

Realtime motion path generation using subtargets in a changing environment

Dennis Bruijnen, Jeroen van Helvoort and René van de Molengraft

Abstract—In this work an algorithm is proposed for path planning in a changing environment. The algorithm is computationally cheap and generates a sub-optimal smooth path with bounds on the allowed velocity, acceleration and jerk. It outperforms potential field algorithms regarding both convergence and optimality. Furthermore, it is able to adapt fast in a changing environment in contrast with computationally more expensive methods, such as wavefront algorithms and global optimization methods. It is applicable to both holonomic systems and a class of nonholonomic systems.

I. INTRODUCTION

Path generation is an essential item in the field of autonomous robotics. There are a lot of interesting application areas including UAVs (Unmanned Aerial Vehicles) [1], [2], autonomous underwater vehicles [3], rescue vehicles, Mars landers and robot soccer [4]–[8]. The main challenge for an autonomous robot is to move to a target position as fast as possible without colliding with other objects in a changing environment. A changing environment is defined as an environment with moving objects. Moreover, to be sure that the robot is indeed able to track the path, the path planning algorithm should at all times comply with the physical limitations of the robot, such as acceleration, velocity bounds and nonholonomic constraints.

In literature, a huge amount of path generation strategies has been proposed, both for offline and online computation. Most strategies are based on potential fields [3], [5], [9], wavefront algorithms [4] or heuristic search methods e.g. Neural Network/Genetic Algorithm methods [10]–[12]. In a changing environment, offline strategies fall short, since they require *a priori* knowledge of all environmental variables. For online strategies to be effective, the algorithms need to be computationally cheap and respond fast to changes in the environment. Global path algorithms like wavefront algorithms and heuristic search methods have the potential to approximate an optimal global path in some sense at the expense of computational costs. Because of these computational costs, the path can not be updated frequently, which renders such methods unsuitable in a changing environment. On the other hand, local path algorithms such as potential fields are less computational expensive, but they often show a lack of convergence towards the desired target.

Our new path algorithm is computationally cheap and is able to adapt rapidly in a changing environment. Moreover, it takes into account the robot’s physical limitations by limiting

the maximum velocity and acceleration of the generated path. Most existing potential field, wavefront and heuristic search algorithms lack this feature and result often in infeasible motion paths with respect to the robots physical limitations. Besides holonomic systems, the method is also applicable to a class of nonholonomic systems, i.e. systems with differential drive, as will be explained in Section IV.

First, the application area of this research and the environment are introduced. After that, the procedure to determine a subtarget is explained. Next, a method to generate a smooth path is presented. Then, several extensions to increase robustness are presented. After that, simulation results are shown and discussed and finally, some conclusions are drawn.

II. APPLICATION AREA

As a benchmark problem, a fastly changing environment is chosen, namely, the application of robot soccer. In robot soccer, path planning is crucial and, moreover, the changes in the environment are in the same order of magnitude as the movements of the robot itself. Furthermore, no *a priori* knowledge of the environment is available, i.e., it is highly unpredictable. Changes in other players’ positions and target position should be coped with flexibly. Moreover, physical limits such as the maximum acceleration and maximum velocity of the robot are an important issue.

III. FIND A SUBTARGET

Consider a typical situation that an autonomous robot might encounter during operation, as shown in Fig. 1. The objective of the robot is to move as fast as possible to the target without colliding with other objects in the field. The robot and the objects are all modeled as circles which bound the physical geometry. The global idea is to determine a subtarget where the robot can go to in a straight line after each position update of the robot and surrounding objects. This way, the robot is able to adapt to a changing environment including movement of objects, (dis)appearing objects (e.g. due to sensing limitations) and a target position change. The procedure of determining the subtarget is explained in 4 steps: first obstructor, grouping, location of subtarget and iteration procedure.

A. First obstructor

From the projection of $\vec{o}_i - \vec{o}_r$ onto $\vec{t} - \vec{o}_r$ we obtain

$$a_i = \frac{(\vec{t} - \vec{o}_r) \cdot (\vec{o}_i - \vec{o}_r)}{|\vec{t} - \vec{o}_r|} \in \mathbb{R} \quad (1)$$

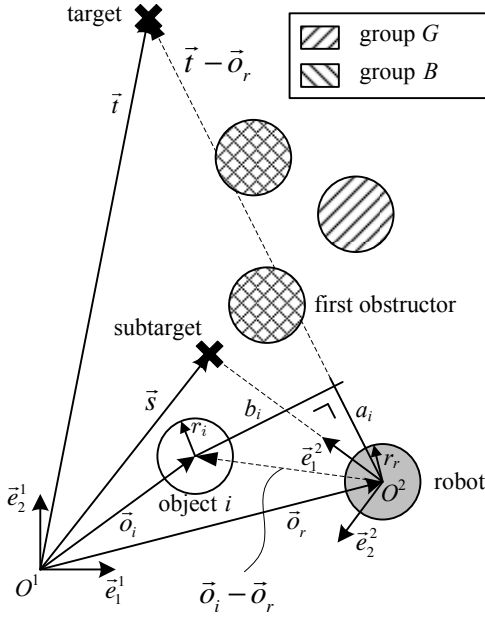


Fig. 1. An example of a situation with blocking objects between the target and the robot

$$b_i = \frac{(\vec{t} - \vec{o}_r) \times (\vec{o}_i - \vec{o}_r)}{|\vec{t} - \vec{o}_r|} \in \mathbb{R} \quad (2)$$

where (a_i, b_i) are the coordinates of object i with respect to the robot in the direction of the target. The set of all objects, represented by their indices, is defined as

$$O = \{1, \dots, n\} \in \mathbb{N}^n \quad (3)$$

with n the amount of objects in the field. By using (1), (2) and (3), the set of blocking objects B , which are all objects that the robot would hit if it would move in a straight line to the target, can be computed as follows

$$B = \{i \in O \mid 0 < a_i < |\vec{t} - \vec{o}_r| \cap |b_i| < r_i + r_r\} \in \mathbb{N}^m \quad (4)$$

where m is the amount of blocking objects and $r_i, r_r \in \mathbb{R}^+$ are the radii of object i and the robot respectively. Using (1) and (4) the first obstructing object f can be determined

$$f = \min_{a_i} B \quad (5)$$

B. Grouping

The next step is to determine the largest set G , containing f , in which for each object i holds that the gap between at least one other object $j \in G, j \neq i$ is smaller than the diameter $2r_r$ of the robot. In pseudo-code, a possible algorithm looks like

$$G = \{\text{first obstructor } f\}$$

REPEAT until the size of the group becomes constant

FOR test_object = 1 to n

IF the gap between test_object and at least one group member is smaller than the diameter of the robot

THEN $G = \{G, \text{test_object}\}$

END FOR

END REPEAT

In our approach, the group G will be avoided instead of the object f only, otherwise the robot could get stuck in the group G .

C. Location of subtarget

The side to pass the group G is determined by selecting the side with the smallest absolute value of the perpendicular distance $b_i \forall i \in G$.

$$|b|^+ = \arg \max_{|b_i + r_r|} \{i \in G \mid b_i > 0\} \quad (6a)$$

$$|b|^- = \arg \max_{|b_i - r_r|} \{i \in G \mid b_i < 0\} \quad (6b)$$

$$\text{if } |b|^+ \begin{cases} \leq |b|^- & \text{select left side of } G \\ > |b|^- & \text{select right side of } G \end{cases} \quad (6c)$$

To determine the subtarget position, the angle to pass each object in the group on the selected side,

$$\alpha_i = \arctan\left(\frac{b_i}{a_i}\right) + \text{sign}(|b|^- - |b|^+) \arcsin\left(\frac{r_r + r_i}{|\vec{o}_i - \vec{o}_r|}\right) \quad (7)$$

is computed. The object with the largest angle α , object a , determines the position of the subtarget which becomes

$$\vec{s} = \vec{o}_r + \begin{bmatrix} \cos \alpha_j & -\sin \alpha_j \\ \sin \alpha_j & \cos \alpha_j \end{bmatrix} \frac{\vec{t} - \vec{o}_r}{|\vec{t} - \vec{o}_r|} |\vec{o}_a - \vec{o}_r| \quad (8)$$

where $j = \max_{|\alpha_i|} i \in G$.

D. Iteration procedure

It is possible, that between the robot and the computed subtarget an object of another group in the field is obstructing. Previous steps should be recomputed while setting the subtarget as the target. This iteration will converge to a collision free subtarget if group G does not imprison the robot. During each iteration cycle, only objects are considered which are closer to the robot than the (sub)target. Worst-case, n iterations will be necessary, but in practical situations, 1 or 2 iterations turn out to be sufficient.

IV. FROM SUBTARGET TO PATH

The procedure, as presented in the previous section, returns a subtarget after each environment update. However, especially in a changing environment, the location of the subtarget might change fast, for instance to reflect sudden changes in the location of the various objects or in the location of the target. Although it is necessary to address these changes, it might result in a ragged path with accelerations that exceed the robot's capabilities. Moreover, for robots with nonholonomic constraints, a ragged path

would conflict with physical possibilities. These physical limitations jeopardize the ability of the robot to actually follow the prescribed path. Therefore, a smoothing algorithm is included.

A path is defined to be smooth, if the accelerations of the path are continuous and bounded functions in time. This implies that also the velocity and the position are continuous functions in time. The jerk (the time-derivative of the acceleration), however, is allowed to be a discontinuous, yet bounded, function of time. To impose these constraints on the generated path, a 'controller' is used, which controls the velocity to the desired direction by manipulating the (2-dimensional) jerk of the path.

A. Smoothing

Consider again Fig. 1. The frame $\{O^2, \underline{e}^2\}$ is fixed onto the robot position and \underline{e}_1^2 is directed towards the position of the subtarget. Let \underline{o}_r^1 be the position of the robot with respect to the fixed frame $\{O^1, \underline{e}^1\}$. Then, the velocity of the robot with respect to $\{O^1, \underline{e}^1\}$ is given by $\dot{\underline{o}}_r^1$. The velocity with respect to the frame $\{O^2, \underline{e}^2\}$ is given by

$$\dot{\underline{o}}_r^2 = \underline{A}^{21} \dot{\underline{o}}_r^1 \quad (9)$$

where

$$\underline{A}^{21} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

and θ is the counter clockwise rotation of frame $\{O^2, \underline{e}^2\}$ with respect to frame $\{O^1, \underline{e}^1\}$.

The goal is to smoothly control the velocity in the direction of the subtarget, $\dot{\underline{o}}_{r1}^2$, to a desired velocity v_{desired} , while simultaneously bringing the velocity perpendicular to the desired direction, $\dot{\underline{o}}_{r2}^2$, to zero. This is accomplished by using a controller that can only influence the jerk of the desired movement. Since the location of the subtarget is updated regularly, it suffices to control the direction of travel, instead of the path. This behavior is more natural, and results in a smooth, short path. Furthermore, as a direct result, the desired direction of travel is a smooth function of time. For nonholonomic systems with e.g. differential drive, the direction of travel is directly linked to its orientation. Therefore, if the desired direction of travel is continuous, the orientation is continuous as well, ergo, the path is feasible.

Assume that the path is to be generated using a fixed sampling rate of 1 kHz, which might for instance be the sampling rate of the motion controllers which control the robot. The aim is to build a controlled system

$$\ddot{\underline{o}}_{r1}^2 = C(z)(v_{\text{desired}} - \dot{\underline{o}}_{r1}^2) \quad (10a)$$

$$\ddot{\underline{o}}_{r2}^2 = -C(z)\dot{\underline{o}}_{r2}^2 \quad (10b)$$

such that $\dot{\underline{o}}_{r1}^2$ converges to v_{desired} and $\dot{\underline{o}}_{r2}^2$ converges to 0. Note that the transition $P(z)$ from jerk $\ddot{\underline{o}}_{r1}^2$ to velocity $\dot{\underline{o}}_{r1}^2$, which in continuous time is given by a double integrator, is

in discrete time given by (with a sampling frequency of 1 kHz):

$$P(z) = \frac{(0.001)^2}{2} \frac{z^{-1} + z^{-2}}{1 - 2z^{-1} + z^{-2}} \quad (11)$$

A discrete time controller that accomplishes the convergence of the velocities, is given by

$$C(z) = \frac{520z^{-1} - 518.6z^{-2}}{1 - 1.726z^{-1} + 0.7545z^{-2}} \quad (12)$$

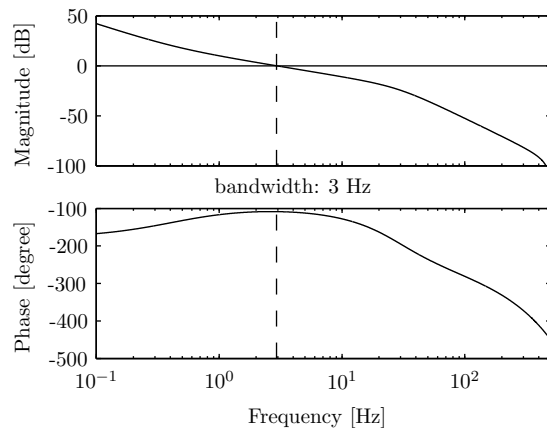


Fig. 2. Bode plot of the open loop $H_{ol} = CP$.

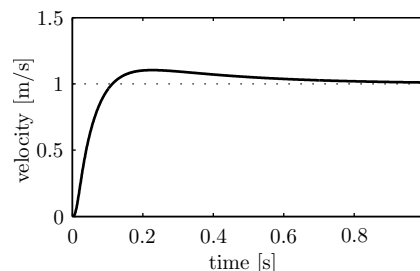


Fig. 3. Stepresponse of the controlled system $H_{cl} = \frac{CP}{1+CP}$.

In Fig. 2, a bode plot of the open loop $H_{ol} = CP$ is shown. It can be seen that the bandwidth of the controlled system is about 3 Hz. The controlled system is stable, with a gain margin of 21 dB and a phase margin of 72° . In Fig. 3, the stepresponse of the closed loop system $H_{cl} = \frac{CP}{1+CP}$ is shown. It can be seen that indeed the controlled system converges to the desired velocity, with a small overshoot and after a time span of 1 s which is reasonable regarding the application area.

B. Saturation

From the jerk, as derived in (10), the update of the acceleration and velocity profile is computed. However, an acceleration and velocity can result, that exceed the capabilities of the robot at hand. Therefore, saturations are imposed on the acceleration and velocity.

Let the maximum velocity of the robot be given by v_{max} and the maximum acceleration by a_{max} . Then, if the bound of the

maximum acceleration is exceeded by the controller output, the acceleration is reset to its maximum value

$$a_{\text{sat}} = \text{sat}(a, a_{\text{max}}) \quad (13)$$

where

$$\text{sat}(x, x_{\text{max}}) = \begin{cases} -x_{\text{max}} & \text{if } x < -x_{\text{max}} \\ x & \text{if } -x_{\text{max}} \leq x \leq x_{\text{max}} \\ x_{\text{max}} & \text{if } x > x_{\text{max}} \end{cases}$$

The acceleration a is considered both in the direction of the subtarget, \ddot{o}_{r1}^2 , and perpendicular to that direction, \ddot{o}_{r2}^2 . The velocity is treated similarly:

$$v_{\text{sat}} = \text{sat}(v, v_{\text{max}}) \quad (14)$$

The velocity v is also considered both in the direction of the subtarget, \dot{o}_{r1}^2 , and perpendicular to that direction, \dot{o}_{r2}^2 . If indeed the velocity is saturated, the acceleration and jerk has to be computed which would have resulted in this velocity. It should be noted, that the adjusted acceleration is always closer to zero than the original acceleration, hence, this procedure will always provide a feasible solution. Of course, a large jerk might result from this nonlinear procedure. The maximum jerk is given by

$$\max(j) = \frac{a_{\text{max}}}{dT} \quad (15)$$

with dT the sampling interval. Hence, although possibly large, the maximum jerk is still bounded. Simulations show that the controller (12) is robust to sudden changes in the target, changes in the desired velocity and disturbances caused by the saturation.

C. Desired velocity

As input for the controlled system, a desired velocity is given. Typically, for a straight trajectory, the desired velocity will be close to the maximum allowed velocity of the robot in order to reach the target as fast as possible. However, when the robot has to make a sharp turn, then the desired velocity is lower due to the acceleration limit. The distance of the robot to the subtarget is a qualitative measure of how close an object is.

Consider a holonomic robot, which is driven by 'omni-wheels' [13]. These omniwheels are frequently applied in the robot soccer competition RoboCup. The robot is a holonomic system due to the omniwheels and every change in movement requires power of the drives. This is in contrast with non-holonomic system, e.g. a car, which can change direction using steering wheels without using power of the engine. Suppose that the robot has to make a turn with turning radius r , then the desired velocity follows from the maximum acceleration of the robot induced by its omniwheel drives. For a circular movement, this maximum acceleration is equal to the centrifugal acceleration

$$a_{\text{centrif.}} = \frac{v^2}{r} \quad (16)$$

Considering (16), an appropriate choice for v_{desired} is

$$v_{\text{desired}} = \text{sat}\left(\sqrt{|\vec{s} - \vec{o}_r| a_{\text{max}}}, v_{\text{max}}\right) \quad (17)$$

which is the maximum possible velocity to circle around an object where its radius plus the radius of the robot is equal to $|\vec{s} - \vec{o}_r|$. Before reaching the object, the robot will already reduce its velocity to avoid large overshoot of the path smoothening algorithm during the turn. Furthermore, if the subtarget is equal to the target, the desired velocity will reduce towards zero when the robot has reached the target location.

Note that only the robot's acceleration is considered. An extension would be to consider the acceleration per omnivheel but this goes beyond the scope of this work.

V. EXTENSIONS TO INCREASE ROBUSTNESS

To increase the robustness of the proposed algorithm in a changing environment, several additional adjustments can be made. Especially, if the algorithm is to operate in an environment with partial and noisy measurements of the surroundings or within a fast changing environment. The determination of a new subtarget can be implemented with every new measurement of the environment. The extensions, as proposed in this section, can be integrated in the main procedure and therefore they are also evaluated every step.

As a first extension to the main procedure, a safety margin around the obstacles can be incorporated. The entire region of the obstacle plus a safety margin is then regarded as the obstacle to be avoided, and the subtarget is located outside this region. With this extension, a collision is avoided even with noisy position measurements of the obstacles or in the presence of slowly moving obstacles. If the path accidentally happens to enter the safety margin region, the subtarget is instantaneously relocated outside the safety margin and the path is sort of "repelled" by the obstacle.

As an additional extension to the safety margin, the velocity of the obstacles can be estimated. The safety margin can then be resized such that in the direction of travel of the obstacle, the path has to reckon with a larger distance from that obstacle.

Another extension is to alter the relation between the distance to the subtarget and the desired velocity (17). A quicker decrease of the the desired velocity results in an increased manoeuvrability in the presence of objects, such that the change of a collision is reduced.

All these extensions can be interpreted as the aggressiveness of the robot. The closer and the faster it passes objects, the more risk is taken regarding collisions with objects.

VI. SIMULATIONS

A. Parameters

In Fig. 4, Fig. 5 and Fig. 6 simulations are shown of the path generation algorithm presented before. The robot and the objects have a radius of 0.3 m. The maximum velocity is 2 m/s and the maximum acceleration is 2.5 m/s². The update rate of the subtarget is 10 Hz and the trajectory is generated

at a rate of 1000 Hz. The dots indicating the velocity of the robot have an interval of 0.2 s. For clarity of the simulation, all objects are standing still during the simulations. However, it will be clear from the previous sections, that changing environments are treated likewise.

The algorithm is implemented in Matlab. Interpreted in script language, the computational time of the subtarget is about 3 ms per sample and the computational time of the smooth trajectory is about 0.3 ms per sample, on a regular pentium 4 processor. The calculation time is then 30% of 1 ms so it is suitable for real time purposes. If the algorithm is implemented more efficiently, for example in C, then this calculation time will decrease.

B. Test situations

The situation shown in Fig. 4 shows how the robot passes 4 groups sequentially. It is clear to see that if the robot comes close to objects, it will decrease its velocity such that it is able to change direction fast enough. If the velocity would not decrease, a large overshoot of the trajectory would occur, possibly resulting in a collision.

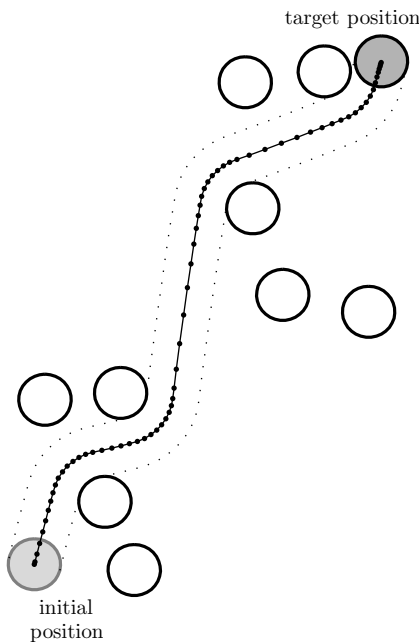


Fig. 4. Simulation with 9 objects scattered across the field.

The situation shown in Fig. 5 shows two groups both in the shape of a cup. A potential field algorithm would most likely get stuck in a cup because the combination of the objects close to each other result in local minima. The proposed algorithm finds a trajectory which is smooth and sub-optimal. only sub-optimality is reached, since if it would go around the cup on the right side, it would arrive faster at the target. In a changing environment such a little performance increase is much less important than the ability to respond fast to sudden changes of the target location and the environment.

In Fig. 6 the robot is partly surrounded by a group of objects. For this situation, a potential field algorithm has

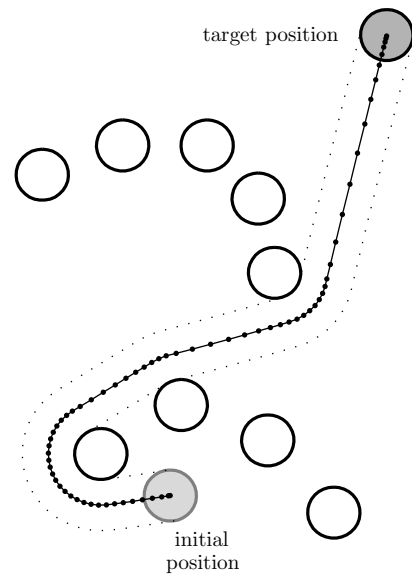


Fig. 5. Simulation with 9 objects forming two groups.

difficulty to find a way to the target, because the robot is located in a local minimum, see Fig. 7. Also in this case, the algorithm generates a near optimal trajectory.

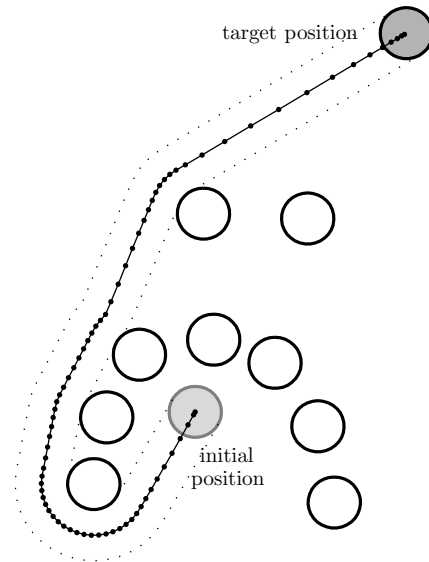


Fig. 6. Simulation with 9 objects where one group encircles the robot.

In Fig. 8, the solution of a wavefront algorithm is shown. The computation time of this path was about 0.2 s. The computed path approximates the optimal path in the sense of shortest distance. However, it is not yet a smooth path satisfying physical limitations of the robot and a choice for the grid size is to be made. So an additional algorithm is required to make a feasible path regarding physical limitations of the robot. This will further increase the computational time and furthermore, if the field of interest is larger while keeping the same resolution of the wavefront algorithm, the computation time will increase proportional to the surface area.

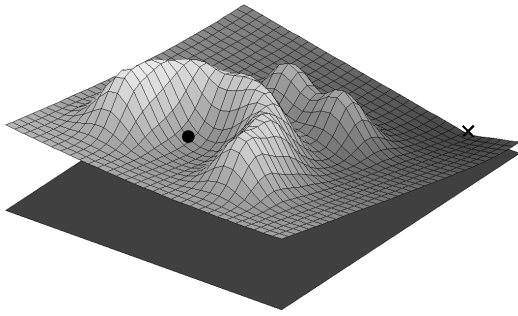


Fig. 7. Potential field for the situation as shown in Fig. 6. The dot is the robot position and the cross is the target position. The robot is located in a local minimum so no path to the target is found.

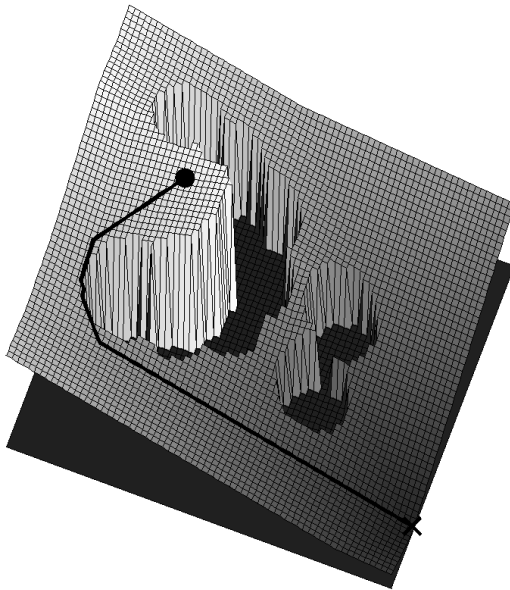


Fig. 8. Wavefront algorithm solution for a situation as shown in Fig. 6. The dot is the robot position and the cross is the target position. The height is equal to the amount of steps needed to arrive at the target. The trajectory may not cross the gaps that are present because objects are located there.

One situation is known where the algorithm does not come up with a trajectory converging to the target position. This situation occurs when a group of objects encircles the target position. For example, if the initial position and target position in Fig. 6 would be swapped. The robot will wait close to the target position with the objects in between, until the group splits up and/or the target position is not encircled anymore. A possible way to overcome this problem is to place the target outside the group in a collision free straight line. Actually, this is equal to using the proposed algorithm the other way around, so try to move the target to the robot.

VII. CONCLUSIONS

The proposed algorithm has shown to generate near optimal smooth paths in difficult situations. It outperforms potential field algorithms with respect to global convergence to the target. Although convergence is not guaranteed for arbitrary situations, it converges in most practical situations and it is able to adapt fast in a changing environment due to the low computational costs, this in contrast with computational expensive global methods like heuristic search algorithms and wavefront algorithms. The generated path is smooth and complies with the robot's capabilities.

REFERENCES

- [1] J. Lee, R. Huang, A. Vaughn, X. Xiao, J. Hedrick, M. Zennaro, and R. Sengupta, "Strategies of path-planning for a uav to track a ground vehicle," in *AINS 2003*, Menlo Park, CA, USA, June 2003.
- [2] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of a uav through uncertain environments," in *Digital Avionics Systems Conference*, 2002.
- [3] M. Eichhorn, "A reactive obstacle avoidance system for an autonomous underwater vehicle," in *IFAC World Congress*. Prague, Czech Republic: IFAC, July 2005, p. CDrom.
- [4] C. Behring, M. Branco, M. Castro, and J. Moreno, "An algorithm for robot path planning with cellular automata," in *Proc. Conf. Cellular Automata for Research and Industry*, 2000.
- [5] P. Vallejos, J. R. del Solar, and A. Duvost, "Cooperative strategy using dynamic role assignment and potential fields path planning," in *Proc. IEEE Latin Amer. Robot. Symp.*, Mexico City, Mexico, October 2004, pp. 48 – 53.
- [6] K. Han and M. Veloso, "Reactive visual control of multiple non-holonomic robotic agents," in *Proceedings of the 1998 IEEE Int. Conf. on Robotics & Automation*, Leuven, Belgium, 1998, pp. 3510–3515.
- [7] R. Emery and T. Balch, "Behavior-based control of a non-holonomic robot in pushing tasks," in *Proceedings of the 2001 IEEE Int. Conf. on Robotics & Automation*, Seoul, Korea, 2001, pp. 2381–2388.
- [8] L. Lei, X. Rui-zhi, W. Ke, and W. Wei, "Key techniques in robocup middle-sized soccer research," in *Proceedings of the 2004 IEEE Int. Conf. on Robotics & Biomimetics*, Shenyang, China, 2004, p. 529–534.
- [9] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 501–518, 1992.
- [10] D. Kang, H. Hashimoto, and F. Harashima, "Path generation for mobile robot navigation using genetic algorithm," in *Int. Conf. Indust. Electr. Control and Instrumentation*, Orlando, FL, USA, November 1995, pp. 167–172.
- [11] D. Lebedev, J. Steil, and H. Ritter, "A neural network model that calculates dynamic distance transform for path planning and exploration in a changing environment," in *Proc. Int. Conf. Robotics & Autom.*, Taipei, Taiwan, September 2003, pp. 4209–4214.
- [12] H. Sfeir, "A neural-network-based path generation technique for mobile robots," in *Proc. Int. Conf. Mech.*, June 2004, pp. 176–181.
- [13] M. Ashmore and N. Barnes, "Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line," in *AI '02: Proceedings of the 15th Australian Joint Conference on Artificial Intelligence*. London, UK: Springer-Verlag, 2002, pp. 225–236.