Incremental, Sensor-Based Motion Generation for Mobile Manipulators in Unknown, Dynamic Environments

Peter Lehner[1] Arne Sieverling Oliver Brock

***Abstract*— We present an incremental method for motion generation in environments with unpredictably moving and initially unknown obstacles. The key to the method is its incremental nature: it locally augments and adapts global motion plans in response to changes in the environment, even if they significantly change the connectivity of the world. The restriction to *local* changes to a global plan results from the fact that in mobile manipulation, robots can ultimately only rely on their on-board sensors to perceive changes in the world. The proposed method addresses three sub-problems of motion generation with three algorithmic components. The first component reactively adapts plans in response to small, continuous changes. The second augments the plan locally in response to connectivity changes. And the third extracts a global, goal-directed motion from the representation maintained by the first two components. In an experimental evaluation of this method, we show a real-world mobile manipulator executing a whole-body motion task in an initially unknown environment, while incrementally maintaining a plan using only on-board sensors.**

I. Introduction

Many scenarios in mobile manipulation do not comply with the assumptions made by common motion generation methods. For example, motion planners based on the PRM or RRT methods [1] require information about the entire environment to determine a valid plan. In contrast, mobile manipulators may have to obtain this information incrementally, as they move through the environment. Planning methods also require substantial computation time to determine paths in complex, large-scale environments, especially for systems with many degrees of freedom. This is not adequate for mobile manipulation scenarios, where a robot cannot "stand and think" for extended periods of time.

We propose an incremental, sensor-based motion generation method. It is based on the simple insight that *(a)* the world changes in a continuous fashion, *(b)* significant connectivity changes occur relatively infrequently, and that *(c)* all of these changes must be observed to be reflected in a plan. These (obvious) insights lead to different assumptions to those of existing planning methods, assumptions that can be met in the context of mobile manipulation. First, we assume that most changes to the world only require local changes to the plan. Second, we assume that most of these changes are relatively simple motion planning problems. And third, we assume that information about the environment is obtainable exclusively through on-board sensors.

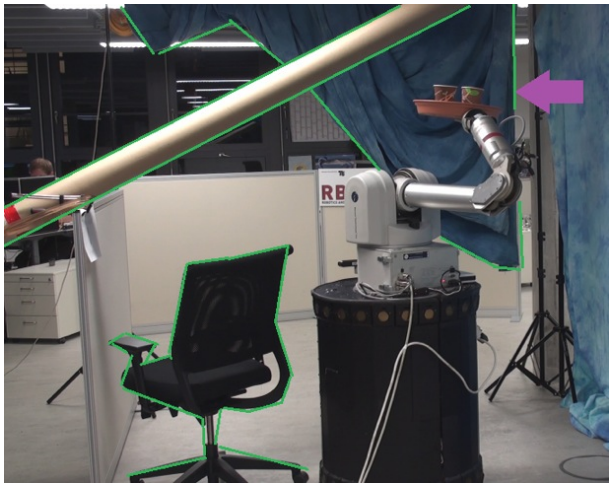

Fig. 1. An example motion generation task for a mobile manipulator. The WAM+XR4000 maintains the orientation of a tray while avoiding unstructured obstacles (green). The robot uses only on-board RGB-D sensors and has no prior model of this environment.

The proposed method is *incremental* in that it continuously constructs, extends, and refines global motion plans (roadmaps) based on sensory input. Incrementality is realized by dividing the overall motion generation problem into three types of sub-problems, each of which can then be solved by the most appropriate method. First, to respond to small, continuous changes in the world, we employ online trajectory modification to adapt the existing edges of the roadmap. Second, to update local connectivity changes or to reflect significant changes to our world model, we employ an efficient motion planner to add new milestones and edges to the roadmap. Third, to ensure the global motion goal is attained, we use graph search in our incrementally maintained roadmap.

We evaluate this incremental approach to motion generation in a real-world mobile manipulation scenario (also shown in Figure 1), demonstrating that it satisfies the following three requirements. First, the motion satisfies *task constraints*: in our experimental evaluation, the robot maintains its end-effector orientation while carrying a tray of objects. Second, the robot deals with *unpredictable dynamic*

[1]Peter Lehner is with the Department of Autonomy and Teleoperation, Institute of Robotics and Mechatronics, German Aerospace Center (DLR) at Oberpfaffenhofen, Germany.

Arne Sieverling and Oliver Brock are with the Robotics and Biology Laboratory, Technische Universität Berlin, Germany

We gratefully acknowledge the funding provided by the Alexander von Humboldt foundation and the Federal Ministry of Education and Research (BMBF). We are equally greateful for funding provided by the SOMA project (European Commission, H2020-ICT-645599), the First-MM project (European Commission, FP7-ICT-248258), the EuRoC project (European Commission, FP7-ICT-608849), and the German Research Foundation (DFG, award number BR 2248/3-1). We thank SimLab for their support.

obstacles: the robot handles significant connectivity change in the environment. Third, the robot relies only on *on-board sensors*: throughout the experiment, the mobile manipulator acts truly autonomous and does not rely on external sensors or pre-built maps of the world.

II. Related Work

A. Control and Local Optimization

The artificial potential field method [2] is one of the earliest obstacle avoidance approaches. It combines repulsive forces from obstacles with an attracting force at the robots goal. The method is computationally efficient and reacts directly to sensor data. Reactive methods can be applied to entire trajectories in configuration space [3] or in the work space [4], maintaining prior planned paths in dynamic environments. Trajectory modification can also be realized with stochastic optimal control [5], covariant optimization [6], or sequential quadratic programming [7]. However, all of methods in this category require approximate global solutions as initialization and are subject to local minima.

B. Motion Planning

Probabilistic roadmap methods [8] and the rapidly exploring random tree planners [1] are widely used in mobile manipulation planning [9, 10]. Sampling-based motion planners can accommodate dynamic obstacles with known trajectories [11] and they also handle sensor-generated world models [12]. However, they are difficult to apply in unpredictable environments as unexpected changes in the environment invalidate large parts of the tree or roadmap.

The high computation time of most global planners can be reduced significantly by guiding the planning process with workspace information. A very relevant planner for our work is the exploring/exploiting tree (EET) planner [13]. We incorporate the EET in our method to solve local planning problems; these solutions are used to incrementally update a roadmap.

C. On-Line and Reactive Planning

Several approaches presented in the literature aim to equip global planners with reactive capabilities. We analyze if these methods satisfy the requirements of applications in mobile manipulation.

1) Replanning: Replanning [14] is a simple method to adapt a solution path in unpredictable environments and has been applied successfully for small mobile manipulation tasks [12]. Replanners observe the feasibility of a planned motion at execution time. The planner plans one trajectory and checks its validity continuously. For invalid trajectories, the planner is simply evoked again. In realistic scenarios, replanning might lead to significant interrupts in the motion of the robot: in the time a new plan is computed, it can be immediately invalidated by new changes in the environment.

2) Online Adaption: Some methods plan one [15] or more [16] initial paths using a global search that subsequently are adapted using trajectory optimization methods. In environments with small changes, these methods lead to satisfactory results. However, when trajectory modification fails, the planner has to be invoked again, falling back into the category of replanning approaches.

3) Feedback Motion Planning: Feedback motion planning methods [17, 18] integrate the strengths of global planning and control to find policies over the whole state space. Such a plan is pre-computed offline for all possible states of the world. The robot then can query for optimal actions depending on its state. However, constructing a complete global feedback motion plan that incorporates unpredictably moving obstacles is infeasible, due to the inherent complexity of motion planning. To use feedback planning in mobile manipulation, one must resort to approximate methods that leverage additional structure present in the problem. This is exactly the route we take with the proposed approach, which can be considered an incremental feedback motion planner.

4) Adaptive and Elastic Roadmaps: Adaptive, roadmap-based methods can be viewed as a specific category of feedback motion planners. They validate edges in the roadmap at run time, thus adapting the plan to environmental changes [19, 20]. In unpredictable dynamic environments, multi-query roadmaps outperform single-query tree-based planners, as the invalidation of an edge simply leads to the selection of another path in the roadmap. Such roadmap-based methods are an efficient way to reuse previously found connectivity.

The work presented here extends the elastic roadmap framework [21, 22], which falls into this category of motion generation approaches. In elastic roadmaps nodes and edges correspond to reactive controllers. These controllers respond to changes in the environment and maintain validity as long as possible with only local methods. Elastic roadmaps combine the global properties of planning, the reactivity of control, and the advantages of multi-query methods. A similar approach [23] combines an adaptive roadmap with local replanning but does not consider real sensor data and task constraints.

III. Incremental Elastic Roadmaps

Above we stated three assumptions for incremental, sensor-based planning: *(a)* most changes in real world problems result from small, continuous object motions, *(b)* significant changes to the connectivity occur rarely, and *(c)* changes in the environment can only influence the plan if they are perceived by the robots on-board sensors. The goal of this section is to present a roadmap-based planning method based on these assumptions.

These three assumptions motivate a division of the problem into three sub-problems. The first component adapts the roadmap within the sensing range to reflect the continuous changes in the environment *(a)*. The second component detects the rarely occurring significant changes *(b)* that can

not be handled by the first component. The component addresses these changes and treats them by solving local motion planning problems. This component is efficient because it only plans within the robots sensing range *(c)*. The following three sections introduce these three main components of our method.

A. Handling continuous change

Most common control or trajectory optimization methods address continuous change in plans efficiently. We will present one particular solution using the previously published elastic roadmap framework [21]. We provide an overview but refer the reader to the prior publication for technical details.

The elastic roadmap framework uses a roadmap, i.e. a graph, as the underlying representation. Edges in the elastic roadmap are not specific paths in configuration space but instead controllers. A vertex or milestone represents a configuration of the robot. Two vertices are connected by an edge in the roadmap iff the controller is able to move the robot between them. By replacing the edge with a controller that includes obstacle avoidance behavior, we gain the flexibility to locally react to small changes in the world without incurring the need to replan. We can imagine these edges as being "elastic".

Each milestone is also associated with a controller that modifies it in response to changes in the environment. Now vertices can also move around. The result is an elastic network of moving milestones reacting to dynamic changes in the environment. Robust motion plans can be generated by sequencing the controllers along the edges so as to connect the start and the goal milestone in the roadmap.

The elastic roadmap framework is computationally efficient and able to maintain the roadmap for large environments [21]. When the controllers associated with vertices and edges are task-constrained operational space controllers [24], the motion generated with the elastic roadmap satisfies global constraints as well as task-constraints while achieving reactive obstacle avoidance. But when the environment changes significantly, the roadmap may not be able to find a solution to the planning problem. To overcome this limitation, the next Section introduces the second component of our method.

B. Handling connectivity changes

The second component of our method addresses significant, rarely occurring changes in the environment. This efficient reasoning about connectivity is the major technical contribution of this paper and completely novel. Therefore we describe technical details in Section IV. When adapting an existing elastic roadmap to a dynamic environment certain changes can not be handled sufficiently only using local adaptation. We subsume all of these changes as changes of *connectivity*. Reasons for changes in connectivity could be the appearance of an obstacle - in this case the roadmap would need to capture paths around the obstacle. It could also be the disappearance of an obstacle, i.e. a door opening up a new possible shortcut in this case the roadmap would need to include new edges passing through the door.

To react to significant changes we need an efficient way to capture connectivity. Reasoning about the connectivity of the high-dimensional configuration space is a hard problem. We will present the *workspace connectivity graph*, a method that approximates configuration space connectivity with workspace connectivity. This graph can be constructed very efficiently at high rates. Using the workspace connectivity we check if the connectivity represented in the roadmap differs from the current state of the environment. If there is a difference, we modify the roadmap locally by inserting or removing milestones and edges. The removal of milestones is simple. We create and insert milestones by using sampling-based motion planners. Configuration space search becomes feasible when we guide the planners with the found workspace connectivity.

C. Global planning

The remaining planning problem is extracting a global plan that leads the mobile manipulator from its current position to a goal. As we already devised methods to create and maintain a roadmap in the previous two Sections this last planning problem is a trivial graph search. We compute the distance between adjacent milestones and use Dijkstra's algorithm on the elastic roadmap to extract the shortest path. As the roadmap changes continuously, we have to recompute the shortest path frequently. In our experiment the roadmaps contained roughly 200 milestones and we achieved satisfying rates. For very large roadmaps, this process should be improved with an incremental graph search algorithm such as D* [25].

IV. Updating roadmap connectivity

In this Section we will present details on our method to augment elastic roadmaps to deal with connectivity changes. Fig. 2 serves as an introductory example for this process. It shows how our method constructs a roadmap iteratively from scratch. In all images, the grey background is the representation of the environment. The upper row of the figure shows five tunnels through the free workspace. These tunnels are the result of a workspace analysis that detects parts of the workspace where the current roadmap does not represent the connectivity sufficiently. The lower row shows the outcome of the method's second step: a local planner generates paths guided by the workspace tunnels. The paths get integrated as milestones into an elastic roadmap. The final roadmap approximates the connectivity of the environment well as shown in the lower right image.

A. Approximating workspace connectivity

The workspace connectivity graph is an efficient representation of the connectivity of the environment. Its creation is based on a sphere-based wavefront algorithm similar to the one used in decomposition-based planning [26]. The algorithm floods the free workspace with spheres (see Fig. 3b). It tries to minimize the amount of spheres by greedily preferring larger spheres over small ones. In a second step we construct an undirected graph where nodes correspond

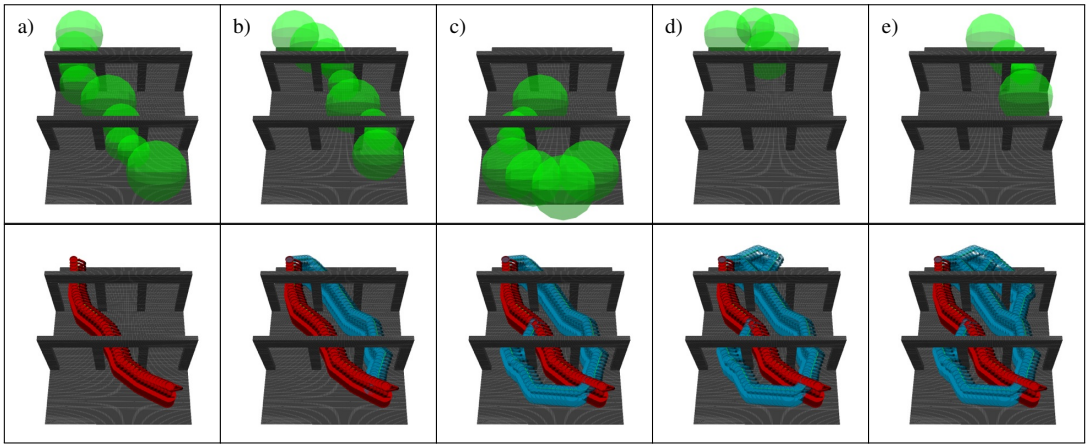

Fig. 2. Illustration of the roadmap generation process: Figure a - e) show the continuous generation of the roadmap from the start configuration in the lower right corner to the goal in the upper left corner. Each figure shows a workspace tunnel (top) and the incremental elastic roadmap after the resulting path was inserted (bottom)

to the sphere centers. We add edges between nodes if two sphere overlap significantly (see Fig. 3c). Each path through this graph is a sequence of overlapping spheres that can be seen as a tunnel through the free workspace.

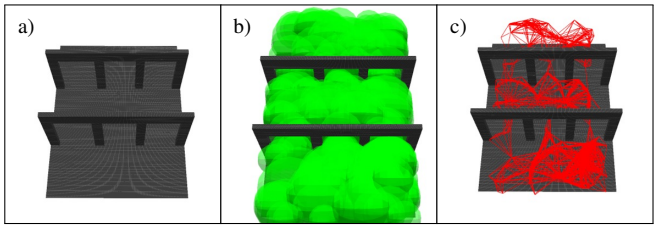

Fig. 3. Illustration of the workspace connectivity graph: a) shows a sample environment with six passages. b) shows the decomposition of the workspace with spheres. c) shows the resulting workspace connectivity graph.

B. Generating paths in workspace

The incremental planner uses the workspace connectivity graph to extract tunnels through the free workspace, These tunnels then guide our underlying sampling-based planner. We only consider tunnels that help us in the current motion generation task by *(1)* leading to the goal and *(2)* distinguishing from the current elastic roadmap. We define the cost of an edge e in the workspace connectivity graph as a sum of the Euclidian distance and a penalty term k_{covered}. k_{covered} is a constant that should be higher than all other distances. It is only added if there is a milestone inside one of the spheres incident to e. We summarize the construction of the workspace connectivity graph and theis cost function in Algorithm 1. After the graph construction we apply Dijkstra's algorithm to compute a minimal cost path. The minimal cost path is a tunnel of overlapping spheres that connect start and goal. The tunnel leads through unoccupied workspace and is as short as possible. We then use several of these tunnels to guide sampling-based motion planners locally, create milestones, and add them to the roadmap.

Algorithm 1: CREATE_WGRAPH($p_{\text{start}}, V_{\text{roadmap}}$)

```
1  Graph G = (V,E); Priority Queue Q ← ∅; Sphere s_start
2  s_start.center ← p_start
3  s_start.radius ← DISTANCE(p_start)
4  Q.PUSH(s_start, s_start.radius)
5  repeat
6      s_top ← Q.POP()
7      G.ADD_VERTEX(s_top)
8      forall the s in V do
9          if OVERLAP(s, s_top) ≥ r_min then
10             cost ← DISTANCE(s, s_top)
11             forall the v in V_roadmap do
12                 if s.COVERS_MILESTONE(v) then
13                     cost ← cost + k_covered
14             G.ADD_EDGE(s, s_top, cost)
15     for i = 1 to N · s_top.radius do
16         s_new.center ← SAMPLE_POINT_ON_SURFACE(s_top)
17         s_new.radius ← DISTANCE(s_top.center)
18         if s_new.radius ≥ r_min AND ¬COVERED(s_new, V) then
19             Q.PUSH(s_new, s_new.radius)
20 until Q = ∅
21 return G
```

V. Implementation

In this Section we will introduce an implementation of the last two remaining components of our method. First we will introduce a sampling-based motion planner that can be guided efficiently by tunnels in the workspace connectivity graph. Then we will introduce a sensor-based world representation that can be updated at the required rates.

A. Locally guided motion planning to find connectivity

Generally, all sampling based motion planners can be used in our method provided they satisfy the following two requirements: first, the planner needs to generate paths that satisfy task constraints. Second, the planner should be able to use workspace information to guide sampling.

In our implementation we integrate the exploring / exploiting tree (EET) planner [13]. The EET satisfies both stated conditions. The EET accommodates task-space constraints easily, although this was not explicitly stated in the original publication. The EET samples task frames and then grows a search tree in configuration space. By restricting the sampling of task frames to those satisfying task constraints the EET only searches paths that lie completely on the task manifold.

The second requirement is that sampling can be guided efficiently by workspace information. While in many cases the guidance of free workspace is helpful, sometimes it can lead the planner in wrong directions. The key of the EET planner is that it adaptively balances the usage of information (which is exploitation) with a broader search in configuration space (which is exploration). This balancing lets the planner rely on workspace information as much as possible if the information is helpful. In regions where the information is misleading the planner relies less on it and searches more broadly. The adaptive usage of workspace information lets the EET perform up to three orders of magnitude better than standard RRT planners [13], on a large number of realistic problems.

Once the EET finds a new path the algorithm places milestones at an regular interval Δq into the elastic roadmap. Each milestone becomes an initial configuration for a controller that reacts to changes in the environment.

B. Environment representation

As our planner is incremental, we need a sensor-generated representation that we extend efficiently with the measurements of on-board RGB-D sensors. We use a distance function computed over a 3D voxel grid [27]. This function, which contains the distance to the closest obstacle, can be updated incrementally in constant time with a brushfire algorithm [28]. The distance function also allows for efficient collision checking, which we need for the local sampling-based motion planning. We approximate the robots volume with bounding spheres and test possible collisions for each sphere separately.

VI. Experiment

In this real-world experiment we show the incremental elastic roadmap is able to satisfy our three initially stated requirements for motion generation. We show the method is able to

- satisfy *task constraints*. The robots carries a tray while avoiding obstacles. During motion it has to solve a whole-body planning task of lowering the tray to avoid collisions.
- deal with *unpredictable dynamics*. The robot reacts to appearing as well as disappearing connectivity in the environment. The robot handles appearing obstacles in the environment but also the removal of previously seen obstacles.
- rely only *on-board sensors*. In the experiment the robot uses only the sensor data from one RGB-D camera mounted on-board. The robot has no prior information about the structure of the environment.

In the next section we will discuss the setup of the experiment and afterwards evaluate the results.

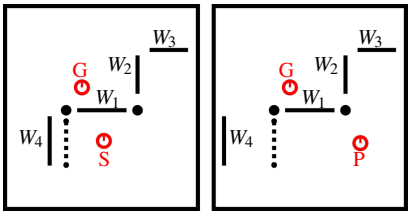

Fig. 4. Sketch of the experiment with walls $W_1 - W_4$ labeled for explanation. Start (S) and Goal (G) and intermediate position (P) are shown in red. W_4 is moved when the robot reaches P.

A. Setup

For the experiment we implemented the proposed method on a mobile manipulator consisting of a Nomad XR4000 base and a Barrett WAM arm, resulting in a robot with ten degrees of freedom. Mounted on the base is an Asus Xtion Pro RGB-D sensor which obtains depth information in front of the robot.

The goal of the experiment is an orientation constrained pick and place task: The mobile manipulator moves a tray upright to a goal location. The mobile manipulator starts with no information about the environment and assumes the unknown environment is free of obstacles. During the experiment a person moves a wall. The robot has no access to global localization during motion.

Figure 4 shows the setup of the experiment. The robot is standing in front of wall W_1 and oriented towards it. The goal of the experiment is to bring the mounted tablet to the goal position behind wall W_1. The robot has a limited sensing range and only observes wall W_1 in the beginning. Due to the setup the robot will first choose a path to the right of wall W_1. Once it reaches a position (P) where it can observe this path is blocked a person moves wall W_4 to the border of the scene. This frees the passage below a curtain. The robot proceeds on a path around the left of W_1 to the goal.

In total the experiment contains three different sources of unpredictable dynamics: First, as the robot starts without prior knowledge, it integrates new measurements continuously in its world model. Every detection of a new obstacle represents a change to the robots perceived environment and invalidates previously planned paths. Second, the manual removal of a wall opens up connectivity that was blocked before. The robot has to notice the change of connectivity and react by planning a new alternative locally. Third, there are additional dynamics present because of the robots odometry error. Without global localization, the world is moving constantly relative to the robot. The roadmap also has to adapt to this motion.

B. Evaluation

The experiment is depicted in Figure 5. After 9 s the planner finds the two paths which describe the connectivity

of the scene at the beginning. After an initial period of switching between these two paths to map out the initial scene the robot starts to move on a path to the right of wall W_1 and detects walls W_2 and W_3. The planner reacts to the change by removing all paths around the right of W_1 and chooses a path around W_4 at 152 s. A person moves W_4 to the border of the scene and enables a new disjoint path alternative. The robot continues on its path and observes that W_4 is missing. The planner adapts the current path to pass below the curtain at 187 s and generates a shortcut through the freed space at 216 s. The robot passes below the curtain at 296 s and retracts the arm to avoid collision. It continues on the path and reaches the goal at 405 s.

The experiment shows that the method continuously finds new connectivity at responsive rates. It approximates the connectivity of the scene throughout all stages of the experiment. At the beginning the planner is able to find the two initial path possibilities in quick succession. Both significant changes in connectivity are incorporated into the roadmap:

- The restriction of the connectivity to one possibility after the detection of wall W_2 and W_3.
- The appearing connectivity after wall W_4 is removed.

The experiment also highlights the necessity of local feedback to preserve the connectivity of the roadmap. The initial paths are adapted to the continuous detection of W_1 and W_4. During the experiment the base of the mobile manipulator is subject to significant odometry error. The roadmap is able to compensate this position uncertainty with feedback as well. During the experiment the mobile manipulator never has to 'stop and think' except for a short time of initial roadmap construction. After initialization the roadmap always supplies alternative paths and the robot fulfils the task in one continuous motion.

VII. Conclusion

We presented an incremental, sensor-based motion generation method, suitable for problems in mobile manipulation. The key contribution of this methods consists in the division of the motion generation problem into three sub-problems: addressing continuous changes in the world, addressing connectivity changes, and finally obtaining global motion. By dividing the problem in this fashion, we can employ the most appropriate and efficient method for solving each subproblem. As a result, the method is able to efficiently generate global, task-constrained, and reactive feedback plans. The generation and execution of these plans is based on a world model the robot acquires throughout its motion. We demonstrate the effectiveness of this approach on a real-world manipulator performing a constrained end-effector task in a complex and unpredictably changing environment. The proposed method can fail because the underlying map only stores the last known state of the world. This map might be in a state where all paths to the goal are blocked even if there is a path to the goal. To handle these cases, we have to integrate our previous results on reasoning about uncertainty in an elastic roadmap planner [22]. This is left for future work.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2004.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 1985, pp. 500–505.

[3] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *1993 IEEE International Conference on Robotics and Automation (ICRA)*, 1993, pp. 802–807.

[4] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[5] M. Toussaint, "Robot trajectory optimization using approximate inference," in *26th Annual International Conference on Machine Learning*, 2009, pp. 1049–1056.

[6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[7] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[8] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[9] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[10] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[11] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[12] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Mobile manipulation in unstructured environments: Perception, planning, and execution," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 58–71, 2012.

[13] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1305–1317, 2014.

[14] K. Hauser, "On responsiveness, safety, and completeness in real-time motion planning," *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, 2012.

[15] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[16] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1199–1212, 2008.

[17] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[18] L. Yang and S. LaValle, "The sampling-based neighborhood graph: an approach to computing and executing feedback motion strategies," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 419–432, 2004.

[19] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[20] A. Nakhaei and F. Lamiraux, "Motion planning for humanoid robots in environments modeled by vision," in *8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 197–204.

[21] Y. Yang and O. Brock, "Elastic roadmaps—motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[22] A. Sieverling, N. Kuhnen, and O. Brock, "Sensor-based, task-constrained motion generation under uncertainty," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4348–4355.

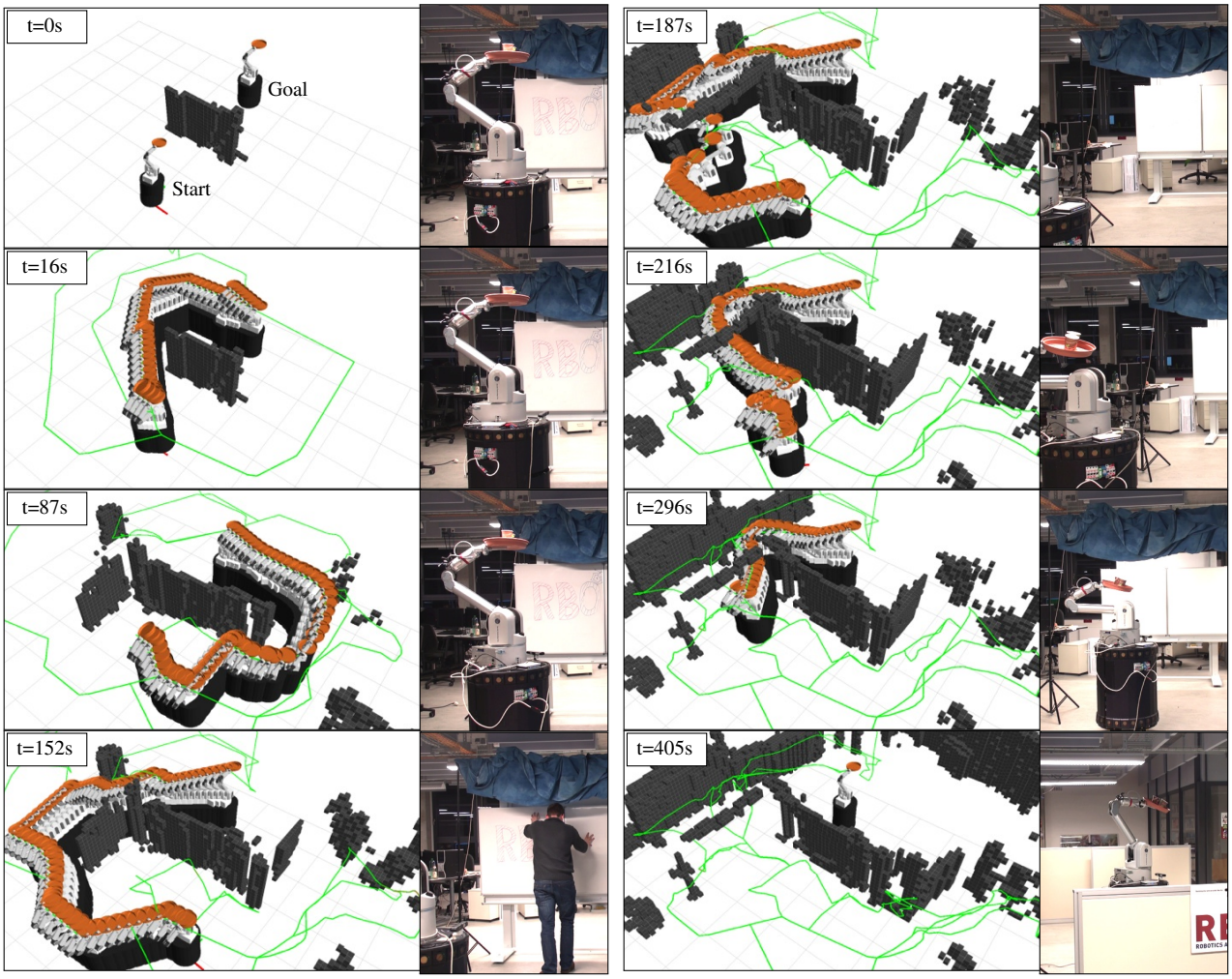

Fig. 5. The roadmap end-effector positions are shown in green, the occupancy grid is shown in dark grey, and the current path is shown as a flow of configurations. At $t = 0$s the robot perceives only the wall W_1 in front of it. It receives the task of moving to a goal configuration behind the wall. Initially, it chooses a path to the left of W_1 at $t = 16$s. After alternating three times between paths to the left and right of W_1 it chooses a path on the right at $t = 87$s. It follows this path around the corner and discovers the right path is blocked at $t = 152$s. The robot instantly chooses a path to the left of W_4. We manually remove W_4 and the algorithm adapts a path through the hole at $t = 187$s. The algorithm perceives the new motion alternative and finds a path underneath the curtain at $t = 216$s. At $t = 296$s the robot passes below the curtain and retracts the arm to avoid collision. At $t = 405$s the robot safely reaches the goal.

[23] E. Yoshida and F. Kanehiro, "Reactive robot motion using path replanning and deformation," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5456–5462.

[24] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[25] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *1994 IEEE International Conference on Robotics and Automation (ICRA)*, 1994, pp. 3310–3317.

[26] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *2001 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2001, pp. 1469–1474.

[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[28] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1116–1130, 2013.