Motion Coordination for a Mobile Robot in Dynamic Environments

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

vorgelegt von
Boris Kluge
aus
Creglingen

Würzburg 2004

Eingereicht am: _____

bei der Fakultät für Mathematik und Informatik

1. Gutachter: _____

2. Gutachter: _____

Tag der mündlichen Prüfung: _____

For Luise, Hubert, and Ulrike.

Abstract

Generating coordinated motion for a mobile robot operating in natural, continuously changing environments among moving obstacles such as humans is a complex task which requires the solution of various sub problems. In this thesis, we will cover the topics of perception and navigation in dynamic environments, as well as reasoning about the motion of the obstacles and of the robot itself.

Perception is mainly considered for a laser range finder, and an according method for obstacle detection and tracking is proposed. Network optimization algorithms are used for data association in the tracking step, resulting in considerable robustness with respect to clutter by small objects.

Navigation in general is accomplished using an adaptation of the velocity obstacle approach to the given vehicle kinematics, and cooperative motion coordination between the robot and a human guide is achieved using an appropriate selection rule for collision-free velocities.

Next, the robot is enabled to compare its path to the path of a human guide using one of a collection of presented distance measures, which permits the detection of exceptional conditions. Furthermore, a taxonomy for the assessment of situations concerning the robot is presented, and following a summary of existing approaches to more intelligent and comprehensive perception, we propose a method for obstruction detection.

Finally, a new approach to reflective navigation behaviors is described where the robot reasons about intelligent moving obstacles in its environment, which allows to adjust the character of the robot motion from regardful and defensive to more self-confident and aggressive behaviors.

Acknowledgments

I am grateful to all the people who helped me in writing this thesis at various stages. Special thanks go to Prof. Hartmut Noltemeier, my thesis advisor at the University of Würzburg for all his support and understanding throughout the work on this thesis. For being provided with the opportunity to work with inspiring colleagues at an extraordinary institute, I am deeply indebted to Prof. Franz Josef Radermacher, head of the Research Institute for Applied Knowledge Processing (FAW) at the University of Ulm, and Dr. Thomas Kämpke, head of the autonomous systems department at the FAW. Especially, numerous discussions with my former colleague Christian Schlegel on topics ranging (besides robotics) from software engineering to computational geometry are thankfully acknowledged. Finally, I want to mention a wonderful time working with the entire team of the MORPHA project.

Parts of this work were supported by the German Department for Education and Research (BMB+F) under grant no. 01 IL 902 F6 as part of the project MORPHA.

Contents

Chapter 1

Introduction

The interaction and cooperation between a human and intelligent robot systems is a research topic which has attracted much attention recently. Under captions such as 'human-friendly robotics' or 'human-robot co-existence' this field covers a large variety of aspects. These aspects reach from a human-robot communication based on 'human-friendly' communication channels such as natural language, gestures, mimics, over an understanding of the context of a task or an understanding of situations where human and robot have to interact to achieve a common task, to physical interaction (robot touches human, human touches robot) and coordination of motion and actions of the human and the robot.

When talking about interaction, the notions of *coordination* and *cooperation* can be distinguished, as Webster's Dictionary (Gove, 1993) defines *coordination* as a

> *combination in suitable relation for most effective or harmonious results*,

and *cooperation* is characterized by a

> *common effort or labor; association of persons for their common benefit*.

Accordingly, we will talk of *coordination* in situations where agents (humans or the robot) pursue their goals while avoiding mutual obstruction, and as a special case of coordination, we will talk of *cooperation* when the involved agents pursue a common goal.

This thesis addresses various problems which arise when the motion of a mobile robot has to be coordinated with the motion of humans in its environment, including the perception, navigation and collision avoidance in such an environment, as well as reasoning about the motion of the obstacles and the robot itself.

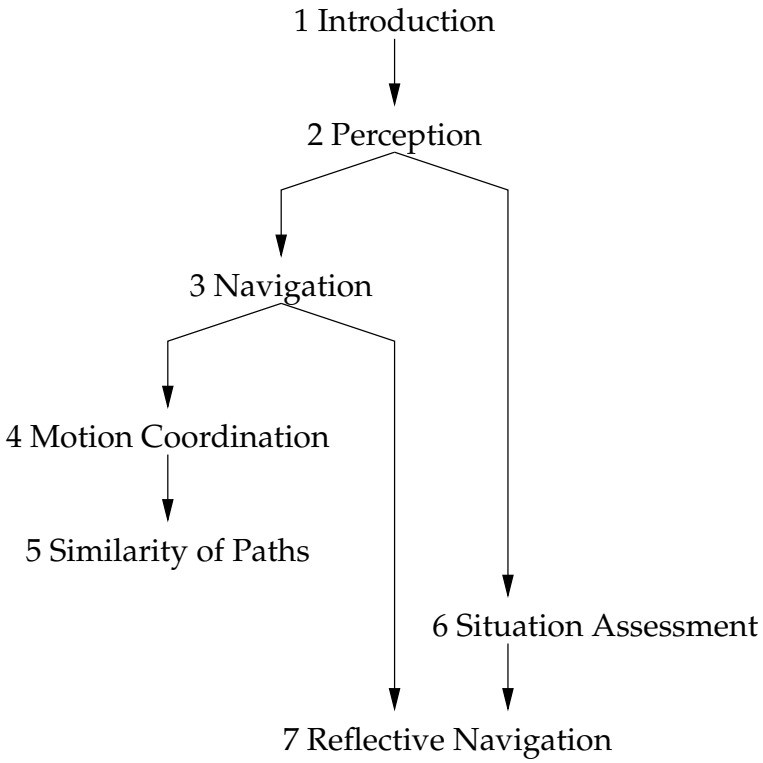


Figure 1.1: Structure of the thesis

Figure 1.2: Wheelchair 'MAid' (mobility aid)

1.1 Overview

The overall structure of this thesis is illustrated by Figure 1.1 together with implications between the respective chapters. In Chapter 2, the problem of perception in a dynamic environment is addressed, which serves as a basis for further reasoning and navigation. Chapter 3 is dedicated to the task of navigation in general and collision-free motion among moving obstacles specifically. Cooperative motion coordination between a human guide and the robot is achieved using the methods presented in Chapter 4. The following Chapter 5 deals with distance measures for paths which can be used to assess the quality of a coordinating behavior. Chapter 6 presents related work on advanced perceptual techniques for situation assessment and an approach to detect deliberate obstructions of a mobile robot. Finally, Chapter 7 proposes an approach to reflective navigation, which means that the robot reasons about the perception and velocity selection of its obstacles and integrates these findings into its own decision making.

The robotic wheelchair 'MAid' (mobility aid (Prassler et al., 1998b), see Figure 1.2) has been used as a target for the implementation of algorithms and experimental evaluation as presented in this thesis. The described system is equipped with a SICK laser range finder and an ultrasonic sensing system. Computations are performed on an on-board PC (Intel Pentium II, 333 MHz, 64 MB RAM) running Linux. The laser range finder is used to observe the environment, whereas the ultrasonic sensors help to avoid collisions with obstacles that are invisible to the range finder.

1.2 Notations

The following conventions and notations conferring to robotics have been adopted from Latombe (1991) and Bruyninckx and De Schutter (2001). The definitions addressing graph theory are mainly taken from Krumke et al. (2000) and sometimes inspired by Bang-Jensen and Gutin (2001). Furthermore, the web-site provided by Weisstein (2004) has been useful in many places.

1.2.1 Robotics

Rigid objects like a robot or an obstacle are denoted by uppercase letters like A or B. For a sequence of similar objects, we will write $A_1, A_2, \ldots$ or $B_1, B_2, \ldots$.

The workspace of a robot is denoted by $\mathcal{W}$. The configuration space is denoted by $\mathcal{C}$, or by $\mathcal{C}_A$ if we want to clarify to which object the configuration space belongs. Vectors are written in bold face, like $\mathbf{p}, \mathbf{v} \in \mathbb{R}^2$.

A coordinate system is a reference frame which is used to describe the position and orientation of an object. In this thesis, coordinate systems will be rigidly attached to an

object or to a space, and each object has at most one coordinate system attached to it. For an object A, its coordinate system will be denoted by $\langle A \rangle$, e.g. the coordinate system $\langle B_i \rangle$ is attached to object B_i (an obstacle, or a robot), and the coordinate system $\langle \mathcal{W} \rangle$ is attached to the workspace $\mathcal{W}$. The axes of a plane coordinate system $\langle A \rangle$ are denoted by $\hat{x}_A$ and $\hat{y}_A$.

A *path of an object* is a planar curve which maps time to the position of the object. A *trajectory of an object* is a curve which maps time to the configuration of the object, containing position and orientation.

1.2.2 Graphs

Definition 1.1 (Directed Graph)
A **directed graph** $G = (V, A, \alpha, \omega)$ *consists of a finite, non-empty set* V *of* **vertices**, *a finite set* A *of* **arcs**, *and two functions* $\alpha, \omega : A \to V$.

For an arc $a \in A$, *the vertex* $\alpha(a)$ *is called the* **source vertex of** a, *and the vertex* $\omega(a)$ *is called the* **target vertex of** a. *An arc* $a \in A$ *is called* **incident** *to its source and target vertices* $\alpha(a)$ *and* $\omega(a)$. *For a vertex* $v \in V$, *the arc set* $A^+(v) = \{a \in A \mid \alpha(a) = v\}$ *is called the* **leaving arcs of** v, *the set of vertices* $succ(v) = \{\omega(a) \mid a \in A^+(v)\}$ *is called the* **successors of vertex** v, *and the number* $d^+(v) := |A^+(v)|$ *is called the* **out-degree of** v. *Analogously, the arc set* $A^-(v) = \{a \in A \mid \omega(a) = v\}$ *is called the* **entering arcs of** v, *the set of vertices* $pred(v) = \{\alpha(a) \mid a \in A^-(v)\}$ *is called the* **predecessors of vertex** v, *and the number* $d^-(v) := |A^-(v)|$ *is called the* **in-degree of** v. *The number* $d(v) := d^+(v) + d^-(v)$ *is called the* **degree** *of a vertex* $v \in V$ *and is the number of arcs* $a \in A$ *which are incident to* v.

An arc $a \in A$ *with* $\alpha(a) = \omega(a)$ *is called* **loop**. *Two arcs* $a_1, a_2 \in A$ *are called* **parallel**, *if* $a_1 \neq a_2$ *and* $\alpha(a_1) = \alpha(a_2)$ *and* $\omega(a_1) = \omega(a_2)$. *Two arcs* $a_1, a_2 \in A$ *are called* **inverse**, *if* $\alpha(a_1) = \omega(a_2)$ *and* $\omega(a_1) = \alpha(a_2)$.

A directed graph $G = (V, A, \alpha, \omega)$ *is called* **simple**, *if it has neither loops nor inverse arcs. In that case, it may be denoted as* $G = (V, A)$ *with* $A \subseteq V \times V$, *and the functions* α *and* ω *are defined implicitly such that* $a = (\alpha(a), \omega(a))$ *holds.*

A directed graph $G = (V, A, \alpha, \omega)$ *is called* **symmetric**, *if for any arc* $a \in A$ *there is an inverse arc* $\bar{a} \in A$. *In a simple, symmetric directed graph, the inverse is uniquely defined.*

Definition 1.2 (Subgraph of a Directed Graph)
For a directed graph $G = (V, A, \alpha, \omega)$, *the directed graph* $G[V'] = (V', A', \alpha', \omega')$ *with*

- $V' \subseteq V$,
- $A' = \{a \in A \mid \alpha(a) \in V' \text{ and } \omega(a) \in V'\}$,
- $\alpha' = \alpha|_{A'}$, *and*
- $\omega' = \omega|_{A'}$

is called the **subgraph of** G **induced by** V'.

Definition 1.3 (Bipartite Graph)
A directed graph $G = (V, A, \alpha, \omega)$ *is called* **bipartite**, *if there is a partition of the vertices* $V = S \cup T$, $S \cap T = \emptyset$, *such that for any arc* $a \in A$, *either* $\alpha(a) \in S$ *and* $\omega(a) \in T$, *or* $\alpha(a) \in T$ *and* $\omega(a) \in S$.

Definition 1.4 (Matching in a Directed Graph)
A **matching** M *in a directed graph* $G = (V, A, \alpha, \omega)$ *is a set of arcs* $M \subseteq A$, *such that each vertex* $v \in V$ *is incident to at most one arc* $a \in M$ *of the matching. A matching* M *is called* **perfect**, *if* $\alpha(M) \cup \omega(M) = V$, *that is, for each vertex* $v \in V$ *there is an incident arc* $a \in M$ *in the matching.*

Definition 1.5 (Paths in Directed Graphs)
Let $G = (V, A, \alpha, \omega)$ *be a directed graph. Let* $p = (v_1, a_1, v_2, a_2, \ldots, a_n, v_{n+1})$ *be a finite sequence of vertices* $v_i \in V$ *and arcs* $a_i \in A$ *with* $\alpha(a_i) = v_i$ *and* $\omega(a_i) = v_{i+1}$ *for* $1 \leq i \leq n$. *Then, this sequence* p *is called a* **path in** G, *the set* $A(p) = \{a_i \mid 1 \leq i \leq n\}$ *is called* **the arcs of** p, *and the set* $V(p) = \{v_i \mid 1 \leq i \leq n+1\}$ *is called* **the vertices of** p. *The* **length** *of path* p *is the number of arcs in this sequence. Furthermore, we will write* $\alpha(p) = v_1$ *for the* **initial vertex** *of* p *and* $\omega(p) = v_{n+1}$ *for the* **terminal vertex** *of* p.

For a path p *and* $v \in V(p)$ *we say* p *is a path* **through** v, *or the path* p **traverses** v. *For a path* p *and* $a \in A(p)$ *we say* p *is a path* **through** a, *or the path* p **traverses** a.

A path p *in* G *is called* **closed** *or* **circuit in** G, *if* $\alpha(p) = \omega(p)$. *A path* $p =$ *in* G *is called* **open** *if* p *is not closed.*

A path $p = (v_1, a_1, v_2, a_2, \ldots, a_n, v_{n+1})$ *in* G *is called* **simple**, *if* $a_i \neq a_j$ *for* $i \neq j$, *i.e. no arc is traversed more than once.*

A path $p = (v_1, a_1, v_2, a_2, \ldots, a_n, v_{n+1})$ *in* G *is called* **elementary**, *if* $v_i \neq v_j$ *for* $1 \leq i, j \leq n$, $i \neq j$, *and* $v_{n+1} \neq v_i$ *for* $1 < i \leq n$, *i.e. no vertex is traversed more than once, unless the path is closed where the initial vertex of* p *appears again as terminal vertex of* p.

Definition 1.6 (Chains and Cycles in Directed Graphs)
Let $G = (V, A, \alpha, \omega)$ *be a directed graph. A* **chain in** G *is a finite sequence*

$$s = (v_1, \delta_1 a_1, v_2, \delta_2 a_2, \ldots, v_n, \delta_n a_n, v_{n+1})$$

with $\delta_i \in \{-, +\}$, $a_i \in A$ *for* $1 \leq i \leq n$, *and* $v_i \in V$ *for* $1 \leq i \leq n+1$, *such that* $\alpha(\delta_i a_i) = v_i$ *and* $\omega(\delta_i a_i) = v_{i+1}$ *for* $1 \leq i \leq n$, *where*

$$\alpha(\delta a) = \begin{cases} \alpha(a) & \text{if } \delta = \text{`+'}, \\ \omega(a) & \text{else}, \end{cases} \tag{1.1}$$

$$\omega(\delta a) = \begin{cases} \omega(a) & \text{if } \delta = \text{`+'}, \\ \alpha(a) & \text{else}. \end{cases} \tag{1.2}$$

The vertex $\alpha(s) = v_1$ *is called* **initial vertex of** s*, and the vertex* $\omega(s) = v_{n+1}$ *is called* **terminal vertex of** s*.*

The chain s *is called* **simple***, if* $a_i \neq a_j$ *for* $i \neq j$, $1 \leq i, j \leq n$*. The chain* s *is called* **reduced***, if* $a_i \neq a_{i+1}$ *for* $1 \leq i < n$*, and* $a_1 \neq a_n$ *if* $v_1 = v_{n+1}$ *(i.e. the chain is "locally simple" and there is no "U-turn").*

The chain s *is called* **elementary***, if* $v_i \neq v_j$ *for* $1 \leq i < j \leq n$*, and* $v_{n+1} \neq v_i$ *for* $1 < i \leq n$*.*

If $v_{n+1} = v_1$*, the chain is called* **closed chain** *or* **cycle in** G*. A chain in* G *is called* **open***, if it is not closed.*

For a chain $s = (v_1, \delta_1 a_1, v_2, \ldots, \delta_n a_n, v_{n+1})$ *we will write* $\delta a \in s$ *if there is an* $i \in \{1, 2, \ldots, n\}$ *with* $\delta_i = \delta$ *and* $a_i = a$*.*

For a chain $s = (v_1, \delta_1 a_1, v_2, \ldots, \delta_n a_n, v_{n+1})$*, we will denote its* **inverse chain** *by*

$$\bar{s} = (v_{n+1}, \bar{\delta}_n a_n, v_n, \ldots, v_2, \bar{\delta}_1 a_1, v_1), \tag{1.3}$$

where

$$\bar{\delta} = \begin{cases} - & \text{if } \delta = '+', \\ + & \text{else.} \end{cases} \tag{1.4}$$

For two chains $s = (v_1, \delta_1 a_1, \ldots, \delta_n a_n, v_{n+1})$ *and* $s' = (v'_1, \delta'_1 a'_1, \ldots, \delta'_m a'_m, v'_{m+1})$ *with* $\omega(s) = \alpha(s')$*, we will call*

$$s \cdot s' := (v_1, \delta_1 a_1, \ldots, \delta_n a_n, v'_1, \delta'_1 a'_1, \ldots, \delta'_m a'_m, v'_{m+1}) \tag{1.5}$$

the **concatenation of** s **and** s'*.*

For chains of non-zero length we may omit denoting the vertices. Alternatively, when the underlying graph does not contain parallel arcs, we may omit denoting the arcs.

Furthermore, we will also write a *in short for* $+a$ *and* $\bar{a}$ *in short for* $-a$*.*

Definition 1.7 (Connectedness of Directed Graphs)
A directed graph $G = (V, A, \alpha, \omega)$ *is called* **weakly connected***, if for any two vertices* $v_1, v_2 \in V$ *there is a chain* s *in* G *with initial vertex* v_1 *and terminal vertex* v_2*.*

A directed graph $G = (V, A, \alpha, \omega)$ *is called* **strongly connected***, if for any two vertices* $v_1, v_2 \in V$ *there is a path* p *in* G *with initial vertex* v_1 *and terminal vertex* v_2*.*

Definition 1.8 (Flow in a Directed Graph)
Let $G = (V, A, \alpha, \omega)$ *be a directed graph. Let* $e : V \to \mathbb{R}$ *be a vertex label, the* **excess** *of a vertex. Let* $u : A \to \mathbb{R}_0^+$ *and* $c : A \to \mathbb{R}_0^+$ *be arc labels, the* **capacity** *and the* **cost** *of an arc. Now we can ask for an arc label* $f : A \to \mathbb{R}_0^+$*, called a* **flow in** G*, which complies with the* **mass balance condition**

$$\sum_{a \in A^-(v)} f(a) - \sum_{a \in A^+(v)} f(a) + e(v) = 0 \tag{1.6}$$

for each vertex $v \in V$ *and the* **capacity bound**

$$f(a) \leq u(a) \tag{1.7}$$

for each arc $a \in A$. *Such a label* f *is called a* **feasible flow** *in* G.

Often only a special case is considered, where there is exactly one **source vertex** $s \in V$ *with* $e(s) > 0$, *and exactly one* **sink vertex** $t \in V$ *with* $e(t) < 0$. *In this case, a well known problem is the question for a* **maximum flow**, *that is, how large may the amount of excess* $e(s) = -e(t)$ *be such that there is still a feasible flow in* G.

If there is a feasible flow in a given labeled graph, one can search for a **feasible flow** f^* **with minimal cost**

$$\sum_{a \in A} f^*(a) \cdot c(a) = \min_{f \text{ is a feasible flow}} \left(\sum_{a \in A} f(a) \cdot c(a) \right) \tag{1.8}$$

among all feasible flows f.

There are several algorithms which solve these problems efficiently. Implementations are available for example via the LEDA library (Mehlhorn and Näher, 1999). For more details on graph theory and network optimization see the monographs by Bang-Jensen and Gutin (2001), Ahuja et al. (1993), Noltemeier (1976), or Lawler (1976).

Definition 1.9 (Undirected Graph)
An **undirected graph** $G = (V, E, \gamma)$ *consists of a non-empty finite set of vertices* V, *a finite set of edges* E, *and a function* $\gamma : E \to \{\{v_1, v_2\} \subseteq V\}$ *which maps an edge* $e \in E$ *to its incident vertices* $\gamma(e) = \{u, v\} \subseteq V$.

An edge $e \in E$ *is called* **loop**, *if* $|\gamma(e)| = 1$. *Two edges* $e_1, e_2 \in E$ *are called* **parallel**, *if* $e_1 \neq e_2$ *and* $\gamma(e_1) = \gamma(e_2)$.

An undirected graph $G = (V, E, \gamma)$ *is called* **simple**, *if* G *has no loops and no parallel edges. In that case,* G *may be denoted as* $G = (V, E)$ *with* $E \subseteq \{\{v_1, v_2\} \subseteq V \mid v_1 \neq v_2\}$.

Definition 1.10 (Paths in Undirected Graphs)
Let $G = (V, E, \gamma)$ *be a directed graph. Let* $p = (v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1})$ *be a finite sequence of vertices* $v_i \in V$ *and edges* $e_i \in E$ *with* $\gamma(e_i) = \{v_i, v_{i+1}\}$ *for* $1 \leq i \leq n$. *Then, this sequence* p *is called a* **path in** G, *the set* $E(p) = \{e_i \mid 1 \leq i \leq n\}$ *is called* **the edges of** p, *and the set* $V(p) = \{v_i \mid 1 \leq i \leq n+1\}$ *is called* **the vertices of** p. *The* **length** *of path* p *is the number of edges in this sequence. Furthermore, we will write* $\alpha(p) = v_1$ *for the* **initial vertex** *of* p *and* $\omega(p) = v_{n+1}$ *for the* **terminal vertex** *of* p.

For a path p *and* $v \in V(p)$ *we say* p *is a path* **through** v, *or the path* p **traverses** v. *For a path* p *and* $e \in E(p)$ *we say* p *is a path* **through** e, *or the path* p **traverses** e.

A path p *in* G *is called* **closed** *or* **cycle in** G, *if* $\alpha(p) = \omega(p)$. *A path* $p =$ *in* G *is called* **open** *if* p *is not closed.*

A path $p = (v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1})$ *in* G *is called* **simple**, *if* $e_i \neq e_j$ *for* $i \neq j$, *i.e. no arc is traversed more than once.*

A path $p = (v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1})$ *in* G *is called* **elementary**, *if* $v_i \neq v_j$ *for* $1 \leq i < j \leq n$, *and* $v_{n+1} \neq v_i$ *for* $1 < i \leq n$, *i.e. no vertex is traversed more than once, unless the path is closed and the initial vertex of* p *appears again as terminal vertex of* p.

Chapter 2

Perception in Dynamic Environments

2.1 Introduction

Today many existing robot systems are not designed to cope well with rapidly changing, dynamic environments. If there is an unforeseen situation, like a human entering the working cell of an assembly robot or people crossing a mobile robot's desired path, at best the robot stops and possibly tries to evade this obstacle in order not to hurt anybody. On the other hand, such a mobile robot might be easily obstructed by a person playing jokes on it. So this lack of awareness is a serious obstacle to a widespread use of such systems, and a basic requirement to this awareness is continuous observation of objects surrounding the robot, which we will address in this chapter of the thesis.

2.1.1 Related Work

Tracking human motion with computer vision is an active field of research (Moeslund, 1999), most approaches using video image sequences. Range images are for example used in intelligent vehicles or driver assistance systems (Meier and Ade, 1998; Sobottka and Bunke, 1998). An object tracking system for a mobile robot using a laser range finder had been designed by Prassler et al. (1998a) previously, and was replaced by the system which is presented in this chapter.

There is a correspondence problem in stereo vision where features from two camera images are to be matched, which is similar to the data association problem encountered when tracking multiple moving objects. As an alternative to the widely used dynamic programming approach, Roy and Cox (1998) use maximum-flow computations in graphs to solve this problem, yielding considerably better results, but unfortunately at increased computational costs.

Associating multiple moving point objects at successive moments in time by comput-

ing minimum cost matchings in bipartite graphs is mentioned by Ahuja et al. (1993) as an application example. The network optimization approach which is presented in this chapter is inspired by this application example, and by the relationship between matchings and flows in bipartite graphs (Ahuja et al., 1993; Lawler, 1976).

2.1.2 Overview

Perception in dynamic environments can be accomplished by various sensors, however, not all of them are equally suited for this task, and we are focusing mainly on laser range finders in this chapter. Our goal is to track the objects around the robot, which is performed by repeated execution of the following steps. At the beginning of each cycle a scan image of the environment is taken. This scan is segmented (Sect. 2.3.1) and further split into point sets representing approximately convex objects (Sect. 2.3.2). Then, the object matching problem is considered as a minimum cost transportation problem, where the shape length of objects is transported from scan to scan at costs depending on the relative position and shape of the objects (Sect. 2.4.1). Results of experiments are shown (Sect. 2.5) and discussed (Sect. 2.6) before concluding this part of the thesis (Sect. 2.7).

2.2 Sensors

Various types of sensors are being used on mobile robots in order to perceive the environment. Typically, ultrasonic proximity sensors, one or more cameras for computer vision, and laser scanners can be be found, each of which has certain advantages and disadvantages.

Recently, some researches started to explore the use of even more exotic sensors like scanning body heat detectors (Cattin, 2002) and stereo microphones (Walthelm and Litza, 2003) for human-robot interaction. But these are considered to be less relevant for the topic of this thesis.

2.2.1 Computer Vision

Using video cameras for mobile robot perception is an appealing idea which is motivated as analogon to human vision. Consequently, research in computer vision has a long tradition. However, due to their susceptibility to changes of lighting conditions and the required bandwidth, computer vision approaches are not yet suitable for many fields. Especially robustness is a problem when it comes to service robots operating under changing conditions, for example outdoors and exposed to the sun.

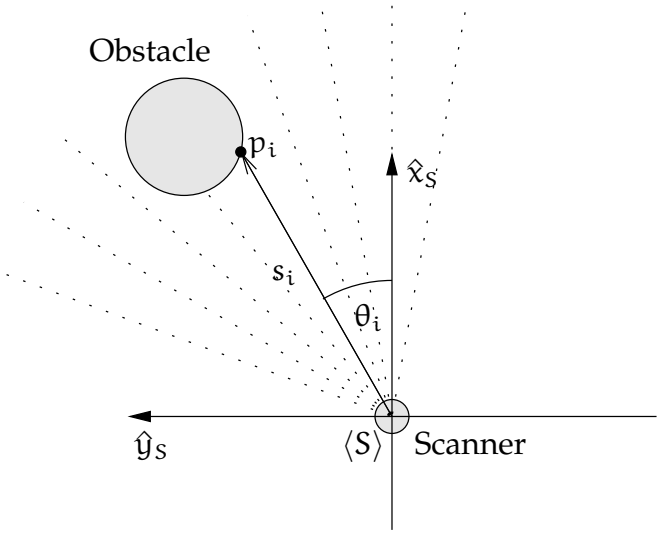


Figure 2.1: Laser Scan, Distance s_i and Direction θ_i

2.2.2 Ultrasonic Sensors

A simple way of sensing distances to obstacles in the environment is based on the emission of ultrasonic pulses and the measurement of the time until an echo returns. Since the width of the sonic beam can range from $\pm 30^\circ$ up to $\pm 60^\circ$ and more, this technique is suitable to detect obstacles in arbitrary directions. However, this also means a lack of angular resolution, which turns sonar sensors unsuitable for tracking multiple objects.

2.2.3 Laser Scanners

Laser scanners are devices which measure distances to opaque obstacles at constant angular intervals with high spatial and temporal resolution, and are thereby well suited for mobile robot applications.

A laser scan image is given by a sequence $S = (s_1, \ldots, s_L)$ of samples $s_i \in \mathbb{R}_0^+$ representing the distances from the range finder to the closest opaque obstacles in the plane of sensing. Let θ_i be the direction in which distance sample s_i is measured, and without loss of generality $\theta_i \leq \theta_j$ for $i \leq j$ (see Figure 2.1). Clearly, we may associate each distance sample s_i to the point $p_i = (s_i \cos\theta_i, s_i \sin\theta_i) \in \mathbb{R}^2$ in the frame of the laser scanner device.

2.3 Finding Objects

Objects are extracted from laser scan images by two heuristic steps. At first the scan is segmented into densely sampled parts. In the second step these parts are split into subsequences describing "almost convex" objects.

2.3.1 Scan Segmentation

Consider two adjacent distance samples s_i and s_{i+1}. Intuitively, if they are taken from the same obstacle, their values will be similar. On the other hand, if one of these samples is taken from a different obstacle, which is located in front of or behind the other sample's obstacle, we will encounter a significant difference $|s_i - s_{i+1}|$ in these two distances. Thus it is plausible to use such distance gaps as split positions for the sequence of samples in order to find distinct objects. A threshold value δ_{max} is chosen in advance for the maximal allowed distance gap. The result of the scan segmentation is a finite sequence $((s_1, \ldots, s_{i_1-1}), (s_{i_1}, \ldots, s_{i_2-1}), \ldots, (s_{i_p}, \ldots, s_L))$ of subsequences of S where indices $i_1, i_2, \ldots, i_p$ denote split positions such that $|s_{i_{k-1}} - s_{i_k}| > \delta_{max}$ for $k = 1, 2, \ldots, p$.

Finding the Threshold

In order to find an adequate threshold value δ_{max} as required by the approach described above, we examine the distribution of distances between successive scan points for a collection of different scans. Figure 2.2 visualizes these distributions in histogram form. Note that distances larger than 0.8 m are not shown.

Two phenomena are prominent. Firstly, the distributions exhibit a bulge at short distances, as we expect it from neighboring scan points taken form the same object. Secondly, there are sporadic occurrences of larger distances as expected from neighboring scan points taken from different objects. The task is now to identify the distance where the bulge ends. Considering the histograms, a threshold value of $\delta_{max} = 0.18\,\mathrm{m}$ appears reasonable.

Adaptive Scan Segmentation

Since adequateness of parameter values often depend on the environment, adaptive approaches are of interest when targeting varying environments. For an adaptive scan segmentation, automated detection of the extents of the bulge is needed.

We conducted some experiments with an approach where the lower distance bound of the first histogram bin with zero count was taken as the threshold. Clearly this is not a parameter free approach, since the result still depends on the size of the histogram bins.

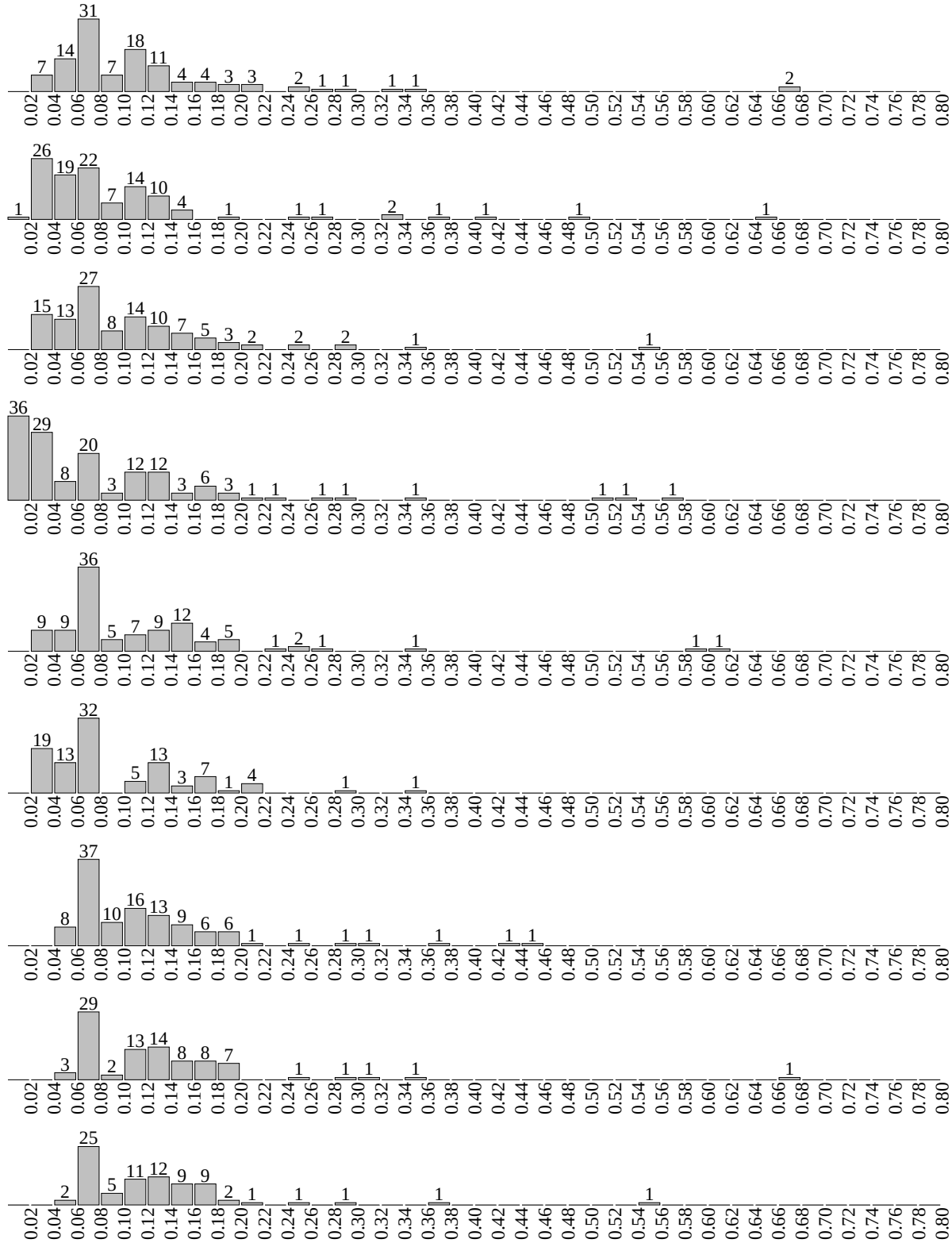


Figure 2.2: Distance Histograms for Successive Scan Points

But one might find that finding a reasonable bin size is rather easy, since the approach is able to adapt itself to the specific environment.

For single scans, results have been reasonable, but for a sequence of scans, different thresholds in successive scans lead to an unstable object extraction which degraded the overall performance of the tracking system.

2.3.2 Object Extraction

The subsequences of sample points yielded by the preceding step are divided further, as we would otherwise encounter problems with objects situated too close to other objects, for example humans leaning against or walking close to a wall or other humans.

In order to extract distinct objects from a scan point segment, a model of the objects is required, i.e. a specification of possible shapes. For example if we knew that all objects were rectangular boxes (or some other specific geometric shape), that information could be used to properly split up scan segments. However, every further assumption about the environment is also a requirement to the environment: if the objects are not from expected shape classes, the extraction step may fail. Therefore, we only assume that the objects of interest in the robot's environment are either almost convex or can be decomposed into almost convex sub-objects in a reasonable way.

Thus our approach is to compute the visible part of the convex hull for each of the given subsequences (see Figure 2.3). For each sample point on the interior side of the convex hull its distance to the line defined by the next preceding and succeeding sample points on the hull is computed. If there is a point whose distance exceeds a threshold value (i.e. the shape is not even "almost convex"), the sequence is divided at a point with a maximum distance, and this split procedure is applied again in a recursive manner. Note that the visible part of the convex hull can be computed efficiently in linear time, since we have an angular ordering of the points around the range finder. The algorithm is an adaptation of Graham's scan (Preparata and Shamos, 1988).

The result of the object extraction step is a set $U = \{u_1, \ldots, u_n\}$ of objects and associated indices $\mathit{begin}(u_i)$ and $\mathit{end}(u_i)$ such that object u_i refers to the sub-sequence of scan points $\{p_{\mathit{begin}(u_i)}, \ldots, p_{\mathit{end}(u_i)}\}$.

2.4 Object Correspondence

As we intend to track objects in a dynamic environment, we have to compare information about objects from successive scans. This is done by a combination of graph algorithms.

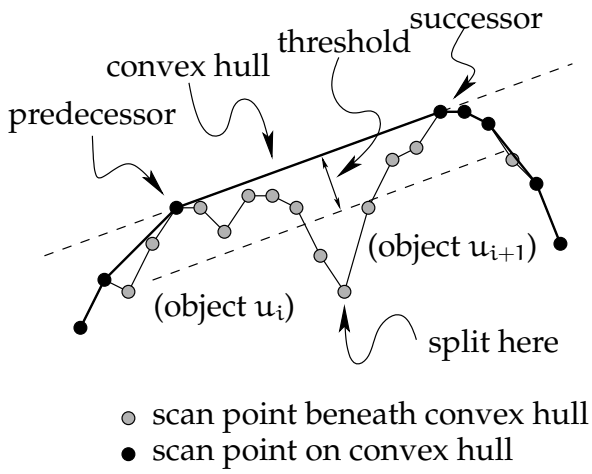


Figure 2.3: Using the convex hull to find split points

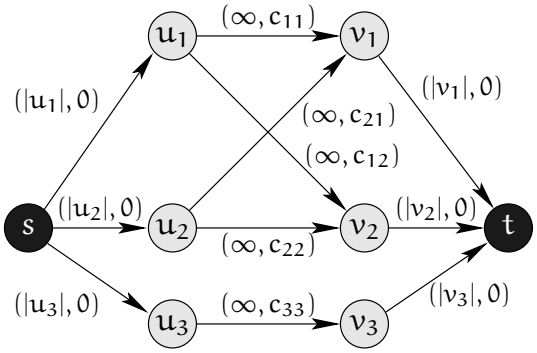


Figure 2.4: Directed graph for object matching. Labels are pairs of arc capacity and cost, where $|u_i|$ denotes the shape length of object u_i, and c_{ij} denotes the distance of objects u_i and v_j with respect to position and shape.

2.4.1 Finding Object Matchings

From the previous and the current scan two sets of objects $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_m\}$ are given. The goal is to find for each object $u_i \in U$ from the previous scan a corresponding object $v_j \in V$ from the current scan. This can be seen as a matching in the bipartite graph $(U \cup V, U \times V)$.

To find a matching representing plausible assignments between objects at successive points of time in the real world, we start by computing a maximum flow with minimal cost in a graph $G = (\{s, t\} \cup U \cup V, A)$ as illustrated by Figure 2.4 and Algorithm 1. We have

$$\begin{aligned} A = \; & \{ \; (s, u) \; | \; u \in U \; \} \; \cup \\ & \{ \; (v, t) \; | \; v \in V \; \} \; \cup \\ & \{ \; (u, v) \; | \; P(u, v) \; \} \end{aligned} \tag{2.1}$$

for some predicate $P : U \times V \to \{true, false\}$ of reasonably low computational complexity. We could use the constant predicate ***true*** here (i.e. accept all arcs), but for practical reasons P is chosen such that $P(u, v)$ is true if the distance between the centers of gravity of u and v does not exceed a threshold value. This is equivalent to the assumption of a speed limit for objects in our environment. The finite scanning frequency does not allow tracking of arbitrary fast objects, anyway. Thus the size of the graph is reduced without imposing a further restriction, resulting in faster computations of minimum cost maximum flows.

Finally an object matching is deduced by retaining only arcs conveying large amounts of this minimum cost maximum flow, i.e. we compute a maximum weight matching. Details on capacity, cost, and weight labels as well as on computational complexities are given below.

Algorithm 1 OBJECT MATCHING

1: **input:** objects $U = \{u_1, \ldots, u_n\}$ from the previous scan
2: **input:** objects $V = \{v_1, \ldots, v_m\}$ from the current scan
3: let $A_{su} = \{(s, u) \mid u \in U\}$
4: let $A_{uv} = \{(u, v) \in U \times V \mid P(u, v)\}$ for some predicate P
5: let $A_{vt} = \{(v, t) \mid v \in V\}$
6: let $A = A_{su} \cup A_{uv} \cup A_{vt}$
7: define graph $G = (\{s, t\} \cup U \cup V, A)$
8: compute a maximum flow $\hat{f}$ in G from source vertex s to sink vertex t w.r.t. capacity labels as defined by Alg. 2.
9: compute vertex excess labels

$$excess(w) = \begin{cases} \sum_{u \in U} \hat{f}(s, u) & \text{if } w = s, \\ \sum_{v \in V} -\hat{f}(v, t) & \text{if } w = t, \\ 0 & else \end{cases}$$

10: compute a minimum cost flow f^* in G w.r.t. capacity and cost labels as defined by Alg. 2 and 3, and excess labels as defined above.
11: let $w = f^*|_{A_{uv}}$ a weight label
12: compute a maximum weight matching M^* in $G[A_{uv}] = (U \cup V, A_{uv})$ w.r.t. weight label w
13: **return** matching $M^* \subseteq U \times V$

Capacity Labels

For each object $u_i \in U$ we compute the length

$$|u_i| = \sum_{i=begin(u_i)}^{end(u_i)-1} d_2(p_i, p_{i+1}) \tag{2.2}$$

of the polygonal chain induced by its scan points. This length is taken as capacity label $u(a)$ for the arc $a = (s, u_i)$, see Algorithm 2). Capacities of arcs (v_j, t), $v_j \in V$, are assigned analogously. Arcs $(u_i, v_j) \in U \times V$ are assigned infinite capacity.

Intuitively, we try to assign as much object shape length as possible from one scan to the next by computing a maximum flow. This is reasonable if we assume small changes of this length for each object between two successive scans.

Algorithm 2 ARC CAPACITIES: $E \to \mathbb{R}$

1: **input:** arc $(p, q) \in A = A_{su} \cup A_{uv} \cup A_{vt}$ (arc set A as in Alg. 1)
2: **if** $(p, q) = (s, u_i)$ for some $u_i \in U$ **then**
3: **return** (length of the polygonal chain representing object u_i)
4: **else if** $(p, q) = (v_j, t)$ for some $v_j \in V$ **then**
5: **return** (length of the polygonal chain representing object v_j)
6: **else**
7: **return** $+\infty$
8: **end if**

Cost Labels

Arcs (s, u_i) incident to the source vertex and arcs (v_j, t) incident to the target vertex are assigned zero costs. Now consider arcs (u_i, v_j) incident only to vertices representing objects. These arcs will be assigned costs that reflect the similarities in shape and position of these objects in the real world, rendering less plausible object matchings more expensive than plausible ones. Our approach to compute these cost labels is to roughly estimate the physical work needed to transform one object into the other. Note that for a resting point of mass, the work that is necessary to move it by a certain distance in a constant period of time is proportional to the square of this distance.

Each object $u_i \in U$ is approximated by a constant number of uniformly distributed sample points $U_i = \{u_1^i, \ldots, u_N^i\} \subseteq \mathbb{R}^2$ which are selected from its shape (i.e. the polygonal chain induced by its scan points). Analogously, each object $v_j \in V$ is approximated by a point set $V_j = \{v_1^j, \ldots, v_N^j\} \subseteq \mathbb{R}^2$. Using these points we construct for each arc $(u_i, v_j) \in A \cap (U \times V)$ of graph G the bipartite graph $H_{ij} = (U_i \cup V_j, U_i \times V_j)$ and an arc label $d_{ij} : U_i \times V_j \to \mathbb{R}$ for H_{ij} with $d_{ij}(u_k^i, v_l^j) = (d_2(u_k^i, v_l^j))^2$, where d_2 denotes the

Euclidean distance in the plane. Since we do not want to make an assumption about the maintenance of the order of points on an object shape between successive scans, we follow a least effort approach and compute a minimum cost perfect matching M^*_{ij} in H_{ij}. The total costs $c_{ij} = \sum_{a \in M^*_{ij}} d_{ij}(a)$ of this matching are taken as a rough estimate for the necessary work and are assigned as the value of the cost label to the according arc (u_i, v_j) of our prior graph G. This approach is also described in Alg. 3.

Algorithm 3 ARC COSTS: $E \to \mathbb{R}$

input: arc $(p, q) \in A = A_{su} \cup A_{uv} \cup A_{vt}$ (arc set A as in Alg. 1)
let $N \in \mathbb{N}$ a constant number
if $p = s$ or $q = t$ **then**
  return 0
else
  $(p, q) = (u_i, v_j)$ for some i, j
  let $C_{u_i} \subset \mathbb{R}^2$ the polygonal chain representing object u_i.
  let $U_i = \{u_1^i, \ldots, u_N^i\} \subset C_{u_i}$ a set of N points uniformly distributed along chain C_{u_i}
  let $C_{v_j} \subset \mathbb{R}^2$ the polygonal chain representing object v_j.
  let $V_j = \{v_1^j, \ldots, v_N^j\} \subset C_{v_j}$ a set of N points uniformly distributed along chain C_{v_j}
  let M^* a minimum cost perfect matching in the bipartite graph $H_{ij} = (U_i \cup V_j, U_i \times V_j)$ w.r.t. squared Euclidean distance d_2^2 in the plane as arc costs.
  return $\sum_{(u_k^i, v_l^j) \in M^*} d_2^2(u_k^i, v_l^j)$
end if

Object Matching.

The computed flow gives an idea of the motion in the environment of the robot but does not yet induce a unique matching. There may be objects that split and rejoin in successive scan images (e.g., consider a human and his arm) and thus induce a flow from one vertex to two successors and reversely. As the length of the shape of an object varies there may be a small flow reflecting these changes as well. Thus it is a good idea to focus attention on arcs with a large amount of flow. Consequently the final step in our approach is to compute a matching of maximum weight in the bipartite subgraph of G induced by the two object sets U and V, using the flow labels computed in the previous step as weight labels (see Algorithm 1 again). Finally, by this matching, we have unique assignments between objects from two successive scans.

Computational Complexity.

We now examine the time complexity of the presented object matching approach. The size of the input is described by the number L of samples per laser scan and the sizes of

the sets $U \neq \emptyset$ and $V \neq \emptyset$ of objects to be matched. Clearly, the graph G as defined in line 7 of Alg. 1 contains $|U| + |V| + 2$ vertices and at most $|U| + |V| + |U| \cdot |V|$ arcs.

Property 2.1
Computing cost labels for all arcs $a \in A$ requires at most $\mathcal{O}(L + (|U| \cdot |V|))$ computational steps.

Proof:

Algorithm 3 uses a set of N uniformly distributed points per object, where N is a constant number. These point sets can be computed by two subsequent sweeps over a scan point sequence S, where object shape lengths are determined during the first sweep and point sets are chosen during the the second sweep. This is accomplished within $\mathcal{O}(L)$ computational steps.

Next, we have to compute a minimum cost perfect matching in a bipartite graph with a constant number of vertices and arcs for each arc of G. Clearly this requires a constant amount of time for each arc of G, so this step is accomplished within $\mathcal{O}(|U| \cdot |V|)$ computational steps.

Finally, arcs with source s or target t are assigned zero costs, which is easily accomplished within $\mathcal{O}(|U| + |V|)$ computational steps. Summing up establishes the claimed time complexity. □

Property 2.2
Computing capacity labels $capacity(e)$ for all arcs $a \in A$ requires at most $\mathcal{O}(L + (|U| \cdot |V|))$ computational steps.

Proof:

As shown in the proof of property 2.1 computation of object shape lengths can be accomplished within $\mathcal{O}(L)$ computational steps. The remaining effort per arc is constant. As there are $\mathcal{O}(|U| \cdot |V|)$ arcs in G, the claimed time complexity is proven. □

Property 2.3
Algorithm 1 can be implemented such that it terminates after at most $\mathcal{O}(L + (|U| + |V|)^3 \log((|U| + |V|) \cdot C))$ computational steps, where $C = \max_{a \in A} costs(a)$ denotes the maximum arc cost in G.

Proof:

The construction of graph G in line 7 of Alg. 1 together with cost and capacity labels requires at most $\mathcal{O}(L + (|U| \cdot |V|))$ computational steps as shown above. The computation of a maximum flow $\hat{f}$ in line 8 requires at most $\mathcal{O}((|U| + |V|)^3)$ computational steps if we use the FIFO preflow push algorithm (Ahuja et al., 1993). The calculation of the

vertex excess labels in line 9 consumes another $\mathcal{O}(|U| + |V|)$ steps. The computation of a minimum cost flow f^* in line 10 requires at most $\mathcal{O}((|U| + |V|)^3 \log((|U| + |V|) \cdot C))$ steps, if we use the cost scaling algorithm (Ahuja et al., 1993). The maximum weight matching M^* in line 10 can be computed within at most $\mathcal{O}((|U| + |V|) \cdot (|U| \cdot |V| + (|U| + |V|) \log(|U| + |V|))) \subseteq \mathcal{O}((|U| + |V|)^3)$ steps (Mehlhorn and Näher, 1999). Summing up establishes the claimed time complexity. □

In other words, for a set of $n = |U| + |V|$ objects and a constant laser scan size L we may need up to $\mathcal{O}(n^3 \log(nC))$ computational steps in order to find an object matching following the approach presented above.

There are several improvements for bipartite network flows (Ahuja et al., 1994). However they require the network to be unbalanced in order to substantially speed up the algorithms, i.e. either $|U| \ll |V|$ or $|U| \gg |V|$, which is not the case in our context.

The complexity of finding an optimal (minimum or maximum weight) matching might be reduced if the cost label is also a metric on the vertex set of the underlying graph. For example if the vertices of the graph are points in the plane and the cost label is the L_1 (*Manhattan*), L_2 (*Euclidean*) or L_∞ metric there are lower time complexity bounds for the problem of finding a minimum weight perfect matching (Vaidya, 1989) than in the general case. However it is not obvious if (and if so, how) this can be applied to the given object correspondence problem.

2.5 Experiments

The described tracking system has been implemented on the robot presented in Section 1.1, and has been tested in various environments, indoors (lab environment, exhibitions, concourse of a railway station) as well as outdoors (pedestrian area). It proved to perform considerably more robust than its predecessor proposed by Prassler et al. (1998a) which was based on a greedy nearest neighbor search among the objects' centers of gravity. The number of objects extracted from a scan typically ranges from ten to twenty, consuming about 50 milliseconds CPU time per cycle on the hardware mentioned above.

Figure 2.5 shows every fourth scan from a longer sequence. The three respective intermediate scans are plotted using lighter shades of grey. The images show one person walking upward, one person walking downward, and two persons walking right. The sequence has been recorded at an exhibition, so the straight segments are billboards. For the same sequence of scan data, Figure 2.6 illustrates extracted objects together with a history of previous object positions which has been obtained by repeated application of the presented approach.

Figure 2.7 shows the path of a guide walking outside on the parking place in front of our lab. The guide has been tracked and accompanied by the robot for 1073 cycles (more

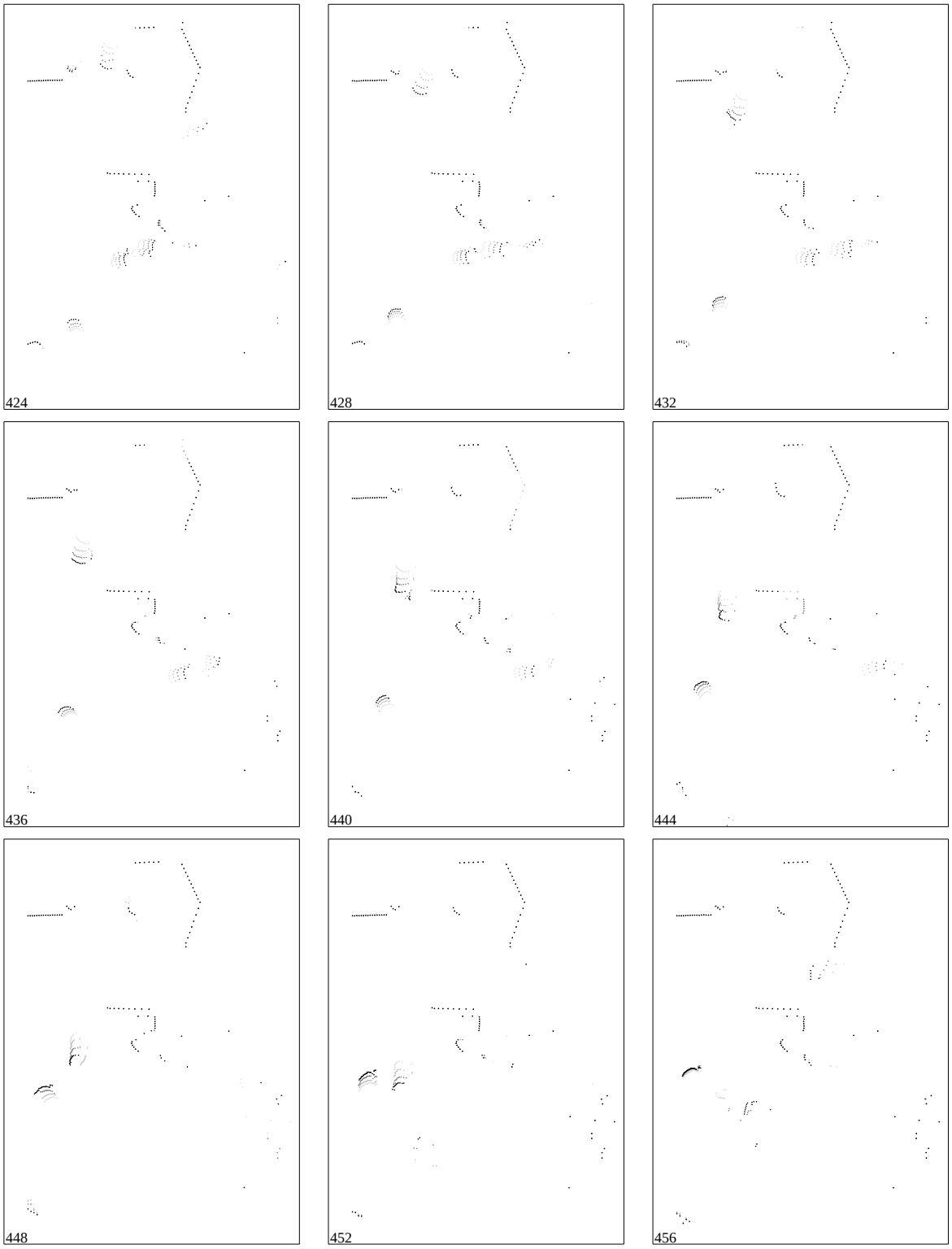


Figure 2.5: Sequence of scans

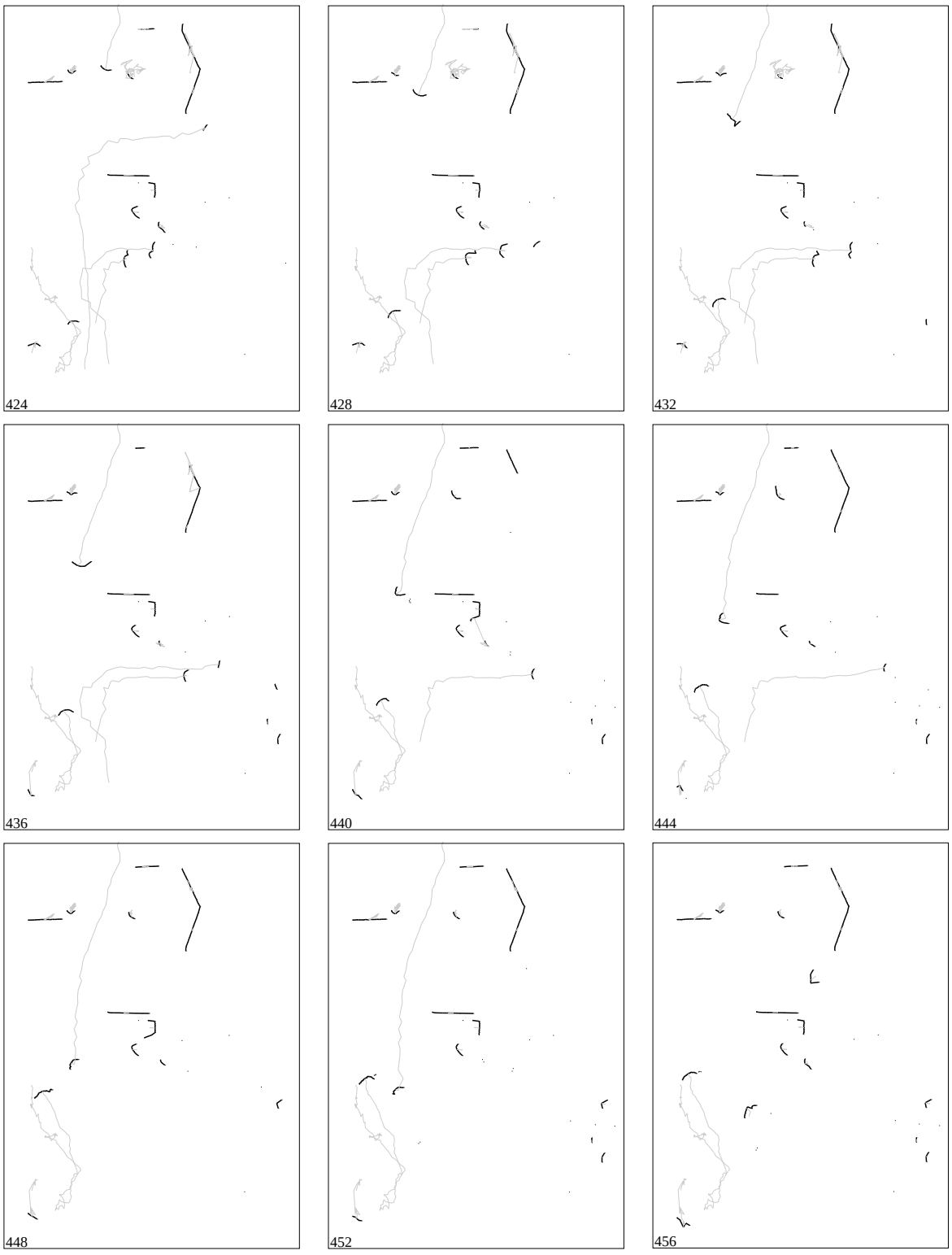


Figure 2.6: Sequence of tracked objects

than five minutes), until he finally became occluded to the range finder. The small loop is caused by the guide walking around the robot.

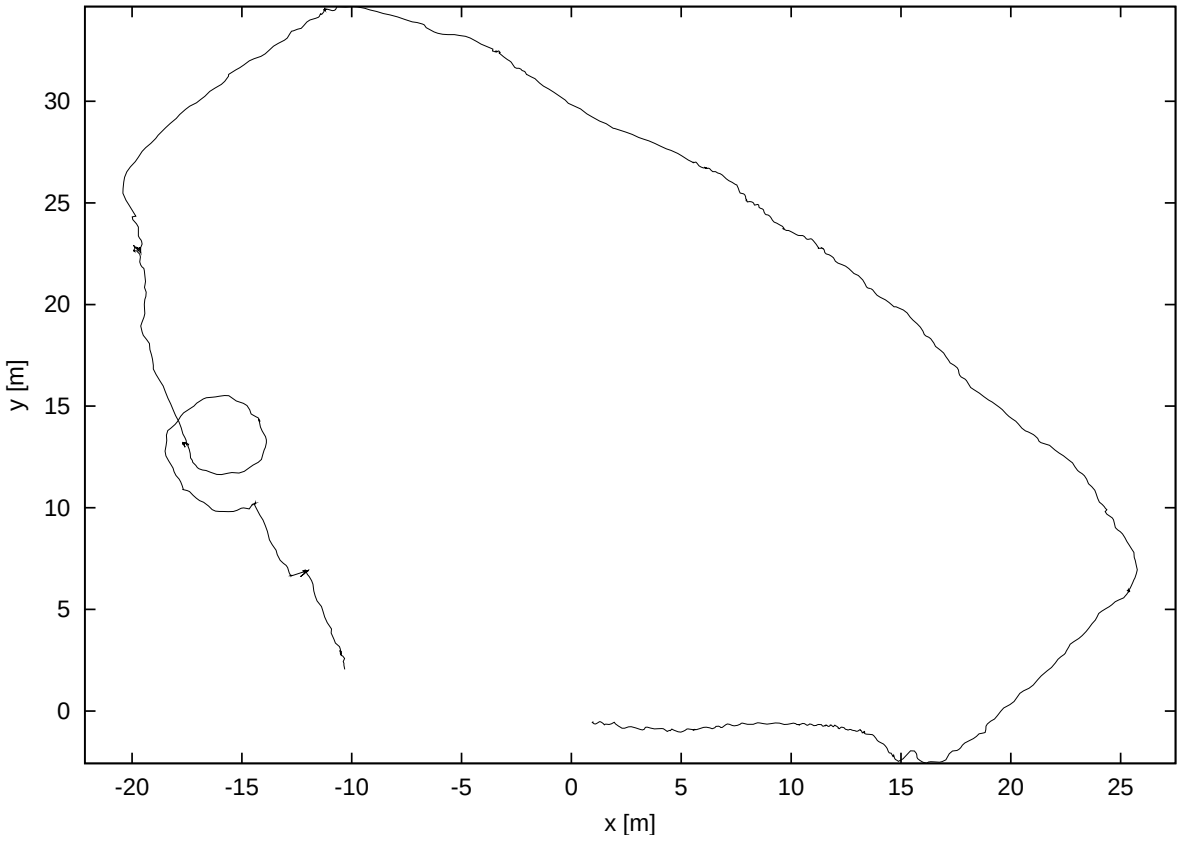


Figure 2.7: Image of the path of a tracked person who is walking outdoors, followed by the robot carrying the tracking system

2.6 Discussion

Unfortunately, the tracking system still loses tracked objects occasionally. One obvious cause is occlusion. It is evident that invisible objects cannot be tracked by any system. But consider an object occluded by another object passing between the range finder and the first object. Such an event canceled the tracking shown in Fig. 2.7, where the guide was hidden for exactly one scan. Hence a future system should be enabled to cope at least with short occlusions.

But tracked objects get lost occasionally even if they are still visible. This might happen for example if new objects appear and old objects disappear simultaneously, as the

visual field of the range finder is limited. To illustrate this, imagine a linear arrangement of three objects. Now delete the leftmost object and insert an object next to the rightmost. A flow computed as described above induces a false assignment, that is a shift to the right. This problem is partially dealt with by restriction to a local search for correspondents as presented in Sect. 2.4.1. It might be further improved if we do not assign any old object to new objects that become visible by a change of perspective due to the robot's motion.

In some cases object extraction fails to properly split composed objects. If these objects are recognized separately in the previous scan, either of them is lost. But this situation may be recognized by looking at the minimum cost flow in the graph, if there is a significant flow into one vertex from two predecessors. This might give a hint to split the newly extracted object.

As object extraction probably cannot be perfect, one might follow the idea to compute the flow based on the scan points before extracting objects by searching for neighboring groups of parallel arcs carrying flow. However this might be computationally infeasible, since the sizes of the graphs involved in the computations of the flows are heavily increased.

Information about the motion of an object drawn from previous scan images could be used to compute an approximation of its current position and thus direct the search for corresponding points. A first implementation of this regarding the motion of centers of gravity showed poor performance in some environments, for example considering walls moving as their visible part grows. Another bad effect of position prediction is its tendency to create errors by a chain effect, as even a single incorrect object assignment results in incorrect prediction of future positions and therefore may result in further incorrect assignments.

2.7 Conclusion

In this part we presented an object tracking system based on laser range finder images and graph algorithms. The basic idea of our tracking approach is to represent the motion of object shapes in successive scan images as flows in bipartite graphs. By optimization (maximum flow, minimum cost, maximum weighted matching) we get plausible assignments of objects from successive scans even in the case of clutter by smaller objects. The approach has been implemented on a mobile robot and proved to be useful for various task ranging from obstacle avoidance to situation assessment.

Chapter 3

Navigation in Dynamic Environments

3.1 Introduction

After Chapter 2 presented an approach to perception in dynamic environments, we will now consider the complementary problem of generating motion for a mobile robot among moving obstacles. In the course of this chapter we will cover the basic principles of mobile robot motion, properties of existing mobile robot drives, and an approach to implement a collision avoiding behavior.

3.1.1 Related Work

Motion planning is a fundamental problem in robotics, and numerous approaches have been proposed. Many of them are dealing with static environments, where obstacles of fixed shape are located at fixed positions. Other approaches accept moving obstacles in the presence of the robot, which renders them more relevant to the topic of this chapter. For example, the comprehensive work of Fujimura (1991) considers motion planning for a mobile robot among obstacles with known motion. Similarly, Fraichard (1998) and Hsu et al. (2000) address motion planning for vehicles with kinematic and dynamic constraints on known trajectories. The latter approach is claimed to be efficient enough for occasional replanning, and therefore might be used even when the obstacle motion is not known in advance. Other approaches, for example by Chakravarthy and Ghose (1998) or Fiorini and Shiller (1998), allow to avoid the computational cost of complete motion planning, and permit fast reactive collision avoidance behaviors in dynamic environments.

3.1.2 Overview

In the following, we will specify some basic concepts and properties of mobile robots, before two popular implementations of a mobile robot drive are considered, i.e. the differential drive and the synchro drive. We will get an impression of their kinematic and dynamic constraints, and how these influence the complexity of motion planning. In order to avoid that complexity when navigating a differentially driven mobile robot, an approach is proposed considering a virtual robot center for which these constraints are substantially simplified. With this heuristic method, we are able to apply the velocity obstacle approach by Fiorini and Shiller (1998) for collision avoidance, which is described subsequently.

3.2 Mobile Robot Motion

3.2.1 Configuration, Motion, and Collisions

In contrast to for example aircraft motion, mobile robot motion takes place in the plane. Accordingly, we will model the rigid shape of involved objects as sets of points.

Definition 3.1 (Rigid Body, Object, Obstacle)
A **rigid body** *is a compact set* $B \subset \mathbb{R}^2$.

The rigid body

$$D(r) = \left\{ \mathbf{p} \in \mathbb{R}^2 \mid |\mathbf{p} - \mathbf{0}| \leq r \right\} \tag{3.1}$$

is called **circular with radius** r*, or* **disc with radius** r.

A rigid body may be called **object** *or* **obstacle***, too.*

The configuration of a rigid body describes its current position and orientation. That is, for a specific configuration, each degree of freedom of the object has to be fixed to a corresponding value. Rigid bodies in the plane have two translational and one rotational degrees of freedom, whereby the rotation axis is perpendicular to the plane of translations. Hence, their configuration space is $SE(2)$, the Special Euclidean group in two dimensions. An element of $SE(2)$ corresponds to a frame in $\mathcal{W}$, or to a displacement of a rigid body in $\mathcal{W}$, and the neutral element of $SE(2)$ corresponds to the frame $\langle \mathcal{W} \rangle$.

Definition 3.2 (Configuration)
The **configuration of a rigid body** B *in the Euclidean plane is an ordered triple*

$$c_B = (x_B, y_B, \theta_B) \in \mathbb{R}^2 \times \mathbb{R} \tag{3.2}$$

where (x_B, y_B) *is the* **position** *of* B*, and* θ_B *is the* **orientation** *of* B *(both relative to* $\langle W \rangle$*, see Figure 3.1).*

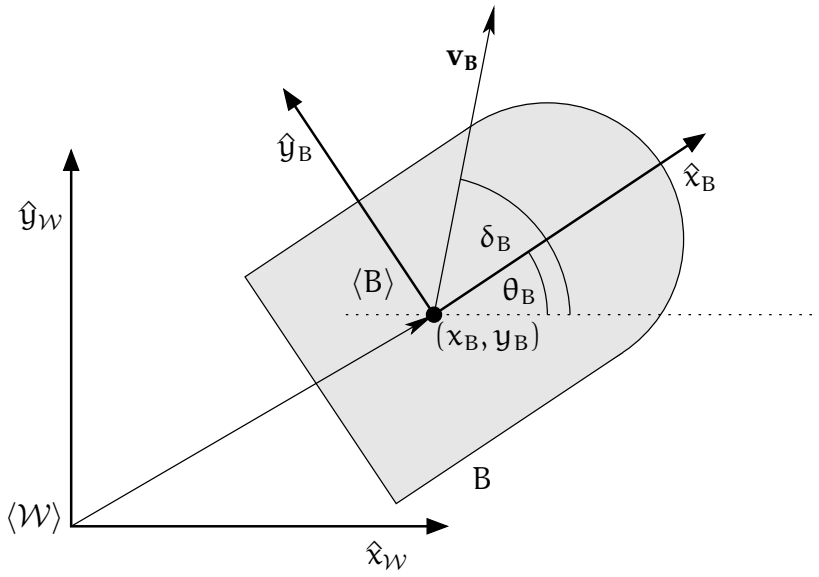


Figure 3.1: Configuration (x_B, y_B, θ_B), velocity $\mathbf{v_B}$, and direction of motion δ_B of a rigid body B

We will also write $\mathbf{c_B} = (x_B, y_B)^T$ *for the position component of configuration* c_B*. Conversely, we will accept a point* $\mathbf{p} = (x_p, y_p)^T \in \mathbb{R}^2$ *as a configuration* $(x_p, y_p, 0)$ *with orientation zero.*

For moving rigid bodies, their configuration changes over time. This is modeled by a function which maps time to configurations, called the trajectory of a body.

Definition 3.3 (Trajectory)
The **trajectory of a rigid body** B *is a continuous function*

$$\tau_B : \mathbb{R} \to \mathbb{R}^2 \times \mathbb{R} \tag{3.3}$$

which maps time t *to configurations* $\tau_B(t) = (x_B(t), y_B(t), \theta_B(t))$*.*

The above definition of a trajectory allows sharp turns on the spot, without decelerating. For practical cases, we will introduce a direction of motion which is not allowed to change instantaneously. Furthermore, we will not require the direction of motion to be identical or in a fixed relation to the orientation of the vehicle. Therefore, these two angular variables are treated separately.

Definition 3.4 (Direction of Motion, Velocity)
Let τ_B *be the trajectory of a rigid body* B *with differentiable components* x_B*,* y_B*, and* θ_B*. If there is a differentiable continuous function* $\delta_B : \mathbb{R} \to \mathbb{R}$ *with*

$$\mathbf{v_B}(t) := \begin{pmatrix} \dot{x}_B(t) \\ \dot{y}_B(t) \end{pmatrix} = |\mathbf{v_B}(t)| \begin{pmatrix} \cos \delta_B(t) \\ \sin \delta_B(t) \end{pmatrix}, \tag{3.4}$$

we call δ_B *the* **direction of motion** *of* B*, and the ordered triple* $(|\mathbf{v_B}|, \dot{\theta}_B, \dot{\delta}_B)$ *is called the* **velocity of** B*. We will also use the letter* ω *to denote the angular rate* $\dot{\delta}$ *of the direction of motion.*

Definition 3.5 (Initial and Final Configuration)
Let τ *be a trajectory. The configuration* $c_1 = \tau(t_1)$ *is called* **initial configuration (of** τ**)**, *if* $\tau(t) = c_1$ *for* $t \leq t_1$*. The configuration* $c_2 = \tau(t_2)$ *is called* **final configuration (of** τ**)**, *if* $\tau(t) = c_2$ *for* $t \geq t_2$*.*

Definition 3.6 (Placed Body)
An ordered pair (B, c_B) *consisting of a rigid body* B *and a configuration* c_B *is called a* **placed body***.*

Definition 3.7 (Moving Body)
An ordered pair (B, τ_B) *consisting of a rigid body* B *and a trajectory* τ_B *is called a* **moving body***.*

Definition 3.8 (Linear Motion)
A trajectory $\tau_B : t \mapsto (x_B(t), y_B(t), \theta_B(t))$ *is called a* **linear motion***, if its first two components* x_B *and* y_B *are linear, and the third component* θ_B *is a constant.*

When we are only interested in the position of objects, we will talk of the path of an object and ignore the orientational components of the configuration.

Definition 3.9 (Path)
Let $\tau_B : t \mapsto (x_B(t), y_B(t), \theta_B(t))$ *the trajectory of a rigid body* B*. Then, the plane curve* $\pi_B : t \mapsto (x_B(t), y_B(t))$ *is called the* **path of rigid body** B*.*

In general, rigid bodies occupy some amount of workspace, which depends on the shape of the body and its (current) configuration.

Definition 3.10 (Occupied Space of a Rigid Body)
Let (B, c) *be a placed body with* $c = (x, y, \theta)$*. Then, the set*

$$\mathcal{S}(B, c) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \mathbf{p} + \begin{pmatrix} x \\ y \end{pmatrix} \mid \mathbf{p} \in B \right\} \subseteq \mathbb{R}^2 \tag{3.5}$$

is called **the occupied space of** B **in configuration** c*.*

Let (B, τ_B) *be a moving body. Then, the set* $\mathcal{S}(B, \tau_B(t))$ *is called* **the occupied space of** (B, τ_B) **at time** t*.*

Definition 3.11 (Collision)
Let (A, c_A) *and* (B, c_B) *be placed bodies. We say* (A, c_A) **collides with** (B, c_B)*, if*

$$\mathcal{S}(A, c_A) \cap \mathcal{S}(B, c_B) \neq \emptyset. \tag{3.6}$$

Two moving bodies (A, τ_A) *and* (B, τ_B) *are called* **colliding at time** t*, if* $(A, \tau_A(t))$ *collides with* $(B, \tau_B(t))$*.*

Two moving bodies (A, τ_A) *and* (B, τ_B) *are called* **colliding***, if there is a* $t \in \mathbb{R}$ *such that they are colliding at time* t*, and they are called* **collision-free***, if they are not colliding at any time.*

Definition 3.12 (Reflected Placed Body)
Let (B, c_B) *be a placed rigid body with* $c_B = (x_B, y_B, \theta_B)$*. Then, the placed rigid body* $(B, \bar{c}_B)$ *with*

$$\bar{c}_B = (-x_B, -y_B, \theta_B + \pi) \tag{3.7}$$

is called the **reflection** *of* (B, c_B)*.*

Definition 3.13 (Minkowski Sum)
Let $A, B \subseteq \mathbb{R}^2$ *be point sets. Then we call the set*

$$A + B = \{a + b \mid a \in A, b \in B\} \tag{3.8}$$

the **Minkowski sum** *of* A *and* B*.*

For $p \in \mathbb{R}^2$*, we write*

$$A + p = A + \{p\}, \text{ and } p + B = \{p\} + B \tag{3.9}$$

for the Minkowski sum of a set and a single point.

If two point sets $A, B \subseteq \mathbb{R}^2$ are represented by their characteristic functions χ_A and χ_B, then the convolution $\chi_A * \chi_B$ maps points from the Minkowski sum $A + B$ to non-zero values, and points from the complement of the Minkowski sum to zero. Due to this property, the Minkowski sum is sometimes called the ***convolution of point sets***.

Collisions of placed rigid bodies can be detected by considering the Minkowski sum of one body with a reflected version of the other body.

Property 3.14 (Characterization of Collisions)
Let (A, c_A) *and* (B, c_B) *be two placed bodies with* $c_A = (x_A, y_A, \theta_A)$ *and* $c_B = (x_B, y_B, \theta_B)$*. Then, the following propositions are equivalent:*

(i) (A, c_A) *and* (B, c_B) *are colliding*
(ii) $\mathbf{0} \in \mathcal{S}(A, \bar{c}_A) + \mathcal{S}(B, c_B)$
(iii) $(x_A, y_A) \in \mathcal{S}(A, (0, 0, \theta_A + \pi)) + \mathcal{S}(B, c_B)$.

Proof:

The two placed rigid bodies (A, c_A) and (B, c_B) with $c_A = (x_A, y_A, \theta_A)$ and $c_B = (x_B, y_B, \theta_B)$ are colliding if and only if there are $\mathbf{a} \in A$ and $\mathbf{b} \in B$ with

$$\begin{pmatrix} \cos\theta_A & -\sin\theta_A \\ \sin\theta_A & \cos\theta_A \end{pmatrix} \cdot \mathbf{a} + \begin{pmatrix} x_A \\ y_A \end{pmatrix} = \begin{pmatrix} \cos\theta_B & -\sin\theta_B \\ \sin\theta_B & \cos\theta_B \end{pmatrix} \cdot \mathbf{b} + \begin{pmatrix} x_B \\ y_B \end{pmatrix}, \tag{3.10}$$

which holds if and only if there are $\mathbf{a} \in A$ and $\mathbf{b} \in B$ with

$$\mathbf{0} = \begin{pmatrix} \cos(\theta_A + \pi) & -\sin(\theta_A + \pi) \\ \sin(\theta_A + \pi) & \cos(\theta_A + \pi) \end{pmatrix} \cdot \mathbf{a} + \begin{pmatrix} -x_A \\ -y_A \end{pmatrix} + \begin{pmatrix} \cos\theta_B & -\sin\theta_B \\ \sin\theta_B & \cos\theta_B \end{pmatrix} \cdot \mathbf{b} + \begin{pmatrix} x_B \\ y_B \end{pmatrix} \quad (3.11)$$

which is equivalent to

$$\mathbf{0} \in \mathcal{S}(A, \bar{c}_A) + \mathcal{S}(B, c_B). \quad (3.12)$$

Furthermore, Equation 3.11 is equivalent to

$$\begin{pmatrix} x_A \\ y_A \end{pmatrix} = \begin{pmatrix} \cos(\theta_A + \pi) & -\sin(\theta_A + \pi) \\ \sin(\theta_A + \pi) & \cos(\theta_A + \pi) \end{pmatrix} \cdot \mathbf{a} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\theta_B & -\sin\theta_B \\ \sin\theta_B & \cos\theta_B \end{pmatrix} \cdot \mathbf{b} + \begin{pmatrix} x_B \\ y_B \end{pmatrix} \quad (3.13)$$

which is equivalent to

$$(x_A, y_A) \in \mathcal{S}(A, (0, 0, \theta_A + \pi)) + \mathcal{S}(B, c_B). \quad (3.14)$$

□

Given a set of obstacles, one can ask for a collision-free trajectory of another body from an initial configuration to a final configuration. The image of such a trajectory must lie within a set of collision-free configurations.

Definition 3.15 (Free Space)
Let $\mathcal{B} = \{(B_i, c_i) \mid i = 1, 2, \ldots, n\}$ be a finite set of placed obstacles, and A a rigid body. Then, the set

$$\mathcal{F}(A, \mathcal{B}) = \left\{ c_A \in \mathbb{R}^2 \times \mathbb{R} \mid \mathcal{S}(A, c_A) \cap \mathcal{S}(\mathcal{B}) = \emptyset \right\} \quad (3.15)$$

with

$$\mathcal{S}(\mathcal{B}) = \cup_{i=1}^{n} \mathcal{S}(B_i, c_i) \quad (3.16)$$

is called the **free space of A with respect to the obstacle set $\mathcal{B}$.**

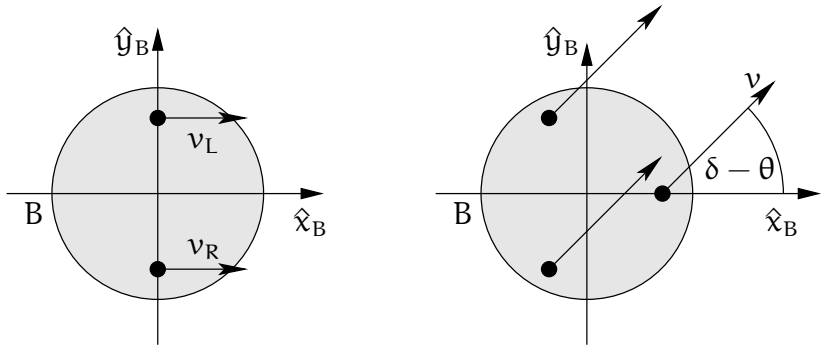
If the robot A is not allowed to rotate, Property 3.14 can be used to compute the free space $\mathcal{F}(A, \mathcal{B})$ of A with respect to placed obstacles $\mathcal{B}$ as

$$\mathcal{F}(A, \mathcal{B}) = \left(\mathbb{R}^2 - \cup_{(B,c) \in \mathcal{B}} \left(\mathcal{S}(A, (0, 0, \theta_A + \pi)) + \mathcal{S}(B, c) \right) \right) \times \{\theta_A\}. \quad (3.17)$$

If the obstacles are allowed to move, too, the more complex problem of dynamic motion planning emerges.

Problem 3.1 (Dynamic Motion Planning)
Let $\mathcal{B} = \{(B_i, \tau_i) \mid i = 1, 2, \ldots, n\}$ be a finite set of moving obstacles. Let $A \subseteq \mathbb{R}^2$ be a rigid body, and c_1 and c_2 configurations.

The **dynamic motion planning decision problem** *is to decide whether there is a trajectory τ_A with initial configuration c_1 and final configuration c_2 such that (A, τ_A) is collision-free with respect to the moving obstacles $\mathcal{B}$.*



(a) Differential drive (b) Synchro drive

Figure 3.2: Drive kinematics

Canny (1987) proved the following theorem on the complexity of dynamic motion planning.

Theorem 3.16 (Complexity of Dynamic Motion Planning)
Let $\mathcal{B} = \{(B_i, \tau_i) \mid i = 1, 2, \ldots, n\}$ be a finite set of moving obstacles, where each obstacle B_i is a convex polygon, and each trajectory τ_i is a linear motion. Let A be a point shaped rigid body, and c_1 and c_2 configurations.

Then, deciding if there is a trajectory $\tau_A = (x_A, y_A, \theta_A)$ with initial configuration c_1, final configuration c_2, and bounded velocity $\dot{x}_A^2 + \dot{y}_A^2 \leq v_{max}^2$ such that (A, τ_A) is collision-free with respect to the moving bodies $\mathcal{B}$, is NP-hard.

3.2.2 Kinematic Model

In order to implement a drive system for a vehicle, one has to solve the problems of propulsion and steering. Due to their relative simplicity, the differential drive and the synchro drive (see Figure 3.2) are popular approaches, and we will focus on their properties in the following. There are other types of drives, e.g. car-like robots with separate driven wheels and steering wheels, or even holonomic drives with omnidirectional wheels. However, these will not be considered in this thesis.

Definition 3.17 (Differential Drive)
A **differential drive** *for a mobile robot B consists of two independently driven wheels which are oriented in parallel to the positive x-axis and located at positions $(0, w)$ (the* **left wheel***) and $(0, -w)$ (the* **right wheel***) relative to coordinate frame $\langle B \rangle$, see Figure 3.2(a).*

Property 3.18 (Direction of Motion and Velocity of Differential Drive)
The direction of motion δ_B of a vehicle B with differential drive is equal to its orientation θ_B. Steering occurs by rotating the entire vehicle. Given the velocities v_L and v_R of

the left and the right wheel, the velocity $(v_B, \dot{\theta}_B, \dot{\delta}_B) = (v_B, \dot{\theta}_B, \dot{\theta}_B)$ *of* B *is given by*

$$v_B = \frac{v_R + v_L}{2}, \text{ and} \tag{3.18}$$

$$\dot{\theta}_B = \frac{v_R - v_L}{2w}. \tag{3.19}$$

In case of a differential drive, we will write $(v, \dot{\theta})$ *in short for its velocity, omitting a second appearance of* $\dot{\theta}$*.*

Property 3.19 (Wheel Velocities from Vehicle Velocity)
Given the velocity (v, ω) *of a differential drive, the corresponding wheel velocities can be obtained as*

$$v_R = v + \omega \cdot w, \text{ and} \tag{3.20}$$

$$v_L = v - \omega \cdot w. \tag{3.21}$$

Definition 3.20 (Synchro Drive)
Another common drive for a vehicle B *is the synchro drive, which consists of* $n \geq 3$ *wheels whose orientations and velocities are constantly equal, see Figure 3.2(b). To change the direction of motion, all wheels are turned synchronously with respect to an axis orthogonal to the plane of motion.*

The orientation of the vehicle B *is constant, without loss of generality* $\theta_B = 0$*, and the direction of steering* δ_B *is the orientation of the wheels. Therefore, the velocity of a synchro drive is* $(v_B, 0, \dot{\delta}_B)$*, where* v_B *is the wheel velocity.*

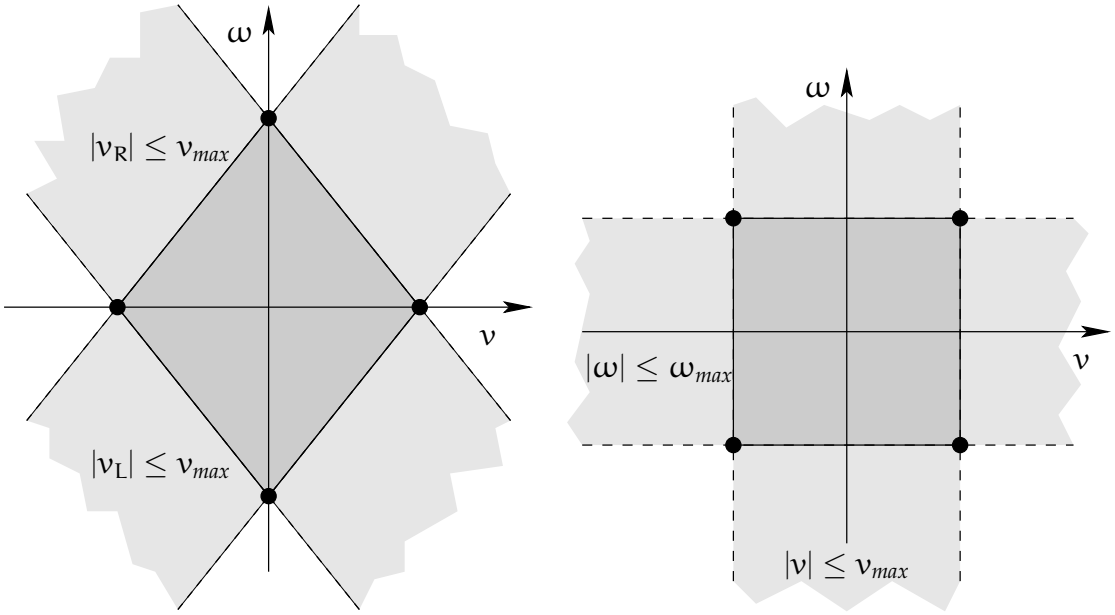
When talking of synchro drives, we will write (v, ω) *for the velocity, omitting the constant body orientation.*

As shown above, we may neglect the given kinematic structure to some degree, speaking of velocity as a pair (v, ω), or of (v, ω)-drives. However, when considering bounds on the controlled velocities, differences will appear.

Assuming that forward and backward velocities of the right and left wheel of a differential drive are equally bounded by a maximum velocity v_{max}, the set of feasible wheel velocities (v_R, v_L) can be transformed into the set of feasible vehicle velocities (v, ω), as illustrated by the rhombic shape in Figure 3.3(a).

In contrast to that, the translational and rotational components of the velocity (v, ω) of a synchro drive propelled vehicle are controlled independently, yielding a rectangular feasibility area as shown in Figure 3.3(b).

For the remainder of this section, we will focus on properties of the differential drive.



(a) Differential drive (b) Synchro drive

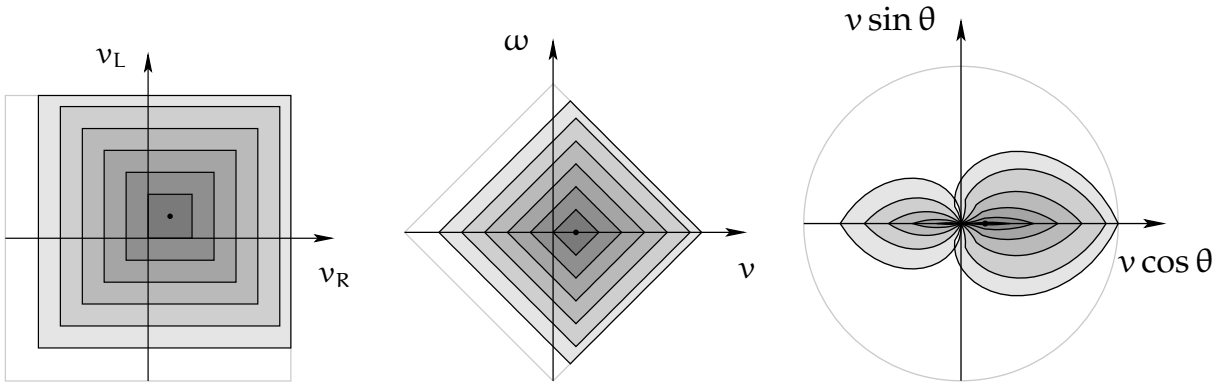
Figure 3.3: Feasible vehicle velocities

Reachable Velocities for a Differential Drive

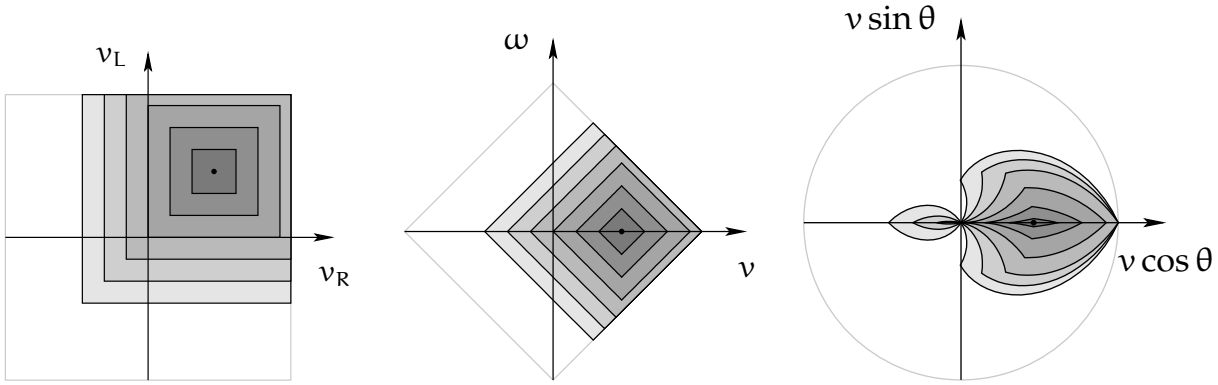
As we have already seen above, there may be forbidden motions for a given type of mobile robot drive kinematics. For example, a vehicle with differential drive cannot move sideways, since there is a kinematic constraint imposed by the interaction of the wheels with the floor. Furthermore, in the real world, forces are bounded and bodies have non-zero mass. Therefore, accelerations and consequently velocities which are reachable starting from a given motion state during a bounded period of time are bounded, too. Both these kinematic and dynamic constraints restrict the sets of velocities and configurations which are reachable within a bounded amount of time.

Figures 3.4 through 3.6 visualize sets of reachable velocities for a differential drive starting from different initial velocities. A common maximum velocity $\hat{v}_W$ and a common maximum acceleration $\hat{a}_W$ are assumed for each for the two independently driven wheels. Starting at $t = 0$ with an initial velocity (v_0, ω_0), constant feasible wheel accelerations a_R and a_L with $-\hat{a}_W \leq a_R, a_L \leq \hat{a}_W$ are applied, without exceeding maximal wheel velocities. The left and center columns in Figures 3.4 through 3.6 visualize the attained wheel velocities and vehicle velocities, respectively.

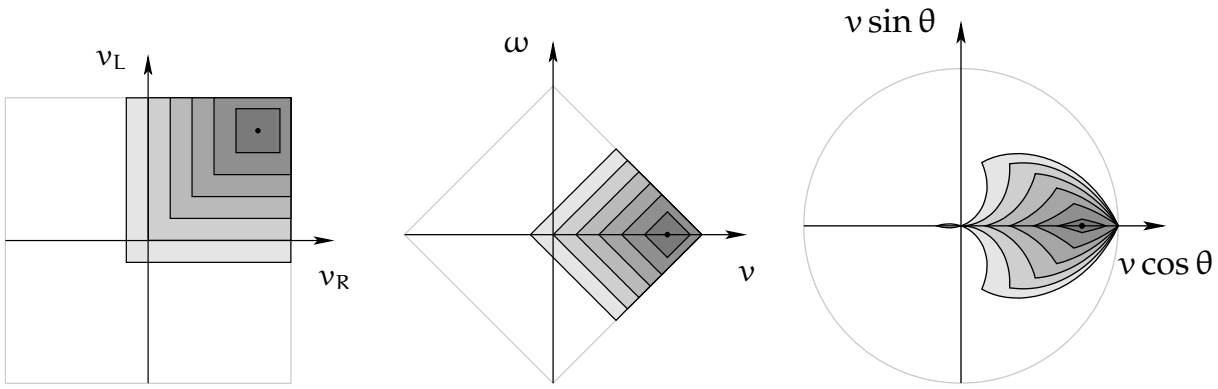
Since in general the rotational velocity component ω is non-zero, the actual direction of motion θ is changing continuously. Integrating this angular rate, the resulting change of direction of motion is visualized in the right column of Figures 3.4 through 3.6. Here a constant wheel acceleration is assumed over the full interval in order to attain the



(a) Initial velocity is $(v_0 = 0.1, \omega_0 = 0.0)$

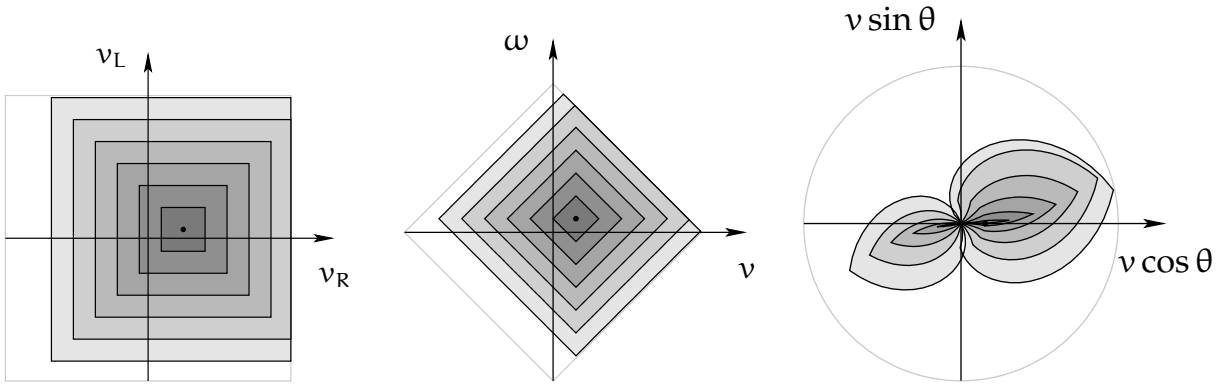


(b) Initial velocity is $(v_0 = 0.3, \omega_0 = 0.0)$

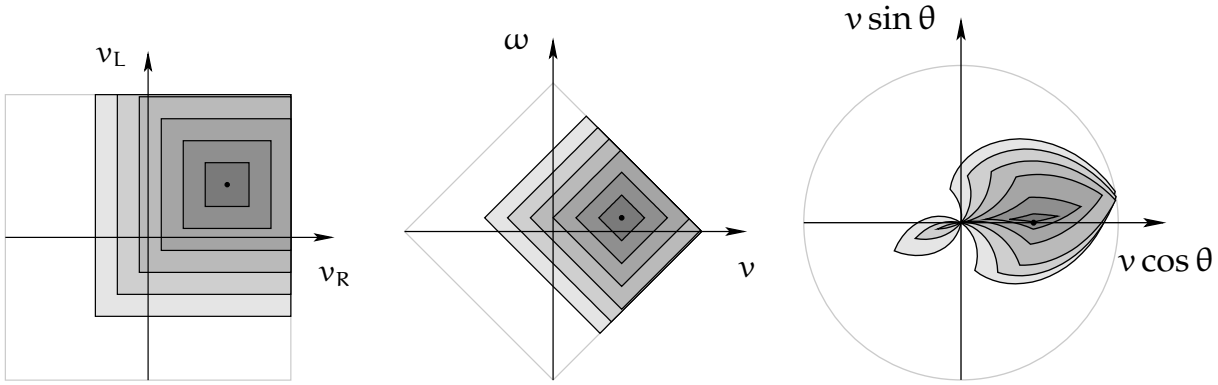


(c) Initial velocity is $(v_0 = 0.5, \omega_0 = 0.0)$

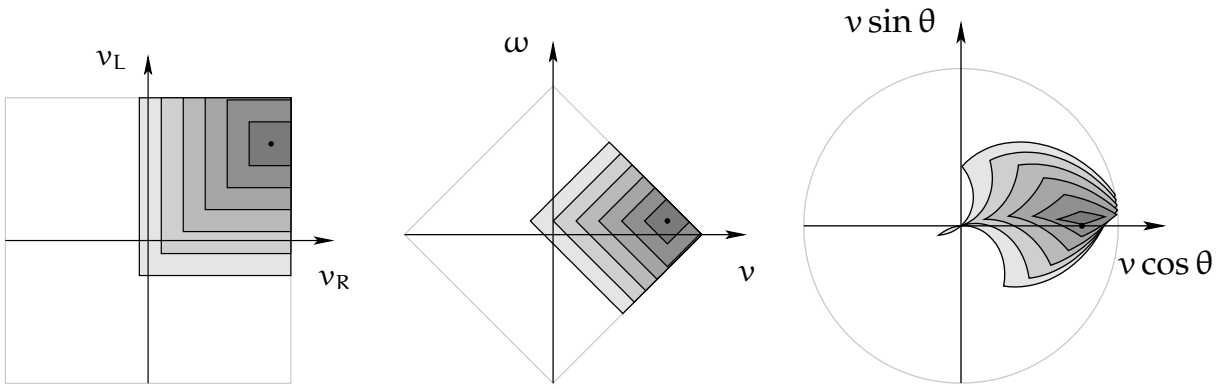
Figure 3.4: Reachable velocities for a differential drive (1)



(a) Initial velocity is $(v_0 = 0.1, \omega_0 = 0.2)$

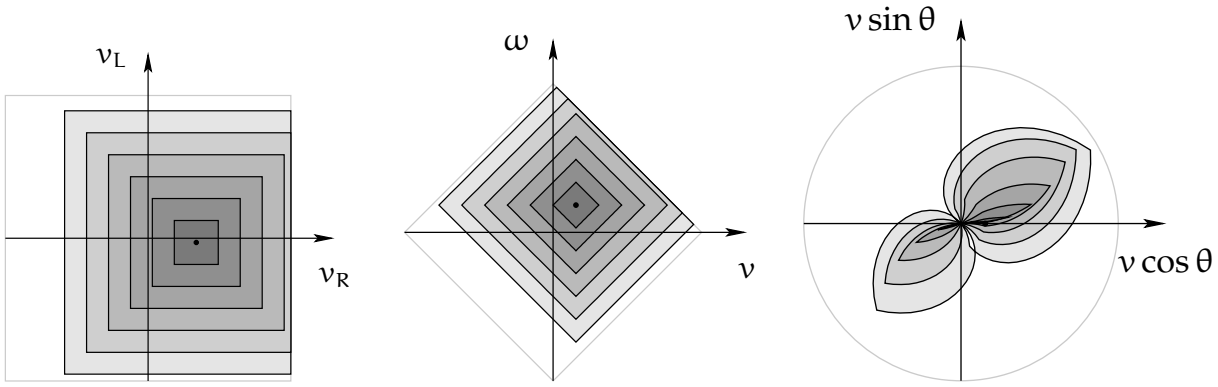


(b) Initial velocity is $(v_0 = 0.3, \omega_0 = 0.2)$

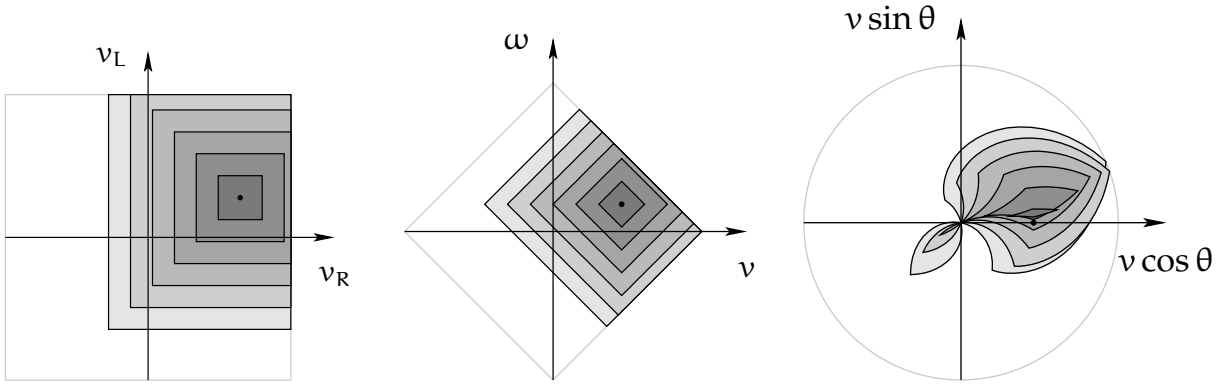


(c) Initial velocity is $(v_0 = 0.5, \omega_0 = 0.2)$

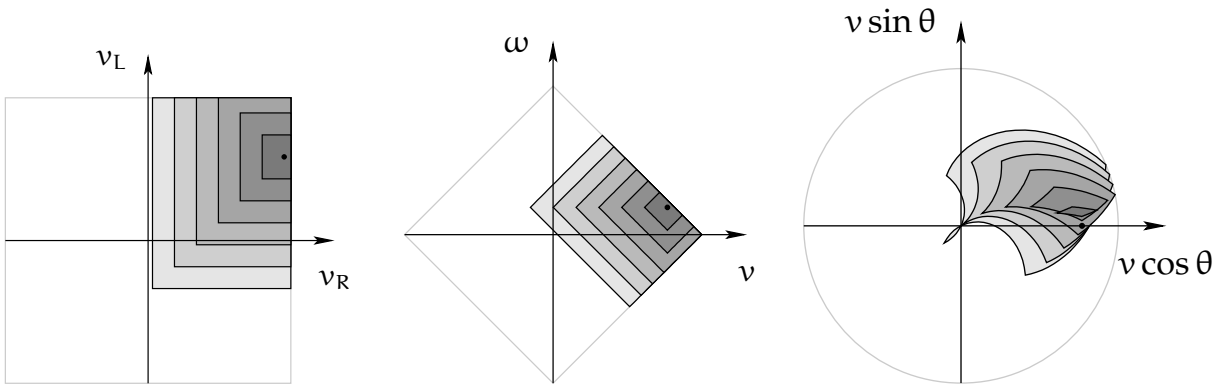
Figure 3.5: Reachable velocities for a differential drive (2)



(a) Initial velocity is $(v_0 = 0.1, \omega_0 = 0.4)$

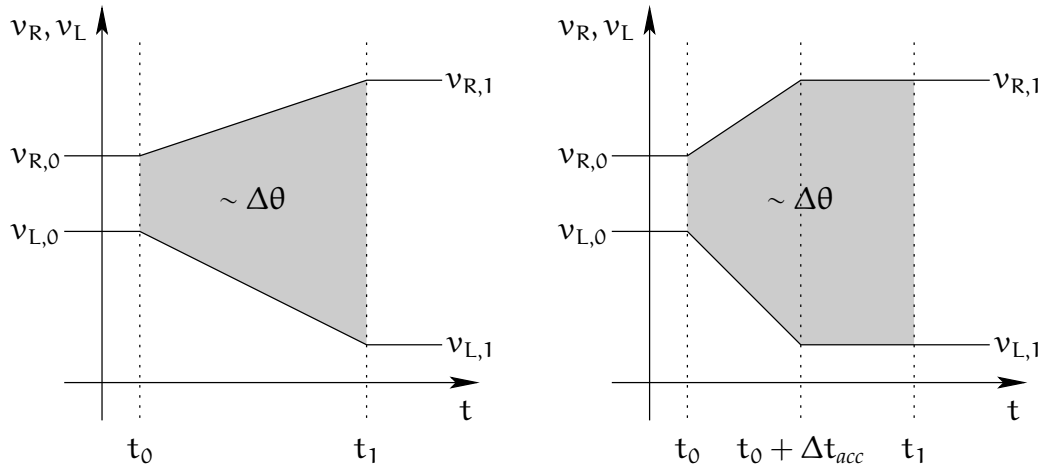


(b) Initial velocity is $(v_0 = 0.3, \omega_0 = 0.4)$



(c) Initial velocity is $(v_0 = 0.5, \omega_0 = 0.4)$

Figure 3.6: Reachable velocities for a differential drive (3)



(a) constant acceleration profile (b) alternative acceleration profile

Figure 3.7: Different acceleration profiles for reaching the same motion state

new motion state, see Figure 3.7(a). However, if the maximal possible acceleration $\hat{a}_W$ is not required, different acceleration profiles may be applied, resulting in the same final wheel velocities. See for example Figure 3.7(b), where constant accelerations $a_R \in [-\hat{a}_W, \hat{a}_W]$ and $a_L = -\hat{a}_W$ are applied for a minimal interval $[t_0, t_0 + \Delta t_{acc}]$, resulting in the same wheel velocities at time t_1 as in Figure 3.7(a). As indicated by these figures, the change of orientation is proportional to the area between the two velocity profiles, since with Equation 3.19

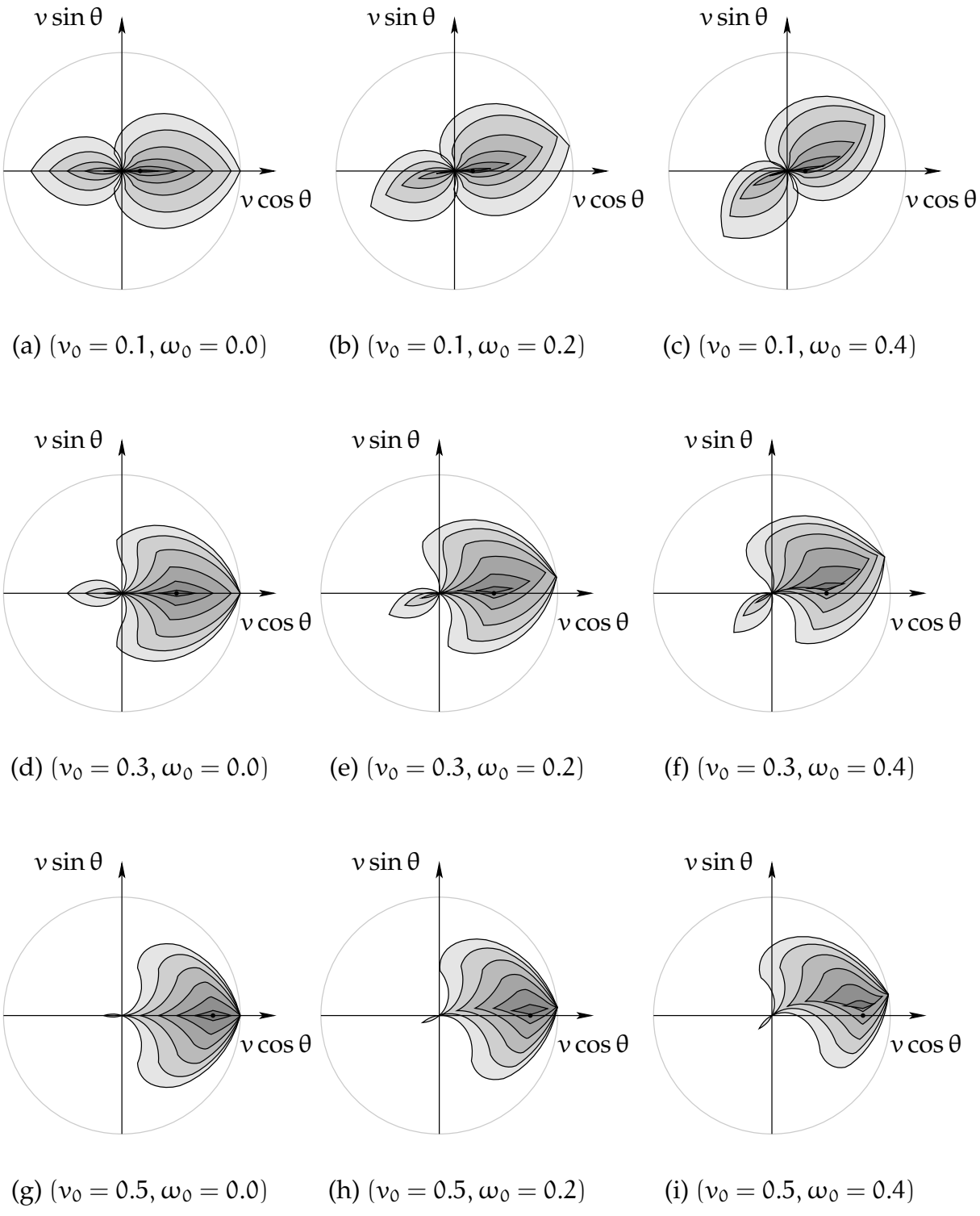
$$\Delta\theta = \theta(t_1) - \theta(t_0) \tag{3.22}$$

$$= \int_{t_0}^{t_1} \omega(t)dt \tag{3.23}$$

$$= \frac{1}{2w} \int_{t_0}^{t_1} v_R(t) - v_L(t)dt \tag{3.24}$$

holds. Therefore, using different types of acceleration profiles will result in different directions of motion. For acceleration profiles as indicated by Figure 3.7(b), the resulting change of direction is reflected by Figure 3.8 for the same initial motion states as in Figures 3.4 through 3.6.

So far, we considered the effect of the wheel velocities and accelerations on the vehicle velocity and orientation only. In the next section, we will examine the effect of constant wheel accelerations on the path of the vehicle.



(a) $(v_0 = 0.1, \omega_0 = 0.0)$ (b) $(v_0 = 0.1, \omega_0 = 0.2)$ (c) $(v_0 = 0.1, \omega_0 = 0.4)$

(d) $(v_0 = 0.3, \omega_0 = 0.0)$ (e) $(v_0 = 0.3, \omega_0 = 0.2)$ (f) $(v_0 = 0.3, \omega_0 = 0.4)$

(g) $(v_0 = 0.5, \omega_0 = 0.0)$ (h) $(v_0 = 0.5, \omega_0 = 0.2)$ (i) $(v_0 = 0.5, \omega_0 = 0.4)$

Figure 3.8: Reachable velocities for a differential drive (4)

Path Types for a Differential Drive

When accelerations $|a_R| = |a_L|$ with the same absolute value (e.g. the maximum value $\hat{a}_W$) are applied to the wheels of a differential drive, prominent types of paths emerge. They differ in the way the curvature changes depending on the traveled arc length.

Definition 3.21 (Curvature of a Plane Curve)
At a given point $c(t)$ *of a plane curve* c*, the* **radius of curvature** $\rho(t)$ *is the radius of the osculating circle at* $c(t)$*, and the* **curvature** $\kappa(t) = 1/\rho(t)$ *is the inverse of the radius of curvature.*

Property 3.22 (Curvature of the Path of a (v, ω)***-Drive)***
Let B *be a vehicle with* (v, ω)*-drive. Then the curvature of the path of* B *is*

$$\kappa = \frac{\omega}{v} \tag{3.25}$$

where the velocity v *is not zero.*

Let (v_R, v_L) *be the wheel velocities of a differentially driven body* B*. Then, the curvature of the path of* B *is*

$$\kappa = \frac{1}{w} \cdot \frac{v_R - v_L}{v_R + v_L} \tag{3.26}$$

where $v_R + v_L \neq 0$*.*

We start examining the simplest case where the accelerations of the right and left wheels are zero, $a_R = a_L = 0$. That is, the wheel velocities are constant. The vehicle is turning on the spot for $v_R + v_L = 0$, or the curvature

$$\kappa = \frac{1}{w} \frac{v_R - v_L}{v_R + v_L} \tag{3.27}$$

is constant for $v_R + v_L \neq 0$. In any case, the robot moves on a circular path, possibly with zero radius.

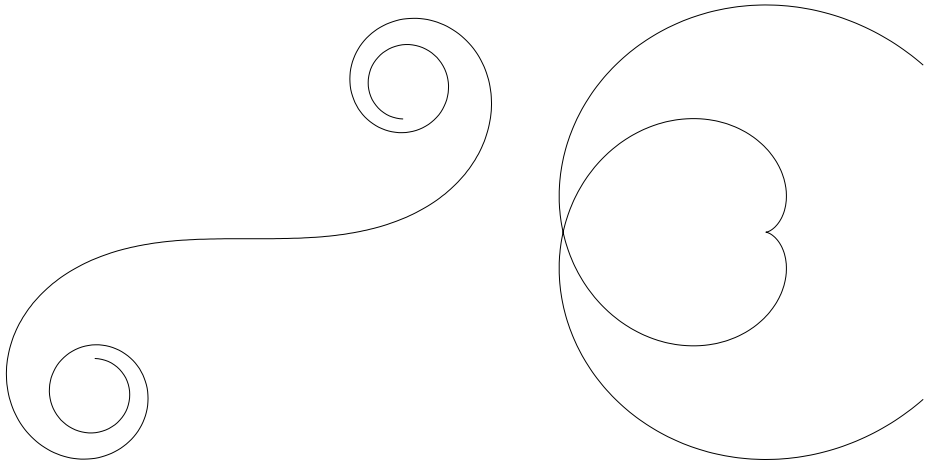
Now consider a vehicle with differential drive where accelerations a_W are applied to the wheels in opposite directions, i.e. $a_R = -a_L = a_W$. Without loss of generality, let $v_R(0) = v_L(0)$. The vehicle moves with constant velocity

$$v = \frac{(v_R(0) + a_W t) + (v_L(0) - a_W t)}{2} = \frac{v_R(0) + v_L(0)}{2}, \tag{3.28}$$

and the curvature of its path is

$$\kappa = \frac{1}{w} \frac{(v_R(0) + a_W t) - (v_L(0) - a_W t)}{(v_R(0) + a_W t) + (v_L(0) - a_W t)} = \frac{a_W}{vw} t = \frac{a_W}{v^2 w} s \tag{3.29}$$

where s is the arc length of the path. Since the curvature is proportional to the arc length, the path is a *clothoid*, also known as *Cornu spiral*. The clothoid is a well-known



(a) Clothoid (b) Involute of a circle

Figure 3.9: Paths of differential drive with extremal wheel accelerations

curve which is used for example in street construction, since driving a car on a clothoid imposes a constant turning velocity of the steering wheel. The bad news is the computational complexity of that curve, since its parametric representation is given by Fresnel integrals as

$$x(t) = S(t) = \int_0^t \sin(\frac{1}{2}\pi z^2)dz \tag{3.30}$$

and

$$y(t) = C(t) = \int_0^t \cos(\frac{1}{2}\pi z^2)dz. \tag{3.31}$$

These integrals cannot be expressed by means of elementary functions.

Now assume a vehicle B with differential drive is traveling with constant wheel accelerations $a_R = a_L = a_W$. Its angular velocity ω_B is constant, since

$$\dot{\omega}_B = \frac{\dot{v}_R - \dot{v}_L}{2w} = \frac{a_R - a_L}{2w} = 0. \tag{3.32}$$

Without loss of generality, let $v_R(0) + v_L(0) = 0$. Then, the velocity of B is

$$v(t) = \frac{(v_R(0) + a_W t) + (v_L(0) + a_W t)}{2} = \frac{v_R(0) + v_L(0)}{2} + \frac{a_W t + a_W t}{2} = a_W t, \tag{3.33}$$

the traveled arc length s of B since time $t = 0$ is

$$s(t) = \frac{1}{2} a_W t^2, \tag{3.34}$$

and the radius of curvature of the path of B is

$$\rho = 1/\kappa = w\frac{(v_R(0) + a_W t) + (v_L(0) + a_W t)}{(v_R(0) + a_W t) - (v_L(0) + a_W t)} = \frac{2w}{v_R(0) - v_L(0)} a_W t = \frac{a_W}{\omega_B} t. \tag{3.35}$$

The curvature of this path as a function of the arc length s is

$$\kappa = \frac{1}{\sqrt{\frac{2a_W}{\omega_B^2}s}}, \tag{3.36}$$

that is, the body will move on a path which is known as the *involute of a circle*, see Figure 3.9(b). Intuitively, this is the path of the end point of a string which is being unwound from a fixed reel. For a circle of radius r, the involute is given by the parametric equations

$$x(t) = r(\cos t + t \sin t), \tag{3.37}$$
$$y(t) = r(\sin t - t \cos t). \tag{3.38}$$

We subsume these observations in the following property.

Property 3.23 (Differential Drive Paths with extremal Wheel Acceleration)
Let B be a vehicle with a differential drive, and let a_R and a_L be the accelerations of the left and right wheel, respectively. Then,

(i) B *moves on a circle, if* $a_R = a_L = 0$,
(ii) B *moves on a clothoid, if* $a_R = -a_L \neq 0$, *and*
(iii) B *moves on the involute of a circle, if* $a_R = a_L \neq 0$.

The circle in (i) degenerates to a straight line if $v_R = v_L$. The clothoid in (ii) degenerates to a single point if $v_R + v_L = 0$. The involute of a circle in (iii) degenerates to a half line if $v_R - v_L = 0$.

3.2.3 Virtual Robot Center

As we have seen in the preceeding section, the complexity of navigating a vehicle with a differential drive or a synchro drive cannot be neglected, since neither the reachable velocities are isotropic (i.e., invariant with respect to direction), nor the traveled paths are simple to describe. Consequently, the dynamic and kinematic constraints can turn motion generation for a mobile robot into a non-trivial problem, which becomes even worse in the presence of obstacles. In fact, only recently a polynomial time approximation algorithm for the kino-dynamic motion planning problem in static environments became known (Reif and Wang, 2000).

Since we address partially known, dynamic environments, unforeseen situations may emerge where the robot must react quickly. Using sophisticated algorithms for kino-dynamic motion planning are hardly an option, due to their high computational complexity. Therefore, heuristic approaches which hide the complexity of kino-dynamic constraints from the actual motion planning approach are of interest.

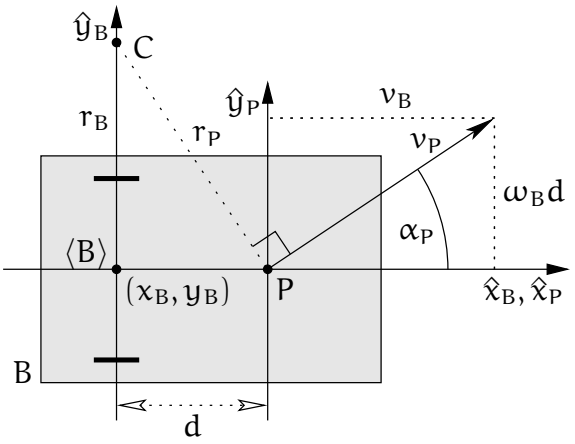


Figure 3.10: The virtual robot center P is a point located at distance d in front of the real robot center (i.e. the origin of the frame $\langle B \rangle$, located in the middle of the wheels)

We will now present a heuristic approach which we call the *virtual robot center approach*. This approach uses a subset of reachable vehicle velocities and feasible vehicle accelerations with the benefit that motion planning for a certain point P in front of the real robot center becomes easier.

So, consider such a point P situated at distance $d > 0$ in front of the point (x_B, y_B) in the middle of the two wheels, see Figure 3.10. If the current velocity of the robot is (v_B, ω_B) with $\omega_B \neq 0$, the motion of point (x_B, y_B) is a rotation around a point C located at distance r_B on the line through the wheels with angular velocity ω_B. From

$$v_B = \omega_B r_B, \tag{3.39}$$

$$v_P = \omega_P r_P = \omega_B r_P, \text{ and} \tag{3.40}$$

$$r_P^2 = r_B^2 + d^2 \tag{3.41}$$

follows

$$\begin{aligned} v_P^2 &= \omega_B^2 r_P^2 \\ &= \omega_B^2 (r_B^2 + d^2) \\ &= v_B^2 + (\omega_B d)^2, \end{aligned} \tag{3.42}$$

that is, the velocity $\mathbf{v_P}$ has a component $\omega_B d$ parallel to $\hat{y}_P$ which stems from the rotation of B, and a component v_B parallel to $\hat{x}_P$ which stems from the translation of B.

Reversing this observation, point P can be moved in any desired direction: if α_P is an arbitrary direction and v_P an arbitrary velocity, point P will move accordingly when the vehicle velocity

$$(v_B, \omega_B) = (v_P \cos \alpha_P, \frac{v_P}{d} \sin \alpha_P) \tag{3.43}$$

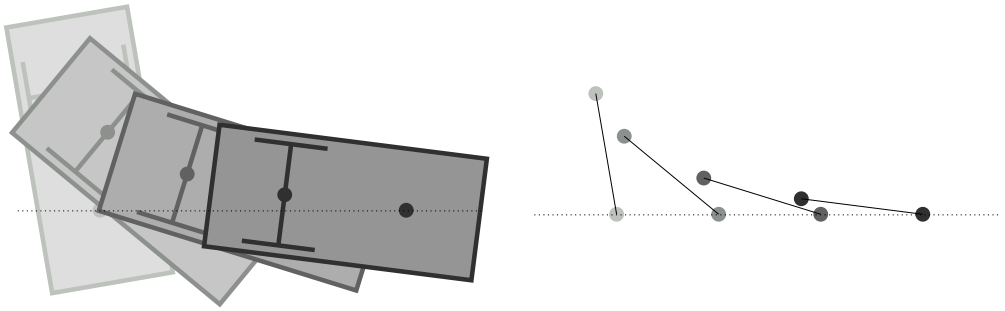


Figure 3.11: Robot motion corresponding to virtual center motion on a straight line

is assumed. Note that α_P is defined relative to $\langle P \rangle$, i.e. it rotates with the robot. In order to attain a stable direction with respect to the world coordinate system $\langle W \rangle$, the additional constraint

$$\dot{\alpha}_P = -\omega_B \tag{3.44}$$

must be fulfilled.

Following this approach, the real robot center will move on a curve known as *tractrix*, see Figure 3.11 and 3.12. Informally, a tractrix is the path of an object which, initially being situated at a lateral offset to a straight line l, is dragged along by a string of fixed length d with one end connected to the object, and the other end moving along line l with velocity v. In our context, the midpoint of the driven wheels is the dragged point, and the virtual robot center is moving on the straight line.

If we intend to use this approach to navigate a mobile robot with differential drive, the following questions arise:

(i) Given the maximum wheel velocity $\hat{v}_W$, what is the maximum feasible velocity $\hat{v}_P$ of the virtual center?

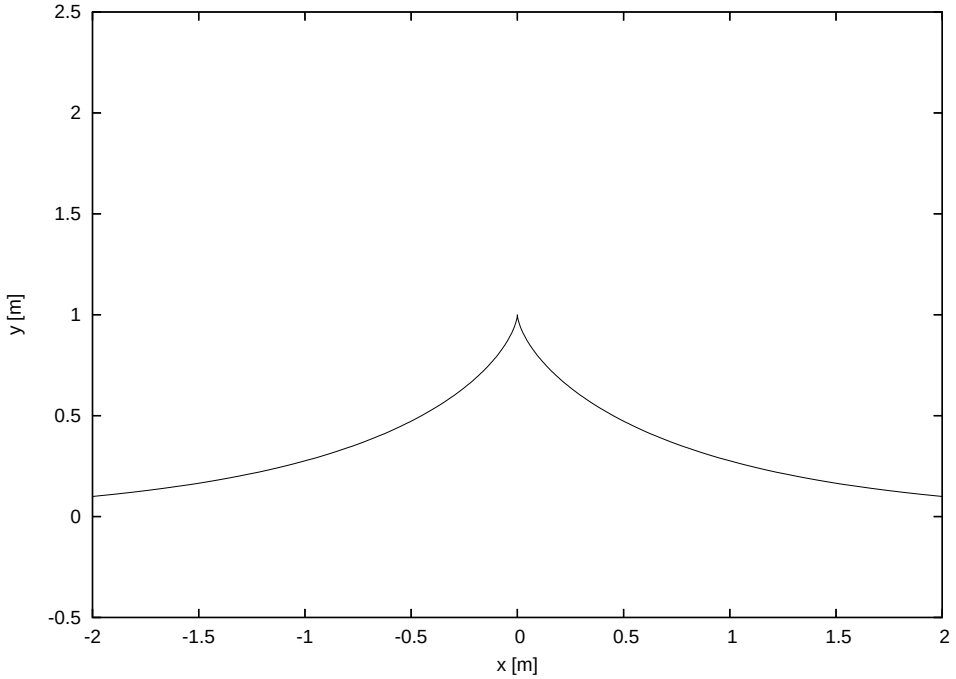
(ii) How are feasible velocities of the virtual center restricted by the maximum wheel acceleration?

(iii) Given the maximum wheel velocities and accelerations, what is the maximum linear acceleration $\dot{\mathbf{v}}_P$ that can be achieved for point P by this approach?

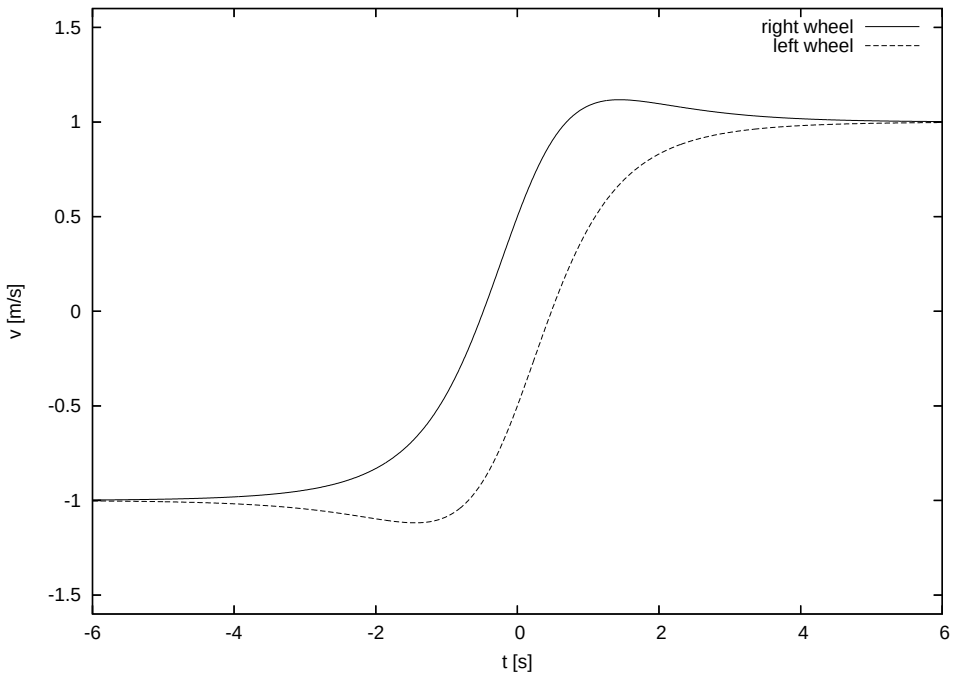
When moving the virtual center with constant velocity v_P on a straight line, the pair of velocities $(v_B, \omega_B d)$ of the robot moves on a circle

$$v_B^2 + (\omega_B d)^2 = v_P^2 \tag{3.45}$$

with radius v_P, see Figure 3.13. Therefore, in order to determine $\hat{v}_P$, we have to look for a largest in-circle of the set of feasible velocities $(v_B, \omega_B d)$. The radius $\hat{v}_P$ of that circle is

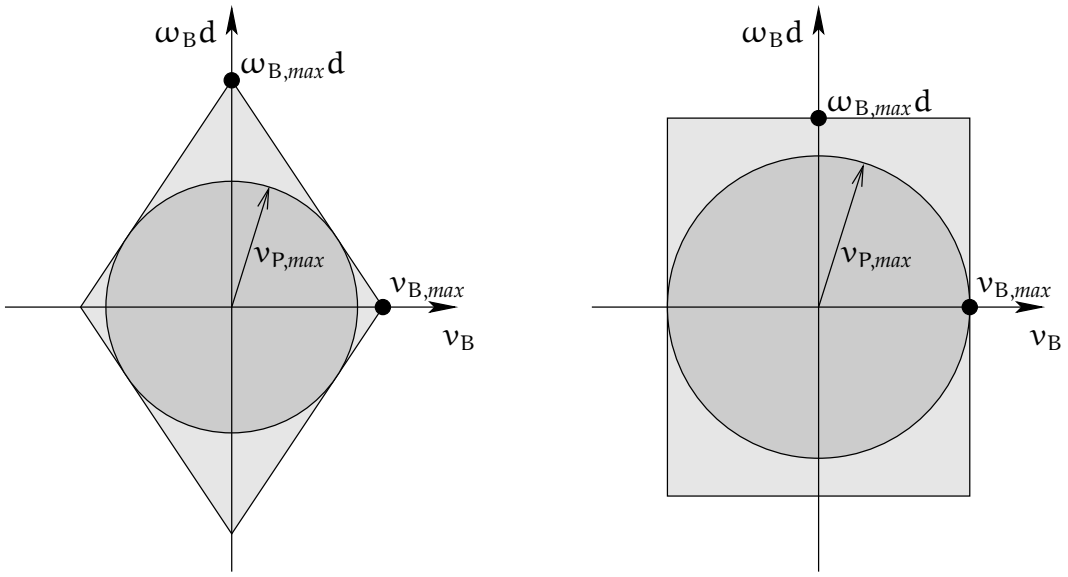


(a) Tractrix with $w/\mathrm{d} = 1$



(b) Wheel velocities for $w/\mathrm{d} = 1$

Figure 3.12: Tractrix Curve and Wheel Velocities



(a) Differential drive (b) Synchro drive

Figure 3.13: Maximum feasible virtual center velocities

easily shown to be

$$\hat{v}_P = \frac{\hat{v}_B}{\sqrt{(\frac{w}{d})^2 + 1}} \tag{3.46}$$

in the case of a differential drive with maximum wheel velocity $\hat{v}_W$, and

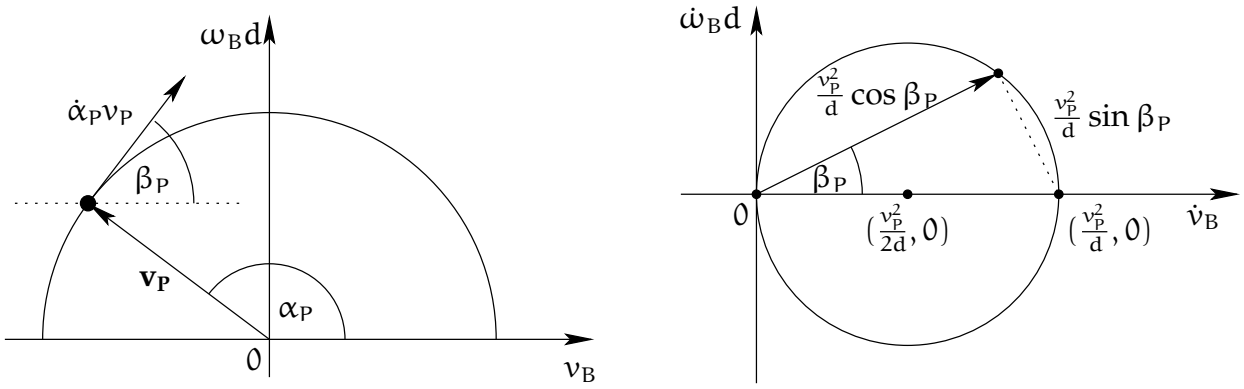
$$\hat{v}_{P,max} = \max\{\hat{v}_W, \hat{\omega}_W d\} \tag{3.47}$$

in the case of a synchro drive with maximum wheel velocity $\hat{v}_W$ and maximum angular velocity $\hat{\omega}_W$.

In the case of the differential drive, the maximum feasible virtual center velocity $\hat{v}_P$ decreases as the relative wheel distance $\frac{w}{d}$ increases. This is plausible, since when moving the virtual center sideways (parallel to the axle of the wheels), the robot rotates around its real center point, and the wheel velocity is $v_P \cdot \frac{w}{d}$, which can be forced to an arbitrary high value by increasing the wheel distance. In contrast, when moving the virtual center more or less in the natural direction of vehicle motion, the wheel velocity approaches the virtual center velocity.

Now we will examine how fast the velocity $\mathbf{v_P} = (v_B, \omega_B d)$ moves on its according circle, as the virtual center moves on a straight line with constant velocity v_P. Consider Figure 3.14(a). We know that $\mathbf{v_P}$ will rotate with angular velocity $\dot{\alpha}_P = -\omega_B$ around $\mathbf{0}$, which corresponds to a translational velocity

$$a_p = \dot{\alpha}_P v_P = -\omega_B v_P = \frac{v_p^2}{d} \sin \alpha_P \tag{3.48}$$



(a) Velocity (b) Acceleration

Figure 3.14: Vehicle accelerations for virtual center motion

in direction $\beta_P = \alpha_P - \frac{\pi}{2}$, since $\omega_B = \frac{v_P}{d}\sin\alpha_P$. Therefore, the velocity a_P of $\mathbf{v_P}$ in direction β_P is

$$a_P(\beta_P) = \frac{v_p^2}{d}\sin\alpha_P = \frac{v_p^2}{d}\sin(\beta_P + \frac{\pi}{2}) = \frac{v_p^2}{d}\cos\beta_P, \tag{3.49}$$

that is, the accelerations a_P are located on a circle, too, which is centered at $(\frac{v_P^2}{2d}, 0)$ and has radius $\frac{v_P^2}{2d}$, see Figure 3.14(b).

The accelerations on this circle must be among the feasible accelerations of the vehicle. We will examine the case of a differential drive, see Figure 3.15 Given a maximum feasible acceleration $\hat{a}_W$ for the right and left wheel, it is

$$\dot{v}_{B,max} = \frac{a_R + a_L}{2} = \hat{a}_W \quad \text{for } a_R = a_L = \hat{a}_W, \text{ and} \tag{3.50}$$

$$\dot{\omega}_{B,max} = \frac{a_R - a_L}{2w} = \frac{\hat{a}_W}{w} \quad \text{for } a_R = -a_L = \hat{a}_W. \tag{3.51}$$

The minimum distance a_S from the circle to the boundary of feasible accelerations is the maximum feasible isotropic acceleration which may be used to steer the vehicle (that is, changing the velocity v_P and line on which P moves). Considering similarity among triangles in Figure 3.15, we get

$$\left(\frac{\hat{v}_P^2}{2d} + a_S\right) : \left(\hat{a}_W - \frac{\hat{v}_P^2}{2d}\right) = \left(\frac{\hat{a}_W d}{w}\right) : \left(\sqrt{\hat{a}_W^2 + \left(\frac{\hat{a}_W d}{w}\right)^2}\right) \tag{3.52}$$

which delivers

$$\hat{a}_W = \left(\sqrt{\left(\frac{w}{d}\right)^2 + 1}\right) a_S + \left(1 + \sqrt{\left(\frac{w}{d}\right)^2 + 1}\right) \frac{\hat{v}_P^2}{2d}. \tag{3.53}$$

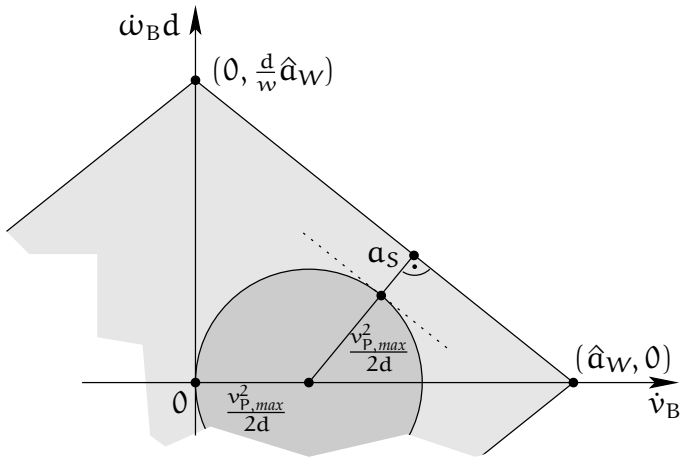


Figure 3.15: Virtual center velocity bounded by wheel acceleration

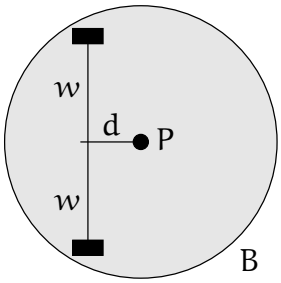


Figure 3.16: Virtual center example

The robot is steerable if $a_S > 0$, which is the case for

$$\hat{a}_W > \left(1 + \sqrt{\left(\frac{w}{d}\right)^2 + 1}\right) \frac{\hat{v}_P^2}{2d}. \tag{3.54}$$

Example

Let B be a robot with a differential drive, $w = 0.2\,m$, and $d = 0.01\,m$, see Figure 3.16. Assume we want to navigate B with virtual center velocities up to $v_P = 0.2\,\frac{m}{s}$ and steering accelerations up to $a_S = 0.1\,\frac{m}{s^2}$. According to Equation 3.46, the robot must be able to realize wheel velocities v_W up to

$$v_W = 0.2\,\frac{m}{s} \cdot \sqrt{5} \approx 0.45\,\frac{m}{s}, \tag{3.55}$$

and, according to Equation 3.53 wheel accelerations a_W up to

$$a_W = \sqrt{5} \cdot 0.1 \, \frac{m}{s^2} + \left(1 + \sqrt{5}\right) \frac{(0.2 \, \frac{m}{s})^2}{2 \cdot 0.1 \, m} \approx 0.87 \, \frac{m}{s^2} \tag{3.56}$$

3.3 Collision Avoidance

When navigating a mobile robot among moving obstacles, an important requirement is a collision-free motion. Many solutions have been suggested for static environments, the steer angle fields approach (Bauer et al., 1994) and the dynamic window approach (Fox et al., 1997) being two well-established representatives. These approaches are able to cope with dynamic environments to some degree by continuously observing the situation and reacting accordingly. However, one can expect that approaches which have been devised for dynamic environments will display better performance than the former as soon as the situation around the robot becomes dominated by changes.

In this section an approach to navigation among moving objects will be presented which has been proposed by Fiorini and Shiller (1998) and is based on *velocity obstacles*. This approach expects circular obstacles and a circular robot. Furthermore, it cannot be applied to (v, ω)-drives directly, but requires an adaptation like the virtual robot center heuristic which has been presented above.

3.3.1 Velocity Obstacles

Throughout this section we will consider a finite set

$$\mathcal{B} = \left\{ \big(D(r_i), \tau_i\big) \mid \tau_i : t \mapsto \big(x_i(t), y_i(t), 0\big), \text{ and } i = 1, 2, \ldots, n \right\} \tag{3.57}$$

of moving discs with linear motion

$$\begin{pmatrix} x_i(t) \\ y_i(t) \end{pmatrix} = \mathbf{c_i} + \mathbf{v_i} t. \tag{3.58}$$

A moving body (B_i, τ_i) from $\mathcal{B}$ collides with a different moving body (B_j, τ_j) from $\mathcal{B}$ at a certain time $t > 0$ in the future if and only if

$$\mathcal{S}\big(B_i, \tau_i(t)\big) \cap \mathcal{S}\big(B_j, \tau_j(t)\big) \neq \emptyset, \tag{3.59}$$

which is the case if and only if there are $\mathbf{p_i} \in B_i$ and $\mathbf{p_j} \in B_j$ with

$$\mathbf{p_i} + \mathbf{c_i} + \mathbf{v_i} t = \mathbf{p_j} + \mathbf{c_j} + \mathbf{v_j} t. \tag{3.60}$$

Since B_i is a disc, it is $\mathbf{p}_i' = -\mathbf{p}_i \in B_i$ for any $\mathbf{p}_i \in B_i$, and we have equivalently

$$(\mathbf{v}_i - \mathbf{v}_j) \cdot t - (\mathbf{c}_j - \mathbf{c}_i) = (\mathbf{p}_i' + \mathbf{p}_j) \tag{3.61}$$

for some $\mathbf{p}_i' \in B_i$ and $\mathbf{p}_j \in B_j$, which means

$$\begin{aligned}(\mathbf{v}_i - \mathbf{v}_j) \cdot t - (\mathbf{c}_j - \mathbf{c}_i) &\in B_i + B_j \\ &= D(r_i) + D(r_j) \\ &= D(r_i + r_j),\end{aligned} \tag{3.62}$$

or equivalently

$$(\mathbf{v}_i - \mathbf{v}_j) \cdot t \in D(r_i + r_j) + (\mathbf{c}_j - \mathbf{c}_i), \tag{3.63}$$

which means that the set of relative velocities leading to a collision at time $t > 0$ lie in a circle with radius $\frac{1}{t}(r_i + r_j)$ and center $\frac{1}{t}(\mathbf{c}_j - \mathbf{c}_i)$. We capture this observation in the following definition and property.

Definition 3.24 (Collision Circle)
We write

$$CC_{ij}(t) = \{\mathbf{v}_{ij} \in \mathbb{R}^2 \mid \mathbf{v}_{ij} \cdot t \in D(r_i + r_j) + (\mathbf{c}_j - \mathbf{c}_i)\} \tag{3.64}$$

for the set of relative velocities which lead to a collision between $(D(r_i), \tau_i)$ *and* $(D(r_j), \tau_j)$ *at time* $t \geq 0$.

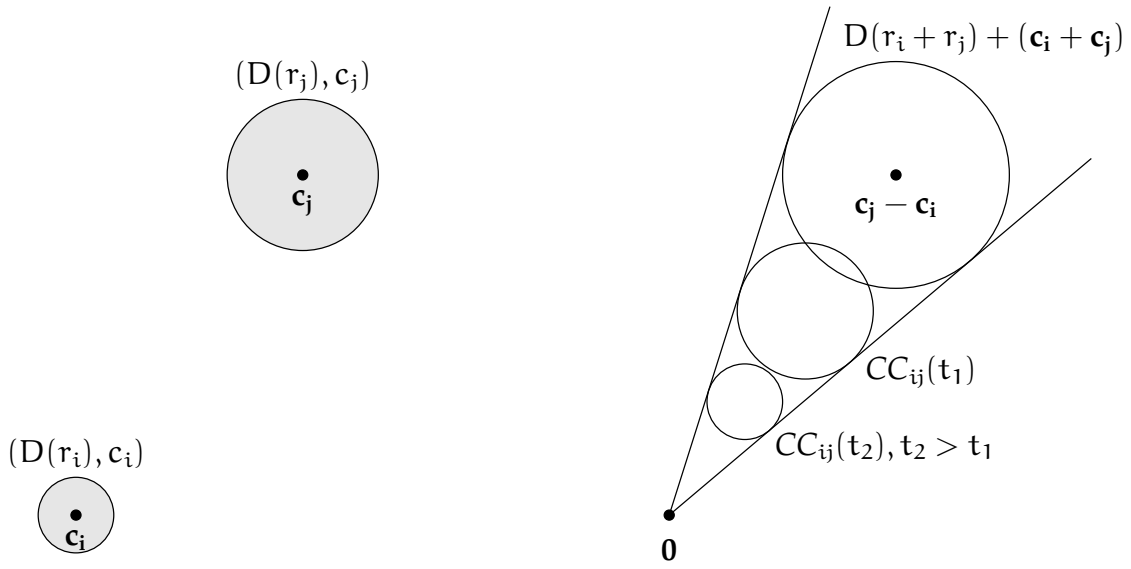
Property 3.25 (Collision Circle)
For two moving objects (B_i, τ_i) *and* (B_j, τ_j) *from a set of moving objects* $\mathcal{B}$ *as defined above, the collision circle is*

$$CC_{ij}(t) = \begin{cases} \mathbb{R}^2, & \text{if } t = 0, \text{ and } (B_i, \tau_i(0)) \text{ and } (B_j, \tau_j(0)) \text{ are colliding,} \\ \emptyset, & \text{if } t = 0, \text{ and } (B_i, \tau_i(0)) \text{ and } (B_j, \tau_j(0)) \text{ are not colliding,} \\ D\left(\frac{r_i + r_j}{t}\right) + \frac{1}{t}(\mathbf{c}_j - \mathbf{c}_i), & \text{else.} \end{cases} \tag{3.65}$$

Given two moving objects (B_i, τ_i) and (B_j, τ_j) from a set of moving objects $\mathcal{B}$ as defined above, any two collision circles $CC_{ij}(t_1)$ and $CC_{ij}(t_2)$ for $t_1, t_2 > 0$ can be transformed into each other by a central dilation with respect to the origin, see Figure 3.17. Therefore, they share the same tangent lines from the origin, and any point between or on the tangent lines except the origin is contained in $CC_{ij}(t)$ for some $t > 0$. Consequently, the union of the collision circles $CC_{ij}(t)$ for $t \in \mathbb{R}^+$ is a cone of colliding relative velocities, and the objects (B_i, τ_i) and (B_j, τ_j) are colliding if their relative velocity $\mathbf{v}_i - \mathbf{v}_j$ is contained in the union of their collision circles for $t \in \mathbb{R}_0^+$.

Definition 3.26 (Collision Cone)
Let $(D(r_i), c_i)$ *and* $(D(r_j), c_j)$ *be placed discs. Then the set*

$$CC\big((D(r_i), c_i), (D(r_j), c_j)\big) = \cup_{t \in \mathbb{R}_0^+} CC_{ij}(t) \tag{3.66}$$



(a) Workspace situation (b) Velocity space situation

Figure 3.17: Collision circles

of relative velocities is called the **collision cone** *of* $(D(r_j), c_j)$ *for* $(D(r_i), c_i)$. *When the placed discs are uniquely identified by their indices* i *and* j*, we will also write* CC_{ij} *for the collision cone of* $(D(r_j), c_j)$ *for* $(D(r_i), c_i)$.

With this notion of a collision cone, the following property collects our observations on colliding relative velocities.

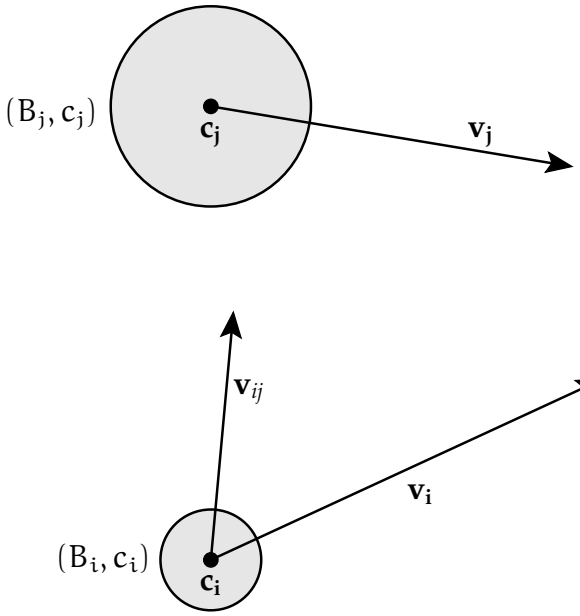
Property 3.27 (Colliding Relative Velocities)
Let $(D(r_i), \tau_i)$ *and* $(D(r_j), \tau_j)$ *be moving discs from a set of moving discs* $\mathcal{B}$ *as defined above. Then, these discs are colliding if and only if*

$$\mathbf{v_i} - \mathbf{v_j} \in CC_{ij}. \tag{3.67}$$

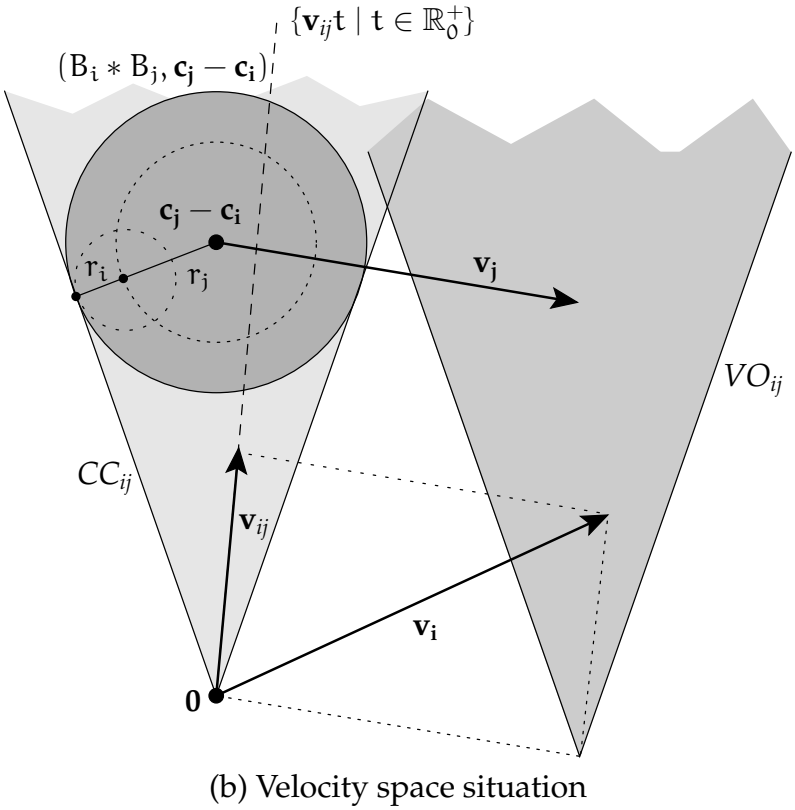
In order to be able to decide if a linear motion of $(D(r_i), \tau_i)$ with velocity $\mathbf{v_i}$ leads to a collision with $(D(r_j), \tau_j)$ moving linearly with velocity $\mathbf{v_j}$, we give the following definition of a velocity obstacle, and a proof that the velocity obstacle is the set of colliding velocities.

Definition 3.28 (Velocity Obstacle)
Let $(D(r_i), c_i)$ *be a placed disc, and* $(D(r_j), \tau_j)$ *a moving disc from a set of moving discs* $\mathcal{B}$ *as defined above. Then the set*

$$VO\big((D(r_i), c_i), (D(r_j), \tau_j)\big) = CC_{ij} + \mathbf{v_j} \tag{3.68}$$



(a) Workspace situation



(b) Velocity space situation

Figure 3.18: Construction of collision cone CC_{ij} and velocity obstacle VO_{ij}

is called the **velocity obstacle** *of* $(D(r_j), \tau_j)$ *for* $(D(r_i), c_i)$. *When the discs are uniquely identified by their indices* i *and* j, *we will also write* VO_{ij} *for the velocity obstacle of* $(D(r_j), \tau_j)$ *for* $(D(r_i), c_i)$.

Since the collision cone CC_{ij} is the area between two rays with common apex, and its Minkowski sum with the set $\{\mathbf{v_j}\}$ is merely a translation by $\mathbf{v_j}$, the velocity obstacle VO_{ij} is the area between two rays with common apex, too, and reasonably simple to compute, see Figure 3.18.

Property 3.29 (Colliding Absolute Velocities)
Let $(D(r_i), \tau_i)$ *and* $(D(r_j), \tau_j)$ *be moving discs from a set of moving discs* $\mathcal{B}$ *as defined above.*

Then, these moving discs are colliding if and only if

$$\mathbf{v_i} \in VO_{ij}. \tag{3.69}$$

Proof:

The moving discs $(D(r_i), \tau_i)$ and $(D(r_j), \tau_j)$ are colliding if and only if

$$\mathbf{v_i} - \mathbf{v_j} \in CC_{ij} \tag{3.70}$$

for $CC_{ij} = CC\big((D(r_i), \tau_i(0)), (D(r_j), \tau_j(0))\big)$, which is equivalent to

$$\exists \mathbf{v_{ij}} \in CC_{ij} : \mathbf{v_i} = \mathbf{v_{ij}} + \mathbf{v_j}. \tag{3.71}$$

This can be written as

$$\mathbf{v_i} \in CC_{ij} + \mathbf{v_j}, \tag{3.72}$$

or even shorter as

$$\mathbf{v_i} \in VO_{ij} \tag{3.73}$$

for $VO_{ij} = VO((D(r_i), \tau_i(0)), (D(r_j), \tau_j))$. □

In general, there is more than a single moving obstacle in the presence of a robot. Therefore, consider a moving disc $(D(r_0), \tau_0)$ being confronted with all the moving discs from a finite set of linearly moving discs $\mathcal{B}$ as defined above. The union of the velocity obstacles of the $(D(r_i), \tau_i)$ for $(D(r_0), \tau_0)$ can be considered as one large velocity obstacle, containing any colliding velocity.

Definition 3.30 (Velocity Obstacle of a Set of Moving Discs)
Let $\mathcal{B}$ *be a set of linearly moving discs as defined above. Let* $(D(r_0), c_0)$ *be a placed disc. The set*

$$VO\big((D(r_0), c_0), \mathcal{B}\big) = \cup_{i=1}^{n} VO\big((D(r_0), c_0), (D(r_i), \tau_i)\big) \tag{3.74}$$

is called the **(composite) velocity obstacle** *of* $\mathcal{B}$ *for the placed body* $(D(r_0), c_0)$.

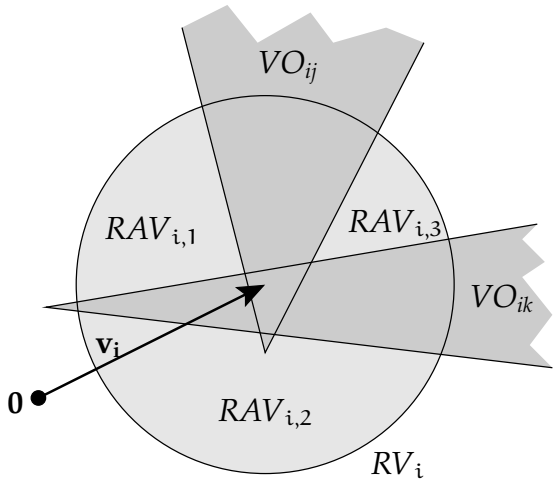


Figure 3.19: Velocity obstacle navigation

3.3.2 Navigation with Velocity Obstacles

Since the velocity obstacle $VO \subseteq \mathbb{R}^2$ of a set of moving discs contains all the colliding velocities for a placed body $(D(r_0), c_0)$, its complement $\mathbb{R}^2 - VO$ represents the velocities of collision-free linear motions.

Figure 3.19 shows the velocity obstacles VO_{ij} and VO_{ik} of two moving obstacles B_j and B_k for a moving object B_i. In this example, object B_i would collide with B_j and with B_k. If we denote the set of reachable velocities for B_i by RV, objects B_i should change its velocity to one of the regions labeled as reachable avoidance velocities $RAV_{i,1}$, $RAV_{i,2}$, and $RAV_{i,3}$ in order to avoid these collisions. Notice that there is some freedom in choosing avoidance velocities, which allow to implement different behaviors based on specific selection rules.

Algorithm 4 VELOCITY OBSTACLE NAVIGATION

1: **input:** the shape $D(r_A) \subset \mathbb{R}^2$ and configuration c_A of the robot
2: **input:** the set of reachable velocities $RV_A \subset \mathbb{R}^2$ for the robot
3: **input:** a finite set of moving obstacles $\mathcal{B}$
4: **input:** a subroutine to select a useful velocity with $select(A) \in A$ for $A \subseteq \mathbb{R}^2$
5: compute the velocity obstacle VO_{A,B_i} of each moving body $B_i \in \mathcal{B}$ for (A, c_A)
6: compute $RAV = RV - \cup_{B_i \in \mathcal{B}} VO_{A,B_i}$
7: **return** $select(RAV)$

Algorithm 4 summarizes the approach. Since each of the elementary velocity obstacles in the finite union is bounded by two rays with common apex, the set $\mathbb{R}^2 - VO$ has a polygonal boundary, which is not connected in general. That is, the composite ve-

locity obstacle and its complement can be efficiently represented on a computer using standard data structures from computational geometry.

3.3.3 Discussion

The presented approach has been implemented on the robot presented in Section 1.1. In doing so, some practical problems have been identified, which are summarized in the following.

Static Obstacles and Time Horizon

In the unmodified approach as presented above the robot is not allowed to drive directly towards an obstacle even if it is still far away. This is a severe drawback since in any indoor environment the robot is surrounded by walls preventing it from any motion. Therefore a time horizon $h \in \mathbb{R}_0^+$ is introduced by Fiorini and Shiller (1998), and only collisions occurring not later than h from the current time are considered. This is achieved geometrically by cutting off an apex part of each velocity obstacle depending on the distance to respective workspace obstacle.

Definition 3.31 (Collision Cone with Time Horizon)
Let $(D(r_i), \tau_i)$ *and* $(D(r_j), \tau_j)$ *be moving objects from a set* $\mathcal{B}$ *of moving objects as defined above, and* $h \in \mathbb{R}_0^+$ *a non-negative real number. Then the set*

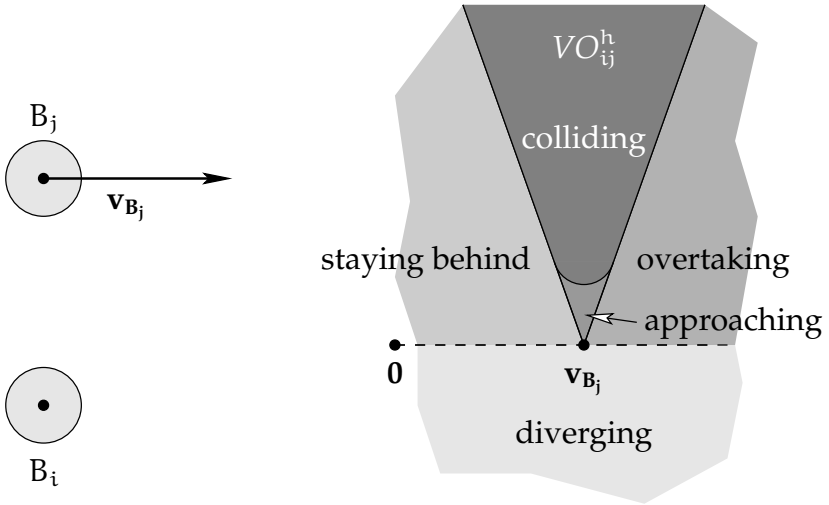
$$CC_{ij}^h = \cup_{t\in[0,h]} CC_{ij}(t) \tag{3.75}$$

of relative velocities is called the **collision cone with time horizon** h *of the placed disc* $(D(r_j), c_j)$ *for the placed disc* $(D(r_i), c_i)$.

In analogy to Definition 3.28, we will talk of a velocity obstacles with time horizon h as the set

$$VO_{ij}^h = CC_{ij}^h + \mathbf{v_j}. \tag{3.76}$$

Using these velocity obstacles with time horizon h, the robot is allowed to approach an obstacle down to arbitrarily small distances $d > 0$, decelerating and never touching it. For example, see Figure 3.20(b). Velocities from the region labeled as "approaching" will move the object B_i directly towards object B_j and will lead to a collision if they are kept. However, as the distance between these two objects decreases, the region of allowed approaching velocities will shrink, since smaller relative velocities become sufficient for a collision within time horizon h. Therefore, object B_i is forced to reduce its velocity, or to change its direction of motion.



(a) Workspace (b) Regions of velocities

Figure 3.20: Regions of absolute velocities of B_i and their meaning with respect to obstacle B_j

Interpretation of Absolute Velocities

Given a moving obstacle (B_j, τ_j), there is a meaningful partition of the set of absolute velocities of a moving object (B_i, τ_i). Figure 3.20 shows such a partition for the case where the velocity obstacle VO_{ij} of (B_j, τ_j) for (B_i, τ_i) does not contains multiples of the velocity $\mathbf{v}_{B_j} \neq \mathbf{0}$. The straight line passing through the origin and $\mathbf{v}_{B_j}$ (that is, the apex of VO_{ij}), separates the sets of diverging and non-diverging velocities. The non-diverging velocities are separated further by VO_{ij}, resulting in velocities such that B_i stays behind of B_j, B_i collides with or approaches B_j, or B_i passes in front of B_j.

Non-circular Shapes of Objects

When navigating with a real robot in real environments, the assumption of circular objects has to be dropped. A simple and general approach would be to approximate non-circular shapes by a set of circles. However, the collision circles are currently implemented as convolution of the reflected robot shape with the obstacle shape given as a polygonal chain. Clearly, these are no circles any more. Note that the robot shape may change over time due to rotations. This effect is neglected at the moment, which unfortunately gives rise to problems in narrow environments. For example the robot may refuse to drive through a narrow door if it is not properly aligned and the wall to the left and to the right of the doorway coalesce after obstacle growing. This could be overcome for example by locally using a configuration space planner, considering the environment as static for a short interval of time.

Integration of Other Sensors

The approach as presented so far uses laser range finder images for obstacle detection and tracking as shown in Chapter 2. Clearly, a laser range finder can only detect objects that intersect its range of view, which is more or less a horizontal plane (neglecting the divergence of the laser beam). Objects below or above that plane cannot be perceived, and collisions with them are not avoided so far.

To overcome this problem, additional sensors have to be deployed. For example it is possible increase the safety of operation by integrating a ultrasonic sensor system. Whenever an ultrasonic sensor reports an obstacle which is not visible in the laser scan, we can declare velocities which might lead to a collision with that obstacle as unreachable, thereby effectively avoiding that obstacle.

Experiments

The presented navigation scheme has been tested in an autonomous mode where the task was to traverse the concourse of a railway station in the presence of pedestrians. More important in the context of this thesis is its role as a basis for the cooperative motion coordination approach which is presented with more experimental details in the succeeding chapter.

3.4 Conclusion

At the beginning of this chapter, the basic concepts for mobile robot motion have been presented. After a technical introduction, two widely used approaches of implementing mobile robot kinematics have been considered together, with their kinematic and dynamic properties and restrictions. As an original contribution, the virtual robot center approach has been proposed and analyzed as a method to circumvent kinematic and dynamic constraints of a differential drive. Furthermore, the concept of velocity obstacles has been introduced on a formal basis. In combination with the virtual robot center approach, velocity obstacles have been shown to be suitable for creating a navigation system which allows collision-free motion among moving obstacles. Implemented on a robotic wheelchair, this system has been experimentally evaluated in public environments such as the concourse of the central station of Ulm.

Chapter 4

Cooperative Motion Coordination

4.1 Introduction

In this part of this thesis we study the problem of coordinating the motion of a mobile robot and a human through a populated, continuously changing natural environment. This problem has an application in the transportation of disabled or elderly people or the transportation of patients in a hospital. There, transportation services are usually carried out by nursing personnel who push the patient or disabled person sitting or lying in some type of carriage, for example a wheelchair or a hospital bed. Since pushing and maneuvering a heavy carriage exposes the back of the pushing person to significant strain, these people often suffer severe long-term back problems. Using a robotic wheelchair or hospital bed, which is able to accompany the nurse side by side like a heeling dog, through arbitrarily populated, continuously changing natural environments would certainly allow the reduction of this problem or even avoid it.

Accompanying an object or a person side by side involves the control of the position relative to the accompanied person. Besides this, there are further constraints which affect the heeling of a person. Ideally, the robot and the person should move at the same velocity. So, accompanying a person side by side is not only a position control problem but at the same time a velocity control problem. As a final constraint, the target position, inferred from the predicted position of the accompanying person may be perturbed through obstacles in the environment.

4.1.1 Related Work

Previously some work has been conducted considering following behaviors for mobile robots, for example by Schlegel et al. (1998) or Sidenbladh et al. (1999), but they tend to focus on computer vision topics (e.g. tracking a moving person), and widely ignore dynamic aspects needed for real motion coordination. Recently, some researchers use

laser range finders to track people in populated environments for interactive robot applications, for example for a museum tour guide as proposed by Schulz et al. (2001).

The related problem of intercepting a moving target is relevant for military applications and has been largely studied in that context (Zarchan, 1998). The main difference is that there the goal is to collide with the moving target (in general with non-zero relative velocity), where we strive to reach and keep a position besides the guide with vanishing relative velocity.

There are some theoretical results on the complexity of pursuit evasion games, that is, Reif and Tate (1993) prove hardness for exponential time for 3-dimensional polyhedral pursuit games with bounded velocities.

Finally, there is another problem called *motion coordination* where one has to plan the simultaneous motion of multiple robots. Clearly, that problem is related to the topic of this thesis only marginally.

4.1.2 Overview

Accompanying a human through busy environments requires the robot to pursue two different, sometimes conflicting goals: staying close to the human and moving parallel on the one hand, and avoiding collisions with environmental obstacles or the human guide himself one the other hand. Achieving this second goal ensures survival of the robot, while accomplishing the first goal means succeeding in the mission.

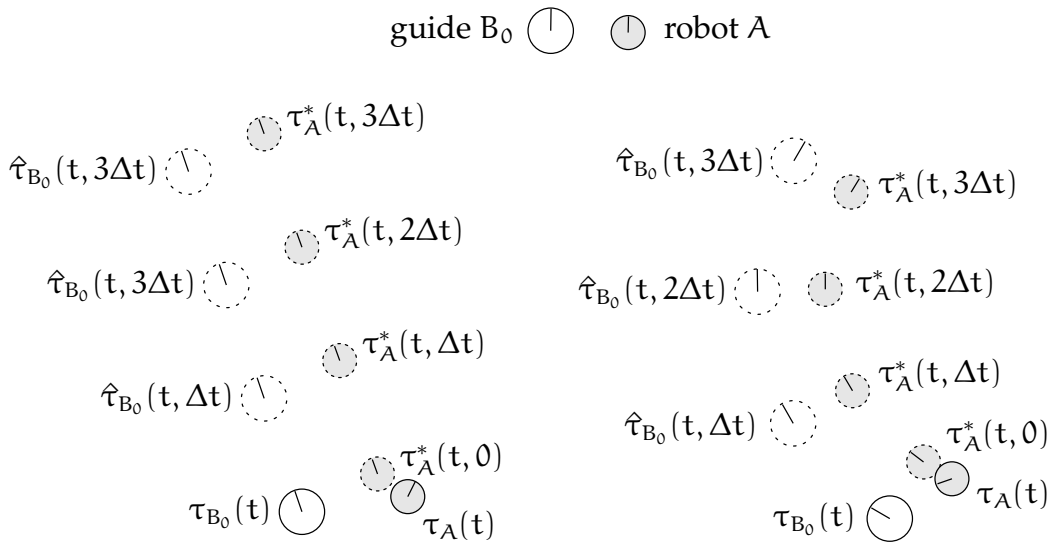
Therefore an approach to motion coordination incorporates at least two layers: a top-level strategic layer to direct the robot towards states desired for motion coordination, and a tactical layer below to avoid collisions with obstacles. The approach described in this thesis employs two layers as above and a basic third low-level layer (the operational layer) that controls the robot velocity.

The remainder of this chapter is organized as follows. In Section 4.2, the problem is introduced in a more formal manner, before Section 4.3 portrays a practical approach to the stated problem. The method is described as an extension to the employed obstacle avoidance scheme presented in the previous chapter. Section 4.4 outlines conducted experiments, and the results are discussed in Section 4.5.

4.2 Problem Description

Let the environment of robot A contain a set of moving objects $\mathcal{B} = \{(B_i, \tau_i) \mid i = 1, 2, \ldots, n\}$. For simplicity we presume that the robot and the objects are of circular shape with radii r_A for the robot and $r_1, r_2, \ldots, r_n$ for the objects in $\mathcal{B}$.

In the following, τ_i denotes the trajectory of object B_i, and $\hat{\tau}_i(t, \Delta t)$ denotes a predicted



(a) Linear motion (b) Circular motion

Figure 4.1: Motion coordination problem

configuration of object B_i, estimated at time t to be occurring at time $t + \Delta t$. Analogously, $\tau^*_A(t, \Delta t)$ denotes a configuration for the robot A that is considered at time t to be optimal at time $t + \Delta t$.

Let one of the objects, say object B_0, be a guiding motion partner of the robot. That is, we want the robot A to stay in a fixed configuration relative to object B_0, for example half a meter to the right of B_0, see Figure 4.1.

At time t, the robot A knows its own configuration $\tau_A(t)$, and detects the position $(x_{B_0}(t), y_{B_0}(t))$ of its guide. From previous guide positions $(x_{B_0}(t - \Delta t), y_{B_0}(t - \Delta t))$, the robot may infer the current guide orientation $\theta_{B_0}(t)$ (thereby the current configuration $\tau_{B_0}(t)$ of the guide) and velocity $(v_{B_0}(t), \omega_{B_0}(t))$, which allows to predict future configurations $\hat{\tau}_{B_0}(t, \Delta t)$ of the guide, and to compute future desired states $\tau^*_A(t, \Delta t)$ for the robot.

4.2.1 Dynamic Game Formulation

In our specific case, the robot A has a differential drive, that is, two independently driven wheels at distance $2w$ with fixed maximum wheel velocity $v_{W,max}$ and maximum wheel acceleration $a_{W,max}$. Let a_R and a_L be the accelerations of the right and left wheel,

respectively. Then, the kinematic equations

$$\dot{x}_A = v_A \cos\theta_A \tag{4.1}$$
$$\dot{y}_A = v_A \sin\theta_A \tag{4.2}$$
$$\dot{\theta}_A = \omega_A \tag{4.3}$$
$$\dot{v}_A = \frac{a_R + a_L}{2} \tag{4.4}$$
$$\dot{\omega}_A = \frac{a_R - a_L}{2w} \tag{4.5}$$

describe feasible robot motion (up to the velocity bounds) with state variables

$$(x_A, y_A, \theta_A, v_A, \omega_A) \in \mathbb{R}^5 \tag{4.6}$$

and control variables

$$(a_R, a_L) \in [-a_{W,max}, a_{W,max}]^2. \tag{4.7}$$

If the guide kinematics and dynamics are modeled similarly to the robot, the motion coordination problem can be formulated as a differential (or dynamic) game (Isaacs, 1965) between the guide and the follower. Then, the state variables of the game are given by $(x_A, y_A, \theta_A, v_A, \omega_A)$ as above for the robot (the *pursuer*), and the corresponding state vector $(x_B, y_B, \theta_B, v_B, \omega_B)$ for the guide (the *evader*). The control variables are the same as in the respective kinematic equations, namely a_R and a_L for the pursuing robot. In the worst case for the robot, the guide will try to evade the follower. The time $t^* - t_0$ elapsed from the start of the game until the robot reaches a desired configuration $\tau_A(t^*) = \tau_A^*(t^*, 0)$ is the payoff that the evading guide strives to maximize and the robot tries to minimize. If we consider a reduced state space where the configuration of the guide is described relative to the robot coordinate frame, a strategy for the robot is a function

$$\phi : (\mathbb{R}^2 \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R}) \to \mathbb{R} \times \mathbb{R} \tag{4.8}$$

which maps the current position and orientation of the guide B_0 relative to $\langle A \rangle$, the velocities (v_B, ω_B) of the guide, and (v_A, ω_A) of the robot to values for the control variables (i.e. the wheel accelerations) of the robot.

This approach follows a clear theoretical concept, however, it displays several drawbacks. First, the kinematics and dynamics of a human guide (or more specifically, of human gait) have to be modeled. Furthermore, presuming the guide to be an antagonistic evader appears rather unrealistic in the context of cooperative motion coordination. Finally, the differential game approach becomes difficult or even inadequate as soon as obstacles in the environment have to be considered (Isaacs, 1965, page 152), which serves as a motivation for an alternative approach.

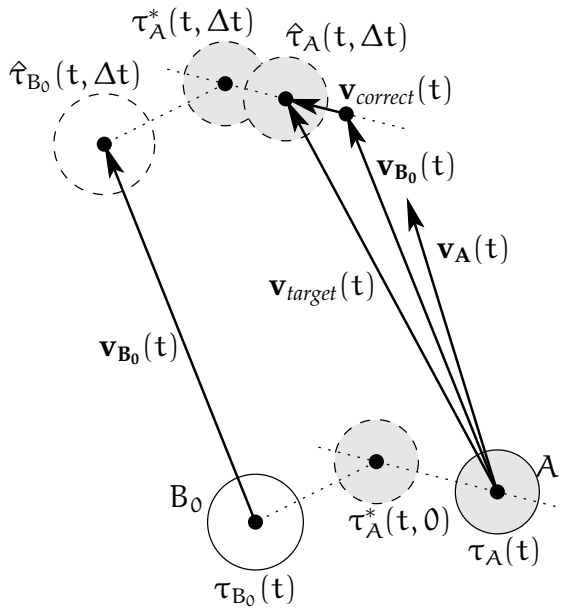


Figure 4.2: Computing the target velocity $\mathbf{v}_{target}(t)$

4.3 Practical Motion Coordination

This section portrays our approach to the given problem, which is described as an extension to the collision avoidance scheme presented in the previous chapter. As suggested before, we exploit the freedom remaining in the choice of an avoidance velocity to implement the motion coordination behavior.

4.3.1 Target Velocity

In general, the robot is neither located at a position (x^*, y^*) nor moving with a velocity $\mathbf{v}^*$ suitable for motion coordination. So the goal is to control the motion of the robot such that it approaches both the desired accompanying position and the velocity $\mathbf{v}_{\mathbf{B_0}}$ of the guide B_0. To achieve this, a target velocity

$$\mathbf{v}_{target}(t) = \mathbf{v}_{\mathbf{B_0}}(t) + \mathbf{v}_{adjust}(t) \tag{4.9}$$

is computed, see Figure 4.2. The addend $\mathbf{v}_{adjust}(t)$ to the current velocity $\mathbf{v}_{\mathbf{B_0}}(t)$ of the guide is used to control the relative position of the robot, and it fulfills

$$\mathbf{v}_{adjust}(t) = \lambda \cdot \left(\tau^*_A(t, 0) - \tau_A(t)\right) \tag{4.10}$$

with $\lambda \in \mathbb{R}^+$. The actual value of λ depends on the distance between the actual position $\tau_A(t)$ and the desired configuration $\tau^*_A(t, 0)$, as well as the dynamic capabilities of the robot. One may think of a virtual link or spring here.

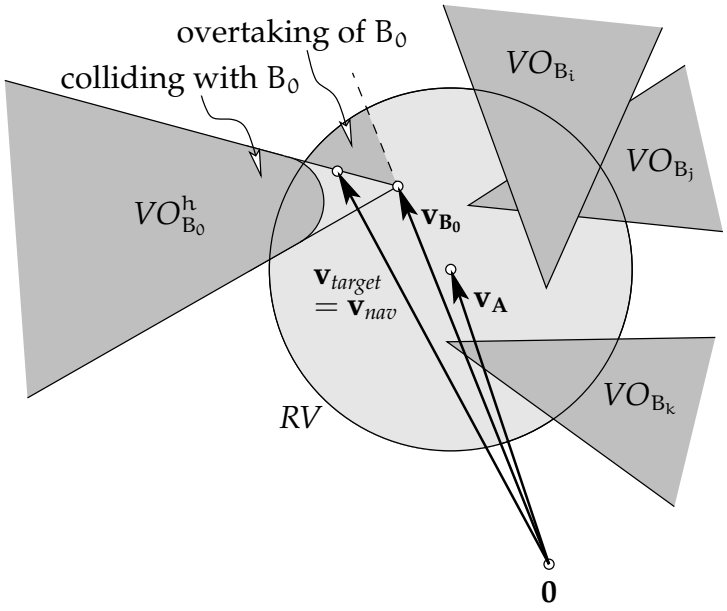


Figure 4.3: Selecting an avoidance velocity $\mathbf{v}_{nav}$ for cooperative motion coordination. The robot selects an avoidance velocity $\mathbf{v}_{nav}$ minimizing the difference to the target velocity $\mathbf{v}_{target}$. Since $\mathbf{v}_{target}$ is neither colliding nor overtaking here, it is taken directly as $\mathbf{v}_{nav}$.

4.3.2 Integration with Obstacle Avoidance

Now the notion of the target velocity is filled with meaning, as the obstacle avoidance module selects an avoidance velocity $\mathbf{v}_{nav}(t)$ minimizing the difference to the target velocity $\mathbf{v}_{target}(t)$.

For the set of moving obstacles $\mathcal{B}$, the composite velocity obstacle

$$VO_{A,\mathcal{B}} = \cup_{i=1}^{n} VO_{A,(B_i,\tau_i)} \tag{4.11}$$

is computed and the set of reachable velocities

$$RV \subset \mathbb{R}^2 \tag{4.12}$$

is determined. From the set of reachable avoidance velocities

$$RAV = RV - VO_{A,\mathcal{B}} \tag{4.13}$$

a velocity $\mathbf{v}_{nav}$ with minimum difference to $\mathbf{v}_{target}$ is selected, that is,

$$\mathbf{v}_{nav} \in RAV, \text{ and} \tag{4.14}$$

$$|\mathbf{v}_{nav} - \mathbf{v}_{target}| = \min_{\mathbf{v} \in RAV} |\mathbf{v} - \mathbf{v}_{target}|. \tag{4.15}$$

This selection is illustrated by Figure 4.3, continuing the example from Figure 4.2.

Velocities resulting in unwanted overtaking of the guide can be identified and forbidden easily by modifying the velocity obstacle VO_{B_0} of the guide.

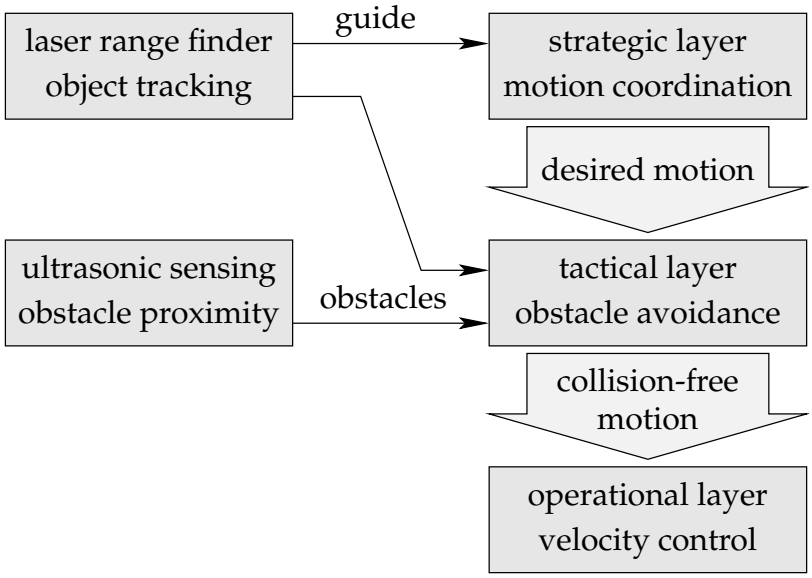


Figure 4.4: Interaction between modules and layers for cooperative motion coordination

4.4 Experiments

The presented approach to cooperative motion coordination has been implemented on the robot presented in Section 1.1. Figure 4.4 visualizes the coarse interaction between the involved hardware and software modules. The system has been extensively and successfully tested in the concourse of the central station and in the pedestrian area of Ulm during regular business hours.

The mission was for the robot to accompany a person side by side in a lateral distance of 60 cm through the concourse. The concourse has a size of about $20 \times 40\,\mathrm{m}^2$, with several rows of seats, an information booth and several ticket machines. During the experiments typically between 50 and 100 people were constantly staying and moving in the concourse.

In the pedestrian area, the robot accompanied a person from the central station to the Münster, which is a distance of about 500 meters (see Figure 4.5) where the sensors where exposed to the sun.

The total mission time adds up to 8–10 hours distributed over several days. The distance traveled during that time adds up to around three kilometers. Due to visibility problems

such as occlusion, the robot several times lost the person which was to accompany and then stopped. There was no collision between the robot and a pedestrian.

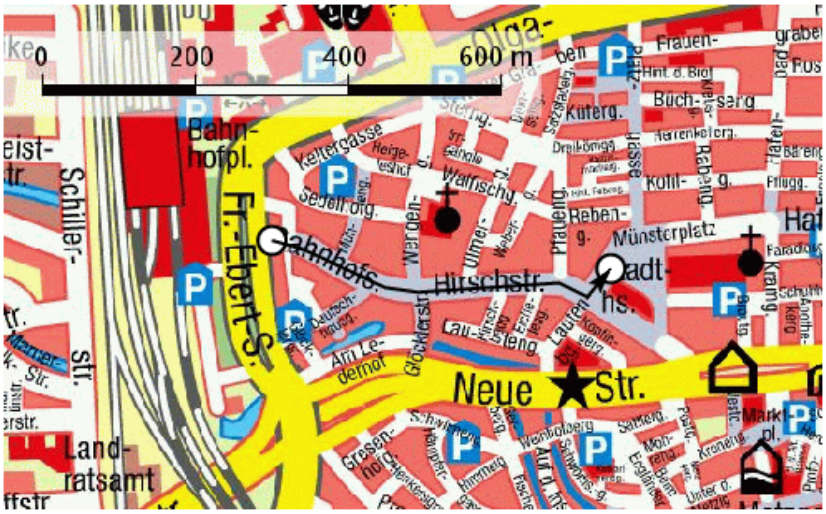


Figure 4.5: Map of Ulm downtown, with path from central station to Münster

4.5 Conclusion

Motion coordination has been presented as a problem related to motion planning in dynamic environments and to kino-dynamic motion planning, both of which are computationally hard or even intractable in their exact form. Therefore we may not expect an exact solution but have to rely on approximations and heuristics.

Our test application is a robotic wheelchair accompanying a person through the concourse of a railway station or a pedestrian area moving side by side with the person. During several experiments the robot successfully managed to accompany a person through a populated concourse over a total distance of around three kilometers with a total mission time of about 8–10 hours.

Currently, the motion of the guide is predicted only by linear extrapolation. It should be not too complicated to consider angular velocity (i.e. circular motion), too. Furthermore, opponent modeling appears to be a promising idea from game theoretic domains, since better models of humans moving in the environment (their goals, attitudes towards other humans or the robot, etc.) allow a more precise prediction of their future motion. Cooperative games might be addressed to model the problem of motion coordination more accurately, too.

Chapter 5

Similarity of Paths

5.1 Introduction

When considering cooperative motion coordination behaviors as presented in Chapter 4, one can ask for the similarity between the motion of the guide and the motion of the robot. This question may arise for example when several approaches to cooperative motion coordination are compared to each other in order to determine a "best" approach which shall be used in an application (e.g. in a museum tour guide, robotic wheelchair etc.), where the similarity outcomes are evaluated by a human engineer.

Another situation where a similarity measure (or, an abstract distance measure) for paths is useful is in the context of user interfaces for mobile robots. For example, Schlegel and Kämpke (2001) presented a system where the user may input an inprecise path for a mobile robot, which is transformed into another path which is similar to the original path but optimized with respect to some objective function. Specifically, the proposed system transforms a given path into a shortest path from the same homotopy class as the original path.

There is some related work on distances between point sets or curves. For example the Hausdorff distance for point sets is well known, and the Fréchet metric is an intuitive distance measure for curves. Furthermore, there are efficient algorithms which compute the values of these distance measures for planar polygonal chains.

However, obstacles in the ambient space of the curves appear to be widely ignored. Since we are mainly interested in the cooperative motion of robots and humans in natural environments, which, in general, may contain an arbitrary number of obstacles, we take the opportunity and define a topological metric on the the set of path homotopy classes.

To accomplish this task, the basic concepts concerning curves and homotopy classes are presented first. Subsequently, the Hausdorff metric and the Fréchet metric are presented

as examples of existing and established distance measures, and efficient algorithms for their computation in the case of planar polygonal chains are described as proposed by Alt et al. (1995) and Alt and Guibas (1996). Finally, the major part of this chapter is dedicated to the definition of a topological metric, and a description of its computation for plane curves.

5.2 Curves

The most basic concept in this chapter is the curve. Intuitively, a curve is something that can be drawn with a pen without jumping from one position to a different position. This idea is specified more abstractly in the following definition.

Definition 5.1 (Curve, Path)
A **curve in a space** S *is a continuous mapping* $c : [a, b] \to S$.

A curve c *is called* **simple**, *if its restriction to* $[a, b)$ *is injective. A curve* c *is called* **closed**, *if* $c(a) = c(b)$.

A curve in $\mathbb{R}^2$ *is called a* **plane curve**. *A simple closed plane curve is called a* **Jordan curve**.

The point $c(a)$ *is called source*(c), *and the point* $c(b)$ *is called target*(c). *A curve* c *is called* (s, t)**-curve**, *if source*$(c) = s$ *and target*$(c) = t$.

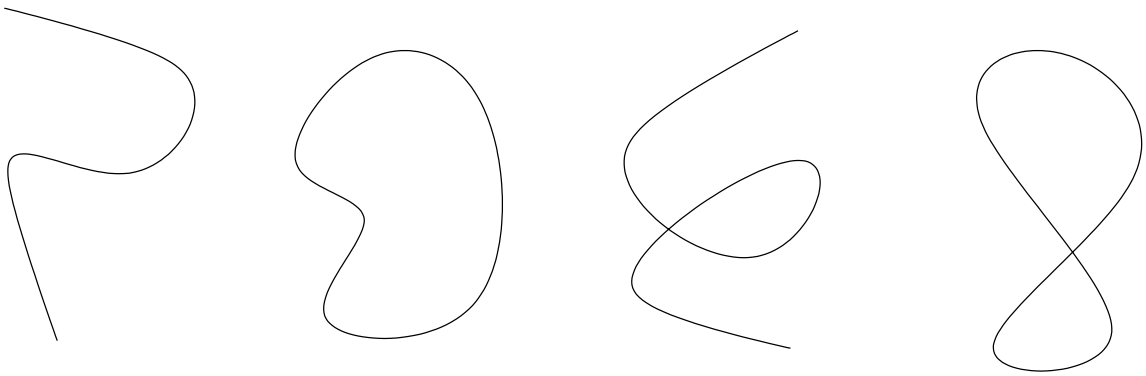


Figure 5.1: Examples of curves

Figure 5.1 shows images of example curves in $\mathbb{R}^2$. Curve c_1 is simple and not closed, curve c_2 is simple and closed, curve c_3 is neither simple nor closed, and curve c_4 is not simple but closed.

Later on we will need the famous Jordan Curve Theorem, which is given below without a proof.

Theorem 5.2 (Jordan Curve Theorem)
If c is a Jordan curve, then the complement of the image of c has two distinct connected components, a bounded interior and an unbounded exterior, with c being the boundary of each.

If one curve stops where another curve starts, these two curves together can be considered as one curve, i.e. as the concatenation of the two former curves. Furthermore, when a plane curve is drawn with a pen, we can reverse the motion of the pen an get a reversed version of the curve.

Definition 5.3 (Concatenation, Reverse Curve)
Let $c_1 : [a_1, b_1] \to S$ and $c_2 : [a_2, b_2] \to S$ be curves with $c_1(b_1) = c_2(a_2)$. Then, the **concatenation** *of c_1 and c_2 is the curve*

$$c_1 \cdot c_2 : [a_1, b_1 + b_2 - a_2] \to S \tag{5.1}$$

with

$$(c_1 \cdot c_2)(t) = \begin{cases} c_1(t) & \text{if } t \leq b_1, \\ c_2(t - b_1 + a_2) & \text{else,} \end{cases} \tag{5.2}$$

and the **reverse curve** *of c_1 is the curve $\overline{c_1} : [a_1, b_1] \to S$ with*

$$\overline{c_1}(t) = c_1(a_1 + b_1 - t) \tag{5.3}$$

Intuitively, particularly simple curve can be drawn using a ruler, with constant speed of the pen. These will be called linear curves, and concatenations of linear curves will be called polygonal chains.

Definition 5.4 (Linear Curve)
A curve $c : [a, b] \to \mathbb{R}^d$ is called **linear**, *if*

$$c(t) = c(a) + \frac{t - a}{b - a}(c(b) - c(a)).$$

Definition 5.5 (Polygonal Chain)
A curve $c : [a, b] \to \mathbb{R}^d$ is called **polygonal chain**, *if it is the concatenation*

$$c = c_1 \cdot c_2 \ldots c_n \tag{5.4}$$

of a finite sequence of linear curves $c_i : [a_i, b_i] \to \mathbb{R}^d$. A point $p = c_i(b_i) \in \mathbb{R}^d$ for $1 < i < n$ is called **an inner vertex of** *c, if $c_i \cdot c_{i+1}$ is not a linear curve. A point $p \in \mathbb{R}^d$ is called* **a vertex of** *c, if $p = c(a)$, or $p = c(b)$, or p is an inner vertex of c.*

Definition 5.6 (Length of a Polygonal Chain)
Let $c : [a, b] \to \mathbb{R}^d$ *be a polygonal chain with* $c = c_1 \cdot c_2 \ldots c_n$ *and* $c_i : [a_i, b_i] \to \mathbb{R}^d$.

Let m *be the number of vertices of* c. *Then the* **link length** $l_\#(c)$ *of the polygonal chain* c *is the number* $m - 1$. *The* **Euclidean length** $l_2(c)$ *of the polygonal chain is*

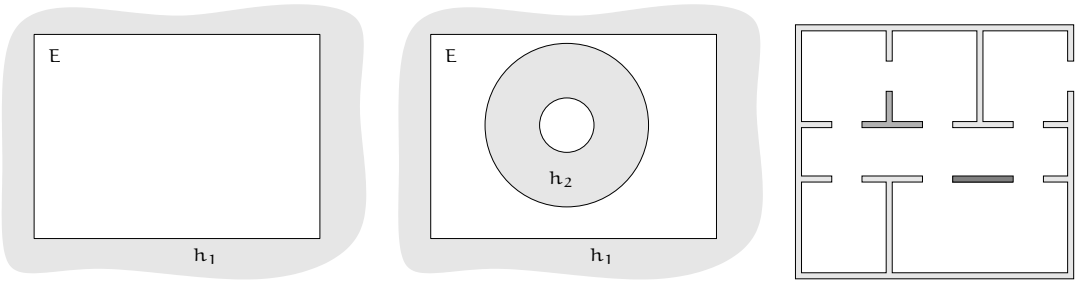
$$l_2(c) = \sum_{i=1}^{n} d_2(c_i(a_i), c_i(b_i)) \tag{5.5}$$

where d_2 *denotes the Euclidean distance.*

5.2.1 Homotopy

In a space with obstacles, there are various ways of moving from a start position to a goal position with respect to the position of the obstacles. A simple example is a room with two doors to a corridor: taking either of them yields paths from different "classes".

Definition 5.7 (Space with Obstacles)
A **space with obstacles (of dimension** d**)** *is a connected set* $E \subseteq \mathbb{R}^d$, *where the* **obstacle set** $H = \mathbb{R}^d \backslash E$ *is the union of a finite number of disjunct, closed, and connected sets* $h_i \subset \mathbb{R}^d$ *(i.e. the obstacles, or holes).*



(a) (b) (c)

Figure 5.2: Examples of spaces with obstacles

As an example, Figure 5.2(a) depicts a bounded 2-dimensional space with one (unbounded) obstacle, but the structure depicted by Figure 5.2(b) does not represent a valid space with obstacles since E is not connected. Finally, Figure 5.2(c) shows an unbounded space with obstacles where the obstacle set consists of three disjunct holes.

Definition 5.8 (Homotopy of Closed Curves)
Let E *be a space with obstacles. Let* $c_1 : [a_1, b_1] \to E$ *and* $c_2 : [a_2, b_2] \to E$ *be closed curves. Curve* c_1 *is said to be* **homotopic** *to curve* c_2, *if there is a continuous mapping* $\Gamma : [0, 1] \times [0, 1] \to E$ *such that*

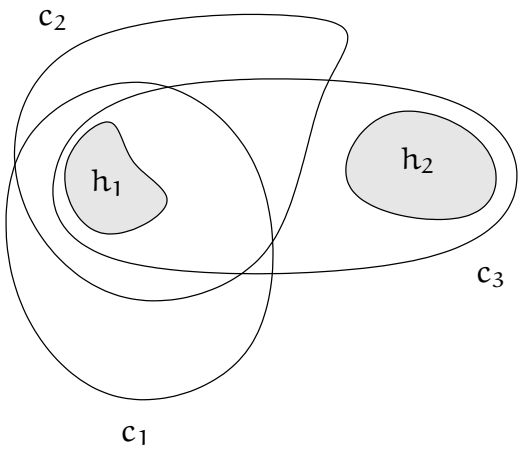


Figure 5.3: Examples of homotopic curves. Curve c_1 is homotopic to curve c_2, but not homotopic to curve c_3.

(i) $\Gamma(0,t) = c_1(a_1 + t(b_1 - a_1))$ *for* $t \in [0,1]$,

(ii) $\Gamma(1,t) = c_2(a_2 + t(b_2 - a_2))$ *for* $t \in [0,1]$,

(iii) $\Gamma(s,0) = \Gamma(s,1)$ *for* $s \in [0,1]$, *and*

(iv) $\Gamma(s,t) \in E$ *for* $s,t \in [0,1]$.

In the definition above, the mapping Γ continuously transforms curve c_1 into curve c_2 with increasing values of its first argument (cf. (i) and (ii)), whereby each intermediate curve is still closed (cf. (iii)) and does not intersect any obstacle (cf. (iv)).

Figure 5.3 shows a space with two obstacles h_1 and h_2, as well as images of three curves c_1, c_2, and c_3. Curve c_1 is homotopic to c_2, but neither c_1 nor c_2 are homotopic to c_3.

Property 5.9 (Homotopy is an Equivalence Relation)
Homotopy is an equivalence relation on the set of closed curves in a space E. *The respective equivalence classes are called* **homotopy classes**.

Proof:

For any closed curve $c : [a,b] \to E$, the mapping $\Gamma(s,t) = c(a + t(b-a))$ proves reflexivity.

Let $c_1 : [a_1,b_1] \to E$ and $c_2 : [a_2,b_2] \to E$ be closed curves, and c_1 homotopic to c_2 via $\Gamma : [0,1] \times [0,1] \to E$. Then, c_2 is homotopic to c_1 via Γ' with $\Gamma'(s,t) = \Gamma(1-s,t)$, i.e. homotopy is symmetric.

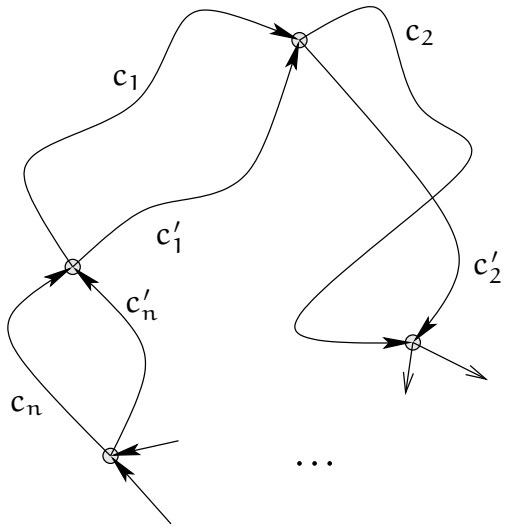


Figure 5.4: Homotopy of composed curves

Let $c_1 : [a_1, b_1] \to E$, $c_2 : [a_2, b_2] \to E$, and $c_3 : [a_3, b_3] \to E$ be closed curves, c_1 homotopic to c_2 via Γ_1, and c_2 homotopic to c_3 via Γ_2. Then, c_1 is homotopic to c_3 via Γ_3 with

$$\Gamma_3(s,t) = \begin{cases} \Gamma_1(2s,t) & \text{if } s \leq 0.5, \\ \Gamma_2(2s-1,t) & \text{if } s > 0.5. \end{cases} \tag{5.6}$$

□

Property 5.10 (Homotopy of Composed Curves)
Let E be a space with obstacles, let c_i and c'_i for $i \in \{1, 2, \ldots, n\}$ be curves in E with

(i) source(c_i) = source(c'_i) and target(c_i) = target(c'_i) for $i \in \{1, 2, \ldots, n\}$,
(ii) target(c_i) = source(c_{i+1}) (mod n), and
(iii) c_i and c'_i are path homotopic for $i \in \{1, 2, \ldots, n\}$,

see Figure 5.4. Then, the closed curves $\prod_{i=1}^{n} c_i$ and $\prod_{i=1}^{n} c'_i$ are homotopic.

Proof:

Let Γ_i be a continuous mapping which transforms the closed curve $c'_i \cdot \overline{c_i}$ to the single point *source*(c_i) without crossing an obstacle. Then, there is a continuous mapping Γ'_i which transforms the curve c_i to the curve $c'_i \cdot \overline{c_i} \cdot c_i$ without crossing an obstacle. A combination of such mappings Γ'_i for $i = 1, 2, \ldots, n$ transforms the closed curve $\prod_{i=1}^{n} c_i$ into the closed curve $\prod_{i=1}^{n} (c'_i \cdot \overline{c_i} \cdot c_i)$ without crossing an obstacle. A simultaneous application of continuous transformations of curves $\overline{c_i} \cdot c_i$ to points *target*(c_i) for $i = 1, 2, \ldots, n$ completes the proof. □

Definition 5.11 (Path Homotopy of (s,t)-Curves)
Let E be a space with obstacles. Two curves $c_1 : [a_1, b_1] \to E$ and $c_2 : [a_2, b_2] \to E$ with $c_1(a_1) = c_2(a_2)$ and $c_1(b_1) = c_2(b_2)$ are called **(path) homotopic***, if the closed curve $c_1 \cdot \overline{c_2}$ is homotopic to the degenerate closed curve consisting of the single point $c_1(a_1)$.*

Property 5.12 (Path Homotopy is an Equivalence Relation)
Path homotopy is an equivalence relation on the set of (s,t)-curves, and the equivalence classes are called **path homotopy classes***.*

Proof:

For any (s,t)-curve $c : [a,b] \to E$, reflexivity of path homotopy is shown by the mapping

$$\Gamma(s,t) = \begin{cases} c(a + 2t(b-a) & \text{if } 2t \leq 1-s, \\ c(a + (2-2t)(b-a) & \text{if } 2-2t \leq 1-s, \text{ and} \\ c(1-s) & \text{else,} \end{cases} \tag{5.7}$$

which transforms the curve $c \cdot \overline{c}$ to the degenerate curve without crossing an obstacle.

Let c_1 and c_2 be (s,t)-curves. Curve c_1 is path homotopic to c_2, iff $c_1 \cdot \overline{c_2}$ is homotopic to the degenerate curve via a mapping Γ, which is equivalent to $\overline{c_1 \cdot \overline{c_2}} = c_2 \cdot \overline{c_1}$ being homotopic to the degenerate curve via a mapping $\Gamma'(s,t) = \Gamma(s, 1-t)$, which is the case if and only if curve c_2 is homotopic to c_1. Therefore, path homotopy is symmetric.

Let c_1, c_2, and c_3 be (s,t)-curves, curve c_1 path homotopic to c_2, and curve c_2 path homotopic to c_3. Then, the closed curve $c_1 \cdot \overline{c_2}$ is homotopic to the degenerate curve consisting of point s via mapping Γ_1, and the closed curve $c_2 \cdot \overline{c_3}$ is homotopic to the degenerate curve consisting of point s via mapping Γ_2. Mappings Γ_1 and Γ_2 can be combined to a mapping Γ_3 which transforms the closed curve $c_1 \cdot \overline{c_2} \cdot c_2 \cdot \overline{c_3}$ to the degenerate curve consisting of point s without crossing an obstacle. The closed curve $\overline{c_2} \cdot c_2$ is homotopic to the degenerate curve consisting of point t via mapping Γ_4, since path homotopy is reflexive. Therefore, the closed curve $c_1 \cdot \overline{c_3}$ can be continuously transformed to the degenerate curve consisting of point s without crossing an obstacle by subsequent application of a mapping Γ_4' which transforms curve $c_1 \cdot \overline{c_3}$ to curve $c_1 \cdot \overline{c_2} \cdot c_2 \cdot \overline{c_3}$, and Γ_3 transforming curve $c_1 \cdot \overline{c_2} \cdot c_2 \cdot \overline{c_3}$ to the degenerate curve consisting of point s. □

Property 5.13 (Path Homotopy of Composed Curves)
Let E be a space with obstacles, let c_1, c_2 and c_1', c_2' be curves in E with

(i) *source*$(c_i) =$ *source*(c_i')*, and* *target*$(c_i) =$ *target*(c_i') *for* $i \in \{1,2\}$*,*

(ii) *target*$(c_1) =$ *source*(c_2)*, and*

(iii) c_i *and* c_i' *are path homotopic for* $i \in \{1,2\}$*,*

see Figure 5.4. Then, the curves $c_1 \cdot c_2$ and $c_1' \cdot c_2'$ are path homotopic.

Proof:

We have to show that there is a continuous mapping Γ which transforms $c_1 \cdot c_2 \cdot \overline{c_2'} \cdot \overline{c_1'}$ into the degenerate closed curve (consisting of a single point) without crossing an obstacle. Since c_1 and c_1' are path homotopic, there is a continuous mapping Γ_1 which transforms the closed curve $\overline{c_1'} \cdot c_1$ into the single point *target*(c_1) without crossing an obstacle. Analogously, there is a continuous mapping Γ_2 which transforms the closed curve $c_2 \cdot \overline{c_2'}$ into the single point *source*(c_2). A combination of mappings Γ_1 and Γ_2 continuously transforms the closed curve $\overline{c_1'} \cdot c_1 \cdot c_2 \cdot \overline{c_2'}$ to the single point *target*(c_1) = *source*(c_2). Clearly, the closed curves $c_1 \cdot c_2 \cdot \overline{c_2'} \cdot \overline{c_1'}$ and $\overline{c_1'} \cdot c_1 \cdot c_2 \cdot \overline{c_2'}$ are homotopic, and the latter is homotopic to a single point. Therefore, the closed curve $c_1 \cdot c_2 \cdot \overline{c_2'} \cdot \overline{c_1'}$ is homotopic to a single point, too, and the (open) curves $c_1 \cdot c_2$ and $c_1' \cdot c_2'$ are path homotopic. □

Corollary 5.14
Let $(c_1, c_2, \ldots, c_n)$ *and* $(c_1', c_2', \ldots, c_n')$ *be sequences of curves with*

(i) *source*(c_i) = *source*(c_i'), *and* *target*(c_i) = *target*(c_i') *for* $1 \leq i \leq n$,

(ii) *target*(c_i) = *source*(c_{i+1}) *for* $1 \leq i < n$, *and*

(iii) c_i *and* c_i' *are path homotopic.*

Then, the curves $c_1 \cdot c_2 \ldots c_n$ *and* $c_1' \cdot c_2' \ldots c_n'$ *are path homotopic.*

Lemma 5.15
Let $c : [a, b] \to E$ *be a closed curve and* $p \in E$. *Then there is a closed curve* c' *with* *source*(c') = *target*(c') = p *which is homotopic to* c.

Proof:

The space E is connected, therefore there is a curve $d : [0, 1] \to E$ with *source*$(d) = p$ and *target*$(d) = c(a)$. Then, the curve $c' := d \cdot c \cdot \overline{d}$ has the claimed properties. □

Definition 5.16 (Concatenation of Homotopy Classes)
Let $[c_1]$ *and* $[c_2]$ *be homotopy classes of closed curves* $c_1 : [a, b] \to E$ *and* $c_2 : [a, b] \to E$ *with* $c_1(a) = c_2(a)$. *Then,* $[c_1 \cdot c_2]$ *is called the* **concatenation of** $[c_1]$ **and** $[c_2]$.

We have to argue that the concatenation of homotopy classes above is well-defined. Let $c_1' \in [c_1]$ with $c_1' \neq c_1$, and $c_2' \in [c_2]$ with $c_2' \neq c_2$. Then there are a continuous mapping Γ_1 which transforms c_1' to c_1 without crossing an obstacles, and a continuous mapping Γ_2 which transforms c_2' to c_2 without crossing an obstacle, such that $\Gamma_1(s, 0) = \Gamma_2(s, 0)$ = *source*(c_1) for $s \in [0, 1]$. Mappings Γ_1 and Γ_2 are now easily combined into a mapping Γ_3 which transforms the curve $c_1' \cdot c_2'$ into the curve $c_1 \cdot c_2$ without crossing an obstacle. That is, curves $c_1' \cdot c_2'$ and $c_1 \cdot c_2$ are homotopic, and $[c_1 \cdot c_2] = [c_1' \cdot c_2']$.

Property 5.17 (Fundamental Group)
The set of homotopy classes of closed curves together with the operation of concatenation form a group called the **Fundamental Group** *or* **Poincaré Group**.

The neutral element of this group is the equivalence class of all closed curves which are homotopic to the degenerate closed curve consisting of a single point $p \in E$.

The inverse element $\overline{[c]}$ *of an element* $[c]$ *of the group is the element* $[\overline{c}]$.

Verification of the group axioms is left to the reader.

Definition 5.18 (Primitive Element of Fundamental Group)
We call an element $e_i = [c_i]$ *of the Poincaré group* **primitive**, *if there is a Jordan curve* $c \in [c_i]$ *which contains exactly one obstacle, i.e. the obstacle* h_i, *in its interior.*

5.3 Distance Measures

When objects have to be compared, a simple binary decision if the objects are equal or not may not be sufficient in many cases, for example when there is noise in the data and a perfect match cannot be expected. In such cases, distance measures which quantify the degree of similarity between two given objects can be useful.

Definition 5.19 (Pseudometric)
Let M *be a set. The function* $d : M \times M \to \mathbb{R}_0^+$ *is called* **pseudometric for** M, *if*

(i) $d(x, x) = 0$,

(ii) $d(x, y) = d(y, x)$ *(symmetry), and*

(iii) $d(x, y) + d(y, z) \geq d(x, z)$ *(triangle inequality)*

hold for any $x, y, z \in M$.

Definition 5.20 (Metric)
Let M *be a set. The function* $d : M \times M \to \mathbb{R}_0^+$ *is called* **a metric for** M, *if* d *is a pseudometric for* M *and*

$$d(x, y) = 0 \quad \Rightarrow \quad x = y \tag{5.8}$$

holds for any $x, y \in M$.

Property 5.21
Any pseudometric d *for a set* M *induces an equivalence relation* $\sim_d$ *on* M *via*

$$x \sim_d y \quad :\Leftrightarrow \quad d(x, y) = 0. \tag{5.9}$$

Proof:

Reflexivity and symmetry of the relation $\sim_d$ follow from properties (i) and (ii) in Definition 5.19. Let $x, y, z \in M$ with $x \sim_d y$ and $y \sim_d z$. According to property (iii) from Definition 5.19,

$$0 \leq d(x,z) \leq d(x,y) + d(y,z) = 0$$

holds, i.e. $d(x,z) = 0$ and therefore $x \sim_d z$. □

Property 5.22
Any pseudometric d for a set M induces a metric δ on the equivalence classes of the relation $\sim_d$ via

$$\delta([x],[y]) := d(x,y). \tag{5.10}$$

Proof:

The function δ is well-defined: let $x, x', y, y' \in M$ with $x' \sim_d x$ and $y' \sim_d y$. Then,

$$\delta([x],[y]) = d(x,y) \leq d(x,x') + d(x',y') + d(y',y) = d(x',y') = \delta([x'],[y'])$$

and

$$\delta([x'],[y']) = d(x',y') \leq d(x',x) + d(x,y) + d(y,y') = d(x,y) = \delta([x],[y])$$

hold, together $\delta([x],[y]) = \delta([x'],[y'])$. Furthermore, δ displays all properties of a metric:

(i) $\delta([x],[x]) = d(x,x) = 0$

(ii) $\delta([x],[y]) = d(x,y) = d(y,x) = \delta([y],[x])$

(iii) $\delta([x],[y]) + \delta([y],[z]) = d(x,y) + d(y,z) \geq d(x,z) = \delta([x],[z])$

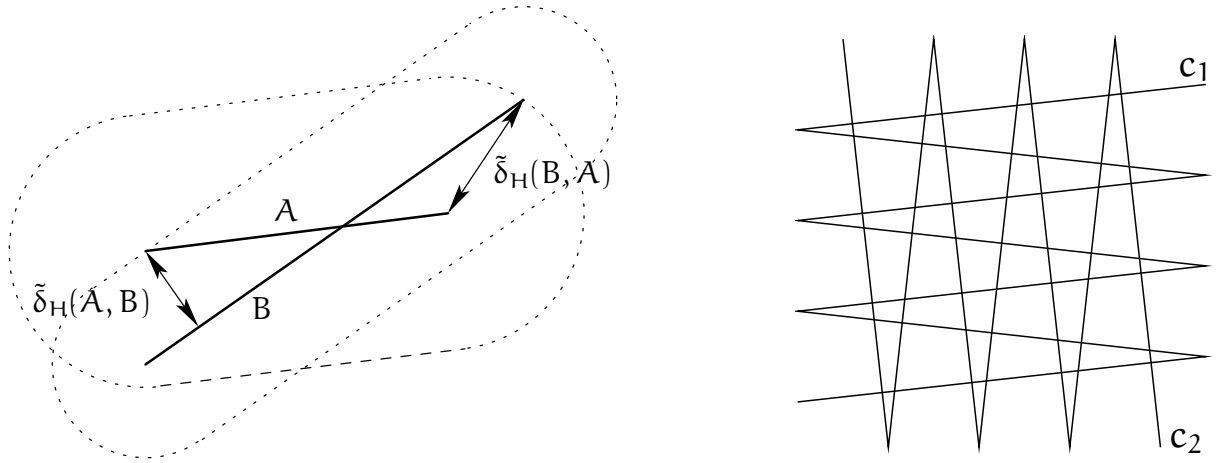
(iv) $\delta([x],[y]) = d(x,y) = 0 \quad \Rightarrow \quad x \sim_d y \quad \Rightarrow \quad [x] = [y]$

□

Distance measures can be defined on any kind of sets. In the following we will focus on distance measures for geometric objects. Two well known measures will be presented, namely the Hausdorff distance for point sets and the Fréchet distance for curves.

Furthermore, a new distance measure for curves will be introduced which considers the relationship between the curves and obstacles in their environment, i.e. topological properties of the curves.

However these distance measures are defined for arbitrary dimensional spaces, when computation of distance values is considered, we restrict ourselves to the 2-dimensional plane in the following.



(a) Hausdorff distance for line segments (b) Pathological example

Figure 5.5: Hausdorff distance examples

5.3.1 Hausdorff Distance

The Hausdorff distance is defined for point sets in Euclidean space. Informally, for two point sets A and B, their distance is the minimum distance d such that no point from A is more than distance d away from a point B and vice versa. This is stated more precisely in the following definition.

Definition 5.23 (Hausdorff Distance)
Let $A, B \subseteq \mathbb{R}^d$ *sets of points,* $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_0^+$ *the Euclidean metric. The function*

$$\tilde{\delta}_H(A, B) := \sup_{a \in A} \inf_{b \in B} d(a, b) \tag{5.11}$$

is called the **directed Hausdorff distance** *of the two point sets* A *and* B*. Furthermore, the* **(undirected) Hausdorff distance** $\delta_H(A, B)$ *of these two point sets is defined by*

$$\delta_H(A, B) := \max\left(\tilde{\delta}_H(A, B), \tilde{\delta}_H(B, A)\right) \tag{5.12}$$

The meaning of the directed Hausdorff distance is illustrated by Figure 5.5(a) for two straight line segments. It should be kept in mind that there are cases where the Hausdorff distance is not suitable to compare curves. One such case is can be seen in Figure 5.5(b), where the distance of the two polygonal chains is small but their similarity is low.

5.3.2 Computation of the Hausdorff Distance

The significant step in the computation of the directed Hausdorff distance of two point sets A and B is the search for a point $q \in B$ with minimum distance to a given point $p \in A$, i.e. this is the spot where naive approaches will differ from sophisticated algorithms. Clearly, efficient implementations will depend on the structure of the point sets A and B.

An efficient algorithm for computing the Hausdorff distance of two simple polygons is presented by Alt et al. (1995). Their approach can be generalized to simple polygonal chains easily.

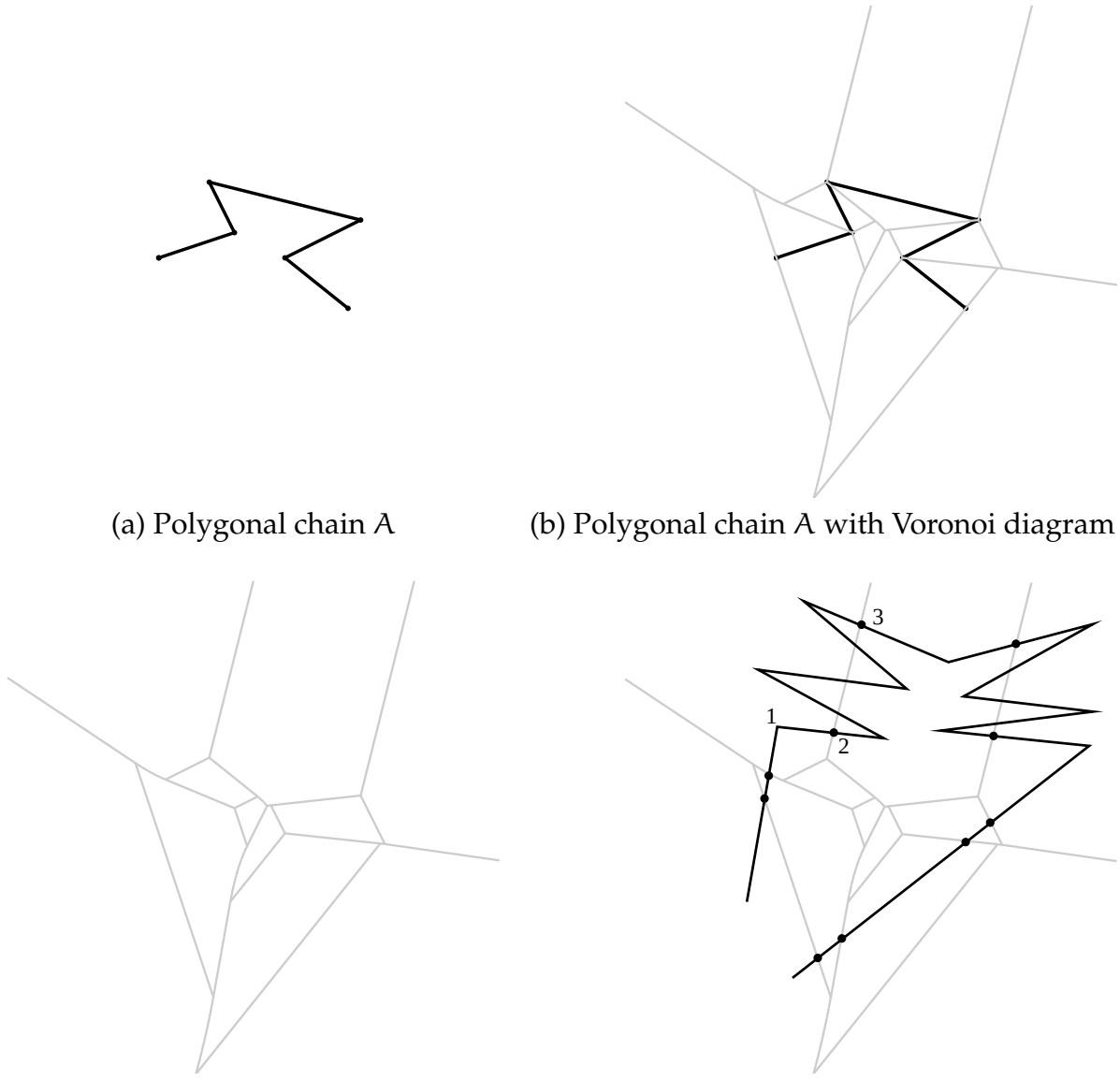
A key to the efficiency of this algorithm are Voronoi diagrams of polygonal chains. The Voronoi diagram $Vor(A)$ of a set of geometric objects A (called *sites* in this context) assigns to each site $a \in A$ its *Voronoi cell*, i.e. the set of points which are closer to a site a than to any other site $a' \in A$. Voronoi cells are bounded by *Voronoi edges*. In the case of polygonal chains, relevant sites are the edges and vertices of the chains, and the Voronoi edges are either line segments (if they separate the cells of two edges or two vertices of the polygonal chain) or parabolic segments (if they separate the cell of an edge from the cell of a vertex of the polygonal chain). The (zero-dimensional) locations where Voronoi edges meet are called *Voronoi vertices*, and they represent center points of empty circles with locally maximum radius (i.e. infinitesimal displacement from this point will decrease the maximum radius of an empty circle). Fortune (1987) gives more details on this data structure and an optimal algorithm for its computation.

In two dimensions, the structure of Voronoi edges and vertices is a planar graph. Therefore, a voronoi diagram for $|A|$ sites will consist of $\mathcal{O}(|A|)$ faces, edges, and vertices. Furthermore, the Voronoi diagram of a set A of straight line segments or circular arcs can be computed in time $\mathcal{O}(|A| \log |A|)$ which is shown by Fortune (1987) for straight lines and by Yap (1987) for circular arcs. As an example, Figures 5.6(a) and (b) show a polygonal chain together with its Voronoi diagram.

Given two polygonal chains A and B as well as the Voronoi diagram $Vor(A)$ of A, the directed Hausdorff distance $\tilde{\delta}_H(B, A)$ can be computed efficiently as following, according to Alt et al. (1995).

Consider the intersection of an edge b from B with a cell C from $Vor(A)$. When moving monotonically on this part of the edge b, the distance to the corresponding site a from A defining cell C is a convex function. Therefore, the maximum distance is assumed at an endpoint of this part of b, i.e. at a vertex of B or at an intersection of b with a Voronoi edge (see for example the points marked as '1' and '2' in Figure 5.6(d)).

On the other hand, when moving monotonically on a Voronoi edge e of $Vor(A)$, the distance to the sites whose cells are bounded by e is a convex function, too. As a consequence, for each Voronoi edge e of $Vor(A)$ only its extremal intersections with sites from B need to be considered (see for example the points marked as '2' and '3' in Figure 5.6(d)).



(a) Polygonal chain A

(b) Polygonal chain A with Voronoi diagram $Vor(A)$

(c) Voronoi diagram $Vor(A)$

(d) Voronoi diagram $Vor(A)$ and polygonal chain B

Figure 5.6: Hausdorff distance of polygonal chains

Therefore, in order to compute the directed Hausdorff distance $\tilde{\delta}_H(B, A)$, the number of points for which the distance to A needs to be considered is only $\mathcal{O}(|A| + |B|)$.

The extremal intersection points of Voronoi edges e from $Vor(A)$ with edges b from B can be found by two subsequent plane sweeps in opposite directions, whereby in each sweep Voronoi edges are removed from the event and sweep structure as soon as a first intersection is detected. Since there are $\mathcal{O}(|A| + |B|)$ event points, each sweep can be accomplished in time $\mathcal{O}((|A| + |B|) \log(|A| + |B|))$. During the sweep, the cells of $Vor(A)$ containing the event points are known, and therefore the distances to the respective sites can be computed in constant time. Note that the time required for constructing the Voronoi diagram is dominated by the time required for the final plane sweeps.

As a result, the Hausdorff distance $\delta_H(A, B)$ of two polygonal chains can be computed in time $\mathcal{O}((|A|+|B|) \log(|A|+|B|))$, too, by computing the maximum of $\tilde{\delta}_H(A, B)$ and $\tilde{\delta}_H(B, A)$.

5.3.3 Fréchet Distance

As we have seen in the example of Figure 5.5(b), the Hausdorff distance is not always suitable for the comparison of curves. These problems are avoided by the Fréchet distance, which is defined for curves only.

Definition 5.24 (Fréchet Distance)
Let $c_1 : [a_1, b_1] \to \mathbb{R}^d$ and $c_2 : [a_2, b_2] \to \mathbb{R}^d$ be curves in $\mathbb{R}^d$, let $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_0^+$ be the Euclidean metric. Then, the function

$$\delta_F(c_1, c_2) := \inf_{\alpha,\beta} \max_{t\in[0,1]} d\left(c_1(\alpha(t)), c_2(\beta(t))\right) \tag{5.13}$$

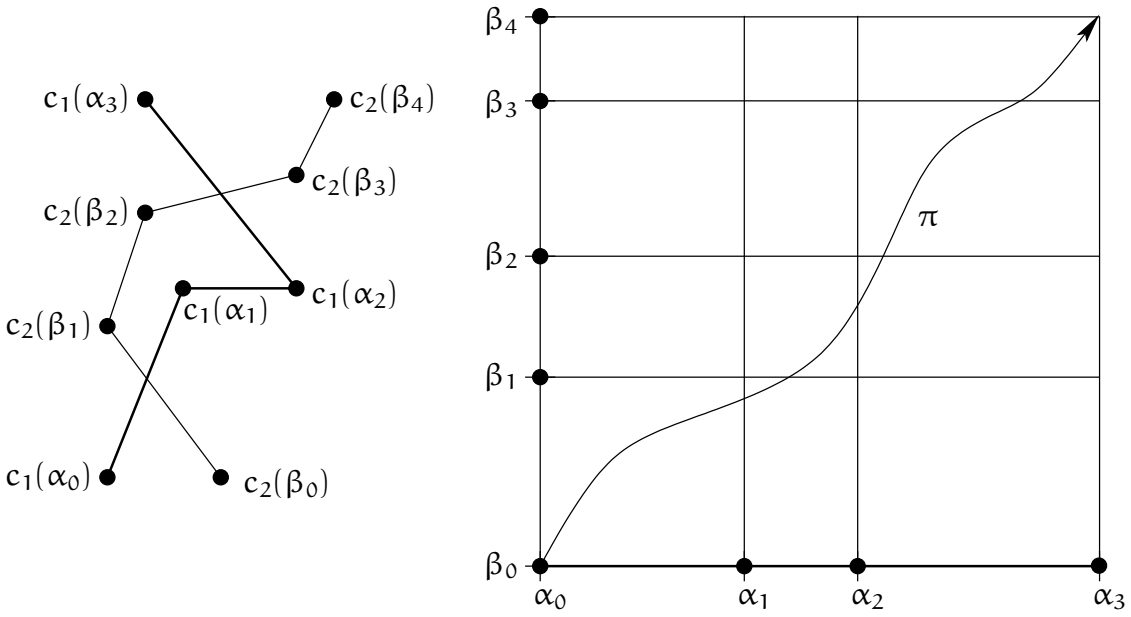
where permissible functions $\alpha : [0, 1] \to [a_1, b_1]$ and $\beta : [0, 1] \to [a_2, b_2]$ are continuous, surjective, and monotonously increasing, is called **Fréchet metric**,

An intuitive description of the Fréchet distance of two curves can be given as follows. Consider a dog and its master on a walk, where the master moves monotonically along c_1 and the dogs moves monotonically along c_2. Then, the Fréchet distance of c_1 and c_2 is the minimum length of the leash which still allows such a walk.

5.3.4 Computation of the Fréchet Distance

An algorithm for computing the Fréchet distance of polygonal chains has been proposed by Alt and Guibas (1996), and will be presented in the following.

Let $c_1 : [\alpha_0, \alpha_m] \to \mathbb{R}^2$ and $c_2 : [\beta_0, \beta_n] \to \mathbb{R}^2$ polygonal chains, without loss of generality with linear parameterization (that is, the Euclidean arc length of the curve from $c(t)$ to $c(t + s)$ is s for any t and s), c_1 linear on each interval $[\alpha_i, \alpha_{i+1}]$ for $0 \le i < m$, and c_2 linear on each interval $[\beta_j, \beta_{j+1}]$ for $0 \le j < n$.



(a) Two polygonal chains (b) Combined parameterization

Figure 5.7: Computation of Fréchet distance of polygonal chains

As a first step, an algorithm is developed which decides if the Fréchet distance of two given polygonal chains is smaller than a given value δ. Then, in a second step, binary or parametric search is applied to incrementally compute the actual distance value.

So in the first step, a distance value δ is given, and the problem is to decide whether the Fréchet distance of the given pair of polygonal chains c_1 and c_2 is not greater than δ (see for example Figure 5.7(a)). That is, we have to decide if there is a curve $\pi : [0,1] \to [\alpha_0, \alpha_m] \times [\beta_0, \beta_n]$ from (α_0, β_0) to (α_m, β_n) with $\pi(t) = (\alpha(t), \beta(t))$, and α and β monotonous as in the definition of the Fréchet distance, such that $d(c_1(\alpha(t)), c_2(\beta(t))) \leq \delta$ for any $t \in [0,1]$ (cf. Figure 5.7(b)).

In the case of polygonal chains, we can partition the set $[\alpha_0, \alpha_m] \times [\beta_0, \beta_n]$ into rectangles $R_{ij} = [\alpha_i, \alpha_{i+1}) \times [\beta_j, \beta_{j+1})$ where each rectangle R_{ij} confers to a specific pair of segments from the polygonal chains. Depending on the supporting lines l_1 and l_2 of the segments $c_1|_{[\alpha_i, \alpha_{i+1}]}$ and $c_2|_{[\beta_j, \beta_{j+1}]}$, the set $\hat{R}_{ij} = \{(\alpha, \beta) \in R_{ij} \mid d(c_1(\alpha), c_2(\beta)) \leq \delta\}$ is the intersection of the rectangle R_{ij} with either

(i) the empty set, if l_1 and l_2 are parallel with distance greater than δ,

(ii) a strip of constant width, if l_1 and l_2 are parallel and their distance is not greater than δ, or

(iii) a circle, if l_1 and l_2 intersect in one point.

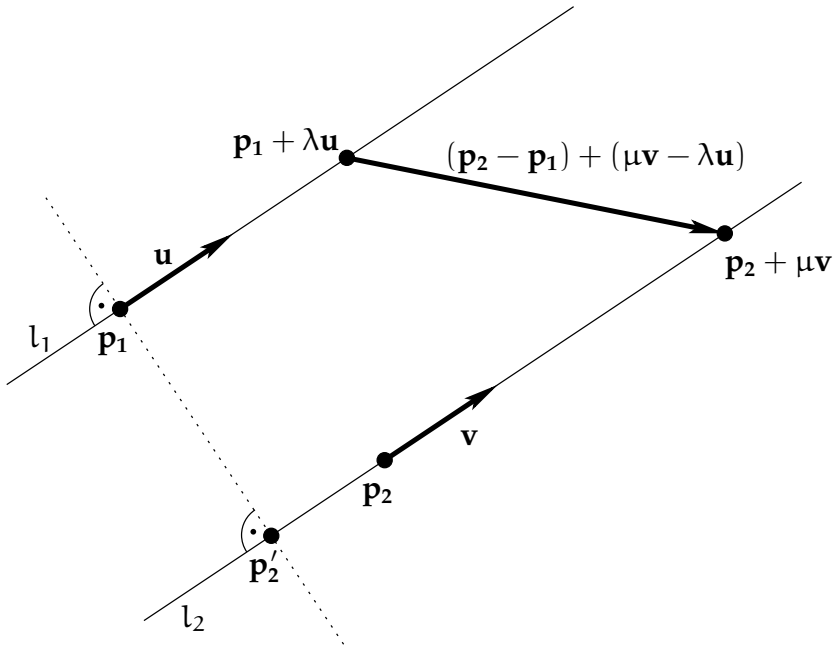


Figure 5.8: Fréchet distance, parallel segment pair

Clearly, only points from the union of the sets $\hat{R}_{ij}$ ($0 \leq i < m, 0 \leq j < n$) may be traversed by a curve π for the combined parameterization of c_1 and c_2 when we want to show $\delta_F(c_1, c_2) \leq \delta$.

We will now examine the three cases for the relative pose of the supporting lines l_1 and l_2. At first, assume that l_1 and l_2 are parallel and given as $l_1 : \lambda \mapsto \mathbf{p_1} + \lambda\mathbf{u}$, $l_2 : \mu \mapsto \mathbf{p_2} + \mu\mathbf{v}$, $|\mathbf{u}| = |\mathbf{v}| = 1$, $\mathbf{v} = k_1\mathbf{u}$ with $k_1 \in \{-1, 1\}$, and $\mathbf{p_2'} = \mathbf{p_2} - k_2\mathbf{v}$ such that $(\mathbf{p_2'} - \mathbf{p_1})$ and $\mathbf{u}$ are orthogonal (cf. Figure 5.8). Obviously, case (i) is trivial. Therefore, assume the distance $d = |\mathbf{p_2'} - \mathbf{p_1}|$ of the parallel lines is not greater than δ. We have

$$
\begin{aligned}
& ((\mathbf{p_2} + \mu\mathbf{v}) - (\mathbf{p_1} + \lambda\mathbf{u}))^2 \leq \delta^2 && (5.14)\\
\Leftrightarrow\quad & ((\mathbf{p_2'} - \mathbf{p_1}) + ((k_2 + \mu)k_1 - \lambda)\mathbf{u})^2 \leq \delta^2 && (5.15)\\
\Leftrightarrow\quad & d^2 + 2((k_2 + \mu)k_1 - \lambda)(\mathbf{p_2'} - \mathbf{p_1})\mathbf{u} + ((k_2 + \mu)k_1 - \lambda)^2\mathbf{u}^2 \leq \delta^2 && (5.16)\\
\Leftrightarrow\quad & (k_2 + \mu)k_1 - \lambda)^2 \leq \delta^2 - d^2 && (5.17)\\
\Leftrightarrow\quad & \mu \in \left[(\lambda/k_1 - k_2) - \sqrt{\delta^2 - d^2}, (\lambda/k_1 - k_2) + \sqrt{\delta^2 - d^2}\right] && (5.18)\\
\Leftrightarrow\quad & \mu \in \left[(k_1\lambda - k_2) - \sqrt{\delta^2 - d^2}, (k_1\lambda - k_2) + \sqrt{\delta^2 - d^2}\right]. && (5.19)
\end{aligned}
$$

Figure 5.9 illustrates this result. The strip is centered around a line $\mu = k_1\lambda - k_2$ with $k_1 \in \{-1, 1\}$, and its width is $\sqrt{\frac{\delta^2 - d^2}{2}}$ orthogonal to this line.

Now, assume that l_1 and l_2 intersect at point $\mathbf{p_0}$ and (without loss of generality) are

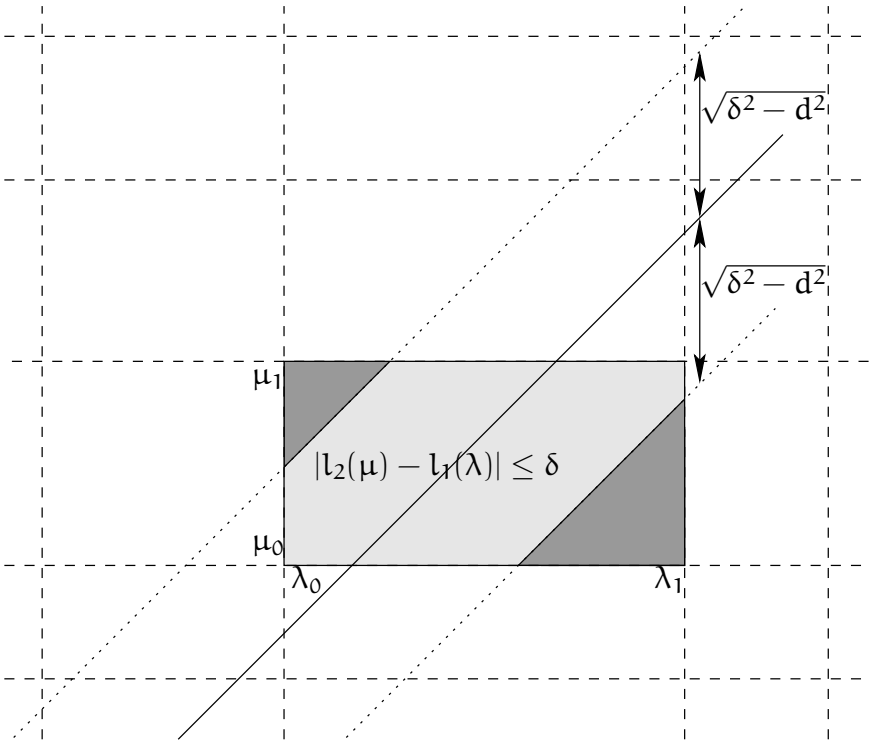


Figure 5.9: Fréchet distance, allowed parameters $\hat{R}$ for parallel lines

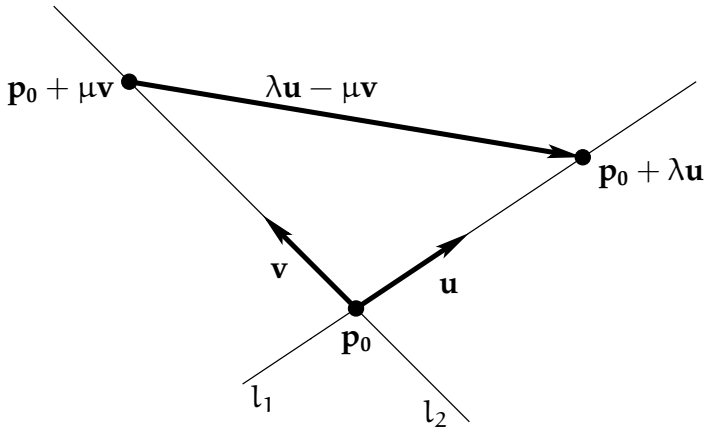


Figure 5.10: Fréchet distance, intersecting segment pair

given as $l_1 : \lambda \mapsto \mathbf{p_0} + \lambda\mathbf{u}$, $l_2 : \mu \mapsto \mathbf{p_0} + \mu\mathbf{v}$, and $|\mathbf{u}| = |\mathbf{v}| = 1$ (cf. Figure 5.10). We examine the set of allowed parameters as follows:

$$(\lambda\mathbf{u} - \mu\mathbf{v})^2 \leq \delta^2 \tag{5.20}$$

$$\Leftrightarrow \quad \lambda^2 + \mu^2 \leq \delta^2 + 2\lambda\mu(\mathbf{uv}). \tag{5.21}$$

With the substitution $r := \mu + \lambda$ and $s := \mu - \lambda$, this equation becomes

$$(r - s)^2 + (r + 2)^2 \leq 4\delta^2 + 2(r - s)(r + s)(\mathbf{uv}) \tag{5.22}$$

$$\Leftrightarrow \quad 2r^2 + 2s^2 \leq 4\delta^2 + 2r^2(\mathbf{uv}) + 2s^2(\mathbf{uv}) \tag{5.23}$$

$$\Leftrightarrow \quad r^2 + s^2 \leq \frac{2\delta^2}{1 - \mathbf{uv}}. \tag{5.24}$$

Reversing the substitution, we get

$$r^2 + s^2 = 2(\lambda^2 + \mu^2) \leq \frac{2\delta^2}{1 - \mathbf{uv}} \tag{5.25}$$

$$\Leftrightarrow \quad \lambda^2 + \mu^2 \leq \frac{\delta^2}{1 - \mathbf{uv}} \tag{5.26}$$

This result indicates that the set of allowed parameters is bounded by a circle with radius $\frac{\delta}{\sqrt{1-\mathbf{uv}}}$ and centered at $(\lambda, \mu) = (0, 0)$.

Figure 5.11 illustrates this result. The position of the center of the circle depends on the actual parameterization of the line segments. That is, if they do not intersect in $l_1(0) = l_2(0)$, the center of the circle is moved accordingly.

As each set $\hat{R}_{ij}$ is the intersection of a rectangle and a convex set, it is convex, too. Since the components α and β of the curve π which expresses the combined parameterization are monotonously increasing, each rectangle R_{ij} can only be entered by π on the left or bottom side, and can only be left by π on the right or top side. As $\hat{R}_{ij}$ is convex, the sets of allowed entrance and exit points on each side of the rectangle are single, connected segments. We will call the sets of allowed parameter pairs on the left and bottom edges the *entrance windows* of the cell and the allowed parameter pairs on the right and top edge the *exit windows* of the cell.

In order to decide if there is a curve π from $(0, 0)$ to (α_m, β_n) which increases monotonically in both components, we have to consider the monotonically reachable parts of entrance and exit windows of each cell (see Figure 5.12).

If a cell has a left entrance window, top exit windows are entirely monotonically reachable (cf. Figure 5.12(a)). If a cell has a bottom entrance window, right exit windows are entirely monotonically reachable (cf. Figure 5.12(b)). If a cell has a left entrance window, but no bottom entrance window, right exit windows are not monotonically reachable below the lower border of the entrance window (cf. Figure 5.12(c)). If a cell has a bottom entrance window, but no left entrance window, top exit windows are not monotonically reachable to the left of the left border of the entrance window (cf. Figure 5.12(d)).

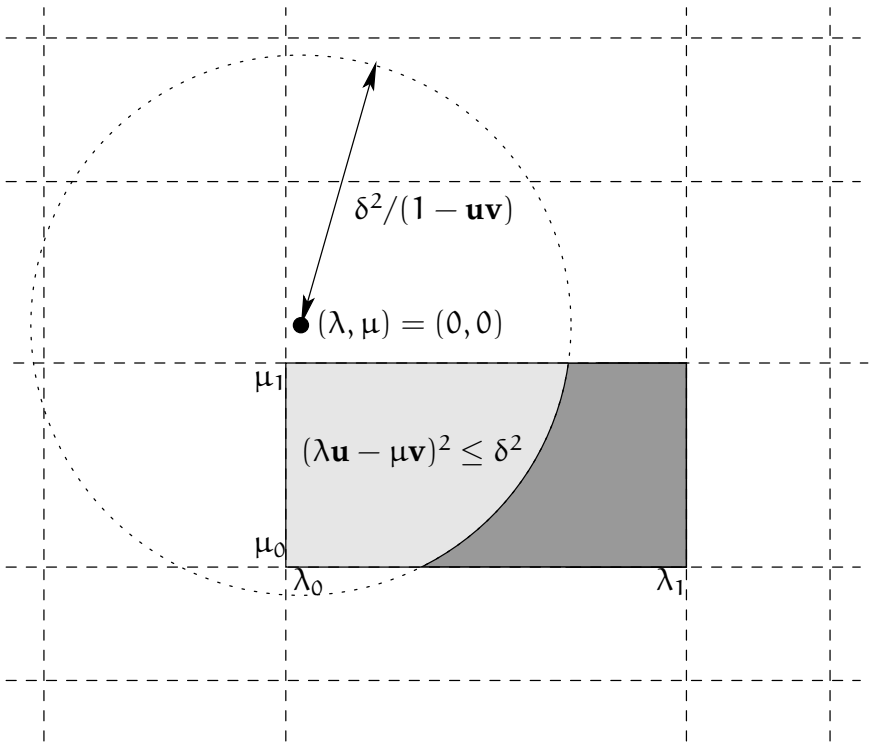
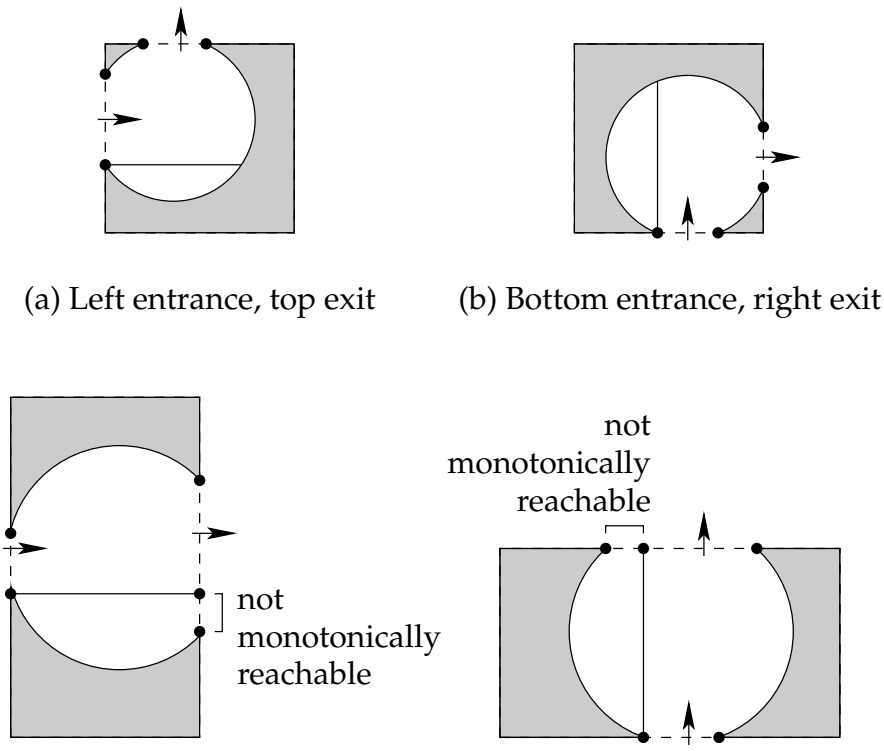


Figure 5.11: Fréchet distance, allowed parameters $\hat{\mathsf{R}}$ for intersecting lines



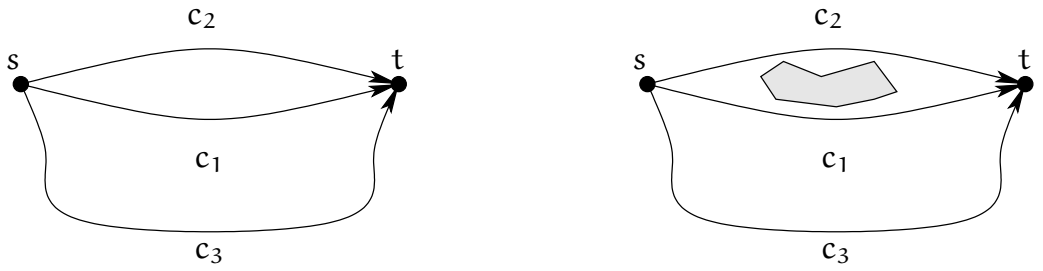
(a) Left entrance, top exit

(b) Bottom entrance, right exit

(c) Left entrance, right exit

(d) Bottom entrance, top exit

Figure 5.12: Fréchet distance, monotonic reachability



(a) c_2 appears "more similar" to c_1 than c_3 (b) c_3 appears "more similar" to c_1 than c_2

Figure 5.13: Motivating example of topological distance

Furthermore, the monotonically reachable right exit window of cell (i, j) is the (possibly restricted) left entrance window of cell $(i+1, j)$, and the monotonically reachable top exit window of cell (i, j) is the (possibly restricted) bottom entrance window of cell $(i, j+1)$.

The monotonically reachable windows can be computed incrementally in constant time per cell, and a curve π representing an allowed combined parameterization exists if $(0, 0)$ and (α_m, β_n) are allowed parameter pairs, and the cell $(m-1, n-1)$ has a monotonically reachable entrance window. This yields an algorithms which decides $\delta_F(c_1, c_2) \leq \delta$ in time $\mathcal{O}(mn)$.

In order to actually compute the distance value, Alt and Guibas (1996) claim using a variant of parametric search yields an algorithm of running time $\mathcal{O}(mn \log(mn))$. However, it might appear equally practical to compute the distance value bit by bit, applying the decision algorithm once per step.

5.3.5 Topological Distance

Another approach to similarity of paths might not only consider geometric properties of the paths themselves, but may also take the relationship to external obstacles into account.

For a motivating example, see Figure 5.13. Sub-figure (a) shows three (s, t)-curves in a space without an obstacle. One could show that the Hausdorff and Fréchet distances of c_1 and c_2 are smaller than the respective distances between c_1 and c_3. Sub-figure (b) displays the same three (s, t)-curves, but now in a space with one obstacle. However the facts about Hausdorff and Fréchet distances of the curves still hold as above, one could argue that c_1 is now more similar to c_3 than to c_2 since it passes on the same side of the obstacle.

In other words, c_1 and c_3 are from the same homotopy classes, and c_2 is from a different homotopy class of (s, t)-curves. Therefore, any continuous mapping $\Gamma : [0, 1] \times [0, 1] \to E$ with the properties (i) to (iii) from Definition 5.8 which transforms $c_1 \cdot \overline{c_2}$ to a degener-

ate closed curve consisting of a single point will touch at least one obstacle. We will define the topological distance of two (s,t)-curves to be the minimum number of obstacle crossings which are caused by such a continuous mapping Γ. Note that the result of that definition will be a distance measure for the set of homotopy classes of (s,t)-curves, since in general $d(c_1,c_2)=0$ will not imply $c_1=c_2$ but only means that c_1 and c_2 are homotopic.

Definition 5.25 (Topological Norm)
Let c be a closed curve in E. Let $\mathcal{M}$ be the set of continuous mappings $\Gamma:[0,1]\times[0,1]\to E$ with

(i) $\Gamma(0,t)=c(t)$ *for* $t\in[0,1]$,

(ii) $\Gamma(1,t)=c(0)$ *for* $t\in[0,1]$, *and*

(iii) $\Gamma(s,0)=\Gamma(s,1)$ *for any* $s\in[0,1]$.

Let $cc(M)$ denote the set of connected components of a metric space M. Then the number

$$|c|_T := \min_{\Gamma\in\mathcal{M}} \left| cc\left(\{(s,t)\in[0,1]^2 \,|\, \Gamma(s,t)\notin E\}\right)\right|, \tag{5.27}$$

is called the **topological norm** *of* c.

For an illustrating example to that definition, see Figure 5.14, where a space with two obstacles and a closed curve which loops once around the first obstacle and twice around the second obstacle are considered. The effect of a continuous mapping Γ is shown for some values of s as well as the set of pairs (s,t) where Γ crosses an obstacle, i.e. where $\Gamma(t,s)\notin E$. Apparently there are three connected components in the set of these pairs, so the topological norm of the closed curve in that example is not greater than 3.

Property 5.26
For any closed curve c, $|c|_T=|\overline{c}|_T$ holds.

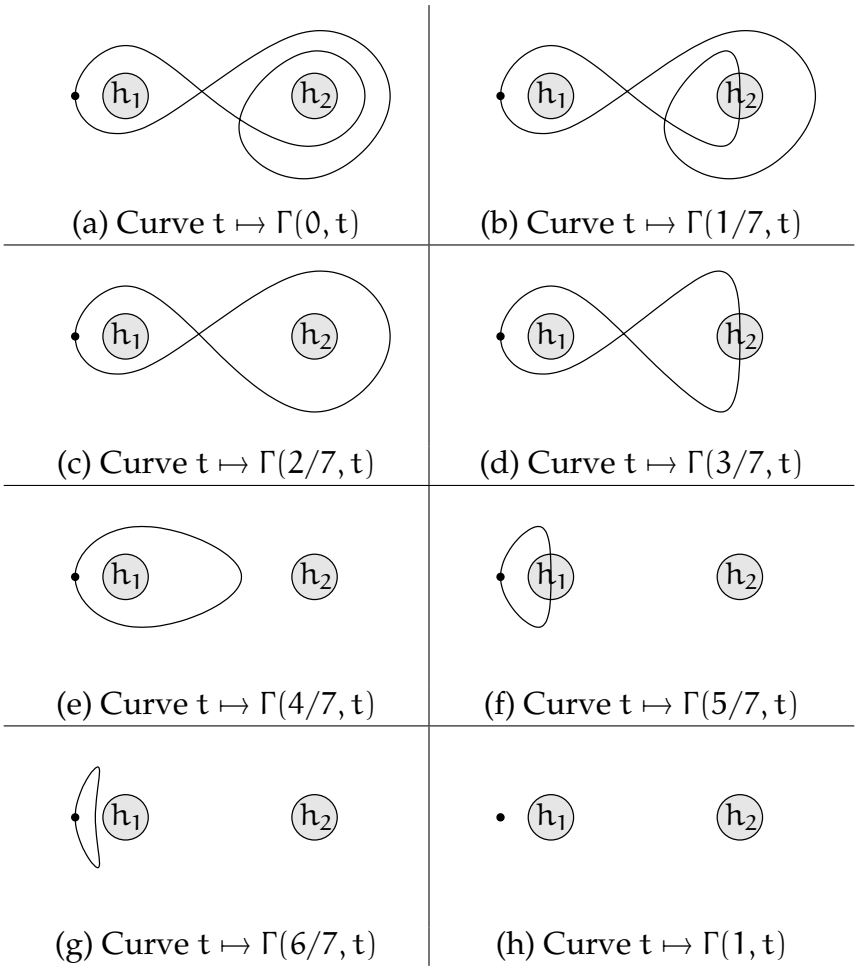
Proof:

Let Γ be a mapping which transforms c into the degenerate curve crossing obstacles $|c|_T$ times. Then,

$$\Gamma' : (s,t)\mapsto\Gamma(s,1-t)$$

transforms $\overline{c}$ into the degenerate curve crossing obstacles $|c|_T$ times, i.e. $|\overline{c}|_T\le|c|_T$. Similarly one can show $|\overline{\overline{c}}|_T\le|\overline{c}|_T$. Since $\overline{\overline{c}}=c$, we have $|c|_T=|\overline{c}|_T$. □

Property 5.27
Let c_1 and c_2 be closed curves in E. If c_1 and c_2 are homotopic, then $|c_1|_T=|c_2|_T$.

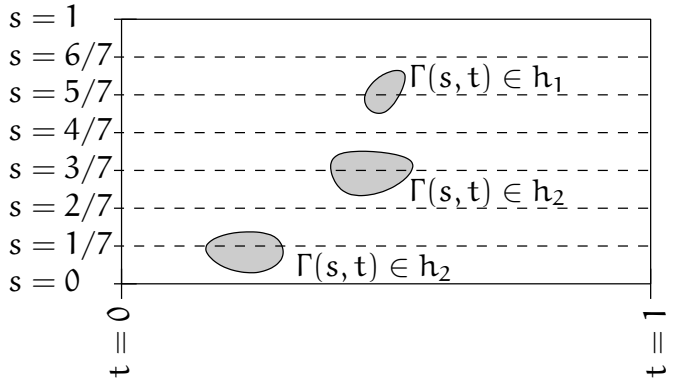


(a) Curve $t \mapsto \Gamma(0, t)$ (b) Curve $t \mapsto \Gamma(1/7, t)$

(c) Curve $t \mapsto \Gamma(2/7, t)$ (d) Curve $t \mapsto \Gamma(3/7, t)$

(e) Curve $t \mapsto \Gamma(4/7, t)$ (f) Curve $t \mapsto \Gamma(5/7, t)$

(g) Curve $t \mapsto \Gamma(6/7, t)$ (h) Curve $t \mapsto \Gamma(1, t)$



(i) Set of obstacle intersections $\{(s, t) \mid \Gamma(s, t) \notin E\}$

Figure 5.14: Definition of topological norm

Proof:

Let Γ_1 be a mapping which proves homotopy of c_1 to c_2, let Γ_2 be a mapping which transforms c_2 to the degenerate closed curve according to Definition 5.25 and crossing obstacles $|c_2|_T$ times during this transformation. Now define a continuous mapping Γ_3 as

$$\Gamma_3(s,t) = \begin{cases} \Gamma_1(2s,t) & \text{if } s \leq 1/2, \\ \Gamma_2(2s-1,t) & \text{else.} \end{cases}$$

Then, Γ_3 transforms c_1 to the degenerate closed curve, crossing obstacles $|c_2|_T$ times during this transformation, i.e. $|c_1|_T \leq |c_2|_T$. Repeating this construction with exchanged roles of c_1 and c_2 yields $|c_2|_T \leq |c_1|_T$. Combined we have $|c_2|_T = |c_1|_T$. □

Property 5.28
Let c_1 and c_2 be closed curves in E with source(c_1) = source(c_2). Then, the triangle inequality

$$|c_1 \cdot c_2|_T \leq |c_1|_T + |c_2|_T \tag{5.28}$$

holds.

Proof:

Let Γ_1 a mapping which transforms c_1 to the degenerate closed curve according to Definition 5.25 and crossing obstacles $|c_1|_T$ times during this transformation. Let Γ_2 a mapping which transforms c_2 to the degenerate closed curve according to Definition 5.25 and crossing obstacles $|c_2|_T$ times during this transformation. Now define a continuous mapping Γ_3 as

$$\Gamma_3(s,t) = \begin{cases} \Gamma_1(s,2t) & \text{if } t \leq 1/2, \\ \Gamma_2(s,2t-1) & \text{else.} \end{cases}$$

Then, Γ_3 transforms the curve $c_1 \cdot c_2$ to the degenerate closed curve and crosses obstacles at most $|c_1|_T + |c_2|_T$ times during this transformation. Mapping Γ_3 proves that $|c_1 \cdot c_2|_T$ cannot be greater than $|c_1|_T + |c_2|_T$. □

Definition 5.29 (Topological Distance)
Let c_1 and c_2 be (s,t)-curves in E. Then the number

$$\delta_T(c_1, c_2) := |c_1 \cdot \overline{c_2}|_T \tag{5.29}$$

is called the **topological distance** *of c_1 and c_2.*

Property 5.30
Let c_1 and c_2 be (s,t)-curves. Then, $\delta_T(c_1, c_2) = 0$ if and only if c_1 and c_2 are path homotopic.

Proof:

Assume $\delta_T(c_1, c_2) = 0$. Then $|c_1 \cdot \overline{c_2}|_T = 0$ which means that $c_1 \cdot \overline{c_2}$ can be transformed into the degenerate curve without touching an obstacle. This matches the definition of homotopy among (s,t)-curves, therefore c_1 and c_2 are homotopic.

Now assume c_1 and c_2 are homotopic, i.e. there is a mapping Γ which transforms the closed curve $c_1 \cdot \overline{c_2}$ into the degenerate closed curve without touching any obstacle. This mapping Γ proves $|c_1 \cdot \overline{c_2}|_T = 0$. □

Property 5.31
The topological distance δ_T is a pseudo-metric on the set of (s,t)-curves.

Proof:

We verify the properties of a pseudometric. The distance of a curve to itself is always zero,

$$\delta_T(c_1, c_1) = |c_1 \cdot \overline{c_1}|_T = 0, \tag{5.30}$$

since path homotopy is reflexive.

Symmetry follows mainly from the facts that $|c|_T = |\overline{c}|_T$ and that $\overline{c_1 \cdot c_2}$ and $\overline{c_2} \cdot \overline{c_1}$ are homotopic (in fact equal up to a shift in parameterization). Therefore,

$$\delta_T(c_1, c_2) = |c_1 \cdot \overline{c_2}|_T = |\overline{c_1 \cdot \overline{c_2}}|_T = |\overline{\overline{c_2}} \cdot \overline{c_1}|_T = |c_2 \cdot \overline{c_1}|_T = \delta_T(c_2, c_1) \tag{5.31}$$

holds.

The triangle inequality is shown as follows,

$$\begin{aligned}
\delta_T(c_1, c_2) + \delta_T(c_2, c_3) &= |c_1 \cdot \overline{c_2}|_T + |c_2 \cdot \overline{c_3}|_T \\
&\geq |c_1 \cdot \overline{c_2} \cdot c_2 \cdot \overline{c_3}|_T \\
&= |\overline{c_2} \cdot c_2 \cdot \overline{c_3} \cdot c_1|_T \\
&= |\overline{c_3} \cdot c_1|_T \\
&= |c_1 \cdot \overline{c_3}|_T \\
&= \delta_T(c_1, c_3),
\end{aligned}$$

using the triangle inequality for the norm, and successively replacing the argument curve of the norm by a homotopic curve. □

Property 5.32
On the set of (s,t)-curves in a space with obstacles, the equivalence relation $\sim_{\delta_T}$ induced by the pseudometric δ_T is identical to the homotopy relation.

Proof:

Let c_1 and c_2 be (s,t)-curves. At first, assume c_1 and c_2 are homotopic. Then, $\delta_T(c_1,c_2) = |c_1 \cdot \overline{c_2}|_T = 0$ holds, which implies $c_1 \sim_{\delta_T} c_2$.

Now assume $c_1 \sim_{\delta_T} c_2$, i.e. $\delta_T(c_1,c_2) = 0$. This implies $|c_1 \cdot \overline{c_2}|_T = 0$ which is equivalent to c_1 and c_2 being homotopic. □

Corollary 5.33
Let E be an environment with obstacles. Then, the topological distance δ_T on the set of (s,t)-curves in E induces a metric on the set of homotopy classes of (s,t)-curves.

5.3.6 Computation of Topological Distances

As before, we will restrict ourselves to two dimensional spaces when considering the computation of a distance function. Furthermore, unbounded obstacles are irrelevant for the topological norm or distance, since they cannot be surrounded by a closed curve. Accordingly, we will also restrict ourselves to bounded obstacles.

The road-map for the computation of topological norms (and thereby distances) is as follows. First, a method to capture the structure of the space with obstacle in an embedded planar graph G is presented. Second, a closed curve c (for which the topological norm is to be computed) is transformed to a homotopic embedded graph cycle s in G. Next, a tree G_T in G is fixed, and from the sequence of non-tree arcs in s, a representation for the homotopy class $[c]$ by means of a composition of primitive elements from the fundamental group is deduced. Finally, this element of the fundamental group is transformed to the neutral element using a minimum number of primitive element deletions.

Skeleton of a Space with Obstacles

In order to process homotopy classes of curves on a computer, we need some kind of discrete representation of these objects. We will use embedded planar graphs to capture the structure of the space with obstacles. This allows us to map homotopy classes of closed curves to cycles in embedded planar graphs.

Definition 5.34 (Planar Embedding of a Planar Graph)
Let $G = (V,A)$ be a simple directed planar graph. A **planar embedding of** G *is a pair (ϕ,ψ) of an injective function $\phi : V \to \mathbb{R}^2$ and a function $\psi : A \times [0,1] \to \mathbb{R}^2$, where each vertex $v \in V$ is embedded as point $\phi(v)$, and each arc $a \in A \subseteq V \times V$ is embedded as a simple curve $c_a : [0,1] \to \mathbb{R}^2$ with*

(i) $c_a : t \mapsto \psi(a,t)$,

(ii) $\psi(a,0) = \phi(\alpha(a))$,

(iii) $\psi(a, 1) = \phi(\omega(a))$*, and*

(iv) $\psi(a_1, [0, 1]) \cap \psi(a_2, (0, 1)) = \emptyset$ *for any* $a_1, a_2 \in A$ *with* $a_1 \neq a_2$*.*

Furthermore, the connected components of $\mathbb{R}^2 - \psi(A, [0, 1])$ *are called the* **faces of the embedded planar graph**.

Given a chain in an embedded planar graph, a plane curve is induced in an intuitive manner by concatenating the embedded curves of the traversed arcs. This idea is specified more precisely as follows.

Definition 5.35 (Curve of a Chain)
Let $G = (V, A)$ *be a simple directed planar graph with embedding* (ϕ, ψ)*. Let*

$$s = (\delta_1 a_1, \delta_2 a_2, \ldots, \delta_n a_n)$$

be a chain in G*. Then, a* **curve** $c(s)$ **is induced by** s *via*

$$c(s) = c(\delta_1 a_1) \cdot c(\delta_2 a_2) \cdot \cdots \cdot c(\delta_n a_n) \tag{5.32}$$

where

$$c(\delta a) = \begin{cases} c_a & \text{if } \delta = '+', \\ \overline{c_a} & \text{else.} \end{cases} \tag{5.33}$$

Note that the definition above maps graph cycles to closed curves as well. Furthermore, by mapping chains in a skeleton to plane curves, we may talk of homotopy and path homotopy among cycles and chains in a skeleton, referring to their embedded curves.

Property 5.36
Let $G = (V, A)$ *be a simple directed planar graph with embedding* (ϕ, ψ)*. For any elementary chain* s *in* G*, the induced curve* $c(s)$ *is simple.*

Proof:

Let $s = (\delta_1 a_1, \delta_2 a_2, \ldots, \delta_n a_n)$ be an elementary chain in G. Assume $c(s)$ is not simple, i.e. there are $i, j \in \{1, 2, \ldots, n\}$ such that $c_{a_i}(t_1) = c_{a_j}(t_2)$ for some $t_1, t_2 \in [0, 1]$. Since arcs are embedded as simple curves, $i \neq j$ holds. Furthermore, since s is elementary, any vertex of s is traversed only once, and since ϕ is injective, each vertex is embedded at a different point in the plane. Therefore, the intersection cannot occur at a vertex. But by definition, the curves which embed the arcs may intersect each other only in their endpoints. This is a contradiction, so $c(s)$ must have been simple. □

Corollary 5.37 (Elementary Cycles Induce Simple Closed Curves)
Let $G = (V, A)$ *be a simple directed planar graph with embedding* (ϕ, ψ)*. For any elementary cycle* s *in* G*, the induced curve* $c(s)$ *is closed and simple.*

According to the Jordan Curve Theorem, we may talk of the (bounded) interior and the (unbounded) exterior of embedded elementary cycles.

Definition 5.38 (Face Cycles of an Embedded Planar Graph)
Let $G = (V, A)$ *be a simple directed planar graph with a planar embedding* (ϕ, ψ)*. An elementary cycle* $s = (\delta_1 a_1, \delta_2 a_2, \ldots, \delta_n a_n)$ *in* G *is called* **a face cycle of** G **(with respect to** (ϕ, ψ)**)***, if the interior or the exterior of its induced Jordan curve is equal to a face of the embedded planar graph.*

Given a space with obstacles, there is a special class of embedded planar graphs, where each face contains exactly one obstacle. Intuitively it is clear that such graphs capture the topological properties of the space with obstacles, since the obstacles may be transformed into the according full faces in a continuous manner. In general, these structures are called ***deformation retracts***. In the given special context of planar environments with obstacles, we will call these graphs ***skeletons***, and their properties are given in the following definition.

Definition 5.39 (Skeleton of a Space with Obstacles)
Let E *be a 2-dimensional space with obstacle set* $H = \{h_1, \ldots, h_k\}$*. A simple directed planar graph* $G = (V, A)$ *with a planar embedding* (ϕ, ψ) *is called* **skeleton of** E*, if*

(i) G *is weakly connected,*

(ii) the embedding does not touch any obstacles, i.e. $\psi(A, [0, 1]) \subseteq E$*,*

(iii) every obstacle $h_i \in H$ *is located inside a bounded face of the embedded planar graph, and*

(iv) every face of the embedded planar graph is either unbounded or contains exactly one obstacle $h_i \in H$*.*

As an example, Figure 5.15 shows a skeleton of a space with four obstacles. The face cycles of this skeleton are

- cycle $a b i$ for the face containing h_1,
- cycle $c n d$ for the face containing h_2,
- cycle $\bar{i} n e \bar{m} \bar{k}$ for the face containing h_3,
- cycle $h k m \bar{f} \bar{g}$ for the face containing h_4, and
- cycle $a b \bar{c} \bar{d} e \bar{f} \bar{g} h$ for the unbounded face

as well their respective inverses.

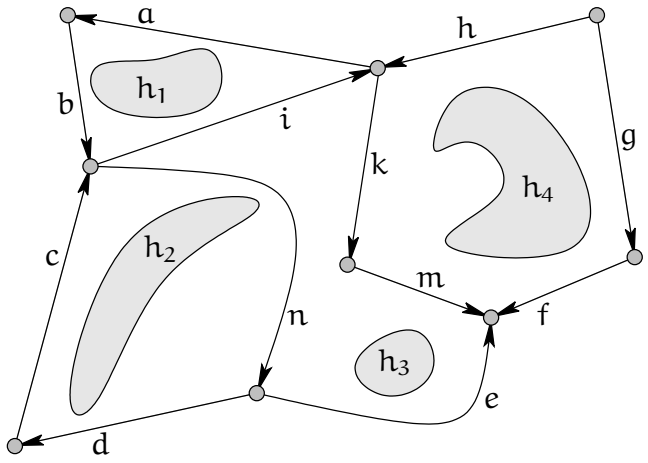


Figure 5.15: Skeleton of space with obstacles

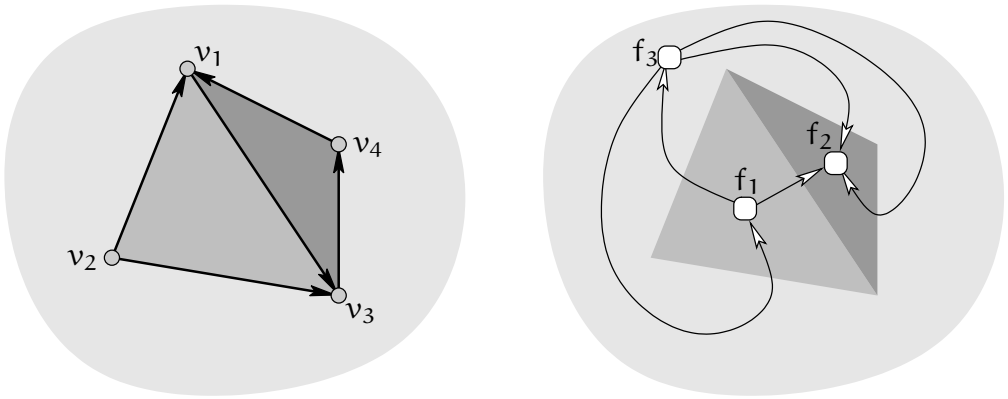
Mapping Closed Curves to Skeleton Cycles

As mentioned above, we aim at mapping a closed planar curve to a homotopic embedded planar graph cycle. In order to accomplish this task, we will use the dual graph of the skeleton and an embedding thereof.

An example of an embedded planar graph G and an embedding of its dual graph G^* is depicted by Figure 5.16. The original planar graph $G = (V, A)$ is shown in Figure 5.16(a), its faces are painted with different shades of gray. In order to find its dual graph $G^* = (V^*, A^*)$, each face f of the original graph is taken as a vertex $f \in V^*$ of the dual graph, and for each arc $a \in A$ which separates faces f_i (on the right side of the arc) and f_j (on the left side) of G, an arc $a^* \in A^*$ exists in the dual graph with source vertex f_i and target vertex f_j (see Figure 5.16(b)). It is an easy exercise to show that each arc a^* of G^* can be embedded in a manner such that no other embedded arc is intersected but the embedded image of its corresponding arc a in G. Furthermore, the faces of G^* correspond to vertices of G, and G^* can be embedded in a way such that each face of G^* contains the embedding of its corresponding vertex $v \in V$ of G. Additionally, we will require that the vertex f_∞ of G^* representing the unbounded face of G is embedded at infinity, i.e. no Jordan curve contains the image of f_∞ in its interior.

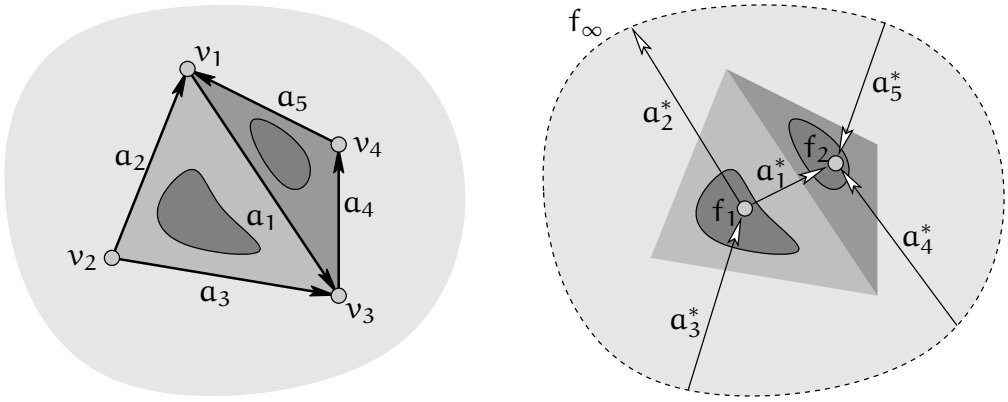
We know that each bounded face of a skeleton G contains exactly one obstacle. Therefore, the dual graph G^* of a skeleton can be embedded such that each of its vertices is located inside a different obstacle, see Figure 5.17. As a consequence, the faces of G^* partition the space with obstacles E. Given a curve c in that space E, we can trace the faces of G^* which are traversed by that curve. Since the vertices of G^* are located in the obstacles, only proper non-degenerate transitions[1] from one face to another face will

[1] We will ignore cases where the curve touches an arc without crossing it.



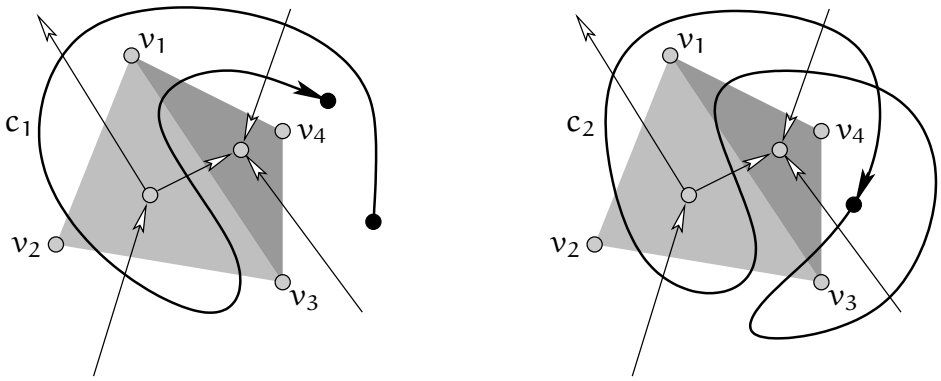
(a) Original graph (b) Dual graph

Figure 5.16: Embedded planar graph and its dual graph



(a) Skeleton (b) Embedded dual graph

Figure 5.17: Skeleton and its embedded dual graph



(a) Induced chain (b) Induced cycle

Figure 5.18: Curves induce cycles or chains in a skeleton

occur on the course of c.

Definition 5.40 (Chain Induced by a Curve)
Let E be a space with obstacles, $G = (V, A)$ a skeleton of E with planar embedding (ϕ, ψ), and $G^* = (V^*, A^*)$ the dual graph of G with embedding (ϕ^*, ψ^*) such that each vertex of G^* is embedded in an according obstacle of E, or at infinity if the vertex corresponds to the unbounded face of G.

Let $c : [a, b] \to E$ be a curve, let

$$(\alpha_1, t_1, \omega_1, \alpha_2, t_2, \omega_2, \ldots, \alpha_n, t_n, \omega_n)$$

be a sequence of parameter values with $\alpha_1 = a$, $\omega_n = b$, and

(i) $a \leq \alpha_i < t_i < \omega_i \leq \alpha_{i+1} \leq b$,

(ii) $c(t_i)$ *and* $c(t_{i+1})$ *are located in different faces of* G^*,

(iii) $c(t)$ *and* $c(t_i)$ *are located in the same face of* G^* *for* $\alpha_i < t < \omega_i$

(iv) $c([\omega_i, \alpha_{i+1}])$ *is part of an arc of* G^*

for any $0 < i \leq n$.[2] *Then, the chain $s = (v_1, v_2, \ldots, v_n)$ in the skeleton G is called* **induced by curve** *c, if $c(t_i)$ is located in the same face of G^* as $\Phi(v_i)$ for $i = 1, 2, \ldots, n$.*

In a similar way, the above definition is applied to closed curves, which induce cycles in the skeleton graph. For example, the curve c_1 in Figure 5.18(a) induces a chain $s_1 = (v_4, v_1, v_2, v_3, v_1, v_4)$ in the skeleton of the previous example. The closed curve c_2 in Figure 5.18(b) induces a cycle $s_2 = (v_4, v_3, v_4, v_2, v_3, v_2, v_1, v_4)$ in the same skeleton.

[2] We ignore cases where a curve oscillates infinitely between two faces.

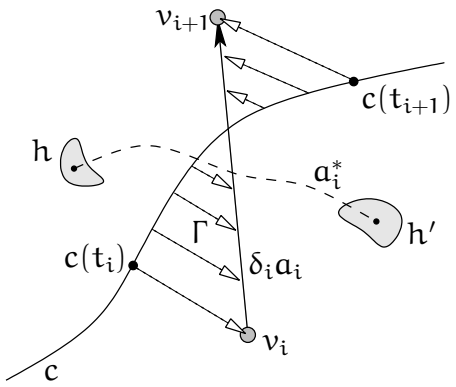


Figure 5.19: Proving homotopy of closed curve and its embedded induced cycle

Property 5.41
Let c *be a closed curve in a space with obstacles* E*, let* c' *be the curve of a cycle* s *which is induced by curve* c*. Then,* c *and* c' *are homotopic.*

Proof:

Let

$$s = (v_1, \delta_1 a_1, v_2, \delta_2 a_2, \ldots, v_n, \delta_n a_n, v_{n+1})$$

be the induced cycle of c,

$$(\alpha_1, t_1, \omega_1, \alpha_2, t_2, \omega_2, \ldots, \alpha_n, t_n, \omega_n)$$

the according parameter values as required by Definition 5.40. Then there is a continuous mapping $\Gamma : [0,1] \times [0,1] \to E$ which

(i) transforms $c(t_i)$ to the embedded skeleton vertex v_i and

(ii) transforms $c|_{[t_i, t_{i+1}]}$ to the embedded skeleton arc $\delta_i a_i$

simultaneously for $1 \leq i \leq n$ without crossing any obstacle (see Figure 5.19), since both the restriction of c to $[t_1, t_{i+1}]$ and the embedded arc $\delta_i a_i$ are completely contained within the union of the image of a_i^* and its incident faces minus the obstacles. This mapping proves the claimed property. □

So far we transformed a given closed plane curve c to a skeleton cycle s whose embedding is homotopic to c, i.e. they have the same topological norm.

Group Representation of Homotopy Classes

Now we will transform a cycle s in the skeleton to a product $p = \Pi_{i=1}^{n} e_i$ of primitive elements of the fundamental group representing the homotopy class $[s]$ of s, i.e. $p = [s]$. To accomplish this task, a tree of the skeleton will be used.

Definition 5.42 (Forest, Tree)
A directed graph $G = (V, A, \alpha, \omega)$ is called **forest***, if G does not contain a simple cycle. A forest $G = (V, A, \alpha, \omega)$ is called* **tree***, if G is weakly connected.*

Definition 5.43 (Spanning Tree of a Graph)
Let $G = (V, A, \alpha, \omega)$ be a directed graph. Then, a tree $G_T = (V, A', \alpha|_{A'}, \omega|_{A'})$ with $A' \subseteq A$ is called **spanning tree of** *G.*

Property 5.44 (Equivalent Definitions of a Spanning Tree)
Let $G = (V, A, \alpha, \omega)$ be a directed graph, and $A' \subseteq A$ a subset of arcs. Then, the following propositions are equivalent:

(i) $G_T = (V, A', \alpha|_{A'}, \omega|_{A'})$ *is a spanning tree of* G.

(ii) A' *is a minimal subset of arcs such that* G_T *is weakly connected.*

(iii) A' *is a maximal subset of arcs such that* G_T *has no simple cycles.*

(iv) For any pair $v_i, v_j \in V$ *of vertices, there is a unique simple chain from* v_i *to* v_j *in* G_T.

For a proof, see for example (Krumke et al., 2000), or any other good textbook on elementary graph theory.

Property 5.45 (Reduced Chains in Trees are Elementary)
Let $T = (V, A, \alpha, \omega)$ be a tree, $s = (v_1, \delta_1 a_1, v_2, \ldots, \delta_n a_n, v_n)$ a reduced chain in T. Then, the chain s is elementary.

Proof:

Assume s is not simple. Then, some arc a_i in s is repeated, i.e.

$$s = (\ldots, \delta_i a_i, \ldots, \delta_j a_j, \ldots)$$

with $a_i = a_j$, and $j \geq i+1$, i.e. there is at least one other arc between the two occurrences of $a_i = a_j$, since s is reduced.

If $\delta_i = \delta_j$, then there are two different simple chains from $\alpha(\delta_i a_i)$ to $\omega(\delta_i a_i)$ in T, namely $(\delta_i a_i)$ and $(\bar{\delta}_{j-1} a_{j-1}, \ldots, \bar{\delta}_{i+1} a_{i+1})$, which is a contradiction to T being a tree.

If $\delta_i \neq \delta_j$, then there are two different simple chains from $\omega(\delta_i a_i)$ to $\omega(\delta_{i+1} a_{i+1})$, namely $(\delta_{i+1} a_{i+1})$ and $(\overline{\delta_{j-1}} a_{j-1}, \ldots, \overline{\delta_{i+2}} a_{i+2})$ (the latter being empty for $j = i + 1$), which is a contradiction to T being a tree. Therefore, chain s must have been simple.

Now assume s is not elementary. Then, some vertex v_i is repeated, i.e.

$$s = (\dots, v_i, \delta_i a_i, \dots, \delta_{j-1} a_{j-1}, v_j, \dots)$$

with $v_i = v_j$, and, without loss of generality, $v_k \neq v_i$ for $i < k < j$. Since T is a tree, arc a_i is not a loop, and therefore $v_{i+1} \neq v_i$. Since s is simple, it is $a_i \neq a_{j-1}$, and therefore $(\delta_i a_i)$ and $(\overline{\delta_{j-1}} a_{j-1}, \dots, \overline{\delta_{i+1}} a_{i+1})$ are two different simple chains from $\alpha(\delta_i a_i)$ to $\omega(\delta_i a_i)$, which is a contradiction to T being a tree. Thus, chains s must have been elementary. □

Corollary 5.46
Let $T = (V, A, \alpha, \omega)$ be a tree, and s a chain in T. Then the following propositions are equivalent:

(i) Chain s is reduced.
(ii) Chain s is simple.
(iii) Chain s is elementary.

Proof:

According to 5.45, (i) implies (iii). The implications (iii) ⇒ (ii) (elementary chains are simple) and (ii) ⇒ (i) (simple chains are reduced) are obvious. □

Definition 5.47 (Rooted Tree)
Let $G = (V, A)$ be a tree, and $r \in V$ a tree vertex which we will call **root of tree** *G. Let $v \in V$ be another vertex of G, and $s = (v_0, v_1, \dots, v_n)$ the unique simple chain from $r = v_0$ to $v = v_n$. Then, for $1 \leq i \leq n$, vertex v_i is called* **child of vertex** *v_{i-1}. If vertex v is a child of vertex u, vertex u is called* **parent of** *v. Vertices of a tree which have no children are called* **leaves**.

Given a chain s in a skeleton $G = (V, A)$ and a spanning tree $G_T = (V, A')$ of G, each arc a of s will be either a tree arc $a \in A'$ or a non-tree arc $a \in A - A'$. Furthermore, if the chain s is a cycle in G, it has to contain at least one non-tree arc $a \in A - A'$.

Definition 5.48 (Induced Chains and Cycles)
Let $G = (V, A, \alpha, \omega)$ be a weakly connected directed graph, and let $T = (V, A_T)$ be a spanning tree of G.

For vertices $v_i, v_j \in V$, we denote by $s_T(v_i, v_j)$ the **unique simple chain from** *v_i* **to** *v_j* **in** *T.*

For a non-tree arc $a \in A - A_T$, we denote by $s_T(\delta a)$ the concatenation of the chain (δa) and the chain $s_T(\omega(\delta a), \alpha(\delta a))$, that is, the **unique simple cycle in** *G* **induced by** *δa* **and** *T. If the tree T is fixed, its specification may be omitted.*

Property 5.49 (Non-tree Edges Define Reduced Cycles)
Let $G = (V, A)$ be a skeleton, $G_T = (V, A')$ a spanning tree of G, and s a reduced cycle in G. Then, the cycle s is uniquely defined by its (cyclic) sequence of non-tree edges.

Proof:

Let $(\delta_1 a_1, \delta_2 a_2, \ldots, \delta_k a_k)$ be the (cyclic) sequence of non-tree arcs of s. Then, according to 5.44 and 5.45, there is a unique reduced chain in G_T from $\omega(\delta_i a_i)$ to $\alpha(\delta_{i+1} a_{i+1})$ for each $i = 1, \ldots, k$. Therefore, the cycle s is uniquely defined by its sequence of non-tree arcs. □

Now we are approaching a point where we can make use of a result from algebraic topology, which continues to undisclose the structure of the fundamental group of an environment with obstacles. But first the notion of a ***free group*** needs to be introduced.

Definition 5.50 (Free Group)
A group G *is called* **free** *if there is a subset* S *of* G *such that any element of* G *can be written in a unique way as a product of finitely many elements of* S *and their inverses (neglecting trivial variations as* $ab = acc^{-1}b$*). The elements of such a subset* S *of* G *are called* **generators** *of the group* G*.*

A proof of the following theorem can be found in the book by Massey (1967).

Theorem 5.51 (Fundamental Group of a Graph)
Let G *be any connected graph, and let* $T = (V, A')$ *be a spanning tree of* G*. Then, fundamental group of* G *is a free group on the set of generator* $\{s_T(a) \mid a \in A - A'\}$*.*

To put it in other words, the homotopy class of any closed curve in an environment with obstacles can be uniquely characterized by a sequence of non-tree arcs, if a skeleton of the environment and a spanning tree thereof are fixed. Yet we have to note that, in general, the cycles which are induced by the non-tree arcs are not guaranteed to contain exactly one obstacle in their interior. That is, the generators are not guaranteed to be primitive.

For example in the special case where the skeleton is an outerplanar graph (together with an according embedding), a spanning tree of the skeleton can be found such that the resulting generators of the fundamental group are primitive. Thereby a planar graph is called outerplanar, if it can be embedded in the plane such that all its vertices are incident to the unbounded face.

In the general case, we will have to choose a set of primitive elements as generators, and then express the homotopy classes of the cycles induced by the non-tree arcs (i.e. the generators from the theorem) using these primitive generators. In order to accomplish this task, we will define an arc labeling scheme, such that each arc label is a product of primitive elements, and the product of arc labels along a cycle is the group element corresponding to the homotopy class of the cycle.

The next two corollaries from the theorem above will be used later, helping to argue that moving on a tree will not interfere with homotopy.

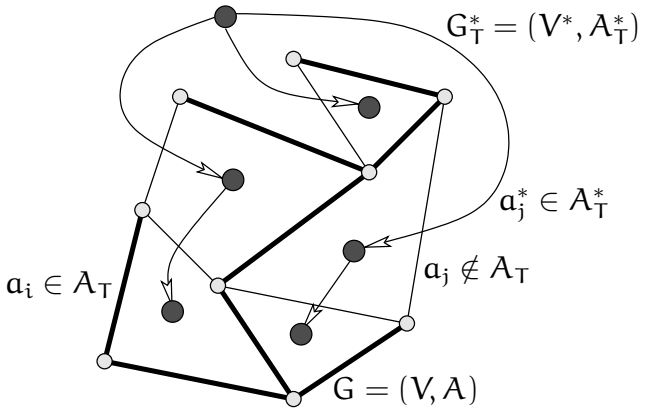


Figure 5.20: Non-tree arcs correspond to a tree in the dual graph

Corollary 5.52 (Homotopy of Cycles in Trees)
Let $G_T = (V, A)$ be an embedded planar tree. Any embedded cycle $c(s)$ in G_T is homotopic to a single point.

Proof:

A cycle in a tree cannot contain any non-tree edge. Therefore, its homotopy class is the empty product of generators. □

Corollary 5.53 (Homotopy of Chains in Trees)
Let $G_T = (V, A)$ be an embedded planar tree. Let s_1, s_2 be chains in G_T with same initial and terminal vertices. Then, the embedded chains $c(s_1)$ and $c(s_2)$ are homotopic.

Property 5.54 (Non-tree Arcs Correspond to a Tree in the Dual Graph)
Let $G = (V, A)$ be a skeleton with dual graph $G^* = (V^*, A^*)$, and $G_T = (V, A_T)$ a spanning tree in G. Then the graph $G_T^* = (V^*, A_T^*)$ with

$$A_T^* = \{a^* \in A^* \mid a^* \text{ is the dual of a non-tree arc } a \notin A_T\} \tag{5.34}$$

is a spanning tree of G^* (see Figure 5.20).

Proof:

Assume G_T^* contains a simple cycle. Then, this cycle separates a proper subset of faces of G^* from the other faces. These separated faces correspond to a set of vertices in G_T which are not connected to the other vertices in G_T. This is a contradiction to G_T being a tree. Therefore, the graph G_T^* must not have contained a simple cycle, i.e. it is a forest.

Now assume G_T^* is not connected. Then, there is a set $V' \subset V^*$ of vertices of G_T^* such that arcs from A^* that connect vertices from V' to vertices from $V^* - V'$ are not contained

in A_T^*. These arcs are the dual arcs of a cycle in A_T, which is a contradiction to G_T being a tree. Therefore, the forest G_T^* must have been connected, i.e. it is a tree. □

In order to convert a skeleton cycle into a product of primitive elements of the fundamental group, we will choose a spanning tree G_T of the skeleton G, and label the non-tree arcs of the skeleton with products of primitive elements of the fundamental group. These labels are multiplied as the non-tree arcs are traversed, and the result is the desired representation of the homotopy class of the embedded original cycle.

These non-tree arc labels are constructed incrementally along a tree G_T^* in the dual graph G^* of the skeleton. The construction starts at the leaves and terminates at the root of G_T^*, which is the vertex representing the unbounded face of G.

Definition 5.55 (Non-tree Arc Labels)
Let $G = (V, A)$ be a skeleton of a space E with obstacles. Let $G_T = (V, A_T)$ be a spanning tree of G, and $G^* = (V^*, A^*)$ the dual graph of G. Let $v_i^* \in V^*$ be the vertex of G^* corresponding to the face of obstacle h_i, and $v_\infty^* \in V^*$ the vertex of G^* corresponding to the unbounded face of G. Let $G_T^* = (V^*, A_T^*)$ a spanning tree of G^* induced by G_T as in 5.54, i.e. each arc a_i^* of G_T^* is the dual of the non-tree arc a_i of G. We choose v_∞^* as root vertex of G_T^*.

Without loss of generality (by means of appropriate index renaming), let $(v_i^*, \delta_j a_j^*, v_j^*)$ denote the (unique) simple chain connecting a parent vertex v_i^* to a child vertex v_j^* in G_T^*, i.e. the non-tree arc a_j is part of the face cycle of obstacle h_j. Furthermore, we will identify each obstacle h_i with a Jordan curve $h_i : [0, 1] \to E$ which contains only obstacle h_i in its interior and surrounds h_i in counterclockwise orientation. Thereby, the classes $[h_i]$ and $[\overline{h_i}]$ are primitive elements of the fundamental group.

Let $a_i \in A - A_T$ be a non-tree arc of G. Let s_i be the face cycle in G surrounding obstacle h_i in counterclockwise orientation. Cycle h_i contains at least one non-tree arc, since otherwise there was a simple cycle in the tree G_T. Assume that arc a_i is the only non-tree arc in cycle s_i, i.e. $s_i = (\dots, \delta_i a_i, \dots)$. Then, we define the **label of arc** *a_i as*

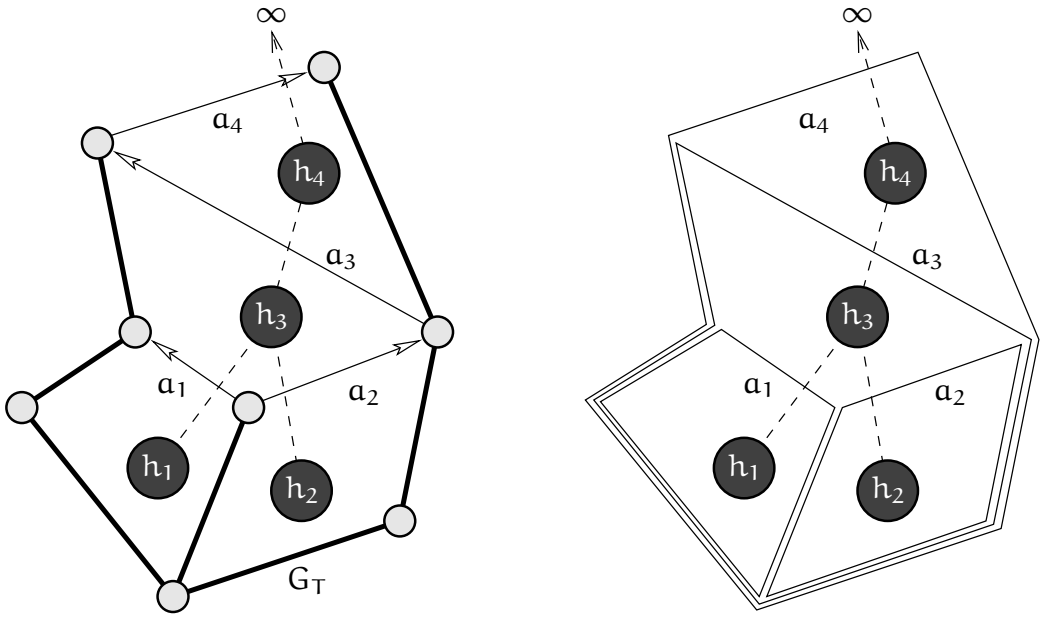
$$l(a_i) := \begin{cases} [h_i] & \text{if } \delta_i = '+', \\ [\overline{h_i}] & \text{else.} \end{cases} \tag{5.35}$$

Furthermore, we will write

$$l(\delta a) := \begin{cases} l(a) & \text{if } \delta = '+', \\ \overline{l(a)} & \text{else.} \end{cases} \tag{5.36}$$

Now assume there are $k + 1$ non-tree arcs $a_i, a_{j_1}, \dots, a_{j_k}$ in cycle s_i, i.e.

$$s_i = (\dots, \delta_i a_i, \dots, \delta_{j_1} a_{j_1}, \dots, \delta_{j_k} a_{j_k}, \dots). \tag{5.37}$$



(a) Tree arcs (bold) and non-tree arcs $a_{1,2,3,4}$ (b) Cycles induced by non-tree arcs

Figure 5.21: Computing arc labels

Then, we define the label of arc a_i *as*

$$l(a_i) := \begin{cases} [h_i] \cdot \overline{l(\delta_{j_k} a_{j_k})} \dots \overline{l(\delta_{j_1} a_{j_1})} & \text{if } \delta_i = '+' \\ l(\delta_{j_1} a_{j_1}) \dots l(\delta_{j_k} a_{j_k}) \cdot [\overline{h_i}] & \text{else.} \end{cases} \tag{5.38}$$

For the sake of completeness, we will define $l(a_t) := 1$ *for any tree arc* $a_t \in A_T$.

Example 5.1 (Arc Labels)
For an example see Figure 5.21. Arc a_1 *will be labeled as* $l(a_1) = [h_1]$*, since* $+a_1$ *is the only non-tree arc in the counterclockwise face cycle of* h_1*. Analogously, arc* a_2 *will be labeled as* $l(a_2) = [\overline{h_1}]$*, since the face cycle of* h_2 *contains* $-a_2$*. Now consider arc* a_3*. The counterclockwise face cycle of* h_3 *contains* $+a_3$*,* $-a_1$*, and* $+a_2$ *in that order. Therefore, according to the definition above,*

$$\begin{aligned} l(a_3) &= [h_3] \cdot \overline{l(+a_2)} \cdot \overline{l(-a_1)} \\ &= [h_3] \cdot [h_2] \cdot [h_1]. \end{aligned}$$

Note that the first line is equivalent to

$$l(+a_3) \cdot l(-a_1) \cdot l(+a_2) = [h_3],$$

and in a short moment we will prove that in general the product of the arc labels along any cycle is the homotopy class of the respective closed curve.

Back to the example, consider arc a_4 at last. The counterclockwise face cycle of h_4 contains $-a_4$ and $-a_3$ in that order, and therefore

$$\begin{aligned} l(a_4) &= l(-a_3) \cdot \overline{[h_4]} \\ &= \overline{[h_3] \cdot [h_2] \cdot [h_1]} \cdot \overline{[h_4]} \\ &= \overline{[h_1]} \cdot \overline{[h_2]} \cdot \overline{[h_3]} \cdot \overline{[h_4]}. \end{aligned}$$

Let us point out that the arc labels are well defined. Equation 5.38 in Definition 5.55 contains a recursive self-reference, and we have to clarify that termination is guaranteed. The non-tree arc a_i, for which the label $l(a_i)$ is currently being defined, has a dual arc a_i^* in the tree G_T^*. In the definition of the label $l(a_i)$, labels of non-tree arcs $a_{j_1}, \ldots, a_{j_k}$ are being used. These arcs have dual arcs $a_{j_1}^*, \ldots, a_{j_k}^*$ in the tree G_T^*, which are the successors of arc a_i^* on chains from the root of G_T^* to leaves of this tree. If an arc $a_{j_r}^*$ is incident to a leaf, the label of arc a_{j_r} is given directly. Otherwise, the label of arc a_{j_r} can be computed without using the labels of any of the arcs a_i, $a_{j_1}, \ldots, a_{j_k}$, since trees are acyclic. Therefore, the arc labels are well defined.

Property 5.56 (Arc Label is the Homotopy Class of Induced Cycle)
Let $G = (V, A, \alpha, \omega)$ be a skeleton with spanning tree $G_T = (V, A_T)$ and arc labels as in Definition 5.55. Let $a \in A - A_T$ be a non-tree arc of G, and let s be a chain in G_T from $\omega(a)$ to $\alpha(a)$. Then, the concatenation of $+a$ and s is a cycle s' in G, and the closed curve $c(s')$ is in the homotopy class $l(a)$.

Proof:

Clearly, the concatenation of a and s is a cycle in G.

Since the embedded curves $c(s)$ of all chains from $\omega(a)$ to $\alpha(a)$ in G_T are homotopic (cf. 5.53), we can assume that, without loss of generality (cf. 5.10), the chain s is the unique simple chain

$$s_{G_T}(\omega(a), \alpha(a)) = (v_1, \delta_1 a_1, v_2, \ldots, v_{n-1}, \delta_{n-1} a_{n-1}, v_n).$$

from $v_1 = \omega(a)$ to $v_n = \alpha(a)$ in G_T.

Since $a \notin A_T$, the cycle s' (i.e. the concatenation of s and a) is simple, and as the concatenation adds no further vertices, the cycle s' is also elementary. Therefore, the embedded closed curve $c(s')$ is a Jordan curve, and contains a certain number of obstacles in its interior. The remaining claim will be proved by induction for all numbers of contained obstacles.

The Jordan curve $c(s')$ contains at least one obstacle in its interior, since the cycle s' is simple. Assume h is the only obstacle in the interior of $c(s')$. Then, the cycle s' is a face cycle of h, and contains only one non-tree arc. If s' surrounds h_i in counterclockwise

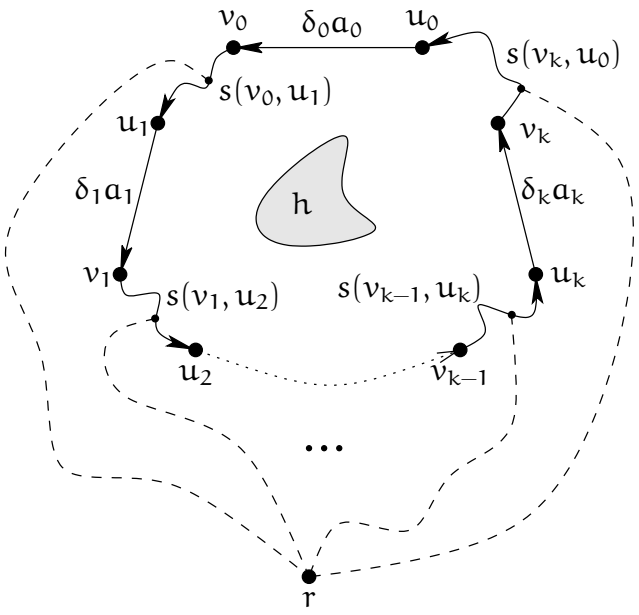


Figure 5.22: Non-tree arc labels are homotopy classes of induced cycles

orientation, then the label $l(a)$ is defined as the homotopy class of closed curves surrounding exactly h_i in counterclockwise orientation, i.e $c(s') \in l(a)$. If s' surrounds h_i in clockwise orientation, the inverse chain of s' will surround h_i in counterclockwise orientation and contains $-a$. Therefore, the label $l(a)$ is defined as the homotopy class of closed curves surrounding exactly h_i in clockwise orientation, i.e. $c(s') \in l(a)$.

Now assume that there are n obstacles in the interior of $c(s')$, and the claim has been proven for elementary cycles whose induced Jordan curve contains up to $n-1$ obstacles.

Let h be the obstacle in the interior of $c(s')$ which corresponds to the face of G which is bounded by arc a. A face cycle s_h of h must contain other non-tree arcs besides a, since otherwise h was the only obstacle in $c(s')$. Without loss of generality, let

$$s_h = (\ldots, \delta_0 a_0, \ldots, \delta_1 a_1, \ldots, \delta_2 a_2, \ldots, \delta_k a_k, \ldots) \tag{5.39}$$

for $a_0 = a$, non-tree arcs a_r, $r = 0, 1, 2, \ldots, k$, and $\delta_0 \in \{-, +\}$ such that s_h surrounds h counterclockwise, i.e. $c(s_h) \in [h]$ (see Figure 5.22). Each non-tree arc a_r, $r = 1, 2, \ldots, k$, induces an elementary cycle s_r which contains at most $n - 1$ obstacles in the interior of its curve $c(s_r)$, and $c(s_r) \in l(a_r)$ by induction hypothesis. Furthermore, for different non-tree arcs a_i and a_j, $i, j \in \{1, 2, \ldots, k\}$, the obstacles contained in the interior of $c(s_i)$ and $c(s_j)$, respectively, are disjunct, since the duals of the non-tree arcs form a tree G^*_T in the dual graph G^*, and a non-empty intersection of these obstacle sets would imply a simple cycle in G^*_T. Therefore, the labels of different non-tree arcs a_i and a_j contain different primitive group elements.

Let $r \in V$ be a vertex of G_T, and let $u_i = \alpha(\delta_i a_i)$ and $v_i = \omega(\delta_i a_i)$ for $i = 0, 1, \ldots, k$. Then, the chains $s(v_i, u_{i+1})$ and $s(v_i, r) \cdot s(r, u_{i+1})$ are path homotopic, and

$$\begin{aligned}
[s_h] &= [(\delta_0 a_0) \cdot s(v_0, u_1) \cdot (\delta_1 a_1) \cdot s(v_1, u_2) \ldots s(v_{k-1}, u_k) \cdot (\delta_k a_k) \cdot s(v_k, u_0)] \\
&= [s(r, u_0) \cdot (\delta_0 a_0) \cdot s(v_0, r) \cdot s(r, u_1) \cdot (\delta_1 a_1) \cdot s(v_1, r) \ldots s(r, u_k) \cdot (\delta_k a_k) \cdot s(v_k, r)] \\
&= [s(r, u_0) \cdot (\delta_0 a_0) \cdot s(v_0, r)] \cdot [s(r, u_1) \cdot (\delta_1 a_1) \cdot s(v_1, r)] \ldots [s(r, u_k) \cdot (\delta_k a_k) \cdot s(v_k, r)] \\
&= [s(\delta_0 a_0)] \cdot l(\delta_1 a_1) \ldots l(\delta_k a_k),
\end{aligned}$$

by induction hypothesis. Multiplication with inverse elements yields

$$[s_h] \cdot \overline{l(\delta_k a_k)} \ldots \overline{l(\delta_1 a_1)} = [s(\delta_0 a_0)], \tag{5.40}$$

where the left hand side is the label of $\delta_0 a_0$ (according to definition 5.55), and the right hand side is the homotopy class of the cycle induced by $\delta_0 a_0$. Checking the cases of $\delta_0 \in \{-, +\}$ proves the claim. □

Corollary 5.57 (Product of Arc Labels is Homotopy Class of Cycle)
Let $G = (V, A)$ be a skeleton with spanning tree G_T and arc labels as in 5.55.

Then, for any cycle $s = (\delta_1 a_1, \ldots, \delta_n a_n)$ in G, the equation

$$[c(s)] = \prod_{i=1}^{n} l(\delta_i a_i). \tag{5.41}$$

holds.

Proof:

Let $u_i = \alpha(\delta_i a_i) \in V$ and $v_i = \omega(\delta_i a_i) \in V$. Let $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ the non-tree arcs in s, $j_i < j_{i+1}$, and $r \in V$ a vertex in G. Then, by simple curve concatenation, and since only non-tree arcs are relevant for the homotopy class,

$$[s] = [(\delta_{j_1} a_{j_1}) \cdot s(v_{j_1}, u_{j_2}) \cdot (\delta_{j_2} a_{j_2}) \ldots (\delta_{j_k} a_{j_k}) \cdot s(v_{j_k}, u_{j_1})]$$

and after adding some detours without affecting homotopy,

$$[s] = \left[\prod_{i=1}^{k} (s(r, u_{j_i}) \cdot (\delta_{j_i} a_{j_i}) \cdot s(v_{j_i}, r)) \right],$$

which can be expressed as

$$[s] = \prod_{i=1}^{k} [s(r, u_{j_i}) \cdot (\delta_{j_i} a_{j_i}) \cdot s(v_{j_i}, r)],$$

using the group product. Since $s(r, u_{j_i})$ and $s(v_{j_i}, r)$ are chains within the tree G_T for $i = 1, \ldots, k$, we may use Property 5.56 and write

$$[s] = \prod_{i=1}^{k} l(\delta_{j_i} a_{j_i})$$

which can be transformed into

$$[s] = \prod_{i=1}^{n} l(\delta_i a_i)$$

by adding some neutral elements, i.e. the labels of the remaining tree arcs, which completed the proof. □

Theorem 5.58 (Correspondence between Cycles and Homotopy Classes)
Let E be a space with obstacles, and G a skeleton of E. For each homotopy class $[c]$ of closed curves, there is a unique (except for cyclic shifts) reduced cycle s in G such that $c(s) \in [c]$.

Proof:

Each homotopy class is an element of a free group (i.e. the fundamental group) which is uniquely determined by a product of generators. This product of generators is uniquely determined by a sequence of non-tree arcs, which are uniquely determined by a reduced cycle in G. □

Neutralization of Group Elements

Let $p = (e_1 \cdot e_2 \cdot \cdots \cdot e_n)$ be a product of primitive elements from the fundamental group of a space with obstacles. By deleting some of the factors e_i of p, other factors may become adjacent to their inverses, and eventually the product itself becomes equivalent to the neutral element. Here we are interested in a minimal sequence of factors to be deleted such that the resulting product becomes equal to the neutral element of the group.

For example, consider the product

$$p = a \cdot c \cdot \bar{a} \cdot b \cdot \bar{c} \cdot \bar{b}.$$

One could first remove the third and fourth factors ($\bar{a}$ and b), which makes the second factor (c) adjacent to its inverse, and then remove the two remaining factors, resulting a neutralization with four deletions. Alternatively, one could remove the second and fifth factors (c and $\bar{c}$), which results in a neutralization with only two deletions. Obviously, it has to be decided which occurrences of an element and its inverse will be used for neutralization.

Definition 5.59 (Deletion from a Sequence)
Let $S = (a_i)_{i\in\mathbb{N}}$ *be a sequence, and let* $I = \{i_1, i_2, \dots, i_m\} \subseteq \mathbb{N}$ *be a set of natural numbers, without loss of generality* $i_1 < i_2 < \dots i_m$*. Then, we will denote by* S/I *the* **sequence** S **after simultaneous removal of the elements indicated by** I*, i.e.*

$$S/I = (a_1, a_2, \dots, a_{i_1-1}, a_{i_1+1}, \dots, a_{i_2-1}, a_{i_2+1}, \dots, a_{i_m-1}, a_{i_m+1}, \dots). \tag{5.42}$$

Note that the removal index set I *specifies the indices of sequence elements, not their positions:*

$$(S/I_1)/I_2 = S/(I_1 \cup I_2). \tag{5.43}$$

Property 5.60 (Number of Deletions Required for Neutralization)
Let $p = (e_1 \cdot e_2 \cdot \dots \cdot e_n)$ *be a product of primitive elements from the fundamental group of a space with obstacles.*

By $w_i^j(p)$ *we will denote the minimum cardinality of an index set* $I \subset \mathbb{N}$ *such that the product of the elements of* $(e_i, e_{i+1}, \dots, e_j)/I$ *is equivalent to the neutral element.*

Then, the values $w_i^j(p)$ *satisfy the following property:*

$$w_i^j(p) = \begin{cases} 0 & \text{if } i > j, \\ 1 & \text{if } i = j, \\ w_{i+1}^{j-1}(p) & \text{if } i < j \text{ and } e_i = \bar{e}_j, \\ \min\left\{ w_i^k(p) + w_{k+1}^j(p) \mid k \in \{i, i+1, \dots, j-1\} \right\} & \text{else} \end{cases} \tag{5.44}$$

Proof:

Clearly, the index set $I = \{1, 2, \dots, j-i+1\}$ neutralizes the product of $(e_i, e_{i+1}, \dots, e_j)$ by deleting all elements. Since only the finite number of subsets of this index set need to be considered, the value $w_i^j(p)$ is well defined for any $1 \leq i, j \leq n$.

The claimed properties $w_i^{i-1}(p) = 0$ (empty product) and $w_i^i(p) = 1$ are trivial. For the proof of the other claimed properties, we use the notation

$$\prod s = e_1 \cdot e_2 \dots e_n \tag{5.45}$$

for the product of the elements of a sequence $s = (e_1, e_2, \dots, e_n)$.

Consider the case $i < j$ and $e_i = \bar{e}_j$, i.e. we have to prove the equality $w_i^j(p) = w_{i+1}^{j-1}(p)$. Clearly, $w_i^j(p) \leq w_{i+1}^{j-1}(p)$ holds, since neutralization of the product $e_{i+1} \cdot e_{i+2} \dots e_{j-1}$ will leave $e_i \cdot e_j = 1$. There is a set $I \subset \mathbb{N}$ with

$$\prod (e_i, \dots, e_j)/I = 1, \tag{5.46}$$

and $|I| = w_i^j(p)$.

We check the four cases of i and j being members or not being members of I. Assume $\{i, j\} \subset I$. Then

$$\prod(e_i, \ldots, e_j)/(I - \{i, j\}) = e_i \cdot e_j = 1, \tag{5.47}$$

$$\prod(e_{i+1}, \ldots, e_{j-1})/(I - \{i, j\}) = 1, \tag{5.48}$$

and therefore

$$w_{i+1}^{j-1}(p) \leq w_i^j(p) - 2 < w_i^j(p) \leq w_{i+1}^{j-1}(p), \tag{5.49}$$

which is a contradiction.

Assume $\{i, j\} \cap I = \emptyset$. Then

$$\prod(e_{i+1}, \ldots, e_{j-1})/I = 1, \tag{5.50}$$

and therefore

$$w_{i+1}^{j-1}(p) \leq w_i^j(p) \leq w_{i+1}^{j-1}(p), \tag{5.51}$$

which implies the claimed equality.

Assume $j \in I$ and $i \notin I$. Then,

$$\prod(e_i, \ldots, e_j)/(I - \{j\}) = e_j, \tag{5.52}$$

and

$$\prod(e_i, \ldots, e_{j-1})/(I - \{j\}) = 1. \tag{5.53}$$

Since $i \notin I$, there must be a $k \notin I$, $i + 1 \leq k \leq j - 1$ with $e_k = \bar{e}_i$ which neutralizes e_i. With this k,

$$\prod(e_{i+1}, \ldots, e_{j-1})/((I - \{j\}) \cup \{k\}) = 1, \tag{5.54}$$

and therefore

$$w_{i+1}^{j-1}(p) \leq |I| - 1 + 1 = w_i^j(p) \leq w_{i+1}^{j-1}(p), \tag{5.55}$$

which proves the claimed equality.

Analogously, assume $i \in I$ and $j \notin I$. Then,

$$\prod(e_i, \ldots, e_j)/(I - \{i\}) = e_i \tag{5.56}$$

and

$$\prod(e_{i+1}, \ldots, e_j)/(I - \{i\}) = 1. \tag{5.57}$$

Since $j \notin I$, there must be a $k \notin I$, $i + 1 \leq k \leq j - 1$ with $e_k = \bar{e}_j$ which neutralizes e_j. With this k,

$$\prod(e_{i+1}, \ldots, e_{j-1})/((I - \{i\}) \cup \{k\}) = 1, \tag{5.58}$$

and therefore

$$w_{i+1}^{j-1}(p) \leq |I| - 1 + 1 = w_i^j(p) \leq w_{i+1}^{j-1}(p), \tag{5.59}$$

which proves the claimed equality, and closes the case $i < j$ and $e_i = \bar{e}_j$.

Now consider the case $i < j$ and $e_i \neq \bar{e}_j$, i.e. we have to prove the equality

$$w_i^j(p) = \min\left\{w_i^k(p) + w_{k+1}^j(p) \mid k \in \{i, i+1, \ldots, j-1\}\right\}. \tag{5.60}$$

Clearly, $w_i^j(p)$ is equal to or less than the right hand side minimum, since each combination of neutralizations of $e_i \ldots e_k$ and $e_{k+1} \ldots e_j$ on the right hand side neutralizes the complete product $e_i \ldots e_j$, too. The other inequality remains to be proven.

Let $k \in \{i, \ldots, j-1\}$ such that $w_i^k(p) + w_{k+1}^j(p)$ is minimal. Let $I \subset \mathbb{N}$ with $|I| = w_i^k$ and

$$\prod(e_i, \ldots, e_j)/I = 1. \tag{5.61}$$

Assume $j \in I$. Then

$$\prod(e_i, \ldots, e_j)/(I - \{j\}) = e_j \tag{5.62}$$

and

$$\prod(e_i, \ldots, e_{j-1})/(I - \{j\}) = 1, \tag{5.63}$$

therefore

$$w_i^{j-1}(p) + w_j^j(p) \leq |I| - 1 + 1 = w_i^j(p) \tag{5.64}$$

which proves the remaining inequality, and thereby the claimed equality.

Now assume $i \in I$. Then

$$\prod(e_i, \ldots, e_j)/(I - \{i\}) = e_i \tag{5.65}$$

and

$$\prod(e_{i+1}, \ldots, e_j)/(I - \{i\}) = 1, \tag{5.66}$$

therefore

$$w_i^i(p) + w_{i+1}^j(p) \leq 1 + |I| - 1 = w_i^j(p) \tag{5.67}$$

which proves the remaining inequality, and thereby the claimed equality.

Finally, assume $\{i, j\} \cap I = \emptyset$, i.e. $I \subseteq \{i+1, \ldots, j-1\}$. Then

$$\prod(e_i, \ldots, e_j)/I = e_i \cdot e_j = 1, \tag{5.68}$$

which implies $e_i = \bar{e}_j$, which is a contradiction to the considered case. Therefore, $\{i, j\} \cap I \neq \emptyset$, and the claimed equality has been proven. □

Property 5.60 indicates that the topological norm of an element p of the fundamental group can be computed using a dynamic programming approach, as illustrated by Algorithm 5. Since it is not important where the concatenation of closed loops starts, we

Algorithm 5 PRODUCT NEUTRALIZATION

1: **input:** a product $p = (e_1 e_2 \dots e_n)$ of primitive group elements
2: consider cyclic shifts of p by treating indices as $i = i + kn \mod n$ for $k \in \mathbb{Z}$
3: let $M = (m_{i,j}) \in \mathbb{N}_0^{n \times n}$ be a matrix
4: **for** $j = 1, 2, \dots, n$ **do** {consider each subproduct of length j}
5: **for** $i = 1, 2, \dots, n$ **do** {consider the subproduct $e_i e_{i+1} \dots e_{i+j-1}$}
6: **if** $j = 1$ **then**
7: $m_{i,j} \leftarrow 1$
8: **else if** $e_i = \bar{e}_{i+j-1}$ **then**
9: **if** $j > 2$ **then**
10: $m_{i,j} \leftarrow m_{i+1,j-2}$
11: **else**
12: $m_{i,j} \leftarrow 0$
13: **end if**
14: **else**
15: $m_{i,j} \leftarrow j$
16: **for** $k = 1, 2, \dots, j-1$ **do**
17: let $s = m_{i,k} + m_{i+k,j-k}$
18: $m_{i,j} \leftarrow \min\{s, m_{i,j}\}$
19: **end for**
20: **end if**
21: **end for**
22: **end for**
23: **return** $\min\{m_{i,n} \mid i = 1, 2, \dots, n\}$

have compute the minimum number of required factor deletions for each cyclic shift of the product, in order to find the topological norm of the product.

Due to the three nested loops starting in lines 4, 5, and 16, the algorithm runs in time at most $\mathcal{O}(n^3)$, where n is the number of factors in the product p, and returns the minimum number of factor deletions required to neutralize a cyclic shifted copy of p. The correctness of the algorithm follows from Property 5.60: After round j of the outer loop, each matrix element $m_{i,r}$ with $1 \leq i \leq n$ and $1 \leq r \leq j$ contains the minimum number of factor deletions required to neutralize the subproduct of p of length r starting at index i, i.e. the value $m_{i,r} = w_i^{i+r-1}(p)$, since the applied operations correspond directly to the cases in Property 5.60. Therefore, after round n of the outer loop, the returned minimum number has the claimed property.

Example 5.2
Let a, b, c be primitive group elements corresponding to three different obstacles. Consider the product

$$p = (a\bar{b}c\bar{a}\bar{c}b).$$

For its neutralization, each substring of length 1 requires the deletion of 1 factor, and each of the substrings $a\bar{b}$, $\bar{b}c$, $c\bar{a}$, $\bar{a}\bar{c}$, $\bar{c}b$, *and* ba *of length 2 requires deletion of 2 factors. The following table shows the required number* w *of deletions for each substring of length greater than 2 to be neutralized.*

String	w	*String*	w	*String*	w	*String*	w
$a\bar{b}c$	*3*	$a\bar{b}c\bar{a}$	*2*	$a\bar{b}c\bar{a}\bar{c}$	*3*	$a\bar{b}c\bar{a}\bar{c}b$	*2*
$\bar{b}c\bar{a}$	*3*	$\bar{b}c\bar{a}\bar{c}$	*2*	$\bar{b}c\bar{a}\bar{c}b$	*3*	$\bar{b}c\bar{a}\bar{c}ba$	*2*
$c\bar{a}\bar{c}$	*1*	$c\bar{a}\bar{c}b$	*2*	$c\bar{a}\bar{c}ba$	*3*	$c\bar{a}\bar{c}ba\bar{b}$	*2*
$\bar{a}\bar{c}b$	*3*	$\bar{a}\bar{c}ba$	*2*	$\bar{a}\bar{c}ba\bar{b}$	*3*	$\bar{a}\bar{c}ba\bar{b}c$	*2*
$\bar{c}ba$	*3*	$\bar{c}ba\bar{b}$	*2*	$\bar{c}ba\bar{b}c$	*1*	$\bar{c}ba\bar{b}c\bar{a}$	*2*
$ba\bar{b}$	*1*	$ba\bar{b}c$	*2*	$ba\bar{b}c\bar{a}$	*3*	$ba\bar{b}c\bar{a}\bar{c}$	*2*

Property 5.61 (Alternative Neutralization Approach)
Let $p = (e_1 \cdot e_2 \cdot \dots \cdot e_n)$ *be a product of primitive elements from the fundamental group of a space with obstacles.*

Let $G = (V, A)$ *be a simple directed graph with* $V = \{1, 2, \dots, n\}$, *arc length* $c : A \to \mathbb{N}$, *and the arc set* A *incrementally defined as follows (vertices modulo* n*):*

- $(i, i+1) \in A$ *and* $c(i, i+1) = 1$ *for* $i \in V$,
- *for* $d = 1, 2, \dots, n-1$,
  - *if* $e_i = \bar{e}_{i+d}$, *then* $(i, i+d+1) \in A$ *and* $c(i, i+d+1)$ *is the length of a shortest path from* $i+1$ *to* $i+d$ *in the graph as constructed so far,*
- *and no other arcs are in* A.

Then, the topological norm of p *is the length of a shortest closed path in* G.

Proof:

Let $c = (v_1, v_2, \dots, v_n)$ with $v_n = v_1$ be a shortest closed path in a graph $G = (V, A)$ defined for a product $p = (e_1 e_2 \dots e_n)$ as described. Clearly, path c is simple, and its length $\sum_{i=1}^{n-1} c(v_i, v_{i+1})$ is not greater than n, since the path $(1, 2, \dots, n)$ is closed and has length n.

One can show by induction over $d > 0$ and by using Property 5.60, that for each arc $a = (i, i+d) \in A$, its arc label $c(a)$ is the minimum number of factor deletions required to neutralize the product $e_i e_{i+1} \dots e_{i+d-1}$, and that for each pair of vertices i and $i+d$, the length of a shortest path from i to $i+d$ is the minimum number of factor deletions required to neutralize the product $e_i e_{i+1} \dots e_{i+d-1}$.

Therefore, the shortest closed path c witnesses that the minimum number of factor deletions required to neutralize the product $e_{v_1} e_{v_1+1} \dots e_{v_1-1}$ is $\sum_{i=1}^{n-1} c(v_i, v_{i+1})$, and no other cyclic shift of the product p can be neutralized with less deletions. □

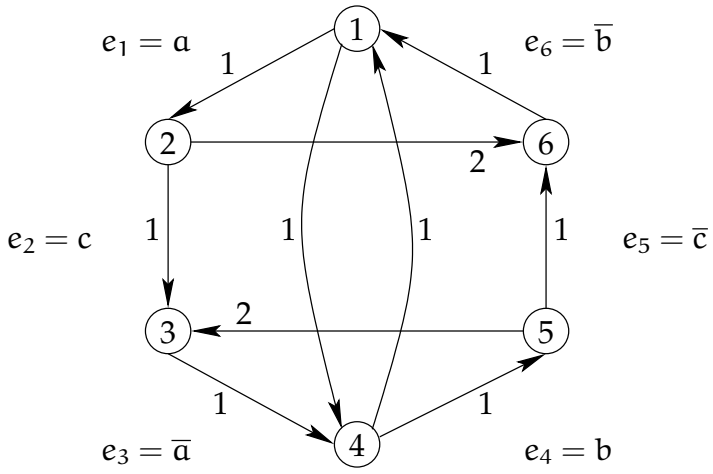


Figure 5.23: Topological norm as length of shortest cycles

Example 5.3
For an example, see Figure 5.23, where a graph as described in Property 5.61 is shown for the product $ac\bar{a}b\bar{c}\bar{b}$*. The shortest cycle traverses vertices 1 and 4, and has length 2.*

Algorithm and Complexity

In order to implement the presented approach, one can use a triangulation of the space with obstacles to generate a skeleton, More specifically, the dual graph of a triangulation of a space with obstacles can serve as a skeleton of that space. Furthermore, it is sufficient to consider only one single point from each obstacle, which makes it possible to use standard computational geometry libraries for the computation of the skeleton. A triangulation G^* of a set of n_h points can be computed in time $\mathcal{O}(n_h \log n_h)$, and the number of edges, faces, and vertices of G^* is $\mathcal{O}(n_h)$, since G^* is a planar graph.

Next, a given closed curve has to be mapped to a cycle in the skeleton. When the curve is a polygonal chain c, this step will take time $\mathcal{O}(n_s + n_t)$ where n_s is the number of segments of c, and n_t is the number of triangulation faces which are traversed by c. In the worst case, curve c may traverse $\Omega(n_s \cdot n_h)$ triangulation faces, i.e. each segment of c may cross all faces of the triangulation (up to a constant factor).

Each non-tree arc in G is labeled with a product of primitive group elements, and the maximum number of factors in these products is $\Omega(n_h)$, since in the worst case an induced cycle $s_{G_T}(\delta a)$ for a non-tree arc a contains all of the n_h obstacles. This results in a group product describing the homotopy class of c with $\Omega(n_s \cdot n_h^2)$ factors in the worst case. Using the dynamic programming approach as described above for the neutralization of the product, the topological norm of c can be computed in time $\mathcal{O}(n_s^3 \cdot n_h^6)$.

Future work is required to investigate if there are more efficient approaches. For exam-

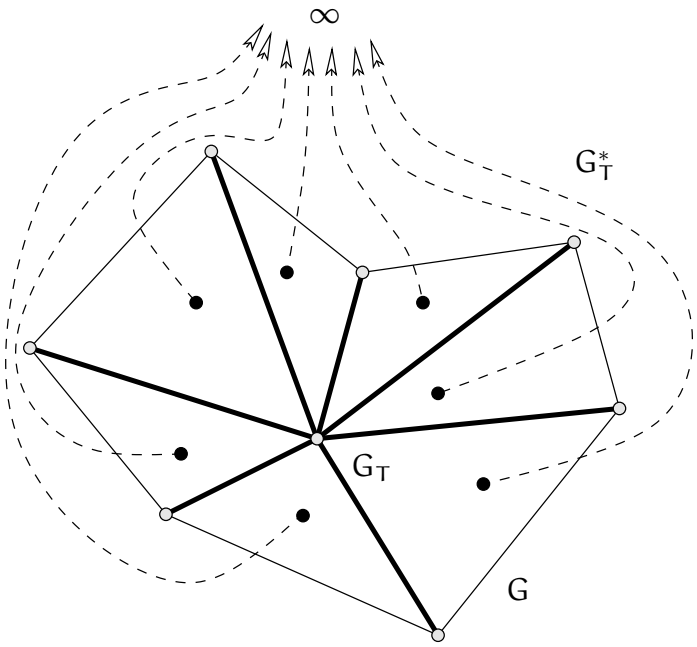


Figure 5.24: Skeleton G with shallow dual tree G_T^*

ple, a skeleton G of a space with obstacles could be chosen as indicated by Figure 5.24, such that any path from the root node of G_T^* to a leaf node has length 1, and thereby each non-tree arc of G is labeled with a degenerate product consisting of a single primitive group element. If such an approach is applicable in general, the complexity of computing the topological norm can be reduced to $\mathcal{O}(n_s^3 \cdot n_h^3)$.

5.4 Conclusion

In this chapter we presented metrics for curves, and algorithms to compute according distance values for plane curves. Motivation for curve metrics in the context of cooperative motion coordination is provided by quality assessment of robot behaviors as well as by prospective failure detection systems which shall be able to decide if a guide is still accompanied well. Furthermore, distance measures for paths could be applied to the paths of obstacles in order to detect some correlation among them. Besides the already known Hausdorff metric and the Fréchet metric, a topological metric on the set of homotopy classes of curves has been proposed, which takes into account the obstacles in the ambient space of the curves. Finally, an approach to the computation of topological distances of plane curves has been given.

Chapter 6

Situation Assessment

6.1 Introduction

In dynamic motion planning literature, moving obstacles are commonly modeled as solid objects traveling at a certain speed and being unaware of their environment. This assumption does not hold well for mobile robots operating in natural environments, where dynamic obstacles are mostly humans that are able to perceive and react on the robot's motion to avoid collisions, too. On the other hand there are situations where the robot unintentionally gains other agents' attention and might be obstructed by them. Thus we should not be concerned only by the problem of collision avoidance but might also want to reason about what level of "self-confident" driving is permissible to a robot in a given situation.

So for real world applications of autonomous mobile robots it is a promising idea not to reason only about shapes, positions and velocities of moving obstacles in the environment but also about their actions and intentions.

6.1.1 Motivating Examples

Figure 6.1(a) shows the situation where an autonomous mobile robot, depicted as a solid circle, encounters dense traffic when its plan was to perform a right turn. An obviously suboptimal reaction would be to stop and wait for the traffic to decrease. A human would recognize that in this situations it is necessary to carefully but resolutely join the stream, as a decrease of traffic or its end is not in sight. Figure 6.1(b) shows a situation where a moderate traffic stream is confronted with a narrow passage, so that involved agents have to line up. A human recognizes immediately that he or she has to join the queue at its end in order to get through the narrow passage. But this is a challenging situation for an autonomous mobile robot. Another example is shown in Figure 6.1(c), where a group of agents is walking together in a highly ordered manner. For instance

this might be a group of pupils on an excursion. A human recognizes that it would be quite a bad idea to try to join or cross this sort of traffic stream and instead prefers to wait. So this is an example where a robot that is able to recognize and join or cross dense traffic should still behave respectfully. Last but not least Figure 6.1(d) shows one human deliberately obstructing the robot. Here a robot that is not aware of this situation surely is trapped.

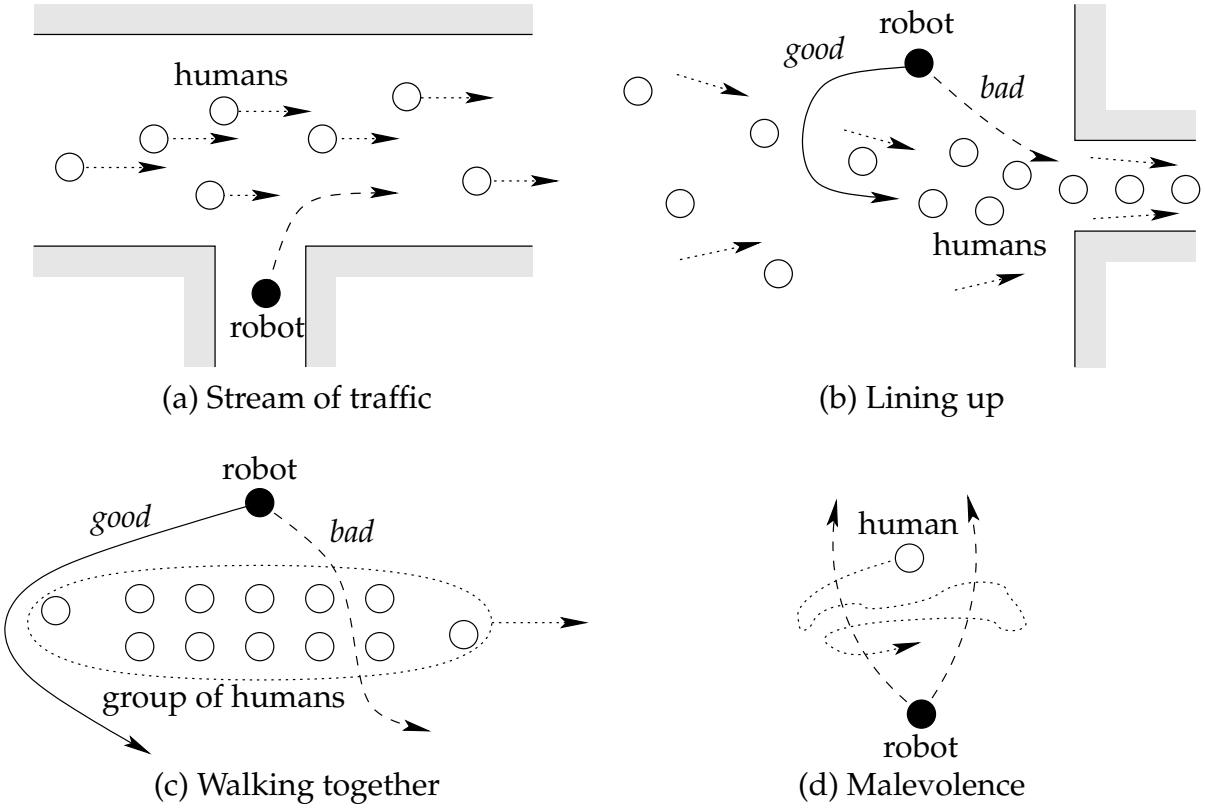


Figure 6.1: Example situations in crowded environments

So we expect mobile robots to behave more efficiently in real-world scenarios by increased situation awareness. This appears plausible as such robots are able to reason about up to which degree other agents behave respectfully (i.e., decelerate to permit the robot to safely join a traffic stream) or disrespectfully (i.e., deliberately obstruct the robot's desired motion) on the robot's behalf in a certain situation. Recognition of a certain situation may move the robot to adapt its current behavior or choose a new behavior to reflect the newly recognized circumstances.

6.1.2 Overview

Section 6.2 proposes a situation taxonomy that is based on the agents being involved and the interactions taking place. Since we do not expect to be able to recognize any imaginable situation, we focus on a collection of important prototypical situations encountered in a crowded environment that are presented in Section 6.3. Subsequently, various existing and related approaches to the recognition of actions, intentions and situation among agents are reviewed in Section 6.4. Finally, a solution to the problem of recognizing deliberate obstructions is proposed in Section 6.5.

6.2 Situation Taxonomy

We consider situations as configurations of sets of agents, where an agent is a human or the unique robot. Thereby, possible sets of agents involved in one situation are (a) a single human, (b) several humans, (c) a single robot, (d) one robot and one human, or (e) one robot and several humans. The robot is supposed to observe the environment and appropriately detect occurrences of situations. Furthermore, we assume the agents involved in one situation to be located close to each other.

At the top-level of our taxonomy, situations are classified according to the interaction among the agents. So there are situations with interaction taking place and the agents being located close to each other deliberately. On the other hand, proximity that is not caused by intended interaction raises situations from the complement class, where agents are situated or moving close to each other merely by accident. Clearly, a situation is not an interaction situation if there is only one agent involved in it.

6.2.1 Situations Driven by Interaction

Interaction between two or more humans occurs frequently and might be some form of communication or exchange of some objects (however we do not expect to be able to further distinguish this case given contemporary sensor and processing techniques).

Interaction of humans with a robot may be motivated in many ways, for example by cooperation, curiosity, or even hostility. Situations of cooperation can be regarded in two different ways. Firstly, the robot may be explicitly told to cooperate with a human (for example to accompany that human). Secondly, the robot might autonomously recognize a situation where cooperation is possible and appreciated (for example opening a door for a human moving towards that door). Situations initiated by curiosity might be driven by the robot as well as by a human agent in order to discover some interesting facts about the other agent. Finally, hostile situations are not provoked by the robot in the ideal case, but might be encountered by the robot in public environments.

6.2.2 Situations Without Deliberate Interaction

Situations that are not induced by interaction are mainly traffic situations, where the participating agents share a common goal or follow up interfering goals of motion. This class contains all situations involving only a single agent, too. Collective traffic situations include streams, queues and jams (as partially shown by Figure 6.1), crossing streams surely being one of the more complex examples. A very common example involving a robot is given if this robot has to navigate through dense pedestrian traffic.

Situations involving only a single agent profoundly correspond to the agent's mental state. For example an agent may be waiting for some event and be bored. So he, she, or it will stand still or wander at low speeds, in contrast to an agent pursuing a clear goal.

6.3 Situations in Crowded Public Areas

The previous section introduced a synthetic approach to a set of observable situations. Now it is also interesting to examine which situations are experienced in practice. Below we describe situations which have been encountered during several experiments in the concourse of the railway station of Ulm.

6.3.1 Situations Involving Humans

From a temporal point of view the most widespread situation for a human in a concourse is waiting for some event to take place. So a large amount of people are sitting, standing still or wandering around slowly. In contrast to that some humans walk at a notably higher speed and obviously towards a clear target, and often even small groups of individuals staying close to each other walk through the hall in a similar manner. The largest group of humans walking together that has been detected was a basic school class of about twenty pupils walking side by side as shown in Figure 6.1(c).

From time to time agents spontaneously change from the state of waiting and wandering to walking directly to a goal, for example into a book store.

Large crowds occur when trains arrive as then the number of humans in the railway station increases suddenly. Those crowds move from the platform towards the exit in a dense stream, but queuing at the exit doors has not been observed. Those streams were geometrically bounded rather well as long as their density permitted the members to walk still sufficiently fast. We observed spontaneous transitions with stream members, too, when some of them stopped in order to talk to each other more easily, hence forcing the stream to fork in front of them.

6.3.2 Situations Involving Humans and the Robot

In our experimental environment we are confronted with several situations that involve the robot. A first situation is the one for which the system has been designed initially: navigation through dense pedestrian traffic with dynamic avoidance of collisions. This situation can be further subdivided according to the class of traffic (sparse, streaming, turbulent, etc.), and the robot's desired direction relative to a predominant direction of traffic. When navigating through dense traffic it is interesting to notice that collision avoidance is performed cooperatively by the robot and humans. There are even people that refuse to approach the robot and prefer to choose another path. However most passers-by are not bothered by the robot's presence.

Another class of situations we want to consider are deliberate obstructions that occur when people become interested in the robot and try to fool it. This is in fact an important situation as robots are not yet quite common and thus do have to cope with curious people when operated in public areas.

6.4 Existing Approaches

This section presents some approaches to situation recognition and related problems and sub problems. Remarkably, only few of them are applied within robotic domains today. This overview neither claims to be complete nor to present a unique applicable classification.

6.4.1 Scene Analysis

Scene analysis is the basis for situation recognition, as before reasoning about actions and intentions of agents we have to be aware of the agents themselves and their geometric configuration (positions and velocities, absolute and relative to each other). This analysis can be done for example by computer vision using video images or by geometric computations on range images, and gives a first clue to the situation.

Scene analysis can be subdivided according to static and dynamic scenes. Static scene analysis is expected to identify occurrences of known or unknown objects in an image of the environment. When a sequence of images of a dynamic scene is given, the problem of single or multiple object tracking arises (Isard and Blake, 1998; Kluge et al., 2001; Schulz et al., 2001; Sobottka and Bunke, 1998), where the goal is to correctly identify the object motion between successive images.

Furthermore, Mohnhaupt and Neumann (1991) accumulate trajectories of tracked objects into a spatio-temporal buffer and abstract from the seen examples by a generalization step (which is similar to a dilatation step) and a convergence step (which is similar

to the computation of a medial axis) in order to obtain generic models of motion in the environment.

6.4.2 Action Recognition

More information about a scene and the current situation is obtained if objects are identified as agents and the motion that an agent performs is interpreted, i.e. the occurring action is recognized. Then an agent's motion can be described by additional attributes such as *wandering*, *purposeful*, or *periodical*, and its relation to other agents' motion may reveal further information.

Visual recognition of single agent actions is an active field of research, and many approaches have been developed (Bobick and Ivanov, 1998; Davis and Bobick, 1997; Nagel et al., 1995; Pentland and Liu, 1995; Rosales and Sclaroff, 1999). However they do not play an important role in our context of crowded public areas as they focus only on a single individual performing the action to be recognized. However, the work of Nagel et al. (1995) who introduced the notion of *situation graphs* is worth mentioning here. A situation graph is a bipartite directed graph with *situation nodes*, *link nodes*, and *argument nodes* at link nodes. Link nodes represent admitted transitions between a current situation and a possible successor situation. A situation represented by an according node comprises action options for the observed agent and the observable state of the world. Hence path in a situation graph represents the situational development of an aspect of the world. The situation is tracked by comparing changes in the world to the successor situation nodes of the current situation node.

More relevant approaches focus on recognition of coordinated multi agent action. Devaney and Ram (1997) analyze the spatial distribution and motion of participants in US Army training battles. From the spatial distribution they deduce significant portions of the battlefield and label them (for example as *left flank rear*). Indicators based on the change of concentration of participants in the labeled areas are used to recognize ongoing actions.

Symbolic names for locations and regions are also used by Intille (1999) in order to recognize American football plays. The plays are recognized by Bayesian belief networks (Charniak, 1991), their input being primitive single agent actions (recognized by Bayesian belief networks, too) along with temporal and logical relationships between these actions and labeled regions of the football field.

In contrast to the two preceding approaches addressing actions of tens or hundreds of agents, Oliver et al. (1999) use coupled hidden Markov models for the recognition of interaction between two human agents. The agents' relative distance, speed, orientation etc. are provided as input to the training process and to the recognition process, respectively. In order to obtain enough training data, the authors had to make use of a software simulator for the addressed types of agent interactions, since the most inter-

esting situations were the most rare, too.

6.4.3 Intention Reasoning

Given a sequence of an agent's actions, one might ask what the agent's intention or plan is. If the observed agent cooperates in the plan recognition process, the problem is known as *intended plan recognition*. If the agent tries to hinder the discovery of its plans, the problem is called *obstructed plan recognition*. The problem of recognizing the plan of an agent that does not try to cooperate nor hinder the recognition is know as *keyhole plan recognition*.

Charniak and Goldman (1993) apply Bayesian belief networks to plan recognition for story understanding. Others like Bobick and Ivanov (1998) or Pynadath (1999) employ stochastic context-free grammars to describe action sequences and recognize complex actions or plans as most likely deductions in this grammar. A comprehensive deterministic framework for plan recognition is presented by Kautz (1991).

These approaches might become interesting within the context of intelligent personal robot assistants that infer their users' intentions from their users' actions and can react by performing some helping action. But the plans considered by them appear far too detailed within the context of intentions of agents in public crowded areas.

However, there are approaches to intention reasoning that are of interest within our domain. For example Bayesian reasoning is applied to plan recognition in a robotic domain by Huber and Durfee (1993). They consider a mobile robot, some points of interest in the environment, and another moving agent. The robot reasons from the perceived motion of the other agent about the other agent's goal position (which is one of the points of interest) and moves to that point in order to meet him or her there.

If the system which observes and reasons about the environment also participates in this environment, recursive agent tracking (Gmytrasiewicz, 1992; Tambe, 1995) is an interesting paradigm. Given the robot R and an agent A, this principle proposes that R should not only track actions of A (i.e. track a model $M_R(A)$ of A's actions) but also should track at least a model $M_R(M_A(R))$ of A's model of R. Depending on the domain, one or several deeper recursive models might be tracked, too.

6.4.4 Opponent Modeling

The opponent modeling paradigm is related to the domain of automated game playing. Given the rules of a game like chess or checkers, one might derive (in theory) an optimal strategy ϕ for a player, where ϕ is a function that maps the current state of the game to an action for the player to take in order to maximize (or minimize) the outcome of the game, under the assumption that the other plays optimally, too. In a complex domain,

an opponent model can help to predict the action of an opponent (or, more general, interaction partner), narrowing the set of actions for which an appropriate reaction has to be generated.

The opponent is modeled as an agent that repeatedly decides about its actions based on the perceived state of the world. This decision making process can be modeled in various ways. A simple representation for the decision process of the opponent is a *deterministic finite automaton*, where its action depends on its state, and transitions depend on its perception (Carmel and Markovitch, 1996). However, inference of a finite automaton by *passively* observing its input/output behavior is infeasible (Rivest and Schapire, 1994).

Other models of the opponent include concepts like *recent history adversaries* and *statistical adversaries* as proposed by Freund et al. (1995). A recent history adversary uses boolean formulae over the preceeding history of play to compute its action, while a statistical adversary uses linear functions of simple statistics over the complete history of the play to determine its next action.

However, the approaches from literature presented above share one common drawback: they model only one agent and its interaction with the subject.

6.4.5 Other Approaches

Dousson et al. (1993) describe situations in complex dynamic systems as temporal patterns, i.e. as a set of events related by temporal constraints. They present an approach to recognize occurrences of such situations efficiently and on the fly. Forthcoming events are predicted depending on which situations are possibly emerging.

In automated highway applications, autonomous vehicles surely will have to track the situation of their environment. The approach of Sukthankar (1997) to situation awareness in this domain associates reasoning objects to each object of interest in the vicinity of the vehicle, which are for example other traffic members, the car's self-state (including its desired speed), neighboring lanes, and exit lanes. Decisions for the autonomous vehicle are made by these reasoning objects using a voting scheme. As an example, the self-state might vote for overtaking a preceding slower vehicle while a car in the neighboring lane vetoes against this overtaking.

6.5 Situation Recognition in Crowded Public Areas

Some clues to situation recognition for a robot operating in crowded public areas can be drawn from the previous section. It appears sensible (and maybe necessary) to describe agents' positions and motions in terms of points and directions of interest in the environment (Devaney and Ram, 1997; Intille, 1999). Thereby we are able to reason about an

agent's motion symbolically, avoiding for example the large training effort necessary for other approaches that use low level data for recognition. Furthermore recursive agent tracking (Tambe, 1995) appears important since the robot's animate presence surely influences its environment.

We implemented an approach to detect deliberate obstructions of an autonomous robot that makes use of these two paradigms, and the remainder of this section is dedicated to its description.

6.5.1 Detecting Deliberate Obstructions

When operated in public areas, an autonomous robot attracts the attention of pedestrians passing by. So from time to time some fearless pedestrian approaches the robot and tries to block its path, forcing the robot to perform evasive maneuvers (see Figure 6.1(d)). If the robot is not aware of this situation, it is trapped and unable to accomplish its task.

Region of Interest

In order to recognize these obstructions we identify a relevant region of interest (ROI) such that the motion of an obstructor can be described basically in terms of this region. Clearly this region comprises some area in front of the robot, as only objects in front of the robot really are obstacles.

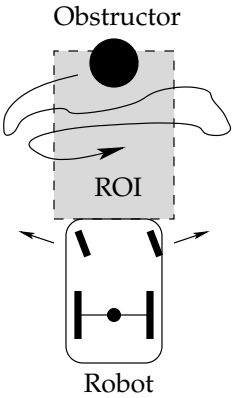


Figure 6.2: Region of Interest (ROI)

We defined the robot's ROI to be a rectangular area immediately in front it (see Figure 6.2). Objects located inside this area but moving at a sufficiently high speed (i.e. 50% of the robot's maximum speed) into the robot's desired direction are treated as if they were outside of this region.

Recursive Agent Tracking

The robot continuously tracks moving objects and humans in its vicinity. It does not reason about any global goal positions of these objects but does note their actual positions relative to its ROI. Hence the robot's model of an opponent for obstruction detection is rather simple: either he or she is located inside the ROI, or not.

Humans located close to the robot can easily estimate its desired direction, as it shows a clear orientation and cannot move sideways or backwards. So the recursive model of an obstructor's model of the robot's local goal (i.e. the direction it intends to drive into) is equal to the robot's actual local goal.

Note that a robot may utilize recursive agent tracking in order to deceive and consequently evade an obstructor. If there are several ways to circumvent an obstructor the robot might choose a plan that it believes to be a behavior that is most unexpected (and thereby hopefully unobstructed) by its opponent. For example the robot might perform a left turn imitating an evasion on the left, but then drive backwards and circumvent a surprised obstructor on the right side. However such deception techniques are not implemented in the current version of our system, as due to the lack of backward sensors the robot is not allowed to drive backwards.

Obstruction Detection

If a human intends to obstruct the robot, he or she will move in front of the robot, i.e. into its ROI, since a human correctly recognizes that this is the only action that might bother the robot. On the other hand humans may cross the robot's region of interest in order to pursue goals of motion that are completely unrelated to the robot. In order to separate these cases the robot has to accumulate evidence about the occurrence of an obstruction. Thus a human has to enter the robot's ROI repeatedly or stay inside this region actively for some time before he or she is recognized as an obstructor. Note that any passive object eventually leaves the ROI (which moves with the robot), since static obstacles are circumvented.

Experimental Results

In our experiments we defined the ROI to be a rectangular area in front of the robot with a width equal to the width of the robot (0.7 m) and a length of 2.0 m. A human is considered to be deliberately obstructing if he or she enters the ROI a third time or actively stays inside the ROI for more than 10 sec. Each time the same agent is considered obstructing the reaction of the system is increased. At first, there is only a spoken notification that the obstruction has been detected. The next time, the system stops for a short period of time and asks the obstructor to let it pass by, the speech output directed to the opponent becoming more and more resolute. Finally, the system gives up.

The system has been tested in our lab environment and in the concourse of the railway station of Ulm. It proved to recognize deliberate obstructions fairly reliably without tending to be over-sensitive. However, a future system should not give up as quickly as our current implementation, but choose an alternative path or employ deception techniques as described above.

6.6 Conclusion

We introduced the problem of situation awareness for autonomous robots operating in crowded public areas and illustrated its importance by means of several examples. A taxonomy of situations among agents in the considered environment was proposed and substantiated with observed examples. Next we presented a collection of approaches related to situation recognition from various domains like action recognition or intention reasoning. Finally, we proposed an approach to detect deliberate obstructions of an autonomous robot, which has been implemented on a robotic wheelchair and tested in public, populated environments.

Chapter 7

Reflective Navigation

7.1 Introduction

Motion planning for a robot in an environment containing obstacles is a fundamental problem in robotics. For the task of navigating a mobile robot among moving obstacles, numerous approaches have been proposed. However, moving obstacles are most commonly assumed to be traveling without having any perception or motion goals (i.e. collision avoidance or goal positions) of their own.

In the expanding domain of mobile service robots deployed in natural, everyday environments, this assumption does not hold, since humans (which are the moving obstacles in this context) do perceive the robot and its motion and adapt their own motion accordingly. Therefore, reflective navigation approaches which include reasoning about other agents' navigational decision processes become increasingly interesting.

In this chapter an approach to reflective navigation is presented which extends the velocity obstacle navigation scheme to incorporate reasoning about other objects' perception and motion goals.

7.1.1 Related Work

Predictive navigation is a domain where a prediction of the future motion of the obstacles is used to yield more successful motion (with respect to travel time or collision avoidance), see for example the work of Foka and Trahanias (2002) and Miura and Shirai (2000). However, reflective navigation approaches are an extension of this concept, since they include further reasoning about perception and navigational processes of moving obstacles.

The velocity obstacle paradigm, which belongs to the class of predictive navigation schemes, has been presented by Fiorini and Shiller (1998) for obstacles moving on straight

lines, and has been extended by Shiller et al. (2001) for obstacles moving on arbitrary (but known) trajectories.

Modeling other agents' decision making similar to the own agent's decision making is used by the recursive agent modeling approach (Gmytrasiewicz, 1992), where the own agent bases its decisions not only on its models of other agents' decision making processes, but also on its models of the other agents' models of its own decision making, and so on (hence the label *recursive*).

7.1.2 Overview

Assume a robot B uses deterministic velocity obstacles (as presented in Section 3.3) for its navigation. As we have seen, there is some freedom in choice of avoiding velocities. That is, a unique velocity $\mathbf{v_B} \in \mathbb{R}^2$ cannot be an adequate prediction of the future velocity of B. Therefore, if velocity obstacles are used in a recursive manner (as it is the objective of this chapter), they have to be extended in a way which allows to express uncertainty about the velocity of the obstacles, i.e. by using (possibly multi-modal) probability distributions. Such a probabilistic extension of the velocity obstacle approach is presented in Section 7.2.

Being able to cope with uncertain obstacle velocities, Section 7.3 describes how to apply the velocity obstacle scheme recursively in order to create a reflective navigation behavior. The proposed method is evaluated for a collection of simulated in Section 7.4, and finally concluded after discussing the presented work.

7.2 Probabilistic Velocity Obstacles

Let B_i and B_j be circular objects with radii r_i and r_j, placed at positions $\mathbf{c_i} \in \mathbb{R}^2$ and $\mathbf{c_j} \in \mathbb{R}^2$, as in the deterministic velocity obstacle approach.

However, now we will consider uncertainty in shape and velocity of the objects. This allows to reflect the limitations of real sensors and object tracking techniques.

7.2.1 Shape Uncertainty

A first source of uncertainty is the actual shape of the obstacles. With real sensors, there will always be measurement errors, which should be reflected by the navigation approach.

It turns out that giving an adequate probabilistic counterpart to the definition of deterministic rigid bodies in Definition 3.1 is not straightforward. One possible way of a definition would be to model uncertainty about the actual shape of a rigid body B by a

function

$$PB : \mathbb{R}^2 \to [0, 1] \tag{7.1}$$

where $PB(\mathbf{p})$ is interpreted as the probability of point $\mathbf{p}$ belonging to B. However, we would have to specify dependencies between the points, too, which will become clear after the following definition of a simple class of "probabilistic" objects.

Definition 7.1 (Disc with Uncertain Radius)
A **disc with uncertain radius** $D(a, b)$ *is a disc centered at the origin whose radius is a variate* R *with range* $[a, b]$ *and*

$$P(R \leq r) = \begin{cases} 0, & \text{if } r < a, \\ \frac{r-a}{b-a} & \text{if } r \in [a, b], \text{ and} \\ 1, & \text{if } r > b. \end{cases} \tag{7.2}$$

For the sake of brevity, we may call a disc with uncertain radius **probabilistic disc** *or* **p-disc***, too.*

Note that discs with uncertain radius cannot be represented by a mapping as in Equation 7.1, since for example for a p-disc B we may have

$$0 \; < \; P((0, r) \in B) = P((r, 0) \in B) < 1,$$

but

$$\begin{aligned} P((0, r) \in B \wedge (r, 0) \in B) &= P((0, r) \in B) \\ &\neq P((0, r) \in B) \cdot P((r, 0) \in B). \end{aligned}$$

The reason is that, as announced, statistical dependencies are not taken into account properly when uncertain shapes are modeled by mappings according to Equation 7.1.

Therefore we will focus on p-discs as probabilistic objects in the following. This is not a severe restriction, since the remainder of this chapter remains valid after changing the definition of probabilistic objects, provided that the definition of probabilistic collision cones is adapted accordingly.

Definition 7.2 (Placed Disc with Uncertain Radius)
The ordered pair $(D(a, b), \mathbf{c})$ *of a p-disc* $D(a, b)$ *and a position* $\mathbf{c} \in \mathbb{R}^2$ *is called a* **placed disc with uncertain radius**.

The ordered triple $(D(a, b), \mathbf{c}, \mathbf{v})$ *of a p-disc* $D(a, b)$*, a position* $\mathbf{c} \in \mathbb{R}^2$*, and a velocity* $\mathbf{v} \in \mathbb{R}^2$ *is called a* **moving disc with uncertain radius***, and the point* $\mathbf{c} + \mathbf{v} \cdot t$ *is called its* **position at time** t.

Property 7.3 (Collision of Discs with Uncertain Radius)
Let $(D(a_i, b_i), \mathbf{c_i})$ *and* $(D(a_j, b_j), \mathbf{c_j})$ *be placed p-discs with variates* R_i *and* R_j *representing their radii. Then, the placed p-discs are colliding if* $R_i + R_j \leq |\mathbf{c_i} - \mathbf{c_j}|$.

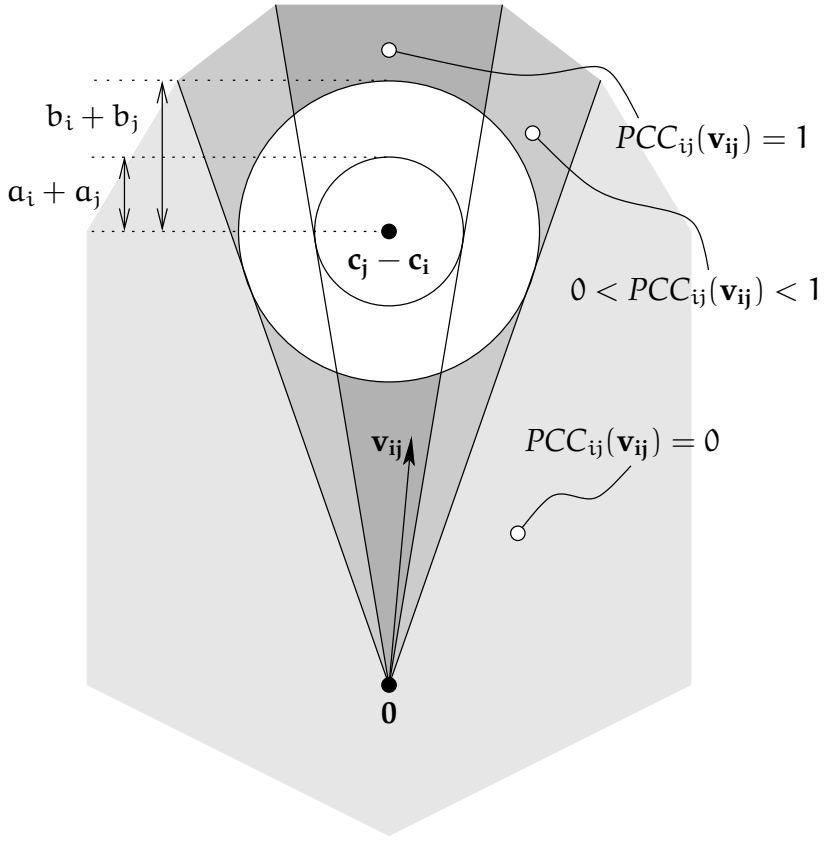


Figure 7.1: Probabilistic collision cone of two discs $(D(a_i, b_i), \mathbf{c_i})$ and $(D(a_j, b_j), \mathbf{c_j})$ with uncertain radii

In the deterministic velocity obstacle approach, the collision cone of an ordered pair of moving objects is a set of relative velocities which lead to a collision. If the shapes of the objects are uncertain, e.g. the radius of a circular objects is only known up to some error, all we can expect as a probabilistic collision cone is a mapping which assigns collision probabilities to relative velocities.

Definition 7.4 (Probabilistic Collision Cone)
The **probabilistic collision cone** *of an ordered pair of placed discs* $(D(a_i, b_i), \mathbf{c_i})$ *and* $(D(a_j, b_j), \mathbf{c_j})$ *with uncertain radii* R_i *and* R_j *is a mapping* $PCC_{ij} : \mathbb{R}^2 \to [0, 1]$ *with*

$$PCC_{ij} : \mathbf{v_{ij}} \mapsto P\left(R_i + R_j \geq \min_{t \in \mathbb{R}_0^+} |\mathbf{c_i} + \mathbf{v_{ij}} \cdot t - \mathbf{c_j}|\right), \tag{7.3}$$

that is, in words, $PCC_{ij}(\mathbf{v_{ij}})$ *is the probability of* $(D(a_i, b_i), \mathbf{c_i} + \mathbf{v_{ij}} \cdot t)$ *colliding with* $(D(a_j, b_j), \mathbf{c_j})$ *for some* $t \in \mathbb{R}_0^+$.

As an example, Figure 7.1 shows the probabilistic collision cone of two discs with uncertain radii.

7.2.2 Velocity Uncertainty

Another source of uncertainty is the motion of the obstacles. In fact, we are confronted with two types of uncertainty here, one which stems from the measurement errors of the sensor system, and another one which stems from unpredictable changes of the motion of the obstacles. Therefore we represent the uncertain velocity of object B_j as a probability density function

$$V_j : \mathbb{R}^2 \to \mathbb{R}_0^+. \tag{7.4}$$

Given such a blurred velocity V_j of a placed p-disc $D_j = (D(a_j, b_j), \mathbf{c_j})$, we may ask for the collision probability with respect to a moving p-disc $D_i = (D(a_i, b_i), \mathbf{c_i}, \mathbf{v_i})$ representing the robot, which leads us to a probabilistic formulation of a velocity obstacle as a function

$$PVO_{ij} : \mathbb{R}^2 \to [0, 1] \tag{7.5}$$

which maps absolute velocities $\mathbf{v_i}$ of B_i to the according probability of colliding with D_j. Assume D_j moves with velocity $\mathbf{v_j} \in \mathbb{R}^2$. Then, the probability of a collision between D_i and D_j is $PCC_{ij}(\mathbf{v_i} - \mathbf{v_j})$. Since the velocity of D_j is uncertain, we have to weigh that collision probability with $V_j(\mathbf{v_j})$, the probability density at $\mathbf{v_j}$. Integrating over all possible velocities $\mathbf{v_j}$ of D_j delivers

$$PVO_{ij}(\mathbf{v_i}) = \int_{\mathbb{R}^2} V_j(\mathbf{v_j}) PCC_{ij}(\mathbf{v_i} - \mathbf{v_j}) d^2\mathbf{v_j}, \tag{7.6}$$

which is equivalent to

$$PVO_{ij} = V_j * PCC_{ij} \tag{7.7}$$

where $*$ denotes the convolution of two function.

When a moving p-disc D_i is confronted with a set of moving p-discs $\mathcal{B} = \{D_j \mid 1 \leq j \leq n, i \neq j\}$, the probability of D_i colliding with any other obstacle $D_j \in \mathcal{B}$ equals the probability of not avoiding collisions with any other moving obstacle. Here and in the remainder of this chapter we will assume that the velocities of the moving obstacles are statistically independent. Therefore, the function $PVO_i : \mathbb{R}^2 \to [0, 1]$ with

$$PVO_i(\mathbf{v_i}) = 1 - \prod_{j \neq i,\, D_j \in \mathcal{B}} \left(1 - PVO_{ij}(\mathbf{v_i})\right). \tag{7.8}$$

assigns to a velocity $\mathbf{v_i}$ of D_i the probability of colliding with any other p-disc from $\mathcal{B}$. That is, PVO_i is the probabilistic counterpart of the composite velocity obstacle.

7.2.3 Navigating with Probabilistic Velocity Obstacles

In the deterministic case, navigating is rather easy since we consider only collision free velocities and can choose a velocity which is optimal for reaching the goal. But here,

we have to balance two objectives: reaching a goal and minimizing the probability of a collision.

Let $U_i : \mathbb{R}^2 \to [0,1]$ be a function representing the utility of velocities $\mathbf{v_i}$ for the motion goal of D_i. However, the full utility of a velocity $\mathbf{v_i}$ is only attained if (a) $\mathbf{v_i}$ is reachable, and (b) $\mathbf{v_i}$ is collision free. Therefore we define the relative utility function

$$RU_i = U_i^{\alpha} \cdot R_i^{\beta} \cdot (1 - PVO_i)^{\gamma}, \tag{7.9}$$

where $R_i : \mathbb{R}^2 \to [0,1]$ describes the reachability of a new velocity, i.e. it corresponds to the set *RV* of reachable velocities in the deterministic velocity obstacle approach. This uncertainty about the reachability of a new velocity may also take into account the uncertainty about the robot's own current velocity. The exponents $\alpha, \beta, \gamma \in \mathbb{R}^+$ are weights for the three factors of the relative utility.

A simple navigation scheme for p-disc D_i based on probabilistic velocity obstacles can be obtained by periodically choosing a velocity $\mathbf{v_i}$ which maximizes the relative utility RU_i. In order to implement this approach, the use of continuous functions has to be replaced by discretized version, and explicitly represented functions have to be restricted to a finite size.

Discretization

Step functions $s : \mathbb{R}^2 \to \mathbb{R}$ with $s(x,y) = s(x',y')$ for $i\kappa \le x, x' < (i+1)\kappa$ and $j\kappa \le y, y' < (j+1)\kappa$ are used for discretization of continuous (in the sense of non-discrete) functions. In other words we use functions which are piecewise constant on squares of size $\kappa \times \kappa$, where κ is a predefined constant.

For a point $p = (x,y) \in \mathbb{R}^2$, its discretization is

$$\mathit{discr}(p) = \mathsf{p} = \left(\left\lfloor \frac{x}{\kappa} \right\rfloor, \left\lfloor \frac{y}{\kappa} \right\rfloor \right) \in \mathbb{Z}^2. \tag{7.10}$$

Conversely, for a discretized point $\mathsf{p} = (z_1, z_2) \in \mathbb{Z}^2$ we define its cell as

$$\begin{aligned} \mathit{cell}(z_1,z_2) &= \{p \in \mathbb{R}^2 \mid \mathit{discr}(p) = (z_1,z_2)\} \\ &= [z_1\kappa, (z_1+1)\kappa) \times [z_2\kappa, (z_2+1)\kappa)\,. \end{aligned} \tag{7.11}$$

For any function $F : \mathbb{R}^2 \to [0,1]$ we define the discretization of F to be the function $\mathsf{F} : \mathbb{Z}^2 \to [0,1]$ with

$$\mathsf{F}(z_1,z_2) = \frac{1}{\kappa^2} \int_{\mathit{cell}(z_1,z_2)} F(x,y)\,dx\,dy, \tag{7.12}$$

i.e. $\mathsf{F}(z_1,z_2)$ is the average of F on $\mathit{cell}(z_1,z_2)$. However, in practise we will only require $\mathsf{F}(z_1,z_2) \in F(\mathit{cell}(z_1,z_2))$ for F to be called a discretization of F, since the computation of the integral is expensive and not negligible. A simple extension to overcome potential difficulties would be to draw a constant number n of random points p_i

from *cell*(p) and use the average value $\frac{1}{n}\sum F(p_i)$ as value for $\mathsf{F}(\mathsf{p})$, approaching the exact value for $n \to \infty$. A more thorough treatment of this problem involves sampling theory, i.e. an analysis of the spectrum of F and the selection of κ according to Shannon's sampling theorem, and goes beyond the scope of this thesis. We will call a function F a *strict* discretization of F, if it fulfills Equation 7.12, and otherwise assume that the value of κ is adequate for F.

Finally, for a discretized function $\mathsf{F} : \mathbb{Z}^2 \to \mathbb{R}_0^+$ the set

$$\sigma(\mathsf{F}) = \{(x, y) \in \mathbb{Z}^2 \,|\, \mathsf{F}(x, y) > 0\} \tag{7.13}$$

is called the *supporting set* of F, which is the set of cells on which the discretized function does not vanish. The property

$$\sigma(\mathsf{FG}) = \sigma(\mathsf{F}) \cap \sigma(\mathsf{G}) \tag{7.14}$$

is easily shown. Furthermore,

$$\sum_{\mathsf{p}\in\mathbb{Z}^2} \mathsf{F}(\mathsf{p})\kappa^2 = \int_{\mathbb{R}^2} F(\mathbf{p})\,d^2\mathbf{p} \tag{7.15}$$

holds for strict discretizations.

Restriction

Now we discuss the restriction problem in the context of navigating a p-disc D_i. Assuming that the velocity of any other p-disc D_j is bounded or is known with bounded error, the supporting set $\sigma(\mathsf{V}_j)$ is finite. Therefore, $\mathsf{PVO}_{ij}(\mathsf{v}_i)$ can be computed for any $\mathbf{v_i}$ by using

$$\mathsf{PVO}_{ij}(\mathsf{v}_i) = \sum_{\mathsf{v}_j\in\sigma(\mathsf{V}_j)} \mathsf{V}_j(\mathsf{v}_j)\mathsf{PCC}_{ij}(\mathsf{v}_i - \mathsf{v}_j)\kappa^2, \tag{7.16}$$

which is the discrete version of Equation 7.6. The unbounded probabilistic collision cones PCC_{ij} have to be represented implicitly by a subroutine which computes the respective collision probabilities on demand.

Furthermore, for any real (i.e. physical) p-disc D_i, the set $\sigma(\mathsf{R}_i)$ describing reachable velocities is finite, as any bounded acceleration applied to a body of non-zero mass for a bounded period of time results in a bounded change of velocity. Since only velocities from $\sigma(\mathsf{RU}_i)$ will be considered for navigating D_i, and since $\sigma(\mathsf{RU}_i) \subseteq \sigma(\mathsf{R}_i)$, we can restrict velocities to the finite domain $\sigma(\mathsf{RU}_i)$.

Algorithm

Combining the results from the previous subsections, we get

$$\mathsf{PVO}_i = 1 - \prod_{j\neq i} (1 - \mathsf{PVO}_{ij}) \tag{7.17}$$

Algorithm 6 RELATIVE UTILITY

1: **input:** a set of placed p-discs $\mathcal{B} = \{D_i = (D(a_i, b_i), c_i) \mid i = 1, 2, \ldots, n\}$
2: **input:** uncertain velocities $V_i : \mathbb{R}^2 \to [0, 1]$ for each p-disc $D_i \in \mathcal{B}$
3: **input:** a function $U_i : \mathbb{R}^2 \to [0, 1]$ describing utility of velocities
4: **input:** a function $R_i : \mathbb{R}^2 \to [0, 1]$ describing reachable velocities
5: **input:** a function $\mathsf{PCC} : \mathbb{N} \times \mathbb{N} \times \mathbb{Z}^2 \to [0, 1]$ with $\mathsf{PCC}(i, j, v_{ij}) = \mathsf{PCC}_{ij}(v_{ij})$
6: **input:** the index i of the p-disc representing the robot
7: **for** $v_i \in \sigma(R_i)$ **do**
8: $\quad \mathsf{RU}_i(v_i) \leftarrow U_i^{\alpha}(v_i) \cdot R_i^{\beta}(v_i)$
9: $\quad$ **for** $j \in \{1, 2, \ldots, n\} - \{i\}$ **do**
10: $\quad\quad \mathsf{PVO}_{ij}(v_i) \leftarrow 0$
11: $\quad\quad$ **for** $v_j \in \sigma(V_j)$ **do**
12: $\quad\quad\quad \mathsf{PVO}_{ij}(v_i) \leftarrow \mathsf{PVO}_{ij}(v_i) + V_j(v_j) \cdot \mathsf{PCC}_{ij}(v_i - v_j) \cdot \kappa^2$
13: $\quad\quad$ **end for**
14: $\quad\quad \mathsf{RU}_i(v_i) \leftarrow \mathsf{RU}_i(v_i) \cdot (1 - \mathsf{PVO}_{ij}(v_i))^{\gamma}$
15: $\quad$ **end for**
16: **end for**
17: **return** RU_i

and further

$$\begin{aligned} \mathsf{RU}_i(v_i) &= U_i(v_i) R_i(v_i) (1 - \mathsf{PVO}_i(v_i)) \\ &= U_i(v_i) R_i(v_i) \prod_{j \neq i} (1 - \mathsf{PVO}_{ij}(v_i)) \end{aligned} \tag{7.18}$$

for any $v_i \in \sigma(\mathsf{RU}_i)$. This observation is summarized in Algorithm 6, too.

Assuming that $\mathsf{PCC}_{ij}(v_i)$, $R_i(v_i)$, and $U_i(v_i)$ can be computed in time $\mathcal{O}(1)$ for $v_i \in \mathbb{Z}^2$, we can compute $\mathsf{PVO}_{ij}(v_i)$ in time $\mathcal{O}(|\sigma(V_j)|)$ (according to Equation 7.16 and lines 10–13 in Algorithm 6), and $\mathsf{RU}_i(v_i)$ in time $\mathcal{O}(\sum_{j \neq i} |\sigma(V_j)|)$ (according to Equation 7.18 and lines 8–15 in Algorithm 6). Finally, a discrete velocity v_i maximizing RU_i can be found in time

$$\mathcal{O}\left(|\sigma(R_i)| \cdot \sum_{j \neq i} |\sigma(V_j)| \right), \tag{7.19}$$

integrating the search into the loop from line 7 to line 16 in Algorithm 6. That is, the dependence of the running time on the number of obstacles is only linear, but the dependence on the discretization is $\mathcal{O}(1/\kappa^4)$.

7.3 Recursive Probabilistic Velocity Obstacles

Traditionally, when navigating a mobile robot among moving obstacles (like humans), these obstacles' abilities to avoid collisions and their resulting motion behaviors are not taken into account. In contrast to this plain obstacle modeling, recursive modeling approaches presume the opponents (or more generally, the interaction partners) to apply decision making processes for navigation similar or equivalent to the own process. In the given context of mobile robot navigation, this means to put the robot into the position of its obstacles, let it reason about their decisions and then integrate the resulting insight into its own decision making. We will call such intelligent moving obstacles (or, obstacles which are considered intelligent) *agents*. Furthermore, we will consider a finite set of agents $\mathcal{B} = \{D_i = (D(a_i, b_i), \mathbf{c_i}) \mid i = 1, 2, \ldots, n\}$ with uncertain velocity V_i for each $D_i \in \mathcal{B}$ for the remainder of this section.

7.3.1 Agent Modeling

Agents are assumed to perceive their environment and deduce according reactions, the reasoning process being similar to that of the robot. That is, any agent D_j is assumed to take actions maximizing its relative utility function RU_j. Therefore, in order to predict the action of agent D_j, we need to know its current utility function U_j, reachable velocities R_j, and velocity obstacle PVO_j.

The utility of velocities can be inferred by recognition of the current motion goal of the moving obstacle. For example, Bennewitz et al. (2002) learn and recognize typical motion patterns of humans. If no global motion goal is available through recognition, one can still assume that there exists such a goal which the agent strives to approach, expecting it to be willing to keep its current speed and heading. By continuous observation of a moving agent it is also possible to deduce a model of its dynamics, which describes feasible accelerations depending on its current speed and heading. These two problems are beyond the scope of this thesis and will not be addressed in detail in the following.

The remaining problem is the computation of a probabilistic velocity obstacle for an agent D_j, and this requires to presume assumptions on the velocities of the other moving agents D_k, $k \neq j$, to agent D_j. In principle, we can base assumptions on the future velocities of an agent on its probabilistic velocity obstacle again and again. This is a recursive description, hence these probabilistic velocity obstacles will be called *recursive probabilistic velocity obstacles*, and will be abbreviated as "RPVO."

However, at some point the recursion has to be terminated, i.e. the velocity obstacle must be based on perceived velocities. Therefore, we may distinguish different levels or depths of recursion, denoted by superscript d as in $PVO_i^{(d)}$ for agent D_i, such that $PVO_i^{(1)}$ is based on perceived velocities of the other agents, and $PVO_i^{(d)}$ for $d > 1$ is based on velocities of the other agents deduced using probabilistic velocity obstacles of recursive

depth $d-1$.

Of course this recursive modeling is not restricted to any constant depth of recursion by a matter of principle. However, computational demands will increase with the depth of the recursion, and intuitively, one does not expect recursion depths of more than three or four to be of broad practical value, since such deeper modeling is not likely to happen when we are walking as human beings among other humans.

Note that accurate recursive models of moving agents are prerequisite for more sophisticated reflective navigation approaches in order to be able to deceive and feint particularly malevolent agents like deliberate obstructors. However being dreams of the future, such potential abilities indicate the importance of reflective navigation approaches and their investigation.

7.3.2 Formal Representation

In order to interact with their surroundings, intelligent agents create models of their environment. If this environment contains other agents, these can become part of the model, as well as these agents' models of the environment and so forth. This section presents a formal representation of recursive models in the given context, which serves as a basis for the implementation later on.

Definition 7.5 (Models of Functions by Agents, Interpretation of Models)
Let F *be the symbol of a function from* $\mathbb{R}^2$ *to* $[0,1]$*. Then, the symbol* $\mu_i[F]$ *denotes a function from* $\mathbb{R}^2$ *to* $[0,1]$ *and is verbalized as* **model of** F **by agent** i*.*

An **interpretation** $\mathcal{I}$ *assigns functions to symbols* $\mu_i[F]$*, that is,* $\mathcal{I}(\mu_i[F]) : \mathbb{R}^2 \to [0,1]$*.*

Informally, we denote by $\mu_i[F]$ the current knowledge of agent i about an entity F. For example, if $R_i : \mathbb{R}^2 \to [0,1]$ is the function which specifies the reachability of velocities for an agent i, we will denote by $\mu_j[R_i]$ the function specifying the reachability of velocities as attributed to agent i by agent j.

Using these symbols, we can now express the basic principle of recursive agent modeling in the context of probabilistic velocity obstacle navigation as follows. Each agent assumes that the others will choose their velocity according to their relative utility function, that is

$$\mu_i[V_j^{(d)}] = \begin{cases} \frac{1}{w}\mu_i[RU_j^{(d)}] & \text{if } d > 0 \text{ and } w := \int RU_j^{(d)}\, d^2\mathbf{v} > 0, \\ \mu_i[V_j] & \text{else.} \end{cases} \tag{7.20}$$

Note that $V_j^{(d)}$ is a probability density, that is

$$\int_{\mathbb{R}^2} V_j(\mathbf{v_j})^{(d)}\, d^2\mathbf{v_j} = 1,$$

but $RU_j^{(d)}$ is a $[0,1]$-valued function with bounded support, that is

$$0 \leq w := \int_{\mathbb{R}^2} RU_j^{(d)}(\mathbf{v_j})\, d^2\mathbf{v_j} < \infty.$$

This is the reason for the scaling factor $\frac{1}{w}$ in the first case of Equation 7.20, and the second case in that equation terminates the recursion for $d = 0$ or is a fallback position for $w = 0$.

For a recursive depth $d = 0$, no reflection about the other agents' motion is assumed, and therefore the relative utility $RU_i^{(0)}$ will depend only on the utility U_i of reachable velocities as indicated by R_i. For a recursive depth $d > 0$, the relative utility $RU_i^{(0)}$ of an agent i depends on its probabilistic velocity obstacle $PVO_i^{(d)}$, too. Together, we have

$$RU_i^{(d)} = \begin{cases} U_i^{\alpha} \cdot R_i^{\beta} & \text{if } d = 0, \\ U_i^{\alpha} \cdot R_i^{\beta} \cdot \left(1 - PVO_i^{(d)}\right)^{\gamma} & \text{else,} \end{cases} \tag{7.21}$$

with weights $\alpha, \beta, \gamma \in \mathbb{R}^+$.

The actual reflection appears in the specification of the recursive probabilistic velocity obstacle $PVO_i^{(d)} : \mathbb{R}^2 \to [0,1]$ of depth d for agent i, since this entity depends on the (recursive) model of other agents' velocities $\mu_i[V_j^{(d-1)}]$ and is defined as

$$PVO_i^{(d)} = 1 - \prod_{j \neq i} \left(1 - \mu_i[V_j^{(d-1)}] * PCC_{ij}\right), \tag{7.22}$$

which completes our specification of RPVO.

Before any utility $RU_i^{(d)}(\mathbf{v})$ for $\mathbf{v} \in \mathbb{R}^2$ can be computed, we have to specify an interpretation of symbols $\mu_i[F]$ for function symbols F, which will be given in a recursive way by a set of rules, and two sets of rules will distinguished. Motivation for the first set stems from the given context of reflective navigation. The second set of rules stems from our assumptions on the way how the agents acquire information about each other, and is more or less specific to a certain implementation.

The first set of interpretation rules is defined as follows.

Definition 7.6 (Interpretation of RPVO Function Models)
Let F *be a symbol for a function from* $\mathbb{R}^2$ *to* $[0,1]$. *Then,* F *will be interpreted as follows*

$$\mathcal{I}(F) = \begin{cases} \mathcal{I}(\mu_i[G]) & \text{if } F = \mu_i[\mu_i[G]], \\ \mathcal{I}(\mu_i[G])\, op\, \mathcal{I}(\mu_i[H]) & \text{if } F = \mu_i[G\, op\, H] \text{ with } op \in \{+, \cdot, *\}, \\ \mathcal{I}(\mu_i[G])^{\alpha} & \text{if } F = \mu_i[G^{\alpha}] \text{ with } \alpha \in \mathbb{R}, \\ \mathcal{I}(C) & \text{if } F = \mu_i[C] \text{ and } C \text{ symbolizes a constant function}, \\ U_i & \text{if } F = \mu_i[U_i], \\ R_i & \text{if } F = \mu_i[R_i], \\ V_i & \text{if } F = \mu_i[V_i], \\ PCC_{ij} & \text{if } F = \mu_i[PCC_{ij}], \text{ and} \\ F & \text{if } F \text{ is not of the shape } \mu_i[G], \end{cases} \tag{7.23}$$

for $i, j \in \{1, 2, \ldots, n\}$.

The first rule, $I(\mu_i[\mu_i[G]]) = \mathcal{I}(\mu_i[G])$, is motivated by the assumption that an agent "knows what it knows," i.e. its model of its model of an entity is the model of that entity.

The second and the third rule are motivated by the assumption that all agents use the same approach for decision making, i.e. they perform the same operations to compute a certain function.

The remaining rules terminate the interpretation, either for a symbol of a constant function (e.g. "1"), or when an agent models itself, since we assume that each agent has accurate information about itself, or when no modeling is involved.

For $d > 1$, and $w_j := \int RU_j^{(d-1)}\, d^2\mathbf{v} > 0$ for $j \neq i$, we get

$$RU_i^{(d)} = U_i^{\alpha} R_i^{\beta} \prod_{j \neq i} \left(1 - \frac{1}{w_j} \mu_i[RU_j^{(d-1)}] * PCC_{ij}\right)^{\gamma}, \tag{7.24}$$

from Equations 7.20–7.22, and with the rules from 7.6 follows

$$\begin{aligned} \mu_i[RU_j^{(d)}] &= \mu_i \left[U_j^{\alpha} R_j^{\beta} \prod_{k \neq j} \left(1 - \frac{1}{w_k} \mu_j[RU_k^{(d)}] * PCC_{jk}\right)^{\gamma} \right] \\ &= \mu_i[U_j]^{\alpha} \mu_i[R_j]^{\beta} \prod_{k \neq j} \left(1 - \frac{1}{w_k} \mu_i[\mu_j[RU_k^{(d)}]] * \mu_i[PCC_{jk}]\right)^{\gamma}, \end{aligned} \tag{7.25}$$

that is, modeling is propagated towards the primitive (i.e. not composed) functions U_i, R_i, V_i, and PCC_{ij}. Furthermore, the number of models $\mu_{i_1}[\ldots \mu_{i_d}[F] \ldots]$ of primitive functions occurring in a full expansion of $RU_i^{(d)}$ may increase exponentially with the recursive depth d, depending on their interpretation.

Interpretation under Equal Information

As seen above, we must specify interpretations of (recursive) models of the functions U_j, R_j, V_j and PCC_{jk} in order to evaluate a relative utility $RU_i^{(d)}$. That is, we must say what agents assume or know about other agents' perception, intention, and reachable velocities.

As a first simple approach, we will assume equal information among the agents. That is, no agent "knows more" or has a "more accurate model" of an entity than an other agent. This is expressed technically in the following definition.

Definition 7.7 (Interpretation of RPVO Function Models under Equal Information) *We say* **all agents have equal information**, *iff*

$$\mathcal{I}(\mu_i[F]) = \mathcal{I}(\mu_j[F]) \tag{7.26}$$

for agents i *and* j, *and* F *symbolizing a function from* $\mathbb{R}^2$ *to* $[0, 1]$.

If all agents have equal information, any recursive model $\mu_{i_1}[\dots\mu_{i_k}[F]\dots]$ collapses to a simple model $\mu_i[F]$ for any $i_1, \dots, i_k, i$:

$$\begin{aligned}\mathcal{I}(\mu_{i_1}[\mu_{i_2}[\dots\mu_{i_k}[F]\dots]]) &= \mathcal{I}(\mu_{i_2}[\mu_{i_2}[\dots\mu_{i_k}[F]\dots]]) \\ &= \mathcal{I}(\mu_{i_2}[\dots\mu_{i_k}[F]\dots]) \\ &\dots \\ &= \mathcal{I}(\mu_{i_k}[F]) \\ &= \mathcal{I}(\mu_i[F]),\end{aligned} \tag{7.27}$$

and with Definition 7.6 we have

$$\mathcal{I}(\mu_{i_1}[\dots\mu_{i_k}[F]\dots]) = F, \text{ for } F \in \{U_i, R_i, V_i, PCC_{ij}\} \tag{7.28}$$

and any agent $i_1, \dots, i_k, i, j$. Consequently, when agents have equal information, we do not reason about mutual perception but on relative positions and velocity selections only. Furthermore, relative utilities $RU_i^{(d)}(\mathbf{v_i})$ of velocities $\mathbf{v_i}$ for an agent i at a recursive depth $d > 0$ can now be computed efficiently using dynamic programming.

7.3.3 Implementation

For the implementation we assume equal information among the agents as defined above. With this simplification, the dependence of the complexity on the recursion depth is reduced to linear, since the number of models to be computed is equal on each level of recursion. Algorithm 7 gives the details of the used dynamic programming approach

Algorithm 7 RECURSIVE RELATIVE UTILITY

input: a set of placed p-discs $\mathcal{B} = \{D_i = (D(a_i, b_i), \mathbf{c_i}) \mid i = 1, 2, \ldots, n\}$
input: uncertain velocities $V_i : \mathbb{R}^2 \to [0, 1]$ for each $D_i \in \mathcal{B}$
input: functions $U_i : \mathbb{R}^2 \to [0, 1]$ describing utility of velocities for each $D_i \in \mathcal{B}$
input: functions $R_i : \mathbb{R}^2 \to [0, 1]$ describing reachable velocities for each $D_i \in \mathcal{B}$
input: a function $\mathsf{PCC} : \mathbb{N} \times \mathbb{N} \times \mathbb{Z}^2 \to [0, 1]$ with $\mathsf{PCC}(i, j, v_{ij}) = \mathsf{PCC}_{ij}(v_{ij})$
input: the desired recursive depth $r \in \mathbb{N}$
for $i = 1, \ldots, n$ **do**
  $V_i^{(0)} \leftarrow \mathit{discr}(V_i)$
  $\mathsf{RU}_i^{(0)} \leftarrow \mathit{discr}(U_i R_i)$
end for
for $d = 1, \ldots, r$ **do**
  for $i = 1, \ldots, n$ **do**
    $\mathsf{RU}_i^{(d)} \leftarrow$ RELATIVE UTILITY as in Algorithm 6 for
      • p-discs $\mathcal{B}$,
      • uncertain velocities $V_j^{(d-1)}$ for each $D_j \in \mathcal{B}$,
      • functions U_j and R_j for each $D_j \in \mathcal{B}$,
      • the function PCC, and
      • considering D_i as the robot.
    $w \leftarrow \kappa^2 \cdot \sum_{v \in \sigma\left(\mathsf{RU}_i^{(d)}\right)} \mathsf{RU}_i^{(d)}(v)$
    if $w > 0$ **then**
      $V_i^{(d)} \leftarrow \frac{1}{w} \mathsf{RU}_i^{(d)}$
    else
      $V_i^{(d)} \leftarrow V_i^{(0)}$
    end if
  end for
end for
output: relative utilities $\mathsf{RU}_i^{(d)}$ for each $D_i \in \mathcal{B}$ and $d = 0, 1, \ldots, r$
output: uncertain velocities $V_i^{(d)}$ for each $D_i \in \mathcal{B}$ and $d = 0, 1, \ldots, r$

Complexity

We begin the complexity assessment by measuring the sizes of the supporting sets of the discretized functions used in Algorithm 7, where line 9 implies

$$\sigma(\mathsf{RU}_i^{(0)}) \subseteq \sigma(\mathsf{R}_i), \tag{7.29}$$

and from line 13 follows

$$\sigma(\mathsf{RU}_i^{(d)}) \subseteq \sigma(\mathsf{R}_i) \tag{7.30}$$

for $d > 0$. Line 8 implies

$$\sigma(\mathsf{V}_i^{(0)}) = \sigma(\mathsf{V}_i), \tag{7.31}$$

and from lines 16 and 18 follows

$$\sigma(\mathsf{V}_i^{(d)}) \subseteq \sigma(\mathsf{R}_i) \cup \sigma(\mathsf{V}_i) \tag{7.32}$$

for $d > 0$, using the three preceding Equations.

Now we count the numbers of operations used in the algorithm, which we write down using $N_i := |\sigma(\mathsf{R}_i) \cup \sigma(\mathsf{V}_i)|$ as an abbreviation. Line 13 requires $\mathcal{O}(N_i \cdot \sum_{j \neq i} N_j)$ operation (cf. Equation 7.19). Lines 14, 16, and 18 require $\mathcal{O}(N_i)$ operations each, and are thus dominated by line 13. Therefore the loop from line 12 to line 20 requires

$$\mathcal{O}(\sum_{i=1}^{n} (N_i \sum_{j \neq i} N_j)) \tag{7.33}$$

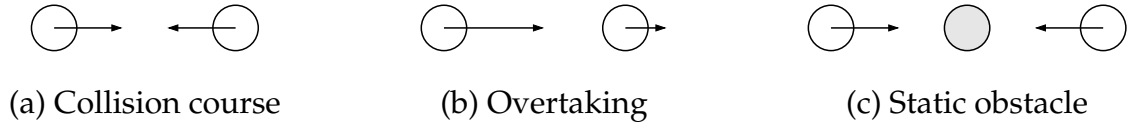
operations, and the loop from line 11 to line 21 requires

$$\mathcal{O}(r \sum_{i=1}^{n} (N_i \sum_{j \neq i} N_j)) \tag{7.34}$$

operations. The complexity of the loop from line 11 to 21 clearly dominates the complexity of the initialization loop from line 7 to 10. Therefore Equation 7.34 gives an upper bound of the overall time complexity of our implementation. That is, the dependence on the maximum recursive depth is linear, the dependence on the number of objects is $\mathcal{O}(n^2)$, and the dependence on the discretization remains $\mathcal{O}(1/\kappa^4)$.

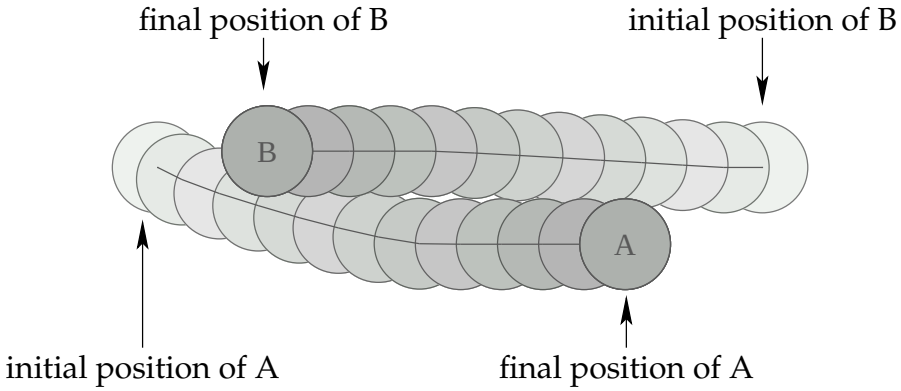
7.4 Results

The approach has been evaluated in different simulated situations, including (a) two objects on a collision course, (b) a faster object approaching and overtaking a slower object, and (c) two objects encountering each other close to a static obstacle, see Figure 7.2.

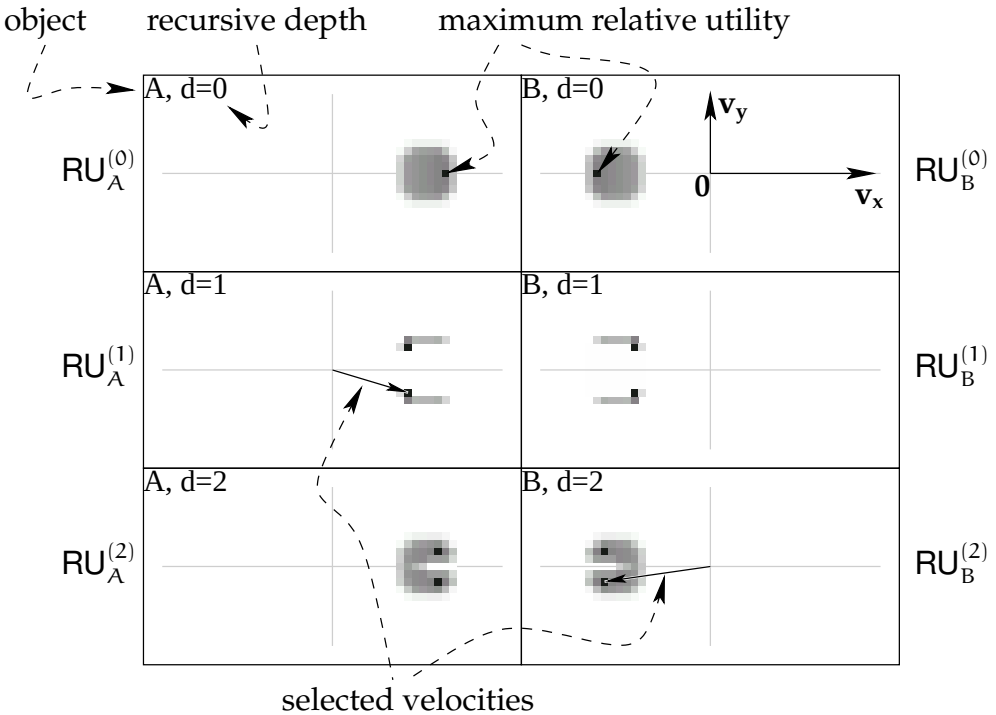


(a) Collision course (b) Overtaking (c) Static obstacle

Figure 7.2: Situations for RPVO simulation



(a) Resulting motion



(b) Relative utilities and velocity selection

Figure 7.3: Legend for simulation results

For each situation, varying values for the recursive depth for each moving object have been used.

The results for each situation and selected recursive depths are presented in the following. For each experiment, the entire resulting motion is depicted as in Figure 7.3(a), where one disc is drawn per four iteration steps. For selected points in time the relative utilities for the involved agents are depicted as in Figure 7.3(b). Higher values of relative utility are indicated by darker shades of grey. For better visibility, maximum values are emphasized in black.

Collision Course

In this situation, two agents are involved which face each other initially. Their desired velocities are conflicting, i.e. they are directed against each other. Both agents have the same maximum velocities and accelerations.

Figure 7.4 shows the collision course experiment with two agents A and B, where agent A from the left uses recursive depth 1 and agent B from the right uses recursive depth 2. That is, agent B models agent A correctly and assumes that A is able to perceive its environment and to avoid collisions. As a result, agent B displays a straighter motion with less deviation from its optimal path than agent A.

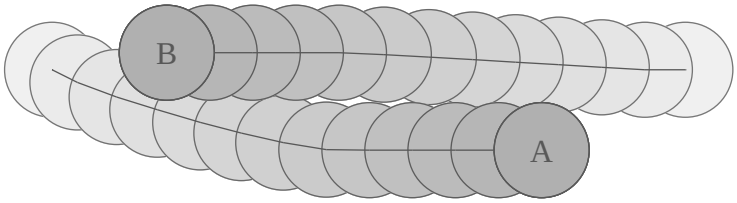
Similarly, Figure 7.5 shows the encounter of agent A from the left and agent B from the right, but now agent A uses recursive depth $d = 3$, and agent B uses depth $d = 2$ as before. Depth 3 means that agent A assumes agent B to move according to depth 2, i.e. in a somewhat "self-confident" way, so agent A appears to choose rather defensive velocities for its motion, and apparently deviates more decidedly and with higher velocity from its optimal path than above. This becomes visible when comparing the velocities of A with maximum relative utility for recursive depths $d = 1$ and $d = 3$ in Figure 7.5(c). Furthermore, the distance between agent A and agent B when they meet is smaller when agent A uses recursive depth 3, compare Figures 7.4(a) and 7.5(a).

Finally, an agent which uses recursive depth $d = 2$, i.e. assuming the other agents to avoid collisions, is still able to avoid collisions with moving obstacles which are oblivious to other agents, as shown in Figure 7.6.

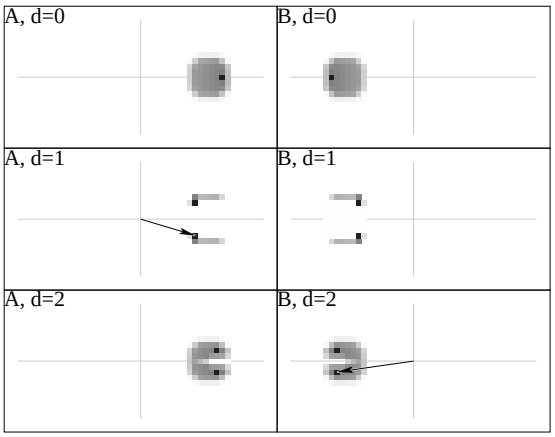
Overtaking

In this situation, two agents are moving in the same direction, agent A behind agent B, whereby agent A desires a much higher velocity than agent B. This creates a conflict that the two agents have to solve.

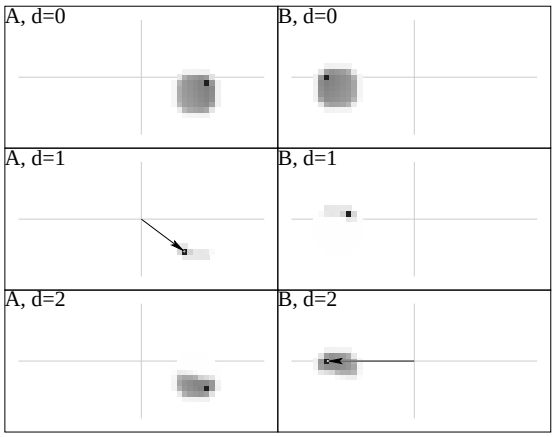
Figure 7.7 shows the result when agent A uses recursive depth 1 and agent B uses recursive depth 2. Agent B does not leave its optimal path as much as agent A does, which



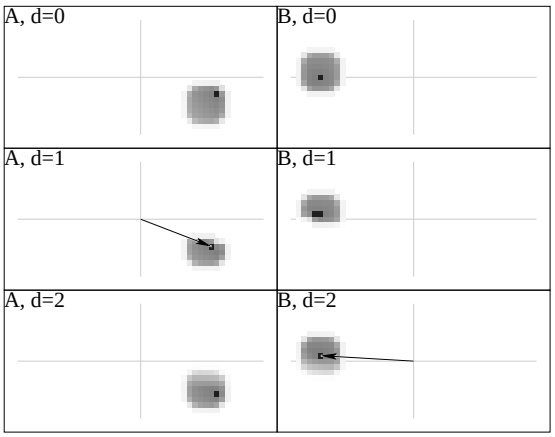
(a) Resulting motion



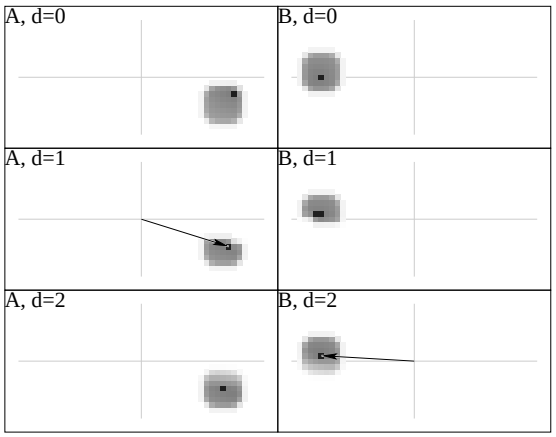
(b) Step 1



(c) Step 2

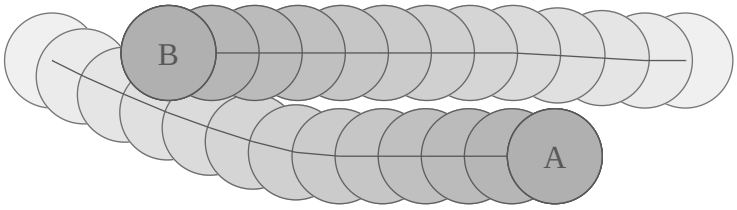


(d) Step 5

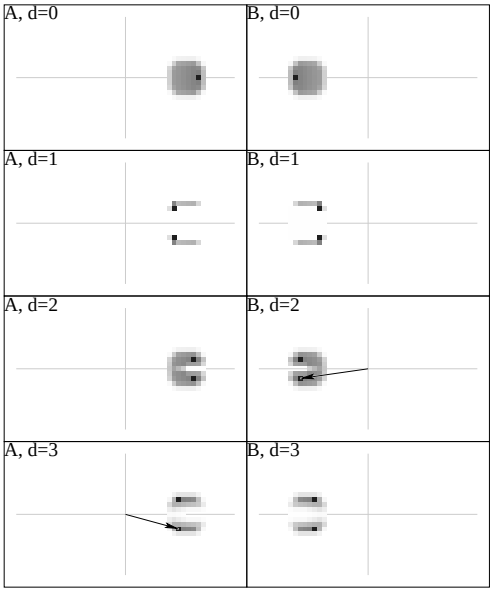


(e) Step 10

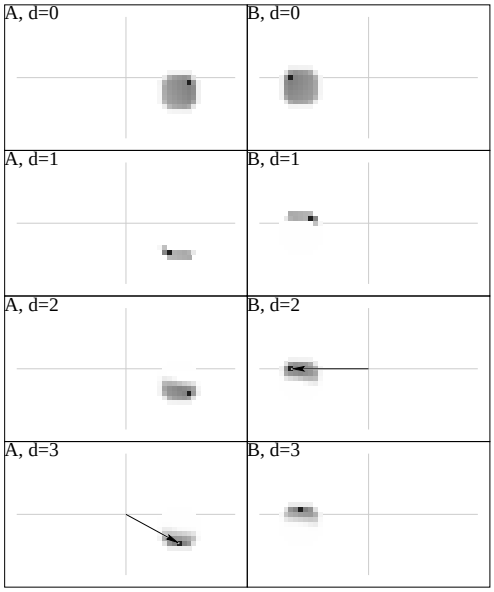
Figure 7.4: Collision course, object A at depth 1 versus object B at depth 2



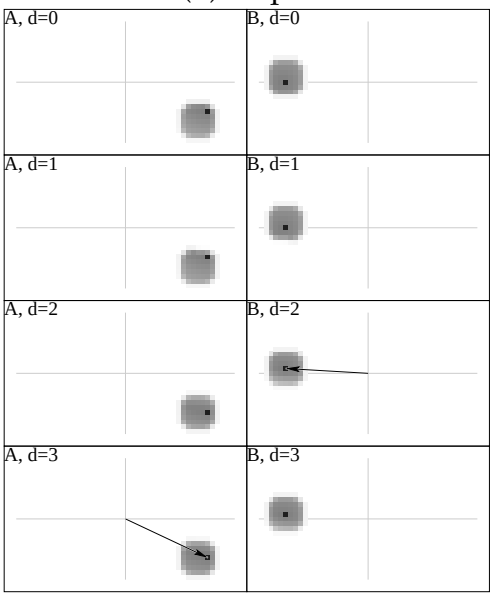
(a) Resulting motion



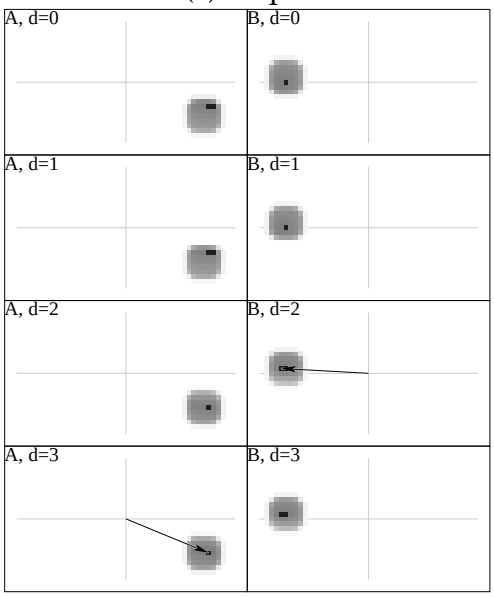
(b) Step 1



(c) Step 2

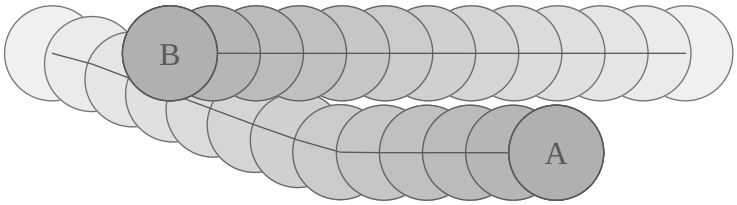


(d) Step 5

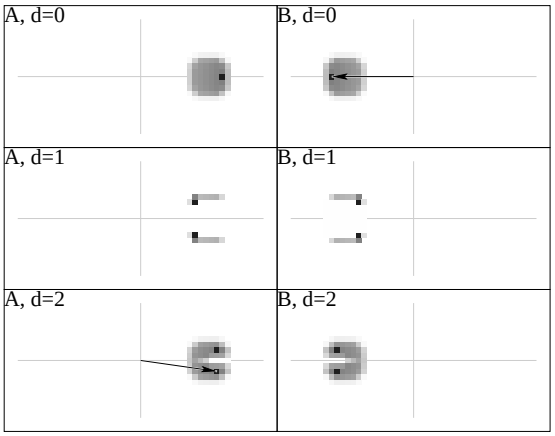


(e) Step 10

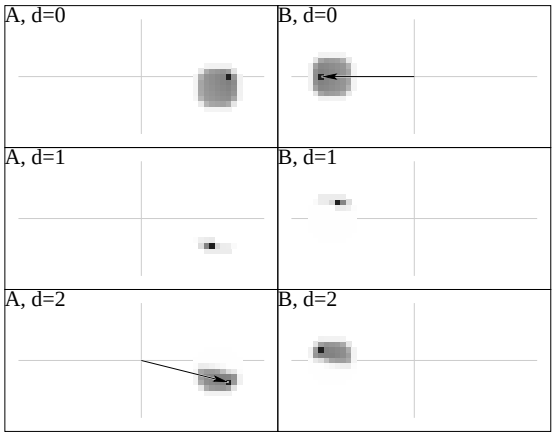
Figure 7.5: Collision course, object A at depth 3 versus object B at depth 2



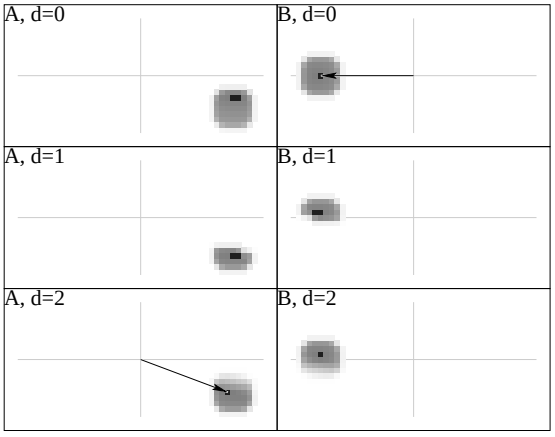
(a) Resulting motion



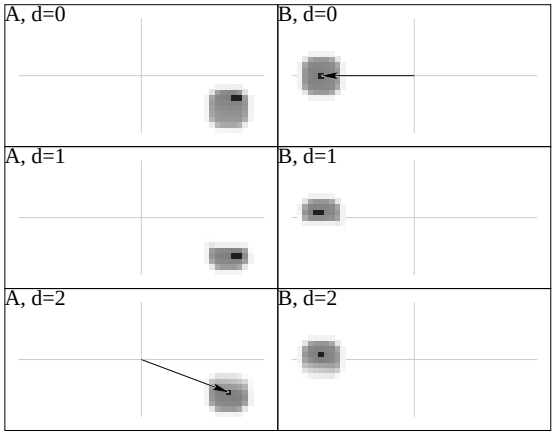
(b) Step 1



(c) Step 2

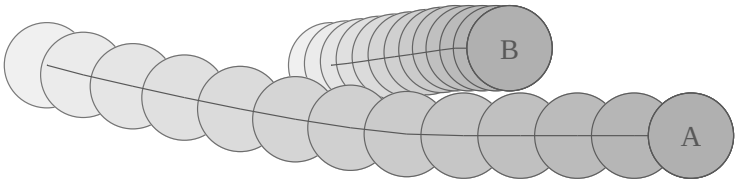


(d) Step 5

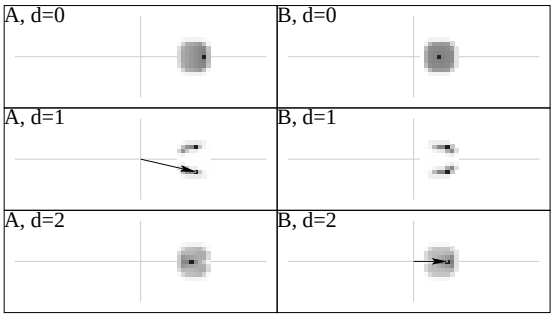


(e) Step 10

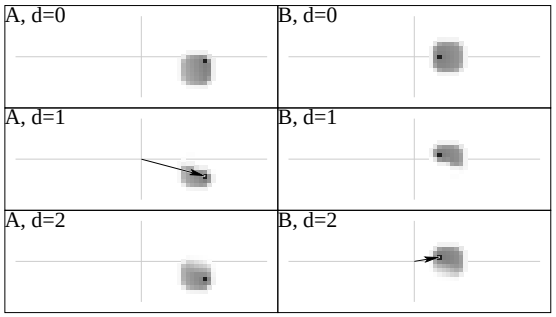
Figure 7.6: Collision course, object A at depth 2 versus object B at depth 0



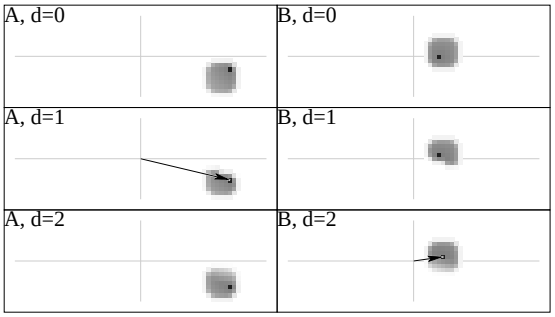
(a) Resulting motion



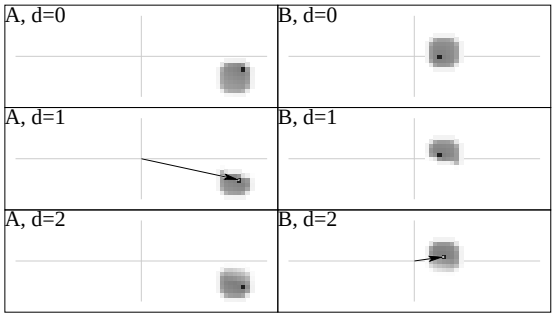
(b) Step 1



(c) Step 2



(d) Step 5



(e) Step 10

Figure 7.7: Overtaking, object A at depth 1 versus object B at depth 2

is what one expects for this chosen pair of depths.

When agent A uses recursive depth 3 instead of 1, its downward velocity component is slightly larger than in the previous experiment, which is visible when comparing its relative utilities and selected velocities at step 10 between Figure 7.7 and Figure 7.8. Furthermore, agent B starts to move horizontally again earlier when agent A uses depth 3, which can be seen when comparing Figure 7.7(a) to Figure 7.8(a). All this indicates that in the latter experiment agent A passes by faster than in the former experiment.

In the overtaking examples until now, we had a slow agent B using recursive depth 2. Now we will consider examples where the fast agent A uses depth 2 and encounters a slow agent B at depth 1 or 3.

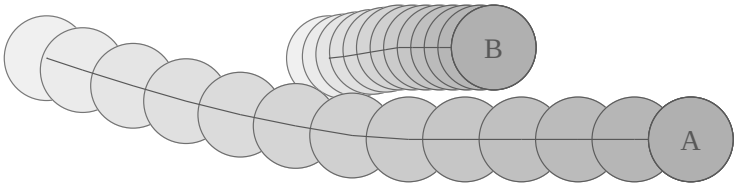
We start with agent B using depth 1. Having seen the experiments above, we would expect the fast agent at depth 2 to force the slow agent to leave its optimal path. This is not the case, as can be seen in Figure 7.9. The reason for this is simple: agent B cannot move fast enough out of agent A's path. In the first step, agent B chooses an avoiding velocity while agent A moves straight ahead, as depicted in Figure 7.9(b). In the next step, agent B has moved a little downward, and therefore agent A starts to move upward, allowing agent A a faster motion in its desired direction (i.e. to the right).

If agent B uses recursive depth 3, it assumes that agent A expects it to avoid collisions, and therefore appears to start moving out of the way more quickly. As a result, the vertical component of agent A's velocity is smaller in this case, which can be seen when comparing the relative utility (which is centered at the current velocity) of agent A for step 10 in both cases. Anyhow, the avoidance maneuver of agent B is more prominent when using depth 3 than when using depth 1, which becomes obvious when comparing Figures 7.9(a) and 7.10(a).

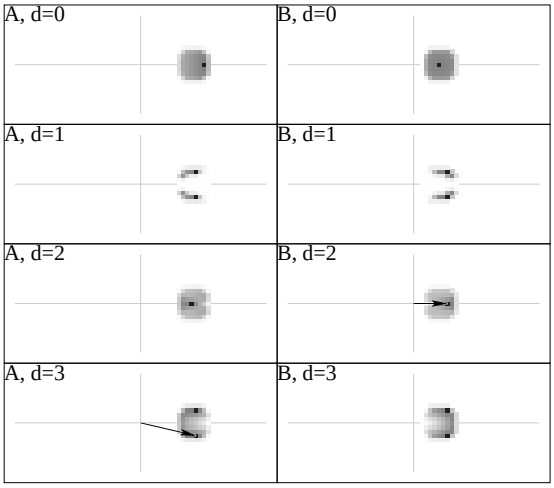
Finally we will consider overtaking examples where both agents use the same recursive depth. We will start with both agents using depth $d = 2$, see Figure 7.11. Due to the symmetry, none of the agents considers deviating from its optimum path, and the initially slower agent B accelerates to avoid a collision.

But if both agents use recursive depth $d = 3$, the conflict is solved in a more intelligent way. In a first step, both agents deviate in the same direction in order to avoid the pending collision, see Figure 7.12. In the next step, agent B still chooses a velocity with a small deviating component, while agent A decides to move horizontally. This asymmetry is amplified during the subsequent steps, such that both agents avoid the collision cooperatively.

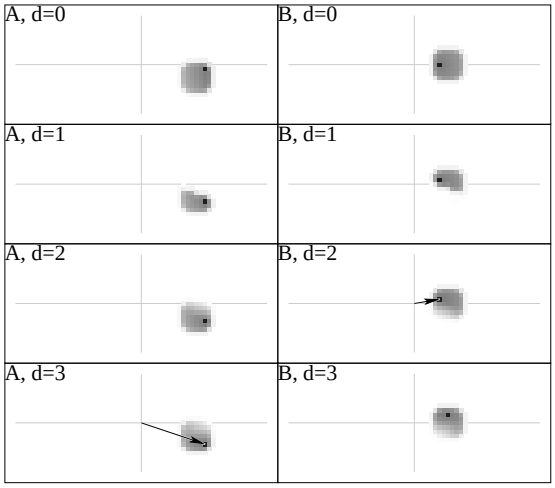
A possible explanation for the different observed behaviours can be seen in Figure 7.12(b). At odd depths, the relative utility functions are bimodal, and at even depths, they are unimodal. Thereby, in the case of even depths, the velocities might be "stabilized" in the center, whereas in the case of odd depths, there might be no "stable" velocities until each object has decided where to move.



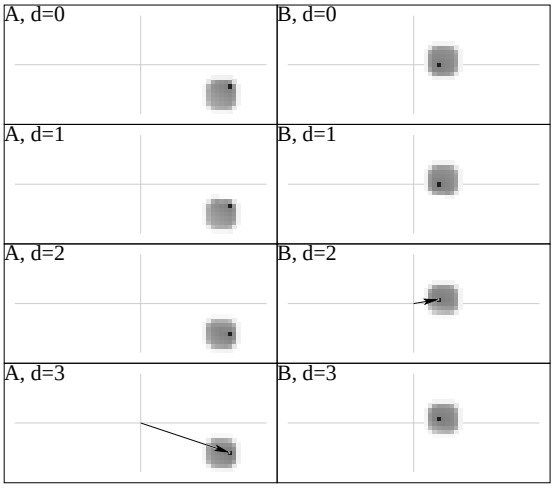
(a) Resulting motion



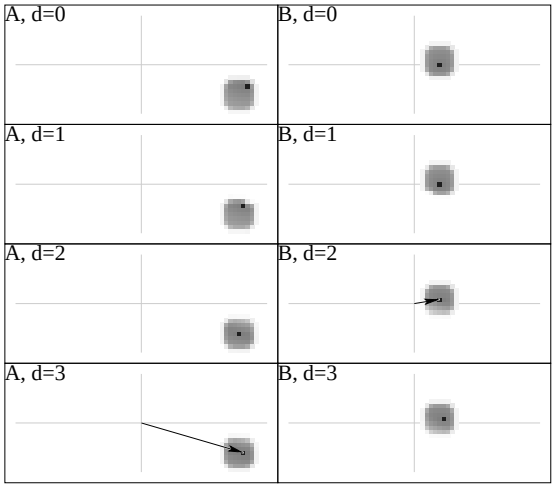
(b) Step 1



(c) Step 2

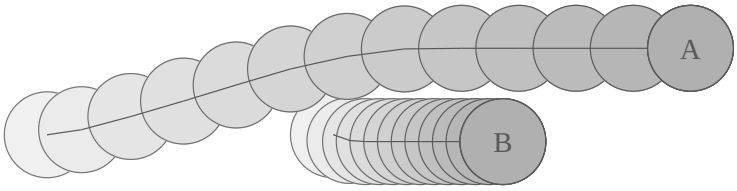


(d) Step 5

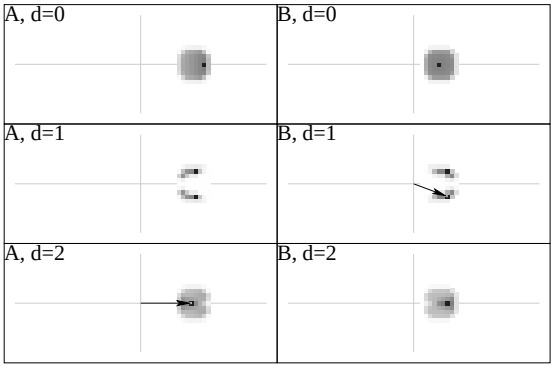


(e) Step 10

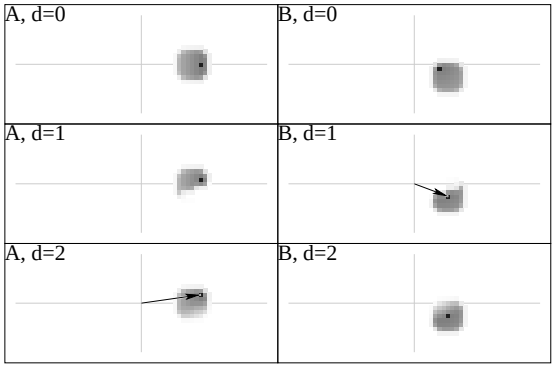
Figure 7.8: Overtaking, object A at depth 3 versus object B at depth 2



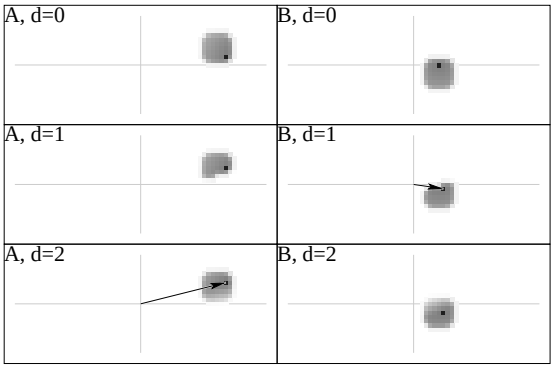
(a) Resulting motion



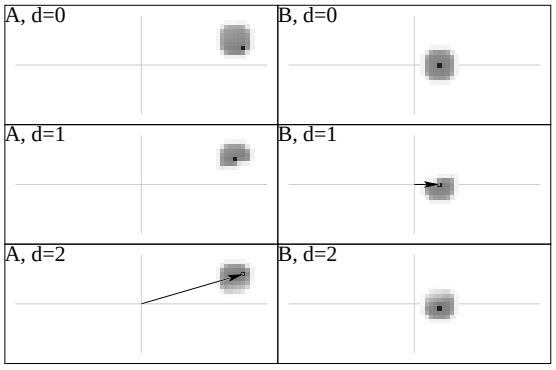
(b) Step 1



(c) Step 2

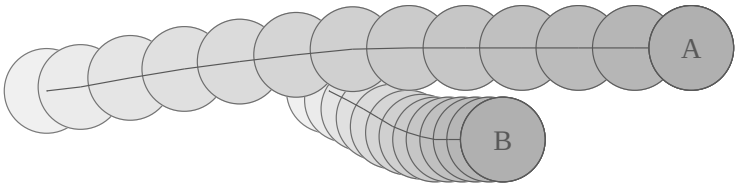


(d) Step 5

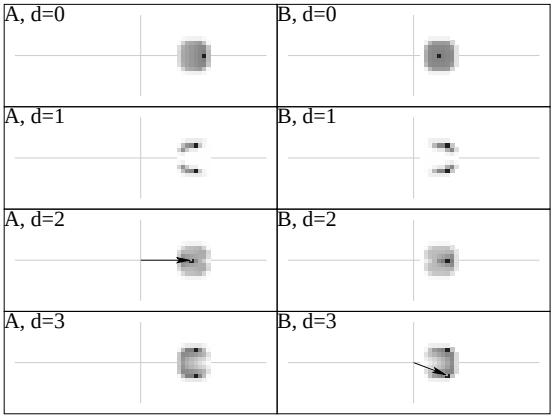


(e) Step 10

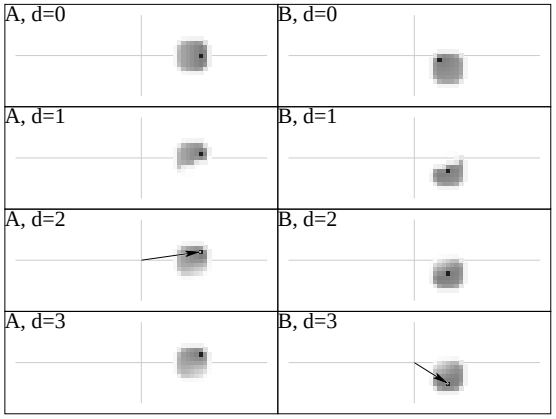
Figure 7.9: Overtaking, object A at depth 2 versus object B at depth 1



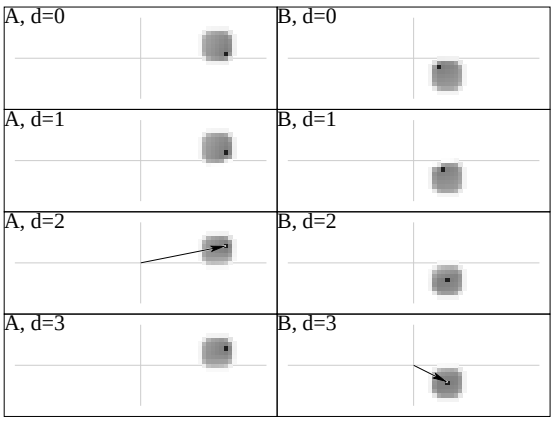
(a) Resulting motion



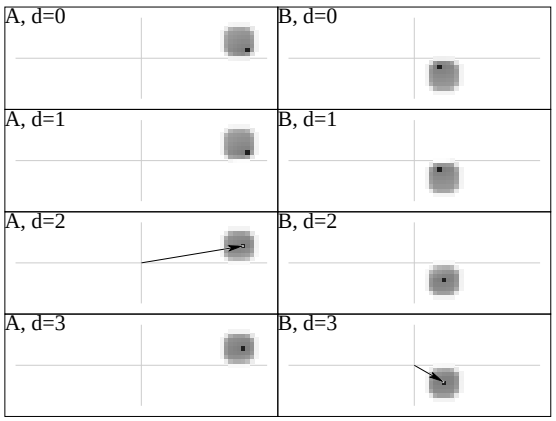
(b) Step 1



(c) Step 2

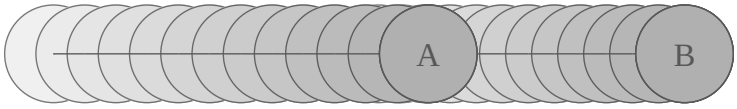


(d) Step 5

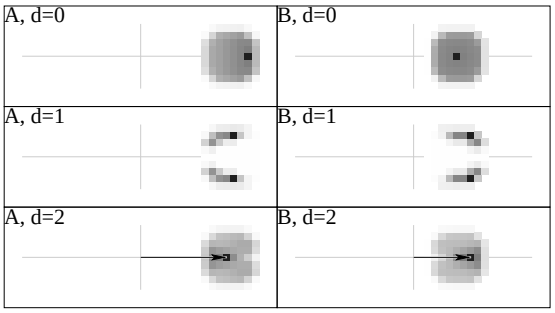


(e) Step 10

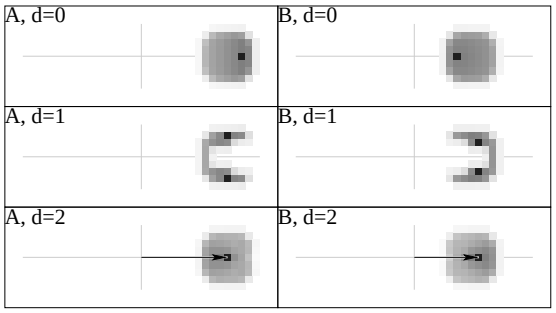
Figure 7.10: Overtaking, object A at depth 2 versus object B at depth 3



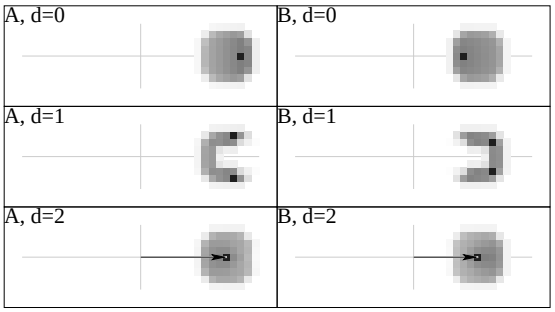
(a) Resulting motion



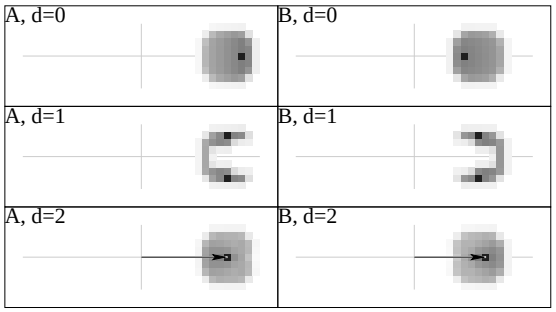
(b) Step 1



(c) Step 2

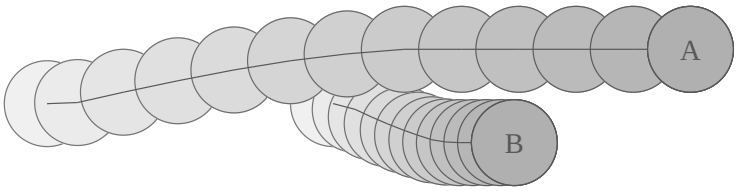


(d) Step 5

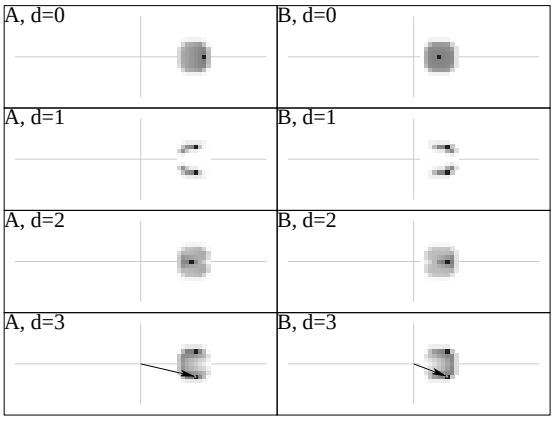


(e) Step 10

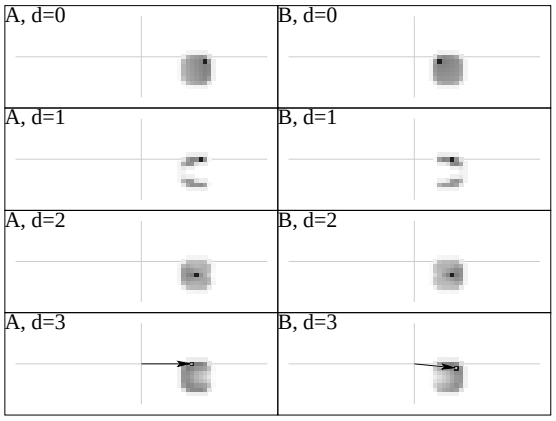
Figure 7.11: Overtaking, object A at depth 2 versus object B at depth 2



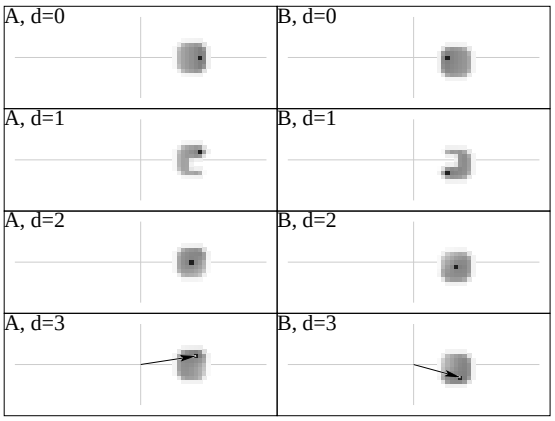
(a) Resulting motion



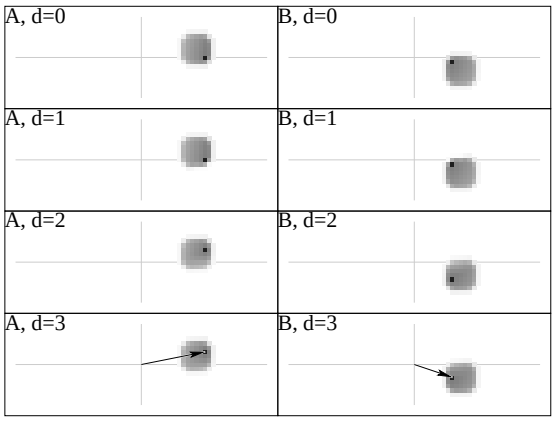
(b) Step 1



(c) Step 2



(d) Step 3



(e) Step 4

Figure 7.12: Overtaking, object A at depth 3 versus object B at depth 3

Static Obstacle

The last set of experiments which we will consider two agents moving in opposite directions with an encounter close to a static obstacle.

In the first example of this type, agent A uses depth 2 and agent B uses depth 1, see Figure 7.13. Having seen the examples above, the result of this experiment is not surprising, since agent A using depth 2 appears to be able to exploit the collision avoiding behavior of agent B and succeeds in moving on the shorter path, closer to the static obstacle C.

Similarly, when agent B uses depth $d = 3$ instead of depth $d = 1$, agent A succeeds in moving on the shorter path, too, see Figure 7.14.

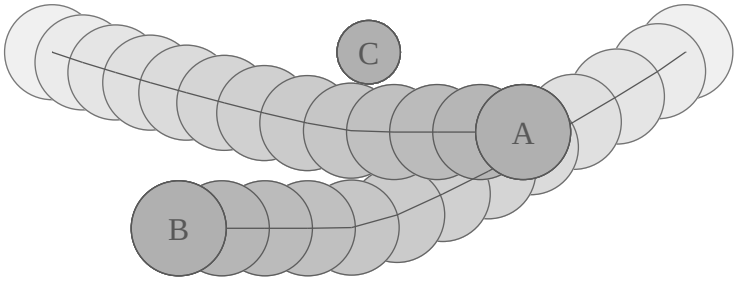
Finally, Figure 7.15 demonstrates that the "defensive" behavior of agent B at depth 3 allows agent A to move on its desired path even when using depth 1.

Note that in no case the agents decided to pass by obstacle C on different sides. The reason is the way a velocity with maximum relative utility is selected. A simple approach is to accept the first velocity with that property, when (discrete) velocities are considered in their lexical order, resulting in the observed velocity selection. Another approach is to select one velocity from the optimal (discrete) velocities by random, which will at least remove artifacts which stem from some velocities being systematically preferred to others.

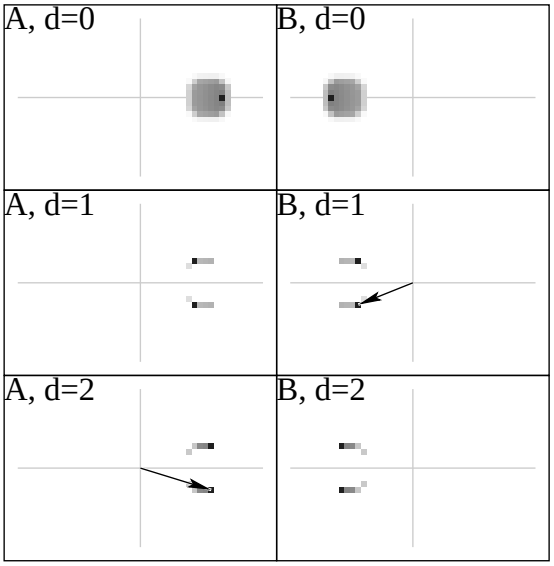
7.5 Discussion

To navigate a mobile robot B_i using depth-d recursive probabilistic velocity obstacles, we repeatedly choose a velocity $\mathbf{v_i}$ maximizing $RU_i^{(d)}$. For $d = 0$, we get a behavior that only obeys the robot's utility function U_i and its dynamic capabilities D_i, but completely ignores other obstacles. For $d = 1$, we get the plain probabilistic velocity obstacle behavior as described in Section 7.2. Something new happens for $d > 1$, when the robot starts modeling the obstacles as perceptive and decision making. Agents navigating at depth $d = 2$ appear to move more aggressively than agents navigating at depths $d = 1$ or $d = 3$, whereby especially depth $d = 3$ appears to result in rather defensive behaviors, and may become an an attractive option for considerate service robots.

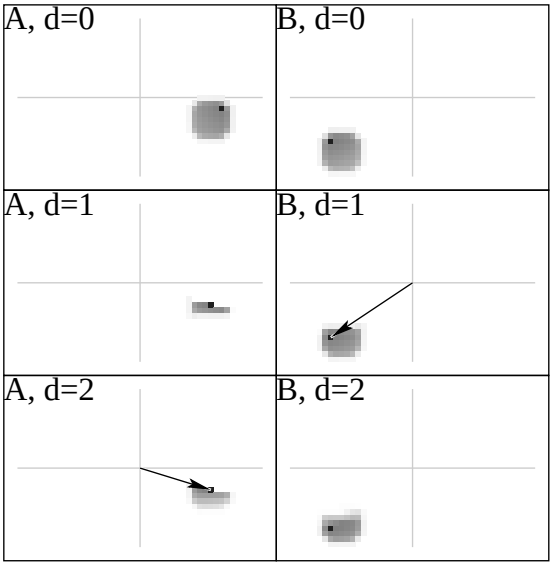
Finding good models of another agent's dynamic capabilities R_j and utility functions U_j is a problem beyond the scope of this thesis. When the action to be taken is considered the first step of a longer sequence, computing the utility function may involve motion planning, or even game-tree search, if reactions of other objects are taken into account. Due to the recursive nature of the approach, such a procedure would have to be applied for any object at any recursive level. This renders such enhancements of utility functions rather infeasible, since already single applications of such procedures are computation-



(a) Resulting motion

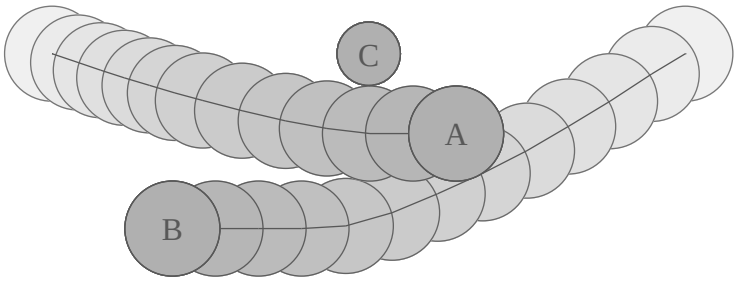


(b) Step 1

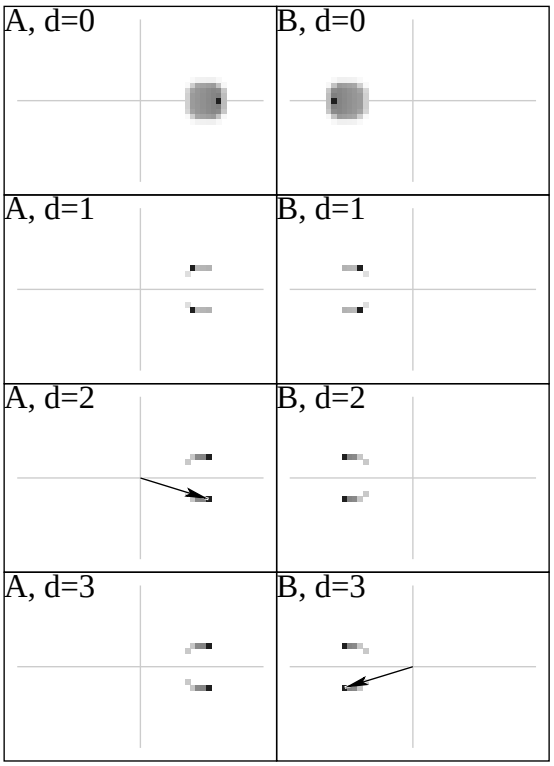


(c) Step 5

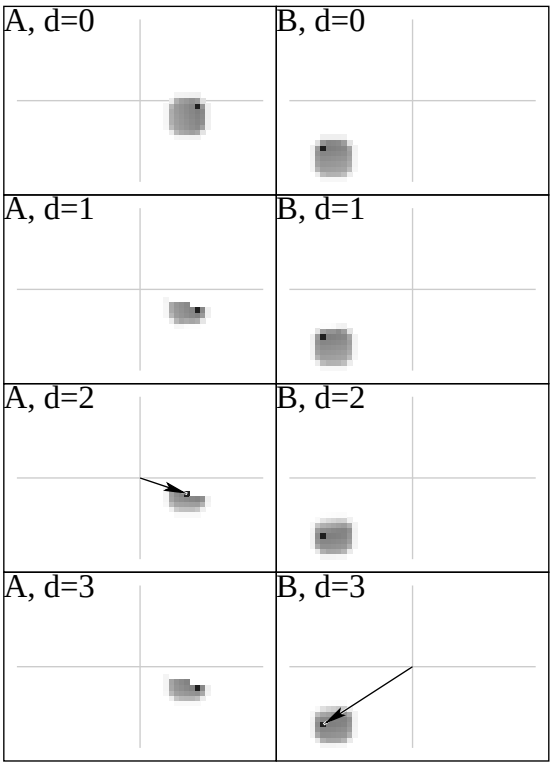
Figure 7.13: Static obstacle C, object A at depth 2 versus object B at depth 1



(a) Resulting motion

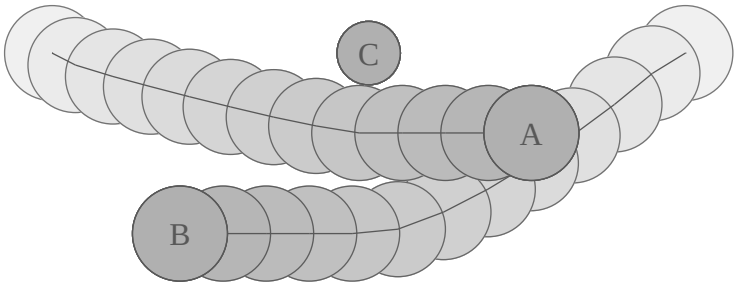


(b) Step 1

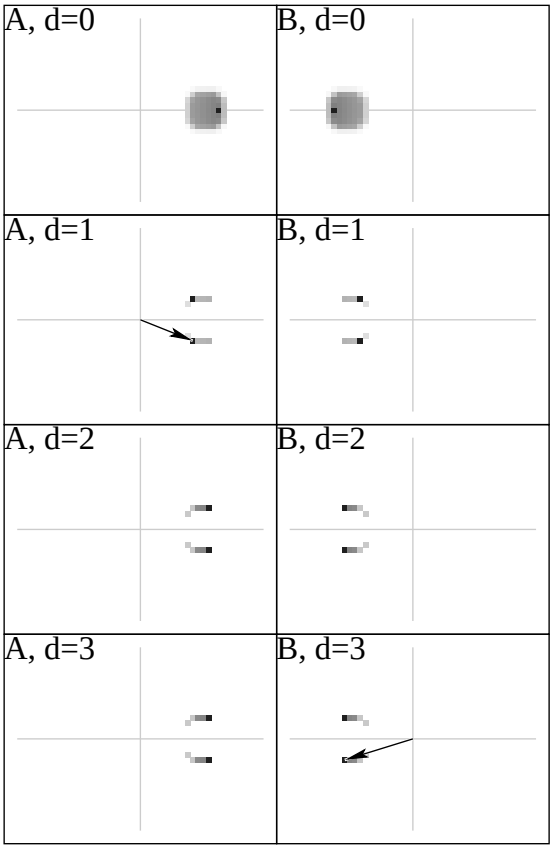


(c) Step 5

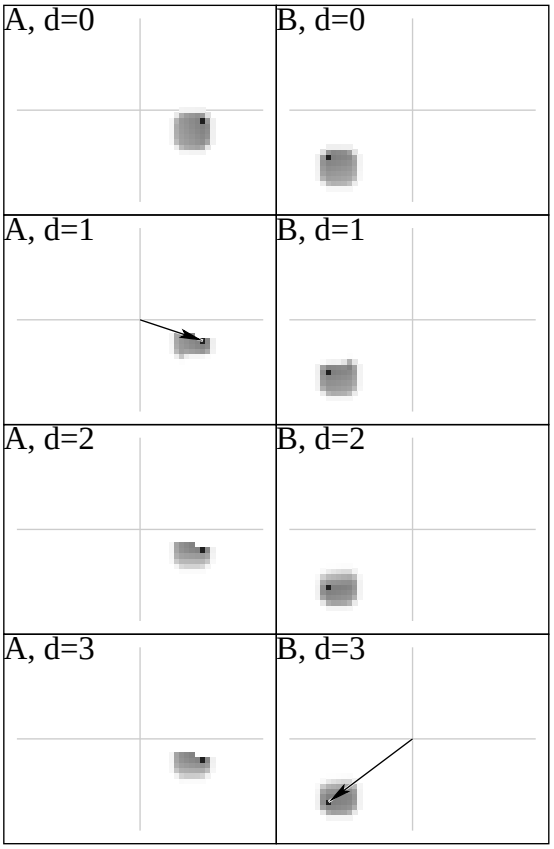
Figure 7.14: Static obstacle C, object A at depth 2 versus object B at depth 3



(a) Resulting motion



(b) Step 1



(c) Step 5

Figure 7.15: Static obstacle C, object A at depth 1 versus object B at depth 3

ally expensive.

The role of the weights α, β, and γ of the three factors of relative utility is largely unexplored. Some first experiments indicated that they in fact do influence the results, but not in a ground-breaking manner. This might change when the uncertainty about shapes and velocities is increased. During the experiments presented above, weights $\alpha = \beta = \gamma = 1$ were used. Note that each agent might use a different set of weights.

Oscillations may appear in models for successive depths. Reconsider the collision course example with both agents facing each other. Assume at depth d, both objects avoid a collision by deviating to the left or to the right. Then at depth $d + 1$, none of the objects will perform an avoidance maneuver, since each object's depth-d model of the other object predicts that other object to avoid the collision. Subsequently, in depth $d+2$, both objects will perform collision avoidance maneuvers again, an so on.

When driving a car on a highway, reasoning similar to the presented approach arises. Cars in front have to be avoided, and when they are already driving on the rightmost lane, they expect faster cars from behind to perform all maneuvers necessary for overtaking without further collaboration. That is, cars from behind are to be modeled with depth 1 or depth 3, and cars in front are to be modeled with depth 0 or depth 2. But the situation is different for emergency cars from behind. They expect any other car to give way to them, and therefore need to be modeled with depth 2.

In the context of pedestrian traffic, a rather different aspect of the presented recursive modeling scheme is that it can serve as a basis for an approach to reasoning about the objects in the environment. One could compare the observed motion of the objects to the motion that was predicted by recursive modeling, possibly discovering relationships among the objects. An example for such a relationship is deliberate obstruction, when one object obtrusively refrains from collision avoidance.

Finally, more accurate models of the interaction partners are required for effectively generating unexpected actions. If $\mu_j[U_i]$ "differs notably" from U_i, but $\mu_i[\mu_j[U_i]]$ is "rather close" to $\mu_j[U_i]$, agent i can detect the difference between $\mu_i[\mu_j[U_i]]$ and U_i, and exploit this situation by doing something that is unexpected, and therefore unobstructed by agent j.

7.5.1 Conclusion

An approach to coordinated motion in dynamic environments has been presented, which reflects the peculiarities of natural, populated environments: obstacles are not only moving, but also perceiving and making decisions based on their perception. This perception and decision making of the intelligent obstacles is taken into account, i.e. it is modeled and integrated into the robot's own decision making.

The approach can be seen as a twofold extension of the velocity obstacle framework. Firstly, object velocities and shapes may be known and processed with respect to some uncertainty (by means of a probabilistic extension). Secondly, the perception and decision making of other objects is modeled and included in the own decision making process (by means of a recursive extension).

Chapter 8

Conclusion

Recently robot technology and applications are leaving the domain of factory automation and spread to our everyday environments, as for example offices, supermarkets, and homes. Today, customers can even buy robotic lawn mowers (Husqvarna AB; Friendly Robotics Ltd.) or vacuum cleaners (Electrolux, 2002; iRobot, 2002) for their personal use. Thus human-machine interaction, especially human-robot interaction is becoming a more and more important topic.

This thesis addressed several problems which arise when mobile robots are operating among humans, for example providing fetch-and-carry, cleaning, or inspection services in public environments like train stations, airports, or even pedestrian areas. For some problems, new solutions have been proposed, whereas new problems haven been opened elsewhere.

8.1 Perception in Dynamic Environments

A basic requirement for a mobile robot operating in populated environments is a robust perception of obstacles and their motion. We devised an approach which uses scans from a laser range finder to detect objects, and generates object matchings between successive scans using network optimization algorithms. The benefit of using globally optimal matchings is an increased robustness with respect to clutter by smaller objects. Thereby, the proposed method served as a solid basis for the approaches presented in subsequent chapters.

Other sensors besides the laser range finder where considered in brief only, and fusion of data from different sensors has not been considered at all. This may serve as a starting point for future research, since the cost measure used in the data association step for object tracking can be easily extended to incorporate information from other sensors (e.g. computer vision), too.

8.2 Navigation and Motion Coordination

Robot motion planning is a wide field of research with numerous existing approaches. Here we did not come up with another new approach, but decided to use a promising existing method based on velocity obstacles. This method has been introduced on a formal basis, and an adaptation to commonly used vehicle kinematics has been proposed.

Furthermore, since the output of the chosen approach is a set of collision-free velocities, a mobile robot behavior for cooperative motion coordination with a human is easily created by an appropriate selection rule for the effective velocity.

Future work in this area might include a better kino-dynamic adaptation, for example aiming at a fusion of the dynamic window approach (which obeys kino-dynamic constraints) and velocity obstacles (incorporating moving obstacles). Considering cooperative motion coordination, better models of the motion of the guide (including his angular velocity when in circular motion) and intentions (goal positions or intermediate goal positions likes doors or points of interest) can be expected to improve the performance of the robot behavior.

8.3 Situation Assessment

For mobile service robots to be widely accepted when operated among humans, they should display at least basic capabilities of situation awareness, which enables them to deliver high performance of service without becoming disregardful with respect to humans. Here, we presented a taxonomy of situations which we substantiated by observed human behavior, and a survey of related work in that area has been given. Unfortunately we were not able to deal with the full potential complexity of situations in populated environments, and therefore only a simple approach to the detection of deliberate obstructions of a mobile robot has been proposed. A lot of future research remains to be done here, and a reasonable point to start from might be to try to transfer one or two approaches from other domains like action recognition or opponent modeling to the context of interactive robots.

Another type of situation assessment is more closely related to failure detection. Here, we presented an approach to determine the similarity of paths in the presence of obstacles, using a metric on the set of path homotopy classes. Prospective applications of such abstract distance measures include the selection and refinement of cooperative motion coordination behaviors as well as failure detection for such behaviors in case the guide gets lost or the quality of the coordination becomes insufficient.

8.4 Reflective Navigation

A final contribution of this thesis is an approach to reflective navigation, where the robot puts itself into the position of the obstacles, reasons about their goals and future velocities, and uses its insight from that reflective reasoning for its own motion decision. This scheme can be applied recursively, resulting in different types of behavior depending on the depth of cascade reasoning. The proposed approach has been evaluated in a simulated environment only, but appears to be a promising starting point for future research, for example addressing the acquisition of accurate models of other agents' goals and capabilities.

List of Figures

List of Algorithms

Bibliography

Ravindra K. Ahuja, Thomas L. Magnati, and James B. Orlin. ***Network Flows: Theory, Algorithms, and Applications***. Prentice Hall, 1993.

Ravindra K. Ahuja, James B. Orlin, Clifford Stein, and Robert E. Tarjan. Improved algorithms for bipartite network flow. ***SIAM Journal on Computing***, 23(5):906–933, October 1994.

Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation — a survey. Technical Report 96-11, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 1996. to appear in: Handbook on Computational Geometry, Eds. Jörg Sack and Jorge Urrutia.

Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. ***Annals of Mathematics and Artificial Intelligence***, 13:251–265, 1995.

Jørgen Bang-Jensen and Gregory Gutin. ***Digraphs: Theory, Algorithms and Applications***. Springer Verlag, 2001.

Rudolf Bauer, Wendelin Feiten, and Gisbert Lawitzky. Steer angle fields: An approach to robust manoeuvring in cluttered, unknown space. ***Robotics and Autonomous Systems***, 12:209–212, 1994.

M. Bennewitz, W. Burgard, and S. Thrun. Learning motion patterns of persons for mobile service robots. In ***Proceedings of the International Conference on Robotics and Automation (ICRA)***, 2002.

A. F. Bobick and Y. A. Ivanov. Action recognition using probabilistic parsing. In ***Proceedings of IEEE Conference on Computer Vision and Pattern Recognition***, Santa Barbara, CA, June 1998.

Herman Bruyninckx and Joris De Schutter. Introduction to intelligent robotics. http://people.mech.kuleuven.ac.be/ bruyninc/robotics.ps.gz, October 2001. (Notes for the robotics course at KU Leuven, Belgium).

John F. Canny. ***The Complexity of Robot Motion Planning***. PhD thesis, Massachusetts Institute of Technology, 1987.

David Carmel and Shaul Markovitch. Opponent modeling in multi–agent systems. In Gerhard Weiß and Sandip Sen, editors, ***Adaptation and Learning in Multi–Agent Systems***, pages 40–52. Springer-Verlag, Heidelberg, Germany, 1996.

Philippe Cattin. Person detector for mobile robots. In ***Robotik 2002***, volume 1679 of ***VDI Report***, 2002.

Animesh Chakravarthy and Debasish Ghose. Obstacle avoidance in a dynamic environment: A collision cone approach. ***IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans***, 28(5):562–574, September 1998.

Eugene Charniak. Bayesian networks without tears. ***AI Magazine***, pages 50–63, 1991.

Eugene Charniak and Robert P. Goldman. A bayesian model of plan recognition. ***Artificial Intelligence***, 64:53–79, 1993.

James W. Davis and Aaron F. Bobick. The representation and recognition of action using temporal templates. In ***Proceedings of IEEE Conference on Computer Vision and Pattern Recognition***, Puerto Rico, June 1997.

Mark Devaney and Ashwin Ram. Situation development in a complex real-world domain. In ***ICML-97 Workshop on Machine Learning Applications in the Real World***, 1997.

Christophe Dousson, Paul Gaborit, and Malik Ghallab. Situation recognition: Representation and algorithms. In ***Proceedings of the 13th IJCAI***, pages 166–174, 1993.

Electrolux. Trilobite. http://www.trilobite.electrolux.se, 2002. Robot Vacuum Cleaner.

Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. ***International Journal of Robotics Research***, 17(7):760–772, July 1998.

Amalia F. Foka and Panos E. Trahanias. Predictive autonomous robot navigation. In ***Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems***, pages 490–495, EPFL, Lausanne, Switzerland, October 2002.

Steven Fortune. A sweepline-algorithm for voronoi diagrams. ***Algorithmica***, 2:153–174, 1987.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. ***IEEE Robotics and Automation***, 4(1), 1997.

Thierry Fraichard. Trajectory planning in dynamic workspace: a 'state-time space' approach. Technical Report 3545, INRIA Rône-Alpes, October 1998.

Yoav Freund, Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, and Robert E. Schapire. Efficient algorithms for learning to play repeated games against computationally bounded adversaries. In ***36th Annual Symposium on Foundations of Computer Science***, pages 332–341, Milwaukee, Wisconsin, October 1995.

Friendly Robotics Ltd. Robomower. http://www.friendlyrobotics.com/. Robotic lawnmower.

Kikuo Fujimura. ***Motion Planning in Dynamic Environments***. Computer Science Workbench. Springer Verlag, 1991.

Piotr J. Gmytrasiewicz. ***A Decision-Theoretic Model of Coordination and Communication in Autonomous Systems (Reasoning Systems)***. PhD thesis, University of Michigan, 1992.

Philip Babcock Gove, editor. ***Webster's Third New International Dictionary of the English Language***. Merriam-Webster, 1993.

David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. In ***The Fourth International Workshop on Algorithmic Foundations of Robotics***, 2000.

Marcus J. Huber and Edmund H. Durfee. Observational uncertainty in plan recognition among interacting robots. In ***Proceedings IJCAI Workshop on Dynamically Interacting Robots***, pages 68–75, Chambery, France, 1993.

Husqvarna AB. Automower. http://www.automower.com/. Robotic Lawnmower.

Stephen Sean Intille. ***Visual Recognition of Multi-Agent Action***. PhD thesis, Massachusetts Institute of Technology, September 1999.

iRobot. Roomba. http://www.irobot.com, 2002. Robot Vacuum Cleaner.

Rufus Isaacs. ***Differential Games—A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization***. John Wiley and Sons, Inc., 1965.

Michael Isard and Andrew Blake. Condensation — conditional densitiy propagation for visual tracking. ***International Journal of Computer Vision***, 29(1):5–28, August 1998.

Henry A. Kautz. A formal theory of plan recognition and its implementation. In ***Reasoning about Plans***, chapter 2, pages 69–125. Morgan Kaufmann Publishers, 1991.

Boris Kluge, Christian Köhler, and Erwin Prassler. Fast and robust tracking of multiple moving objects with a laser range finder. In ***Proc. of Int. Conf. on Robotics and Automation***, Seoul, Korea, May 2001.

Sven Oliver Krumke, Hartmut Noltemeier, and Hans-Christoph Wirth. ***Graphentheoretische Konzepte und Algorithmen – Skript zur Vorlesung***. via WWW, Lehrstuhl für Informatik I an der Bayerischen Julius-Maximilians-Universität Würzburg, 2000.

Jean-Claude Latombe. ***Robot Motion Planning***. Kluwer Academic Publishers, 1991.

Eugene L. Lawler. ***Combinatorial optimization: networks and matroids***. Rinehart and Winston, New York, 1976.

William S. Massey. ***Algebraic Topology: An Introduction***. Harcourt, Brace & World, Inc., 1967.

Kurt Mehlhorn and Stefan Näher. ***LEDA—A Platform for Combinatorial and Geometric Computing***. Cambridge University Press, 1999.

Esther B. Meier and Frank Ade. Object detection and tracking in range image sequences by separation of image features. In ***IEEE International Conference on Intelligent Vehicles***, pages 280–284, 1998.

Jun Miura and Yoshiaki Shirai. Modeling motion uncertainity of moving obstacles for robot motion planning. In ***Proc. of Int. Conf. on Robotics and Automation (ICRA)***, 2000.

Thomas B. Moeslund. Computer vision-based human motion capture – a survey. Technical Report LIA 99-02, University of Aalborg, March 1999.

Michael Mohnhaupt and Bernd Neumann. Understanding object motion: Recognition, learning and spatiotemporal reasoning. ***Robotics and Autonomous Systems***, 8:65–91, 1991.

H.-H. Nagel, H. Kollnig, M. Haag, and H. Damm. The association of situation graphs with temporal variations in image sequences. In ***Working Notes AAAI-95 Fall Symposium Series 'Computational Models for Integrating Language and Vision'***, pages 1–8, Cambridge/MA, USA, November 1995.

Hartmut Noltemeier. ***Graphentheorie: mit Algorithmen und Anwendungen***. de Gruyter, 1976.

Nuria M. Oliver, Barbara Rosario, and Alex P. Pentland. A bayesian computer vision system for modeling human interactions. In Henrik I. Christensen, editor, ***Proc. of Int. Conf. on Computer Vision Systems (ICVS)***, volume 1542 of ***Lecture Notes in Computer Science***, pages 255–272, Gran Canaria, Spain, January 1999. Springer Verlag.

Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. Perceptual Computing Technical Report 433, M.I.T. Media Lab, 1995. Originally published as: Towards Augmented Control Systems, in IEEE Intelligent Vehicles 1995.

E. Prassler, J. Scholz, M. Schuster, and D. Schwammkrug. Tracking a large number of moving objects in a crowded environment. In ***IEEE Workshop on Perception for Mobile Agents***, Santa Barbara, June 1998a.

E. Prassler, J. Scholz, and M. Strobel. Maid: Mobility assistance for elderly and disabled people. In ***Proc. of the 24th Int. Conf. of the IEEE Industrial Electronics Soc. IECON'98***, Aachen, Germany, 1998b.

Franco P. Preparata and Michael I. Shamos. ***Computational geometry : an introduction***. Springer Verlag, 1988.

David V. Pynadath. ***Probabilistic Grammars for Plan Recognition***. PhD thesis, University of Michigan, 1999.

J. H. Reif and S. Tate. Continuous alternation: The complexity of pursuit in continuous domains. ***Algorithmica***, 10(2–4):156–181, 1993.

John H. Reif and Hongyan Wang. Nonuniform discretization for kinodynamic motion planning and its applications. ***SIAM Journal on Computing***, 30(1):161–190, 2000.

Ronald L. Rivest and Robert E. Schapire. Diversity-based inference of finite automata. ***Journal of the ACM***, 41(3):555–589, May 1994.

Rómer Rosales and Stan Sclaroff. Trajectory guided tracking and recognition of actions. Technical Report BU-CS-TR-99-002, Computer Science Department, Boston University, 111 Cummington St, Boston, MA 02215, 1999.

Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In ***Proceedings of the International Conference on Computer Vision***, Bombai, January 1998.

Christian Schlegel and Thomas Kämpke. Recognizing user intentions for trajectory specification. In ***Fourth European Workshop on Advanced Mobile Robots (Eurobot '01)***, pages 231–238, Lund, Sweden, September 2001.

Christian Schlegel, Jörg Illmann, Heiko Jaberg, Matthias Schuster, and Robert Wörz. Vision based person tracking with a mobile robot. In ***Proc. 9th British Machine Vision Conference (BMVC)***, pages 418–427, Southampton, UK, 1998.

Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In ***Proc. of IEEE Int. Conf. on Robotics and Automation***, Seoul, Korea, 2001.

Zvi Shiller, Frederic Large, and Sepanta Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In ***Proceedings of the 2001 IEEE International Conference on Robotics and Automation***, pages 3716–3721, Seoul, Korea, May 2001.

Hedvig Sidenbladh, Danica Kragic, and Henrik I. Christensen. A person following behaviour for a mobile robot. In ***IEEE International Conference on Robotics and Automation***, volume 1, pages 670–675, 1999.

Karin Sobottka and Horst Bunke. Vision-based driver assistance using range imagery. In ***IEEE International Conference on Intelligent Vehicles***, pages 280–284, 1998.

Rahul Sukthankar. ***Situation Awareness for Tactical Driving***. PhD thesis, Carnegie Mellon University, 1997.

Milind Tambe. Recursive agent and agent-group tracking in a real-time, dynamic environment. In ***International Conference on Multi-agent Systems***, 1995.

Pravin M. Vaidya. Geometry helps in matching. ***SIAM Journal on Computing***, 18(6): 1201–1225, December 1989.

A. Walthelm and M. Litza. Stereo microphone sound source localization for service robots. In ***Proceedings of 1st International Workshop on Advances in Service Robotics (ASER)***, Bardolino, Italy, 2003. Fraunhofer IRB Verlag, Stuttgart.

Eric W. Weisstein. ***MathWorld***–a wolfram web resource, 2004. URL `http://mathworld.wolfram.com`.

C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. ***Discrete and Computational Geometry***, 2:365–393, 1987.

Paul Zarchan. ***Tactical and Strategic Missile Guidance***. American Institute of Aeronautics and Astronautics, 1998.