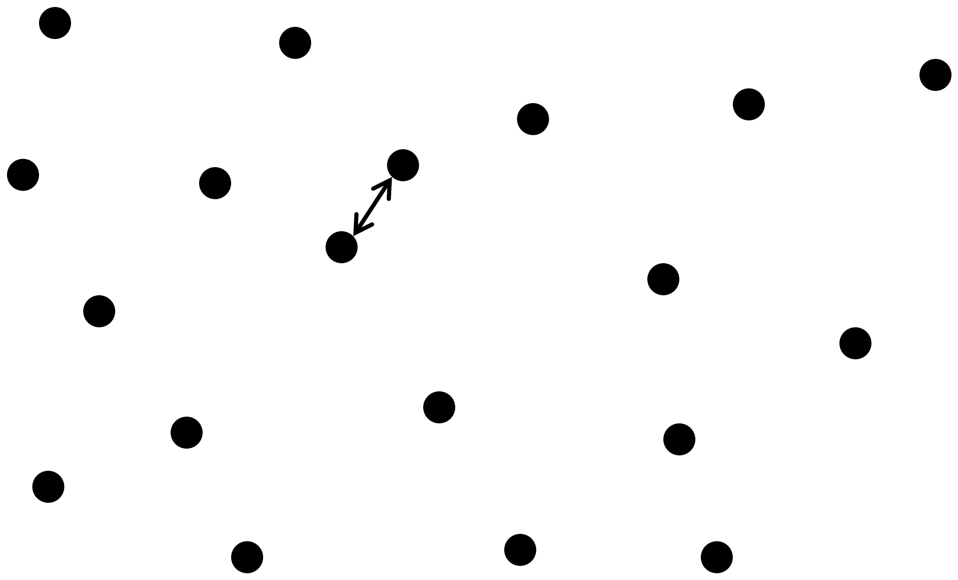
CS133
Computational Geometry

Closest/Farthest Pairs

Closest Pair

- Given a set P of points, find the distance between the closest pair of points



Naïve Algorithm

- Compute all distances
- Find the minimum distance
- Running time: $O(n^2)$

- Can we do better?

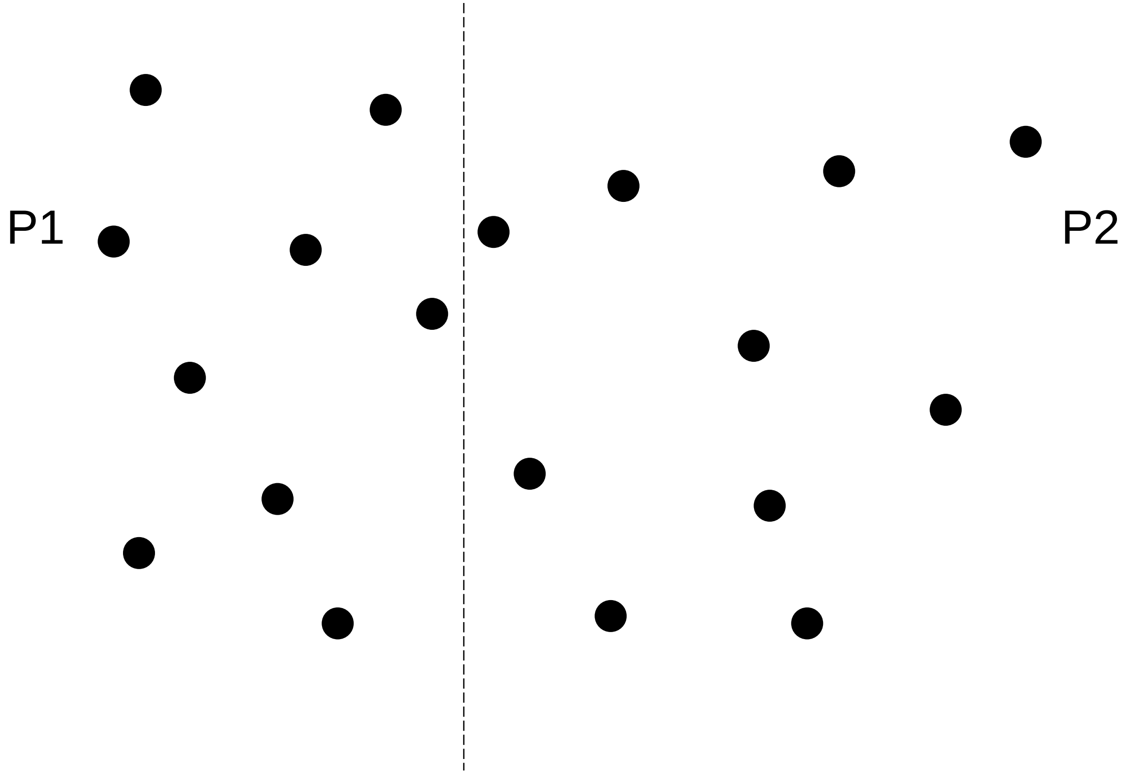
Divide-and-conquer Algorithm

- ClosestPair(P)
  - Split P into two subsets P_1 and P_2
  - cp_1 = ClosestPair(P_1)
  - cp_2 = ClosestPair(P_2)
  - Return Minimum$\{cp_1, cp_2\}$

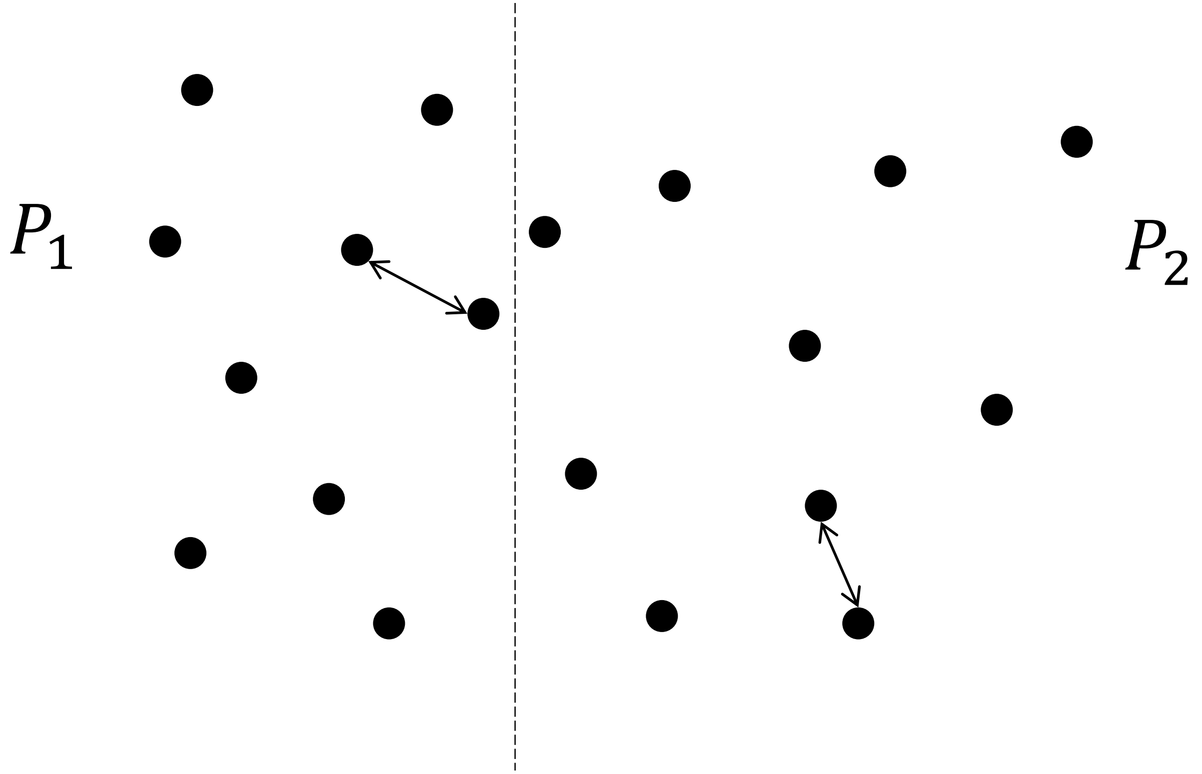
Divide-and-conquer Algorithm

- ClosestPair(P)
  - Split P into two subsets P_1 and P_2
  - cp_1 = ClosestPair(P_1)
  - cp_2 = ClosestPair(P_2)
  - cp_{12}= ClosestPairMiddle(P_1, P_2, min($cp_1.d$, $cp_2.d$))
  - Return Minimum$\{cp_1, cp_2, cp_{12}\}$

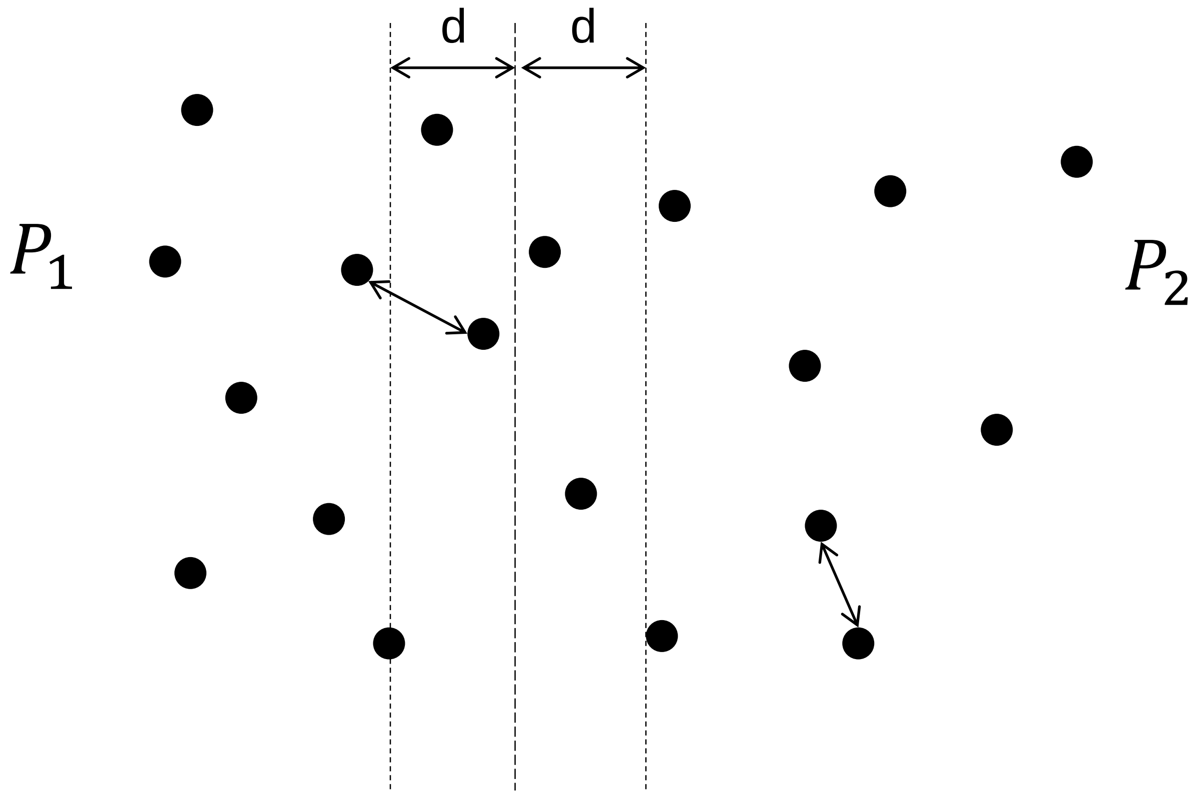
Closest Pair Example



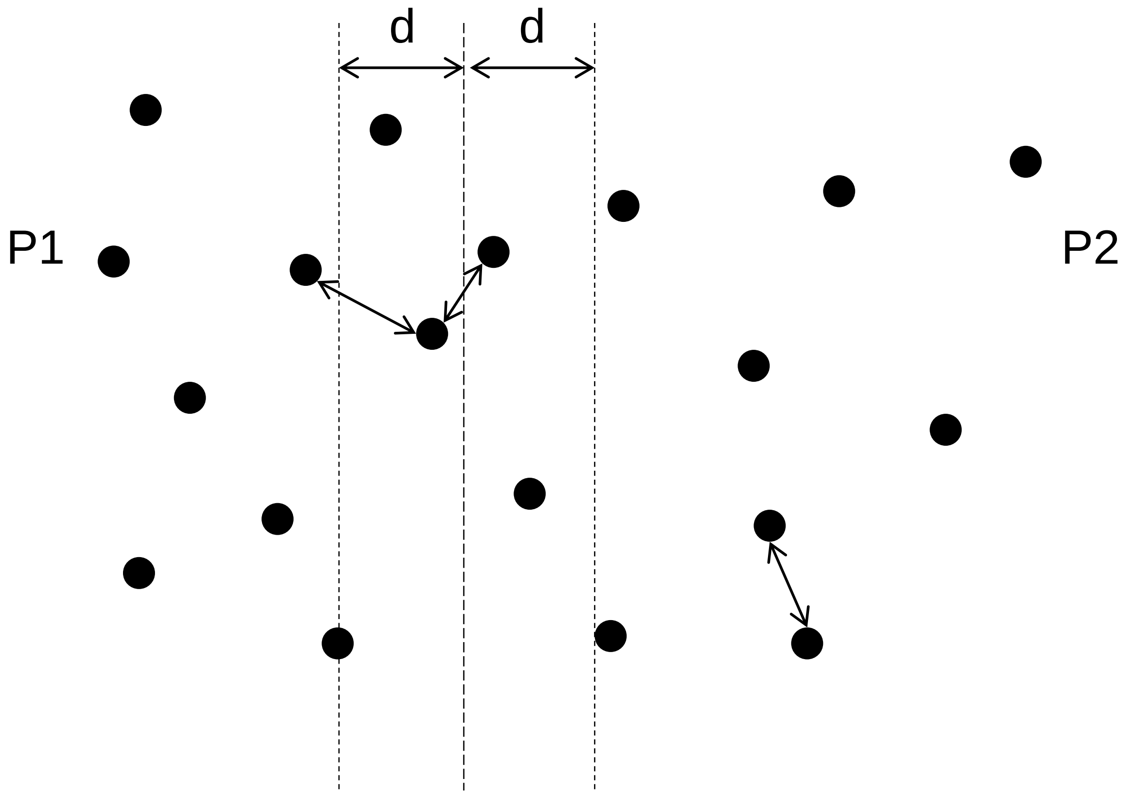
Closest Pair Example



Closest Pair Example



Closest Pair Example



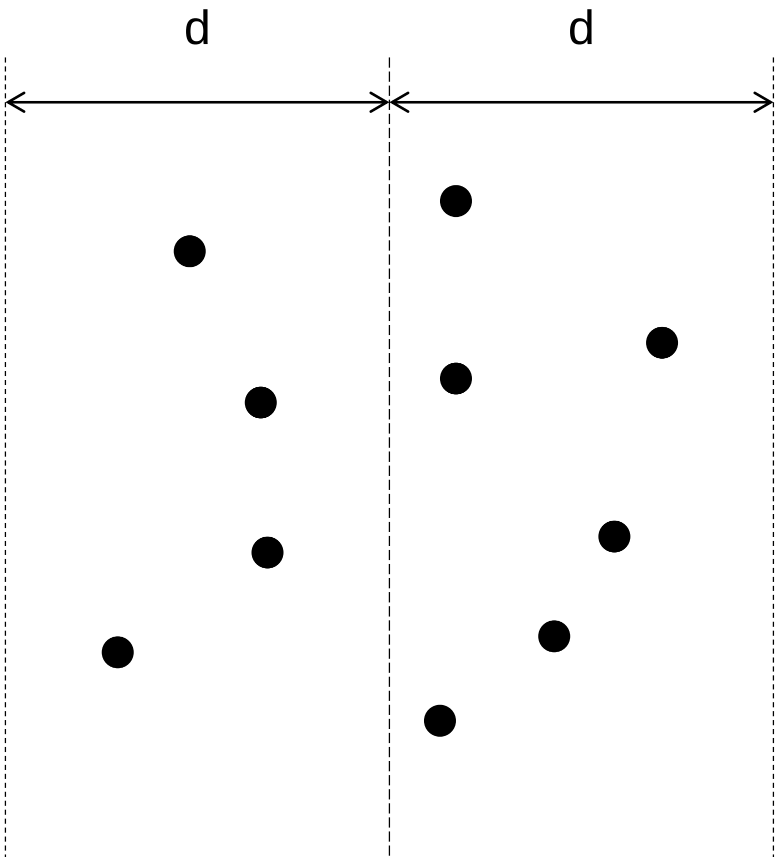
Middle Strip

Find the closest pair of points with a distance at most d within the middle strip

Sort by y and test each point with the next r points

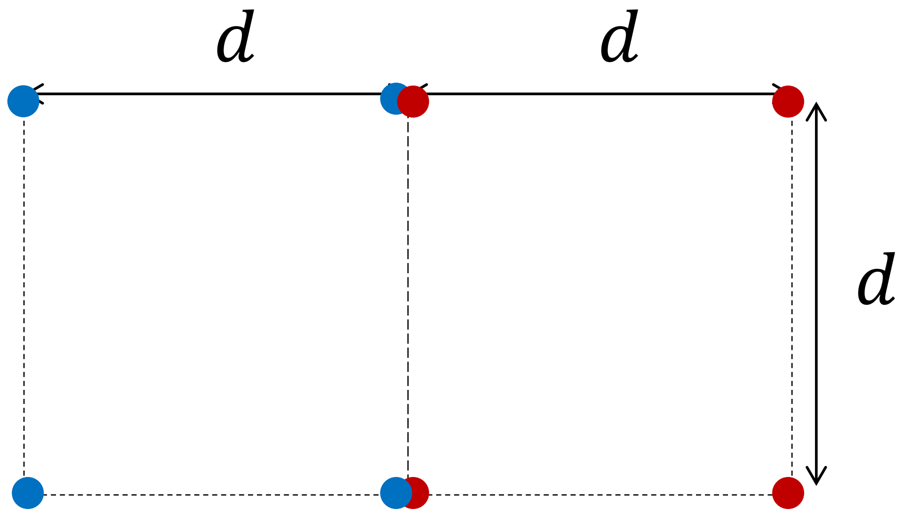
Seemingly $O(n^2)$

Really $O(n)$



Middle Strip

- The search can terminate when $\Delta y > d$
- This limits the search box to $2d \times d$
- The search box can contain up-to eight points
- Each point can be compared to at most seven points



Pseudo Code

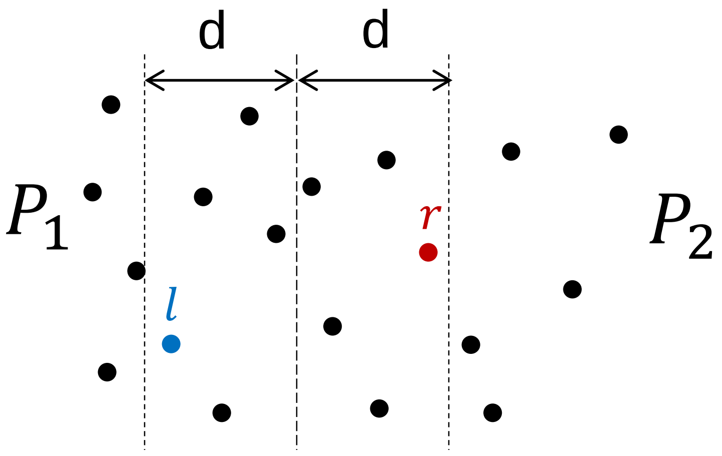
- ClosestPair(P)
  - Sort P by x
  - Return ClosestPairRecursive(P, 1, $|P|$)

ClosestPairRecursive(P, s, e)

- If $e - s = 2$
  - Return $(s, e, \|e - s\|)$
- If $e - s = 1$
  - Return $(-1, -1, \infty)$
- $m = \frac{s+e}{2}$
- cp_1 =ClosestPairRecursive(P,s,m)
- cp_2 =ClosestPairRecursive(P,$m + 1$,e)
- cp_{12} =ClosestPairMiddle($P, s, e, m, \min\{cp_1.d, cp_2.d\}$)
- Return $\min\{cp_1, cp_2, cp_{12}\}$

ClosestPairMiddle(P, s, e, m, d)

- $x_m = P[m].x$
- l =binarySearch+($P, s, m, x_m - d$)
- r = binarySearch−($P, m, e, x_m + d$)
- $P_r = P[l, r]$
- Sort P_r by y
- $cp = (-1, -1, \infty)$
- For $p_i \in P_r$
  - For ($j = i; j < |P_r| \wedge p_j.y - p_i.y < d; j++$)
    - If $\|p_j - p_i\| < cp.d$
      - $cp = (i, j, \|p_j - p_i\|)$
- Return cp



Running Time

- Initial sorting $O(n \log n)$
- Recursive part:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$$

- Overall running time = $O(n \log n)$
- Question: In the ClosestPairMiddle algorithm, how to sort P_r by y in linear time?

Stable Splitting

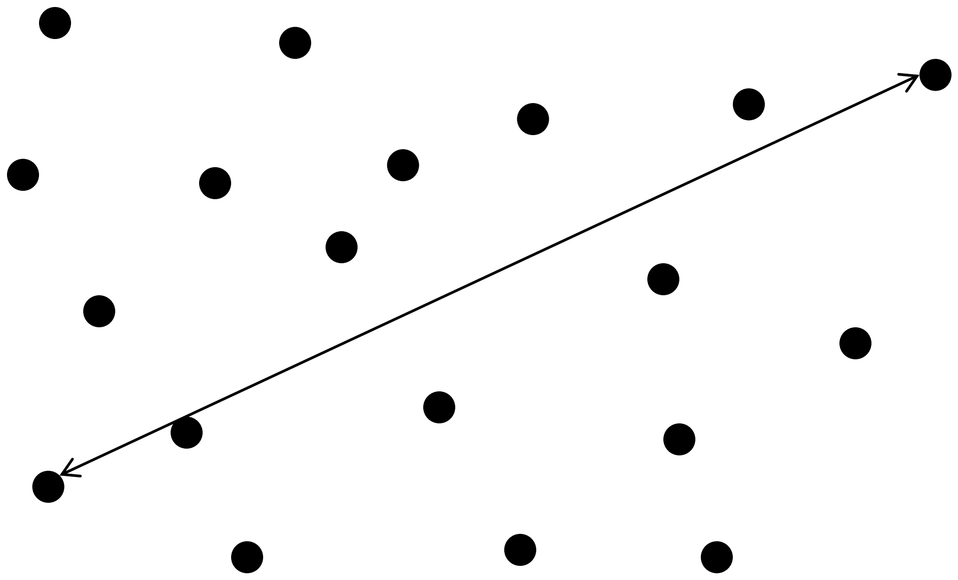
- Presort P by x and y. Let's call them P_x and P_y, respectively.
- When splitting P along a vertical line, we split both P_x and P_y in linear time
- Assume that x_m is the x-coordinate of the split line where $m = \frac{s+e}{2}$
- $P_{1_x} = P[s..m]$ and $P_{2_x} = P[m+1..e]$
- Initialize P_{1_y} and P_{2_y} as empty arrays of the same size of P_{1_x} and P_{2_x}, respectively

Stable Splitting

- Scan P_y and append each point p_j to either P_{1_y} or P_{2_y}, depending on the x-coordinate
- If $p_j.x \leq x_m$
  - Append p_j to P_{1_y}
- Else if $p_j.x > x_m$
  - Append p_j to P_{2_y}
- The same approach can be used to find all the points in the vertical strip, which is bounded by two x-coordinates, sorted by y

Farthest Pair

- Given a set P of points, find the distance of the farthest pair of points

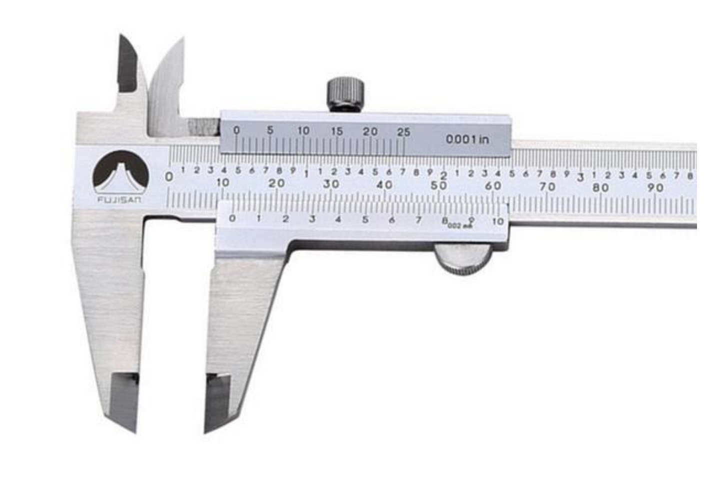


Naïve Algorithm

- Compute all pair-wise distances
- Find the maximum
- Running time $O(n^2)$

Rotating Calipers

- Rotate a pair of calipers around the points
- Find the largest distance that the calipers made

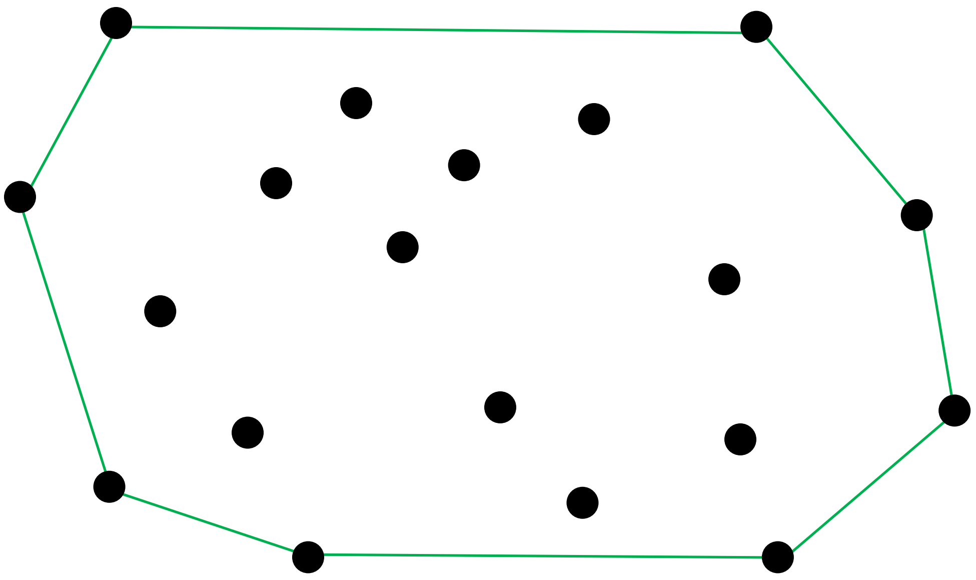


Revisit Convex Hull

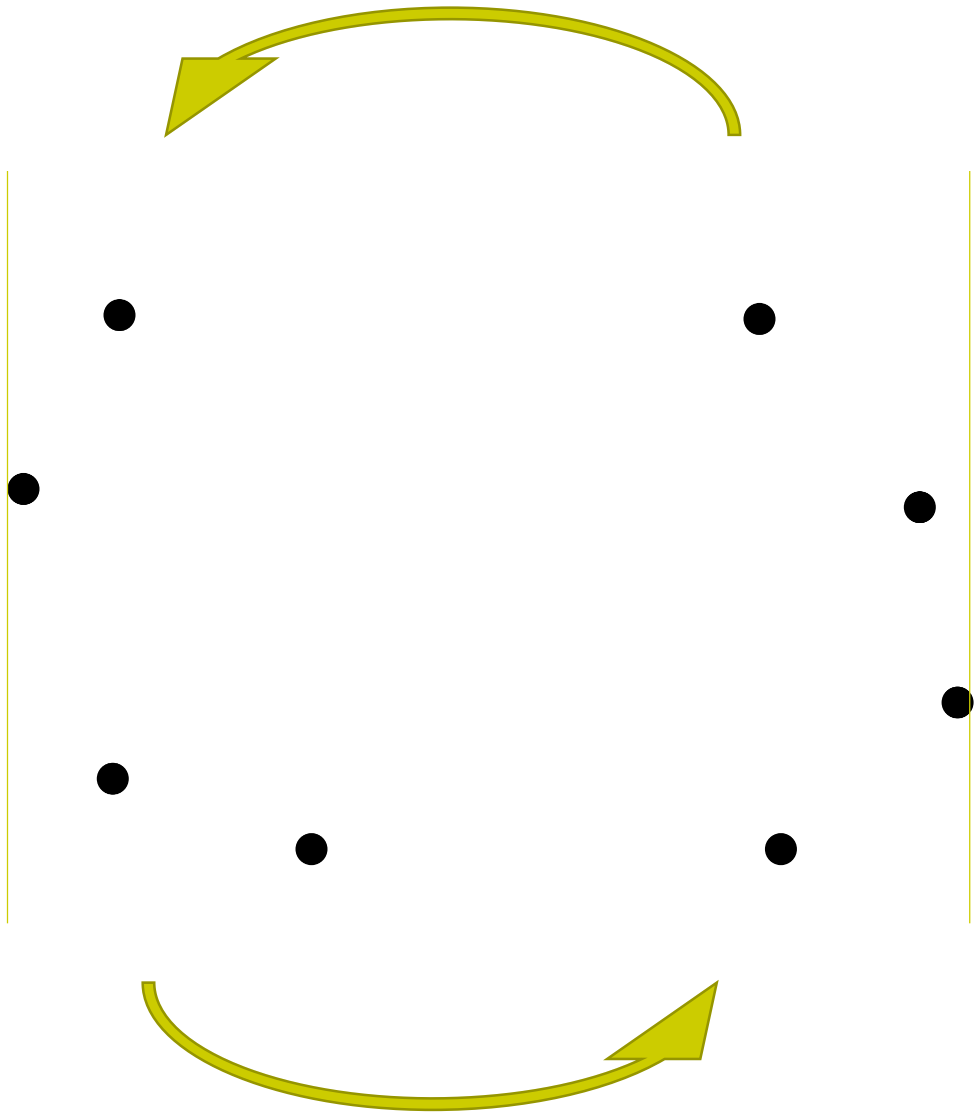
- The farthest pair of points have to be on the convex hull
- Proof by contradiction

Rotating Calipers

- For simplicity, we apply the rotting calipers algorithm on the convex hull

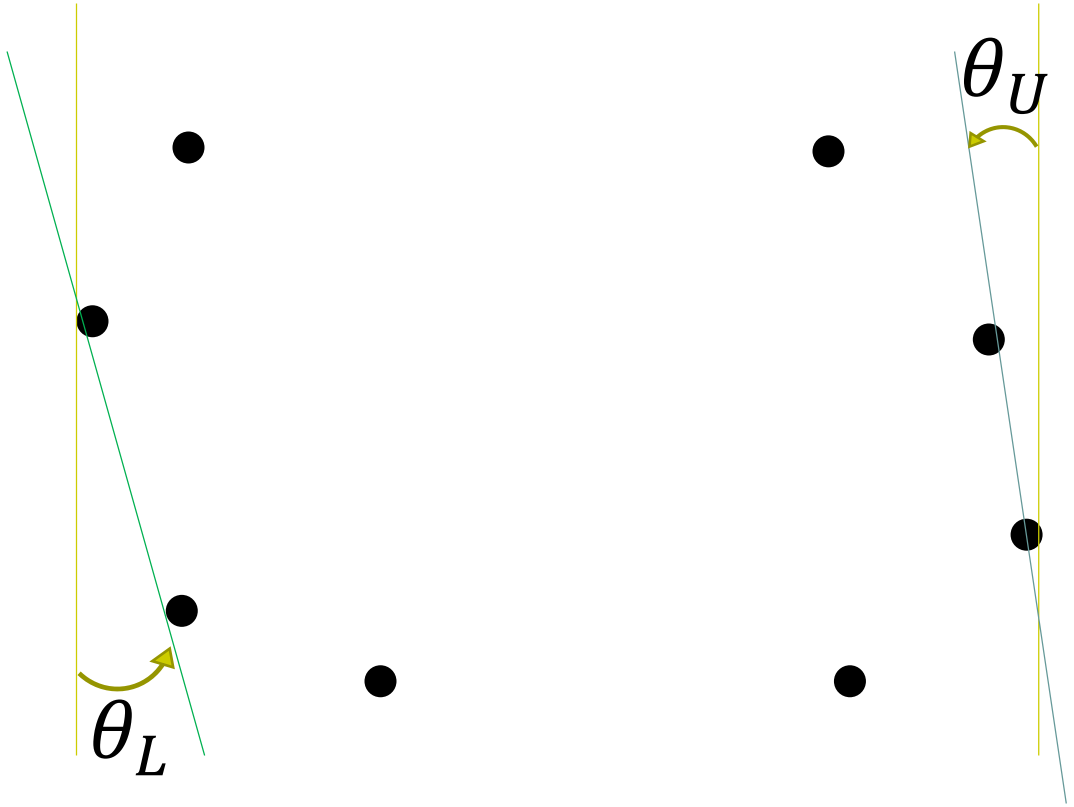


Rotating Calipers Example



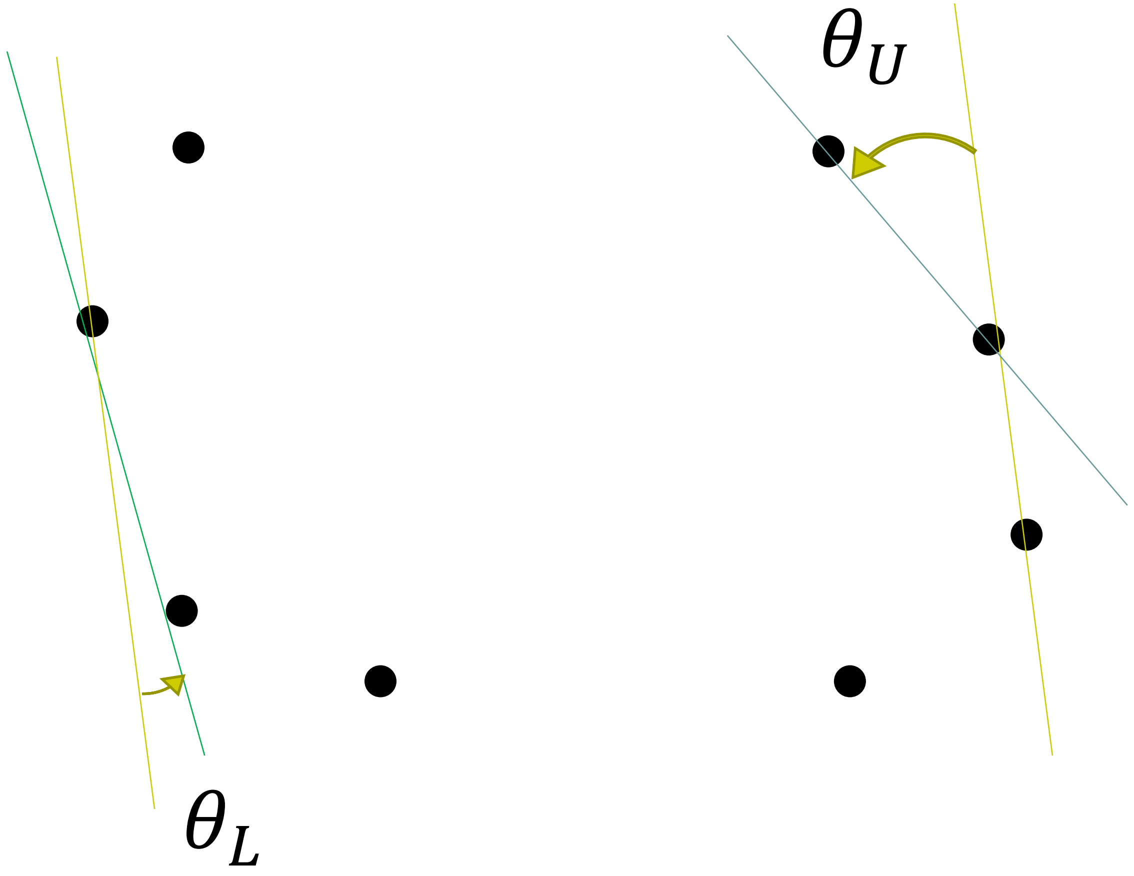
Rotating Calipers Example

Rotate the calipers by $\min\{\theta_L, \theta_U\}$



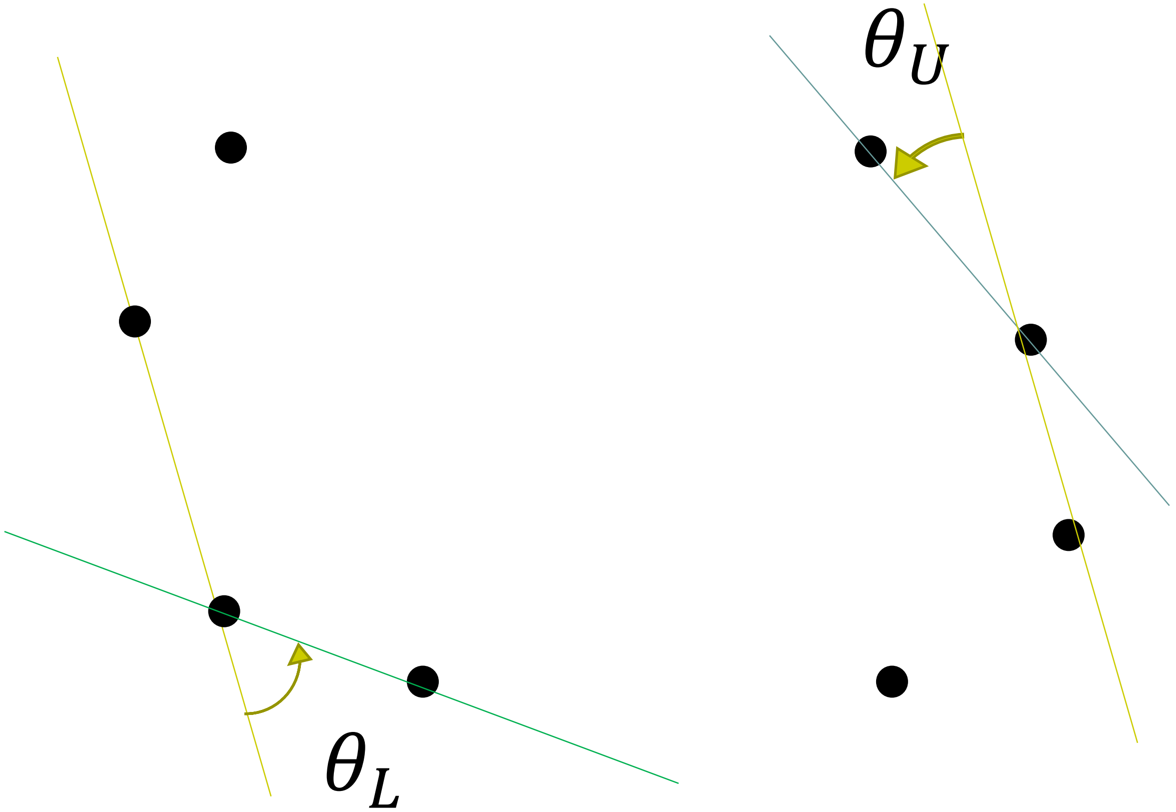
Rotating Calipers Example

Rotate the calipers by $\min\{\theta_L, \theta_U\}$

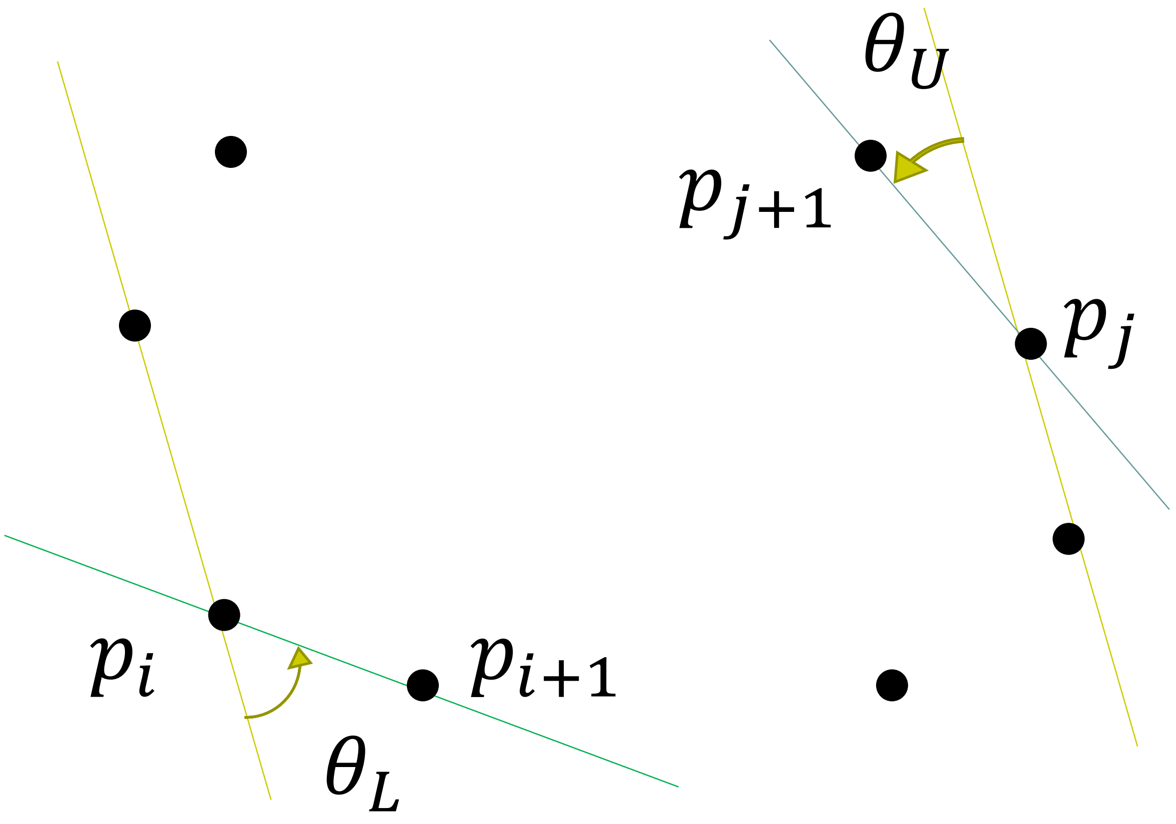


Rotating Calipers Example

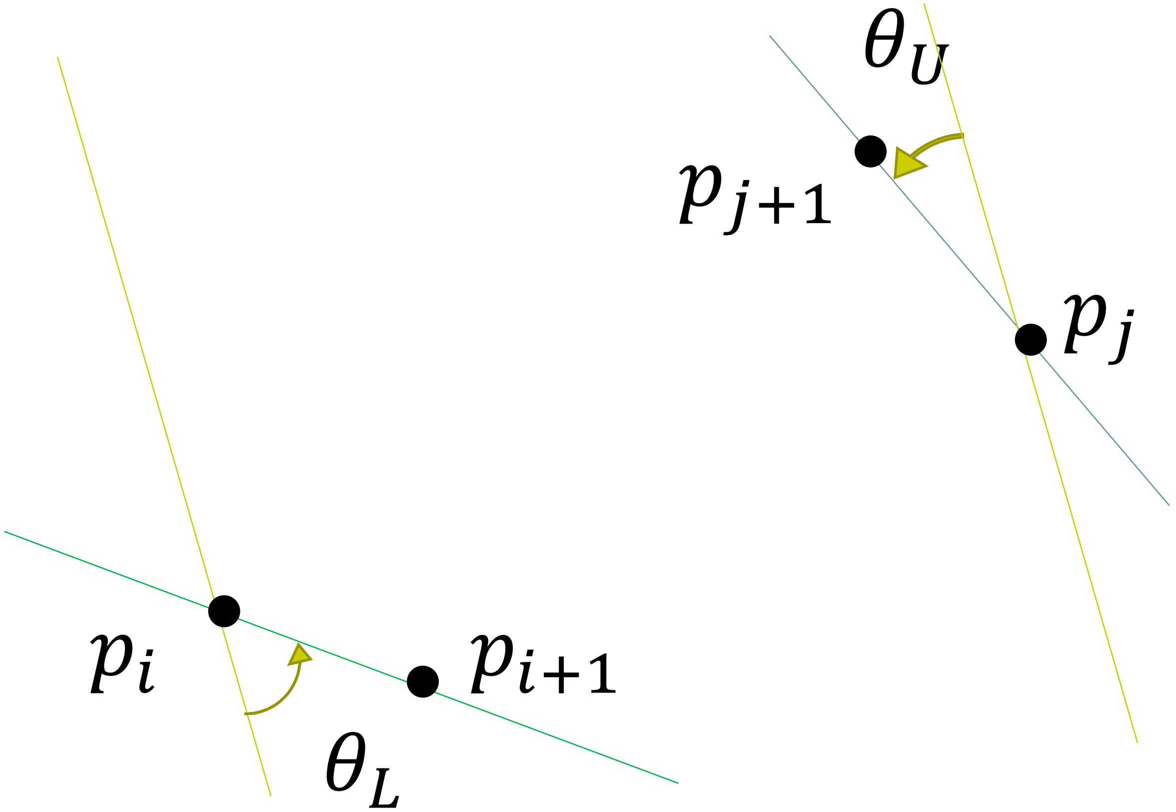
Rotate the calipers by $\min\{\theta_L, \theta_U\}$



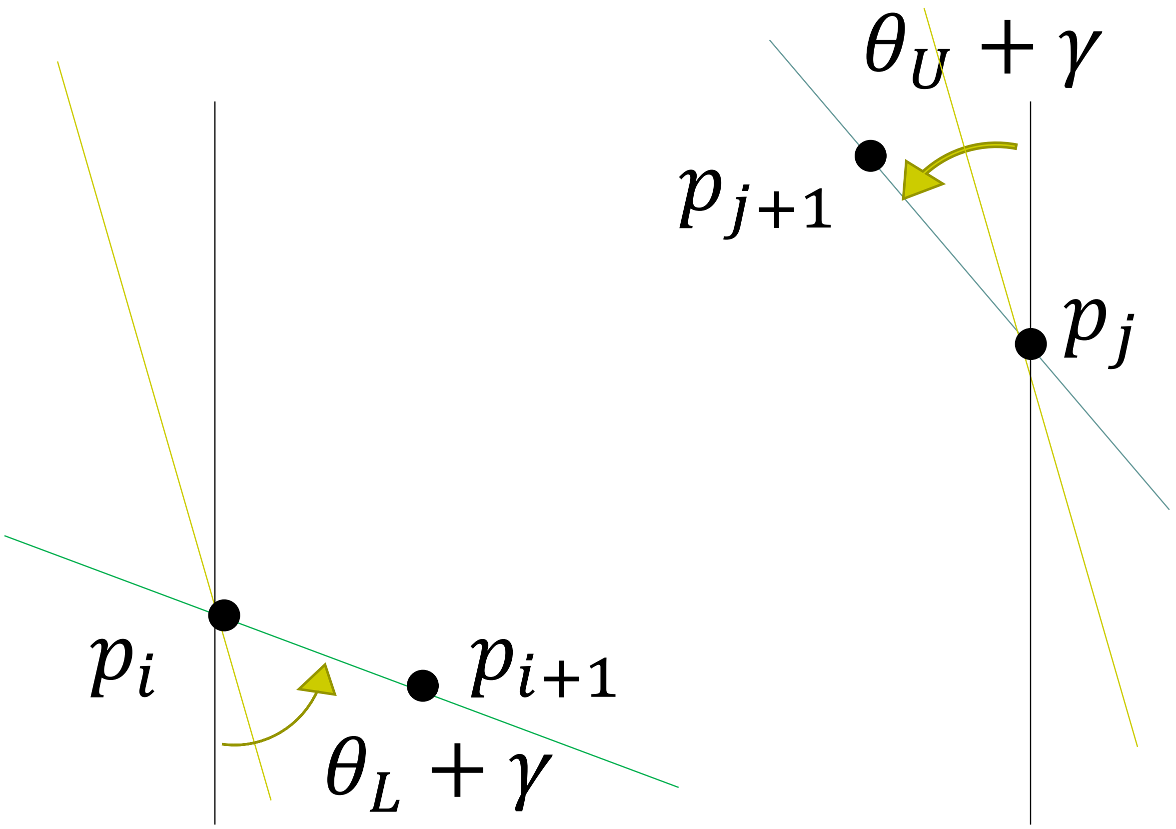
Finding Minimum Angle



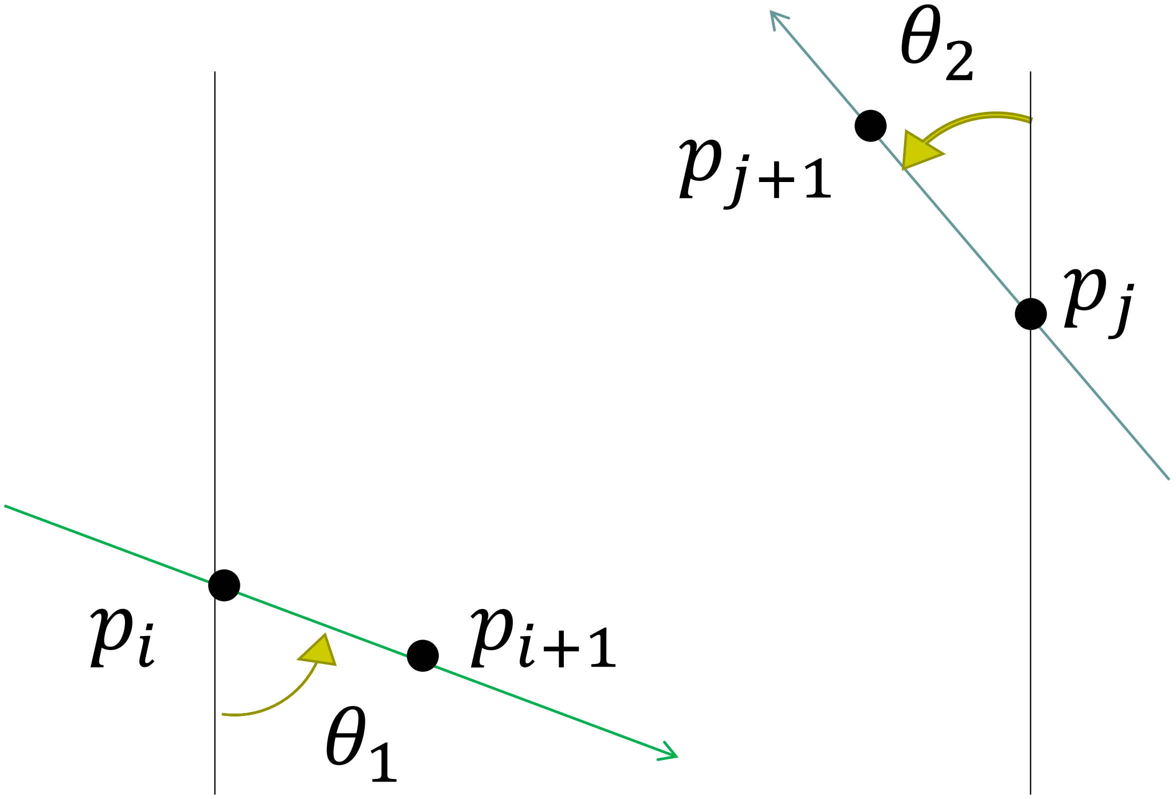
Finding Minimum Angle



Finding Minimum Angle

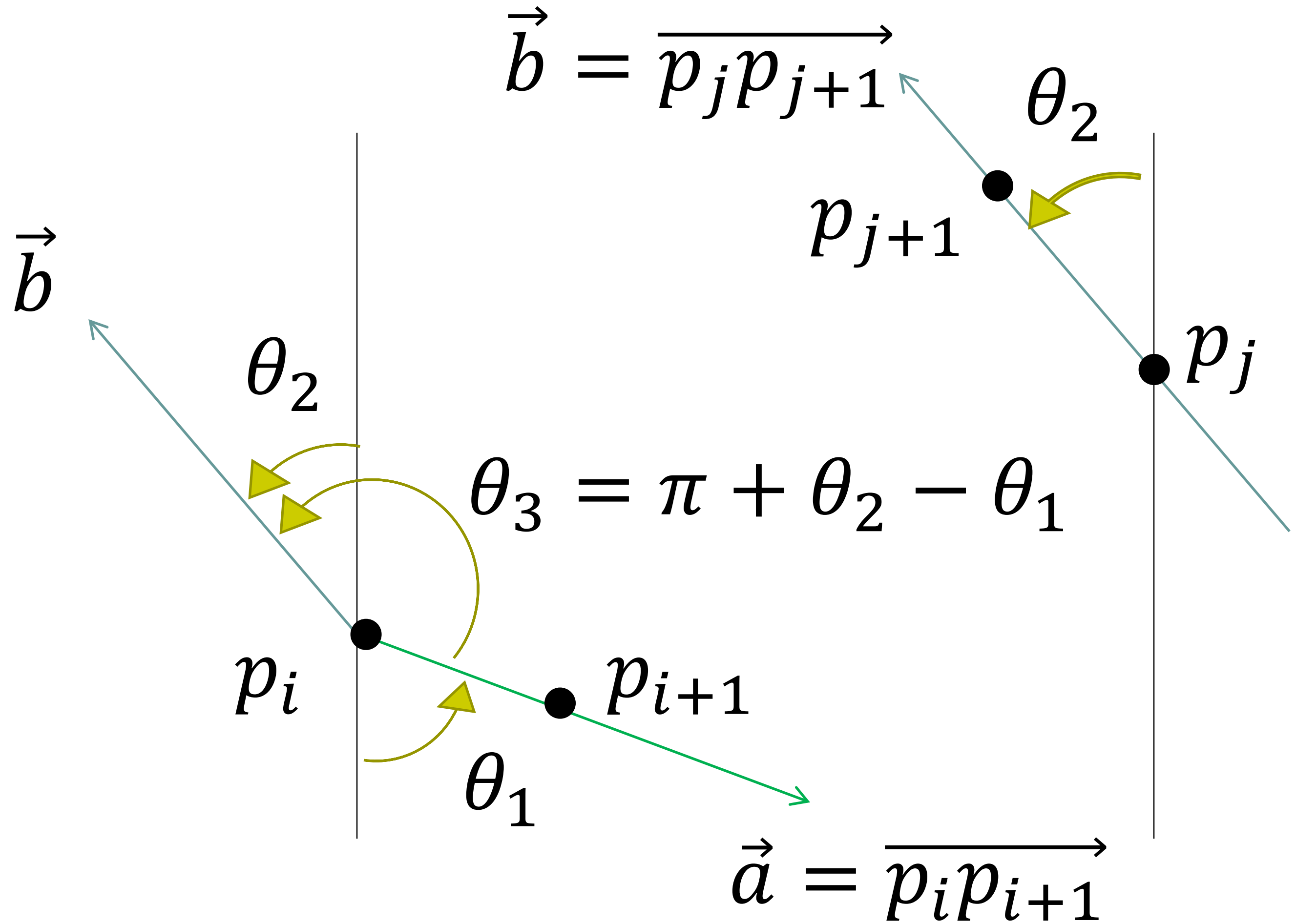


Finding Minimum Angle



Finding Minimum Angle

$$\vec{a} \times \vec{b} = \|a\| \cdot \|b\| \cdot \sin\theta_3 > 0?$$



Finding Minimum Angle

- $\vec{a} \times \vec{b} = \|a\| \cdot \|b\| \cdot \sin\theta_3 > 0$
- $\Rightarrow \sin\theta_3 > 0$
- $\Rightarrow 0 < \theta_3 < \pi$
- $\Rightarrow 0 < \pi + \theta_2 - \theta_1 < \pi$
- $\Rightarrow 0 < \theta_1 - \theta_2 < \pi$
- $\Rightarrow 0 < \theta_L + \gamma - \theta_R - \gamma < \pi$
- $\Rightarrow 0 < \theta_L - \theta_U < \pi$