Proceedings of the 2014 IEEE/SICE International
Symposium on System Integration, Chuo University,
Tokyo, Japan, December 13-15, 2014

Dynamic Obstacle Avoidance based on Curvature Arcs

Eduardo Molinos, Ángel Llamazares, Manuel Ocaña and Fernando Herranz

Abstract— **Traditionally, obstacle avoidance algorithms have been developed and applied successfully to mobile robots that work in controlled and static environments. But, when working in real scenarios the problem becomes complex since the scenario is dynamic and the algorithms must be enhanced in order to deal with moving objects. In this paper we propose a new method based on the well known Curvature Velocity Method (CVM) and a probabilistic 3D occupancy and velocity grid, developed by the authors, that can deal with these dynamic scenarios. The proposal is validated in real and simulated environments.**

I. Introduction

Robots teams have been used to support a variety of human activities, including planning and decision making. In recent years, these teams have started to include humans as active components that sense, act, and collaborate with their automated counterparts. In this way the human-robot collaboration means to work side by side between team's members, i.e. robots guiding humans in museums, human-robot surveillance teams, etc. While robots can acquire vast amounts of processed, analysed, interpreted, and fused data, humans can provide their ability to understand the scenarios and solve complex problems to ensure the right functioning of human-robot teams in terms of coordination, negotiation, distribution, and cooperation [1].

To achieve an effective collaboration of humans and robots is especially important to develop a robust autonomous navigation system which keeps human integrity. This kind of systems have been deeply studied in the literature [2] [3]. Autonomous navigation systems usually execute the next four stages: perception (provides information about the surroundings of robot), localization (determines the position of the robot in a map), cognitive processes (global and local planning, decision making, etc.) and motors modulation.

Movement planning can be divided into global planning processes and obstacle avoidance systems. While global planning processes determine what is the better way to approach the target, local planning is in charge of avoiding obstacles and keep the robot-human integrity. For this planning is important to use maps, which can be generated in manual or semi-autonomus way, and they can include occupancy or occupancy and velocity information [4].

Obstacle avoidance systems can also be divided into global or local depending of the available information about the environment. The first ones assume a complete model of the environment such as potential field methods [5]. Local methods can be considered faster and can be programmed like reactive tasks [6] [7] [8]. These reactive methods control the robot when an obstacle is detected and avoid the collision. They use the nearest portion of the environment and update the world model according to the current sensor observation.

Authors are with Department of Electronics, University of Alcalá, Madrid, Spain `emolinos, allamazares, mocana, fherranz @depeca.uah.es`

In this work, we propose an extension of a well known reactive method called Curvature Velocity Method (CVM) to include the information provided by our previous developed DOMap [4]. We test the proposal both in simulated and real scenarios to achieve a robust and safety autonomous navigation. In addition, we evaluate the effect of dealing with mobile obstacles.

The rest of the paper is organized as follows: section 2 shows a selection of related works; the proposed method is shown in section 3; section 4 describes the test-bed and some experimental results; and finally, section 5 enumerates conclusions and future works.

II. Related works

This section provides a description of previous works in obstacle avoidance. As the real world is dynamic, with people walking around, cars or others robots moving, obstacle avoidance algorithms have to deal with these moving obstacles. For that reason, is crucial to know not only the position of those objects but also the velocity of them. The classical methods, such as [5] [6] [7], do not take into account these changes, then for each time step all the obstacles are considered static. This sometimes becomes a problem, because a moving obstacle in front of the robot can be considered as a wall, so the robot assumes an unavoidable obstacle or spends too much time on the manoeuvre. For that reason obtaining the obstacle's speed and direction is important in order to determinate its next position. However getting that information is not always possible if the movement of the obstacle is either erratic or too far from the sensor to get an accurate measurement. For that reason, obstacle avoidance can be divided into two stages: local mapping stage and obstacle avoidance stage.

So many works, that detect the movement of the obstacles, limit their movement to linear and inside an area, such as the road [9]. Or only track one kind of object, as [10] that introduce a method to detect people using the local minimums that produce the legs. In [11] authors extend the previous one by needing that the objects enter in a object-free regions.

On the other hand, the goal of an obstacle avoidance algorithm is to follow a previously computed path (usually by a global planner working at a different layer) avoiding obstacles that are not previously mapped and can be detected by the local mapping stage.

Several approximations for this problem have been developed over the years. The first ones are based on vectors summations, such as Vector Field Forces (VFF) [6] or Potential Fields (PF) [5]. These ideas work well in simulation and are computationally simple but can not deal very well with uncertainties in the local mapping stage. Then, another algorithms based on angular sectors histogram of potential risk were developed, such as Vector Field Histogram (VFH) [12] and VFH+ [7]. In these algorithms the movement of the robot is selected by a weight function with different parameters, such as security or distance to the goal, and can deal better with uncertainties of the measures but they have problems with local minimums at certain situations. Trying to avoid these local minimums another kind of algorithms classify the environment into different kind of scenarios and follow different strategies of movement on each one. Examples of these algorithms are Nearness Diagram (ND) [13] and Smooth Nearness Diagram (SND) [14].

There is other kind of algorithms that try to deal with the dynamic of real robots. In order to do that, they work in a velocities space that is restricted by the real capacities of the robot. Examples of this kind of algorithms are Dynamic Window Approach [15], which is the most used algorithm and is integrated into ROS (Robot Operating System) standard packages, and Curvature Velocity Method (CVM) [8], which we will extensively describe in the proposal section.

Another idea to deal with complex environment is to create a stage between global path planning and obstacle avoidance to plan local goals that should be reach using another obstacle avoidance algorithm. Examples of this kind of algorithms are Lane Curvature Method [16] and Beam Curvature Method [17] that divide the environment into different shapes (lanes and angular sectors) and use CVM in order to reach local goals. Authors of this paper compared several of these algorithms in [18] [19].

These algorithms deal relatively well with static obstacles, but in the last years the need for dynamic obstacle avoidance has increased, especially in the automated cars area. Automated cars use expensive sensors to obtain 3D information of the environment and can detect and deal well with moving obstacles. In [20], [21] and [22] are explained different techniques used by automated cars. But these algorithms for cars have several constraints (velocity limits, lanes of the road, non-holonomic turns), that are very different with respect to the differential mobile robots used in this paper.

III. Proposed method

In this section we show our proposal to autonomously navigate avoiding dynamic obstacles into the ABSYNTHE project framework. We propose modifications in two stages: local mapping stage and obstacle avoidance stage.

The local mapping stage is based on a dynamic occupancy grid called DOMap (Dynamic Occupancy Mapping) [4]. Each cell of this grid stores the probability of occupancy and the estimation of its velocities (v_x, v_y). The occupancy likelihood is obtained by a Bayesian Occupancy Filter and the velocity estimation is based on a detection of the obstacles movement using the pyramidal implementation of Lukas Kanade optical Flow [23] and a Kalman filter to track the objects. With this combination of occupancy likelihood and velocity estimation of the obstacles, the algorithm is able to obtain a complex estimation of the surrounding environment.

The obstacle avoidance stage was studied in our previous works [18] [19] where we proved that CVM has a good performance in static environments and is easily understandable. We extend this idea to a dynamic obstacle avoidance aided by the DOMap perception stage. First we describes our implementation of CVM and then our extensions (Predicted CVM and Dynamic CVM) of it to avoid dynamic obstacles.

1) Curvature Velocity Method (CVM): this algorithm is based on the idea that an holonomic robot moves in curvature arcs formed by a pair of angular and linear velocities. In an obstacles' free environment a robot can choose any pair of angular and linear velocities only restricted by its kinematic constraints (limits in acceleration and velocities). In a practical way this is computationally expensive so it has to be discretized. The best way to reduce the number of pair of angular and linear velocities to choose is to add restrictions, and some restrictions can be created by the obstacles in the environment.

In order to create these restrictions, the environment is modelled as angular sectors. If there are no obstacles, all the environment is considered as a single angular sector. When obstacles are detected, this single sector is modified and divided into more angular sectors. In order to do that, each point detected in the range sensor is considered as a circumference augmented by a security radius with four coordinates (x and y, $maximum$ and $minimum$). From this detection two different curvature arcs can be calculated: one to the nearest point and one to the furthest. So, for each obstacle we obtain two more sectors. Sectors are described by its limits (curvature arcs) and distance free of obstacles d_{free}. Also a curvature arc to the goal is created in order to calculate trajectories to it. Once all the sectors are computed, a merging algorithm deletes the curvature sectors that are very thin or has similar distance free of obstacles to the neighbours to reduce the computational cost.

Our implementation also discretizes each sector in five curvature arcs that the robot can follow: pair of maximum linear and angular velocities (two pairs, right and left to the robot), curvature arcs that bound the sector and one arc that turns the robot to the goal. All the options are restricted by the kinematic of the robot. For each pair of velocities, a weight function is computed ((1), which depends of the linear (v_r) and angular (ω_r) velocities) that has three terms: velocity, heading and distance free of obstacles, each of them with different weights.

$$f(v_r, \omega_r) = \alpha_1 * speed(v_r) + \alpha_2 * dist(v_r, \omega_r) + \alpha_3 * head(\omega_r) \quad (1)$$

Speed term prizes higher linear speeds, heading term prizes arcs that faces the goal and distance free of obstacles term prizes curvature arcs with more distance to travel.

2) Predicted CVM (PCVM): in this approximation we change the perception stage of CVM using the information provided by DOMap perception. Instead of using the actual position of the obstacle we can use the prediction of the next position (given a certain time) if the obstacle is moving, so the algorithm can anticipate the avoidance manoeuvre. We tested this idea in [19].

3) Dynamic CVM (DCVM): the idea of this approach is to use DOMap as perception stage to predict the future positions of moving obstacles. Fig. 1 shows a scheme of the DCVM working.

The first step of the algorithm is to compute all the curvature arcs from the robot to static obstacles (blue boxes in Fig. 1) and the goal (red circle in Fig. 1) as CVM does. As each curvature arc is described by a pair of linear and angular velocities (v_r and ω_r) it can be modelled as a circumference described by its radius r and its centre points (x_c, y_c) (curved dashed lines in Fig. 1), knowing the robot's position (x_r, y_r). We also obtain the obstacles' movement from DOMap and model them as straight lines (blue dotted line in Fig. 1).

Moreover, using trigonometry we calculate if the robot and the obstacle movements intersects. If both movements intersects, there is a potential risk of collision (red and yellow stars in Fig. 1), so we calculate which time both (obstacle and robot) will be at this position, and if they are close in time, the collision is probable. In order to do that, we use (2) and (3) where x_p and y_p are the intersection points, t_{obs} and t_r are the times where the obstacle and the robot will be in the intersection point, x_{obs} the obstacle's position and v_{xobs} the obstacle's velocity in x axis. When a collision is probable, the parameter of the curvature arc that leads to the collision (d_{free}) is modified with the distance that the robot can travel until the collision with dynamic obstacle d'_{free}.

$$t_{obs} = x_p - x_{obs}/v_{xobs} \tag{2}$$

$$t_r = \frac{2\pi((atan(y_c - y_r, x_c - x_r)) - (atan(y_c - y_p, x_c - x_p))}{|\omega_r|} \tag{3}$$

In this way, the algorithm takes into account the dynamic obstacles and can predict where they will be and avoid them. But, if the velocity measures are not exact, the avoidance manoeuvre may not be good. So we propose a new weight function in the curvature arc calculation (4) to deal with these uncertainties. The new equation depends on the previous weight of static CVM ($f(v_r, \omega_r)$) and the *riskiness* value. As *riskiness* increases the probability of choosing a curvature arc decreases.

$$f'(v_r, \omega_r) = f(v_r, \omega_r) - \alpha_4 * (f(v_r, \omega_r)/3) * riskiness \tag{4}$$

The *riskiness* parameter takes three values, from smaller to greater *riskiness*: no intersection with dynamic obstacles (green dashed lines in Fig. 1), intersection with dynamic obstacles (yellow dashed line in Fig. 1) and possible collision with a dynamic obstacle (red dashed line in Fig. 1). If there are no intersections with dynamic obstacles, the weight of the curvature arc is not modified. If there is a possible collision the weight function drastically decreases. Therefore, we add a third possibility to deal with uncertainties in the velocities: when the robot and the obstacle paths intersect but there is not a probable collision. In this case it decreases the weight of the curvature arc. We consider that it is riskier to approach a dynamic obstacle, even when a collision is not certain, than a static one.

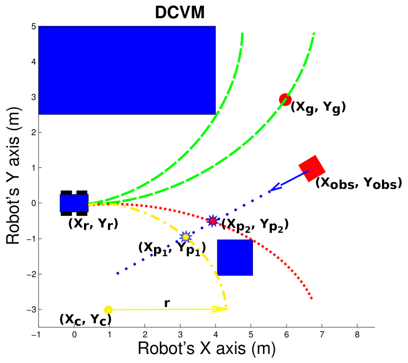


Fig. 1. Dynamic CVM: Scheme

IV. Test-bed and Results

Several tests have been carried out in simulation and real robots. ROS has been used as developing system.

A. Test-bed

A MobileRobots Pioneer 3-AT has been used as robotic platform. The maximum velocities of the robot are limited to 0.4 m/s (linear velocity) and 40 °/s (angular velocity). The range sensor used is a Hokuyo URG-04LX with 5 meters coverage in real robots and a Hokuyo UTM-30LX with 30 meters coverage in simulation. Both sensors have a field of view of 180 °. We use Gazebo 3D as simulator due to it allows us to simulate full 3D environments. We simulate moving obstacles as boxes of 0.6 m each side that move at a constant velocity and can be easily detected, while in real tests the moving obstacles are people which are harder to detect and their velocities are not constant. Laser is mounted approximately at 0.5 m from the floor to detect the legs.

We use a full autonomous robot framework (under our ABSYNTHE project). For all tests the robot starts in a known position and is localized only by its odometric sensors (without error in simulation). A Dijkstra path planner calculates the best path to the goal given and provides intermediate goals. Due to this path has usually too many intermediate goals, we filter it to one intermediate goal each three meters.

Same parameters are set for every test and algorithm in order to compare their behaviour. In the next paragraphs we described some of the most important ones like resolution in perception stage, tolerance to the goal, weight function in obstacle avoidance algorithms, and so on.

The perception stage (BOF and DOMap) uses a grid of 5 m in front of the robot (maximum coverage of real laser) divided into 20 cm occupancy cells. For PCVM we use a prediction of 2 seconds.

To consider that the robot has reached the final goal we set a tolerance of 30 cm. For intermediate goals we consider that the robot has reached any of them if it is nearer than 30 cm or is nearer to the next intermediate goal than to the current goal. This way we ensure that the robot approaches the global goal even when the obstacles in the environment make it hard to reach some intermediate goal.

For the obstacle avoidance algorithm, the parameters of the weight function in curvature arc selection are the following: 0.3 for distance weight α_2, 0.6 for heading weight α_3 and 0.1 for speed weight α_1. Using these parameters the robot chooses a curvature arc based on three criteria in order of importance: maximizing the chances of selecting an arc that faces the goal, then selecting a curvature arc with maximum distance without obstacles and finally selecting a path that maximizes the linear velocity of the robot. With this configuration that was selected via trial and error stage the algorithm is able to reach a global goal safely.

For DCVM we consider that if an obstacle and the robot are in the same point within 2 seconds difference there is a possible collision which makes this curvature arc maximum risky. If the obstacle and the robot are in the same point within 5 seconds difference we consider this curvature arc as medium risk. Also, security weight α_4 is set to 1 in order that $f'(v_r, \omega_r)$ has one third of the value of $f(v_r, \omega_r)$ if there is a possible collision.

B. Simulation results

We test different scenarios with the three combinations of algorithms and perception stages in order to compare them: normal CVM, DOMap + CVM (PCVM) and DOMap + DCVM. To perform this comparison we evaluate the obstacle avoidance manoeuvre performed and the parameters presented in table I. Some of these parameters measure the velocity of the manoeuvre: distance travelled d, time expended t and medium linear velocity $\overline{v}$, while others measure the smoothness of the path: medium absolute angular velocity $\overline{|\omega|}$ and velocities standard deviation: σ_v^2 and σ_ω^2.

1) Two obstacles crossing: in this scenario, the final goal is in front of the robot and two objects cross its natural path at 0.4 m/s (Fig. 2(a)).

When the first obstacle approaches, CVM and PCVM turn right to avoid it, which is risky because the obstacle is moving that way. Due to the fact that the obstacle travels faster than the robot turns, the obstacle surpasses the robot and both algorithms return to the straight trajectory. The behaviour is similar to the second obstacle. PCVM has a better performance because it avoids where the obstacle will be two seconds later. So the obstacles stop blocking the path earlier than CVM but is riskier. DCVM is able to predict that both obstacles will collide with the robot path even if it turns in the same way than CVM, so it avoids both obstacles behind them which is a better and safer avoidance even though the robot travels more distance.

2) Corner and two obstacles: in this experiment (Fig. 2(b)) the goal is at 45° from the start orientation of the robot and it is blocked by a big obstacle that must be rounded. There are two moving obstacles, one crossing the natural path of the robot from right to left at 0.4 m/s and one overtaking the robot at its left side at 0.75 m/s. Both obstacles cross at the big obstacle corner.

In this scenario all the CVM configurations are able to reach the goal but the avoidance manoeuvre is different. Both CVM and PCVM begin turning to the direction of the goal because they do not know where the moving obstacles will be and get stuck in the corner until both obstacles disappear. Once again, the prediction of the obstacles of PCVM make this performance better than CVM. DCVM is able to perform a safer avoidance. Since DOMap perception is able to calculate that both obstacles block the direct path to the goal the algorithm chooses an avoidance manoeuvre behind the crossing obstacle and is able to reach the goal following a more logical, safer, smoother and faster trajectory, even when the travelled distance is longer than the previous ones.

3) Corner and two obstacles approaching the robot: this experiment is set up in the same environment as the previous one with different positions of the moving obstacles (Fig. 2(c)): one crosses the natural path of the robot from left to right at 0.4 m/s and the other one approaches the robot at its left side at 0.75 m/s.

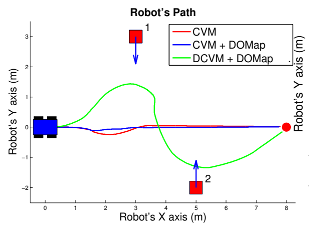
In this experiment, the three algorithms begin turning to the goal, but DCVM corrects its turning because the approaching obstacle intersects its path. Standard CVM and PCVM end their turns when the obstacle is close to the robot. The crossing obstacle affects all the experiments equally, the obstacle is detected very near to the robot because it is occluded by the big obstacle and cannot be avoided. So the algorithms wait until the obstacle does not block the path. As DCVM and PCVM use the future position of the obstacles, they perform a faster and better avoidance of the approaching obstacle than CVM.

4) Multiple obstacles and moving obstacle approaching the robot: this scenario is a more complex environment that mixes static and dynamic obstacles (Fig. 2(d)). Two more static obstacles are added to the previous scenario: one box in the path of the robot that forms a ”corridor” between the big obstacle and the box and another box in the corner turn that forms another (and wider) ”corridor”. In addition there is one moving obstacle to the robot by its diagonal and in collision course to it through the first corridor.

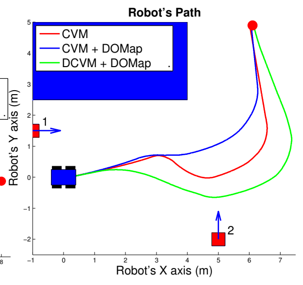
If there are only static obstacles the logical path is through the first corridor, so all the three algorithms try it first. When the obstacle approaches and occludes the corridor, it is so close to the robot that CVM and PCVM can not avoid it and collide with it. However, DCVM is able to predict the collision course of the obstacle earlier and turns right in order to avoid it. Once the first static obstacle is avoided, DCVM is able to avoid the other one and reach the goal.

5) Crossroad: in this scenario there are no static obstacles and the goal is at 45° of the robot orientation. Two moving obstacles in opposite directions simulate a crossroad in the street (Fig. 2(e)). In this scenario there is no information of the moving obstacles until the robot turns, so the obstacles are very close to the robot when they are detected.

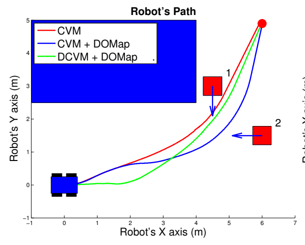
Since there is no information about the obstacles, all the algorithms begin turning left to face the goal. As CVM does not have any information of the velocity of the obstacles and the detection happens very close to the robot, it is not able to avoid them and crashes with the second one. As PCVM uses the future prediction of the obstacles, when they are detected their future prediction are in collision with the robot, and makes the algorithm stop. When the obstacle moves the robot continues toward its goal. DCVM is able to recognize that the first moving obstacle intersects the path of the robot and turns until it can be avoided with security. It avoids the second obstacle in the same way performing a safer and better avoidance even when PCVM obtains a shorter and faster path overall.



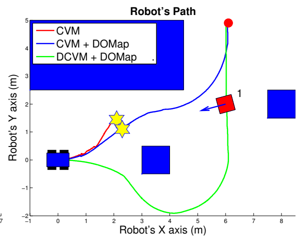
(a) Two Obstacles crossing



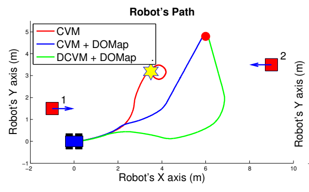
(b) Corner and two obstacles



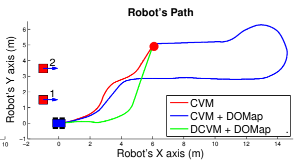
(c) Corner and two obstacles approaching the robot



(d) Multiple obstacles and moving obstacle approaching the robot



(e) Crossroad



(f) Lane changing

Fig. 2. Simulated Experiments: Paths

6) Lane changing: this scenario is a variation of the previous one with the moving obstacles travelling in the same direction as the robot, so it simulates a lane changing in a road (Fig. 2(f)). The obstacle nearer the robot travels at 0.75 m/s and the further one at 0.4 m/s.

The three algorithms begin turning to face the goal. When the first obstacle appears, CVM is able to avoid it slowing down its velocity and turning until the obstacle is not blocking the path (it travels faster than the robot). CVM avoid the second obstacle in the same way. In this case PCVM almost failed: it is able to avoid the first obstacle but when the second obstacle appears a local minimum happens: the obstacle and the robot move at the same speed. It makes the avoidance manoeuvre (and the arc selected) be the same on each iteration until a detection error slightly alter the obstacle detection allowing the algorithm to escape from the minimum and reach the goal. DCVM begins turning to the goal but when the robot detects that the first obstacle will cross its path turns again until it can be avoided by behind in a safer way. Once the first obstacle is avoided the robot continues in a straight path to the goal. In this case CVM obtains the shorter and faster path to the goal but the avoidance manoeuvre of DCVM is safer.

TABLE I
SIMULATED EXPERIMENTS: RESULTS

Experiment	Algorithm	d	t	$\overline{v}$	σ_v^2	$\overline{\lvert\omega\rvert}$	σ_ω^2
2 Obstacles Crossing	CVM	7.88	36.62	0.21	0.18	3.23	6.41
	CVM+DOMap	**7.79**	**25.07**	0.31	0.14	**3.01**	**5.66**
	DCVM+DOMap	10.32	28.92	**0.35**	**0.09**	11.38	13.96
Corner + 2 Obstacles	CVM	10.95	37.55	0.29	0.13	6.08	8.44
	CVM+DOMap	**9.42**	**33.12**	0.28	0.14	**5.27**	**6.78**
	DCVM+DOMap	12.31	37.70	**0.32**	**0.11**	5.49	7.05
Corner + 2 Obstacles Approaching	CVM	**8.10**	58.42	0.13	0.17	**1.69**	**3.50**
	CVM+DOMap	8.57	**26.47**	**0.32**	**0.11**	5.15	5.42
	DCVM+DOMap	8.33	28.20	0.29	0.12	3.70	4.64
Multiple Obstacles	DCVM+DOMap	**12.58**	**41.37**	**0.30**	**0.12**	**6.26**	**7.95**
Crossroad	CVM+DOMap	**8.17**	**27.50**	**0.29**	**0.14**	**4.00**	**6.14**
	DCVM+DOMap	10.81	44.42	0.24	0.18	4.14	6.57
Lane Changing	CVM	**8.08**	**44.42**	0.18	**0.17**	**4.14**	6.60
	CVM+DOMap	26.23	97.20	**0.26**	**0.17**	4.22	7.13
	DCVM+DOMap	8.93	47.70	0.18	0.18	2.76	**4.99**

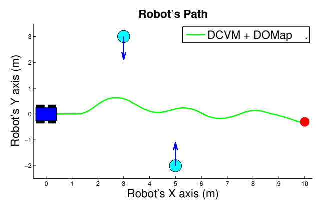
C. *Real results*

Due to the fact that DCVM has obtained the safer performance in the simulated tests, and the impossibility of repeating exactly the same scenario for all the algorithms we only test DCVM in real robots. We replicate some simulated scenarios in order to prove that the algorithm performs in a similar way with the real robot. The environment cannot be exactly replicated because unexpected obstacles which are not in the path of the robot (such as walls) can affect the avoidance maneouvre.

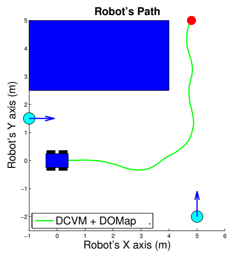
1) Two people crossing: comparing the real experiment (Fig. 3(a)) with the simulated one (Fig. 2(a)), the avoidance manoeuvre is similar. The perception stage predicts where the obstacle will be and avoids both obstacles by behind, which is a safer manoeuvre. The difference of amplitude of the curvature arcs is due to the fact that both obstacles does not move at the same velocity and the variations of it. Once both obstacles are avoided the robot tries to follow a straight path to the goal, but it begins turning left and right. This happens because the robot works at 10 Hz instead of 50 Hz of the simulation and there are some errors in the motors. Nevertheless, the robot is able to reach the goal in a safe way.

2) Corner and two people: in this scenario (Fig. 3(b)) the avoidance manoeuvre is similar to the simulated one (Fig. 2(b)). Once the obstacle that overtakes the robot is detected, it turns right to avoid it (the shortest path to the goal should be turning left) and then turns left again when it can be avoided by behind. The second obstacle is also avoided by behind and the algorithm predicts a straight path to the goal in the last part of the big obstacle avoidance that is followed by the robot with some errors.

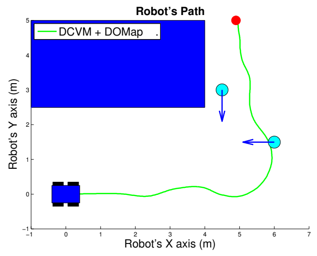
3) Corner and two people approaching the robot: in this scenario (Fig. 3(c)) the robot is able to reach the goal but the obstacle avoidance manoeuvre is slightly different to the simulated one (Fig. 2(c)). The robot begins travelling straight to avoid the big obstacle (instead of turning to the goal) and when the approaching obstacle appears it turns right to separate from it. After the crossing obstacle appears and the robot turns left to avoid it from behind. Finally the robot rounds the corner and travels to the goal in the straighter path possible due to the odometry and actuators errors.



(a) Two people crossing



(b) Corner and two people



(c) Corner and two people approaching the robot

Fig. 3. Real Experiments: Paths

V. Conclusions and Future Works

This work demonstrates the feasibility of obtaining information of variations of the environment from a range sensor and using this information to improve the autonomous capabilities of a mobile robot.

An extension of a well known algorithm, to deal with dynamic obstacles, has been developed (DCVM) and its performance has been tested in simulated and real environments, proven that it is able to avoid obstacles in a safer and better way than previous obstacle avoidance algorithms.

In the near future, we will test DCVM in more real robotic platforms, specially in the ones that have outdoor capabilities. We also want to extend the perception stage to full 3D capabilities in order to increase its robustness and improves the detection of the environment.

Also, nowadays we are generating more multimedia content of these experiments that will be available in our Robesafe YouTube Channel (*www.youtube.com/Robesafe*).

VI. Acknowledgements

This work has been funded by TIN2011-29824-C02-01 and TIN2011-29824-C02-02 (ABSYNTHE project) from Ministerio de Economía y Competitividad and CCG2013/EXP-066 (ROBOSHOP project) from University of Alcalá.

References

[1] J. M. Alonso, K. LeBlanc, M. Ocaña, and E. Ruspini, "ABSYNTHE: Abstraction, Synthesis, and Integration of Information for Human-Robot Teams," in *International Workshop on Perception in Robotics, IEEE Intelligent Vehicles Symposium*, 2012, pp. P14.1–P14.6.

[2] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, S. Thrun, *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, 2008.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, *et al.*, "Winning the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[4] Á. Llamazares, V. Ivan, E. Molinos, M. Ocaña, and S. Vijayakumar, "Dynamic Obstacle Avoidance Using Bayesian Occupancy Filter and Approximate Inference," *Sensors*, vol. 13, no. 3, pp. 2929–2944, mar 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/3/2929

[5] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 1, pp. 23–32, 1992.

[6] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 19, no. 5, pp. 1179–1187, 1989.

[7] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," *ICRA*, pp. 1572–1577, 1998.

[8] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *International Conference on Robotics and Automation*, April 1996.

[9] T.-D. Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driven Markov chain Monte Carlo." in *ICRA*. IEEE, 2009, pp. 3800–3806.

[10] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *ICRA*, 2001.

[11] T.-D. Vu, O. Aycard, and N. Appenrodt, "Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments," in *2007 IEEE Intelligent Vehicles Symposium*, Istanbul, Turquie, 2007. [Online]. Available: http://hal.inria.fr/inria-00194152

[12] J. Borenstein and Y. Koren, "The vector field histogram and fast obstacle-avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[13] J. Mínguez and L. Montano, "Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation 20(1)*, pp. 45–59, February 2004.

[14] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*. IEEE, 2008, pp. 690–695.

[15] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles." in *ICRA*. IEEE, 2007, pp. 1986–1991. [Online]. Available: http://dblp.uni-trier.de/db/conf/icra/icra2007.html

[16] N. Y. Ko and R. Simmons, "The lane-curvature method for local obstacle avoidance," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3, oct 1998, pp. 1615 –1621 vol.3.

[17] J. L. Fernández, R. Sanz, J. A. Benayas, and A. R. Diéguez, "Improving collision avoidance for mobile robots in partially known environments: the beam curvature method," *Robotics and Autonomous Systems*, vol. 46, no. 4, pp. 205–219, 2004.

[18] E. Molinos, J. Pozuelo, Á. Llamazares, M. Ocaña, and J. López, "Comparison of local obstacle avoidance algorithms," in *Computer Aided Systems Theory - EUROCAST 2013*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8112, pp. 39–46.

[19] Á. Llamazares, E. Molinos, M. Ocaña, and F. Herranz, "Integrating absynthe autonomous navigation system into ROS," *In Proc. of the ICRA 2014 workshop*, 2014.

[20] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[21] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[22] Q. Baig, M. Perrollaz, and C. Laugier, "A robust motion detection technique for dynamic environment monitoring: A framework for grid-based monitoring of the dynamic environment," *IEEE Robotics and Automation Magazine*, vol. 21, 2014.

[23] J. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.