清華大學
Tsinghua University

Tsinghua University
Convex Optimization: Theory and Application

Complete Coverage Path Planning

Course Project

Author
Chenghua Liu, Chenyu Wang, Ziyue Li, Xinran Gu

December 27, 2020

Contents

Chapter 1

Body

1.1 Preface

The cleaning vehicle path planning problem is an intriguing and challenging optimization problem. In this report, we provide both theoretical analysis and a heuristic stepwise approach to solve the problem. Our report is organized as follows: Section 1.2 reviews the problem definition and proposes the formulation of typical optimization problem for this task. At the same time, we discuss the potential difficulties in directly solving the optimization problem, leading to a stepwise approach proposed in Section 1.3. As the last part, some further discussions and extensions are involved in Section 1.4 such as upper and lower bound analysis and potential generalization of our method. To sum up, **we derive the shortest path with length 24.90**. The path is shown in Figure 1.6.

1.2 Problem Description

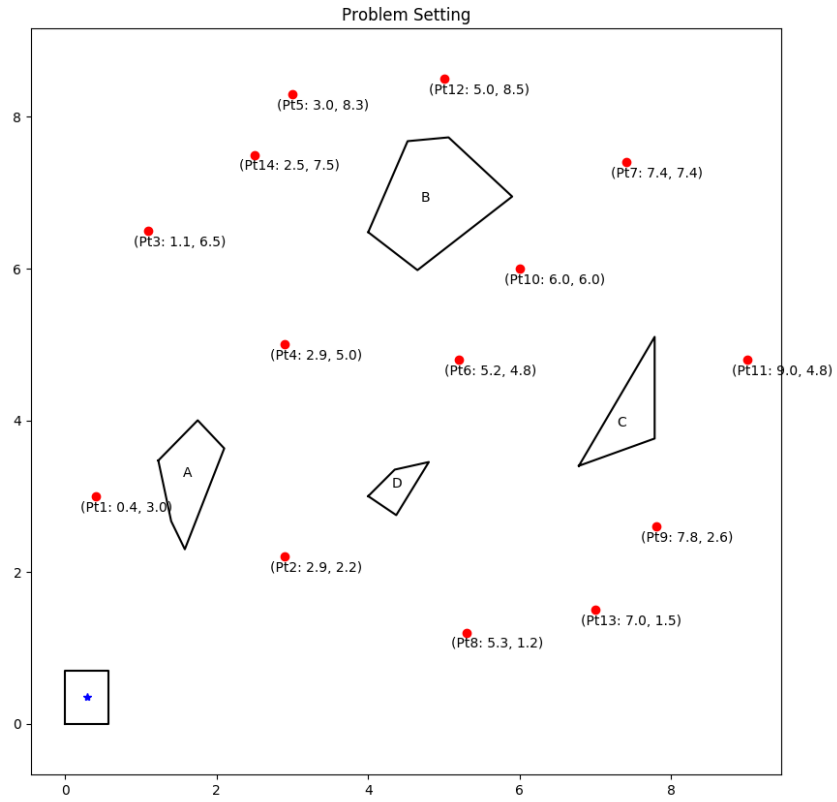

Figure 1.1: Problem setting

Basic setups for the shortest path planning problem are shown in Figure 1.1. As an optimization problem, we first discuss the three key related issues: objective function, constraint conditions and decision variables. Then a discussion related to such optimization problem is proposed to lead to our solution in Section 1.3.

1.2.1 Decision Variables

It's typically hard to formulate the whole path of the cleaning robot. Thus a commonly used way is to sample enough points in the path at equal time intervals and utilize these waypoints to represent the whole track. For each sampled point, three variables are utilized to describe: (x_i, y_i, θ_i) where x_i, y_i represents the coordinate of the robot center and θ_i represents its rotation angle with respect to the horizontal axis. Suppose that N points are sampled in total. Thus the decision variables for this problem is $\mathbf{z} = [\mathbf{z}_1^T, \ldots, \mathbf{z}_n^T]^T$, where $\mathbf{z}_i = [x_i, y_i, \theta_i]^T$. Besides, for simplicity, we also denote $\mathbf{z}_0$ as $[0, 0, 0]^T$ to represent the initial point.

1.2.2 Constraints

Three constraints, rubbish coverage, obstacle avoidance and limited rotation, are discussed respectively as the following.

- **Rubbish Coverage**

  For simplicity, we require that the m rubbishes is covered by the robot in one of the sampling points and neglect the potential coverage in path between two points. Thus this constraint can be formulated as:

  $$r_i \in \cup\{V_j, j = 0, \ldots, n\}, \forall i = 1, \ldots, m$$

  where r_i represents the coordinate of the ith rubbish and V_j represents the rectangular area of the jth sampling point of the robot.

  Given $\mathbf{z}_j = [x_j, y_j, \theta_j]^T$, we can derive the expression for $r_i \in V_j$ when $\theta_j \neq k\pi/2, k \in \mathbb{N}$ (cases when $\theta_j = k\pi/2$ are degenerated and easy to deal with):

  $$\begin{pmatrix} -\tan\theta_j & 1 \\ \tan\theta_j & -1 \\ \frac{1}{\tan\theta_j} & 1 \\ -\frac{1}{\tan\theta_j} & -1 \end{pmatrix} \cdot r_i \leq \begin{pmatrix} -\tan\theta_j(x_j - a\sin\theta_j) + y_j + a\cos\theta_j \\ \tan\theta_j(x_j + a\sin\theta_j) - (y_j - a\cos\theta_j) \\ \frac{1}{\tan\theta_j}(x_j + b\cos\theta_j) + y_j + b\sin\theta_j \\ -\frac{1}{\tan\theta_j}(x_j - b\cos\theta_j) - (y_j - b\sin\theta_j) \end{pmatrix}$$

  where $2a = 0.57$, $2b = 0.70$ are the width and length of the vehicle respectively.

- **Obstacle Avoidance**

  For non-convex obstacles, we can turn it to a convex one by split them into convex sub-obstacles or approximate them by their convex hull. Therefore, we can only consider the case with convex obstacles. For the condition that the rectangular area of the robot V_j does not intersect with the convex obstacle $Poly_k$, it can be naturally viewed as the lack of feasible solution for the problem

  $$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in V_j, \mathbf{x} \in Poly_k \end{aligned}$$

  However, since this is hard to be expressed as a constraint, an alternative way involves line separation of two polygons:

  $$\exists l_w, \quad \text{s.t.}\, \min\{a_i^{l_w}\} > \max\{b_k^{l_w}\} \quad \text{or} \quad \min\{b_k^{l_w}\} > \max\{a_i^{l_w}\}$$

  where $a_i^{l_w}, b_k^{l_w}$ are the coordinates of apexes' one-dimensional projections on line l_w for V_j and $Poly_k$ respectively, and l_w is the vertical line of edge w of V_j or $Poly_k$. This is to say, two convex polygons do not intersect with each other if one of the two polygons' edges can separate the two polygons, i.e. projections of the two polygons' apexes on the vertical direction of this edge do not intersect with each other in one-dimensional space.

- **Limited Rotation**

  The constraint is levied on two adjacent sampling points, indicating

  $$-\alpha_{max} \leq \theta_{i+1} - \theta_i \leq \alpha_{max}, \quad i = 0, \ldots, n-1$$

  where $a_{max} = 0.2\pi$ is the maximum permittable rotation angle.

1.2.3 Objective Function

Suppose that p sampling points are utilized satisfying $p \leq 100$, then our target to minimize the length of planned path can be formulated to the objective function:

$$\begin{aligned}\min_{\mathbf{z}} L(\mathbf{z}) &= \sum_{i=0}^{p-1} ||(x_{i+1}, y_{i+1}), (x_i, y_i)||_2 \\ &= \sum_{i=0}^{p-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}\end{aligned}$$

1.2.4 Discussions

Although an optimization problem can be formulated as above, there are some difficulties. Many unions of convex sets occurred in constraints, which gives rise to non-convex constraints. This can be solved by penalize some constraints into the objective function, or undertaking constraint relaxation. As for our approaches, we make the relaxation by separating the problem into several sub-problems, solving each step-wise with optimization methods, integrating them together and making slight adjustments. This will be discussed in detail in Section 1.3.

1.3 Proposed Approach

As discussed in Section 1.2, the problem involves **arranging the visiting order, ensuring coverage, avoiding collision and determining the turning angles**. While the four sub-problems are intensely coupled with one another and jointly determines the optimal solution, we have great difficulty solving them simultaneously due to the complexity of constraints. To break out of the rut, we propose a sequential approach to tackle the four sub-problems one by one, which provides a sufficiently satisfying, if not optimal, solution.

1.3.1 Visiting Order: A Variant of Travelling Salesman Problem

When we use the location of rubbish as a proxy for the location of the cleaning vehicle, the determination of visiting order is almost the same as the well-known **Travelling Salesman Problem (TSP)**, except that:

- A loop is not required. That is, the vehicle does not need to return to the starting point.
- The starting point is fixed.

We add a dummy node, say node 15 in this specific problem (numbering starting from 0), whose distances from all other nodes are set as 0. We use Google OR-Tools [1] to solve the problem, which allows us to designate the starting point (i.e the center of the given starting position) and the ending point (i.e. the dummy node). The derived optimal visiting order is $0 \to 1 \to 2 \to 8 \to 13 \to 9 \to 11 \to 7 \to 10 \to 6 \to 4 \to 3 \to 14 \to 5 \to 12 \to 15$, with a total travelling distance of 29.27. (See Figure 1.2)

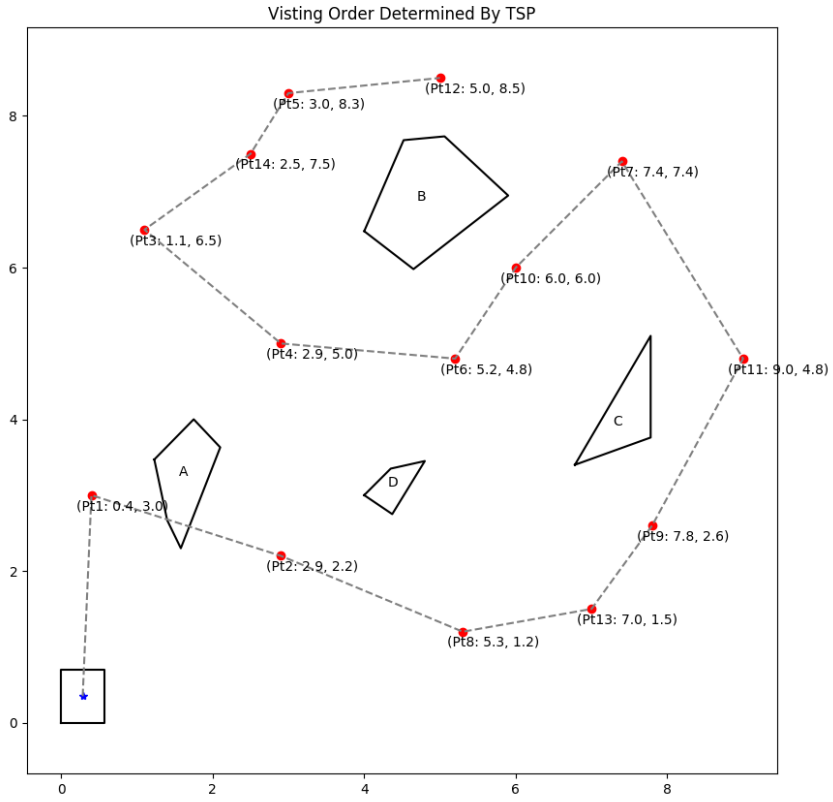

Figure 1.2: Visiting Ordered Determined By TSP

1.3.2 Rubbish Coverage: A Mathematical Programming

The travelling distance obtained in Section 1.2 serves as a rough estimate for the objective function value. It can be further improved since we do not force the center of the cleaning vehicle to coincide with the rubbish. Instead, the cleaning vehicle can rotate around its center to cover the rubbish. Therefore, its center should fall within the circle, whose center is the rubbish and radius is $\frac{1}{2}\sqrt{w^2+h^2}$(i.e. the radius of the circumcircle of the vehicle).

In this subsection, we formulate a mathematical programming with a path minimization objective and rubbish coverage constraints, without considering collision avoidance.

Denote the centers of the vehicle $x_i, i = 0, \ldots, 14$, where $x_0 = 0.57/2, y_0 = 0.7/2$ is the starting point. Denote the location of rubbish (indexed by the visiting order)i as $(a_i, b_i), i = 1, \ldots, 14$. Let $w = 0.57, h = 0.7$, we derive the following mathematical programming:

$$\begin{aligned} \min \quad & \sum_{i=0}^{13} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \\ \text{s.t.} \quad & (x_i - a_i)^2 + (y_i - b_i)^2 \leq \frac{1}{4}(w^2 + h^2) \qquad i = 1, \ldots, 14 \end{aligned}$$

Note that the objective function and each constraint are convex. We use scipy.optimize[2] to solve the convex programming. The optimal value is 24.75 with the trail shown in Figure 1.3.

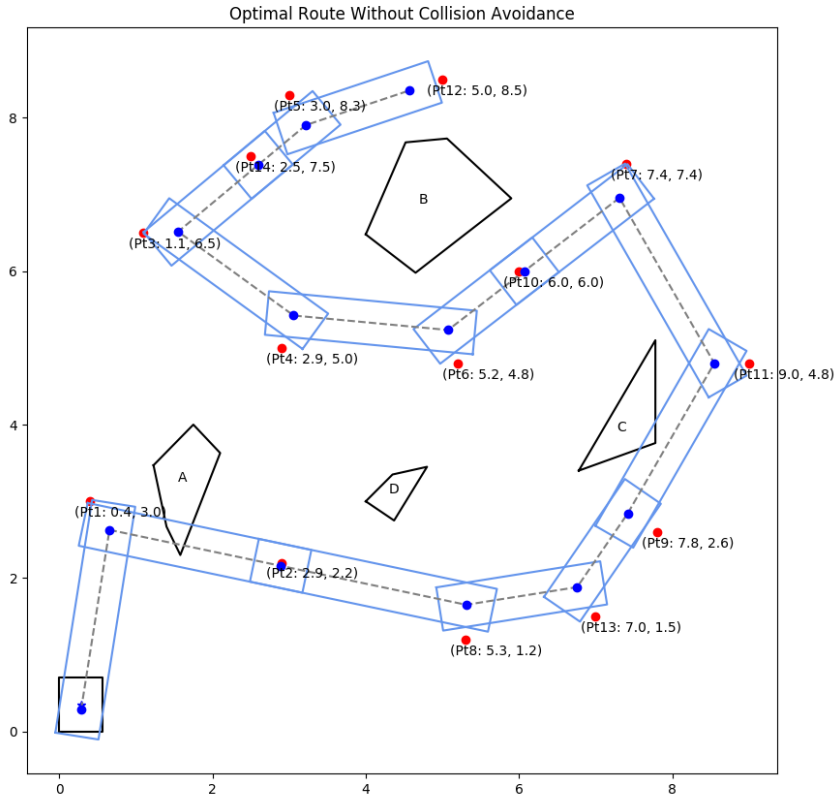

Figure 1.3: Optimal Routes Derived From Convex Programming

1.3.3 Collision Avoidance: Adding Constraints

Given the visiting order, the result in Section 1.3.2 serves as a lower bound. As shown in Figure 1.3, the cleaning vehicle will collide with polygon A and C. Adjustments should be made to avoid collisions with A and C. Note that no collision with polygon A and C is only a necessary condition for a feasible path. We still take a sequential approach here and check feasibility at the end.

First, we take a closer look at polygon A (See Figure 1.4). Of course, we can solve the two locations with mathematical programming, with additional constraints, which describe the relationship between polygon A and the rectangular region of this segment of trajectory. While we can check whether two convex sets intersect through the feasibility of a convex programming, it is difficult to express this 'no intersection' relationship as constraints. One possible way is to add binary variables to express 'either-or' and 'if-then' conditions, which exacerbates the complexity of the problem. Instead, we consider one of its necessary condition, which can be easily expressed as a linear constraint.

To cover rubbish 1 and 2, the center of the cleaning vehicle must pass some points in

the two orange circles. Therefore, the x coordinate on its trajectory must vary from, say, 1.23 to 2.1, continuously. A necessary condition for no collision with polygon A is when the x coordinate on the trajectory is equal to 1.58 (the lowest point), the vehicle should not collide with the vertex $(1.58, 2.3)$. A sufficient condition for this necessary condition is the corresponding y is less than $2.3 - r$, where r denotes the radius of the circumcirle (See the arrow in Figure 1.4).

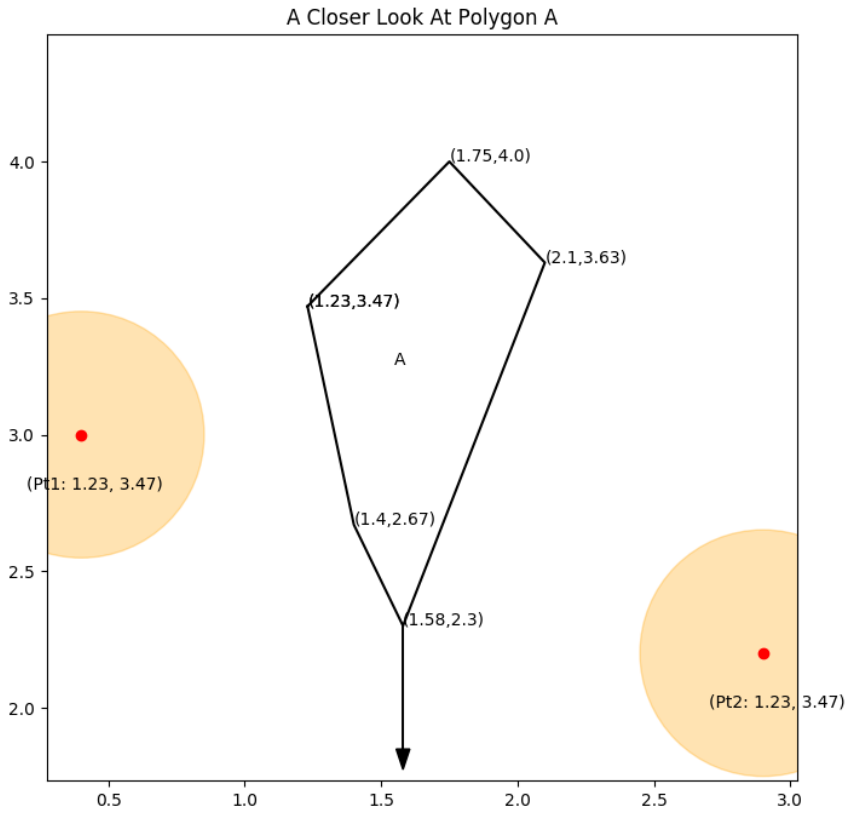

Figure 1.4: A Closer Look At Polygon A

Similarly, for polygon C, we sample the trajectory at $x = 7.78$, and require $y \leq 3.76 - r$. Now, the mathematical programming is:

$$
\begin{aligned}
\min \quad & \sum_{i=0}^{15} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \\
\text{s.t.} \quad & (x_1 - a_1)^2 + (y_1 - b_1)^2 \leq \frac{1}{4}(w^2 + h^2) \\
& (x_i - a_{i-1})^2 + (y_i - b_{i-1})^2 \leq \frac{1}{4}(w^2 + h^2) && i = 3, \ldots, 6 \\
& (x_i - a_{i-2})^2 + (y_i - b_{i-2})^2 \leq \frac{1}{4}(w^2 + h^2) && i = 8, \ldots, 16 \\
& x_2 = 1.58 \\
& x_7 = 7.78 \\
& y_2 \leq 2.3 - r \\
& x_7 \leq 7.78 - r
\end{aligned}
$$

Again, it is a convex programming, the derived trajectory is shown in Figure 1.5. The path length is 24.9276. As highlighted by the purple circles, there is still some space left between the trajectory and polygon A while it intersects with polygon C. So the constraint $y_2 \leq 2.3 - r$ can be tightened while $x_7 \leq 7.78 - r$ should be relaxed. Here we apply the idea of **grid search** to adjust the constraints:

$$
\begin{aligned}
\min \quad & \sum_{i=0}^{15} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \\
\text{s.t.} \quad & (x_1 - a_1)^2 + (y_1 - b_1)^2 \leq \frac{1}{4}(w^2 + h^2) \\
& (x_i - a_{i-1})^2 + (y_i - b_{i-1})^2 \leq \frac{1}{4}(w^2 + h^2) && i = 3, \ldots, 6 \\
& (x_i - a_{i-2})^2 + (y_i - b_{i-2})^2 \leq \frac{1}{4}(w^2 + h^2) && i = 8, \ldots, 16 \\
& x_2 = 1.58 \\
& x_7 = 7.78 \\
& y_2 \leq 2.3 - r + d_1 \\
& x_7 \leq 7.78 - r - d_2
\end{aligned}
$$

Note that $d_1, d2 \geq 0$. We start by $d_1 = d_2 = 0$ and increase d_1 with a stepsize of 0.01 while keeping $d_2 = 0$ until the trajectory does not intersect with polygon. The we increase d_2 with a stepsize of 0.01 while keeping d_1 constant. The intuition behind this method is explained as follows:

- **Why adjusting the constraints in this way will improve the solution?** For polygon C, this adjustment pulls the solution to the feasible region. For polygon A, to go from the neighborhood of rubbish 1 to that of rubbish 2, the vehicle generally moves downwards to circumvent the vertex $(1.58, 2.3)$. So intuitively, when we shift the trajectory upward, the path length will be shortened. We will check this claim in our grid search experiment.

- **Will adjusting d_2 make the vehicle collide with polygon A?** For general mathematical problem the answer is uncertain. But for this specific problem, an intuitive explanation is that increasing d_1 shifts then trajectory upwards and increasing d_2 pulls it downward. It is only possible that the vehicle will be pulled away from polygon A when d_2 is increased. (See the purple arrows in 1.5). We will check the whether the vehicle collides with polygon A in every iteration when adjusting d_2.

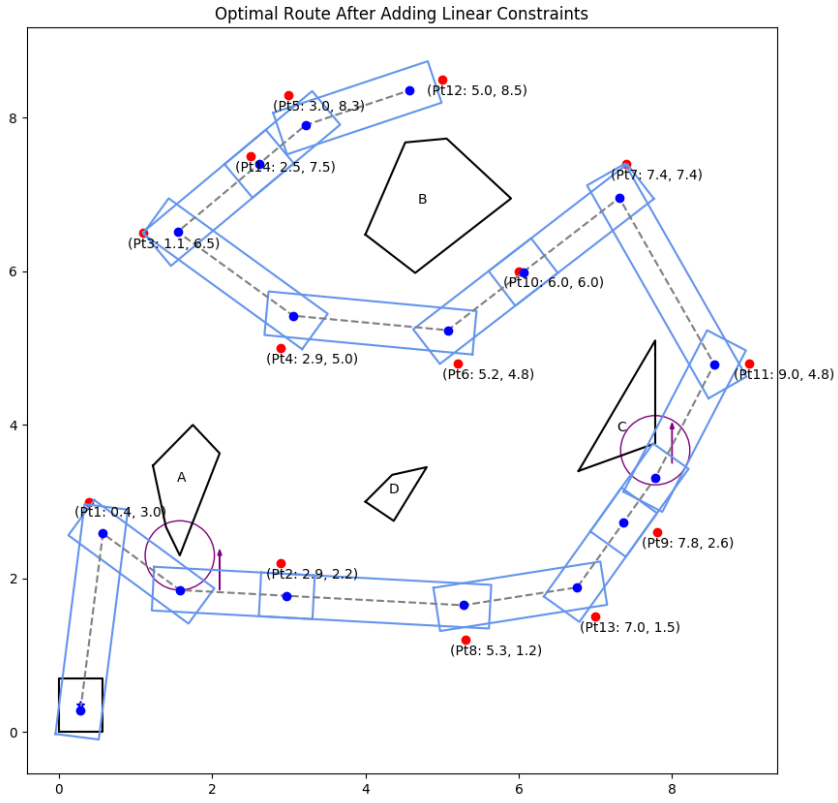

Figure 1.5: Optimal Routes After Adding Linear Constraints

The trajectory obtained from grid search is shown in Figure 1.6, with a path length of 24.9012 and $d_1 = 0.11, d_2 = 0.25$. Fortunately, this solution is feasible and, the trajectory does not intersect with polygon A when we increase d_2 alone, as expected.

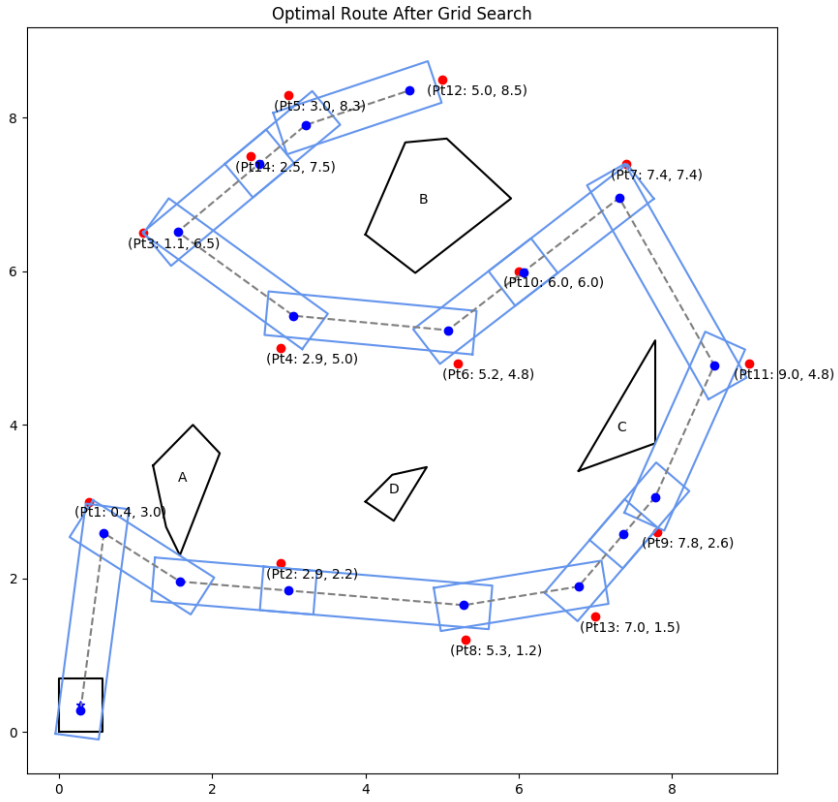

Figure 1.6: Optimal Routes After Grid Search

Figure 1.7 and 1.8 depict the relationship between the optimal objective value and distance constraints. The optimal objective value decreases monotonically as the center gets closer to the vertex. So we have exactly find the largest d_1 and smallest d_2 to make our trajectory as close to the vertex as possible.

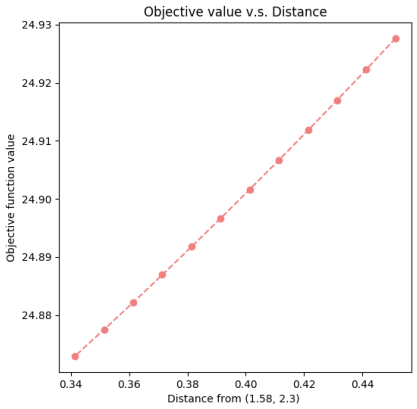

Figure 1.7: Optimal value v.s. $-d_1 + r$

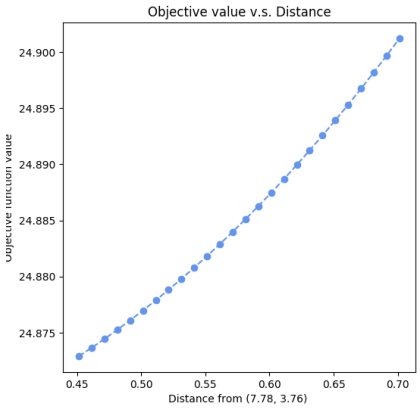

Figure 1.8: Optimal Value v.s. $d_2 + r$

Table summarizes how the solution evolves as we try to solve the problem step by

step.

Table 1.1: The Evolution of Solutions

Stage	Feasibility	Path length
TSP	×	29.27
Convex Programming	×	24.75
Add Constraints	×	24.92
Grid Search	✓	24.90

1.3.4 Robot Steering and Sampling Points Selection

To find the optimal steering method is somewhat off-topic to the question of optimal path of length, since 100 sample points are sufficient for the robot to finish the steering task. However, as a part of the problem, it's essential for us to find a feasible and satisfactory way of steering, which is not the simplest but easier-for-coding.

Robot Steering Angles

There are two types of steering for us to consider, one of which is in cleaning rubbish scenario, and the other in turning-for-next-waypoint scenario.

When the robot stops in the target point for cleaning the rubbish, first we need to specify whether the rubbish has already been cleaned by the robot, which is equal to whether the rubbish is already inside the square of the robot. Then we find the angle of steering, which is less than π, achieved by *shaking head* or *twisting ass*. After cleaning the rubbish, we force it back to the previous state, and finally the robot steers to the next direction pointed to the next target point. It's somewhat easy for coding if we force it back, although it may be easier for the robot to only steer a small angle back or continue to steer a small angle. The details of the steering process are shown in the table below. We set the machine epsilon as 1e-3 (radian).

Table 1.2: Angle and Times of Steering

Description	Angle (radian)	Times of steering
From start to the first target point	0.129	1
Cleaning Rubbish #1	0.114	1
Move to next direction	2.118	4
Transfer rotate	1.212	2
Cleaning Rubbish #2	0.353	1
Move to next direction	0.353	1
Cleaning Rubbish #8	0.765	2
Move to next direction	1.009	2
Cleaning Rubbish #13	0.537	1
Move to next direction	1.238	2
Cleaning Rubbish #9	0.094	1
Move to next direction	0.094	1
Transfer rotate	1.358	3
Cleaning Rubbish #11	0.419	1
Move to next direction	1.356	3
Cleaning Rubbish #7	0.032	1
Move to next direction	1.743	3
Cleaning Rubbish #10	0	0
Move to next direction	0	0
Cleaning Rubbish #6	0.512	1
Move to next direction	1.262	3
Cleaning Rubbish #4	0.620	1
Move to next direction	1.154	2
Cleaning Rubbish #3	0.022	1
Move to next direction	1.842	3
Cleaning Rubbish #14	0	0
Move to next direction	0	0
Cleaning Rubbish #5	0.701	2
Move to next direction	1.074	2
Cleaning Rubbish #12	0.683	2
Total	-	47

Some Explanations of Robot Steering

We take rubbish #8 for example. When the robot stopping in the target point, it firstly steers 0.765 radian clockwise (2 sample points, less than 0.2π for one new sample point), then steers back and to the next target point 1.009 radian counterclockwise (2 sample points), and finally goes ahead to the next target point. Below is an illustration.

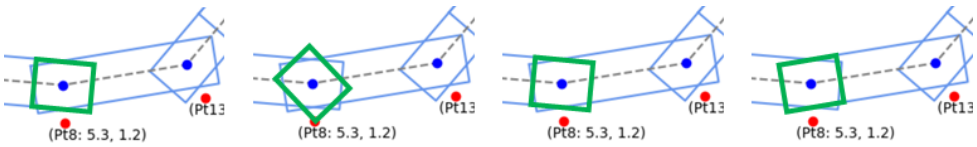

Figure 1.9: Illustration of Cleaning Rubbish #8

Total Sampling Points

As shown in the table above, the total sampling point for steering is 47, and the critical target points number is 16. So the **total sampling points of our solution is 63**. For more details, please refer to the Appendix. Actually, we only give an upper bound of total sampling points. Sometimes the robot doesn't have to stop to clean and rotate back, which means the robot has already cleaned the rubbish when it steers to the direction of next target point. As the number of sampling points is not a tight constraint, we only give a feasible and satisfactory solution way of steering and sampling as stated before.

1.4 Discussion and Conclusion

In summary, our idea is to first obtain the visiting order of garbage points by solving the traveling salesman problem, then obtain the solution under barrier-free conditions by solving the following mathematical programming problems, and finally make subtle route adjustments based on obstacles. It is worth noting that the optimal access sequence obtained by the traveling salesman problem does not guarantee that the corresponding mathematical programming problem is optimal.But the following inequality holds

$$f_{tsp} - n \times \frac{1}{2}\sqrt{w^2 + h^2} \leq f_{math} \leq f_{tsp} + n \times \frac{1}{2}\sqrt{w^2 + h^2}$$

f_{tsp} is the route length corresponding to a feasible solution to the traveling salesman problem, and f_{math} is the route length corresponding to the mathematical programming problem. We can get the above inequality by taking two straight lines between the adjacent access points. Obviously, this inequality is not tight. According to the rotation angle, a tighter bound should be obtained. We will not discuss it because of the complexity and time. According to the above calculation, we get that $n \times \frac{1}{2}\sqrt{w^2 + h^2} = 6.31903$. Therefore, the solution to the traveling salesman problem that is greater than or equal to 31.22 must not be as good as our existing solution. On this basis, we verified that **the above solution 24.90 is the best solution for this problem.**

Appendix A

Sample Points

No.	x	y	Angle (radian)	Angle (degree)	Description
0	0.350	0.285	1.571	90.0	Start Point
1	0.350	0.285			Steering to Rubbish #1
2	0.590	2.590	1.435	82.3	Near Rubbish #1
3	0.590	2.590			Cleaning Rubbish #1
4	0.590	2.590			Move to Next Direction - 1
5	0.590	2.590			Move to Next Direction - 2
6	0.590	2.590			Move to Next Direction - 3
7	0.590	2.590			Move to Next Direction - 4
8	1.580	1.959	5.715	327.5	Near Transfer Point
9	1.580	1.959			Transfer Point Rotate - 1
10	1.580	1.959			Transfer Point Rotate - 2
11	2.995	1.841	6.200	355.3	Near Rubbish #2
12	2.995	1.841			Cleaning Rubbish #2
13	2.995	1.841			Move to Next Direction
14	5.282	1.651	6.200	355.3	Near Rubbish #8
15	5.282	1.651			Cleaning Rubbish #8 - 1
16	5.282	1.651			Cleaning Rubbish #8 - 2
17	5.282	1.651			Move to Next Direction -1
18	5.282	1.651			Move to Next Direction -2
19	6.779	1.894	0.161	9.2	Near Rubbish #13
20	6.779	1.894			Cleaning Rubbish #13
21	6.779	1.894			Move to Next Direction -1
22	6.779	1.894			Move to Next Direction -2
23	7.369	2.580	0.861	49.3	Near Rubbish #9
24	7.369	2.580			Cleaning Rubbish #9
25	7.369	2.580			Move to Next Direction

No.	x	y	Angle (radian)	Angle (degree)	Description
26	7.780	3.059	0.861	49.3	Near Transfer Point
27	7.780	3.059			Transfer Point Rotate - 1
28	7.780	3.059			Transfer Point Rotate - 2
29	7.780	3.059			Transfer Point Rotate - 3
30	8.549	4.778	1.150	65.9	Near Rubbish #11
31	8.549	4.778			Cleaning Rubbish #11
32	8.549	4.778			Move to Next Direction -1
33	8.549	4.778			Move to Next Direction -2
34	8.549	4.778			Move to Next Direction -3
35	7.311	6.958	2.088	119.6	Near Rubbish #7
36	7.311	6.958			Cleaning Rubbish #7
37	7.311	6.958			Move to Next Direction -1
38	7.311	6.958			Move to Next Direction -2
39	7.311	6.958			Move to Next Direction -3
40	6.057	5.991	3.798	217.6	Near Rubbish #10
41	5.075	5.234	3.798	217.6	Near Rubbish #6
42	5.075	5.234			Cleaning Rubbish #6
43	5.075	5.234			Move to Next Direction -1
44	5.075	5.234			Move to Next Direction -2
45	5.075	5.234			Move to Next Direction -3
46	3.059	5.422	3.048	174.7	Near Rubbish #4
47	3.059	5.422			Cleaning Rubbish #4
48	3.059	5.422			Move to Next Direction -1
49	3.059	5.422			Move to Next Direction -2
50	1.551	6.515	2.514	144.1	Near Rubbish #3
51	1.551	6.515			Cleaning Rubbish #3
52	1.551	6.515			Move to Next Direction -1
53	1.551	6.515			Move to Next Direction -2
54	1.551	6.515			Move to Next Direction -3
55	2.618	7.405	0.695	39.8	Near Rubbish #14
56	3.220	7.906	0.695	39.8	Near Rubbish #5
57	3.220	7.906			Cleaning Rubbish #5 -1
58	3.220	7.906			Cleaning Rubbish #5 -2
59	3.220	7.906			Move to Next Direction -1
60	3.220	7.906			Move to Next Direction -2
61	4.572	8.357	0.322	18.5	Near Rubbish #12
62	4.572	8.357			Cleaning Rubbish #12 -1
63	4.572	8.357			Cleaning Rubbish #12 -2