2008 IEEE International Conference on Robotics and Automation Pasadena, CA, USA, May 19-23, 2008

Replanning with Uncertainty in Position: Sensor Updates vs. Prior Map Updates

Juan P. Gonzalez and Anthony Stentz, *Member, IEEE*

Abstract— **This paper presents two new approaches to planning with uncertainty in position that achieve better performance than existing techniques and that are able to incorporate changes in the environment in near real-time. Both approaches reuse previous searches and replan when changes in the environment are detected.**

The first approach, called *replanning with prior map updates,* assumes that changes in the prior map originate from the same source as the original prior map. Therefore, the updates are registered with the existing map, but not with the position of the robot. The resulting path after applying the updates is the same as if the updates had been present in the original prior map.

The second approach, called *replanning with sensor updates*, assumes that changes in the prior map originate from on-board sensors. Therefore, the updates are registered with the robot, but not with the existing map. The resulting path after applying the updates is not the same path that would be found if the updates had taken place in the original prior map.

***Replanning with prior map updates* achieves a speed-up to one order of magnitude with respect to forward planning from scratch, while *replanning with sensor updates* achieves a speed-up of almost two orders of magnitude.**

I. Introduction

Planning with uncertainty in position is a computationally intensive and often intractable problem. Even the most efficient approaches take tens of seconds or minutes to find a path in large environments, which is appropriate only for off-line planning. In practice, however, off-line planners are of limited use as prior maps are imperfect and obstacles such as cars and other dynamic objects are not usually represented in prior maps.

This paper presents two new approaches to planning with uncertainty in position that achieve better performance than existing techniques and that are able to incorporate changes in the environment in near real-time. Both approaches reuse previous searches and replan when changes in the environment are detected.

The first approach, called *replanning with prior map updates,* assumes that changes in the prior map originate from the same source as the original prior map. Therefore, the updates are registered with the existing map, but not with the position of the robot. The resulting path after applying the updates is the same as if the updates had been present in the original prior map.

The second approach, called *replanning with sensor updates*, assumes that changes in the prior map originate from on-board sensors. Therefore, the updates are registered with the robot, but not with the existing map. The resulting path after applying the updates is not the same path that would be found if the updates had taken place in the original prior map.

II. Related Work

Existing approaches to planning with uncertainty in position for outdoor environments take from tens of seconds to hours to calculate paths in large environments. Most are also limited to indoor applications where the world can be described as FREE or OBSTACLE, and where the search space is significantly reduced [2][3][7][8][9][10].

In the field of Partially Observable Markov Decision Processes (POMDPs), the problem of planning with uncertainty in position has been frequently addressed. However, most algorithms become computationally intractable when dealing with worlds with a large number of states. Only Roy and Thrun [13] have solved the problem of finding optimal paths for large, continuous-cost worlds in the presence of uncertainty. This approach requires pre-processing of all the states in the search space, which later allows for very fast planning. However, the approach does not handle changes in the environments, and the total planning time (including the pre-processing stage) can take from several minutes to a few hours [14].

In classical path planning for outdoor environments only Hait *et al* [6] and Gonzalez and Stentz [4][5] have proposed planners that can handle the larger environments and continuous cost representation required. While these planners are much faster than the POMDP-based planners, they are still too slow to be used as online planners in worlds greater than 100x100 cells.

The research presented here extends the work of Gonzalez and Stentz [5] by adding replanning capabilities and allowing for sensor data to be incorporated into the planning process.

III. Problem Statement

The problem we are trying to solve is navigating autonomously in an outdoor environment without GPS

Manuscript received September 14, 2007. This work was sponsored by the U.S. Army Research Laboratory under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012). The views and conclusions contained in this document do not represent the official policies or endorsements of the U.S. Government.

J.P. Gonzalez is with General Dynamics Robotic Systems and with the Robotics Institute at Carnegie Mellon University, Pittsburgh PA, 15213 (e-mail: jpgonzal@cmu.edu).

A. Stentz is with the Robotics Institute at Carnegie Mellon University, Pittsburgh PA, 15213 (e-mail: axs@cmu.edu).

through the use of high resolution prior maps and a good dead-reckoning system. We assume that the initial position of the robot is known within a few meters, and the initial heading is known within a few degrees. As the robot moves towards the goal, its estimates of the traversal cost for some areas of the prior map will be updated with data from external sources or from the onboard sensors.

We assume a high-resolution map that allows the identification of landmarks and the approximate estimation of terrain types by automatic or manual methods. We assume that landmarks can be reliably identified in the prior map, and with the onboard sensors of the robot. We also assume that landmarks may not be unique

The resulting path should minimize the expected value of the objective function along the path, while ensuring that the uncertainty in the position of the robot does not compromise its safety or the reachability of the goal.

IV. Planning with Uncertainty in Position

The approach presented here extends the planner with uncertainty in position (PUP) presented in [5], which takes advantage of the low drift rate in the inertial navigation system of many outdoor mobile robots. The planner uses an isometric Gaussian distribution to model position uncertainty and uses deterministic search to efficiently find paths that minimize expected cost while considering uncertainty in position. A linear error propagation model is used, which assumes that the dominant term in the uncertainty propagation is the error in the initial heading.

The high-resolution map is translated into a cost map, in which the value of each cell corresponds to the cost of traveling from the center of the cell to its nearest edge. Non-traversable areas are assigned infinite cost and considered obstacles. This map is often called a *prior map*.

As in [5] we use *unique detection regions* to disambiguate landmarks. *Unique detection regions* are areas in the map in which a non-unique landmarks can be uniquely identified for a given detection range *R*.

A. State Space Representation

The probability density function (pdf) of the error is modeled as a Gaussian distribution, centered at the most likely location of the robot at step *k*:

$$\mathbf{q}_k = (x_k, y_k)$$

$$\mathbf{q}_k : N(\mu_k, \sigma_k) \tag{1}$$

$$p_{\mu_k,\varepsilon_k}(\mathbf{q}_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2}\frac{(\mathbf{q}_k-\mu_k)^T(\mathbf{q}_k-\mu_k)}{\sigma_k^2}}$$

where $\mu_k = (\mu_{x_k}, \mu_{y_k})$ is the most likely location of the robot at step *k*, and $\sigma_k = \sigma_{x_k} = \sigma_{y_k}$ is the standard deviation of the distribution at step *k*.

Let us define:

$$\varepsilon_k = 2 \cdot \sigma_k \tag{2}$$

We can then model the boundary of the uncertainty region as a disk centered at μ_k with radius ε_k. This model is a conservative estimate of the true error propagation model and, depending on the type of error that is dominant in the system, can provide an accurate approximation of the true model.

Under these assumptions, the augmented state vector

$$\mathbf{r} = (\mu, \varepsilon) \tag{3}$$

defines a 3-D configuration space, which is also a complete *belief space* [1].

In the cost map, the cost C_o of a cell $\mathbf{q}$ is defined as the cost to travel from the center of the cell to its nearest edge. We extend the idea of this 2-D cost map into the 3-D configuration space by defining the cost to move from the center of the 3-D cell $\mathbf{r}$ to its nearest edge. This cost can be expressed as:

$$C_{\mathbf{r}}(\mathbf{r}_k) = C_{\mathbf{r}}(\mu_k, \varepsilon_k) = \sum_i C_o(\mathbf{q}_i) p_{\mu_k,\varepsilon_k}(\mathbf{q}_i) \tag{4}$$

where $C_{\mathbf{o}}(\mathbf{q}_i)$ is the deterministic traversal cost as defined by the 2-D cost map at location $\mathbf{q}_i$.

B. Uncertainty Propagation

1) Outside of Unique Detection Regions

Outside of *unique detection regions* the position estimate of the robot is calculated using dead-reckoning. For traverses up to a few kilometers and with a good dead-reckoning system, the dominant term in the error propagation is the error in the initial heading, which increases linearly with distance traveled. We therefore use the following model to propagate uncertainty:

$$\varepsilon_k = \varepsilon_{k-1} + \alpha_u d(\mu_{k-1}, \mu_k) \tag{5}$$

where α_u is the uncertainty accrued per unit of distance traveled, μ_{k-1} is the previous position along the path, ε_{k-1} is the uncertainty at the previous position, and $d(\mu_{k-1}, \mu_k)$ is the distance between the two adjacent path locations μ_{k-1} and μ_k. The uncertainty rate α_u is typically between 0.01 and 0.1 (1% to 10%) of distance traveled.

2) Inside Unique Detection Regions

If all the possible locations for a configuration $\mathbf{r}_k$ are inside a *unique detection* region, we can guarantee that the feature that created the region can be detected, and that no other features will be visible within the field of view of the robot.

For practical purposes we make the simplifying assumption that the disk with radius $\varepsilon_k = 2 \cdot \sigma_k$ completely contains all possible locations on *(x,y)* of a given configuration $\mathbf{r}_k = (\mu_k, \varepsilon_k)$. Therefore, if the disk of radius ε_k centered at μ_k is completely contained within a *unique detection* region *i,* we assume that the configuration $\mathbf{r}_k$ is inside the *unique detection region.* As such, we can guarantee that feature *i* will be detected and assume that the uncertainty ε_k will be reduced to a small amount δ.

C. Using Deterministic Search to Plan with Uncertainty in Position

The belief space and 3-D cost map defined above define a graph with positive traversal costs. If the transitions between nodes are deterministic, we can use deterministic search to find the lowest cost path between any two points in the graph.

We use the following assumptions to ensure that the transitions between nodes are deterministic. In areas outside *unique detection regions*, planning takes place without sensing landmarks, and can be modeled as deterministic transitions in belief space. If landmarks can be reliably detected, then the areas inside *unique detection regions* can also be modeled as deterministic transitions, as the detection of landmarks is guaranteed. In the transitional areas that are not completely contained within *unique detection regions* the detection of landmarks cannot be either guaranteed or ruled out. However, by assuming that landmarks will only be detected when the uncertainty contour is completely contained within the detection region, we can still model these regions in a deterministic fashion, at the expense of having an overly conservative approach.

We search this graph using a modified version of A* in 3-D in which the successors of each state are calculated only in a 2-D plane, and state dominance is used to prune unnecessary states. Fig 1 shows an example of using PUP to find a path in a sample world. Notice how the planner avoids high cost regions and localizes only when needed.

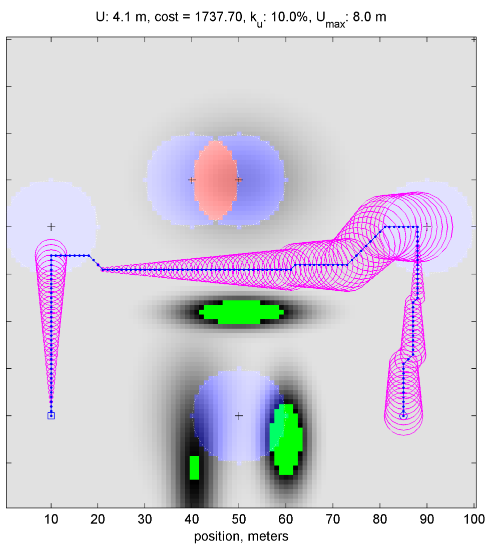

Fig 1. Path planned using PUP. Lighter regions in the cost map represent lower cost, and darker regions represent higher cost. Green areas are non-traversable obstacles. The blue path corresponds to the mean of the expected path, and the circles around each point are the uncertainty at each step along the path. The blue circular regions represent the *unique detection region* region of each landmark

V. Proposed Approach

We propose an approach that implements replanning within the Planning with Uncertainty in Position approach described above. RPUP uses D*-lite [11][12] to incorporate changes in the environment by reusing previous search results that minimize the changes in the search graph. However, there are significant challenges when implementing such approach.

The main limitation is that many of the performance gains in PUP come from using forward search and state dominance. However, when replanning for goal-directed navigation, it is much more efficient to plan backward from the goal than forward from the start. While in forward search the uncertainty at the beginning of the search (the start location) is known, in backward search the uncertainty at the beginning of the search (the goal location) is unknown. Instead of having a single start state with known uncertainty, all possible uncertainty values at the goal have to be considered in the search. Furthermore, most of the search will take place near the goal, where the uncertainty is higher (as opposed to planning forward, in which case the uncertainty near the start is usually lower).

We propose two ways to incorporate updates in the prior map, depending on whether the updates are registered with the prior map or with the robot. Each type of update requires a somewhat different replanning approach and results in different paths and performance.

A. Prior Map Updates

This approach assumes that the updates in the prior map are provided by a source that is well registered with the original map. For example, if the original map is a satellite image, updates to this map will also come from a satellite image that is well registered to the original satellite image. These updates, like the initial prior map, are not perfectly registered to the position of the robot.

Because *prior map updates* are registered with the prior map, they can be directly applied to the prior map. However, when the prior map is updated, all the cells in the 3-D configuration space that are within 2σ of the updated prior map cell need to be updated as well. Changes in the prior map $C_\mathbf{o}$ at location x_k, y_k require changing $C_\mathbf{r}(\mu_x, \mu_y, \varepsilon)$ at all locations such that

$$(\mu_x - x_k)^2 + (\mu_y - y_k)^2 \le \varepsilon^2 \tag{6}$$

for all values of ε.

Paths found by replanning with prior map updates are equivalent to the paths that would have been found if the new information had been present in the original prior map.

Fig 2 shows an example of RPUP with prior map updates. The path shown is the resulting path after replanning the path from Fig 1 because a prior map update is received showing an obstacle to the left of the workspace.

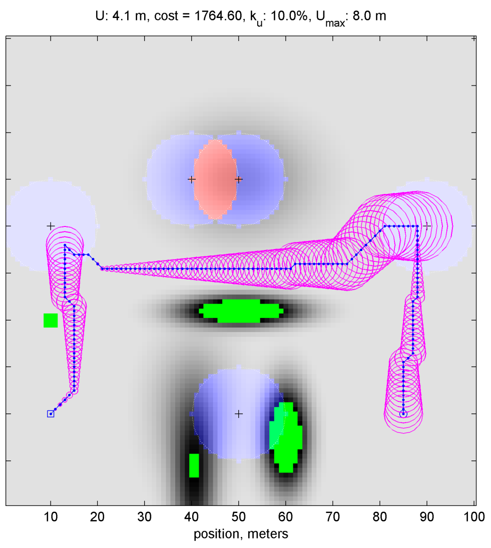

Fig 2. Path updated and re-planned because of a prior map obstacle.

B. Sensor Updates

This approach assumes that the updates in the prior map are provided by a source that is well registered with the position of the robot such as updates provided by the onboard sensors in the robot. The position of the robot with respect to these updates is known, but the position of these updates in the prior map is not.

These updates are represented in a separate sensor layer, which is combined with the prior map as follows:

$$C_{TE}(\mu_x,\mu_y,\varepsilon) = \sum_{\forall i}\Big(\alpha C_s(x_i,y_i)+(1-\alpha)C_o(x_i,y_i)\Big)p_{\mu,\varepsilon}(x_i,y_i) \tag{7}$$

where C_s is the sensor update, and $\alpha \in [0,1]$ is a linear term that determines the importance of the sensor data with respect to the prior data. Since the sensor data is assumed to be perfectly registered with the position of the robot, its cost does not depend on the probability distribution of the position of the robot, and (7) can be simplified as follows:

$$C_{TE}(\mu_x,\mu_y,\varepsilon) = \alpha C_s(\mu_x,\mu_y) + \sum_{\forall i}\Big((1-\alpha)C_o(x_i,y_i)\Big)p_{\mu,\varepsilon}(x_i,y_i) \tag{8}$$

$$C_{TE}(\mu_x,\mu_y,\varepsilon) = \alpha C_s(\mu_x,\mu_y) + (1-\alpha)C_{\mathbf{r}}(\mu_x,\mu_y,\varepsilon)$$

Using this update model, when a change at location (μ_x,μ_y) is detected, the combined map C_{TE} is updated by recalculating $C_{TE}(\mu_x,\mu_y,\varepsilon)$ for all ε values at (μ_x,μ_y). This change affects significantly fewer cells than prior map updates, since neither the prior map C_o nor the configuration space map $C_{\mathbf{r}}$ change when sensor updates are received.

The parameter α can have different interpretations, but in general reflects the confidence on the sensor data. One possible approach is to use sensor fusion assuming that both the prior cost and the sensor updates are normally distributed with means C_o and C_s, and variances (or confidences) σ_o^2 and σ_s^2. In such case

$$\alpha = \frac{\frac{1}{\sigma_s^2}}{\frac{1}{\sigma_s^2}+\frac{1}{\sigma_o^2}} \tag{9}$$

If the confidence in the sensor information is much higher than the confidence in the prior map, $\sigma_s \ll \sigma_o$ and $\alpha = 1$, but only for cells with sensor information. For cells that have not received sensor data $\sigma_s \to \infty$, therefore:

$$\alpha = \begin{cases} 1 & \textit{for cells with sensor data} \\ 0 & \textit{otherwise} \end{cases} \tag{10}$$

Paths found by replanning with sensor updates are not equivalent to the paths that would have been found if the new information had been present in the original prior map, because prior map information is not registered to the robot and sensor information is.

Fig 3 shows the resulting path after replanning the path from Fig 1 because a sensor update is received showing an obstacle to the left of the workspace. Notice how only the mean of the path avoids the obstacle, not the 2σ uncertainty contour.

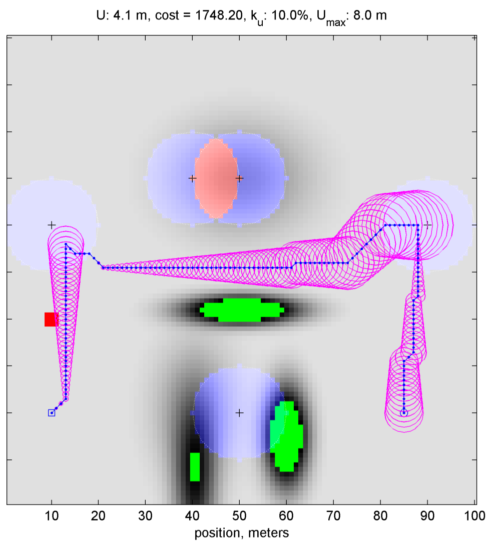

Fig 3. Path updated and re-planned because of a sensor obstacle

C. Discussion

The approach proposed here handles two of the most common cases for data updates with uncertainty in position for mobile robot navigation. However, there are scenarios that fall between prior map updates and sensor updates which are not properly handled. One such scenario is if the robot were to revisit a previously sensed area. In this case, the uncertainty with respect to the previously sensed data is no longer negligible. The uncertainty in the old sensed data is not the same uncertainty as that of the prior map either.

The correct way of dealing with old sensed data would be to keep track of the uncertainty of the robot with respect to each data point and to update this uncertainty as the robot moves. However this would be extremely inefficient, as all data points would change every time the robot moves.

Because the robot is assumed to be moving towards a goal, the scenario described above is not as critical as it would be for other tasks. In tasks that require sensed areas to be frequently revisited it may be possible to discard sensed data after a given time or after some distance has been traveled. This would maintain the consistency of the data while keeping updates efficient.

VI. Results

A. *Performance*

In order to measure the performance gains achieved by using replanning vs. planning from scratch the following experiment was performed. 50 different worlds were randomly generated using a fractal world generator, varying in size from 100x100 to 1000x1000 pixels. The number of uncertainty levels was kept fixed at 100. In each of these worlds the prior map used was a low resolution version of each world. Then the robot's motion along the path was simulated and higher resolution data was obtained around the robot in a 20x20 cells area by retrieving the original, high-resolution map (see Fig 4). These data updates were incorporated into the plan as prior map updates (registered to the original map) or sensor updates (registered to the robot). The planner used for planning from scratch was the one described in [5], which is a very efficient forward planner with uncertainty in position. The simulations were performed on an Intel(R) Xeon(TM) CPU running at 3.80GHz, with 4GB of memory. While some parts of the algorithm could easily be run in parallel, the simulations only use one of the four processors present. Additional improvements in the run-time of the algorithm could be achieved by taking advantage of the additional processors available.

When using prior map updates, the performance gains were only noticeable for worlds larger than 500x500, and continuously improving for larger worlds. The best result was at 1000x1000, where planning from scratch took 50 seconds on average, while replanning took approximately 7.5 seconds (7.5 times faster). Fig 5 shows the average online planning times (top) and the speed-up factor for different world sizes (bottom).

Replanning with sensor updates produced significantly better performance gains. Performance improvements were significant for worlds of size 300x300 and higher, with greater improvements in larger worlds. The best result was at 1000x1000, where planning from scratch took 38 seconds on average, while replanning took approximately 0.6 seconds (62 times faster). Because replanning takes on average less than one second, this approach can be considered near real-time: the planner can be implemented online incorporating real-time updates from the onboard sensors in the robot. This solves some of the problems with off-line planners that are limited to local obstacle avoidance when executing the original path. Fig 6 shows the average the average online planning times (top) and the speed-up factor for different world sizes (bottom).

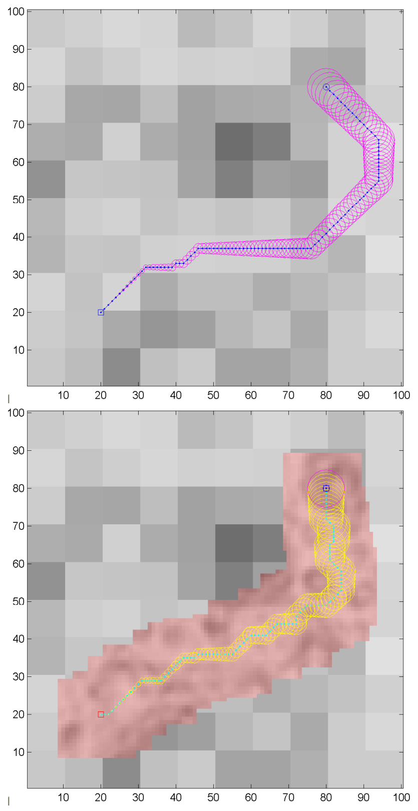

Fig 4. Initial path planned in low-resolution prior map (top) and final path followed by the robot after high-resolution sensor updates (bottom)

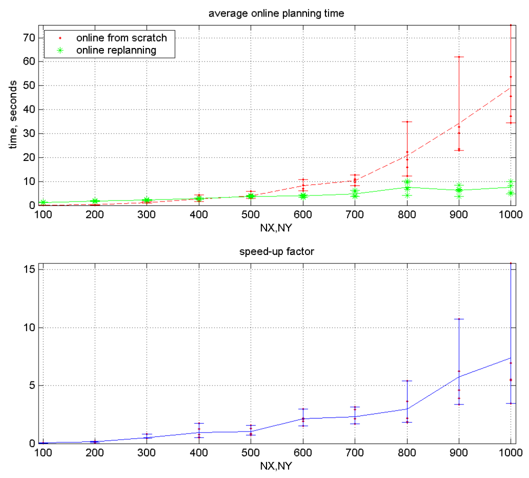

Fig 5. Average online planning time for forward search vs. re-planning with prior map updates (top). Average speed-up (bottom)

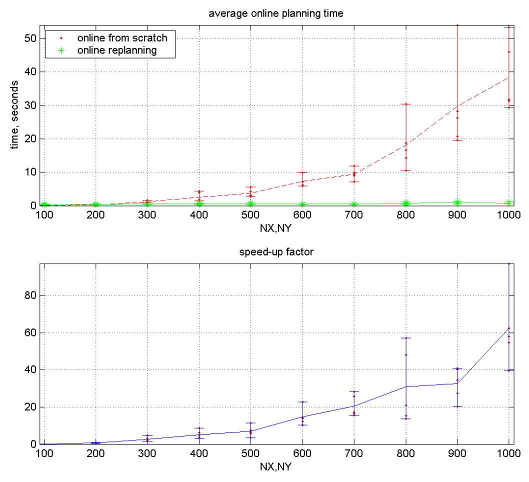

Fig 6. Average online planning time for forward search vs. re-planning with sensor updates (top). Average speed-up (bottom)

While re-planning is significantly faster with either prior map updates or sensor updates, it has some important disadvantages with respect to the original PUP implementation (forward search). While in forward PUP the expanded states defined a thin volume, in backward PUP with D*-Lite the search space is usually a full 3-D volume. The average thickness of the search volume in forward search is between 1 and 2 cells, while the average thickness for replanning with backward search is about 20 cells. Thus, the memory requirements for backward search are 10 to 20 times greater.

Additionally, the initial planning time when performing backward search is much longer than in forward search. Fig 7 shows the comparison between initial planning times for forward vs. backward search. The worst case is at 1000x1000, where the initial planning time for backward search is 1500 seconds (25 minutes), while for forward search is 120 seconds (2 minutes). While this is an important limiting factor, most parts of the initial planning can be computed off-line, therefore reducing the initial planning time significantly.

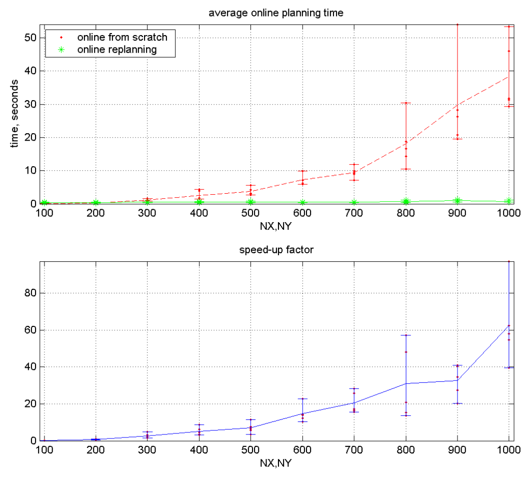

Fig 7. Average initial planning time for forward search vs. re-planning with sensor updates.

B. Field Tests

The following test setup was implemented on the e-gator autonomous vehicle shown in Fig 8. We used an aerial photograph covering a 200x250m region at 30 cm/pixel as the source for our prior data. From this photograph, we estimated a traversal cost map by training a Bayes classifier and manually annotating roads and buildings. Additionally, the electric poles used as landmarks were also identified in the image and were manually labeled.

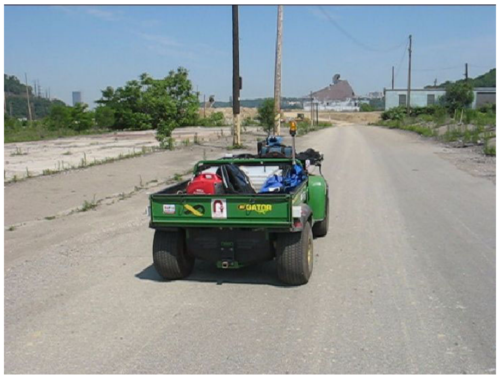

Fig 8. E-gator autonomous vehicle used for testing and electric poles used for localization at test site. The vehicle equipped with wheel encoders and a KVH E- core 1000 fiber-optic gyro for dead reckoning, and a tilting SICK ladar and onboard computing for navigation and obstacle detection

An initial path was planned based on this estimated cost map (prior map), the landmarks, and the estimated initial position and heading of the robot. Once the robot started moving, its sensors provided updates to the cost map (*sensor updates*), as well as detection information about the landmarks. A new path was replanned every second using the combination of the prior map, the sensor updates and the landmarks. Local obstacle avoidance was also performed at 10Hz.

As a reference, the same setup was also tested without PUP: using only the prior map (without landmarks), and only dead-reckoning for localization, while incorporating sensor updates into a standard planner (D*lite) that did not consider uncertainty.

Each approach was run 5 times, for a total of 10 runs. Fig 9 and Fig 10 show the paths planned and executed for one the runs in each setup, as well as the position reported by a WAAS-enabled GPS (for reference). Notice that the error between the robot's position estimate and the GPS position is much smaller when using RPUP and that only when using RPUP the sensor obstacles are correctly positioned in the global map. Notice also that the path executed in the top part Fig 9 is significantly different from the path planned. This is due to replanning around a blockage detected by the sensors.

Fig 11 Shows the error in the position of the robot with respect to the position reported by the GPS. Notice how this error is less than 4 meters at all times when using RPUP, and grows continuously when using only dead-reckoning.

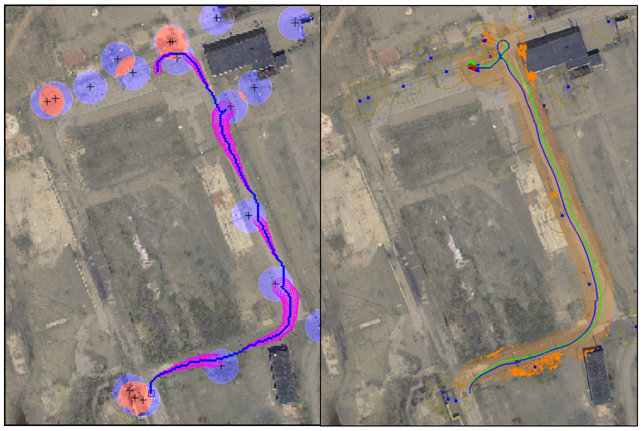

Fig 9. Path planned (left) and executed (right) without GPS using RPUP. In the left image the blue circles are *unique detection regions* generated by landmarks. In the right image, the blue line is the position estimate of the Kalman filter on the robot and the green line is the position reported by a WAAS differential GPS with accuracy of approximately 2 meters (for reference only). The orange regions represent sensor data, with obstacles shown as bright orange.

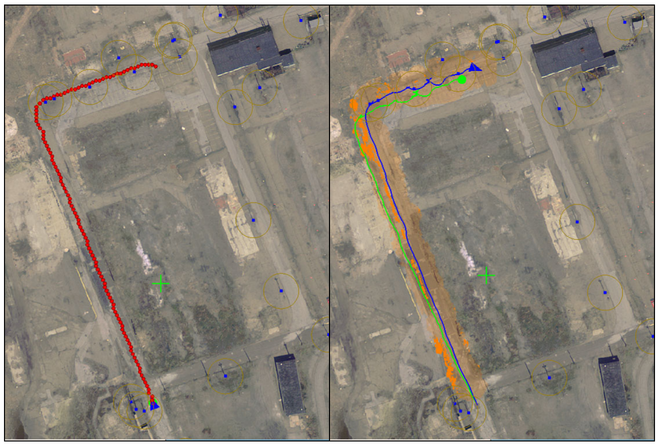

Fig 10. Path planned (left) and executed (right) without GPS, using a standard planner. In the right image, the blue line is the position estimate of the Kalman filter on the robot and the green line is the position reported by a WAAS differential GPS with accuracy of approximately 2 meters (for reference only). The orange regions represent sensor data, with obstacles shown as bright orange.

Fig 12 shows the replanning times for the same run using RPUP. Notice how the times are usually less than one second and always less than two seconds. The average replanning time was 0.22 seconds, with a standard deviation of 0.23 seconds.

While each of the 10 runs had slightly different characteristics, using RPUP always outperformed the regular planner. All five runs using RPUP were successfully completed, with uncertainties at the goal well below 5m. Only four of the runs with the regular planner were completed. The first run had to be aborted because the robot was unable to find a way around a group of rocks by the side of the road. Of the remaining runs, only the second run had an uncertainty at the goal below 5 meters. The other three runs had uncertainties of 10, 12 and 7 meters respectively.

Our approach was also tested in a larger, 600x700m area. Using the same test setup as in the previous experiments, a single 850m run was planned and successfully executed. Fig 13 shows the test site and the path executed.

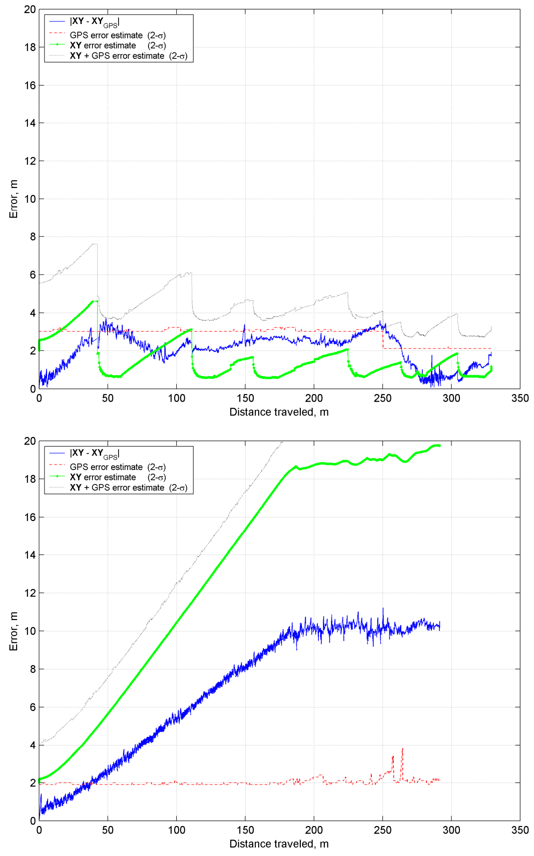

Fig 11. Position error when executing path using RPUP with landmarks (top) and when using a regular planner without landmarks (bottom). The blue solid line is the error in the position of the robot (difference between the robot's position estimate and the GPS). The red dashed line is the GPS's own error estimate. The green solid line is the robot's error estimate. The dotted gray line is the sum of the error estimates of the GPS and the robot.

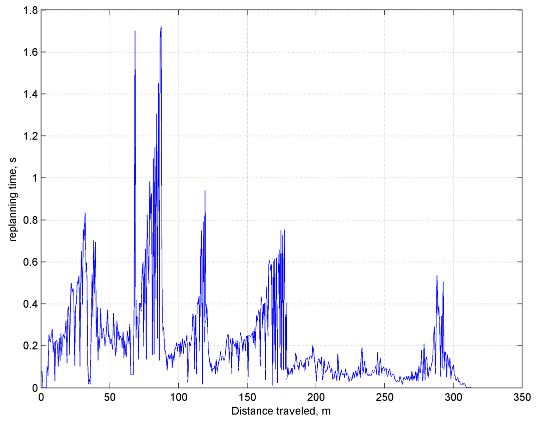

Fig 12. Replanning times for the run using RPUP. Notice how replanning times are usually less than 1 second, and always less than 2 seconds.

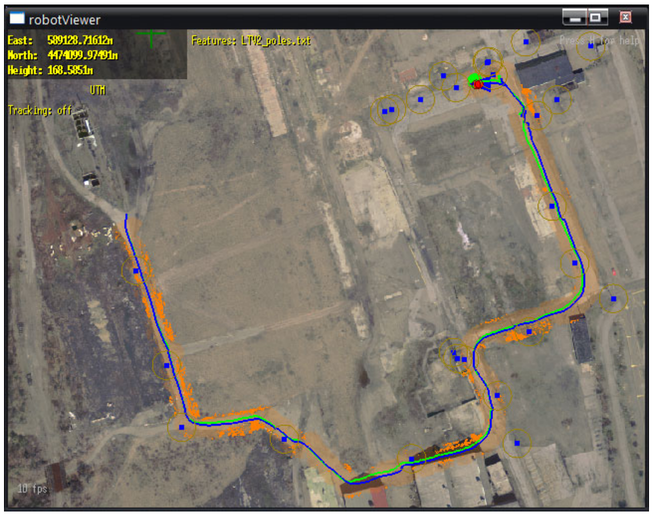

Fig 13. Long run using RPUP. The blue line is the position estimate of the Kalman filter on the robot and the green line is the position reported by a WAAS differential GPS with accuracy of approximately 2 meters (for reference only). The orange regions represent sensor data, with obstacles shown as bright orange. The total length of the run was 850 meters.

VII. Conclusions and Future Work

We have introduced two novel approaches to planning with uncertainty in position that enable a robot to plan paths that incorporate changes detected in the environment and to navigate with imperfect prior maps. These approaches implement D*-lite as part of the planning with uncertainty algorithm proposed in [5] and incorporate changes that are registered with either the prior map (*prior map updates*) or with the robot (*sensor updates*).

Replanning with prior map updates provides performance improvements with respect to the original PUP approach of almost one order of magnitude. While performance is highly dependent on the quality of the original map and the specific changes that take place in the world, the results presented here show a performance gain of up to 7.5 (with a 1000x1000 world).

Replanning with sensor updates provides significant performance gains with respect to the original PUP approach, with the most significant gains being obtained with 1000x1000 worlds. At this size, the average performance gain is 62. The average time to replan a path was about 0.6 seconds, enabling this approach to be used as an on-line planner for some applications. While this approach may still be too slow for other applications, it is a significant step towards real-time, on-line path planning with uncertainty in position.

We also presented results from several field tests confirming the advantages of using RPUP in GPS-denied environments, as well as the ability to replan in response to sensor updates. To the best of our knowledge, this is the first time that a planner that considers uncertainty in position for outdoor environments has been validated through experimental results.

While the approaches presented here do not consider all the possibilities for incorporating new data into an existing plan, they represent the most common cases for goal-directed navigation.

Future work includes evaluating the integration of sensor data with prior map data in more complex environments, as well as using more complex sensor models to perform the fusion of sensor and prior map data.

References

[1] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in Proceedings of the 6th International Conference on Artificial Intelligence in Planning Systems (AIPS), pp. 52-61, AAAI Press, 2000

[2] B. Bouilly. “Planification de Strategies de Deplacement Robuste pour Robot Mobile”. PhD thesis, Insitut National Polytechnique, Tolouse, France, 1997

[3] B. Bouilly, T. Siméon, and R. Alami. “A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty”. In Proc. of the IEEE Int. Conf. on Robotics and Automation, volume 2, pages 1327--1332, Nagoya (JP), May 1995.

[4] J.P. Gonzalez and A. Stentz, "Planning with Uncertainty in Position: An Optimal and Efficient Planner," Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '05), August, 2005.

[5] J.P. Gonzalez and A. Stentz, "Planning with Uncertainty in Position Using High-Resolution Maps," In Proceedings of the IEEE Int. Conference on Robotics & Automation (ICRA) 2007, April, 2007.

[6] A. Haït, T. Simeon, and M. Taïx, "Robust motion planning for rough terrain navigation".Published in IEEE Int. Conf. on Intelligent Robots and Systems Kyongju, Korea, 1999.

[7] Th. Fraichard and R. Mermond "Integrating Uncertainty And Landmarks In Path Planning For Car-Like Robots" Proc. IFAC Symp. on Intelligent Autonomous Vehicles March 25-27, 1998.

[8] J.C. Latombe, A. Lazanas, and S. Shekhar, "Robot Motion Planning with Uncertainty in Control and Sensing," Artificial Intelligence J., 52(1), 1991, pp. 1-47.

[9] J.C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers. 1990

[10] A. Lazanas, and J.C. Latombe, “Landmark-based robot navigation”. In Proc. 10th National Conf. on Artificial Intelligence (AAAI-92), 816--822. Cambridge, MA: AAAI Press/The MIT Press.

[11] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. Technical Report GITCOGSCI -2002.

[12] S. Koenig and M. Likhachev, "D*lite". Eighteenth national conference on Artificial intelligence, American Association for Artificial Intelligence, 476-483, 2002.

[13] N. Roy, and S. Thrun, “Coastal navigation with mobile robots”. In Advances in Neural Processing Systems 12, volume 12, pages 1043—1049, 1999.

[14] N. Roy, Department of Aeronautics and Astronautics, MIT. Private Conversation. September 1, 2004