IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2015

Using Environment Objects As Tools In Unknown Environments

Martin Levihn Henrik Christensen

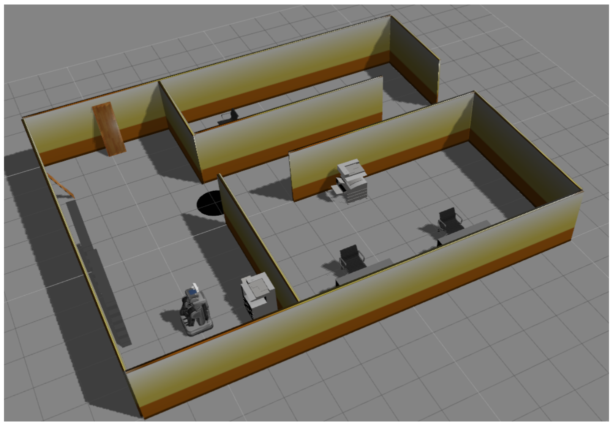
Abstract— **Robots should be able to reason about using objects in the environment to assist task completion. If faced with an unexpected situation, such as a critically damaged floor, robots should be able to evaluate different ways to resolve the situation rather than directly declaring failure. In this paper, we present the first online framework that enables robots to reason about using environment objects as tools to overcome obstructions, allowing robots to solve previously unsolvable tasks. The framework utilizes constraint relaxed planning to detect unavoidable obstructions. In case the robot becomes disconnected from the goal by an obstruction, the system guides the robot to search for and utilize an object in the environment to overcome the obstruction using the concept of inverse affordances. We verified the framework in a realistic simulation environment for the PR2 robot.**

I. Introduction

Experienced humans do not hesitate to *use their environments*. Robots should not either. Suppose one is trapped in a room with burning gasoline. The human searches the environment for something that can be placed over the gasoline, finds a long enough board, places it over the fire and escapes. Typically, such situations can not be anticipated a-priori and no predetermined action plan exists. Rather, the present situation and objects have to guide the actions *online*. Possessing such capabilities becomes essential for robots that are expected to operate autonomously in complex, unstructured environments such as disaster areas.

To bring robots closer to such capabilities, this paper presents the first framework that allows a robot to reason *online* about using an object in the environment to facilitate its task completion. To understand the value of such a framework, consider the example visualized in Figure 1(a) in which a robot is tasked with escaping a damaged building. As typical for such scenarios, we assume that the robot has access to a map containing the static environment properties, but has no knowledge about the existence, size or location of non-static environment objects and no knowledge about any potential structural damages. Unaware that the floor has been critically damaged, making the only exit unreachable, our system starts by computing a motion plan for the robot to escape the building using the exit. As the robot executes the motion plan it obtains sensor readings about the environment and eventually detects the hole in the floor. Similar to existing systems such as [1], our system re-evaluates the current path based on this new information and realizes that there is no alternative path to exit the building. In contrast to existing work however, our method does not just declare failure and stop the robot. Instead, it guides the robot to evaluate the environment for ways of overcoming the obstruction. For the example in Figure 1(a), the robot searches its environment, finds a long enough board in the room, places it over the hole and successfully escapes the building. Figure 1(b) visualizes this successful escape. We are not aware of any other method that would have enabled the robot to escape this situation.

The authors are affiliated with the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, Georgia 30332, USA. Email: levihn@gatech.edu, hic@cc.gatech.edu


(a) Initial environment configuration. The robot is not aware of the location of any non-static environment objects or the hole in the floor, which prevents the robot from escaping the building.

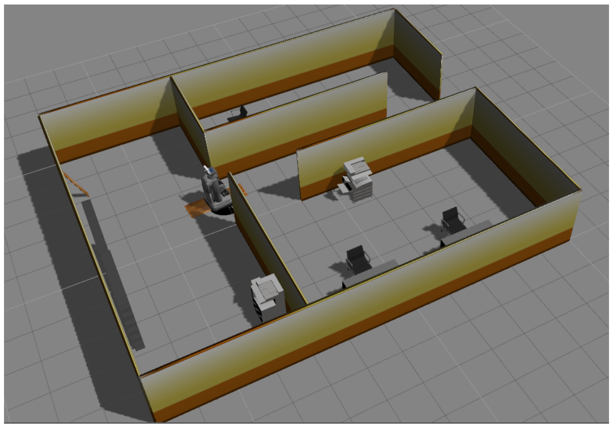

(b) The robot successfully escapes the building by using a board it found in the environment to negotiate the hole.

Fig. 1. Example execution of the proposed framework.

To achieve such behaviors, we present a framework that combines constraint relaxed planning [2] with the concept of *inverse affordances*, a mapping from a failed action to a set of object properties required to make the action feasible. Constraint relaxed planning allows the robot to decide when an obstruction can not be avoided, and the inverse affordance mapping allows the robot to determine what properties (*e.g.* dimensions) an object needs to have to be helpful in overcoming a given obstruction. The robot can then search the environment for any object with these properties. If a suitable object is found, the framework computes the necessary grasp and drop motions and finally guides the robot to use the object to overcome the obstruction.

This paper focuses on the general concept ideas and intuitive example implementations as each of the steps in the framework represent active research areas in itself. However, we anticipate that the framework presented here will be used to bring currently disjoint research contributions together and encourage other researchers to consider the problem discussed here as a new application domain.

The remainder of the paper is organized as follows: Section II presents related work and Section III provides an overview of the proposed framework. After discussing implementation details in Section IV, the system is evaluated in Section V. The paper concludes with final remarks in Section VI.

II. Related Work

The work discussed in this paper allows a robot to make progress towards a goal even if an obstruction is blocking its path. This general problem statement has been addressed before for cases that require the robot to move objects out of its way to clear a path, in which case the domain is referred to as Navigation Among Movable Obstacles (NAMO). The problem discussed in this paper of *using* environment objects to overcome an obstruction can be seen as a generalization of the NAMO concept. We now cover related work in the NAMO domain as well as in more general robotic tool use.

A. NAMO

Navigation and manipulation planning poses a significant computational challenge even with *complete environment information*. Wilfong [3] first proved that deterministic NAMO with any number of obstacles is NP-hard. Demaine [4] further showed that even the simplified version of this problem, in which only unit square obstacles are considered, is also NP-hard.

In [5], Stilman presented a planner that solved a subclass of NAMO problems termed LP_1 where disconnected components of free-space could be connected independently by moving a single obstacle. The planner was able to solve the hard problems presented in [6]. From an algorithmic perspective, our framework solves problems similar to the LP_1 NAMO problem classification, as it focuses on scenarios where each obstruction can be resolved independently using a single object. However, in contrast to [5], our system reasons about *using* environment objects rather than just seeing them as obstacles and does not assume full environment knowledge.

Wu [7] and Kakiuchi [8] introduced the first extensions to NAMO in *Unknown Environments*. In [7] a planner was presented that could solve NAMO problems with substantial lack of initial state information. We extended this planner in [9] to yield locally optimal solutions. [8] presented a system that executes NAMO in unknown environments on the humanoid robot HRP-2 with only onboard sensing. While these systems bring robots closer to making decisions online about how to overcome obstructions, the NAMO domain retains the inherent restriction of only allowing a robot to reason about moving objects out of its way. In contrast, we argue that a robot should also be abe to reason about actively using environment objects to create a path in the first place.

B. Tool Use

Most existing forms of robotic object use are focused on accurate positioning and control of specific tools such as welding instruments [10], spray guns [11], drills [12] and surgical instruments [13, 14]. In all of these scenarios the robot performs a well defined task with the tool. While the control methods developed for these scenarios are complementary to the proposed system, we do not assume that the robot is given a specific task to accomplish with a specific object. Instead, the robot has to autonomously determine if and how it needs to use environment objects to accomplish its overall task.

As an initial step towards this goal, we presented a system that allowed a HRP-2 robot to autonomously utilize environment objects to create itself a path in [2]. The robot used a box to create a stair step for itself and placed a board on the ground to cross a gap. Similarly, in [15] we presented a planning system that allowed a robot to reason about force transmission properties of environment objects. However, both methods required full a-priori environment knowledge. In contrast, the framework presented in this paper is designed for the more realistic case of substantial lack of knowledge about the initial state and allows the robot to operate and make decisions using only onboard sensing.

III. Overview

We now provide an overview of the proposed system before discussing implementation details in the following section.

A. Assumptions

We assume that the world is static and that the robot has access to a map containing immobile environment objects such as walls and stairs. We do not assume that the robot has a-priori knowledge of any structural differences between the real world and the map or of the existence or location of manipulable environment objects prior to any sensing actions.

The framework focuses on cases in which a single object can be used to resolve an obstruction. We also assume that obstructions can be solved independently.

B. Framework

The framework, as visualized in Figure 2, is initialized by computing a motion plan for the robot from its current location to the goal configuration. To be able to handle obstructions, the framework uses constraint relaxed path planning [2]. In contrast to traditional motion planning systems, that either return a sound path or no path at all [16], the constraint relaxed planning system might return a path that violates robot constraints as it treats obstructions as soft constraints rather than hard constraints. Consequently, if no alternative path existed, this planning step could return

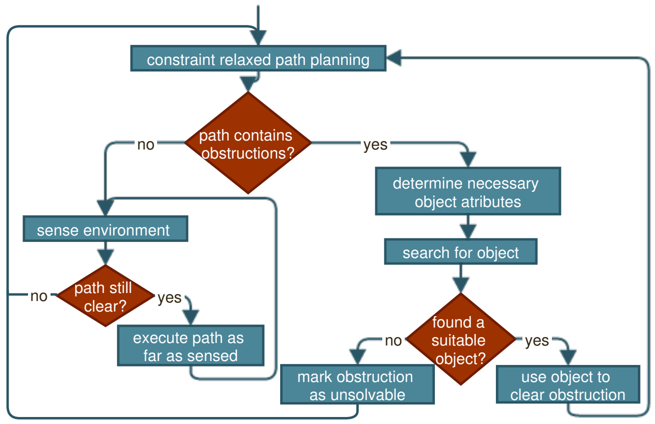

constraint relaxed path planning
path contains obstructions?
no
yes
determine necessary object atributes
search for object
sense environment
path still clear?
no
yes
execute path as far as sensed
found a suitable object?
no
yes
mark obstruction as unsolvable
use object to clear obstruction

Fig. 2. Flowchart of the proposed framework.

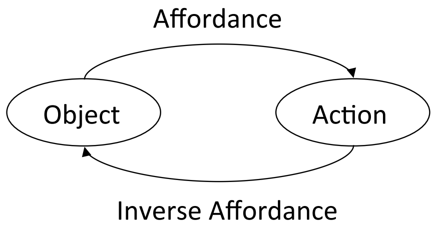


Fig. 3. Inverse affordance mapping.

a path that requires the robot to move over a gap. While such a path would not directly be executable by the robot, constraint relaxed planning provides two crucial insights. First, it establishes whether a low cost path to the goal without obstructions exists. Second, if no such paths exists and obstructions need to be overcome, it provides information to subsequent planning steps about which obstruction needs to be cleared. While this property was only utilized to reduce the search space in [2], the proposed framework explicitly uses this information to gain insight into the kind of object the robot needs to search for in order to be able to clear the obstruction.

To handle the fact that the constraint relaxed planning step could either output a sound path or a path containing constraint violations, the proposed framework branches. If the constraint relaxed planning step finds a path to the goal without obstructions, the robot is tasked with moving along the path while continuously sensing the environment for potential obstructions. In case an obstruction is intersecting the path, the framework guides the robot to evaluate the environment for options to overcome the obstruction.

To find helpful objects in the environment, the framework directly utilizes the output of the constraint relaxed planning step to determine the necessary properties any suitable object needs to have. We call this mapping *inverse affordance mapping*. While the term affordance is not uniquely defined in the literature [17, 18], here we refer to affordance as a mapping from properties of an object (*e.g.* appearance) to a set of possible actions for a given agent. Given this definition, inverse affordance represents the opposite mapping: A mapping from a set of actions to the properties of an object[1]. Figure 3 visualizes this distinction.

Given these object properties, the framework then controls the robot to search the environment for such an object. Note that this process stands in contrast to traditional object recognition problems (*e.g.* [19]) in which the task is to find a specific object. Here, the goal is to find *any* object that is usable by the robot. If a suitable object is found, the framework proceeds by guiding the robot to use the object to overcome the obstruction. Independent of whether this operation succeeds or fails, the framework loops. This looping mechanism allows the robot to treat the environment configuration resulting from the repositioning of an object as a new problem instance. In turn, this enables the robot to recover from failure situations in which the usage of an object did not result in the anticipated outcome (*e.g.* a gap is not completely covered).

[1]While a detailed discussion of this concept is outside the scope of this work, we hope to draw attention to this crucial reasoning process and anticipate numerous new research concepts.

IV. Implementation

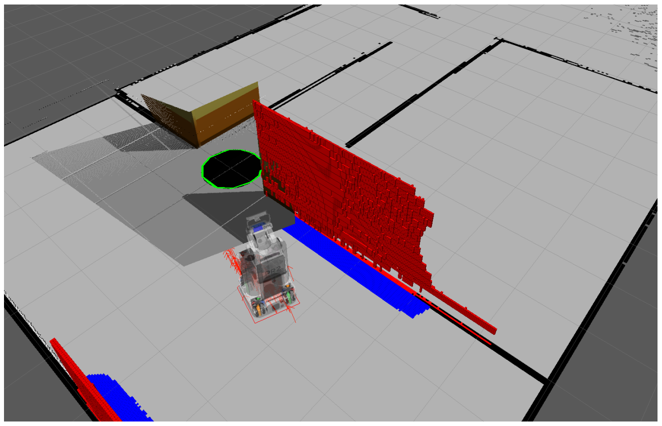
While the previous section provided a general overview of the proposed framework, we now provide a detailed description of an actual implementation and demonstrate example outputs. To reiterate, the proposed general framework is modular and any of the following subsystems can easily be replaced by more advanced methods. Our specific realization of the framework described above focuses on cases where ground obstructions such as holes or oil spills could prevent the robot from reaching its goal.

The framework requires execution monitoring to ensure that the robot does not collide with previously unknown objects and to detect obstructions. We avoid unknown objects by not allowing the robot to enter any space that it has not previously scanned and for which the scans do not indicate free space. Detecting obstructions, such as structural damages, however, is more challenging. We now detail upon our specific implementation for detecting holes and oil spills.

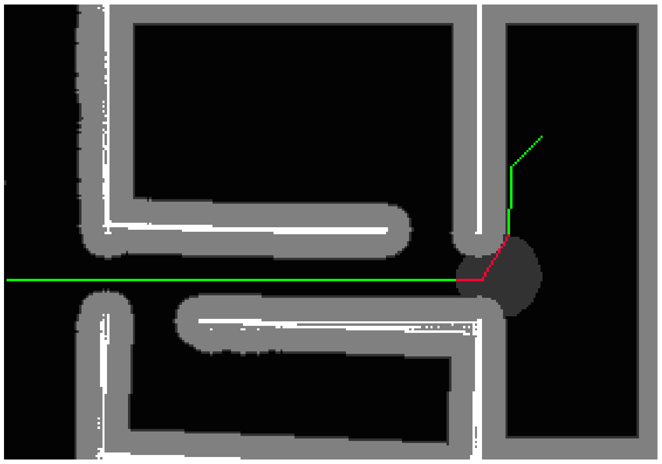
A. Detecting Obstructions

To detect crucial ground obstructions, we are utilizing a head mounted Kinect sensor [20]. To identify liquid ground obstructions as well as structural damages such as a hole, our implementation utilizes the fact that these kinds of obstruction usually result in a very different ground color than regular, traversable ground and color threshold the Kinect's RGB camera data. This allows us to detect obstructions in real-time. To not sacrifice 3D information about potential obstructions by restricting ourselves to vision methods, we are associating the color information provided by the Kinect's RGB camera with the depth information obtained by the Kinect's depth sensor. The world coordinates of the obstruction's bounding polygon are then directly given by the 3D information associated with the thresholded pixels.

This process is sufficient for cases where the robot either observes a complete obstruction or no obstruction at all. However, if the robot drives towards an obstruction, each



(a) Obstruction detection. The green polygon indicates the detected obstruction.



(b) Constraint relaxed planning step output. The red portion of the plan indicates a constraint violation, requiring the robot to resolve the constraint prior to following this part of the plan.

Fig. 4. Obstruction detection and constraint relaxed planning step examples.

scan could reveal more parts of an obstruction, resulting in a frequently changing bounding polygon and consequently many subsequent calls to the path planning system. To minimize the occurrence of such partial obstruction reports, our implementation is not reporting detected obstructions to the planning system until either the obstruction's dimensions do not increase anymore, or the robot is getting too close to the obstruction. Figure 4(a) visualizes an example output of this obstruction detection method for the scenario shown in Figure 1.

B. Planning

If the obstruction detection algorithm reports an obstruction to the planning system, the obstruction is added to the internal cost map and re-planning is triggered. The planner performs an A* search over the discretized representation of the configuration space of the robot but considers collisions with obstructions as soft constraints rather than hard constraints. A heuristic cost penalty is applied for each initial intersection with an obstruction. The A* search returns a motion path for the robot as well as a list of obstructions that have to be resolved to clear the path. This is similar to previous work in the NAMO domain (*e.g.* [5, 21, 22]) but adopted to handle the case of general obstructions rather than just intersections with movable obstacles. Figure 4(b) shows the output of the constraint relaxed planning step following the detection of the hole in Figure 4(a). The output indicates that the robot has to cross the hole in order to reach the goal.

C. Object Search

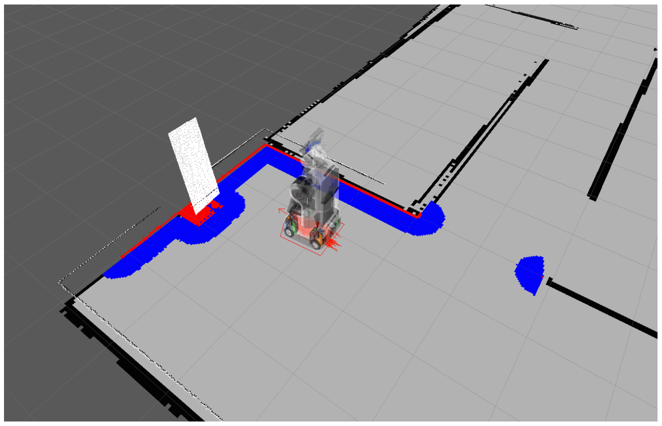
If the constraint relaxed planning step indicates that an obstruction needs to be resolved for the robot to be able to get to the goal, the framework continues by abandoning the process of attempting to reach the goal through pure navigation. Instead, the robot is now tasked with finding a suitable object in the environment to overcome the obstruction. For example, in the scenario visualized in Figure 4, the robot needs to find an object to help it cross the hole.

As mentioned above, in contrast to most existing research in object detection, this step does not focus on detecting a known object, but rather on finding *any* object that the robot could utilize to overcome the obstruction. We therefore use an instance of inverse affordance mapping to base the object detection algorithm on the output of the constraint relaxed planning step. Recall that the constraint relaxed planning step returns the exact obstruction that is currently blocking the robot. The inverse affordance mapping step now utilizes this information to determine the minimum dimensions for a suitable object and uses these dimensions to guide the search. This represents a specific realization of the general concept of inverse affordance mapping. For the example visualized in Figure 4, we need to find an object that has at least the length of the hole and is at least as wide as the robot base. The robot needs to explore the environment to find such an object.

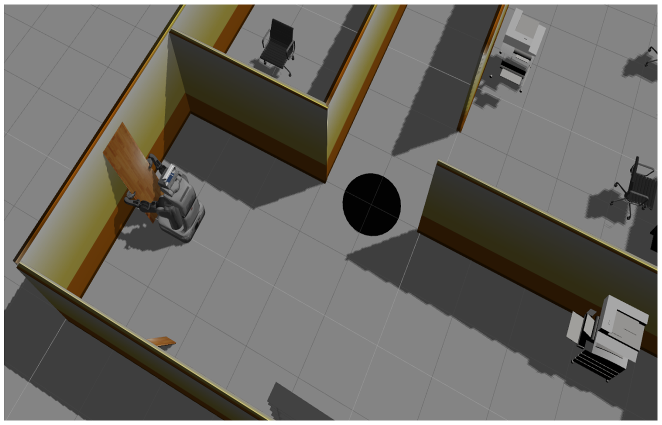
While reasoning about most likely locations of candidate objects is an interesting research area in itself (*e.g.* [23, 24]), our implementation takes advantage of simple heuristics such as on-the spot rotations and wall following to compute exploration waypoints for the robot that cover the entire reachable space as defined by the current cost-map[2].

The robot now proceeds by navigating to the exploration waypoints while scanning the environment for candidate objects using the Kinect's point cloud data. To detect suitable objects, we segment the point cloud data into individual object clusters. This is achieved by using the environment map to remove the ground plane as well as walls from the point cloud prior to clustering. Clusters that do not fulfill the dimensionality requirements, as determined above, are then rejected. Further, any clusters that are above a size threshold, indicating that the robot would likely not be able to manipulate the corresponding object, are rejected. The remaining clusters are then sorted based on a custom cost function. We used a scoring function that attempts to capture the notion of "manipulable" using surface smoothness and object width. In the order defined by the cost function, the robot is now tasked with attempting to use the objects to overcome the current obstruction.

[2]We do not consider resolving obstructions just to increase the searchable space for the robot.



(a) Robot detects suitable obstacle.



(b) Robot executes grasping subroutine.

Fig. 5. Constraint resolution example.

Figure 5(a) visualizes an example output of this obstacle detection method.

D. Grasping and Dropping

If a candidate object has been found, our realization computes a navigation plan to the object. When the robot has reached the object grasp position, which in our implementation is determined to be $20cm$ in front of the cluster, our implementation computes a detailed motion plan to grasp the object. We use the cluster information to determine pre-grasp configurations for the grippers. These configurations are computed to be $5cm$ from the edges of the cluster on each side. Given these pre-grasp configurations, the system performs a RRT-connect search with smoothing [25] for each arm individually and moves the grippers into those configurations. The arms are then controlled to move the grippers inwards until contact with the object is established. Upon contact, the grippers are closed and the shoulder joint of each arm controlled to lift the object from the ground.

If the object is successfully grasped, the algorithm computes a motion plan to guide the robot to the obstruction and align the robot with the obstruction. The alignment is determined based on the initial path direction. If the robot has reached this drop location, the robot executes a drop motion. We implemented the drop motion as a reversion of the grasp motion with the addition of a slight motion of the robot base in the direction of the obstruction. This is done to ensure that the object falls in the correct direction. If the drop was successful, the algorithm marks the new location of the object as traversable space and re-starts the constraint relaxed planning system. Figure 1(b) shows the behavior of the robot after the successful drop of the object.

V. Evaluation

We implemented the proposed framework in simulation on the PR2 robot using the physics simulator Gazebo as well as ROS [26]. We performed multiple runs on environments similar to Figure 1 with varying start locations of the robot and varying positions of the objects. The accompanying video demonstrates a complete run of the system.

Prior to running the experiments, we generated a map of the static environment properties by teleoperating the robot in the empty environment and running a SLAM algorithm [27]. For each experiment run, the robot was given access to the according environment map.

A. Runtime

Obstruction Detection: The obstruction detection subroutine took an average of $74ms$, allowing the robot to constantly monitor the ground in front of it.

Planning: The constraint relaxed planning subroutine took an average of $1.7s$ if no obstruction was blocking the goal and an average of $14.6s$ if all paths to the goal were blocked by obstructions. We can observe that if no direct path to the goal existed, this step takes substantially longer. This is caused by the fact that we used a very high penalty for initial path intersections with obstruction to avoid unnecessary environment modifications. Consequently the planner explored a larger portion of the space before intersecting obstructions. If no known obstructions were disconnecting the robot from the goal, the planner could explore the space more efficiently. This behavior is similar to what was observed in [2].

Object Search: Evaluating the point clouds for potential candidate objects to resolve a given obstruction took an average of $2.72s$. In our experiments, the robot needed to repeat this procedure an average of 43 times at different exploration poses before finding a suitable object.

Object Grasp and Drop: Planning for object grasp motions took an average of $1.3s$. As the drop motions were implemented as a reversion of the grasp motion, no planning was necessary for dropping the objects.

B. Failure Cases

While all our experimental setups were solvable in principle, we encountered failure cases. First, we encountered task level failures due to insufficient grasps. As our implementation computes grasp configurations solely based on the cluster information and does not reason about force closure or stable grasp configurations, the object occasionally slipped during locomotion. Frequently, such slippage caused the object to fall into a configuration that did not allow the robot to pick the object up again. If no other object was available in the environment, the robot was not able to exit the building, resulting in task level failure.

Second, the drop motion of the object would not always result in a satisfactory positioning of the object. While, as discussed above, the framework can handle such cases to some degree due to the fact that the algorithm loops and the robot just treats the current environment configuration as a new problem instance, we encountered cases in which the object would critically block the hole, again resulting in task level failure.

We anticipate that future iterations of our implementation will deploy more sophisticated grasp and drop methods.

VI. Conclusion

This paper introduced the first framework that allows robots to reason *online* about using the environment to make progress towards a goal that is not directly reachable. The framework utilizes constraint relaxed planning to detect potential obstructions and the novel concept of inverse affordance mapping to determine the object properties necessary to overcome the obstructions. We presented a complete implementation of the framework in simulation on the PR2 robot, allowing it to solve previously unsolvable problems.

While this paper presents a crucial first step towards allowing robots to achieve the intelligent goal-orientated behavior which is characteristic of human beings, much work remains to be done. We anticipate that future work based on the work presented in this paper will extend the framework to reason about multiple object solutions and allow for more reliable object grasp and drop planning. In addition, we plan to extend the framework to support reasoning about information gathering actions. A robot typically cannot directly perceive material properties of an environment object and therefore should test an object's applicability to the task before deciding to use it. For example, a real robot should test the strength of a board before deciding to use it to cross a gap.

Nevertheless, the core reasoning methods described in this work are vital for robots to operate autonomously in unstructured environments. If in the near future robots are to replace humans in dangerous search and rescue missions, assist humans in household environments, work at construction sides or operate more autonomously over large distances, robots need to be able to get to their goal by all means necessary. A robot that is kept from reaching a victim or critical instruments and tools just because it ignores its manipulation capabilities during navigation planning, is not a valuable replacement for a human counterpart.

Acknowledgments

This work is dedicated to the memory of Mike Stilman, whose encouragement, support and enthusiasm will never be forgotten.

References

[1] S. Koenig and M. Likhachev, "d^* lite," in *Proceedings of the national conference on artificial intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 476–483.

[2] M. Levihn, K. Nishiwaki, S. Kagami, and M. Stilman, "Autonomous environment manipulation to assist humanoid locomotion," in *IEEE International Conference on Robotics and Automation*, 2014.

[3] G. Wilfong, "Motion planning in the presence of movable obstacles," in *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1988, pp. 279–288.

[4] E. Demaine, J. O'Rourke, and M. L. Demaine, "PushPush and Push-1 are NP-hard in 2D," in *In Proceedings of the 12th Canadian Conference on Computational Geometry*, 2000, pp. 211–219.

[5] M. Stilman and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," in *Journal of Humanoid Robotics*, 2004, pp. 322–341.

[6] P. Chen and Y. Hwang, "Practical Path Planning among Movable Obstacles," in *In Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp. 444–449.

[7] H.Wu, M. Levihn, and M. Stilman, "Navigation Among Movable Obstacles in Unknown Environments," in *IROS*, October 2010.

[8] Y. Kakiuchi, R. Ueda, K. Kobayashi, K. Okada, and M. Inaba, "Working with movable obstacles using on-line environment perception reconstruction using active sensing and color range sensor," in *IROS*, 2010.

[9] M. Levihn, M. Stilman, and H. Christensen, "Locally optimal navigation among movable obstacles in unknown environments," in *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[10] G. E. Cook, "Robotic arc welding: research in sensory feedback control," *Industrial Electronics, IEEE Transactions on*, no. 3, 1983.

[11] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, "Automated robot trajectory planning for spray painting of free-form surfaces in automotive manufacturing," in *IEEE Int. Conf. on Robotics and Automation*, 2002.

[12] M. A. Diftler, C. Culbert, R. Ambrose, R. Platt Jr, and W. Bluethmann, "Evolution of the nasa/darpa robonaut control system," in *IEEE Int. Conf. on Robotics and Automation*, 2003.

[13] B. Davies, S. Harris, W. Lin, R. Hibberd, R. Middleton, and J. Cobb, "Active compliance in robotic surgerythe use of force control as a dynamic constraint," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 211, no. 4, 1997.

[14] R. Alterovitz, K. Goldberg, and A. Okamura, "Planning for steerable bevel-tip needle insertion through 2d soft tissue with obstacles," in *IEEE Int. Conf. on Robotics and Automation*, 2005.

[15] M. Levihn and M. Stilman, "Using environment objects as tools: Unconventional door opening," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[16] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[17] E. J. Gibson, "The concept of affordances in development: The renascence of functionalism," in *The concept of development: The Minnesota symposia on child psychology*, vol. 15. Lawrence Erlbaum Hillsdale, NJ, 1982, pp. 55–81.

[18] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3060–3065.

[19] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[20] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4–10, 2012.

[21] M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner, "Planning and Executing Navigation Among Movable Obstacles," in *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS 06)*, October 2006, pp. 820 – 826.

[22] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[23] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Not seeing is also believing: Combining object and metric spatial information." ICRA, 2014.

[24] ——, "Manipulation-based active search for occluded objects," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2814–2819.

[25] J. Kuffner Jr and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, 2000.

[26] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[27] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.