Roadmap

The first group of discretization approaches that you will learn is referred to by the name Roadmap. These methods represent the configuration space using a simple connected graph - similar to how a city can be represented by a metro map.

Roadmap methods are typically implemented in two phases:

- The **construction phase** builds up a graph from a continuous representation of the space. This phase usually takes a significant amount of time and effort, but the resultant graph can be used for multiple queries with minimal modifications.
- The **query phase** evaluates the graph to find a path from a start location to a goal location. This is done with the help of a search algorithm.

In this Discretization section, we will only discuss and evaluate the construction phase of each Roadmap method. Whereas the query phase will be discussed in more detail in the Graph Search section, following Discretization.

The two roadmap methods that you will learn next are the Visibility Graph, and Voronoi Diagram methods.

NEXT

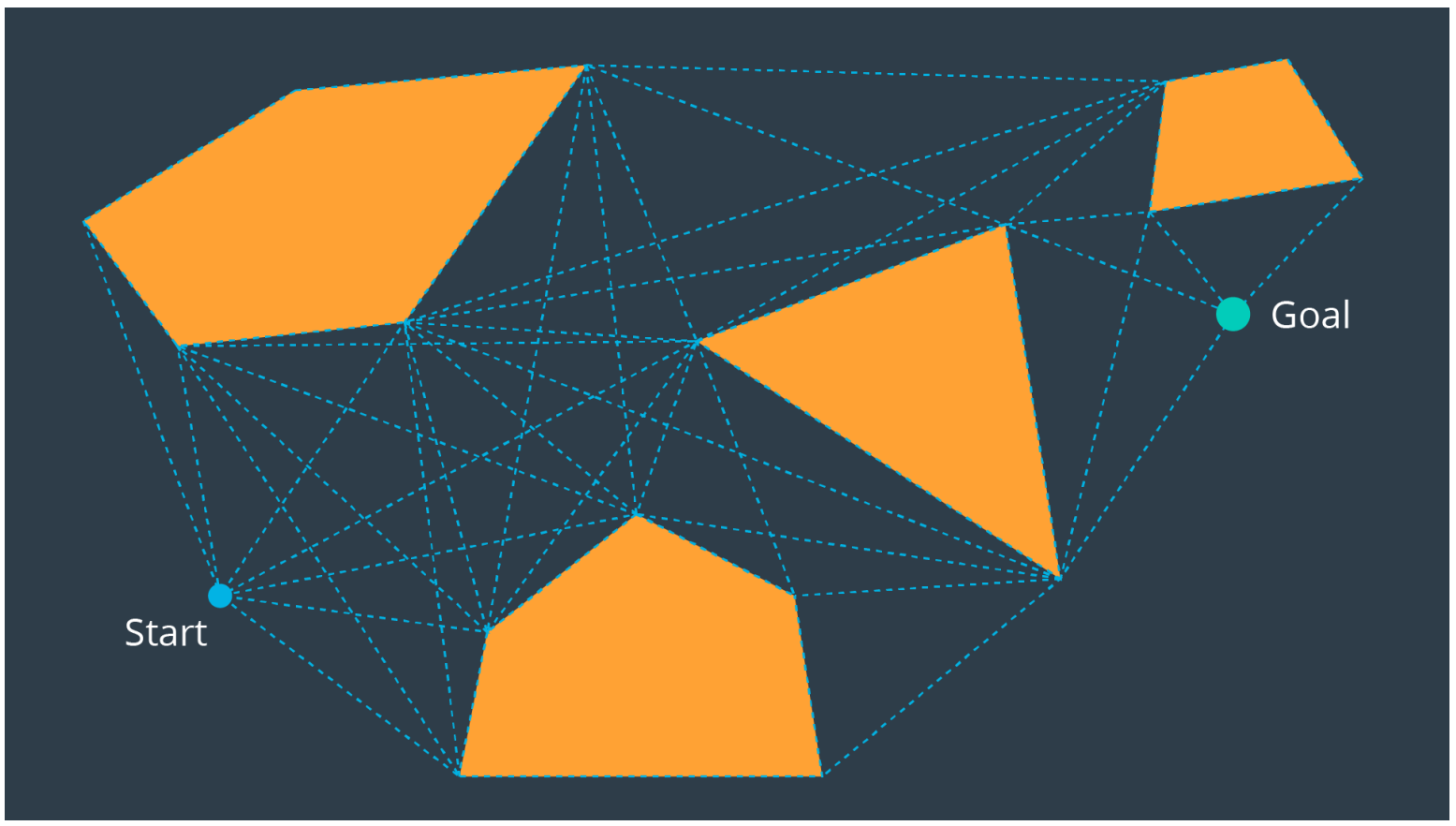
Visibility Graph

The Visibility Graph builds a roadmap by connecting the start node, all of the obstacles' vertices, and goal node to each other - except those that would result in collisions with obstacles. The Visibility Graph has its name for a reason - it connects every node to all other nodes that are 'visible' from its location.

> ***Nodes:*** *Start, Goal, and all obstacle vertices.*

> ***Edges:*** *An edge between two nodes that does not intersect an obstacle, including obstacle edges.*

The following image illustrates a visibility graph for a configuration space containing polygonal obstacles.

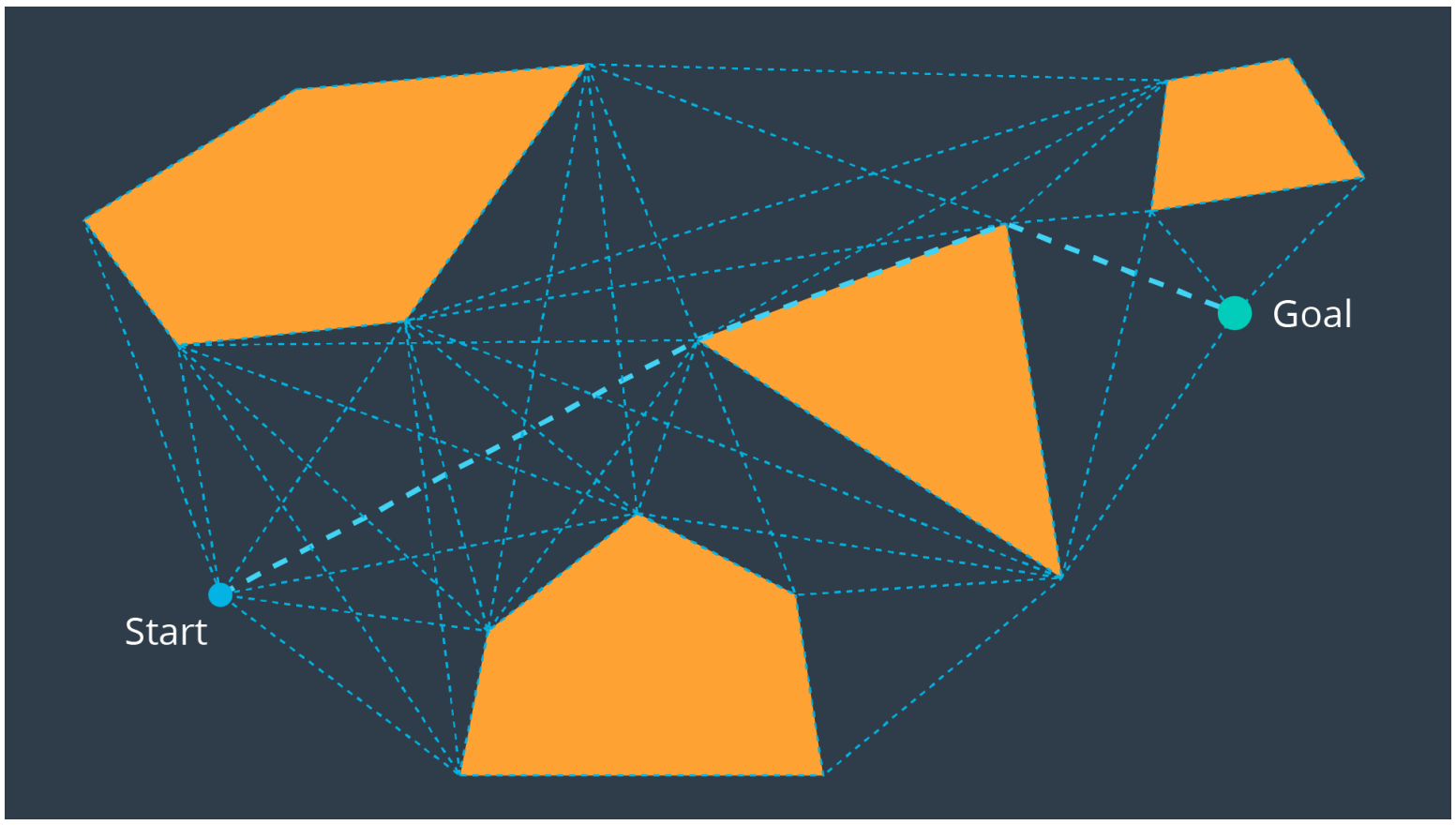


The motivation for building Visibility Graphs is that the shortest path from the start node to the goal node will be a piecewise linear path that bends only at the obstacles' vertices. This makes sense intuitively - the path would want to hug the obstacles' corners as tightly as possible, as not to add any additional length.

Once the Visibility Graph is built, a search algorithm can be applied to find the shortest path from Start to Goal. The image below displays the shortest path in this visibility graph.

additional length.

Once the Visibility Graph is built, a search algorithm can be applied to find the shortest path from Start to Goal. The image below displays the shortest path in this visibility graph.



Quiz

Although the algorithms used to search the roadmap have not yet been introduced - it is still worth analysing whether *any* algorithm would be able to find a path from start to goal, and whether the optimal path lies within the roadmap.

QUIZ QUESTION

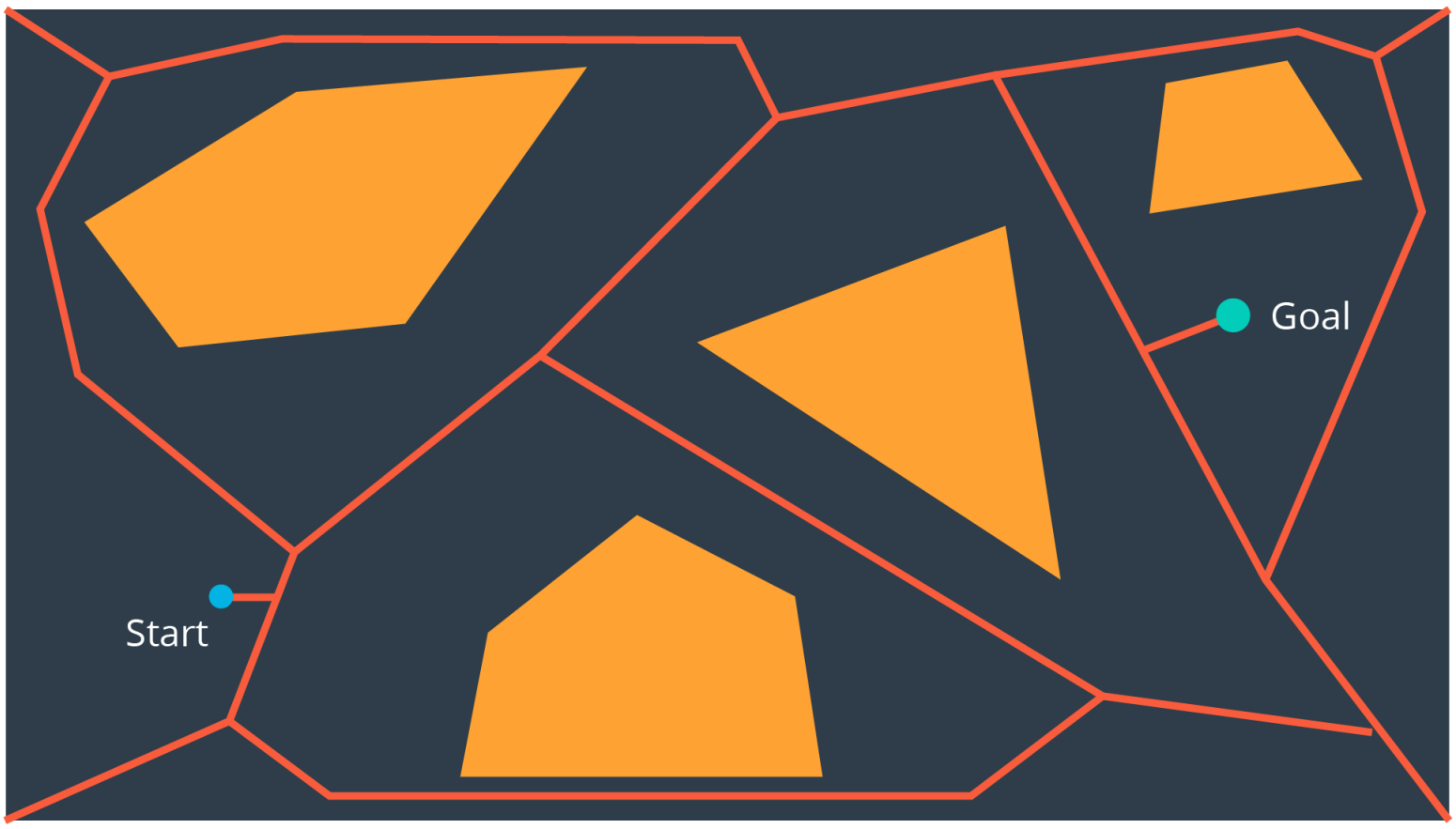
Is the visibility graph complete? Does it contain the optimal path?

- [x] Complete
- [x] Optimal
- [ ] Neither

Voronoi Diagram

Another discretization method that generates a roadmap is called the Voronoi Diagram. Unlike the visibility graph method which generates the shortest paths, Voronoi Diagrams maximize clearance between obstacles.

A Voronoi Diagram is a graph whose edges bisect the free space in between obstacles. Every edge lies equidistant from each obstacle around it - with the greatest amount of clearance possible. You can see a Voronoi Diagram for our configuration space in the graphic below.



Once a Voronoi Diagram is constructed for a workspace, it can be used for multiple queries. Start and goal nodes can be connected into the graph by constructing the paths from the nodes to the edge closest to each of them.

Every edge will either be a straight line, if it lies between the edges of two obstacles, or it will be a quadratic, if it passes by the vertex of an obstacle.

Quiz

Once again, it is worth investigating - will the roadmap built by the voronoi diagram contain a path from start to goal, and will it contain the optimal path.

QUIZ QUESTION

Is the Voronoi Diagram complete? Does it contain the optimal path?

- [x] Complete
- [ ] Optimal
- [ ] Neither

SUBMIT