A Stable Qualitative Movement Control System based on Propositional Dynamic Logic

Przemysław Andrzej Wałęga[1]*, Emilio Muñoz-Velasco[2]†

[1] *University of Warsaw, Institute of Philosophy, Poland,*
[2] *Universidad de Málaga, Departsmento de Matemática Aplicada, Spain.*

We present a qualitative movement control system based on Propositional Dynamic Logic (PDL). Firstly, we obtain an stable qualitative representation by means of hysteresis loops. Second, we build the qualitative movement control system, in order to represent relative movement of an object with respect to another, that is capable to perform qualitative composition of qualitative relations. Third, we express the control rules of the system by using PDL. Finally, we implement the framework in Robotic Operating System (ROS) and test it with the computer simulator STAGE, which indicate possibility to use our system in real world applications.

Key words: Qualitative Reasoning, Spatial Reasoning, Movement Control, Collision Avoidance, Mobile Robots, Propositional Dynamic Logic.

1 INTRODUCTION

Qualitative Reasoning (QR) [6] is an area of AI where the reasoning task with concepts represented usually by numbers (such as velocity, orientation, distance) is performed by using qualitative values (small, medium, large, close, distant, etc.) instead of the exact values. In fact, humans perform the majority of their tasks without knowing the exact values of velocity, distance, angles, etc. For instance, when you are driving a car, you do not need to know the

* email: `p.a.walega@gmail.com`
† email: `emilio@ctima.uma.es`

exact value of your velocity, distance to the other cars, etc. Instead, you use rules such as: if you are approaching *fast* the car in front of you and the car is *very close* to you, you have to slow down. Qualitative data are simpler and easier to obtain and transmit. This is crucial in cases where an immediate decision needs to be made, such as in movement control systems, which are designed to control vehicles or robotic manipulators – see, e.g., [5]. In this context, QR has been applied to Qualitative Spatial Reasoning, which is focused on representing and reasoning with spatial entities (topology, orientation, shape, size, distance, etc.) with use of qualitative techniques. Qualitative Spatial Reasoning has numerous applications, such as [10, 11].

The use of logic in AI, (and, in particular in QR) may improve the capability of formal representation of real world problems, and provides an interesting point of view in order to face different reasoning tasks. Moreover, *logic-ideas* as theorem-proving and model-construction techniques are present in AI. For this reason, we will use the Propositional Dynamic Logic (PDL) for dealing with movements presented in [13] and exploit specifically two of these advantages: on the one hand, its *expressivity*, that will allow us to consider the special relations in order to control the movement; on the other hand, its capability to reason, in order to obtain *new information* by using the *composition* of movements.

When we consider movement control systems (for instance, for a robot or a car), the information gathered by sensors can be noisy or inaccurate due to external factors such as bad weather conditions, low quality of the sensors, etc. Additionally, sometimes spatial reasoning needs to be performed using incomplete data and intuitive information. In general, this information is easy to understand for humans but not for machines. For instance, when a human being is catching a ball, there is a lot of intuitive and incomplete information about the position, velocity and direction of the ball, which is very difficult and slow to process for one machine.

In this paper, we present an approach for obtaining a stable qualitative representations (Section 2) which is particularly important when the system is dealing with noisy data. Our method uses the hysteresis system which overcomes the problem of fast changes of the qualitative representation. We compare this method with the basic approach for qualitative representation, i.e., using non-overlapping intervals. Based on the hysteresis system, we build a module for relative qualitative movement representation (Section 3.1) based on the one introduced in [13], which may be treated as a generalization of the approaches for relative movement presented in [4, 12, 17]. Afterwards, we consider the qualitative composition of movements (Section 3.2) by means of the composition tables, which is the basic reasoning method in Qualitative Spatial Reasoning [3]. We construct a table-based control system which, as we show, is expressible in the Propositional Dynamic Logic presented in [13] (Section 3.4). We describe also the implementation of the approach in Robotic Operating System (ROS) [14] (Section 4), which is currently one of the most popular open source frameworks for robot software. Moreover, we present two applications of the approach, namely for a task of following another object and for avoiding collisions and discuss the corresponding experiments performed in robotic simulator STAGE [7] (Section 4), and we finish the paper with some conclusions and prospects of future work (Section 5).

2 QUALITATIVE VALUES

Movement control systems often use qualitative values while reasoning, e.g., about a distance between objects or their velocity. For instance, in order to avoid a collision it might be enough to know if the obstacle is *close* or *far away*, whereas information about an exact distance might not be needed. One of the most common methods to represent qualitative values is constituted by means of non-overlapping subintervals [6]. However, such method is affected by small changes of the underlying quantitative values in borderline cases (i.e., when a quantitative value is on the border between two qualitative values). We present the method of non-overlapping subintervals and a hysteresis system in order to avoid these changes in the border of the subintervals.

2.1 Non-overlapping Subintervals

The method consists of dividing the continuous scale of values into a finite number of non-overlapping subintervals, where each subinterval is denoted by its two boundary values (also called distinguishing values or landmark values), i.e., the beginning of the subinterval and its end. As depicted in Figure 1, we can represent the subintervals by means of a function that assigns values from the continuous scale of values (x) to qualitative values ($\mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3$). The function presented in the figure translates values from the subinterval $[0, x_1)$ to a qualitative value q_0, the ones from $[x_1, x_2)$ to a qualitative value q_2, etc. Usually the subintervals cover the whole scale which means that the set of qualitative values becomes jointly exhaustive and pairwise disjoint. As a result, every value from the continuous scale belongs to exactly one quality. The above property enables to use algorithms that reason by exclusion which implies that powerful programming methods can be performed.

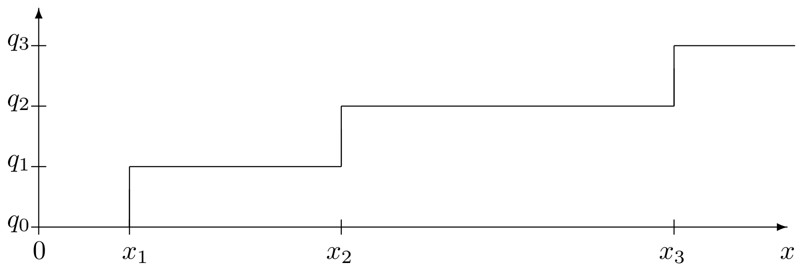


Figure 1: Non-overlapping subintervals as a function from continuous scale to qualitative values.

2.2 Hysteresis System

In order to obtain a stable qualitative representation, we present a novel method based on a nonlinear system with hysteresis loops[⋆] . The aim of the approach is to obtain a representation which lacks of frequent changes even if the input contains noisy quantitative data. Our solution is to extend the non-overlapping subintervals approach with hysteresis loops located on the borders of the subintervals as depicted in Figure 2. To be more precise, we introduce sharp hysteresis loops which enable to distinguish the borderline value between qualitative values in case of increasing (denoted by x_{i+}) and decreasing (denoted by x_{i-}) quantitative values. As a result, the number of hysteresis loops is always equal to the number of qualitative values decreased by 1.

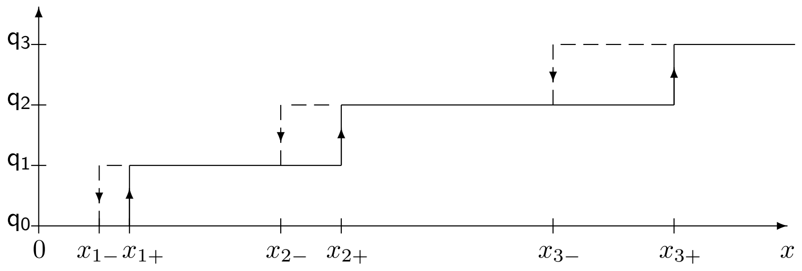


Figure 2: Hysteresis method for translating quantitative data into qualitative values.

As an example, let us consider the translation system presented in Figure 2. There are 4 qualitative values, namely $\mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3$. The nonlinear system consists of 3 hysteresis located on the borders of qualitative class, i.e., be-

[⋆] For an exhaustive analysis of hysteresis loops and nonlinear systems in general see [9].

tween q_0 and q_1, between q_1 and q_2, etc. We describe now how the system prevents from frequent changes of qualitative values. Consider noisy data that need to be mapped into qualitative values. We compare a result of using (a) non-overlapping subintervals translation function from Figure 1 and (b) hysteresis method from Figure 2. Assume that x_i from the non-overlapping subintervals approach equals x_{i+} from hysteresis approach.

(a) If the starting quantitative value is 0, then the starting qualitative value is q_0. Afterwards the quantitative value continuously increases up to a value of x_i which is when the qualitative values changes from q_0 to q_1. Then (e.g., because of noisy data), the measured quantitative value slightly decreases below x_i increases above x_i, and so on. As a consequence, the qualitative value changes between q_0 and q_1 which in turn may lead to unwanted behaviour of the system using such a representation.

(b) Again, the qualitative value equals q_0. Afterwards the quantitative value continuously increases, becomes x_{1-} then increases even more, up to $x_{1+} = x_i$ which is when the qualitative values changes to q_1. To this point there is no difference between (a) and (b). However later on the noisy part of data occurs. Although the quantitative value decreases below x_{1+} the qualitative values does not change (it would change only if the quantitative value becomes less then x_{1-}). As a result the noise does not lead to frequent changes of the qualitative representation, as in case of (a).

The most straight forward way of establishing hysteresis is such that

$$(x_{i+} - x_{i-}) > noise_{avr},$$

where $noise_{avr}$ is the average value of the noise – may be computed, e.g., from previously gathered data or inferred from quality of sensor. On the other hand, the narrower the hysteresis, the more granulated will be the representation. Hence, the most desirable system is constructed with a hysteresis width only slightly bigger than the $noise_{avr}$.

In some qualitative reasoning systems [6, 13], there are additional limitations on the length of qualitative values, e.g., in [13] it is needed that every next qualitative value (more precisely a subinterval corresponding to that qualitative value) is at least twice longer than the previous one. In other words, $2 \cdot x_{i+} \leq x_{(i+1)+}$. Such a restriction decreases the number of possible outcomes of summing qualitative values – for instance adding values from

q_0 and q_1 cannot result in q_3. Then in order to obtain desirable properties of the reasoning system, the width of the hysteresis should also fulfil a similar condition, i.e., $2 \cdot (x_{i+} - x_{i-}) \leq (x_{(i+1)+} - x_{(i+1)-})$. We exploit the same restrictions in our approach.

3 QUALITATIVE MOVEMENT CONTROL SYSTEM

Common methods for robot motion planing in dynamic environment are based on computation of Velocity Obstacle (VO) [2, 16], i.e., a set of robot velocities resulting in potential collisions and select robot velocity that is outside VO. As a result, a first order method based on current position and velocity of the robot or a second order method that takes into account the current velocity and path curvature of the moving obstacle [16] are constructed. Even more complex systems are obtained by further modifications, for instance, by means of Kalman-based observer for estimating the obstacles velocities [2].

In what follows, we present a qualitative table-based controlling approach which is much simpler and human-like than mentioned systems for robot motion planing. The approach uses the above introduced hysteresis system for translating quantitative information into qualitative values. Afterwards, we explain how to infer new information about the movement by means of composition tables. Then, we describe a control movement approach which is expressible in the PDL introduced in [13].

3.1 Movement Representation

In this section, we explain how the movement is represented in our approach. We will use the ideas presented in [13], where the movement of an object with respect to another[†] is given by a tuple of qualitative values which represent, respectively, absolute velocity, absolute orientation (of the movement), relative direction of movement, allowed directions of movement and relative position. This set of parameters enables us to express a number of complex scenarios – see [13]. Formally, the movement of one object with respect to another is represented by a tuple $(x_1, \ldots, x_7) \in L$, where $L = L_1 \times \cdots \times L_7$ is described in Table 1.

It is worth mentioning that the movement representation enables us to capture uncertainty of information, e.g. the following velocity description $x_2 = \{\mathsf{v}_1, \mathsf{v}_2\}$ has an intuitive meaning that the velocity is slow or normal. As an example of movement representation let us consider the tuple presented

[†] The objects may be moving as well as static (obstacles).

Table 1: Description of the tuple L used in $\mathsf{PDL}_{\mathsf{M}}^{\mathsf{F}}$ movement representation.

	Description
$L_1 = A \times A$	states that the tuple describes movement of the first object of the L_1 ordered pair with respect to the second one, where $A = \{\mathsf{A}_1, \ldots, \mathsf{A}_\mathsf{n}\}$ is a set of all objects
$L_2 = 2^{\{\mathsf{v}_\mathsf{o}, \mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3\}}$	first object absolute velocity (zero v_o, slow v_1, normal v_2, quick v_3)
$L_3 = 2^{\{\mathsf{o}_\mathsf{o}, \mathsf{o}_1, \mathsf{o}_2, \mathsf{o}_3, \mathsf{o}_4\}}$	absolute orientation of the first object's movement (unknown o_o, North o_1, South o_2, East o_3, West o_4)
$L_4 = 2^{\{0,-,+\}} \times 2^{\{0,-,+\}}$	relative movement direction which consists of the first object movement relative to second object (stable 0, moving towards $+$, moving away from $-$) and analogously second object movement with respect to the first one
$L_5 = 2^{\{\mathsf{o}_\mathsf{o}, \mathsf{o}_1, \mathsf{o}_2, \mathsf{o}_3, \mathsf{o}_4\}}$	allowed absolute orientations of first object movement (unknown o_o, North o_1, South o_2, East o_3, West o_3)
$L_6 = 2^{\{\mathsf{o}_1, \mathsf{o}_2\}} \times 2^{\{\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_3\}}$	latitude position of first object with respect to the second one which consists of orientation (North o_1, South o_2) and distance (zero d_0, close d_1, normal d_2, distant d_3)
$L_7 = 2^{\{\mathsf{o}_3, \mathsf{o}_4\}} \times 2^{\{\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2, \mathsf{d}_3\}}$	longitude position of first object with respect to the second one which consists of orientation (East o_3, West o_4) and distance (zero d_0, close d_1, normal d_2, distant d_3)

by:[‡]

$$\mathsf{A}_\mathsf{i}, \mathsf{A}_\mathsf{j}; \mathsf{v}_2\mathsf{v}_3; \mathsf{o}_3; +-; \mathsf{o}_1\mathsf{o}_3; \mathsf{o}_1, \mathsf{d}_1\mathsf{d}_2; \mathsf{o}_3, \mathsf{d}_2.$$

This tuple represents a movement of an object A_i with respect to A_j; A_i has a normal or quick velocity, $\mathsf{v}_2\mathsf{v}_3$; and east orientation of movement o_3; A_i is moving away from A_j, $+$, and A_j is moving towards A_i, $-$; A_i possible orientations of movement are north or east, $\mathsf{o}_1\mathsf{o}_3$; A_i is to the north at a close or normal distance with respect to A_j, $\mathsf{o}_1, \mathsf{d}_1\mathsf{d}_2$ (qualitative latitude); and A_i is to the east at a normal distance with respect to A_j, $\mathsf{o}_3, \mathsf{d}_2$ (qualitative longitude). Notice that there is a difference between the real orientation (direction) of the movement of the first object (represented by L_3) and the allowed orientations

[‡] Henceforth, for a better reading, we eliminate the curly brackets in the sets and the parenthesis in the tuples, and we use semicolon "; " to separate two consecutive components of the movement.

(directions) of this movement (given by L_5). For instance, when a car is moving North on a highway, its real orientation L_3 is North (o_1), while its allowed directions may be North, East and West ($\mathsf{o}_1\mathsf{o}_3\mathsf{o}_4$) in order to allow the car to change lanes, but it is not allowed move South (o_2), because it may cause an accident.

3.2 Composition Tables

Composition tables constitute the most common method for obtaining new information in Qualitative Spatial Reasoning [3] . In our system (see Table 2), we consider the *composition of movements*, i.e., given the movement description of an object A_i with respect to A_j and a description of A_j movement with respect to A_k, we will infer movement description of A_i with respect to A_k.

Notice that, in the last row and column of the Table 2, for the composition of two movements with the same orientation o_r in the latitude (or, similarly longitude) and distances d_s and d_u, will be a movement with the same orientation o_r, and the distance obtained as the qualitative sum of d_s and d_u, which is the maximum of these two qualitative values, i.e., $\mathsf{d}_{\max\{\mathsf{s},\mathsf{u}\}}$. This is a specific choice in our system, motivated for the applications we have taken into consideration so far (presented in Section 4), but could be changed easily for other applications. The method provides "safe" reasoning (the smallest possible distance between object and obstacle is always taken into account). Notice that such a behavior is highly desirable when safety is considered as the most important goal, e.g., in the collision avoidance system that will be introduced in what follows.

Table 2: Composition table for qualitative latitude (longitude), where $r \in \{1,2,3,4\}$ and $s,u \in \{0,1,2,3\}$.

$\mathsf{A}_\mathsf{i}\mathsf{A}_\mathsf{j}$ \ $\mathsf{A}_\mathsf{j}\mathsf{A}_\mathsf{k}$	$\mathsf{o}_\mathsf{r},\mathsf{d}_0$	$\mathsf{o}_\mathsf{r},\mathsf{d}_\mathsf{u}$
$\mathsf{o}_\mathsf{r},\mathsf{d}_0$	$\mathsf{o}_\mathsf{r},\mathsf{d}_0$	$\mathsf{o}_\mathsf{r},\mathsf{d}_\mathsf{u}$
$\mathsf{o}_\mathsf{r},\mathsf{d}_\mathsf{s}$	$\mathsf{o}_\mathsf{r},\mathsf{d}_\mathsf{s}$	$\mathsf{o}_\mathsf{r},\mathsf{d}_{\max\{\mathsf{s},\mathsf{u}\}}$

3.3 Table-based Decision

We introduce a movement control system which is able to modify the velocity and orientation of the object's movement, in order to achieve the desired goal (mainly collision avoidance). We focus on the modification of velocity, because of the applications presented in the next section. However, the same ideas can be applied to control the orientation of the movement.

The change of the velocity of object $\mathsf{A_i}$ (denoted by $v_{\mathsf{A_i}}$) is determined by the distance and the velocity difference between the $\mathsf{A_i}$ and the closest obstacle. We make use of the PDL introduced in [13] and introduce three relations (called *programs* in the context of PDL) for modifying the velocity denoted by Dec, Inc and Man, meaning decreasing, increasing and maintaining the velocity, respectively. The table-based control system is presented in Table 3, where d denotes qualitative distance between the object and the closest obstacle, whereas dv stands for the qualitative velocity difference between the object and the closest obstacle.

Table 3: The movement control rules.

d \ dv	v_{-3}	v_{-2}	v_{-1}	v_0	v_1	v_2	v_3
d_0	Man	Man	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$
d_1	$\boldsymbol{Inc}$	Man	Man	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$
d_2	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	Man	Man	Man	$\boldsymbol{Dec}$	$\boldsymbol{Dec}$
d_3	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$	$\boldsymbol{Inc}$

The velocity difference dv can take seven qualitative values, namely: v_{-3}, v_{-2}, v_{-1} ,v_0, v_1, v_2, v_3, meaning zero, slow, normal and quick, with positive and negative values. The table is interpreted as a set of so called "if–then" rules in order to decide which program needs to be used, given the values of v and dv. For instance, the element of the second row and seventh column can be interpreted as follows: if the object and the obstacle are close and the object is moving *much faster* than the obstacle, then *decrease* the velocity of the object, that is:

"If $d = \mathsf{d}_1$ and $dv = \mathsf{v}_3$ then use the program Dec".

The intuitive meaning of all control rules from the Table 3 can be explained in the following general way:

- if the controlled object moves much faster than the obstacle or the distance between them is small, then the controlled object should decrease its velocity (Dec),
- if the controlled object moves much slower than the obstacle or the distance between them is big, then the controlled object should increase its velocity (Inc),

- otherwise maintain the velocity of the controlled object (Man).

The movement control system enables to maintain a safe distance between the controlled object and the obstacle. Therefore, it may be used in such applications as avoiding collisions with obstacles or following another object, as we will see in section 4. The rules presented in the Table 3 are based on these applications however, they can be easily adapted to other applications related to controlling movements.

3.4 Using Logic in our system

Firstly, we present the Syntax and Semantics logic used in our approach, that is, the PDL for movements presented in [13].

The PDL for movements

The language of logic consists of a set of formulas Φ and a set of programs Π, which are defined recursively on disjoint sets Φ_0 and Π_0, respectively. Φ_0 is called the set of *atomic formulas* which can be thought of as abstractions of properties of states. Similarly, Π_0 is called the set of *atomic programs* which are intended to represent basic instructions:

- $\Phi_0 = \mathbb{V} \cup L$, where $\mathbb{V}$ is a denumerable set consisting of propositional variables and $L = L_1 \times \cdots \times L_7$, intended to represent atomic labels.

- If φ and ψ are formulas and a is a program, then $\varphi \rightarrow \psi$ (propositional implication), $\perp$ (propositional falsity) and $[a]\varphi$ (program necessity) are also formulas. As usual, $\vee$ and $\wedge$ represent logical disjunction and conjunction, respectively; whereas $\langle a \rangle$ represents program possibility.

- The set Π_0 of *specific* programs is defined as follows:

$$\begin{aligned} \Pi_0 \quad = \quad & \{\mathtt{rev}_x \mid x \in L\} \cup \{\otimes_{x,y} \mid x, y \in L\} \cup \\ & \cup \{\mathtt{Dec}^{\mathsf{s}}_x, \mathtt{Man}^{\mathsf{s}}_x, \mathtt{Inc}^{\mathsf{s}}_x \mid \mathsf{s} \in \{0,1,2,3,4\}, x \in L\}. \end{aligned}$$

- If a and b are programs and φ is a formula, then $(a; b)$ ("do a followed by b"), $a \cup b$ ("do either a or b, nondeterministically"), a^* ("repeat a a nondeterministically chosen finite number of times") and $\varphi?$ ("proceed if φ is true, else fail") are also programs.

The intuitive meaning of programs $\mathtt{rev}_x$ is considered to be the *reverse* of the movement x, that is, if x represents a movement of $\mathsf{A_i}$ with respect to $\mathsf{A_j}$, then $\mathtt{rev}_x$ is the movement of $\mathsf{A_j}$ with respect to $\mathsf{A_i}$. In addition, $\otimes_{x,y}$

is *compose* the movement labeled by x, with the movement labeled by y. Moreover, programs $\texttt{Dec}^{\mathsf{s}}_x$, $\texttt{Man}^{\mathsf{s}}_x$, and $\texttt{Inc}^{\mathsf{s}}_x$ for $\mathsf{s} \in \{0, 1, 2, 3, 4\}$ have the intuitive meaning of *modifying* (increasing, maintaining, or decreasing) the velocity of the movement labeled by x and change its orientation towards o_s, for $\mathsf{s} = 1, 2, 3, 4$ (to the North, South, East and West, resp.).

The *semantics* of the logic is defined as usual in PDL [1]. We focus only in the definition of the specific programs $\texttt{Dec}^{\mathsf{s}}_x$, $\texttt{Man}^{\mathsf{s}}_x$ and $\texttt{Inc}^{\mathsf{s}}_x$, used in our system. For every $\mathsf{s} \in \{0, 1, 2, 3, 4\}$, and $x = (x_1; \dots; x_7) \in L$:

- $m(\texttt{Dec}^{\mathsf{s}}_x)(m(x)) \subseteq m(y)$, where[¶] $y = (y_1; \dots; y_7) \in L$, being $y_1 = x_1$,
$$y_2 = \begin{cases} \mathsf{v}_{\mathsf{k}_1 - 1} \dots \mathsf{v}_{\mathsf{k}_r - 1} & \text{if } x_2 = \mathsf{v}_{\mathsf{k}_1} \dots \mathsf{v}_{\mathsf{k}_r}, \mathsf{k}_1 > 0 \\ \mathsf{v}_0 \mathsf{v}_{\mathsf{k}_1 - 1} \dots \mathsf{v}_{\mathsf{k}_r - 1} & \text{if } x_2 = \mathsf{v}_0 \mathsf{v}_{\mathsf{k}_1} \dots \mathsf{v}_{\mathsf{k}_r} \end{cases} \quad \text{and } y_3 = \mathsf{o}_\mathsf{s}.$$

- $m(\texttt{Man}^{\mathsf{s}}_x)(m(x)) \subseteq m(y)$, where $y = (y_1; \dots; y_7)$, being $y_1 = x_1$, $y_2 = x_2$ and $y_3 = \mathsf{o}_\mathsf{s}$.

- $m(\texttt{Inc}^{\mathsf{s}}_x)(m(x)) \subseteq m(y)$, where $y = (y_1; \dots; y_7)$, being $y_1 = x_1$,
$$y_2 = \begin{cases} \mathsf{v}_{\mathsf{k}_1 + 1} \dots \mathsf{v}_{\mathsf{k}_r + 1} & \text{if } x_2 = \mathsf{v}_{\mathsf{k}_1} \dots \mathsf{v}_{\mathsf{k}_r}, \mathsf{k}_r < 3 \\ \mathsf{v}_{\mathsf{k}_1 + 1} \dots \mathsf{v}_{\mathsf{k}_r + 1} \mathsf{v}_3 & \text{if } x_2 = \mathsf{v}_{\mathsf{k}_1} \dots \mathsf{v}_{\mathsf{k}_r} \mathsf{v}_3 \end{cases} \quad \text{and } y_3 = \mathsf{o}_\mathsf{s}.$$

Notice that the previous definition formalizes the intuitive meaning of $\texttt{Dec}^{\mathsf{s}}_x$ as a binary relation such that u is related to v iff v gives the description of a movement obtained by *decreasing* the velocity and modifying the orientation towards o_s. Similarly for $\texttt{Man}^{\mathsf{s}}_x$ and $\texttt{Inc}^{\mathsf{s}}_x$.

Our specific system

In order to describe the collision avoidance procedure for an object A_i with respect to an object A_j we distinguish the following states:

- "safe" – A_i can increase its velocity without danger of a collision with A_j;

- "stable" – A_i should maintain its velocity in order to keep a safe distance with respect to A_j;

- "danger" – there is a danger of a collision with A_j, therefore A_i should brake.

[¶] Note that the left part of the inclusion represents a relation, $m(\texttt{Dec}^{\mathsf{s}}_x)$, applied to a set, $m(x)$, with the usual meaning of the set of all the elements which are related to some element in $m(x)$.

Given objects A_i, A_j we denote:

$$\varphi_{\mathsf{d}_m,\mathsf{v}_n} = (A_1, A_2; \mathsf{d}_m; \mathsf{v}_n; l_4; l_5; l_6; l_7),$$

where $m \in (0, 3)$, $n \in (-3, 3)$ and l_4, l_5, l_6, l_7 are arbitrary values such that $l_4 \in L_4, l_5 \in L_5, l_6 \in L_6, l_7 \in L_7$. Then we express the above mentioned states by means of the following formulas, corresponding to procedure from Table 3:

$$\begin{aligned}
\varphi_{safe} =& (\varphi_{\mathsf{d}_1,\mathsf{v}_{-3}} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_{-3}} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_{-2}} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_{-3}} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_{-2}} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_{-1}} \\
& \vee \varphi_{\mathsf{d}_3,\mathsf{v}_{-0}} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_1} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_2} \vee \varphi_{\mathsf{d}_3,\mathsf{v}_3}); \\
\varphi_{stable} =& (\varphi_{\mathsf{d}_0,\mathsf{v}_{-3}} \vee \varphi_{\mathsf{d}_0,\mathsf{v}_{-2}} \vee \varphi_{\mathsf{d}_1,\mathsf{v}_{-2}} \vee \varphi_{\mathsf{d}_1,\mathsf{v}_{-1}} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_{-1}} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_0} \\
& \vee \varphi_{\mathsf{d}_2,\mathsf{v}_1}); \\
\varphi_{danger} =& (\varphi_{\mathsf{d}_0,\mathsf{v}_{-1}} \vee \varphi_{\mathsf{d}_0,\mathsf{v}_0} \vee \varphi_{\mathsf{d}_0,\mathsf{v}_1} \vee \varphi_{\mathsf{d}_0,\mathsf{v}_2} \vee \varphi_{\mathsf{d}_0,\mathsf{v}_3} \vee \varphi_{\mathsf{d}_1,\mathsf{v}_0} \\
& \vee \varphi_{\mathsf{d}_1,\mathsf{v}_1} \vee \varphi_{\mathsf{d}_1,\mathsf{v}_2} \vee \varphi_{\mathsf{d}_1,\mathsf{v}_3} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_2} \vee \varphi_{\mathsf{d}_2,\mathsf{v}_3}).
\end{aligned}$$

The language of PDL enables to express basic programming instructions [1], e.g., "if – then" condition and "while" loop:

$$\begin{aligned}
\textbf{if } \varphi \textbf{ then } \pi_1 \textbf{ else } \pi_2 &= \big((\varphi?; \pi_1) \cup (\neg\varphi?; \pi_2)\big); \\
\textbf{while } \varphi \textbf{ do } \pi &= \big((\varphi?; \pi); \neg\varphi?\big).
\end{aligned}$$

As a result we are able to express the movement control procedure[§] as follows:

$$\begin{aligned}
&\Big(\varphi_{safe}?; \big((\varphi_{safe}?; Inc)^*; \varphi_{stable}?\big)\Big) \\
\cup &\Big(\varphi_{stable}?; \big((\varphi_{stable}?; Man)^*; \neg\varphi_{stable}?\big)\Big) \\
\cup &\Big(\varphi_{danger}?; \big((\varphi_{danger}?; Dec)^*; \varphi_{stable}?\big)\Big),
\end{aligned}$$

which intuitively means that if the state is "safe" then increase the velocity until you obtain the "stable" state. If the state is "stable" then maintain the current velocity until the state changes. Finally if the state is "danger" decrease the velocity until you obtain "stable" state.

Since language of our PDL for movements is expressive enough to describe our movement control approach and it is known [13] that this logic is decidable and possesses a sound and complete axiomatization, there is a possibility of tight integration of both approaches which is one of our future plans.

[§] For simplicity, we omit here the subscripts and superscripts from Inc, Man and Dec, because they will be clear from the context in our scenarios.

4 IMPLEMENTATION AND EXPERIMENTS

The approach is implemented in ROS, which is one of currently most popular open source frameworks for writing robot software [14], capable to control numerous robot platforms, such as: PR2, Robonaut 2, REEM, TurtleBot, iRobot Roomba, Lego Mindstorm and many more. ROS enables us to use the same implementation in real robots and perform tests by using simulators such as STAGE – a 2D freeware robotic simulator [7], which provides a virtual word where different mobile robots together with sensors and objects can be simulated and visualized. Moreover, the implementation may be tested in 3D simulators, e.g., V-rep [15] (which we consider as a future work).

The implementation is divided into several separate modules, called nodes. Every node gathers messages through input topics, processes data and sends messages via output topics to other nodes. The constructed structure of nodes and topics is presented in Figure 3, where the oval shapes denote nodes, while the arrows represent topics.

- The `/stageros` node is the simulator node which publishes information from simulated sensors.
- The node `/qualitative_values` translate quantitative data into qualitative values as described in the Section 2.2.
- Then, the `/composition` node is used in order to infer new information with the method introduced in Section 3.2.
- Afterwards, `/control` modifies the velocity of the object, depending on its own velocity and the velocity and distance to the obstacle, as described in the Section 3.3.

All this information is sent to the simulator. The whole process is in a loop which repeats until the end of the simulation. The STAGE simulator displays objects and informs about collision whenever it occurs.

In what follows, we present two scenarios, namely *following an object* and *collision avoidance*. In both scenarios we will use the following notation, as depicted in Figure 4.:

- $\mathsf{A_i}$, $\mathsf{A_j}$, $\mathsf{A_k}$ denote cars, whereas v_{A_i}, v_{A_j}, v_{A_k} denote their velocities,
- d_{rs} denote the qualitative distance between vehicles $\mathsf{A_r}$ and $\mathsf{A_s}$, where $r, s \in \{i, j, k\}$.

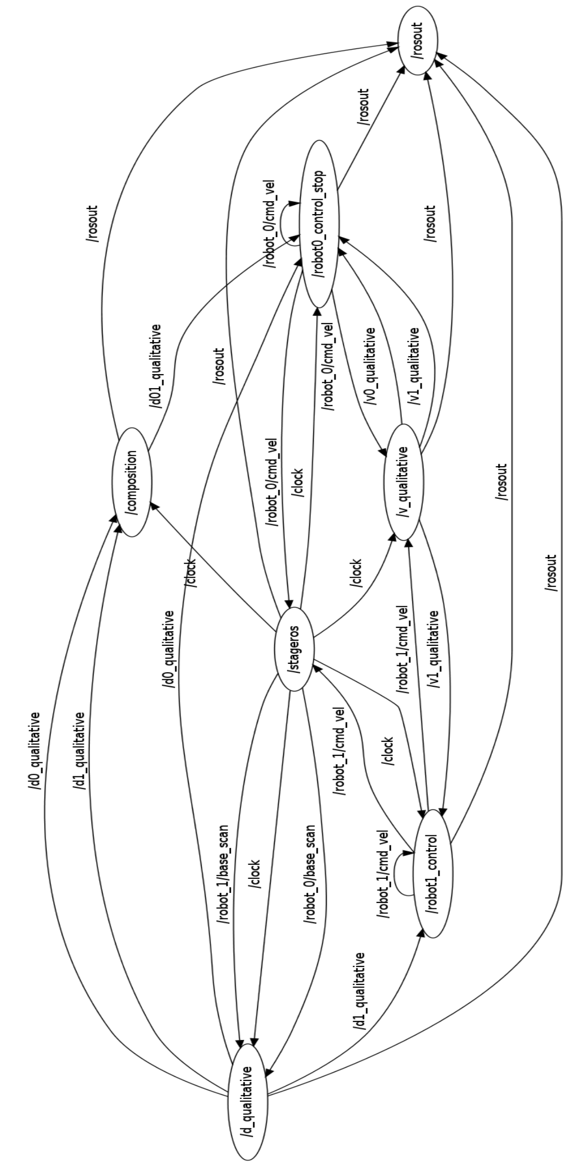


Figure 3: The program graph of our implementation.

Notice that d_{ij} and d_{jk} are obtained after translation of quantitative data from sensors, as explained in Section 2.2. On the other hand, d_{ik} is not available

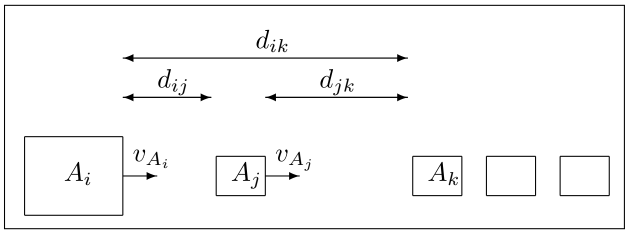


Figure 4: The scenario setup.

from sensors but inferred by means of the composition tables for distances (Table 2). The movement of $\mathsf{A_i}$ is controlled using the information of the distances and the difference of velocities $v_{\mathsf{A_i}}$ and $v_{\mathsf{A_j}}$ (Table 3) – we assume that $\mathsf{A_i}$ has an access to such an information.

4.1 Experiment 1: Following an Object

In the first experiment, we consider only two of objects $\mathsf{A_i}$ and $\mathsf{A_j}$, where $\mathsf{A_j}$ is moving with a constant velocity and direction, whereas $\mathsf{A_i}$ has to follow $\mathsf{A_j}$. The cars behavior is presented in Figure 5 by means of 2 screenshots obtained from the simulator. Notice, that since only qualitative values are used, the exact distance between objects is unknown and as a result, it is impossible to maintain an exact constant distance between the $\mathsf{A_i}$ and the $\mathsf{A_j}$. The distance between them increases and decreases during the simulation (see Figure 8a), however, it always remains safe. Consequently, $\mathsf{A_i}$ follows $\mathsf{A_j}$ without having a collision or letting $\mathsf{A_j}$ go too far from $\mathsf{A_i}$.

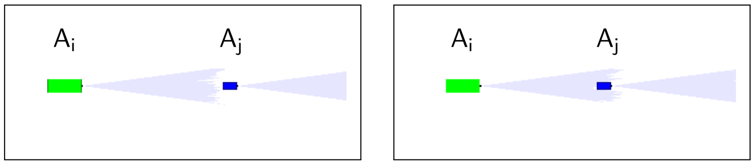
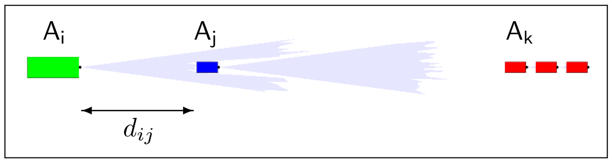


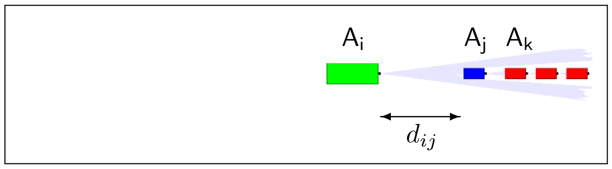
Figure 5: Experiment 1: Following an object. The distance between $\mathsf{A_i}$ and $\mathsf{A_j}$ constantly fluctuates from the situations presented in (a) and (b).

4.2 Experiment 2: Collision Avoidance

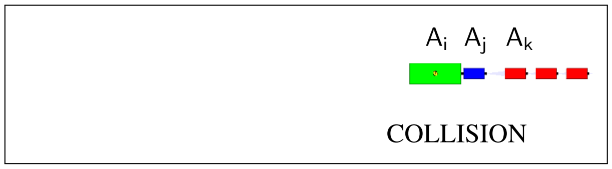
The second experiment models a more complex situation. A number of cars are stuck in a traffic jam, whereas other cars are moving towards the same



(a)



(b)



(c)

Figure 6: Experiment 2, test 1. $\mathsf{A_i}$ is unable to infer d_{ik} and, as a result a collision occurs.

traffic jam and may not be able to stop in order to avoid a collision. Such a situation is common on highways and leads to dangerous collisions of numerous vehicles. The scenario consists of two moving vehicles, namely $\mathsf{A_i}$ and $\mathsf{A_j}$ and cars stuck in a traffic jam with $\mathsf{A_k}$ being the last car in the traffic jam. To make it more complicated, we assume also that $\mathsf{A_i}$ is a truck with long braking distance. We assume that each vehicle has information about its distance to the closest object in front of it. We will study how the information gathered by $\mathsf{A_j}$ (namely, the d_{jk} distance) can be used by $\mathsf{A_i}$ to avoid the collision. We perform two tests: in the first one, we assume that $\mathsf{A_i}$ does not have the information about the distance d_{jk}; while in the second test, this information is available to $\mathsf{A_i}$, so it can infer (by means of the composition table) its distance to the traffic jam d_{jk}. In the first test, we consider what happens if $\mathsf{A_j}$ stops suddenly, while $\mathsf{A_i}$ cannot infer d_{jk}. In such a case the input values to the movement control of $\mathsf{A_i}$ are: the qualitative distance d_{ij} between $\mathsf{A_i}$ and $\mathsf{A_j}$ and the qualitative velocity difference dv between velocities of $\mathsf{A_i}$ and $\mathsf{A_j}$. In this case, $\mathsf{A_j}$ stops suddenly just before the traffic jam, while $\mathsf{A_i}$, having a longer braking distance, hits $\mathsf{A_j}$ and the collision occurs. The simulation of the test 1 is presented in the Figure 6, with a collision occurring in frame (c). In the second test, the distance value d_{jk} is sent from $\mathsf{A_j}$ to $\mathsf{A_i}$. Since, $\mathsf{A_i}$ has an access to d_{ij} and d_{jk}, it can infer d_{ik} (the composition Table 2). Afterwards, the movement control module is used twice: (a) to maintain a safe distance between $\mathsf{A_i}$ and $\mathsf{A_j}$, and (b) to maintain a safe distance between $\mathsf{A_i}$ and $\mathsf{A_k}$. Finally, the safer of two programs for movement control is chosen, i.e., the one that leads to smaller velocity of $\mathsf{A_i}$. In other words, $\mathsf{A_i}$ movement control performs two tasks, namely (a) tries to avoid the collision with $\mathsf{A_j}$ using d_{ij} value and (b) tries to avoid the collision with $\mathsf{A_k}$ using the inferred d_{ik} distance. The task (b) may be also considered as a prediction of future $\mathsf{A_j}$ movement which enables $\mathsf{A_i}$ to start braking earlier and avoiding the collision. The simulation of the second test is presented in Figure 7.

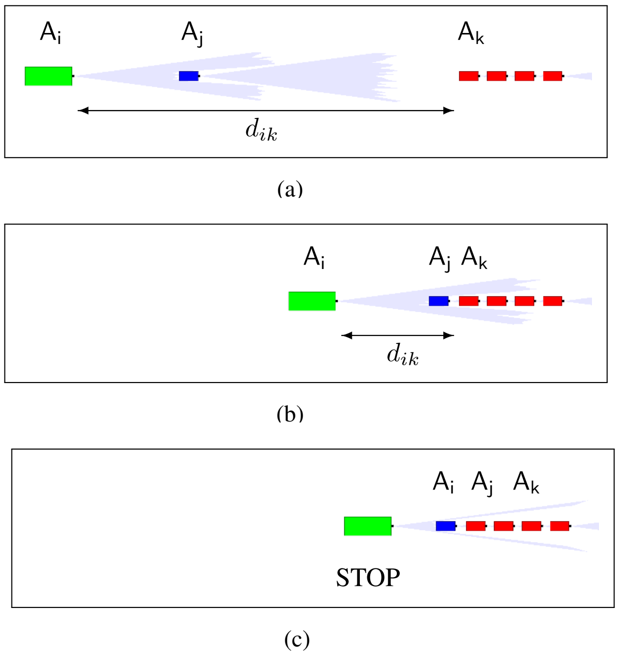


Figure 7: Experiment 2, test 2. $\mathsf{A_i}$ is able to infer d_{ik} and as a result, it avoids the collision.

The comparison of $\mathsf{A_i}$ speed in case of test 1 and test 2 is presented in Table 4. In the first case, i.e., without access to d_{jk}, $\mathsf{A_i}$ deceleration time (293 ms) is not long enough to stop the vehicle, therefore a crash occurs. On the other hand, in the test 2, A_i begins deceleration much earlier which results in longer deceleration time (472 ms) and collision avoidance.

Table 4: Comparison of deceleration with and without an access to d_{jk}.

Experiment	deceleration time	beginning of deceleration
Test 1: Collision Avoidance without access to d_{jk}	293 ms	945 ms
Test 2: Collision Avoidance with access to d_{jk}	472 ms	663 ms

A detailed comparison of the A_i velocity is presented in Figure 8b where the thin line corresponds to the test 1 and the thick line corresponds to the test 2. In the test 2, A_i starts breaking earlier than in the test 1 which results in collision avoidance. In fact the early breaking enables to obtain a “stable” state, i.e., a state in which there is no danger of collision and no need to change the velocity – it corresponds to the horizontal segment in the middle of a velocity graph for test 2.

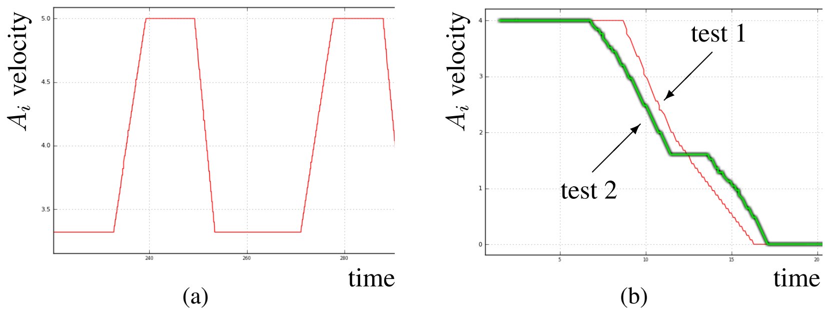


Figure 8: Velocity of object A_i in Experiment 1 – following an object (a), and Experiment 2 – collision avoidance (b).

5 CONCLUSIONS AND FUTURE WORK

We have presented a movement control system expressible in the logic designed in [13]. Some of the advantages of this logic have been exploited in order to represent the qualitative composition of movements, which enables the control of movements, in particular for collision avoidance. On the other hand, we have obtained stable qualitative values by considering hysteresis loops. Finally, we have implemented the system in ROS and tested it with computer simulator STAGE. Two experiments have been performed related to following an object and collision avoidance in traffic jam scenarios. The results confirm the possibility of using our system in practical applications, and our approach is flexible enough in order to accomplish other tasks related to movement of vehicles or robots. Since the movement control procedure is expressible in PDL, we consider a tight integration with logic as a future work, by considering systems for model checking and satisfiability checkers, which will allow us to reason in the system in a more general way. We also consider temporal extensions of PDL, in the line of [8] and further experiments of the obtained methods.

ACKNOWLEDGMENTS

The work presented in this paper is partially supported by the Polish National Science Centre grant DEC-2011/02/A/HS1/00395 (P.A.W.) and by the Spanish Project TIN15-70266-C2-P-1 (E.M.V.).

REFERENCES

[1] Patrick Blackburn, Johan van Benthem, and Frank Wolter. (2006). *Handbook of modal logic*, volume 3. Elsevier.

[2] Andrea Cherubini, Boris Grechanichenko, Fabien Spindler, and François Chaumette. (2013). Avoiding moving obstacles during visual navigation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3069–3074. IEEE.

[3] Anthony G. Cohn and Jochen Renz. (2008). Qualitative spatial representation and reasoning. *Handbook of knowledge representation*, 3:551–596.

[4] Matthias Delafontaine, Peter Bogaert, Anthony G Cohn, Frank Witlox, Philippe De Maeyer, and Nico Van de Weghe. (2011). Inferring additional knowledge from qtcn relations. *Information Sciences*, 181(9):1573–1590.

[5] Zoe Falomir, VICENT CASTELLÓ, M Teresa Escrig, and Juan Carlos Peris. (2011). Fuzzy distance sensor data integration and interpretation. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 19(3):499–528.

[6] Kenneth D. Forbus. (2008). Qualitative modeling. *Handbook of knowledge representation*, 3:361–393.

[7] Brian Gerkey, Richard T Vaughan, and Andrew Howard. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, volume 1, pages 317–323.

[8] Laura Giordano, Alberto Martelli, and Camilla Schwind. (2001). Reasoning about actions in dynamic linear time temporal logic. *Logic Journal of IGPL*, 9(2):273–288.

[9] Hassan K Khalil and JW Grizzle. (2002). *Nonlinear systems*, volume 3. Prentice Hall Upper Saddle River.

[10] Honghai Liu, David J. Brown, and George Macleod Coghill. (2008). Fuzzy qualitative robot kinematics. *Fuzzy Systems, IEEE Transactions on*, 16(3):808–822.

[11] W. Liu, S. Li, and J. Renz. (2009). Combining RCC-8 with qualitative direction calculi: Algorithms and complexity. *Proceedings of IJCAI-09*, pages 854–859.

[12] M Teresa Escrig Monferrer and Francisco Toledo Lobo. (2002). Qualitative velocity. In *Topics in Artificial Intelligence*, pages 29–39. Springer.

[13] Emilio Muñoz-Velasco, Alfredo Burrieza, and Manuel Ojeda-Aciego. (2014). A logic framework for reasoning with movement based on fuzzy qualitative representation. *Fuzzy Sets and Systems*, 242:114–131.

[14] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. (2009). ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3.2, page 5.

[15] Eric Rohmer, Surya PN Singh, and Marc Freese. (2013). V-rep: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE.

[16] Zvi Shiller, Frederic Large, and Sepanta Sekhavat. (2001). Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3716–3721. IEEE.

[17] Nico Van de Weghe, Bart Kuijpers, Peter Bogaert, and Philippe De Maeyer. (2005). A qualitative trajectory calculus and the composition of its relations. In *GeoSpatial Semantics*, pages 60–76. Springer.