Efficient Opening Detection

Martin Levihn and Mike Stilman

***Abstract*—We present an efficient and powerful algorithm for detecting openings. Openings indicate the existence of a new path for the robot. The reliable detection of new openings is of great relevance for the domain of moving objects [2] as a robot typically needs to detect openings for itself to navigate through. It is also especially relevant to the domain of Navigation Among Movable Obstacles in known [8] as well as unknown [3] environments. In these domains a robot has to plan for object manipulations that help it to navigate to the goal. Tremendous speed-ups for algorithms in these domains can be achieved by limiting the considerations of obstacle manipulations to cases where manipulations create new openings. The presented algorithm can detect openings for obstacles of arbitrary shapes being displaced or moving by themselves, in arbitrarily directions in changing environments.**

To the knowledge of the authors, this is the first algorithm to achieve efficient opening detection for arbitrary shaped obstacles.

I. Introduction

Efficiently detecting openings is a vital requirement for multiple domains. In the domain of moving objects [2] a robot has to navigate to a goal configuration in an environment with moving objects. To reach the goal the robot has to navigate through openings. Efficiently detecting when new openings appear is essential for this domain.

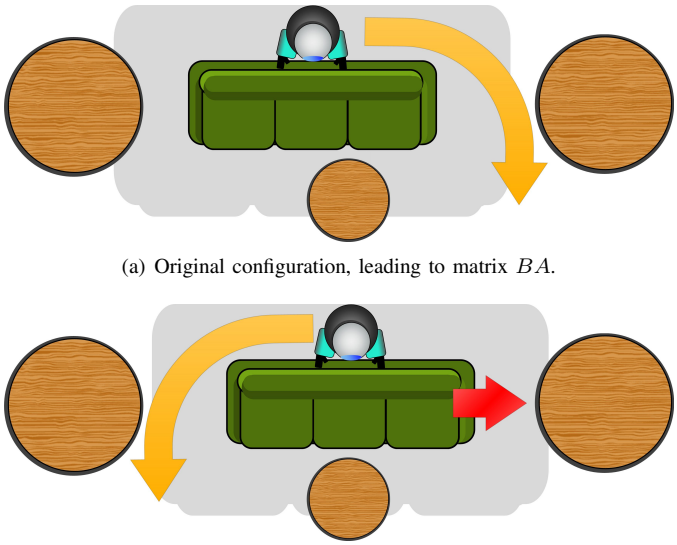
Detecting openings is also crucial for algorithms in the domain of Navigation Among Movable Obstacles (NAMO) [8]. Planners developed for the NAMO domain drastically improve a robot's navigation capabilities by allowing it to reason about its environment and manipulating it in order to reach a goal configuration, just as humans do. However, this domain's state space is exponential in the number of objects and therefore difficult to handle. Consequently all state-of-the-art planners for this domain, whether for known or unknown environments, ([8], [9], [3], [1], [5]) are relying on techniques to reduce the searched portion of the state space. An obvious and effective approach, used by all the aforementioned NAMO planners, is to only consider obstacle manipulations where the resulting displacement of the obstacle is creating a new opening, and therefore potentially a new path to the goal.

Besides being a vital part of almost all algorithms of the NAMO domain, there is no simple, general and efficient algorithm to detect such new openings.

To simplify the following discussions and to pay tribute to the complexity of the domain, we will just focus on the NAMO domain. The reader may keep in mind that the following discussions can be modified readily to suit the domain of moving obstacles.

We propose an algorithm that can detect openings not just for a specific scenario but in general. Our proposed method detects openings for arbitrarily shaped objects, being displaced in arbitrary directions in discretized known or unknown environments in an efficient manner.

The authors are with the Robotics and Intelligent Machines Center in the Department of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA. email: levihn@gatech.edu, mstilman@cc.gatech.edu



(a) Original configuration, leading to matrix BA.

(b) Configuration after manipulation leading to matrix BA_s.

Fig. 1. Example setup. The robot is moving the couch to the right and checks for new openings. Gray: the extended object.

The algorithm is based on the idea of tracking areas that prevent the robot from passing the currently manipulated obstacle over multiple displacements. The area tracking is done by simple operations on matrices. To ensure the algorithms applicability to the domain of NAMO in unknown environments without sacrificing efficiency, these areas are constructed without requiring all known objects to be represented in configuration space. This eliminates the necessity to constantly recompute the configuration space representations of obstacles, as the shape and size of obstacles typically changes frequently for the domain of NAMO in unknown environments.

However, the proposed algorithm only considers the local neighborhood of the currently manipulated obstacle. This might result in false positives as the global path may still be blocked in latter sections of the path. This is inherent in all opening detection algorithms that do not repeatedly construct full paths to the goal. Nevertheless, our algorithm will not produce false negatives in reference to the global path, making it applicable for optimal planners.

This work assumes that the robot is circular, as in all NAMO planning papers known to the authors. Extensions of the algorithm presented here for non-circular robots are possible.

Our contributions are a general, efficient and simple algo-

rithm for opening detection that can be used in a variety of domains and algorithms along with a formal definition of a new opening.

The remainder of the paper is organized as follows. First Section II will provide on overview of related work. After formally defining openings in Section III, Section IV will provide an overview of the algorithm. Section V guides the reader through a detailed implementation of the algorithm accompanied by the example shown in Fig. 1. Usage of the proposed algorithm in the most challenging NAMO domain known the authors is presented in Section VI prior to concluding the paper in Section VII.

II. Related Work

Opening detection is a vital part of all NAMO applications as the manipulation of obstacles is just considered *because* it can create new openings and therefore new paths to the goal. We will consequently provide an overview of methods that have been used for different NAMO domains and algorithms.

Stilman, et.al. presented an algorithm in [8] that is capable of detecting openings for the domain of NAMO in known environments. The algorithm relies on local search after each considered object manipulation. An A^* [7] call is triggered for each manipulation action to check if two free-space regions have been combined. This is highly inefficient. Given that checking for openings is done for each possible manipulation of an obstacle this results in frequent searches within the NAMO search itself.

Wu, et.al. utilized opening detection in [3] for the domain of NAMO in unknown environments. The algorithm did not rely on search but simply observed the amount of adjacent free spaces on corners of the manipulated obstacle. While efficient, this algorithm is only applicable for world configurations populated with simple rectangular shaped static and movable obstacles. This is not realistic.

In [5] NAMO in unknown environments was implemented on an HRP-2. Opening detection was again performed through repeated calls to the motion planner. In contrast to [8], the motion planner attempted to actually construct a full path to the goal for each manipulation. This is intractable for realistically sized scenarios as it, again, performs frequent searches within a search itself.

Our algorithm can be used on all the aforementioned works without modification and is expected to introduce substantial runtime savings. It could further be used on extensions of these works, as for example arbitrarily shaped obstacles in [3].

III. Definitions

Prior to discussing our algorithm in the following sections, we will provide a formal definition of a *new opening*. The definition is based on the notion of *homotopic* paths, which will be defined below following general definitions.

A. General Definitions

We consider a world $\mathcal{W}$ as a 2D-workspace populated with:

- **Static obstacles:** $\mathcal{O} = \{O_0, O_1, ..., O_k\}$
- **Movable obstacles:** $\mathcal{M} = \{M_0, M_1, ..., M_n\}$
- **Robot:** $\mathcal{R}$

A world configuration at time t ($q^t_{\mathcal{W}}$) is defined as:

$$q^t_{\mathcal{W}} = \{q^t_{\mathcal{R}}, q^t_0, q^t_1, ..., q^t_n\}$$

where $q^t_{\mathcal{R}}$ and q^t_i denotes $\mathcal{R}$ and M_i positions at time t, respectively. We assume $\mathcal{R}$ to be circular.

Definition 1 (Path)**.** *Given a robot's start configuration $q^s_r \in q^t_W$ and goal configuration $q^g_r \in q^t_W$ a path τ is a continuous function $\tau : [0, 1] \rightarrow q_r \in q^t_W$ that follows $\tau[0] = q^s_r$ and $\tau[1] = q^g_r$.*

This definition is adopted from [6]. Note that the term path is used in reference to the start and goal, in comparison to a general path.

Definition 2 (Set of Paths)**.** *$\mathcal{T}^t$ is the set of all paths in q^t_W.*

B. Opening Definition

Definition 3 (Homotipc Paths)**.** *Two paths $\tau_1, \tau_2 \in \mathcal{T}^t$ are homotopic if and only if there exists no known obstacle in the area enclosed by the paths. Otherwise they are ahomotopic.*

This definition is adopted from [4]. Based on this definition the definition of a new opening can be given.

Definition 4 (New Opening)**.** *A new opening at time $t + 1$ exists if and only if the set $\mathcal{T}^{t+1}$ has at least one path that is ahomotopic to all the paths in $\mathcal{T}^t$.*

C. Discussion

Definition 4 is given in accordance to the set $\mathcal{T}$ and as such in relation to the goal. Every opening detection algorithm used for any NAMO domain is expected to return openings according to def. 4.

No false negatives should occur as this will potentially influence properties of the NAMO planner such as optimality. False positives on the other side, while ideally kept at a minimum, are of limited impact. The optimality of a NAMO planner itself is generally not influenced by false positives, as the planer would typically detect that no lower cost plan can be constructed [3]. Additionally, false positives could always be rejected in a verification step that checks if a new path to the goal was actually created.

IV. Algorithm

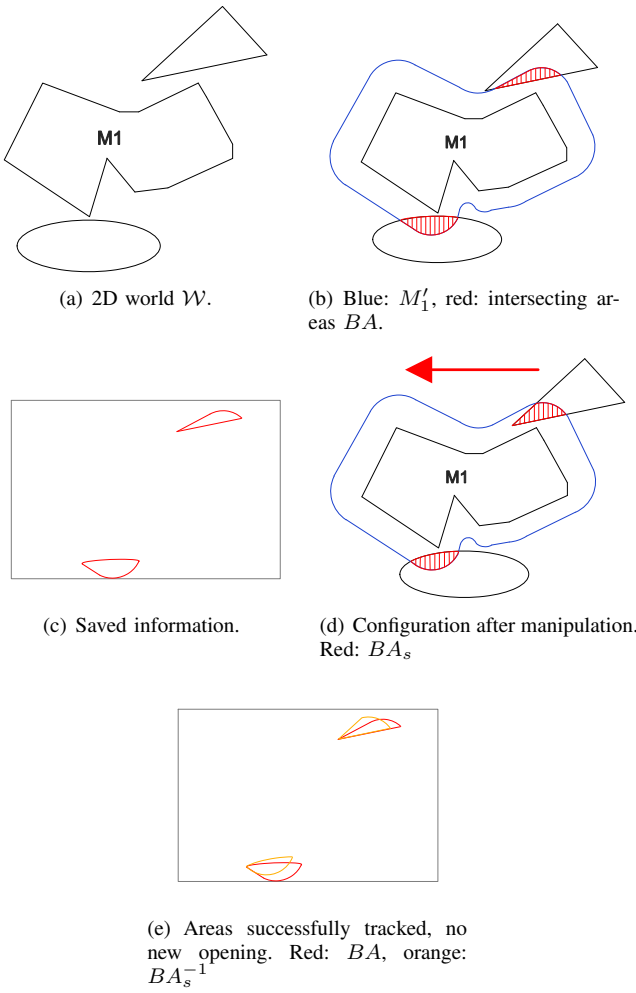
A. Outline

The inputs to the following algorithm are

1) an occupancy grid map G of the workspace $\mathcal{W}$
2) a movable obstacle M_i and
3) a single or set of manipulation actions $\mathcal{A}_M$.

The explicit definitions are dependent on the NAMO planner utilizing the algorithm and are not necessary for the following discussion.

The output of the algorithm is a simple binary variable indicating if executing $\mathcal{A}_M$ on M_i in $\mathcal{W}$ creates a new opening or not.



(a) 2D world $\mathcal{W}$.

(b) Blue: M_1', red: intersecting areas BA.

(c) Saved information.

(d) Configuration after manipulation. Red: BA_s

(e) Areas successfully tracked, no new opening. Red: BA, orange: BA_s^{-1}

Fig. 2. Example of opening detection

The core idea of the proposed opening detection algorithm is to track areas that prevent the robot from passing the currently manipulated obstacle over different configurations of that obstacle. If, as a consequence of a manipulation, such an area disappears then a new opening is reported.

B. Detecting Blocking Areas

As the proposed algorithm is intended as a general framework for NAMO planners, minimum assumptions about the planner itself are made. For example, no representation of the configuration space is required. Planners for the domain of NAMO in unknown environments may not plan in configuration space and consequently may not be able to provide our algorithm with such a representation. This is due to the fact that for the domain of NAMO in unknown environments the obstacle shapes, as perceived by the robot, may change frequently as more information becomes available. Constantly maintaining an accurate configuration space representation would therefore be computationally expensive.

To find the blocking areas we extend the obstacle M_i by the robots *diameter*, yielding M_i'. Note, we extend M_i *not by the radius*. This is not the typical Minkowski sum. However, by extending M_i by the robots diameter we do not need to incur the computational overhead of constructing the full configuration space or to reason about which portion of the configuration space is relevant for the algorithm. Informally speaking, we can now simply overlay M_i' on $\mathcal{W}$. Intersections yield the areas that are blocking the robot from passing M_i.

Fig. 2(a)-2(b) show this concept. Fig. 2(a) shows a 2D setup for which the algorithm is trying to detect new openings for manipulations of M_1. Fig. 2(a) shows M_1' in blue as M_1 is being extended by the robots diameter (robot not shown). The red areas are the blocking areas BA. These areas are now saved, Fig. 2(c).

C. Tracking Areas and Detecting Openings

Once the initial blocking areas are determined, the obstacle is shifted according to $\mathcal{A}_M$. The shifted M_1' is again intersected with $\mathcal{W}$. The resulting blocked areas BA_s are determined, Fig. 2(d). The areas BA_s are now shifted back according to $\mathcal{A}_M^{-1}$, yielding BA_s^{-1}. This allows us to directly compare the areas that block the robot from passing the obstacle, eliminating the offset. BA and BA_s^{-1} can now simply be compared, Fig. 2(e). If every area in BA has an intersection with an area in BA_s^{-1}, no new opening is detected, otherwise a new opening is reported. This is because a new opening occurs if at least one area that prevented the robot from passing the robot prior to the manipulation disappeared after the manipulation.

These steps can be easily and efficiently implemented as the next section will show.

D. Discussion

The presented algorithm works by observing the local space adjacent to M_i for the manipulation action $\mathcal{A}_M$. This is sufficient in finding all true positives according to definition 4. However, as the full world is not taken into account, this introduces the risk of false positives.

The authors nevertheless followed this approach because of

- the computation time advantages of only considering the local space,
- the resulting independence of the actual world size,
- the limited impact of false positives on a NAMO planner and
- the option of a post verification step for false positives rejection if desired.

V. Implementation and Example

This section will provide implementation details, Algorithm 1, and use Fig. 1 to demonstrate each step. We assume that W is represented by an occupancy grid G and M_i' as a binary matrix M. M has the size of the bounding box of M_i' and uses unity valued entries to represent parts of M_i. This allows for arbitrarily shaped obstacles to be treated equally by the algorithm below. M only needs to be constructed once and can be saved for later use in all NAMO domains. However, in the NAMO in unknown environments domain, M has to be recomputed if more information about M_i becomes available. This is summarized in the function GET-M_i'-Matrix in line 1.

Algorithm 1: CHECK-NEW-OPENING(G, M_i, $\mathcal{A}_M$)

1: M =GET-M_i'-MATRIX(M_i);
2: $x\_offset = M_i.x$;
3: $y\_offset = M_i.y$;
4: BA =GET-BLOCKING-AREAS(x_offset, y_offset)
5: x_offset =GET-NEW-X-POS($\mathcal{A}_M$, M_i);
6: y_offset =GET-NEW-Y-POS($\mathcal{A}_M$, M_i);
7: BA_s =GET-BLOCKING-AREAS(x_offset, y_offset)
8: $BA_s^* = [0][0]$; {0-Matrix; dim()=dim(M)}
9: **for** $i = 0 \rightarrow |BA_s^*|$ **do**
10: **for** $j = 0 \rightarrow |BA_s^*[i]|$ **do**
11: $x = (x\_offset - M_i.x) + i$;
12: $y = (y\_offset - M_i.y) + j$;
13: **if** $0 < x < |BA_s^*|$ AND $0 < y < |BA_s^[]x] * |$ **then**
14: $BA_s^*[x][y] = BA_s[i][j]$;
15: **end if**
16: **end for**
17: **end for**
18: Z =COMPARE(BA, BA_s^*);
19: **if** $Z \equiv [0][0]$ **then**
20: return $false$;
21: **end if**
22: return $true$;

Function GET-BLOCKING-AREAS(x_off, y_off)

1: $index = 1$;
2: $BA = [0][0]$; {0-Matrix; dim()=dim(M)}
3: **for** $x = 0 \rightarrow |M|$ **do**
4: **for** $y = 0 \rightarrow |M[x]|$ **do**
5: **if** $M[x][y] \neq 0$ AND $G[x + x\_off][y + y\_off] \neq 0$ **then**
6: ASSIGN-NR(BA, x, y, $index$);
7: **end if**
8: **end for**
9: **end for**
10: return BA;

Function ASSIGN-NR(BA, x, y, $index$)

1: **for** $i = -1 \rightarrow 1$ **do**
2: **for** $j = -1 \rightarrow 1$ **do**
3: **if** $BA[x + i][y + j] \neq 0$ **then**
4: $BA[x][y] = BA[x + i][y + j]$;
5: return;
6: **end if**
7: **end for**
8: **end for**
9: $BA[x][y] = index$;
10: $index = index + 1$;
11: return;

Function COMPARE(BA, BA_s^*)

1: $del\_num := \emptyset$;
2: **for** $x = 0 \rightarrow |BS|$ **do**
3: **for** $y = 0 \rightarrow |BS[x]|$ **do**
4: **if** $BA[x][y] \in del\_num$ **then**
5: $BS[x][y] = 0$;
6: **end if**
7: **if** $BA[x][y] \neq 0$ AND $BA_s^*[x][y] \neq 0$ **then**
8: $del\_num = del\_num \cup BS[x][y]$;
9: $BS[x][y] = 0$
10: **end if**
11: **end for**
12: **end for**
13: return BA;

A. Blocking Areas

To determine the blocking areas, an empty matrix BA with dimensions equal to M is constructed and initialized, line 2.

The algorithm now iterates over M, line 3-4. For all indices that show a non-zero entry in M, the corresponding index in G is checked for blockage. This is done by simply adding the offset for the object to the indices, line 5. If this is the case, a number unequal to zero is assigned to the corresponding entry in BA. The number is assigned based on algorithm ASSIGN-NR.

Assignment is performed based on the 3×3 neighborhood of the currently to assign entry in BA. If a number has already been assigned within this neighborhood, the same number is assigned. In case no number has been assigned yet, a number not used in BA so far is assigned. This encodes the blocking areas. For the example in Fig. 1(a) the following matrix is obtained:

$$BA = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{2} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{2} & \mathbf{2} & \mathbf{2} & \mathbf{2} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The offsets are now updated according to $\mathcal{A}_M$, line 5-6 and BS_s obtained. For the example in Fig. 1(b) the following matrix is constructed:

$$BS_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{2} & \mathbf{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{2} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

B. Opening Detection

Openings are then detected by shifting BA_s according to $\mathcal{A}^-1$, resulting in line 9-17.

$$BA_s^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

BA_s^* and BS are then compared using the function COMPARE, which checks for *non-zero entries* in both matrices. If such an entry is detected, all entries in BS having the same

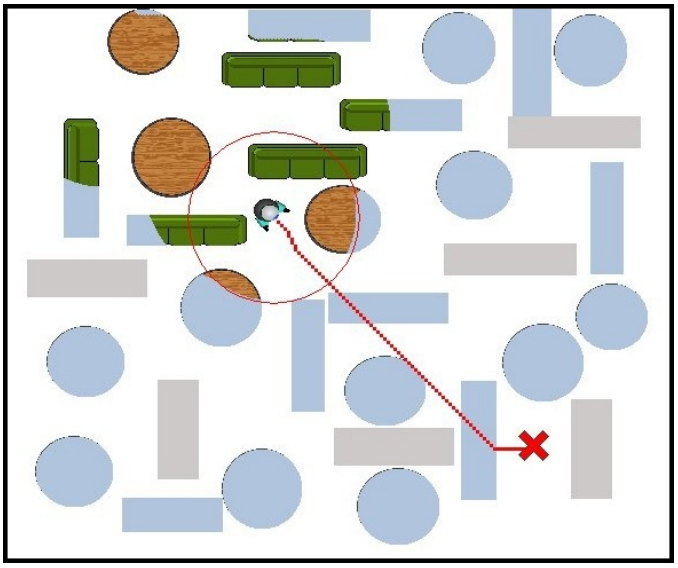


Fig. 3. Execution example of the used simulator. Grayed out objects are not known to the robot.

number as detected are set to 0. This is done because if a part of a previous blocking area is detected, the robot is still blocked by the same cause, no matter how much of the area still exists.

Let Z denote the resulting matrix. For the example in Fig. 1 the following Z is obtained:

$$Z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

If Z equals the zero-matrix after this operation, no new openings were detected, as all blocking areas are still accounted for after the manipulation. If Z does not equal the zero matrix, one intersecting area could not be found anymore and the possibility of an opening is returned. Since the matrix Z obtained for example Fig. 1 is not equal to the zero-matrix, an opening is detected.

C. Discussion

The provided description above is intentionally very detailed. An actual implementation can summarize many of these steps. Additionally, BS can be saved for repeated calls to the function with different manipulation actions as long as the local environment around M_i and M_i itself does not change.

Note that the algorithm does not make any assumptions about the shape of M_i, or the manipulation direction of $\mathcal{A}_M$.

VI. Evaluation

In order to verify the effectiveness of our algorithm, we did implement it in the most challenging NAMO domain we are aware of: NAMO in unknown environments, allowing arbitrary displacements of arbitrarily shaped obstacles. Note that no opening detection algorithm exists for this domain so far.

The NAMO algorithm reasons about objects by iterating over the objects, evaluating possible manipulations of each object and finally executing the lowest cost plan. This procedure is repeated whenever the currently executed plan is being intercepted by newly detected obstacles. This is an extension of [3]. Fig. VI shows an execution example of the simulator. Grayed out objects are not known to the robot. The robot constantly receives information about the part of the environment that is within its sensor range, indicated by the red circle around the robot. The robot continuously updates its internal map. Based on that map it reasons about the obstacles.

Our experiments were performed on 50 randomly generated environments, with complexity ranging from simple maps with only two obstacles to complex maps with more than 70 obstacles. We ran the algorithm both with our opening detection algorithm and without.

We measured:

1) the overall runtime savings,
2) the savings in obstacle evaluations, and
3) the savings in navigation planner calls.

As the algorithm reasons about obstacles in the environment, (2) measures whether the amount of obstacles that are being reasoned about are affected at all by our algorithm. (3) measures how frequently the navigation planner is called. In general, the navigation planner is called if the NAMO planner considers a different path to the goal and needs to determine its actual cost.

The effect of opening detection depends on the map configuration. For maps where the robot hardly encounters any obstacles on its path to the goal, the NAMO planner will not reason about obstacle manipulations frequently. The opening detection algorithm can therefore not take strong affect. For denser maps, in contrast, the NAMO planner will frequently reason about obstacle manipulations and opening detection can introduce substantial savings.

Consequently, we evaluated the opening detection algorithm in relation to the ratio between navigation actions and manipulation actions that are being executed by the robot while navigating to the goal. If the ratio is low, this indicated that the environment was dense and the robot had to frequently manipulate objects to reach the goal, and therefore reason about the obstacles. The opposite is true for a high ratio.

Fig. 4 shows the results. The graph is interpolated and bezier-smoothed to allow a clearer observation of the tendencies. We can clearly see that for world configurations with a higher ratio the savings introduced by our algorithm decrease. However, for more dense environments, yielding a lower ratio, our algorithm introduces substantial savings. The runtime savings tend to be lower than the navigation planner calls. This is due to the computations performed by the opening algorithm itself.

The number of obstacle evaluations proved to not be affected by our algorithm.

VII. Conclusion

We presented a simple opening detection algorithm and demonstrated it for the NAMO domain. Additionally, a formal definition of a new opening based on the concept of homoptic paths was provided.

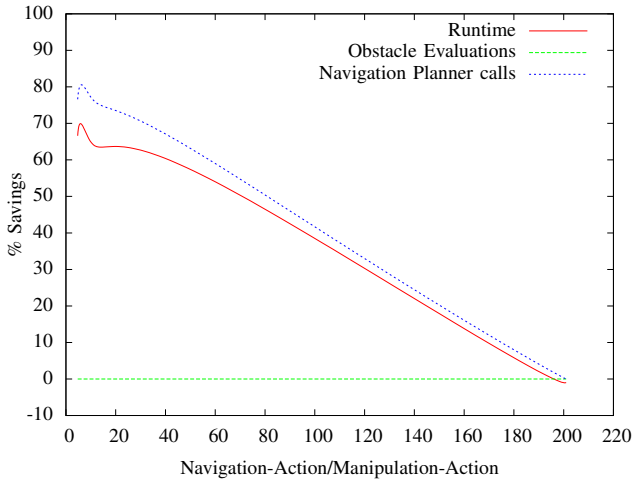


Fig. 4. savings if opening detection is used

The presented algorithm is capable of handling arbitrarily shaped obstacles, being displaced in arbitrary directions. At the same time the algorithm is easy to understand, implement and is directly applicable to the domain of NAMO with known environments as well as unknown environments. The algorithm does not require a representation of the configuration space, as not always present for NAMO planner in unknown environments. Rather, the algorithm is simply creating a representation of the currently considered obstacle increased by the robots diameter. This representation is intersected with the occupancy grid map of the world and intersecting areas, which would block the robot from passing the obstacle, are tracked. Upon the disappearance of such an area a new opening is reported. While this may result in false positives, it will not cause false negatives. We also demonstrated a simple implementation of the algorithm.

The algorithms applicability was shown for the most complex NAMO domain currently considered in research, NAMO in unknown environments with arbitrarily shaped obstacles being displaced in arbitrarily directions. The results showed substantial savings for environments that require frequent manipulations.

The presented algorithm can be used in all NAMO planners the authors are aware of.

REFERENCES

[1] Salvatore Candido, Yong-Tae Kim, and Seth Hutchinson. An improved hierarchical motion planner for humanoid robots. In *Proceedings of the IEEE-RAS Internation Conference on Humanoid Robots*, Daejeon, Korea, December 2008.

[2] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles [mobile robot]. *Robotics and Automation, IEEE Transactions on*, 5(1):61 –69, feb 1989.

[3] H.Wu, M. Levihn, and M. Stilman. Navigation among movable obstacles in unknown environments. In *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS 10)*, October 2010.

[4] Takeo Igarashi and Mike Stilman. Homotopic path planning on manifolds for cabled mobile robots. In David Hsu, Volkan Isler, Jean-Claude Latombe, and Ming Lin, editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 1–18. Springer Berlin / Heidelberg, 2011.

[5] 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 *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1696 –1701, oct. 2010.

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

[7] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[8] M. Stilman and J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robotics (Humanoids'04)*, volume 1, pages 322 – 341, December 2004.

[9] Mike Stilman and James J. Kuffner. Planning among movable obstacles with artificial constraints. *I. J. Robotic Res.*, 27(11-12):1295–1307, 2008.