Optimal Motion Planning of Wheeled Mobile Robots Using Convex Optimization and Receding Horizon Concept

Theodor Chakhachiro

Fall 2020

Content

Introduction
What is motion planning?

- The range of applications of wheeled mobile robots is getting wider and wider
- Applications include space exploration, underwater navigation, search and rescue missions. More common applications are industrial, manufacturing and construction robotics
- All these examples rely on a solid motion planning algorithm to ensure the wheeled robot achieves the given task without collision

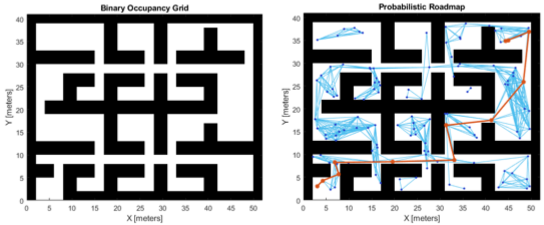


Figure: Occupancy grid map trajectory planning [3]

Introduction
Why Convex Optimization?

- Convex optimization in motion planning applications has received a lot of attention lately, due to the progress in current state of the art computational algorithms
- These problems can be solved quickly and reliably up to a very large size of variables and constraints, that is, if the problem can be convexified

Literature Review
Other Methods

- A wide range of approaches successfully tackled the problem over the years, providing several solutions, each with advantages and disadvantages.
- Early preliminary work in this field used a stochastic model to determine an appropriate trajectory for robots with multiple degrees of freedom [1]
- Novel approaches apply neural network-based planning due to the computational complexity of the motion planning problem [4]
- Vision based path planning is also heavily studied due to the vast availability of camera equipped mobile robots [2]

Chosen Paper
Information

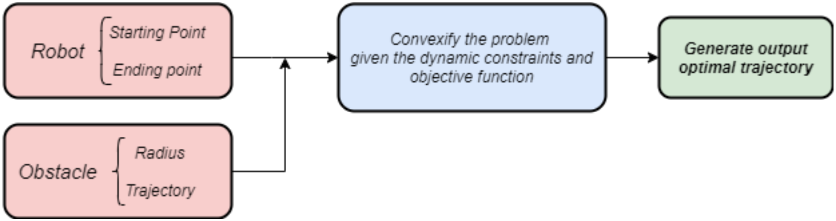
- **Title**: Experimental Study on Optimal Motion Planning of Wheeled Mobile Robot Using Convex Optimization and Receding Horizon Concept [5]
- **Authors**: Mojtaba Zarei, Mehdi Tale Masouleh, and Roya Sabbagh Novin
- **Year**: 2016
- **Conference**: International Conference on Robotics and Mechatronics

Chosen Paper
Summary

- A novel algorithm for collision-free motion planning of two wheeled mobile robots is presented
- The proposed approach stands on discrete motion planning, convex optimization, model-based control, and RHC
- The shortest time and feasible path as the objective functions are considered
- The results of this paper demonstrated that the solution time is less than 0.2 seconds at each stage
- By considering the maximum velocity of this category of the mobile robots, it means the proposed algorithm is quite suitable for real-time applications in motion planning
- All safety conditions and no-obstacle collision was satisfied with the results

Problem Statement

Flowchart



Robot
Starting Point
Ending point
Obstacle
Radius
Trajectory
Convexify the problem given the dynamic constraints and objective function
Generate output optimal trajectory

List of Symbols

Symbol	Interpretation	Units
x_e	Robot x position in the global frame	m
y_e	Robot y position in the global frame	m
θ_e	Robot orientation in the robot frame	rad
v_R	Robot right wheel velocity	m/s
v_L	Robot left wheel velocity	m/s
l	Distance between the wheels	m
h	Control Horizon	-
$z_e(k) \in \mathbb{R}^2$	$2D$ coordinate of the robot at time step k	m
$dt(k) \in \mathbb{R}$	time taken at each step k	s
$z_g \in \mathbb{R}^2$	$2D$ coordinate of the final robot pose	m
s	Number of sides in the polygon	—
v_{max}	Maximum Velocity of the Robot	m/s

Problem Formulation
Dynamic Model

- In [5], the robot used is an E-puck, which is a mobile robot with two wheels. The problem can be scaled to other robots, taking into account their dynamic and kinematic constraints

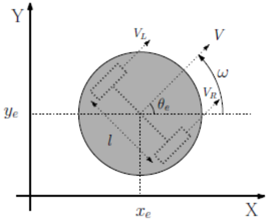


Figure: Global and Robot Frames

- From figure 2, the coordinates of the robot in the global frame can be expressed as:

$$X = [x_e, y_e, \theta_e]^T \tag{1}$$

Problem Formulation

Dynamic Model Cont.

- Taking the derivative with respect to time:

$$\dot{X} = [\dot{x}_e, \dot{y}_e, \dot{\theta}_e]^T \tag{2}$$

- Given the right and left wheel velocities v_R and v_L as well as the distance between the wheels l, we can model the kinematics of the mobile robot:

$$\dot{X}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} \frac{(v_R+v_L)\cos\theta_e}{2} \\ \frac{(v_R+v_L)\sin\theta_e}{2} \\ \frac{(v_R-v_L)}{l} \end{bmatrix} \tag{3}$$

Problem Formulation

Convexification: *Objective Function*

- As stated earlier, the objective function can be a weighted sum of multiple objective functions, depending on the task to be accomplished
- **Minimizing the trajectory length**: In discrete programming, the trajectory can be discretized into k time steps, thus, minimizing the overall path length in done by minimizing the sum of the distance between each consecutive location of the robot in the global frame. Since we are working in 2D, our set of possible coordinates is in $\mathbb{R}^2$, and the first part of our objective function is minimizing the sum of the $\ell - 2$ norm of the difference between each two consecutive robot positions:

$$\min \sum_{k=1}^{h} ||z_e(k+1) - z_e(k)||_2^2 \tag{4}$$

Problem Formulation

Convexification: *Objective Function Cont.*

- **Minimizing the total time**: In the same manner, to minimize the total time taken T, we minimize the sum of the time taken at each step k, labeled as $dt(k)$:

$$\min \sum_{k=1}^{h} dt(k) \tag{5}$$

- **Minimizing Final Pose Error**: To make sure the final point in the trajectory $z_e(h)$ is close to the ending goal z_g given as input, we minimize the distance between the two vectors which is the $\ell - 2$ norm in this case:

$$\min ||z_e(h) - z_g||_2^2 \tag{6}$$

Problem Formulation

Convexification: *Constraints*

- **Collision Avoidance**: To ensure the problem is convex, we need to approximate each obstacle, whether convex or non-convex, as a polygon which is the intersection of half-spaces which in terms, is convex. Let set S denote the set of all points ξ that lie inside the polygon, defined as:

$$S = \{\xi | A\xi < b\} \tag{7}$$

Where $A \in \mathbb{R}^{s \times k}$ and $b \in \mathbb{R}^{s}$. Thus a point ξ lies outside the polygon if one of the Linear Matrix Inequalities is not satisfied, formulated as:

$$A\xi \geq b + (v - 1)M \tag{8}$$

$$\sum_{i=1}^{s} v_i \geq 1 \tag{9}$$

Where M is a constant and $v \in \mathbb{R}^{s}$ is a binary vector such that $v_i \in \{0, 1\}$. This additional constraint will ensure that at least one point ξ will lie outside the polygon defined by the set S.

Problem Formulation

Convexification: *Constraints Cont.*

- **Maximum Velocity Constraint**: We require that the instantaneous velocity at any time step k, between two consecutive points seperated by a period Δt, is less than the maximal velocity of the robot v_{max}:

$$|\frac{z_e(k+1) - z_e(k)}{\Delta t}| \leq v_{max}, \qquad k = 1, ..., (h-1) \qquad (10)$$

Problem Formulation
Final Formulation

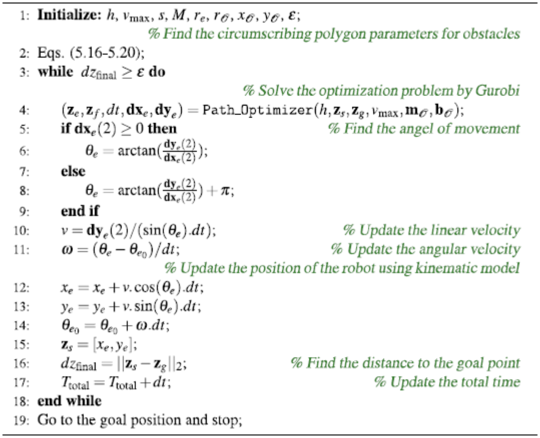
- The problem can be formulated as a weighted sum of the objective functions subject to the stated constraints:

$$\min w_1 \times T + w_2 \times \sum_{k=1}^{h} ||z_e(k+1) - z_e(k)||_2^2 + w_3 \times ||z_e(h) - z_g||_2^2$$

$$\begin{aligned} \textit{subject to} \qquad & A_C(k) z_e(k) \geq b_C(k) + (v(k) - 1)M \\ & \sum_{i=1}^{s} v_i \geq 1 \\ & z_e(1) = z_s \\ & z_e(h) = z_g \\ & \left|\frac{z_e(k+1) - z_e(k)}{\Delta t}\right| \leq v_{max}, \qquad \forall k = 1, ..., (h-1) \end{aligned}$$

Problem Formulation

Proposed Algorithm



```
1: Initialize: $h$, $v_{\max}$, $s$, $M$, $r_e$, $r_{\mathcal{O}}$, $x_{\mathcal{O}}$, $y_{\mathcal{O}}$, $\varepsilon$;
                                  % Find the circumscribing polygon parameters for obstacles
2: Eqs. (5.16-5.20);
3: while $dz_{\text{final}} \geq \varepsilon$ do
                                                  % Solve the optimization problem by Gurobi
4:     $(\mathbf{z}_e, \mathbf{z}_f, dt, \mathbf{dx}_e, \mathbf{dy}_e) = \texttt{Path\_Optimizer}(h, \mathbf{z}_s, \mathbf{z}_g, v_{\max}, \mathbf{m}_{\mathcal{O}}, \mathbf{b}_{\mathcal{O}})$;
5:     if $\mathbf{dx}_e(2) \geq 0$ then                          % Find the angel of movement
6:         $\theta_e = \arctan(\frac{\mathbf{dy}_e(2)}{\mathbf{dx}_e(2)})$;
7:     else
8:         $\theta_e = \arctan(\frac{\mathbf{dy}_e(2)}{\mathbf{dx}_e(2)}) + \pi$;
9:     end if
10:    $v = \mathbf{dy}_e(2)/(\sin(\theta_e).dt)$;                    % Update the linear velocity
11:    $\omega = (\theta_e - \theta_{e_0})/dt$;                       % Update the angular velocity
                                  % Update the position of the robot using kinematic model
12:    $x_e = x_e + v.\cos(\theta_e).dt$;
13:    $y_e = y_e + v.\sin(\theta_e).dt$;
14:    $\theta_{e_0} = \theta_{e_0} + \omega.dt$;
15:    $\mathbf{z}_s = [x_e, y_e]$;
16:    $dz_{\text{final}} = ||\mathbf{z}_s - \mathbf{z}_g||_2$;          % Find the distance to the goal point
17:    $T_{\text{total}} = T_{\text{total}} + dt$;                               % Update the total time
18: end while
19: Go to the goal position and stop;
```

Figure: Algorithm [5]

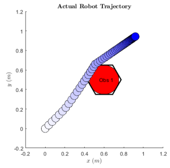
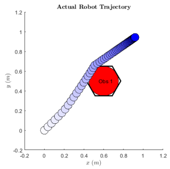
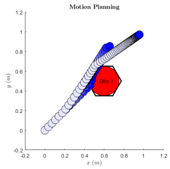
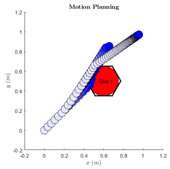
Results
General Overview

- **Solver**: Since SDPT3 and SEDUMI cannot deal with MIP, the solvers used are Gurobi and Mosek in MATLAB CVX
- Adding/Removing Constraints
- Changing the Objective Function
- Sensitivity Analysis on the weights in the ojective function
- Dynamic Obstacles
- Real-time Gazebo Simulation

Results

Comparison Between Gurobi and Mosek

- Minimizing Trajectory Length, One Obstacle, constant *dt*:

Gurobi	Mosek
	
	

Results

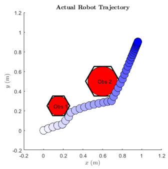
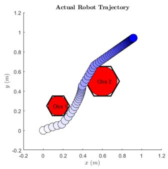
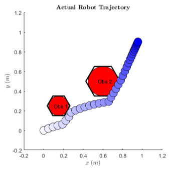
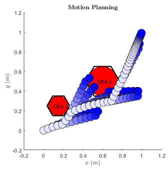
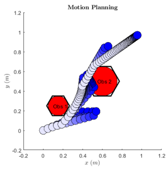
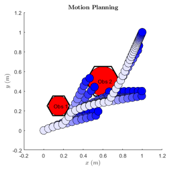
Comparison Between Gurobi and Mosek Cont.

Output	**Value Gurobi**	**Value Mosek**	**Units**
Solving Time	86.177985	78.529866	s
Time Taken	48.000000	48.000000	s
Trajectory Length	1.451358	1.451237	m
Final Robot Pose	[1.005776,0.902083]	[1.005689,0.901989]	m

- **Note**: The reason behind the large solving time is because plots were being generating simultaneously. It should be noted that the solving time for each iteration without plotting is around 0.3 s

Results

Changing the Objective Function for two obstacles

Time	Trajectory Length	Final Pose Error
		
		

Results

Changing the Objective Function for two obstacles Cont.

Table: Minimizing Total Time

Output	**My Values**	**Paper Values**	**Units**
Solving Time	110.442174	-	s
Time Taken	37.000000	26.2	s
Trajectory Length	1.451682	1.68	m
Final Robot Pose	[0.963372,0.911120]	-	m

Table: Minimizing Trajectory Length

Output	**My Values**	**Paper Values**	**Units**
Solving Time	324.707134	-	s
Time Taken	60.000002	28	s
Trajectory Length	1.379427	1.49	m
Final Robot Pose	[0.919334,0.943355]	-	m

Results

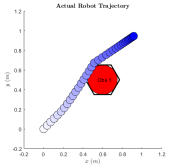
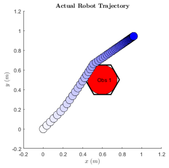
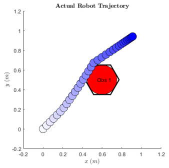
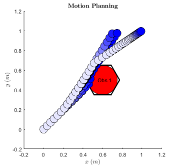
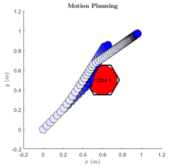
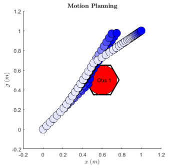
Changing the Objective Function for two obstacles Cont.

Table: Minimizing Final Pose Error

Output	**My Values**	**Paper Values**	**Units**
Solving Time	99.559352	-	s
Time Taken	35.000001	-	s
Trajectory Length	1.438994	-	m
Final Robot Pose	[0.960890,0.912268]	-	m

Results

Changing the Objective Function for one obstacle

Time	Trajectory Length	Final Pose Error
		
		

Results

Changing the Objective Function for one obstacle Cont.

Table: Minimizing Total Time

Output	**My Values**	**Units**
Solving Time	106.568922	s
Time Taken	35.000000	s
Trajectory Length	1.345827	m
Final Robot Pose	[0.925170,0.949502]	m

Table: Minimizing Trajectory Length

Output	**My Values**	**Units**
Solving Time	223.089158	s
Time Taken	53.000000	s
Trajectory Length	1.339049	m
Final Robot Pose	[0.922097,0.945878]	m

Results

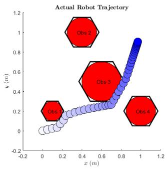
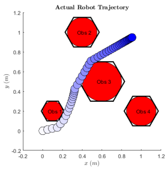
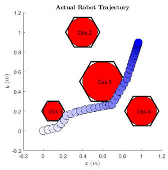
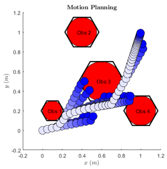
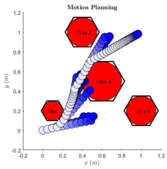
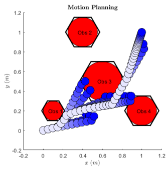
Changing the Objective Function for one obstacle Cont.

Table: Minimizing Final Pose Error

Output	**My Values**	**Units**
Solving Time	103.409956	s
Time Taken	33.000001	s
Trajectory Length	1.340034	m
Final Robot Pose	[0.920406,0.946300]	m

Results

Changing the Objective Function for four obstacles

Time	Trajectory Length	Final Pose Error
		
		

Results

Changing the Objective Function for four obstacles Cont.

Table: Minimizing Total Time

Output	**My Values**	**Units**
Solving Time	132.113480	*s*
Time Taken	38.999999	*s*
Trajectory Length	1.484363	*m*
Final Robot Pose	[0.971783,0.912304]	*m*

Table: Minimizing Trajectory Length

Output	**My Values**	**Units**
Solving Time	185.934544	*s*
Time Taken	47.000002	*s*
Trajectory Length	1.404373	*m*
Final Robot Pose	[0.913716,0.950895]	*m*

Results

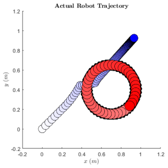
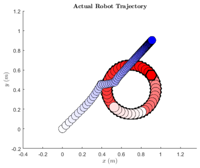
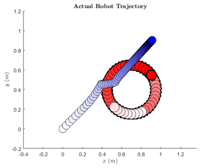
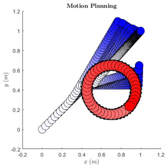
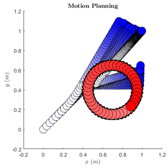
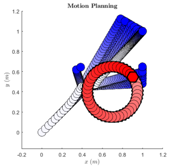
Changing the Objective Function for four obstacles Cont.

Table: Minimizing Final Pose Error

Output	**My Values**	**Units**
Solving Time	127.135271	s
Time Taken	37.000001	s
Trajectory Length	1.477996	m
Final Robot Pose	[0.969828,0.906250]	m

Results

Changing the Objective Function for a dynamic obstacle

Time	Trajectory Length	Final Pose Error
		
		

Results

Changing the Objective Function for a dynamic obstacle Cont.

Table: *Minimizing Total Time*

Output	**My Values**	**Units**
Solving Time	4266.559812	s
Time Taken	64.000001	s
Trajectory Length	1.408223	m
Final Robot Pose	[0.925913,0.917499]	m

Table: *Minimizing Trajectory Length*

Output	**My Values**	**Units**
Solving Time	5036.113212	s
Time Taken	82.000001	s
Trajectory Length	1.350057	m
Final Robot Pose	[0.933590,0.926204]	m

Results

Changing the Objective Function for a dynamic obstacle Cont.

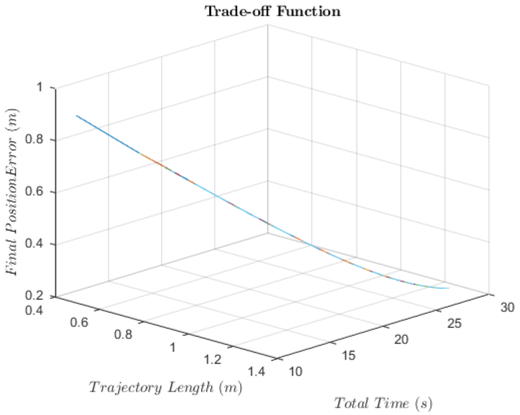
Table: Minimizing Final Pose Error

Output	**My Values**	**Units**
Solving Time	4141.456682	s
Time Taken	73.000004	s
Trajectory Length	1.387274	m
Final Robot Pose	[0.933940,0.946300]	m

Results

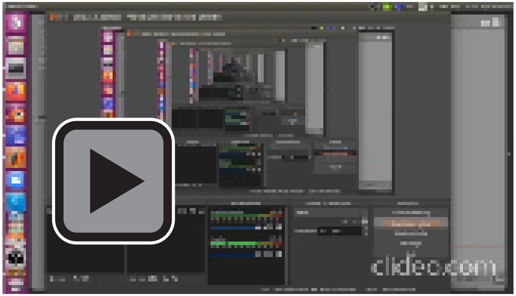
Trade-off Function for one obstacle

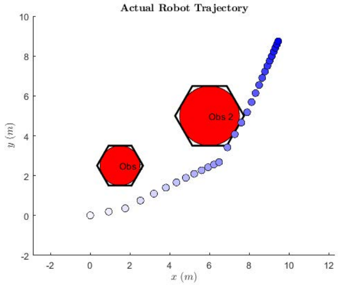
- Fixing the weight for the minimum length objective function and varying the remaining weights:



Results

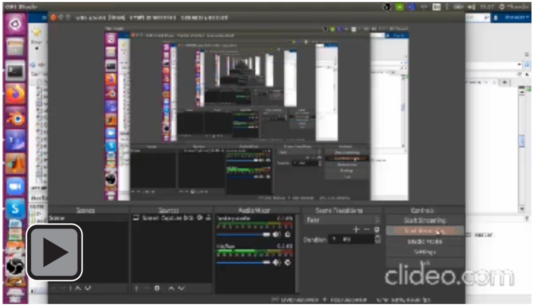
Gazebo Simulation: Minimizing total time for two obstacles

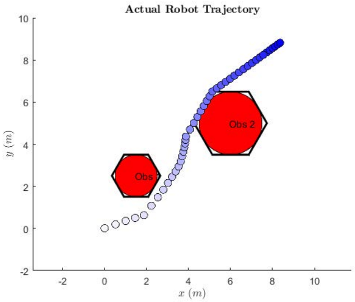




Results

Gazebo Simulation: Minimizing trajectory length for two obstacles





Conclusion and Possible Extensions

- Motion planning and control of mobile robots using convex optimization and discrete planning structure and receding horizon concept were discussed
- Proposed algorithm was implemented in MATLAB using CVX with Mosek and Gurobi Solvers
- This method can be extended to other mobile vehicles, taking into consideration dynamic and kinematic constraints
- This method can also be extended for different environments with different obstacles

References I

[1] J. Barraquand and J. . Latombe, *A monte-carlo algorithm for path planning with many degrees of freedom*, Proceedings., IEEE International Conference on Robotics and Automation, 1990, pp. 1712–1717 vol.3.

[2] B. Hummel, S. Kammel, Thao Dang, C. Duchow, and C. Stiller, *Vision-based path-planning in unstructured environments*, 2006 IEEE Intelligent Vehicles Symposium, 2006, pp. 176–181.

[3] Mathworks, *Matlab learning*, 2020.

[4] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, *Motion planning networks*, 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 2118–2124.

References II

[5] M. Zarei, R. S. Novin, and M. T. Masouleh, *Experimental study on optimal motion planning of wheeled mobile robot using convex optimization and receding horizon concept*, 2016 4th International Conference on Robotics and Mechatronics (ICROM), 2016, pp. 386–391.