a')$ than w_1' and w_2'. Thus, the resulting situation is similar to the one shown in Figure 4.36: The map polygon is divided into three regions, two of which contain the witnesses of the two embeddings (drawn gray in the figure) and a third region ("between" them) that contains the four candidate edges.

From this division it directly follows that the artificial edges $h(a)$ and $h'(a')$ cannot intersect, which in consequence contradicts Lemma 4.18. Thus, the assumption that the two embeddings overlap must be wrong. □

4.8 Summary

In this chapter we investigated overlapping embeddings, that is, embeddings of the same skeleton, where the embedded skeleton polygons overlap. We introduced the notion of valid embeddings and gave worst-case examples of $\Omega(r)$ overlapping valid embeddings. Note that such overlappings are the main source of complication when we further investigate the structure of the localization problem. In particular, most of the results of the following chapter could be proved much easier if we may exclude the case of overlapping embeddings from our considerations.

Towards our main result, we investigated the structure of general overlapping embeddings (that is, embeddings of skeletons, which are not necessarily identical) and proved some properties of their intersection region (Lemma 4.5 to Corollary 4.10), namely that it is a single polygonal region consisting of only a few types of edges. Moreover, we analyzed the different types of intersecting skeleton edges in detail and showed that certain patterns may occur only a constant number of times.

The main result of this chapter (Theorem 4.2) describes the connection between two overlapping embeddings (of the same skeleton) and the visibility of their witnesses and states that these witnesses cannot see each other. We proved this by first investigating the special case where also the kernels of the embeddings overlap (Theorem 4.3), for which the claim is much easier to verify. Then, we showed that one of the key structures that we used in order to prove Theorem 4.3 also exists in the general case, such that the proof could be adapted to it in a straightforward way.

This result on the visibility of the witnesses plays a central role in the next chapter and is used as a powerful tool to circumvent the complications caused by overlapping embeddings mentioned above. It also was used in order to prove some results on crossing candidate edges, which were the last point of consideration in this chapter. Crossing candidate edges occur if the lines of sight from two embeddings to two corresponding candidate edges intersect. We showed (using Theorem 4.2) that in the case of overlapping embeddings crossing candidate edges may occur only in certain patterns, which also is a helpful property in the next chapter.

5 Revisiting the Complexity of an Equivalence Class

"And shall th' ungrateful traitor go," she said,
"My land forsaken, and my love betray'd?
Shall we not arm? not rush from ev'ry street,
To follow, sink, and burn his perjur'd fleet?
Haste, haul my galleys out! pursue the foe!
Bring flaming brands! set sail, and swiftly row!
What have I said? ***Where am I?*** *Fury turns*
My brain; and my distemper'd bosom burns.
Then, when I gave my person and my throne,
This hate, this rage, had been more timely shown."

Queen Dido in "Aeneid"
VERGIL (70–19 B. C.)

IN CHAPTER 3 we stated a result of Guibas et al. that provided us with an upper bound of $O(n^2)$ on the worst-case complexity $|\mathcal{EC}_{V^*}|$ of an equivalence class, see Theorem 3.10 on page 47. Recall that the complexity of an equivalence class $\mathcal{EC}_{V^*}$ is defined as the total number of vertices and edges of all visibility cells with skeleton V^*.

In this chapter we establish a refined upper bound of $O(n + r^2)$ on $|\mathcal{EC}_{V^*}|$, where the dependence on the number r of reflex vertices becomes clearer. We also show that this bound is worst-case optimal. The main idea of the proof is to determine the complexity of an equivalence class $\mathcal{EC}_{V^*}$ by accurately counting the vertices and edges of all visibility cells whose skeleton equals V^*. Therefore, we first study the structure of the visibility cells and classify their boundary edges into two groups. As it turns out, only the edges of one type, which are determined by the embedded candidate edges of V^*, are difficult to count, and we first give an upper bound on their number in a *single* embedding of V^*.

Unfortunately, summing up these numbers for all embeddings of the skeleton V^* does not yield the desired outcome. The reason is that one edge may serve as a candidate edge in several embeddings as already shown in Example 3.3 on page 38. The whole setting gets even more complicated in the case of overlapping embeddings, which we have discussed in the previous chapter. Therefore, we examine such situations and show how to summarize these numbers in a more sophisticated way.

To this end, we have to further investigate the candidate edges and distinguish between *single* and *multiple* candidate edges in order to determine tight upper bounds on their total numbers. At this point, the results of the previous chapter will be very helpful.

For the following, let V^* be a fixed skeleton with s artificial edges $a_1, \ldots, a_s$ (with corresponding lines $l_1, \ldots, l_s$) and t valid embeddings $h_1, \ldots, h_t$ into the map. For an artificial edge a_i and an embedding h_j, we denote by $C_{i,j}$ the set of all valid candidate edges of a_i in embedding $h_j(V^*)$, which lie on line $h_j(l_i)$, and by $c_{i,j}$ the cardinality of $C_{i,j}$. For example, the sets of (valid) candidate edges of the skeleton V_2^* in Figure 3.9 on page 37 are $C_{1,1} = \{e_1, e_2\}$, $C_{1,2} = \{e_1, e_2, e_3\}$, $C_{1,3} = \{e_2, e_3\}$, with a_1 being the only artificial edge of V_2^*.

5.1 The Structure of the Visibility Cells

For a fixed skeleton V^*, we examine the visibility cells whose skeleton equals V^*. Each edge of such a visibility cell in an embedding $h(V^*)$ is either a *kernel edge* or a *wedge edge* as defined below.

Definition 5.1 (Visibility Wedge)
*For an artificial edge a_i of V^* and a candidate edge e of a_i in embedding h_j, the corresponding **visibility wedge** $w_{i,j}^e$ consists of all points that can see the candidate edge e "through" the embedded artificial edge $h_j(a_i)$. That is,*

$$w_{i,j}^e := H^+(r_r) \cap H^-(r_l) .$$

*Here, r_l is the uniquely defined ray that is anchored at a vertex of a_i and emanates from a vertex of e such that the remaining vertices of a_i and e both lie to the **left** of r_l. Analogously, r_r is the uniquely defined ray that is anchored at a vertex of a_i and emanates from a vertex of e such that the remaining vertices of a_i and e both lie to the **right** of r_r.*

The notion of visibility wedges is illustrated in Figure 5.1 (a) on the facing page, which again shows the map of Example 3.1 on page 24, concentrating on embedding $h_2(V^*)$. In the upper part, each candidate edge e_1, e_2, e_3 of the artificial edge a_1 in embedding h_2 induces a corresponding visibility wedge

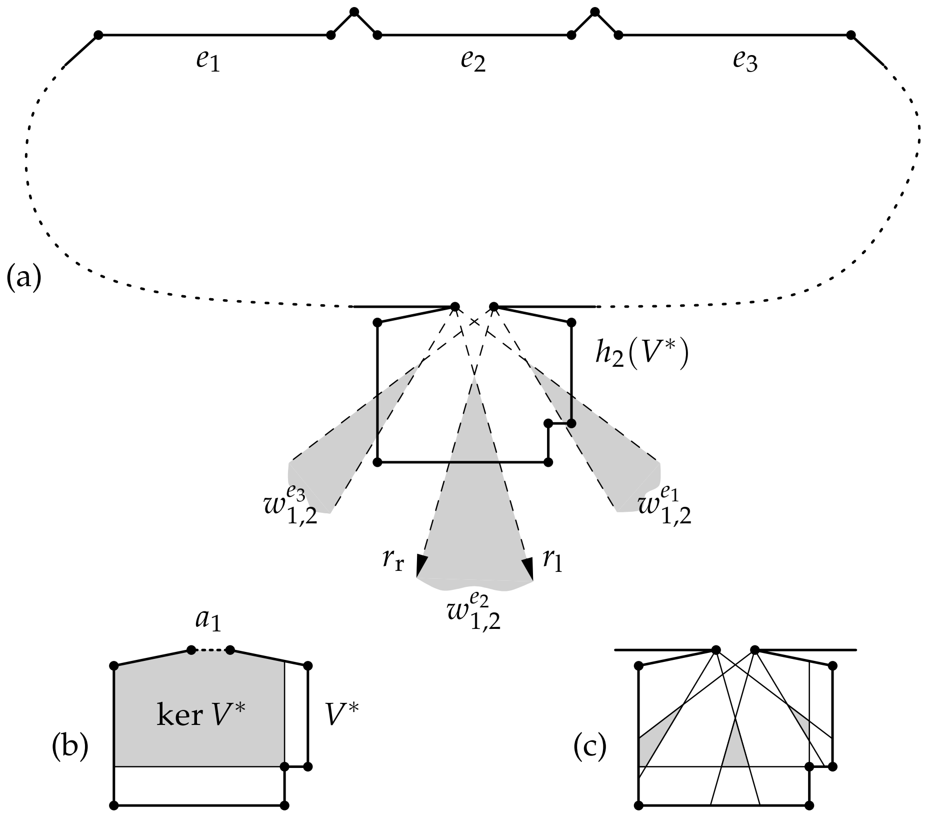


FIGURE 5.1: Three candidate edges with corresponding visibility wedges

$w^{e_1}_{1,2}, w^{e_2}_{1,2}, w^{e_3}_{1,2}$. The visibility wedge $w^e_{i,j}$ can alternatively be regarded as the union of view cones with respect to $h_j(a_i)$ of points on e without the union of view cones of points on $l \setminus e$, where l denotes the straight line corresponding to e. This reflects the intuitive definition of the set of all points that can see e "through" $h_j(a_i)$.

Definition 5.2 (Kernel and Wedge Edges)
Let C be a visibility cell with skeleton V^*, which lies in embedding $h_j(V^*)$. An edge of C that lies on the boundary of the embedded kernel $h_j(\ker V^*)$ is called a ***kernel edge*** *of C. An edge of C that lies on the boundary of any of the visibility wedges $w^e_{i,j}$ is called a* ***wedge edge*** *of C.*

For an example of the definition of kernel and wedge edges, again look at Figure 5.1, which depicts the three visibility wedges of the artificial edge a_1 in embedding h_2. If we intersect these wedges with the kernel of V^*, see Figure 5.1 (b), we get the three visibility cells in embedding h_2, see Figure 5.1 (c).

The horizontal and vertical edges of the cells are therefore kernel edges, the remaining ones are wedge edges.

The classification of the edges of a visibility cell $\mathcal{C}$ into kernel edges and wedge edges is actually a partition of the set of edges of $\mathcal{C}$, because of the following arguments: Let e be an edge of a visibility cell $\mathcal{C}$.

1. If e is part of a map edge $\overline{u\,v}$, the two map vertices u and v must be visible from every point $p \in \mathcal{C}$, cf. Theorem 3.3. Thus, $\overline{u\,v}$ belongs to the kernel of $V^*_{\mathcal{C}}$ and e must be a kernel edge.
2. In the other case, e lies on a visibility ray, anchored at some reflex vertex v_{r} (visible from $\mathcal{C}$) and emanating from a vertex u.
    a) If (besides v_{r}) also u is visible from $\mathcal{C}$, $\overline{v_{\mathrm{r}}\,u}$ must be an edge of V^*, either a full edge or an artificial edge, since any other vertex in V^* "between" v_{r} and u would constitute an additional visibility ray, which would divide $\mathcal{C}$. Consequently, $\overline{v_{\mathrm{r}}\,u}$ induces an edge of ker V^*, on which e lies, and e is a kernel edge.
    b) If u is not visible from $\mathcal{C}$, it becomes visible by crossing e. Therefore, v_{r} is a blocking vertex of an artificial edge a of V^* and u is a vertex of a candidate edge of a. That is, the visibility ray anchored at v_{r} and emanating from u is one of the two bounding rays of the corresponding visibility wedge. Thus, e is a wedge edge.

We summarize this consideration:

Observation 5.3 *Each edge of a given visibility cell $\mathcal{C}$ is either a kernel edge of $\mathcal{C}$ or a wedge edge of $\mathcal{C}$.*

The next lemma gives an upper bound on the total number of kernel edges in $\mathcal{EC}_{V^*}$. It shows that it suffices to count only the wedge edges, in order to establish an upper bound of $O(n + r^2)$ on the complexity $|\mathcal{EC}_{V^*}|$.

Lemma 5.1 *For a given skeleton V^* the total number of kernel edges in $\mathcal{EC}_{V^*}$ is in $O(n + r^2 + W)$, where W denotes the total number of wedge edges in $\mathcal{EC}_{V^*}$.*

Proof. As we have already seen in the above discussion, a kernel edge either lies on a map edge (case 1) or on a certain kind of visibility ray (case 2a). Let K denote the set of all such map edges and visibility rays, on which kernel edges of $\mathcal{EC}_{V^*}$ lie. Since each reflex vertex contributes at most two visibility rays to each of the $O(r)$ embedded kernels, it follows that $|K| \in O(n + r^2)$.

If more than one kernel edge lies on an edge from K (see for example Figure 5.1 (c), where two kernel edges lie on the horizontal edge of ker V^*), these additional edges are generated by the intersection of a visibility wedge with an edge of K. Since each visibility wedge may intersect each edge of K only once, producing at least one wedge edge, the number of additional kernel edges is bounded by the number W of wedge edges (up to a constant factor). □

5.2 Counting the Wedge Edges of an Equivalence Class

From the preceding Lemma 5.1 it follows that in order to prove our desired upper bound of $O(n + r^2)$ on the complexity $|\mathcal{EC}_{V^*}|$, it remains to count the wedge edges of all visibility cells in an equivalence class $\mathcal{EC}_{V^*}$. To this end, we consider each embedding h_j of the skeleton V^* and count the number of wedge edges of cells in $h_j(\ker V^*)$. Summing up over all embeddings yields the total number of wedge edges in $\mathcal{EC}_{V^*}$. The following lemma limits the number of those edges in a *single* embedding h_j.

Lemma 5.2 *The total number of wedge edges of cells that lie in embedding h_j of a given skeleton V^* is in*

$$O\left(r + \left(\sum_{i=1}^{s}(c_{i,j}-1)\right)^2 - \sum_{i=1}^{s}(c_{i,j}-1)^2\right). \tag{5.1}$$

Proof. Recall from Definition 5.1 that a visibility wedge $w^e_{i,j}$ consists of exactly those points that can see the candidate edge e through the embedded artificial edge $h_j(a_i)$. Extending the argumentation of Observation 5.3 it follows that each visibility cell with skeleton V^* in embedding $h_j(V^*)$ must lie in exactly one visibility wedge, for each of the s artificial edges. Figure 5.2 on the following page illustrates this situation for the embedding $h_j(V^*)$ of a rectangular skeleton with two artificial edges a_1 and a_2: Each one of the six visibility cells lies either in $w^{e_1}_{1,j}$, $w^{e_2}_{1,j}$, or $w^{e_3}_{1,j}$, and also either in $w^{f_1}_{2,j}$ or $w^{f_2}_{2,j}$.

Therefore, the set of all visibility cells from $\mathcal{EC}_{V^*}$ in embedding h_j can be seen as the intersection of the set

$$\overline{W}_j := \bigcap_{i=1}^{s} W_{i,j} \tag{5.2}$$

with the kernel $h_j(\ker V^*)$ of the embedding, where

$$W_{i,j} := \bigcup_{x \in C_{i,j}} w^x_{i,j} \tag{5.3}$$

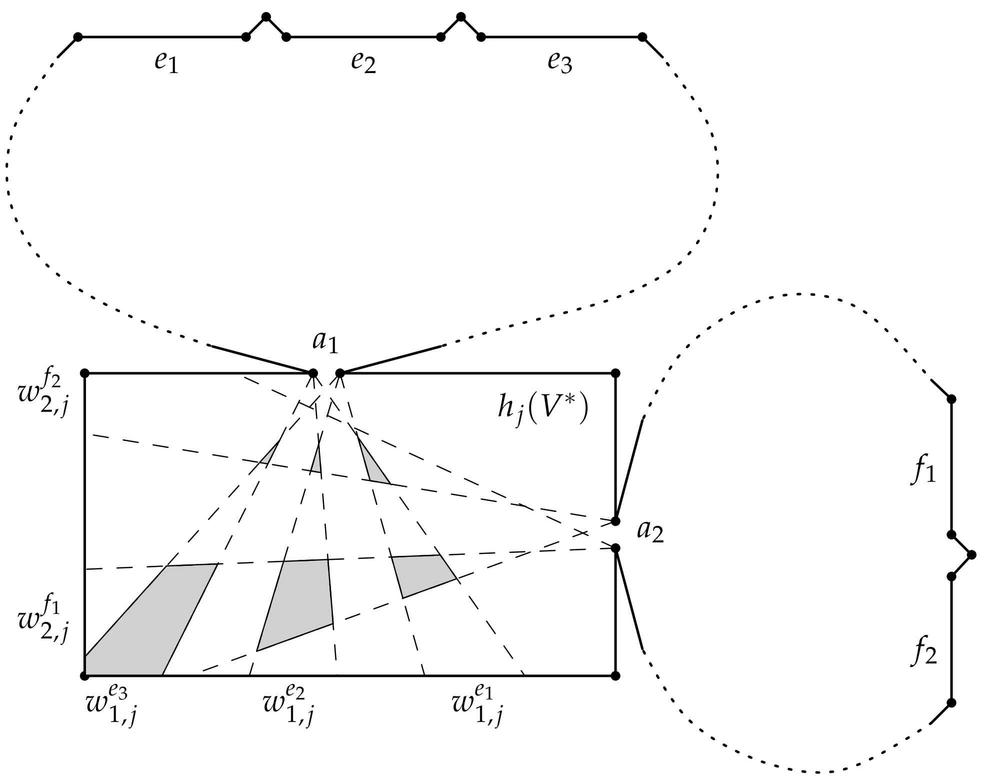
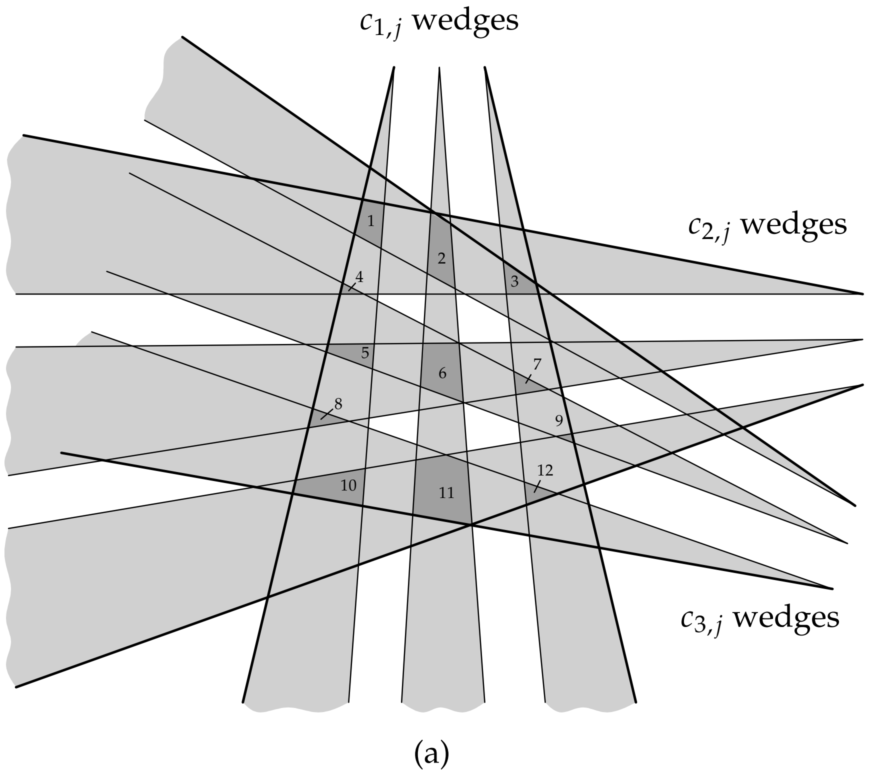


FIGURE 5.2: Visibility cells created by intersecting sets of visibility wedges

denotes the union of all wedges for a single artificial edge $h_j(a_i)$. In particular, the complexity of $\overline{W_j}$ (that is, the number of vertices and edges) is up to a constant factor an upper bound on the number of wedge edges in $h_j(V^*)$, since intersecting $\overline{W_j}$ with $h_j(\ker V^*)$ only decreases the number of wedge edges.

In order to determine the complexity of $\overline{W_j}$, we have to carefully count those vertices and edges of the arrangement of the sets $W_{i,j}$ that belong to visibility cells of V^*. To this end, examine Figure 5.3 (a) on the next page, which shows the intersection of three sets $W_{i,j}$, each of which consists of $c_{1,j} = c_{2,j} = c_{3,j} = 3$ pairwise non-intersecting visibility wedges. Note that only a fractional amount of all vertices of the arrangement belongs to the dark-gray visibility cells.

Further note that if the width of the interspaces between the wedges is reduced, the total number of intersection points does not change, although some additional visibility cells may be introduced. If the width is reduced to zero, we arrive at an arrangement of $\sum_{i=1}^{s}(c_{i,j} - 1)$ rays, as illustrated by Figure 5.3 (b). In other words, the complexity of the arrangement of these rays (which represent the interspaces between the wedges) and a single convex polygon (which



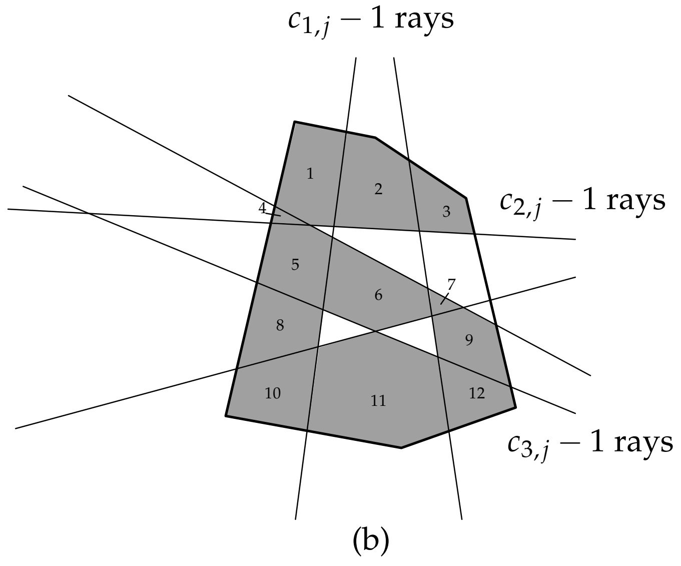


FIGURE 5.3: The complexity of the dark-gray visibility cells in (a) is not greater than the complexity of the arrangement of the rays and the single polygon in (b)

represents the intersection of the two outermost half planes corresponding to each of the artificial edges) is up to a constant factor an upper bound on the complexity of $\overline{W}_j$.

The total complexity of the arrangement of the s sets of rays can then be determined in a straightforward manner: Each one of the $c_{1,j} - 1$ rays, which represent the interspaces between the wedges of $W_{1,j}$, may intersect each one of the $\sum_{i=2}^{s}(c_{i,j} - 1)$ remaining rays. Analogously, each one of the $c_{2,j} - 1$ rays corresponding to the set $W_{2,j}$, may intersect each one of the $\sum_{i=3}^{s}(c_{i,j} - 1)$ remaining rays, and so on. This way, we get a total complexity of

$$\begin{aligned}(c_{1,j} - 1)\left(\sum_{i=2}^{s}(c_{i,j} - 1)\right) + (c_{2,j} - 1)\left(\sum_{i=3}^{s}(c_{i,j} - 1)\right)\\ + \cdots + (c_{s-2,j} - 1)\left(\sum_{i=s-1}^{s}(c_{i,j} - 1)\right) + (c_{s-1,j} - 1)(c_{s,j} - 1)\\ = O\left(\left(\sum_{i=1}^{s}(c_{i,j} - 1)\right)^2 - \sum_{i=1}^{s}(c_{i,j} - 1)^2\right).\end{aligned}$$

The single *convex* polygon, which represents the intersection of the two outermost half planes corresponding to each of the s artificial edges, obviously has at most two intersection points with each of the edges of the arrangement. Thus, for the total complexity we only have to consider the at most $2s$ polygon vertices that are introduced. Since the number s of artificial edges cannot be larger than the number r of reflex vertices (cf. Definition 3.7), summing up the complexities yields the desired bound of Equation (5.1). □

Remember that our goal is to show an upper bound of $O(n + r^2)$ on the worst-case complexity $|\mathcal{EC}_{V^*}|$ of an equivalence class and that to this end (using the results of Observation 5.3 and Lemma 5.1) we have to prove an $O(n + r^2)$ bound on the total number of wedge edges in $\mathcal{EC}_{V^*}$. Taking advantage of Lemma 3.7 and Lemma 5.2 this number is in

$$O\left(r^2 + \sum_{j=1}^{t}\left(\left(\sum_{i=1}^{s}(c_{i,j} - 1)\right)^2 - \sum_{i=1}^{s}(c_{i,j} - 1)^2\right)\right), \tag{5.4}$$

such that it suffices to show an $O(n + r^2)$ bound for the second term of this sum. Note that each one of the numbers s, t, and $c_{i,j}$, if viewed on their own, may have a worst-case bound of $\Omega(r)$, cf. Lemma 3.7. This means, that if we perform the summation in Equation (5.4) in a naïve way, we only get an unreasonable upper bound of $O(r^5)$ on the number of wedge edges. However, in the following sections we show that we can summarize these numbers more elaborately such that we actually are able to prove a worst-case upper bound of $O(r^2)$ for this complexity.

5.3 Single and Multiple Candidate Edges

Since the numbers $c_{i,j}$ of (valid) candidate edges apparently play an important role in determining the complexity $|\mathcal{EC}_{V^*}|$, we study them in the following two sections. We present some worst-case examples and also prove some upper bounds, which are useful later. Note that even the trivial upper bound of $O(n)$ on the total number of candidate edges for a given skeleton V^* can be shown to be tight in the worst-case. To this end, consider a map with $\Omega(n)$ niches (like in the bottom part of Figure 3.15 on page 47) such that the skeleton of a suitably chosen viewpoint contains $\Omega(n)$ artificial edges (one per niche), each of them with one corresponding candidate edge. Thus, the total number of candidate edges of this skeleton also is in $\Omega(n)$.

However, for this construction also the number r of reflex vertices is in $\Omega(n)$, that is, the total number of candidate edges actually is in $O(r)$. But in the following we present a worst-case example with $\Omega(n)$ valid candidate edges using only $O(\sqrt{n})$ reflex vertices. In other words, we show that for small numbers r of reflex vertices the total number of possible candidate edges of cells in $\mathcal{EC}_{V^*}$ may be as high as $\Omega(r^2)$ and particularly cannot be bounded by $O(r)$.

In contrast we show for a certain subset of candidate edges, the so-called *multiple candidate edges*, an upper bound of only $O(r)$ on their number in the next section. It turns out that only these multiple candidate edges are responsible for large numbers of visibility cells such that a small upper bound of only $O(r)$ on their number is one of the key ingredients to the proof that the complexity of Equation (5.4) actually is in $O(n + r^2)$.

To this end, firstly let C_{all} be the set of all valid candidate edges for all artificial edges in all embeddings of the skeleton V^*, that is,

$$C_{\text{all}} := \bigcup_{\substack{1 \le i \le s \\ 1 \le j \le t}} C_{i,j} \,.$$

As already shown above, the cardinality of C_{all} may be as high as $\Omega(n)$ in the worst-case. Now we distinguish between *single* and *multiple* candidate edges and show tight upper bounds on their cardinalities in the following sections:

Definition 5.4 (Single and Multiple Candidate Edges)
A valid candidate edge $c \in C_{\text{all}}$ is called a ***multiple candidate edge*** *if there exists indices i and j such that $c \in C_{i,j}$ and $c_{i,j} = |C_{i,j}| > 1$. Otherwise, c is called a* ***single candidate edge****. We denote with C_{mult} the subset of C_{all} that consists of all multiple candidate edges and with $C_{\text{sing}} := C_{\text{all}} \setminus C_{\text{mult}}$ the remaining set of single candidate edges.*

Lemma 5.3 *For a skeleton V^* the total number $|C_{\text{sing}}|$ of single candidate edges is in $O(\min\{n, r^2\})$ and this bound is tight.*

Proof. Each candidate edge $c \in C_{\text{all}}$ is contained in at least one set $C_{i,j}$ of candidate edges, which means, c is a candidate edge of the artificial edge a_i in embedding h_j. (But note that c might be contained in more than one of these sets.) If c additionally is a *single* candidate edge, it directly follows that $C_{i,j} = \{c\}$. Thus, the number of sets of candidate edges is an upper bound on the number of single candidate edges and therefore $|C_{\text{sing}}| \leq s \cdot t \in O(r^2)$, cf. Definition 3.7 and Lemma 3.7.

It remains to show the tightness of the bound. To this end, consider the skeleton V^* that is sketched in Figure 5.4 (a) on the next page. It is similar to the skeleton that we used for the proof of Lemma 4.1 on page 55 to show the $\Omega(r)$ bound on the number of intersecting embeddings and consists of $s - 2$ identical horizontal artificial edges $a_3, \ldots, a_s$ plus two artificial edges a_1 and a_2 on the left and on the right side of the skeleton. The main difference to the skeleton of Lemma 4.1 is that the corresponding lines $l_3, \ldots, l_s$ of the artificial edges $a_3, \ldots, a_s$ are not identical. Instead, each line can be viewed as a tangent to a *convex* niche, as depicted in Figure 5.4 (a). This way, if we overlay the $\Theta(s)$ (identical) niches, the corresponding tangent lines form a *convex* polygon that consists of $\Theta(s)$ vertices.

Now consider the map polygon $\mathcal{M}$ shown in Figure 5.4 (b), which consists of $s + t - 1$ identical convex polygonal niches (each one containing $\Theta(s)$ non-reflex vertices, as described above) such that the skeleton V^* has t different embeddings $h_1, \ldots, h_t$ into $\mathcal{M}$. Obviously, the total number of reflex vertices of $\mathcal{M}$ is in $O(s + t)$ and the total number $|C_{\text{sing}}|$ of single candidate edges is in $\Omega(s \cdot t)$. Thus, if we choose $s \in \Theta(t)$, we get a total number of $\Omega(r^2)$ single candidate edges. □

5.4 The Total Number of Multiple Candidate Edges

In this section we show a tight upper bound of $O(r)$ on the total number of multiple candidate edges. To this end, we first introduce the notion of *gaps*

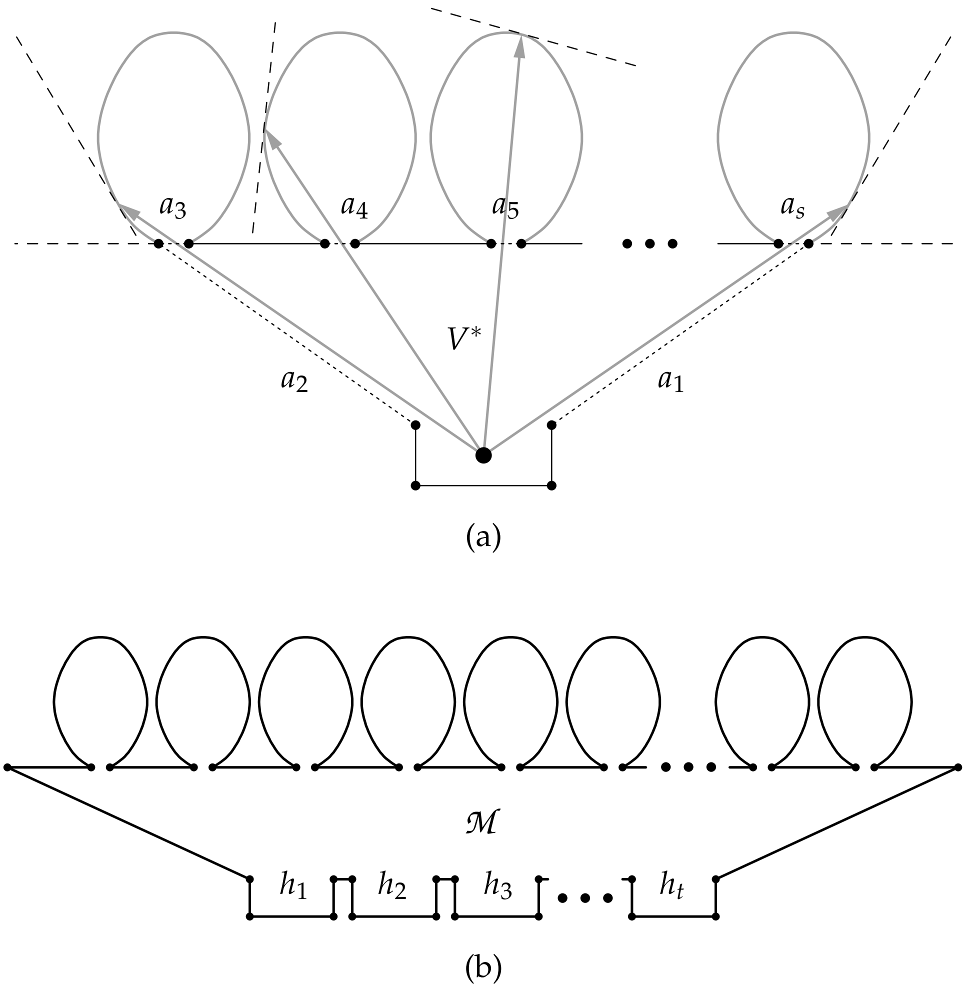


FIGURE 5.4: The skeleton V^* in (a) consists of s artificial edges and has t embeddings into the map $\mathcal{M}$ in (b), which consists of only $O(s+t)$ reflex vertices but results in a total of $\Omega(s \cdot t)$ single candidate edges

between consecutive candidate edges, as defined below, and show that their number is of the same order of magnitude as the number of multiple candidate edges.

The main problem in accurately and correctly counting the gaps (or candidate edges, respectively) is then the case where several gaps overlap or are nested into each other. These cases will be the main subject of investigation in the following.

5.4.1 Gaps between Consecutive Candidate Edges

Definition 5.5 (Gap between Candidate Edges)
Let $s, e \in C_{i,j}$ be two multiple candidate edges lying on (the oriented) line $h_j(l_i)$ such that s occurs before e on $h_j(l_i)$ and let $\overline{s\,s_1}, \overline{s_1\,s_2}, \ldots, \overline{e_2\,e_1}, \overline{e_1\,e}$ be the polygonal chain of the map polygon from s to e.

*If s and e are consecutive candidate edges (that is, no other candidate edge of $C_{i,j}$ lies between s and e), we call the chain $g(s,e) := \overline{s_1\,s_2}, \ldots, \overline{e_2\,e_1}$ the corresponding **gap** between the consecutive candidate edges s and e.*

From this definition it directly follows that the total number of gaps differs at most by a factor of two from the total number $|C_{\text{mult}}|$ of multiple candidate edges:

Lemma 5.4 *The total number of gaps for a skeleton V^* and the total number $|C_{\text{mult}}|$ of multiple candidate edges for V^* are of the same order of magnitude.*

Furthermore, it is clear that each gap $g(s,e)$ (that is, the polygonal chain between s and e) must contain at least one reflex vertex. Thus, if we assume that the gaps are neither *nested* nor *interweaved* (as defined below), we could easily map each gap *injectively* onto a reflex vertex. This way, the resulting injective mapping of the set of all gaps into the set of reflex vertices would guarantee that there are at most $O(r)$ gaps. Unfortunately, this is not always the case, as Example 5.1 illustrates.

Definition 5.6 (Nested and Interweaved Gaps)
Two different gaps $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$ are called

- ***nested** if either s_1 and e_1 are part of the polygonal chain g_2 (i. e., g_1 is nested into g_2) or s_2 and e_2 are part of the polygonal chain g_1 (i. e., g_2 is nested into g_1),*
- ***interweaved** if s_1 or e_1 is part of the polygonal chain g_2 and s_2 or e_2 is part of the polygonal chain g_1.*

Example 5.1 (A map with nested and interweaved gaps)
Figure 5.5 on the facing page shows a skeleton V^* with two artificial edges a and b, which has two valid embeddings h_1 and h_2. Each of the five witnesses $p_1, \ldots, p_5$ sees a different pair of candidate edges. The artificial edge $h_2(a)$ has three multiple candidate edges a_1, a_2, and a_3, that is, we have two gaps $g(a_1, a_2)$ and $g(a_2, a_3)$. The artificial edge $h_1(b)$ has two multiple candidate edges b_1 and b_2, that is, there is one gap $g(b_1, b_2)$.

Thus, the gap $g(a_2, a_3)$ is nested into the gap $g(b_1, b_2)$, which itself is interweaved with the gap $g(a_1, a_2)$. ◁

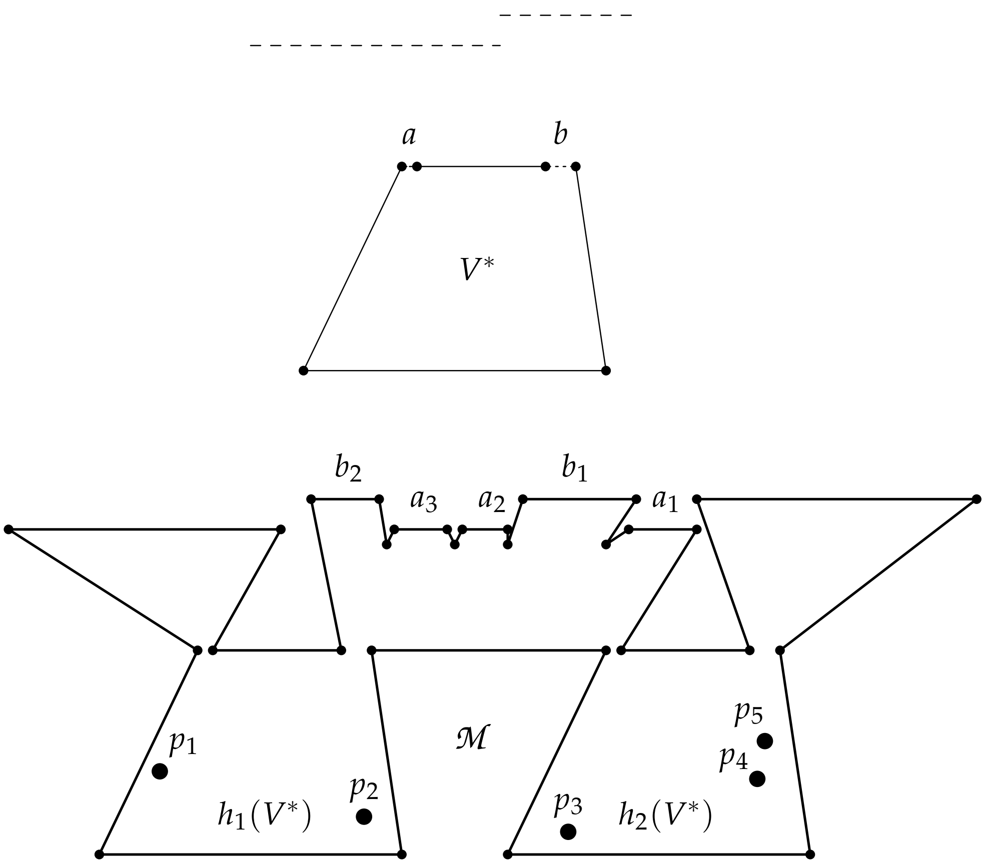


FIGURE 5.5: An example of nested and interweaved gaps

As the previous example illustrates, we cannot guarantee the injectivity of our mapping using the simple argument from above, although the number of (multiple) candidate edges actually is less than the number of reflex vertices in Example 5.1. For such cases, we have to take a closer look on nested and interweaved gaps. Here, the main problem is that in the case of interweaved gaps *concave chains* of candidate edges may occur in the map polygon without any reflex vertex between the individual candidate edges. This is sketched in Figure 5.6 on the next page, which shows a map with four pairs of collinear edges (which may serve as candidate edges) and only three reflex vertices. In such a case the idea of mapping each gap onto the reflex vertex that it must contain fails, since this mapping need not be injective. In order to cope with this problem, we first state some basic properties of nested and interweaved gaps in the following.

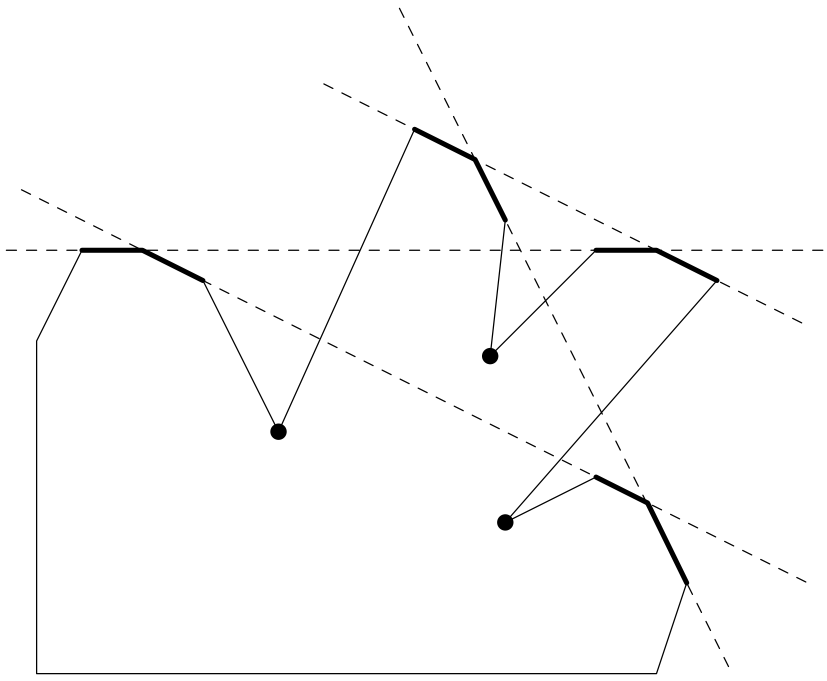


FIGURE 5.6: Concave chains of collinear edges, for which the injective-mapping-idea fails

5.4.2 Basic Properties of Nested and Interweaved Gaps

At first, recall that for a given gap $g(s,e)$ there may exist several sets $C_{i,j} \ni s, e$ such that s and e are consecutive candidate edges of an artificial edge a_i in embedding h_j (see, for example, the right part of Figure 3.9 on page 37). But in order to keep our considerations as simple as possible, we assume in the subsequent considerations w. l. o. g. that there exists *exactly one* such embedding. Then, we call h_j the *corresponding embedding* and a_i the *corresponding artificial edge*[1] of the respective gap $g(s,e)$. For this embedding h_j we prove some properties that restrict the conditions under which h_j may overlap with corresponding embeddings of other gaps. Since we are investigating an upper bound on the number of gaps (subject to the number of reflex vertices) and because additional corresponding embeddings for the same gap would only *increase* the number of required reflex vertices, this assumption surely is no oversimplification for our following reasoning.

1 Note that it directly follows from Lemma 4.17 that more than one corresponding artificial edge for a gap in a *single* embedding (i. e., $C_{i,j} \ni s, e \in C_{i',j}$ with $i \neq i'$) is not possible. Thus, each gap has an uniquely defined corresponding artificial edge in its corresponding embedding.

Lemma 5.5 *Let $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$ be two gaps that are nested or interweaved and let h_1 and h_2 be the corresponding embeddings. Then, $h_1 \neq h_2$. That is, two gaps of a single embedding can neither be nested nor be interweaved.*

Proof. Assume that $h_1 = h_2 = h$. Then, it follows from Definition 5.5 that the corresponding artificial edges a_1 and a_2 must be different, since s_1 and e_1 (as well as s_2 and e_2) are *consecutive* candidate edges. Thus, at least one of the lines of sight from inside h to s_1 and e_1 must cross one of the lines of sight from inside h to s_2 and e_2. Then, the claim directly follows from Lemma 4.17 on page 93. □

In the following we concentrate our investigations on interweaved gaps. Nested gaps will be handled later on.

Lemma 5.6 *Let $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$ be two interweaved gaps. Then, the corresponding embeddings cannot overlap.*

Proof. From the previous Lemma 5.5 we know that the corresponding embeddings are not identical. Furthermore, we conclude from Definition 5.6 that the order in which the four candidate edges s_1, e_1, s_2, and e_2 appear on the boundary of the map polygon is either s_1, s_2, e_1, e_2 or s_2, s_1, e_2, e_1. Then, the claim directly follows from Corollary 4.21 on page 104. □

The next lemma shows that in the case of two interweaved gaps (with corresponding artificial edges a_1 and a_2) the corresponding embeddings cannot have additional interweaved gaps besides those with corresponding artificial edges a_1 and a_2.

Lemma 5.7 *Let g_1 and g_2 be two interweaved gaps with corresponding embeddings h_1 and h_2 and corresponding artificial edges a_1 and a_2. Then, for any two interweaved gaps g'_1 and g'_2 with corresponding embeddings h_1 and h_2 their corresponding artificial edges must be a_1 and a_2.*

Proof. If we assume that such interweaved gaps g'_1 and g'_2 with corresponding artificial edges a'_1 and a'_2 exist, such that $a_i \neq a'_i$ for at least one $i \in \{1, 2\}$ (w. l. o. g. we assume $a_1 \neq a'_1$ and $g_2 = g'_2$), we get a situation like the one depicted in Figure 5.7 on the next page. Note that we already know from the previous Lemma 5.6 that the two embeddings h_1 and h_2 do not overlap.

Since g_1 (through a_1) as well as g'_1 (through a'_1) is interweaved with a gap belonging to h_2 ($g_2 = g'_2$ in our setting), the lines of sight from inside the embeddings (drawn dashed in the figure) cross in such a way that the map must

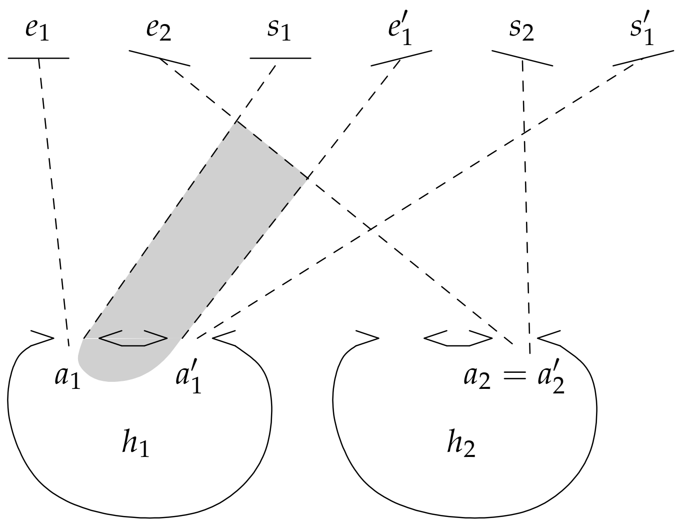


FIGURE 5.7: The gaps g_1 and g'_1 have different corresponding artificial edges

contain a hole (drawn gray), which contradicts Observation 3.3 on page 27. Thus, our first assumption must be wrong.

If $g_2 \neq g'_2$ (or $a_2 \neq a'_2$), the same argumentation can be applied with an even "larger" hole in the map polygon. □

From the preceding lemmas we conclude that the notion of interweaved gaps can be carried over to embeddings in the following straightforward manner:

- Two *embeddings* h_1 and h_2 are called *interweaved*, if there exist interweaved gaps $g_1(s_1, e_2)$ and $g_2(s_2, e_2)$ such that h_1 and h_2 are the corresponding embeddings of g_1 and g_2. (That is, there exist corresponding artificial edges a_1 and a_2 such that s_i and e_i lie on line $h_i(l_i)$ for $i = 1, 2$.) We then call g_1 and g_2 a pair of *witnesses*[2] for the interweaved embeddings; $h_1(a_1)$ and $h_2(a_2)$ are called the *interweaved artificial edges*.

- If the embeddings h_1 and h_2 are interweaved, every pair (g_1, g_2) of witnesses must have the same pair of corresponding artificial edges. (Lemma 5.7)

- Two interweaved embeddings cannot overlap. (Lemma 5.6)

2 Do not confuse these witnesses with the witnesses for an embedding defined on page 54!

- For two interweaved embeddings h_1 and h_2 there exist lines of sight from inside h_i to s_i or e_i (for $i = 1, 2$) that intersect outside of the skeleton polygons. (This follows from Lemma 5.6 and Definition 5.6.)

5.4.3 The Interweavement Graph

In order to represent only the topological structure of the interweaved embeddings, we define the *interweavement graph*, where each embedding is represented by a single node. To this end, we first inductively define for a given embedding h the *equivalence class of overlapping embeddings*, which contains all embeddings (besides h itself) that either directly overlap with h or overlap with embeddings that are equivalent to h.

Definition 5.7 (Equivalence Class of Overlapping Embeddings)
Let h_j be a valid embedding of a skeleton V^*. The ***equivalence class of overlapping embeddings*** *$\mathcal{E}^{\mathrm{ov}}(h_j)$ of h_j is inductively defined as follows:*

- $\mathcal{E}^{\mathrm{ov}}_1(h_j) := \{h_j\}$
- $\forall_{k\in\mathbb{N}}\ \mathcal{E}^{\mathrm{ov}}_{k+1}(h_j) := \mathcal{E}^{\mathrm{ov}}_k(h_j) \cup$
  $\{\, h_{j'} \colon \exists_{j''\in\mathbb{N}}\ h_{j''} \in \mathcal{E}^{\mathrm{ov}}_k(h_j)\ \wedge\ h_{j'}$ *and* $h_{j''}$ *overlap* $\}$
- $\mathcal{E}^{\mathrm{ov}}(h_j) := \bigcup_{k\in\mathbb{N}} \mathcal{E}^{\mathrm{ov}}_k(h_j)$

The set $\mathcal{E}^{\mathrm{ov}}$ consists of the equivalence classes $\mathcal{E}^{\mathrm{ov}}(h)$ of those embeddings h that actually overlap with another embedding (that is, for which $|\mathcal{E}^{\mathrm{ov}}(h)| > 1$).

It should be clear from the definition that the relation $h \sim h' :\Leftrightarrow h \in \mathcal{E}^{\mathrm{ov}}(h')$ indeed is an equivalence relation. In the same way we also define for a given embedding h_j and an artificial edge a_i the equivalence class of interweaved embeddings.

Definition 5.8 (Equivalence Class of Interweaved Embeddings)
Let h_j be a valid embedding of a skeleton V^* and let a_i be an artificial edge of V^*. The ***equivalence class of interweaved embeddings*** *$\mathcal{E}^{\mathrm{iw}}(h_j, a_i)$ is inductively defined as follows:*

- $\mathcal{E}^{\mathrm{iw}}_1(h_j, a_i) := \{(h_j, a_i)\}$
- $\forall_{k\in\mathbb{N}}\ \mathcal{E}^{\mathrm{iw}}_{k+1}(h_j, a_i) := \mathcal{E}^{\mathrm{iw}}_k(h_j, a_i) \cup$
  $\{\, (h_{j'}, a_{i'}) \colon \exists_{(j'',i'')\in\mathbb{N}^2}\ (h_{j''}, a_{i''}) \in \mathcal{E}^{\mathrm{iw}}_k(h_j, a_i)\ \wedge$
  $h_{j'}$ *and* $h_{j''}$ *are interweaved with artificial edges* $h_{j'}(a_{i'})$ *and* $h_{j''}(a_{i''})\,\}$
- $\mathcal{E}^{\mathrm{iw}}(h_j, a_i) := \bigcup_{k\in\mathbb{N}} \mathcal{E}^{\mathrm{iw}}_k(h_j, a_i)$

The set $\mathcal{E}^{\text{iw}}$ consists of the equivalence classes $\mathcal{E}^{\text{iw}}(h, a)$ of those pairs (h, a) for which h actually is interweaved with some other embedding (that is, for which $|\mathcal{E}^{\text{iw}}(h, a)| > 1$) with a being the corresponding interweaved artificial edge.

Now we are able to define the *interweavement graph* that describes the topological relationships between overlapping and interweaved embeddings, using the previous definitions.

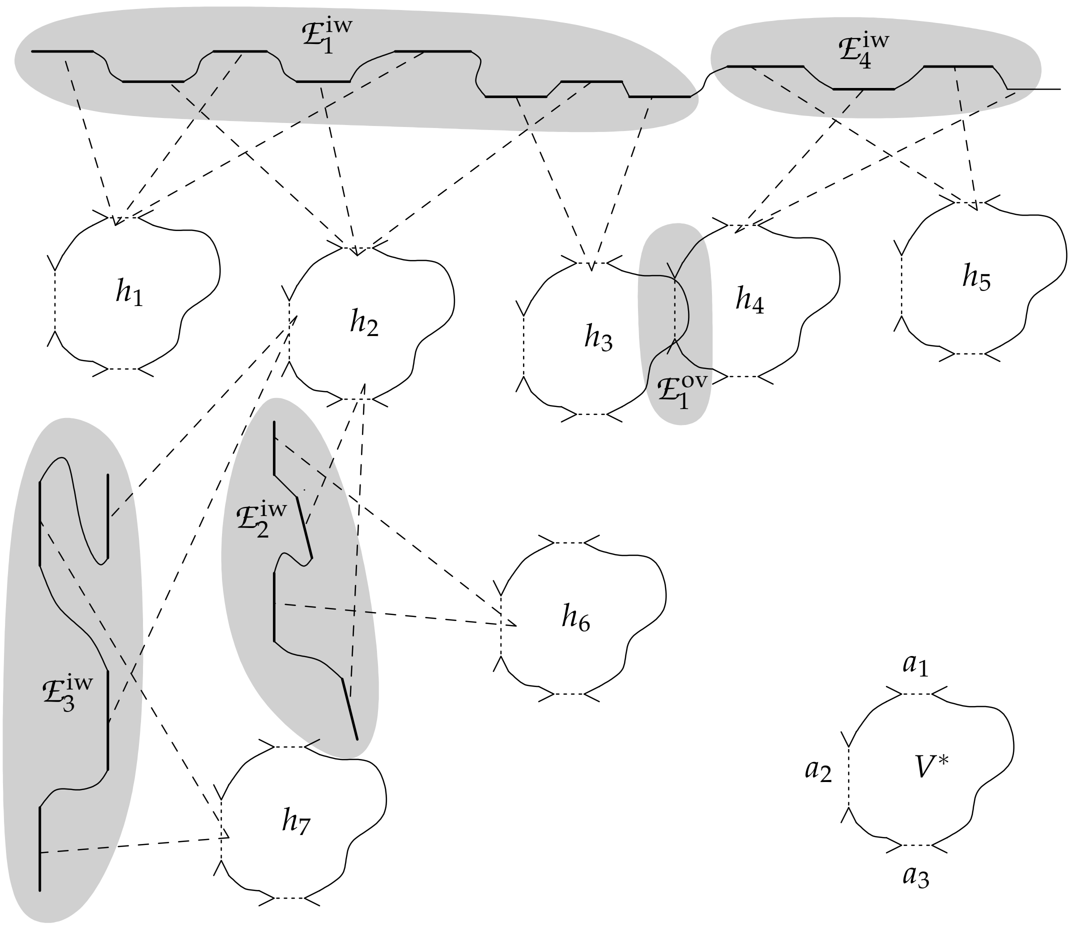
Definition 5.9 (Interweavement Graph)
*For a given skeleton V^* with t embeddings $h_1, \ldots, h_t$ the **interweavement graph** $IG(V, E)$ is an undirected bipartite graph with vertex set V partitioned into the set of **embedding nodes** $\{h_1, \ldots, h_t\}$ and the set of **special nodes** $\mathcal{E}^{\text{ov}} \cup \mathcal{E}^{\text{iw}}$. The edge set E is defined as follows:*

- *There is an edge between an embedding node h_j and a special node $\mathcal{E}_k^{\text{ov}} \in \mathcal{E}^{\text{ov}}$ if and only if $h_j \in \mathcal{E}_k^{\text{ov}}$.*
- *There is an edge between an embedding node h_j and a special node $\mathcal{E}_k^{\text{iw}} \in \mathcal{E}^{\text{iw}}$ if and only if there is an a_i such that $(h_j, a_i) \in \mathcal{E}_k^{\text{iw}}$.*

Note that embedding nodes are only connected to special nodes and vice versa. The following example illustrates the above definitions.

Example 5.2 Figure 5.8 on the next page sketches a polygonal map with seven embeddings of a skeleton V^*, which has three artificial edges a_1, a_2, and a_3. The artificial edges are drawn as dotted lines and the lines of sight from inside the embeddings to the corresponding candidate edges are drawn as dashed lines. Some of the embeddings are interweaved and some of them overlap, as described below:

- Embedding h_1 is interweaved with h_2, which itself is interweaved with h_3 (with the same corresponding artificial edge a_1). Thus, the three embeddings (each one together with its artificial edge a_1) belong to the same equivalence class $\mathcal{E}_1^{\text{iw}} = \{(h_1, a_1), (h_2, a_1), (h_3, a_1)\}$. (The equivalence classes are marked gray in the figure.)
- Furthermore, h_2 is also interweaved with h_6 (through its artificial edge a_2) and with h_7 (through its artificial edge a_3). Therefore, h_2 is further contained in the equivalence classes $\mathcal{E}_2^{\text{iw}} = \{(h_2, a_3), (h_6, a_2)\}$ and $\mathcal{E}_3^{\text{iw}} = \{(h_2, a_2), (h_7, a_2)\}$. Note that the corresponding artificial edges of interweaved embeddings need not be identical as can be seen, for example, at $\mathcal{E}_2^{\text{iw}}$, which represents the interweavement of h_2 and h_3.



$\mathcal{E}_1^{\text{iw}}$
$\mathcal{E}_4^{\text{iw}}$
h_1
h_2
h_3
h_4
h_5
$\mathcal{E}_1^{\text{ov}}$
$\mathcal{E}_2^{\text{iw}}$
$\mathcal{E}_3^{\text{iw}}$
h_6
h_7
a_1
a_2
V^*
a_3

Figure 5.8: A map with overlapping and interweaved embeddings …

- The last equivalence classes of interweaved embeddings contains h_4 and h_5, which are interweaved through the artificial edge a_1, $\mathcal{E}_4^{\text{iw}} = \{(h_4, a_1), (h_5, a_1)\}$.

- Finally, the embeddings h_3 and h_4 overlap, such that the only equivalence class of overlapping embeddings consists of these two embeddings, $\mathcal{E}_1^{\text{ov}} = \{h_3, h_4\}$. Note that the overlapping is only sketched in the figure. (For example, there must be involved at least one artificial edge of h_3, which is ignored in Figure 5.8).

The corresponding interweavement graph is depicted in Figure 5.9 on the following page. The embedding nodes are represented by bullets »•« and the special nodes by circles »∘«. ◁

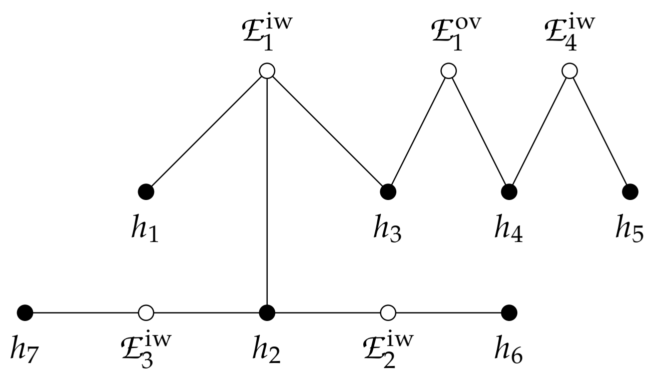

$\mathcal{E}_1^{\text{iw}}$
$\mathcal{E}_1^{\text{ov}}$
$\mathcal{E}_4^{\text{iw}}$
h_1
h_3
h_4
h_5
h_7
$\mathcal{E}_3^{\text{iw}}$
h_2
$\mathcal{E}_2^{\text{iw}}$
h_6

FIGURE 5.9: ... and the corresponding interweavement graph

In the following we state some properties of the interweavement graph.

Lemma 5.8 *The interweavement graph does not contain any circle.*

Proof. The claim follows from the properties of interweaved embeddings that we have stated above. We use the fact that for two interweaved embeddings there exist lines of sight from inside the embeddings to the involved candidate edges that intersect outside of the skeleton polygons. This way, we would be able to construct a circle of lines of sight containing at least one map vertex (which would contradict Observation 3.3 on page 27) if the embeddings are circularly interweaved (or overlapping, respectively). □

Lemma 5.9 *For an interweavement graph the following holds:*

1. *The degree of each special node is at least* 2*.*

2. *For each embedding node there is at most one edge to a special node* $\mathcal{E}_k^{\text{ov}} \in \mathcal{E}^{\text{ov}}$*.*

3. *The interweavement graph is a forest of trees with embedding nodes as leaves.*

4. *The number of edges is at most two times the number of embedding nodes.*

Proof. The proof of the four propositions is straightforward:

1. The claim directly follows from Definition 5.7, 5.8, and 5.9.

2. This also directly follows from Definition 5.7.

3. Lemma 5.8 states that an interweavement graph does not contain any circle. Thus, it must be a forest of trees. From (proposition 1) follows that a leaf of such a tree cannot be a special node, from which the claim follows.

4. The claim follows by a simple inductive argument using (proposition 1): We build up each tree of the interweavement graph starting with a single embedding node and counting the number of edges and embedding nodes, which initially is zero and one. Adding a new embedding node increases both numbers by one. Adding a new special node requires also an additional embedding node due to (proposition 1) such that the number of edges increases by two and the number of embedding nodes increases by one. □

From Lemma 5.9 (4) particularly follows that in the average case each embedding contributes at least 50 percent to each of the interweavements in which it is involved. This will be the key in the following in order to find an upper bound on the number of candidate edges.

5.4.4 Determining the Upper Bound on C_{mult}

In order to count the multiple candidate edges, we may restrict ourselves to count the edges of nested and interweaved gaps, since in the other case the injective mapping described on page 118 already proves a bound of $O(r)$ on their number. Furthermore, we additionally make the following assumptions, which will be justified later on.

Assumption 5.10 *For the subsequent considerations we assume that*

1. *there are* no overlapping embeddings *(since in our case of counting the candidate edges, the fact that two embeddings overlap can be ignored, as we will see below), that is, $\mathcal{E}^{\text{ov}} = \varnothing$,*

2. *for each interweavement $\mathcal{E}_k^{\text{iw}} \in \mathcal{E}^{\text{iw}}$ the corresponding* lines l_i *of the interweaved artificial edges a_i are pairwise* non-parallel*, and*

3. *there are* no nested gaps *(as already stated above, this case will be handled separately).*

Using our above considerations on interweaved gaps, we are now able to cope with the problem of concave chains of candidate edges, which we have mentioned on page 119. The following Lemma 5.11 uses some results on the maximum complexity of m disjoint concave chains in an arrangement of

n pseudo lines by Halperin and Sharir [HS91, HS94b]. Similar bounds were proven by Hershberger and Snoeyink [HS98] investigating the maximum complexity of m faces in an *erased arrangement* of n lines. We cite the theorem of Halperin and Sharir, since it plays a crucial role in the proof of Lemma 5.11:

Theorem 5.10 *Let $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ be a collection of n* pseudo lines*, each pair of which intersect at most once. Let $\mathcal{A} = \mathcal{A}(\mathcal{L})$ be the arrangement of the pseudo lines. A* concave chain *c in $\mathcal{A}$ is a connected path in the union of the pseudo lines of $\mathcal{L}$, such that as we traverse c from left to right, whenever we reach a vertex of $\mathcal{A}$, we can either continue along the pseudo line we are currently on, or make a right turn onto the other pseudo line, but we cannot make a left turn.*

Then, the maximum joint combinatorial complexity of m disjoint concave chains in an arrangement of n pseudo lines is

$$\Theta\left(m^{2/3} \cdot n^{2/3} + n\right) . \tag{5.5}$$

Now we state the lemma, which limits the number of candidate edges in each single interweavement, such that our desired bound directly follows.

Lemma 5.11 *Under Assumption 5.10 the following holds: For each interweavement $\mathcal{E}_k^{\mathrm{iw}} \in \mathcal{E}^{\mathrm{iw}}$ we can assign r_k reflex vertices to $\mathcal{E}_k^{\mathrm{iw}}$ such that*

- *the number of (multiple) candidate edges, which is involved in interweavement $\mathcal{E}_k^{\mathrm{iw}} = \{(h_{j_1}, a_{i_1}), \ldots, (h_{j_u}, a_{i_u})\}$ is in $O(r_k)$, that is, $c_{i_1,j_1} + \cdots + c_{i_u,j_u} \in O(r_k)$, and*
- *each reflex vertex is assigned to at most $O(1)$ interweavements (that is, in particular $\sum_k r_k \in O(r)$ holds).*

Proof. We consider an interweavement $\mathcal{E}_k^{\mathrm{iw}} = \{(h_{j_1}, a_{i_1}), \ldots, (h_{j_u}, a_{i_u})\}$ with u involved embeddings (that is, the corresponding special node in the interweavement graph has degree u).[3]

From Assumption 5.10 (2) follows that the u artificial edges must be pairwise different, since no two of their corresponding lines $l_{i_1}, \ldots, l_{i_u}$ can be identical. Furthermore, from Lemma 5.6 we know that no two of the embeddings $h_{j_1}, \ldots, h_{j_u}$ overlap. Thus, each one of the u artificial edges has its own pair of map vertices in each one of the u embeddings, therefore requiring a total of $\Omega(u^2)$ reflex vertices (in the u disjoint embeddings).

3 To be correct, we should use u_k for the number of involved embeddings, since this number of course depends on the respective interweavement. But in order to avoid triple-indices, we accept this little abuse of notation.

From Lemma 5.9 (4) we then conclude that for the u edges incident to $\mathcal{E}_k^{\mathrm{iw}}$ in the interweavement graph we can determine $\Omega(u)$ embeddings (namely $u/2$ embeddings), which can be *exclusively* assigned to $\mathcal{E}_k^{\mathrm{iw}}$. Thus, we can find a first set of reflex vertices of size $\bar{r}_k \in \Omega(u^2)$, which fulfills the requirement from above that each reflex vertex is assigned to at most $O(1)$ interweavements. Consequently, the number u of involved embeddings is then in $O(\bar{r}_k^{1/2})$.

In the following we apply the result of Halperin and Sharir cited above to our setting. We consider the arrangement of the u corresponding lines $h_{j_1}(l_{i_1}), \ldots, h_{j_u}(l_{i_u})$, on which the multiple candidate edges lie. The part of the map boundary that corresponds to $\mathcal{E}_k^{\mathrm{iw}}$ can then be seen as a collection of disjoint concave chains of candidate edges (and possibly also of non-candidate edges, but this plays no role here) in the arrangement of the u lines. Between two consecutive chains there lies at least one reflex vertex, which also can be *exclusively* assigned to $\mathcal{E}_k^{\mathrm{iw}}$. (This follows directly from the construction of the equivalence class of interweaved embeddings, cf. Definition 5.8). Thus, we can find a second set of reflex vertices with cardinality $\bar{\bar{r}}_k$, which also fulfills the requirement from above and whose cardinality is of the same order of magnitude as the number of concave chains.

Then, due to Theorem 5.10 on the facing page the maximum number of candidate edges that lie on the lines $h_{j_1}(l_{i_1}), \ldots, h_{j_u}(l_{i_u})$, is in

$$\Theta\left(\bar{\bar{r}}_k^{2/3} \cdot \left(\bar{r}_k^{1/2}\right)^{2/3} + \bar{r}_k^{1/2}\right) . \tag{5.6}$$

Using $r_k := \max\{\bar{r}_k, \bar{\bar{r}}_k\}$ this can be rewritten as

$$O\left(r_k^{2/3} \cdot r_k^{1/3} + r_k^{1/2}\right) = O(r_k) ,$$

which completes the proof. □

As already stated above, from the previous lemma an upper bound of $O(r)$ on the total number of multiple candidate edges directly follows by summing up over all interweavements $\mathcal{E}_k^{\mathrm{iw}}$ and using the fact that $\sum_k r_k \in O(r)$. So, in order to prove this bound also for the general case, we only have to show that Assumption 5.10 is not too restrictive. This is done in the following for each of the three assumptions.

No overlapping embeddings – Assumption 5.10 (1)

In the case of overlapping embeddings in the map, let $\mathcal{E}_k^{\mathrm{ov}} = \{h_{j_1}, \ldots, h_{j_u}\}$ be such an equivalence class of overlapping embeddings. We now reduce this case to a non-overlapping situation by substituting the embeddings $h_{j_1}, \ldots, h_{j_u}$ in the interweavement graph by a *single virtual embedding* $\tilde{h}_k$, as depicted in

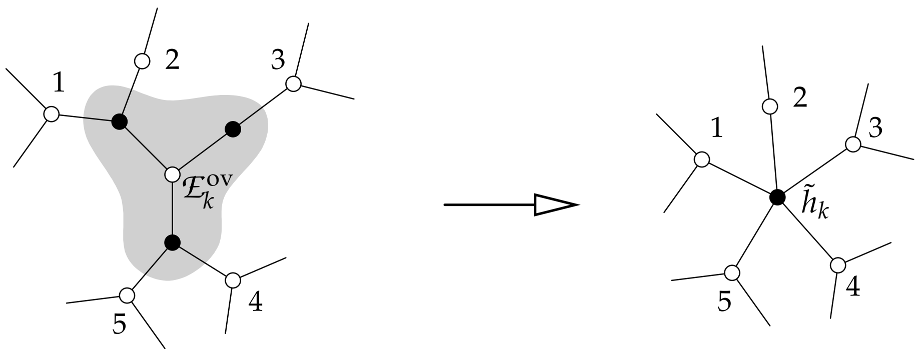


FIGURE 5.10: Substituting an equivalence class of overlapping embeddings in the interweavement graph

Figure 5.10. Note that Lemma 5.9 (2) states that all embeddings $h_{j_1}, \ldots, h_{j_u}$ are connected to only *one* equivalence class of overlapping embeddings (i. e., to $\mathcal{E}_k^{\mathrm{ov}}$). Thus, this substitution process poses no problems in the interweavement graph. Furthermore, the substitution also does not influence the averaging argument of Lemma 5.9 (4), since the resulting interweavement graph has the same properties as the original one.

In other words, we regard a set of overlapping embeddings as a new object,[4] which is interweaved with the same embeddings as the original ones, but count it only as one embedding in order to determine the number of reflex vertices. Thus, we safely *under*estimate the number of reflex vertices, in order to find an *upper* bound on the number of candidate edges.

No parallel lines l_i – Assumption 5.10 (2)

The problem of parallel corresponding lines l_i for an interweavement $\mathcal{E}_k^{\mathrm{iw}} \in \mathcal{E}^{\mathrm{iw}}$ can be circumvented in a similar way as the previous problem, namely by a suitable transformation of the map and the interweavement graph, respectively. To this end, assume an interweavement $\mathcal{E}_k^{\mathrm{iw}} = \{(h_{j_1}, a_{i_1}), \ldots, (h_{j_u}, a_{i_u})\}$ such that a subset $h_{\tilde{\jmath}_1}(l_{\tilde{\imath}_1}), \ldots, h_{\tilde{\jmath}_{\tilde{u}}}(l_{\tilde{\imath}_{\tilde{u}}})$ of the corresponding lines is parallel. Then we transform the map in the following way:

Let w. l. o. g. the lines $h_{\tilde{\jmath}_1}(l_{\tilde{\imath}_1}), \ldots, h_{\tilde{\jmath}_{\tilde{u}}}(l_{\tilde{\imath}_{\tilde{u}}})$ be horizontal such that the corresponding embedded skeletons are placed below and let $h_{\tilde{\jmath}_1}(l_{\tilde{\imath}_1})$ be the topmost line. Then, we first shift all candidate edges, which lie on one of the other lines $h_{\tilde{\jmath}_2}(l_{\tilde{\imath}_2}), \ldots, h_{\tilde{\jmath}_{\tilde{u}}}(l_{\tilde{\imath}_{\tilde{u}}})$ onto $h_{\tilde{\jmath}_1}(l_{\tilde{\imath}_1})$, as depicted in Figure 5.11 on the next page. Note that this transformation does *not* introduce any new reflex vertex. Furthermore, we additionally eliminate the edges between $\mathcal{E}_k^{\mathrm{iw}}$ and

4 The notion *virtual embedding* may be a bit misleading, since this object surely is no embedding of V^*.

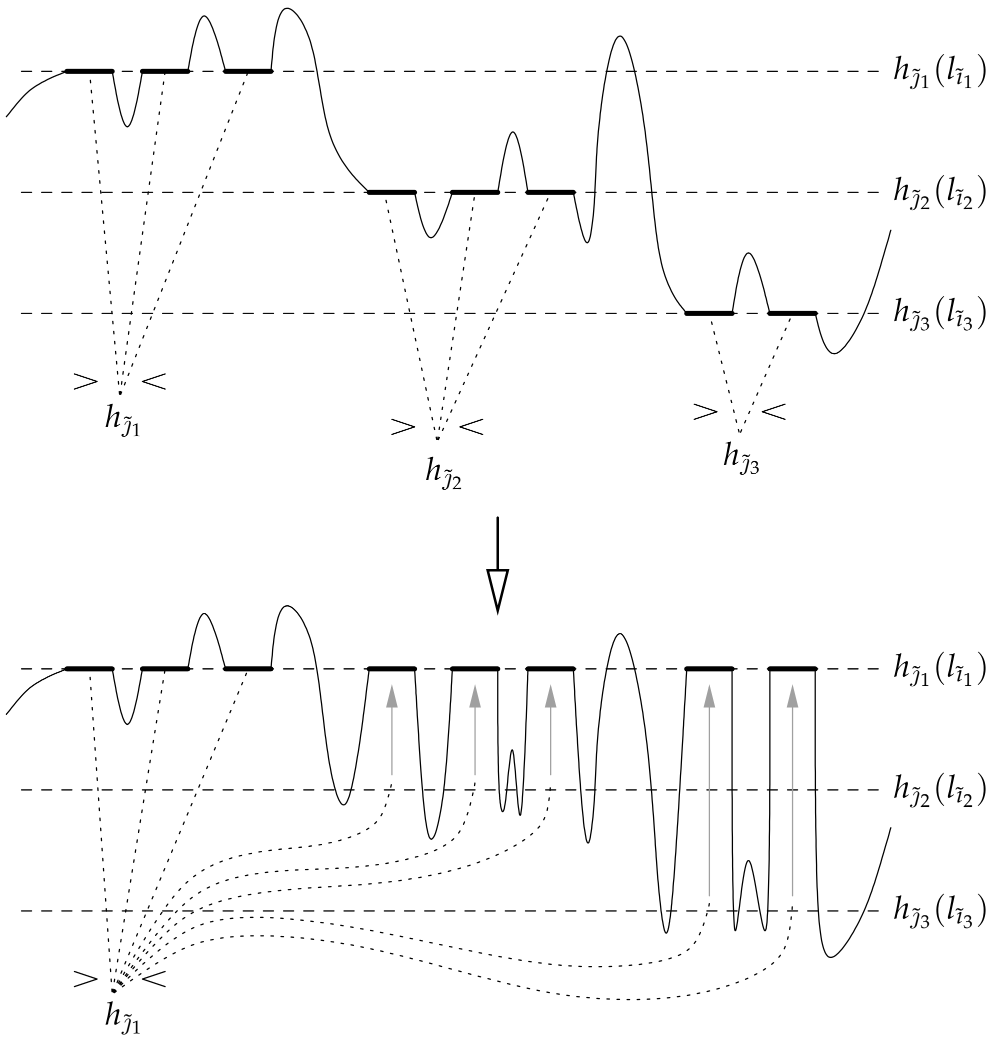

$h_{\tilde{\jmath}_1}(l_{\tilde{\imath}_1})$
$h_{\tilde{\jmath}_2}(l_{\tilde{\imath}_2})$
$h_{\tilde{\jmath}_3}(l_{\tilde{\imath}_3})$
$>$ $<$
$h_{\tilde{\jmath}_1}$
$h_{\tilde{\jmath}_2}$
$h_{\tilde{\jmath}_3}$

FIGURE 5.11: Transforming the map to avoid parallel lines l_i

$(h_{\tilde{\jmath}_2}, a_{\tilde{\imath}_2}), \dots, (h_{\tilde{\jmath}_{\tilde{u}}}, a_{\tilde{\imath}_{\tilde{u}}})$ and assign the transformed candidate edges to embedding $h_{\tilde{\jmath}_1}$. (If $h_{\tilde{\jmath}_1}$ is the only embedding that remains we could even eliminate the whole interweavement $\mathcal{E}_k^{\mathrm{iw}}$.)

In other words, in the case of parallel lines we choose *one representative* embedding, to which we assign the corresponding candidate edges, and simply ignore the remaining embeddings. This way, the number of candidate edges is not changed, but the number of involved embeddings (and therefore, the number of reflex vertices) is *decreased* by our transformation.

No nested gaps – Assumption 5.10 (3)
The problem that we have with nested gaps (in contrast to interweaved gaps) is that the corresponding embeddings could overlap (cf. Lemma 5.6 on page 121 where this property was disproved for interweaved gaps). Therefore, we have to differentiate between nested gaps with overlapping and non-overlapping embeddings:

Nested gaps with non-overlapping embeddings If the corresponding embeddings do not overlap, we get the same situation as with interweaved gaps and can handle it in exactly the same way. In particular, lines of sight from inside the embeddings to the corresponding candidate edges, which cross outside of the embeddings, exist also for the case of nested gaps. Thus, all the claims about interweaved gaps and embeddings (e. g., Lemma 5.7 and the subsequent ones) can be transferred and analogously shown for nested gaps.

Nested gaps with overlapping embeddings If we assume two gaps $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$ with corresponding artificial edges a_1 and a_2 and corresponding overlapping embeddings h_1 and h_2 such that g_2 is nested into g_1, the situation must be like the one shown in Figure 5.12 on the next page.

In particular, the artificial edge $h_2(a_2)$ must lie *between* the lines of sight from h_1 to s_1 and e_1, since otherwise the two embeddings could not overlap without creating a hole in the map. Thus, we conclude that $h_1(a_1)$ must intersect $h_2(V^\diamond)$, since this is the only possible way of an overlapping between h_1 and h_2 (without creating a hole). From this situation it follows that another embedding h_3 that is interweaved with h_2 can neither be interweaved with h_1 nor can it overlap h_1. That is, all corresponding gaps of h_3 must be nested into g_1.

Thus, we can safely ignore the nested gaps $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$ and simply regard the two embeddings h_1 and h_2 as overlapping embeddings, which then can be handled as described above. This way, we do not count the nested gaps (or candidate edges, respectively) $g_1(s_1, e_1)$ and $g_2(s_2, e_2)$, but their total number is limited by the maximum number of embeddings, that is, it must be in $O(r)$.

Using the above arguments, which justify Assumption 5.10, we can combine Lemma 5.4 and Lemma 5.11 in order to get our desired bound on the number of multiple candidate edges:

Theorem 5.12 *For a skeleton V^* the total number $|C_{\text{mult}}|$ of multiple candidate edges is in $O(r)$.*

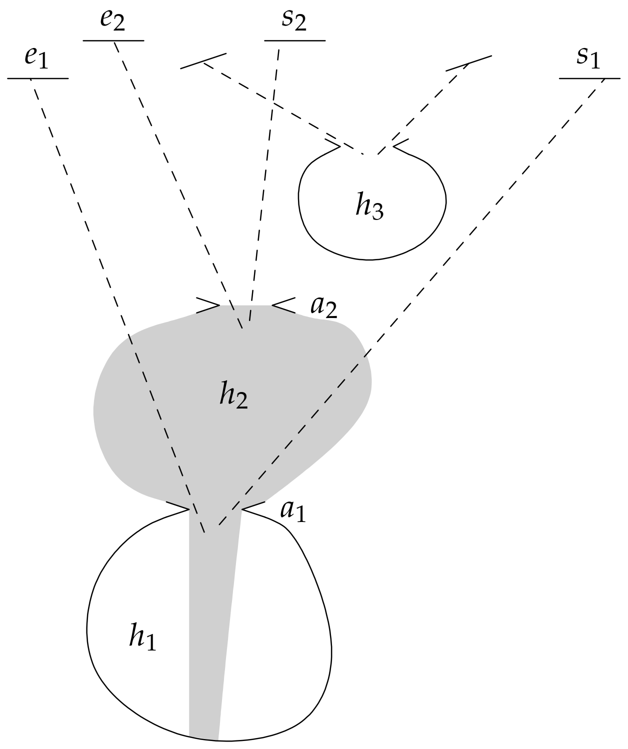


FIGURE 5.12: Nested gaps with overlapping corresponding embeddings

5.5 A Result on Pairs of Multiple Candidate Edges

We have already seen in the preceding sections that multiple candidate edges play a crucial role in determining the worst-case complexity $|\mathcal{EC}_{V^*}|$ of an equivalence class. Moreover, in the subsequent section it will come out that in particular we have to count the number of *pairs* of multiple candidate edges. To this end, we show in this section that a given pair (e, f) of map edges may be a pair of *multiple* candidate edges in at most a constant number of embeddings.

The following two lemmas first investigate pairs of candidate edges in many embeddings and show that at least a constant fraction of these embeddings must overlap.

Lemma 5.13 *Let e and f be two map edges and let $h_{j_1}, \ldots, h_{j_k}$ be a number of $k \geq 2$ embeddings such that for each h_{j_u}, $1 \leq u \leq k$, the following holds:*

- *e is a valid candidate edge of some artificial edge $h_{j_u}(a_{i_u})$*
- *f is a valid candidate edge of some artificial edge $h_{j_u}(\tilde{a}_{i_u})$ with $\tilde{a}_{i_u} \neq a_{i_u}$*

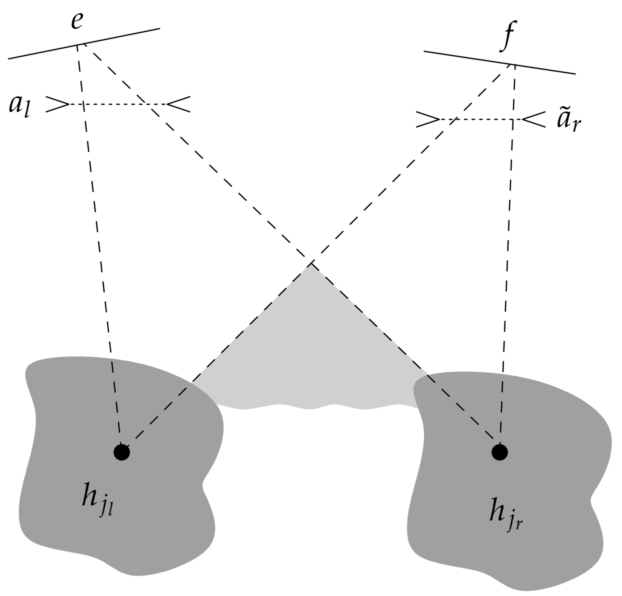


FIGURE 5.13: A pair of candidate edges visible from two embeddings

Then, there exist two embeddings h_{j_l} and h_{j_r} and artificial edges a_l and $\tilde{a}_r$ such that for all h_{j_u}, $1 \leq u \leq k$,

- *the lines of sight from inside h_{j_u} to e cross $h_{j_l}(a_l)$, and*
- *the lines of sight from inside h_{j_u} to f cross $h_{j_r}(\tilde{a}_r)$.*

That is, all lines of sight to e (or to f, respectively) pass through the same artificial edges $h_{j_l}(a_l)$ or $h_{j_r}(\tilde{a}_r)$, respectively.

Proof. Without loss of generality let e and f be horizontal, directed from right to left and placed in the map such that e lies to the left of f (see Figure 5.13). Note that for two different embeddings the lines of sight to e and f must intersect as depicted in the figure, since otherwise the two embeddings would intersect and encircle at least one map vertex (which would contradict Observation 3.3) due to Theorem 4.2.

Now let h_{j_l} be the embedding with the *leftmost* line of sight to e and let h_{j_r} be the embedding with the *rightmost* line of sight to f. Furthermore, let a_l be the artificial edge of h_{j_l} that belongs to e and let $\tilde{a}_r$ be the artificial edge of h_{j_r} that belongs to f. Then, the right vertex of $h_{j_l}(a_l)$ cannot be placed below the line of sight from h_{j_l} to f (that is, in the light-gray region in the figure), because otherwise also the line of sight to f would cross $h_{j_l}(a_l)$. Thus, it must be placed above this line of sight and *to the right* of all others lines of sight from to e, since

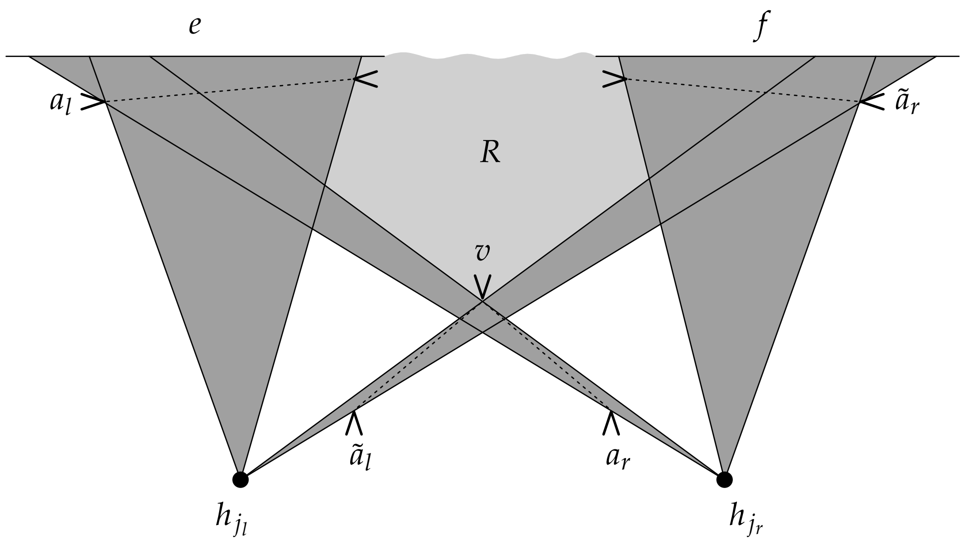


FIGURE 5.14: The one exceptional case where h_{j_l} and h_{j_r} do not overlap

in the other case a hole in the map would have been created. Therefore, all lines of sight to e pass through the artificial edge $h_{j_l}(a_l)$. Analogously, we show that all lines of sight to f pass through $h_{j_r}(\tilde{a}_r)$. □

Lemma 5.14 *Let e and f be two map edges and let $h_{j_1}, \dots, h_{j_k}$ be a number of $k \geq 4$ embeddings such that for each h_{j_u}, $1 \leq u \leq k$, the following holds:*

- *e is a valid candidate edge of some artificial edge $h_{j_u}(a_{i_u})$*
- *f is a valid candidate edge of some artificial edge $h_{j_u}(\tilde{a}_{i_u})$ with $\tilde{a}_{i_u} \neq a_{i_u}$*

Then, at least $\lceil k/3 \rceil$ of the embeddings pairwise overlap.

Proof. Let h_{j_l} and h_{j_r} be the embeddings claimed by Lemma 5.13. We now conclude for $1 \leq u \leq k$ that the *right* vertex of the artificial edge $h_{j_u}(a_{i_u})$, that is, the artificial edge corresponding to the *left* candidate edge e, and the *left* vertex of $h_{j_u}(\tilde{a}_{i_u})$, that is, the artificial edge corresponding to f, must lie in the same region R as the right vertex of $h_{j_r}(a_r)$ and the left vertex of $h_{j_l}(\tilde{a}_l)$, bounded by the map and by the lines of sight from h_{j_r} to e and from h_{j_l} to f. See Figure 5.14 where this situation is sketched (including the view cones of the witnesses); the respective region R is drawn in light-gray.

Note that we cannot conclude where the respective other vertices of the above artificial edges have to lie (that is, the left vertices of the a_{i_u} and the right vertices of the $\tilde{a}_{i_u}$), but in all cases (except for one, see below), the two embeddings overlap, since $h_{j_l}(\tilde{a}_l)$ intersects $h_{j_r}(V^\diamond)$ and $h_{j_r}(a_r)$ intersects $h_{j_l}(V^\diamond)$. The

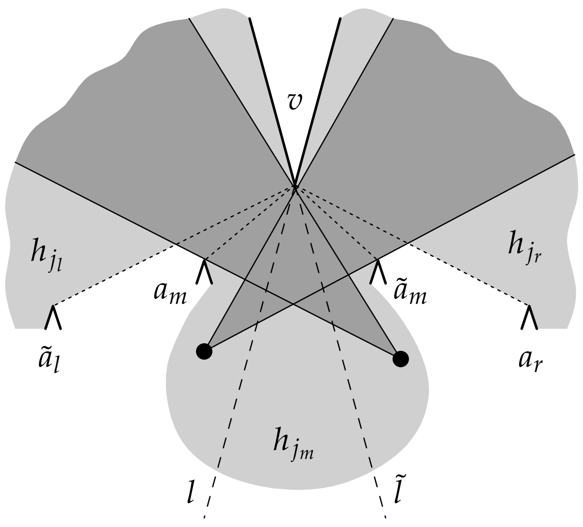


FIGURE 5.15: The case of three non-overlapping embeddings h_{j_l}, h_{j_m} and h_{j_r}

one exception occurs if the lef vertex of $h_{j_l}(\tilde{a}_l)$ and the right vertex of $h_{j_r}(a_r)$ are identical. This is depicted in Figure 5.14, where their common vertex is denoted by v.

Now we assume another embedding h_{j_m}, $1 \leq m \leq k$. It is clear that if h_{j_l} and h_{j_r} overlap, the embedding h_{j_m} must overlap *both* of them, since each straight line into R "from below" crosses a view cone of h_{j_l} as well as a view cone of h_{j_r}. Thus, we can restrict our considerations to the exceptional case shown in Figure 5.14 where the two artificial edges $h_{j_l}(\tilde{a}_l)$ and $h_{j_r}(a_r)$ have a common vertex v. In order to achieve that also these *three* embeddings do not overlap, the vertex v additionally must be the right vertex of $h_{j_m}(a_m)$ *and* the left vertex of $h_{j_m}(\tilde{a}_m)$. We then get the situation sketched in Figure 5.15, which focuses on the region around vertex v. The view cones through the artificial edges $h_{j_m}(a_m)$ and $h_{j_m}(\tilde{a}_m)$ are drawn dark-gray and the three embeddings h_{j_l}, h_{j_m}, and h_{j_r} are drawn light-gray.

The lines that are induced by the two incident edges of v are denoted by l and $\tilde{l}$. Note that the *left* artificial edge $h_{j_m}(a_m)$ must lie to the left of l and that the *right* artificial edge $h_{j_m}(\tilde{a}_m)$ must lie to the right of $\tilde{l}$. Thus, any additional embedding, which does overlap neither h_{j_l} nor h_{j_r} must be positioned in the same way as h_{j_m} and consequently overlap h_{j_m}.

Therefore, any additional embedding must overlap at least one of the three embeddings h_{j_l}, h_{j_m} or h_{j_r}. By the above construction (cf. Figure 5.14) it further follows that if such an embedding overlaps h_{j_x} ($x = l, m, r$), it also overlaps all embeddings that overlap h_{j_x}. From this, the claim follows using the pigeon hole principle. □

We now use the result of the previous lemma in order to show that for a given pair (e, f) of map edges the number of embeddings, for which (e, f) is a pair of *multiple* candidate edges, is limited by a constant.

Theorem 5.15 *Let e and f be two map edges and let $h_{j_1}, \ldots, h_{j_k}$ be a number of k embeddings such that for each h_{j_u}, $1 \leq u \leq k$, the following holds:*

- *e is a valid candidate edge of some artificial edge $h_{j_u}(a_{i_u})$*
- *f is a valid* multiple *candidate edge of some artificial edge $h_{j_u}(\tilde{a}_{i_u})$ with $\tilde{a}_{i_u} \neq a_{i_u}$*

Then, k is at most six, that is, $k \leq 6$.

Proof. We assume that $k \geq 7$ and conclude from the preceding Lemma 5.14 that in this case at least three embeddings h_{j_1}, h_{j_2}, and h_{j_3} exist that overlap. Since f is a multiple candidate edge in each of the three embeddings, there must exist candidate edges f_1, f_2, f_3 such that $f, f_u \in C_{i_u, j_u}$ for $1 \leq u \leq 3$, all lying on the same line l. Without loss of generality we assume that l is horizontal and directed from right to left. It follows that at least two of the candidate edges f_1, f_2, f_3 must lie on the same side of f on l. Without loss of generality we assume that the edges f_1 and f_2 lie to the right of f. For the following considerations let $h := h_{j_1}$ and $h' := h_{j_2}$. Furthermore, let w_1 and w_2 be the witnesses of $\gg f$ and f_1 are valid candidate edges in $h \ll$ and let let w'_1 and w'_2 be the witnesses of $\gg f$ and f_2 are valid candidate edges in $h' \ll$. Then, we get a situation as depicted in Figure 5.16 on the next page.

Now observe the following:

1. Since f_1 and f_2 lie on the same side of f, a straight line s exists (as depicted in the figure) that separates the witness w_1 from w_2, as well as it separates the witness w'_1 from w'_2. (The gray regions in the figure denote the visibility wedges of the candidate edges f and f_1/f_2 with respect to the artificial edge $h_{j_r}(\tilde{a}_r)$ from Lemma 5.13.)

2. From each of the four witnesses exist lines of sight to the candidate edges e and f. Moreover, w_1 and w_2 see each other, as well as w'_1 and w'_2 do.

3. The lines of sight between w_1 and w'_i $(i = 1, 2)$ must be blocked by map vertices due to Theorem 4.2, as well as the lines of sight between w_2 and w'_i $(i = 1, 2)$.

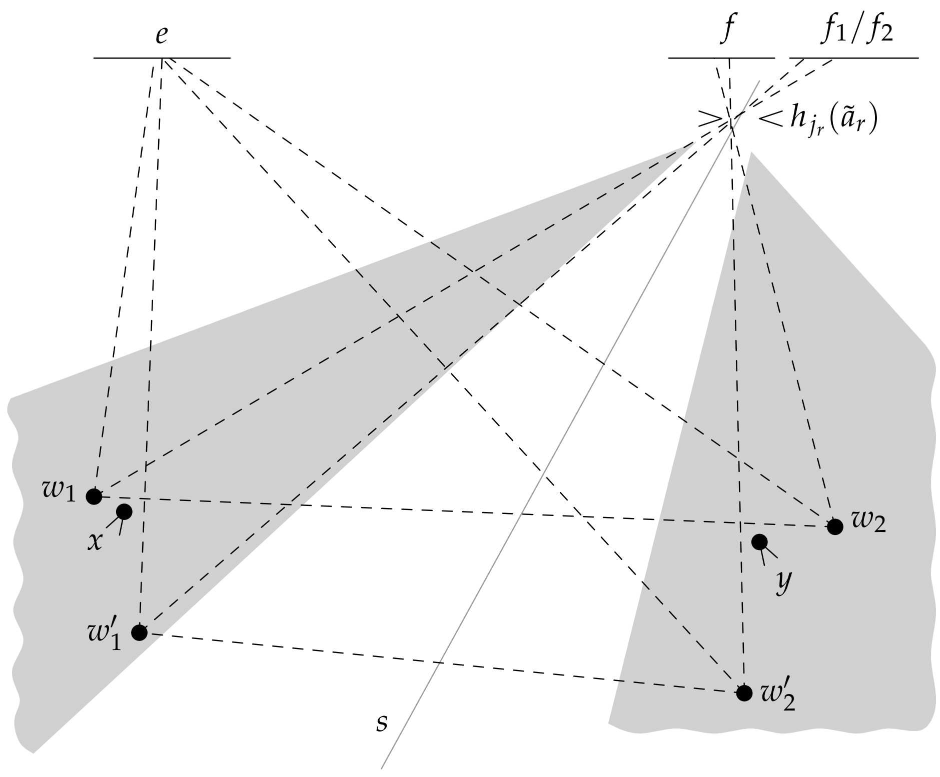


FIGURE 5.16: The placement of the four witnesses w_1, w_2, w'_1, and w'_2

4. From (proposition 1) to (3) follows that the line of sight between w_1 and w_2 and the line of sight between w'_1 and w'_2 cannot intersect, since otherwise the map would contain a hole.

5. With the same reasoning as in (4) we conclude that no one of the triangles, which consist of two corresponding witnesses (that is, either w_1 and w_2 or w'_1 and w'_2) and a point on e or f, could contain one of the two remaining witnesses (cf. Figure 5.16).

6. Then, we finally conclude from (proposition 4) and (5) that the four witnesses must be placed as depicted in Figure 5.16, that is, one pair of witnesses lies "below" the remaining pair. Without loss of generality we assume that w'_1 and w'_2 lie below w_1 and w_2. Furthermore, let x and y denote the blocking vertices between w_i and $w'_{i'}$.

Now compare the placement of the witnesses w_1, w_2, and w'_2 with the situation in the proof of Theorem 4.19. Although we have no *crossing* candidate edges here (since f_1 and f_2 need not be different map edges), the situation is absolutely identical to the one in Theorem 4.19. In particular, we could even copy the proof (starting at page 102) almost one-to-one, in order to show, that also the placement of the witnesses w_1, w_2, and w_2' leads to a contradiction.

Thus, in the same way as in the proof of Theorem 4.19 we show that the map polygon must contain a hole and conclude that our initial assumption »$k \geq 7$« must be wrong. □

This result, that the number of *pairs* of multiple candidate (for a given pair of map edges) is bounded by a constant, is profitably applied in the following section.

5.6 Putting it Together –The Bound on the Number of Wedge Edges

Now we are able to use the results of the preceding sections, in order to bound the total number of wedge edges of an equivalence class.

Theorem 5.16 *For a given skeleton V^* the total number of wedge edges in $\mathcal{EC}_{V^*}$ is in $O(r^2)$.*

Proof. Equation (5.1) from Lemma 5.2 on page 111 describes the number of wedge edges in a *single* embedding h_j. Summing up (and taking advantage of Lemma 3.7), we yield Equation (5.4) from page 114

$$O\left(r^2 + \sum_{j=1}^{t}\left(\left(\sum_{i=1}^{s}(c_{i,j}-1)\right)^2 - \sum_{i=1}^{s}(c_{i,j}-1)^2\right)\right) . \tag{5.4}$$

Using the following equality

$$\left(\sum_{i=1}^{s}(c_{i,j}-1)\right)^2 - \sum_{i=1}^{s}(c_{i,j}-1)^2 = 2\sum_{1\leq i_1<i_2\leq s}(c_{i_1,j}-1)\,(c_{i_2,j}-1) \;,$$

Equation (5.4) can be rewritten to

$$O\left(r^2 + \sum_{j=1}^{t}\sum_{1\leq i_1<i_2\leq s}(c_{i_1,j}-1)\,(c_{i_2,j}-1)\right) . \tag{5.7}$$

For the double sum in Equation (5.7) we further conclude

$$\sum_{j=1}^{t}\sum_{1\leq i_1<i_2\leq s}(c_{i_1,j}-1)\,(c_{i_2,j}-1) \leq \sum_{j=1}^{t}\sum_{\substack{1\leq i_1<i_2\leq s\\ |C_{i_1,j}|,|C_{i_2,j}|>1}}\left|C_{i_1,j}\times C_{i_2,j}\right| \;,$$

since $c_{i_1,j}, c_{i_2,j} \geq 1$, and using the fact that for a single embedding h_j the sets $C_{i,j}$ of valid candidate edges must be pairwise disjoint (cf. Lemma 4.17) we get

$$= \sum_{j=1}^{t} \left| \bigcup_{\substack{1 \leq i_1 < i_2 \leq s \\ |C_{i_1,j}|, |C_{i_2,j}| > 1}} C_{i_1,j} \times C_{i_2,j} \right| . \tag{5.8}$$

Now observe that the union in Equation (5.8) consists of pairs of *multiple* candidate edges (because of $|C_{i_1,j}|, |C_{i_2,j}| > 1$), such that we can apply Theorem 5.15 as follows

$$\leq 6 \cdot \left| \bigcup_{j=1}^{t} \bigcup_{\substack{1 \leq i_1 < i_2 \leq s \\ |C_{i_1,j}|, |C_{i_2,j}| > 1}} C_{i_1,j} \times C_{i_2,j} \right|$$
$$\leq 6 \cdot |C_{\text{mult}}|^2 . \tag{5.9}$$

Thus, plugging Equation (5.9) into Equation (5.7) and applying Theorem 5.12, which bounds the total number of multiple candidate edges, we conclude that the total number of wedge edges is in $\mathcal{EC}_{V^*}$ is in $O(r^2)$. □

Using the previous result together with Lemma 5.1 on page 110, we immediately get the following claim about the total complexity of an equivalence class $\mathcal{EC}_{V^*}$.

Theorem 5.17 *The total number of cells in any equivalence class $\mathcal{EC}_{V^*}$ (as well as their total complexity) is in $O(n + r^2)$.*

This improves the $O(n^2)$ bound of Theorem 3.10 on page 47 in the sense that the quadratic term now depends only on the number of *reflex* vertices and not on the total number of map vertices.

Referring to Equation (3.2) we analyze the preprocessing costs analogously to Theorem 3.11 on page 49 and yield the succeeding corollary.

Corollary 5.18 *The preprocessing of a polygonal map $\mathcal{M}$ with n vertices in total, hereof r reflex vertices, using Algorithm 3.14 has a time and space complexity of $O(N(n + r^2))$, where $N \in O(n^2 r)$ describes the number of visibility cells of $\mathcal{M}$.*

5.7 Summary

The main goal of this chapter was to establish a refined bound on the worst-case complexity $|\mathcal{EC}_{V^*}|$ of an equivalence class compared to the $O(n^2)$ result of Guibas et al. (see Theorem 3.10). To this end, we first investigated the structure of the visibility cells of $\mathcal{EC}_{V^*}$ in detail and reduced the problem of determining $|\mathcal{EC}_{V^*}|$ to the problem of counting the wedge edges of $\mathcal{EC}_{V^*}$. An accurately upper bound on their number in a single embedding was given in Lemma 5.2.

We further showed that the number of wedge edges heavily depends on the number of single and multiple candidate edges. For the total number of single candidate edges we showed that the trivial upper bound of $O(\min\{n, r^2\})$ is tight and gave a worst-case example (Lemma 5.3). In order to establish also a tight upper bound on the number of multiple candidate edges, we further investigated the gaps between the multiple candidate edges, which can be nested or interweaved in the worst-case, such that all known simple counting strategies fail.

Thus, we introduced the interweavement graph that abstracts from the geometric properties of the setting and only represents the topological structure of the interweaved (and overlapping) embeddings. We proved several properties about interweaved, overlapping, and nested embeddings and finally established a tight upper bound of $O(r)$ on the total number of multiple candidate edges.

Moreover, using the results of Chapter 4 we proved a claim about pairs of multiple candidate edges (Lemma 5.13), which states that any two map edges may be a pair of candidate edges in at most a constant number of embeddings. Using this result and the previously established bound on the number of candidate edges, we finally proved our main theorems on the total number of wedge edges and the total complexity of an equivalence class $\mathcal{EC}_{V^*}$ (Theorem 5.16 and 5.17).

6

Realistic Scenarios

Far down the giant hallway,
almost as tiny as Danny himself,
was a dark figure. Tony.
*"**Where am I?**" he called softly to Tony.*
"Sleeping," Tony said.
"Sleeping in your mommy and daddy's bedroom."
There was sadness in Tony's voice.

Danny in "The Shining"
STEPHEN KING (∗ 1947)

THE PRECEDING CHAPTERS dealt with the robot localization problem by introducing some restrictive and simplifying assumptions (cf. Assumption 3.1 on page 24). This way, we obtained an idealized version of the problem, which was analyzed and solved using methods from the field of computational geometry. In the following we return to the original real-world problem of localizing a robot, which is equipped with a range sensor (e. g., a laser scanner) and which was dropped somewhere in its environment.

The objective of this chapter is to adapt the solution of the idealized problem (shown in Chapter 3) to the properties of real-world environments such that the restrictive assumptions can be loosened. In order to do this, we first discuss in the subsequent section the problems that occur if we try to use the computational geometry solution of the previous chapters for realistic scenarios. We then introduce the concept of *distance functions*, which model the resemblance between the sensor data and the map, and show how we may use this concept to circumvent some of the problems illustrated before.

Moreover, we introduce a distance function that seems to be well suited for the localization problem, the *polar coordinate metric*, and show how this distance can be used in our framework. Finally, we present our implementation ROLOPRO, where the idealizing scheme of Chapter 3 as well as our approach for realistic scenarios was implemented.

6.1 Problems in Realistic Scenarios

In the following we compare the idealizing assumptions of the previous chapters with the characteristics of real-world scenarios and study how the computational geometry solution would behave in such scenarios and what problems would occur, respectively.

Exact visibility polygon vs. noisy range scan Realistic range sensors do not generate a visibility *polygon* $\mathcal{V}$ as assumed for the idealizing method, but only a finite sequence S of *scan points* (usually, measured at equidistant angles). Furthermore, these scan points do not lie exactly on the robot's visibility polygon, but are perturbed due to sensor uncertainties. An example is depicted in Figure 6.1 on the next page, which shows a part of a polygonal map of the Computer Science Department at the University of Würzburg. Here, Figure 6.1 (a) depicts the exact visibility polygon $\mathcal{V}_p$ of a robot standing at point p, whereas Figure 6.1 (c) shows a real noisy laser range scan S_p taken at the corresponding position in the department hall using a SICK LMS 200 laser scanner. Even if we connect the scan points by straight line segments as shown in Figure 6.1 (b) in order to obtain a shape representation of the scan points, we only get an approximation $\mathcal{V}_{S_p}$ of the exact visibility polygon $\mathcal{V}_p$ based on the map.

Missing compass For the localization process we assume that we already know the exact orientation of the robot. But in practice this is often not the case. Usually, a robot has no compass at all and gains information about its orientation only from its odometry sensors, which have a large drift if the travelled distance increases. Thus, we only have inexact knowledge or no knowledge at all about the robot's orientation.

Map polygon is not simple Of course, a realistic map cannot be modeled by a *simple* polygon as assumed in Chapter 3, since there are almost always pillars in the robot's environment, which have to be modeled by *holes* in the map polygon.

Unknown obstacles and incomplete map There may be obstacles in the environment that are not considered in the map and which may affect the robot's view. For example, furniture that is too small to be took into account for the map generation process or even dynamic obstacles like people or other robots. Such obstacles can also be recognized in the right part of the example scan shown in Figure 6.1 (c).

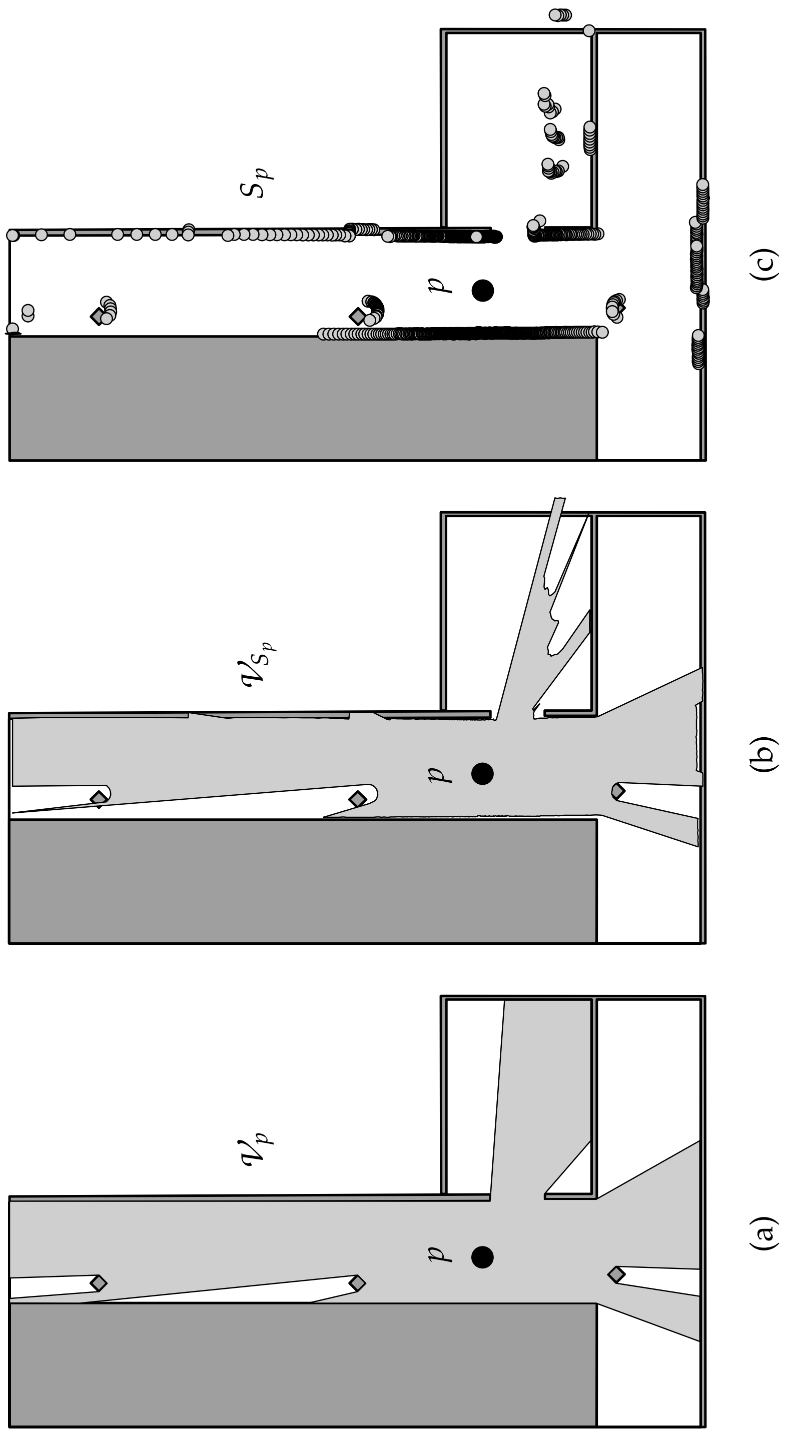


FIGURE 6.1: Exact visibility polygon (a) and approximated visibility polygon (b) of a noisy scan (c)

Limited sensing range Realistic range sensors only have a limited sensing range (about 80 m for a typical laser scanner) and obstacles that have a greater distance to the robot cannot be detected. If the robot has to operate in an environment with greater distances between positions that are visible to each other, the scans at such positions significantly differ from the respective visibility polygons.

The consequence of the problems described above is in all cases that the (approximated) visibility skeleton V_S^* , which the robot computes from its approximated visibility polygon $\mathcal{V}_S$, usually does not match any of the preprocessed skeletons exactly. That is, Step 4 of Algorithm 3.13 on page 44 fails, as no leaf in the d-dimensional search tree T_d could be found, which contains the corresponding encoding vector $[V_S^*]$. Therefore, no equivalence class of visibility cells could be determined, for which the point location query of Step 5 had to be performed, and as a consequence the localization process completely fails.

6.2 Adaptation to Practice Using Distance Functions

In the following sections we introduce an approach to tackling the problems described above, which at least reduces them and which allows us to apply the ideas of Section 3.8 also to real-world scenarios. To this end, let S be a given range scan (from the sensor). Then, we search for the preprocessed skeleton that is *most similar* to S. In order to model the resemblance between a scan S and a skeleton V^*, we use an appropriate *distance function* $d(S, V^*)$. Then, instead of performing an *exact match* query as in the original Algorithm 3.13, we carry out a *nearest-neighbor* query in the set of skeletons with respect to the chosen distance function $d(S, V^*)$ in order to find the skeleton with the highest resemblance to the scan.

Depending on the distance function, we then additionally have to apply a local matching algorithm to the scan and the skeleton in order to determine the position of the robot. The reason for this procedure is that not all methods for determining a distance measure yield an optimal matching (i. e., a translation vector and a rotation angle) as well. Consider, for example, the *Arkin metric*, see [AC+91], which compares two shapes by means of their *turning functions*. The turning function can be seen as a parametrisation of the angle of rotation of a point that traverses the boundary subject to its arc length. It is invariant under scaling and translation and in the case of polygons it is piecewise constant (with discontinuities at the polygon vertices). Thus, an algorithm that computes the Arkin metric only provides the optimal rotation angle and no translation vector (besides the distance measure). In contrast to this, algorithms for computing the *minimum Hausdorff distance* under rigid motions, see

[ABB95, AG99] and Section 6.6, may provide both the distance measure and the corresponding matching.

6.3 Requirements to the Distance Function

In order to be useful in practice, a distance function $d(S, V^*)$ should meet at least some of the following properties, which are motivated by the intuitive definition of similarity between geometric objects (cf. [Vel01, Sect. 3]):

Continuity So as to be robust against small distortions, the distance function should be *continuous* in the sense that small changes in the scan (e. g., caused by noisy sensors) or even in the skeleton (e. g., caused by an inexact map) should result in only small changes of the distance. More precisely: Let $d_S(S_1, S_2)$ and $d_{V^*}(V_1^*, V_2^*)$ be functions that measure the resemblance between two scans S_1 and S_2 and between two skeletons V_1^* and V_2^*, respectively. Possible reference distance measures for $d_S(S_1, S_2)$ are, for example, the Hausdorff distance (see Section 6.6 and [ABB95, AG99]) or the *symmetric difference* [AF$^+$98]. (In the latter case we have to use the shape representation of the scan points, that is, their approximated visibility polygon $\mathcal{V}_S$.)

Then, we define the continuity of the distance $d(S, V^*)$ as follows.

Definition 6.1 (Continuity of a Scan-Skeleton Distance)
For a given reference distance measure $d_S(S_1, S_2)$ we say that the distance function $d(S, V^*)$ is ***continuous with respect to scans*** *if*

$$\forall_{\varepsilon>0}\,\exists_{\delta>0}\,\forall_{S_1,S_2}\;\; d_S(S_1, S_2) < \delta \implies |d(S_1, V^*) - d(S_2, V^*)| < \varepsilon$$

holds for all skeletons V^*.

Analogously, for a given reference distance measure $d_{V^*}(V_1^*, V_2^*)$ between skeletons the distance $d(S, V^*)$ is said to be ***continuous with respect to skeletons*** *if*

$$\forall_{\varepsilon>0}\,\exists_{\delta>0}\,\forall_{V_1^*,V_2^*}\;\; d_{V^*}(V_1^*, V_2^*) < \delta \implies |d(S, V_1^*) - d(S, V_2^*)| < \varepsilon$$

holds for all scans S.

The requirement of continuity is also motivated by the classification of the edges of the visibility polygon being sorted into different types (spurious edges, partially visible edges, etc.) in Step 1 of Algorithm 3.13 on page 44. This particularly makes the original method susceptible to perturbations: Even a small translation of a vertex can change the type of an edge, which yields a skeleton that does not match any equivalence class. In this sense, the exact match query in Step 4 can also be interpreted as a discrete distance measure between a visibility polygon and a skeleton, which, however, strongly violates the continuity requirement, because it attains only two values (e. g., 0 – “match” and 1 – “no match”).

Similarity preservation A skeleton V^* that is “similar” to a scan S should have a small distance value $d(S, V^*)$. Otherwise, the distance would not give any advice for finding a well-matching skeleton and therefore would be useless for the localization algorithm. In particular, if we take a scan S_p from a point p whose skeleton equals V_p^*, we want the distance $d(S_p, V_p^*)$ to be zero or at least small, depending on the amount of noise and the resolution of the scan.

Analogously, if a scan S and a skeleton V^* have a small distance $d(S, V^*)$, we should safely be able to assume that there exists a position p in the map such that S_p is “similar” to S and $V_p^* = V^*$ holds.

Translational invariance As the robot has no knowledge about the relative position of the coordinate systems of the scan and the skeleton to each other, a translation of the scan or the skeleton in their local coordinate systems must not influence the distance. Rather finding this position is the goal of the localization algorithm.

Rotational invariance If the robot does not have a compass, the distance must also be invariant under rotations of the scan (or the skeleton, respectively).

Fast computability It turns out in the following section that the distance $d(S, V^*)$ has to be determined several times for a single localization query (for different skeletons). Thus, the computation costs should not be too high.

6.4 Maintaining the Skeletons

As we do not want to compare a scan with all skeletons in order to find the skeleton with the highest resemblance (remember that their number can be in $\Omega(n^2 r)$ even for simple map polygons, see Theorem 3.9 on page 46), the skeletons should be stored in an appropriate spatial data structure that we can search through efficiently.

For this purpose we use the *Monotonous Bisector Tree* of Noltemeier et al. [NVZ93], a *spatial index* that allows to partition the set of skeletons hierarchically with respect to a second distance function $D(V_1^*, V_2^*)$, which models the resemblance between two skeletons V_1^* and V_2^*. Using the distance function $D(V_1^*, V_2^*)$, the set of skeletons is recursively divided into *clusters* in a preprocessing step. Such a cluster of skeletons is characterized by its cluster center (one of the skeletons of the respective cluster) and its cluster radius, that is, the maximum distance between the center and any other skeleton of the cluster. In the recursion step each cluster with sufficiently many skeletons is divided into two preferably equally-sized subclusters. This is accomplished by means of the bisector with respect to $D(V_1^*, V_2^*)$ between the two new cluster centers, which guarantees monotonously decreasing cluster radii. The resulting division then represents the similarities of the skeletons among each other.

The distance function $D(V_1^*, V_2^*)$, which is used in order to compute the cluster radii, should be "compatible" to the function $d(S, V^*)$, such that in the nearest-neighbor query not all clusters have to be investigated. That is, at least the *triangle inequality*

$$d(S, V_1^*) + D(V_1^*, V_2^*) \geq d(S, V_2^*) \tag{6.1}$$

should be satisfied. This way, we can determine lower bounds on the distance values $d(S, V^*)$ of complete clusters, when traversing the tree. Such a cluster can then be rejected and need not be examined at all.

The computation of the skeletons can be accomplished similar to Algorithm 3.14 on page 45, except that we need not overlay the resulting visibility cells (which is only required for the exact localization algorithm). Furthermore, we can also withdraw the assumption of a simple map and instead allow holes in the map polygon, since the notions of skeletons and the visibility cell decomposition are also well-defined for map polygons that have holes. But of course, most of the complexities of Chapter 3 become worse in this case; for example, the worst-case number of visibility cells (which is in $O(n^2r)$ for map polygons *without* holes, cf. Theorem 3.9) may then be in $\Omega(n^2r^2)$. This way, we circumvent the problems with non-simple map polygons, which was mentioned in Section 6.1 by accepting a trade-off in the execution speed of our modified algorithm.

6.5 Suitable Distance Functions

It is not easy to find distance functions that have all the properties described in Section 6.3. In particular, the prerequisite of a fast computability is contrary to the remaining ones. Moreover, it is often not possible to simply use existing polygon distances, because in our setting we have to cope with scans and skeletons instead of polygons. Therefore, a careful adaptation of the chosen distance functions is almost always necessary. And of course, it is even more difficult to find for a given scan-skeleton distance $d(S, V^*)$ a compatible distance $D(V_1^*, V_2^*)$, which we need in order to perform the nearest-neighbor query efficiently.

In the following sections we investigate two distance functions, the *Hausdorff distance* and the *polar coordinate metric*, and illustrate the occurring problems.

6.6 The Hausdorff Distance

For two point sets $A, B \subset \mathbb{R}^2$, their *Hausdorff distance* $\delta(A, B)$ is defined as the maximum of $\vec{\delta}(A, B)$ and $\vec{\delta}(B, A)$,

$$\delta(A, B) := \max\{\vec{\delta}(A, B), \vec{\delta}(B, A)\}\,, \tag{6.2}$$

where

$$\vec{\delta}(A, B) := \sup_{a \in A} \inf_{b \in B} \|a - b\| \tag{6.3}$$

is the *directed Hausdorff distance* from A to B. Accordingly, the term $\vec{\delta}(A, B)$ stands for the maximum distance of some point from A to the set B.[1]

Let $\mathcal{T}$ be the set of all Euclidean transformations (i. e., combinations of translations and rotations). The undirected and directed *minimum Hausdorff distances* with respect to these transformations are defined as

$$\delta_{\min}(A, B) := \inf_{t \in \mathcal{T}} \delta(A, t(B)) \quad \text{and} \quad \vec{\delta}_{\min}(A, B) := \inf_{t \in \mathcal{T}} \vec{\delta}(A, t(B))\,. \tag{6.4}$$

It can easily be seen that the minimum Hausdorff distances are continuous and by definition also are invariant under translations and rotations. Thus, they fulfill most of the requirements described in Section 6.3. But their computation is very expensive. According to [ABB95], the minimum Hausdorff distance can be computed in time $O((ms)^4(m+s)\log(m+s))$ if m is the complexity of the scan and s is the complexity of the skeleton. This is surely too expensive in order to be used in our application.

On the other hand, the computation of the Hausdorff distance without minimization over transformation application is relatively cheap, cf. [ABB95], namely in $O((m+s)\log(m+s))$. The property of continuity is also not affected, but we now have to choose a suitable translation vector and a rotation angle by hand.

1 Note that in the case of closed sets (e. g., the boundary and the shape of a polygon or finite sets of scan points) there always exists a pair of points $(a, b) \in A \times B$ such that their distance $\|a - b\|$ equals $\vec{\delta}(A, B)$.

An obvious choice for such a translation vector for a scan S and a skeleton V^* is the vector that moves the scan origin (i. e., the position of the robot) somewhere into one of its corresponding visibility cells $\mathcal{C}$ (e. g., the center of gravity of $\mathcal{C}$). This is reasonable, because according to their definition exactly the points in the cells of $|\mathcal{EC}_{V^*}|$ induce the skeleton V^* (for example, see Figure 3.12 on page 43). Of course, the consequence of this approach is that all cells with the same skeleton (e. g., the six cells drawn in gray in Figure 3.12) must be *handled separately*, because the distance $d(S, V^*)$ now does not only depend on V^*, but also on the visibility cell itself. Besides, the intersection of all these cells of a given equivalence class $\mathcal{EC}_{V^*}$ may be empty and we might not find a common translation vector for all of them.

Note that for this new setting, where the distance $d(S, V^*)$ also depends on a certain visibility cell, the notation $d(S, V^*)$ is a bit misleading, since there usually exist several cells that have the same skeleton V^*. To be correct, we should use the notation $d(S, \mathcal{C})$, where the dependence of the distance from the respective cell is expressed more clearly. But in the following, we use the more intuitive expression $d(S, V^*)$.

Of course, the bigger the cell is that the scan has to be placed into, the bigger is the error of this approach, compared with the minimum Hausdorff distance.

Approximate Matching Strategies

A compromise for computing a good matching, which has the advantages of the previous algorithms, is using an *approximate matching strategy*, which yields only a pseudo-optimal solution. This means, the algorithm finds a transformation $t \in \mathcal{T}$ with

$$\delta(A, t(B)) \leq c \cdot \delta_{\min}(A, B)\,,$$

for a constant $c \geq 1$.

Alt et al. [ABB95] showed that for any constant $c > 1$ an approximate matching with respect to Euclidean transformations can be computed in time $O(ms \log(ms) \log^*(ms))$ using so-called *reference points*. If we are only interested in an approximate matching with respect to translations instead of Euclidean transformations, the time complexity would be even better, namely in $O((m+s) \log(m+s))$.

Using the Hausdorff Distance in Practice

Another point that we have to consider is that a skeleton (interpreted as a point set) in general is not bounded, because it includes a straight line l_i labeled to each artificial edge a_i in order to represent the possible positions of its candidate edges. The result is that the directed distances $\vec{\delta}(V^*, S)$ and $\vec{\delta}_{\min}(V^*, S)$ almost always return an infinite value (except for the trivial case when V^* equals the convex map polygon and thus has no artificial edges). Therefore, we either may only use the directed distances $\vec{\delta}(S, V^*)$ and $\vec{\delta}_{\min}(S, V^*)$ in order to define $d(S, V^*)$ or we must modify the skeleton representation that we use in order to compute useful distance values. If we pursue the first approach, it remains the problem of finding a suitable distance function $D(V_1^*, V_2^*)$, since then, neither one of the Hausdorff distances may be used here. But note that in this case the distance $\vec{\delta}_{\min}(S, V^*)$ is also similarity preserving, provided that the resolution of the scan is high enough such that no edge (in particular, no artificial edge) is missed.

The second approach of modifying the skeleton representation in order to get a bounded point set seems to be more promising, since it allows us to define *both* distances $d(S, V^*)$ and $D(V_1^*, V_2^*)$ in an appropriate way. One possibility of doing so is described on the next page, as we use the same method in order to compute the polar coordinate metric, which is described in the following section.

6.7 The Polar Coordinate Metric

A more specialized distance function for our problem than the Hausdorff distance is the *polar coordinate metric* (PCM for short) introduced by Wahl [Wah97, KW99], which takes a fundamental property of the localization problem into account: All occurring polygons are *star-shaped* in the following sense, and we even know a kernel point:

- The approximate visibility polygon $\mathcal{V}_S$ (generated from the scan points) is star-shaped by construction, with the origin as a kernel point.
- Every skeleton V^* is star-shaped in the sense that from every point in its kernel $\ker V^*$ all full edges are completely visible, and for each artificial edge a_i a part of the corresponding straight line l_i is visible.

In order to define the PCM between two (star-shaped) polygons P and Q with kernel points p_{K} and q_{K}, we first define their *polar coordinate function*.

Definition 6.2 (Polar Coordinate Function)
Let P be a star-shaped polygon with a kernel point $p_{\mathrm{K}} \in \ker P$. For any angle φ let $b(\varphi)$ be the unique intersection point of the boundary of the polygon P with a ray starting at p_{K} in direction φ. Then, the ***polar coordinate function*** *(PCF*

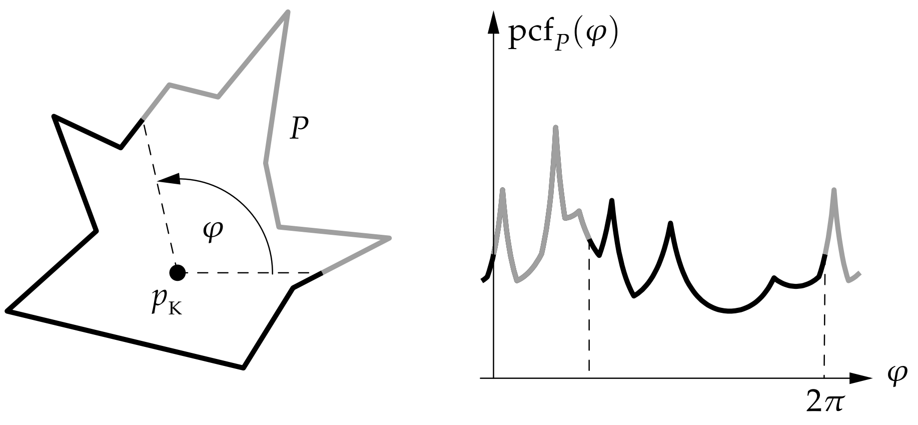


FIGURE 6.2: The polar coordinate function $\mathrm{pcf}_P(\varphi)$ for a star-shaped polygon P with kernel point p_K

for short) of P (with respect to p_K) is defined as

$$\mathrm{pcf}_P\colon \mathbb{R} \to \mathbb{R}_{\geq 0}\,, \qquad \mathrm{pcf}_P(\varphi) := d_2(p, b(\varphi))\,. \tag{6.5}$$

That is, the function $\mathrm{pcf}_P(\varphi)$ corresponds to a description of the polygon P in polar coordinates (with p_K as the origin) and is periodical with a period of 2π. Figure 6.2 depicts the PCF for a star-shaped polygon P as an example.

We then define the value of the PCM between the polygons P and Q as the minimum integral norm between the functions pcf_P and pcf_Q in the interval $[0, 2\pi[$ over all horizontal translations between the two graphs (i. e., rotations between the corresponding polygons).

Definition 6.3 (Polar Coordinate Metric)
Let P and Q be two star-shaped polygons with kernel points $p_K \in \ker P$ and $q_K \in \ker Q$. Then, the value of the ***polar coordinate metric*** *between P and Q is defined as*

$$\mathrm{pcm}(P, Q) := \min_{t \in [0, 2\pi[} \sqrt{\int_0^{2\pi} \Big(\mathrm{pcf}_P(\varphi - t) - \mathrm{pcf}_Q(\varphi)\Big)^2 \,\mathrm{d}\varphi}\,. \tag{6.6}$$

Figure 6.3 on the next page shows an example, where the two graphs are already translated in such a way that the integral norm is minimized.

Using the PCM as Distance $d(S, V^*)$

If we want to use the PCM as a distance function $d(S, V^*)$, a similar problem as with the Hausdorff distance occurs: We need corresponding star-shaped

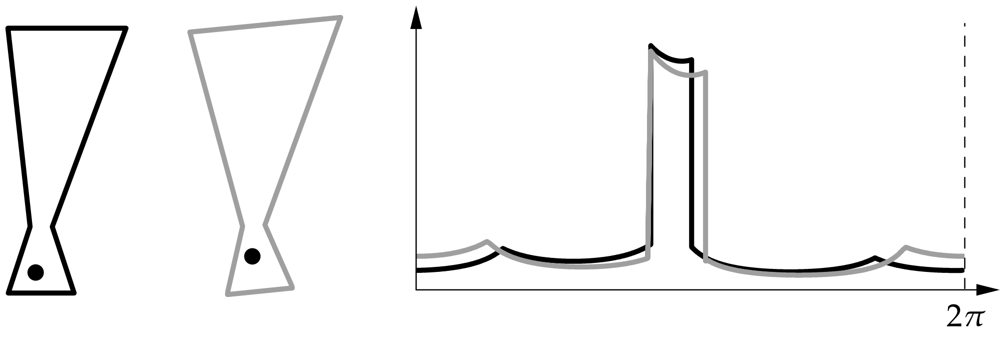


FIGURE 6.3: Computation of the polar coordinate metric as minimum integral norm

polygons for S and V^* that can be used as polygonal representatives for the scan and the skeletons in order to compute their PCM value:

- For the scan, we choose the approximated visibility polygon $\mathcal{V}_S$, which is star-shaped by construction. As already stated above, the coordinate origin can be used as a kernel point.

- In order to generate a polygon from a skeleton V^*, we choose for each visibility cell $\mathcal{C} \in |\mathcal{EC}_{V^*}|$ a point $p \in \mathcal{C}$ inside (e. g., the center of gravity of $\mathcal{C}$) and determine its visibility polygon $\mathcal{V}_p$. Then, p is a kernel point of $\mathcal{V}_p$ by construction. That is, for each visibility cell we simply pretend to stand at its center of gravity in order to get a bounded polygonal representation of its corresponding skeleton.

  As discussed in Section 6.6 on on page 151 there is some minor drawback of this idea: Since the generated polygons not only depend on the skeletons but also on the corresponding visibility cells, we should keep in mind that also the computed distance does. Furthermore, using this approach we produce an additional error, which (in the worst-case, where the robot stands near the boundary of a visibility cell) increases with the size of the visibility cells.

In the sequel we use $\mathcal{V}_S$ and $\mathcal{V}_p$ as polygonal representatives of S and V^* for determining their distance measures. Then, our goal is to find that polygon $\mathcal{V}_p$ that is most similar to the approximated visibility polygon $\mathcal{V}_S$ with respect to $\text{pcm}(\mathcal{V}_S, \mathcal{V}_p)$.

Using the PCM as Distance $D(V_1^*, V_2^*)$

It can be shown (see [Wah97]) that the function $\text{pcm}(P, Q)$ is a polygon *metric* if the computed kernel points are invariant under Euclidean transformations. That is, if p'_K denotes the kernel point of a polygon $P' = t(P)$ for a transformation $t \in \mathcal{T}$, the equality $t(p_K) = p'_K$ must hold, for all polygons P and all transformations $t \in \mathcal{T}$. For example, the center of gravity of the kernel of the polygon has this property.

Since the PCM has all metric properties, it particularly fulfills the triangle inequality. Therefore, we can use it not only for defining the scan-skeleton distance $d(S, V^*)$, but also for defining a *compatible* skeleton-skeleton distance $D(V_1^*, V_2^*)$.

For this task, we again use for each visibility cell $\mathcal{C}$ (with corresponding skeleton V^*) its polygonal representative $\mathcal{V}_p$ where p is the center of gravity of $\mathcal{C}$. Then, the triangle inequality (6.1) on page 149 directly follows from the triangle inequality of the PCM. Thus, we are able to apply the Monotonous Bisector Tree to the set of skeletons in order to increase the performance of the nearest-neighbor query.

6.8 Efficient Computation of the PCM

The exact computation of the minimum in Equation (6.6) on page 153 seems to be difficult and time consuming, since the polar coordinate functions of a polygon with m edges consists of m pieces of functions of the form $c_i / \sin(\varphi + \alpha_i)$. For one fixed translation t (this corresponds to the case, when the robot already knows its exact orientation), the integral

$$\sqrt{\int_0^{2\pi} \Big(\text{pcf}_P(\varphi - t) - \text{pcf}_Q(\varphi)\Big)^2 \, d\varphi} \tag{6.7}$$

can be computed straightforward in linear time $O(m + n)$, where m and n stand for the complexities of the two polygons. But the global minimum over all possible values of $t \in [0, 2\pi[$ seems to be much harder to determine.

Therefore, we use two different approximative approaches for computing a suitable PCM value. Both approaches use a set of supporting angles (or points, respectively) for each of the two involved polar coordinate functions. For a given polygon its supporting angles are the angles that correspond to a vertex of the polygon plus the angles of the local minima of the inverse sine functions, see Figure 6.4 (a). The ideas of the two approximative approaches are then as follows.

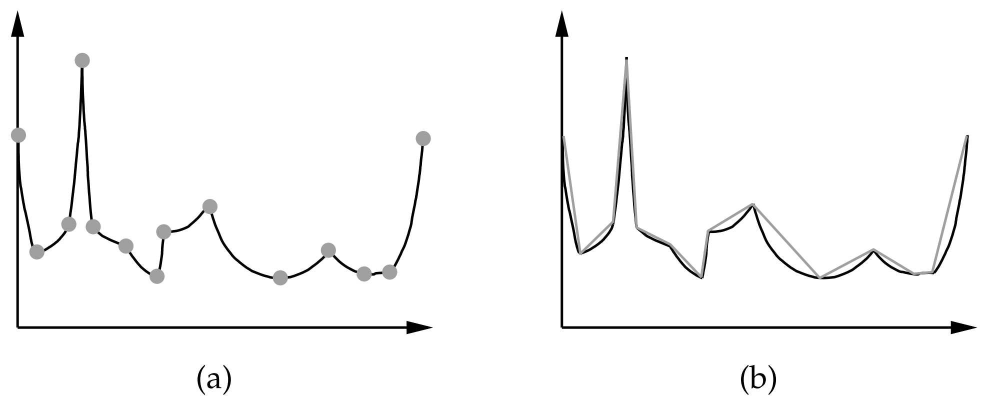


FIGURE 6.4: Two approximative approaches for computing the PCM

Minimizing only over a subset of rotation angles The first approach concentrates on the $O(m)$ or $O(n)$, respectively, supporting angles, see Figure 6.4 (a). Namely, we do not minimize over *all rotation angles* of the two polygons, but only over the $O(mn)$ rotation angles that place one supporting angle of the first polygon on a supporting angle of the second one. Then, for each of the $O(mn)$ rotation angles t its integral value (6.7) is computed exactly in time $O(m+n)$. Summing up, we need $O(mn(m+n))$ time to compute this approximated value of the PCM.

But note that the resulting approximative distance function is no longer a metric, since the triangle inequality does not hold anymore (although the identity and symmetry properties are not affected by the approximation).

Using a linear approximated PCF The second approach computes an *exact minimum* over all rotation angles, but uses a modified polar coordinate function. Namely, we introduce a linear approximation of the PCM, which also has all metric properties and which suffices for our applications. This approximation is depicted in Figure 6.4 (b): The supporting points (that correspond to a polygon vertex or a local minimum of the PCF) are connected by straight line segments in order to get a modified distance function. The minimum integral norm is then defined like in the non-approximated version of the PCM, see Equation (6.6).

Following an idea of [AC+91], the actual computation of the minimum integral norm between the two *piecewise linear* functions can now be carried out much faster than the computation of the original PCM: Arkin et al. compute the minimum integral norm between two *piecewise constant* functions in time $O(mn)$. This idea can be generalized to compute the approximated PCM in time $O(mn \cdot (m+n))$. Of course, if we do not want to minimize over the rotations, the computation time is again in $O(m+n)$ like for the non-approximated version.

6.9 Using the PCM in Realistic Scenarios

In order to examine the usability of the polar coordinate metric in realistic environments, we first investigate the continuity property (cf. page 147): For a fixed kernel point Wahl [Wah97] showed that the function pcf_P is continuous in φ except for one special case: When we move a vertex of a polygon edge such that the edge becomes collinear to p_K, the function pcf_P has a discontinuity at the corresponding angle, the height of which represents the length of the collinear edge. Moreover, the PCF is also continuous in the sense of the definitions in Section 6.3 with respect to translations of the polygon vertices or translations of the kernel point, unless this special case occurs. But as pcf_P and pcf_Q may have only a finite number of such discontinuities, the integration makes them continuous with respect to *all* translations of polygon vertices and translations of the kernel points, provided that P and Q remain star-shaped with kernel points p_K and q_K.

It can easily be verified that the PCM fulfills the continuity requirement of Section 6.3 in the following sense: Given two polygons P and Q with kernel points p_K and q_K and an $\varepsilon > 0$, we can find a $\delta > 0$ such that $|\text{pcm}(P, Q) - \text{pcm}(P', Q)| < \varepsilon$, for all polygons P' that are created from P by moving each vertex by at most δ, provided that p_K remains to be a kernel point also of P'. Furthermore, if the kernel points are considered as a part of the polygons (i. e., part of the *input* of the PCM) the above statement holds even if we also are allowed to move the kernel points by at most δ.

Moreover, if the kernel points are *not* considered as part of the input of the PCM (that is, they are computed from P and Q by the algorithm that computes the $\text{pcm}(P, Q)$ value), the PCM is continuous as well, provided that the kernel points continuously depend on the polygons. For example, the center of gravity of the kernel of a polygon P depends continuously on P and can be used as a kernel point p_K, whereas, for example, the leftmost kernel point does *not* depend continuously on the polygon.

Revisiting the Problems in Realistic Scenarios

Since the polar coordinate metric is continuous (in the above sense), we can hope that the first problem mentioned in Section 6.1 (the problem of noisy scans) is resolved satisfactorily if we use the polar coordinate metric in order to define the distance functions $d(S, V^*)$ and $D(V_1^*, V_2^*)$. The PCM also copes with the problem of an unknown robot orientation, since it is invariant under translations and rotations by definition. Furthermore, using the ideas of the previous section, the linear approximated variant of the PCM can be computed relatively fast, that is, in $O(mn \cdot (m+n))$. And if we additionally assume that the robot has a compass, we are even able to compute the distance measures in linear time.

Since the problems with non-simple map polygons were already circumvented in Section 6.4 by accepting a trade-off in the execution speed, there remain two unresolved problems, which we have to investigate: Unknown obstacles in the map and a limited sensing range.

Unknown obstacles and incomplete map Unknown obstacles in the environment do not strongly influence the PCM as long as they take up only a small interval of the whole scanning angle. But the other case, where the obstacles occupy a large part of the robot's view, poses a problem to our localization method. Here, additional (heuristic) algorithms are needed, which detect such cases and guide the search in the correct direction.

Limited sensing range If the robot's sensing range is smaller than the maximum sensing distance, which may occur in the environment, this problem can be tackled in a straightforward heuristic manner: When we compute the distances $d(S, V^*)$ and $D(V_1^*, V_2^*)$, we simply cut off all distance values larger than the sensing range, that is, for all occurring polygons we use a modified PCF,

$$\mathrm{pcf}'_P(\varphi) := \min\{\mathrm{pcf}_P(\varphi), \text{sensing range}\}\,. \tag{6.8}$$

But we should keep in mind that the resulting distance function is no longer a metric.

6.10 The Implementation RoLoPro

Using LEDA, the Library of Efficient Datatypes and Algorithms [MN99], we have implemented two versions of the localization algorithm in C++, namely the original method described in Section 3.8 for exact sensors as well as the modification for realistic scenarios introduced in the current chapter. For our implementation, the original algorithm was modified and simplified at some points, since we did not focus our efforts on handling sophisticated but rather complicated data structures and algorithmic ideas that were suggested by Guibas et al. Rather, we wanted to have an instrument to experiment with different inputs for the algorithm that is reasonably stable to be used in real-life environments and that can serve as a basis for own modifications. Of course, a consequence of these simplifications is that the program does not keep to all theoretical time and space bounds of Section 3.9 and the original paper [GMR97], as this would have required a tremendous programming effort. Nevertheless, the algorithm is stable and reasonably efficient for sufficiently small environments.

We implemented the two distance functions described in the previous sections, the Hausdorff distance and the polar coordinate metric. Note that in both cases the localization algorithm only tries to find the best-matching skeleton and does *not* compute an optimal matching as well (although some implemented versions of the PCM algorithm at least compute an optimal rotation angle, but no optimal translation vector). As already mentioned in Section 6.2, this matching has to be determined by a local matching algorithm in a postprocessing step, which uses the result of the localization process as a first approximation. This postprocessing step was not implemented.

6.10.1 The Hausdorff Distance

In order to define the distance $d(S, V^*)$, we implemented several versions of the Hausdorff distance:

The directed distance $\vec{\delta}(S, V^*)$, which computes the maximum over all minimum distances from a scan point of S to the skeleton V^*;

the undirected version $\delta(S, V^*)$, for which the skeleton representation was modified as described on page 153, in order to get a polygonal representative for V^*;

the k-Hausdorff distance, another variant of the Hausdorff distance (for point sets A and B, A finite), where the k-th smallest value

$$\min_{a \in A}{}_k \min_{b \in B} \|a - b\| \qquad (6.9)$$

rather than the largest value $\max_{a \in A} \min_{b \in B} \|a - b\|$ determines the distance value, cf. [AG99]. With a carefully chosen parameter k this may help to overcome the sensitivity of the Hausdorff distance to *outliers*.

In all above cases we do *not* compute the *minimum* Hausdorff distances, but instead assume that the robot has a compass (i. e., we need not take care about the rotation angle) and furthermore use a heuristic translation vector as described on page 150. Moreover, we have not implemented the efficient skeleton management described in Section 6.4, since we could not find a suitable distance $D(V_1^*, V_2^*)$, which is compatible to one of the Hausdorff distances above, that is, which fulfills the triangle inequality (6.1). Therefore, in this case the scan has to be compared with *all skeletons*, which is much more time consuming than the PCM approach, which uses the Monotonous Bisector Tree.

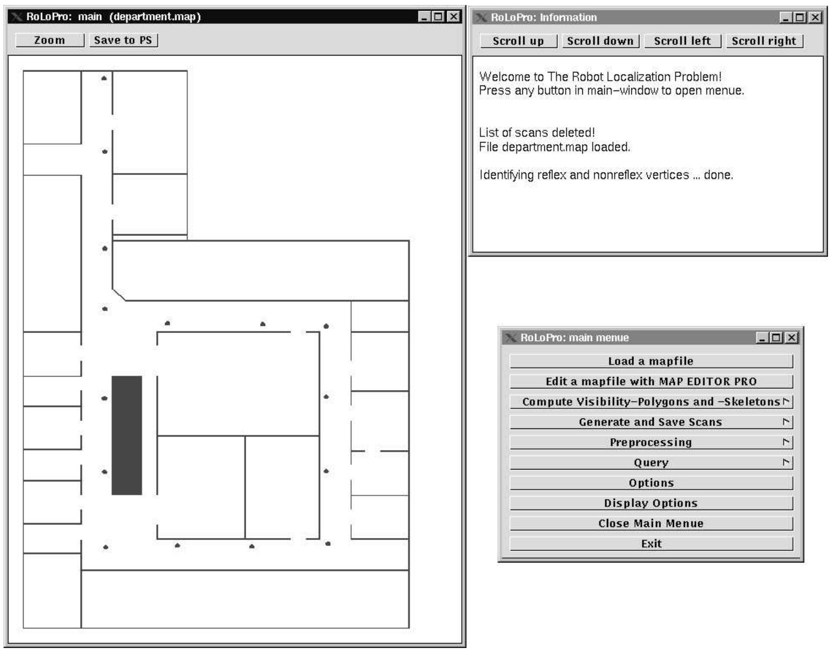
6.10.2 The Polar Coordinate Metric

We have implemented both approximative approaches, which were presented in Section 6.8, the strategy that minimizes the integral norm only over a subset of rotation angles as well as the linear approximated PCF. Furthermore, for both approaches two variants are implemented, one variant for robots *without a compass*, which has a cubic computation time, and another variant for robots *with a compass*, where no optimal rotation angle has to be computed and which has a linear time complexity.

As described on page 153, we use polygonal representatives for the scans as well as for the skeletons in order to compute their respective distances. This way, we were able to use the PCM for both distances, $d(S, V^*)$ as well as $D(V_1^*, V_2^*)$, and to ensure that the two distances are compatible. In particular, we were able to use the Monotonous Bisector Tree described in Section 6.4 for efficient nearest-neighbor queries in the set of preprocessed skeletons.

6.10.3 Some Screen Shots of RoLoPro

The Figures 6.5 to 6.10 on the following pages show some typical situations using RoLoPro: Figure 6.5 depicts the three major windows of RoLoPro, the main window, which shows the map, the information window and the main menu window, where all actions are displayed and can be chosen. The four screen shots of Figure 6.6 show a map of a small room with three niches and three obstacles in front of the niches. The visibility polygons and their corresponding skeletons at four different positions are depicted. Note that the skeletons of the two topmost positions are identical. The next screen shot of Figure 6.7 is taken during the preprocessing step and depicts the visibility cell decomposition of the room from the previous example, whereas in Figure 6.8 an exact localization query in this room is visualized: For a robot standing in the left niche (marked by a bullet »•«) two visibility cells are detected that have the same skeleton as the robot. Thus, in this case two possible robot placements exist. In contrast to the exact localization query, the Figures 6.9 and 6.10 show a localization query for a noisy scan, which is simulated by RoLoPro, using the polar coordinate metric. In Figure 6.9 the robot's position is marked by a square »■« and the center of gravity of the best-matching visibility cell (with



Welcome to The Robot Localization Problem!
Press any button in main–window to open menue.

List of scans deleted!
File department.map loaded.

Identifying reflex and nonreflex vertices ... done.

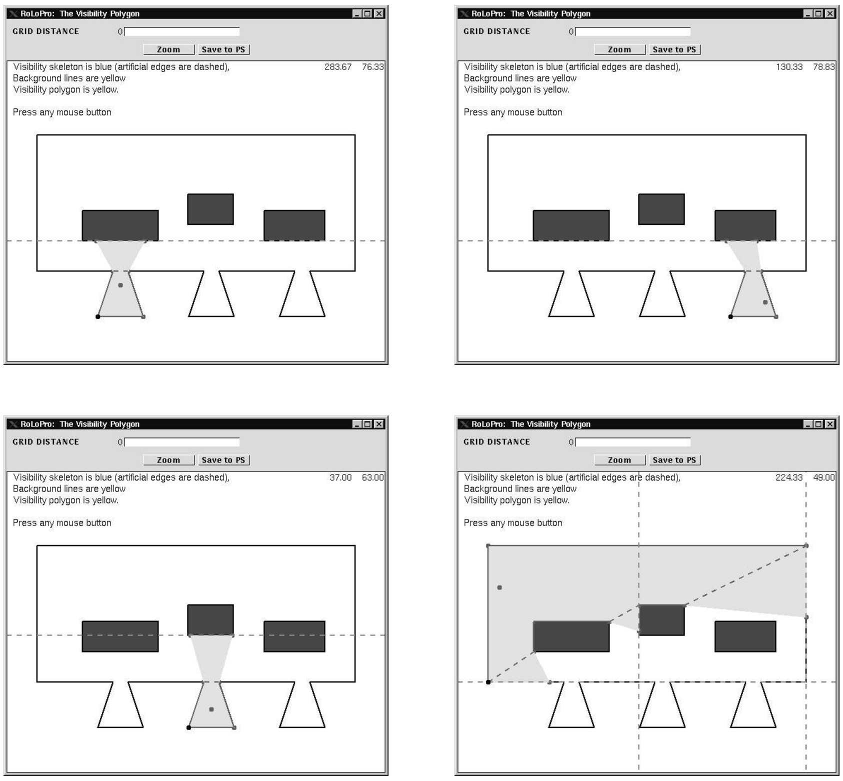
FIGURE 6.5: The three main windows of ROLOPRO

respect to the PCM) is marked by a bullet »•«. Figure 6.10 depicts the corresponding polygonal representatives of the scan and the skeleton as well as their polar coordinate functions.

6.10.4 Additional Heuristic Features of ROLOPRO

We have implemented some additional features to ROLOPRO, which are motivated by practical needs rather than by theoretical investigations. Thus, we neither proved any theoretical results on the quality of these modifications of the localization algorithms nor could we guarantee that properties like, for example, the triangle inequality remain true. But nevertheless we believe that they may be useful in practice.

Noisy Compass For the polar coordinate metric, which we have described in the previous sections, we have only two choices concerning the existence of a compass: Either the robot *has* a compass, which implies a fixed orientation of the scan with respect to the preprocessed skeletons, or the the robot has *no compass at all*, such that we have to use the version of the PCM that is invariant under rotations and has a cubic computation time (compared to a liner running time in the other case). But with a rather simple modification concerning the



Visibility skeleton is blue (artificial edges are dashed),
Background lines are yellow
Visibility polygon is yellow.

Press any mouse button

283.67 76.33

130.33 78.83

37.00 63.00

224.33 49.00

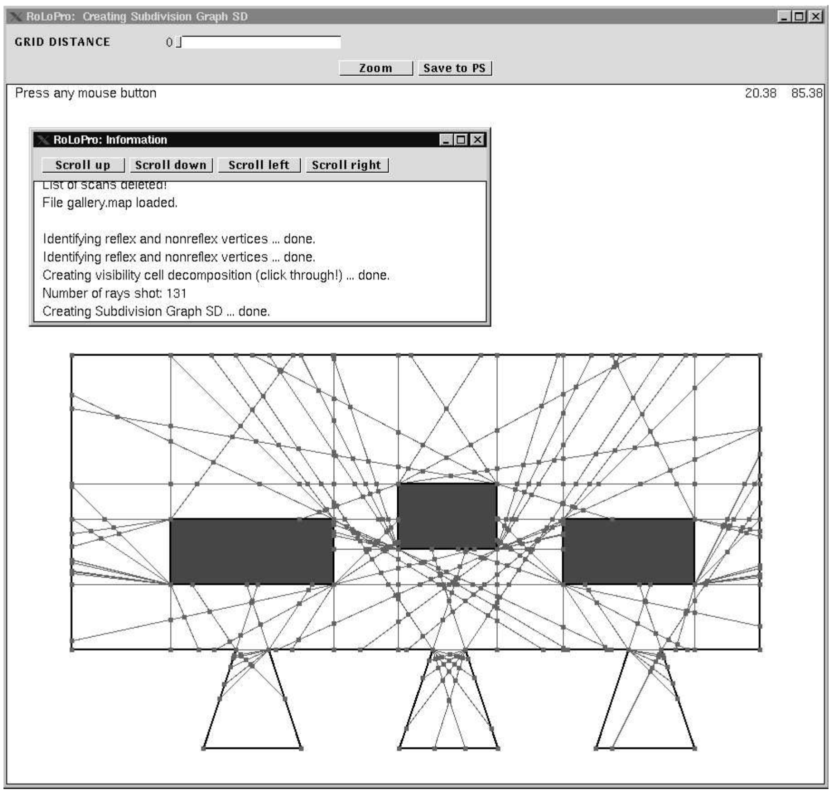
FIGURE 6.6: Computing visibility polygons and skeletons

minimization interval in Equation (6.6) on page 153 we also can model a *noisy* compass:

No compass at all This is the case in Equation (6.6) on page 153, where we have no knowledge at all about the robot's orientation and therefore have to minimize the integral norm over all possible rotation angles between the scan and the skeleton. Thus, the minimization interval is $[0, 2\pi[$.

Exact compass In the case of an exact compass we already know the orientation o of the scan with respect to the map, such that the "minimization interval" consists only of the single value o, which represents the rotation angle between the scan and the map, cf. Equation (6.7) on page 155.

Noisy compass If we know the orientation o of the robot with respect to the map only with some uncertainty $\varepsilon < \pi$, we simply use a minimization interval $[o - \varepsilon, o + \varepsilon[$ in Equation (6.6). Thus, we can use a possibly existing compass, even if it is noisy, and furthermore have a reasonable trade-off between the running time and the localization quality.



```
List of scans deleted!
File gallery.map loaded.

Identifying reflex and nonreflex vertices ... done.
Identifying reflex and nonreflex vertices ... done.
Creating visibility cell decomposition (click through!) ... done.
Number of rays shot: 131
Creating Subdivision Graph SD ... done.
```

FIGURE 6.7: The visibility cell decomposition (preprocessing step)

The main disadvantage of this idea is that the triangle inequality does not remain true, which has the consequence that the best-matching skeleton may be missed by the localization algorithm.

Partial range scans A typical laser scanner only has a 180° scanning angle. Thus, in order to get a full 360° scan, either two laser scanners have to be used, which might raise problems concerning the robot's weight or its power consumption, or the robot has to take two 180°-scans with opposite viewing directions, which might be too slow or inaccurate due to the rotation between the two shots.

As long as we can assume that the scan direction is fixed with respect to the map (that is, the robot has a compass and always adjusts its viewing direction before taking a scan), we can easily model such a situation of a partial scan by suitably adjusting the integration interval in Equation (6.6) on page 153. The metric properties of the PCM remain untouched, but of course we have to use this modification also for the preprocessing step and cannot change the

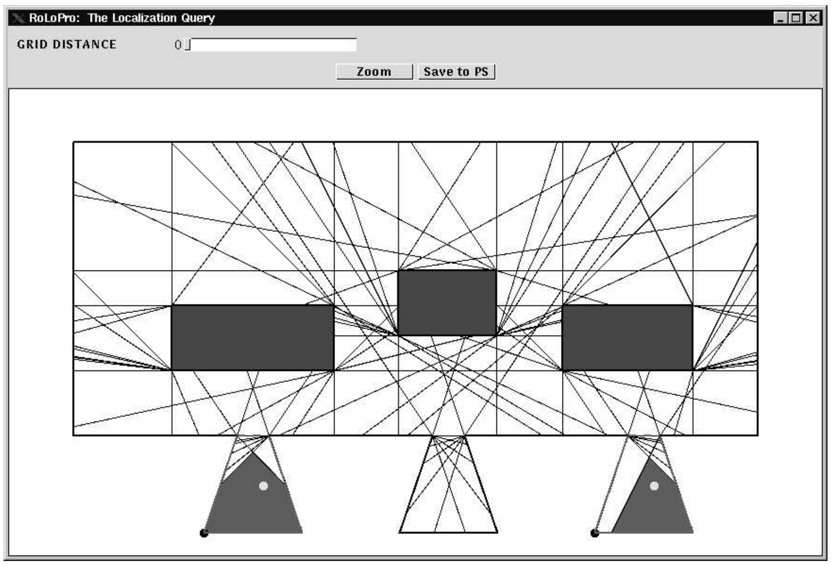


FIGURE 6.8: An *exact* localization query: A robot standing in the left niche may also be positioned in the right niche

robot's viewing direction without repeating the preprocessing procedure. For all other cases, where the robot has no compass or where the scanning interval is different from the interval used in the preprocessing step, at least the triangle property gets lost, with the same consequences as described above.

Additional coarsening step The complexity of the visibility cell decomposition, which is necessary for the geometrical approach described in Section 3.6 and which is the basis for our approach described in this chapter, is quite high (especially if we allow holes in the map). But in practice, where we have noisy sensors, many of the visibility cells need not be considered, because their skeletons differ only slightly from the skeletons of neighboring cells such that the noise of the laser scan dominates possible differences between the skeletons. Furthermore, the cells themselves are often very small such that even if the localization algorithm determines only a visibility cell in the neighborhood of the "correct" cell, the spatial displacement is often not very large and can be handled by the local matching algorithm in the postprocessing step.

Therefore, we have implemented an additional coarsening step, where neighboring cells with small distances $D(V_1^*, V_2^*)$ are combined into one bigger cell. This way, we need less space for storing the cells, less time for constructing the Monotonous Bisector Tree as well as less time for answering a localization query. Of course, this coarsening procedure reduces the quality of

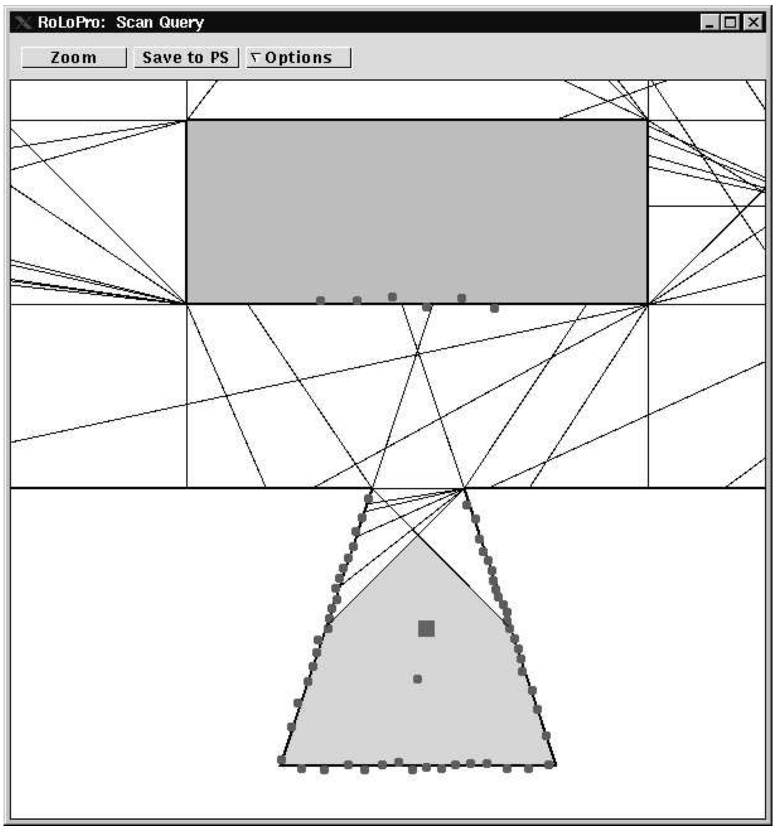
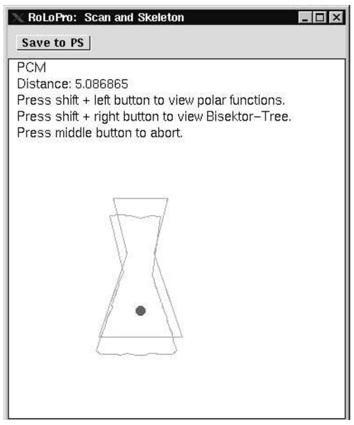


FIGURE 6.9: A localization query for a simulated *noisy* scan: The robot's position is marked by a square ≫■≪ and the center of gravity of the best-matching cell is marked by a bullet ≫●≪



PCM
Distance: 5.086865
Press shift + left button to view polar functions.
Press shift + right button to view Bisektor–Tree.
Press middle button to abort.

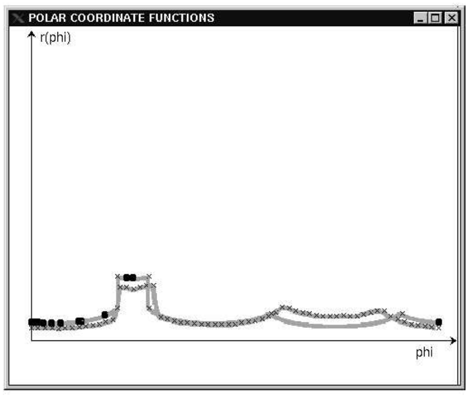


FIGURE 6.10: The PCF of the scan and the skeleton of Figure 6.9

the localization, since we obtain both, a fewer number of distance values and larger visibility cells. Thus, the threshold value for the merging step must be carefully chosen by the user.

Local queries If the robot already has a rough estimate of its position, a localization query as described in the previous sections would not be very reasonable. Therefore, we also have implemented a *local query*, where the search is restricted to the neighborhood of the appraised robot position. That is, only skeletons of cells are compared to the scan that lie within a distance to the estimated position.

6.11 Experimental Tests

We have evaluated our approaches using the implementation RoLoPro on a Sun Enterprise 250 with 512 MByte main memory, both in simulated and in real-world environments. Thereby, we used the Hausdorff distance as well as the polar coordinate metric as distance function. The criterion whether the result of a localization query was counted as a "success" or as a "failure" was quite simple: Only if the reported visibility cell contained the origin of the respective scan, the localization was regarded as successful. Of course, also less conservative success measures are conceivable, where, for example, also the spatial distance between the reported visibility cell and the scan origin is taken into account.

Simulated Environments

Figure 6.11 on the facing page depicts four representative artificial maps, which were used in order to test our algorithms. The total number of vertices typically varied between 20 and 60 and the number of generated visibility cells between 250 and 2000.

The Hausdorff distance As already stated in Section 6.10, we have neither implemented the *minimum* Hausdorff distance nor the skeleton management using the Monotonous Bisector Tree. The consequences were that firstly we had to assume a robot with a compass and secondly the localization query required a relative large amount of time (several minutes for a single query). Beyond that, we only got success rates of approximately 60 percent, even for the k-Hausdorff distance. Thus, the Hausdorff distance seems to be not very well suited for the problem of robot localization.

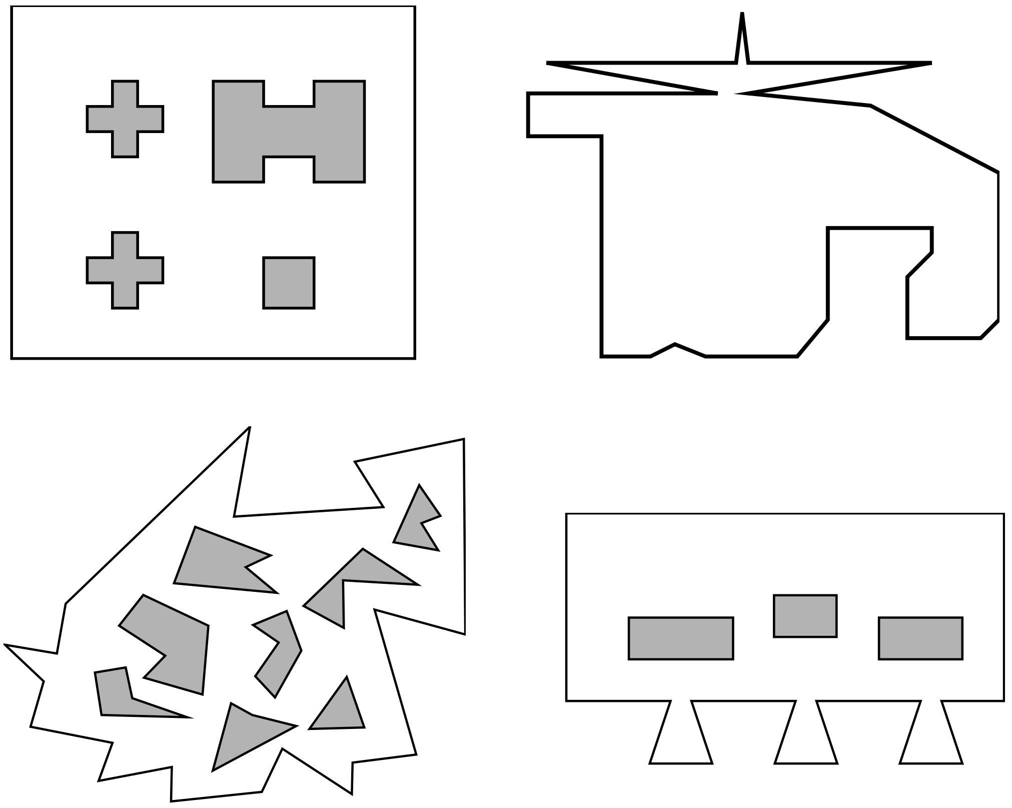


FIGURE 6.11: Artificial maps used for evaluating ROLOPRO

The polar coordinate metric In the same scenes, which we have used for evaluating the Hausdorff distance, the PCM (for a robot with a compass) behaved much better and achieved success rates of about 80 to 90 percent. Furthermore, the performance was much faster than in the above case (only a few seconds for one query). This probably was caused by the faster computation of a single distance value (linear time complexity versus an $O(n \log n)$-complexity) and by the usage of the Monotonous Bisector Tree, which saves us from visiting all skeletons.

On the other hand, when we used the variant of the PCM that minimizes over the rotation angles (that is, we assumed a robot without a compass), the success rates and the performance both became worse: The percentage of correctly localized positions dropped down to 50 to 70 percent, depending on the scene and on the number of self-similarities the environment contained. Moreover, due to the cubic time complexity of the distance computation the overall running time dramatically increased (up to fifteen minutes for one query) such that this variant could only be used either in very small scenes or in conjunction with one of the additional heuristic features of ROLOPRO (for example, a noisy compass instead of no compass at all).

Real-World Environments

On the basis of our results and experiences with the performance of the Hausdorff distance and the polar coordinate metric in the simulation, we decided to evaluate only one distance function in real-world scenes, namely the variant of the PCM that has a linear running time and requires the correct orientation of the scan with respect to the map. Two different environments were used: The Computer Science Department at the University of Würzburg and a "less friendly" supermarket environment, where the scans were very noisy. In both cases the scans were generated by a SICK LMS 200 laser scanner.

Figure 6.12 on the next page shows some localization examples for the first environment: We used a partial map of the department, which consists of 76 vertices, 54 of them reflex. This led to a total number of about 4300 visibility cells, which were finally reduced to about 3300 cells by our heuristic coarsening step (using a suitable threshold). Each of the 360°-scans consisted of 720 range measurements.

Almost all localization queries resulted in solutions like the two topmost examples in Figure 6.12. Only about 5 to 10 percent of the localization queries failed. For example, the reason for the bad result of the bottommost example in Figure 6.12 probably were some scan points at a distance of about 50 m in the right part of the scan (cut off in the figure), which were produced by scanning through a window. Since the PCM "tries" to minimize a kind of *average* quadratic error, the scan was shifted in the opposite direction to the left.

In the supermarket environment the localization approach completely failed, probably because of the very noisy range scans, which produced too much effects like in the third example of Figure 6.12 where a number of completely non-matching scan points dominates the average quadratic error in Equation (6.6). Moreover, in this environment only 180°-scans were available such that the localization problem was even harder.

6.12 Summary

We introduced an approach to tackling the problems that occur if we try to apply the idealizing scheme investigated in the preceding chapters to real-world problems. To this end, we decided to use distance functions that model the resemblance between a range scan (i. e., the sensor data) and a skeleton (i. e., the map). Furthermore, in order to efficiently compare a scan with all skeletons, we store the skeletons in a spatial data structure, the Monotonous Bisector Tree

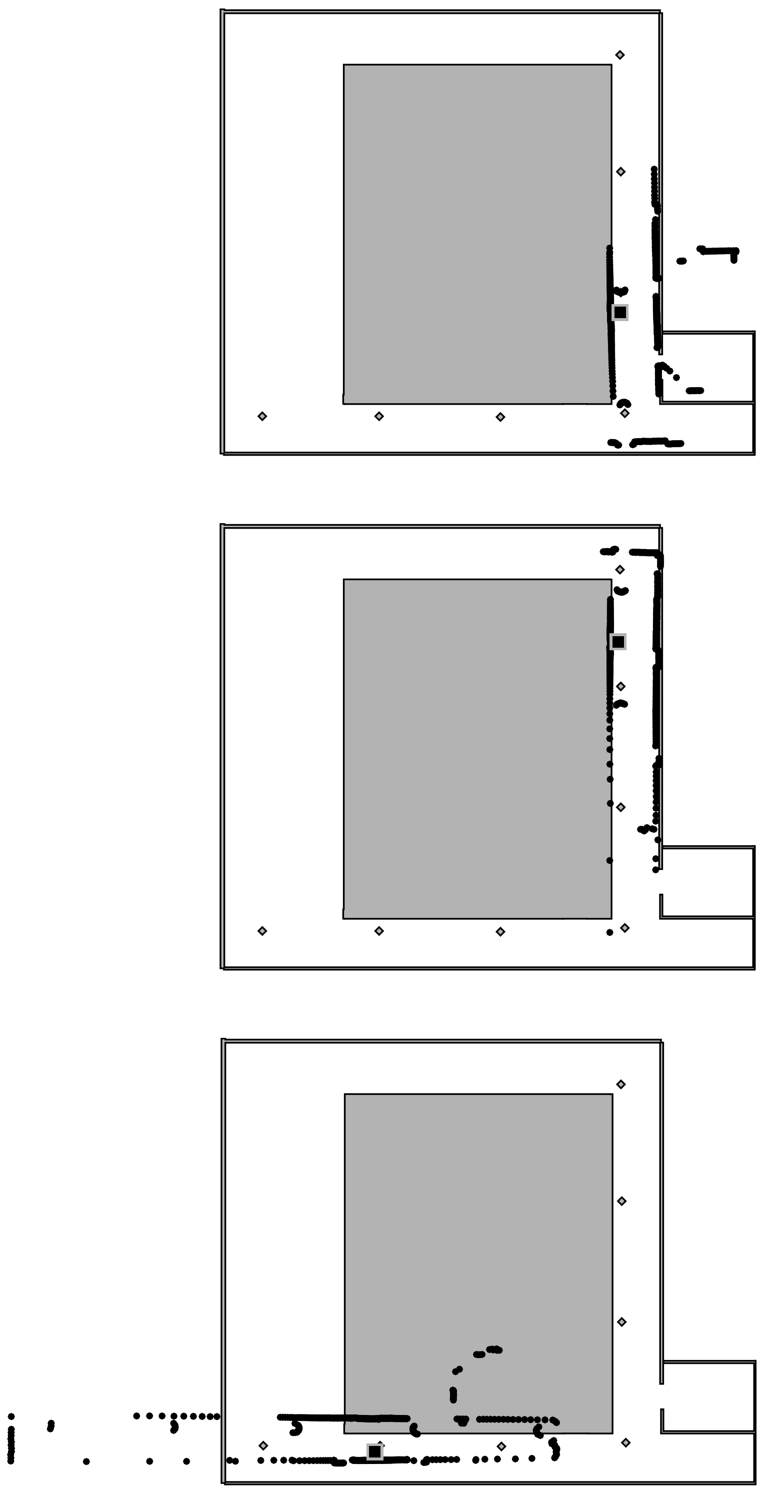


FIGURE 6.12: Real-world localization examples using the PCM: The scan origin is marked by a square ≫■≪

in our case, and use a second distance functions that describes the similarity between skeletons.

We investigated several different distance functions and presented techniques to adapt them to the special requirements of the localization problem. In particular, we introduced the polar coordinate metric (PCM) for star-shaped polygons and showed how this polygon distance can be used in our setting. Furthermore, we explained two possibilities to efficiently compute the PCM and a linear approximation of it, respectively.

We presented the implementation RoLoPro, where the idealizing scheme of Chapter 3 as well as our approach for realistic scenarios was implemented. Several distance functions (including the polar coordinate metric) and some variants were included into RoLoPro and tested in simulated environments and in real-world scenarios. Although the results using the PCM approach in small simulated and real environments were quite promising (compared to other distance functions), we also discovered the limits of this method: First, the preprocessing costs (time as well as space) are very high such that it can be used only for small scenes. Second, without any further preprocessing of the scan data the method fails if the noise of the range scan is too high. Even the PCM, which "tries" to minimize a kind of average quadratic error, does badly for very noisy scans, especially in cluttered environments. But it should be noted that compared to the remaining distance measures that can be used in our setting (e. g., Hausdorff distance, Arkin metric, symmetric difference) the polar coordinate metric performs very well due to its robustness against (small) changes of the involved polygons and its fast computability.

The localization problem seems to be one of that kind of problems where the gap between the solution of an *idealized* formulation of it and the solution of the actual *real* problem is particularly wide. Thus, the presented approach may only be *one* way of a variety of possibilities to deal with this problem. In order to find a reliable and usable solution, one surely has to combine different strategies: For example, a partitioning of the map in variably sized "regions of interest" (like in the method of Guibas et al.) combined with a suitable map and scan preprocessing, which significantly reduces the complexity, and a well-suited distance measure like, for example, the presented polar coordinate metric.

List of Figures and Algorithms

Figures are marked by »F« and algorithms are marked by »A«.

List of Definitions, Assumptions, and Observations

List of Theorems, Lemmas, and Corollaries

Bibliography

Some of the literature is also available online. Note that in this case, the given URL was up-to-date in December 2002, but may become invalid in the future.

[ABB95] H. ALT, B. BEHRENDS, AND J. BLÖMER · *Approximate Matching of Polygonal Shapes* · in »Formal Methods in 2-D Shape Analysis«, J. Csirik and H. Bunke, eds., vol. 13(3-4) of Annals of Mathematics and Artificial Intelligence, 1995, pp. 251–266. Online available at *ftp://fubinf.inf.fu-berlin.de/pub/reports/tr-b-93-10.ps.gz* . (5 citations on pages 147, 150, and 151)

[AC$^+$91] E. M. ARKIN, L. P. CHEW, D. P. HUTTENLOCHER, K. KEDEM, AND J. S. B. MITCHELL · *An Efficiently Computable Metric for Comparing Polygonal Shapes* · IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13(3), 1991, pp. 209–215. (Two citations on pages 146 and 156)

[Ada99] M. D. ADAMS · *Sensor Modelling, Design and Data Processing for Autonomous Navigation* · vol. 13 of Robotics and Intelligent Systems, World Scientific, 1999. (Two citations on pages 6 and 7)

[AF$^+$98] H. ALT, U. FUCHS, G. ROTE, AND G. WEBER · *Matching Convex Shapes with Respect to the Symmetric Difference* · Algorithmica, vol. 21, 1998, pp. 89–103. (One citation on page 147)

[AG99] H. ALT AND L. J. GUIBAS · *Discrete Geometric Shapes: Matching, Interpolation, and Approximation* · in »Handbook of Computational Geometry«, J. R. Sack and J. Urrutia, eds., Elsevier/North-Holland, 1999, pp. 121–153. Online available at *ftp://fubinf.inf.fu-berlin.de/pub/reports/tr-b-96-11.ps.gz* . (3 citations on pages 147 and 159)

[AH$^+$98] A. ANDERSSON, T. HAGERUP, S. NILSSON, AND R. RAMAN · *Sorting in Linear Time?* · Journal of Computer and System Sciences, vol. 57, 1998, pp. 74–93. Online available at *http://www.cs.lth.se/Research/Algorithms/Papers/stefan2.ps* . (One citation on page 22)

[AHU74] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN · *The Design and Analysis of Computer Algorithms* · Addison-Wesley, 1974. (3 citations on page 21)

[AKM98] P. K. AGARWAL, L. E. KAVRAKI, AND M. T. MASON, eds. · *Robotics – The Algorithmic Perspective (WAFR '98)* · A K Peters, 1998. (One citation on page 19)

[ASV97] A. ARSÉNIO AND J. SANTOS-VICTOR · *Robust Visual Tracking by an Active Observer* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 1342–1347. (One citation on page 7)

[BBN97] R. C. BOLLES, H. BUNKE, AND H. NOLTEMEIER, eds. · *Intelligent Robots – Sensing, Modelling and Planning* · vol. 27 of Machine Perception and Artificial Intelligence, World Scientific, 1997. (One citation on page 19)

[BC$^+$99] W. BURGARD, A. B. CREMERS, D. FOX, D. HÄHNEL, G. LAKEMEYER, D. SCHULZ, W. STEINER, AND S. THRUN · *Experiences with an Interactive Museum Tour-Guide Robot* · Artificial Intelligence, vol. 114(1-2), 1999, pp. 3–55. Online available at *http://www.cs.washington.edu/homes/fox/postscripts/museum-ai-00.ps.gz* . (One citation on page 1)

[BEF96] J. BORENSTEIN, H. R. EVERETT, AND L. FENG · *"Where am I?" – Systems and Methods for Mobile Robot Positioning* · University of Michigan, 1996. (7 citations on pages 2, 5, 6, and 8)

[BK$^+$99] M. BUCK, B. KLUGE, H. NOLTEMEIER, AND D. SCHÄFER · *RoLoPro – Simulationssoftware für die Selbstlokalisation eines autonomen mobilen Roboters* · in »Proceedings of the 1999 Autonome Mobile Systeme (AMS '99)«, G. Schmidt, U. Hanebeck, and F. Freyberger, eds., Springer, 1999, pp. 118–127. (One citation on page 8)

[BKN95] H. BUNKE, T. KANADE, AND H. NOLTEMEIER, eds. · *Modelling and Planning for Sensor Based Intelligent Robot Systems* · vol. 21 of Machine Perception and Artificial Intelligence, World Scientific, 1995. (One citation on page 19)

[BO83] M. BEN-OR · *Lower Bounds for Algebraic Computation Trees* · in »Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC '83)«, 1983, pp. 80–86. (One citation on page 22)

[C$^+$96] B. CHAZELLE ET AL. · *The CG Impact Task Force Report – Application Challenges to Computational Geometry* · Tech. Rep. TR-521-96, Princeton University, 1996. (One citation on page 21)

[Can88] J. F. CANNY · *The Complexity of Robot Motion Planning* · MIT Press, 1988. (One citation on page 19)

[Cas00] R. CASSINIS · *Landmines Detection Methods Using Swarms of Simple Robots* · in Pagello et al. [PG+00], pp. 212–218. (One citation on page 4)

[CBN99] H. I. CHRISTENSEN, H. BUNKE, AND H. NOLTEMEIER, eds. · *Sensor Based Intelligent Robots* · vol. 1724 of Lecture Notes in Artificial Intelligence, Springer, 1999. (4 citations on pages 19, 182, and 186)

[CE92] B. CHAZELLE AND H. EDELSBRUNNER · *An Optimal Algorithm for Intersecting Line Segments in the Plane* · Journal of the ACM, vol. 39(1), 1992, pp. 1–54. (Two citations on pages 48 and 49)

[CE+94] B. CHAZELLE, H. EDELSBRUNNER, M. GRIGNI, L. J. GUIBAS, J. E. HERSHBERGER, M. SHARIR, AND J. SNOEYINK · *Ray Shooting in Polygons Using Geodesic Triangulations* · Algorithmica, vol. 12, 1994, pp. 54–68. (One citation on page 48)

[Cha95] B. CHAZELLE · *Computational Geometry – A Retrospective* · in »Lecture Notes Series on Computing«, vol. 1, World Scientific, 1995, pp. 22–46. (One citation on page 21)

[CLR92] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVES · *Introduction to Algorithms* · MIT Press, 6th ed., 1992. (One citation on page 20)

[Cop68] E. T. COPSON · *Metric Spaces* · Cambridge University Press, 1968. (Two citations on pages 12 and 13)

[Cox91] I. J. COX · *Blanche – An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle* · IEEE Transactions on Robotics and Automation, vol. 7(2), 1991, pp. 193–204. (One citation on page 8)

[dBvK+97] M. DE BERG, M. VAN KREVELD, M. OVERMARS, AND O. SCHWARZKOPF · *Computational Geometry – Algorithms and Applications* · Springer, 1997. (5 citations on pages 14, 48, 49, and 50)

[DHS96] K. DANIILIDIS, M. HANSEN, AND G. SOMMER · *Real Time Pursuit and Vergence Control with an Active Binocular Head* · in »Proceedings of the 1996 Autonome Mobile Systeme (AMS '96)«, G. Schmidt and F. Freyberger, eds., Springer, 1996, pp. 78–87. (One citation on page 7)

[Die00] R. DIESTEL · *Graphentheorie* · Springer, 2nd ed., 2000. (One citation on page 20)

[DLR00] B. DONALD, K. LYNCH, AND D. RUS, eds. · *Algorithmic and Computational Robotics – New Directions (WAFR '2000)* · A K Peters, 2000. (One citation on page 19)

[DM95] N. I. DURLACH AND A. S. MAVOR, eds. · *Virtual Reality: Scientific and Technological Challenges* · National Academy Press, 1995. Online available at *http://books.nap.edu/catalog/4761.html*. (One citation on page 4)

[Ede87] H. EDELSBRUNNER · *Algorithms in Combinatorial Geometry* · Springer, 1987. (One citation on page 14)

[Eve95] H. R. EVERETT · *Sensors for Mobile Robots – Theory and Application* · A K Peters, 1995. (One citation on page 6)

[EW95] T. EDLINGER AND G. WEISS · *Exploration, Navigation and Self-Localization in an Autonomous Mobile Robot* · in »Proceedings of the 1995 Autonome Mobile Systeme (AMS '95)«, R. Dillmann, U. Rembold, and T. Lüth, eds., Springer, 1995, pp. 142–151. (One citation on page 8)

[FBL94] W. FEITEN, R. BAUER, AND G. LAWITZKY · *Robust Obstacle Avoidance in Unknown and Cramped Environments* · in »Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA '94)«, 1994, pp. 2412–2417. (Two citations on pages 1 and 7)

[FBT99] D. FOX, W. BURGARD, AND S. THRUN · *Markov Localization for Reliable Robot Navigation and People Detection* · in Christensen et al. [CBN99], pp. 1–20. (One citation on page 1)

[FM97] P. FREEDMAN AND P. MACKENZIE · *A Computer-Based Training Environment for Forestry Telemanipulation* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 1826–1831. (One citation on page 4)

[Fox98] D. FOX · *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation* · PhD thesis, Institute of Computer Science III, University of Bonn, Germany, December 1998. Online available at *http://www.cs.washington.edu/homes/fox/postscripts/fox-thesis.ps.gz* . (Two citations on pages 1 and 2)

[FW93] M. L. FREDMAN AND D. E. WILLARD · *Surpassing the Information Theoretic Bound with Fusion Trees* · Journal of Computer and System Sciences, vol. 47, 1993, pp. 424–436. (One citation on page 22)

[GdP98] K. GUPTA AND A. P. DEL POBIL · *Practical Motion Planning in Robotics – Current Approaches and Future Directions* · John Wiley & Sons, 1998. (One citation on page 19)

[GH$^+$87] L. J. GUIBAS, J. E. HERSHBERGER, D. LEVEN, M. SHARIR, AND R. E. TARJAN · *Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons* · Algorithmica, vol. 2, 1987, pp. 209–233. (One citation on page 48)

[GH$^+$94] K. GOLDBERG, D. HALPERIN, J.-C. LATOMBE, AND R. WILSON, eds. · *Algorithmic Foundations of Robotics (WAFR '94)* · A K Peters, 1994. (3 citations on pages 19, 183, and 184)

[GJ79] M. R. GAREY AND D. S. JOHNSON · *Computers and Intractability – A Guide to the Theory of NP-Completeness* · W. H. Freeman and Company, 1979. (One citation on page 21)

[GMR97] L. J. GUIBAS, R. MOTWANI, AND P. RAGHAVAN · *The Robot Localization Problem* · SIAM Journal on Computing, vol. 26(4), 1997, pp. 1120–1138. Online available at *http://theory.stanford.edu/people/motwani/postscripts/localiz.ps.gz* . (9 citations on pages 26, 27, 42, 46, 49, 50, and 159)

[Gut] *The Project Gutenberg*. Online available at *http://www.gutenberg.net* .

[HS91] D. HALPERIN AND M. SHARIR · *On Disjoint Concave Chains in Arrangements of (Pseudo) Lines* · Information Processing Letters, vol. 40(4), 1991, pp. 189–192. (One citation on page 128)

[HS94a] ——— · *Arrangements and their Applications in Robotics – Recent Developments* · in Goldberg et al. [GH$^+$94], pp. 495–511. (One citation on page 20)

[HS94b] ——— · *Corrigendum: On Disjoint Concave Chains in Arrangements of (Pseudo) Lines* · Information Processing Letters, vol. 51(1), 1994, pp. 53–56. (One citation on page 128)

[HS98] J. E. HERSHBERGER AND J. S. SNOEYINK · *Erased Arrangements of Lines and Convex Decompositions of Polyhedra* · Computational Geometry: Theory and Applications, vol. 9(3), 1998, pp. 129–143. Online available at *http://www.cs.ubc.ca/spider/snoeyink/papers/erased.ps.gz* . (One citation on page 128)

[HU79] J. E. HOPCROFT AND J. D. ULLMAN · *Introduction to Automata Theory, Languages, and Computation* · Addison-Wesley, 1979. (One citation on page 21)

[Joy96] D. E. JOYCE · *Euclid's Elements* · 1996. Online available at *http://aleph0.clarku.edu/˜djoyce/java/elements/toc.html* . (One citation on page 13)

[Kal60] R. E. KALMAN · *A New Approach to Linear Filtering and Prediction Problems* · Transactions of the ASME – Journal of Basic Engineering, 1960, pp. 35–45. (One citation on page 8)

[Kir83] D. G. KIRKPATRICK · *Optimal Search in Planar Subdivisions* · SIAM Journal on Computing, vol. 12, 1983, pp. 28–35. (Two citations on pages 49 and 50)

[Kle97] R. KLEIN · *Algorithmische Geometrie* · Addison-Wesley, 1997. (3 citations on pages 14, 21, and 48)

[Klu99] B. KLUGE · *Lokale Featurs – Erkennung und Matching in Lokalisationsszenarien* · Master's thesis, Department of Computer Science I, University of Würzburg, June 1999. (One citation on page 8)

[KNS00] B. KLUGE, H. NOLTEMEIER, AND D. SCHÄFER · *Feature-based Localization of an Autonomous Mobile Robot: An Experimental Case Study* · in Pagello et al. [PG+00], pp. 487–492. (One citation on page 8)

[KW99] O. KARCH AND T. WAHL · *Relocalization — Theory and Practice* · Discrete Applied Mathematics (Special Issue on Computational Geometry), vol. 93, 1999, pp. 89–108. (Two citations on pages 10 and 152)

[KZK97] M. KAM, X. ZHU, AND P. KALATA · *Sensor Fusion for Mobile Robot Navigation* · in »Proceedings of the IEEE«, vol. 85, 1997, pp. 108–119. (One citation on page 8)

[Lat91] J.-C. LATOMBE · *Robot Motion Planning* · Kluwer Academic Publishers, 1991. (One citation on page 19)

[Lat94] ——— · *Robot Algorithms* · in Goldberg et al. [GH+94], pp. 1–18. (One citation on page 19)

[LBG97] L. M. LORIGO, R. A. BROOKS, AND W. E. L. GRIMSON · *Visually-Guided Obstacle Avoidance in Unstructured Environments* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 373–379. (One citation on page 7)

[LD97] S. LACROIX AND G. DUDEK · *On the Identification of Sonar Features* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 586–592. (Two citations on pages 7 and 8)

[LDW91] J. J. LEONARD AND H. F. DURRANT-WHYTE · *Mobile Robot Localization by Tracking Geometric Beacons* · IEEE Transactions on Robotics and Automation, vol. 7(3), 1991, pp. 376–382. (One citation on page 5)

[LM97] F. LU AND E. MILIOS · *Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans* · Journal of Intelligent and Robotic Systems, vol. 18, 1997, pp. 249–275. (One citation on page 8)

[LO96] J.-P. LAUMOND AND M. OVERMARS, eds. · *Algorithms for Robotic Motion and Manipulation (WAFR '96)* · A K Peters, 1996. (One citation on page 19)

[May90] P. S. MAYBECK · *The Kalman Filter: An Introduction to Concepts* · in »Autonomous Robot Vehicles«, I. J. Cox and G. T. Wilfong, eds., Springer, 1990, pp. 193–204. (One citation on page 8)

[Meh84a] K. MEHLHORN · *Data Structures and Algorithms 1 – Sorting and Searching* · Springer, 1984. (One citation on page 22)

[Meh84b] —— · *Data Structures and Algorithms 3 – Multi-Dimensional Searching and Computational Geometry* · Springer, 1984. (Two citations on pages 48 and 49)

[MFS] *Mars Fact Sheet*. Online available at *http://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html*. (One citation on page 5)

[MN99] K. MEHLHORN AND S. NÄHER · *LEDA – A Platform for Combinatorial and Geometric Computing* · Cambridge University Press, 1999. (One citation on page 158)

[MRF] *Mars Rover Fact Sheet*. Online available at *http://telerobotics.jpl.nasa.gov/tasks/scirover/factsheet/homepage.html*. (Two citations on page 4)

[Mun75] J. R. MUNKRES · *Topology: A First Course* · Prentice-Hall, 1975. (One citation on page 16)

[Nol76] H. NOLTEMEIER · *Graphentheorie: mit Algorithmen und Anwendungen* · de Gruyter, 1976. (One citation on page 20)

[NVZ93] H. NOLTEMEIER, K. VERBARG, AND C. ZIRKELBACH · *A Data Structure for Representing and Efficient Querying Large Scenes of Geometric Objects: MB*–Trees* · in »Geometric Modelling«, G. Farin, H. Hagen, and H. Noltemeier, eds., vol. 8 of Computing Supplement, Springer, 1993, pp. 211–226. (One citation on page 148)

[O'R98] J. O'ROURKE · *Computational Geometry in C* · Cambridge University Press, 2nd ed., 1998. (One citation on page 14)

[OS97] M. OLLIS AND A. STENTZ · *Vision-Based Perception for an Automated Harvester* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 1838–1844. (One citation on page 4)

[Pap94] C. H. PAPADIMITRIOU · *Computational Complexity* · Addison-Wesley, 1994. (One citation on page 21)

[PG$^+$00] E. PAGELLO, F. GROEN, T. ARAI, R. DILLMANN, AND A. STENTZ, eds. · *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-6)* · IOS Press, 2000. (5 citations on pages 181, 184, 186, and 188)

[PM95] E. PRASSLER AND E. E. MILIOS · *Position Estimation Using Equidistance Lines* · in »Proceedings of the 1995 IEEE International Conference on Robotics and Automation (ICRA '95)«, 1995, pp. 85–92. (One citation on page 8)

[PS85] F. P. PREPARATA AND M. I. SHAMOS · *Computational Geometry – An Introduction* · Springer, 2nd ed., 1985. (4 citations on pages 14, 21, and 22)

[PS+99] E. PRASSLER, J. SCHOLZ, M. STROBEL, AND P. FIORINI · *MAid: A Robotic Wheelchair Operating in Public Environments* · in Christensen et al. [CBN99], pp. 68–95. (Two citations on page 3)

[PSS98] E. PRASSLER, J. SCHOLZ, AND M. STROBEL · *MAid: Mobility Assistance for Elderly and Disabled People* · in »Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON '98)«, 1998, pp. 2493–2498. (One citation on page 3)

[Rei79] J. H. REIF · *Complexity of the Mover's Problem and Generalizations* · in »Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS '79)«, 1979, pp. 421–427. (One citation on page 20)

[Ren94] W. D. RENCKEN · *Autonomous Sonar Navigation in Indoor, Unknown, and Unstructured Environments* · in »Proceedings of the 7th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '94)«, 1994, pp. 127–134. (Two citations on pages 1 and 7)

[RFZ99] W. D. RENCKEN, W. FEITEN, AND R. ZÖLLNER · *Relocalisation by Partial Map Matching* · in Christensen et al. [CBN99], pp. 21–35. (Two citations on pages 1 and 8)

[Rhi] *The Rhino-Project Homepage*. Online available at *http://www.cs.uni-bonn.de/~rhino/* . (One citation on page 1)

[Soi97a] M. SOIKA · *A Sensor Failure Detection Framework for Autonomous Mobile Robots* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 1735–1740. (One citation on page 7)

[Soi97b] —— · *Grid Based Fault Detection and Calibration of Sensors on Mobile Robots* · in »Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA '97)«, 1997, pp. 2589–2594. (One citation on page 7)

[SR+00] S. A. STOETER, P. E. RYBSKI, M. D. ERICKSON, M. GINI, D. F. HOUGEN, D. G. KRANTZ, N. PAPANIKOLOPOULOS, AND M. WYMAN · *A Robot Team for Exploration and Surveillance: Design and Architecture* · in Pagello et al. [PG+00], pp. 767–774. (One citation on page 4)

[SS90] J. T. SCHWARTZ AND M. SHARIR · *Algorithmic Motion Planning in Robotics* · in van Leeuwen [vL90a]. (One citation on page 19)

[Sto96] H. W. STONE · *Mars Pathfinder Microrover – A Small, Low-Cost, Low-Power Spacecraft* · in »Proceedings of the AIAA Forum on Advanced Developments in Space Robotics«, 1996. (One citation on page 4)

[SW99] J. Z. SASIADEK AND Q. WANG · *Sensor Fusion Based on Fuzzy Kalman Filtering for Autonomous Robot Vehicle* · in »Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA '99)«, 1999, pp. 2970–2976. (One citation on page 8)

[TGVS99] N. TSCHICHOLD-GÜRMAN, S. J. VESTLI, AND G. SCHWEITZER · *Operating Experiences with the Service Robot MOPS* · in »Proceedings of the 3rd European Workshop on Advanced Mobile Robots (EUROBOT '99«, 1999. Online available at *http://www.ifr.mavt.ethz.ch/research/mops/eurobot99.pdf*. (One citation on page 1)

[Vel01] R. C. VELTKAMP · *Shape Matching: Similarity Measures and Algorithms* · Tech. Rep. UU-CS-2001-03, Institute of Information and Computer Science, Utrecht University, February 2001. Online available at *http://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2001/2001-03.pdf*. (One citation on page 147)

[VH01] R. C. VELTKAMP AND M. HAGEDOORN · *State-of-the-Art in Shape Matching* · in »Principles of Visual Information Retrieval (Advances in Pattern Recognition)«, M. S. Lew, ed., Springer, 2001, pp. 87–119. (One citation on page 12)

[vL90a] J. VAN LEEUWEN, ed. · *Handbook of Theoretical Computer Science – Algorithms and Complexity* · vol. A, Elsevier/North-Holland, 1990. (3 citations on pages 21, 186, and 188)

[vL90b] ——, ed. · *Handbook of Theoretical Computer Science – Formal Models and Semantics* · vol. B, Elsevier/North-Holland, 1990. (One citation on page 21)

[VO+97] R. VOLPE, T. OHM, R. PETRAS, R. WELCH, J. B. BALARAM, AND R. IVLEV · *A Prototype Manipulation System for Mars Rover Science Operations* · in »Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)«, 1997, pp. 1486–1492. (One citation on page 4)

[VTG96] S. J. VESTLI AND N. TSCHICHOLD-GÜRMAN · *MOPS, a System for Mail Distribution in Office Type Buildings* · Service Robot: An International Journal, vol. 2(2), 1996, pp. 29–35. Online available at *http://www.ifr.mavt.ethz.ch/research/mops/servicerobot.ps*. (One citation on page 1)

[Wah97] T. WAHL · *Distanzfunktionen für Polygone und ihre Anwendung in der Roboterlokalisation* · Master's thesis, Department of Computer Science I, University of Würzburg, June 1997. (4 citations on pages 10, 152, 155, and 157)

[Wil92] D. E. WILLARD · *Applications of the Fusion Tree Method to Computational Geometry and Searching* · in »Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '92)«, 1992, pp. 286–295. (One citation on page 22)

[WJvP00] J. WEBER, K.-W. JÖRG, AND E. VON PUTTKAMER · *APR – Global Scan Matching Using Anchor Point Relationships* · in Pagello et al. [PG$^+$00], pp. 471–478. (One citation on page 8)

[XvB$^+$95] H. XU, H. VAN BRUSSEL, J. DE SCHUTTER, AND J. VANDORPE · *Sensor Fusion and Positioning of the Mobile Robot LiAS* · in »Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-4)«, U. Rembold, R. Dillmann, L. O. Hertzberger, and T. Kanade, eds., IOS Press, 1995, pp. 246–253. (One citation on page 8)

[Yao90] F. F. YAO · *Computational Geometry* · in van Leeuwen [vL90a]. (3 citations on pages 14, 48, and 49)

[YI00] I. E. YAIRI AND S. IGI · *A Self-sustained Moving Support System for Aged and Disabled Persons* · in Pagello et al. [PG$^+$00], pp. 692–697. (One citation on page 3)

[Zha97] Y. ZHAO · *Vehicle Location and Navigation Systems* · Artech House, 1997. (One citation on page 19)

Index

Underlined page numbers refer to the page where a notion is defined, first used, or described in more detail than on the other pages.