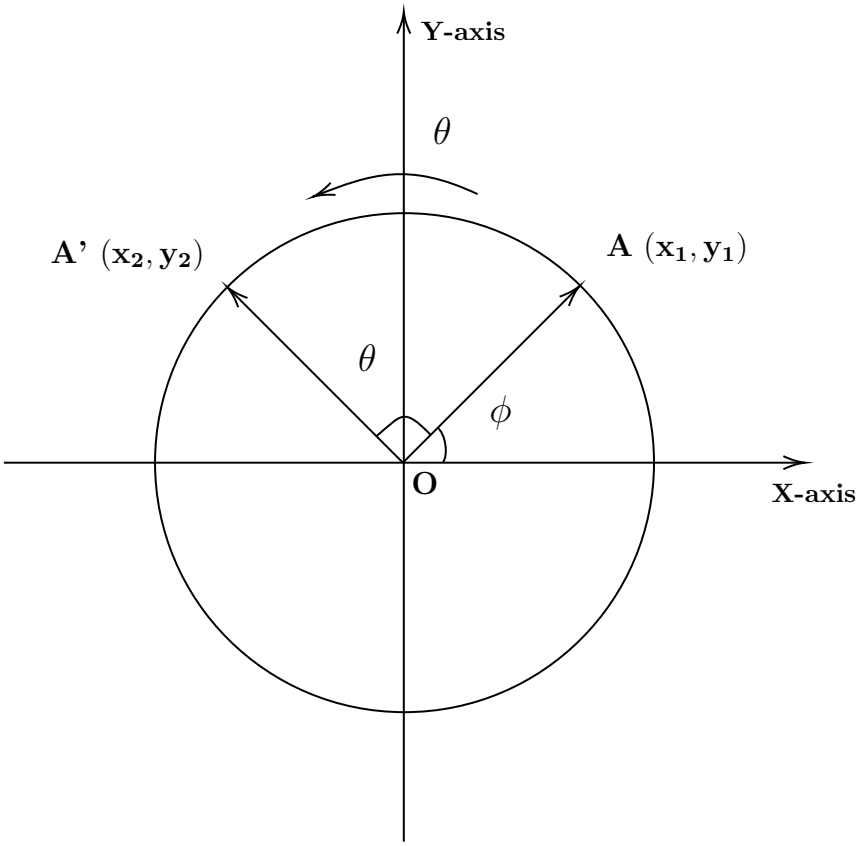
RBE550: Motion Planning

Project 1: Fundamentals

Student 1: Jeel, Chatrola

1. Given a unit disc $\mathcal{A}$ centered at origin in workspace $\mathcal{W} = \mathbb{R}^2$.
   $\mathcal{A}$ represented by single algebraic primitive H = $\{(x, y) \mid x^2 + y^2 \leq 1\}$.



Consider a point A (x_1, y_1) in the primitive H $\implies$ H = $\{(x_1, y_1) \mid x_1^2 + y_1^2 \leq 1\}$

We rotate the primitive H (circle) anticlockwise around the origin θ so the point A becomes a new point A' (x_2, y_2) in rotate primitive H' (circle - as rotation doesn't change the shape).

We have to prove that primitive H remains unchanged after rotation. Now, finding new rotated point (x', y') using Rotation Matrix R.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \implies x' = x_1\cos\theta - y_1\sin\theta,\ y' = y_1\cos\theta + x_1\sin\theta$$

Comparing it with the new primitive H'.
$(x_2, y_2) = (x', y') \implies (x_2, y_2) = (x_1\cos\theta - y_1\sin\theta, y_1\cos\theta + x_1\sin\theta)$

To prove this new point is inside the new primitive H', we substitute the point in primitive H to prove that it lies inside.

$H = \{(x_2, y_2) \mid x_2^2 + y_2^2 \leq 1\}$

$H_{(x_2,y_2)} = \{(x_1\cos\theta - y_1\sin\theta)^2 + (y_1\cos\theta + x_1\sin\theta)^2 \leq 1\}$ ($\because$ Using the rotation matrix above)

$H_{(x_2,y_2)} = \{(x_1\cos\theta)^2 - 2(x_1\cos\theta)(y_1\sin\theta) + (y_1\sin\theta)^2 + (y_1\cos\theta) + 2(y_1\cos\theta)(x_1\sin\theta) + (x_1\sin\theta)^2 \leq 1\}$

$H_{(x_2,y_2)} = \{(x_1\cos\theta)^2 + (y_1\sin\theta)^2 + (y_1\cos\theta)^2 + (x_1\sin\theta)^2 \leq 1\}$

$H_{(x_2,y_2)} = \{x_1^2\cos\theta^2 + y_1^2\sin\theta^2 + y_1^2\cos\theta^2 + x_1^2\sin\theta^2 \leq 1\}$

$H_{(x_2,y_2)} = \{x_1^2\cos\theta^2 + x_1^2\sin\theta^2 + y_1^2\sin\theta^2 + y_1^2\cos\theta^2 \leq 1\}$

$H_{(x_2,y_2)} = \{x_1^2(\cos\theta^2 + \sin\theta^2) + y_1^2(\sin\theta^2 + \cos\theta^2) \leq 1\} \qquad (\because \sin\theta^2 + \cos\theta^2 = 1)$

$H_{(x_2,y_2)} = \{x_1^2 + y_1^2 \leq 1\}$

$H_{(x_2,y_2)} = H_{(x_1,y_1)}$

This proves that our transformed point (x_2, y_2) lies in the primitive H. Additionally, the length of vector A and A' remains unchanged $\left(\because \sqrt{x_2^2 + y_2^2} = \sqrt{(x_1\cos\theta - y_1\sin\theta)^2 + (y_1\cos\theta + x_1\sin\theta)^2} \implies L_1 = L_2\right)$

Our point (x_2, y_2) was inside primitive H' which proves that transformation/rotation of primitive H remains unchanged

---

2. Pseudo code to calculate intersection point given two line segments A_1B_1 and A_2B_2 in a 2D Workspace:

Notations:

- L_i is Line Segment i.
- $\vec{L_i}$ a vector for end points A_iB_i.
- m_i is slope of line segment i.
- $\vec{L_i}^{\perp}$ denotes vector perpendicular to $\vec{L_i}$.
- $A_i\vec{A}_j$ denotes a vector from points A_i and A_j.
- P_i - parameters.

Algorithm 1 Algorithm for Line Segment Intersection

function SEGMENT_PAIR_INTERSECTION(L_1, L_2)
  $\vec{L_1} = (x_2 - x_1, y_2 - y_1)$ ▷ Create a Vector from given endpoints
  $\vec{L_2} = (x_3 - x_4, y_3 - y_4)$

  if $(x_2 - x_1) \neq 0$ or $(x_3 - x_4) \neq 0$ **then** ▷ Eliminate points if ∞ slope on both segments

    if $(\vec{L_1})^{\perp} * (\vec{L_2})$!= 0 and $(\vec{L_2})^{\perp} * (\vec{L_1})$!= 0 **then** ▷ Eliminate out parallel/coinciding lines

      $P_1 = \left(\dfrac{\vec{L_2}^{\perp} * (A_2\vec{A}_1)}{\vec{L_2}^{\perp} * (\vec{L_1})}\right)$, $P_2 = \left(\dfrac{\vec{L_1}^{\perp} * (A_1\vec{A}_2)}{\vec{L_1}^{\perp} * (\vec{L_2})}\right)$

      if $P_1 \in [0, 1]$ and $P_2 \in [0, 1]$ **then** ▷ Return the point if it lies on both line segments

        Point = $\left[A_1 + P_1 * (A_1 \vec{-} A_2)\right]$

        return Point
      else
        return None
    else
      return None
  else
    return None

Line Segment Representation used:
$F_1(P_1) = A_1 + P_1 * (\vec{L_1})$ where, $(\vec{L_1}) = A_1 - B_1$ and $P_1 \in [0, 1]$ represents Line Segment L_1.
$F_2(P_2) = A_2 + P_2 * (\vec{L_2})$ where, $(\vec{L_2}) = A_2 - B_2$ and $P_2 \in [0, 1]$ represents Line Segment L_2.

Solving for parameters P_i.
$A_1 + P_1 * (A_1 - B_1) = A_2 + P_2 * (A_2 - B_2)$

Now, at intersection $F_1(P_1) = F_2(P_2)$
$P_1 * (A_1\vec{B_1}) = (A_2 - A_1) + P_2 * (A_2\vec{B_2})$
Multiplying by $\vec{L_2}^{\perp}$

$$P_1 * (\vec{L_2}^{\perp})(\vec{L_1}) = (\vec{L_2}^{\perp})(A_2 - A_1) + P_2 * (\vec{L_2}^{\perp})(\vec{L_2}) \qquad (\because (\vec{a}^{\perp})(\vec{a}) = 0)$$

$$P_1 = \frac{(\vec{L_2}^{\perp})(A_2 - A_1)}{(\vec{L_2}^{\perp})(\vec{L_1})} = \frac{(\vec{L_2}^{\perp})(A_2\vec{A_1})}{(\vec{L_2}^{\perp})(\vec{L_1})}$$

Similarly, Parameter $P_2 = \frac{(\vec{L_1}^{\perp})(A_1 - A_2)}{(\vec{L_1}^{\perp})(\vec{L_2})} = \frac{(\vec{L_1}^{\perp})(A_1\vec{A_2})}{(\vec{L_1}^{\perp})(\vec{L_2})}$.

Now using this parameter equation we derive our algorithm, we need to consider all the possible cases for given two line segments.

(a) **Case 1: Parallel Lines**

Parallel Lines don't intersect, so if we take a perpendicular to L_1 (i.e. $L_1^{\perp}$) it will also be perpendicular L_2 so their dot product will be zero. (i.e. $perpdot(L_1, L_2) = 0$). So only one of them can be vertical (i.e. Slope $= \infty$.)

(b) **Case 2: Coinciding or Overlapping Lines**

There are multiple cases where a line can be coincident like complete or partial overlap, or lie on a same infinite line. If the lines coincide A_2, A_1 lie on the same line, then $(A_2\vec{A_1}^{\perp}).\vec{L_1} = 0$. Further classification is not required for our specific algorithm.

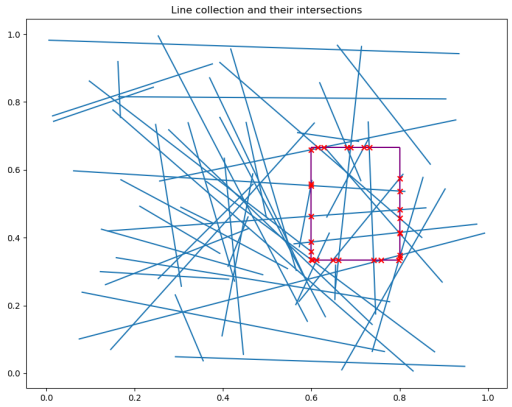
(c) **Case 3: Non-parallel Lines**

As per line segment definition if parameter $P_i \in [0, 1]$ then the point is in that line segment L_i. Hence, for a intersection point to be one both lines that parameters P_1 and $P_2 \in [0, 1]$. If not the point lies outside one or both of the segments.

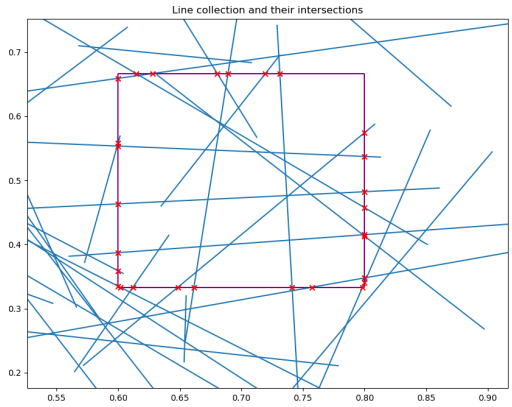
i. Point on both line segments - $P_1 \in [0, 1]$ and $P_2 \in [0, 1]$
ii. Point on one line segment - Only one of $P_1 or P_2 \in [0, 1]$.
iii. Point outside of both line segments - $P_1 \notin [0, 1]$ or $P_2 \notin [0, 1]$

For our algorithm the point need to be on both the line segments for true intersection point so we use Case 2(i). Additionally we check $(\vec{L_1}^{\perp})(\vec{L_2}) = 0$, as if it is zero the parameter equation we derived earlier becomes undefined. It also has geometrically interpretation as if perpendicular of L_1 is also perpendicular to L_2 it implies that both L_1 and L_2 are parallel or coincide. In other cases this can never be zero.

3. (a) Plots for Test Cases for the implementation of segment_pair_intersection().



(a) Case 1: Normal View



(b) Case 1: Zoom-in View

Figure 1: Case1

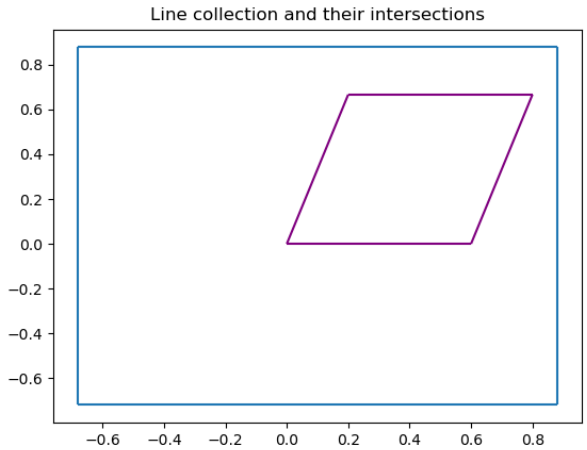
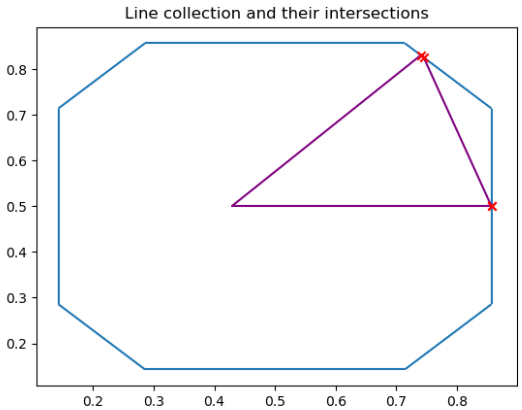
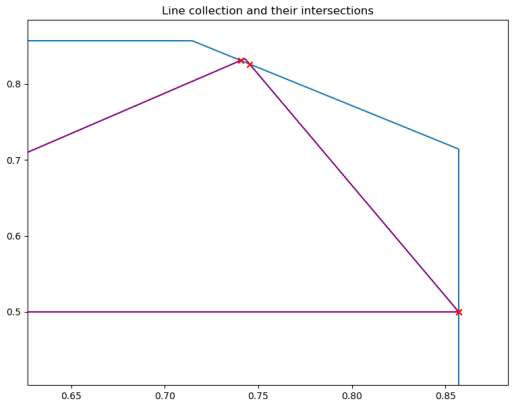


Figure 2: Case 2: Normal View



(a) Case 3: Normal View



(b) Case 3: Zoom-in View

Figure 3: Case3

(b) Complexity Comparison in Big O notation. Where N is total number of line segments. Complexity of segment_pair_intersection() is $\mathbf{O(n^2)}$

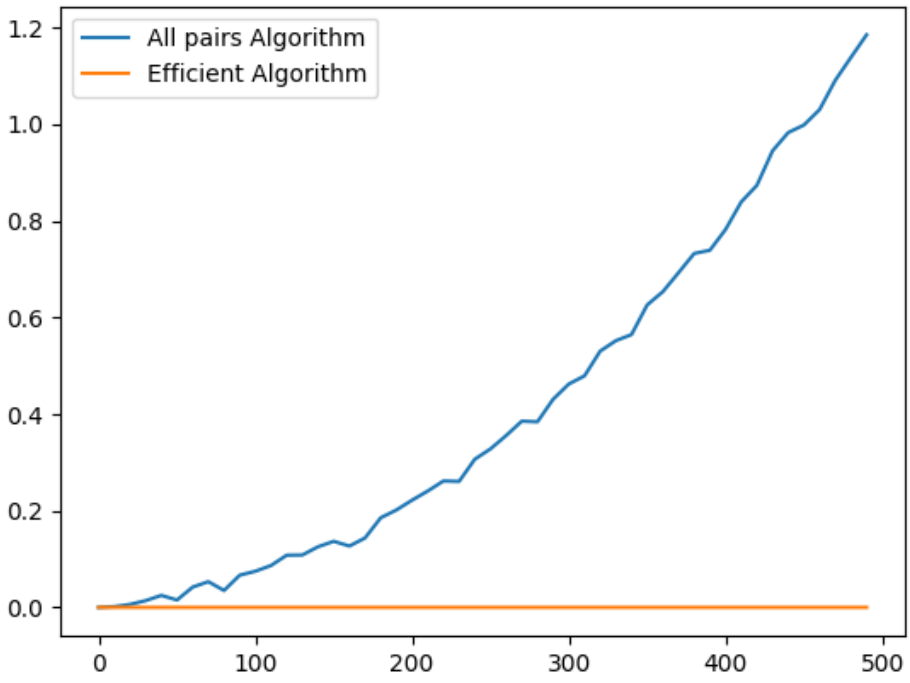


Figure 4: Case 2: Normal View