# RRTs Postprocessing for Uncertain Environments

Agris Nikitenko
Department of Systems Theory and Design
Riga Technical University
Riga, Latvia
Agris.Nikitenko@rtu.lv

Martins Ekmanis
Department of Systems Theory and Design
Riga Technical University
Riga, Latvia
Martins.Ekmanis@rtu.lv

Aleksis Liekna
Department of Systems Theory and Design
Riga Technical University
Riga, Latvia
Aleksis.Liekna@rtu.lv

***Abstract*—Path planning is one of the central tasks to be solved in mobile robotics. Rapidly Exploring Random Tree is one of possible alternatives addressing path planning. While it does not deliver optimal solution it provides a good performance that is crucial for most cases in mobile robotic systems. Unfortunately the algorithm may produce unnecessary or even dangerous waypoints that might lead to collisions with obstacles or other robotic systems. A significant part of the possible collisions are caused by uncertainty of robot positioning or obstacle sensing. In this paper we propose a set of plan post processing steps to estimate and decrease the possibility of collisions during plan execution.**

***Keywords*— RRT planner; RRT post processing; Planning under uncertainty;**

## I. Introduction

Path planning is one of the central tasks to be solved in mobile robotics. Along with methods groups like Potential field planning [1] and Combinatorial planning [2], Rapidly Exploring Random Tree [3] (RRT) planning has found its application in mobile robotics. While it does not ensure optimal solutions and does not guarantee solution at all it provides sufficient performance [4] for most applications in mobile robotics.

Since its first implementations RRT planning has experienced a variety of modifications [4] that differ with implementations of particular algorithm steps and provide different overall performance under particular constraints [5]. However in real applications the robotic systems operate with noisy sensor data that leads to uncertainties in positioning and obstacle position detection. This causes uncertainties in robot's environment model and may result in incorrect assignment of free and occupied space in the map of the environment. These uncertainties entail collision risks that are highly unwanted for mobile robotic systems.

Within this paper we propose a set of steps that enables to estimate and decrease the collision possibility by modifying the initial plan generated by RRT.

The paper is organized as follows: Section II gives a brief overview of the RRT algorithm and related work regarding planning under uncertainty, Section III presents steps of the proposed method and their descriptions, Section IV provides experimental evaluation of the proposed method, Section V gives conclusions and insight of future work.

## II. Related Work

### A. The essence of RRT

The RRT was introduced as planning technique for wide range of motion planning problems [3], which can accommodate particular kinematic or geometric constraints of a given system. The planning goal is to generate a path from initial configuration $q_0$ to the goal configuration $q_g$. At each iteration i, a random configuration $q_{rnd}$ is selected. Then the closest configuration $q_c$ from the graph is found and algorithm tries to extend the planning graph towards $q_{rnd}$, by adding an arc from $q_c$ towards $q_{rnd}$ with length d. Thereby a new configuration $q_i$ is added to the planning graph. This step is depicted in Fig. 1.

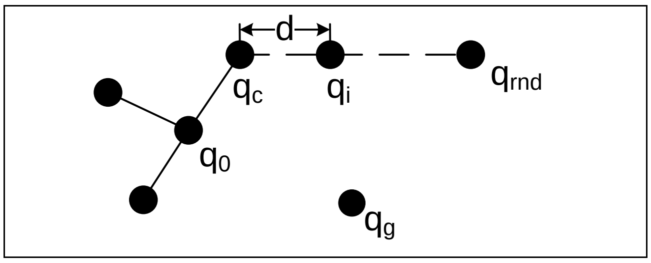

Fig. 1. RRT extending towards $q_{rnd}$

The planning stops when the newly added configuration $q_i$ is in a predefined proximity from the goal configuration $q_g$. A number of variations of the RRT exist [4], which provide better performance under particular constraints [5]. Unfortunately the considered modifications and extensions [4,5] do not provide means to consider uncertainties of the planning environment or the robotic systems itself.

### B. Planning under uncertainty

In mobile robotics one of the fundamental problems is robot pose (position and heading) estimation, which due to imperfection of the used sensors and physical implementation of the robot is always coupled with random noise. In practice it means that during the planning the actual pose of the robot is known only with some probability, what introduces an uncertainty into planning process.

Besides robot position estimation error another important source of uncertainty is obstacle positioning, which in real applications is obtained using noisy on-board or off-board sensors. Therefore some existing techniques have to be discussed.

The Particle RRT planner (pRRT) [6] uses particle cloud, where each particle represents a possible trajectory of a robotic system considering uncertainties of the robot. In order to reduce the number of trajectories particles are clustered according to their qualitative features thereby each configuration in the planning graph is defined by a set of possible configurations. This approach enables to reduce number of calculated path alternatives significantly but requires some distinctive features or parameters of the environment like rises or sharp slopes that allows to group alternatives into clusters. The pRRT changes some parameters like wheel friction of the robot in order to generate alternatives causing robot to perform differently. Environments like office premises do not provide these features limiting applicability of the method.

Adam Bry and Nicholas Roy in [7] propose to model future states of robot sensors according to the prior knowledge of the environment. It allows consideration of a set of path alternatives and estimate feasibility of the alternatives defined in terms of collision free motion. The sensor and motion models of the robotic system are complemented by uncertainties expressed using normal probability distribution. The proposed method enables to effectively combine robot motion model with probability distribution model, which provide more realistic view on path planning for particular system. As indicated in [8] the probability distribution of the robot position being an uncertain variable over time does not comply with restrictions of normal distribution and may produce distributions of different shapes that are hard to describe analytically.

An attempt to describe localization error distribution analytically is presented in [9], where exponential coordinates are applied. The authors show basic distribution propagation on differential drive robotic system with noisy sensor data in operation scenario, where robot motion is limited to straight or circular motion with known rotation radius. This method estimates the position distribution significantly better in comparison to the existing normal distribution based methods. However some effort is still required in order to develop a particle filter approach applicable for robot position estimation and to use it with all kinds of robot motion modes including backward motion and rotation around its own mass centre.

In order to deal with obstacle position uncertainties in [10] the Guided Cluster Sampling (GCS) is proposed providing a way of sampling based global planning, where the belief space is partitioned into subspaces. Thereby the search space is reduced significantly allowing application of the method for many practical robot motion planning problems. The method combines motion and sensing steps with different values providing means for guiding the search. A set of distinct obstacle and goal features such as obstacle corners in 3D or vertices in 2D space are used to partition the belief space.

While the method provides a straight forward approach to converge to globally optimal path it requires a set of well-defined distinctive features that are not always available with random shaped objects where some discretization like occupancy grid is applied.

The authors are unaware of published research proposing complete analytical methods that might accommodate both realistic robot and obstacles position error distribution description and its propagation at the same time. Therefore we propose an alternative heuristic approach that comprises a set of plan post processing steps. These steps are applied after RRT algorithm has produced its result – plan consisting of set of waypoints in a continuous environment and a set of arcs connecting waypoints thus forming a path.

## III. Description of the proposed post processing method

### A. Brief overview of the method

The proposed RRT plan post processing method comprises the following steps:

1. Removal of unnecessary plan waypoints – as shown later this step allows to reduce unnecessary heading changes of the robot;
2. Addition of waypoints around obstacles – within this step additional waypoints are added where the path leads close enough to obstacles thereby decreasing collision risk;
3. Point realignment – plan waypoints are realigned away from obstacles to reduce risk of collisions;
4. Smoothing of the plan – modified plan is smoothed using filtering eliminating sharp turning angles to improve motion speed;
5. Feasibility estimation – the step employs particle cloud simulation to estimate the collision possibility.

Each step is explained in details in the following sub-sections.

### B. Removal of unnecessary plan waypoints

RRT is not an optimal planner and therefore the typical planning result is a broken line that comprises a set of unnecessary manoeuvers like in Fig.2. While the main pose error is caused by the heading component, each additional change of the heading adds error to the final pose estimation.

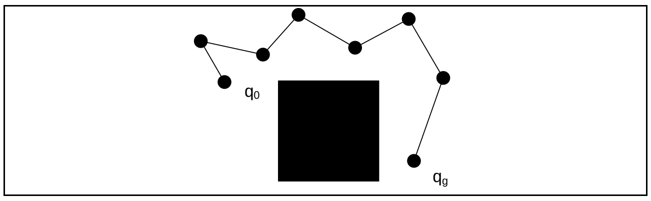

Fig. 2. Typical RRT result $q_0$ – start configuration, $q_g$ – goal configuration

Therefore it is important to reduce number of unnecessary turns especially with narrow angles that might require stopping the robot. Due to wheel slipping stopping and acceleration actions are additional sources of errors.

To reduce the number of unnecessary turns slowing the robot, a straight forward heuristic is applied to linearize the plan as long as it stays within the specified minimal distance from the obstacles.

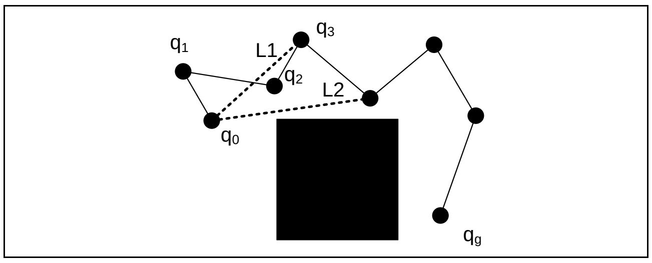

Fig. 3. Elimination of vertices

For example in Figure 3 distance from line L1 to the obstacle (black square) is acceptable while distance from line L2 is not. Therefore the line L1 enables to eliminate vertices q1 and q2. After going through the other vertices the resulting example path is depicted in Fig. 4, which has reduced the total number of vertices and the related manoeuvers of the robot.

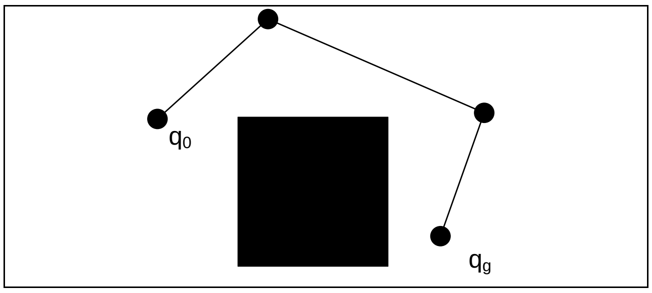

Fig. 4. The resulting example path

Assuming that the path is a list of waypoints this step is implemented by the following algorithm:

```
PROCEDURE waypoint_Elimination (PointList Path)
SET     newPath = $q_0$
SET     Current = $q_0$
SET     Last = $q_0$
FOR     each waypoint p in Path starting from $q_0$+1
        IF segment_is_safe(Current,p) = true THEN
                Last = p
        ELSE
                newPath = newPath + Last
                Current = p
        END IF
END FOR
RETURN newPath
```

Fig. 5. Unnecessary waypoint removal algorithm

In the algorithm depicted in Fig. 5 function *segment_is_safe* returns true if the appropriate plan segment is far enough from obstacles.

## C. Addition of waypoint around obstacles

Let us assume that in a particular example case the plan after removal of unnecessary waypoints is like the one depicted in the Fig. 6, where each square represents an occupied cell in occupancy grid.

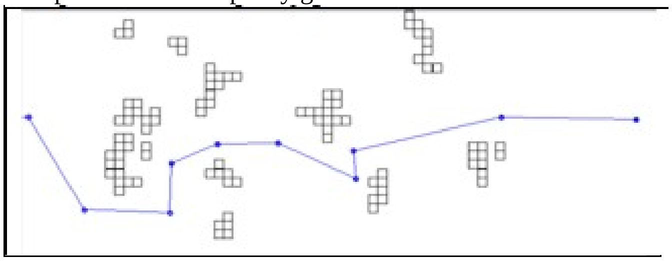

Fig. 6. Example plan after removal of unnecessary waypoints

In order to bend the plan around obstacles it is necessary to split the plan segments into smaller ones thereby providing a discrete representation of the segments that reduces the necessary computation. The heuristic behind this step is to add more waypoints where the segment closer to the obstacles and less where the segment is far enough from the obstacles.

```
PROCEDURE AddPoints (PointList Path)
SET     Current = 0
WHILE   Current < Length(Path) – 1
        IF (Dist_to_Obstacle(Path[Current], Path[Current + 1]) < Safe
        AND
        SegmentLength(Path[Current], Path[Current + 1])) > MinLength)
        OR
        SegmentLength(Path[Current], Path[Current + 1])) > MaxLength)
        THEN
                Instert_new_point(Current, Current + 1)
        ELSE
                Current = Current + 1
        END IF
END FOR
RETURN Path
```

Fig. 7. Adding new waypoints around obstacles

Within the algorithm in figure Fig.7 function *Distance_to_Obstacle* returns distance of a segment with ending waypoint from the list Path with indexes *Current* and *Current+1* to the closest obstacle. If the distance is less than constant threshold *Safe* then the procedure *Insert_new_point* splits the segment into two equal subsegments by inserting a new waypoint in the middle of the segment. This is done only if the segment is longer than a set threshold *MinLength* in order to avoid addition of too many waypoints. Constant MaxLength determines the maximum length of segment, which being exceeded causes collision threats after smoothing the plan (See section A). The result of this step is depicted in Fig. 8.

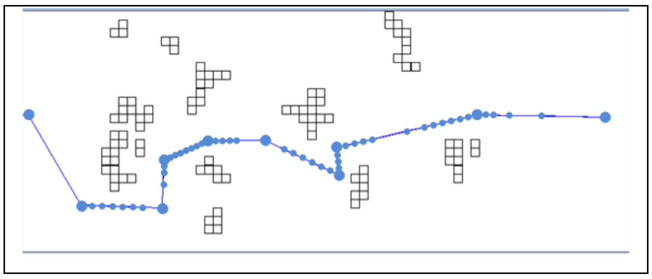

Fig. 8. Added waypoints around obstacles

As it is seen in Fig. 8 the closer the segment is to obstacle the higher the density of the new waypoints. Thereby the applied heuristic allows paying more attention to plan segments with higher collision possibility.

## *D. Point realignment*

Within this step we exploit idea of elastic strip planning framework presented in [11], which proposes to perceive plan as an elastic strip in dynamic environment. Obstacles in the environment are the source of forces being applied to the strip allowing bending it around the obstacles. The main advantage of the method is the possibility to respond to changing environments. Within the step we use this idea with few modifications:

1) Any changes of the plan may appear only at the plan waypoints. By this hard discretization assumption a significant amount of calculations is reduced;

2) The waypoints are treated only relatively to the nearby obstacles and previous waypoint. This modification reduces smoothness of the result but saves a lot of calculation efforts because it requires treating all of the plan waypoints only once.

3) The positive force that pushes a waypoint away from the obstacle remains as in the original elastic strip but the negative force is produced by the waypoint itself i.e. the waypoint tends to preserve its position. This also reduces smoothness of the result but requires consideration only of the obstacles and the given waypoint without considering inter-waypoint relations.

To simulate positive and negative forces we use abstraction of linear springs.

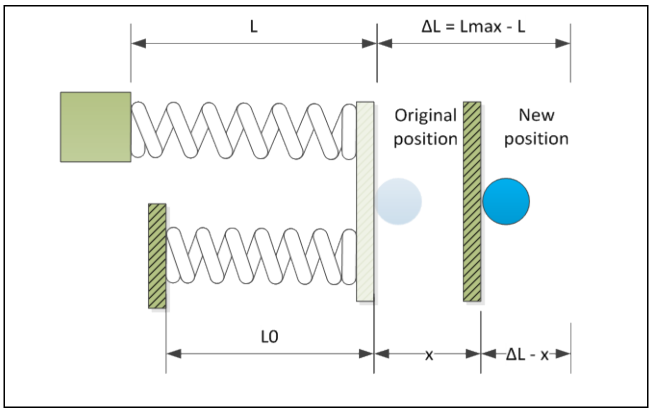

Fig. 9. Force simulation

One dimension case is depicted in Fig. 9. If the waypoint is closer than the maximum distance to obstacle $L_{max}$, where the spring force is 0, the spring is being compressed by a distance ΔL, producing a positive force that is calculated using formula (1).

$$F^+ = k^+ \cdot \Delta L \quad (1)$$

Therefore the positive force tends to move the waypoint away. At the same time the other spring while being pulled by the first one produces negative force that is calculated using formula (2).

$$F^- = k^- \cdot x \quad (2),$$

where x – displacement of the waypoint is not known. However, while both forces act against each other there is a waypoint of equilibrium, where sum of forces is 0. That is expressed in equation (3).

$$F^+ = F^- \xrightarrow{Follows} k^+ \cdot (\Delta L - x) = k^- \cdot x \quad (3)$$

Thereby calculation of x is straight forward:

$$x = \frac{k^+ \cdot \Delta L}{(k^+ + k^-)} \quad (4)$$

Equation (4) is applicable for one dimension, while the planner operates in 2D environment. Therefore the forces and appropriate calculations are decomposed into x and y components.

In case of many obstacles around the waypoint each obstacle is treated relatively to the waypoint. In Fig. 10 two obstacles produce forces in opposite directions. The resulting force is a vector sum of both. The actual number of considered obstacles depends on distance threshold providing a way to eliminate irrelevant ones.

Given the resulting $F^+$ for x and y force components it is possible to calculate the final displacement of the waypoint in both dimensions.

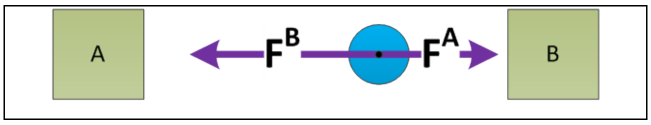

Fig. 10. Multiple obstacles

The result is depicted in figure Fig. 11, where the blue line is the original plan and the black line is the realigned one.

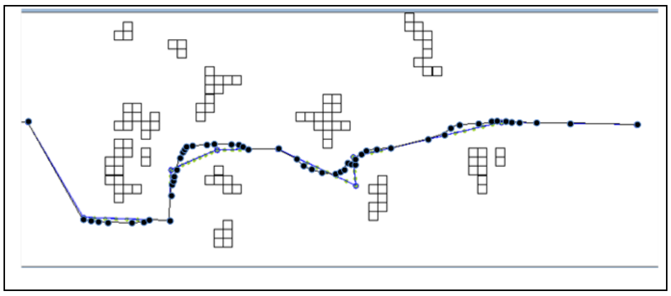

Fig. 11. Result of waypoint relialignment

## *E. Plan smoothing*

As it is seen in Fig.11 those segments that are close to the obstacles are bended around them thus making the robot motion safer. In order to ensure smooth motion of the robotic system the roughness has to be smoothed. As none of the dimensions (x or y) has constant discretization step we propose to use a Kalman filter that does not require such constraints being met [12].

In order to avoid collisions with obstacles we use a unity process model (5) that assumes constant waypoint positions.

$$\begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_{i-1} \\ Y_{i-1} \end{pmatrix} \qquad (5),$$

where $X_i$ and $Y_i$ – coordinates of i-th waypoint, $X_{i-1}$ and $Y_{i-1}$ – coordinates of previous waypoints.

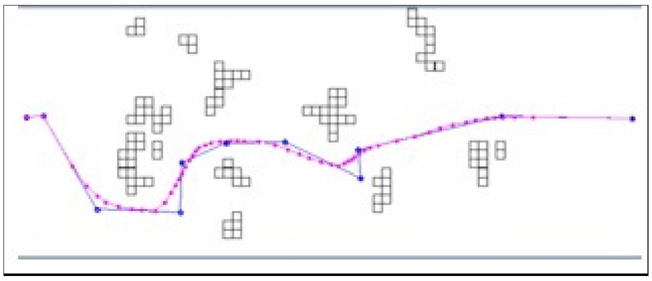
Fig. 12. Result of filtering step

The measurement in this context is the actual i-th waypoint coordinates. This assumption with equal weight of modelled and actual coordinates of i-th waypoint, in fact, turns the filter into an averaging or low-pass filter that is well known in signal processing. The result of the filtering step is shown in the Fig. 12, where the blue line is the original plan while the rose one is smoothed one.

At this point the bend plan already provides significant improvements to reduce collision possibility. The experimental results are provided in section IV.

## *F. Feasibility estimation*

As mentioned above the main goal of the plan post processing is the reduction of collision risk. The previous steps provide means to do it, but they do not provide any information about the actual collision risk. As demonstrated in [8] and [9] the actual robot position estimation analytically is possible only under hard assumptions on robot motion specifics. Therefore as an alternative we apply a set of particles where each particle simulates a robotic system with appropriate motion end error models. We use two different error models to acquire as adequate feasibility estimation as possible.

### *1) Signal transfer* model

As the first model we propose to use signal response function instead of the one offered by Linear time-invariant system theory [13]. The actual mechanism behind that is based on Laplace transformation of functions from time domain to Laplace domain. By definition Laplace transformation F(s) of time function f(t) is given by integral (6) [13]:

$$\mathcal{L}[\mathrm{f}(\mathrm{t})] = F(s) = \int_{-\infty}^{\infty} f(t)e^{-st}dt \quad , \qquad (6)$$

where $\mathcal{L}[\mathrm{f}(\mathrm{t})]$ – indicates Laplace transform of the time function f(t), s – complex Laplace variable of the form s = σ+jω, F(s) – the transformed function in Laplace domain.

In Laplace domain, signal response is defined by the ratio between output Y(s) and input X(s), while in time domain the multiplication corresponds to convolution [13]. This correspondence is indicated in equation (7):

$$y(t) = \boldsymbol{h}(t) * x(t) \Rightarrow Y(s) = H(s)X(s) \qquad (7)$$

The signal transfer function H(s) is acquired by using (6) and replacing f(t) by h(t). The convolution itself is defined by the equation (8) [13]:

$$y(t) = x(t) * \boldsymbol{h}(t) = \int_{-\infty}^{\infty} x(\tau)\boldsymbol{h}(t-\tau)d\tau \qquad (8)$$

For modelling the input signal x we use a simple step function that fully corresponds to the incoming robot control signals. Use of the impulse function in practice is difficult as it is infinitely short. Instead, we use a step function, which is defined by integral of impulse function.

$$u(t) = \int_{-\infty}^{t} \delta(t)dt, \qquad u'(t) = \delta(t) \qquad (9)$$

As shown in (9), the derivative of the step function is an impulse at the time instant t. This allows transformation of the equation (8):

$$y(t) = x(t) * \boldsymbol{h}(t) = \int_{-\infty}^{\infty} \delta(\tau)\boldsymbol{h}(t-\tau)d\tau \qquad (10)$$

Having expression (10) it is possible to calculate the actual value of y over the time.

The actual signal response function has to be acquired experimentally due to technical specification of the robot, its imperfection and environmental constraints, such as ground cover, traction of the surface, etc. In our case we model an indoor differential drive robot

In reality the input is defined by a vector $[L(t), R(t)]$, and the expected output from the model is $[v(t), \omega(t)]$, where v(t) – linear speed, and $\omega(t)$ – angular speed of the robot, $L(t), R(t)$ – left and right wheel speeds. Now, having in mind the definition of the model (7), it is possible to define the model in Laplace domain (11):

$$\begin{bmatrix} v(s) \\ \omega(s) \end{bmatrix} = \begin{bmatrix} H_{Lv}(s) & H_{Rv}(s) \\ H_{L\omega}(s) & H_{R\omega}(s) \end{bmatrix} \cdot \begin{bmatrix} L(s) \\ R(s) \end{bmatrix} \qquad (11)$$

Knowing that Laplace transformation of the matrix (12) is matrix (13) with the transformed elements and having the transformations of product and convolutions i.e. (7) and (8) it is possible to rewrite (11) in time domain as (14):

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad (12)$$

$$\mathcal{L}[x(t)] = \begin{bmatrix} \mathcal{L}[x_1(t)] \\ \mathcal{L}[x_2(t)] \end{bmatrix} = X(s)$$

(13)

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{d}{dt} u_{Lv}(t) * L(t) + \frac{d}{dt} u_{Rv}(t) * R(t) \\ \frac{d}{dt} u_{L\omega}(t) * L(t) + \frac{d}{dt} u_{R\omega}(t) * R(t) \end{bmatrix} \quad (14)$$

In the equation (14) step functions $u_{Lv}(t)$, $u_{Rv}(t)$, $u_{L\varphi'}(t)$ and $u_{R\varphi'}(t)$ can be obtained empirically by simply switching on each motor at a full speed and collecting the actual $v(t)$ and $\omega(t)$ data until the speed difference over a single time step achieves 0 meaning that the top speed is reached and transient processes are finished. Here we assume that both output functions are normally distributed random variables: and . Now it is possible to write the modelled kinematic model of differential drive robot:

$$\hat{X}_{t+1} = \begin{cases} \begin{bmatrix} x_t + v \cdot \Delta t \cdot \cos(\theta_t) \\ y_t + v \cdot \Delta t \cdot \sin(\theta_t) \\ \theta_t \end{bmatrix}, & \omega = 0 \\ \begin{bmatrix} x_t - R \cdot \sin(\theta_t) + R \cdot \sin(\theta_t + \omega \cdot \Delta t) \\ y_t + R \cdot \cos(\theta_t) - R \cdot \cos(\theta_t + \omega \cdot \Delta t) \\ \theta_t + \omega \cdot \Delta t \end{bmatrix}, & \omega \neq 0 \end{cases}$$

(15)

In (15) R = v/ω, $x_t$ and $y_t$ – position of the robot, $\hat{X}_t = [x_t \quad y_t \quad \theta_t]$ – x, y coordinates and heading of the robot at time instant t, ω – angular and v – linear velocities.

To estimate the variance and the final values of the model it is necessary to use algebra of random variables.

*2) Sensor error model*

Within this model we use the same kinematic model (15) but add noise to the velocities thus simulating noisy wheel speed readings.

$$R = \frac{l\,(v_r + v_l)}{2\,(v_r - v_l)}, \quad (16)$$

$$\omega = \frac{(v_r - v_l)}{l}, \quad (17)$$

In (16) and (17) both speed readings are normally distributed around their actual values: and .

The actual standard deviation values have to be obtained experimentally or from the manufacturer of the used sensors.

Both models allow generation of a set of possible positions of the robot. The particle motion is simulated using the same motion control algorithms as implemented on robotic system.

While particle cloud follows the plan we count the number of particles having collisions (see figure Fig. 13). Thereby at the end of simulation it is possible to count the percentage of failed and successful particles.

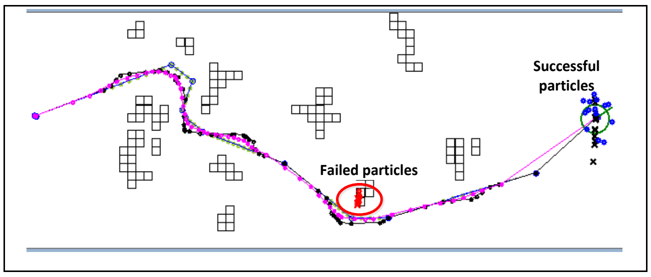

Fig. 13. Result feasibility estimation

These numbers provide data to estimate the possibility of collision if the number of particles is large enough.

## IV. Experimental verification of the proposed method

### A. *Experimental setup*

We have conducted two separate experiment sets: planning performance, which was focused on steps 1 to 4 of the proposed method, and plan feasibility tests, which was focusing on the step 5 of the proposed method.

*1) Plan performance experimental setup*

The main goal of this experiment set is to compare the proposed method with the RRT-Connect planning technique [14] (a slight modification of the original RRT) in terms of collision risk.

The first experiment set was conducted on a real robotic system that implements RRT-Connect, whose result is processed by the proposed method and then executed by the robotic system.

The robotic system comprises a differential drive chassis (iRobot Roomba vacuum cleaner [15]) and a laptop with added web camera for localization using artificial landmarks. The positioning method using artificial landmarks in details is discussed in [16].

Two different scenarios were used: the first one with full set of obstacles (see Fig. 14) and the second one with reduced number of obstacles (see Fig. 15).

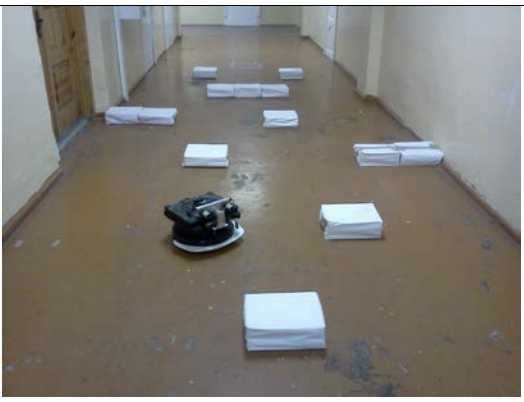

Fig. 14. The first scenario wih full set of experiments

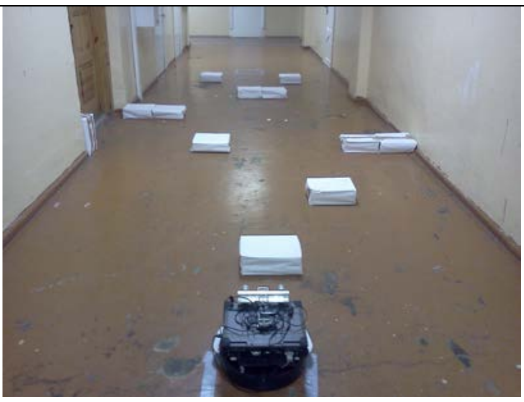

Fig. 15. The second scenario with reduced number of obstacles

The total area of the experimental setup is 240 x 1000 cm. Before each scenario the environment was explored by robotic system using all of the available sensors in order to decrease uncertainties of obstacles position. However due to positioning errors and limitations of the obstacle detection sensors of the used chassis the map still accommodates uncertainties. Tables I and II list the parameters of the planning and plan post-processing methods. The actual values were found by trials and errors before the experiments because we believe parameter estimation is beyond the scope of this paper.

TABLE I. RRT-CONNECT PARAMETERS

| Parameter | Value |
|---|---|
| Direction of planning | One-directional, Start-to-Goal |
| Probability of goal selection for the current direction of tree extension | 0.1 |
| Maximum Length of the current segment (mm) – the actual length is selected randomly | 500 |
| Goal met condition – max. distance from the goal (mm) | 50 |
| Safe point condition - min. distance to obstacle (mm) from robot center considered to be safe / collision free. | 200 |

TABLE II. PROPOSED POST-PROCESSING METHOD'S PARAMETERS

| Parameter | Value |
|---|---|
| Point addition distance (mm) – if a plan segment is closer to obstacle then an additional point is put in the middle of the segment this forming two new segments instead of the current. | 300 |
| Min. segment length (mm) – if the segment is shorter or equal to this distance then no additional points are put on the segment | 60 |
| Max distance from obstacles (mm) – the distance, at which (or shorter) the obstacles are taken into account for realignment forces calculations | 400 |
| Obstacle force constant (N/m) | 3 |
| Point resistance force constant (N/m) | 20 |

Within each scenario we on purpose selected plans produced by RRT-Connect that fail to be executed by robotic system without collisions with obstacles thereby proving the concept of the proposed method.
With each scenario we calculated success rate of the post-processed plan, which is the relative number of robot runs without collisions using the provided plan. Success rate in our case is percentage of the plans accomplished without collisions with obstacles.
Each plan was tested by robot runs with full positioning (see [16]) that included use of artificial landmarks keeping positioning error variance within an interval of 10 - 15 centimetres, and with odometric positioning only, where the positioning error variance is growing by each manoeuver throughout the plan execution.
Positioning with odometric position estimations is critical due to fact that artificial landmarks are not always in the sight of robot sensors thereby the position is estimated using odometric sensors only.
We tested three different plans: 1 with full set of obstacles and 2 with reduced set of obstacles, where each trial was composed of 20 robot runs under the same conditions.

### 2) *Plan feasibility estimation experimental setup*

Plan feasibility experiment goal is to determine if the particle cloud simulation produces the same success rate as the actual robot runs thereby providing means to forecast the possible collision probability before plan execution by robotic system.
The plan feasibility estimation experiments were conducted using software simulator, which implements the kinematic and error models discussed above. The planning and post-processing methods parameters are the same as listed in Table I and Table II. Table III lists sensor error variance values that were found empirically to match the used robotic system.

TABLE III. ERROR VARIANCES

| Parameter | Value |
|---|---|
| Left wheel encoder variance (mm) | 5,5 |
| Right wheel encoder variance (mm) | 5,5 |
| Angular speed variance (degrees/s) | 0,35 |
| Linear speed variance (m/s) | 0,08 |

Plans and maps produced by robotic system in the plan performance experiments were loaded into the simulation system thus providing consistency with the real situation.
Each of the produced plans was tested in 20 runs and the average percentage of successful particles was counted. Afterwards difference between success rates of simulation and actual robot runs was calculated.
During the experiment only odometric positioning was simulated because use of landmarks keeps positioning error in a certain interval preserving it from growth during the pan execution. As it is not possible to determine, when and where robot sensors will not be able to detect landmarks due to light condition changes or other unpredictable

circumstances, the worst scenario has to be considered, which in this case is pure odometric positioning.

### B. *Plan performance runs*

The RRT-Connect and the post-processed with the proposed method is depicted in Fig. 16, where white cells represent free space, black cells represent obstacles and grey cells represent unexplored space.

TABLE IV. FIRST RUN RESULTS

| Method | Success rate (%) |
|---|---|
| Post-processed | 100% |
| Post-processed only odometry | 15% |

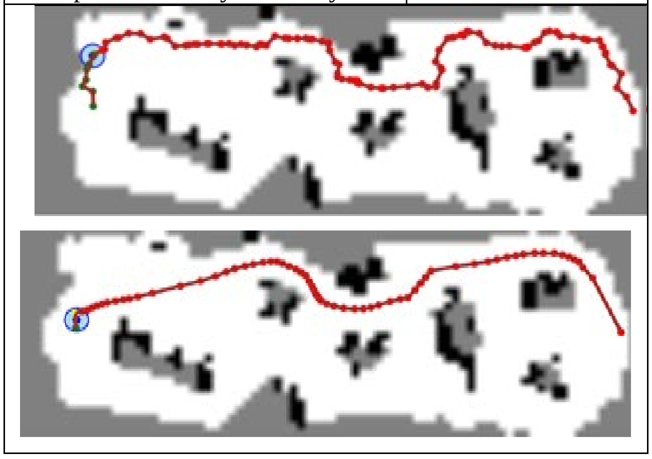
Fig. 16. First run RRT-Connect (on top) and post-processed plan (at the bottom)

The next runs were performed slightly reducing the number of obstacles. We removed those obstacles where the robotic system experienced the most number of collisions. The next scenario is depicted in figure Fig.15.
With the reduced complexity environment we performed two trials using the same approach. The results are depicted in figure Fig. 17 and Fig.18.

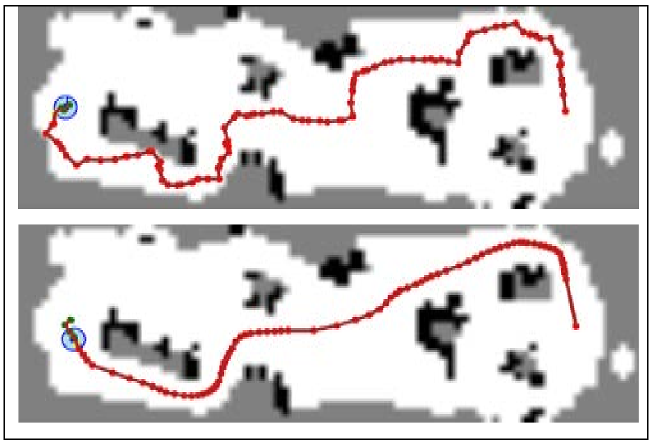
Fig. 17. Second run RRT-Connect (on top) and post-processed plan (at the bottom)

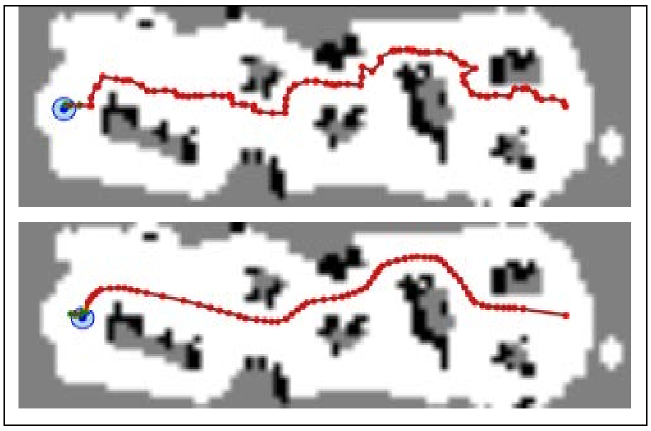
Fig. 18. Third run RRT-Connect (on top) and post-processed plan (at the bottom)

Again each plan was tested with 20 runs and the post-processed plan was tested both only with odometry and with full positioning.

TABLE V. SECOND TRIAL RESULTS

| Method | Success rate (%) |
|---|---|
| Post-processed | 100% |
| Post-processed only odometry | 55% |

TABLE VI. THIRD TRIAL RESULTS

| Method | Success rate (%) |
|---|---|
| Post-processed | 100% |
| Post-processed only odometry | 15% |

As shown in Table IV, Table V and Table VI the proposed method is capable to deliver a plan that the robotic system can accomplish without collisions in situations when RRT-Connect fails.

Here it has to be emphasized that RRT-Connect operates with a black-box step validation mechanism, which in our case is simple distance – to obstacle estimator. If the safe distance is increased the RRT would not be able to produce plan at all due to several bottle necks in the environment.

### C. *Plan feasibility tests*

In order to verify the plan feasibility each of the plans was simulated in both scenarios using previously described particles cloud with 100 particles. The actual noise variances shown in Table III where tailored empirically for the particular robotic system and remain constant during the tests on all plans. The results are summarized in the table VII. As explained above during the tests only pure odometry operation was tested because use of artificial landmarks keeps position variance in certain interval and does not allow considering the worst possible scenario.

TABLE VII. PLAN FEASIBILITY TESTS RESULTS

| Method | Success rate (%) | Difference to robot runs results (% |
|---|---|---|

| | | points) |
|---|---|---|
| First run | | |
| Post-processed only odometry | 31% | 16% |
| Second run | | |
| Post-processed only odometry | 46 | 4% |
| Third run | | |
| Post-processed only odometry | 22% | 7% |

As it is seen in the table VII, the feasibility estimation provides reasonable difference from the actual robot runs success rate, which is caused by idealized simulation model and does not take into account factors like surface roughness that might cause the deviation. This enables to forecast collision probability before the plan execution and avoid unwanted possible damage of the robotic system.

## V. Conclusions and future work

The proposed method provides a set of plan post-processing steps that allow reduction of robot collision possibility in environments where significant uncertainties of robots position estimation and obstacle mapping are present. The presented experimental tests show that the proposed plan post-processing method delivers plans that can be executed by robotic system without collisions in cases when the used RRT planning algorithm fails.

The waypoint removal and addition steps provide means to reduce overall amount of necessary calculations with varying discretization step depending on the distance from obstacles.

While the proposed method has been tested on plans generated by RRT-Connect, we see that it could be applied to other planning methods because it does not depend on actual planning methods used to generate the initial plan. Therefore we envision future study of the method application in conjunction with other planning techniques like combinatorial planning and its derivatives that are widely uses in mobile robotics.

The feasibility estimation provides data to make a decision of plan execution before the actual execution thus reducing possibility of robot damage during potentially dangerous plan execution.

For the future we see potential to apply the proposed method for multi-dimensional robots motion planning with persistence of high uncertainty.

## Acknowledgment

The research described in this paper is supported by funding of the ERDF Project “Development of technology for multi-agent robotic intelligent system”, project implementation contract No. 2010/0258/2DP/ 2.1.1.1.0/10/ APIA/VIAA/005.

## References

[1] Steven M. LaValle, Planning Algorithms, 2006, Cambridge University Press, ISBN 0-521-86205-1.

[2] J.C. Latombe, Robot motion planning, 1991, Springer, ISBN 9780792391296,

[3] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. In Technical Report No. 98-11. October 1998.

[4] A. Abbadi, R. Matousek, P. Minar, P. Soustek, RRTs Review and Options, computational Engineering in Systems Applications, Volume II, 2011, IAASAT Press, pp. 194-199, Iasi, Romania. ISBN: 978-1-61804-014-5, ISSN: 2223-9812.

[5] A.Abbadi, R.Matousek, RRTs Review and Statistical Analysis. International Journal of Mathematics and Computers in Simulation, 6, 2012.

[6] N.Melchior, R.Simmons, Particle RRT for Path Planning with Uncertainty, 2007 IEEE International Conference on Robotics and Automation, April, 2007, pp. 1617-1624.

[7] Adam Bry, Nicholas Roy. Rapidly-exploring Random Belief Trees for motion planning under uncertainty. In IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011. pages 723-730, IEEE, 2011.

[8] Ioannis Rekleitis. A Particle Filter Tutorial for Mobile Robot Localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, Montreal, Quebec, Canada, 2004.

[9] A.Long, K.Wolfe, M.Mashner, S.Gregory, The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates. In: Robotics: Science and Systems, 2012.

[10] H. Kurniawati, T. Bandyopadhyay, and N.M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. In Proc. Robotics: Science & Systems, 2011.

[11] Oliver Brock and Oussama Khatib Elastic Strips: A Framework for Motion Generation in Human Environments. International Journal of Robotics Research 21(12):1031-1052, 2002.

[12] H.B. Mitchell Data Fusion: Concepts and Ideas, Springer – Verlag, 2007, ISBN 9783540714637,281 pages.

[13] C.L.Phillips, J.M.Parr Signals, Systems and Transforms: 4th edition, Pearson Education, 2008, 795 pages

[14] J.J.Kuffner,S.M.LaValle, RRT-Connect: An Efficient Approach to Single-Query Path Planning, Robotics and Automation, Proceedings. ICRA '00. IEEE International Conference on on Robotics and Automation (Volume:2 ), pp 995 – 1001, 2000, ISSN 1050-4729

[15] http://www.irobot.com/us/robots/home/roomba.aspx cited: 16.06.2013.

[16] Agris Nikitenko**,** Aleksis Liekna, Martins Ekmanis, Guntis Kulikovskis, Ilze Andersone, Single robot localization approach for indoor robotic systems integrating odometry and artificial landmarks, Scientific Proceedings of Riga Technical University: 5th series “Computer science, Applied Computer Systems”, Riga, RTU Publishing, 2013, ISSN 22558683.