Path Algebra for Mobile Robots

Frederic Maire and Gavin Suddrey

School of Electrical Engineering and Computer Science,
Science and Engineering Faculty, Queensland University of Technology,
Gardens Point, Brisbane, 4000, Australia
f.maire,g.suddrey@qut.edu.au
http://www.qut.edu.au

Abstract. In this paper, we introduce a path algebra well suited for navigation in environments that can be abstracted as topological graphs. From this path algebra, we derive algorithms to reduce routes in such environments. The routes are reduced in the sense that they are shorter (contain fewer edges), but still connect the endpoints of the initial routes. Contrary to planning methods descended from Disjktra's Shortest Path Algorithm like $D^\star$, the navigation methods derived from our path algebra do not require any graph representation. We prove that the reduced routes are optimal when the graphs are without cycles. In the case of graphs with cycles, we prove that whatever the length of the initial route, the length of the reduced route is bounded by a constant that only depends on the structure of the environment.

1 Introduction

Most current navigation systems generate global, metric representation of the environment with either obstacle-based grid maps or feature-based metric maps [4]. While suitable for small areas, global metric maps have inefficiencies of scale [10, 8]. Arguably, topological mapping is more efficient as it concisely represents a partial view of the world as a graph [14]. The vertices of the graph may correspond to anchored points in the environment with physical meaning that the robot can precisely navigate to, like a door or a corridor intersection as illustrated in Figure 1. The vertices can also correspond to the vertices of a Voronoi diagram [6].

In this paper, we focus on mobile robots equipped with a sensor suite enabling the robots to

- detect when they reach a topological node.
- determine the degree of a node (number of incident edges).
- select an edge, and drive along this edge.

We show that when the graph is without cycles, a map is superfluous for navigation purpose. More precisely, we define a path algebra that provides a very compact representation of the route followed by the robot. This path algebra enables the computation of return routes as well as the automatic simplification of

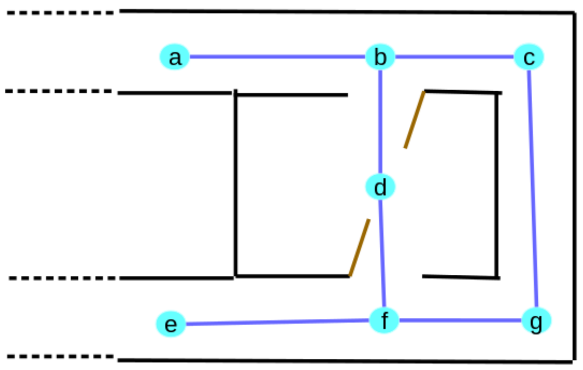

Fig. 1. A topological map derived from a floor map. The node *d* represents a room. The two doorways of this room correspond to the edge *bd* and the edge *df*.

long routes containing detours. We also extend this method to graph containing cycles.

In Section 2, we review previous work on mapping and localisation. In Section 3, we introduce a path algebra. In Section 4, we demonstrate how the path algebra can be used on different types of graphs.

2 Related Work

The major mobile robot navigation schemes can roughly be classified into two categories: navigation with position information and navigation without position information [9].

Metric maps belong to the first category. They are popular because they are suitable for path planning with a high degree of accuracy. Their drawback is that they are often expensive to calculate, and do not scale well in large, unstructured and dynamic environments like most outdoor places [1].

The alternative to metric maps are topological maps, which are graphs that in their pure form do not store metric information. Topology is mainly concerned with the connectivity properties of a space. In the context of an indoor mobile robot, the space refers to the expanse containing fixed structural components of the building like rooms, walls, doors and corridors, as well as the mobile entities like people, furniture and equipment [15]. Distinct places are represented as nodes, adjacency between different locations is represented by the graph edges (see Figure 1). Because of their sparse representation, topological maps can be a memory efficient representation of the environment and provide good scalability [5].

Hybrid approaches build topometric maps, allowing the robot to use the topological information to plan a global path and to exploit metric information to find shortcuts [3].

Appearance-based mapping and localisation approaches include appearance-only systems like FAB-MAP [2] which can be considered as an extreme case of topological map. Topometric systems using a more primitive data association than FAB-MAP, like RatSLAM [12] or CAT-Graph [11] require a rough estimate of the distance travelled. More recently, these approaches have been combined in [7] to achieve better robustness to variation in the environment appearance and changes in illumination and structure.

Although the $D^\star$ algorithm accepts partially known environments, it has to build a graphical representation of its environment to plan paths [13]. The path algebra that we introduce does not have this requirement.

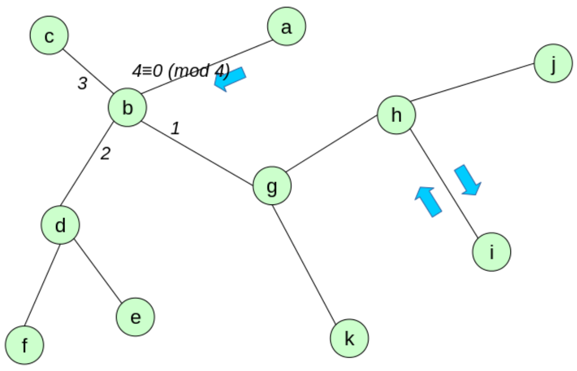

Fig. 2. The route from the arc ab to the arc hi can be coded as the tuple $(1, 1, 2)$. The labels on the edges incident to Node b are the indices with respect to arc ab.

3 Route Representation

When following the instructions of a navigation GPS device, the driver of a car receives instructions of the form "at the next roundabout, take the third exit". This command format is well suited for logging the itinerary followed by a mobile robot in an environment that has a graphical topology like a road-network or the corridors of a building. Figure 2 illustrates such an environment. The agent/robot traverses the graph following its edges and using its vertices as roundabouts. In order to indicate the direction the robot is facing on an edge, we specify the location of the robot by an arc. We adhere to the standard terminology of graph theory, and reserve the term *arc* for an edge that has been given an orientation. The blue arrow on the left of arc $a \to b$ represents the position of the robot going from Node a to Node b.

The navigational route instructions from the arc $a \rightarrow b$ as the starting position, to the arc labeled $h \rightarrow i$ as the destination, would sound as follows if told by a GPS device:

1. "drive to the next the intersection"
2. "take the first left"
3. "drive to the next the intersection"
4. "take the first left"
5. "drive to the next the intersection"
6. "take the second left"
7. "drive to the next the intersection"
8. "you have arrived at your destination"

A less verbose representation would code the route into the integer sequence $(1, 1, 2)$.

3.1 Forward Route

Definition 1 *A* **forward route** *of length k is a sequence of integers* $(c_1, c_2, \ldots, c_k)$ *coding for each node the agent traverses, the relative edge, with respect to the entering edge, at which the agent should leave the node. At the* j^{th} *node of its journey, the agent takes the* c_j^{th} *relative edge counting clockwise from the entering edge.*

The integers $(c_1, c_2, \ldots, c_k)$ are signed. The command c for a node with n edges can be reduced modulo n. In the graph shown in Figure 2, if the agent is on the arc ab, and the driving command c is equal to 6, then the agent will continue its journey via the arc bd. The commands 6 and 2 have the same effect at this node because 6 is equivalent to 2 modulo 4. Formally, we write $6 \equiv 2 \pmod 4$. More generally,

Proposition 1. *When arriving at a node with* n *edges, the commands* c *and* c' *have the same effect if and only if* $c \equiv c' \ (mod\, n)$.

Observing that $-c \equiv n - c \pmod n$, we derive the following special case that becomes useful when backtracking:

Proposition 2. *When arriving at a node with* n *edges, the commands* $-c$ *and* $n - c$ *have the same effect.*

Considering again Figure 2, if the agent is on the arc ab and the driving command c is equal to 0 or 4, then the agent will perform a U-turn at b, and ends up on arc ba. More generally,

Proposition 3. *When arriving at a node with* n *edge, the command* c *will trigger a U-turn if and only if* $c \equiv 0 \ (mod\, n)$.

When the agent arrives at a leaf, the value of the command c does not matter:

Proposition 4. *At a leaf node (node with exactly one incident branch), all commands* c *have the same effect. Whatever the value* c *takes, the agent performs a U-turn.*

Proof. It is enough to observe that $\forall c \in \mathbb{Z}, \quad c \equiv 0 \pmod 1$

3.2 Return Route

While driving, if we take the 2^{nd} leftmost exit at a roundabout during the forward leg of a trip, we should take the 2^{nd} rightmost exit at the same roundabout during the return trip. More generally, suppose that an agent on a forward journey visits the nodes $x_1, x_2, \ldots, x_k$, and that the agent traverses the node x_i with the command c_i on its way forward. If the agent wants to backtrack to its starting position, the agent should traverse the node x_i with the command $-c_i$ on its return journey. In this paper, the reverse arc of an arc α will be denoted by $\overline{\alpha}$. That is, if $\alpha = x \rightarrow y$, then $\overline{\alpha} = y \rightarrow x$.

Proposition 5. *If command c takes an agent positioned on an arc α to an arc β, then the command $-c$ will take an agent positioned on the arc $\overline{\beta}$ to the arc $\overline{\alpha}$.*

Proposition 5 can be generalized to longer paths by a simple induction on the number of nodes in the path.

Proposition 6. *If $(c_1, c_2, \ldots, c_k)$ is a command sequence that takes an agent positioned on an arc α to an arc β via the nodes $x_1, x_2, \ldots, x_k$, then the command sequence $(-c_k, -c_{k-1}, \ldots, -c_1)$ will take an agent positioned on the arc $\overline{\beta}$ to the arc $\overline{\alpha}$ via the nodes $(x_k, x_{k-1}, \ldots, x_1)$.*

3.3 Route Simplification

Consider Figure 3 where a robot starting at the position of the blue arrow executes the command sequence $c = (1, 5, 2)$ that takes the robot through Node a, then to Node b, where a U-turn is made, then back to Node a and onto Node c. In this example, $c_2 = 5$ is equal to the number of incident edges to the second node $x_2 = b$ of the route. Therefore, we could have used the equivalent sequence $c' = (1, 0, 2)$. This sequence can be further reduced to the singleton $c'' = (1 + 2) = (3)$.

More generally, we always have

Proposition 7. *If (c_1, c_2, c_3) is a command triplet that takes an agent positioned on an arc α to an arc β via the nodes x_1, x_2 and x_3; and if moreover $c_2 \equiv 0 \pmod{n_2}$ where n_2 is the degree of the node x_2, then the sequence (c_1, c_2, c_3) has the same effect as the singleton sequence $(c_1 + c_3)$. In other words, if an agent starts on the arc α and executes the single command $(c_1 + c_3)$, it will ends up positionned on the arc β. These conditions also imply that the nodes x_1 and x_3 are the same.*

Proof. Without loss of generality, we assign the index 0 to the arc α entering x_1. When the U-turn at Node x_2 is performed, the robot returns to Node $x_1 = x_3$ via the edge of x_1 indexed c_1. The index of the exit edge is obtained by adding c_3. Therefore the relative index of the arc β with respect to the arc α is $c_1 + c_3$ as illustrated in Figure 4.

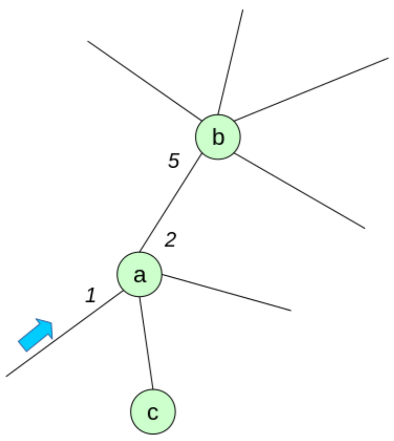

Fig. 3. The robot starts at the location of the blue arrow. The integer on the left of an arc corresponds to the command c_i for the i^{th} node. In this, example when arriving at Node a the robot takes the first exit, then follows the arc $a \to b$ to Node b where it takes the 5^{th} exit. This U-turn brings back the robot on the arc $b \to a$. At the second visit of Node a, the robot takes the 2^{nd} exit with respect to arc $b \to a$. The whole sequence $c = (1, 5, 2)$ can be contracted into $c = (3)$.

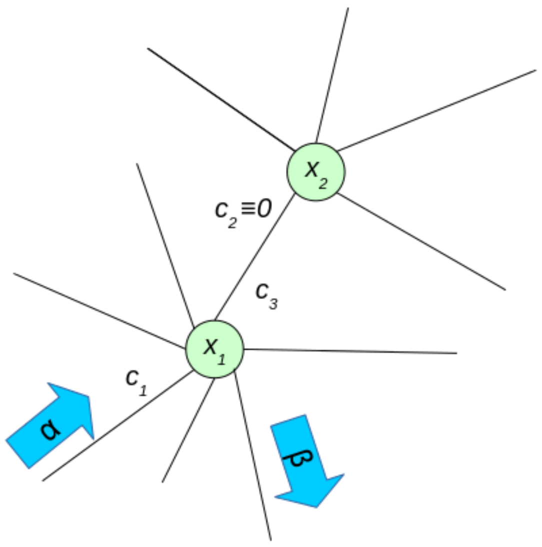

Fig. 4. Sequence reduction; whenever $c_2 \equiv 0 \pmod{n_2}$ where n_2 is the degree of the node x_2, the sequence (c_1, c_2, c_3) has the same effect as the singleton sequence $(c_1 + c_3)$.

Often, we are mainly concerned with the starting arc α and the destination arc β of a given route c. This leads us to the following definition;

Definition 2 *If two routes c and c' start with the same arc α and end with the same destination arc β, then the routes are said to be* equivalent. *We will write $c \sim c'$ to express this equivalence.*

In Figure 5, the robot starts on the arc labeled 1, and executes the command sequence $c = (1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1)$ that leads the robot to the arc labeled 16. Proposition 7 allows us to simplify the route by iteratively reducing command triplets of the form $(c_{i-1}, 0, c_{i+1})$. The first occurence of 0 is in the triplet $(1, 0, 2)$. The triplet reduces to (3). This first reduction shows that the sequence c is equivalent to the sequence $(3, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1)$. Further reductions can be performed. The next occurence of 0 is $(2, 0, 2)$ which reduces to (4). Therefore, we now have $c \sim (3, 4, 0, 2, 3, 2, 0, 2, 1, 0, 1)$. Continuing the reduction, $(4, 0, 2)$ is replaced with (6). Therefore, $c \sim (3, 6, 3, 2, 0, 2, 1, 0, 1)$. The next reduction replaces $(2, 0, 2)$ with (4), and entails that $c \sim (3, 6, 3, 4, 1, 0, 1)$. Finally, $(1, 0, 1)$ is replaced with (2), yielding $c \sim (3, 6, 3, 4, 2)$. Although, our automatic substitutions have eleminated all occurences of 0, we have not completely exploited Proposition 7. Indeed, the degree sequence $(4, 3, 4, 3, 3)$ of the nodes traversed in the reduced route $(3, 6, 3, 4, 2)$ requires more attention.

As $6 \equiv 0 \pmod 3$, we have $(3, 6, 3, 4, 2) \sim (3, 0, 3, 4, 2) \sim (6, 4, 2)$. As the degree sequence of the nodes traversed for the command sequence $(6, 4, 2)$ is $(4, 3, 3)$, c can be put in the more canonical form $(2, 1, 2)$. To sum up, by iterating the reduction described in Proposition 7, we have shown that the command sequence $c = (1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1)$ is equivalent to the command sequence $(2, 1, 2)$.

From this toy example, we can generalize our approach to a generic algorithm for the minimization of any route (pseudo-code listed in Algorithm 1).

4 Navigation Applications

In this section, we consider in turn the navigation task on graphs without cycles, then on graphs with cycles.

4.1 Navigation on Trees

Given a route $\mathcal{R} = (c_1, c_2, \ldots, c_n)$, we write $\overline{\mathcal{R}} = (-c_n, -c_{n-1}, \ldots, -c_1)$ to denote the reverse of route $\mathcal{R}$.

Proposition 8. *If a route $\mathcal{R}_1$ takes a robot from a place α_0 to a place α_1, and the route $\mathcal{R}_2$ takes the robot from the same α_0 to a second place α_2, then the route $\overline{\mathcal{R}_1}\mathcal{R}_2$ will take the robot from the $\overline{\alpha_1}$ to α_2.*

The place α_0 could be the base of the robot (charging station), with α_1 and α_2 being places of interest for which the robot has stored the respective routes $\mathcal{R}_1$ and $\mathcal{R}_2$. By applying Algorithm 1 on $\overline{\mathcal{R}_1}\mathcal{R}_2$ we derive a direct route from $\overline{\alpha_1}$ to α_2.

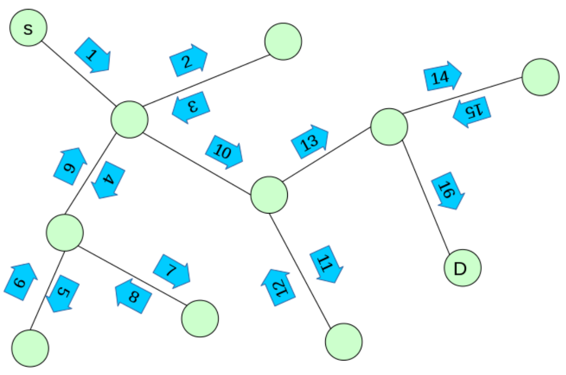

Fig. 5. A route traversing 15 nodes taking the robot from the arc labeled 1 to the arc labeled 16. The command sequence is $c = (1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1)$. Thanks to Proposition 7, we can reduce the long route $(1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1)$ to the direct route $(2, 1, 2)$.

4.2 Navigation on Graphs with Cycles

In Section 3, we introduced a route reduction algorithm that is applicable to any graph. However, it is only for trees (connected graphs without cycles) that we can guarantee that the returned contracted route corresponds to the shortest path between the starting arc α and the destination arc β. Without extra information, it is impossible to detect a loop closure.

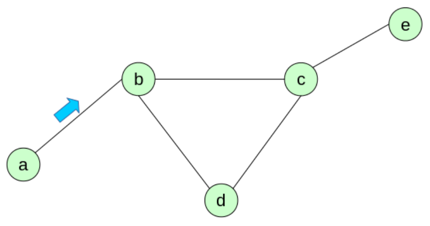

Fig. 6. Starting on the blue arrow, the route $c = (1, 2, 1, 2, 1)$ brings the robot to the arc $c \to e$. Unfortunately, Algorithm 1 cannot simplify c into the equivalent route $c' = (1, 1)$. Moreover, from a topological point of view, the robot cannot distinguish the journey on this graph from the journey experienced on the graph of Figure 7.

The route $c = (1, 2, 1, 2, 1)$ executed on the graphs of Figure 6 and Figure 7 provides the same topological experience to the robot, in the sense that the robot visits two sequences of nodes with identical degree sequences, and selects edges with the same relative indices in the two cases.

```
input :
a route with the degrees of the traversed nodes
$\mathcal{C}$ : a command sequence $(c_1, c_2, \ldots, c_n)$ where the $c_i$'s are integer values
(possibly negative)
$\mathcal{D}$ : a degree sequence $(d_1, d_2, \ldots, d_n)$ where $d_i$ is the number of edges incident
to the $i^{th}$ traversed node
output:
the shortest route $\mathcal{C}'$ equivalent to $\mathcal{C}$
1  begin
2      $\mathcal{C}' = \mathcal{C}$                          /* initialize the reduced route */
3      $\mathcal{D}' = \mathcal{D}$          /* initialize the associated degree sequence */
4      repeat
5          $k = \text{length}(\mathcal{C}')$   /* recompute the length of the reduced route */
6          for $i \in [2, k-1]$ do
7              if $c_i \equiv 0\ (mod\, d_i)$ then
                   /* apply Proposition 7 to contract $(c_{i-1}, c_i, c_{i+1})$ into
                      $(c_{i-1} + c_{i+1})$ in $\mathcal{C}'$                        */
8                  $\mathcal{C}' \longleftarrow (c_1, c_2, \ldots, c_{i-2}, c_{i-1} + c_{i+1}, c_{i+2}, \ldots, c_k)$
                   /* remove the degrees of the $i^{th}$ and $(i+1)^{th}$ nodes from
                      $\mathcal{D}'$                                                 */
9                  $\mathcal{D}' \longleftarrow (d_1, d_2, \ldots, d_{i-2}, d_{i-1}, d_{i+2}, \ldots, d_k)$
10                 break
11             end
12         end
13     until no index $i \in [2, k-1]$ such that $c_i \equiv 0\ (mod\, d_i)$ can be found
14 end
```

Algorithm 1: Route Minimization

To deal with cycles, it is sufficient to mark enough arcs with unique identifiers so that all cycles of the graph have at least one marked arc. In the example of Figure 6, imagine that the arc $b \to c$ is marked with γ. The robot can then annotate the sequence $c = (1, 2, 1, 2, 1)$ with the marker γ to indicate the traversing of a distinguished arc. The annotated version of the sequence c is $(1, \gamma, 2, 1, 2, \gamma, 1)$. The annotated sequence $(\gamma, 2, 1, 2, \gamma)$ can be reduced to (γ). Therefore the sequence $(1, \gamma, 2, 1, 2, \gamma, 1)$ can be replaced with the sequence $(1, \gamma, 1)$ In practice, either the path between Place b and Place c corresponding to the arc $b \to c$ is uniquely identifiable thanks to natural features, or we have to install a uniquely identifiable landmark on this path.

Algorithm 2 extends Algorithm 1 to graphs with cycles. We need first to create a set $\mathcal{M}$ of uniquely marked arcs such that $\mathcal{M}$ intersects every cycle of the graph. To reduce an annotated route $\mathcal{C}$ on the graph, we first replace any subsequence of the form $(\gamma, \mathcal{P}, \gamma)$ of $\mathcal{C}$ with (γ), where $\mathcal{P}$ is a route and $\gamma \in \mathcal{M}$. Finally, when no further substitutions are applicable, we run Algorithm 1.

Theorem 1. *Algorithm 2 reduces any route $\mathcal{C}$ (possibly long random walks) to an equivalent route $\mathcal{C}'$ whose length is bounded by $q + m \times (q + 1)$, where m is the number of edges of the graph, and q is the number of marked arcs.*

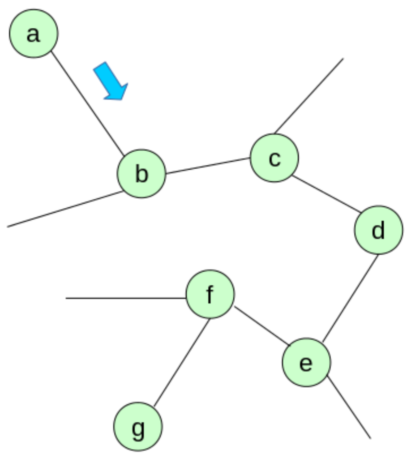

Fig. 7. Starting on the blue arrow, the route $c = (1, 2, 1, 2, 1)$ brings the robot to the arc $f \to g$. From a topological point of view, the robot cannot distinguish the journey on this graph from the journey experienced on the graph of Figure 6.

Proof. Consider $\mathcal{C}'$ the reduced route returned by Algorithm 2. The route $\mathcal{C}'$ is the concatenation of runs of non-marked arcs and marked arcs. Each marked arc appears at most once in $\mathcal{C}'$. This accounts for the the term q in the upper bound formula. Recall that the graph induced by the non-marked arcs is a tree, as it contains no cycles by construction of $\mathcal{M}$. Therefore each run of non-marked arcs is of length at most m as Algorithm 1 has eliminated all U-turns in $\mathcal{C}'$. We have at most $q+1$ runs of non-marked arcs in $\mathcal{C}'$. This accounts for second term $m \times (q + 1)$ in the upper bound formula.

5 Experiments

We validated the algorithms of Section 4 in simulation as well as with real robots. We programmed Lego Mindstorms robots to navigate in networks drawn on laminated posters (see Figure 8). The robot performed a random walk with route logging. Upon an external trigger (detecting a grey patch on the floor, or sound clap), the robot had to return to some specific location (either the starting position or the place where another grey patch was found).

6 Conclusion

In this paper, we have introduced a path algebra that is well suited for navigation in environments whose topology is a graph. We showed that when the graph contains no cycles, our route minimization algorithm returns the optimal route.

```
  input :
  $\mathcal{M}$ : a set of marked arcs such that $\mathcal{M}$ intersects every cycle of the graph
  $\mathcal{C}$ : an annotated route with marked arcs from $\mathcal{M}$
  output:
  a reduced route $\mathcal{C}'$ equivalent to $\mathcal{C}$
1 begin
2     $\mathcal{C}' = \mathcal{C}$                    /* initialize the reduced route */
3     repeat
4         scan $\mathcal{C}'$ for multiple occurences of any marked arc $\gamma$
5         let $(\gamma, \mathcal{P}, \gamma)$ be the longest subsequence containing the same marked arc
          $\gamma$
6         replace $(\gamma, \mathcal{P}, \gamma)$ in $\mathcal{C}'$ with $(\gamma)$
7     until no $\gamma \in \mathcal{M}$ appears more than once in $\mathcal{C}'$
8     run Algorithm 1 on $\mathcal{C}'$
9 end
```

Algorithm 2: Route Reduction on Graphs with Cycles

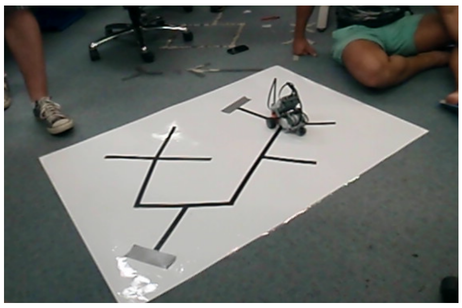

Fig. 8. Testing Algorithm 1 on a Lego-Mindstorms robot. The robot wanders on the network while logging its route. Upon detecting the second grey patch or hearing a clap, it returns to the first grey patch using the reduced route.

The path algebra is not restricted to planar environments. For example, using a lift in a building can be abstracted as traversing a node with each destination floor corresponding to an edge of the node.

If cycles are present in the graph, distinguishing some places becomes necessary for localisation. We showed that in order to navigate, it is sufficient to be able to recognize a set of places on the ground that corresponds to a set of arcs that intersects every cycle of the topological graph representing the environment. In future work, we plan to test our approach in a building environment with a mobile robot capable of detecting doors and reading signs on doors.

References

1. Marcus Augustine, Frank Ortmeier, Elmar Mair, Darius Burschka, Annett Stelzer, and Michael Suppa. Landmark-tree map: a biologically inspired topological map

for long-distance robot navigation. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 128–135. IEEE, 2012.

2. Mark Cummins and Paul Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.
3. Feras Dayoub, Timothy Morris, Ben Upcroft, and Peter Corke. Vision-only autonomous navigation using topometric maps. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1923–1929. IEEE, 2013.
4. David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. A review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.
5. David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.
6. Santiago Garrido, Luis Moreno, Dolores Blanco, and Piotr Jurewicz. Path planning for mobile robot navigation using voronoi diagram and fast marching. *Int. J. Robot. Autom*, 2(1):42–64, 2011.
7. Arren J Glover, William P Maddern, Michael J Milford, and Gordon Fraser Wyeth. Fab-map+ ratslam: appearance-based slam for multiple times of day. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3507–3512. IEEE, 2010.
8. J. Hartmann, J.H. Klussendorff, and E. Maehle. A unified visual graph-based approach to navigation for wheeled mobile robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1915–1922, Nov 2013.
9. Jiang Jehn-Ruey, Lai Yung-Liang, and Deng Fu-Cheng. Mobile Robot Coordination and navigation with directional antennas in positionless Wireless Sensor Networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 7(4):272–280, jan 2011.
10. K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3041–3047, May 2011.
11. W. Maddern, M. Milford, and G. Wyeth. Towards persistent indoor appearance-based localization, mapping and navigation using cat-graph. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4224–4230, Oct 2012.
12. Michael Milford, Adam Jacobson, Zetao Chen, and Gordon Wyeth. Ratslam: using models of rodent hippocampus for robot navigation and beyond. 2013.
13. Anthony Stentz and Is Carnegle Mellon. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10:89–100, 1993.
14. Stephen Tully, George Kantor, and Howie Choset. A unified bayesian framework for global localization and slam in hybrid metric/topological maps. *The International Journal of Robotics Research*, page 0278364911433617, 2012.
15. Michael Worboys. Modeling indoor space. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 1–6. ACM, 2011.