Project name:- FCND-Motion-Planning
Submitted by:- Bishwajit Kumar

Explain the starter code:-
Comparison between motion_planning.py and backyard_flyer_solution.py

Motion_planning.py	Backyard_flyer_solution.py
Import statements	
Extra:-Import functions from planning_utils.py where A star and Other functions are implemented, Imports auto and argparse as well	Imports basic udacidrone api's
States class	
Assigns auto instances for different values in State class	Uses predefined values
Difference in defined functions	
Extra send_waypoints and plan_path methods	
Also uses methods defined in planning_utils.py file	
local_position_callback method	
No calculate box,uses waypoint_transition	Uses calculate_box to move drone in a rectanglular shape
velocity_callback	
Same in both the files	Same in both the files
state_callback	
Added Planning state	
arming_transition	
No call to set_home_position	
Order change between arm and take_control	
takeoff_transition	
Target altitude not set inside this method	
Removes target position from this method	
waypoint_transition	
Uses heading parameter(target_position[3]) in self.cmd_position instead of 0	
landing_transition	
Same except order change in assigning values	Same
disarming_transition	
Same except order change in assigning values	Same
manual_transition	
Same except order change in assigning values	Same
send_waypoint	
New method introduced to send waypoints to simulator	
plan_path	
Main method introduced used to plan complete path. Described in below sections	

Safety distance is set to 5 and target altitude to 30.	

Implementing Your Path Planning Algorithm:-

1. Read lat,lon value from colliders.csv file and set global_home.
   Used lstrip() method to remove leading whitespace.
2. Retrieve current global position using (self._longitude, self._latitude, self._altitude)
3. Convert to current local position using global_to_local method(In NED format).
4. Set start position to current position relative to map center using local position found in step 3.(Convert it to int)
5. Set goal to random geotic coordinates (used -122.399612, 37.795933) and then used global_to_local method to convert it to local map frame.
6. Called A* method provided in planning_utils.py where added diagonal motion with a cost of sqrt(2).Modified Action class along with valid_actions method in planning_utils.py.
7. Implemented bresenham to prune path(reduce waypoints) in planning_utils.py
8. Passed the new path to waypoints which calculates and executes the same.

Executing the flight:-

Does it work? Yes it works for values provided from the map.
Check video named Motion_Planning.mov in the same folder