IROS 2015 Workshop on On-line decision-making in multi-robot coordination (DEMUR'15)
http://robotics.fel.cvut.cz/demur15/

TOWARDS HUMAN-SAFE NAVIGATION WITH PRO-ACTIVE COLLISION AVOIDANCE IN A SHARED WORKSPACE

Daniel Claes[a,*], Daniel Hennes[b], K. Tuyls[a]

[a] *smARTLab, Faculty of Computer Sciences, University of Liverpool, UK*
[b] *European Space Agency, Advanced Concepts Team, The Netherlands*
[*] corresponding author: dclaes@liv.ac.uk

Abstract. This paper presents a human-safe navigation algorithm for shared human-robot workspaces, based on the velocity obstacles paradigm. By extending the velocity obstacle paradigm with different cost factors accounting for humans and robots, the approach allows human workers to use the same navigation space as robots. It does not rely on any external sensors and shows its feasibility even in densely packed environments. Experiments show that our approach leads to safer and smoother paths at a small cost of additional traveled time and distance.

Keywords: Multi-robot collision avoidance, Velocity Obstacles.

1. Introduction

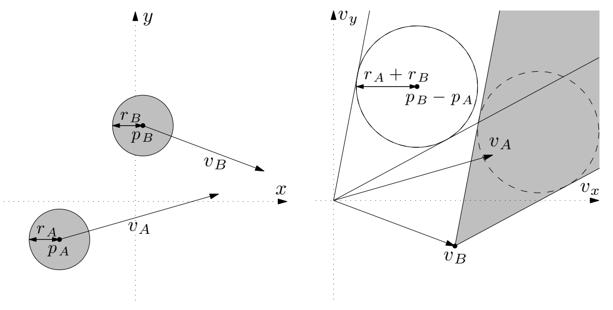
Current research in mobile robotics focuses more and more on enabling robots and humans to share a common workspace. Two well known research initiatives in this direction are the *Factories of the Future* and *Industry 4.0*, which both have the goal to develop smart factories with networked tools, devices, and mobile manipulation platforms (e.g. the KUKA youBot).

Nowadays robots in manufacturing are typically not designed to be mobile and human-safe. They are placed inside cages and operation is interrupted as soon as a human enters the safety zones. Current solutions for mobile robots in manufacturing settings are restricted to predefined paths, e.g., tracks on the floor, or restricted to movement in a grid to ensure easy navigation. Humans are not allowed to enter the navigation zone of the robots in order to ensure safety.

Relying on predefined paths and grids for navigation is too restrictive and does not allow for a flexible and generally applicable setup of a mobile multi robot system. Ideally, robots should be able to plan their paths through any open space and ensure safety without any external limitations such as restricted zones. Additionally, in an unstructured work-space there are no traffic rules that restrict the navigation. To safely navigate in such a shared multi-robot and human setting, the robot system has to take into account that the surrounding moving 'obstacles' are essentially also pro-active agents and thus aim to avoid collisions too.

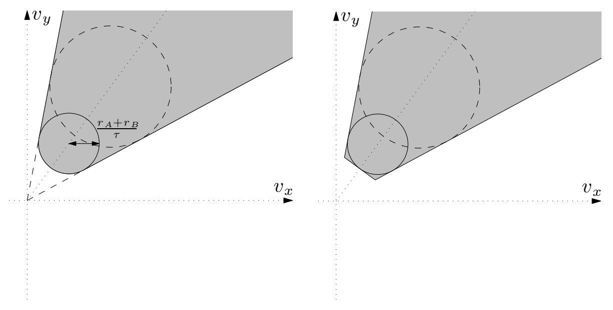
Recently, some algorithms based on the velocity obstacle paradigm were introduced, e.g. ORCA [1], and have successfully been applied to multi-robot collision avoidance [2, 3]. While these algorithms provide guaranteed safety and even optimality for the individual agents, there are still some limitations that remain. Specifically, the algorithms calculate the velocity that is closest to the preferred velocity and still safe. This implies that the robots always pass each other within only marginal distances. While this approach is feasible in simulation, in real world applications it is not possible to exactly control the velocity of the robots. With only marginal distances between the robots that pass each other, there is an increased risk that the smallest error in control will lead to a collision. An additional limitation is that all agents, either human or robot, are treated in the same way, while it would be desirable to have more distance from humans than from other robots. Also, if a robot knows that another robot is running the same algorithm (e.g. by using communication), it can make use of the assumption that the other robot will partly take avoiding actions as well. Hence, the robot can drive closer to that robot, while towards other robots and in particular in the presence of humans more distance is advisable.

In this paper, we introduce a pro-active local collision avoidance system for multi-robot systems in a shared workspace, that aims to overcome the stated limitations. The robots use the velocity obstacle paradigm to choose their velocities in the input space. But instead of choosing only the closest velocity to the preferred velocity, more cost features are introduced in order to evaluate which one is the best velocity to choose. This allows us to apply different weights or importance factors for passing humans, other robots, and static obstacles. Furthermore, we introduce a smart sampling technique that limits the need to sample throughout the whole velocity space. The resulting algorithm is decentralized with low computational complexity, such that the calculations can be performed online in real time, even on lower-end on board computers. Our empirical evaluation shows that the resulting paths of the robots are safer and smoother, while not adding much overhead w.r.t. distance and time travelled. The evaluation also shows that the approach leads to substantially less collisions in very dense configurations.



(a) Workspace configuration (b) $VO_{A|B}$

Figure 1. (a) A workspace configuration with two robots R_A and R_B on collision course. They are described by a position, radius and velocity (i.e. p_A, r_A and v_A for robot R_A). (b) Translating the workspace configuration into velocity space and the resulting velocity obstacle ($VO_{A|B}$) for R_A.



(a) Truncated VO (VO^τ) (b) Approximated VO^τ

Figure 2. Truncation. (a) Truncation of a VO of a static obstacle at $\tau = 2$. (b) Approximating the truncation by a line for easier calculation.

The remainder of the paper is structured as follows: The next section introduces the velocity obstacle paradigm and discusses the state of the art. Section 3 presents our new approach to improve safety in a shared workspace environment. In Section 4, the algorithm is empirically evaluated and compared against the existing COCALU approach. Related work is reviewed in Section 5. Section 6 concludes the paper and describes current and future work.

2. Background

2.1. Velocity Obstacles

The velocity obstacle (VO) was first introduced by Fiorini et al. [4] for local collision avoidance and navigation in dynamic environments with multiple moving objects. In the following, we will introduce the basic concept of velocity obstacles and some of its extensions.

Let us assume a workspace configuration with two robots on a collision course as shown in Figure 1a. If the position and speed of the moving object (robot R_B) is known to R_A, we can mark a region in the robot's velocity space that leads to collision under current velocities and is thus unsafe. This region resembles a cone with the apex at R_B's velocity v_B, and two rays that are tangential to the convex hull of the Minkowski sum of the footprints of the two robots. The Minkowski sum for two sets of points A and B is defined as:

$$A \oplus B = \{a + b | a \in A, b \in B\} \tag{1}$$

The direction of the left and right ray is then defined as:

$$\theta_{left} = \max_{p_i \in \mathcal{F}_A \oplus \mathcal{F}_B} atan2((p_{rel}+p_i)^\perp \cdot p_{rel}, (p_{rel}+p_i) \cdot p_{rel})$$

$$\theta_{right} = \min_{p_i \in \mathcal{F}_A \oplus \mathcal{F}_B} atan2((p_{rel}+p_i)^\perp \cdot p_{rel}, (p_{rel}+p_i) \cdot p_{rel})$$

where p_{rel} is the relative position of the two robots and $\mathcal{F}_A \oplus \mathcal{F}_B$ is the convex hull of the Minkowski sum of the footprints of the two robots, in the case of round robots, this is a circle with the sum of the robots' radii as exhibited in Figure 1b. The $atan2$ expression computes the signed angle between two vectors. The resulting angles θ_{left} and θ_{right} are left and right of p_{rel}.

When the workspace is cluttered with many robots that do not move or only move slowly, the apexes of the VOs are close to or even at the origin in velocity space; thus rendering the robots immobile, since the whole velocity space gets invalidated. This problem can be solved using truncation.

The idea of a truncating a VO can be best explained by imagining a static obstacle. Any velocity in the direction of the obstacle will eventually lead to collision, but not immediately. Hence, we can define an area in which the selected velocities are safe for at least τ time steps (see Figure 2a). The truncation can be closely approximated by a line perpendicular to the relative position and tangential to the Minkowski sum of the two footprints as shown in Figure 2b.

The velocity obstacle paradigm was extended to incorporate reciprocality to the reciprocal velocity obstacle (RVO) [5]. This approach assumes that each agents takes half the responsibility for the collision avoidance. This result was further refined to the hybrid reciprocal velocity obstacle (HRVO) [6] to overcome situations in which the reciprocal velocity obstacle could lead to reciprocal dances [7], since the sides on which the robot wants to pass switches with each time step.

In order to execute the computed collision free velocity, the robot has to be able to instantaneously accelerate to any velocity in the two dimensional velocity space. This implies that the velocity obstacle approach requires a fully actuated holonomic platform, able to accelerate into any direction from any state. However, differential drive robots with only two motorized wheels are much more common due to their lower price point. Additionally, robots can only accelerate and decelerate within certain dynamic constraints.

If the acceleration limits and motion model of the robot are known, a region of admissible velocities can be calculated and approximated by a convex polygon.

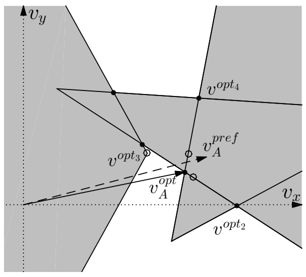


FIGURE 3. ClearPath enumerates intersection points for all pairs of VOs (solid dots). In addition the preferred velocity v_A is projected on the closest leg of each VO (open dots). The point closest to the preferred velocity (dashed line) and outside of all VOs is selected as new velocity (solid line). The next best points are shown for reference.

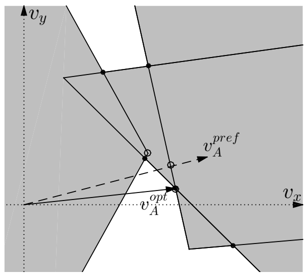


FIGURE 4. Increasing the footprint of the other robots as one way to create more safety. However this also reduces the available safe velocities to choose from and could lead to potential problems in dense configurations, where the whole velocity space becomes unavailable.

These constraints can easily be added to the VO formulation to be incorporated in the calculation of the new velocity, by dynamically restricting the velocity space to only achievable velocities. Another approach is using non-linear velocity obstacles as presented in [8].

To incorporate the differential drive constraints, Kluge et al. introduced a method to calculate the effective center of a differential drive robot [9]. The effective center represents a translation of the center of rotation to a point that can virtually move into all directions. Another method to handle non-holonomic robot kinematics has been introduced by Alonso-Mora et al [10]. This is based on the idea that every robot can track a holonomic speed with an error, i.e. driving an arc and then following the holonomic speed vector.

2.2. COCALU

The COCALU algorithm was introduced [2] to efficiently compute collision free velocities in multi-robot systems. It is based on ClearPath, introduced by Guy et al. [11]. This algorithm is applicable to many variations of velocity obstacles (VO, RVO or HRVO) represented by line segments or rays. ClearPath follows the general idea that the collision free velocity that is closest to preferred velocity is: (a) on the intersection of two line segments of any two velocity obstacles, or (b) the projection of the preferred velocity onto the closest leg of each velocity obstacle. All points that are within another obstacle are discarded and from the remaining set the one closest to the preferred velocity is selected. Figure 3 shows the graphical interpretation of the algorithm.

In COCALU, the algorithm is further adapted to incorporate sensing uncertainties by enlarging the footprint using a localization uncertainty measure. Assuming that the robot uses Adaptive Monte Carlo Localization (AMCL) [12], the weighted particle cloud can be used as an approximation of the probability distribution of the robots' location. By calculating the Minkowski sum of the robots' footprint and the convex hull of the particle cloud, the robots' localization uncertainty can be taken into account in straight forward manner. In order to prevent that the footprint enlargement becomes too big, convex hull peeling is used to remove particles from the set until a ϵ-threshold is exceeded. This effectively limits the footprint enlargement while still providing extra safety during navigation.

2.3. PROBLEMS

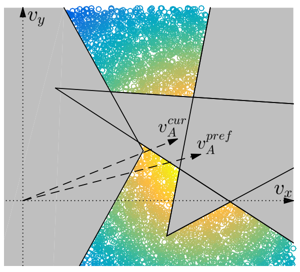
As explained in the introduction, some problems and limitations remain. For one, optimality can be defined in many ways. In the case of COCALU, optimality means driving as close as possible to the desired speed. In many cases this implies that robots using this algorithm pass each other close to zero distances, i.e. there is no margin for error. In real life, where control of the robot is not instantaneous and perfectly accurate, this can lead to collisions. While COCALU implicitly provides safety by enlarging the robots' footprints by the localization uncertainty, this is not an optimal solution. Especially when the localization accuracy is high, the point cloud converges to the actual robots position and the safety region decreases. Thus we need to explicitly take this into account.

A second limitation of the approach is that it is perceived as uncomfortable or even unsafe by humans when the robots pass unnecessarily close by. An intrusion of ones personal space is seldomly appreciated, especially when it concerns a robot.

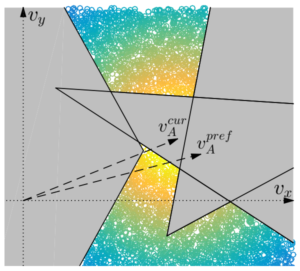
3. TOWARDS HUMAN-SAFE PRO-ACTIVE COLLISION AVOIDANCE

Our algorithm has the same assumptions as COCALU. The robots have to be able to sense velocity and shape of other robots and humans. Since robot-robot detection is a whole research field in itself, we rely on communication between the robots. More specifically, the robots use the same global reference frame and constantly broadcast their positions via WiFi.

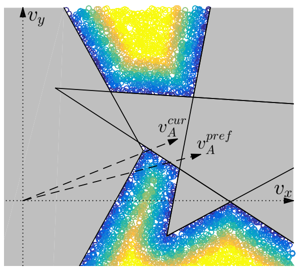
There are multiple ways to ensure that the robots are passing with more distance to each other. One straightforward idea is to virtually increase the size of the robots' footprints. This results in larger velocity



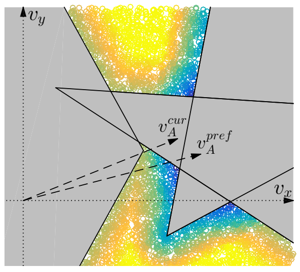
(a) Costmap v^{pref}



(b) Costmap v^{cur}

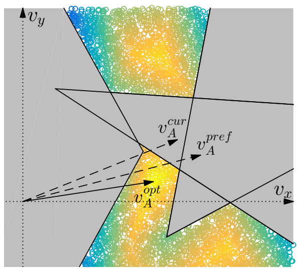


(c) Costmap $dist_{VO}$ with same weights

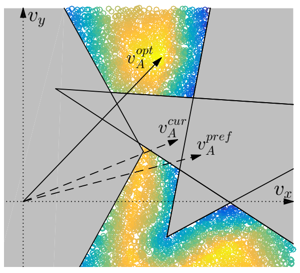


(d) Costmap $dist_{VO}$ with different weights

FIGURE 5. Different cost functions. Yellow depicts lower, i.e. better, and blue higher costs. The distances to the preferred velocity (a) and current velocity (b) as cost where further away yields higher cost; (c) and (d) show the distances to the VOs as costmaps, where points closer to the VOs yield higher cost.



(a) v^{opt} with all VOs weighted equally



(b) v^{opt} with different VO weights

FIGURE 6. Selecting the optimal velocity based on different combinations of the costmaps and sampling throughout the full velocity space. (a) All VOs are weighted equally. (b) The VO on the right has additional weight.

obstacles and thus the robots will have more distance between one another. However, this also drastically reduces the safe velocity space as shown in Figure 4. This approach marks more regions in the velocity unsafe and therefore reduces the options to choose from. It can lead to problems in dense situations, when many other robots are present and the entire velocity space is marked unsafe, while it still would be possible to maneuver without collisions.

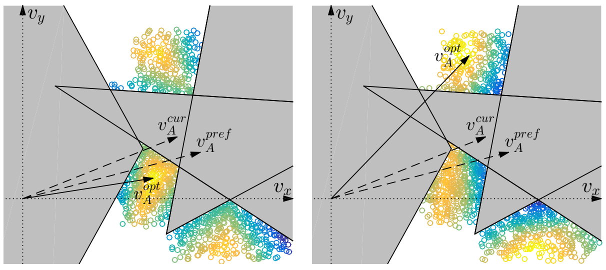
To overcome this problem, we use a sampling based approach with multiple cost functions. This means that the chosen velocities get evaluated not only by their distances from the preferred goal velocity but multiple other evaluation functions. Figure 5 shows the result of different evaluation functions in the example setting. The distances of the sampled velocity against the preferred velocity but also against the current velocity are shown. Additionally, the closest distance to any velocity obstacle can be modeled as negative cost. The resulting distance can be limited, i.e. that points which are further away than a set distance do not get scored higher. This can effectively control the behaviour of the robot. Additionally, if we assume that a velocity obstacle is induced by a human, this can be weighted differently than the distances from the other velocity obstacles. The effect is shown in Figures 5c and 5d, where the right most velocity obstacle is weighted with double the cost than the other two velocity obstacles. Using this approach we can model the personal space of a human by setting the cost for intrusion very high up to a certain distance. For personal space, a distance of 50 cm is usually regarded as applicable [13].

In order to select the optimal velocity, we can sample the velocity space and then evaluate samples based on the weighted sum of the different costmaps. A velocity sample that points inside a VO is disregarded, since it is unsafe. Figure 6 shows the costmaps and the resulting optimal velocity. As can be seen in Figure 6a, the resulting velocity is close to the originally calculated optimal velocity when using ClearPath. However, when the VOs are weighted differently, the optimal velocity is in a different region of the velocity space as shown in Figure 6b.

We can combine the ClearPath algorithm with the above idea to incorporate a smarter sampling algorithm. The ranked velocities calculated by ClearPath are used as a seed (see Figure 3: points marked as v^{opt_i}), such that samples are only created in the vicinity of these velocities. Figure 7 shows the idea of this algorithm. The trade-off of this approach is that it might miss the global optimum in favor of being computationally faster.

An advantage when using a velocity obstacle based approach is that we can easily have pro-active collision avoidance, even when the robots are standing still. When standing still, the preferred velocity is zero, that can be evaluated using the same approach as while driving. Thus, when staying still would result in a collision, the robots using this approach will pro-actively take actions and avoid the incoming robot or human. This is of course only necessary when the incoming robot is not already taking care of the avoidance itself.

Lastly, we can also adapt the truncation factor to improve the safety against other uncontrolled robots and humans. A higher truncation time results in safer velocities, since, as stated in Section 2, it determines the time the chosen velocity is guaranteed to be collision free in the current configuration of the system.



(a) v^{opt} with same VO weights and smart sampling

(b) v^{opt} with different VO weights and smart sampling

FIGURE 7. Applying smart sampling only around the best v_{cp}^{opt} points as calculated by ClearPath. (a) All VOs are weighted equally. (b) The VO on the right has additional weight.

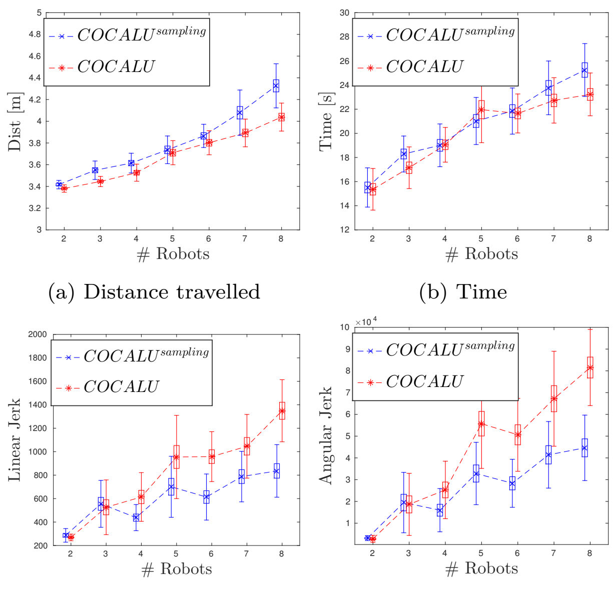
4. Evaluation and Results

The presented algorithms are implemented in the framework of the open source *Robot Operating System (ROS)* [14]. As described above we rely on communication between the robots to broadcast their positions in a common reference frame. The robots are controlled with 10 Hz, and at each timestep the robots evaluate the current position and independently choose their preferred velocity.

We have evaluated our approach in simulation using *Stage* [15, 16] and in real-world settings. Simulation allows us to investigate the system performance using many repetition and various extreme settings.

All experiments in simulation are run on a single machine with a quad-core 3.4 GHz Intel i7 processor and 16 GB of memory. Each setting is repeated 50 times and the results are averaged. Runs in which collisions occurred or which exceeded a time limit of 60 seconds are excluded from the averages. The completed runs are split into seven bins, and the variance of the batch means is used to calculate 90% confidence intervals using the students t-distribution with six degrees of freedom and $\alpha = 0.1$. The simulation are run in real time, since the message passing is an essential component of the described approach. As the ROS message passing uses real time serialization and deserialization, increasing the simulation speed would lead to inaccurate results.

We compare the original $COCALU$ with the our newly proposed $COCALU^{sampling}$. A common scenario for evaluation dense movements are a different number of robots located on a circle (equally spaced). The goals are located on the antipodal positions, i.e. each robot's shortest path is through the center of the circle (see [1, 10]). We use a circle with a radius of 1.7 meter in simulation. The goal is assumed to be reached when the robots center is within a 0.15 meter radius of the true goal. We evaluate several performance measures: a) number of collisions, b) time to complete a single run, c) distance travelled and d) jerk cost. The jerk cost measures the smoothness of a



(a) Distance travelled (b) Time

(c) Linear Jerk (d) Angular Jerk

FIGURE 8. Results of the antipodal circle setting. (a) Distance travelled, (b) time to complete, and linear (c) and angular (d) jerk are shown.

path and is defined as:

$$Jerk_{lin} = \frac{1}{2} \int \dddot{\mathbf{x}}(t)\ dt \tag{2}$$

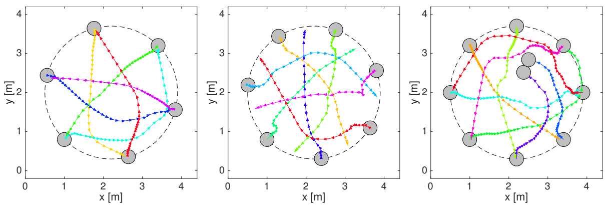
$$Jerk_{ang} = \frac{1}{2} \int \dddot{\theta}(t)\ dt \tag{3}$$

where $\mathbf{x}$ is the forward displacement of the robot, i.e. the linear speed is $\dot{\mathbf{x}}$ and θ the robot's heading, i.e. $\dot{\theta}$ is the angular speed.

For both algorithms we use truncation of the velocity obstacles with $\tau = 10$. For humans and uncontrolled robots, we increase the τ to 40. The $\epsilon = 0.1$ for the localization uncertainty is set to, thus we include 90% of the particles in our footprint enlargement.

All costmaps are included for the sampling approach and weighted equally. In the cases where there is an velocity obstacle induced by an uncontrolled robot or a human present, the minimum distance to these velocity obstacles is weighted double.

Figure 8 shows the results of the antipodal circle setting. The boxes are the confidence intervals and the errorbars show the standard deviation over the 50 runs. The distance travelled and time used are longer for the new $COCALU^{sampling}$ when compared to the original $COCALU$ formulation, (Figure 8a and 8b). This is to be expected, since the robots deviate from the fastest path in order to improve safety. However, it can also be seen that the resulting paths are significantly smoother for the new approach. The linear and angular jerk (Figure 8c and 8d) show that especially for more robots, the sampling based approach improves the smoothness of the paths. This can be explained by the fact that with the original $COCALU$, the robots are driving very close to the


(a) Standard *COCALU*.

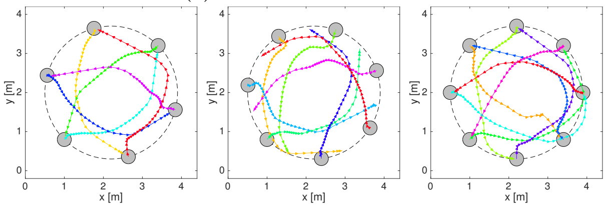

(b) $COCALU^{sampling}$

FIGURE 9. Sample trajectories for the antipodal circle setting, with 6, 7 and 8 robots from left to right. For the standard *COCALU* (a) and 8 robots, we can see a collision between the robots with the blue and purple traces. With $COCALU^{sampling}$ no collision is happening, but the general distances between the robots is larger.

other robots, leading almost to collisions. Especially if the control is not perfect, some radical maneuvers might be needed to get out the collision. Additionally, the $COCALU^{sampling}$ approach specifically models the smoothness of the path by taking the distance to the current velocity into account.

Regarding the amount of collisions, with $COCALU^{sampling}$ only in one run with eight robots a "touch and go" collision occurred, while for standard *COCALU* already for five robots "touch and go" collisions occurred. With seven robots two not resolvable collisions and with eight robots 14 not resolvable collisions were recorded. That collisions still do happen is mainly due to the limited update rate of 10 Hz and the low fidelity of the simulator. The localization uncertainty epsilon was set to 0.1, so there is also a small chance that collisions happen, when AMCL is unable to track the robots' positions well enough.

Additionally, since we are using truncation, we can only guarantee safety of a velocity for $\tau = 10$ timesteps. This might lead to states where a collision is inevitable, especially when using the original *COCALU* formulation.

Some example trajectories of the two approaches are exhibited in Figure 9. The top row shows the results for *COCALU* and six, seven and eight robots, and the bottom row shows corresponding trajectories for $COCALU^{sampling}$. The trajectories confirm the results before, the trajectories with *COCALU* are much closer together and even show a collision with eight robots. The robots with the blue and purple traces collided in a way that they could not resolve it. For $COCALU^{sampling}$, the trajectories are generally more spaced from the center.

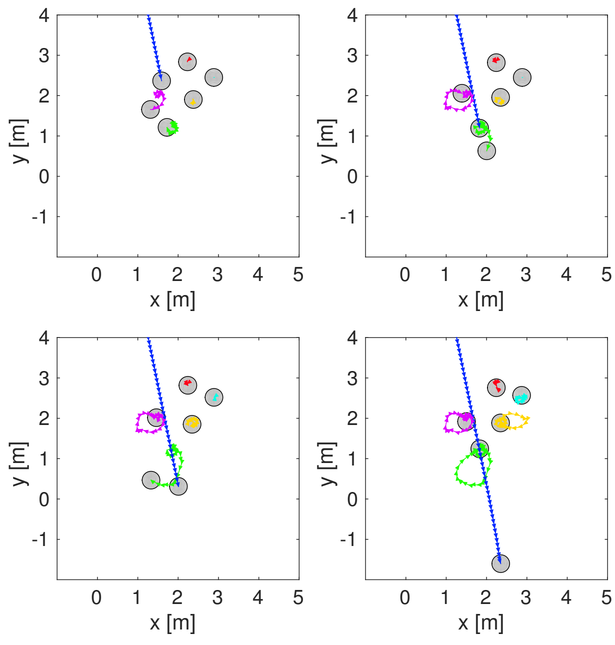


FIGURE 10. Pro-active collision avoidance. The "uncontrolled" robot (blue trace) neglects the presence of the other robots, so they have to move out of its way to ensure safety.

To show how the pro-active collision avoidance works, Figure 10 shows the trajectories of one "uncontrolled" robot passing trough a crowd of robots. "Uncontrolled" in this experiment means that the robot disregards the existence of the other agents and just drives straight, without taking any avoiding measures. Thus, the five robots in the center have to pro-actively move out of the way in order to ensure safety. The robot with the blue trace, is approaching while the pink and green traced robots start moving out of the way (top-left). As soon as the uncontrolled robot has passed, pink returns to its position (top-right). The same happens with the green robot. The final positions and complete trajectories can be sees in the bottom-right corner.

4.1. Human-robot experiments

We evaluated the performance in a the real-word setting using up to three differential drive Turtlebot 2's[1]. In addition to the usual sensors, they are equipped with a Hokuyo URG laser-range finder to enable better localization in large spaces. All computation is performed on-board on a Intel i3 380UM 1.3 GHz dual core CPU notebook. Communication between the robots is realized via a 2.4 GHz WiFi link using a UDP connection and the LCM library [17]. For human detection we use the SPENCER project code[2]. A video showing the results can be found here: `https://youtu.be/hcd6phm2ocs`.

[1]For more information see: `http://turtlebot.com`.
[2]`https://github.com/spencer-project/spencer_people_tracking`

5. Related Work

This work introduces an local collision avoidance approach that deals with the problem of robots sharing the same workspace as humans. An overview over existing (global and local) approaches for human aware navigation [13] shows that the main focus of current research is about the comfort, naturalness and sociability of robots in human environments. This usually entails only one robot acting in a group of humans, i.e. as a personal assistent. However our approach is aimed at a different distribution of agents, namely many robots navigating together with many humans in the same shared workspace.

An example of the single robot, multi-human navigation approach is the stochastic CAO approach [18], which models the discomfort of humans and uses the prediction of human movement to navigate safely around people. Another similar approach is described in [19]. It is based on layered costmaps in the configuration space and it also describes a user study where gaze-detection was used to determine the intended heading of the humans to update the costs. This layered costmaps idea is similar to the multiple evaluation functions in our approach; however it is based on the configuration space, while our approach uses the velocity space. This approach does not explicitly cover the dynamic nature of moving obstacles.

Other approaches for multi-robot collision avoidance use auctions [20] at a rather high communication overhead; or stigmergy [21–23], which relies on pheromones that are hard to apply in a real world setting. Additionally, these approaches do not implement robot-human avoidance.

6. Conclusions and future work

In this paper, we introduced a pro-active local collision avoidance system for multi-robot systems in a shared workspace. It improves upon previous work by incorporating different importance factors for humans, other robots, and static obstacles. Instead of only relying on calculating the optimal velocity based on the preferred velocity, we use Monte Carlo sampling throughout the velocity space and evaluate the samples using multiple different cost functions. We presented a smart sampling technique, using the previously computed velocities from ClearPath as seeds, so that we can limit the sampling to only interesting regions in the velocity space. The resulting algorithm is decentralized with low computational complexity, that can be run even on low-end robots, as for instance the Turtlebot 2.

In a setting where the robots have to drive to the antipodal positions on a circle, we have evaluated our adaptations and compared against the original CO-CALU formulation, showing that it results in smoother and safer at the small cost of slightly longer paths. We also presented how robots can pro-actively avoid collisions, by showing an "uncontrolled" robot driving through a crowd of other robots. In our real world settings, up to three Turtlebots avoid a human sharing their workspace. Current work includes further comparisons to evaluate the influences of the different costmaps and the effect of different weights.

The approach can be generalized to any input space, using for instance the Generalized Velocity Obstacles [24], or the Acceleration Velocity Obstacles [8]. Additionally, this approach can be combined with existing approaches that use trajectory rollouts as for instance the Dynamic Window Approach [25], in a straight forward manner. Adding human gaze detection to improve the prediction of the human velocities is also a promising direction for future work.

References

[1] J. van den Berg, S. Guy, M. Lin, D. Manocha. Reciprocal n-body collision avoidance. *Springer Tracts in Advanced Robotics* **70**:3–19, 2011.

[2] D. Claes, D. Hennes, K. Tuyls, W. Meeussen. Collision avoidance under bounded localization uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Algarve, Portugal, 2012.

[3] D. Hennes, D. Claes, K. Tuyls, W. Meeussen. Multi-robot collision avoidance with localization uncertainty. In *Int. Conf. on Autonomous Agents and Multiagent Systems*. 2012.

[4] P. Fiorini, Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Robotics Research* **17**:760–772, 1998.

[5] J. van den Berg, M. Lin, D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Int. Conf. on Robotics & Automation (ICRA)*. 2008.

[6] J. Snape, J. van den Berg, S. J. Guy, D. Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Int. Conf. on Intelligent Robots and Systems*. 2009.

[7] B. Kluge, E. Prassler. Reflective navigation: Individual behaviors and group behaviors. In *Proceedings of the IEEE International Conference on Robots and Automation (ICRA)*, pp. 4172–4177. 2004.

[8] J. van den Berg, J. Snape, S. Guy, D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. In *Int. Conf. on Robotics & Automation (ICRA)*. 2011.

[9] B. Kluge, D. Bank, E. Prassler, M. Strobel. Coordinating the motion of a human and a robot in a crowded, natural environment. In *Advances in Human-Robot Interaction*, vol. 14, pp. 207–219. Springer, 2005.

[10] J. Alonso-Mora, A. Breitenmoser, M. Rufli, et al. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Symposium on Distributed Autonomous Robotic Systems (DARS)*. 2010.

[11] S. J. Guy, J. Chhugani, C. Kim, et al. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Symposium on Computer Animation*. 2009.

[12] D. Fox. Adapting the sample size in particle filters through kld-sampling. *Robotics Research* **22**, 2003.

[13] T. Kruse, A. K. Pandey, R. Alami, A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems* **61**(12):1726–1743, 2013.

[14] M. Quigley, et al. ROS: An open-source Robot Operating System. In *Proc. of the Open-Source Software Workshop (ICRA)*. 2009.

[15] B. P. Gerkey, R. T. Vaughan, A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317–323. 2003.

[16] R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence* **2**(2):189–208, 2008.

[17] A. S. Huang, E. Olson, D. C. Moore. Lcm: Lightweight communications and marshalling. In *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*, pp. 4057–4062. IEEE, 2010.

[18] J. Rios-Martinez, A. Renzaglia, A. Spalanzani, et al. Navigating between people: A stochastic optimization approach. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2880–2885. IEEE, 2012.

[19] D. V. Lu. *Contextualized Robot Navigation*. Ph.D. thesis, Washington University, 2014.

[20] J.-P. Calliess, D. Lyons, U. D. Hanebeck. Lazy auctions for multi-robot collision avoidance and motion control under uncertainty. In *Advanced Agent Technology*, pp. 295–312. Springer, 2012.

[21] G. Theraulaz, E. Bonabeau. A brief history of stigmergy. *Artificial life* **5**(2):97–116, 1999.

[22] N. Lemmens, K. Tuyls. Stigmergic landmark optimization. *Advances in Complex Systems* **15**(8), 2012.

[23] F. B. von der Osten, M. Kirley, T. Miller. Anticipatory stigmergic collision avoidance under noise. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pp. 65–72. ACM, 2014.

[24] D. Wilkie, J. Van den Berg, D. Manocha. Generalized velocity obstacles. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 5573–5578. IEEE, 2009.

[25] D. Fox, W. Burgard, S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE* **4**(1):23 –33, 1997. DOI:10.1109/100.580977.