Applications of Convex Optimisation in Robotic Motion Planning

Convex Optimization project

Athira Krishnan R, Tadipatri Uday Kiran Reddy

AI22RESCH01001, EE19BTECH11038
Group 22

Motivation

- With the recent boom in Autonomous Navigation, we would want our robot to plan and navigate safely and optimally.
- Convex optimization techniques have the capability to find a feasible solution.
- Posing motion planning problem as convex optimisation helps us to exploit the speed with which cvx solvers can find the solution.

Introduction

- Convex optimization is a domain that can be applied to get feasible solutions for the robot in its motion planning for a faster traversal, obstacle avoidance and many more.
- Most SOTA like D*, A*, TEB are have proved to find a feasible path.
- One of the widely used planner using Artificial Potential Field, has problem of getting stuck at local minima.
- Real time planning and velocity optimization needs more careful modelling.
- Some works [2] optimize and as also plan the path if any obstacles are found in the way.

Problem Statements

We try to address three classical problems in the field of robot motion planning using convex optimization techniques.

1. To generate a collision free path that a robot can traverse from a start point/ current robot position to a target point.[3]
2. To generate a optimized smooth trajectory that the robot can further use while actually tracking the path.[3]
3. To find control inputs, to traverse the optimized path in minimum time considering the robot dynamics.[1]

Assumptions

Assumptions Made:

1. Obstacle geometry is polygonal and is known in prior.
2. Robot dimensions and dynamics are known.
3. Environment is static.
4. Robot configuration is car-like model, with radius=2m, mass=1kg, gravity g=9.8m/s.

All implementations were done using *CVXPY* library in python.
https://github.com/Athirakr94/robot-nav-convex

Path Generation

- Path generation is an iterative process of finding set of feasible waypoints for a robot to traverse from a known start to destination point without any collision.
- In the environment the points are sampled adaptive to the remaining distance to goal.
- Let C(x1,y1),..,(xn,yn) represents the static obstacle region known.
- x,y coordinates are taken as

$$x = r * cos(\theta) \tag{1}$$

$$y = r * sin(\theta) \tag{2}$$

$$theta = arctan((y_g - y_r)/(x_g - x_r)) \tag{3}$$

where (x_g, y_g) is goal and (x_r, y_r) is the current robot position.

$$distancetogoal = ||(x_g, y_g), (x_r, y_r)||_2 \tag{4}$$

Contd..

$$r = \begin{cases} threshold, if distance > threshold \\ distance, if distance <= threshold \end{cases} \tag{5}$$

where threshold=5m.

- Any point selected as next waypoint must be such that they do not lie within the obstacle region.
- To ensure the same, a convex hull is formed around the obstacle as reference as shown in Figure 1.

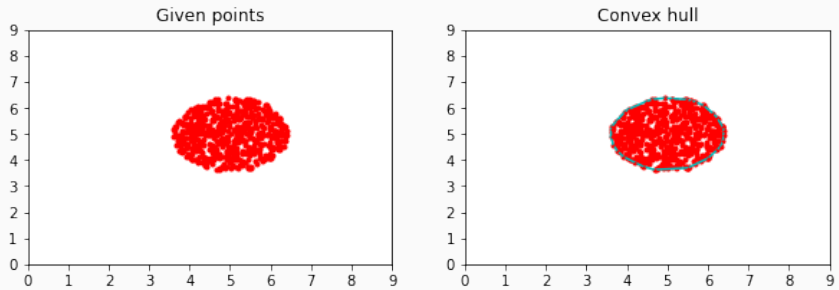


Figure 1: Known obstacle and hull formed around it.

Path generation algorithm at a glance

Algorithm 1 Path generation with static obstacle

procedure PLANPATH($pose, target, obstacles$)
  $tolerace = 1$ **while** $||target - start||_2 > tolerace$ **do**
  end
  $point = Getpoint$
  $waypoint.append(point)$

procedure GETPOINT($pose, target, obstacles$)
  threshold=5
  start=pose
  $hull = ConvexHull(obstaclepoints)$
  if $||target - start||_2 > threshold$ **then**
  end
  $r = threshold$
  $point = start + (random * r)$) **else**
  end
  $r = ||target - start||_2$
  $point = target$

  $hullnew = ConvexHull(obstaclepoints, point)$
  while $(hullnew == hull)$ **do**
  end
  $point = start + (random * r)$)
  $hullnew = ConvexHull(obstaclepoints, point)$

  return point

Path Generated

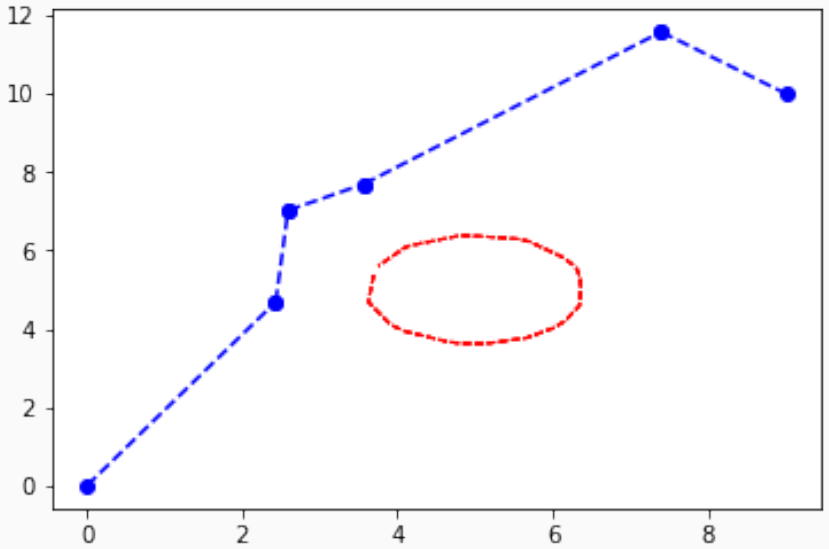


Figure 2: Path generated from [0,0] to [9,10] in known static environment

Path Optimization/ Smoothing

To smoothen a generated path, we need to know the obstacle region as if it is not considered, possibility of collision is high.

Smooth path=bubble formation+path optimization

Bubble formation

- Here we try to find the feasible radius, r with which we could construct a circle around each way point.
- An optimal r would lie between rmin (radius of vehicle=2m) and rmax(=7m).
- This problem can be framed as a bisection problem as shown below,

$$\begin{aligned} &\text{minimize} && 0 \\ &\text{subject to} && Ai \geq 0, ||P - Onear||_2 \geq 2r \end{aligned} \tag{6}$$

where Onear is the nearest obstacle point to the current point in waypoint and Ai is the new center to replace Pi ϵ**P** such that distance to Onear is atleast rmin.

Bubble Formation Contd..

- We iterate untill,

$$rmax - rmin <= 0.1 \tag{7}$$

rmax(=7m) rmin(=2m) and r=0.5*(rmin+rmax)

- During iteration if an optimal solution is attained, rmin and rmax are updated based on status of the problem as ,

$$update - rule = \begin{cases} center = Pi \quad rmin = r, if \\ optimal \\ center = Pi \quad rmax = r, otherwise \end{cases} \tag{8}$$

Note: If rmin<r, rmin=r and Ai is new shifted point instead of Pi and Ai will now act as center unlike other scenario where Pi=Ai.

Bubble formation Algorithm

Through each waypoint bubble formation is performed and translated to next.

Algorithm 2 Bubble generation

procedure PLANPATH(*point*, *nextpoint*)
rmin = 2
rmin = 5
tolerance = 1
r=0.5(rmax+rmin) **while** *rmax* − *rmin* > *tolerace* **do**
end
r = 0.5(*rmax* + *rmin*)
solveconvexproblem **if** *optimal* **then**
end
center = *Pi*
$r_m in$ = *r* **if** *r* **then**
end
center = *Ai*
r = *rmin*
else
end
center = *Pi*
$r_m ax$ = *r*

Bubble Generation Result

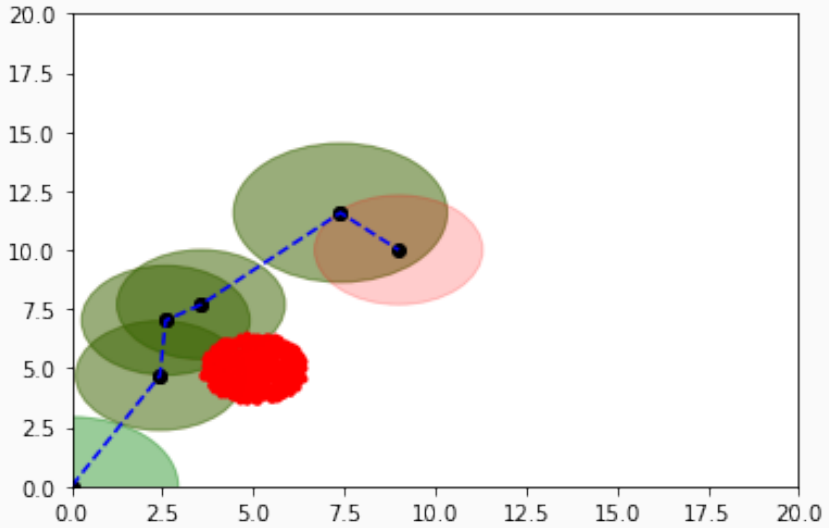


Figure 3: Bubble generated at each waypoint from generated path

Path Optimization

Let Q[k] and Q[k+1] be points in bubbles k and k + 1, with k = 1, . . . , n − 1. We consider an artificial tensile force F[k] between Q[k] and Q[k+1] given by,

$$F[k] = Q[k+1] - Q[k] \tag{9}$$

Balancing force at point Q[k] , k = 1, . . . , n is given by,

$$N[k] = F[k+1] - F[k] \tag{10}$$

Our objective is to find a set of optimized waypoints Q that balances the forces between 2 waypoint and return a shorter smooth trajectory for the robot to follow. Initially we assume v, α and d as random values they could be updated and taken in recursion when considering a real time dynamic environment.

Contd..

Considering the elastic stretching forces, vehicle dynamics (velocity and turning radius), the path optimization problem is modelled as follows,

$$
\begin{aligned}
\text{minimize} \quad & ||2 * Q[k] - Q[k-1] - Q[k+1]||^2 \\
\text{subject to} \quad & Q[1] = P[1], \\
& Q[n] = P[n] \\
& Q[2] = P[1] + d.(v[1]/||v[1]||) \\
& Q[n-1] = P[n] - d.(v[n-1]/||v[n-1]||) \\
& ||2Q[k] - Q[k-1] - Q[k+1]|| <= \\
& max((d^2/Rmin), \alpha[k].(d/v[k])^2)
\end{aligned}
\tag{11}
$$

where k is the iteration index and n is the number of way points in the path. Objective function being quadratic form, problem is formulated as *QCQP*.

Path Smoothing/Optimization Result

- Through path optimization we could reduce path length by **26.7%**.
- Length of the generated path =14.255m. Length of optimized path =10.436m.
- Optimal cumulative balancing cost/force measured as 5.36N.

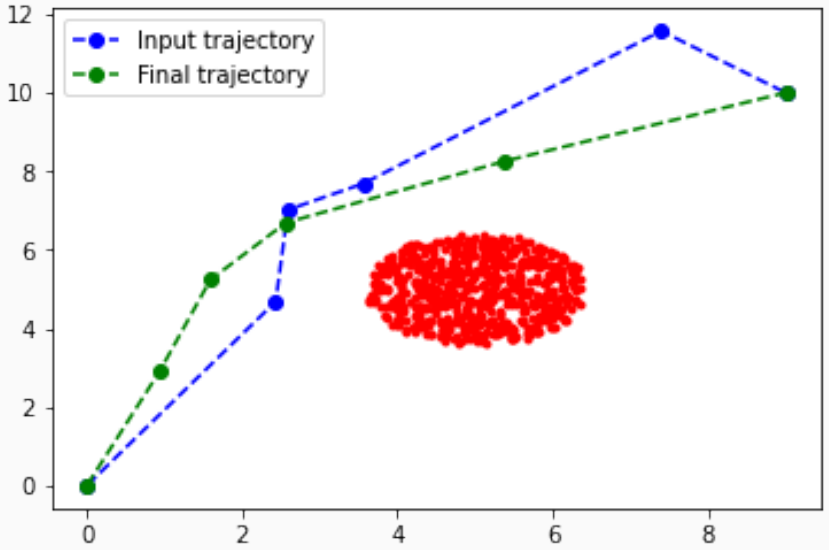


Figure 4: Smooth path from generated path through the collision free bubble regions identified

Velocity Optimization

1. Objective here is to minimize total time taken to traverse the whole path, T.
2. Control inputs here are forces in all possible degree of freedom.
3. Constraints here are,
   - 3.1 Newton laws of motion
   - 3.2 Initial conditions
   - 3.3 Velocity and acceleration limits.

Velocity Optimization

p degrees of motion and r actuators.

Dynamics

Second-order dynamics,

$$R(q)u = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + d(q) \tag{12}$$

$q \in \mathcal{R}^p$ is position vector, $u \in \mathcal{R}^r$ is control inputs,i.e, force, R is the control matrix, M is mass matrix, C is centrifugal matrix and d is force which is dependent on position vector. In our setup,

$$R = \begin{bmatrix} 0.403 & -0.9153 \\ 0.9153 & 0.403 \end{bmatrix}; M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Path interpretation

$$s(\theta(t)) = q(t), t \in [0, T] \tag{13}$$

$\theta : [0, T] \to [0, 1]$, $s : [0, 1] \to \mathcal{R}^p$

Pose the optimisation problem

$$
\begin{aligned}
\min \quad & T \\
\text{s.t.} \quad & R(q(t))u(t) = M(q(t))\ddot{q(t)} + C(q(t))\dot{q(t)}^2 + d(q(t)) \\
& s(\theta(t)) = q(t) \\
& (\dot{q(t)}^2, \ddot{q(t)}, u(t)) \in \mathcal{C}(q(t)) \\
& t \in [0, T]
\end{aligned} \tag{14}
$$

Let's transform few variables, let $b(\theta) = \dot{\theta}^2$ and $a(\theta) = \ddot{\theta}$.

$$
\begin{aligned}
\min \quad & \int_0^1 b(\theta)^{-1/2} d\theta \\
\text{s.t.} \quad & \tilde{R}(\theta)u(\theta) = \tilde{m}(\theta) + \tilde{c}(\theta)b(\theta) + \tilde{d}(\theta) \\
& b^{'}(\theta) = 2a(\theta), \theta \in [0, 1] \\
& (a(\theta)), b(\theta), u(\theta)) \in \tilde{\mathcal{C}}(s(\theta))
\end{aligned} \tag{15}
$$

Discretize the optimisation problem

$$
\begin{aligned}
\min \quad & 2\sum_{i=0}^{N-1}\left(\frac{\theta_{i+1}-\theta_i}{{b_{i+1}}^{1/2}+{b_i}^{1/2}}\right) \\
\text{s.t.} \quad & \tilde{R}(\overline{\theta_i})u_i = \tilde{m}(\overline{\theta_i}) + \tilde{c}(\overline{\theta_i})\frac{b_i+b_{i+1}}{2} + \tilde{d}(\overline{\theta_i}) \qquad (16) \\
& b_i - b_{i-1} = 2a_i d\theta \\
& (a(\theta)), b(\theta), u(\theta)) \in \tilde{\mathcal{C}}(s(\theta))
\end{aligned}
$$

Is this a problem a convex optimisation problem?

Objective: Apply epigraph trick,

$$
g_i \geq \frac{2d\theta}{b_{i+1}^{1/2}+b_i^{1/2}} \implies \frac{2d\theta}{g_i} - b_{i+1}^{1/2} - b_i^{1/2} \leq 0 \implies \text{Convex}
$$

Constraints: Affine functions which are convex.

Implementation challenges

The Objective function is does not comply *DCP rules* .

$$2\sum_{i=0}^{N-1}\left(\frac{\theta_{i+1}-\theta_i}{{b_{i+1}}^{1/2}+{b_i}^{1/2}}\right)$$

Let's check the epigraph form of optimisation problem,

$$\begin{aligned}
\textbf{min} \quad & \sum_{i=0}^{N-1} g_i \\
\text{s.t.} \quad & \frac{2d\theta}{g_i} - b_{i+1}^{1/2} - b_i^{1/2} \leq 0 \\
& \tilde{R}(\overline{\theta_i})u_i = \tilde{m}(\overline{\theta_i}) + \tilde{c}(\overline{\theta_i})\frac{b_i + b_{i+1}}{2} + \tilde{d}(\overline{\theta_i}) \\
& b_i - b_{i-1} = 2a_i d\theta \\
& (a(\theta)), b(\theta), u(\theta)) \in \tilde{\mathcal{C}}(s(\theta))
\end{aligned} \tag{17}$$

Satisfies DCP rules!

Which class of cvx problem is it?

It is Second-Order Cone Programming (SOCP) problem.

Velocity Optimization Results

- From velocity optimizer, path traversal time is **9.56s**.

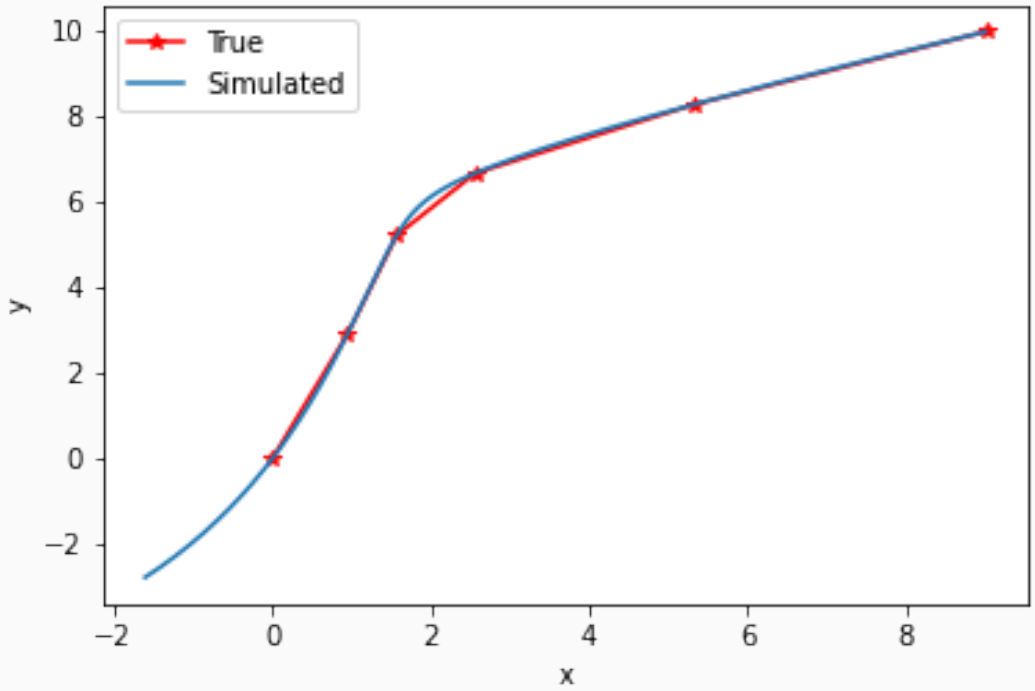
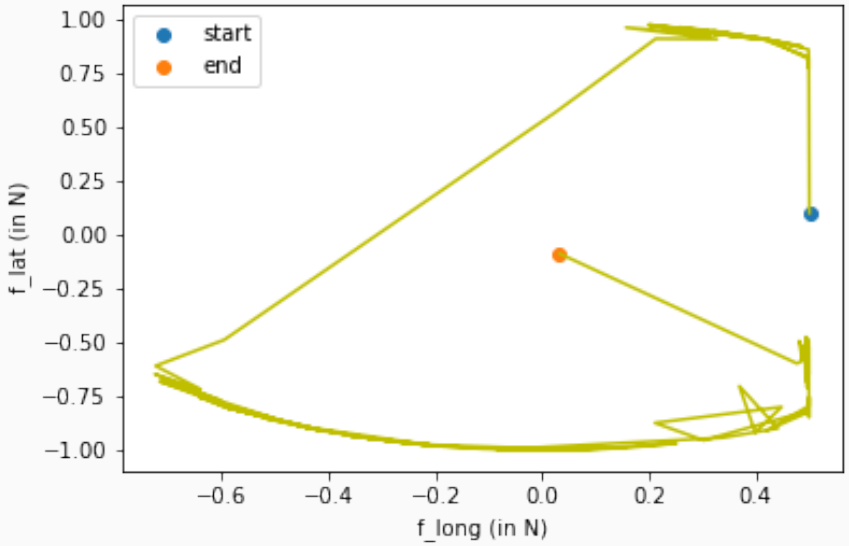
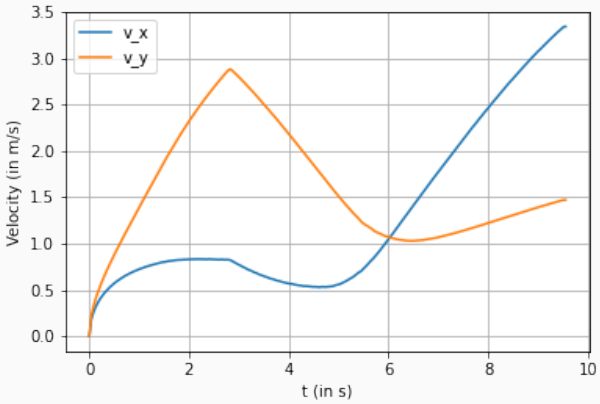


Figure 5: Input path and tracked path through velocity optimization

Contd..





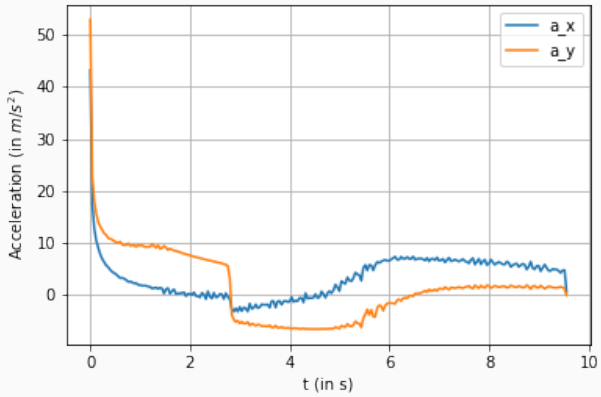


Figure 6: Force, Velocity and Acceleration plots

Conclusions

- We posed path planning with static obstacles as a optimisation problem, and classified the problem as convex optimisation problem. Later we went ahead and solved this using *CVXPY*.
- We solved 3 sub problems but this is sub-optimal, we could instead pose one problem and check it's feasibility but convexity of this problem may not be retained.
- In future we could try extending the work to dynamic environment,i.e, even obstacles are in motion.

References


T. Lipp and S. Boyd.
Minimum-time speed optimisation over a fixed path.
International Journal of Control, 87(6):1297–1311, 2014.


J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel.
Motion planning with sequential convex optimization and convex collision checking.
The International Journal of Robotics Research, 33(9):1251–1270, 2014.


Z. Zhu, E. Schmerling, and M. Pavone.
A convex optimization approach to smooth trajectories for motion planning with car-like robots.
In *2015 54th IEEE conference on decision and control (CDC)*, pages 835–842. IEEE, 2015.