# Motion planning and control of mobile robot using Linear Matrix Inequalities (LMIs)

**Conference Paper** · October 2007

DOI: 10.1109/IROS.2007.4399641 · Source: DBLP



**2 authors:**

Ellips Masehian
California State Polytechnic University, Pomona
**80** PUBLICATIONS **1,867** CITATIONS

SEE PROFILE

Golnaz Habibi
MIT
**14** PUBLICATIONS **389** CITATIONS

SEE PROFILE

Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007



# Motion Planning and Control of Mobile Robot Using Linear Matrix Inequalities (LMIs)

Ellips Masehian and Golnaz Habibi

*Abstract*— **A new motion planning algorithm is proposed for point and disc robots. In this approach, the problem is first formulated as a Binary Integer Programming with variables taken from Delaunay Triangulation of the 2D or *n*-D Free Configuration Space, and then transformed into LMIs and solved to obtain an optimal channel made of connected triangles. The channel is then used to build safe and short paths within from Start to Goal. It is also possible to weight certain passageways of the space so that the robot can avoid costly routes, which is especially useful for traffic control applications. Finally, a tracking control strategy along trajectory based on preplanned path is applied for a synchronous drive robot. The algorithm is simple, complete, and does not suffer from local minima. It is also extendable to 3 and higher C-spaces.**

## I. Introduction

Most of the solution methods for the robot motion planning problem are variations of a few general approaches: Roadmap, Cell Decomposition, Potential Fields, mathematical programming, and heuristic methods, which are broadly surveyed in [1], [2], and [3].

In this paper a new solution approach is proposed for the classical path planning problem for a mobile robot in offline mode. The presented model is a novel combination of the Cell Decomposition and linear algebra techniques. First, we will briefly review some path planning methods and Linear Matrix Inequalities (LMIs). Next, the model is presented in 3 main phases: workspace tessellation, optimal channel finding, and path finding. The last sections of the paper deal with the tracking control of three wheeled mobile robots along trajectories generated based on obtained paths. Also, experimental results and comparisons, as well as further discussions on the nature of the model are presented.

### A. A Brief Review of Motion Planning

The *Roadmap* approach involves retracting or reducing the robot's free C-space ($C_{free}$) onto a network of one-dimensional lines (i.e. a graph). Motion planning is then reduced to a graph-searching problem. A well-known Roadmap, the *Visibility Graph*, is the collection of lines that connect a feature of an object (usually vertices of polygons) to that of another. The *Voronoi Diagram* is defined as the set of points equidistant from two or more object features [3]. Voronoi Diagrams have found a wide application in motion planning problems, especially in low dimensions, such as [5], [6], and [7]. The *Delaunay Triangulation* DT($S$) is obtained with a line segment between any two points $p$, $q$ of $S$ for which a circle $C$ exists that passes through $p$ and $q$ and does not contain any other site of $S$ in its interior or boundary. Two points of $S$ are joined by a Delaunay edge if and only if their Voronoi regions are edge-adjacent. Therefore, the Delaunay Triangulation is the graph-theoretical dual of the Voronoi Diagram. Fig. 1 shows the relation between Voronoi Diagram and Delaunay Triangulation.

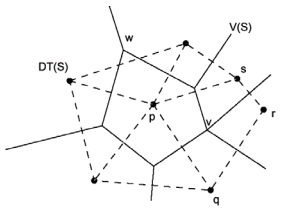


Fig.1. Voronoi Diagram (solid lines) and Delaunay Triangulation (dashed lines) of the set $S$ of points.

The *Cell Decomposition* approach divides the $C_{free}$ into a set of non-overlapping cells. The adjacency relationships between cells are represented by a *Connectivity Graph*. The graph is then searched for a collision free path. A solution is obtained by finding the cells that contain the initial and final configurations and then connecting them using a sequence of connected cells [2].

### B. Linear Matrix Inequalities

The Linear Matrix Inequalities (LMIs) are known as an efficient tool for solving linear and convex quadratic programming models. A wide variety of optimization problems in system and control theories, identification, and signal processing can be reduced to a few standard convex or quasi-convex optimization problems involving LMIs. By means of LMIs, many problems that do not have analytical solutions can be solved.

An LMI is in the form of the inequality $F(X) < 0$, where $F$ is an affine function of the matrix variable $X$. Mathematically, $X$ is the set of $n$ real matrices such that

$$F(X) = F_0 + \sum_{j=1}^{n} X_j F_j \ . \qquad (1)$$

The 'less than zero' condition in $F(X) < 0$ means negative

Manuscript received September 4, 2007.

E. Masehian is with the Industrial Engineering Department, Faculty of Engineering, Tarbiat Modares University, Tehran, Iran (corresponding author, e-mail: masehian@modares.ac.ir).

G. Habibi is M.S. student of Electrical Engineering at Faculty of Engineering, Tarbiat Modares University, Tehran, Iran.

definite; i.e., $u^T F(X) u < 0$ holds true for all nonzero real vectors $u$. A *System of LMIs* is a finite set of LMIs. Since the LMI defines a convex constraint on the variable $X$, and the set of solutions of the LMI $F(X) < 0$ is convex, the optimization problems involving maximizing or minimizing a objective function $f$: $s \rightarrow \Re$ with $s = \{X \mid F(X) < 0\}$ belongs to the class of convex optimization problems and is called *optimization with LMI constraints* [8]. Currently several LMI solvers are available as Matlab toolboxes, such as SDP3 and Sedumi [9].

The applications of LMI in Robotics have mainly been focused on the robust control problem, such as in [4], [10], and [11]. As far as we reviewed the literature, there is no known method for mobile robot path planning using LMIs.

## II. Structure of the Proposed Model

The presented path planner lies within the category of Cell Decomposition approach. However, unlike the usual procedure in Decomposition-based models, which builds a connectivity graph and then searches it to find an optimal channel of cells, this model implements mathematical programming for finding the optimal channel. The general phases of the algorithm are the following:

1) The $C_{free}$ is tessellated through the Delaunay Triangulation such that it is exactly decomposed into a set of triangles (Section III).

2) Each triangle is associated with a variable that is incorporated in a 0-1 Integer Programming model, with an appropriate objective function. This model is then converted to a system of LMIs and solved (Sections IV, V). The solution gives a set of connected triangles forming a channel.

3) The channel is further segmented into a number of convex fragments, and a safe path is calculated passing through the medians of the convex fragments (Section VI).

The next sections explain the above phases in detail.

## III. Triangulating the Workspace

The first step for path planning is to express the workspace (or C-space) as a mathematical formulation. For this purpose, the workspace is decomposed into arrangements of cells through Delaunay Triangulation. The reason for choosing Delaunay triangulation for tessellating the workspace is its many useful optimization properties discussed in [12].

The input information for the triangulation consists of the coordinates of all obstacles' vertices and the inner corners of the workspace's rectangular border. The Delaunay algorithm then builds a connected network of these points, forming a set of triangles. These triangles lie both inside the obstacles and the $C_{free}$. By applying a simple algorithm, all the triangles inside obstacles can be identified and omitted from the triangles set, leaving 'free' triangles that build the $C_{free}$.

Fig. 2 shows the result of the abovementioned operations.

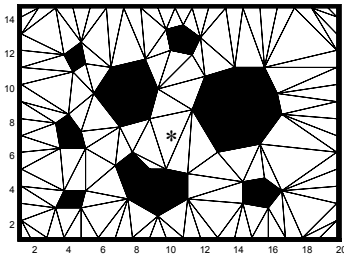


Fig. 2. Delaunay Triangulation of a workspace

## IV. Weighting Functions and Optimality Criteria

Since the objective function of the optimization problem with LMI constraints determines which channel, among all possibilities, is considered optimal (i.e. is supposed to contain the global shortest path), it is very crucial to establish sound and robust criteria for defining the objective function.

The objective function is the product of two matrices: the weighting and the variables matrix (discussed in Section V). We worked out 4 different optimality criteria as follows:

1) *Number of Triangle* ($W_N$): This kind of weighting merely depends on the number of triangles, regardless of their size. The weighting matrix is a row vector of 1s, and so the system selects a Start-to-Goal channel with minimal number of constituting triangles.

2) *Area of Triangles* ($W_A$): In this criterion, larger triangles have larger weights, and the objective function turns into the weighted sum of variables. The algorithm searches for a channel with the smallest area.

3) *Perimeter of Triangles* ($W_P$): According to this criterion, larger triangles have larger weights. The algorithm searches for a channel with the smallest total perimeter.

4) *The Median Length of Channel* ($W_M$): The most reliable and effective weighting function we found is the median length of channel.

Each triangle is represented by the line connecting the middles of its two free edges (Fig. 3). Free edges are totally located in $C_{free}$ and do not belong to any obstacle's border. These lines (medians) are representatives of their respective triangles and their lengths are considered as the weights of their triangles in the objective function.

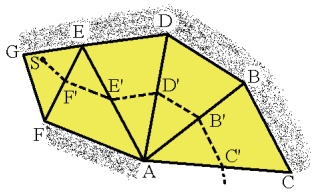


Fig. 3. A channel with its median line.

It may happen that all the 3 vertices of a free triangle belong to 3 different obstacles (like the starred one in Fig. 2. In this case, all edges are free, and the average length of all three median lines is considered as the triangle's weight in the objective function.

Also, in order to care about the distances of Start and Goal points to their neighboring triangles (e.g. SF′ in Fig. 3),

these distances are added to the medians of the neighboring triangles as their weighs (e.g. SF′+F′E′ for the triangle AEF).

It should be noted that a weighted combination of these weighting methods can also be applied as

$$W = \alpha_1 \times W_N + \alpha_2 \times W_A + \alpha_3 \times W_P + \alpha_4 \times W_M . \quad (2)$$

## V. Mathematical Formulation

In this phase a minimization problem with LMI constraints is developed. The objective function to be minimized is the weighted sum of the variables representing all triangles in $C_{free}$, which reflects a measure of optimality for the resulting channel. Following is a basic formulation of the problem, which is then transformed to an LMI model.

### A. Integer Programming Model

The path planning problem can be modeled as a 0-1 Binary Integer Programming (BIP) with variables defined as

$$t_i = \begin{cases} 1 & \text{if triangle } i \text{ is selected} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

In order to guarantee a continuous channel from the Start to Goal points, triangles building the trajectory channel must satisfy the following requirements:

1. The Start and Goal triangles ($t_S$ and $t_G$, respectively) must be selected.
2. If any triangle (other than $t_S$ and $t_G$) is selected, two of its adjacent triangles must also be selected (continuity condition).
3. Each of the Start and Goal triangles must have only one selected adjacent triangle (loop avoidance condition).
4. To avoid looping, only two adjacent triangles of triangles with three free edges must be selected.

With the above considerations, the BIP model for finding the optimal channel will be:

$$\text{Minimize } J = W^T.T$$

Subject to:

$$\begin{cases} t_1 + \sum_{j=1}^{n_1} t_1^j \geq 3t_1 \\ \vdots \\ t_{k-2} + \sum_{j=1}^{n_{k-2}} t_{k-2}^j \geq 3t_{k-2} \end{cases}, \begin{cases} t'_1 + \sum_{j=1}^{3} t'^j_1 \leq 3 \\ \vdots \\ t'_p + \sum_{j=1}^{3} t'^j_p \leq 3 \end{cases}, \begin{cases} t_S + \sum_{j=1}^{n_S} t_S^j = 2t_S \\ t_G + \sum_{j=1}^{n_G} t_G^j = 2t_G \end{cases} \quad (4)$$

$$t_S = 1,\ t_G = 1,\ t_i \in \{0, 1\},\ i \notin \{S, G\}.$$

In this model, $W$ is the column vector of weights (calculated from (2) with desired coefficients $\alpha_i$), and $T$ is the vector of variables $[t_i, \ldots, t_k]^T$. The $t_i^j$ is the $j^{\text{th}}$ adjacent triangle of $t_i$, and $j = 1, \ldots, n_i$, in which $n_i = 1, 2$, or 3.

In (4), $\{t'_k\}$ ($k = 1, \ldots, p$) is the set of $p$ triangles with three free adjacent triangles and $t'^j_k$ is the $j^{\text{th}}$ adjacent triangle of triangle $t'_k$.

There is one constraint for each triangle, and so the problem of path planning turns into an optimal planning with $k+p$ constraints ($k+p-2$ inequalities and 2 equalities), with an optimality criterion defined by the objective function $J$.

### B. LMI Formulation

The above minimization model can be converted into an optimization problem with LMI constraints. The objective function would be:

$$\text{Minimize } J = W^T.T$$

where $T_{k\times 1}$ is the column vector of variables $[t_i]$, and $W_{k\times 1}$ is the column vector of weights defined in Section V.*A*.

As for constraints, inequalities and equalities (4) can be reformulated to a system of three LMIs, as below:

$$\begin{aligned} H &= diag(AT - CT) \geq 0 \\ U &= diag(A'T - C'T) = 0 \\ V &= diag(A''T - C'') \leq 0 \end{aligned} \quad (5)$$

$A_{(k-2)\times k}$ is the *neighborhood* matrix. It is a binary matrix in which the elements $a_{ij}$ show the adjacency relationship of triangles $i$ and $j$ ($i, j \notin \{S, G\}$). $a_{ij} = 1$ if they are neighbors, and $a_{ij} = 0$ otherwise. Also, a triangle is considered the neighbor of itself.

$A'_{2\times k}$ is similar to the matrix $A$, except that it is just defined for the Start and Goal triangles. $A''_{p\times k}$ is similar to the matrix $A$, except that it is defined for triangles with 3 free neighbors.

$C_{(k-2)\times k}$ is a diagonal matrix with $c_{ii} = 3$, $C'_{2\times k}$ is a diagonal matrix with $c'_{ii} = 2$, and $C''_{p\times 1}$ is a diagonal matrix with $c''_i = 3$.

The above LMIs can be solved by different available solvers, after which variables taking a value of 1 represent the triangles which build the optimal channel. The channel lies entirely in $C_{free}$. Note that solving the integer programming model gives the same answers the LMI finds, but, the LMI model is easier to build and faster to solve.

## VI. Path Calculation

Upon finding the optimal channel, the next phase is to find an appropriate trajectory through it. A simple path could be through the centers of gravity of each triangle in the channel. However, it usually takes a zigzag form, which is not suitable for robot motion and acts poorly is presence of large triangles (the dashed line in Fig. 4).

The typical method for path finding in the classic Cell Decomposition approach is to connect the middles of free edges of cells in the optimal channel (the dotted line in Fig. 4). While this method works well in trapezoidal cells and the resulting path is much smoother and shorter than the path through the centers of gravity, it usually travels unduly lengthy distances such that it is considerably longer than the shortest path.

Another path can be suggested which passes through the middles of *convex fragments* edges of the optimal channel.

For determining the convex fragments of the optimal channel, beginning from the Start triangle, neighboring triangles are appended incrementally until reaching a tri-

angle that makes the entire set concave. By this, the largest possible convex fragment is formed. The next convex fragment starts from the next triangle, and this continues toward the goal triangle. Then, a path is calculated based on the midpoints of borders of convex fragments.

The obtained trajectory is much shorter than the other paths discussed above. Besides, the fact that a convex polygon contains any line connecting any two points in the polygon, guarantees that the path is definitely located in the channel and does not cross the obstacles. Fig. 4 shows the paths calculated based on three types of trajectories: 1) centers of gravity, 2) middles of free edges, and 3) middles of convex fragments. It is shown that the trajectory based on middle edges of the convex fragments is the shortest path among them.

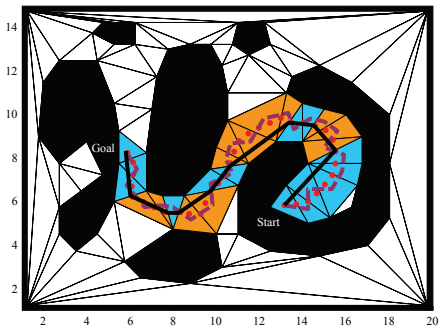


Fig. 4. Path planning with the optimal channel and three trajectories based on centers of gravity (dashed) (path length = 29.12), middles of edges (dotted) (path length =22.14), and middles of edges of 9 convex fragments, colored alternately (solid) (path length = 19.1).

For designing a shorter and smoother trajectory, sometimes the *concaveness* of a fragment can be ignored. The amount of concaveness can be measured by the *degree of concaveness*, which is related to the degree of the concave angle in the fragment. Thus, the greater the degree of the concave angle, the more the degree of concaveness. The range of the degree of concaveness is $180° \le \alpha < 360°$, in which $\alpha = 180°$ represents zero concaveness. For $\alpha \le 180°$ the polygon is convex, and the degree $\alpha \approx 360°$ yields the maximum concaveness.

When the $\alpha$ takes a value between 180 and 360 degrees, some amount of concaveness is tolerated, and the fragments are allowed to be somewhat concave. In return, this causes a shorter path passing through the middles of fragments. For our experiments we set the degree of tolerance about 190 degrees. Note that with large tolerances the path may be out of the channel and cross the obstacles. So the shortness of the path and the tolerance of concaveness must be balanced.

We also limited the cumulative errors of ignoring the concaveness of consecutive neighboring triangles by setting $\beta = 20°$ as the upper bound of the cumulative error.

## VII. Experimental Results and Comparison

In this section, we present an example of path planning with the new method. At first, all obstacles are defined by the user. Fig. 5 shows a sample workspace with 5 obstacles (one concave and four convex) having a total of 213 vertices. The Start and Goal points are then specified. Next, the workspace is triangulated by Delaunay Triangulation, which took 0.01 seconds. The next stage is to build the LMI model and solve it. Fig. 5 illustrates the obtained optimal channel. It took 0.84 seconds to find the channel with the Matlab toolbox GLPKMEX, on a 2.3 GHz PC. In this example, the objective function was the median length of the channel. Also, a spline function was implemented to smoothen the obtained path.

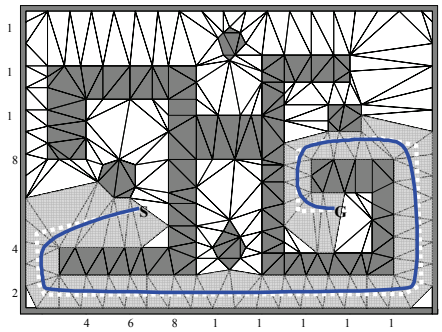


Fig. 5. The optimal channel (light gray), path connecting the middles of fragments (dotted), and the spline path (solid).

In order to test and compare the new planner, 30 problems with different numbers and shapes of obstacles were designed, with their total number of obstacles ($m$) ranging from 5 to 50, and each obstacle having 5 vertices. We solved 5 problems in each category by the proposed, as well as the Dijkstra search methods using Matlab. The graph used for the Dijkstra search was the Generalized Voronoi Diagram of the workspace. The obtained values for each problem size are averaged and summarized in Table I. It is shown that the new method is efficient in terms of both time and path length.

TABLE I.
AVERAGE CPU TIMES OF THE EXPERIMENTATION

| | $m$ | Triangulation | LMI solving | Path calculation | Total time (s) | Path length |
|---|---|---|---|---|---|---|
| LMI planner | 5 | 0.040 | 0.016 | 0.163 | 0.219 | 57.95 |
| | 10 | 0.033 | 0.047 | 0.233 | 0.313 | 58.85 |
| | 20 | 0.070 | 1.092 | 0.315 | 1.447 | 59.22 |
| | 30 | 0.098 | 10.469 | 0.367 | 10.934 | 60.11 |
| | 40 | 0.126 | 113.641 | 0.533 | 114.300 | 60.01 |
| | 50 | 0.155 | 350.670 | 0.534 | 351.359 | 60.36 |
| | $m$ | Voronoi construction | | Dijkstra search | Total time (s) | Path length |
| Dijkstra | 5 | 2.063 | | 0.219 | 2.282 | 66.85 |
| | 10 | 4.735 | | 1.734 | 6.469 | 68.79 |
| | 20 | 12.320 | | 77.253 | 89.573 | 69.30 |
| | 30 | 19.540 | | 189.338 | 208.878 | 69.67 |
| | 40 | 27.870 | | 358.457 | 386.327 | 71.58 |
| | 50 | 40.950 | | 578.468 | 619.418 | 72.14 |

## VIII. Extension to Higher Dimensions

An interesting feature of the new algorithm is the ease of its extension to high dimensional workspaces. Since the algorithm's variables are taken from the cells resulting from

Delaunay Triangulation, the cells can be triangles (in 2D), tetrahedrons (in 3D), or $n$-polytopes (in $n$-D). The constraints remain the same in higher dimensions; the only difference is in the maximum number of free neighbors for each $n$-polytope, which is $n$+1 for $n$-D space. The BIP formulation for cells other than Start and Goal cells will be

$$\begin{cases} t_1^{(n-1)} + \sum_{j=1}^{n+1} t_1^{(n-1)j} \le 3 \\ \vdots \\ t_{p_n}^{(n-1)} + \sum_{j=1}^{n+1} t_{p_n}^{(n-1)j} \le 3 \end{cases} \cdots \begin{cases} t'_1 + \sum_{j=1}^{3} t'^{j}_1 \le 3 \\ \vdots \\ t'_p + \sum_{j=1}^{3} t'^{j}_p \le 3 \end{cases}, \quad (6)$$

in which $\{t_p^{(d)}\}$ ($p = 1, \ldots, p_n$; $d = 2, \ldots, n$) is the set of $p$ $n$-polytopes with $d$+1 free adjacent $n$-polytopes, and $t_p^{(d)j}$ is the $j^{\text{th}}$ adjacent $n$-polytope of the $n$-polytope $t_p^{(d)}$. Fig. 6 illustrates a path planning using LMIs in 3D workspace.

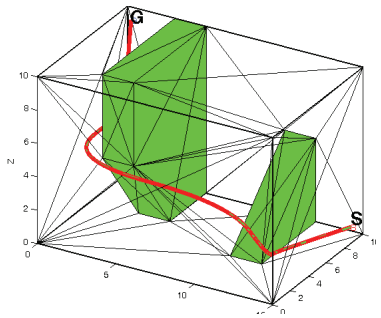


Fig. 6. Path planning using LMIs in 3D space.

## IX. Regulating the Path

In previous sections the mobile robot was assumed to be a point, and so could pass through any passageway. However, in real applications, the robot's size, kinematic and dynamic constraints should be incorporated into the model. In this section, we present a simple method to inhibit the planner from selecting narrow or undesirable locations for the robot's navigation. This is done by assigning weights (penalties) to the triangles.

Generally, triangles in $C_{free}$ may be categorized into three classes: Safe, Forbidden, and Unsafe triangles. Safe triangles are wide enough so that the mobile robot can maneuver safely through them. Forbidden triangles are triangles that are too narrow for the robot to pass through. Also, for planning paths with smoother spins, triangles with sharp corners are considered forbidden. Unsafe triangles indicate rough or slippery terrains, where passing through them is risky for the robot. Based on these definitions, the new weighting function is proposed as

$$W = \alpha_1 \times W_N + \alpha_2 \times W_A + \alpha_3 \times W_P + \alpha_4 \times W_M + \alpha_5 \times W_C. \quad (7)$$

In (7), $W_C$ (weight of constraints) includes both environmental constrains (road terrain), and robot's constraints (size and maximum rotation). These constraints can also be considered and weighted separately:

$$W_c = \alpha_6 \times W_R + \alpha_7 \times W_E \quad (8)$$

where $W_R$ and $W_E$ are the robotic and environmental constraints, respectively. $W_R = d(r) / H(t_i)$, in which $d(r)$ is the size (diameter) of the disc robot, and $H(t_i)$ the main height of triangle $t_i$ (length of the edges belonging to the border or obstacles). For a triangle with three free edges, the diameter of its inscribed circle is taken as $H(t_i)$. It is obvious that for safe triangles, $W_R < 1$. For triangles having $W_R > 1$, either the size of the robot is larger than the width of the channel, or the bend of the channel is too sharp for the robot (applicable for triangles with three free edges). These triangles are considered as forbidden, and the $W_R$ is set to a large positive value to stop the planner from selecting them.

$W_E$ is set by the user for defining rough, slippery or dangerous areas. It can also be used for integrating traffic or congestion zones data. The planner prefers selecting safe triangles (e.g. smooth, dry, or unoccupied roads) rather than unsafe triangles (e.g. rough, slippery, or congested terrains) for robot motion.

Fig. 7 shows how the initial path is regulated by new weighting function. In this special case, the new path is longer than the first one due to preferring path safety rather than channel length.

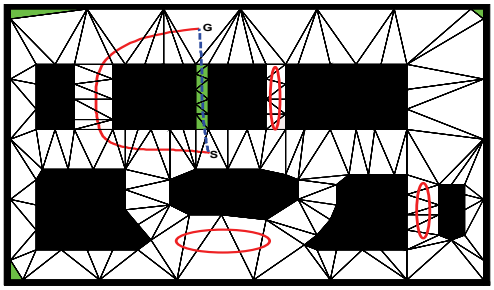


Fig. 7. Path planning in presence of weighted triangles: Forbidden triangles are colored darker (as in the narrow channel), and Unsafe triangles are specified by ovals. The initial (dashed) and regulated paths are also shown.

## X. Motion Control of Synchro-Drive Mobile Robots

In this section we apply a trajectory generation and control strategy for a disc robot with synchro-drive unit. First the dynamics and structure of the synchro-drive mobile robot is described, followed by a discussion on trajectory generation and tracking control.

### A. Structure and Dynamics of the Synchro-drive Robot

The synchro-drive robot is a three-wheeled vehicle equipped with two motors; one for rotating all wheels and producing motion (Drive motor), and the other for turning the wheels and changing direction (Steering motor). Fig. 8 shows the structure of a disc synchro-drive robot.

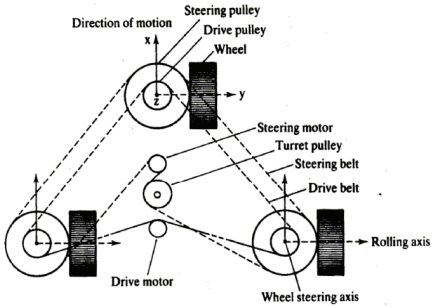


Fig. 8. Structure of a Synchro-drive robot [13].

In this model, all three wheels point at the same direction and turn at the same rate in order to guarantee straight line translation when the Drive motor is not actuated [13]. The kinematics of the synchro-drive robot is given by:

$$\begin{aligned} x(t) &= \int_0^t V(t)\cos(\theta(t))dt \\ y(t) &= \int_0^t V(t)\sin(\theta(t))dt \\ \theta(t) &= \int_0^t \omega(t)dt \\ V(t) &= r_1\omega_1(t), \end{aligned} \tag{9}$$

where $V(t)$, $\theta(t)$ and $r_1$ are the robot's linear velocity, orientation, and the radius of each wheel, respectively. $\omega(t)$ is the robot's angular velocity, and $\omega_l(t)$ is the angular velocity of each wheel when the Drive motor is applied in order.

### B. Trajectory Generation and Control Strategy

As mentioned previously, the planned path consists of a set of points that are connected together with straight lines. Considering that the synchro-drive robot's mechanism allows controlling the orientation and position separately, a simple strategy for point-to-point control of the robot can be applied in two phases:

1. If no turning is needed, the robot is to be driven along the straight line between two points of the planned path with high speed ($V = V_{max}$, $\omega(t) = 0$) (translation mode).
2. At path's nodes the robot stops and rotates about its center to reach to the next direction ($V = 0$, $\omega(t) = \omega_{max}$) (rotation mode).

By applying a PID controller, control laws can be defined as:

$$\begin{aligned} u(t) &= k_P e(t) + k_D \dot{e}(t) + k_I \int e(t)dt \\ u(t) &= k'\omega'(t) \end{aligned} \tag{10}$$

where, $k'$ is the motor constant and $\omega'$ the angular velocity of wheels. $e(t)$ is the feedback error given by:

$$e(t) = \begin{cases} r(t) - r_d(t) & \text{if } \omega = 0 \text{ (translation mode)} \\ \theta(t) - \theta_d(t) & \text{if } V = 0 \text{ (rotation mode),} \end{cases} \tag{11}$$

in which $r(t)$ and $r_d(t)$ are the real and desired positions, and $\theta(t)$ and $\theta_d(t)$ the real and desired directions of the robot. The position and direction control can be done simultaneously, in which case there would be two control loops: one loop for position, and one for direction control.

## XI. Time Complexity

In order to analyze the time complexity of the path planning algorithm, we consider its three phases separately. The time complexity of constructing a Delaunay Triangulation is $O(n\log n)$, $n$ being the number of all obstacle vertices [3]. Since we have binary integer variables, the complexity of BIP solution depends on the number of variables (i.e. triangles). There are $2n+2$ triangles in the workspace, of which $n+2m+2$ triangles are located in $C_{free}$, where $m$ is the number of obstacles. On the other hand, the branching of the triangles' connectivity graph occurs only at triangles having 3 neighbors, which are corresponding to Voronoi vertices, and their number is in $O(m)$. By using the Depth-Best-First search (expanding in depth and then the best solution), the complexity of BIP/LMI solution becomes $O(2^m)$, and so is independent of the number of vertices. The time complexity of path finding is linear in the number of triangles building the optimal channel.

## XII. Conclusion

This paper presents a new method for motion planning of mobile robots. First, the free Configuration space ($C_{free}$) is decomposed into Delaunay triangles, then an optimum channel from initial to goal configurations is found by solving an LMI system to, and finally a solution path is generates within this channel. By assigning weights to some triangles the preplanned path can be regulated and corrected. A control strategy for synchro-drive robots has also been applied for tracking the path. A very important property of this method is its robustness to workspace complexities and local minima, thanks to the first condition of (4).

The obtained path is shorter than the Voronoi roadmap and is near-optimal, with safety considerations applied. The algorithm is complete: it finds the optimal solution if there is one, and reports no solution otherwise. By assigning weights to the triangles, many real-world conditions can be incorporated in the model. The LMIs can be easily extended to $n$-D workspaces by tessellating the space into Delaunay $n$-polytopes.

## References


[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Boston, 2005.

[2] Y. K. Hwang and N. Ahuja, "Gross motion planning - a survey", *ACM Comp. Surveys*, 24(3), (1992), pp. 219-291.

[3] J. C. Latombe, *Robot motion planning*, Kluwer Academic publishers, London, 1991.

[4] P. Bigras, and T. Wong, "An LMI Approach to feedback path control for an Articulated Mining Vehicle", *Electronics*, Aug. 2002.

[5] J. F. Canny, "A Voronoi method for the piano movers' problem", in *Proc. ICRA* 1985, pp.530-535.

[6] H. Choset, and J. Burdick, "Sensor-based exploration: the hierarchical generalized Voronoi graph", *Int. J. of Robotics Research*, Vol. 19, No. 2, (2000), pp. 96-125.

[7] N. S. V. Rao, N. Stolzfus, and S. S. Iyengar, "A 'retraction' method for learned navigation in unknown terrains for a circular robot", *IEEE Trans. Rob. Autom.,* Vol. 7, No. 5, (1991), pp. 699-707.

[8] C. Scherer, and S. Weiland, *Linear Matrix Inequalities in Control*, Feb. 2005, pp. 1-29.

[9] D. P. Didier, "User's guide for SEDUMI INTERFACE 1.04", available online at http://citeseer.ist.psu.edu/616476.html.

[10] Y. J. Lou, J. F. Liu, and Z. X. Li, "An LMI Based Optimal Design of Parallel Manipulators", *Intelligent Robots and Systems*, 2003.

[11] R. K. Prasanth, J. D. Boskovic, and R. K. Mehra, "Mixed integer/LMI programs for low-level path planning," in *Proc. American Control Conference*, Vol. 1, Issue 2002, pp. 608-613.

[12] F. Aurenhammer, and R. Klein, "*Voronoi diagrams*", in J. Sack and G. Urrutia (eds), Handbook of Computational Geometry, Elsevier Science Pub., 2000.

[13] S. Böttcher, "Principles of Robot Locomotion", *in Proc. Human Robot Interaction Seminar SS2006*, Univ. of Dortmund, 2006.\