AIAA SciTech Forum
7-11 January 2019, San Diego, California
AIAA Scitech 2019 Forum

10.2514/6.2019-0958

Real-Time Path Planning for Multi-copters flying in UTM -TCL4

Anjan Chakrabarty[*], Vahram Stepanyan[†], Kalmanje Krishnakumar[‡]

& Corey Ippolito[§]

NASA Ames Research Center, Moffett Field, CA 94035

NASA's UAS Traffic management (UTM) -TCL-4 focuses on safely enabling large scale small UAS operations in low altitude airspace in dense urban environment. This paper presents an operational architecture of an autonomous unmanned aerial vehicle operating in TCL4 . An on-line path planning scheme is proposed which can effectively plan for feasible paths in real time with TCL-4 constraints. The real time path planner avoids other obstacles and other UAVs flying in the shared airspace. An end-to-end system is designed and tested in high fidelity Reflection simulation architecture which demonstrates the feasibility of the approach.

Keywords: UTM, Autonomy, Path planning, BVLS

I. Introduction

The need for Unmanned Aerial Vehicles(UAVs) for different commercial applications has been realized by different industries like package delivery,[1] inspection,[2] security[3] and disaster management.[4] More and more vehicles are being built for complete autonomous operations which can operate beyond visual line of sight (BVLOS).[5] The industry is in the midst of a transitional phase where active research is pushing UAVs to be used in many commercial applications. A rapid rise of autonomous UAVs will bring a sea change to the industry. As we see a rapid growth in UAV technology the need to regulate them becomes increasingly important. Since these vehicles shares the same airspace with their bigger counterparts it is absolutely imperative to include small UASs in the National Airspace System (NAS).

Current airspace management systems are looking to incorporate high density of UASs traffic, operating BVLOS and sharing the airspace with larger aircraft. NASA's UAS Traffic Management (UTM) research initiative is being designed precisely for handling this kind of operations.[6] NASA has taken a series of activities under the UTM umbrella increasing in complexity to address the issue called "Technology Capability Levels (TCL)". Starting with operations in sparsely populated rural environments the complexity of operations in dense urban regions is being tacked in each increasing TCL. For details see NASA UTM website. The current progress made under different national air campaigns with different technological partners is described in.[7] It describes the series of flight tests concluded recently under TCL-3 in sparsely populated urban regions.

UTM TCL4 will focus on UAS operations in higher density urban areas.[8] Higher desnisty urban operations include operations in high population density areas, as well as multiple unmanned vehicles operating and sharing the same airspace. Low altitude autonomous operation of small UASs in populated urban areas will require some fundamental technological as well as infrastructural developments. Autonomous operations of UAS in urban environments requires guaranteed safe and orderly operations of UASs. UAV operations must be contained within approved flight boundaries and within stipulated time. Only highly autonomous UAS will be able to accomplish this task in a safe and efficient manner.

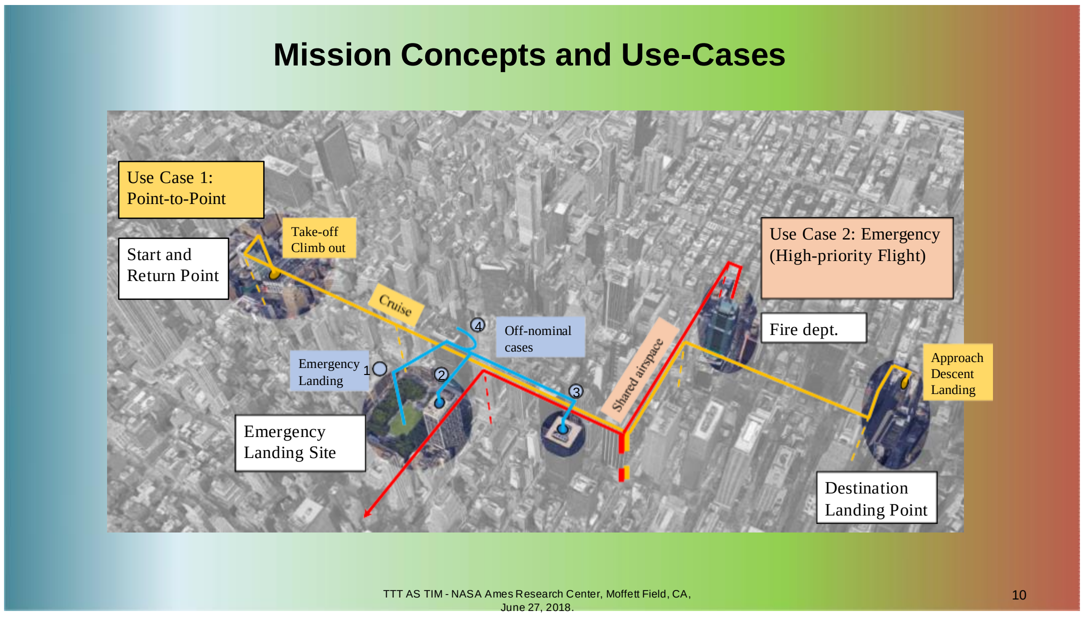
A range of challenges are encountered by UAVs operating in city environment. Unmanned aircraft system navigation in urban environments requires handling of static as well as dynamics objects. Dynamic objects can be other co-operating UAVs operating in the same environment, or other non-co-operative objects. The planning system should be able to handle all the different contingency options.

[*]Research Engineer, SGT Inc, NASA Ames Research Center, Moffet Field, CA 94035. AIAA Member. anjan.chakrabarty@nasa.gov

[†]Senior Research Scientist, Universities Space Research Association, NASA Ames Research Center, Moffet Field, CA 94035, AIAA Senior Member

[‡]Research Scientist, NASA Ames Research Center, Moffet Field, CA 94035, AIAA Member

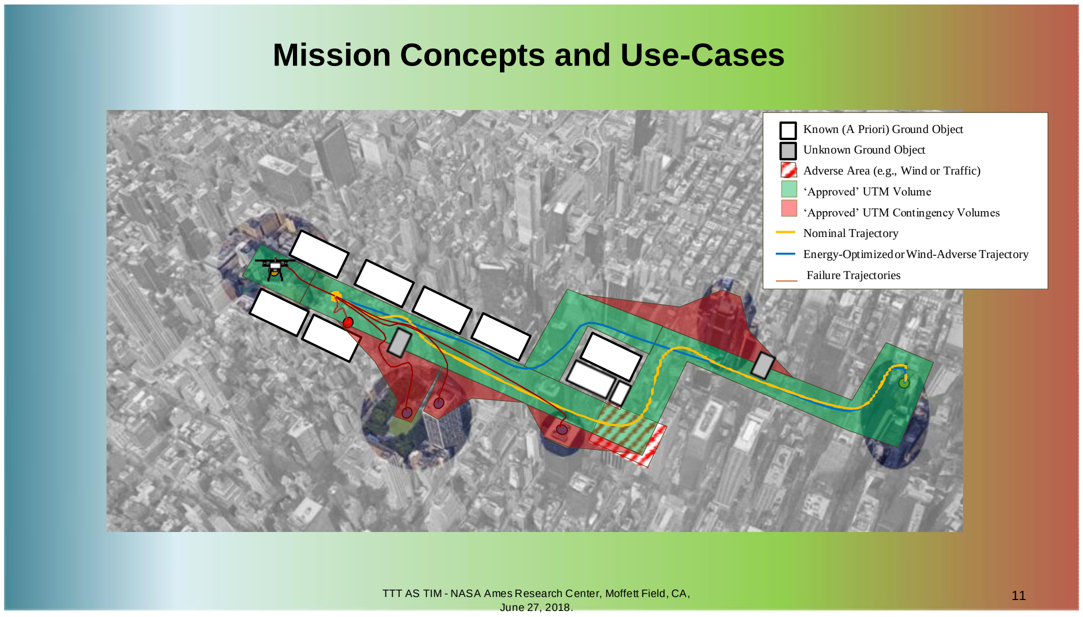
[§]Aerospace Scientist, NASA Ames Research Center, Moffett Field, CA 94035, AIAA Member



Mission Concepts and Use-Cases

TTT AS TIM - NASA Ames Research Center, Moffett Field, CA, June 27, 2018. 10

(a) A point to point UAV operation example in dense urban operations.



Mission Concepts and Use-Cases

Known (A Priori) Ground Object
Unknown Ground Object
Adverse Area (e.g., Wind or Traffic)
'Approved' UTM Volume
'Approved' UTM Contingency Volumes
Nominal Trajectory
Energy-Optimized or Wind-Adverse Trajectory
Failure Trajectories

TTT AS TIM - NASA Ames Research Center, Moffett Field, CA, June 27, 2018. 11

(b) Approved volumes by UTM is designated in Green and shows that it is approved before the flight.

Figure 1. Typical UTM TCL-4 Operation Scenario

Concepts of TCL-4 are under development. Federal Aviation Administratin(FAA), working with NASA has published Unmanned Aircraft Systems Traffic Management (UTM) Concepts and Architecture Overview,[9].[10] The UTM architecture will define "identify services, roles/ responsibilities, information architecture, data exchange protocols, software functions, infrastructure, and performance requirements to enable the management of low-altitude uncontrolled UAS operations",[9].[10]

Several proposal are being considered for conflict mitigation in TCL 4. The NASA SAFE50 study[11] aimed to develop an autonomous UAS for notional last 50 feet of operations. The study aimed to find on-board autonomy requirements for autonomous UAS operating BVOLOS. On-board path planning is one of the main requirements that were identified for autonomous UAS operations. This paper aims to develop the on-board path planning capabilities required for a vehicle to safely operate in UTM.

This paper aims to develop an end to end system which can be implemented within the current framework of UTM operations. Some of the technology developed in this context will enable UAS operations in urban environments. The paper will show in simulation a complete system which will avoid obstacles and other vehicles communication with DSRC or V2V protocol. Another contribution of the paper is to specify the hardware requirements for vehicles operating in TCL4.

A. Motivation

Package delivery[1] is one of the motivating examples for UAS operations in dense urban environments. A basic requirements for package delivery is a point-to-point BVLOS operation. As shown in figure 1a, a typical package delivery flight will consist of take-off from designated launching site, travel to a designated delivery point and return back to the launching site. The vehicle have to encounter known/unknown objects along the way and other vehicles operating in the vicinity. Figure 1a shows the typical setup in an urban environment where the vehicle have to share same airspace with other vehicles (emergency high priority flight) and has designated emegency landing sote.

It is assumed that the vehicle will have to operate inside an UTM approved volume during the whole operation. As shown in 1b the typical TCL-4 operation will have approved volume and contingency volumes.The contingency volumes, which can be used only during emergency include local parks or rooftops. See current UTM operations and concepts at,[9].[10]Autonomous vehicles are required to travel in the approved volumes and use the contingency volumes as and when required.

(a) A Typical dense urban environment where we propose UAV flight.

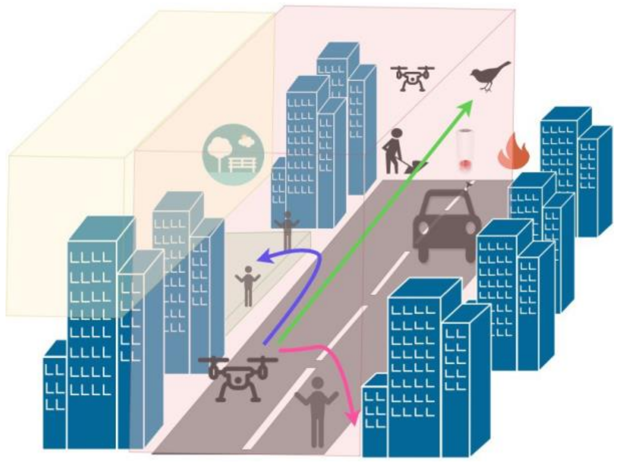

(b) Cartoon representing different contingencies that has to be mitigated

Figure 2. Task of the local planner

The main contribution of this paper is the local planner which ensures safe operation of an autonomous UAV inside the UTM approved volume to fly. Figure 2 shows a typical scenario for point to point operation in a dense urban environment. With in the UTM approved volumes an autonomous UAV has to avoid other UAVs sharing the same airspace or other unknown objects that may enter the airspace. It is assumed that other vehicles participating in UTM TCL-4 and sharing the same airspace will be equipped with DSRC/V2V communication exchanging relevant information for safe navigation. A tree based local planner is developed in this paper which develops collision free trajectories in this shared airspace. We describe an end-to-end simulation of an UAV safely traversing from one point to the next using a recursive local planner which plans obstacle free trajectories for the vehicle.

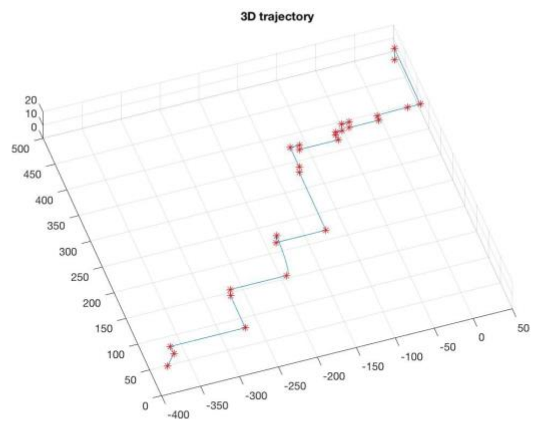
The rest of the paper is arranged as follows: Section II describes the UTM approved volumes and the requirements for a local planner. Section III describes the tree based planning algorithm. Section IV describes in details the implementation of the method in the Reflection simulation architecture. Section V shows the simulation results for both static and dynamic obstacles. Finally Section VI concludes the paper with some recommendation for future research and flight tests.

II. Technical Approach

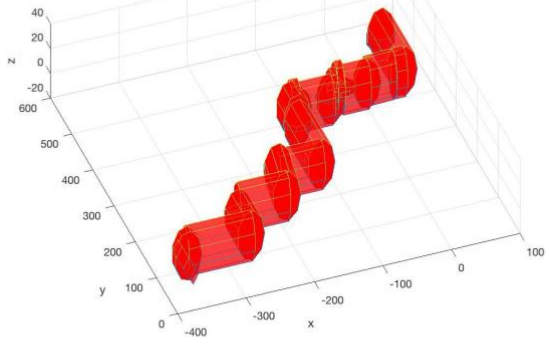
A complete autonomous operations includes a vehicle traversing autonomously from point A to point B, avoiding obstacles along the way.For vehicles participating in UTM, each operator has to submit a flight plan to UTM. The flight plan consists of a volume with specific time information.

It is assumed that the an operator will have an approved volume to fly before take off. An approved volume contains a list of waypoints along the way and an enclosed volume around the waypoint which have been approved by UTM to fly. In UTM-TCL4 it is envisioned that more than one UAVs have to share the same volume in dense urban landscapes. UTM will only control the density of vehicles in a particular volume but individual vehicles are responsible for object detection and collision avoidance.

This work focuses on autonomy developments of a single vehicle rather than co-operative control of all the vehicles participating in the environment. This is important because the same air space can be shared by different vehicles with different capabilities but obeying some basic requirements. Those requirements are still being evaluated but V2V communication of neighboring vehicles is one of the integral components of this infrastructure.



(a) Approved Flight Plan



(b) Approved volume along the flight plan

Figure 3. Approved Flight Plan and volume.

Figure 3 shows a typical approved plan. Figure 3a shows the result of an A* algorithm that defines the path that a particular UAV has to fly. Figure 3b shows an example of typical approved volume from UTM. This is based on local terrain of an urban environment, a typical cluttered polygonal environment. The goal of the planner is to plan feasible trajectories along the approved volume with all the constraints included. We introduce a recursive tree based local path planning approach to handle all the different uncertainties.

The main functions of the local planner is to compute and evaluate different possible trajectories the vehicle could take at a particular time. Given the list of the waypoints the vehicles has to follow, the function of the local planner is to provide a feasible trajectory for the internal controller to follow.

III. Recursive Tree based Local Planning

Motion Planning is an active area of research and major breakthroughs have been achieved in the community in the last few years. Sampling based methods, including probabilistic roadmaps and rapidly exploring random trees have been a particularly useful for different robotic path planning problems. We use a tree based planning approach for the current work.[12] The major advantages of using a tree based planning method are: a) branches of the tree can

Algorithm 1: The optimal Kinematic Tree algorithm.

```
Function :KinematicTree*($x_{init}$);
G.init($x_{init}$);
G.extended $\leftarrow x_{init}$ ;
G.not_extended $\leftarrow \varphi$
for $i$=1 to $K$ do
    $x_{lowest\_cost} \leftarrow choose\_state(G)$;
    Extend(G,$x_{lowest\ cost}$)
end
Return
Function :Extend(G,$x_{lowest\_cost}$);
$X_{next\_states} \leftarrow Steer(x_{lowest\_cost}, U, \Delta t)$;
for all $x_{state} \in X_{next\_states}$ do
    if $collision\_free\_path(x_{lowest\_cost},\ x_{state})$ then
        $G.add\_node(x_{state})$;
        $G.add\_edges(x_{lowest\_cost}, x_{state}, u))\ G.extended(x_{lowest\_cost}))\ G.not\ extended \longleftarrow G/G.extended$;
        $G.cost \longleftarrow cost(x_{lowest\_cost}) + g(x_{lowest\_cost}, x_{state}) + h(x_{state}))$
    end
end
Return G
Function :Choose State(G);
$x_{lowest\_cost} \leftarrow find\_min\ cost(G)$
Return x lowest cost;
```

be precomputed and thus save a lot of computation cost. b) different cost functions can be easily explored by a tree based approach.

The tree based planning approach is described in details.[12] We briefly describe the algorithm here for completeness. The basic algorithm is included in Algorithm 1. The local planner based on the tree algorithm incrementally builds upon a tree $G = (V, E)$ defined by its set of vertices V and edges E. The set of vertices encodes the inertial positions ($V = x_1,\ x_2,\ x_3\ x_i$) and the edges set consists of the path defined by joining the corresponding vertices ($E_{i,\ j} = \{x_i,\ x_j\}$). The tree is initiated at the vehicles start position. From this start configuration the tree is expanded by computing a set of reachable configurations. This set of reachable states can be pre-computed from a set of motion primitives described lates.

X_{next_states} define the all the possible configurations. The configuration (x_{lowest_cost}) is picked from the robot's configuration space (X_{next_states}) which has the minimum cost function. The node that is selected for expansion at each stage minimizes the cost $\hat{f}_i = \hat{g}_i + \hat{h}_i$, where the total cost $\hat{f}_i$ for the i^{th} node is the sum of the cost of navigating to that node $\hat{g}_i = g_i$ (cost of traveling to a node is always known) and $\hat{h}_i$ is a heuristic estimate of the remaining cost to reach the goal. In case of robot navigating a obstacle field environment the cost $\hat{h}_i$ is typically the remaining Euclidean distance between the node and the goal. The main purpose of the heuristic cost function is same as that served by a heuristic cost function in an A* algorithm; i.e. to grow the tree in relevant areas of the search space. If the path joining them is collision free then that new point is added to the vertex set V and the edge joining the points is added to the edge set E. This process is repeated until the goal is reached.

The tree based planning algorithm has been proved to be complete, i.e. it will find a path if a path exists. Also the tree based planner has been proved to be optimal up-to resolution if the cost function is admissible (heuristic cost function does not overestimate the actual cost).[12] Different costs can be incorporated in this formulations. Method to incorporate cost of translation that minimizes the jerk is discussed in the next section.

The Recursive Tree algorithm builds upon the tree algorithm at each time step (Algorithm 2). it is assumed that the local controller will be able to follow the developed plan closely. Thus current position is used to re-initialize the tree at each time. Note that continuity is always maintained in the path as the current position is assumed to be a subset of the path generated by the planner at the earlier time step. At each iteration a new tree is being built with current position and desired heading. The desired heading is calculated based in the next waypoint. This is done at each step to avoid any drift in the final solution.

The Recursive tree algorithm sets a local goal at each iteration. A local goal is the next waypoint from the list of approved way-points. The algorithm explores all the paths connecting the current position to the local goal using the tree based planner and selects the path that corresponds to the minimum cost. The local goal is shifted to the next way-point when the current position reaches the local goal.

The Recursive tree algorithm is run at 1 Hz. It is assumed that the on-board controller is able to follow the generated trajectory during that time. As the algorithm develops new trajectories the current position of the vehicle is used to re-initialize the tree at each iteration.

Algorithm 2: The Recursive Kinematic Tree algorithm.

```
while $LocalGoal$ is not $Goal$ do
    while $X_{current\_state}$ is not $LocalGoal$ do
        Main
        $X_{current\_state} \leftarrow x_{lowest\_cost\ prev}$ ;
        Function :KinematicTree*($X_{current\ state}$);
        $x_{init} \leftarrow X_{current\_state}$
        G.Re-initialize($x_{init}$);
        G.extended$\leftarrow x_{init}$ ;
        G.not_extended $\leftarrow \varphi$
        for $i=1$ to $K$ do
            $x_{lowest\ \_cost} \leftarrow choose\_state(G)$;
            Extend(G,$x_{lowest\ cost}$)
        end
        Return
        Function :Extend(G,$x_{lowest\_cost}$);
        $X_{next\_states} \leftarrow Steer(x_{lowest\_cost}, U, \Delta t)$;
        for all $x_{state} \in X_{next\_states}$ do
            if $collision\_free\_path(x_{lowest\_cost},\ x_{state})$ then
                $G.add\_node(x_{state})$;
                $G.add\_edges(x_{lowest\ cost}, x_{state}, u))$ $G.extended(x_{lowest\ cost}))$ $G.not\ extended \longleftarrow G/G.extended$;
                $G.cost \longleftarrow cost(x_{lowest\ cost}) + g(x_{lowest\ cost}, x_{state}) + h(x_{state}))$
            end
        end
        Return G
        Function :Choose_State(G);
        $x_{lowest\ \_cost} \leftarrow find\_min\ cost(G)$
        $LocalGoal \leftarrow next_{waypoint}$
        Return $x\_lowest\ cost$;
    end
    Return
end
Return
```

Figure 5 shows the tree based trajectory generator for a particular time. All the branches are given random wight and the planner picks the branch with the minimum cost. Branches that are outside the approved volumes are assigned high costs so they are not picked for expansion at the next iteration. How the local plan is generated from the set of motion primitives is discussed next. We will begin by defining the motion primitives.

A. Motion Primitives

A set of motion primitives which are used to construct the local plan at each step is calculated. Defining the cost function of jerk, the local path is computed which minimizes the jerk for a translation.

$$C = {}_{0}^{\Delta t} L \, dt \qquad (1)$$

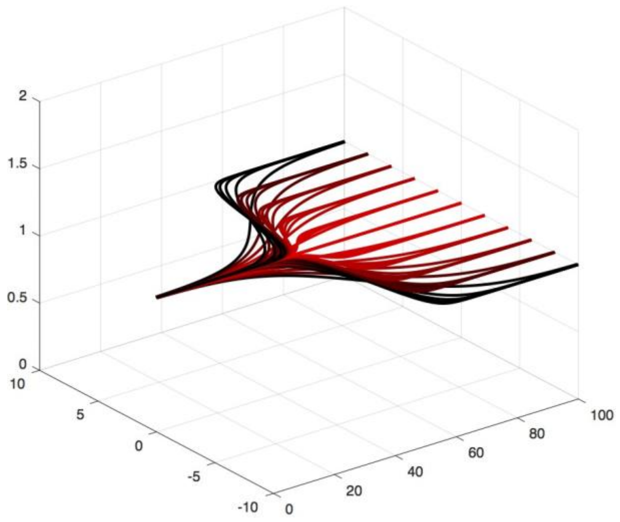
where,

$$L = \left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2 \tag{2}$$

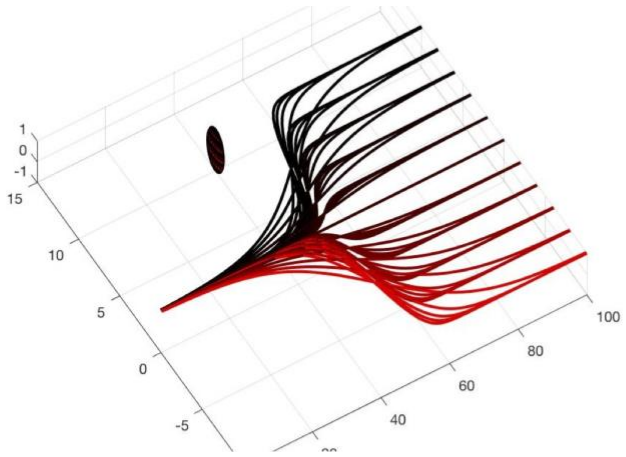
Using Euler equations it can shown that the solution of the differential equation is of the form

$$x(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \tag{3}$$
$$y(t) = b_0 + b_1t + b_2t^2 + b_3t^3 + b_4t^4 + b_5t^5 \tag{4}$$

These equations can be solved for different initial and final conditions. The X direction is fixed with the $\mathbf{X_b}$ body frame of the muticopter. The y-direction gives the lateral position. Thus different positions and velocity can be encoded in this pre-computation of branches. With initial position set at the origin, several initial and final velocities can be encoded in this formulation. Different final position in the lateral directions lead to possible deviation of the trajectory from the center-line (equation 5 to equation 10). The center line depicts the ideal path which is directly found joining the approved way-points.



(a) Motion Primitives with energy cost function



(b) Motion Primitives with cost function depicting nearness to an obstacle. .

Figure 4. Motion primitives with different cost function (Darker color indicates higher cost)

$$x(t=0) = 0 \qquad y(t=0) = 0 \tag{5}$$
$$\dot{x}(t=0) = v_{initial} \qquad \dot{y}(t=0) = 0 \tag{6}$$
$$\ddot{x}(t=0) = 0 \qquad \ddot{y}(t=0) = 0 \tag{7}$$
$$x(t=\Delta t) = x_{final} \qquad y(t=\Delta t) = y_{final} \tag{8}$$
$$\dot{x}(t=\Delta t) = v_{final} \qquad \dot{y}(t=\Delta t) = 0 \tag{9}$$
$$\ddot{x}(t=\Delta t) = 0 \qquad \ddot{y}(t=\Delta t) = 0 \tag{10}$$

For details of minimal jerk based trajectory calculations for multi-copters see,[13].[14] Figure 4 shows the motion primitives developed for different final conditions. Figure 4 a shows the path cost as a function of energy expended for the maneuver. Straight line motion intuitively has lower cost than other possible candidates. Figure 4 b shows cost function as a distance to neighboring obstacle. Similarly other cost functions can be encoded in the motion primitives. Other costs includes are distance to other objects as well as distance to the approved UTM volume. Branches which end outside the UTM approved volume are assigned a very high cost. This ensures that those branches are never picked for expansion in the *Extend* routine (in algorithm 1).

These motion primitives are used to build up the tree at each instant of time. The set of branches are precomputed to save a lot of computational cost at run time. The set of branches $\mathbf{X}^b$ consists of all possible changes in position and velocity given the set of possible inputs $\mathbf{U}$, and is formed by concatenating the vectors of all the branches:

$$\Delta\mathbf{X}^b = [\Delta\mathbf{x}^b_{1111} \dots \Delta\mathbf{x}^b_{IJKL}] \tag{11}$$

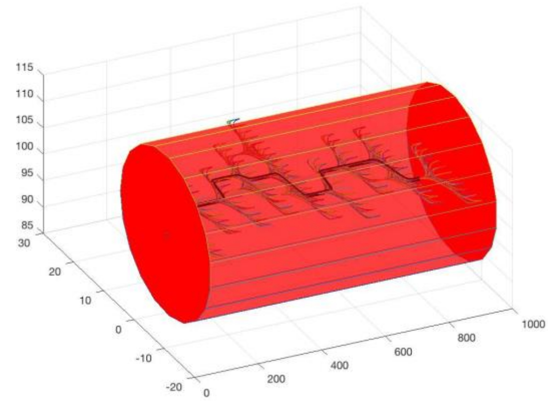
The selected node with position $\mathbf{x}_i$ and time t_i is expanded using the pre-computed branches. Wind speed is obtained from the forecast for the position and time of the selected node and is assumed to be constant over the time interval Δt, and a set of new candidate nodes is computed as

$$\mathbf{X}_{i,new} = \mathbf{X}_{i,current}\mathbf{1} + \mathbf{T}_i\Delta\mathbf{X}^b + \mathbf{w}_i\Delta t\mathbf{1} \tag{12}$$

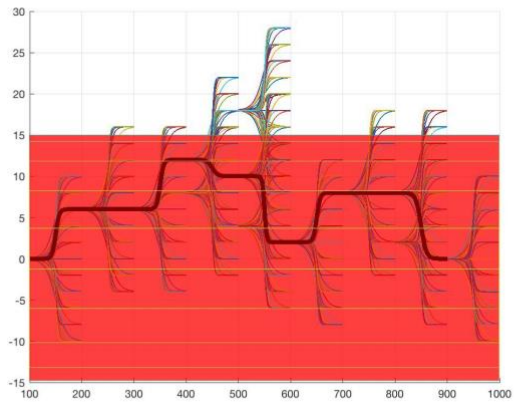
where $\mathbf{T_i}$, defined as

$$\mathbf{T}_i = \begin{bmatrix} cos(\theta_i)cos(\psi_i) & sin(\varphi_i)sin(\theta_i)cos(\psi_i) - cos(\varphi_i)sin(\psi_i) & cos(\varphi_i)sin(\theta_i)cos(\psi_i) + sin(\varphi_i)sin(\psi_i) \\ cos(\theta_i)sin(\psi_i) & sin(\varphi_i)sin(\theta_i)sin(\psi_i) - cos(\varphi_i)cos(\psi_i) & cos(\varphi_i)sin(\theta_i)sin(\psi_i) - sin(\varphi_i)cos(\psi_i) \\ -sin(\theta_i) & sin(\varphi_i)cos(\theta_i) & cos(\varphi_i)cos(\theta_i) \end{bmatrix} \tag{13}$$

is the transformation which rotates the set of precomputed branches to the local frame defined by the orientation angles $[\varphi_i, \theta_i, \psi_i]$. Cost function based on wind is not considered in this paper, but can can be used as an input to the cost function.[15,16]



(a) Local Plan generated inside the approved volume



(b) Top view of the local plan that was selected.

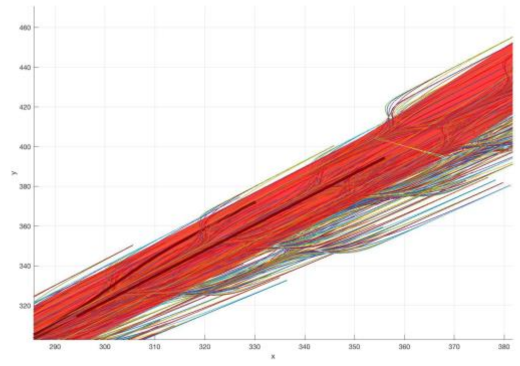
Figure 5. Local Plan generated at each instant of time

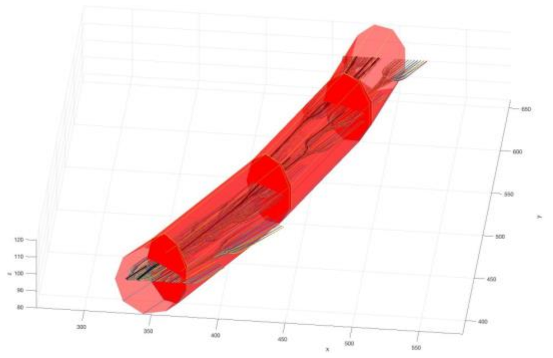
Figure 5 shows the local plan being developed at a particular instant of time. Motion primitives which goes outside the approved volume are given a very high cost. They are still kept in the tree structure in the event that no other paths are available.

The output of the local planner is a list of waypoints. This is then given to the multi-copter at each instant of time. It is assumed that the controller is able to faithfully follow the plan. The output that is received from the simulator is the current position of the multi-copter. The current position is a part of the list of local waypoints given in the previous time step. The current position is then used to generate the next local plan. Sufficient over-lap is maintained so that there is no arbitrary jump in the solution.

Figure 6 shows the successful development of local planner along the path of the vehicle. It is assumed that the current position of the vehicle belongs the the local plan developed at the earlier time step. How far along the local plan the vehicle actually gets will depend on the controller. But the local plan generation doest not depend on any particular time requirements on the controller. To ensure continuity sufficient points are taken from the planner in the previous time step so that there are no abrupt jumps in the overall path. It is important to generate the local plans at each instant so that there is no drift in the final solution.

The proposed architecture was first tested in MatLab as shown in figures 5 and 6. Once satisfacutory performance was generated, the entire solution in tested in the realistic simulation developed here at NASA called Reflection.[17]





(a) Top view of the local plan that was selected. (b) Local Plan generated inside the approved volume

Figure 6. Local Plans generated along the path.

IV. Implementation

The Reflection Architecture is a real time component based plug and play architecture for rapid development of embedded vehicle systems developed here at NASA Ames.[17] This system has been used in may applications at NASA and we have used this system with UTM to build an end-to-end simulation of UTM TCL4 simulation.

Detailed design and implementation of a fully autonomous and programmable autopilot system for small scale autonomous unmanned aerial vehicle (UAV) aircraft using the Reflection architecture is describe here.[18] In this paper we discuss in details the subsystems that are relevant to the Local Planner.

Figure 7 shows the overall Reflection architecture for the overall system. With the Reflection plug-and-play architecture, simulation and hardware can be mixed on the fly for in-situ simulation testing of hardware components at any level of granularity. As shown in figure 7, the same architecture can be be used interchangeable with simulation or flight tests.

The main components of the Reflection software architecture can be divided into sensing and perception, decision making and planning and control. The sensing and perception unit consists of SLAM processing, Object detection and vehicle sensors. The planning and control subsystem comprises of path planning subsystem, local planner and autopilot. The Decision making module is responsible for the overall behavior of the system. The Flight Management System(FMS) handles the overall data management (figure 7,).

The control system of the architecture is an instance of the Autopilot System (AP) class as shown in figure 7. The AutopilotSystem class is responsible for communicating with the rest of the Reflection system and maintaining the two main objects in the system: the FMS (flight management system) and the Controller. The FMS is responsible for maintaining the list of commands which specify FMS mode instructions. The mode instructions are used by the FMS to provide targets to the controller. The controller is responsible for implementing the control loops which control the aircraft through the vehicle?s actuators.

The Decision Making(DM) module is responsible for the overall behavior of the UAS.[19] The DM communicates with the rest of the UTM system and ensures the overall feasibility if the system. From several feasible Trajectories (1,2,...n) the DM decided the final waypoints the vehicle has to fly. Together with the list of way-points and the approved volume around them, the Local Planner module generates feasible trajectories inside the approved volume avoiding other vehicles sharing the same volume.

A. Local Planner Subsystem

The Local Planner system is implemented as a stand alone plug-in component in the Reflection Architecture. The system is designed in such a way that it can be easily be integrated with current autopilot systems.

Figure 8 shows the top level classes in the Local planner Subsystem. The local planner class gets inputs from the communication class which we have designed. It is assumed that other vehicles sharing the same airspace should be able to communicate through DRSC/V2V communication in close range. We have simulated the V2V communication using socket programming. All the vehicles sharing the same airspace are registered with the V2VCommunication class which communicates the position of all the vehicles in close proximity.

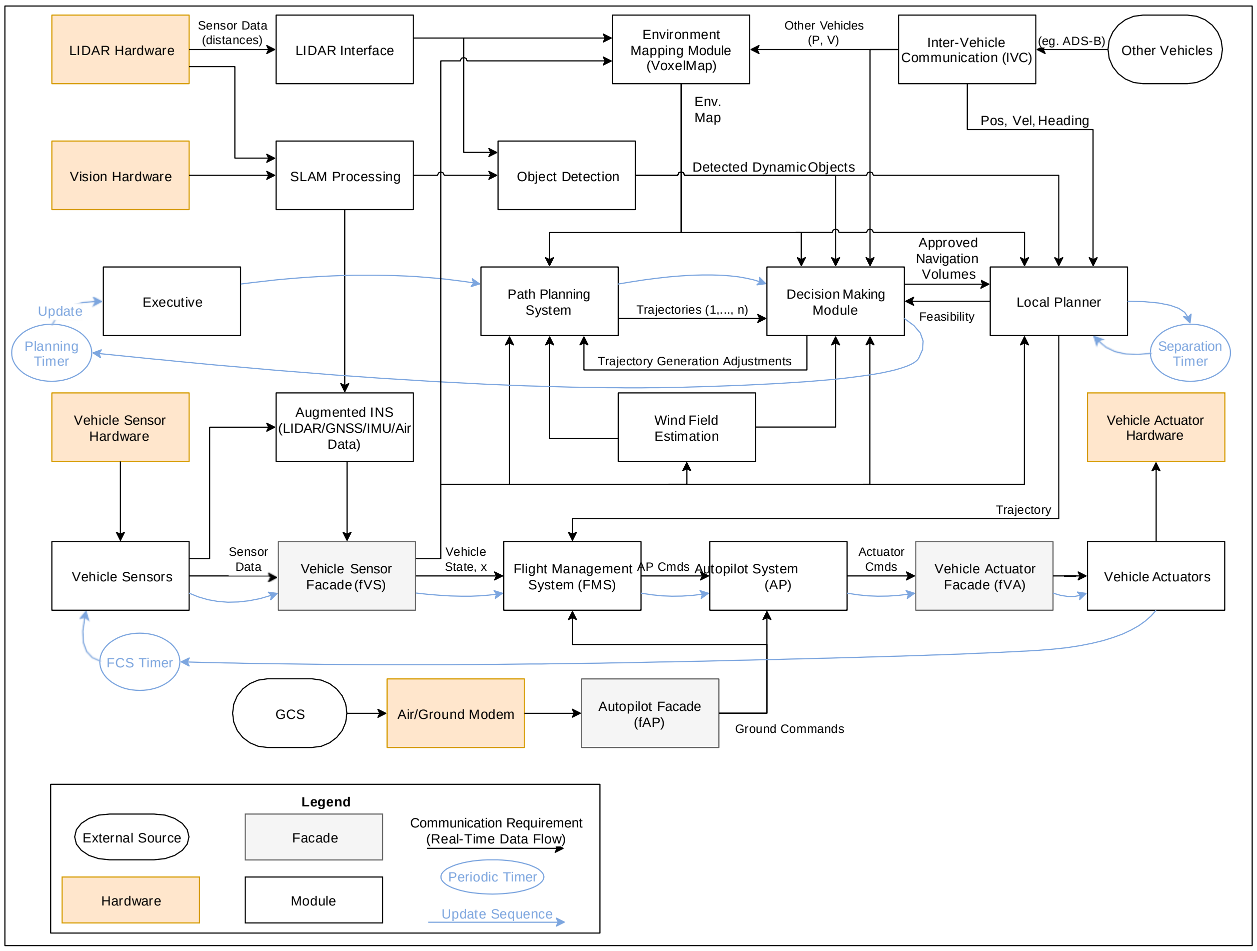


Figure 7. Reflection Software Architecture Diagram.

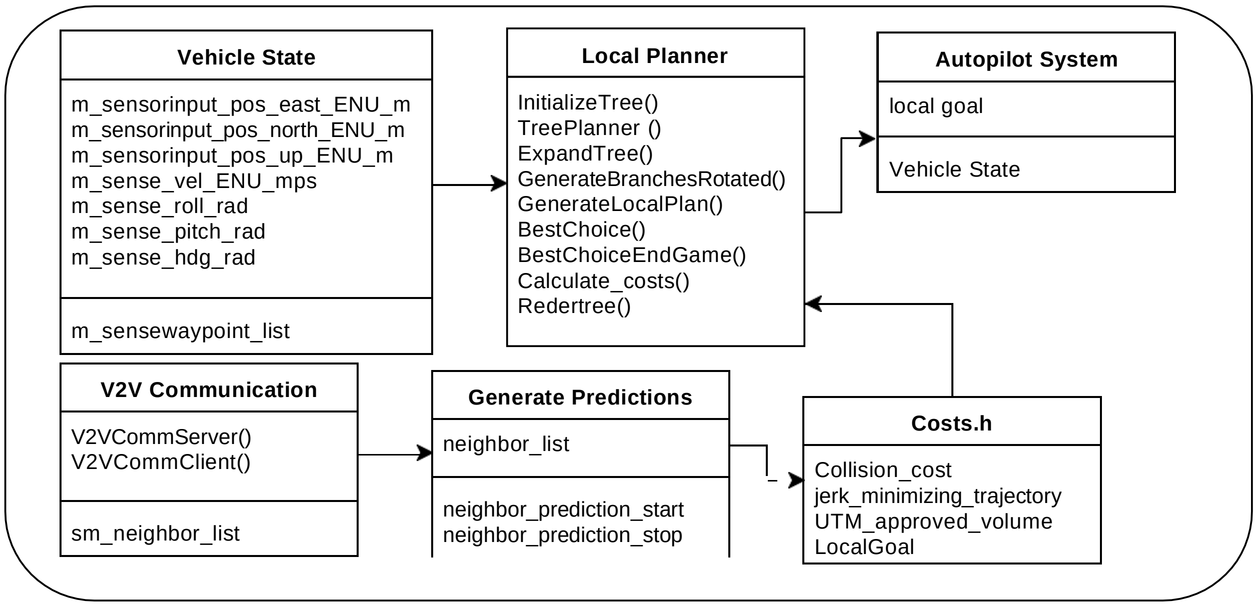


Figure 8. Top-Level Classes in the Local Planner System

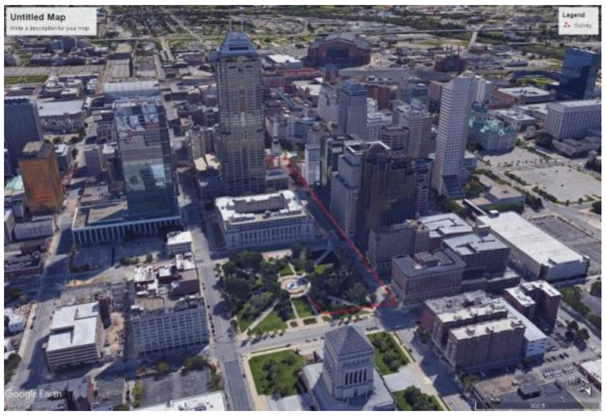
The raw position measurements through V2V communication is not useful for the local planner. Rather a prediction of the states of the surrounding vehicles are important for the local planner. The Generate prediction class takes inputs from the time stamped positions and velocity of the near-by vehicles and generates possible trajectories for all the vehicles.

From the Vehicle Sensor Facade the planner obtains the Vehicle state and uses it to initialize the tree. The ExpandTree() routine generates the motion primitives and the GenerateBranchesRotated() transforms the generated branches to the desired orientation according to equation 12.

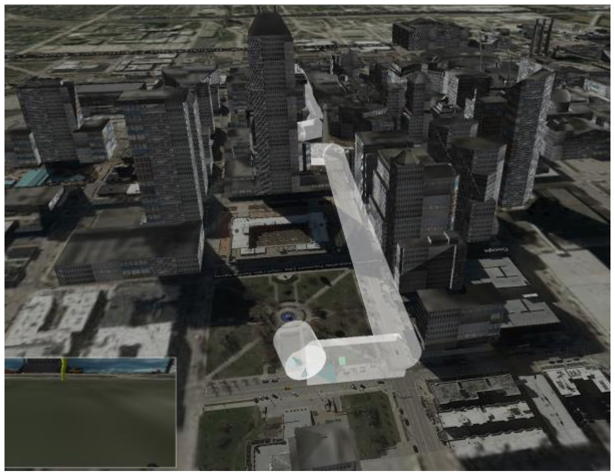
The cost function for each branch of the local planner calculates the collision cost of each vehicles from the possible trajectories. The cost function also calculates cost of individual trajectories which includes local vehicle conditions. The local planner calculates the best trajectory for the UAS to follow based on minimization of the cost function.

The local planner subsystem runs at 1Hz. V2V communication module also runs at 1 Hz. The overall system runs at much higher rate of 30Hz. This entire system was tested in different scenarios in an urban landscape.

V. Simulation Results



(a) List of waypoints in Downtown Indianapolis (Google Earth).



(b) Reflection software simulating the same environment with UTM approved volume.

Figure 9. Algorithm being tested in reflection architecture.

We simulated a point to point BVLOS UAS operation in a urban environment. Downtown Indianapolis in the WholeSale District offers an example of typical urban neighborhood for UTM TCL-4 operations. Here we consider a BVLOS flight between two points as shown in Figure 9. The waypoints are shown as *red* line in figure 9a. It is assumed that the UTM will approve a volume around the list of waypoints before the flight.

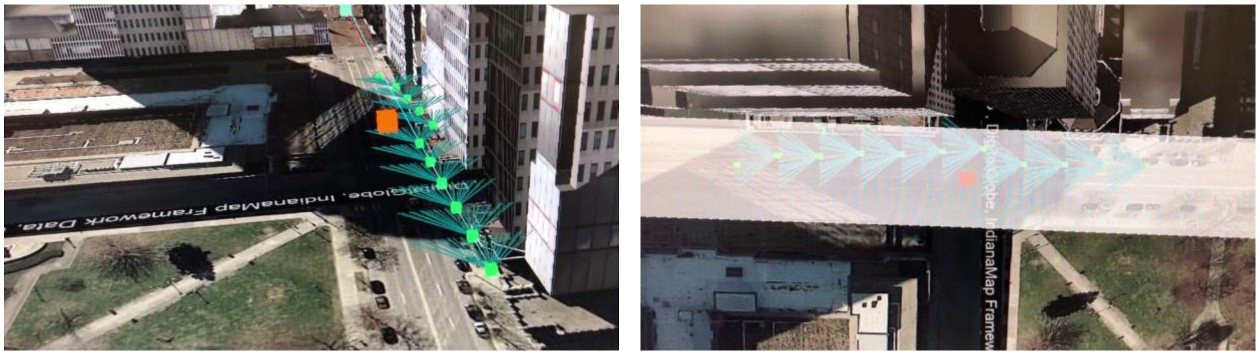
The Reflection simulation architecture simulates the downtown Indianapolis as shown in figure 9b. The urban landscape rendering and collision model was created for large section of downtown Indianapolis, see[20] for details.Figure 9b shows the screen shot from Reflection software and the UTM approved volume for the given list of way-points.

The task of the local planner is to navigate a UAS through this approved volume avoiding static and dynamic obstacles.

It is assumed that the prior permission to fly was obtained and the Decision maker module has selected the current list of waypoints and this has been passed on to the Planning subsystem. Now the task of the local planner is to safely guide the vehicle along the approved UTM volume while avoiding static and dynamic obstacles which were not know to the system a-priori.

A. Static Obstacle avoidance

To test the efficacy of the the Tree based planner a static obstacle avoidance is tested in the scenario described above. A static obstacle is introduced in the middle of the approved volume as shown in figure 10 a. It is assumed that the local planner detects the obstacle either through its on-board sensors or is communicated though UTM SDSPs.



(a) Tree based trajectory generations (b) All the branches of the tree are inside the UTM approved volume

Figure 10. Static Obstacle Avoidance in UTM TCL4 implemented in Reflection

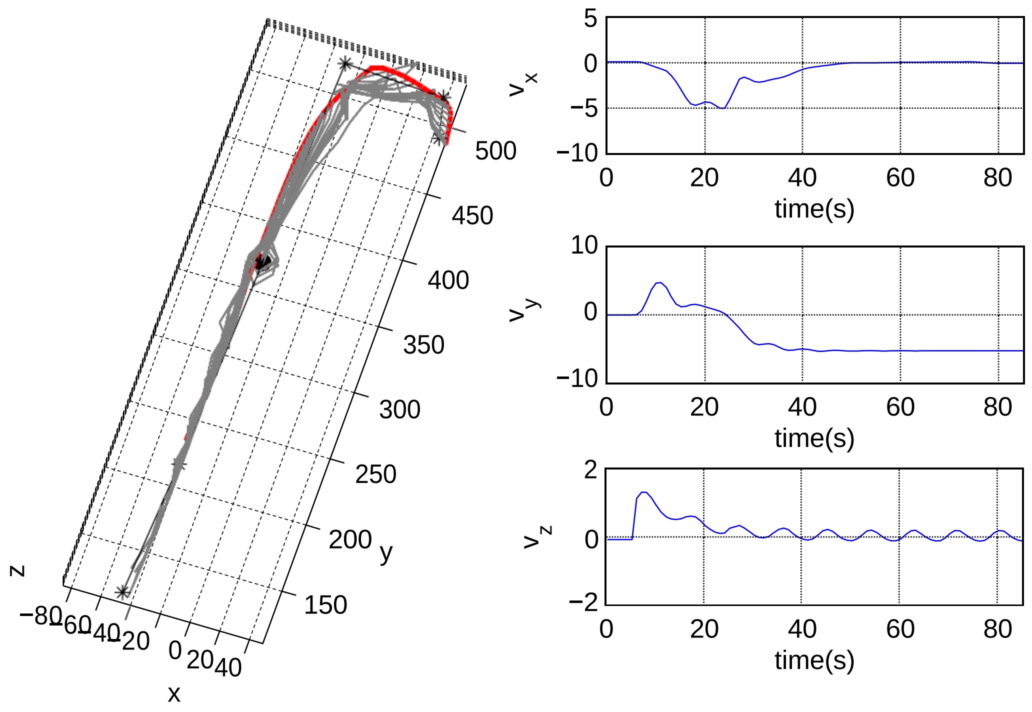


Figure 11. Trajectory and Velocity solution using the tree based trajectory planner

The local planner subsequently plans paths for the UAS which satisfies all the UTM constrains as well as avoids the static obstacle. Figure 10a shows the screen shots of local planner being generated in Reflection. The orange cube in figure 10a is the static obstacle that the tree planner avoids. The selected path is shown as a string of green dots along the way.

Figure 10b shows that solution obtained from the recursive tree planner remains within the approved UTM volume while avoiding the obstacle. Thus all the constraints of the planning problem have been successfully addressed.

Figure 11 shows the complete trajectory and the velocity of the vehicle along the path. Figure 11 also shows all the possible trajectories generated by the recursive tree planner along the path. It is evident from the plot how the trajectory solution evolve with time. The planner finally chose a path that satisfied all the constraints.

The planner was thus shown to successfully plan static obstacle free paths for vehicles participating in UTM. The recursive tree based trajectory was next tested to handle dynamic obstacles.

B. Dynamic Obstacle avoidance in TCL4

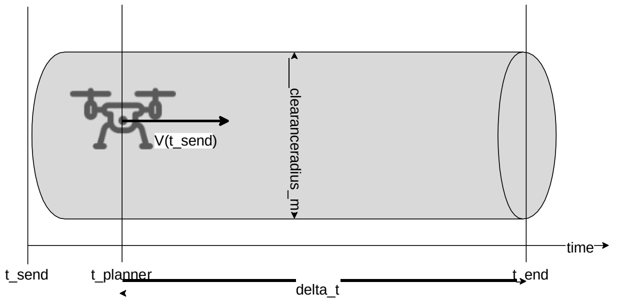
We have developed multiple vehicle simulation in the Reflection architecture. The vehicles are controlled independently using their own individual control systems. All the vehicles can follow a desired set of waypoints independently.It is assumed that all the vehicles participating in UTM will communicate with each via V2V communication.

We assume that we plan for only one vehicle. We call this vehicle the planning vehicle (UAV0) for the rest of the discussion. All the other vehicles follows a set of pre-approved way-points.

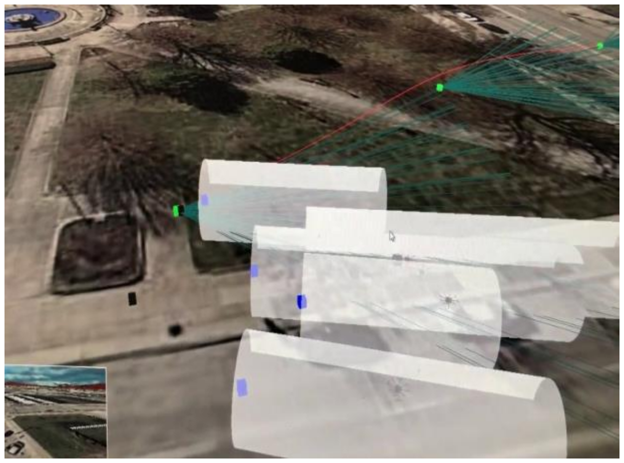
1. V2V Communication

As mentioned in the Section IVA the V2V communication class communicates with all the connected vehicles using a TCP/IP protocol. The vehicles do not communicate their entire trajectory or intent. Each V2V message contains position and velocity information time stamped for each vehicle. The message also contains desired clearance radius for each vehicle.

The planning vehicle (UAV0) maintains a list of all the connected vehicles. The local planner estimates the position of the all the other vehicles at the current time by interpolating the position with velocity information. Also the local planner estimates the possible position of the vehicle during the next projected time interval. These two positions along with the clearance radius is used to define an avoidance volume for each of the connected vehicles (figure 12a).



(a) Avoidance Volume



(b) Avoidance volume implementation in Reflection

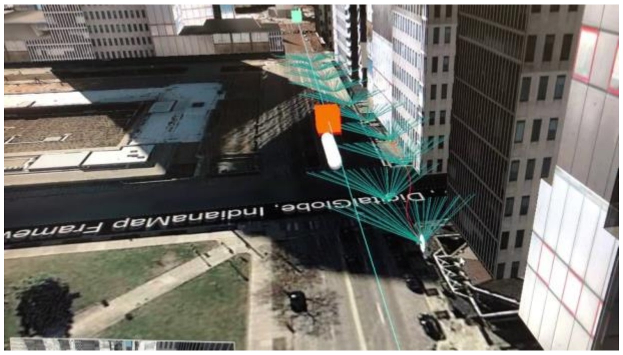
Figure 12. Estimated Volume to avoid for other UAVs communicating over V2V

Figure 12b shows the implementation in Reflection. The black dot in the figure is the position of each vehicle that was received by the vehicle at t_{send}. The blue dot represents the position interpolated at the time that corresponds to the planner. And the white translucent volume is the avoidance volume for each vehicle. The recursive tree planner needs to avoid all the avoidance volumes while maintaining the UTM constrains.

2. *Results*

We again consider the same scenario SectionVA. The waypoints are shown as *red* line in figure 9a. But in this case we consider flight by two UAVs starting at opposite ends of the the approved waypoints. It is assumed that the UTM will approve a volume around the list of waypoints before the flight.

Our planning UAV (UAV0) follow the same set of waypoints as described in sectionVA. While the other UAV(UAV1) follows the reverse set of way-points. Both the UAVs share their position and velocity using the V2V communication described above.


(a) Tree based trajectory generations

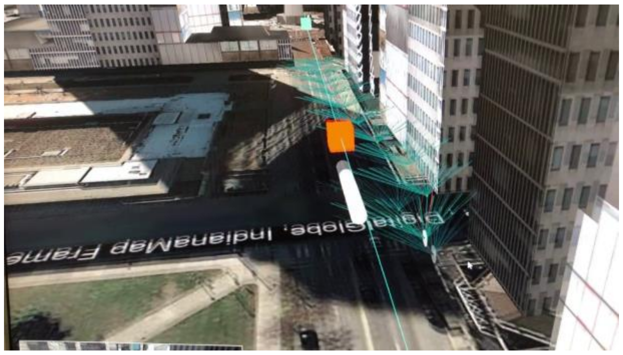
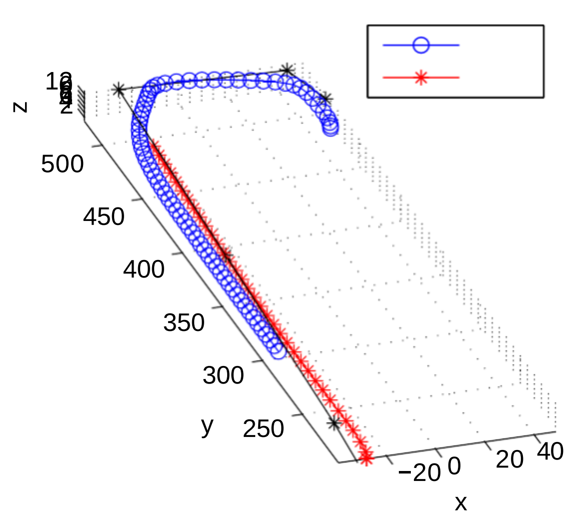
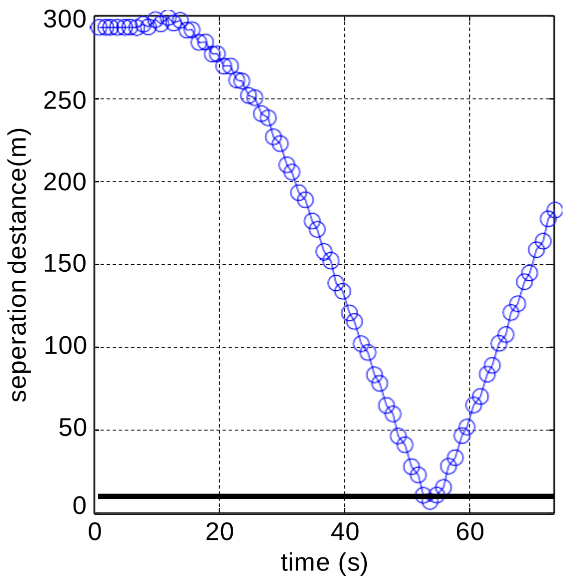

(b) All the branches of the tree are inside the UTM approved volume

Figure 13. Dynamic Obstacle Avoidance using V2V communication and Recursive Tree Planner

Figure 13 shows the screenshot of the successful implementation of the dynamic obstacle avoidance using the recursive tree algorithm in Reflection. As seen from the figure the developed branches of the tree avoid the entire avoidance volume of the other UAV(UAV1). Also, like the earlier case the developed branches maintains the UTM constrains as well.



(a) De-conflicted Trajectories of the two UAVs



(b) Separation distance between the two UAVs.

Figure 14. Implementation in reflection

Figure 14 shows the collision free trajectories of the two UAVs sharing the same approved UTM volume. The separation distance between the two UAVs are also plotted. The minimum separation distance between the UAVs maintains the clearance radius requested by UAV1 through the V2V messages.

Figure 15 shows the trajectory and velocity of the planning UAV (UAV0). Figure 15 also shows the evolving

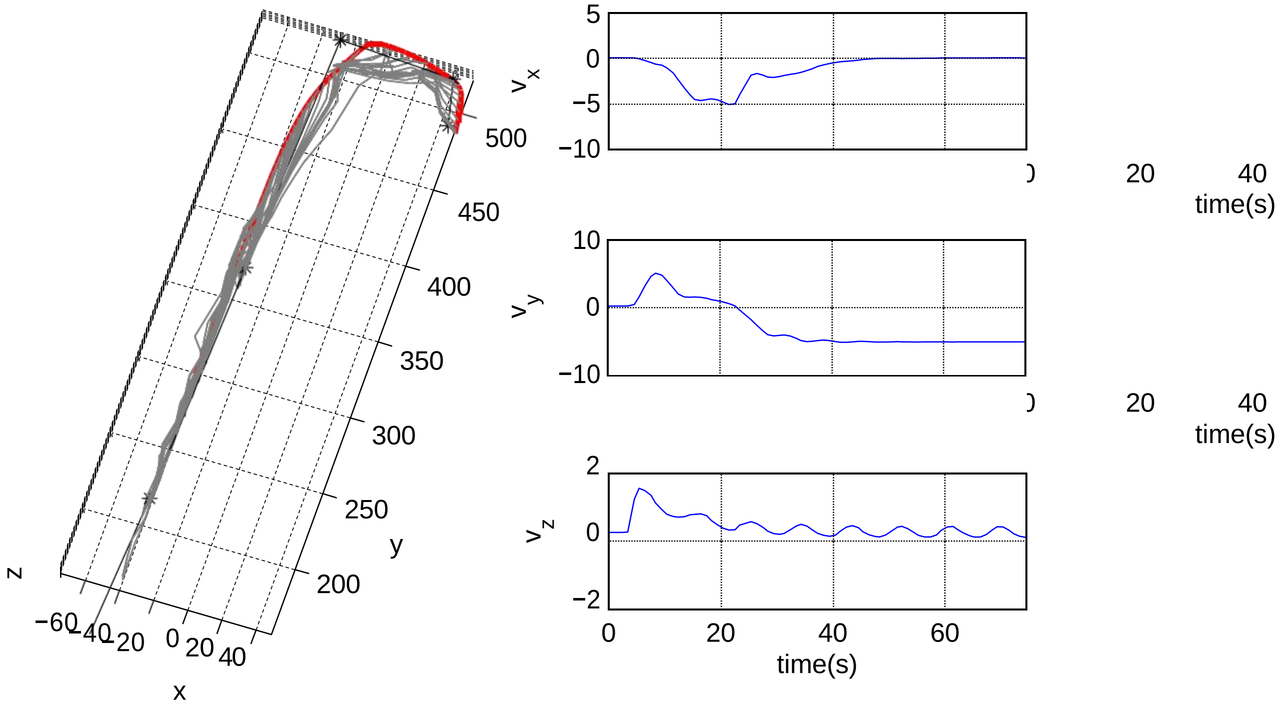


Figure 15. Trajectory and Velocity solution of UAV0 using the tree based trajectory planner

trajectories of UAV0 with time. The recursive tree planner thus successfully finds de-conflicted trajectories with other UAVs sharing the same UTM approved volume. This technology enables vehicles operating in BVLOS UTM TCL-4 flights with on-board real time re-planning capabilities and sharing the same volume with other similar vehicles.

VI. Conclusion

This paper has described in details a end-end system where multiple vehicles can fly autonomously in an UTM approved airspace. The paper describes a tree based trajectory planning algorithm which takes into considerations the UTM volume boundaries and other UAVs flying in and sharing the same airspace. The vehicles communicates over vehicle to vehicle communicating protocol and shares position and velocity information among themselves. The paper has demonstrated an entire operational architecture of successful implementation of the on-board path planner in Reflection Software. Simulation flight tests in downtown Indianapolis were described in the test results.

Future endeavors will continue to test more vehicles in the same simulation environment. Maximum number of UAVs and their communication requirements will be tested as we add more vehicles in this scenario. Flight tests are also planned to test this on-board path planner as a part of TCL-4 flight tests.

Acknowledgments

The authors would like to thank our colleagues in the NASA SAFE50 and UAS Traffic Management (UTM) projects for their collaboration and insight.

References

[1]G. Hoareau, J. J. Liebenberg, J. G. Musial, and T. R. Whitman, "Package transport by unmanned aerial vehicles," Aug. 15 2017, uS Patent 9,731,821.

[2]J. Katrasnik, F. Pernus, and B. Likar, "A survey of mobile robots for distribution power line inspection," *IEEE Transactions on Power Delivery*, vol. 25, no. 1, pp. 485–493, 2010.

[3]N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple uavs for persistent surveillance: algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2012.

[4]S. M. Adams and C. J. Friedland, "A survey of unmanned aerial vehicle (uav) usage for imagery collection in disaster research and management," in *9th International Workshop on Remote Sensing for Disaster Response*, 2011, p. 8.

[5]C. Stöcker, R. Bennett, F. Nex, M. Gerke, and J. Zevenbergen, "Review of the current state of uav regulations," *Remote sensing*, vol. 9, no. 5, p. 459, 2017.

[6]P. H. Kopardekar, "Unmanned aerial system (uas) traffic management (utm): Enabling low-altitude airspace and uas operations," 2014.

[7]M. Johnson, J. Jung, J. Rios, J. Mercer, J. Homola, T. Prevot, D. Mulfinger, and P. Kopardekar, "Flight test evaluation of an unmanned aircraft system traffic management (utm) concept for multiple beyond-visual-line-of-sight operations," 2017.

[8]T. Prevot, J. Rios, P. Kopardekar, J. E. Robinson III, M. Johnson, and J. Jung, "Uas traffic management (utm) concept of operations to safely enable low altitude flight operations," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3292.

[9]M. A. Johnson, "Unmanned aircraft system traffic management (utm): Defining the future of the drone industry," 2017.

[10]J. Rios and M. Johnson, "Unmanned aircraft systems traffic management (utm) concepts and architecture overview," 2018.

[11]K. S. Krishnakumar, P. H. Kopardekar, C. A. Ippolito, J. Melton, V. Stepanyan, S. Sankararaman, and B. Nikaido, "Safe autonomous flight environment (safe50) for the notional last ?50 ft? of operation of ?55 lb? class of uas," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0445.

[12]A. Chakrabarty and J. Langelaan, "Uav flight path planning in time varying complex wind-fields," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 2568–2574.

[13]M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[14]V. Stepanyan and K. S. Krishnakumar, "Estimation, navigation and control of multi-rotor drones in an urban wind field," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0670.

[15]A. Chakrabarty and J. Langelaan, "Flight path planning for uav atmospheric energy harvesting using heuristic search," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 8033.

[16]A. Chakrabarty and J. W. Langelaan, "Energy-based long-range path planning for soaring-capable unmanned aerial vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1002–1015, 2011.

[17]C. Ippolito, G. Pisanich, and K. Al-Ali, "Component-based plug-and-play methodologies for rapid embedded technology development," in *Infotech@ Aerospace*, 2005, p. 7122.

[18]C. Ippolito, G. J. Pai, and E. W. Denney, "An autonomous autopilot control system design for small-scale uavs," 2012.

[19]A. Chakrabarty, R. A. Morris, X. Bouyssounouse, and R. Hunt, "An integrated system for autonomous search and track with a small unmanned aerial vehicle," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0671.

[20]C. A. Ippolito, S. Hening, S. Sankararaman, and V. Stepanyan, "A modeling, simulation and control framework for small unmanned multicopter platforms in urban environments," in *2018 AIAA Modeling and Simulation Technologies Conference*, 2018, p. 1915.