Collision Checking Slides

Course 4, Module 6, Lesson 2

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Collision Checking Challenges

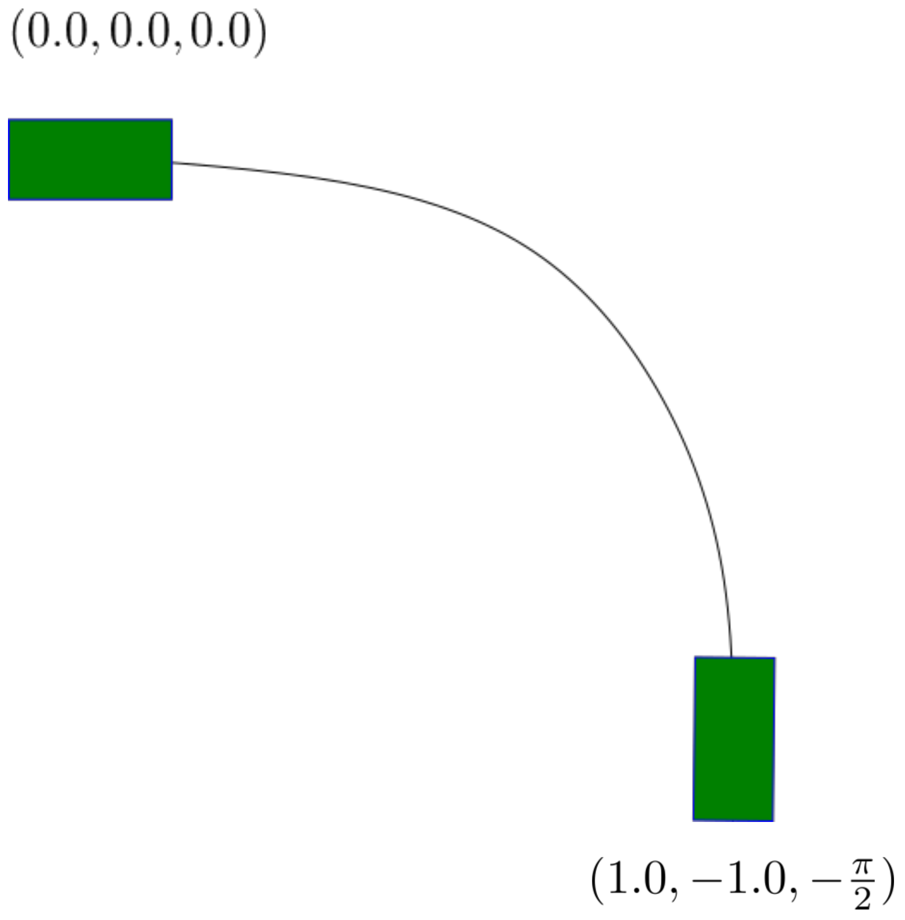
- Computationally intensive
- Requires perfect information to guarantee safety
- Needs to be approximated, and must be robust to noise

occupancy grid : an imperfect estimate

need buffers to be robust

Swath Computation

- Area occupied by car along path generated by rotating the car's footprint by each x, y, θ along the path



union

a function returning the set point

translated

rotated

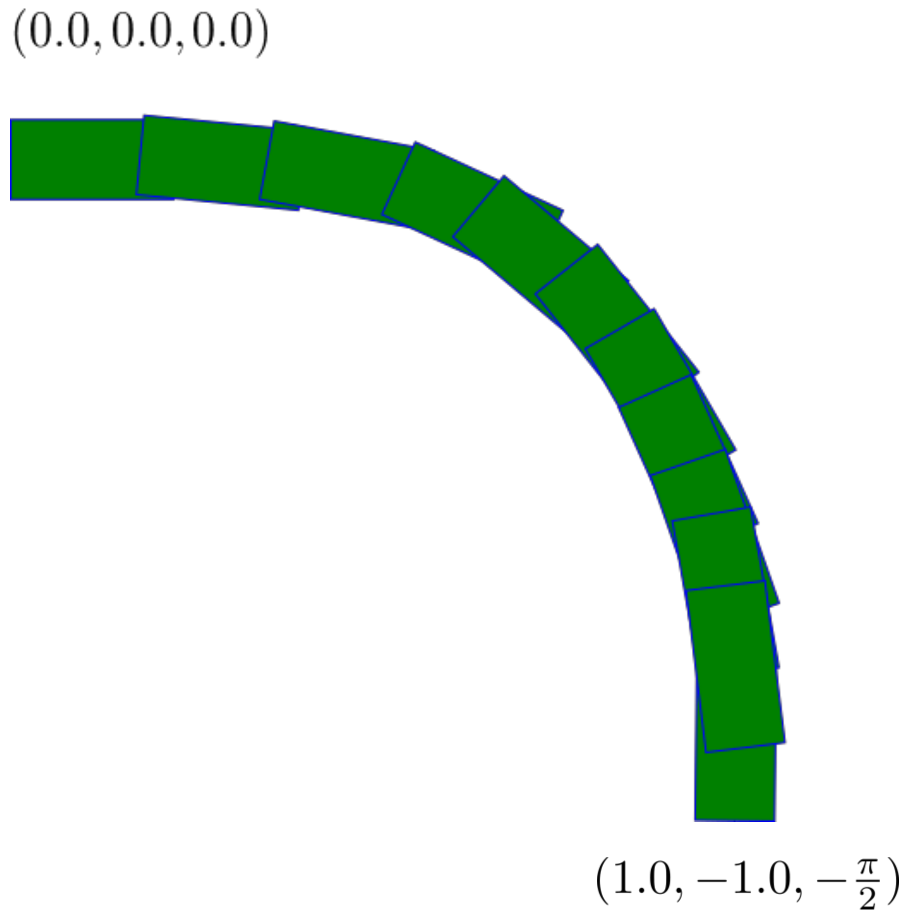
$$S = \bigcup_{p \in P} F\big(x(p), y(p), \theta(p)\big)$$

a path consists of a set of points p

Swath Computation

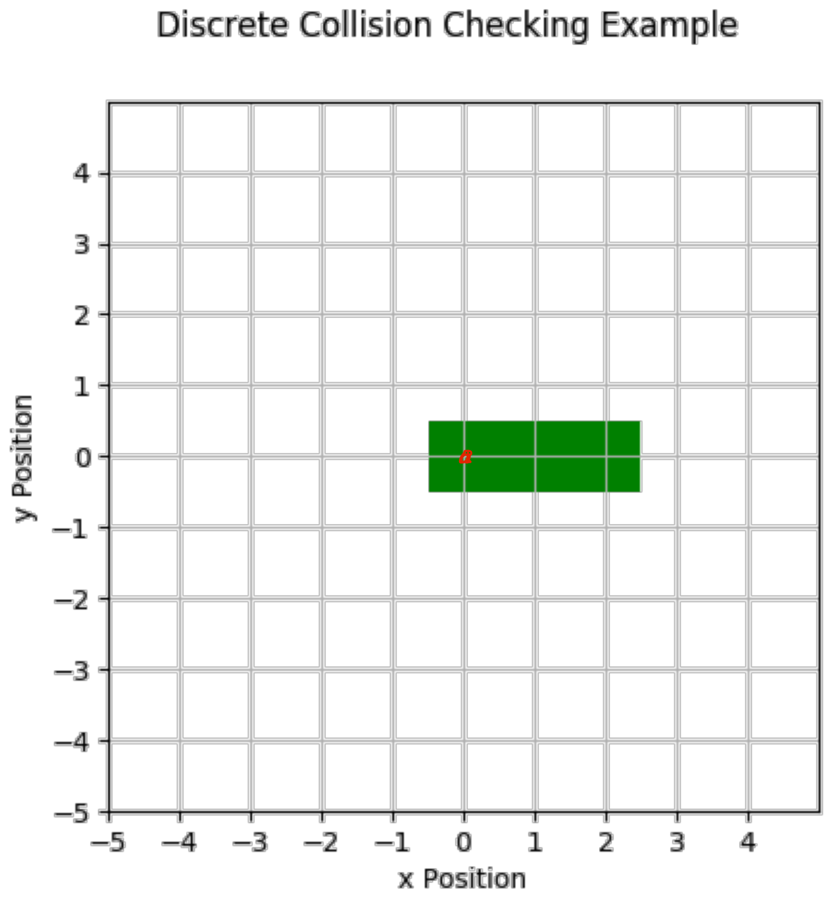
- Area occupied by car along path generated by rotating the car's footprint by each x, y, θ along the path
- Swath along path is the union of each rotated and translated footprint
- Swath can then be checked for collisions

$$S = \cup_{p \in P} F\big(x(p), y(p), \theta(p)\big)$$



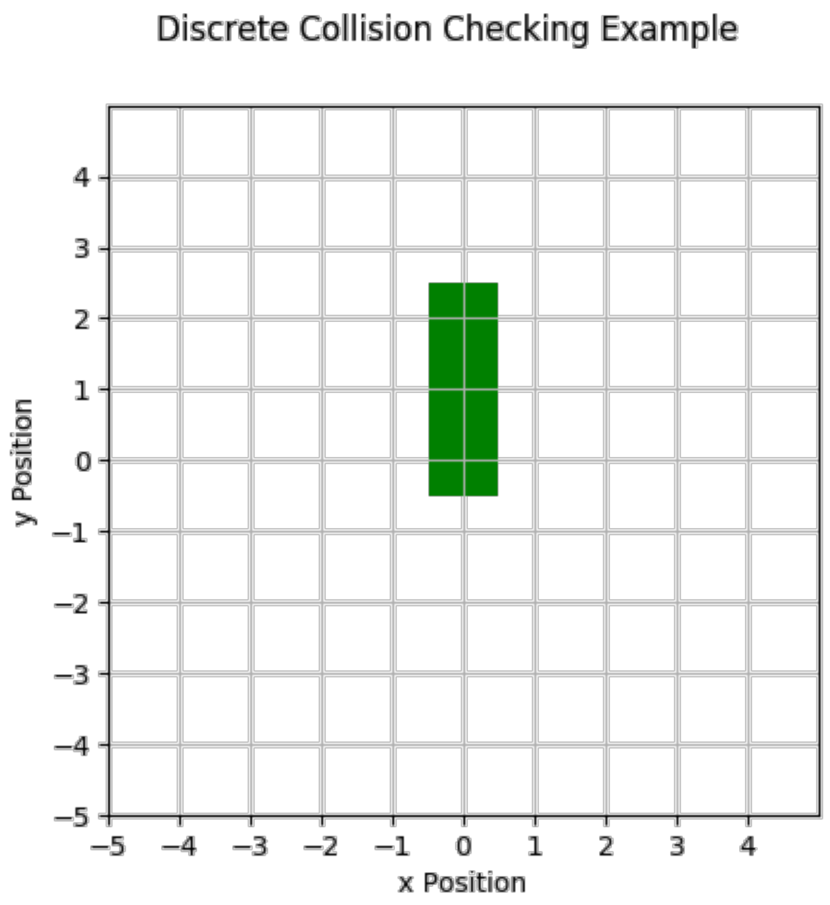
Discretized Example

- Initial state of the vehicle in the occupancy grid, with base link at the origin
- Will need to rotate and translate to get the new footprint at point $(1.0, 2.0, \frac{\pi}{2})$



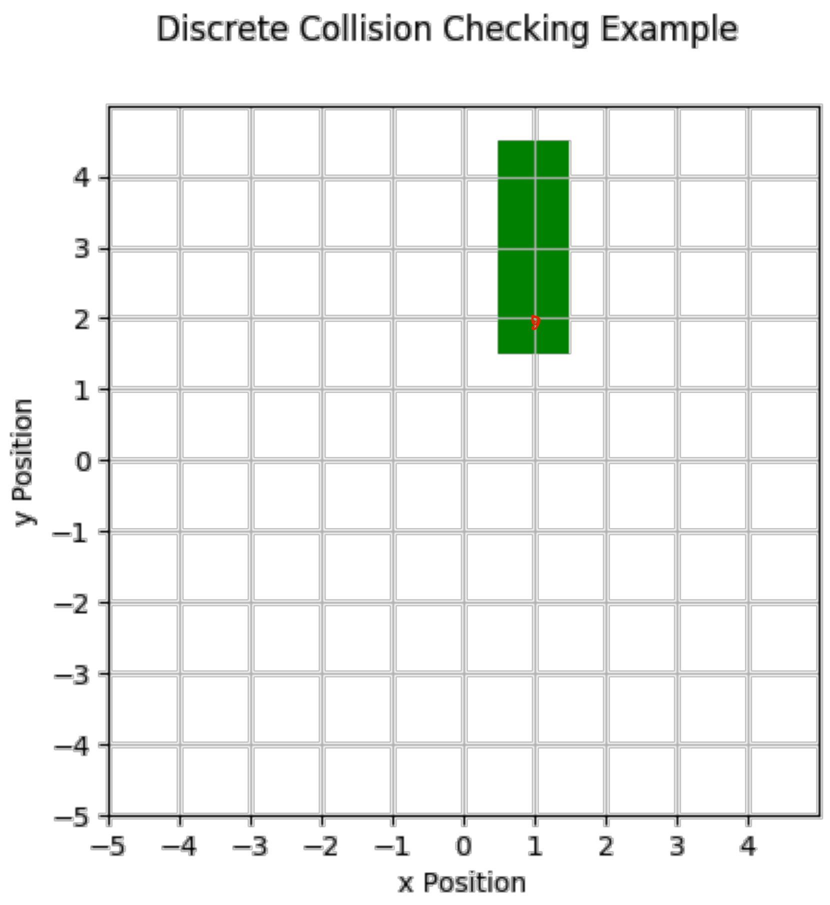
Discretized Example

- First, rotate the footprint about the origin by $\frac{\pi}{2}$



Discretized Example

- First, rotate the footprint about the origin by $\frac{\pi}{2}$
- Next, translate each point by (1.0, 2.0)



Discretized Example

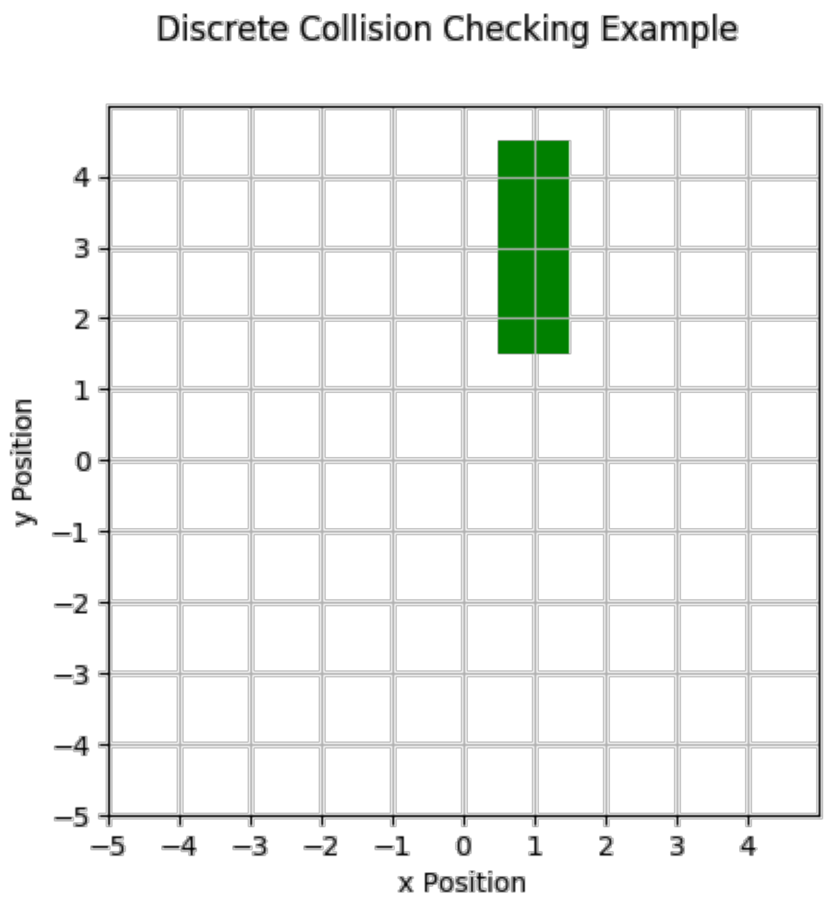
- To compute the occupancy grid index for each point in the footprint, add half the width/height of the occupancy grid, and divide by the grid resolution δ
- Swath is then the union of these indices

no duplicates

$$x_i = \frac{x(p) + \frac{X}{2}}{\delta}$$

$$y_i = \frac{y(p) + \frac{Y}{2}}{\delta}$$

$$S = S \cup (x_i, y_i)$$

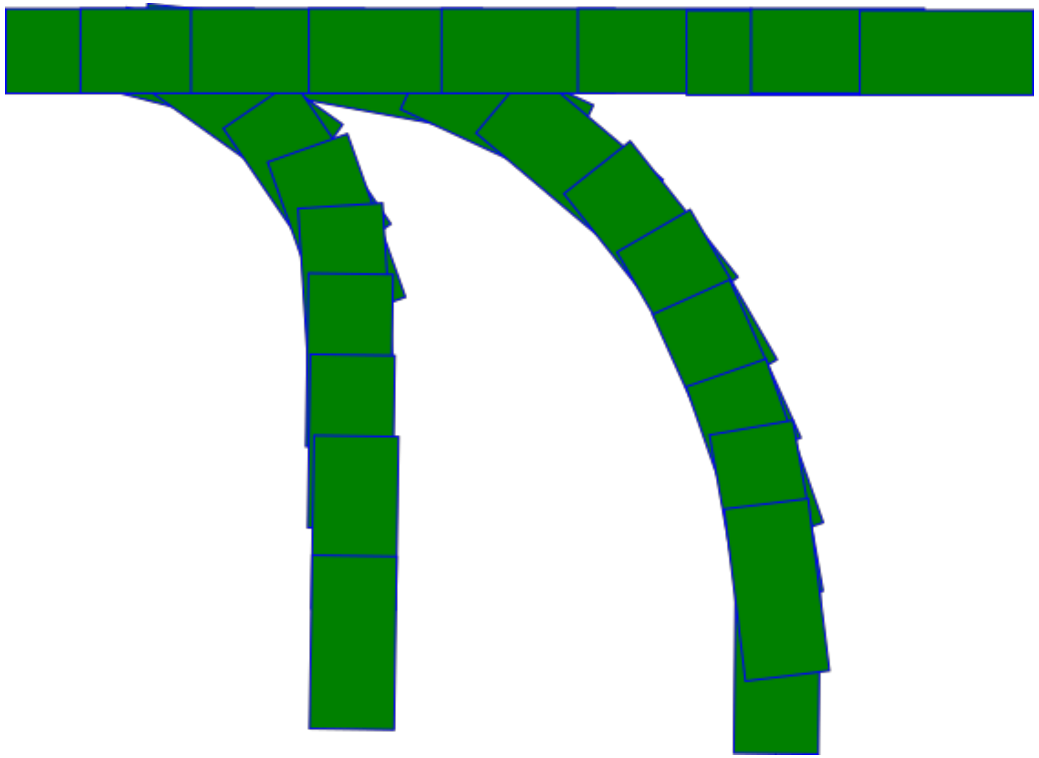


- computationally expensive
- no buffers or errors in obstacle positions

Lattice Planner Swaths

when exploiting alot of repetition

- Swath based methods are useful for lattice planners, as the swath sets can be computed offline
- Online collision checking is then simplified using lookup tables

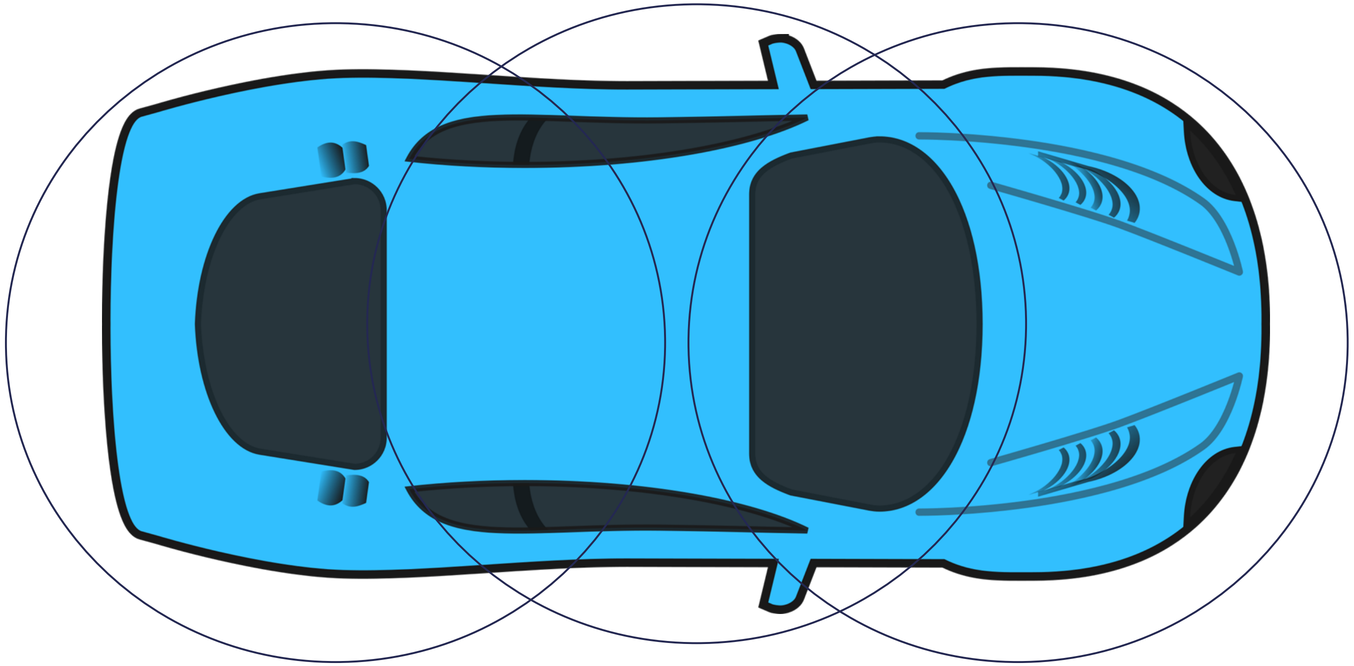


Speed and Robustness

- Need to improve speed
- Need to be robust to noise
- Use conservative approximations to solve both of these problems
- Want algorithmic speedup without sacrificing path quality

Conservative Approximations

- Conservative approximations may report a collision even if there isn’t one, but will never miss a collision if it were to actually happen
- The car can be completely encapsulated by three circles

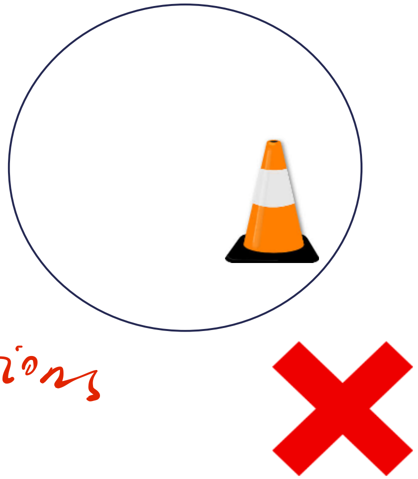


Circle Collision Checking

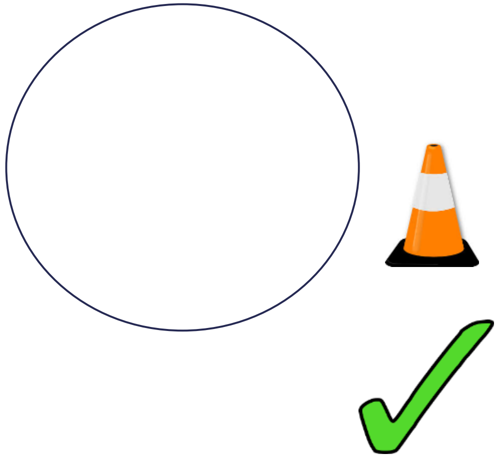
- Circle approximation is effective because it is fast to check if an occupancy grid point lies within a circle of radius r centered at (x_c, y_c)

faster than polygon intersections

- If obstacle in occupancy grid lies within circle, a collision is reported
- Otherwise, due to conservative approximation, no collision is possible



$$\|(x_i, y_i) - (x_c, y_c)\| \leq r$$

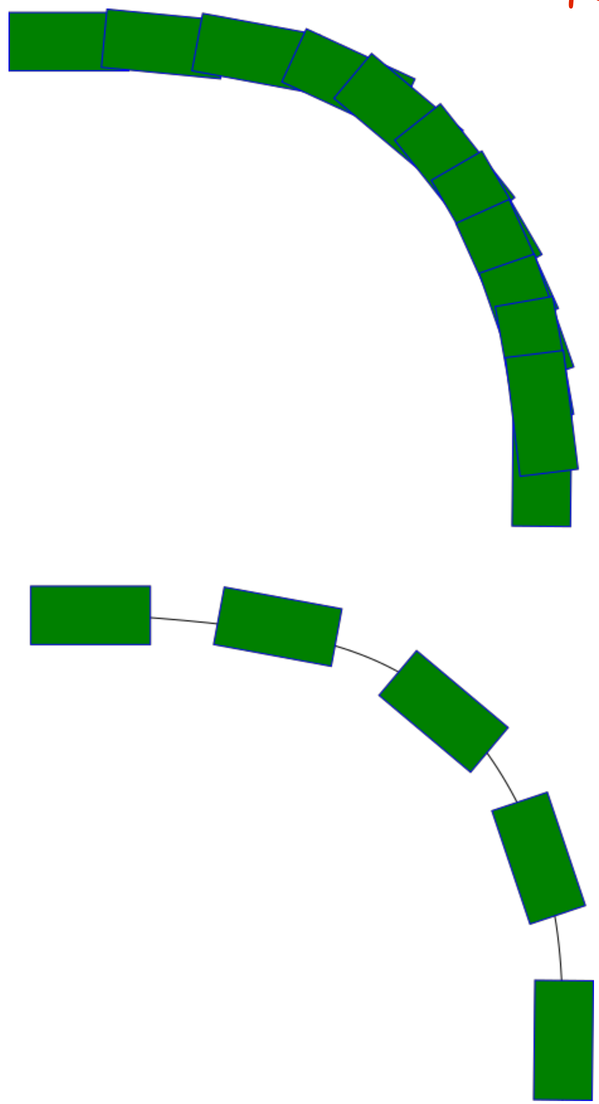


$$\|(x_i, y_i) - (x_c, y_c)\| > r$$

Discretization Resolution

- Collision checking accuracy is impacted by the resolution of our discretization
- Higher fidelity collision checking requires a finer resolution for occupancy grids and path points, and will require more computational resources

same footprints & paths



Summary

- Learned how to use occupancy grid to implement collision checking algorithms
- Introduced swath-based and circle-based collision checking