Reflective Navigation

Boris Kluge
InMach Intelligente Maschinen GmbH
Helmholtzstr. 16, 89081 Ulm, Germany
kluge@inmach.de

Erwin Prassler
Bonn-Rhein-Sieg University of Applied Science
Grantham-Allee 20, 53757 Sankt Augustin, Germany
erwin.prassler@fh-brs.de

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 paper an approach to reflective navigation is presented which extends the velocity obstacle navigation scheme to incorporate reasoning about other objects' perception and motion goals.

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 [3] and Miura and Shirai [5]. 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 [2] for obstacles moving on straight lines, and has been extended by Shiller et al. [6] 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 [4], 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*).

1.2 Overview

Assume a robot B uses deterministic velocity obstacles for its navigation. Then, 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, 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 2.

Being able to cope with uncertain obstacle velocities, Section 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 4, and finally concluded after discussing the presented work.

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.

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 defining the notion of an "uncertain shape" is not straightforward. One idea might be to model uncertainty about the actual shape of a rigid body B by a function

$$PB : \mathbb{R}^2 \to [0, 1] \tag{1}$$

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 2.1 (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*

$$P(R \leq r) = \begin{cases} 0, & \text{if } r < a, \\ \frac{r-a}{b-a} & \text{if } r \in [a, b], \text{ and} \\ 1, & \text{if } r > b. \end{cases} \tag{2}$$

For the sake of brevity, we may call a disc with uncertain radius probabilistic disc *or* p-disc*, too.*

Discs with uncertain radius cannot be represented by a mapping $PB : \mathbb{R}^2 \to [0, 1]$ as above alone, since for example the events $(0, r) \in B$ and $(r, 0) \in B$ for $r \in \mathbb{R}$ are not independent if B is a disc with uncertain radius. Therefore we will focus on p-discs as probabilistic objects in the following. This is not a severe restriction, since the remainder of this paper remains valid after changing the definition of probabilistic objects, provided that the definition of a probabilistic collision cone is adapted accordingly.

Definition 2.2 (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 2.3 (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* $R_i + R_j \leq |\mathbf{c_i} - \mathbf{c_j}|$*.*

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 2.4 (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 is a mapping* $PCC_{ij} : \mathbb{R}^2 \to [0, 1]$ *with*

$$PCC_{ij} : \mathbf{v_{ij}} \mapsto P\left(X_i + X_j \geq \min_{t \in \mathbb{R}_0^+} |\mathbf{c_i} + \mathbf{v_{ij}} \cdot t - \mathbf{c_j}|\right), \tag{3}$$

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* $t \in \mathbb{R}_0^+$*.*

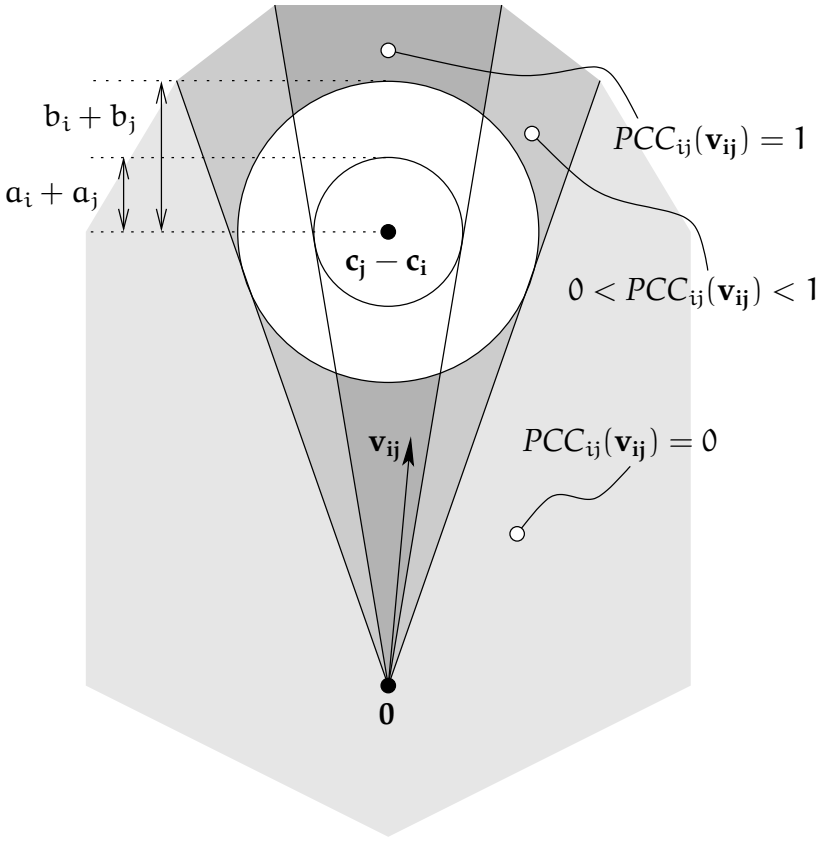


Figure 1: Probabilistic collision cone of two discs $(D(a_i, b_i), \mathbf{c_i})$ and $(D(a_j, b_j), \mathbf{c_j})$ with uncertain radii

As an example, Figure 1 shows the probabilistic collision cone of two discs with uncertain radii.

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

$$V_j : \mathbb{R}^2 \to \mathbb{R}_0^+. \tag{4}$$

Given such an 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

$$PVO_{ij} : \mathbb{R}^2 \to [0, 1] \tag{5}$$

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

$$PVO_{ij}(\mathbf{v_i}) = \int_{\mathbb{R}^2} V_j(\mathbf{v_j}) PCC_{ij}(\mathbf{v_i} - \mathbf{v_j}) d^2\mathbf{v_j}, \tag{6}$$

which is equivalent to

$$PVO_{ij} = V_j * PCC_{ij} \tag{7}$$

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. Therefore, the function $PVO_i : \mathbb{R}^2 \to [0, 1]$ with

$$PVO_i(\mathbf{v_i}) = 1 - \prod_{j \neq i,\, D_j \in \mathcal{B}} \left(1 - PVO_{ij}(\mathbf{v_i})\right). \tag{8}$$

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.

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

$$RU_i = U_i^{\alpha} \cdot R_i^{\beta} \cdot (1 - PVO_i)^{\gamma}, \tag{9}$$

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. 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.

2.3.1 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

$$discr(p) = \mathsf{p} = \left(\left\lfloor \frac{x}{\kappa} \right\rfloor, \left\lfloor \frac{y}{\kappa} \right\rfloor \right) \in \mathbb{Z}^2. \tag{10}$$

Conversely, for a discretized point $\mathsf{p} = (z_1, z_2) \in \mathbb{Z}^2$ we define its cell as

$$\begin{aligned} cell(z_1,z_2) &= \{p \in \mathbb{R}^2 \mid discr(p) = (z_1,z_2)\} \\ &= [z_1\kappa, (z_1+1)\kappa) \times [z_2\kappa, (z_2+1)\kappa)\,. \end{aligned} \tag{11}$$

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

$$\mathsf{F}(z_1,z_2) = \frac{1}{\kappa^2} \int_{cell(z_1,z_2)} F(x,y)\,dx\,dy, \tag{12}$$

i.e. $F(z_1, z_2)$ is the average of f on $cell(z_1, z_2)$. However, in practise we will only require $F(z_1, z_2) \in f(cell(z_1, z_2))$ 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 $F(p)$, 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 12, and otherwise assume that the value of κ is adequate for f.

Finally, for a discretized function $F : \mathbb{Z}^2 \to \mathbb{R}_0^+$ the set

$$\sigma(F) = \{(x, y) \in \mathbb{Z}^2 \,|\, F(x, y) > 0\} \tag{13}$$

is called the *supporting set* of F, which is the set of cells on which the discretized function does not vanish. The property

$$\sigma(FG) = \sigma(F) \cap \sigma(G) \tag{14}$$

is easily shown. Furthermore,

$$\sum_{p \in \mathbb{Z}^2} F(p)\kappa^2 = \int_{\mathbb{R}^2} f(\mathbf{p})\,d^2\mathbf{p} \tag{15}$$

holds for strict discretizations.

2.3.2 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 $\sigma(V_j)$ is finite. Therefore, $PVO_{ij}(v_i)$ can be computed for any $\mathbf{v_i}$ by using

$$PVO_{ij}(v_i) = \sum_{v_j \in \sigma(V_j)} V_j(v_j) PCC_{ij}(v_i - v_j)\kappa^2, \tag{16}$$

which is the discrete version of Equation 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 $\sigma(R_i)$ 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 $\sigma(RU_i)$ will be considered for navigating D_i, and since $\sigma(RU_i) \subseteq \sigma(R_i)$, we can restrict velocities to the finite domain $\sigma(RU_i)$.

Algorithm 1 RELATIVE UTILITY

1: **input:** a set of placed p-discs $\mathcal{B} = \{D_i = (D(a_i, b_i), \mathbf{c_i}) \mid i = 1, 2, \ldots, n\}$
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

2.3.3 Algorithm

Combining the results from the previous subsections, we get

$$\mathsf{PVO}_i = 1 - \prod_{j \neq i} (1 - \mathsf{PVO}_{ij}) \tag{17}$$

and further

$$\begin{aligned} \mathsf{RU}_i(v_i) &= U_i(v_i) R_i(v_i) (1 - \mathsf{PVO}_i(v_i)) \\ &= U_i(v_i) R_i(v_i) \prod_{j \neq i} (1 - \mathsf{PVO}_{ij}(v_i)) \end{aligned} \tag{18}$$

for any $v_i \in \sigma(\mathsf{RU}_i)$. This observation is summarized in Algorithm 1, 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 16 and lines 10–13 in Algorithm 1), and $\mathsf{RU}_i(v_i)$ in time $\mathcal{O}(\sum_{j \neq i} |\sigma(V_j)|)$ (according to Equation 18 and lines 8–15 in Algorithm 1). Finally, a discrete velocity v_i maximizing RU_i can be found in time

$$\mathcal{O}\left(|\sigma(R_i)| \cdot \sum_{j \neq i} |\sigma(V_j)| \right), \tag{19}$$

integrating the search into the loop from line 7 to line 16 in Algorithm 1. That is, the dependence of the running time on the number of obstacles is only linear, but the dependence on the discretization is $\mathcal{O}(1/\kappa^4)$.

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.

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. [1] 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 observed 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.

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 3.1 (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

$$\mu_i[V_j^{(d)}] = \begin{cases} \frac{1}{w}\mu_i[RU_j^{(d)}] & \text{if } d > 0 \text{ and } w := \int RU_j^{(d)}\, d^2\mathbf{v} > 0, \\ \mu_i[V_j] & \text{else.} \end{cases} \tag{20}$$

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 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

$$RU_i^{(d)} = \begin{cases} U_i^{\alpha} \cdot R_i^{\beta} & \text{if } d = 0, \\ U_i^{\alpha} \cdot R_i^{\beta} \cdot \left(1 - PVO_i^{(d)}\right)^{\gamma} & \text{else,} \end{cases} \tag{21}$$

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

$$PVO_i^{(d)} = 1 - \prod_{j \neq i} \left(1 - \mu_i[V_j^{(d-1)}] * PCC_{ij}\right), \tag{22}$$

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 3.2 (Interpretation of RPVO Function Models) *Let F be a symbol for a func-*

tion from $\mathbb{R}^2$ *to* $[0,1]$. *Then,* F *will be interpreted as follows*

$$\mathcal{I}(F) = \begin{cases} \mathcal{I}(\mu_i[G]) & \text{if } F = \mu_i[\mu_i[G]], \\ \mathcal{I}(\mu_i[G])\ op\ \mathcal{I}(\mu_i[H]) & \text{if } F = \mu_i[G\ op\ H] \text{ with } op \in \{+,\cdot,*\}, \\ \mathcal{I}(\mu_i[G])^{\alpha} & \text{if } F = \mu_i[G^{\alpha}] \text{ with } \alpha \in \mathbb{R}, \\ \mathcal{I}(C) & \text{if } F = \mu_i[C] \text{ and } C \text{ symbolizes a constant function}, \\ U_i & \text{if } F = \mu_i[U_i], \\ R_i & \text{if } F = \mu_i[R_i], \\ V_i & \text{if } F = \mu_i[V_i], \\ PCC_{ij} & \text{if } F = \mu_i[PCC_{ij}], \text{ and} \\ F & \text{if } F \text{ is not of the shape } \mu_i[G], \end{cases} \tag{23}$$

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

$$RU_i^{(d)} = U_i^{\alpha} R_i^{\beta} \prod_{j \neq i} \left(1 - \frac{1}{w_j}\mu_i[RU_j^{(d-1)}] * PCC_{ij}\right)^{\gamma}, \tag{24}$$

from Equations 20–22, and with the rules from 3.2 follows

$$\begin{aligned} \mu_i[RU_j^{(d)}] &= \mu_i\left[U_j^{\alpha} R_j^{\beta} \prod_{k \neq j} \left(1 - \frac{1}{w_k}\mu_j[RU_k^{(d)}] * PCC_{jk}\right)^{\gamma}\right] \\ &= \mu_i[U_j]^{\alpha}\mu_i[R_j]^{\beta} \prod_{k \neq j} \left(1 - \frac{1}{w_k}\mu_i[\mu_j[RU_k^{(d)}]] * \mu_i[PCC_{jk}]\right)^{\gamma}, \end{aligned} \tag{25}$$

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.

3.2.1 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 3.3 (Interpretation under Equal Information) *We say* all agents have equal information, *iff*

$$\mathcal{I}(\mu_i[F]) = \mathcal{I}(\mu_j[F]) \tag{26}$$

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 $i_1, \dots, i_k, i$:

$$\begin{aligned}\mathcal{I}(\mu_{i_1}[\mu_{i_2}[\dots\mu_{i_k}[F]\dots]]) &= \mathcal{I}(\mu_{i_2}[\mu_{i_2}[\dots\mu_{i_k}[F]\dots]]) \\ &= \mathcal{I}(\mu_{i_2}[\dots\mu_{i_k}[F]\dots]) \\ &\dots \\ &= \mathcal{I}(\mu_{i_k}[F]) \\ &= \mathcal{I}(\mu_i[F]),\end{aligned} \tag{27}$$

and with Definition 3.2 we have

$$\mathcal{I}(\mu_{i_1}[\dots\mu_{i_k}[F]\dots]) = F, \text{ for } F \in \{U_i, R_i, V_i, PCC_{ij}\} \tag{28}$$

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.

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 2 gives the details of the used dynamic programming approach

Algorithm 2 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 1 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 $d = 0, 1, \ldots, r$

3.3.1 Complexity

We begin the complexity assessment by measuring the sizes of the supporting sets of the discretized functions used in Algorithm 2, where line 9 implies

$$\sigma(\mathsf{RU}_i^{(0)}) \subseteq \sigma(\mathsf{R}_i), \tag{29}$$

and from line 13 follows

$$\sigma(\mathsf{RU}_i^{(d)}) \subseteq \sigma(\mathsf{R}_i) \tag{30}$$

for $d > 0$. Line 8 implies

$$\sigma(\mathsf{V}_i^{(0)}) = \sigma(\mathsf{V}_i), \tag{31}$$

and from lines 16 and 18 follows

$$\sigma(\mathsf{V}_i^{(d)}) \subseteq \sigma(\mathsf{R}_i) \cup \sigma(\mathsf{V}_i) \tag{32}$$

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 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

$$\mathcal{O}(\sum_{i=1}^{n} (N_i \sum_{j \neq i} N_j)) \tag{33}$$

operations, and the loop from line 11 to line 21 requires

$$\mathcal{O}(r \sum_{i=1}^{n} (N_i \sum_{j \neq i} N_j)) \tag{34}$$

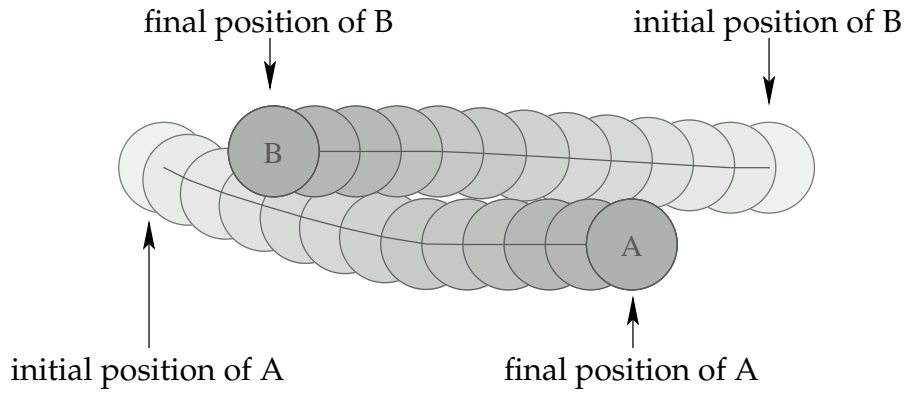
operations. The complexity of the loop from line 11 to 21 clearly dominates the complexity of the initialization loop from line 7 to 10. Therefore Equation 34 gives an upper bound of the overall time complexity of our implementation. That is, the dependence on the maximum recursive depth is linear, the dependence on the number of objects is $\mathcal{O}(n^2)$, and the dependence on the discretization remains $\mathcal{O}(1/\kappa^4)$.

4 Results

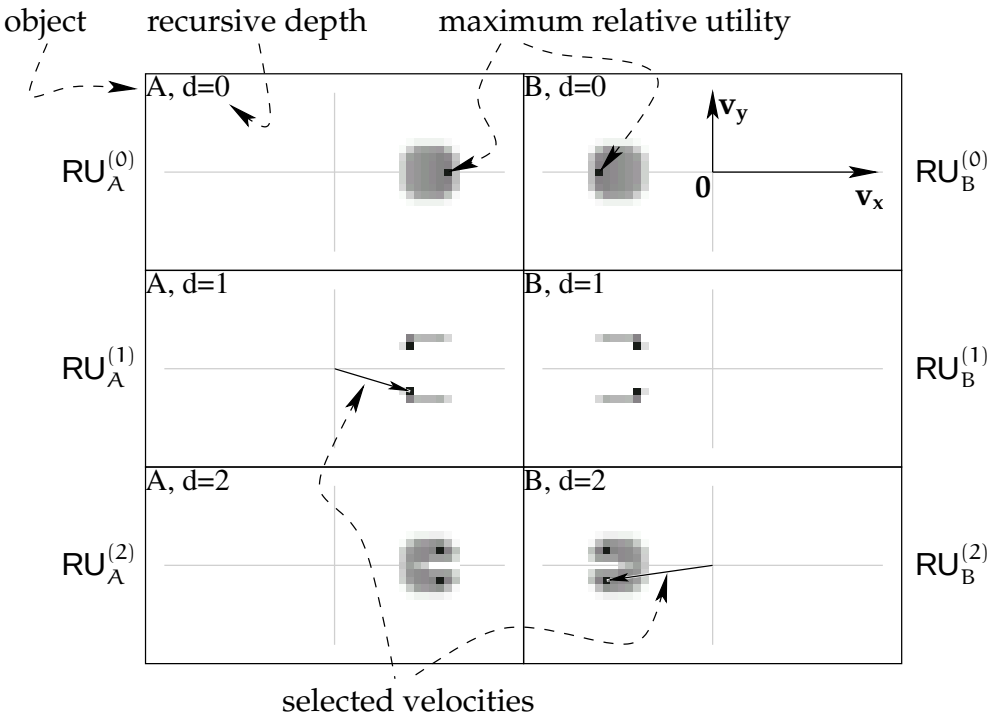
The approach has been evaluated in different simulated situations, including (a) two objects on a collision course, (b) a faster object approaching and overtaking a slower object, and (c) two objects encountering each other close to a static obstacle, see Figure 2.

(a) Collision course (b) Overtaking (c) Static obstacle

Figure 2: Situations for RPVO simulation



(a) Resulting motion



(b) Relative utilities and velocity selection

Figure 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 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 3(b). Higher values of relative utility are indicated by darker shades of grey. For better visibility, maximum values are emphasized in black.

4.1 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 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. Therefore agent B is moving more aggressively and with less deviation from its optimal path than agent A.

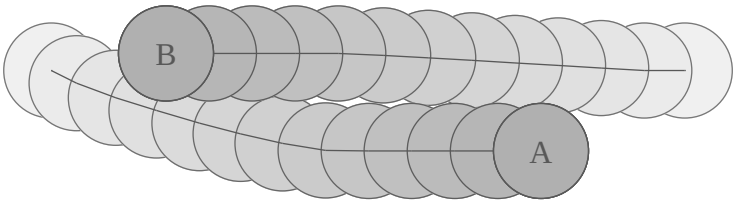
Similarly, Figure 5 shows the encounter of agent A from the left and agent B from the right, but now agent A uses recursive depth $d = 3$, and agent B uses depth $d = 2$ as before. Depth 3 means that agent A assumes agent B to move according to depth 2, i.e. in a somewhat self-confident way, so agent A chooses rather defensive velocities for its motion, and 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 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 4(a) and 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 6.

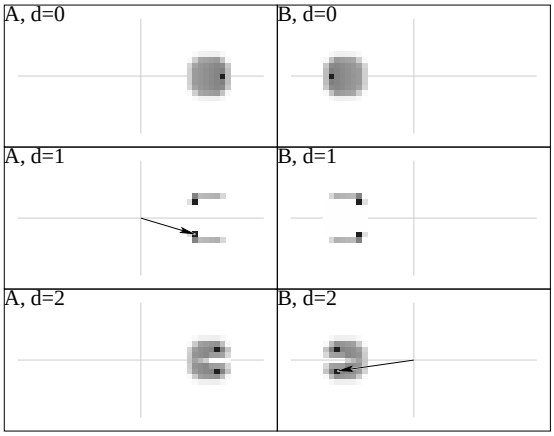
4.2 Overtaking

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.

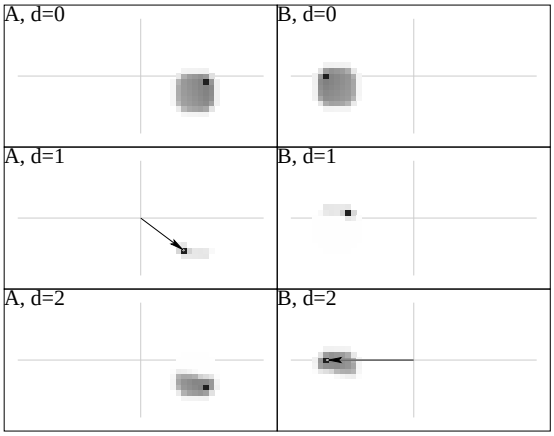
Figure 7 shows the result when agent A uses recursive depth 1 and agent B uses recursive depth 2. Agent B does not leave its optimal path as much as agent A does, which



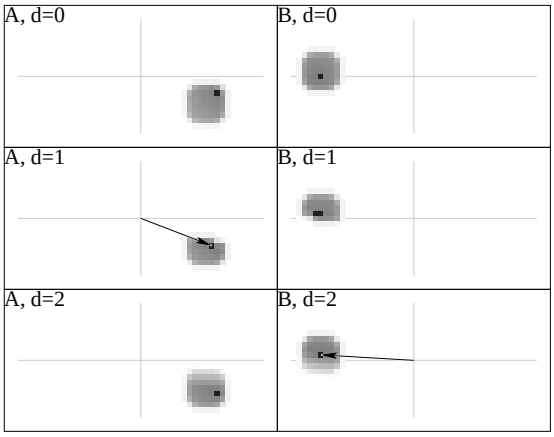
(a) Resulting motion



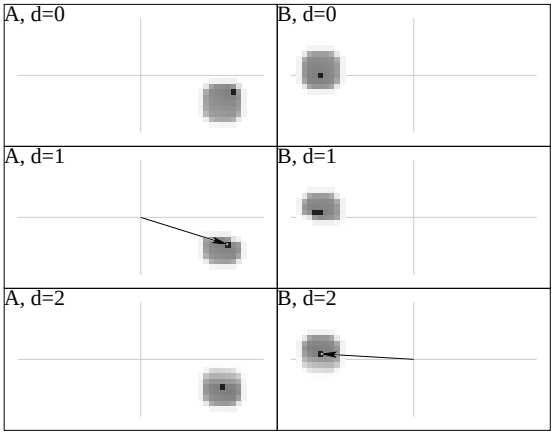
(b) Step 1



(c) Step 2

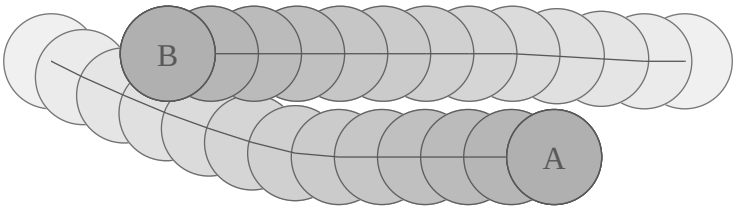


(d) Step 5

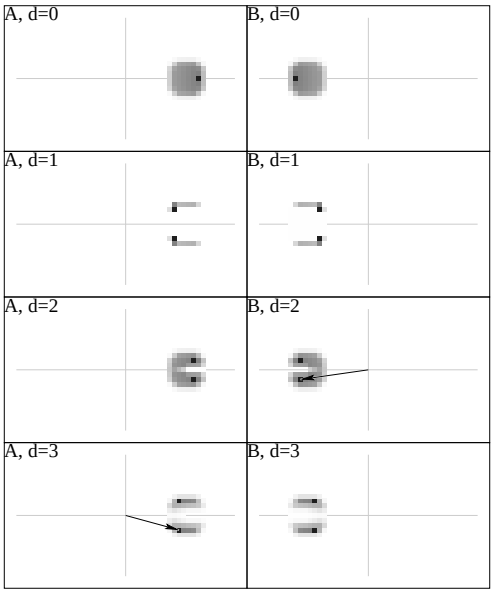


(e) Step 10

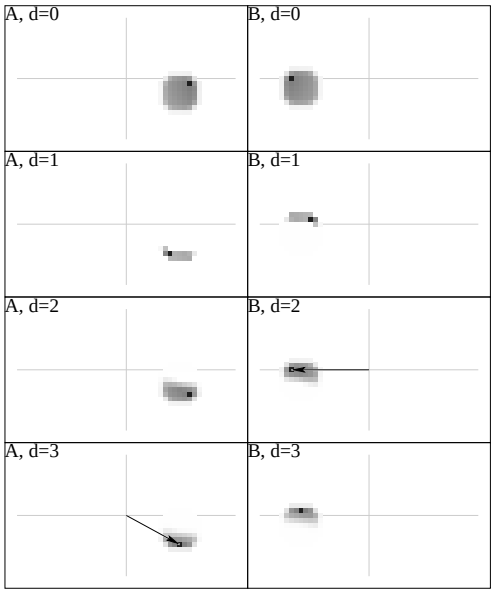
Figure 4: Collision course, object A at depth 1 versus object B at depth 2



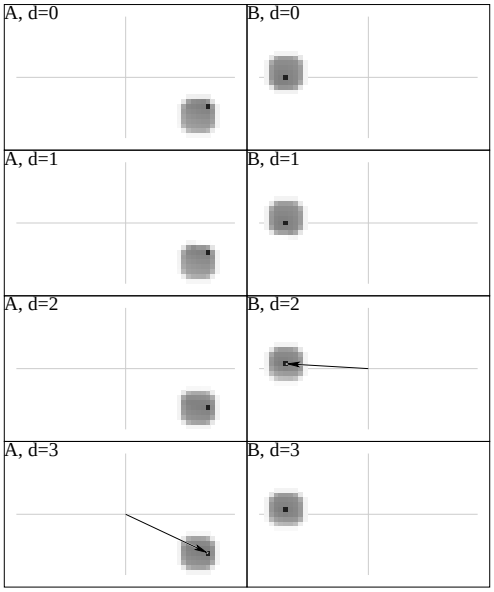
(a) Resulting motion



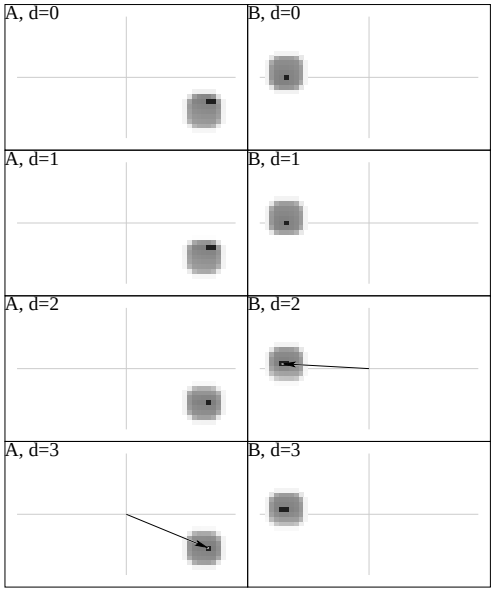
(b) Step 1



(c) Step 2

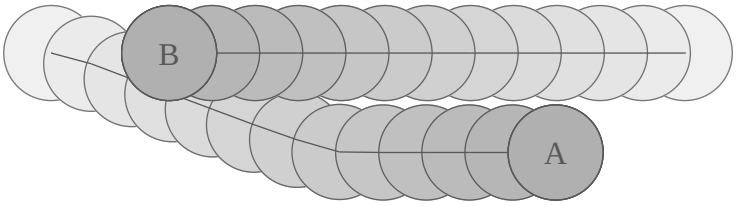


(d) Step 5

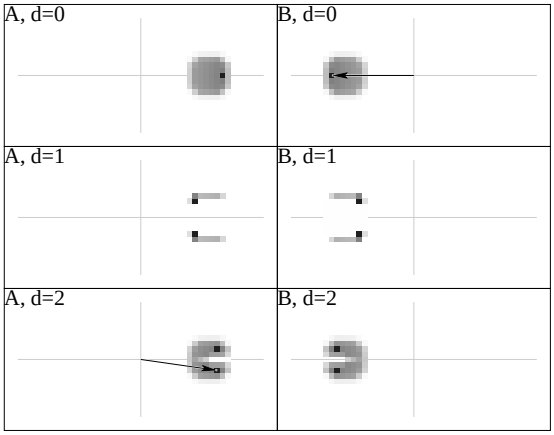


(e) Step 10

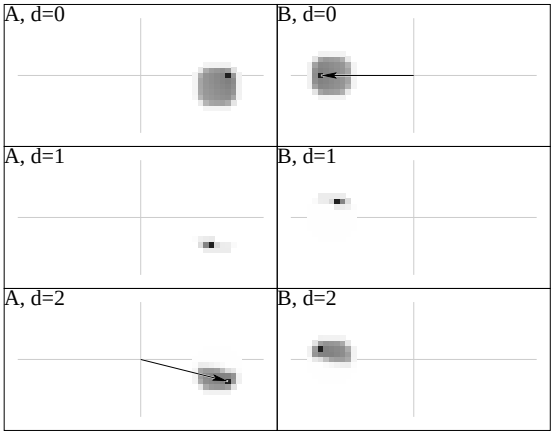
Figure 5: Collision course, object A at depth 3 versus object B at depth 2



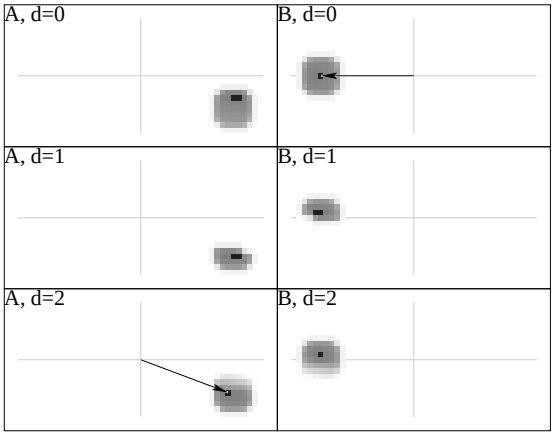
(a) Resulting motion



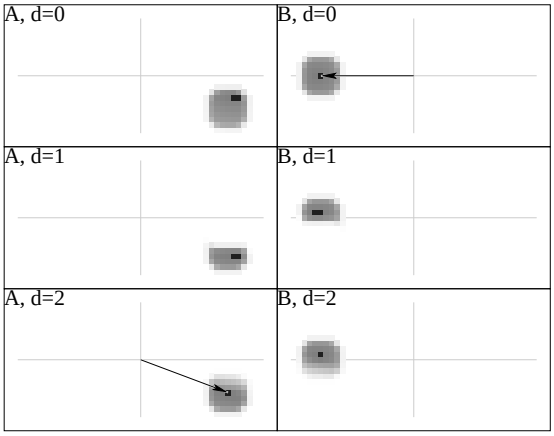
(b) Step 1



(c) Step 2

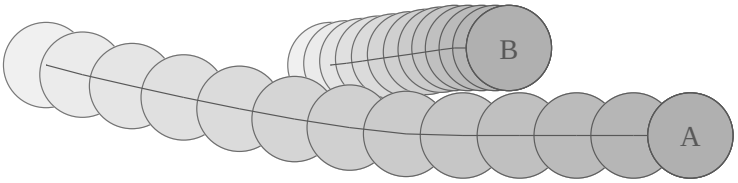


(d) Step 5

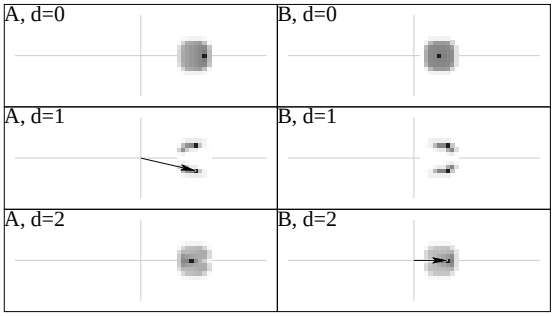


(e) Step 10

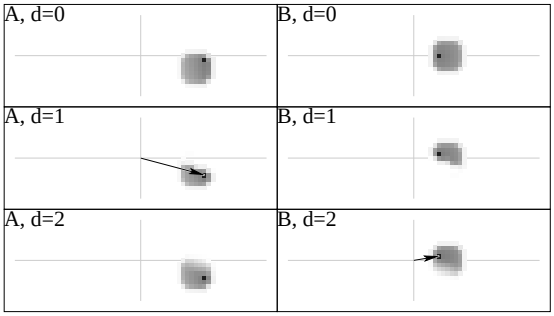
Figure 6: Collision course, object A at depth 2 versus object B at depth 0



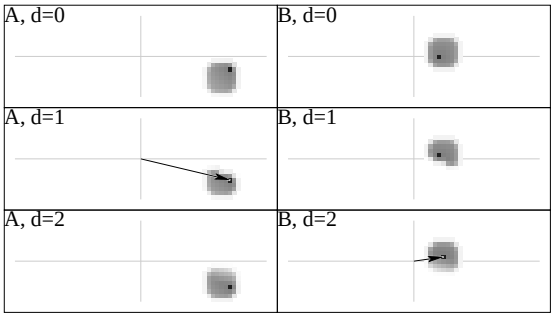
(a) Resulting motion



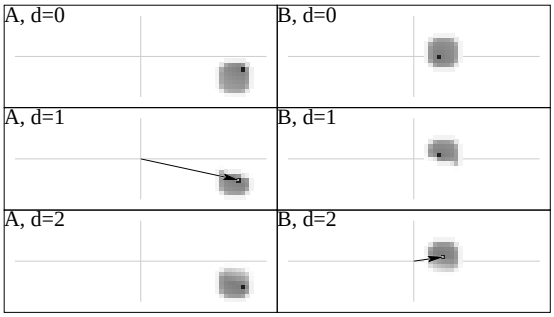
(b) Step 1



(c) Step 2



(d) Step 5



(e) Step 10

Figure 7: Overtaking, object A at depth 1 versus object B at depth 2

is what one expects for 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 and Figure 8. Furthermore, agent B starts to move horizontally again earlier when agent A uses depth 3, which can be seen when comparing Figure 7(a) to Figure 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 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 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).

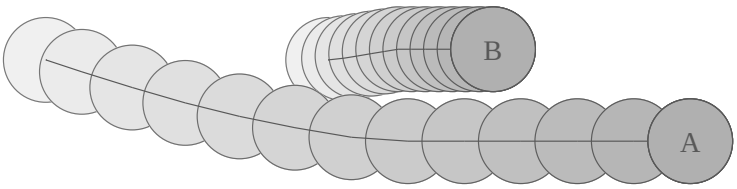
If agent B uses recursive depth 3, it assumes that agent A expects it to avoid collisions, and therefore starts moving out of the way more quickly. As a result, the vertical component of agent A's velocity is smaller in this case, which can be seen when comparing the relative utility (which is centered 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 9(a) and 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 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.

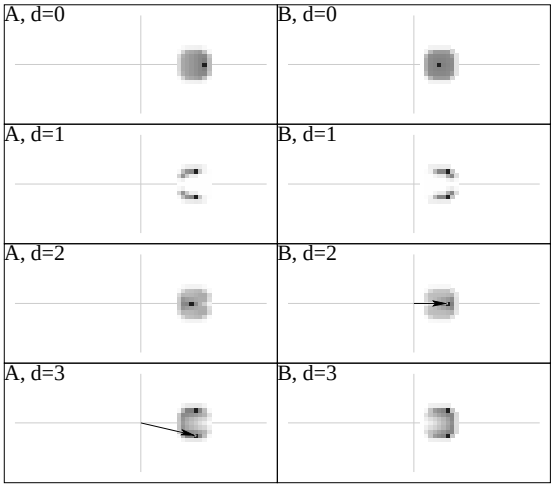
But if both agents use recursive depth $d = 3$, the conflict is solved in a more intelligent way. In a first step, both agents deviate in the same direction in order to avoid the pending collision, see Figure 12. In the next step, agent B still chooses a velocity with a small deviating component, while agent A decides to move horizontally. This asymmetry is amplified during the subsequent steps, such that both agents avoid the collision cooperatively.

4.3 Static Obstacle

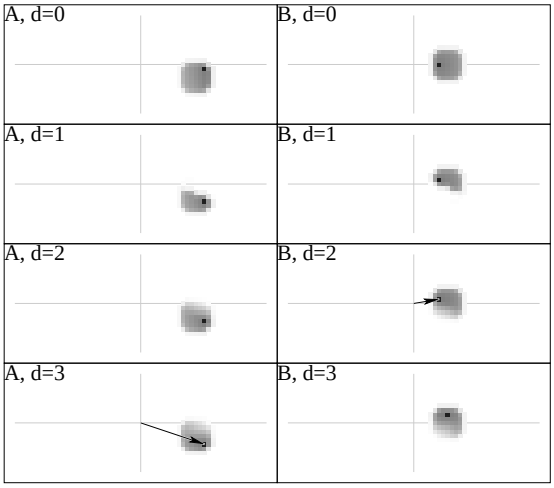
The last type of experiments which we will consider involves two agents moving in opposite directions with an encounter close to a static obstacle.



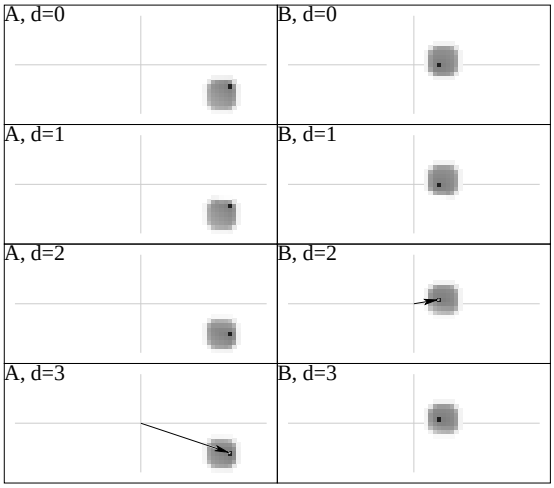
(a) Resulting motion



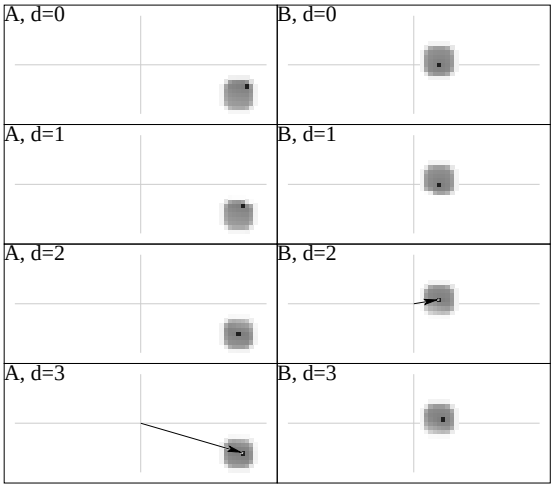
(b) Step 1



(c) Step 2

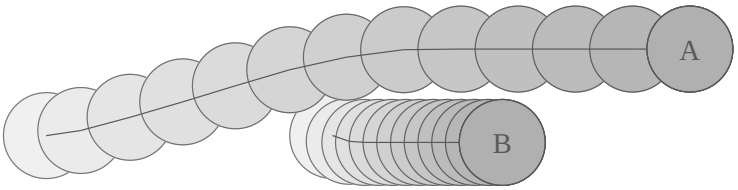


(d) Step 5

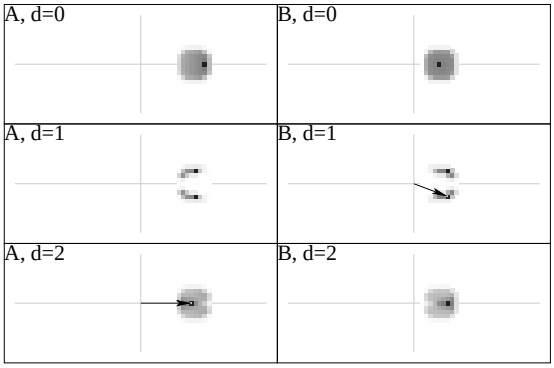


(e) Step 10

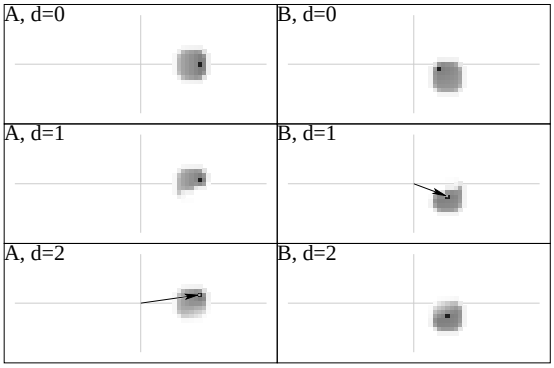
Figure 8: Overtaking, object A at depth 3 versus object B at depth 2



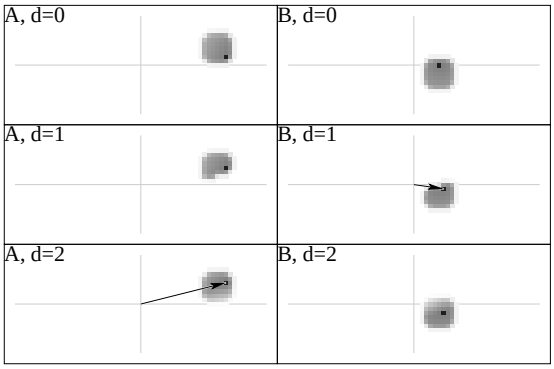
(a) Resulting motion



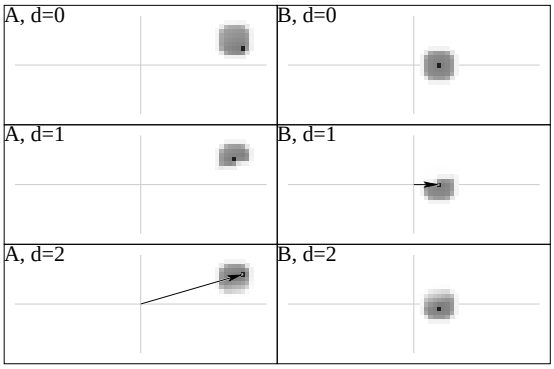
(b) Step 1



(c) Step 2

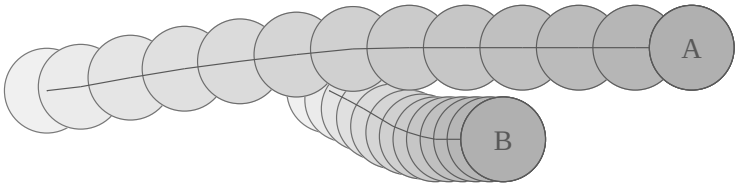


(d) Step 5

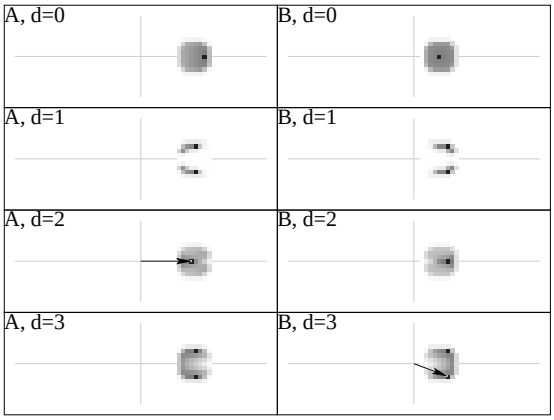


(e) Step 10

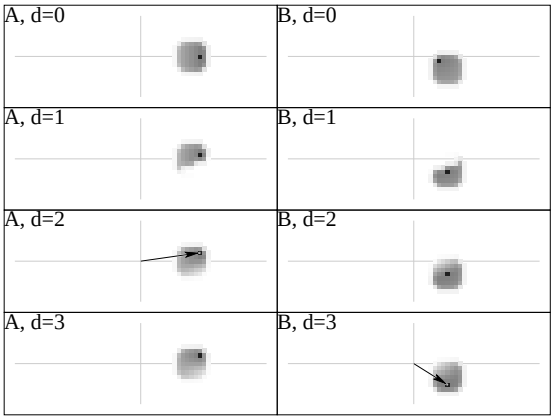
Figure 9: Overtaking, object A at depth 2 versus object B at depth 1



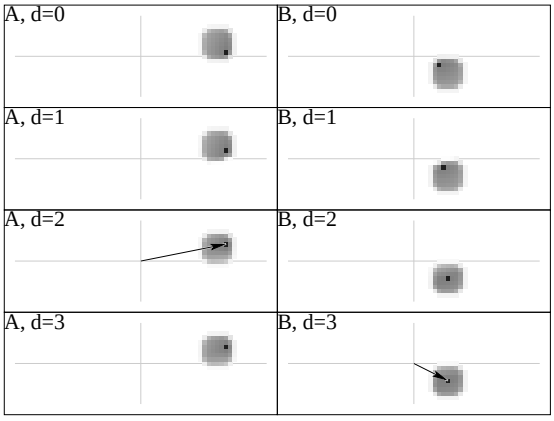
(a) Resulting motion



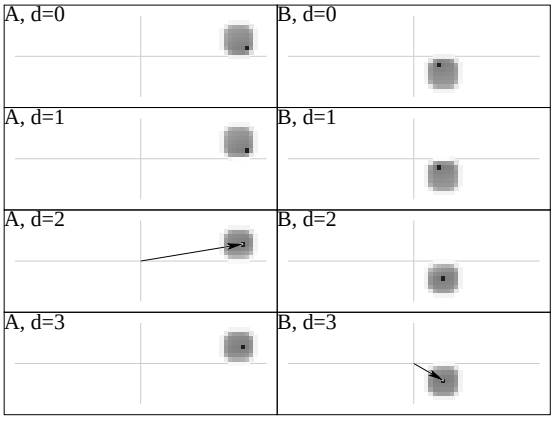
(b) Step 1



(c) Step 2

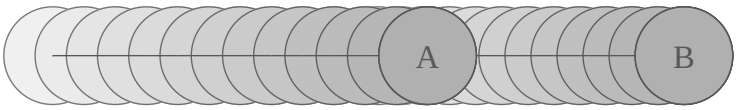


(d) Step 5

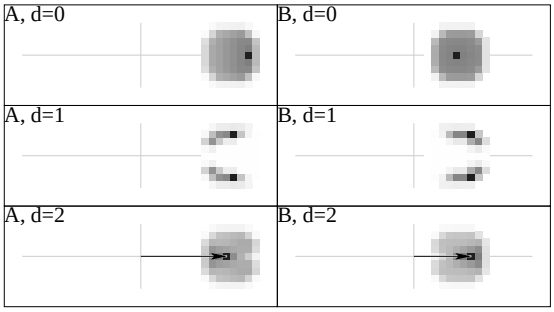


(e) Step 10

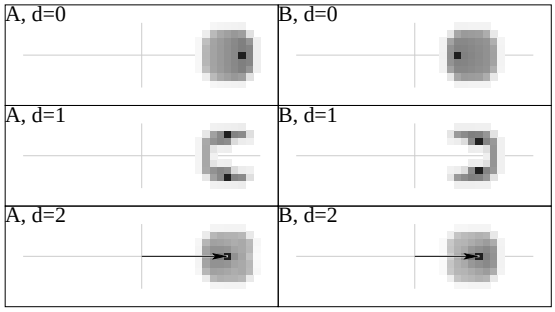
Figure 10: Overtaking, object A at depth 2 versus object B at depth 3



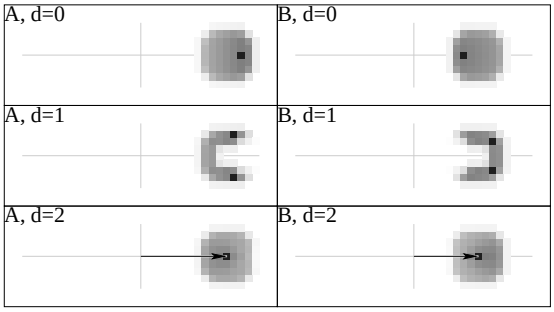
(a) Resulting motion



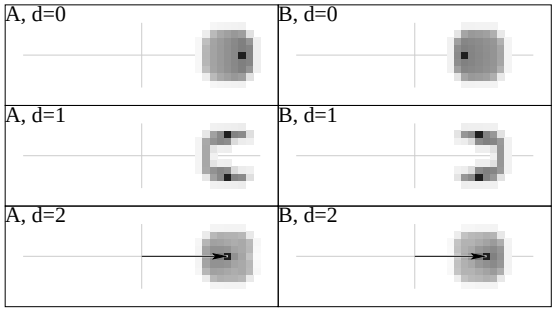
(b) Step 1



(c) Step 2

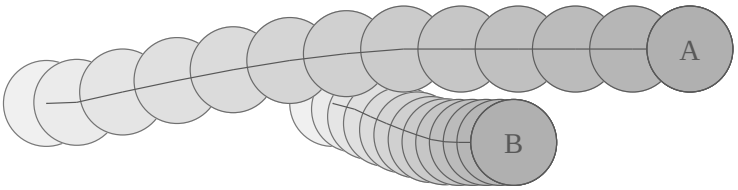


(d) Step 5

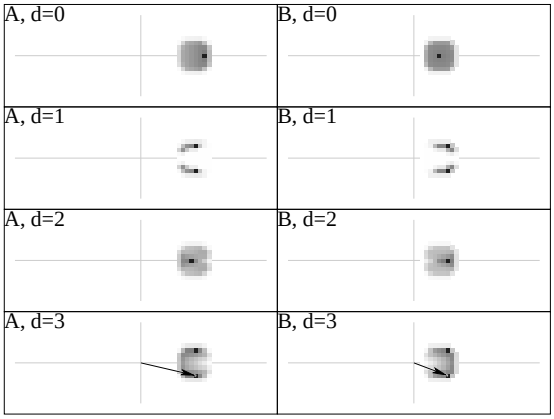


(e) Step 10

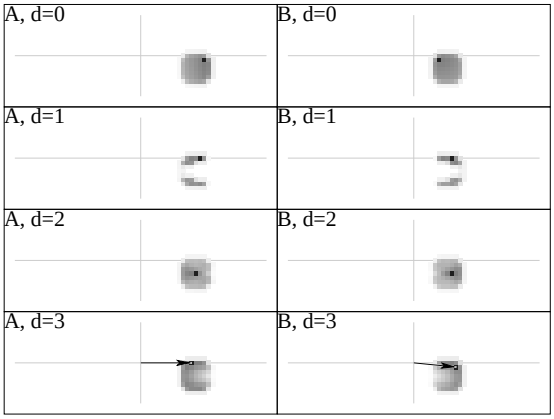
Figure 11: Overtaking, object A at depth 2 versus object B at depth 2



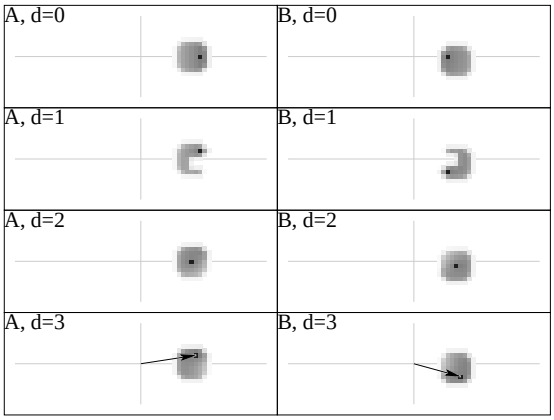
(a) Resulting motion



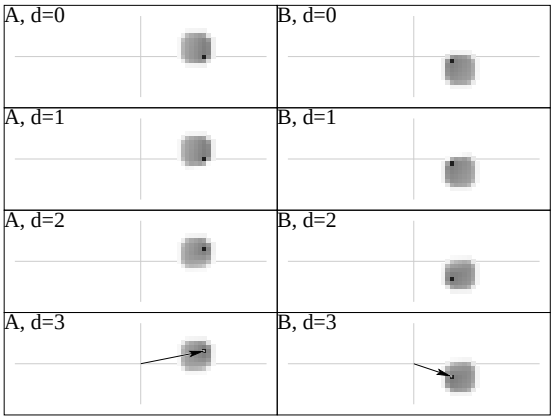
(b) Step 1



(c) Step 2

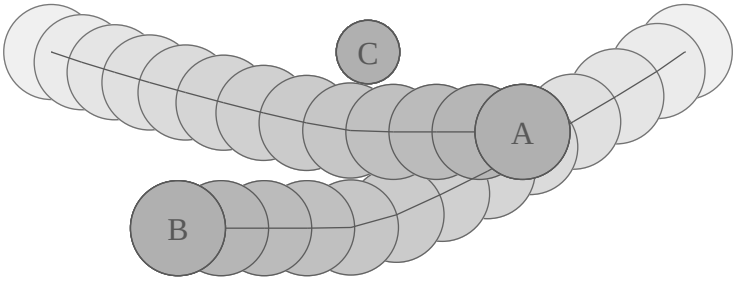


(d) Step 3

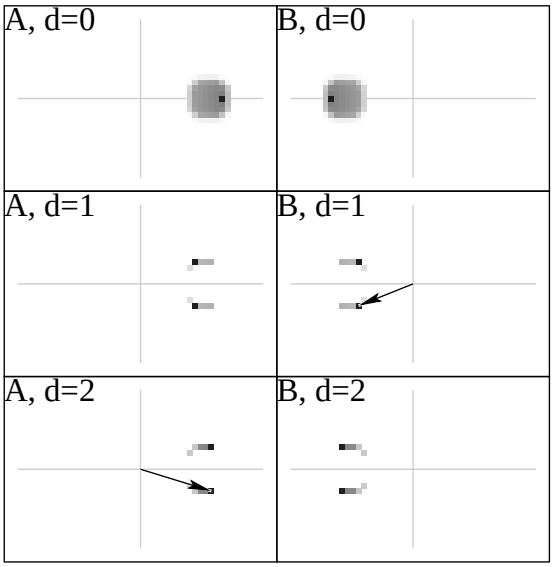


(e) Step 4

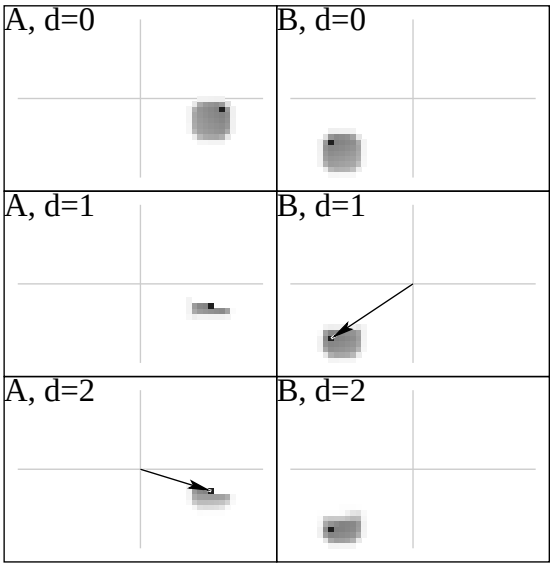
Figure 12: Overtaking, object A at depth 3 versus object B at depth 3



(a) Resulting motion



(b) Step 1



(c) Step 5

Figure 13: Static obstacle, object A at depth 2 versus object B at depth 1

In the first example of this type, agent A uses depth 2 and agent B uses depth 1, see Figure 13. Having seen the examples above, the result of this experiment is not surprising, since agent A using depth 2 is 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 14.

Finally, Figure 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.

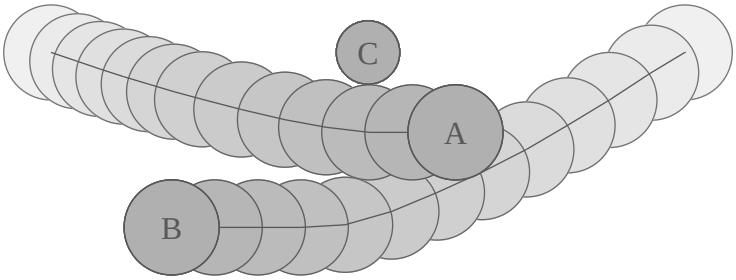
Note that in no case the agents decided to pass by obstacle C on different sides. The reason is the way a velocity with maximum relative utility is selected. A simple approach is to accept the first velocity with that property, when (discrete) velocities are considered in their lexical order, resulting 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.

5 Discussion

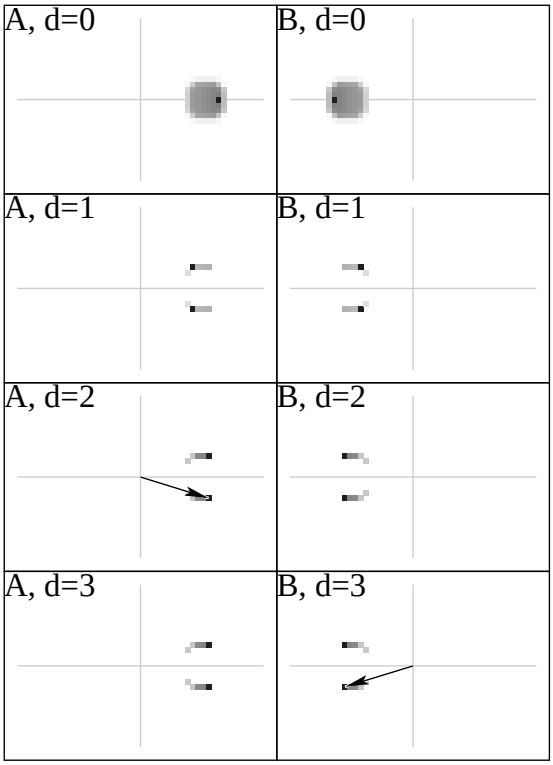
To navigate a mobile robot B_i using depth-d recursive probabilistic velocity obstacles, we repeatedly choose a velocity $\mathbf{v_i}$ maximizing $RU_i^{(d)}$. For $d = 0$, we get a behavior that only obeys the robot's utility function U_i and its dynamic capabilities D_i, but completely ignores other obstacles. For $d = 1$, we get the plain probabilistic velocity obstacle behavior as described in Section 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 computationally expensive.

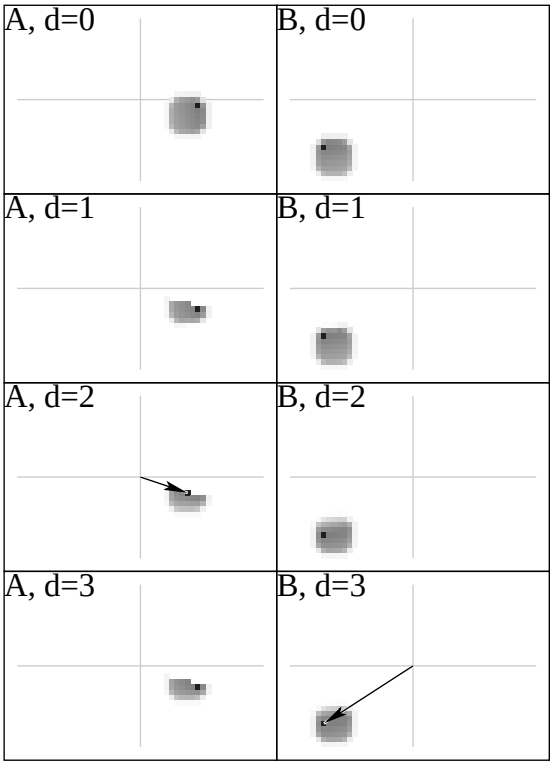
The role of the weights α, β, and γ of the three factors of relative utility is largely unexplored. Some first experiments indicated that they in fact do influence the results, but not in a ground-breaking manner. This might change when the uncertainty about shapes and velocities is increased. During the experiments presented above,



(a) Resulting motion

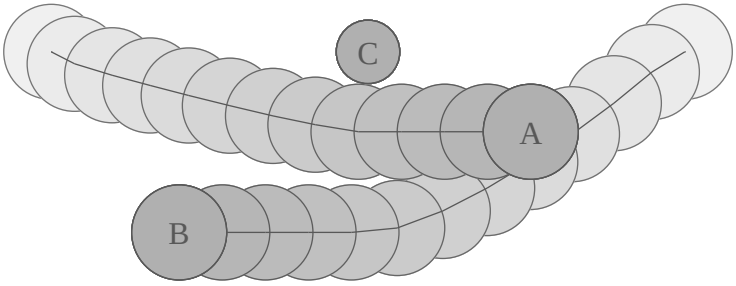


(b) Step 1

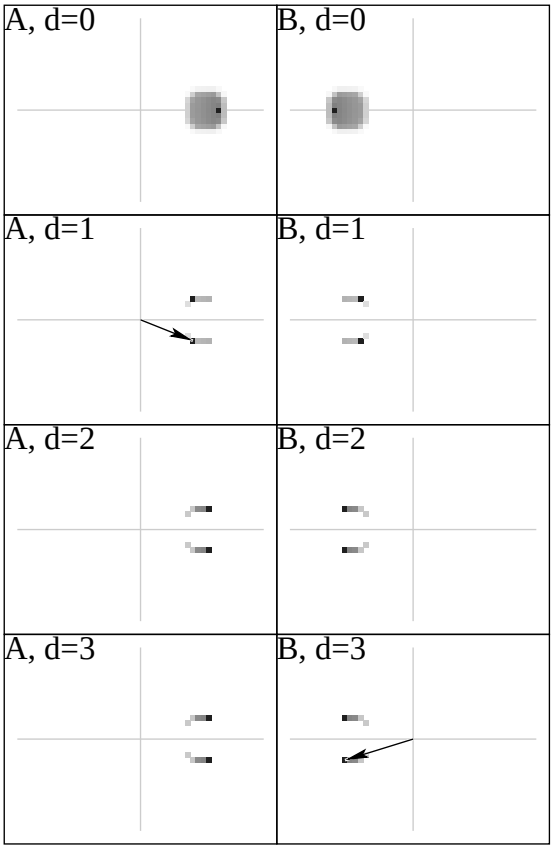


(c) Step 5

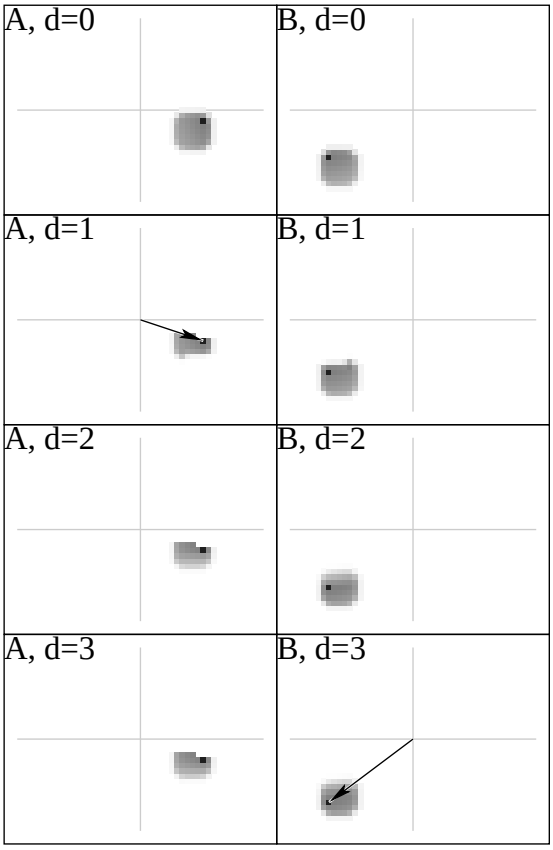
Figure 14: Static obstacle, object A at depth 2 versus object B at depth 3



(a) Resulting motion



(b) Step 1



(c) Step 5

Figure 15: Static obstacle, object A at depth 1 versus object B at depth 3

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.

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).

6 Acknowledgments

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

References

[1] 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.

[2] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, July 1998.

[3] A. F. Foka and P. 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, Oct. 2002.

[4] P. J. Gmytrasiewicz. *A Decision-Theoretic Model of Coordination and Communication in Autonomous Systems (Reasoning Systems)*. PhD thesis, University of Michigan, 1992.

[5] J. Miura and Y. Shirai. Modeling motion uncertainity of moving obstacles for robot motion planning. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[6] Z. Shiller, F. Large, and S. 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.