IMPROVED PATH PLANNING IN HIGHLY DYNAMIC ENVIRONMENTS BASED ON TIME VARIANT POTENTIAL FIELDS

Patrick Heinemann — **University of Tübingen, WSI-RA** — **Germany**

Hannes Becker — **University of Tübingen, WSI-RA** — **Germany**

Andreas Zell — **University of Tübingen, WSI-RA** — **Germany**

Speaker: Patrick Heinemann, University of Tübingen, WSI-RA, Sand 1, 72076 Tübingen, 07071 / 29 - 78989, patrick.heinemann@uni-tuebingen.de

Topic: Mobile robot navigation
Keywords: potential field, path planning, obstacle avoidance

Introduction

Planning collision-free paths is one of the basic skills for a mobile robot performing a goal-oriented task. Especially in highly dynamic environments such as robot soccer there is a need for smooth navigation avoiding the cooperating and competing players. Today, robots in RoboCup [5] are moving at speeds of up to 3 m/s. Navigation thus requires real time path planning considering the movement of the obstacles. Although there are publications on path planning in the robot soccer domain (e.g. [1],[3],[7]), the presented approaches do not incorporate the speed of the obstacles into the planned paths. The majority of the teams in the RoboCup middle size league competition of 2005 are actually completely unaware of the speed of the other robots [6]. To the best knowledge of the authors no team uses information on moving obstacles in their path planning algorithms.
This paper presents a new method of path planning for the RoboCup domain that extends the approach introduced by Weigel *et al.* [7] to time variant potential fields that consider the movement of the obstacles over time. Using this method results in smoother paths and less collisions for several scenarios, that frequently occur in RoboCup. The remainder of the paper explains the method in detail and presents first results.

The Path Planning Algorithm

As our method is an improvement of the path planner of Weigel *et al.* [7], a brief overview of this planner is given first.
The approach of Weigel *et al.* is one of the most efficient approaches for path planning in the RoboCup domain. It uses a combination of a potential field and grid-based path planning to plan out of local minima.

Potential Field

The idea of using a potential field for mobile robot movement was first presented by Khatib [4]. The robot is exposed to an attractive potential towards the target and repulsive potentials away from obstacles and moves according to an artificial force computed as the negative gradient of this potential field. The path the robot follows in such a potential field is comparable to the path an electric particle would follow in a potential field produced by other attractive or repulsive electric charges. The robot is always heading towards the next position with lower potential, thus automatically avoiding the high potential of the obstacles.
The potential field at position $x = (x_1, x_2)$ is composed of an attractive potential well $p_{att}(x)$ at the target $g = (g_1, g_2)$ and repulsive potential barriers $p_{rep,\,i}(x)$ at the position of obstacles $o_i = (o_{1,i}, o_{2,i})$ and the walls $w_j = (0, w_{2,j})$ for walls in x-direction, $w_j = (w_{1,j}, 0)$ for walls in y-direction. The attractive potential is modelled as a conic potential well

$$p_{att}(x) = \rho_{att} \left\| d_{att}(x) \right\|$$

with $d_{att}(x) = (x - g)$ that results in a movement towards the target at constant velocity, as the artificial force

$$f_{att}(x) = -\nabla p_{att}(x) = -\frac{\rho_{att}}{\left\| d_{att}(x) \right\|} d_{att}(x)$$

linearly decreases towards the goal. The repulsive potentials are modelled as potential barriers with a potential inversely proportional to the squared distance to the obstacle or wall

$$p_{obs,i}(x) = \begin{cases} \rho_{obs} & if \quad \mu_{obs}^2 \geq \left\| d_{obs,i}(x) \right\|^2 \\ \rho_{obs}\kappa_{obs}\left(\frac{1}{\left\| d_{obs,i}(x) \right\|^2} - \frac{1}{\mathrm{M}_{obs}^2} \right) & if \quad \mathrm{M}_{obs}^2 > \left\| d_{obs,i}(x) \right\|^2 > \mu_{obs}^2 \\ 0 & if \quad \left\| d_{obs,i}(x) \right\|^2 \geq \mathrm{M}_{obs}^2 \end{cases}$$

for the obstacles, with vectors $d_{obs,i}(x) = (x - o_i)$. Here $\kappa_{obs} = \frac{\mu_{obs}^2 \mathrm{M}_{obs}^2}{\mathrm{M}_{obs}^2 - \mu_{obs}^2}$ is a normalization factor used to make $p_{obs,i}(x)$ continuous at $\left\| d_{obs,i}(x) \right\|^2 = \mathrm{M}_{obs}^2$ and $\left\| d_{obs,i}(x) \right\|^2 = \mu_{obs}^2$. As the potential $p_{obs,i}(x)$ would infinitely increase towards the obstacle, a minimum distance μ_{obs} is introduced. For positions that are nearer to the obstacle than this distance, the potential is at the maximum possible value for obstacles ρ_{obs}. Since RoboCup robots are limited to a size of 50x50cm they are assumed to be round, which simplifies the computation of the potential field of obstacles. Thus, the minimum distance is composed of the obstacle radius r_{obs}, the robot's radius r_{rob} and a security distance ε

$$\mu_{obs} = r_{obs} + r_{rob} + \varepsilon$$

As obstacles that are far away from the robot should not influence its path, a maximum distance M_{obs} is used to cut off the influence of the obstacles. This also reduces the amount of local minima resulting from the superposition of the fields of many obstacles. The artificial force for the obstacles is computed as

$$f_{obs,i} = \begin{cases} -\nabla p_{obs,i}(x) = 2\frac{\rho_{obs}\kappa_{obs}\mu_{obs}^2}{\left\| d_{obs,i}(x) \right\|^4} d_{obs,i}(x) & if \quad \mathrm{M}_{obs}^2 > \left\| d_{obs,i}(x) \right\|^2 > \mu_{obs}^2 \\ \vec{0} & if \quad \mu_{obs}^2 > \left\| d_{obs,i}(x) \right\|^2 \vee \left\| d_{obs,i}(x) \right\|^2 > \mathrm{M}_{obs}^2 \end{cases}$$

As in robot soccer and other applications of robotics the space for driving with the robot is limited, walls and artificial limitations of this space are modelled as potential barriers, as well. Their computation is similar to that of the obstacles

$$p_{wall,j}(x) = \begin{cases} \rho_{wall} & if \quad \mu_{wall}^2 \geq \left\| d_{wall,j}(x) \right\|^2 \\ \rho_{wall}\kappa_{wall}\left(\frac{1}{\left\| d_{wall,j}(x) \right\|^2} - \frac{1}{\mathrm{M}_{wall}^2} \right) & if \quad \mathrm{M}_{wall}^2 > \left\| d_{wall,j}(x) \right\|^2 > \mu_{wall}^2 \\ 0 & if \quad \left\| d_{wall,j}(x) \right\|^2 \geq \mathrm{M}_{wall}^2 \end{cases}$$

with $d_{wall,j}(x) = \begin{pmatrix} 0 \\ x_2 - w_{2,j} \end{pmatrix}$ for walls in x-direction or $d_{wall,j}(x) = \begin{pmatrix} x_1 - w_{1,j} \\ 0 \end{pmatrix}$ for walls in y-direction. Again $\kappa_{wall} = \frac{\mu_{wall}^2 \mathrm{M}_{wall}^2}{\mathrm{M}_{wall}^2 - \mu_{wall}^2}$ is a normalization factor used to make $p_{wall,j}(x)$ continuous at $\left\| d_{wall,j}(x) \right\|^2 = \mathrm{M}_{wall}^2$ and $\left\| d_{wall,j}(x) \right\|^2 = \mu_{wall}^2$. Here, the minimum distance $\mu_{wall} = r_{rob} + \varepsilon$ only consists of the robot's radius r_{rob} and a security distance ε. The artificial force for the walls is computed as

$$f_{wall,j} = \begin{cases} -\nabla p_{wall,j}(x) = 2\frac{\rho_{wall}\kappa_{wall}\mu_{wall}^2}{\left\|d_{wall,j}(x)\right\|^4} d_{wall,j}(x) & if \quad \mathrm{M}_{wall}^2 > \left\|d_{wall,j}(x)\right\|^2 > \mu_{wall}^2 \\ \vec{0} & if \quad \mu_{wall}^2 > \left\|d_{wall,j}(x)\right\|^2 \vee \left\|d_{wall,j}(x)\right\|^2 > \mathrm{M}_{wall}^2 \end{cases}$$

As the attractive and repulsive potentials are independent of each other, the final potential field that attracts the robot towards the target while keeping it away from the obstacles can be computed as a superposition of the singular potential fields

$$P(x) = p_{att}(x) + \sum_i p_{obs,i}(x) + \sum_j p_{wall,j}(x)$$

and the resulting force is given by

$$F(x) = -\nabla P(x) = -\nabla p_{att}(x) - \sum_i \nabla p_{obs,i}(x) + \sum_j \nabla p_{wall,j}(x) = f_{att}(x) + \sum_i f_{obs,i} + \sum_j f_{wall,j}$$

Although the potential field of obstacles and walls is not continuously differentiable at the minimum and maximum distances, this does not affect the presented approach, as in the final algorithm the gradient will be approximated.
Please note, that for the computation of the potentials only the attractive potential uses the norm of the distance vectors. For the obstacles and walls the squared norm is used, avoiding the extraction of the root which is computationally expensive. Especially if many obstacles and walls are present, this results in an enormous performance benefit.

Grid Step Planner

Moving the robot directly into the direction of the negative gradient $F(x) = -\nabla P(x)$ takes the risk of ending up in a local minimum. A mechanism is needed to detect that the robot is trapped and to generate a reasonable movement out of such local minima. Therefore, the state space is subdivided into an equally spaced grid with α being the size of a grid cell and a complete set of waypoints is planned from the current position of the robot towards the target. When planning the path, the next waypoint is the next grid cell following the direction of the gradient. The error made in choosing a discrete grid cell in the direction of the gradient is accumulated and affects the grid cell chosen in the next step similarly to Bresenham's line-drawing algorithm [2]. The gradient is approximated by evaluating the potential field locally using the difference quotient which reduces the computational costs for evalutating the whole potential field

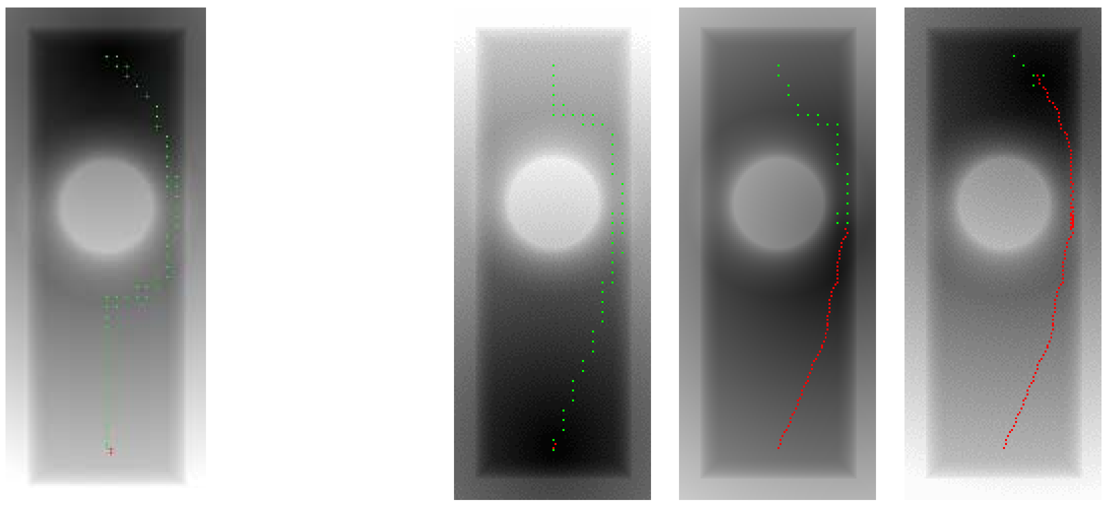
$$g(u,v) = \frac{1}{2\alpha}\big(P(u+1,v) - P(u-1,v), P(u,v+1) - P(u,v-1)\big)$$

If the next grid cell is already contained in the path, the algorithm has detected a local minimum. Then a best-first search begins for the adjacent grid cells that ends if either a cell is found with a potential lower than the potential of the cell where the search started, or the target cell is found.
As the set of grid cells generated as path by the grid step planning algorithm is very square-edged due to the discretization on the grid, the vector of movement followed by the robot is calculated as average over the first m waypoints. If $w_1, \ldots, w_n$ denote the waypoints of the planned path and $s = (s_1, s_2)$ denotes the current position of the robot, the robot moves into the direction

$$\lambda = \frac{1}{m}\sum_{i=1}^{m} w_i - s$$

Backwards Planning

The main advantage of the approach of Weigel *et al.*, however, is the idea to plan the path backwards from the target to the robot to avoid heading directly into an obstacle and then following a curve around it. If the path planning is reversed, the robot directly enters a trajectory that leads around the obstacle. The left image in Figure 1 shows the potential field for a typical situation. It ranges from a low potential (black) at the target located in the upper middle to a high potential at the location of an obstacle in the middle. Furthermore, a path generated by planning from the start in the lower middle towards the target is shown. The path directly approaches the obstacle and gets very near until it starts to surround the obstacle in a very close curve. After passing the obstacle, the robot moves to the goal in a smooth curve (cf. left side of Figure 1). Although our implementation of the path planning algorithm is able to constantly replan the paths in every cycle of the robot control algorithm, the path that the robot finally takes when moving is identical to the path planned in the first cycle as the potential field does not change.
Instead, if the path is planned backwards by changing the target and the start location and is then followed by the robot in reverse order, the robot finally takes a smooth curve around the obstacle. In the first cycle shown on the right side of Figure 1, the planned path seems to be identical to the path planned forwards, taking into account that the start and the target location were exchanged. However, if the robot follows the path in reverse order from its real start location in the



The potential field for a typical situation. Lower potentials are shown in darker color than higher potentials. The robot is located in the lower half and plans a path (green waypoints) towards the target in the upper half around an obstacle in the middle.

The same situation, but the path is planned backwards from the target to the robot. By constantly replanning the path after moving to the next waypoint the robot finally follows a smooth trajectory (red points) around the stationary obstacle.

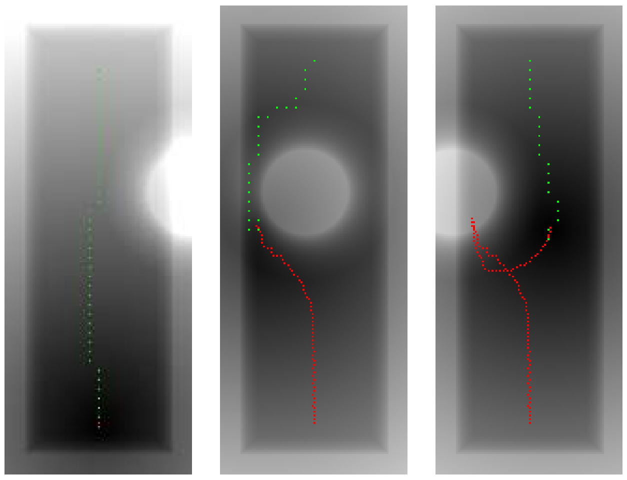
Figure 1

lower middle towards the target, it directly enters a smooth curve around the obstacle. The constant replanning of the path in each cycle changes the path to a smooth curve even behind the obstacle as the potential well follows the robot position when moving (cf. the images for cycle 60 and cycle 120 on the right side in Figure 1).
Although this idea of Weigel *et al.* to plan the path backwards from the target to the goal results in very smooth and efficient paths for slowly moving obstacles, there are many situations where the planned path is inefficient because of the unconsidered movement of faster moving obstacles.

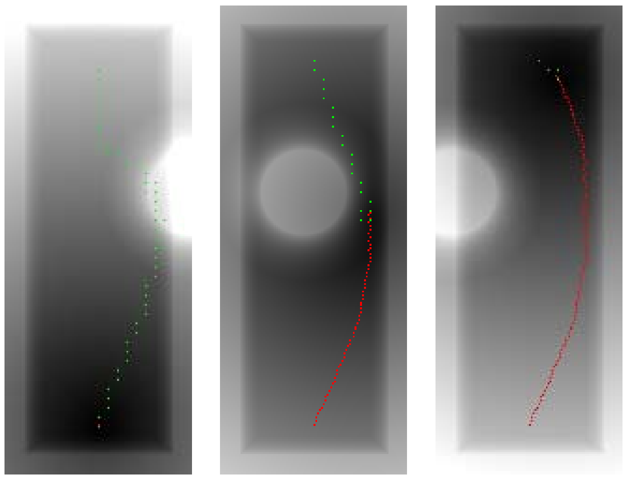
Time Variant Potential Field
To overcome this shortcoming of the approach of Weigel *et al.* the improved path planning method presented in this paper extends the approach to cope with moving obstacles. For that, the position of the obstacles is no longer assumed to be static for the whole planning process. Instead, whenever the next grid cell is reached in the planning process, the obstacles are moved to a new position

$$o_i(t+\tau) = (o_{1,i}(t+\tau), o_{2,i}(t+\tau)) = (o_{1,i}(t) + \tau v_{1,i}, o_{2,i}(t) + \tau v_{2,i})$$

with $o_i(0) = o_i$, $v_i = (v_{1,i}, v_{2,i})$ being the observed velocity of obstacle o_i at $t = 0$ and τ being the time the robot needs to reach the next grid cell, which only depends on the maximum speed v_{max} of the robot as the robot should always move with this speed. This process results in a *time variant potential field* reflecting the changed obstacle situation in each planning step. Planning in this time variant potential field avoids paths that interfere with the predicted trajectories of the moving obstacles while a conventional path planner based on the approach of Weigel *et al.* fails to plan an efficient as shown in Figure 2.
As the original method included backwards planning from the target to the starting point of the robot, the proposed algorithm must know the time T the robot needs to reach the target a priori to predict the obstacle positions. However, T depends on the planned path, which is unknown before the planning. To overcome this problem two proposals are made and compared in this paper. Firstly, the *Euclidian distance estimator* uses the Euclidian distance from the start to the target point to calculate an a priori estimation of the time needed to reach the target



Again, the robot is located in the lower middle and plans a path towards the target in the upper middle. In this scenario, a moving obstacle crosses the path from right to left. The conventional path planner assumes the obstacle to be static and tries to get around the left side of the obstacle (cycle 60). As the object moves, the passage between the obstacle and the wall gets too narrow and the path has to be planned out of the emerging local minimum and finally surrounds the obstacle on the right.



The path planner using the time variant potential field presented in this paper incorporates the velocity of the obstacle and predicts the position over time. Thus, for the same scenario the path is planned around the right side of the obstacle right from the beginning, resulting in a smooth and efficient path. While the conventional path planner is still avoiding the obstacle in cycle 100, the new path planner already reaches the target at the same time.

Figure 2

$$\hat{T}_{e,0} = \frac{\|g - s\|}{v_{max}}$$

where s is the current position of the robot and g is the target position, as defined earlier. This time is an underestimation of the time needed to follow the resulting path. Starting with this a priori estimation, an iterative process is started to estimate T. Denoting the a priori estimation in iteration k with $\hat{T}_{e,k}$ and the a posteriori time as

$$\tilde{T}_{e,k} = \frac{L_{b,k}}{v_{max}}$$

with $L_{b,k}$ being the length of the path planned backwards using $\hat{T}_{e,k}$ as estimation, we compute the a priori estimation for the next iteration by

$$\hat{T}_{e,k+1} = \hat{T}_{e,k} + \gamma(\tilde{T}_{e,k} - \hat{T}_{e,k})$$

Thus, in each iteration we add a fraction of the estimation error between a priori estimation and a posteriori time to the a priori estimation to approach the real value of T. The iterative process stops, if either the estimation error is below a threshold

$$\tilde{T}_{e,k} - \hat{T}_{e,k} \le \xi$$

or if a maximum number of iterations k_{max} is reached, to stop the process in cases where the estimation oscillates.

Secondly, a different a priori estimation of the time is generated by planning the path in forward direction and calculating the time needed to follow this path

$$\hat{T}_f = \frac{L_f}{v_{max}}$$

where L_f is the length of the path planned in forward direction. This estimator is called the *forward planning estimator*. The subsequent iterative process is identical to that used for the Euclidian distance estimator.

The predicted position of the moving obstacles in each time step of the path planner extremely depends on the estimation of T. An obstacle moving at 2m/s changes its position by up to 40cm if the time estimation varies by 0.2s. While the algorithm might be able to plan a smooth curve behind the obstacle in one iteration, the same obstacle can force the planning into a local minimum in the next iteration, now being predicted to a different position. Thus, small variations in the a priori estimation $\hat{T}$ can result in extreme variations of $\tilde{T}$. Therefore, the parameters γ and ξ have a strong influence on the performance of the algorithm which is investigated thoroughly in the next section.

Results

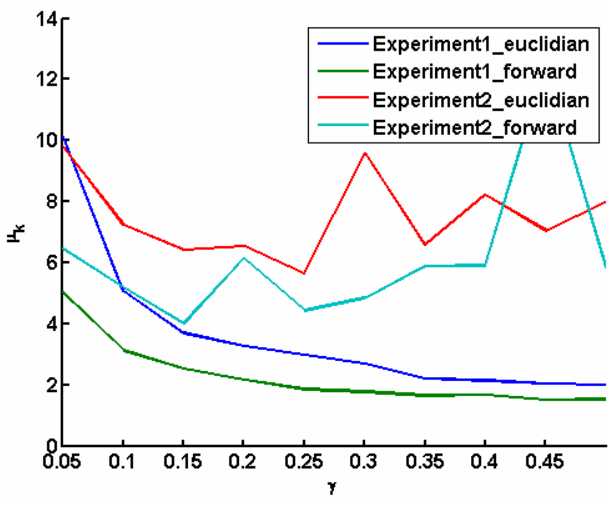
In this section, first results of the new path planning algorithm based on time variant potential fields are presented. Initially the influence of the factor γ and the threshold ξ on the number of iterations is tested. Then, the paths planned by the method of Weigel *et al.* are compared to those planned with the proposed algorithm for two scenarios with moving obstacles. Finally, the length of the paths planned by these two methods is compared for several random scenarios, as the length of the path is a measure of the efficiency of the planned path. For all experiments the parameters given in Table 1 were used.

Parameters used for the experiments										
ρ_{att}	ρ_{obs}	ρ_{wall}	r_{obs}	r_{rob}	ε	M_{obj}	M_{wall}	α	m	v_{max}
$1 \cdot 10^6$	$4 \cdot 10^5$	$2 \cdot 10^5$	25cm	20cm	5cm	50cm	20cm	10cm	5	2m/s

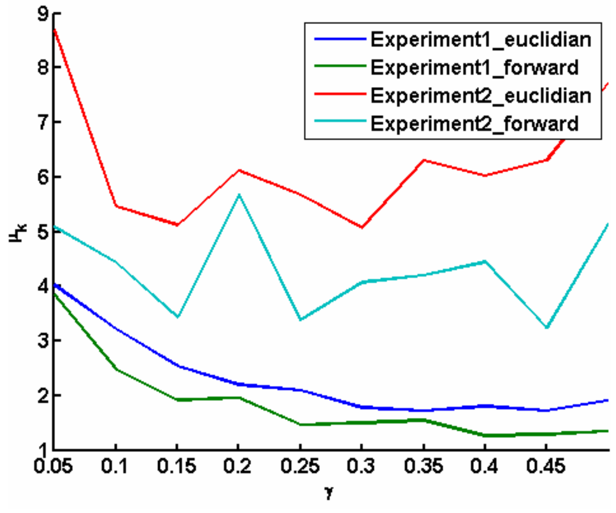
Table 1

To investigate the influence of the factor γ and the threshold ξ on the number of iterations made in each cycle of the planning algorithm, two different experiments were carried out for different values of γ and ξ. The first Experiment was the simple scenario that was used to create the snapshots in Figure 2, while the second Experiment was a more complex scenario including three moving obstacles on a field of $12 \text{x} 8 \text{m}^2$ size where a path was planned over more than 12m from the lower right corner of the field to the upper left. One of the obstacles starts on the left side of the robot and moves into nearly the same direction crossing the diagonal at the middle of the field, thus extremely interfering with the direct path of the robot. Another obstacle moves at a low speed from near the target point into the direction of the robot, so that the robot has to surround the obstacle at the end of the path. The third obstacle moves in the upper half from right to left obstructing the pathway around the second obstacle on the right side (cf. Figure 4). Four different values for the threshold ξ controlling the difference between a priori estimation and a posteriori time were used ranging from 0.1s to 0.25s. The maximum error in the position estimation of an obstacle for these values ranges from 0.2m to 0.5m for obstacles moving at 2m/s. However, this maximum position error affects only obstacles near the target, as the difference in the predicted time is very low at the beginning of the path. Thus, the direction of movement averaged over the first waypoints does not contain this maximum error, and the robot still avoids bumping into the obstacles. For each value of ξ 10 different values of the factor γ controlling the amount of the estimation error that is added to the estimation for the next iteration were tested ranging from 0.05 to 0.5. In addition, all runs were carried out once for the Euclidian distance estimator and for the forward planning estimator. Finally, for these runs the maximum number of iterations was raised to $k_{max} = 50$.

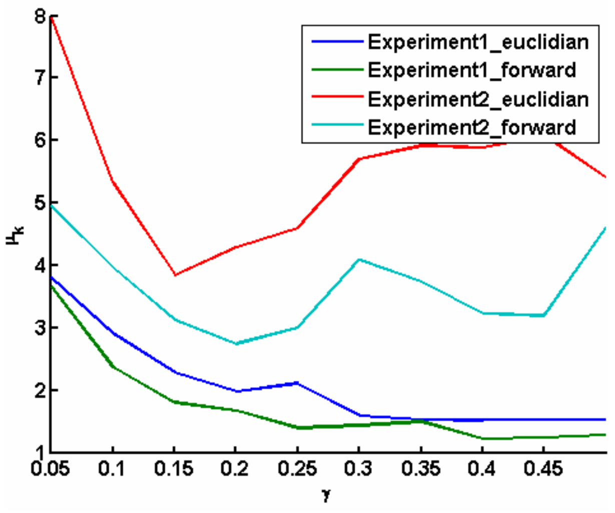
Figure 3 shows the resulting mean number of iterations $\mu(k)$ over all cycles of the path planning for each parameter set and estimator used. Several aspects of the influence of γ and ξ can be gathered from these results. Firstly, it is obvious, that the mean number of iterations for the simple experiment is lower than the mean number of iterations for the more complex experiment in nearly all runs, as the influence of the obstacles on the time estimation is higher for the complex scenario. Also, as only a single obstacle is interfering with the robot in the simple experiment, the a posteriori times are a good estimation and therefore lower values of γ result in a higher number of iterations as only a small fraction of the estimation error is added to the a priori estimation, while raising the factor results in less iterations needed to let the estimation error drop below the threshold ξ. For the complex scenario, the influence of the factor γ is diverse. On the one hand, values of γ above 0.25 change the estimated time from one iteration to the next so much, that the a posteriori time using these different a priori estimations oscillates. Using these values the maximum number of iterations was reached more often, as the iterative process could not converge. On the other hand, values lower than 0.15 converge reliably but too slow, again needing a higher number of iterations. The total number of iterations needed to push the estimation error below the threshold ξ rises for lower values of the threshold. The few exceptions to this statement concerning the complex experiment come from a different movement in the first cycles due to a higher estimation error, resulting in a less optimal position for the later cycles. Finally, the mean number of iterations needed in each cycle of the



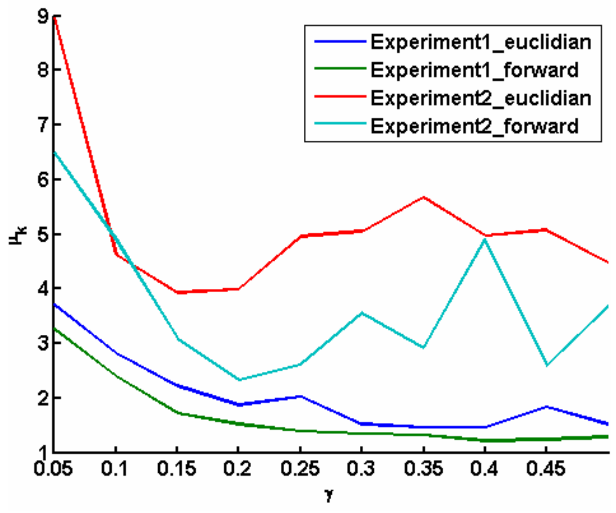
$\xi = 0.10\text{s}$



$\xi = 0.15\text{s}$



$\xi = 0.20\text{s}$



$\xi = 0.25\text{s}$

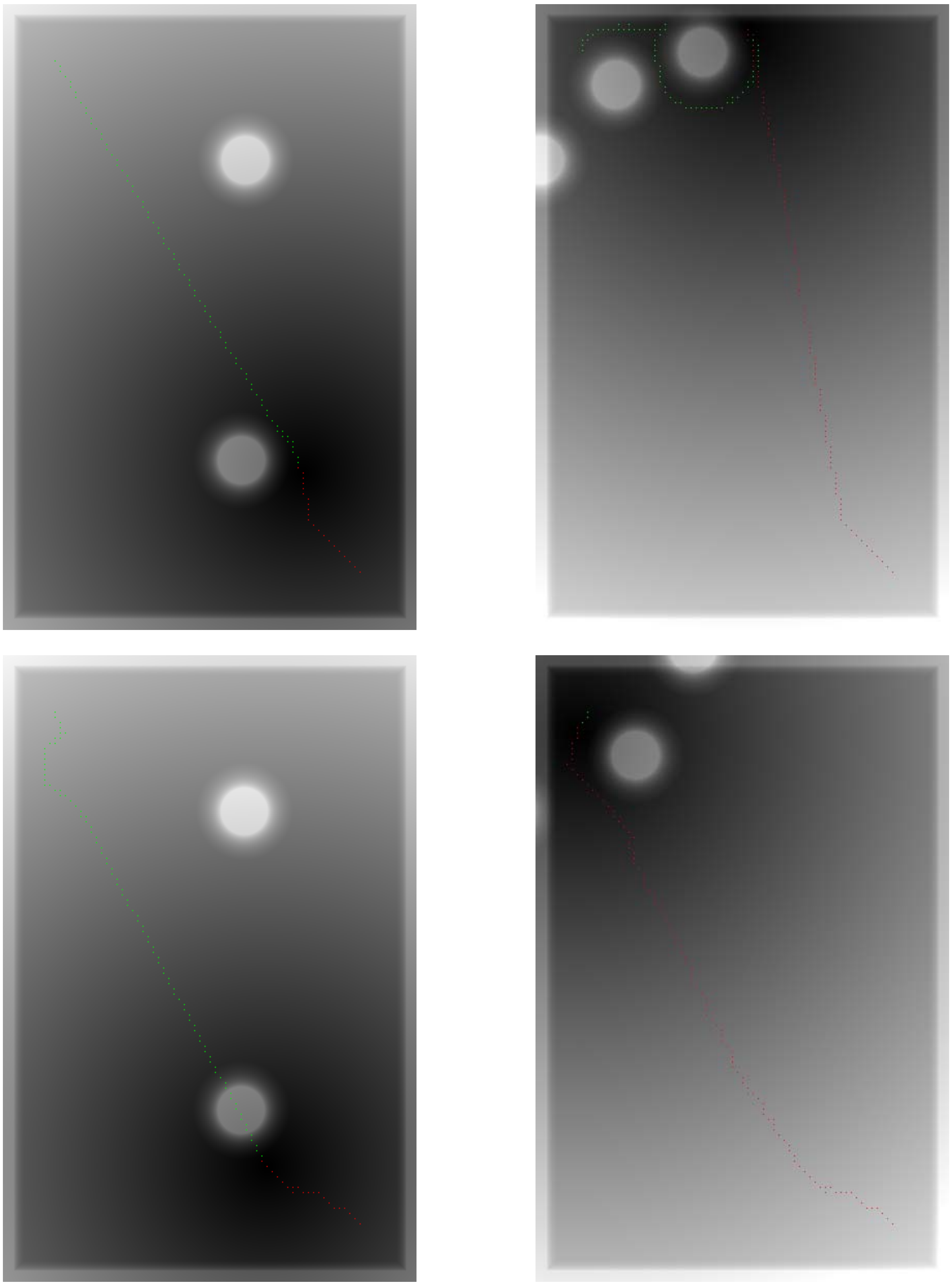
These graphs investigate the influence of the factor γ and the threshold ξ on the number of iterations made in each cycle of the planning algorithm. The mean number of iterations $\mu(k)$ over all cycles of the path planning for two different Experiments is compared for different values of γ and ξ. The first Experiment is the simple scenario that was used to create the snapshots in Figure 2, while the second Experiment is a more complex scenario including three moving obstacles on a field of 12x8m^2 size where a path is planned over more than 12m from one corner of the field to the other.

Figure 3

Euclidian distance estimator is higher than that of the forward planning estimator. For the computation time, however, one has to include the forward planning step to estimate the time in the forward planning estimator. This is a complete forward planning that is comparable to one iteration regarding the computation time.

Based on these experiments, values of $\gamma = 0.25$ and $\xi = 0.2$ were chosen for the following experiments as these values seemed to be a good compromise for a low number of iterations in all types of scenarios, simple as well as complex.

For the chosen values, the two experiments are compared concerning the efficiency on the Pentium-M 2GHz hardware of our middle size league RoboCup team. As computation time on our robot hardware is extremely rare, the maximum number of iterations is limited to $k_{max} = 5$. The mean computation time per cycle $\mu(t_n)$ and the length of the planned path in cycles N is shown in Table 2. Graphical visualizations of the planned path are shown in Figure 2 for the simple and Figure 4 for the complex experiment. Although the computation time is much higher for the more complex scenario when using the proposed approach, it is still able to run in the 20ms global cycle of our RoboCup robots. However, the other processes like image processing and environment modelling were not running at the same time.



In this experiment the robot is located in the lower right corner and plans a path towards the target in the upper left corner. A moving obstacle starts on the left of the robot and moves into nearly the same direction crossing the diagonal at the middle of the field. The conventional path planner (upper images) tries to get around the right side of the obstacle until it finally has to completely surround the obstacle in cycle 105. The path planner using the time variant potential field presented in this paper incorporates the velocity of the obstacle and plans its path to the left of the obstacle directly from the beginning.

Figure 4

In order to compare the performance of the path planning algorithm based on time variant potential fields to the conventional path planner of Weigel *et al.* in typical RoboCup scenarios, 100 random scenarios were created. On a field of 12x8m^2 size, which is the standard field size for the middle size league in the WorldCup 2006, seven obstacles were created at random positions with a random vector of movement. The number of obstacles was chosen to be seven, as this is the usual number of field players in the middle size leage of the RoboCup apart from the own robot. The obstacles keep their movement until they reach the field boundaries, where their movement is reflected back into the field. The start and target points of the robot are chosen randomly, too, such that they do not lie inside an obstacle. Furthermore, the target is only accepted if the Euclidian distance to the start is not less than 6m to get reasonable scenarios and not more than 8m, as this is the maximum distance to the target used in our RoboCup software. Again, the maximum number of iterations is limited to $k_{max} = 5$.

	Simple experiment		Complex experiment	
	N	$\mu(t_n)$	N	$\mu(t_n)$
Conventional path planner	52	0.64ms	140	2.61ms
Euclidian distance estimator	46	1.47ms	106	9.34ms
Forward distance estimator	46	1.43ms	107	8.70ms

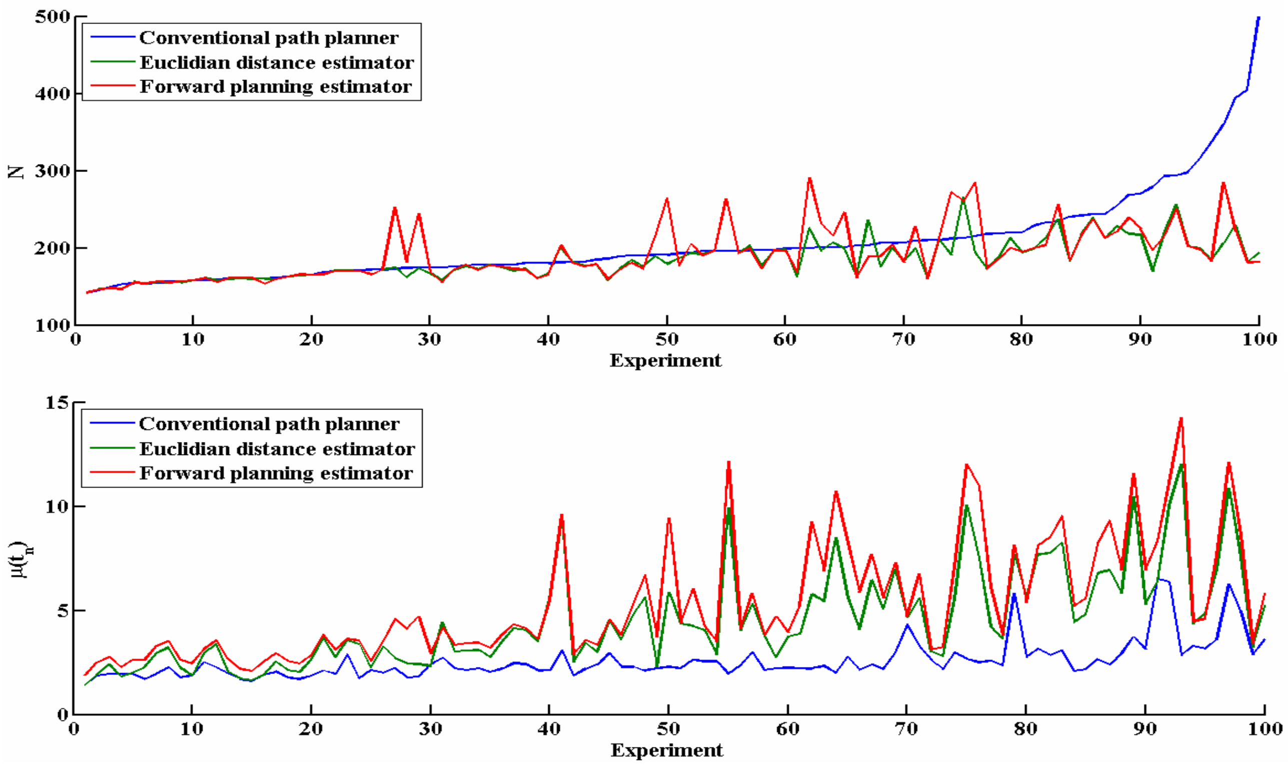
Table 2

Figure 5 presents the results of the 100 runs for the conventional method and each estimator. The upper graph displays the number of cycles N it took to follow the planned path, as this is directly linked to the efficiency of the planned path concerning the time needed to follow the path and the length of the path. The lower graph shows the mean time per cycle $\mu(t_n)$ as a measure of the computational load needed to compute the path. For better visibility the random runs are sorted by the number of cycles of the conventional planner. In the first 60 experiments all three path planners are comparable in terms of the path length apart from single outliers for the forward planning estimator. Although not meaningful in its absolute value (these experiments were run on an AMD-Opteron cluster), the mean time per cycle is higher for the proposed path planner, as it needs more iterations per cycle to estimate the time T. But only from experiment 50 to 60, the computation time per cycle is significantly higher. From experiment 60 on, the computation time for the proposed path planning algorithm is extremely high compared to the conventional planner most of the time. Nevertheless, the path planned by the proposed algorithm is extremely more efficient in nearly all experiments. The most impressive difference appears in experiment 100, where the path planned by the conventional planner is 150% longer than the path planned by the proposed algorithm. Although this is an exceptional result, the fact that the new path planner could reach the target 300 cycles before the conventional path planner is impressive, as this means an extra pathway of 12m or a time gap of 6s at 2m/s maximum speed. Taking all experiments into account, the mean number of cycles per experiment $\mu(N)$ is 184.62 for the Euclidian distance estimator and 191.41 for the forward planning estimator compared to 205.72 for the conventional planner. Thus, the average path length for the Euclidian distance estimator is only 89.74% of the path length planned with the conventional planner and 93.04% for the forward planning estimator. On the other hand, the mean time per cycle needed to compute the path averaged over all experiments $\mu(\mu(t_n))$ is higher for the proposed planning algorithm by a factor of 1.77 for the Euclidian distance estimator and 2.09 for the forward planning algorithm. See Table 3 for the exact values.

Finally, the comparison between the two different time estimators contains interesting results. Although the forward planning estimator should give a better estimation of T than the Euclidian distance estimator, the latter performs better on the conducted experiments. Apart from the mean number of cycles, the Euclidian distance estimator is also more efficient in planning its paths as the cycle time shows. In only 3% of the experiments the paths planned by this estimator are more than 10 cycles longer than the paths planned by the forward planning estimator, while in 17% it has planned paths that are more than 10 cycles shorter.

	$\mu(N)$	$\sigma(N)$	$\mu(\mu(t_n))$	$\sigma(\mu(t_n))$
Conventional path planner	205.72	58.29	2.56ms	0.95ms
Euclidian distance estimator	184.62	26.30	4.53ms	2.36ms
Forward planning estimator	191.41	35.09	5.34ms	2.80ms

Table 3



The performance of the conventional path planner compared to the presented path planner based on time variant potential fields. 100 random scenarios were created on a field of 12x8m^2 size by randomly choosing the position and velocity of 7 obstacles and start and target point of the path. The graphs show the results concerning the cycles N needed to follow the planned path and the mean time per cycle needed to compute the path $\mu(t_n)$.

Figure 5

Conclusions

In this paper a path planning algorithm based on time variant potential fields is presented that incorporates the movement of dynamic obstacles to plan more efficient paths. The results of this algorithm show, that the iterative path planning process is able to improve the paths planned for many scenarios with dynamic obstacles. However, the additional computation time needed to iteratively estimate the time T needed for the prediction of the obstacle position for the backwards planning uses more computational resources, which might not be available, depending on the given application. From the two presented estimators the Euclidian distance estimator outperforms the forward planning estimator both in efficiency and in computation time of the planned paths. The reason for this might be the underestimation of T of the Euclidian distance estimator and the used factor of $\gamma = 0.25$ which is very small so that the correct estimation of T is approached in small steps.

Future work on this topic will include a modification of the Euclidian distance estimator that uses the Euclidian Distance as a first underestimation and then raises the a priori time estimation by ξ until the estimation error becomes negative and then lowers the estimation by $\xi/2$ to return to a value below the threshold. Additionally, experiments with real robot data will be conducted to investigate the influence of opponents reacting to the movements that result from the new path planner. Up to now all obstacles in the experiments were moving with a constant speed. However, it is very difficult to prepare real robot scenarios that can be repeated to compare the performance of different planning algorithms. Nevertheless, our first results using the path planner on our RoboCup robots were promising.

References

[1] J. Baltes and N. Hildreth: *Adaptive Path Planner for Highly Dynamic Environments*. In *RoboCup 2000: Robot Soccer World Cup IV*, LNCS, 2019:76-85, Springer, 2001.

[2] J. Bresenham: *Algorithm for computer control of a digital plotter*. In *IBM Systems Journal*, Vol.4, No. 1, 1965.

[3] B. Damas, P. Lima, and L. Custódio: *A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer*. In *RoboCup 2002: Robot Soccer World Cup VI*, LNCS, 2752:65-77, Springer, 2003.

[4] O. Khatib: *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*. In *Proceedings of the International Conference on Robotics and Automation*, pp. 500-505, 1985.

[5] H. Kitano, M. Asada, Y. Kuniyoshi, et al.: *RoboCup: The Robot World Cup Initiative*. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pp. 340-347, ACM Press, 1997.

[6] D. Sorrenti and H. Fujii: *Summary of Team Questionnaires*. Homepage of the Robocup middle-size league for 2005. http://old.disco.unimib.it/robocup05msl.

[7] T. Weigel, A. Kleiner, F. Diesch, et al.: *CS Freiburg 2001*. In *RoboCup 2001: Robot Soccer World Cup V*, LNCS, 2377:26-38, Springer, 2002.

Authors

Patrick Heinemann received his diploma in technical computer science (2001) from the University of Kaiserslautern, Germany. From 2001-2002 he worked as researcher for driver assisting systems for the DaimlerChrysler AG and is a PhD-student at the W.-Schickard-Institute of Computer Science at the Eberhard-Karls-University of Tübingen since 2003. As team leader of the Attempto Tübingen Robot Soccer Team his main research interests are cooperating mobile robots for RoboCup, including real-time algorithms for robot vision and environment modelling.

Hannes Becker studies computer science at the University of Tübingen since 2001 and is a member of the Attempto Tübingen Robot Soccer Team for 2 years. His main research interests are path planning and high level robot control.

Andreas H. Zell is Professor at the W.-Schickard-Institute of Computer Science at the Eberhard-Karls-University of Tübingen and director of the Centre for Bioinformatics Tübingen (ZBIT). He received his diploma in computer science (1987) from the University of Kaiserslautern, Germany, an M.S. in CS (1987) from Stanford University, CA, USA, and his Ph.D. (1989) and the venia legendi (1994) from the University of Stuttgart, Germany. In 1995 he was appointed full professor for computer architecture at the University of Tübingen, Germany, where he established the bioinformatics curriculum and the ZBIT. From 2000-2002 he was dean of the CS department at the University of Tübingen. His research interests include artificial neural networks, evolutionary algorithms and their applications, autonomous mobile robots including robot vision and other sensors, bioinformatics and computational chemistry applications.