# Trajectory Planning for Robots in Dynamic Human Environments

Svenstrup, Mikael; Bak, Thomas; Andersen, Hans Jørgen

*Published in:*
Trajectory Planning for Robots in Dynamic Human Environments

*DOI (link to publication from Publisher):*
10.1109/IROS.2010.5651531

*Publication date:*
2010

*Document Version*
Publisher's PDF, also known as Version of record

The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

# Trajectory Planning for Robots in Dynamic Human Environments

Mikael Svenstrup, Thomas Bak and Hans Jørgen Andersen

*Abstract*— **This paper presents a trajectory planning algorithm for a robot operating in dynamic human environments. Environments such as pedestrian streets, hospital corridors, train stations or airports. We formulate the problem as planning a minimal cost trajectory through a potential field, defined from the perceived position and motion of persons in the environment.**

**A Rapidly-exploring Random Tree (RRT) algorithm is proposed as a solution to the planning problem, and a new method for selecting the best trajectory in the RRT, according to the cost of traversing a potential field, is presented. The RRT expansion is enhanced to account for the kinodynamic robot constraints by using a robot motion model and a controller to add a reachable vertex to the tree.**

**Instead of executing a whole trajectory, when planned, the algorithm uses a Model Predictive Control (MPC) approach, where only a short segment of the trajectory is executed while a new iteration of the RRT is computed.**

**The planning algorithm is demonstrated in a simulated pedestrian street environment.**

## I. Introduction

As robots integrate further into our living environments, it becomes necessary to develop methods that enable them to navigate in a safe, reliable, comfortable and natural way around humans.

One way to view this problem is to see humans as dynamic obstacles that have social zones, which must be respected. Such zones can be represented by potential fields [1], [2]. The navigation problem can then be addressed as a trajectory planning problem for dynamic environments with a potential field representation. Given the fast dynamic nature of the problem, robotic kinodynamic and nonholonomic constraints must also be considered.

In the recent decade sampling based planning methods have proved successful for trajectory planning [3]. They do not guarantee an optimal solution, but are often good at finding solutions in complex and high dimensional problems. Specifically for kinodynamic systems Rapidly-exploring Random Trees (RRT's), where a tree with nodes correspond to connected configurations (vertices) of the robot trajectory, has received attention [4].

Various approaches improving the basic RRT algorithm have been investigated. In [5], a dynamic model of the robot and a cost function is used to expand and prune the nodes of the tree. A Model Predictive Control (MPC) approach is taken, where only a small part of the trajectory is executed, while a new trajectory is calculated. When expanding a vertex, a random vertex and a random control input is chosen.

In [6], an approach for better choices of vertices to expand, is proposed. It is based on a reachable set of configurations for each vertex.

It is often desirable to run the planning algorithm in real time, hence requiring bounded solution time. One approach is to use anytime algorithms, which initially find a quick suboptimal solution, and then keep improving the solution until time runs out [7].

Methods for incorporating dynamic environments, have also been investigated. Solutions include extending the configuration space with a time dimension ($\mathcal{C} - \mathcal{T}$ space), in which the obstacles are static [8], as well as pruning and rebuilding the tree when changes occur [9], [10].

All these result only focus on avoiding collisions with obstacles. However, there have been no attempts to navigate through a human crowd taking into account the dynamics of the environment and at the same time generate comfortable and natural trajectories around the humans. E. Hall [11] has analysed how people position themselves socially relative to each other. He divides the area around a person into four zones (public, social, personal and intimate). These zones can be used to plan how a robot should move to make the motion natural and comfortable.

In this paper will formulate the problem of navigating through a dynamic human environment, as planning a trajectory through a potential field. The overall mission of the robot is, to move forward through the environment with a desired average speed and direction, which can be set by a top level planner. This paper contributes by enhancing the basic RRT planning algorithm to accommodate for navigation in a potential field and take into account the kinodynamic constraints of the robot. The RRT is expanded using a control strategy, which ensures feasibility of the trajectory and a better coverage of the configuration space. The planning is done in $\mathcal{C}-\mathcal{T}$ space using an MPC scheme to incorporate the dynamics of the environment. To be able to run the algorithm on-line, the anytime concept is used to quickly generate a possible trajectory. The RRT keeps being improved until a new trajectory is required by the robot, which can happen at any time.

The trajectory planning problem is formulated in Section II, and the algorithm for generating the trajectory is described in Section III. Finally in Sections IV-V the algorithm is demonstrated in an experiment, where the robot plans the trajectory through a simulated pedestrian street.

M. Svenstrup and T. Bak are with the Department of Electronic Systems, Automation & Control, Aalborg University, 9220 Aalborg, Denmark {ms,tba}@es.aau.dk

H.J. Andersen is with the Department of Media Technology, Aalborg University, 9220 Aalborg, Denmark hja@imi.aau.dk

## II. Trajectory Generation Problem

### A. Robot Dynamics

The robot is modelled as a unicycle type robot, i.e. like a Pioneer, an iRobot Create or a Segway. A good motion model for the robot is necessary because it operates in dynamic environments, where even small deviations from the expected trajectory may result in collisions. So instead of using a purely kinematic robot model of the robot, it is modelled as a dynamical system, with accelerations as input. This describes the physics better, since acceleration and applied force are proportional. This dynamical model can be described by the five states:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \begin{matrix} \rightarrow \text{x position} \\ \rightarrow \text{y position} \\ \rightarrow \text{linear velocity} \\ \rightarrow \text{rotation angle} \\ \rightarrow \text{rotational velocity} \end{matrix} \quad (1)$$

The differential equation governing the robot behaviour is:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\theta}(t) \\ \ddot{\theta}(t) \end{bmatrix} \\ &= \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ u_v(t) \\ \dot{\theta}(t) \\ u_\theta(t) \end{bmatrix} = \begin{bmatrix} x_3(t)\cos(x_4(t)) \\ x_3(t)\sin(x_4(t)) \\ u_1(t) \\ x_5(t) \\ u_2(t) \end{bmatrix} \end{aligned} \quad (2)$$

where $u_1 = u_v$ is the linear acceleration input and $u_2 = u_\theta$ is the rotational acceleration input.

Without loss of generality the starting time can be set to 0, and the trajectory is then calculated as:

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right) d\tau \quad (3)$$

### B. Dynamic Potential Field

The value of the potential field, denoted $G$, at a point in the environment is calculated as a sum of the cost associated with three different aspects:

1) A cost related to the robots position in the environment without obstacles. For example high costs might be assigned close to the edges.
2) A cost associated with the robot position relative to humans in the area.
3) A cost rewarding moving towards a goal.

The combined cost can be written as:

$$G(t) = g_1(\boldsymbol{x}(t))) + g_2(\boldsymbol{x}(t), \mathcal{P}(t)) + g_3(\boldsymbol{x}(t)) \quad . \quad (4)$$

$\mathcal{P}(t)$ is a matrix containing the position and orientation of persons in the environment at the given time. $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$ are the three cost functions. They are further described below.

*1) Cost related to environment:* This cost function is currently designed for non agoraphobic behaviour of the robot, i.e. in open spaces, such as a pedestrian street. It has the shape of a valley, such that it is more expensive to go towards the sides, but cheap to stay in the middle:

$$g_1(\boldsymbol{x}(t)) = c_y y^2(t) \quad (5)$$

where $c_y$ is a constant determining how much the robot is drawn towards the middle.

*2) Cost of proximity to humans:* This is not a straightforward calculation, and for more detail, see [2]. The shape of the potential field is related to how humans position themselves around others, and is based on Hall's proxemic distances [11]. For example the potential is lower in front of the person than behind, because it is more comfortable to have other persons, where you can see them.

Fig. 1 shows a potential field around a person. The person stands in the point $(0, 0)$ and is looking to the left. A robot should try to move towards the lower parts of the potential function, i.e. towards the dark blue areas, and avoid the red area. The formula for calculating the potential around one person is a summation of four normalized bi-variate Gaussian distributions:

$$g_2(\boldsymbol{x}_{1:2}) = \sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x}_{1:2} - \boldsymbol{0}]^T \Sigma_k^{-1} [\boldsymbol{x}_{1:2} - \boldsymbol{0}]) \quad (6)$$

where $c_k$ are a normalizing constants, $\boldsymbol{x}_{1:2}$ are the first two states of the robot state, i.e. the position relative to the person, where the cost function is evaluated. $\boldsymbol{0}$ is the position of the person, in this case the origin, and $\Sigma_k$ are the covariances of each of the Gaussian distributions. The covariances are adjusted according to the orientation of the person. The total cost, $g_2$ is a summation over all of the persons in the area.

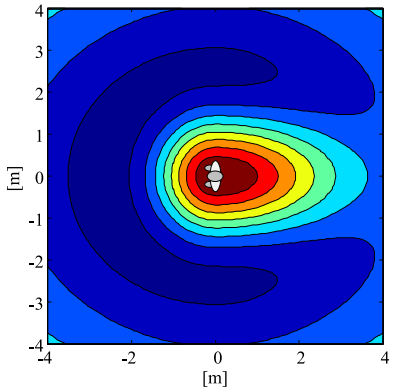


Fig. 1. Potential field around a person standing at $(0, 0)$ and looking to the left. The robot should try to get towards the lower points, i.e. the dark blue areas. The size of the person in this figure is approximate equal to a normal human being.

*3) Cost of end point in trajectory:* The cost at the end point penalizes if the robot does not move forward, and if the robot orientation is not in a forward direction. An exponential function is used to penalize the position. It is set up such that short distances are penalized much, while it is close to the

same value for larger distances, i.e. it does not change much if the robot goes 19 or 20 meters from its starting position.

$$g_3(\boldsymbol{x}(t)) = c_{e1}\exp(c_{e2}(x(t) - \tilde{x}(0))) + c_\theta \theta^4(t) \quad , \quad (7)$$

where $c_{(\cdot)}$ are scaling constants and $\tilde{x}(0)$ is the desired position at $t = 0$. The reason that $\theta$ is raised to the fourth, is to keep the term closer to zero in a larger neighbourhood of the origin. This means that the robot will almost not be penalized for small turns. On the other hand larger turns, like going the wrong way, will be penalized more.

### C. Minimization Problem

Given the above cost functions a potential landscape may be formed. Fig. 2 illustrates an example of a pedestrian street landscape with five persons. The robot is initially positioned at position $(2, 0)$ and has to move to the right. The area is bounded to be $20m$ wide, i.e. $10m$ to each side of the robot from the initial position. Examples of three different randomly chosen trajectories are shown in the figure.

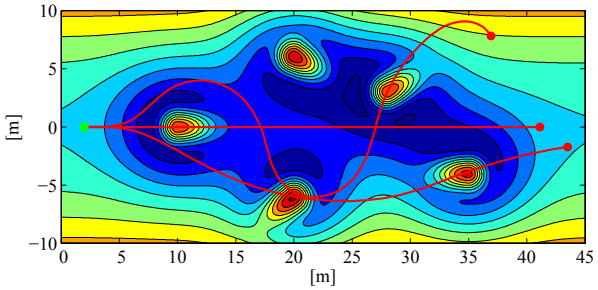


Fig. 2. Person potential field landscape, which the robot has to move through. The robot starting point is the green dot at the point $(2, 0)$. Three examples of potential robot trajectories are shown.

At a first glance it looks like all three trajectories would run into at least one human, but since the persons move while the robot advances along the trajectory, this might not be the case. Conversely the robot may also run into a person, who was not originally on the path. Therefor it is important to take into account the dynamics of the obstacles (i.e. the humans), when planning trajectories.

If the current time is $t = 0$, the planning problem can be posed as follows. Given an initial robot state $\boldsymbol{x}_0$, and trajectory information for all persons until the given time $\tilde{\mathcal{P}}_{start:0}$. Determine the control input $\tilde{\boldsymbol{u}}_{0:T}$, which minimizes the cost of traversing the potential field, subject to the dynamical robot model constraints:

$$\begin{aligned}
&\textit{minimize} && I(\tilde{\boldsymbol{u}}_{0:T}) = \quad (8)\\
&&& \int_0^T [g_1(\boldsymbol{x}(t)) + g_2(\boldsymbol{x}(t), \mathcal{P}(t))]\, dt + g_3(\boldsymbol{x}(T))\\
&\textit{s.t.} && \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\,(\boldsymbol{x}(t), \boldsymbol{u}_t)\\
&\textit{where} && g_1(\boldsymbol{x}(t)) = c_y x_2(t)^2\\
&&& g_2(\boldsymbol{x}(t), \mathcal{P}(t)) =\\
&&& \sum_{j=1}^{p}\sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x_{1:2}} - \boldsymbol{\mu}_j]^T \Sigma_{j,k}^{-1}[\boldsymbol{x_{1:2}} - \boldsymbol{\mu}_j])\\
&&& g_3(\boldsymbol{x}(T)) = c_{e1}\exp(c_{e2}(x_1(T) - x_1(0))) + c_\theta x_4^4(T),
\end{aligned}$$

were $\boldsymbol{x_{1:2}} = [x_1(t), x_2(t)]$ is the position of the robot at time $t$, $\tilde{\boldsymbol{u}}_{0:T}$ is the discrete input sequence to the robot. $T$ is the ending time horizon of the trajectory, $g_x(\cdot)$ are cost functions and $p$ is the number of persons in the area. The position and orientation of all persons at time $t$ is given by $\mathcal{P}(t)$ and $\boldsymbol{\mu}_j$ is the center of the $j$-th person at a given time.

To be able to calculate the cost of a trajectory according to Eq. (8), only the person trajectories remains to be defined. A simple model is that the person will continue with the same speed and direction [12]. More advanced human motion models could be used without changing the planning algorithm, but it is outside the scope of this paper to derive a complex human motion model.

## III. RRT Based Trajectory Planning

The structure of the planning algorithm can be seen in Fig.3. The idea is that while a trajectory is executed, a new is calculated on-line. Input to the trajectory planner is the previous best trajectory, the person trajectory estimates, and the dynamic model of the robot.

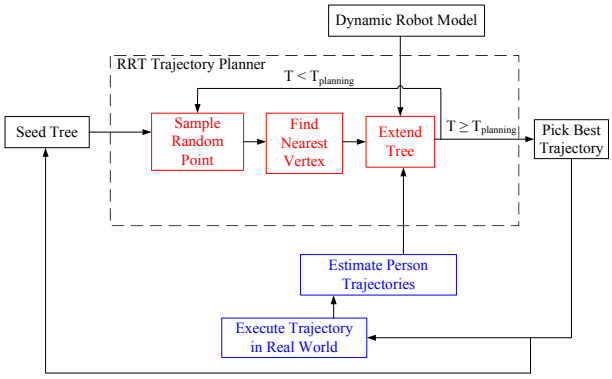


Fig. 3. The overall structure of the trajectory generator. The blue real world part and the red trajectory planning part are executed simultaneously.

The minimization problem stated in Eq. (8) is addressed by RRT's. A standard RRT algorithm is shown in Algorithm 1, where the lines $4, 5, 6$ correspond to the three blocks in the larger *RRT Trajectory Planner* box in Fig. 3.

The method presented here differs from the standard RRT in lines $1, 3, 6, 9$, which are marked red. Furthermore, between line 6 and 7, node pruning is introduced. Since an MPC scheme is used, only a small portion of the planned trajectory is executed, while the planner is restarted to plan a new trajectory on-line. When the small portion has been executed, the planner has an updated trajectory ready. To facilitate this, the stopping condition in line 3 is changed. When a the robot needs a new trajectory, or when certain maximum number of vertices have been extended, the RRT is stopped. Even though the robot only executes a small part of the trajectory, the rest of the trajectory should still be valid. Therefore, in line 1, the tree is seeded with the remaining trajectory.

In line 9 the trajectory with the least cost is returned, instead of returning the trajectory to the newest vertex. The

**Algorithm 1** Standard RRT (see [13])

**RRTmain**()

1: Tree = q.start
2: q.new = q.start
3: **while** **Dist**(q.new , q.goal) < ErrTolerance **do**
4: q.target = **SampleTarget**()
5: q.nearest = **NearestVertex**(Tree , q.target)
6: q.new = **ExtendTowards**(q.nearest,q.target)
7: Tree.add(q.new)
8: **end while**
9: **return** **Trajectory**(Tree,q.new)

**SampleTarget**()

1: **if** **Rand**() < GoalSamplingProb **then**
2: **return** q.goal
3: **else**
4: **return** **RandomConfiguration**()
5: **end if**

tree extension function and the pruning method are described below.

### A. RRT Control Input Sampling

When working with nonholonomic kinodynamic constrained systems, it is not straightforward to expand the tree towards a newly sampled point in the configuration space (line 6 in Algorithm 1). It is a whole motion planning problem in itself to find inputs, that drive the robot towards a given point [14]. The algorithm proposed here uses a controller to turn the robot towards the sampled point and to keep a desired speed. A velocity controller is set to control the speed towards an average speed around a reference velocity. The probabilistic completeness of RRT's in general, is ensured by the randomness of the input. So to maintain this randomness in the input signals, a random value sampled from a Gaussian distribution is added to the controller input. The velocity controller is implemented as a standard proportional controller. The rotation angle is a second order system with $\theta$ and $\dot{\theta}$ as states, and therefore a state space controller is used for control of the orientation. The control input can be written as:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_v(\mu_v - v(t)) \\ k_{\theta 1}(\phi_{point} - \theta(t)) - k_{\theta 2}\dot{\theta} \end{bmatrix} + \begin{bmatrix} \mathcal{N}(0, \sigma_v) \\ \mathcal{N}(0, \sigma_\theta) \end{bmatrix}. \quad (9)$$

$\mu_v$ is the desired average speed of the robot and $\phi_{point}$ is the angle towards the sampled point. $k_{(\cdot)}$ are controller constants and $\sigma_v, \sigma_\theta$ are the standard deviations of the added Gaussian distributed input.

The new vertex to be added to the tree is now found by using the dynamic robot motion model and the controller to simulate the trajectory one time step. This ensures that the added vertex will be reachable.

### B. Tree Pruning and Trajectory Selection

A simple pruning scheme, based on several different properties of a node, is used. If the vertex corresponding to the node ends up in a place where the potential field has a value above a specific threshold, then the node is not added to the tree. Furthermore a node is pruned if $|\theta(t)| > \frac{\pi}{2}$, which means that the robot is on the way back again, or if the simulated trajectory goes out of bounds of the environment. It is not desirable to let the tree grow too far in time, since the processing power is much better spend on the near future, because of the uncertainty of person positions further into the future. Therefore the node is also pruned if the time taken to reach the node is above a given threshold. Finally, instead of returning the trajectory to the vertex of the last added node, the trajectory with the lowest cost (calculated from Eq. (8)), is returned. But to avoid the risk of selecting a node, which is not very far in time, all nodes with a small time are thrown away before selecting the best node.

The final algorithm is shown in Algorithm 2.

**Algorithm 2** Modified RRT for human environments

**RRTmain**()

1: Tree = q.oldBestTrajectory
2: **while** (Nnodes < maxNodes) and (t < tMax) **do**
3: q.target = **SampleTarget**()
4: q.nearest = **NearestVertex**(Tree , q.target)
5: q.new = **CalculateControlInput**(q.nearest,q.target)
6: **if** **PruneNode**(q.new) == false **then**
7: Tree.add(q.new)
8: **end if**
9: **end while**
10: **return** **BestTrajectory**(Tree)

**SampleTarget**()

1: **if** **Rand** < GoalSamplingProb **then**
2: **return** q.goal
3: **else**
4: **return** **RandomConfiguration**()
5: **end if**

## IV. Simulations

The above described algorithm is implemented, and demonstrated to work on a simulated pedestrian street, as shown in Fig. 2. The experiments consist of two parts. First, the algorithm is applied on the environment shown in Fig. 2. This will demonstrate that the algorithm is capable of planning a trajectory, which does not collide with any persons. It will also demonstrate how the tree expands. The algorithm is compared to an algorithm where a random vertex and a random control input is chosen, as suggested in [4] and used to different extends in e.g. [5], [6]. Next, a simulated navigation through several randomly generated worlds is performed. This will demonstrate the robustness of the algorithm over time.

The following parameters for the potential field are used: $c_y = 0.1$, $c_{e1} = 20$, $c_{e2} = -0.1$, $c_\theta = 10$, and the parameters for the Gaussian distributions can be seen in [2]. The poles of the controllers, the reference velocity and the standard deviation of the velocity input, are the only other parameters to set. The poles have experimentally been determined, such that the robot has a relatively quick response, but the exact pole placement does not influence the trajectory generation much. The pole of the velocity controller is placed in $s =$

$-2$, and both the poles of the rotational controller are placed in $s = -2$ as well. The standard deviation of the added random velocity input is set to $\sigma_v = 2\frac{m}{s}$ and the standard deviation of the rotational input is set to $\sigma_\theta = 0.5\frac{rad}{s}$. The reference velocity is set to $1.5\frac{m}{s}$, which is considered as a normal human walking speed.

### A. Robustness Test

The robustness test is performed in 50 different randomly generated environments, where the robot has to navigate forwards in one minute. With an average speed of $1.5\frac{m}{s}$, this corresponds to the robot moving approximately $90m$ ahead along the street. In each simulation the robot's initial state is:

$$\boldsymbol{x}_0 = [2\ 0\ 0\ 0\ 0]^T \tag{10}$$

First the motion of all the persons in the world are simulated. Initially a random number of persons (between 10 and 20) are placed randomly in the world. Their velocity is sampled randomly from a Gaussian distribution. The motion of each person is simulated as moving towards a goal $10m$ ahead of them. The goal position of the $y - axis$ of the street is sampled randomly, and will also change every few seconds. Additional Brownian motion is added to each person to include randomness of the motion. Over time new persons will enter at the end of the street according to a Poisson process. This means that at any given time, persons will appear and disappear at the ends of the street. Because of this randomness, the number of persons can differ from the initial number of persons, and ranges from 10 to around 40, which is different for each simulation.

At each time instant, the robot will only know the current position and velocity of each of the persons within a range of $45m$ in front of the robot, and it has no knowledge about where the persons will go in the future.

As it is a simulation, there is no real time performance issues, and nothing has been done to optimize the code for faster performance. So the tree is set to grow a fixed number of 2000 vertices at each iteration. The planning horizon is set to 20 seconds and at each iteration the robot executes 2 seconds of the trajectory, while a new trajectory is planned.

## V. Results

An example of a grown RRT, with 2000 vertices, from the initial state can be seen in Fig. 4. The simulated trajectories of the robot are the red lines, and the red dots are vertices of the tree. It is seen how the RRT is spread out to explore the configuration space, although only every 10th vertex is plotted to avoid clutter on the graph. Note that the persons are static at their initial position on the figure, and some trajectories seem to pass through persons. But in reality, the persons have moved when the robot passes the point of the trajectory. The best of the all trajectories, which is calculated using Eq. (8), is the green trajectory.

In Fig. 5 a RRT has been run for the same environment, but by choosing a random vertex to expand, instead of the one closest to a random sampled point. The algorithm is a little faster, since it does not have to calculate distances to all

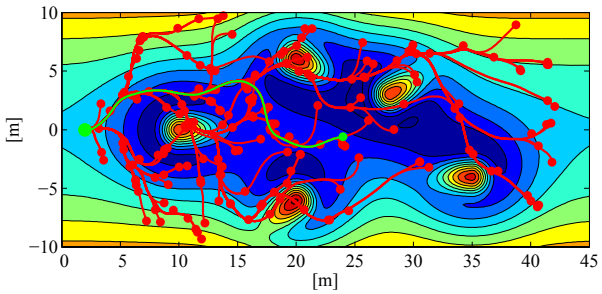


Fig. 4. An RRT for a robot starting at $(2, 0)$ and the task of moving forward through the human populated environment. Only every 10th vertex is shown to avoid clutter of the graph. The vertices are the red dots, and the lines are the simulated trajectories. The green trajectory is the least cost trajectory.

vertices. But as can be seen, the tree does almost not expand over the configuration space.

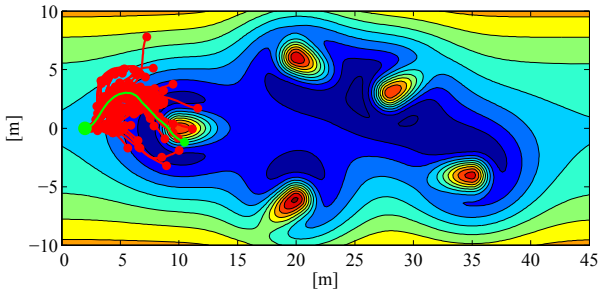


Fig. 5. The RRT after 2000 expansions, where a random vertex is chosen to be expanded. The tree expands very slow over the configuration space. The lowest cost trajectory is shown in green.

An example of the tree after 2000 expansions, when choosing the nearest vertex, but using a random control input , is illustrated in Fig. 6. The configuration space is not covered very well, since there are only two major branches of the tree, and the top area above the first person has not been explored at all. When comparing to Fig. 4, it is clear that our control sampling method covers the configuration space much better.

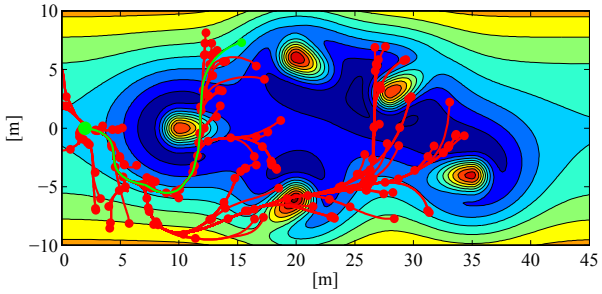


Fig. 6. An example of choosing random control input. The tree expands better than if choosing random nodes, but it still does not cover the configuration space very well. The lowest cost trajectory is shown in green.

In Fig. 7 a typical scene from one of the 50 simulations can be seen. The blue dots are persons, and the arrows are

velocity vectors, with a length proportional to the speed. The black star with the red arrow, is the robot position and orientation.

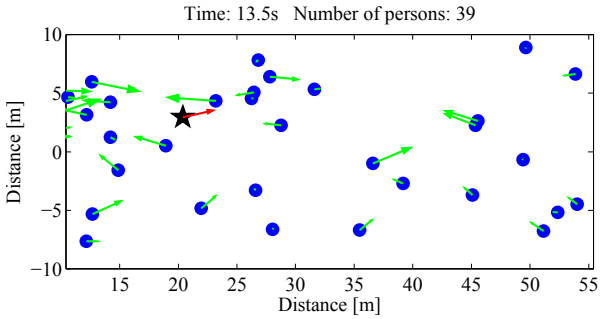


Fig. 7. A scene from one of the 50 simulations. The blue dots are persons, with their corresponding current velocity vectors. The black star is the robot.

In none of the 50 simulations the robot ran into a person. This demonstrates that the algorithm is robust enough to handle simulated human motion with changing goals and additional random motion, even though using the simple human motion model when planning. A few close passes in the combined 50 minute run, down the pedestrian street, are seen. But, as seen in Fig. 8, the robot stays out of the personal zone of any of the persons in more than $97.5\%$ of the time, and out of the intimate zone (a distance closer than $0.45m$) $99.7\%$ of the time. This only occurs on 9 separate instances of the 50 minutes of driving.

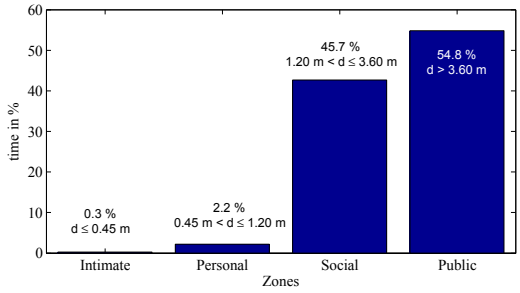


Fig. 8. The bar plot shows how large a part of the time the closest person to the robot has been in each zone. The robot should try to stay out of the personal and intimate zones. The distance interval for each zone, can also be seen in the plot.

Except in very densely populated environments, the model runs approximately one third of real time on a 2.0 $GHz$ CPU running MATLAB. This is considered to be reasonable, since no optimization for speed has been done.

In very dense environments, the planning takes longer, since many new added vertices, are pruned again, and hence more points has to be sampled before 2000 vertices are expanded. Additionally the more persons in the area, the longer it takes to evaluate the cost of traversing the potential field.

On average approximately $\frac{1}{4}$ of the sampled points led to an expansion of the tree and correspondingly $\frac{3}{4}$ led to a pruned node.

## VI. Conclusions

In this paper a new algorithm for trajectory planning for a kinodynamic constrained robot, has been described. The robot is navigating in a highly dynamic environment, which in this case is populated with humans. The algorithm is based on RRT's, but with a new trajectory selection method. The method enables the costs of traversing a potential field to be minimized, thereby supporting planning of comfortable and natural trajectories. Further, a new control input sampling strategy has been presented. This leads to better tree coverage over the configuration space, than if sampling e.g. a random control input. Together with a dynamic model of the robot, an MPC scheme is used to enable the planner to continuously plan a reachable trajectory on an on-line system.

The algorithm is challenged when the environments become very densely populated, but so are humans. Humans react by mutual adaptation and allowing violation of the social zones. This is not done here, where the robot takes on all the responsibility for finding a trajectory.

Potential future work include real life experiments, and incorporation of human to human motion correlation into the algorithm.

## References


[1] E. Sisbot, A. Clodic, L. Marin U., M. Fontmarty, L. Brethes, and R. Alami, "Implementing a human-aware robot system," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006.*, 6-8 Sept. 2006, pp. 727–732.

[2] M. Svenstrup, S. T. Hansen, H. J. Andersen, and T. Bak, "Pose estimation and adaptive robot behaviour for human-robot interaction," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, Kobe, Japan, May 2009, pp. 3571–3576.

[3] S. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.

[4] S. LaValle and J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, p. 378, 2001.

[5] A. Brooks, T. Kaupp, and A. Makarenko, "Randomised mpc-based motion-planning for mobile robot obstacle avoidance," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 3962–3967.

[6] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 2859–2865.

[7] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1310–1315.

[8] J. van den Berg, "Path planning in dynamic environments," Ph.D. dissertation, Ph. D. dissertation, Universiteit Utrecht, 2007.

[9] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proc. IEEE International Conference on Robotics and Automation ICRA 2006*, 2006, pp. 1243–1248.

[10] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1603–1609.

[11] E. T. Hall, "A system for the notation of proxemic behavior," *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.

[12] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Robotics and Automation, 2004. ICRA '04. IEEE International Conference on*, April 2004.

[13] D. Ferguson and A. Stentz, "Anytime rrts," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[14] B. Siciliano and O. Khatib, *Handbook of Robotics*. Springer-Verlag, Heidelberg, 2008.