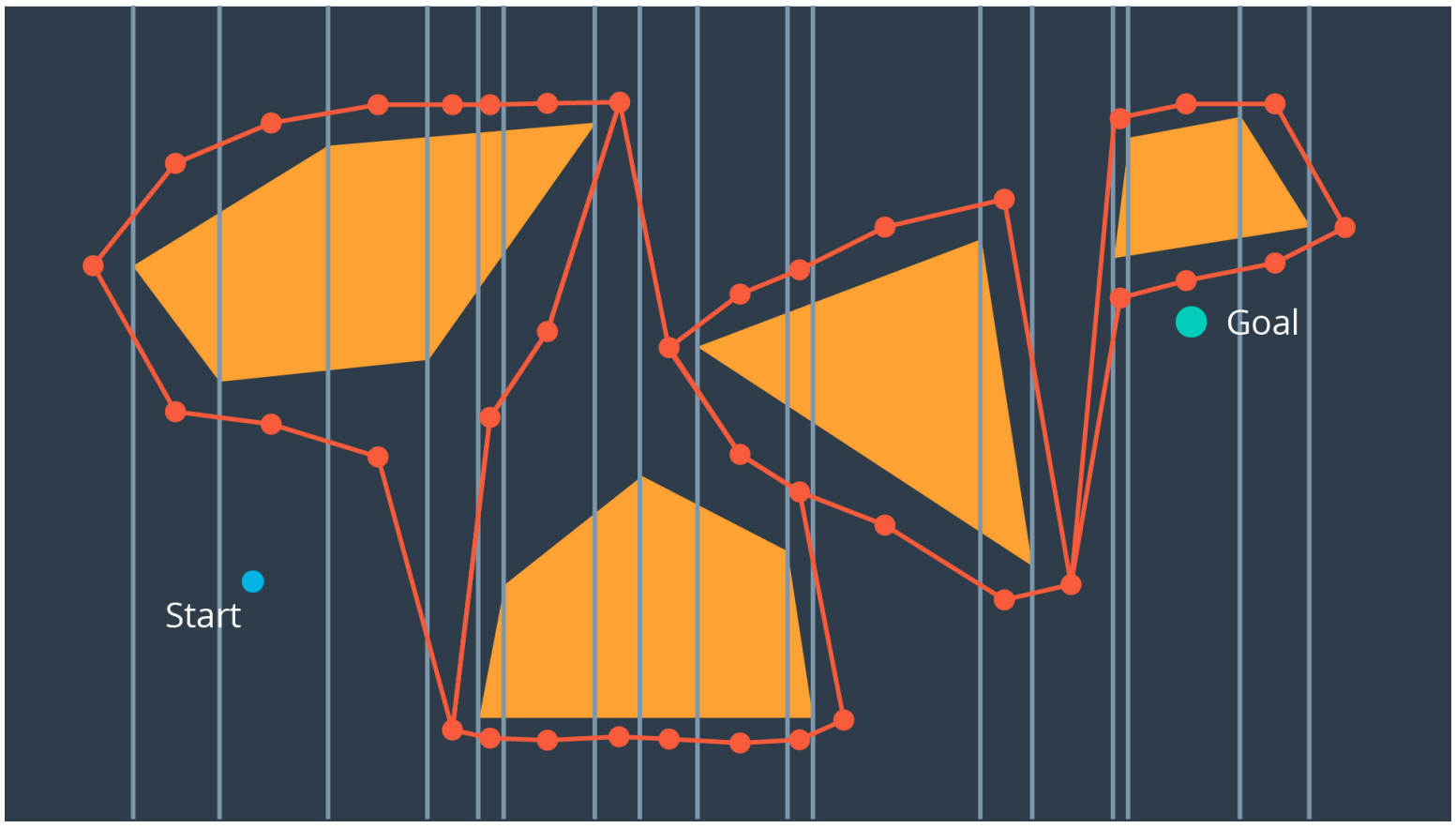
Cell Decomposition

Another discretization method that can be used to convert a configuration space into a representation that can easily be explored by a search algorithm is cell decomposition. Cell decomposition divides the space into discrete cells, where each cell is a node and adjacent cells are connected with edges. There are two distinct types of cell decomposition:

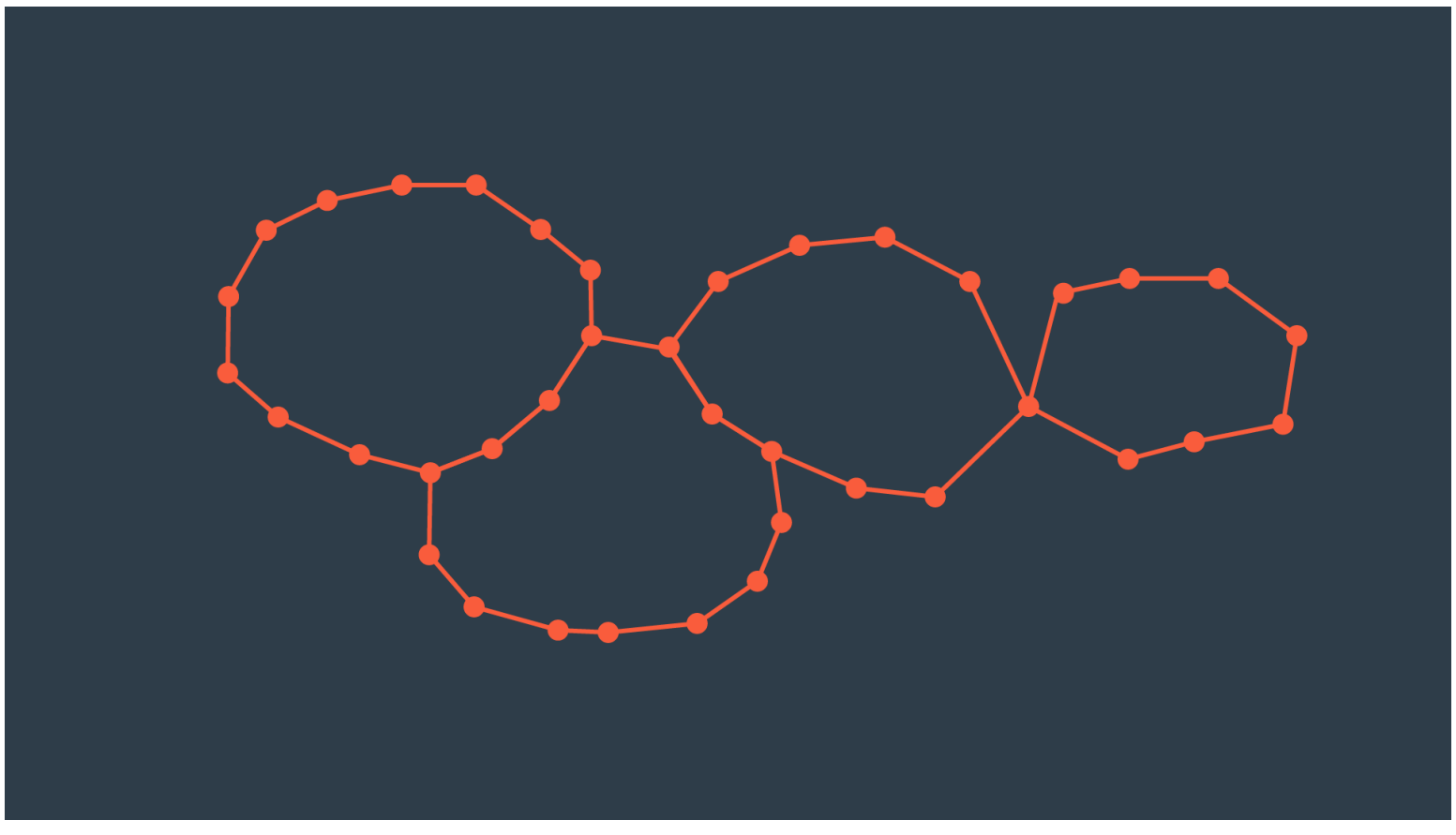
- Exact Cell Decomposition
- Approximate Cell Decomposition.

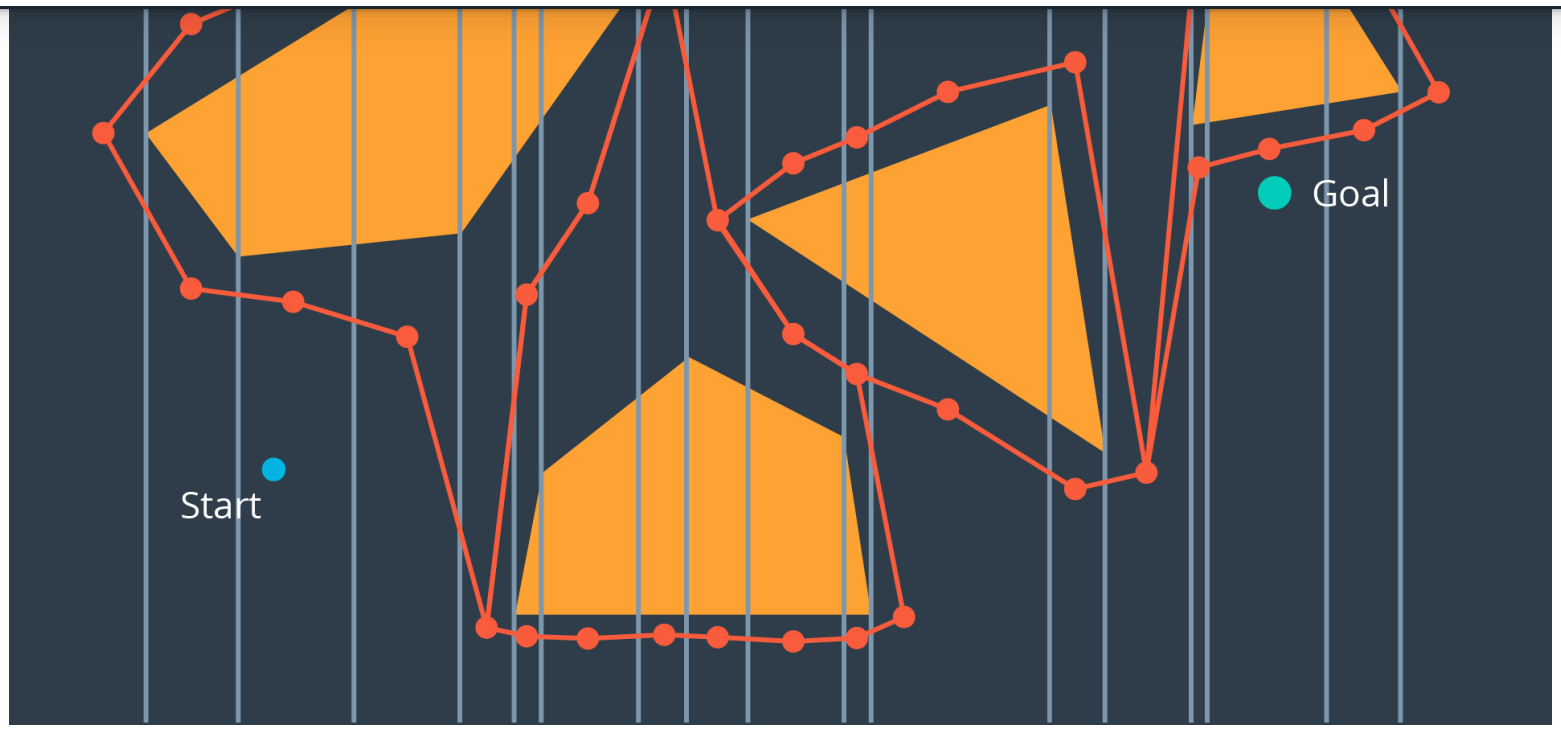
Exact Cell Decomposition

Exact cell decomposition divides the space into *non-overlapping* cells. This is commonly done by breaking up the space into triangles and trapezoids, which can be accomplished by adding vertical line segments at every obstacle's vertex. You can see the result of exact cell decomposition of a configuration space in the image below.

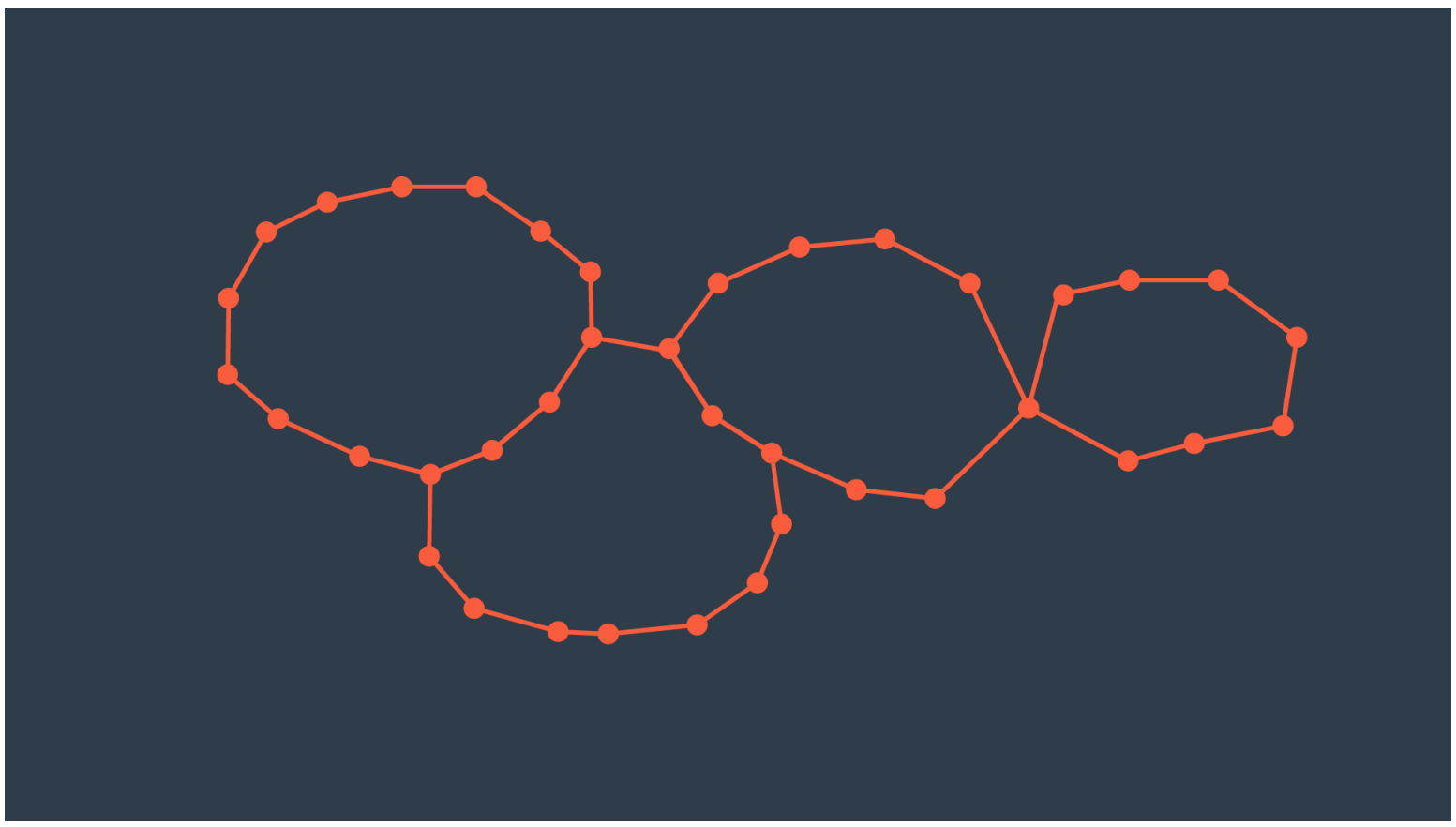


Once a space has been decomposed, the resultant graph can be used to search for the shortest path from start to goal. The resultant graph can be seen in the image below.





Once a space has been decomposed, the resultant graph can be used to search for the shortest path from start to goal. The resultant graph can be seen in the image below.



Exact cell decomposition is elegant because of its precision and completeness. Every cell is either 'full', meaning it is completely occupied by an obstacle, or it is 'empty', meaning it is free. And the union of all cells exactly represents the configuration space. If a path exists from start to goal, the resultant graph *will* contain it.

To implement exact cell decomposition, the algorithm must order all obstacle vertices along the x-axis, and then for every vertex determine whether a new cell must be created or whether two cells should be merged together. Such an algorithm is called the Plane Sweep algorithm.

Exact cell decomposition results in cells of awkward shapes. Collections of uniquely-shaped trapezoids and triangles are more difficult to work with than a regular rectangular grid. This results in an added computational complexity, especially for environments with greater numbers of dimensions. It is also difficult to compute the decomposition when obstacles are not polygonal, but of an irregular shape.

For this reason, there is an alternate type of cell decomposition, that is much more practical in its implementation.

NEXT

Approximate Cell Decomposition

Approximate cell decomposition divides a configuration space into discrete cells of simple, regular shapes - such as rectangles and squares (or their multidimensional equivalents). Aside from simplifying the computation of the cells, this method also supports hierarchical decomposition of space (more on this below).

Simple Decomposition

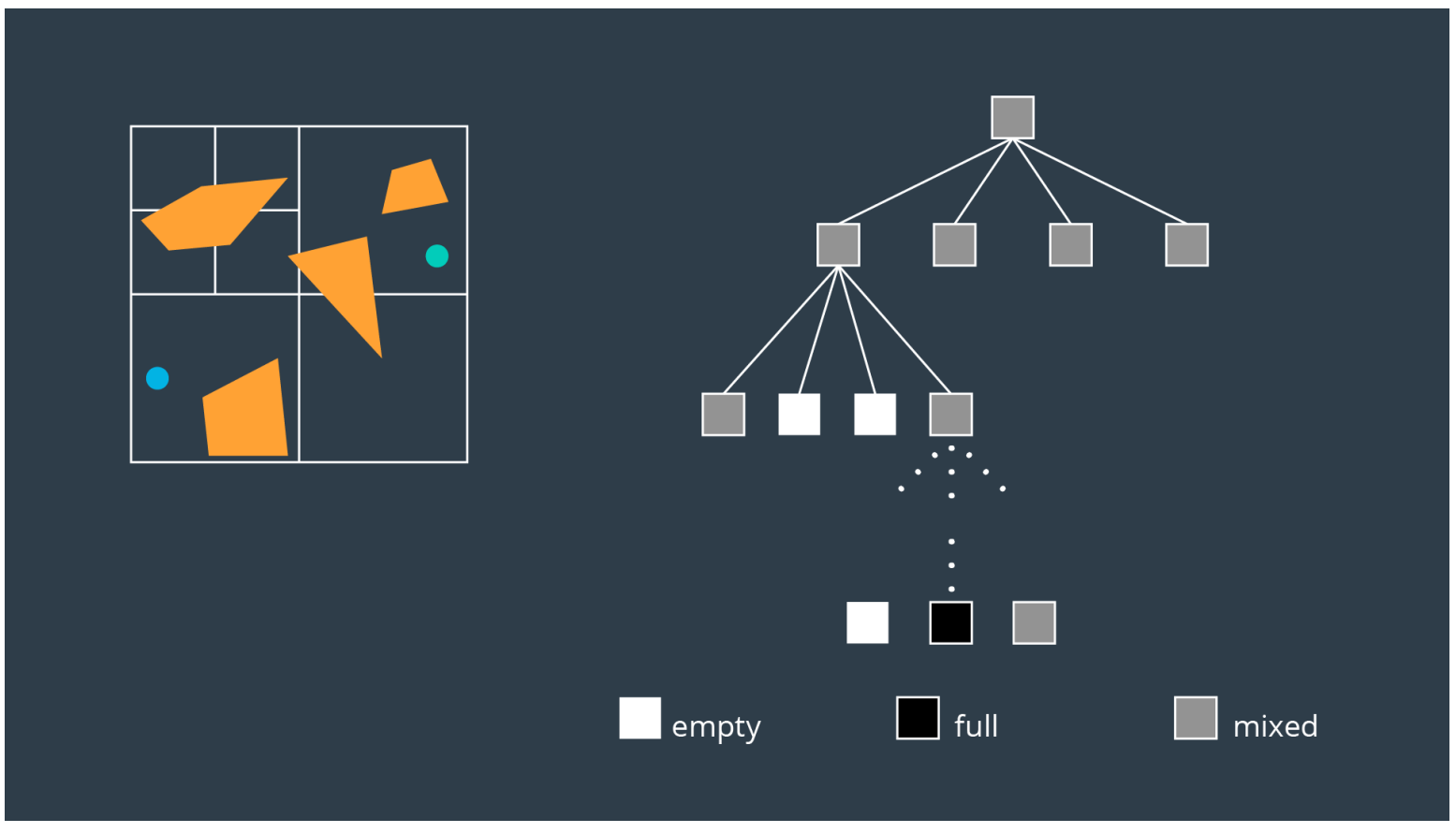
A 2-dimensional configuration space can be decomposed into a grid of rectangular cells. Then, each cell could be marked full or empty, as before. A search algorithm can then look for a sequence of free cells to connect the start node to the goal node.

Such a method is more efficient than exact cell decomposition, but it loses its completeness. It is possible that a particular configuration space contains a feasible path, but the resolution of the cells results in some of the cells encompassing the path to be marked 'full' due to the presence of obstacles. A cell will be marked 'full' whether 99% of the space is occupied by an obstacle or a mere 1%. Evidently, this is not practical.

Iterative Decomposition

An alternate method of partitioning a space into simple cells exists. Instead of immediately decomposing the space into *small* cells of equal size, the method *recursively* decomposes a space into four quadrants. Each quadrant is marked full, empty, or a new label called 'mixed' - used to represent cells that are somewhat occupied by an obstacle, but also contain some free space. If a sequence of free cells cannot be found from start to goal, then the mixed cells will be further decomposed into another four quadrants. Through this process, more free cells will emerge, eventually revealing a path if one exists.

The 2-dimensional implementation of this method is called quadtree decomposition. It can be seen in the graphic below.

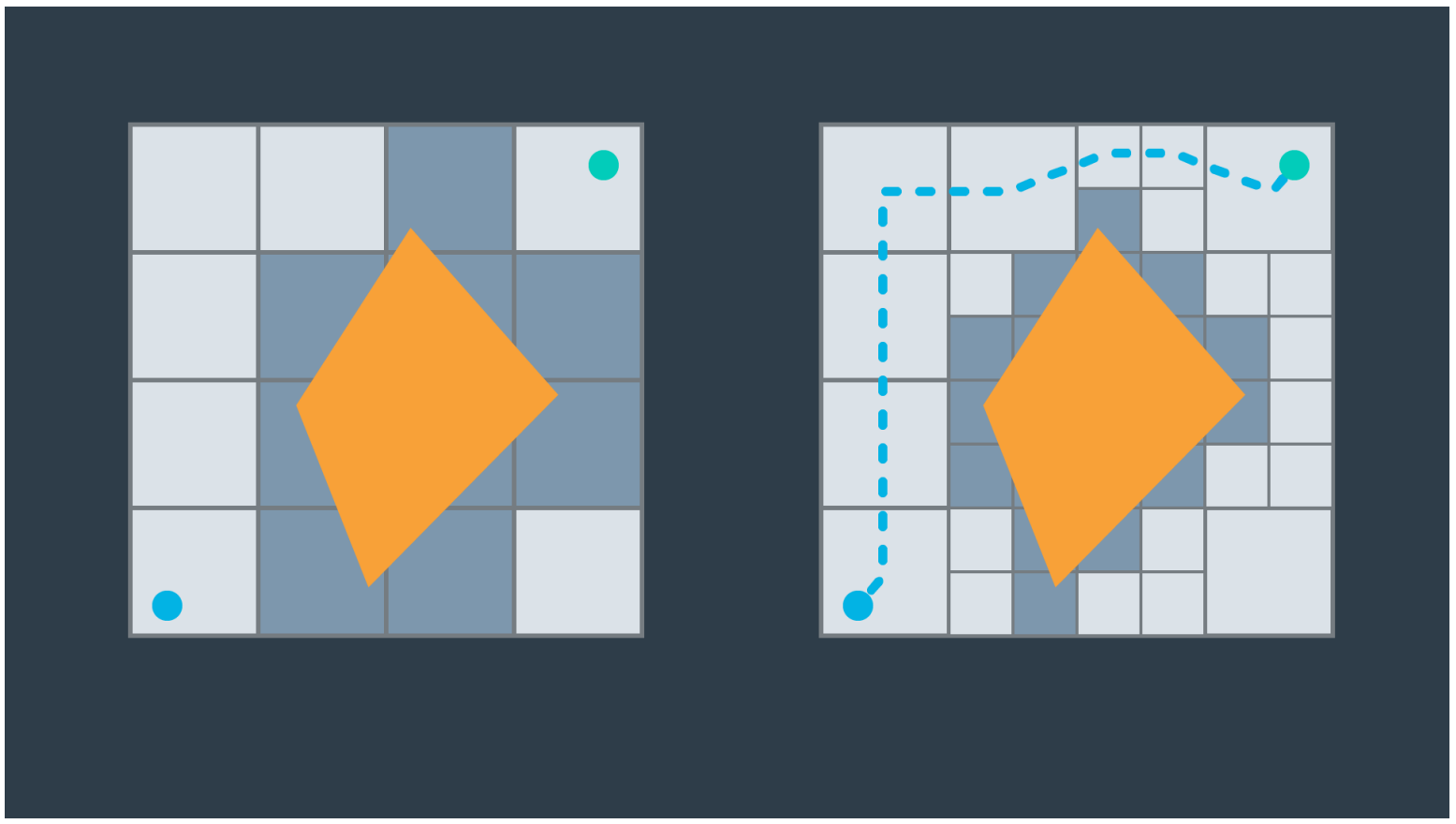


Algorithm

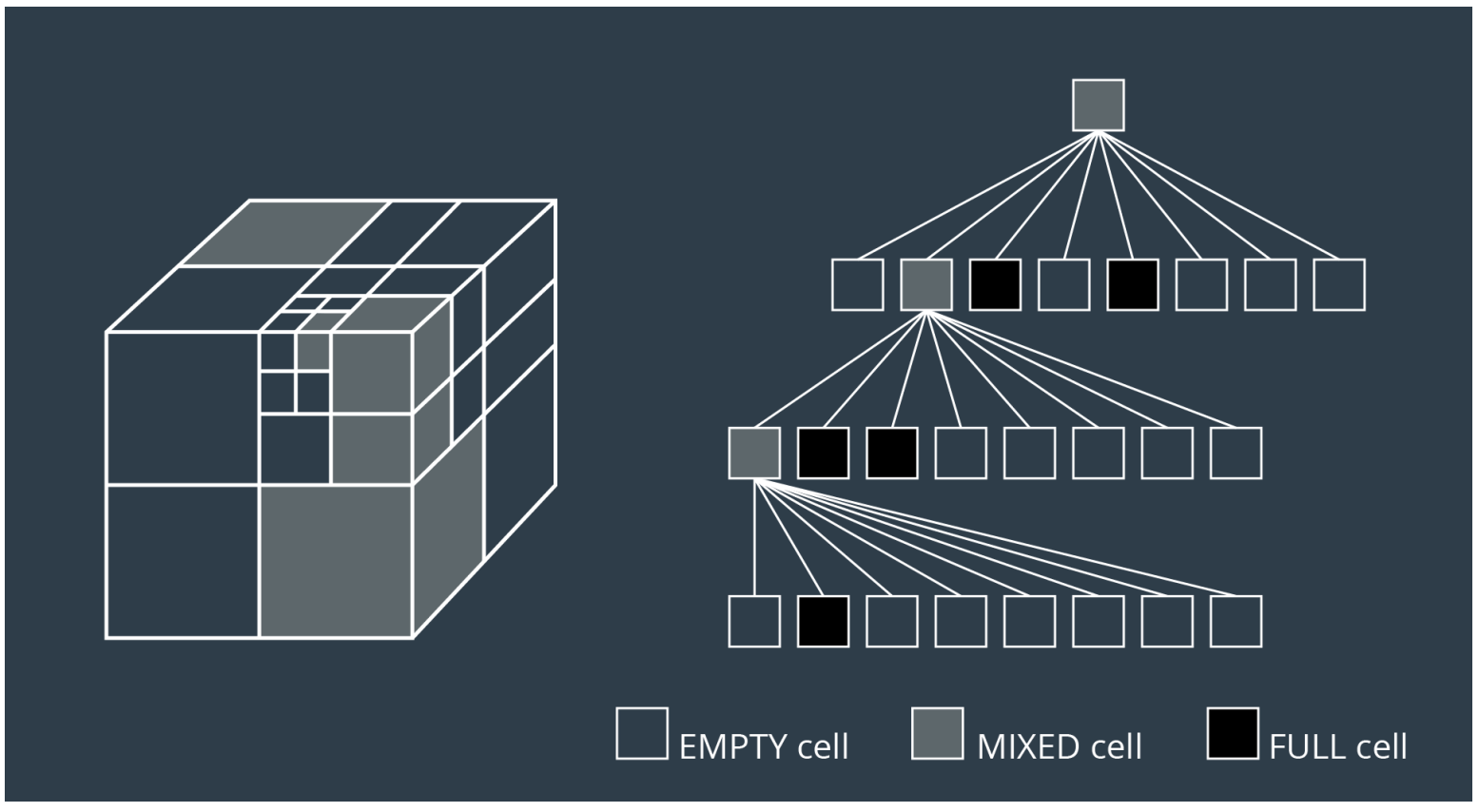
The algorithm behind approximate cell decomposition is much simpler than the exact cell decomposition algorithm. The pseudocode for the algorithm is provided below.

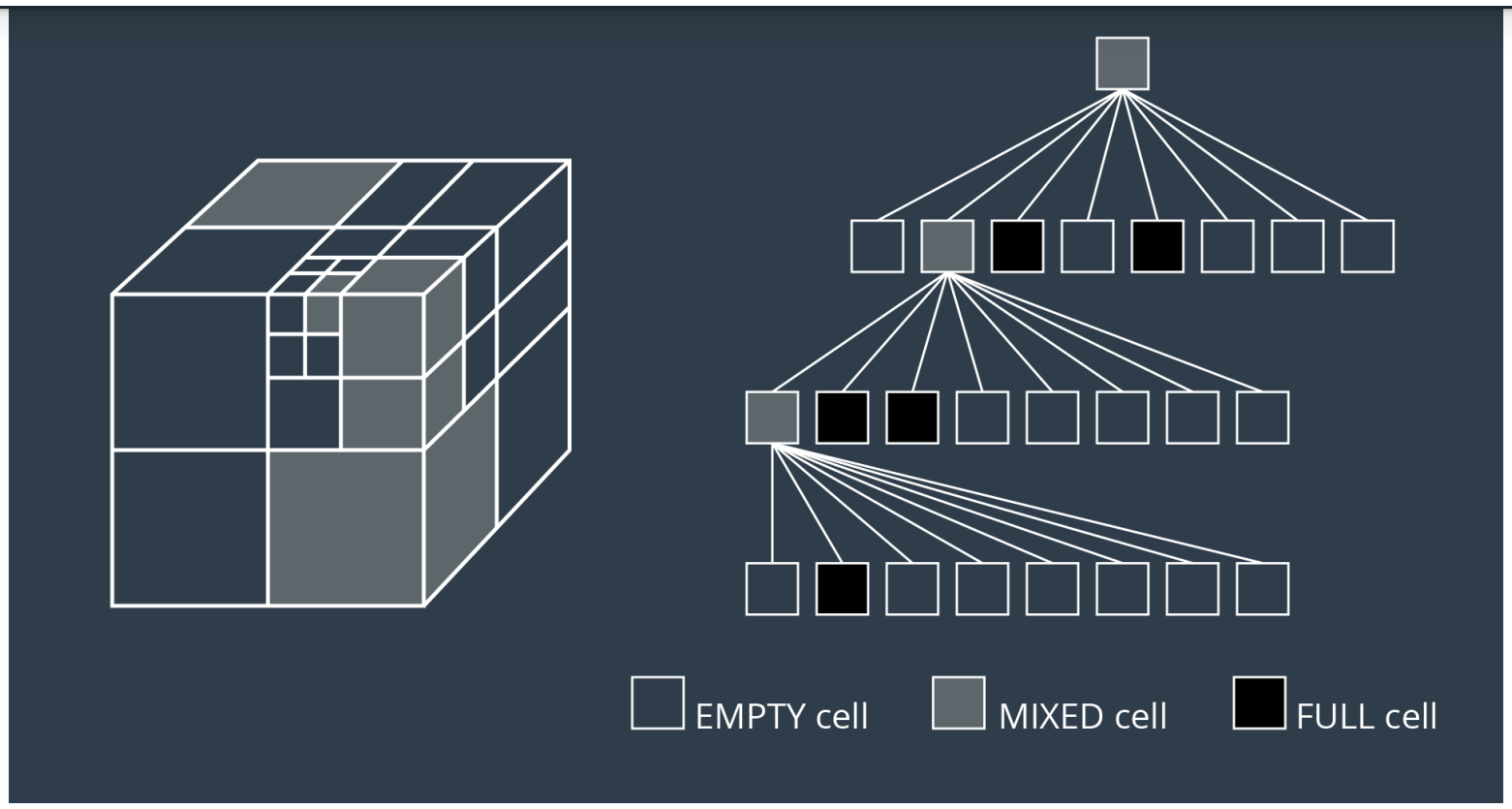
decomposition algorithm. The pseudocode for the algorithm is provided below:

> *Decompose the configuration space into four cells, label cells free, mixed, or full.*
>
> *Search for a sequence of free cells that connect the start node to the goal node.*
>
> *If such a sequence exists:*
>
> *Return path*
>
> *Else:*
>
> *Further decompose the mixed cells*



The three dimensional equivalent of quadtrees are octrees, depicted in the image below. The method of discretizing a space with any number of dimensions follows the same procedure as the algorithm described above, but expanded to accommodate the additional dimensions.





Although exact cell decomposition is a more elegant method, it is much more computationally expensive than approximate cell decomposition for non-trivial environments. For this reason,

approximate cell decomposition is commonly used in practice.

With enough computation, approximate cell decomposition approaches completeness. However, it is not optimal - the resultant path depends on how cells are decomposed. Approximate cell decomposition finds the obvious solution quickly. It is possible that the optimal path squeezes through a minuscule opening between obstacles, but the resultant path takes a much longer route through wide open spaces - one that the recursively-decomposing algorithms would find first.

Approximate cell decomposition is functional, but like all discrete/combinatorial path planning methods - it starts to be computationally intractable for use with high-dimensional environments.

Quiz

QUIZ QUESTION

Which of the following statements are true about cell decomposition?

- [x] In practice, approximate cell decomposition is preferred due to its more manageable computation.
- [x] Approximate cell decomposition is not optimal because obvious (wide/open) paths are found first.
- [ ] The optimality of exact cell decomposition depends on how the configuration space is decomposed.
- [x] The quadtree and octree methods recursively decompose mixed cells until they find a sequence of free cells from start to goal.

SUBMIT