Mid Term Answer

1. For this vehicle, the motion will be easier to plane in the c-space. Thus, the topology becomes an S2, that is controlling only the two angles of the agent. The system is them reduced to a 2d plane.

Explanation: The aerial vehicle shows in this problem as 3 coordinates and 2 rotational angle s. Thus, the workspace of the will have a topology od R3S2. It would be difficult and required a lot of computer resources to plane the motion in this space as we have about 5 parameters to tweak.

However, we the equations given, the controller inputs, the constraint, with careful examination; one can see that the system can be controlled using the two angles of the rotation. In fact, the input the system is the rate of change of the angles and the acceleration. One can reverse calculate the coordinate of the agent at every sampled angle and check if it is obstacle free.



|  |  |  |
| --- | --- | --- |
| Planning Algorithms | Advantages | Disadvantages |
| Gradient descent planner with a potential function | * Can find the path to goal with no local minimum and one global minimum * Fast * Can allow control over controller input | * Local minimum * Invalid point: the next point could be invalid due to constraints set. * Implementation if been done in c-space not the workspace. * Unable to control acceleration. |
| Wave-front planner | * Guaranteed to find path if it exists * Easy to implement * Can be implemented in the chosen space (c-space) | * Little room to manipulate controller input because points are pre-sampled * Will need path smoothing * Cannot control controller input |
| Probabilistic roadmap planner | * find path if it exists * Easy to implement * Can be implemented in the chosen space (c-space) | * Can become computationally intensive * Is probabilistic complete * Could lead to many invalid samples thus not path due to constraint if sample point is too sparse * Will need path smoothing * Can not control input to controller. |
| Randomized tree-based planner | * Ability to control or tweak controller inputs * Could be fast depending on the version chosen * Will find path if it exists * Might not need path smoothing | * Could take too long to complete depending on the planning flavor. * Can be computationally intensive. |

1. Pseudocode RRT\_Goal\_Bias
2. Using the chosen algorithm, RRT goal bias, we drone can find the path to goal if it existed given enough time. This is guaranteed because we will sample some data point near goal at x% of time. Thus, if the algorithm is given enough time I should converge to goal.
3. With RRT Goal Bias, we sample a point in the space of the agent then we then choose a new P\_new sample that is a in a radius r that we define along the closes point in our graph valid points. With this approach we can have control over the chosen P\_new. Thus, for the point p\_new, the team can optimize the controller input that allow to get the best point a.k.a best edge. The team can then construct the graph with points that are optimal. We can define each point with more than one optimal parameter.

Fr example if each point has the following structure

Struct edge {

tuple p1;

tuple p2;

float time;

float distance;

float other\_parm;

}

The team can then find best path using for example time or distance of each edge.