Multi-T-RRT\* Algorithm for the Path Planning of safe UAS operations

% Last Updated: 15th March 2019

%

% \*Author:\* Uluhan Cem Kaya

%

% \*Description:\* This program implements the Multi Tree variant of

% Transition based optimal Rapidly-Exploring Random Trees (RRT\*).

% Path planning is defined on a real 2D map scenario where the building

% footprints are used to construct the risk map of the area for the risk of

% exposure to the UAS impact (Cost) with Gaussian distributions. A Utility

% based cost function is developed to measure the path quality and used to

% generate Forward and Backward tree branches originating from specified

% start and goal locations. Algorithm runs until the maximum iteration

% number is reached, then, outputs all the successful connections of trees

% selecting the Final Path with highest utility.

%

% \*Features:\*

% - Multiple Trees (1 Forward and multiple Backward Trees)

% - Heruistics: Goal Bias and Transition Test (rejection sampling)

% - Obstacle Avoidance (polygonal obstacles with circular avoid radii)

% - Utility Maximization.

% (min UAS impact risk/max UAS operation utility)

%

% \*To Do List:\*

% - Faster Near and Nearest search are needed.

% (current implementation of KDTrees are slow - the bottleneck)

% - Adaptive Sampling Heuristics instead of uniform sampling

% - Transition Test requires a fine initial tuning. There should be

% a way of making it more generic.

% - Object Oriented Programming can be more efficient and cleaner

% (passing all the parameters into many funtions are unnecessary and

% inefficient even though they are under a main struct)

% - Potential places that requires optimization of the method and code:

% \Utility Calculation: Large matrix operations PREM and PURM

% \Rewiring: unsorted rewire trials may exp. increase

%

% \*Usage:\* Use TESTRun.m script to load and pass the required parameters

% that are defined in ParameterFile.m.

% Check ParameterFile.m for details of individual parameters.

% Load the Parameters for the Path Planning to pass the RRT

% algorithm. Dot (.) notation of the struct can be used here to  
 modify the desired parameters. Note that this sample run

% is using a specific scenario map with building footprints. To generate

% different scenario maps, new footprint.shp file are required from GIS database and buildingFootprints\_GIS.m file needs to be updated accordingly.

\*Function Hierarchy:

* TESTRun  
  %%%%%%%%%%%%%%%%%%%% Define Parameters %%%%%%%%%%%%%%%%%%%%%
  + - ParameterFile

%%%%%%%%%%% PREM Construction %%%%%%

* + - * buildingFootprints\_GIS

%%%%%%%%%%% Plot PREM %%%%%%%%%%%%%%

* + - * PREMplot

%%%%%%%%%%% Construct Obstacles %%%%

* + - * polygonalObstacles

%%%%%%%%%%%%%%%%%%%% Main Function %%%%%%%%%%%%%%%%%%%%%%

* + - Multi\_T\_RRTstar
      * Initialize\_Tree

%%%%%%%%%%% Forward Tree %%%%%%%%%%%

* + - * ChooseTarget\_Multi
      * ExtendAndRewire\_Forward
        + Extend\_1

collCheck

UtilitySegment\_Forward

* + - * + RewireNodes\_Forward

Extend\_1

ExtendMultiple\_Fwd

Extend\_1

* + - * ConnectionCheck\_MultiTree
      * CreatePath
        + bidirectionalBacktrack

%%%%%%%%%%%% Backward Trees %%%%%%%%%%%

* + - * ChooseTarget\_Multi
      * ExtendAndRewire\_Backwards
        + Extend\_1

collCheck

UtilitySegment\_Backwards

* + - * + RewireNodes\_Forward

Extend\_1

ExtendMultiple\_Fwd

Extend\_1

* + - * ConnectionCheck\_MultiTree
      * CreatePath
        + bidirectionalBacktrack
      * CheckPrevPaths

%%%%%%%%%%%%%%%%%%%% Return Main Loop %%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%% End of Run %%%%%%%%%%%%%%%%%%%%%%%%%

PARAMETERS Description:

% \*Description:\* This file defines the Parameters for the Path Planning

% problem. These parameters are divided into 9 categories. List of

% categories and the short description is below. At the end of the file,

% all the parameters are put under a main struct (AllParam) for the

% accessability. For more detail, check the comments on the individual parameters.

%

% \Categories:

% - RRTParam:

% (RRT specific and planning parameters are grouped

% together. i.e. Step Size, maximum iteration, display

% tree branches, start/goal configurations etc.)

%

% - GridMapParam:

% (Resolution of the Cost Mapping, grid representation

% parameters, planning map sizes, and geo-references

% are under this category)

%

% - HeuristicParam:

% (Heuristics for randomized sampling and planning

% parameters such as Goal Biases, Transition Test

% temperatures, RRT\* rewiring parameters, and sample

% distribution methods)

%

% - PREM:

% (Composed of Probabilistic Risk Exposure Map layers,

% and their relative weight factors. Map is represented

% as a 2 dimensional Matrix created by the meshgrid.)

%

% - PURM:

% (Originally defined as Probabilistic UAS Reachability

% Map, and it is intended to group the parameters related

% to UAS Failure and Reachability parameters. It includes

% Failure Rates and impact zone parameters where the

% impact zones are represented as elliptical regions.)

%

% - UtilityThreshold:

% (Utility thresholds are defined for the Planning to

% eliminate some of the paths according to specified

% thresholds. These thresholds can be selected as maximum

% length of a path, maximum risk exposure of a node,

% or minimum cumulative utility along the path.)

%

% - obstacles:

% (obstacle parameters are saved in this struct. Here,

% the obstacles are assumed to be polygonal, and given

% the corners of a polygon, centroid of the polygon and

% the circle enclosing all the corners are computed and

% saved in the struct.)

%

% - Constraints:

% (Vehicle specific constraints such as kinematic/dynamic

% constraints, i.e. maximum yaw rate, acceleration, speed,

% and also the path planning solution constraints such as

% maximum delta\_t or the step size of the solution.

%

% - figHandle:

% (Figure and Solution of the Path figures are saved as

% figure handles during the iterations. It is used to

% visualize the solutions during the process.)

%

%

% NOTE: As mentioned before, all these categories are put under a main

% struct to pass around easily. Therefore, using a dot (.) notation,

% they could be easily accessed and modified.

%

% Exm: AllParam.RRTParam.iterMax = 10000;

%