Novel Motion Planning Method for Mobile Robots Using Velocity Obstacle

Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss

Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
Magyar tudósok körútja 2, 1117 Budapest, Hungary
e-mail: zgyenes@iit.bme.hu, szadeczky@iit.bme.hu

The main concept of this article is to introduce a cost function based algorithm for the reactive motion planning problem of mobile robots. With the usage of the cost function, different strategies can be combined. Next to speed and safety, several aspects can play a role in the motion. i.e. Traffic Regulation rules or lane-keeping. The different methods can be used separately as well, but the combination of methods provides an appropriate solution. The introduced reactive motion planning algorithm generates a collision-free solution for the agent in dynamic environment.

Keywords: motion planning; mobile robots; cost function; dynamic environment; obstacle avoidance

1 Introduction

Robot technology is one of the most important fields of modern technology. The robots are used in different fields of the world to perform tasks faster or more accurate than humans (e.g, [1]-[3]). The main task of mobile agents is to execute a collision-free motion to the target position from a start position. The agent has to fulfill several requirements during its motion. One of the most critical tasks is to reach the target quickly as possible. Another important goal of motion planning is to ensure the safety of the environment and the agent.

Using motion planning algorithms, the velocity and the path profile of the robot can be generated. These algorithms can be divided into subsets considering the knowledge of the environment. If every information is given as a priori information, then offline global motion planning methods (e.g.: *Rapidly-exploring random tree (RRT)* [4]-[5], *Hybrid A** [6], *Metaheuristic Global Path planning* [7]) can ensure a suitable solution. If there is only local information about the environment, online reactive motion planning algorithms should be applied. The main goal of local motion planning algorithms is to plan local obstacle-avoidance maneuvers.

At the *Artificial Potential Field (APF)* method, all of the obstacles appear as repulsive forces, and the target generates an attractive virtual force. The result of the summation of the presented forces should be calculated [8]-[12]. With this methodology, the actual direction and the magnitude of the velocity of the robot can be calculated. The method was defined for the static environment first; after that, it was extended for dynamic environment. For example, it was applied in robot soccer [12]. It can also be used for nonholonomic robots too. Sometimes this motion planning method finds only a local optimum. A novel optimization method was introduced for AMOEBA-I robot using the modified potential field method [13]. In that case, the robot has the opportunity to find a solution in divergent cases when the normal *APF* method fails (narrow spaces).

Introducing a time dimension, the *State time-space* method [14] can generate a solution in dynamic environment. The task is to find a time-optimal trajectory for the robot using a confined set of canonical trajectories. The algorithm was also used for car-like robots.

The main idea of the *Dynamic velocity space (DVS)* motion planning method is to map the obstacles (static and moving) and the robot from the workspace to the velocity space. Two components should be used to accomplish this mapping: times to escape from collision and times to collision. Several aspects can be taken into account during the motion, i.e.: shortest path, minimum time trajectories [15].

The *Dynamic window (DWA)* method was introduced as a velocity space-based local motion planning method [16]. The control commands are selected in the velocity space for the robot. The final trajectory can be defined as the series of straight line and circular arcs as it was introduced in [17]. The actual velocity vector can be calculated, considering the kinematic and dynamic constraints of the robot.

The set of *Inevitable collision states (ICS)* contains every state of the robot when the collision is inevitable between the agent and the obstacle in the future. The state of the robot is in *ICS* if no control would result in a collision-free motion for the robot [18]. In a collision-free guaranteed motion, the robotic system never finds itself in an ICS situation. The algorithm was also extended for the dense and dynamic environment [19].

The *Directive circle (DC)* method is an extension of the *Velocity Obstacles (VO)* method [20]-[21]. At this method, the velocity of the agent will be selected from *DC*. That is drawn using the maximum velocity of the robot for the radius of the *DC*. Ensuring the kinematic constraints of the robot, the best solution is selected from the DC that is in the optimal direction to the target position. The DC method prevents the robot from staying in a local minima situation.

The *Evolutionary Algorithm* can also be used to generate an optimal path for the agent [22]-[23]. A novel concept was introduced using Gravitational Search Algorithm [24]. In that case, the optimal path can be generated in a partially

known environment. First, an optimal path is constructed offline, and later on, the agent follows this optimal path until a new obstacle blocks the path. At that time, using the sensor data, the robot can determine the positions of the other obstacles, and it is able to plan a collision-free path. The implemented algorithm was successfully used in robotic system. The evolutionary algorithm can be used with Artificial Neural Networks for optimization problems [25].

At the *Bug* algorithm, the robot moves in the direction of the target if the path is free. If the agent reaches an obstacle, then it has a tangential motion on the boundary of the obstacle. After that, the agent continues the motion to the target position [26].

There are even more mobile robots that have been appeared in the technology using automatization. In the future, they may have to use similar traffic rules and lanes in the factories as the cars on the roads. The main novelty of our work is that the introduced reactive motion planning methods can consider these new ideas (such as lane-keeping or traffic rules) during the motion planning of the mobile robots next to the obstacle avoidance strategy considering dynamic environment. Our main motivation was to combine and develop our previously introduced motion planning algorithms [27]- [29] using an extension of the lane keeping algorithm.

The article is constructed in the following order: Section 1.1 presents the *Velocity Obstacles* method that is the basic of our introduced motion planning methods. Section 2 presents the *Traffic Regulation Velocity Obstacles (TRVO)* method that can handle the traffic rules during the motion planning. After that, different aspects and strategies of the motion planning algorithms are introduced in Section 3 (using cost function based methods). Later, the simulation results will be presented in Section 4. At the end of the paper, conclusions are given.

1.1 Velocity Obstacles Method

The *Velocity Obstacles* (*VO*) method is a reactive motion planning method that uses the velocities and the positions of the agent and the obstacles [30]-[34]. The *VO* method was first developed for omnidirectional robots. So in this paper, an omnidirectional robot is used too, considering that at a velocity selection step, it can change its velocity vector immediately. The limitation of this usage is that most of the autonomous vehicles are nonholonomic like differential-driven mobile robots or cars. There were also attempts to use the *VO* for differential-drive mobile robots [31]. To increase the maneuverability, an *Effective center* and *Effective radius* were defined [35]. The kinematic constraints were fulfilled for this new center-point of the agent.

As an assumption, the obstacles B_i ($i = 1 \ldots n$ where n denotes the number of obstacles) and the robot A are presented as disk-shaped objects and their radii are known. In the motion planning algorithms, the robot is usually presented as a

point by decreasing its radius to zero and increasing the radii of obstacles. $\mathbf{p}_A$ is the position of the robot, $\mathbf{p}_{Bi}$ is the position of the obstacle B_i. $\mathbf{v}_A(0)$ represents the initial velocity vector of the agent, and $\mathbf{v}_{Bi}$ is the velocity vector of B_i. Both the position and velocity vectors are two-dimensional vectors in the workspace.

The VO_i is a cone that consists of every velocity of the agent ($\mathbf{v}_A$) that would result in a collision with B_i in the future if $\mathbf{p}_A$, $\mathbf{p}_{Bi}$ and $\mathbf{v}_{Bi}$ are given. The cone of VO_i can be defined as:

$$VO_i = \{ \ \mathbf{v}_A \mid \exists t: \mathbf{p}_A + \mathbf{v}_A t \cap \ \mathbf{p}_{Bi} + \mathbf{v}_{Bi} t \neq 0\} \tag{1}$$

It is assumed in (1) that the velocities of the robot and also the obstacles will not change until t. Every velocity vector of the robot is always represented with the endpoint of the velocity vector and the starting point is always the position of the robot.

If there are more obstacles in the workspace, the whole *VO* set can be defined by the union of the different VO_i cones:

$$VO = \cup_{i=1}^{n} VO_i \tag{2}$$

Figure 1 shows an example of the workspace of the robot with a moving obstacle (with position $\mathbf{p}_{B1}$ and velocity $\mathbf{v}_{B1}$) and a static obstacle (with position $\mathbf{p}_{B2}$). The grey cones mean the *VO*.

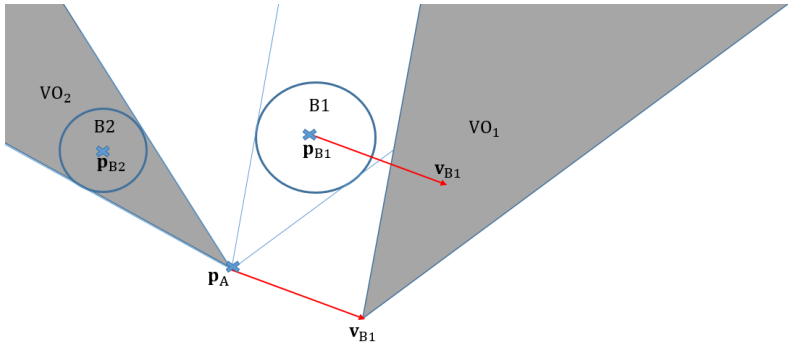

Figure 1
Example for Velocity Obstacles

Using the kinematic and dynamic constraints of the robot, the *Reachable Velocities* (*RV*) can be calculated that consists of every $\mathbf{v}_A$ velocity vector of the robot that is reachable from $\mathbf{v}_A(0)$ in the next sampling time. By subtracting the *VO* from the *RV*, the *Reachable Avoidance Velocities (RAV)* can be calculated. The velocity selection from the *RAV* area is presented in Figure 2, where the yellow area presents the *RAV*, R means the the robot, O is the obstacle, the grey area represents the *VO* set, the selected velocity vector is presented with a blue circle, and the G shows the goal. Usually, a heuristic method is used to choose a velocity vector from *RAV*. For example, the robot can select such a velocity vector that will cause the nearest motion to the target position.

Using the *Velocity Obstacle method*, it is also possible that the robot selects a velocity from the boundary of the *VO* and *RAV* to ensure the fastest target reaching. In that case, the robot will move tangentially to the obstacle during its motion. If information about the obstacles $(\mathbf{p}_{Bi}, \mathbf{v}_{Bi})$ are not accurate, this type of motion could cause a collision between the agent and the obstacle. To ensure a collision-free motion, different aspects can be taken into account.

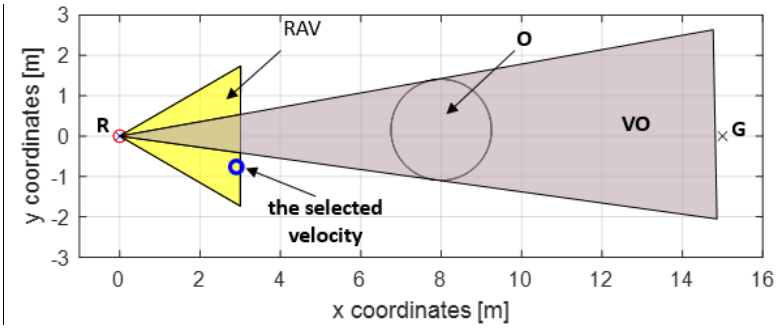

Figure 2

Selecting a velocity vector from RAV

2 Traffic Regulation Velocity Obstacles (TRVO) Method

The basic of the *Traffic Regulation Velocity Obstacles (TRVO)* method was first introduced by us in [27]. In this paper, a developed version of the *TRVO* method is introduced with lower calculating cost. The main idea of this method is to choose a velocity for the agent that will satisfy the basic rules of the Traffic Regulation. This method was inspired by [36]-[37], where *COLREGS (International Regulations for Preventing Collisions at Sea)* were used for *Unmanned Surface Vehicles (USV).*

The *TRVO* method considers four rules for motion planning:

- Crossing from the left, Crossing from the right
- Overtaking, Head-on

The main concept of this method is to select the velocity areas from the *RAV* that will ensure compliance with the rules. These areas can be denoted for every obstacle by S_r, S_f, and S_d. Using a velocity vector for the agent from these areas will result in different maneuvers for the robot during its motion. The r has a meaning of rear maneuver, d is the divergent, and f is the front maneuver in consideration of the obstacle and the robot. For a given $\mathbf{v}_A$, if it is not parallel to $\mathbf{v}_{Bi}$ the intersection point of the paths of A and B_i can be determined as:

$$\mathbf{p}_A + \mathbf{v}_A t_A = \mathbf{p}_{Bi} + \mathbf{v}_{Bi} t_{Bi} = \mathbf{p}_x \tag{3}$$

where $\mathbf{p}_x$ is a point in the workspace where the obstacle and the robot would intersect their path during their motion in the future or the past.

The different subsets of *RAV* can be defined using the value of t_A and t_{Bi} as:

$$0 < t_A < t_{Bi} \Rightarrow \mathbf{v}_A \in S_f \tag{4}$$

$$0 < t_{Bi} < t_A \Rightarrow \mathbf{v}_A \in S_r \tag{5}$$

$$\min(t_A, \ t_{Bi}) \ < 0 \Rightarrow \mathbf{v}_A \in S_d \tag{6}$$

$$t_A = t_{Bi} \ and \ \ min(t_A, t_{Bi}) > 0 \Rightarrow collision \tag{7}$$

Figure 3 shows a situation where the velocities can be divided into three subsets. The red-colored subset represents the velocity vectors of the agent that would result in a front maneuver to the obstacles. The green area consists the velocities resulting in a rear maneuver. The blue area means the divergent velocity vectors of the agent.

In every sampling time, these subsets must be constructed. With the knowledge of the measured position and velocities of the agent and the obstacles, the actual rule can be constructed. At *Crossing from the right* situation, the velocity can be chosen from the union of S_r and S_d. At *Crossing from the left* situation, the velocity can be chosen from the union of S_r, S_f. and S_d, so form the *RAV*. At *Head-on* situation, every velocity vector can be chosen that will result in a right maneuver to the corresponding obstacle. At the *Overtaking* situation, those velocities can be selected that will contribute a left maneuver to the corresponding obstacle. (The algorithm is applied in right-hand traffic situation).

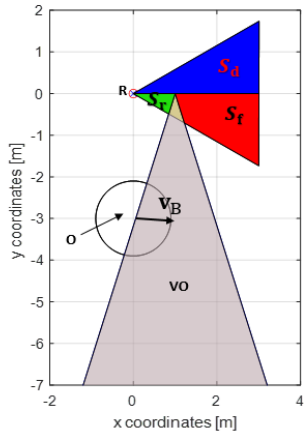

Figure 3
3 subsets of the RAV

3 Strategies for Velocity Selection of the Mobile Robot

In this section, different methods will be presented that can be used for selecting velocities of the robot considering the introduced *TRVO* method.

During motion planning, not every obstacle will play a role at every sampling time. Only those obstacles have an impact on the motion, whose distance to the robot is inside of a limit [27]. For every obstacle B_i, the time $t_{min_{A,\mathrm{Bi}}}$ can be defined as:

$$t_{min_{A,\mathrm{Bi}}} = \frac{-(\mathbf{p}_A - \mathbf{p}_{Bi})\cdot(\mathbf{v}_A - \mathbf{v}_{Bi})}{||\mathbf{v}_A - \mathbf{v}_{Bi}||} \tag{8}$$

where $t_{min_{A,\mathrm{Bi}}}$ is the time when the obstacle is at the nearest to the agent if $\mathbf{v}_A$ and $\mathbf{v}_{Bi}$ remain constant. The notation $||.||$ means the secondary norm, so the Euclidean distance of the presented vectors.

The minimal distance at the calculated $t_{min_{A,Bi}}$ is:

$$d_{min_i} = ||(\mathbf{p}_A + \mathbf{v}_A t_{min_{A,Bi}}) - (\mathbf{p}_{Bi} + \mathbf{v}_{Bi} t_{min_{A,Bi}})|| \tag{9}$$

At a given time moment, only those obstacles will be considered that satisfy the following relation:

$$0 < t_{min_{A,Bi}} < t_{max} \quad and \quad d_{min_i} < d_{max} \tag{10}$$

where t_{max} is a specified time limit, and d_{max} is a specified maximal limit of the distance between the agent and the obstacle. This equatuion can be calculated using precheck algorithm that is presented in Figure 4.

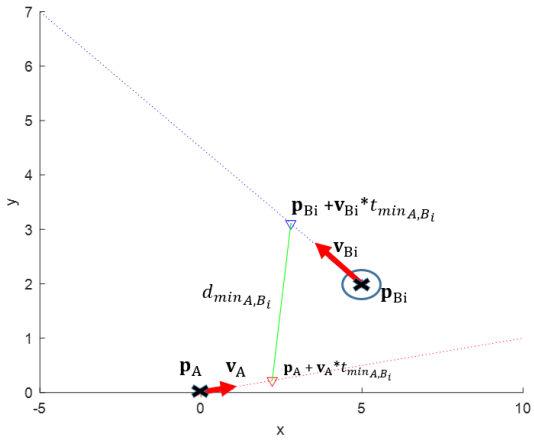

Figure 4
Precheck algorithm

3.1 Safety Velocity Obstacles (SVO) Method

Safety Velocity Obstacles (SVO) was first introduced by us in [28]. The main goal of the *SVO* method is to find the velocity vector for the agent that will result in the safest motion. For the solution, the furthest vector from *RAV* must be selected in consideration of the nearest VO cone. To ensure this solution, the minimal 2-norm must be calculated between the VO cone and the given velocity vector as it is presented in (11). $\mathbf{p}_{VO}$ is the point of the nearest VO cone and $D_S(\mathbf{v}_A)$ means the minimal distance:

$$D_S(\mathbf{v}_A) = \min\{\min_{\mathbf{p}_{VO} \in VO} ||\mathbf{v}_A - \mathbf{p}_{VO}||, D_{max}\} \tag{11}$$

where D_{max} represents the maximum distance that is considered.

$D_S(\mathbf{v}_A)$ can be transformed into [0,1] by dividing $D_S(\mathbf{v}_A)$ by D_{max}.

$$C_S(\mathbf{v}_A) = 1 - \frac{D_S(\mathbf{v}_A)}{D_{max}} \tag{12}$$

The safest motion will be resulted by selecting the velocity vector for the robot that has the minimum value of $C_S(\mathbf{v}_A)$.

Using only the above-presented algorithm, it can be resulted that selecting the safest velocity will not ensure the target reaching during the motion.

3.1.1 Extended Cost Function of SVO Method

Using an extended cost function at the *SVO* method, the safety of the robot and the environment, and also the target reaching will influence the motion planning algorithm. The value of the cost function can be calculated as:

$$Cost(\mathbf{v}_A) = \alpha\, C_S(\mathbf{v}_A) + \beta\, C_G(\mathbf{v}_A) \tag{13}$$

In (13), always the minimal value of $Cost(\mathbf{v}_A)$ must be found. $C_G(\mathbf{v}_A)$ represents the cost value of the fastest target reaching:

$$C_G(\mathbf{v}_A) = \frac{||\mathbf{p}_A + \mathbf{v}_A T_s - \mathbf{p}_{goal}||}{||\mathbf{p}_A(0) - \mathbf{p}_{goal}||} \tag{14}$$

where T_s means the sampling time, $\mathbf{p}_{goal}$ means the position of the goal, $\mathbf{p}_A(0)$ is the start position of the robot. So $C_G(\mathbf{v}_A)$ means the distance between the robot and the desired position using the investigated velocity vector, and it is normalized with the distance of the target and start position.

In the cost function, the motion planning is influenced by the $\alpha \geq 0$ and $\beta \geq 0$ parameters. If $\beta = 0$ and $\alpha \neq 0$, then the safest, if $\alpha = 0$ and $\beta \neq 0$, then the fastest solution will be generated. In other cases, safety and speed will play a different role during the algorithm.

3.2 Lane Keeping Velocity Obstacles (LKVO) Method

The *Velocity Obstacles* method and the *Safety Velocity Obstacles* method can be used even if there is no lane inside of the workspace of the agent. However, if there is a lane, then the *Lane Keeping Velocity Obstacles (LKVO)* method can ensure an appropriate solution. The basic of this motion planning method was introduced in our previous work for straight lanes [29]. In this work, as an extension, the structure of the lanes are constructed in a more general way using Bezier splines [38].

3.2.1 The Basic of Bezier Splines

For creating the lanes, splines can be used as an adequate solution opportunity. Different types of splines are usable, e.g.: Bezier [39], B-splines [40], Catmull-Rom [41]. The spline can be divided into segments, and each segment is an n degree polynomial. The Bezier spline was chosen for constructing the lane. The lane is fixed during the motion, and as an assumption, the control points of the spline are known at the beginning of the motion planning. The Bezier splines can be created as:

$$\mathbf{Bez}(\boldsymbol{t}) = \sum_{i=0}^{n} \; b_{i,n}(t)\, \mathbf{P}_i \tag{15}$$

where $t = 0\ldots1$, $\mathbf{P}_i$ is a two-dimensional control point of a segment, there are *n+1* control points in each segment and $\mathbf{Bez}(t)\colon [0..1] \subset \mathbb{R} \to \mathbb{R}^2$. The Bernstein polynomial can be calculated as:

$$b_{i,n}(t) = \binom{n}{i} \quad t^i \quad (1-t)^{n-i} \tag{16}$$

where $i = 0\ldots n$ and

$$\binom{n}{i} = \frac{n!}{i! \;\; (n-i)!} \tag{17}$$

3.2.2 Structure of the Lanes

The borders of the lanes are calculated using second-degree Bezier splines. First, one side of the borders shall be calculated. The other border can be defined by using an offset for the spline. For a second degree, Bezier spline three control points have to be used in every segment ($\mathbf{P}_0,\ \mathbf{P}_1, \mathbf{P}_2$). Every control point has two coordinates (x, y). The second-degree Bezier spline is represented in (18), by using (15) with $n = 2$ substitution:

$$\mathbf{Bez}(\boldsymbol{t}) = \mathbf{P}_0\,(1-t)^2 + 2\,\mathbf{P}_1 t\,(1-t) + \mathbf{P}_2\,t^2 \tag{18}$$

As an expectation, the border has to ensure the zero and first-order continuity. For the zero-order continuity, the first control point of the next segment must be the same as the last control point of the previous segment. If $\mathbf{Bez_1}$ means the Bezier curves to the first segment (with control points $\mathbf{P}_0, \mathbf{P}_1$, and $\mathbf{P}_2$) and $\mathbf{Bez_2}$ means

the Bezier curves to the second segment (with control points $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{P}_4$), then for the first order continuity, the next equations must be fulfilled:

$$\mathbf{Bez_1}'(1) = \mathbf{Bez_2}'(0) \tag{19}$$

Where $\mathbf{Bez_1'}(1)$ means the time-derivation of the $\mathbf{Bez_1(t)}$ and substituted the value of 1 into the derivated equation.

After the substitution, the result is:

$$\mathbf{P}_3 = -\mathbf{P}_1 + 2\,\mathbf{P}_2 \tag{20}$$

If all the points are known that the spline has to contain, then the Bezier spline can be already calculated using (18) and (20).

Figure 5 illustrates a Bezier spline with two segments. The second control point of the second segment ($\mathbf{P}_3$) is calculated using (20), ensuring the first-order continuity. If the Bezier spline has more than two segments, the control points of the segments can be calculated with the same algorithm.

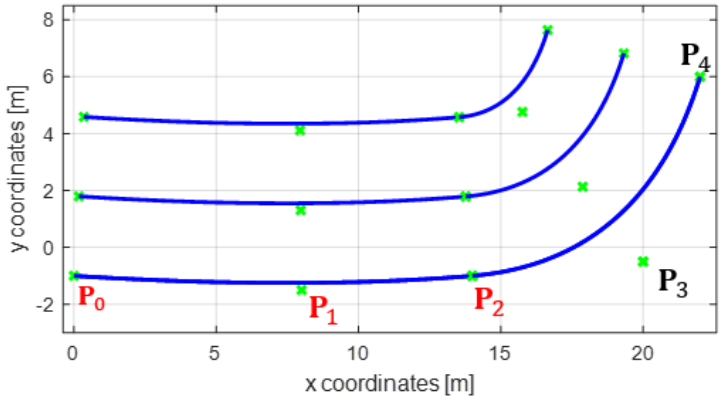

Figure 5
Lanes with the control points

After all of the control points of all segments have been defined, the whole Bezier spline can be established. To get the other side of the corridor, another Bezier spline is needed that always has the same distance from the previous spline. The resulted Bezier splines that generate the lanes are presented in Figure 5. The control points are marked with green shaped x-s, and the borders of the lanes are shown with blue color.

3.2.3 Steps of LKVO

The main concept of the $LKVO$ method is to select a velocity vector for the robot with that the agent will stay inside of the lane if it is possible or reach the boundary of the lane in the furthest time.

To detect, when the robot would reach the boundary of the lane, the intersection of the line of the velocity and second-order Bezier spline has to be calculated. Suppose the line has the (normal vector) equation.

$$ax + by = c \tag{21}$$

In vector form:

$$\mathbf{A}^{\mathrm{T}} \cdot \mathbf{X} = c \tag{22}$$

Now $\mathbf{A}^T = (a, b)$, $\mathbf{X}(t) = \mathbf{Bez}(t)$ from (18) because for the border of the lanes, second-order Bezier spline is used. After the substitution of (18) into (22), the result is:

$$(1-t)^2(\mathbf{A}^{\mathrm{T}} \cdot \mathbf{P}_0) + 2\,t(1-t)\,(\mathbf{A}^{\mathrm{T}} \cdot \mathbf{P}_1) + t^2(\mathbf{A}^{\mathrm{T}} \cdot \mathbf{P}_2) - c = 0 \tag{23}$$

Using (23) for every investigated velocity vector from the corresponding subset that satisfies the *TRVO* algorithm, every intersection point can be defined. It is possible that for a selected velocity of the robot, there are more intersection points on the Bezier spline. In that case, the algorithm has to choose the closest intersection point to the position of the robot with the right orientation. For every velocity $\mathbf{v}_A$, it has to be calculated when the robot would reach the right $(t_R(\mathbf{v}_A))$ and the left boundary $(t_L(\mathbf{v}_A))$ of the lane and use the minimum from them.

The cost value of the $LKVO$ method can be defined after a normalization:

$$C_{LK}(\mathbf{v}_A) = 1 - \frac{\min(t_R(\mathbf{v}_A),\ t_L(\mathbf{v}_A))}{t_H} \tag{24}$$

where t_H is a given time horizon, the cost value is even smaller if the robot will reach the boundary of the lane in a further time. If both of $t_R(\mathbf{v}_A)$ and $t_L(\mathbf{v}_A)$ are infinite numbers, then the robot will never reach the boundary, it will move in the lane. In that case, t_H must be used in the cost function instead of $\min(t_R(\mathbf{v}_A),\ t_L(\mathbf{v}_A))$. (24) should be used if the robot is inside of the lane. A new logical variable can be introduced ($inLane$) that has a value of 1 if the robot is inside of the lane. The value of the variable is 0 if the agent is outside of the lane. As an assumption, the robot leaves the lane on the left side. So the extended cost value for this method can be defined as:

$$C_{LK}(\mathbf{v}_A) = inLane\left(1 - \frac{\min(t_R(\mathbf{v}_A),\ t_L(\mathbf{v}_A))}{t_H}\right) + (1 - inLane)\ \frac{t_R(\mathbf{v}_A)}{t_H} \tag{25}$$

because if the robot is outside of the lane, it has to select a velocity vector that will result in the lane entering back to the right as fast as it is possible (if $t_R(\mathbf{v}_A)$ exists). If there is no opportunity to select a velocity vector resulting lane reaching, the same strategy must be used as in the case when the robot is inside of the lane.

As in Section 3.1, at this algorithm, it is also a weakness that using only the $LKVO$ method, the agent has a slow motion inside of the lane because the main goal is to keep the lane if it is possible. The target position reaching has in this algorithm no effect.

So an appropriate solution idea is using a similar cost function as it was used in Section 3.1.1:

$$Cost(\mathbf{v}_A) = \gamma\, C_{LK}(\mathbf{v}_A) + \beta\, C_G(\mathbf{v}_A) \tag{26}$$

where $\gamma \geq 0$. In that case, the motion planning can be influenced by the target reaching and the lane-keeping at the same time.

3.3 Combination of the Different Methods

A cost function can be constructed that contains every above-presented strategy:

$$Cost_{total}(\mathbf{v}_A) = \alpha\, C_S(\mathbf{v}_A) + \beta\, C_G(\mathbf{v}_A) + \gamma\, C_{LK}(\mathbf{v}_A) \tag{27}$$

where every part of the cost function is the same as it was introduced in Section 3.1.1 and Section 3.2.3. The α, β, and γ parameters will influence which strategy will play a higher role during the motion planning at a specific sampling time. These parameters are given at the beginning of the motion planning algorithm, and they have the same value during the whole motion. The exact values of the parameters can be specified considering the expected solution strategy using the empirical parameter tuning methodology with the experimental results.

A cost value can be calculated for the velocity vectors of the appropriate subsets of the *RAV* (see Section 2). The best option can be selected using an optimization method (e.g., genetic algorithm [22]-[24]).

4 Simulation Results

In this section, several simulation results are presented.

Because of the low calculation cost, a grid-based solution is introduced. In our scenario, a 5*5 velocity grid is used in *RAV*. The $Cost_{total}(\mathbf{v}_A)$ value is calculated in every grid point. The optimal solution can be generated selecting the velocity vector from the grid that has the minimal cost value hence ensures a collision-free motion for the robot using the traffic rules and the lane-keeping algorithm.

So the steps of the motion planning algorithm are:

- Calculate the *VO* sets for every obstacle (Section 1.1).
- Calculate the subsets that satisfy the Traffic Regulation rules (Section 2).
- Make a grid from the investigated velocity vectors.
- Calculate the cost value for every grid point in the *RAV* set (after calculating every part of the cost function described in Section 3).
- Select the velocity vector that has minimal cost value.

There are two obstacles in the workspace of the agent: the first obstacle is a moving obstacle that approaches to the agent in the opposite lane; the second

obstacle is a static obstacle located in front of the robot in the same lane where the agent is (at the start). The parameters of the robot model are: radius: 0.3 m, maximum velocity: 3 m/s (the absolute value of the velocity vector can be in the interval of 0-3 m/s). The parameters of the obstacles are: radii: 0.6, 0.7 m, the velocity of the first obstacle is changing during the motion considering the curvature of the lane, d_{max} = 5 m, D_{max} = 3m, t_H = 1 s. The parameters have been calculated using empirical results. These examples show the differences between the introduced strategies and the result of the strategy where every part plays a role in the cost function.

In every example, (27) is used with different parameter values considering the desired strategy. As an assumption, the *TRVO* method, which was introduced in Section 2, is used in every example.

4.1 The Fastest Solution

To get the fastest solution for the target reaching the parameters must be set as:

- $\alpha = 0, \beta = 1, \gamma = 0$

In this case, the agent will select a velocity vector that causes the fastest motion, as it is represented in Figure 6, where the grid is represented using the little red x-s, the black line shows the previous path of the robot and the other notations are the same as in Figure 2. It can be recognized that the robot has a tangential situation with the static obstacle during the motion. The robot executed the overtaking maneuver before the moving obstacle crosses the path of the robot, ensuring the target reaching as fast as it is possible. The soft-landing algorithm is used during the method. In that case, the closer the robot is to the goal the smaller velocity vector will be selected.

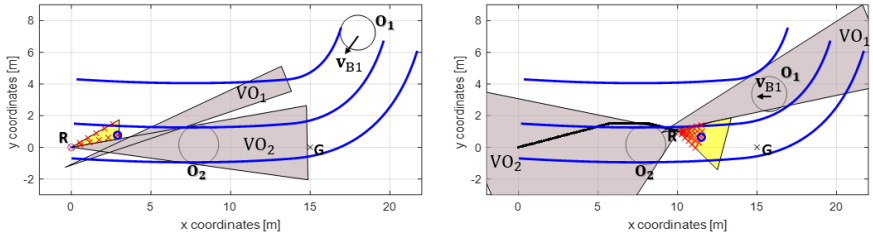

Figure 6
Selecting the velocity vector for the robot using the fastest solution

4.2 Lane-Keeping Algorithm

To execute the lane-keeping algorithm, the parameters must be set as:

- $\alpha = 0, \beta = 0, \gamma = 1$

In that case, the robot will select a velocity vector that will result to stay inside of the lane for the longest time. The result of this method is presented in Figure 7. As it is shown, the moving obstacle has already gone before the agent would have started the overtaking maneuver.

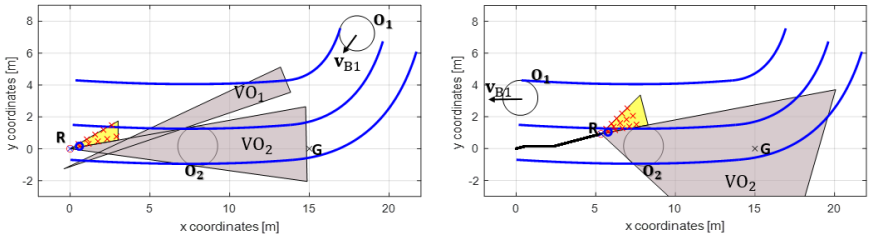

Figure 7

Selecting the velocity vector for the robot using LKVO

On the left side of Figure 8, the lane-keeping algorithm is presented in the situation when the robot is outside of the lane. In that case, if the robot has the opportunity, then a velocity vector is selected that results in the lane entering in the next time interval.

On the right side of Figure 8, the final path of the motion is presented. If the robot is outside of the lane, then the domination of the lane-entering maneuver is remarkable. **S** means the start position of the robot.

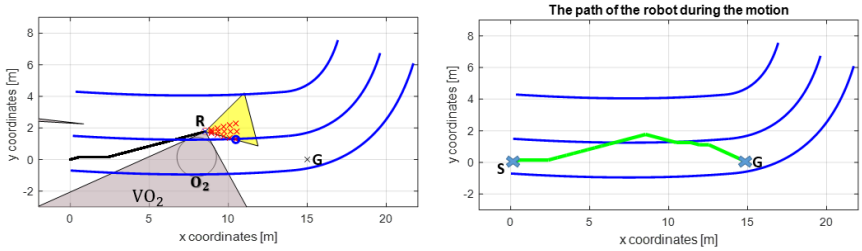

Figure 8

Selecting the velocity vector for the robot using LKVO outside of the lane and path

4.3 Combination of Every Method

To use all of the introduced methods together, the parameters must be set as:

- $\alpha = 1, \beta = 1, \gamma = 1$

At the beginning of the motion, the *LKVO* method plays a higher role. The velocity vector with the minimal cost value ensures the lane-keeping as it is presented in Figure 9 (t = 1s).

As the agent nears to the static obstacle, it has a small velocity until the moving obstacles execute its motion in the next lane. After that, the agent starts the overtaking maneuver. Velocity is selected, resulting in a safe motion for the robot because the static obstacle is close to the robot. So at this moment, the *SVO* method has a higher impact on the cost (t = 8 s). If the robot is out of the lane and it has the chance to come back, then it will execute this maneuver immediately.

After the robot passed the static obstacle, it reaches the target position as fast as it is possible (t = 10 s and t = 12 s). This example illustrated if the combined cost function is used with all components presented in Section 3, how the location of the obstacles and their velocities influence which component of the cost function will dominate at the velocity selection.

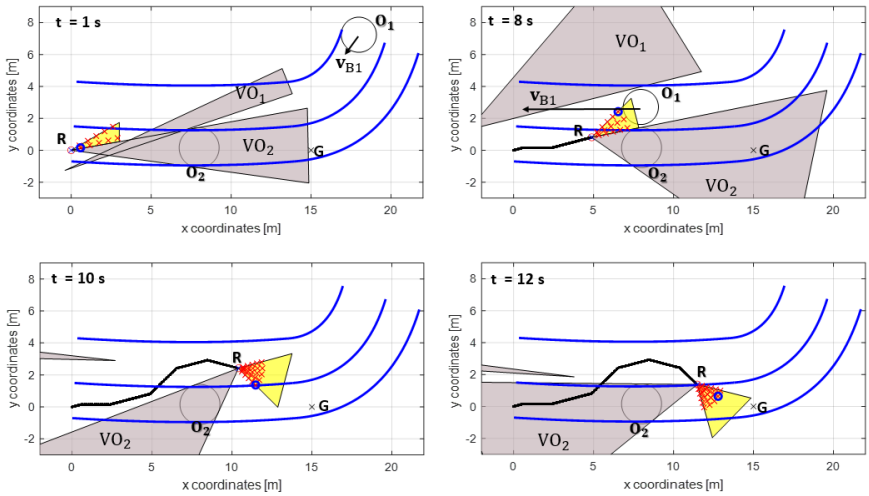

Figure 9
The result of the motion planning using the combined cost

4.4 Simulation in V-REP

V-REP [42] provides an appropriate solution opportunity to test mobile and non-mobile robots using motion planning algorithms. Several types of robots can be used for test cases.

The result of the defined algorithm was also tested in V-REP simulation environment. The motion planning algorithm was implemented in MATLAB, and there is the opportunity to connect the V-REP simulation environment with the MATLAB.

At the aspect of the mobile robot selection, an omnidirectional robot was used (blue colored). In that case, the direction of the movement can be changed in every sampling time if it is necessary. In the first example, there is a static

obstacle (presented with a grey colored cylinder) between the robot and the target point. The goal will be reached using the fastest solution without the *TRVO* method. The video of the solution in V-REP can be checked in [43].

In the second example, the *TRVO* algorithm is also considered. The workspace of the robot is presented in Figure 10. There are a static (O_1 presented with grey colored cylinder), and two moving obstacles (O_2, O_3 – presented with grey colored differential driven robots) in the environment of the robot. In consideration of the O_3, an overtaking maneuver is presented, ensuring an evasive maneuver. O_2 is crossing from the left, so the agent can execute its motion to the goal position without giving priority to the obstacle. The video of the motion is presented in [44].

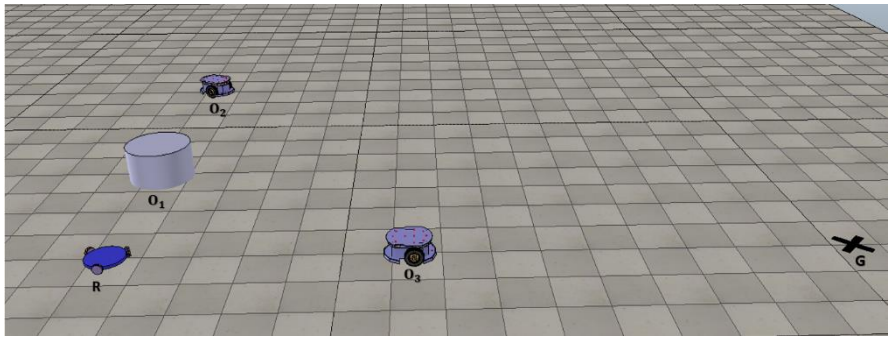

Figure 10
V-REP simulation

Conclusions

In this work, a reactive motion planning algorithm was presented with a cost function. The cost can be determined according to the requirements of safety, speed, and lane-keeping. An appropriate solution can be provided by using the combined cost function with all of these components. Some Traffic Regulation rules can also be considered during motion planning. The motion planning algorithm was also simulated in V-REP simulation environment and generated an appropriate solution using only the kinematic constraints of the robot. The main novelty of our approach was to use these different aspects of the motion planning problem at the same time, generating an optimal collision-free target reaching solution for the mobile robot.

As a future plan, the algorithm will be implemented on a real robotic system. The parameters of the cost function could also be changed in real-time during the motion considering the uncertainty of the sensor measurement data. The cost function could also be extended with new aspects of motion planning.

The algorithm could also be extended for autonomous vehicles, not only for holonomic mobile robots.

Acknowledgement

The research reported in this paper and carried out at the Budapest University of Technology and Economics has been supported by the National Research Development and Innovation Fund (TKP2020 Institution Excellence Subprogram, Grant No. BME-IE-MIFM) based on the charter of bolster issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology.

References

[1] T. Haidegger and Z. Benyo. Surgical robotic support for long duration space missions. Acta Astronautica, 63(7-10):996,1005, 2008

[2] Tamás Haidegger, Levente Kovács, Radu Emil Precup, Stefan Preitl, Balázs Benyó, Zoltán Benyó, Cascade control for telerobotic systems serving space medicine, Proceedings of the 18th IFAC World Congress, Milano (Italy) 2011, August 28-September 2

[3] Tamás Haidegger, Levente Kovács, Stefan Preitl, Radu Emil Precup, Balázs Benyó, Zoltán Benyó, Controller design solutions for long distance telesurgical applications, International Journal of Artificial Intelligence, 2011, 6, pp. 48-71

[4] Noreen I, Khan A, Ryu H, et al. Optimal path planning in cluttered environment using RRT*-AB. Intelligent Service Robotics. 2018;11(1):41–52

[5] Dong Y, Camci E, Kayacan E. Faster RRT-based Nonholonomic Path Planning in 2D Building Environments Using Skeleton-constrained Path Biasing. Intelligent Robot System. 2018; 387-401

[6] Petereit J, Emter T, Frey CW, et al. Application of Hybrid A * to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments. In: Proceedings -ROBOTIK 2012; 7th German Conference on Robotics; 2012 May 21-22; Munich, Germany; 2012, pp. 227-232

[7] Panov S, Koceski S. Metaheuristic global path planning algorithm for mobile robots. International Journal of Reasoning-based Intelligent Systems. 2015;7(1/2):35

[8] Malone N, Chiang HT, Lesser K, et al. Hybrid Dynamic Moving Obstacle Avoidance Using a Stochastic Reachable Set-Based Potential Field. IEEE Transactions on Robotics. 2017;33(5):1124-1138

[9] Li G, Tamura Y, Yamashita A, et al. Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning. International Journal of Mechatronics and Automation. 2013;3(3):141

[10] Kovács B, Szayer G, Tajti F, et al. A novel potential field method for path planning of mobile robots by adapting animal motion attributes. Robotics and Autonomous Systems. 2016;82:24-34

[11] Chiang HT, Malone N, Lesser K, et al. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In: Proceedings - IEEE International Conference on Robotics and Automation; 2015 May 26-30; Seattle, WA; 2015, pp. 2347-2354

[12] Qixin C, Yanwen H, Jingliang Z. An Evolutionary Artificial Potential Field Algorithm for Dynamic Path Planning of Mobile Robot. In: Proceedings - IEEE International Conference on Intelligent Robots and Systems; 2006 Oct 9-15; Beijin, China; 2006, pp. 3331-3336

[13] Tonglin Liu, Chengdong Wu, Bin Li, Shugen Ma, Jinguo Liu. A novel auto- adapted path- planning method for a shape- shifting robot, International Journal of Intelligent Computing and Cybernetics, 2011, Vol. 4. pp. 61-80

[14] Fraichard T. Trajectory planning in a dynamic workspace: a 'state-time space' approach. Advanced Robotics. 1999;13(1):75-94

[15] Petti S, Fraichard T. Safe motion planning in dynamic environments. In: 2005 IEEE/,RSJ International Conference on Intelligent Robots and Systems, IROS; 2015 Aug 2-6; Edmonton, Alta., Canada; 2005, pp. 3726-3731

[16] Fox D, Burgard W, Thrun S. The Dynamic Window Approach to Collision Avoidance 1 Introduction. IEEE Robotics & Automation Magazine. 1997;4(1):1-23

[17] Seder M, Petrović I. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In: Proceedings - IEEE International Conference on Robotics and Automation; 2007 April 10-14; Rome, Italy; 2007, pp. 1986-1991

[18] Fraichard T, Asama H. Inevitable collision states - A step towards safer robots. Advanced Robotics. 2004;18(10):1001-1024

[19] Martinez-Gomez L, Fraichard T. Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation. In: Proceedings - IEEE International Conference on Robotics and Automation; 2009 May 12-17; Kobe, Japan; 2009, pp. 100-105

[20] Masehian E, Katebi Y. Robot motion planning in dynamic environments with moving obstacles and target. International Journal of Mechanical Systems Science and Engineering. 2007;1(5):107-112

[21] Masehian E, Katebi Y. Sensor-based motion planning of wheeled mobile robots in unknown dynamic environments. Journal of Intelligent and Robotic Systems: Theory and Applications. 2014;74(3-4):893-914

[22] Farahani, S. M., Abshouri, A. A., Nasiri, B., Meybodi, M. R., 2012, Some hybrid models to improve firefly algorithm performance. International Journal of Artificial Intelligence 8, S12, 97-117

[23] Garcia, M. A., Montiel, O., Castillo, O., Sepúlveda, R., Melin, P., 2009, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. Applied Soft Computing 9, 3, 1102-1110

[24] Constantin Purcaru, Radu-Emil Precup, Daniel Iercan, Lucian-Ovidiu Fedorovici, Radu-Codrut David, Florin Dragan. Optimal Robot Path Planning Using Gravitational Search Algorithm, International Journal of Artificial Intelligence, Vol. 10, No. S13, pp. 1-20, 2013

[25] Alexandru-Ciprian Zavoianu, Gerd Bramerdorfer, Edwin Lughofer a Siegfried Silber, Wolfgang Amrhein, Erich Peter Klement, Hybridization of Multi-Objective Evolutionary Algorithms and Artificial Neural Networks for Optimizing the Performance of Electrical Drives, 2013. September Engineering Applications of Artificial Intelligence 26(8):1781-1794

[26] Buniyamin N, Ngah WW, Wan Ngah WaJ, et al. A simple local path planning algorithm for autonomous mobile robots. International journal of systems applications, Engineering & development. 2011;5(2):151-159

[27] Gyenes Z, Gincsaine Szadeczky-Kardoss E. Traffic Regulation Velocity Obstacles method. In: Proceedings- 2019 20th International Carpathian Control Conference (ICCC); 2019 May 26-29; Krakow-Wieliczka, Poland; 2019

[28] Gyenes Z, Gincsaine Szadeczky-Kardoss E. Motion planning for mobile robots using the safety velocity obstacles method. In: Proceedings - 2018 19th International Carpathian Control Conference (ICCC); 2018 May 28-31; Szilvásvárad, Hungary; 2018. pp. 389-394

[29] Gyenes Z, Gincsaine Szadeczky-Kardoss E. A novel concept of lane-keeping algorithms for mobile robots. In: Proceedings - 2019 17th IEEE International Symposium on Intelligent Systems and Informatics (SISY); 2019 September 12-14; Subotica, Serbia; 2019

[30] Fiorini P, Shiller Z. Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research. 1998;17(7):760-772

[31] Snape J, Van Den Berg J, Guy SJ, et al. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS; 2009 Oct 11-15; St. Louis,USA; 2009, pp. 5917-5922

[32] Wilkie D, Van Den Berg J, Manocha D. Generalized velocity obstacles. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS; 2009 Oct 11-15; St. Louis,USA; 2009, pp. 5573-5578

[33] Van Den Berg J, Snape J, Guy SJ, et al. Reciprocal collision avoidance with acceleration-velocity obstacles. In: Proceedings - IEEE International Conference on Robotics and Automation; 2011 May 9-13; Shanghai, China; 2011, pp. 3475-3482

[34] Kim M, Oh JH. Study on optimal velocity selection using velocity obstacle (OVVO) in dynamic and crowded environment. Autonomous Robots. 2016;40(8):1459-1470

[35] Jamie Snape JSJG Jur van den Berg, Manocha D. Smooth and Collision-Free Navigation for Multiple Robots Under Differential-Drive Constraints. In: Proceedings - IEEE/RSJ International Conference on Intelligent Robots and Systems; 2010 Oct 18-22; Taipei, Taiwan; 2010, pp. 4584-4589

[36] Zhuang JY, Su YM, Liao YL, et al. Motion planning of USV based on Marine rules. Procedia Engineering. 2011;15:269-276, Available from: http://dx.doi.org/10.1016/j.proeng.2011.08.053

[37] Kuwata Y, Wolf MT, Zarzhitsky D, et al. Safe maritime autonomous navigation with COLREGS, using velocity obstacles. IEEE Journal of Oceanic Engineering. 2014;39(1):110-119

[38] Pottmann H, Farin G. Developable rational Bézier and B-spline surfaces. Computer Aided Geometric Design. 1995;12(5):513-531

[39] Huang X, Gao F, Xu G, et al. Extended-Search, Bézier Curve-Based Lane Detection and Reconstruction System for an Intelligent Vehicle. International Journal of Advanced Robotic Systems. 2015;12(9)

[40] Loose H, Franke U. B-spline-based road model for 3D lane recognition. In: Proceedings- 13th International IEEE Conference on Intelligent Transportation Systems ITSC; 2010 September 19-22; Funchal, Portugal; 2010, pp. 91-98

[41] Wang Y, Shen D, Teoh EK. Lane detection using spline model. Pattern Recognition Letters. 2000;21(8):677-689

[42] https://www.coppeliarobotics.com/

[43] https://youtu.be/86qM-XcqZJo

[44] https://youtu.be/Mg0b9ZqB0go