To appear in Proc. 10th IEEE Intl. Symp. on Safety, Security, and Rescue Robotics (SSRR), 2012.

Multi-Objective Sensor-Based Replanning for a Car-Like Robot

D. K. Grady*, M. Moll*, C. Hegde†, A. C. Sankaranarayanan†, R. G. Baraniuk†, and L. E. Kavraki*
*Dept. of Computer Science, Rice University, Houston, TX 77005, USA. Email: {dkg1,mmoll,kavraki}@rice.edu
†Dept. of Elec. & Computer Engineering, Rice University, Houston, TX 77005, USA. Email: {ch3,saswin,richb}@rice.edu

Abstract — This paper studies a core problem in multi-objective mission planning for robots governed by differential constraints. The problem considered is the following. A car-like robot computes a plan to move from a start configuration to a goal region. The robot is equipped with a sensor that can alert it if an anomaly appears within some range while the robot is moving. In that case, the robot tries to deviate from its computed path and gather more information about the target without incurring considerable delays in fulfilling its primary mission, which is to move to its final destination. This problem is important in, e.g., surveillance, where inspection of possible threats needs to be balanced with completing a nominal route. The paper presents a simple and intuitive framework to study the trade-offs present in the above problem. Our work utilizes a state-of-the-art sampling-based planner, which employs both a high-level discrete guide and low-level planning. We show that modifications to the distance function used by the planner and to the weights that the planner employs to compute the high-level guide can help the robot react online to new secondary objectives that were unknown at the outset of the mission. The modifications are computed using information obtained from a conventional camera model. We find that for small percentage increases in path length, the robot can achieve significant gains in information about an unexpected target.

Keywords: *motion planning, navigation, sensor-based planning*

I. Introduction

A classic task for a robot is that of motion planning. Given a start state A and a destination region B, the problem is to find a continuous path that the robot can execute to reach B. The robot is in general governed by a set of differential constraints [1], [2]. In this paper we consider a variation of the problem above. The robot computes a plan to move from A to region B. However, the robot is equipped with a sensor that can alert it if an anomaly (a target of interest to be sensed) is detected within some range while the robot is moving. In that case, the robot tries to deviate from its computed path and gather more information about the target without incurring a big delay while fulfilling its primary mission, which is to move to destination B (see Figure 1). The paper develops a framework in which this problem can be studied. A powerful state-of-the art sampling based planner (SyCLoP [3]) is modified and used in a replanning loop. The generality of the planner allows the consideration of many realistic robots with complex dynamics. However, only car-like robots are considered in this paper.

The novelty of this work lies in the fact that the planner automatically computes deviations from the original path without asking the user to specify viewpoints and without forcing the robot to always go close to the point of interest. This is achieved by modifying a key step of the planner and providing a single parameter that can be easily tuned to control the trade-off between gathering information about the secondary target and the time required to complete the primary mission.

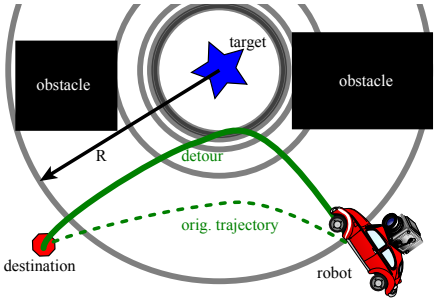

Fig. 1. A schematic representation of a car-like robot making a detour from a path towards its primary destination to opportunistically gather additional information about a secondary target (indicated by a blue star) once the presence of the latter has been detected at distance R. The concentric circles indicate isocontours of probability of gathering enough information to, e.g., classify a target. The probability decreases quadratically with the distance to the target with the sensor model used in this paper.

This work touches upon the broader topic of the need to balance multiple objectives mainly for mobile robots. It would be ideal if a planning algorithm manages this balancing process with minimal or no human intervention. Implementation on a physical platform would introduce additional complexity that we do not explore, and is well described in [4]. Applications exist in search and rescue operations, reconnaissance missions, maintenance of mobile wireless networks, mine clearing, and others. What complicates multi-objective planning in these domains is that some of the objectives may not be known beforehand, but are triggered by sensor readings (e.g., when anomalies are detected). It is challenging to balance the primary objectives stated at the start of the mission with secondary objectives as they arise. In this paper we consider moving a robot to a final destination as the primary task, and maximizing sensor information gain about a detected anomaly as the secondary task. A more formal definition of this scenario is presented in Section II. Although the problem is presented in the case of a simple robot, it also makes sense (and is more challenging) in a multi-robot setting.

Related Work: This paper brings together ideas from diverse topics in robotics, including kinodynamic motion planning, replanning, and view selection. For systems with second-order dynamics, such as the car-like robot considered in this work, it is hard to find solutions for motion planning problems [1], [2], [5], [6], but many practical approaches have been proposed. Sampling-based algorithms [6] have been successfully used for replanning for dynamic systems [7]–[9]. The typical approach in replanning is to use relatively short time periods called *planning cycles* to plan a robot's motion

for the next planning cycle while at the same time the robot is executing the plan computed in the previous cycle.

Sampling-based algorithms have also been combined with cost maps to find paths that minimize some cost function over paths (such as the amount of work) [10]. Cost maps typically consist of a grid decomposition of the workspace with a traversal cost associated with each grid region. Cost maps are very relevant to our work. They have been used successfully in [11] in combination with a rich set of precomputed maneuvers to effectively plan motions for a car in the DARPA Urban Challenge. In our work a generalization of cost maps will be used that not only depend on distance or path length, but also on sensor-derived information about possible viewpoints of a target. Unlike many other grid-based planning techniques, the completeness of the planner is not limited by the grid resolution.

Most of the work on viewpoint selection ignores whether a viewpoint is reachable for a given robot. This is problematic, since a robot may be asked to go to a position that is unreachable. At the same time, many other viewpoints that are almost as good may be easily reachable. In [12] the authors present an algorithm to find points with large visibility polygons at frontiers of explored areas of unknown environments. Reachability is considered, but viewpoint selection and path planning are decoupled and treated as separate problems. A similar approach has been presented in [13] for model reconstruction; here, the authors use a sampling-based planner to compute collision-free paths to the best viewpoints, but no differential constraints are considered. The approach in [14] combines the next best view (NBV) with cost maps that include secondary costs. In that work, a candidate path from the current location to the NBV is recursively deformed to obtain good views "on the way" to the best view, as opposed to directly constructing a sensor-based plan. Bayesian approaches often do not capture the motion constraints underlying the problem. In [15], a Bayesian sensor model computes a cost map and waypoints are selected. Unlike our work, the considered motion planner does not receive this cost information. It receives only the waypoints, and reachability is assumed. However, the proposed Bayesian sensor model could be used in our framework with minimal modification and this is in our future plans.

Overall approach: Our work relies on a sampling-based planner called SYCLOP [3]. It poses few constraints on the underlying dynamics of the robot. SYCLOP works by automatically defining a decomposition of the workspace, creating an adjacency/abstraction graph, and searching that graph for a high-level guide. A low-level planning layer computes the actual dynamically feasible paths and informs the upper layer for how to assign informative weights to the edges of the abstraction graph, so that new meaningful leads can be recomputed if needed. This is critical for (probabilistic) completeness. SYCLOP has been shown to perform well experimentally with a variety of widely-used sampling-based planners such as RRT [16] and EST [17]. In this paper, the weights of the abstraction graph are further influenced via cost factors that encode the presence of new information as explained in Section III. Our work does not explicitly utilize a cost *map*, but rather a cost *graph* between regions that form an arbitrary decomposition of the configuration space. The costs in our approach encode both the quality of potential viewpoints as well as the travel time required to reach them with a car-like robot.

Organization of this paper: In Section II we give a formal definition of the problem addressed in this paper as well as the models used for the robot's dynamics and its sensing capabilities, respectively. Section III describes our sensor-based replanning algorithm. In Section IV we present simulation results. In Section V we conclude with a brief discussion and describe directions for future research.

II. OPPORTUNISTIC PLANNING

Problem Statement: The problem addressed in this paper can be formulated as follows: given a car-like robot in an initial workspace pose A, compute a dynamically feasible path to region B while optimizing for both low path length and high information gain about an unexpected target, if the latter is detected while executing an initial trajectory that leads to B. Getting to B is considered the robot's primary objective, while collecting information about a target T is considered its secondary objective. Depending on the application, the information is used to, e.g., classify, recognize, or model the target. The robot is able to detect a target's position when it is within a distance R, but a stochastic model (described below) is used to define when sufficient information about the target has been obtained to, e.g., classify it. Due to differential constraints on the robot's motion, the robot cannot simply drive straight toward a good viewpoint for T, but needs to carefully plan a path that respects these constraints, avoids obstacles, gets close to T and quickly reaches B.

Because this work relies on the interaction between a sensing framework and a navigation framework, we need to carefully define several terms that we will use to describe our methods. The robot *discovers* the target T when it is within R of it. The robot *senses* the target when a sensor measurement of the target is deemed to be of high quality, as described in Section II. The mission is *complete* when the robot arrives at B. The mission is *successful* when it is *completed* and the robot has *sensed* every target that was *discovered*.

Robot Model: The robot used in this paper has second-order, car-like dynamics:

$$\dot{x} = v\cos\phi\cos\theta, \quad \dot{y} = v\cos\phi\sin\theta, \quad \dot{\theta} = v\sin\phi,$$
$$\dot{v} = u_0, \qquad\qquad \dot{\phi} = u_1,$$

where (x, y, θ) is the pose of the robot, v its velocity, ϕ its steering angle, and the control vector $u = (u_0, u_1)$ controls the acceleration and turning rate. The velocities and controls are bounded, so the robot cannot stop or change direction instantaneously. This makes navigation in environments with obstacles very challenging; the robot cannot rely on purely reactive behaviors and needs to plan ahead. While only car-like robots are considered in this paper, our planning algorithm

is applicable to robots with even more general differential constraints.

As discussed above, due to the complex dynamics, the robot cannot simply follow a potential based on some function of sensor payoff and distance to goal, but needs to *plan* to obtain control inputs that drive it through low-cost areas.

Sensor Model: This paper uses a simple model of a sensor to provide a proof of concept on the method, but our approach generalizes to other kinds of sensors. There are two types of sensor measurements, as indicated in Section II:

1) Presence and location of a discovered target within distance R from the sensor/robot, and
2) Determination if a high-quality sensor measurement of the target has been achieved.

Both measurements are obtained once every planning cycle as described in Section III. High-quality measurements (type 2) are more complicated because the modeled sensor parameters must be taken into account.

Our sensing model is similar to the models that can be found in field robotics research [18]. We model the event of acquiring a high-quality measurement as a Gaussian process: the robot could fail to detect a target even if, with a perfect sensor, it is in theory able to do so. This is more realistic than simply choosing a distance cut-off within which a high-quality measurement is always obtained.

We have used the parameters from an amateur-level digital still camera as our sensor. Specifically, we assume a sensor size of $23.6\text{mm} \times 15.7\text{mm}$ (APS-C standard), an image pixel size of 4592×3056, a focal length of 24mm, and a target pixel size of 100×100. Additionally, we need to choose a typical target size, which was arbitrarily set to 1m^2. We assume that the camera is pointed directly at the target once it is within discovery range. For comparison, the environment that the robot is operating in is $400\text{m} \times 400\text{m}$ and the robot is approximately $0.5\text{m} \times 0.25\text{m}$.

Once these parameters are defined, we define a high-quality sensing event (e.g., for classifying a target) as occurring when a sample from a Gaussian distribution centered at 0 has a magnitude greater than 1. The standard deviation σ of this Gaussian takes into account the distance r between the robot and the target as follows:

$$\sigma = \frac{4592}{r \cdot \frac{0.024}{0.0236}} \cdot \frac{3056}{r \cdot \frac{0.024}{0.0157}} \cdot \frac{1}{10000} \approx \frac{902}{r^2}$$

The first two terms describe the number of pixels that the modeled camera sensor will capture that are of the target. There are 4592 horizontal pixels, which is divided by r times the horizontal field of view of the lens. The horizontal field of view is computed as the focal length of 24mm divided by the horizontal sensor size of 23.6mm. The result of the first term is the horizontal size of the target in pixels for a single image frame. Vertical size is calculated similarly. The target is assumed to present a square visible region to our sensor to make this calculation easier to understand, however, this is certainly not a requirement. Once the total number of target pixels in a particular image frame is computed, we compare it to our set goal of $10,000$ target pixels. The factor of $10,000$ pixels is chosen using the following heuristic: suppose that the pixels on target can be robustly identified using foreground-background subtraction. Then, an affine-invariant feature extraction algorithm (e.g., SIFT [19]) can be employed on these pixels for robust target detection. For highly textured images, it is typical to obtain around 100 reliable SIFT feature descriptors in a 100×100 pixel image [20]. Therefore, at a range of approximately 30m, $\sigma = 1$, and the robot has a 32% chance to obtain a high-quality measurement of the target. When the robot gets closer, success probability increases as r^2, and of course decreases in a similar fashion as the robot moves farther away. There is a hard cutoff at R, assuming that if the robot cannot tell the location of the target, it cannot be expected to classify it either.

III. Algorithm

To build a solution framework, we extend a previously proposed sampling-based planner called SyCLoP [3], implemented in the Open Motion Planning Library (OMPL) [21]. The use of SyCLoP is critical to our solution because its structure allows information to flow between a high-level discrete planning layer and low-level continuous sampling-based planning layer. An illustration is offered in Figure 2. The discrete layer of SyCLoP can be used to encode information about coarse characteristics of the workspace (e.g., where the obstacles are) but also, as shown in this paper, other information such as the information gathered by sensors. A critical advantage of SyCLoP is that it incorporates a structured way to pass information from the discrete layer to the continuous layer and vice-versa, an advantage that boosts its performance as shown in [3]. In the scenario investigated in this paper, the bidirectional information flow enables the motion planner to take into account both reachability and sensor information at the same time. Our implementation of SyCLoP uses RRT [22] as the underlying continuous planner.

The inputs to our modified SyCLoP are a motion planning problem with dynamics (workspace obstacles, current and goal positions), and expected sensor informativeness of regions in the workspace supplied by the sensor model. SyCLoP and the sensor model operate in a replanning loop as explained in [9]. The duration of each iteration of the loop is called the planning cycle. During each cycle the robot executes the plan from the previous cycle, while it plans a new path to the destination with the sensor measurements passed to SyCLoP at the end of the previous planning cycle. The sensor model estimates help guide the robot to interesting areas as described below. The replanning loops continues until the mission is complete.

In the original SyCLoP, the discrete guide (or high-level plan as indicated in Figure 2) is recomputed at fixed small intervals to take into account new reachability information "discovered" by the analysis done by the continuous sampling-based motion planner. The discrete model is typically chosen to be a 2D grid decomposition of the workspace. A directed acyclic graph (DAG) is then constructed by the high-level discrete planner between adjacent regions of the decomposition, starting from the region that the robot is currently in. Edge

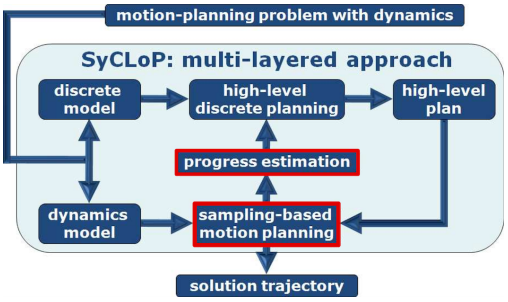

Fig. 2. Diagram illustrating the interaction between the modules in the SYCLOP planner. The highlighted modules have been modified by adding information about the expected sensor payoff.

weights are assigned to this graph of adjacent regions, starting with all equal values. These edge weights will be modified by the progress estimates that the sampling-based planner provides. A guide path (high-level plan) in the discrete model is computed. The sampling-based motion planning tree is mapped to the regions of the discrete model, and tree nodes in regions further along the guide path are more likely to be selected for a exploration by the sampling-based planner. This exploration runs RRT for k iterations, where random samples are only drawn from the subset of the state space that projects to the selected region. Starting from these samples, RRT operates without modification. The same discrete guide path is sampled for a region to expand from in this manner 100 times. After the total of $100k$ RRT expansions, the reachability information is updated in the progress estimation module, changing the weights on the discrete layer, and a new guide path is computed, taking these new weights into account. SYCLOP repeats this series of steps until the end of the planning cycle has been reached. The discrete guide path on the DAG is computed by a shortest path algorithm 95% of the time, and by a random depth first search, ignoring edge weight, the other 5%. All SYCLOP internal parameters were left at their defaults. For additional details on the inner workings of SYCLOP see [3].

We incorporate predicted sensor payoff in the above scheme by adding a new multiplicative edge weight factor in the progress estimation module for the discrete layer. The same multiplicative factor is used for the distance function in the continuous sampling-based motion planning. This new factor causes the discrete guide path to prefer regions of interest, while simultaneously considering reachability. The idea behind this multiplier, simply called *factor* from here on, is to allow the incorporation of a secondary objective, while the planner still only explicitly considers the primary objective. The secondary objective is seen as an online optimization problem, where *factor* determines the relative costs of poor performance between the primary and secondary objectives. It typically causes a deviation of the robot from its original path.

Each planning cycle, a new sensor measurement is available. This is used to modify the edge weights of the discrete model and help guide the robot to areas where the sensor is likely to be most effective. The sensor model associates a payoff value to each region of the discrete model. The payoff of a region is calculated as the estimated σ value of the sensor model, as described in Section II, as if the robot were at the center of the region, and normalized to be between 0 and 1. A payoff of 0 means that no sensor information is expected and 1 means that this is the best possible location to take a sensor reading. If no anomaly is in range, the sensor model reports the same value everywhere. In this case, *factor* will be equal across every region and not influence SYCLOP at all. Otherwise, *factor* will be calculated as in the formula below:

$$factor = \left(1.5 - \tfrac{1}{2}(\text{payoff}\,[region1] + \text{payoff}\,[region2])\right)^c$$

A region with sensor payoff greater than 0.5 causes *factor* to be less than one, and therefore the DAG edge will have a decreased weight. Regions where the sensor payoff is less than 0.5 will yield an increase in edge weight. The values of *factor* fall in the range $[0.5^c, 1.5^c]$. The c parameter controls how important the sensor information is compared to path length and reachability. Another way of looking it is that c controls the relative importance of the secondary target with respect to the primary target. There is no easy way to decide this automatically and, thus, c has to be specified by the user. Although in our sensing model the payoffs do not change other than when targets enter or leave the maximum range, there is no restriction against a sensor model that provides different payoffs for every planning cycle.

Finally, to bias the sampling-based motion planner towards informative paths, the distance function used in this motion planner is similarly multiplied by *factor*, turning it into a problem-specific cost function. This is because the discrete guide may encourage the continuous layer in the correct direction, but the randomized continuous layer finds and selects a path that is closer to the goal than one that acquires extra sensor information. By weighting distance as $distance \times factor$, we ensure that the distance from the goal is correctly balanced against the potential payoff of greater sensor information at both planning levels.

The exponent c is hence a control parameter of the system, where $c = [0, \infty)$. This is used to define how much the sensor information influences SYCLOP by specifying the balance of how much sensor information is worth compared to extra distance traveled. $c = 0$ reduces to the case where sensor information is not utilized in the planning, and we use this for a baseline benchmark.

The output at the end of each planning cycle is a sequence of control inputs for the robot that last for the duration of the next planning cycle. The robot then begins executing that plan and takes a new sensor measurement. For the first planning cycle, the robot does not move, and all subsequent cycles follow until the robot arrives at the destination.

IV. EXPERIMENTAL RESULTS

To provide proof of concept, we change c in our scenario. As the exponential control parameter c increases, we expect to see that the robot takes longer to reach the goal, but successfully senses the target more often.

In each experimental run below, the robot executes a replanning loop with cycle time of 2s. The world model,

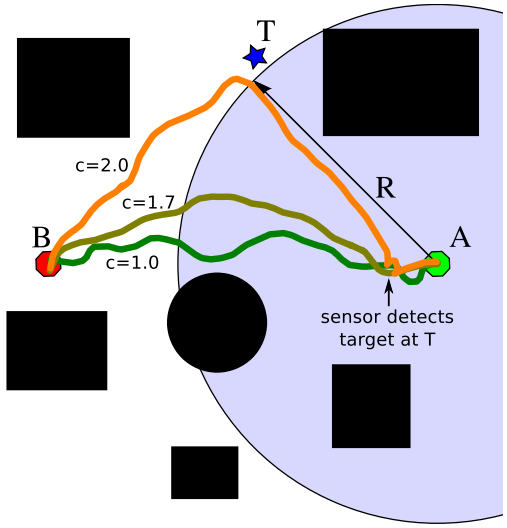

Fig. 3. Representative execution paths of the robot with $c = 1.0, 1.7, 2.0$. The robot starts at position A, needs to travel to destination B, and discovers a secondary objective to sense a target at position T.

robot, and visualization utilize the interprocess communication infrastructure available in ROS [23].

Figure 3 shows the environment that was simulated with three paths overlaid on it. The first, with $c = 1.0$, shows no meaningful deviation from the $c = 0.0$ case. We do not show $c = 0.0$ because of the significant overlap it would have. The second, with $c = 1.7$, is a path with clear deviation from the case with little or no sensor information, but still does not deviate too far from the initial path of the robot. Finally, the third path, with $c = 2.0$, has a very large deviation, where the planner leads the robot almost to the target. When the robot gets close, it has a very high probability of sensing the target as defined in Section II. Once the sensor model detects a high quality measurement, it no longer considers the target to be an anomaly, and reports that no region in the workspace is expected to provide sensor information. Thus, after a success, the subsequent replanning operations head to the goal without further influence by the sensor.

An important graph to present is Figure 4, showing the success of missions as the c exponent is varied across different mission executions. Each data point represents the mean of 30 independent runs. As can be seen, the average mission success probability increases when we increase c to make the expected sensor information more valuable. It is interesting to note that success rate is near 100% at $c \approx 2.0$ and thus does not increase at higher c values. This is because the system already deviates enough to acquire good sensor information, so more deviation does not help.

The data shows that under the Gaussian sample sensor criteria, the robot is never successful at c values under 1.5. Above 1.5, success rates very quickly jump to 90%. Upon observing this jump, more experiments were run at intermediate values. It is still a very fast jump, and the authors plan to investigate whether this behavior is a property of the environment, the weighting factor, the robot dynamics, or another unforeseen interaction.

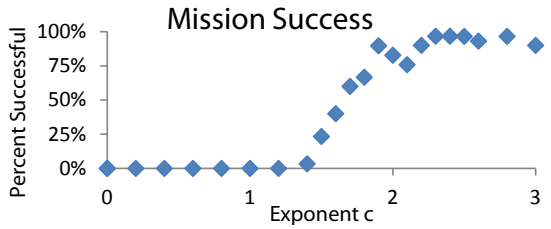

Fig. 4. The percentage of successful missions, leveling off at approximately $c = 2.2$

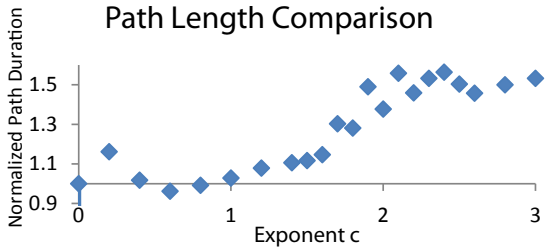

Fig. 5. Higher c values result in higher completion times, as expected.

We can see from Figure 5 that, as expected, modifying the distance function and DAG weights guide the robot along longer paths. At around $c = 2.0$ and above, the system is heading directly for the target until the target has been in a high quality sensor measurement and then it heads away towards the destination. This causes a plateau in the completion time, as the sensor values stop influencing the behavior of the planner once the target has been determined to be successfully sensed. The Euclidean distance between the start and destination is 300m, while the distance from start to target to destination is ≈ 424m. The plateau seen in this graph is at approximately $1.5\times$, which is near the theoretically expected $\frac{424}{300} \approx 1.42\times$, if the robot could turn instantly. At the intermediate value of $c = 1.7$ we observe that the average deviation is approximately $1.3\times$, and the system is successfully sensing the target 60% of the time. At a high value of $c = 2.0$ we see that the average deviation is approximately $1.4\times$, and the system is successfully sensing the target more than 80% of the time. This makes sense in the context of success probability changing as r^2.

To demonstrate the generality of our approach, we also simulated the same start, goal and target locations in a different workspace. There, the obstacles are much harder to navigate around for a car-like robot. Figure 6 shows solution paths for the $c = 0$ and $c = 3.0$ cases. In both paths, the car-like robot has to make several changes in direction, which are costly because the robot reverses direction and turns to a new heading. The system does not collide with any obstacles although it does get quite close—the line drawn along the path is thicker than the robot is wide. Figure 7 shows a very similar trend as in the simpler environment, where each data point is the success rate across 10 executions from start to destination.

V. Discussion

We have developed a structured way of taking into account secondary sensing objectives that are not completely specified when the robot starts its mission. Depending on how important the secondary objective is, a parameter c can be used for the

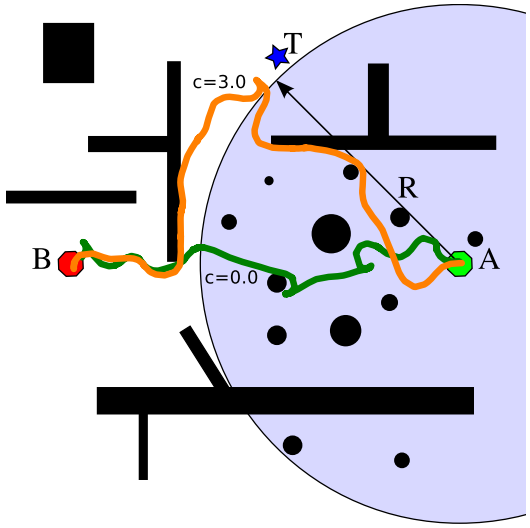

Fig. 6. A complex environment, with sensor range (R), start (A), destination (B) and target location (T) marked, as well as two example execution paths.

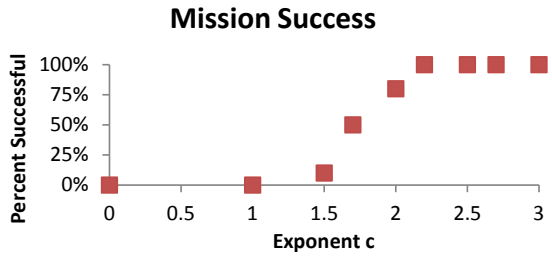

Fig. 7. A very similar trend across c values can be seen when success rates are plotted in the complex environment.

robot to automatically adjust its path an appropriate amount to capture sensor information about the target once it has been discovered. More secondary objectives could be considered in a number of ways, e.g., by making the payoff of a region the weighted sum of information gain related to each objective or by making *factor* a weighted sum of terms for each objective.

The method presented here can be improved by incorporating more realistic sensor models. This can be done in a fairly straightforward fashion if sensor values can be translated into payoff values for the planner. Extending the framework in this paper to multiple robots is another direction to pursue in future work. Coordinating multiple robots with secondary objectives adds some very challenging coordination, communication, and optimization problems. We theorize that the framework used in [9] could be used here as well. Finally, we are interested in extending this framework to handle many-layered mission parameters rather than only primary navigation and secondary sensing.

Acknowledgments

The authors are grateful to the PRACSYS team (http://www.cse.unr.edu/robotics/pracsys/) for their simulator framework, and to Matt Maly for his work implementing SYCLOP in OMPL [21] and working with the authors to debug their particular use of it.

This work was supported by grants NSF CCF-0431150, CCF-0926127, CCF-1117939, OCI-0959097, 0713623, 0920721, 1018798; DARPA N66001-11-C-4092 and N66001-11-1-4090; ONR N00014-10-1-0989, N00014-11-1-0714, and N00014-12-10579; AFOSR FA9550-09-1-0432; ARO MURI W911NF-07-1-0185 and W911NF-09-1-0383; and the Texas Instruments Leadership University Program. Additional support for DKG was also provided by an NSF Graduate Fellowship.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[3] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion Planning With Dynamics by a Synergistic Combination of Layers of Planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, June 2010.
[4] R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct. 2009, pp. 4245 –4252.
[5] J. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1987.
[6] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
[7] J. R. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. 2002 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2002, pp. 2383–2388.
[8] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *IEEE Intl. Conf. on Robotics and Automation*, 2007, pp. 704–710.
[9] K. E. Bekris, D. K. Grady, M. Moll, and L. E. Kavraki, "Safe distributed motion coordination for second-order systems with different planning cycles," *Intl. J. of Robotics Research*, vol. 31, no. 2, pp. 129–149, Feb. 2012.
[10] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-Based Path Planning on Configuration-Space Costmaps," *IEEE Trans. on Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.
[11] M. Likhachev and D. Ferguson, "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles," *Intl. J. of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.
[12] H. H. González-Baños and J.-C. Latombe, "Navigation Strategies for Exploring Indoor Environments," *Intl. J. of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.
[13] L. Torabi and K. Gupta, "Integrated view and path planning for an autonomous six-DOF eye-in-hand object modeling system," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct. 2010, pp. 4516–4521.
[14] E. Dunn, J. van den Berg, and J.-M. Frahm, "Developing visual sensing strategies through next best view planning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct. 2009, pp. 4001–4008.
[15] J. Velez, G. Hemann, A. S. Huang, I. Posner, and N. Roy, "Planning to perceive: Exploiting mobility for robust object detection," in *Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, 2011.
[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Intl. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.
[17] J. M. Phillips, N. Bedrosian, and L. E. Kavraki, "Guided Expansive Spaces Trees: A Search Strategy for Motion- and Cost-Constrained State Spaces," in *Proc. 2004 IEEE Intl. Conf. on Robotics and Automation*, Apr. 2004, pp. 3968–3973.
[18] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, Jan. 2008.
[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
[20] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1615–1630, 2005.
[21] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, 2012, accepted for publication. http://ompl.kavrakilab.org.
[22] S. M. LaValle, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.
[23] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Proc. Open-Source Software workshop at the Intl. Conf. on Robotics and Automation*, 2009.