Hybrid A* Algorithm

The Hybrid A* algorithm is a powerful path planning approach that combines the benefits of A* search in continuous space with a discretized set of headings. It enables the generation of efficient and smooth paths for nonholonomic vehicles navigating complex environments.

The basic A* algorithm is considered a traditional search method, it has been widely used in the field of robot path planning (manipulator and mobile robot). This algorithm has many features that give it superiority over other methods. On the other hand, the Basic A* suffers from a bad convergence rate (it takes a long time until finding the goal point), especially in environments with cluttered obstacles, or whose targets are located in narrow passages. There is an improved method called (Hybrid A*) to overcome the shortcomings of the original A*, specifically slow convergence and cost rate.

Example of the algorithm:

```
#ifndef ALGORITHM_H

#define ALGORITHM_H

#include <ompl/base/spaces/ReedsSheppStateSpace.h>

#include <ompl/base/spaces/DubinsStateSpace.h>

#include <ompl/base/spaces/SE2StateSpace.h>

#include <ompl/base/State.h>

typedef ompl::base::SE2StateSpace::StateType State;

#include "node3d.h"

#include "node2d.h"

#include "visualize.h"

#include "collisiondetection.h"

namespace HybridAStar {

class Node3D;

class Node2D;

class Visualize;

/*!

 * \brief A class that encompasses the functions central to the search.
```

```
 */
class Algorithm {
 public:
  /// The deault constructor
  Algorithm() {}
  // HYBRID A* ALGORITHM
  /*!
     \brief The heart of the planner, the main algorithm starting the search for a collision free and drivable path.
     \param start the start pose
     \param goal the goal pose
     \param nodes3D the array of 3D nodes representing the configuration space C in R^3
     \param nodes2D the array of 2D nodes representing the configuration space C in R^2
     \param width the width of the grid in number of cells
     \param height the height of the grid in number of cells
     \param configurationSpace the lookup of configurations and their spatial occupancy enumeration
     \param dubinsLookup the lookup of analytical solutions (Dubin's paths)
     \param visualization the visualization object publishing the search to RViz
     \return the pointer to the node satisfying the goal condition
  */
  static Node3D* hybridAStar(Node3D& start,
                             const Node3D& goal,
                             Node3D* nodes3D,
                             Node2D* nodes2D,
                             int width,
                             int height,
                             CollisionDetection& configurationSpace,
```

```
                    float* dubinsLookup,
                    Visualize& visualization);

};
}
#endif // ALGORITHM_H
```

This code is a central part of a hybrid A* pathfinding implementation that leverages both 2D and 3D nodes to find a collision-free, drivable path for a robot or vehicle in a grid. It also interacts with RViz for visualization, which is common in robotic systems that use the Robot Operating System (ROS).