An Automatic Approach for the Generation of the Roadmap for Multi-AGV Systems in an Industrial Environment

Valerio Digani, Lorenzo Sabattini, Cristian Secchi and Cesare Fantuzzi

Abstract—This paper deals with the automatic generation of a roadmap. We propose an approach to build a roadmap for Automated Guided Vehicles (AGVs) used for logistics operations in industrial environments. The algorithm computes a roadmap in such a way that the coverage, the connectivity and the redundancy of the paths are maximized. In this way the flexibility and the efficiency of the AGV system can be increased. The proposed approach is validated by means of comparison with different roadmaps manually built in real plants.

I. INTRODUCTION

This paper deals with the automatic generation of a roadmap for the navigation of Automated Guided Vehicles (AGVs) in an automatic warehouse. As illustrated in [1], [2], in the vast majority of modern automatic warehouses, AGVs are constrained to move along a set of (virtual) paths and this set is usually called roadmap. Up to sixty AGVs can travel in an automatic warehouse and the way the roadmap is designed tremendously affects the way traffic can be managed and, consequently, the efficiency of the overall system. In the current industrial practice, given the layout of a plant, the roadmap is manually designed. This process is very time consuming and the achieved set of paths can be quite far from the best one. Thus, we propose an automatic procedure for designing a roadmap which dramatically decreases the installation time and the cost of an AGV system.

A roadmap can be represented by a topological graph spanning the free space [3]. Most roadmap based algorithms have been designed for motion planning for a single robot in a static environment and are generated based on random sampling techniques [4]. Several algorithms, known collectively as probabilistic roadmap methods (PRMs), have been shown to perform well in a number of practical situations [5], [6]. These methods run quickly and are easy to implement, but they can perform poorly in some situation [7]. Furthermore there are several extensions based on the PRM algorithm. Visibility-PRMs [4], [8] try to build small roadmap avoiding redundant elements, instead Reachability Roadmap Methods (RRM) emphasize the coverage and the maximal connectivity of the free space \( C_{free} \) [9] by creating small, resolution complete roadmaps [10], [11]. Nonetheless many techniques generate low quality paths. The approach described in [12] aims at providing high-clearance paths by retracting the roadmap to the medial axis [13].

Some algorithms have been proposed to extend the roadmap based algorithms to dynamic environments [14] and multiple agents [15]. All these approaches generate roadmaps that cover the environment without guaranteeing redundancy, that will be formally defined later on and that is a crucial characteristic for obtaining an efficient traffic flow. Moreover these methods usually require large amount of memory that is not always available in the control unit of an AGV system for automatic warehouses.

The contribution of this paper consists of a new automatic algorithm that, given the layout of the plant, provides a roadmap such that the coverage, the connectivity and the redundancy are maximized. The generated roadmap is suitable to be used with the traffic coordination strategy reported in [16] where the coordination and the roadmap generation problems are treated as a whole. An evidence of this claim is that the performance of the coordination algorithm is strongly related to the structure of the roadmap where the coordination takes place.

The rest of the paper is organized as follows. Section II provides some background notions on the medial axis transform and on algebraic graph theory, that will be used in the paper. Section III describes the problem and provides some definitions. Section IV shows in detail the algorithm for the generation of the roadmap. Finally Section V explains the experiments and simulations, and Section VI deals with the conclusions and the future work.

II. PRELIMINARIES

A. Medial Axis Transform (MAT)

The medial axis of an object is the set of all points having more than one closest point to the object’s boundary. In the 2D case, the following facts hold [13]. Given a planar region \( B \) bounded by a curve \( L \), the medial axis of \( B \) is the set of the centers of the circles that are contained in \( B \) and that are tangent to \( L \) in at least two points. The medial axis of a simple polygon is a tree whose leaves are the vertices of the polygon, and whose edges are either straight segments or arcs of parabolas. Let \( Q \) be the set of segments of line that compose the medial axis.

B. Algebraic Graph Theory

In this section we summarize some of the main notions on graph theory used in the paper. For more exhaustive information, the reader can see for instance [17]. Given a directed graph (or digraph) \( G \) with \( N \) nodes, let \( V \) represent...
the set of vertices (or nodes) of the graph, and let \( E \subseteq V \times V \) represent the set of edges.

A directed graph is strongly connected if it contains a directed path that connects every pair of nodes of the graph. A node of a graph is defined balanced if its out-degree is equal to its in-degree, that is the sum of the incoming (possibly weighted) edges equals the sum of the outgoing (possibly weighted) edges. A balanced graph is defined as a graph whose nodes are all balanced.

A communication graph can be described by means of the adjacency matrix \( A \in \mathbb{R}^{N \times N} \). Let \( N_i \) be the neighborhood of the \( i \)-th node. Each element \( a_{ij} \) is defined as the weight of the edge between the \( i \)-th and the \( j \)-th node, and is a positive degree if \( j \in N_i \), zero otherwise.

The degree matrix of the graph is defined as \( D = \text{diag} \{ d_i \} \), where \( d_i \) is the out-degree of the \( i \)-th node of the graph, that is \( d_i = \sum_{j=1}^{N} a_{ij} \). The (weighted) Laplacian matrix of the graph is defined as \( L = D - A \).

Let \( M \) be the number of edges and let \( w = [w_1, \ldots, w_M] \in \mathbb{R}^M \) be the weights associated to each edge, then the weight matrix \( W \in \mathbb{R}^{M \times M} \) is defined as \( W = \text{diag}(w) \). The Laplacian matrix \( L \) of the directed graph associated with the weight matrix \( W \) can be defined as follows:

\[
L_w = T \cdot W \cdot T^T
\]

Where \( T \in \mathbb{R}^{N \times M} \) is the Incidence matrix. The unweighted Laplacian matrix \( L_w \) is a special case of Laplacian matrix, where all non-zero entries of the adjacency matrix are equal to one and \( L_w = T \cdot T^T \).

The Laplacian matrix exhibits some remarkable properties [17], [18]:

1. Let \( T \) be the column vector of all ones. Since the row sum of the Laplacian matrix is equal to zero, then \( L \cdot T^T = 0 \).
2. Let \( \lambda_i^G \), \( i = 1, \ldots, N \) be the eigenvalues of the Laplacian matrix of the graph \( G \). The eigenvalues can be ordered such that \( 0 = |\lambda_1^G| \leq |\lambda_2^G| \leq \ldots \leq |\lambda_N^G| \).
3. For a strongly connected digraph, \( \Re \{ \lambda_i^G \} \geq 0 \) \( \forall i = 1, \ldots, N \) where \( \Re \{ \cdot \} \) defines the real part of a complex number.
4. In a balanced digraph, the column sum of the Laplacian matrix is equal to zero as well. Then, \( T^T L = 0 \).
5. If a graph is strongly connected, then \( \Re \{ \lambda_2^G \} > 0 \). In case the graph is balanced, then \( \Re \{ \lambda_2^G \} > 0 \) if and only if the graph is strongly connected: then, as in the undirected graph case, we refer to \( \lambda_2^G \) as the algebraic connectivity of the graph.
6. The value of \( \Re \{ \lambda_i^G \} \) increases as the number of edges and with increasing the weight of the edges.

It is worth noting that the higher the value of \( \Re \{ \lambda_2^G \} \) the higher the efficiency of the communication graph in solving distributed computation problems [18].

III. Problem Statement

The approach that we propose aims at filling an industrial warehouse with feasible paths.

Given the layout of the plant, we discretize the configuration space \( C \) by building a grid map and we indicated with \( C_{\text{free}} \) the portion of \( C \) that is not occupied by obstacles. A euclidean Distance Transformation (DT) [19] is applied on the grid map in order to discretize the distance from the obstacles.

In an industrial warehouse two main identities can be identified: corridors and intersections. A corridor is a part of \( C_{\text{free}} \) that can be represented as a straight line and that is bounded by obstacles. Given a polygonal environment and its medial axis \( Q \), a corridor is formally expressed as follows.

**Definition 1 Corridor** A corridor is the set of points in \( C_{\text{free}} \) bounded by at least two obstacles and such that \( Q \) is a segment of line parallel to the boundary of the obstacles.

An intersection is a bounded portion of \( C_{\text{free}} \) in which more corridors flow. Formally:

**Definition 2 Intersection** An intersection is the set of points in \( C_{\text{free}} \) bounded by corridors and whose medial axis \( Q \) is not parallel to the boundary of any of the obstacles.

A path in the real space consists of a path in the configuration space \( C \). We introduce the following definition of feasible path.

**Definition 3 Feasible Path** A path \( P \in C \) between two configurations \( i,j \in C \) is feasible, when \( P \in C_{\text{free}} \).

A roadmap can be represented as a graph \( G = (V,E) \), where the vertices \( V \) represent the cells \( C_{\text{free}} \), and the edges \( E \) represent the roads that connect the cells. We can introduce the following definitions:

**Definition 4 Coverage** [12] Given any two configurations \( c_1,c_2 \in C_{\text{free}} \), \( G \) covers \( C_{\text{free}} \) if a feasible path between \( c_1 \) and \( c_2 \) exists in \( G \).

**Definition 5 Strong Connectivity** [17] \( G \) is strongly connected when for all nodes \( v_i,v_j \in V \), there exists a path in \( G \) between \( v_i,v_j \).

**Definition 6 Redundancy** For each the ordered pairs of nodes \( (v_i,v_j) \in V \), the redundancy of \( (v_i,v_j) \) is the number of the different paths in \( G \) that connect \( v_i \) to \( v_j \).

When designing the roadmap, increasing the redundancy between each pair of nodes implies increasing the possibility of multiple choices of the paths. Thus, redundancy is an important index of quality to evaluate the roadmap for automatic warehouses. A roadmap with high redundancy globally entails a smaller number of stops of the AGVs due to traffic reasons, and a subsequent benefit for the efficiency of the system.

Our method provides a roadmap which is feasible, redundant, that covers all the free space and that is strongly connected.

IV. The Algorithm

In this section, we describe the algorithm for the generation of the roadmap into detail. The layout of the industrial
plant is provided as an input and it contains geometric information such as the position of the static obstacles in the environment.

The algorithm is structured as follows: first the free space has to be covered, then the number of roads is maximized to fill this space. A direction is subsequently assigned to each road in such a way that the algebraic connectivity of the associated graph is maximized. Finally the roadmap is smoothed considering the geometry of the AGV. Each part of the algorithm is described in the following subsections.

A. Find Corridors and Intersections

First of all, the free space has to be detected. Without loss of generality, assume that the obstacles are 2D polygons (e.g. bounding box). The free space can be captured by means of the medial axis transform (MAT).

In this way, the free space is modeled as a topological graph where each node represents an intersection or a corridor. This graph covers all $C_{\text{free}}$ and it is characterized also by high clearance, that is the graph lies on the middle axis of the free space. The MAT process generates a set of segments of line. By analyzing the MAT it is possible to identify corridors and intersections, exploiting Definition 1 and Definition 2. Fig. 1a shows the intersections and corridors found in a portion of a real warehouse.

B. Fill the Corridors

The redundancy requirement is satisfied maximizing the number of roads. The roads have to be collision free as much as possible (collisions can take place in proximity of the intersections). Let $\delta$ be the maximum width of an AGV. Each road has to guarantee a minimal distance $\delta$ form the other roads in order to avoid collisions.

In general, the dimension of the corridor can be deduced by the grid map introduced in Section III. This information is used to determine the maximum number of roads that guarantee at least the minimal distance $\delta$ from each other. Thus a set of roads is generated in each corridor, Fig. 1b shows an example.

The filling procedure is described in details in Algorithm 1.

Algorithm 1: Fill the corridors

| Data: Roadmap $\mathcal{G}$, corridors, AGV’s dimension |
| Result: Roadmap with filled corridors |

1 for all the corridor of $C_{\text{free}}$ do
2     calculate distance transform $DT$;
3     compute maximum number of roads based on AGV’s dimension;
4     for $i=1$ to number of roads do
5         generate road $l$ ;
6         put $l$ in $\mathcal{G}$ ;
7     end
8 end

C. Build the Intersections

In this section we show how to connect the roads of different corridors in an intersection. At this stage, roads are straight lines. An intersection can be modeled as a polygonal area bounded by obstacles and corridors, as shown in Fig. 2a. We simply propose to generate a grid that extends all the roads, see Fig. 2b. In general, the extended roads can overlap each other and the grid is pruned considering the following constraints:

- the distance from obstacles has to be more than $\delta$
- the distance form other roads has to be more than $\delta$

In case that one of the constraints is not respected, the road is merged with another one or deleted, see Fig. 2c.

The result of this process applied on a portion of a real environment is shown in Fig. 1c.

The details of the algorithm are described in Algorithm 2.

D. Assign Directions

This step aims at assigning the direction to each road maximizing the connectivity of the roadmap.

The environment is topologically described as a weighted directed graph $\mathcal{T}(V, E)$ with weight function $\Omega : \mathcal{E} \rightarrow \mathbb{R}$. The vertices $V$ model the intersections and the edges $E$ model the corridors among the intersections. Each corridor can be represented by one or two edges, based on the number of roads obtained after applying Algorithm 1.

The weight of each edge in a corridor is a function of the number of roads and of the length of the corridor itself.

In particular, the traffic flow of the AGVs among the intersections is modeled as the communication among nodes interconnected with the graph $\mathcal{T}$. The communication efficiency on a graph increases as the edge weights increase. Therefore, for each corridor, we define the corresponding edge weight as a quantity that is:

- directly proportional to the number of roads (more roads allow more AGVs to simultaneously travel on the corridor)
- inversely proportional to the length of the corridor (long corridors require more time to be traversed)

Formally the weight of the edge between the $i$-th and $j$-th

![Fig. 2: The intersections are created merging or deleting roads form different corridors. The intersection region is outlined with a green polygon and the corridors with a red one. The roads in the corridors are shown as black lines. This process is outlined in Algorithm 2.](Image 313x678 to 387x742)

![Fig. 2: The intersections are created merging or deleting roads form different corridors. The intersection region is outlined with a green polygon and the corridors with a red one. The roads in the corridors are shown as black lines. This process is outlined in Algorithm 2.](Image 397x678 to 471x742)
Algorithm 2: Build the Intersections

Data: Roadmap \( G \), intersection
Result: Roadmap

\begin{verbatim}
for all the intersection do
    find the coming corridors \( C_{in} \);
    initialize grid \( \text{intersect}_\text{grid} \);
    for all the \( C_{in} \) do
        for all the roads in \( C_{in} \) do
            extend the road until the next corridor;
            put new extended line in \( \text{intersect}_\text{grid} \);
        end
    end
    for all the \( i = 1 \) to lines in \( \text{intersect}_\text{grid} \) do
        for all the \( j > i \) to lines in \( \text{intersect}_\text{grid} \) do
            if lines \( j \) is directed to obstacles then
                delete lines \( i \);
                break;
            end
            if lines \( j \) is directed to obstacles then
                delete lines \( j \);
                continue;
            end
            if distance \( i,j \) < distance \( \text{min} \) then
                merge lines \( i \) and lines \( j \);
            end
        end
    end
end
\end{verbatim}

The corridors can be divided into two classes: bi-directional corridors and mono-directional corridors.

Corridors with more than one road are defined as bi-directional corridors and are represented by two edges. A corridor with an even number of roads from \( i \) to \( j \) is characterized by \( n_{\text{roads},i \rightarrow j} = n_{\text{roads},j \rightarrow i} \) which implies \( w_{i,j} = w_{j,i} \).

Vice-versa a corridor with an odd number of roads from \( i \) to \( j \) is characterized by \( n_{\text{roads},i \rightarrow j} = n_{\text{roads},j \rightarrow i} \pm 1 \) which implies \( w_{i,j} \neq w_{j,i} \). The exact values of \( n_{\text{roads},i \rightarrow j} \) and \( n_{\text{roads},j \rightarrow i} \) are computed solving the Direction Assignment Problem.

Corridors with only one road are defined as mono-directional corridors and are represented by one edge, whose direction is computed solving the Direction Assignment Problem. If the edge is directed form \( i \) to \( j \), its weight is computed as \( w_{i,j} = 1/\Omega_{i,j} \).

The Direction Assignment Problem is an optimization problem that will be solved in the following steps:

1) Randomized direction assignment
2) Strong connectivity verification
3) Algebraic connectivity evaluation

1) Randomized direction assignment: Define \( n_k \) as the number of roads obtained in the \( k \)-th corridor after applying Algorithm 1. Assume, without loss of generality, that the \( k \)-th corridor connects the \( i \)-th and \( j \)-th intersections. The directions of the roads are assigned consistently with the previously introduced definitions of bi-directional and mono-directional corridors. This can be obtained as follows:

- \[ \left\lfloor \frac{n_k}{2} \right\rfloor \text{ roads with direction } i \rightarrow j \]
- \[ \left\lfloor \frac{n_k}{2} \right\rfloor \text{ roads with direction } j \rightarrow i \]
- \[ n_k - 2 \cdot \left\lfloor \frac{n_k}{2} \right\rfloor \in \{0,1\} \text{ roads with random direction} \]

Summarizing, only the road of each mono-directional corridor and one road of each odd bi-directional corridor have to be defined. In this way the problem of assigning the directions is drastically reduced.

2) Strong connectivity verification: In this step, we introduce a procedure to verify that a particular solution of the randomized direction assignment described in Section IV-D.1 generates a strongly connected roadmap.
As described in Section II-B, the algebraic connectivity is an indicator of strong connectivity for balanced graphs. It is worth noting that, in general, \( \mathcal{T} \) is not balanced. However, a balanced version of the graph \( \mathcal{T} \) can always be obtained by means of opportunely computed weights of the edges \( p = [p_1, \cdots, p_M] \) where \( M \) is the number of the edges [20]. The weight vector that balances a directed graph is the solution of \( \mathcal{I} \cdot p = 0 \), where \( \mathcal{I} \) is Incidence matrix of \( \mathcal{T} \) and \( 0 \) is the null vector. The graph \( \mathcal{T} \) is the balanced version of the graph \( \mathcal{T} \), and according to Eq. (1), its Laplacian matrix is defined as follows:

\[
\mathcal{L}_p = \mathcal{I} \cdot \text{diag}(p) \cdot \mathcal{I}^T
\]  

(3)

In other words, \( \mathcal{T} \) is a balanced graph with the edges directed as those of \( \mathcal{T} \). Therefore, \( \mathcal{T} \) is strongly connected if and only if \( \mathcal{T} \) is strongly connected as well.

The algebraic connectivity can then be used to verify the strong connectivity of \( \mathcal{T} \). In particular if \( \Re \{ \lambda_2 \} > 0 \) then the previously assigned directions of the roads generate a strongly connected roadmap, that represents an admissible solution.

3) Algebraic connectivity evaluation: In general, different solutions for the randomized direction assignment may exist that lead to a strongly connected graph. In order to compare these solutions, we select the algebraic connectivity of \( \mathcal{T} \) as an index to be maximized. As described in Section II-B, increasing the algebraic connectivity improves the efficiency of the graph: in this application, the algebraic connectivity is then chosen as a heuristic to maximize the traffic flow.

The maximization of the algebraic connectivity consists of maximizing the real part of the second smallest eigenvalue of the Laplacian matrix associated with the strongly connected graph \( \mathcal{T} \).

The overall Direction Assignment Problem can now be modeled as a binary non-linear optimization problem. The optimization parameters assume only two scalar values (0,1) and they represent the directions defined in the randomized direction assignment described in Section IV-D.1. Formally they are represented as \( d_1 \cdots d_{\pi} \) where \( \pi \) is the number of roads to which a direction has to be assigned. The optimization variables can then be collected in the vector \( x \), defined as follows: \( x = [d_1 \cdots d_{\pi}] \in \{0,1\}^\pi \)

The optimization problem is formalized as:

\[
\begin{align*}
& \text{maximize} & & \Re \{ \lambda_2^T (x) \} \\
& \text{subject to} & & x \in \{0,1\}^\pi \\
& & & \Re \{ \lambda_2^T (x) \} > 0
\end{align*}
\]  

(4a) \hspace{1cm} (4b) \hspace{1cm} (4c)

The constraint defined by the Eq. (4c) represents the strong connectivity verification described in Section IV-D.2. As noted this optimization problem is generally NP-hard. However there are several algorithms that can generate a good approximate solution to this class of problems for an effort that increases only slowly as a function of \( \pi \). In our approach a local search method is used. In particular, a Tabu Search approach is combined with a Monte Carlo method [21].

E. Smooth the Roadmap

The roadmap has to be smoothed in order to generate feasible trajectories with respect to the AGV’s kinematics and geometry. The previous steps of the algorithm build a roadmap composed by straight lines. Naturally these lines intersect each other forming sharp corners. Our method consists of substituting the sharp corners with Bezier curves [22] which are built with respect to the minimal radius of curvature of the AGVs. In this way the roadmap is composed by a sequence of segments of line and Bezier curves. In particular, the curves are used to connect the roads which are intersecting. The result is visible in Fig. 1d

V. EXPERIMENTAL VALIDATION

The proposed methodology was evaluated on real industrial plants comparing the obtained results with currently utilized roadmaps. It is worth noting that current industrial state of the art techniques are based on manually built roadmaps. Hereafter we refer to the roadmaps generated by our method as automatic roadmaps, and to the roadmaps currently used in industrial warehouses as manual roadmaps. The comparison is performed by analyzing the connectivity of the roadmaps and the redundancy. The algebraic connectivity is then used as a heuristic for evaluating the quality of a roadmap and the maximum flow [17] is used as index of redundancy. The comparison was performed on plants with different dimensions. Due to confidentiality reasons, the manual roadmaps cannot be shown or discussed in detail. In general we can divide the plants into three classes based on the dimension. Namely:

- small: less than 10 AGVs
- medium: from 10 to 30 AGVs
- big: more than 30 AGVs

Table I shows the results obtained on three typical plants representing the above mentioned classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Dimension (AGVs)</th>
<th>Manual Roadmap ( \Re { \lambda_2^T } ) max flow</th>
<th>Automatic Roadmap ( \Re { \lambda_2^T } ) max flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>5</td>
<td>0.0031</td>
<td>0.0043</td>
</tr>
<tr>
<td>medium</td>
<td>25</td>
<td>0.00002</td>
<td>0.0027</td>
</tr>
<tr>
<td>big</td>
<td>50</td>
<td>0.00075</td>
<td>0.0019</td>
</tr>
</tbody>
</table>

TABLE I: Values of connectivity and maximum flow

It is worth noting that our roadmaps are comparable with the manual ones in terms of connectivity and redundancy of the graph. Observing the algebraic connectivity term (real part of the second smallest eigenvalue), it is possible to state that the connectivity of the automatic roadmaps is higher than the connectivity achieved by the manual ones. The automatic roadmaps are also more redundant (more paths) than the manual ones. Figs. 3a, 3b, 3c show the automatic roadmaps generated with our approach.

VI. CONCLUSION

The paper describes an automatic roadmap generation process that autonomously defines a set of near-optimal paths in order to cover and connect all the free space.
An algorithm is proposed to build a roadmap in such a way that the coverage, the redundancy and the connectivity are maximized. A roadmap has to cover all the free space in order to reach all the positions of interest, that is the operation points. The coverage is guaranteed by means of the medial axis transform (MAT) of the free space. The redundancy is an important index of quality to evaluate the roadmap of industrial environments. A roadmap with high redundancy entails a decrease of the down time of the overall system due to traffic reasons. The redundancy is achieved by maximizing the number of roads and paths. The connectivity of the roadmap affects the efficiency of the associated graph. The algebraic connectivity is then chosen as a heuristic to maximize the traffic flow. Naturally the roadmaps have to be strongly connected. The simulations have shown that it is possible to generate an automatic roadmap with higher connectivity than a manual roadmap built on the same real plant of an industrial warehouse. Current work aims at formally analyzing the complexity of the proposed algorithm in an analytic manner and to compare our approach to others algorithm to build a roadmap. Moreover, future works will aim at introducing several real factors to modify the weights of the graph, such as the real logistic flows of material in the warehouse and the possibility of preferred paths.

REFERENCES


