A modified Hybrid Reciprocal Velocity Obstacles approach for multi-robot motion planning without communication

Maxime Sainte Catherine, Eric Lucet

A modified Hybrid Reciprocal Velocity Obstacles approach for multi-robot motion planning without communication

Maxime Sainte Catherine[a] and Eric Lucet[a,*]

***Abstract*— Ensuring a safe online motion planning despite a large number of moving agents is the problem addressed in this paper. Collision avoidance is achieved without communication between the agents and without global localization system. The proposed solution is a modification of the Hybrid Reciprocal Velocity Obstacles (HRVO) combined with a tracking error estimation, in order to adapt the Velocity Obstacle paradigm to agents with kinodynamic constraints and unreliable velocity estimates. This solution, evaluated in simulation and in real test scenario with three dynamic unicycle type robots, shows an improvement over HRVO.**

I. INTRODUCTION

Most of the autonomous guided vehicles (AGV) use guides such as tapes or magnets but it is not the only method as Amazon Robotics, Cimcorp and Seegrid introduced AGVs that were not restrained by guides. Those solutions, adopted by Amazon for example [1] aim at improving productivity by integrating unmanned distribution activities in the warehouse management system.

In order to allow these robots to navigate freely, a feasible motion must be produced. Several constraints need to be taken into account such as kinematics, dynamics and obstacle avoidance. Moreover, in a shared workspace, the robot motion planning must take into consideration other robots which are active agents.

Motion planning approaches can be classified in two categories. Reactive approaches, such as the Dynamic Window Approach (DWA) [2], consider the best motion that can be chosen at a certain time instant. On the other hand, trajectory planning method, such as Timed-Elastic-Bands (TEB) [3], plan the motion over a certain horizon of time. This category tends to produce smoother resulting trajectory and give results closer to optimality. For multi-robot motion planning, many trajectory generation approaches are centralized and use optimization [4], [5], [6]. However, the robustness relies on the central intelligence and on the communication between this intelligence and the robots. Furthermore, centralized approach suffer from scaling, as increasing the number of robots increases the computational load on the central computer.

Decentralized approaches such as the Receding Horizon Planning [7] or the Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) [8] allow to generate a trajectory for multiple agents but rely on communication as estimating the trajectory of another active agent is a complex task.

[a]CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191,France

*corresponding author: eric.lucet@cea.fr

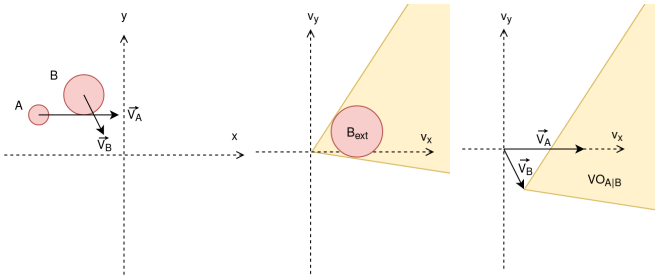
Reactive methods, while resulting in less smooth trajectories, offer the advantage of only requiring the current state of the other agents implying that they can be achieved using only sensing and no communication. The Velocity Obstacles (VO) approach [9] allows reactive local collision avoidance. Knowing the relative positions and velocities of moving obstacles, a set of forbidden velocities is created. Choosing a velocity oustide of those forbidden velocities leads to an avoidance of moving obstacles. This approach has been expanded to take into account uncertainty in the position and velocity of obstacles, Uncertainty-aware Velocity Obstacles (UVO), and has been combined with a global planner, Rapidly-exploring Random Belief Tree (RRBT) [10]. However, active agents also participate in the collision avoidance process. This aspect, overlooked by the VO approach, results in oscillations. Those oscillations are partly removed by Reciprocal Velocity Obstacles (RVO) [11] and further reduced by Hybrid Reciprocal Velocity Obstacles (HRVO) [12] that indroduce an implicit choice on which side an agent should pass. RVO was turned into a probabilistic approach [13] which is an effective means of dealing with uncertainty in position, velocity and actuation compared to bounding approaches [14].

Once the VOs, RVOs or HRVOs are built, a speed must be chosen and applied to the agent. To this end, a solution is Clearpath [15]. This algorithm finds an admissible speed that is the closest to the preferred one for the agent. To do so, it classifies the intersection points of the VOs as inside or outside of the VOs and extracts the segments that are outside of the VOs. It then returns the point from the outside segments that is the closest to the preferred velocity. Another solution to select an admissible speed is to sample the velocity space [16] and score the samples.

Optimal Reciprocal Collision Avoidance (ORCA) is both a VO formulation and a speed choice method. VOs are turned into half planes reducing the problem of finding the velocity the closest to the preferred one to a linear program.

The study performed by J. A. Douthwaite et al. [17] compares the different VO formulations and concludes ORCA performs the best in an ideal setting but is outperformed by HRVO in the presence of sensor noise.

In the following, the required VO principles are first briefly introduced. Based on them, the proposed method is then developed and evaluated in simulation and real test scenarios.



(a) Workspace configuration (b) Collision cone (c) Velocity obstacle

Fig. 1: Velocity obstacle faced by agent A

II. Background

A. Velocity Obstacles (VO)

The velocity obstacle approach [9] allows a local reactive collision avoidance of moving obstacles.

Considering two moving agents in a two dimensional space represented in Fig. 1(a), with Agent A defined by its velocity $\overrightarrow{v_A}$ and footprint A and B by $\overrightarrow{v_B}$ and B, the definition of a velocity obstacle is:

$$VO_{A|B} = \left\{ \overrightarrow{v} | \exists t > 0, t(\overrightarrow{v} - \overrightarrow{v_B}) \in B_{ext} \right\}$$

Where B_{ext} is the mapping of B in the configuration space of A: $B_{ext} = B \oplus -A$. The construction of the velocity obstacle can be achieved by constructing the collision cone which is the smallest cone with its apex at the origin containing B_{ext} as illustrated in Fig. 1(b). The collision cone is then translated by $\overrightarrow{v_B}$ to become the velocity obstacle for A induced by B, $VO_{A|B}$, as illustrated in Fig. 1(c).

In other words, the velocity obstacle is the set of velocities that would lead to collisions if chosen for A assuming:

- A instantly applies the chosen velocity and keeps it over an infinite time horizon
- B keeps a constant velocity over an infinite time horizon

On the contrary, if A chooses a velocity outside of $VO_{A|B}$, collisions will be avoided under the same assumptions. Those assumptions are strong and seldom fulfilled in practice. Instead, the velocity obstacle algorithm is run at a high frequency to overcome the assumption that both agents will keep the same speed over an infinite time horizon.

B. Reciprocal Velocity Obstacles (RVO) and Hybrid Reciprocal Velocity Obstacles (HRVO)

The VO representation was designed for an agent avoiding passive moving obstacles but can cause oscillations when an agent is avoiding another active agent. This is due to the fact that VO forces an agent to take care of the whole collision avoidance.

To counter this behavior, RVO were designed [11]. An agent is now allowed to choose the average between its current velocity and the collision free velocity chosen out of the VO leading to the following definition:

$$RVO_{A|B} = \left\{ \overrightarrow{v} | 2\overrightarrow{v} - \overrightarrow{v_A} \in VO_{A|B} \right\}$$

Therefore, the cone of collision defined for velocity obstacles is translated by $\frac{\overrightarrow{v_A}+\overrightarrow{v_B}}{2}$. This leads to each agent taking care of half of the avoidance and removes some oscillations that occurred when using the VO.

However, RVO can still cause oscillations that occurs when both agents choose to pass on the same side. Those oscillations called reciprocal dances can be observed with humans too. The HRVO [12] solve this issue by introducing an implicit agreement on the side on which the avoidance will be performed. It is constructed by expanding the RVO to a chosen boundary of the VO, forcing the crossing to be on a certain side. The side on which to expand the RVO is chosen as the opposite side of which $\overrightarrow{v_A}$ lies compared to the centerline of the RVO. For symmetry reasons, the other agent also chooses the same side.

C. Static obstacles

Since it has no velocity, the VO corresponding to a static obstacle has its apex at the origin, preventing all motion towards this obstacle. As a consequence, it needs to be truncated by a certain factor τ, the time horizon over which collisions will be avoided with that obstacle.

$$VO_{A|B,\tau} = \left\{ \overrightarrow{v} | \exists t > \tau, t(\overrightarrow{v} - \overrightarrow{v_B}) \in B_{ext} \right\}$$

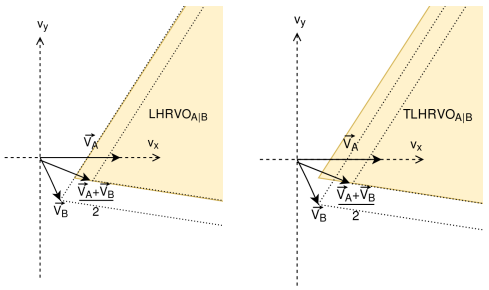
τ must be greater than the current deceleration time of the robot to ensure that the robot is able to stop before colliding with the obstacle.

III. Proposed method

A. Modification of inter-robot collision avoidance

HRVOs allow the robots to implicitly agree on which side to pass. However, the quality of the choice relies on an accurate speed estimation of the agents. To eliminate this restrictive dependence, the robots are given an incentive to pass on the right side by modifying the HRVO.

The first modification is the choice on which side the leg of the HRVO is extended. It is fixed so that the second leg of the RVO in trigonometric order is changed and the first one remains the same, as seen in Fig. 2(a). This modification, called Left Hybrid Reciprocal Velocity Obstacles (LHRVO), gives the incentive to pass on the right side most of the time.



(a) LHRVO (b) TLHRVO

Fig. 2: Velocity Obstacles constructions for inter-agent collision avoidance

However, when two robots have opposite speeds in a head to head encounter for example, or when the two robots are stopped, the RVO and the VO are overlapped. Therefore, the LHRVO is translated along the first leg as seen in Fig. 2(b). This gives a further incentive to pass on the right in all situations. This second construction is the Translated Left Hybrid Reciprocal Velocity Obstacles (TLHRVO).

B. Static obstacle estimation

To collect information about the environment the robot uses depth sensors such as Lidar or a RGB-D camera. The data resulting from the depth sensor is then mapped onto a two-dimensional occupancy grid representing the close surrounding of the robot. Most local planners can use the occupancy grid as is but it is not possible with velocity obstacles which require a geometrical representation of the obstacles footprints. To this end, obstacles are clustered using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [18] and are extracted as polygons by taking the convex-hull of a cluster with Graham's algorithm [19]. Finding the convex hull is easier than finding the concave polygon and does not impact the resulting VO.

C. Constraints and tracking error estimates of a unicycle robot

The following unicycle dynamic model is considered:

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \\ \dot{\omega} &= a_\omega \\ \dot{v} &= a_v \end{cases} \tag{1}$$

While the unicycle can perform forward motion due to the fact that $\dot{x}\cos\theta + \dot{y}\sin\theta = v$, it cannot move laterally as:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \tag{2}$$

This second equation, that constrains the kinematics and that cannot be integrated, is a nonholonomic constraint.

Moreover, the robot is assumed to have the following constraints on accelerations:

$$\begin{cases} a_{v\ min} \leq & a_v & \leq a_{v\ max} \\ -a_{\omega\ max} \leq & a_\omega & \leq a_{\omega\ max} \end{cases} \tag{3}$$

and the following speed limitations:

$$\begin{cases} 0 \leq & v & \leq v_{max} \\ -\omega_{max} \leq & \omega & \leq \omega_{max} \end{cases} \tag{4}$$

The VO approach was initially created for holonomic agents with dynamic constraints [9]. As a consequence, the VO paradigm cannot be used the way it was originally thought as it would lead to a risk of collision. Indeed, in this paradigm, the chosen velocity, which is out of the VOs, is only ensured to be collision free if applied instantaneously to the robot. However, the nonholonomic nature of the unicycle model (2), the speed limit imposed (4) and the dynamic constraint (3) imply that the chosen velocity cannot be reachable instantly.

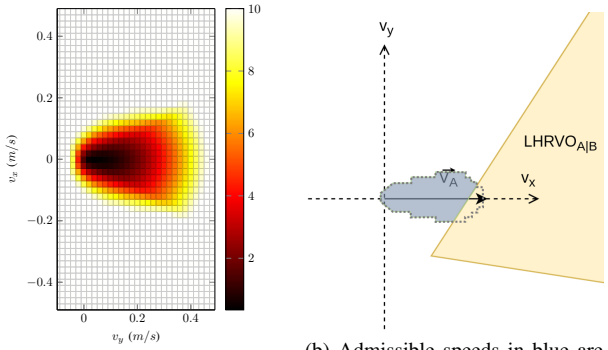
An approach is to consider that the nonholonomic robot will attempt to follow the speed that was chosen in the velocity obstacle paradigm but will have a tracking error [20]. Instead of computing the tracking error after choosing a speed, the allowed tracking error is bounded thus limiting the admissible velocity space. Knowing that the error will be inferior to this bound, all the Minkowski sums used to build the VOs are expanded by this maximum tracking error allowed by performing another Minkowski sum with a disk of radius the maximum tracking error. This approach, which was applied to ORCA [21], can also be applied to the considered velocity obstacles formulation.

To store the information about the tracking error, a set of maps is generated. The coordinates of a map are v_x and v_y, the coordinates that would be used as input for the controller. The value of the cell given by a certain pair (v_x, v_y) is $tracking\_error(v_x, v_y, v_0, w_0)$. It is important to note that the tracking error does not depend on the position and orientation of the robot. As a consequence, the maps are invariant by translation and rotation. However, the error depends on the initial linear and angular velocities. This implies that a map must be generated for each initial condition. In practice the initial conditions are sampled uniformly. Fig. 3(a) shows an example of a map for a robot with $v_0 = 0$ and $w_0 = 0$.

This tracking error is obtained by numerically solving the differential equations using Runge-Kutta method allowing the use of any controller. As the generation of the tracking error maps is quite long, it is done offline. Online, knowing the maximum tracking error allowed, a polygon of allowed velocities is extracted from the map corresponding to the current speed of the robot, its outline being displayed in blue dotted lines in Fig. 3(b).

D. Speed choice in the velocity obstacle paradigm

Before making any speed choice in the velocity space, the preferred velocity, $\overrightarrow{v}_{pref}$, must be determined. This preferred velocity is computed using the global plan as an indication. A point from the global plan is chosen using a look ahead distance. The direction of this point in the robot's frame is kept for the chosen speed which will have a norm



(a) Tracking error map for $v_0 = w_0 = 0$, saturated at $10cm$.

(b) Admissible speeds in blue are inside the tracking error polygon and outside of the VOs

Fig. 3: Tracking error map and admissible velocities.

proportional to the distance to the look ahead point, saturated in order not to exceed v_{max}.

Once the velocity space has been restricted due to static obstacles, other robots and the kinematic and dynamic constraints of the robot, a speed must be extracted from the admissible speeds. The Clearpath algorithm is not demanding computationally. However, it tends to switch abruptly from one intersection point to another as only the euclidean distance to the preferred speed is considered when comparing the different points.

To counter this behavior, a scoring function is applied and takes into account the following elements:

- distance to the preferred speed
- angle to the preferred speed
- distance to the previous speed
- angle to the previous speed
- distance to the speed chosen by Clearpath without considering dynamic and kinematic constraints
- angle to the speed chosen by Clearpath without considering dynamic and kinematic constraints

Algorithm 1 CTE

Input preferred velocity v_{pref}, tracking error polygon $\mathcal{P}_{te}$ and a list of VOs $\mathcal{L}_{vo}$
Output velocity to apply v_{chosen}

if admissible(v_{pref}) **then** ▷ admissible(v_{pref}) return true if v_{pref} is inside $\mathcal{P}_{te}$ and outside of all the VOs of $\mathcal{L}_{vo}$
  $admissible\_velocities$.append(v_{pref})
end if
for all velocity obstacle vo_1 in $\mathcal{L}_{vo}$ **do**
  for all velocity obstacle vo_2 in $\mathcal{L}_{vo}$ deprived of vo_1 **do**
    $potential\_velocities$.append(intersections(vo_1,vo_2))
  end for
  $potential\_velocities$.append(intersections(vo_1,$\mathcal{P}_{te}$))
  $potential\_velocities$.append(projectionss(v_{pref},vo_1))
end for ▷ projections_points(v_{pref},vo_1) returns the velocities resulting from the projection of v_{pref} on the legs of vo_1
$potential\_velocities$.append(points($\mathcal{P}_{te}$))
for all potential velocity v_{pot} in $potential\_velocities$ **do**
  if admissible(v_{pot}) **then**
    $admissible\_velocities$.append(v_{pot})
  end if
end for
if $admissible\_velocities$ is empty **then**
  $v_{chosen} \leftarrow (0, 0)$
else
  v_{chosen} ←best_score($admissible\_velocities$)
end if ▷ best_score($admissible\_velocities$) returns the velocity with the highest score of $admissible\_velocities$
return v_{chosen}

The proposed algorithm to perform the speed choice, that we called Clearpath with Tracking Error (CTE), builds a list of potential velocities from the intersections of the VOs, the projections of the preferred speed on the VOs' legs, the intersections of the VOs with the tracking error polygon and the the points from the tracking error polygon. From the list of potential velocities, all the ones that are within the tracking error polygon and outside of the VOs are added to the list of admissible velocities and are scored. The speed with the best score is returned.

Once the speed has been chosen, it needs to be transformed before being applied to the robot. A lower level proportional controller is in charge of adapting the wheels velocities to reach the wanted linear and angular velocities (v, w). To determine (v, w), an approach similar to the one proposed by J. Alonso-Mora et al. [20] is used.

$$v = \begin{cases} ||v_{chosen}||_2, & \text{if } |\frac{\theta_H}{T}| < \omega_{max} \\ 0, & \text{otherwise} \end{cases}$$

$$w = \begin{cases} \frac{\theta_H}{T}, & \text{if } |\frac{\theta_H}{T}| < \omega_{max} \\ \text{sign}(\theta_H)\omega_{max}, & \text{otherwise} \end{cases}$$

θ_H is the angle between the heading of the robot and the chosen speed. T is a parameter representing the duration to reach the wanted orientation over which the robot rotates on the spot.

The linear and angular accelerations that this command would imply are then computed and (v, ω) are adapted so that the acceleration fits in the saturation bounds defined by Eq. (3).

IV. Experiments

The solution is implemented in C++ in the ROS navigation stack[1]. Each robot performs its own localization using the Kullback–Leibler Divergence-sampling Monte Carlo Localization from the ROS package amcl[2]. The package obstacle_detector[3] is used by the robots to detect and track moving obstacles using data from the depth sensors.

To evaluate the proposed solution, experiments were performed in both simulation and in real world settings. For each setting, two experiments were considered. The first one only focuses on the avoidance of static obstacles while the second one focuses on multi-robot collision avoidance with three robots (see attached video).

A. Simulations

The Gazebo engine was used to perform the simulations. The simulated robot is a Turtlebot 2. Its parameters are summarized in Table I.

TABLE I: Settings used for the simulated Turtlebot 2

Setting	Value
v_{max}	0.5 m/s
ω_{max}	1.25 rad/s
$a_{v\ min}$	-1 m·s^{-2}
$a_{v\ max}$	1 m·s^{-2}
$a_{\omega\ max}$	4 rad·s^{-2}
local planner frequency	10Hz
global planner frequency	1Hz

[1]Eitan Marder-Eppstein. 2D navigation stack ROS. http://wiki.ros.org/navigation. Accessed: 2019-12-04.
[2]Brian P. Gerkey. amcl ROS. http://wiki.ros.org/amcl. Accessed: 2019-12-05.
[3]Mateusz Przybyla. obstacle_detector. https://github.com/tysik/obstacle_detector. Accessed: 2020-01-07.

1) Static obstacles avoidance: Although the primary focus of VO is not the avoidance of static obstacles, it is a prerequisite for which it is relevant to assess performance of the proposed solution against the state of the art. Thus, the first experiment was performed with two static unmapped obstacles that are the two blue cylinders. The proposed solution is compared to the TEB approach from the package teb_local_planner[4], which was configured to follow the constraints from Table I.

A trajectory generation approach cannot perform multi-robot collision avoidance without communication since it needs the planned trajectory of other robots. On the other hand, a reactive approach only requires the current state of the world. The goal of this experiment is to evaluate the loss of performance of the proposed reactive approach compared to a trajectory generation approach in regards to static obstacle avoidance.

TABLE II: Performance with static obstacles over ten successive simulation runs. al and aa are respectively the average absolute linear and angular accelerations.

	VO, $\tau = 1s$		**TEB**	
	mean	std	mean	std
al $(m \cdot s^{-2})$	0.08	0.04	0.11	0.09
aa $(rad \cdot s^{-2})$	0.41	0.13	0.90	0.36
travel time (s)	36.9	0.46	35.4	2.4

The results, gathered in Table II, show that the TEB approach is around 4% faster, which remains quite similar. It also leads to a smoother resulting trajectory. This is explained by the fact that the VO approach is reactive and only finds the optimal command until the next iteration. Taking into account the distance and angle to the previous chosen speed in the scoring of the admissible speeds allows the VO approach to have lower accelerations. On the other hand, the TEB package is set up to optimize travel time and we can assume it could use much less acceleration by changing the objective function used in the optimization.

2) Multi-robot collision avoidance: The second experiment was performed with three robots navigating in open space so that they can detect and track each other. The robots were given navigation goals so that they would all meet approximately at the same time with conflicting paths, as displayed in Fig. 4.

The moving obstacle detection faces some issue with the ego motion of the robot leading to false positives or inaccurate speed estimates. Table III gathers the results from fifty successive runs for each VO formulation. Fixing the choice on which side to pass using either LHRVO or TLHRVO greatly reduces the amount of collisions in the case of unreliable speed estimates. This is due to the fact that the HRVO formulation performs a choice on which side to pass based on the speed estimates and on the RVO centerline. TLHRVO causes less collisions than LHRVO because it is more conservative since the velocity space is reduced and as it makes the incentive to pass on the right side stronger.

[4]Christoph Rösmann. teb_local_planner http://wiki.ros.org/teb_local_planner. Accessed: 2019-12-04.

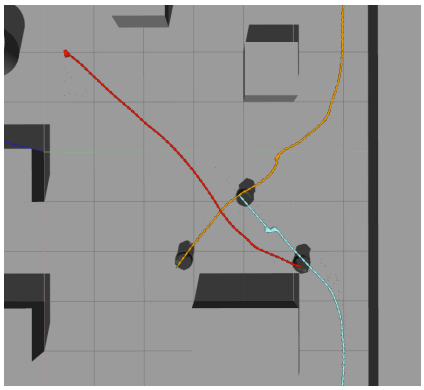


Fig. 4: Resulting rajectories example for a simulation with three robots (using LHRVO)

TABLE III: Performance with three Turtlebots 2 in simulation over 50 successive run. All values but the collisions are averaged over the three robots. al and aa are respectively the average absolute linear and angular accelerations.

	HRVO		**LHRVO**		**TLHRVO**	
	mean	std	mean	std	mean	std
al $(m \cdot s^{-2})$	0.11	0.06	0.10	0.05	0.11	0.06
aa $(rad \cdot s^{-2})$	0.54	0.29	0.50	0.22	0.53	0.31
travel time (s)	23.1	3.91	21.0	1.55	21.2	1.29
collisions	14		6		3	

B. Real world experiments

Similar experiments were performed on Turtlebots 3 waffle pi [22]. All the computations were done on the Raspberry pi 3 B+. The parameters used for the robots are summarized in Table IV.

TABLE IV: Settings used for the Turtlebot 3

Setting	**Value**
v_{max}	0.26 m/s
ω_{max}	1.82 rad/s
$a_{v\ min}$	-2.5 m·s^{-2}
$a_{v\ max}$	2.5 m·s^{-2}
$a_{\omega\ max}$	3.2 rad·s^{-2}
local planner frequency	10Hz
global planner frequency	1Hz

1) Static obstacles avoidance: An experiment was performed with only one Turtlebot 3 and static obstacles conflicting with its path. The results from this experiment are summarized in Table V. Both methods were highly perturbed by sensor noise leading to the robot stopping. This behavior was particularly frequent for the VO approach. However, the TEB approach suffered from it as well as, during one run, the robot remained stuck until the it entered in a recovery behavior. This run was not taken into account for Table V. In

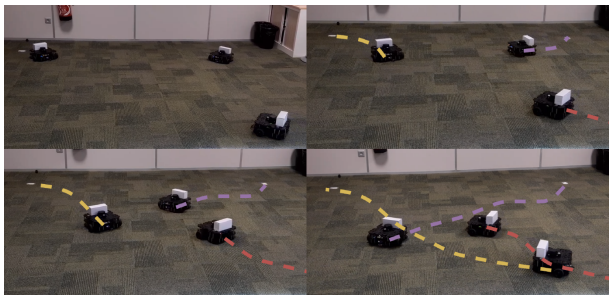


Fig. 5: Experiment with three Turtlebots 3. From left to right, top to bottom.

real world setting, the TEB approach is approximately 10% faster.

TABLE V: Performance with one Turtlebot 3 and static obstacles over ten successive runs. al and aa are respectively the average absolute linear and angular accelerations.

	VO, $\tau = 1s$		**TEB**	
	mean	std	mean	std
al $(m \cdot s^{-2})$	0.26	0.23	0.13	0.12
aa $(rad \cdot s^{-2})$	0.86	0.81	0.92	0.52
travel time (s)	20.2	3.46	18.0	1.55

2) Multi-robot collision avoidance: An experiment was performed with the three Turtlebots positioned on the corners of a square. They were given the opposite corner as a navigation goal ten times for each VO formulation, see Fig. 5. The results are gathered in Table VI. Acceleration, both linear and angular, is much higher than in simulation due to sensor noise and more frequent false positives in moving obstacles detection.

TABLE VI: Performance with three Turtlebots 3 over ten successive simulation runs. All values but the collisions are averaged over the three robots. al and aa are respectively the average absolute linear and angular accelerations.

	HRVO		**LHRVO**		**TLHRVO**	
	mean	std	mean	std	mean	std
al $(m \cdot s^{-2})$	0.26	0.22	0.23	0.20	0.25	0.20
aa $(rad \cdot s^{-2})$	0.64	0.44	0.84	0.35	0.80	0.42
travel time (s)	27.4	1.7	27.5	2.2	29.4	3.0
collisions	2		0		0	

HRVO leads to collisions due to the fact that the choice on which side to pass is not reliable with the detection and tracking of moving obstacles used. The two modifications, LHRVO and TLHRVO, make this choice safe thus resulting in no collisions. TLHRVO yields slower travel times since it is more conservative in regards to restricting the velocity space.

V. Conclusions

A modified online motion planning solution based on VO was presented for a multi-robot system. A static obstacle estimation was presented to convert the data from a depth sensor to a polygonal form usable by the VO approach. The consideration of the tracking error allows this method to rigorously take into account the kinematic and dynamic constraints of the nonholonomic robot and to keep the collision free nature of the VO paradigm. Furthermore, the polygon representing the velocities that are admissible in regards to the maximum allowed tracking error is used for a fast sampling of velocity space. Lastly, the adaptation of HRVO to give the incentive to pass on a certain side allows a much more reliable collision avoidance in a setting where the velocity estimates of the other agent are unreliable. While this proposed algorithm sees a slight decrease in performance when faced against a trajectory generation method in a setting with just static obstacles, this complete approach allows decentralized real-time dynamic collision avoidance without inter-agent communication. Furthermore, the computations are light enough to run on a single board computer at a frequency sufficiently high to avoid dynamic obstacles.

Related works show that the VO approach is prone to deadlocks when the number of agents increases [16] or when the environment becomes too cluttered. To solve this specific problem not addressed in this paper, a higher level decision unit could be set up for conflict resolution.

Finally, the integration of this algorithm in parallel with a trajectory generation and tracking method would give it its full meaning. Indeed, most of the time the multi-agent aspect is not problematic since only some portions of the robots' trajectories end up being conflicting. Using a trajectory generation method to perform navigation and handing over to the VO approach when the intended velocities are no longer safe would allow to benefit from the advantages of the trajectory planning while maintaining the safety the proposed solution approach provides in a setting without communication and multiple agents.

References

[1] Bruce Rolfsen. Amazon's growing robot army keeps warehouses humming. https://news.bloombergenvironment.com/safety/amazons-growing-robot-army-keeps-warehouses-humming. Accessed: 2019-11-04.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, March 1997.

[3] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6, May 2012.

[4] M. R. Panda, P. K. Das, S. K. Pradhan, and H. S. Behera. An improved gravitational search algorithm and its performance analysis for multi-robot path planning. In *2015 International Conference on Man and Machine Interfacing (MAMI)*, pages 1–8, December 2015.

[5] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1917–1922, October 2012.

[6] D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE International Conference on Robotics and Automation*, pages 477–483, May 2012.

[7] J. M. Filho and E. Lucet. Multi-robot motion planning: A modified receding horizon approach for reaching goal states. In *Acta Polytechnica*, volume 56, September 2015.

[8] V. R. Desaraju and J. P. How. Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In *2011 IEEE International Conference on Robotics and Automation*, pages 4956–4961, May 2011.

[9] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. In *The International Journal of Robotics Research*, volume 17, pages 760–772, July 1998.

[10] H. Yang, J. Lim, and S. Yoon. Anytime rrbt for handling uncertainty and dynamic objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4786–4793, October 2016.

[11] J. van den Berg, M. C. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935, May 2008.

[12] J. Snape, J. van den Berg, S. Guy, and D. Manocha. The hybrid reciprocal velocity obstacle. In *IEEE Transactions on Robotics*, volume 27, pages 696–706, August 2011.

[13] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha. Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1089–1096, September 2017.

[14] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen. Collision avoidance under bounded localization uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1192–1198, October 2012.

[15] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. C. Lin, D. Manocha, and P. Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 177–187, August 2009.

[16] D. Claes and K. Tuyls. Multi robot collision avoidance in a shared workspace. In *Autonomous Robots*, volume 42, pages 1749–1770, December 2018.

[17] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova. A comparative study of velocity obstacle approaches for multi-agent systems. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pages 289–294, September 2018.

[18] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[19] R.L. Graham. An efficient algorith for determining the convex hull of a finite planar set. In *Information Processing Letters*, volume 1, pages 132–133, 1972.

[20] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed Autonomous Robotic Systems: The 10th International Symposium*, volume 83, pages 203–216, January 2013.

[21] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart. Reciprocal collision avoidance for multiple car-like robots. In *2012 IEEE International Conference on Robotics and Automation*, pages 360–366, May 2012.

[22] Turtlebot 3. http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications. Accessed: 2020-02-24.