Recursive Probabilistic Velocity Obstacles for Reflective Navigation

Boris Kluge and Erwin Prassler

Research Institute for Applied Knowledge Processing
Helmholtzstr. 16, 89081 Ulm, Germany
{kluge, prassler}@faw.uni-ulm.de
http://www.faw.uni-ulm.de

Abstract. An approach to motion planning among moving obstacles is presented, whereby obstacles are modeled as intelligent decision-making agents. The decision-making processes of the obstacles are assumed to be similar to that of the mobile robot. A probabilistic extension to the velocity obstacle approach is used as a means for navigation as well as modeling uncertainty about the moving obstacles' decisions.

1 Introduction

For the task of navigating a mobile robot among moving obstacles, numerous approaches have been proposed previously. 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

Some recent approaches (see for example [3,5]) use a prediction of the future motion of the obstacles in order to yield more successful motion (with respect to travel time or collision avoidance). 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 [2] is able to cope with obstacles moving on straight lines and has been extended [6] for the case of 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

S. Yuta et al. (Eds.): Field and Service Robotics, STAR 24, pp. 71–79, 2006.

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

This paper is organized as follows: The basic velocity obstacle approach is introduced in Section 2, and its probabilistic extension is presented in Section 3. Now being able to cope with uncertain obstacle velocities, Section 4 describes how to recursively apply the velocity obstacle scheme in order to create a reflective navigation behavior. An implementation of the approach and an experiment are given in Section 5. After discussing the presented work in Section 6, Section 7 concludes the paper.

2 Velocity Obstacle Approach

Let B_i and B_j be circular objects with centers c_i and c_j and radii r_i and r_j, moving with constant velocities $v_i = \dot{c}_i$ and $v_j = \dot{c}_j$. To decide if these two objects are on a collision course, it is sufficient to consider their current positions together with their relative velocity $v_{ij} = v_i - v_j$, see Fig. 1. Let

$$\hat{B}_{ij} = \{c_j + r \,|\, r \in \mathbb{R}^2, |r| \leq r_i + r_j\}, \tag{1}$$

$$\lambda_{ij}(v_{ij}) = \{c_i + \mu v_{ij} \,|\, \mu \geq 0\}. \tag{2}$$

Then B_i and B_j are on a collision course, if and only if $\hat{B}_{ij} \cap \lambda_{ij} \neq \emptyset$.

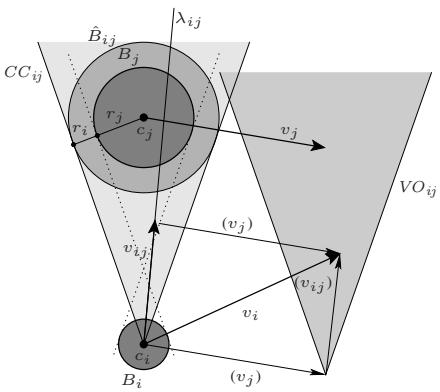


Fig. 1. Collision cone and velocity obstacle

Therefore we can define a set of colliding relative velocities, which is called the *collision cone* CC_{ij}, as

$$CC_{ij} = \{v_{ij} \,|\, \hat{B}_{ij} \cap \lambda_{ij}(v_{ij}) \neq \emptyset\}. \tag{3}$$

In order to be able to decide if an absolute velocity v_i of B_i leads to a collision with B_j, we define the *velocity obstacle* of B_j for B_i to be the set

$$VO_{ij} = \{v_i \,|\, (v_i - v_j) \in CC_{ij}\}, \tag{4}$$

which is, in other words,

$$VO_{ij} = v_j + CC_{ij}. \tag{5}$$

Now for B_i, any velocity $v_i \in VO_{ij}$ will lead to a collision with B_j, and any velocity $v_i \notin VO_{ij}$ for B_i will avoid any collision with B_j.

In general, object B_i is confronted with more than one other moving object. Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ the set of moving objects under consideration. The velocity obstacle of $\mathcal{B}$ for B_i is defined as the set

$$VO_i = \cup_{j \neq i} VO_{ij}. \tag{6}$$

For any velocity $v_i \notin VO_i$, object B_i will not collide with any other object.

Finally, a simple navigation scheme based on velocity obstacles (VO) can be constructed as following. The moving and non-moving obstacles in the environment are continuously tracked, and the corresponding velocity obstacles are repeatedly computed. In each cycle, a velocity is chosen which avoids collisions and approaches a motion goal, for example a maximum velocity towards a goal position.

3 Probabilistic Velocity Obstacles

The velocity obstacle approach as presented in the preceeding section can be extended to deal with uncertainty in shape and velocity of the objects. This allows to reflect the limitations of real sensors and object tracking techniques.

3.1 Representing Uncertainty

Uncertainty in the exact shape of an object is reflected by uncertainty in the corresponding collision cone. Therefore, we define the *probabilistic collision cone* of object B_j relative to object B_i to be a function

$$PCC_{ij} : \mathbb{R}^2 \rightarrow [0, 1] \tag{7}$$

where $PCC_{ij}(v_{ij})$ is the probability of B_i to collide with B_j if B_i moves with velocity v_{ij} relative to B_j.

Similarily, we represent the *uncertain velocity* of object B_j by a probability density function

$$V_j : \mathbb{R}^2 \rightarrow \mathbb{R}_0^+. \tag{8}$$

With these two definition, we get the *probabilistic velocity obstacle* of object B_j relative to object B_i as a function

$$PVO_{ij} : \mathbb{R}^2 \rightarrow [0, 1] \tag{9}$$

which maps absolute velocities v_i of B_i to the according probability of colliding with B_j. It is

$$\begin{aligned} PVO_{ij}(v_i) &= \int_{\mathbb{R}^2} V_j(v) PCC_{ij}(v_i - v) d^2v \\ &= (V_j * PCC_{ij})(v_i) \end{aligned} \tag{10}$$

where $*$ denotes the convolution of two function.

3.2 Probabilistic Velocity Obstacle

The probability of B_i colliding with any other obstacle when traveling with velocity v_i is the probability of not avoiding collisions with each other moving obstacle. Therefore we may define the *probabilistic velocity obstacle* for B_i as the function

$$PVO_i = 1 - \prod_{j \neq i} (1 - PVO_{ij}) \,. \tag{11}$$

3.3 Navigating with Probabilistic VO

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 now, we are confronted with two objectives: reaching a goal and minimizing the probability of a collision.

Let $U_i : \mathbb{R}^2 \rightarrow [0, 1]$ a function representing the utility of velocities v_i for the motion goal of B_i. However, the full utility of a velocity v_i is only attained if (a) v_i is dynamically reachable, and (b) v_i is collision free. Therefore we define the relative utility function

$$RU_i = U_i \cdot D_i \cdot (1 - PVO_i), \tag{12}$$

where $D_i : \mathbb{R}^2 \rightarrow [0, 1]$ describes the reachability of a new velocity.

Now a simple navigation scheme for object B_i based on probabilistic velocity obstacles (PVO) is obtained by repeatedly choosing a velocity v_i which maximizes the relative utility RU_i.

4 Recursive Probabilistic VO

In contrast to traditional approaches, recursive modeling for mobile robot navigation as presented in this paper presumes moving obstacles to deploy navigational decision

making processes similar to the approach used by the robot. This means any object B_j is assumed to take actions maximizing its relative utility function RU_j. Therefore, in order to predict the action of obstacle B_j, we need to know its current utility function U_j, dynamic capabilities D_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 obstacle strives to approach, expecting it to be willing to keep its current speed and heading. By continuous observation of a moving obstacle it is also possible to deduce a model of its dynamics, which describes feasible accelerations depending on its current speed and heading. Further details about aquiring models of velocity utilities and dynamic capabilities of other objects are beyond the scope of this paper.

Finally, the velocity obstacle PVO_j for object B_j is computed in a recursive manner, where predicted velocities are used in all but the terminal recursion step.

4.1 Formal Representation

Let $d \in \ltimes$ the current recursive depth. Then the following equation

$$RU_j^d = \begin{cases} U_j D_j (1 - PVO_j^{d-1}) & \text{if } d > 0, \\ U_j D_j & \text{else,} \end{cases} \tag{13}$$

expresses that each object is assumed to derive its relative utility from recursive PVO considerations, and equation

$$V_j^d = \begin{cases} 1/w \, RU_j^d & \text{if } d > 0, \\ & w = \int_{\mathbb{R}^2} RU_j^d(v) d^2 v \text{ exists,} \\ & \text{and } w > 0, \\ V_j & \text{else.} \end{cases} \tag{14}$$

expresses the assumption that objects will move according to their relative utility function. Probabilistic velocity obstacles PVO_j^d of depth $d \geq 0$ are computed in the obvious way from depth-d models of other objects' velocities.

Computational demands will increase with the depth of the recursion, but, intuitively, one does not expect recursion depths of more than two or three to be of broad practical value, since such deeper modeling is not observed when we are walking as human beings among other humans.

4.2 Navigation with Recursive PVO

To navigate a mobile robot B_i using depth-d recursive probabilistic velocity obstacles, we repeatedly choose a velocity 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 3. Finally, for $d > 1$, the robot starts modeling the obstacles as perceptive and decision making.

5 Implementation

The implementation is given by Algorithm 1. Recursive function calls are not used, the models are computed starting from depth zero up to a predefined maximum depth.

Algorithm 1 RPVO(depth r, n objects)

1: **Input:** for $i, j = 1, \ldots, n, j \neq i$
   - object descriptions for PCC_{ij}
   - velocities V_i
   - dynamic capabilities D_i
   - utilities U_i

2: **for** $i = 1, \ldots, n$ **do**
3: $V_i^0 \leftarrow V_i$
4: $RU_i^0 \leftarrow D_i U_i$
5: **end for**
6: **for** $d = 1, \ldots, r$ **do**
7: **for** $i = 1, \ldots, n$ **do**
8: $RU_i^d \leftarrow D_i U_i \prod_{j \neq i} (1 - V_j^{d-1} * PCC_{ij})$
9: $w \leftarrow \int_{\mathbb{R}^2} RU_i^d(v) d^2 v$
10: **if** $w > 0$ **then**
11: $V_i^d \leftarrow (1/w) RU_i^d$
12: **else**
13: $V_i^d \leftarrow V_i^0$
14: **end if**
15: **end for**
16: **end for**
17: **Output:** recursive models V_i^d and RU_i^d for $i = 1, \ldots, n$ and $0 \leq d \leq r$

For implementation, objects like uncertain velocities, probabilistic collision cones and velocity obstacles have to be discretized. A function f is called discrete with respect to a partition $\Pi = \{\pi_1, \pi_2, \ldots\}$ of $\mathbb{R}^2$ if the restriction of f to any $\pi_i \in \Pi$ is constant. We assume a unique partition for all functions in the algorithm. The supporting set $\sigma(f) \subseteq \Pi$ of a discrete function f is the set of cells $\pi_i \in \Pi$ where f is non-zero.

5.1 Complexity

We begin the complexity assessment by measuring the sizes of the supporting sets of the discretized functions used in Algorithm 1. Line 4 implies

$$\sigma(RU_i^0) \subseteq \sigma(D_i), \tag{15}$$

and from line 8 follows

$$\sigma(RU_i^d) \subseteq \sigma(D_i) \tag{16}$$

for $d > 0$. Line 3 implies

$$\sigma(V_i^0) = \sigma(V_i), \tag{17}$$

and from lines 11 and 13 follows

$$\sigma(V_i^d) \subseteq \sigma(D_i) \cup \sigma(V_i) \tag{18}$$

for $d > 0$, using the three preceding Equations.

Now we count the numbers of operations used in the algorithm, which we write down using $N_i = |\sigma(D_i) \cup \sigma(V_i)|$. Line 8 can be implemented to use $\mathcal{O}(N_i \cdot \sum_{j \neq i} N_j)$ operations. Lines 9, 11, and 13 each require $\mathcal{O}(N_i)$ operations, and are thus dominated by line 8. Therefore the loop starting in line 7 requires

$$\mathcal{O}(\sum_{i=1}^{n}(N_i \sum_{j \neq i} N_j)) \tag{19}$$

operations, and the loop starting in line 6 requires

$$\mathcal{O}(r \sum_{i=1}^{n}(N_i \sum_{j \neq i} N_j)) \tag{20}$$

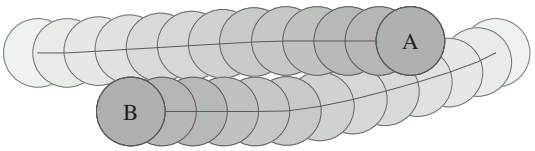
operations. The complexity of the loop starting in line 6 clearly dominates the complexity of the initialization loop starting in line 2. Therefore Equation 20 gives an upper bound of the overall time complexity of our implementation. That is to say the dependence on the recursion depth is linear, and the dependence on the number of objects is $\mathcal{O}(n^2)$.

5.2 Experiments

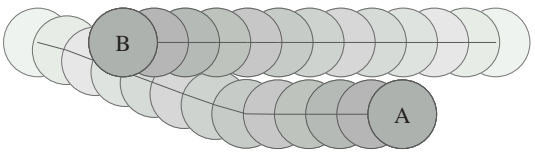
A simulation of a dynamic environment has been used as a testbed for the presented approach. Results for some example situations are given below.

For two objects initially on an exact collision course, Fig. 2 shows the resulting motion for two pairs of depths. In Fig. 2(a) object A (depth 2) correctly models object B (depth 1) to be able to avoid collisions. As a result, object A stays on its course, forcing object B to deviate. Now if object A's assumption failed, i.e. object B won't avoid collisions, Fig. 2(b) shows that object A is still able to prevent a crash in this situation.

Another example is the situation where a fast object approaches a slower one from behind, i.e. where overtaking is imminent, which is depicted by Fig. 3. If the slow object B (depth 2) expects the fast object A (depth 1) from behind to avoid collisions, it mostly stays in its lane (Fig. 3(a)). On the other hand, if object B (now depth 3) assumes other objects to expect it to avoid collisions, a rather defensive driving style is realized (Fig. 3(b)).

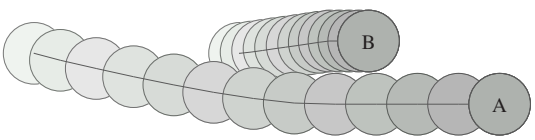


(a) Object A at depth 2 vs. object B at depth 1

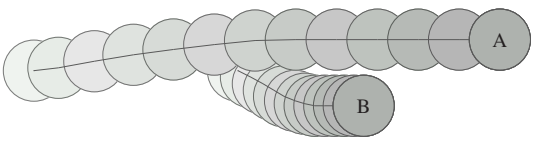


(b) Object A at depth 2 vs. object B at depth 0

Fig. 2. Collision Course Examples



(a) Object A at depth 1 vs. object B at depth 2



(b) Object A at depth 2 vs. object B at depth 3

Fig. 3. Overtaking Examples

6 Discussion

Considering the experiments from the previous section, the nature of a robot's motion behavior appears to be adjustable by changing the evaluation depth. Depth 1 corresponds to a plain collision avoidance behavior. A robot using depth 2 will reflect on its environment and is able to exploit obstacle avoiding capabilities of other moving agents. This sometimes results in more aggresive navigation, which nevertheless may be desireable in certain situations: a robot which is navigating too defensively will surely get stuck in dense pedestrian traffic. In general, depth 3 seems to be more appealing, since a robot usign that level of reflection assumes that the other agents expect it to avoid collisions, which results in a very defensive behavior with anticipating collision avoidance.

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.

That is to say, 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.

7 Conclusion

An approach to motion coordination 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.

The presented 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. Secondly, the perception and decision making of other objects is modeled and included in the own decision making process.

Due to its relfective capabilities, the proposed navigation scheme may represent an interesting option for mobile robots sharing the environment with humans.

Acknowledgment

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, October 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.