# Robot trajectory optimization

*EE159 Course Project

***Abstract*—This project is a method of robot trajectory planning. The problem will be formulated as a convex optimization problem. The main purpose of the method is to get a shortest path from point to point which avoid all obstacles. The method will use the objective function to limit the length of the path. And the obstacles will be represent as constrains in the optimization problem.**
**Since the terrain in the real world in highly non-convex, one important work of this method is to make the constrains of the function convex. Then the problem could be solved in the way for solving convex optimization problems.**

***Index Terms*—Convex optimization, robotics, trajectory planning, convexification**

## I. Introduction

Trajectory optimization algorithm plays an important role in robotics area. By working on motion planning, it helps a robot or a self-driving car moving in a short route, avoid changing direction rapidly and keep away from obstacles in the environment.

The first step for getting a good moving trajectory is to get a path avoid all obstacles. So, the main purpose of this project is to compute a path avoid all obstacles with a convex optimization way.

Besides getting a accessible way, the main request of trajectory optimization is the computing complexity. For a robot working in an unknown environment, the planning job will not be able to be done before the robot starts working. For example, the self-driving car should be able to face a changing traffic environment and an assistant robot may face unknown obstacles. Therefore, a real time planning algorithm is needed for replanning when new constrains from environment occur. And a low computing complexity makes this demand possible and affordable.

Among several methods could be used on dealing the trajectory planning problem, convex optimization is a relative efficient way. Therefore, we want to formulate this problem as a convex optimization problem.

The trajectory optimization problem usually try to minimize the time spent of a movement. Therefore, it usually has a linear function of time as an objective function, which is convex. But because of the complex and unknown environment(i.e. non-convex obstacles), this problem may have non-convex constrains. This makes the trajectory optimization problem a non-convex problem.

Since non-convex optimization cost a high calculated load and usually can not guarantee a global optimal result, to meet the request of computing complexity people try to do convexification on non-convex problem. There already exist several methods for solving convex optimization problem, they are efficient and reliable. By formulate the trajectory optimization problem as a convex problem, we will be able to use these mature ways to solve this problem.

This project will propose a method of convexification for the trajectory optimization problem, which has fully considered characteristics of the environment therefore is efficient on this problem. Based on the generated convex problem, this paper will contribute an algorithm to solve the problem and do an experiment case study to prove the algorithm.

## II. Related workds

There are mainly three ways to solve Trajectory optimization problem, Numerical Integration (NI), Dynamics Programming (DP) and Convex Optimization (CO) [1]. They all guarantee the global optimal.

### A. Numerical Integration

For NI algorithm, Bobrow et al. proposed an algorithm to minimize the time spend of a manipulator control system. It can be used for any known dynamic equations of motion when all limitations are known [2]. They proves the optimal result is composed by maximally accelerating and decelerating segments [1]. But computing the point to switch between accelerating and decelerating segments cost a high time spend, especially when constrains are complex.

### B. Dynamics Programming

The idea of dynamic programming is to separate the space to a several grids, then treat them as a matrix as people usually do in dynamic programming process. The transform function is the minimize value of the time spend all eligible paths from a start position. The problem for this method is to get a ideal path, the algorithm should decompose the space to a fine degree. This cost a high time spend when computing.

### C. Convex Optimization

Since there already exist several efficient method to solve convex problem and convex optimization guarantee the global optimal, formulate the problem as a convex one is a good choice.

If we formulate the robot motion planning problem as an optimization problem, usually the object function is convex. But the constrain functions could be non convex because of the environment.

One way to deal with this problem is convexification, make a non convex problem to a convex one, then it will be much

easier to deal with.
There is a relative mature way for this convexification, sequential quadratic programming (SQP) [3]. This method would change a non convex problem to a sequence of convex problems, then can be deal in ways for convex problems. But this method is designed to be used in a general way, in other words, not designed to be used in motion planning problems. Therefore it does not consider geometric characteristics in motion planning problems.
In improve the effectiveness and efficiency, someone implements another algorithm, the convex feasible set algorithm (CFS) [4]. This algorithm also transform a non convex problem to several convex problems and solve it iteratively. And it has a different way to obtain the convex subproblems, so it could be faster than SQP is. But its problem is it sometimes may not converge to local optima under nonlinear equality constraints. Then the author implements an advanced algorithm to fix it by relax the nonlinear equality constraints to several nonlinear inequality constraints using slack variables [5].

### D. Non-Convex Optimization

Non-Convex Optimization is also a way to solve this problem.
Gasparetto et al. suggest an objective function composed by two parts : a first term proportional to the total execution time and another one proportional to the integral of the squared jerk (defined as the derivative of the acceleration) along the trajectory. [6] By minimizing them the algorithm got a fast and smooth path to the target.
However, the disadvantages of this method is it cost a high time complexity and does not guarantee the global optimal, also sometimes it needs parameter turning for specific tasks in order to get an eligible result.

## III. Problem Formulation

### A. Objective Function

An eligible path for motion planning should avoid obstacles, meet all constrains from requirements from robots functions and mechanical constrain. And it should be as short as possible.
We express the result trajectory as a sequence of point. The connection between these points is the whole trajectory. We could decide how many points will used to composed a trajectory based on the request of accuracy. A large number of point could make the result more close to the optimal choice but cost more time to compute.
We initialize a reference path, which link the start point and the end point with a straight line. Obviously, it is the shortest path.
Since the reference is the shortest way to the target position, we want the new trajectory be close to the reference trajectory. And to limit the time spend, we also want the trajectory try to make itself be short.
The result trajectory might go around a long way to make itself be closed to the reference trajectory without a part of the objective function to limit the length of the trajectory itself.
Therefore the objective function is composed by two parts,

$$minimize$$

$$f(x) = a * \sum_{i=2}^{n} \|x_i - x_{i-1}\|_2 + b * \|x - reference\|_2 ,$$

where x is the decision variable, a and b are positive parameters.
Assumption 1: the function f should be convex.
We use a constant speed for this analysis now. But if there is any demand about speed, acceleration and steering angle, they can all be added to the first part of this objective function as long as they can be express in a convex function of x. And such a modify preserves the convexity of the whole function. It would not change the way we solving the problem, so this method also works on these problems.

### B. Constrain Functions

Since the feasible set for x is not convex, we try to extract n convex set from it. (n is the number of positions in the trajectory)

$$F_i \subseteq F,$$

where F is the whole feasible set of the area.
Then we have

$$x_i \in F_i,$$

as constrains, where $x_i$ are all points in the trajectory. And all $F_i$ are convex.
Assumption 2: the whole space for the problem is a close rectangle area.
Also, if there is any limit of speed, acceleration and steering angle, it can also be added to constrain functions as long as it can be express as a convex function of x.

## IV. Feasible sets Formulation

### A. Obstacles Expression

To facilitate analysis, we assume all obstacles can be approximate to several straight lines. And the original expression of an obstacles is a set of points, which are the vertices of the polygons of the obstacles.These vertices are arranged clockwise along the edge of the polygon.
These polygons are not necessarily convex.

### B. Obstacles Approximate

We approximate all individual obstacles to a circle. Then by limit the distance between a step point to the center of circles, we can formulate constrain functions of points in the result trajectory for all obstacles.
But this methodwill lead a lead to huge error if the shape of an

obstacles is long and narrow. So before do the approximating, we try to cut such a obstacle to several pieces. These pieces' scale ratio in x, y direction needs to be close to one to one and their longest edge should be less than a tolerance value. Then we could do approximate on these pieces separately. The approximate could be done by solving a simple convex optimization problem.

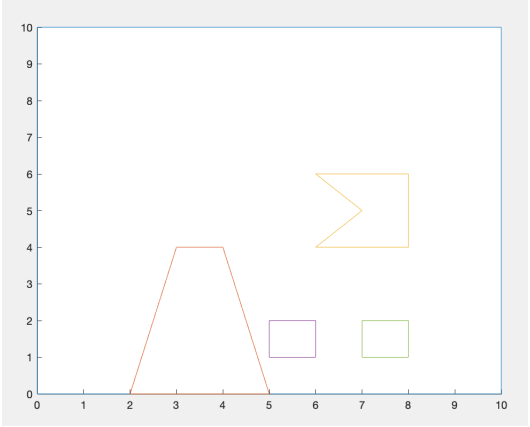
$$minimize$$

$$\pi r^2 - S$$

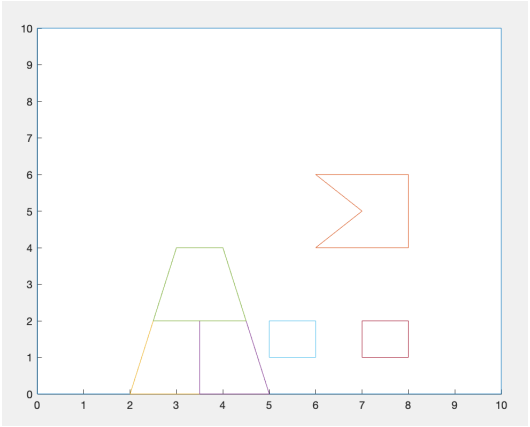
$$subject \quad to$$

$$o_i - c \succeq 0 \text{ for all } o_i$$

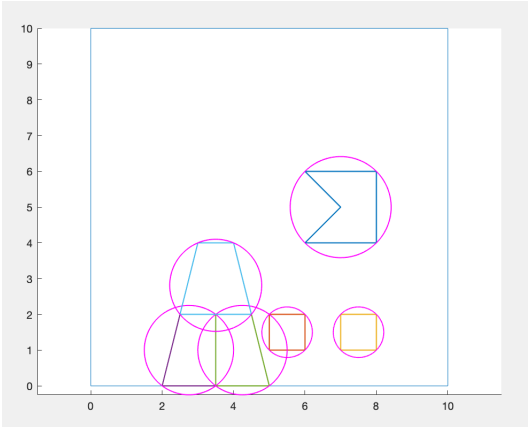
where S and $o_i$ are parameters, represent the area of the obstacle and the vertices of it, r and c are variables represent the radius and the center of the circle.
The objective function is to limit the gap of this approximating by making the feasible space in the obstacle circle as small as possible. Constrain functions guarantee the whole obstacle is inside the circle.
In this example, we choose $\sqrt{22}$ as the tolerance value. To get a higher accuracy, this value should be set lower, but it will cost a higher time spend.



pic1:The boundary of the whole area and obstacles



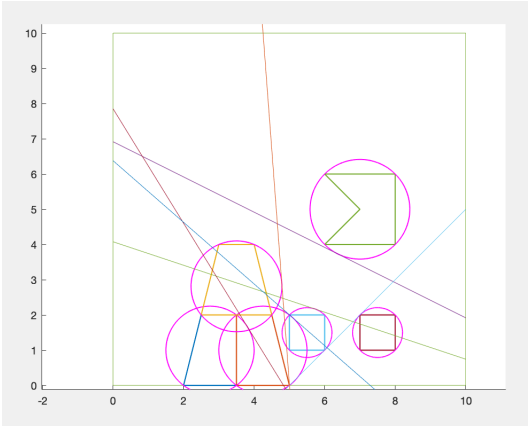
pic2:Obstacles after division



pic3:Approximated obstacles

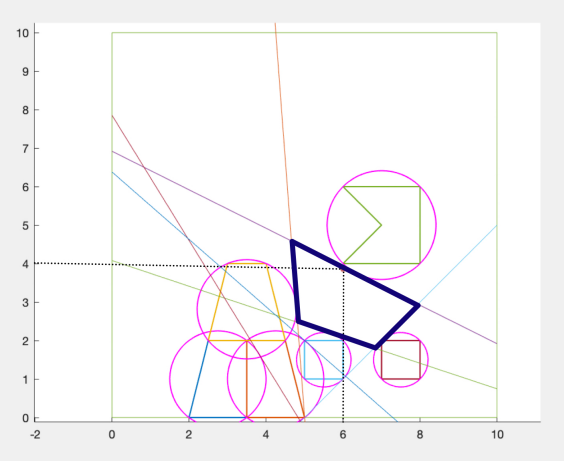
### C. Compute Feasible Sets

The number of points in the result trajectory is same as it in the reference trajectory. Every point gets its feasible set based on the point corresponding to the sequence in the reference trajectory.
For each point in the reference trajectory, we compute the intersection of an obstacle circle and the line connects the center point of the circle and the point of trajectory. Then calculate the tangent of the circle at this point. For m obstacles, we get m lines. These line enclose a closed area, which is the feasible set for the new point. Since this area is constrained by a finite number of linear inequalities, it is a polyhedra and is convex.
For example, for a point (6, 4), we get the set of obstacles and set of tangent.



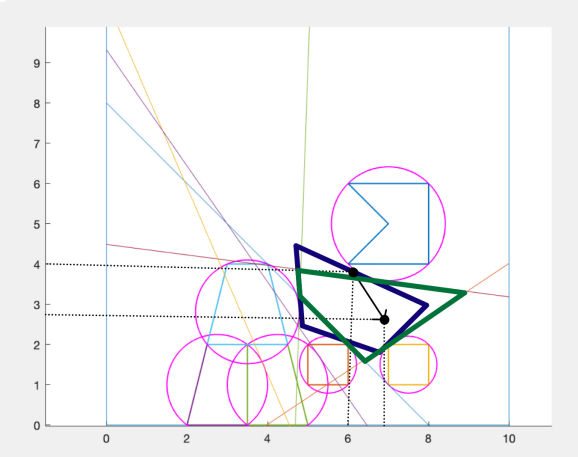
pic4:Set of obstacles and set of tangent for (6,4)

The feasible set based on this point is the blue area in the picture.



pic4:Feasible set for (6,4)

For a possible next step point (6.7, 2,7), we can get a new feasible set. The line between them is a partial trajectory of the problem.



pic4:Feasible set for (6.7,2.7)

The green area in the picture is the feasible set for (6.7, 2.7). The black line is the partially trajectory between these two points.
In addition, there is an extra return value when computing tangents. It is used to record whether the reference point is under or above the tangent lines.

## V. Algorithm

The algorithm needs inputs of the start point, the target point and information about the environment. As we discussed in the section of Problem Formulation, other convex limits and demands like speed and steering angle could also be added into the objective function or constrain functions. Information about those should also be included in the input if they are considered.
When implement this method, we need to know all information about all the obstacles and the boundary of the area. If a new obstacle appears in the environment, we should re-plan the trajectory by re-run the whole algorithm and use the current point as the new start point.

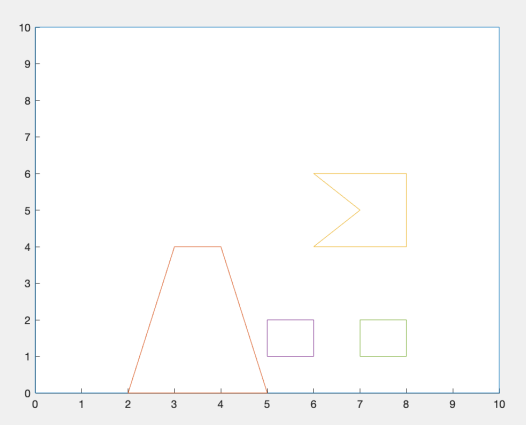
The whole algorithm is :

- Initialize a straight line between the start position and the target position as the reference trajectory. Because it must be the shortest way if we do not consider any obstacles. If any point is inside a obstacle move it vertically until out of the obstacles.
- Compute the feasible set based on points in the reference trajectory.
- Use all the position points in the trajectory as decision variables, the line between them is the new trajectory. Solve the convex problem that minimize the distance from the reference trajectory to the decision variables and the length of the new trajectory itself.
- Use the solution of this convex problem as a new reference trajectory and go back to step 2 solve the convex problem again.
- Break this loop if the optimal value stop decreasing.

The main idea of the algorithm is using the whole trajectory instead of one point as the optimization variable. The traditional way is to compute the next point based on the current point. This could only make the trajectory move towards the target point in a relative short way, but could not consider whether the whole trajectory is close to the optimal path or not. So, the result of the old method might not be ideal if the environment is tortuous. Our algorithm fix this defect by using the whole trajectory as the optimization variable.

## VI. Result

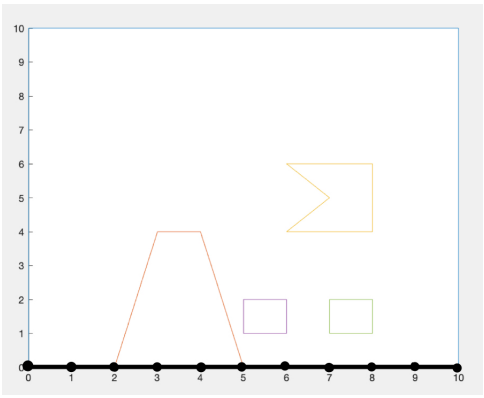
Using this method on the example of this graph.



Suppose the robot is trying to move from the lower left corner to the lower right corner. The start point in this problem is (0,0) and the target point is (10,0). The obstacles is expressed as follow.

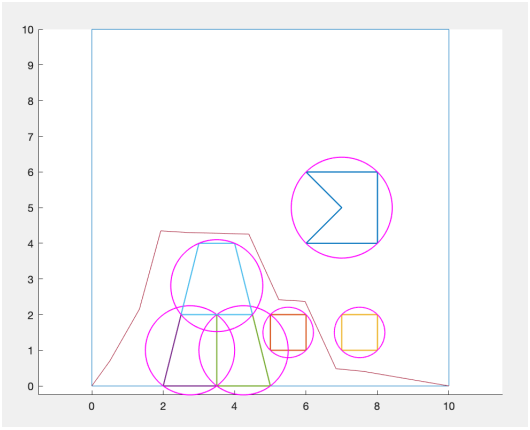
```
o1 =

     2     0
     3     4
     4     4
     5     0
     2     0

o2 =

     6     4
     7     5
     6     6
     8     6
     8     4
     6     4

o3 =

     5     1
     5     2
     6     2
     6     1
     5     1

o4 =

     7     1
     7     2
     8     2
     8     1
     7     1
```

We first get the initial reference trajectory.



pic5:Initial Reference Trajectory

At last, by running the algorithm and solving the convex optimization problem, we get a result trajectory,



pic6:Result Trajectory

where the orange line is the result trajectory. In this example, we use 10 steps in the optimization variable. So it is still a rough broken line. Using a larger number of steps would make the trajectory more like a smooth curve.

## VII. Future Works

This algorithm uses a circle to approximate all obstacles. To using different ways to measure different type of obstacles, for example convex obstacles or non convex obstacles, and using a more complex and efficient way to approximate obstacles might reduce the error. Computing how many steps are required at least to achieve the target is also necessary to make this algorithm can be used in a real scene. Also, the Matlab implement of this algorithm sometimes still makes some mistakes like abscissa drift. This should be fixed in the follow-up work. And a pipeline of all parts of this algorithm should be implement integrally.

## References


[1] Tien Hung Pham. Time-optimal motion planning and control. 2019.

[2] James E Bobrow, Steven Dubowsky, and John S Gibson. Time-optimal control of robotic manipulators along specified paths. *The international journal of robotics research*, 4(3):3–17, 1985.

[3] Peter Spellucci. A new technique for inconsistent qp problems in the sqp method. *Mathematical Methods of Operations Research*, 47(3):355–400, 1998.

[4] Changliu Liu, Chung-Yen Lin, and Masayoshi Tomizuka. The convex feasible set algorithm for real time optimization in motion planning. *SIAM Journal on Control and optimization*, 56(4):2712–2733, 2018.

[5] Changliu Liu and Masayoshi Tomizuka. Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification. *Systems & Control Letters*, 108:56–63, 2017.

[6] Alessandro Gasparetto and Vanni Zanotto. A technique for time-jerk optimal planning of robot trajectories. *Robotics and Computer-Integrated Manufacturing*, 24(3):415–426, 2008.

[7] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009.