Graduate research project Report

UAS Airport Ground Traffic Simulation for vision-based UAV ground Operation

E Zhang

# Background and purpose

Background

* Runway safety is a top priority at both the Federal Aviation Administration (FAA) and the National Transportation Safety Board. Reducing collision on the airport field is a major component of the effort to improve runway safety for the benefit of all air travelers.
* Sometimes, foreign objects such as vehicles may occur on runways. On an active runway involving the movement of aircraft, the presence of FOD may lead to an air crash and consequential loss of life resulting in significant losses to airline companies. Accordingly, there is a need to provide a version-based ground operation system, which seeks to maintain order and security of the airport

Purpose

* Provide a version-based ground operation surveillance system.
* Provide a simulation of the airport operation involving aircraft and airport traffic. (Create scenario for demonstration)
* Set up one or more camera on the airport tower or other building which used to monitor the aircraft movement. (Obtain the image data)
* With this information, the surveillance system will locate the aircraft and give feedback to control or steer the traffic on the runway. (Feedback Control UAV)

## System Environment

OS:

Windows operating system (7 or higher edition)

Framework:

.NET Framework 3.5 or higher

Software:

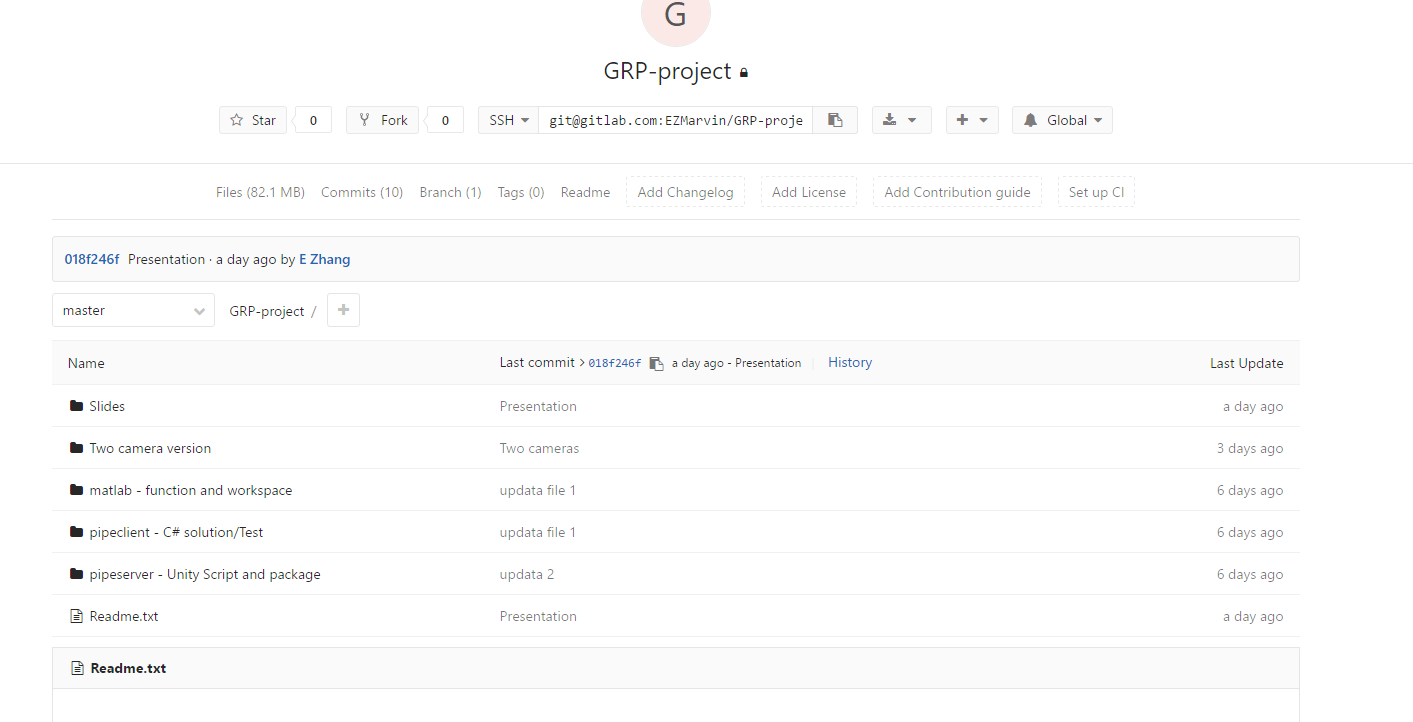
Unity3D (5.5 or higher edition)

MatLab 2015(or other edition with image process toolbox)

Visual studio 2015 (or other C# IDE )

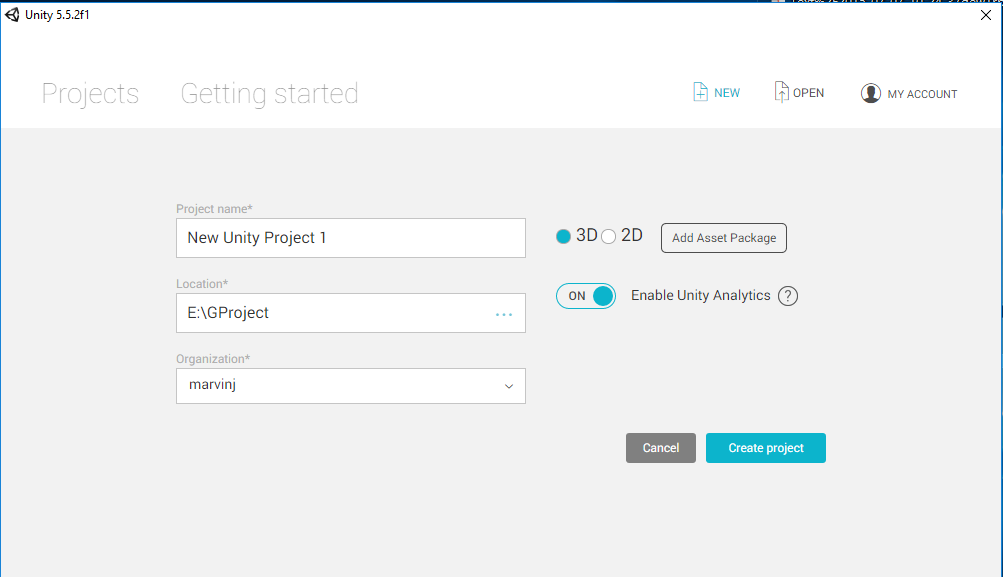
## System Deployment

1. Get project files in GitLab. https://EZMarvin@gitlab.com/EZMarvin/GRP-project.git

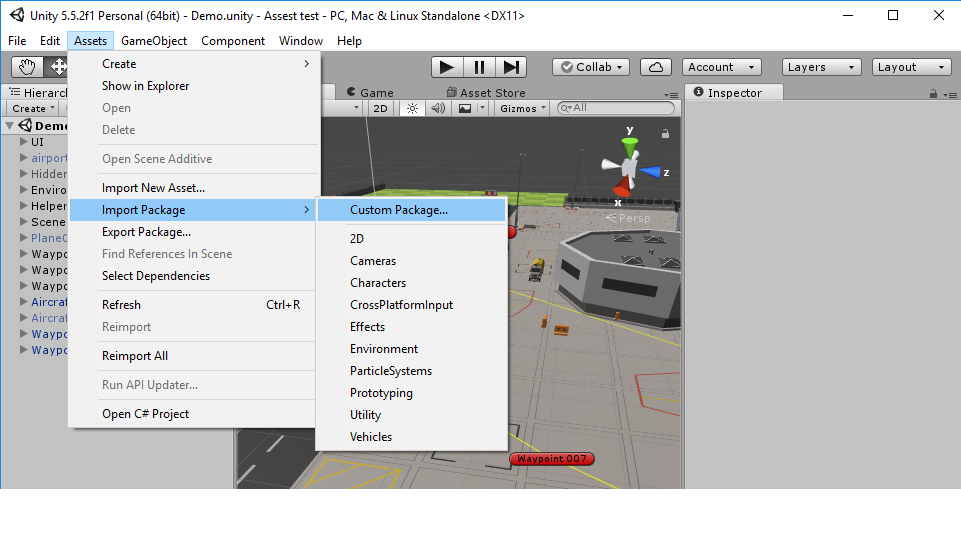


Picture 1. GitLab folder

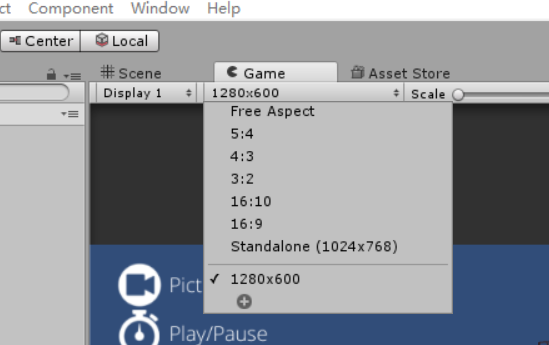
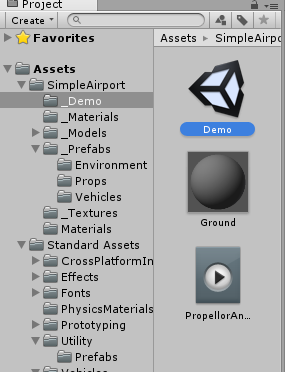
1. Open Unity3D, create new project and import unity3d package. Select “Demo” scene under Asset/SimpleAirport/\_Demo folder, and change resolution into 1280\*600



Picture 2. Create new project

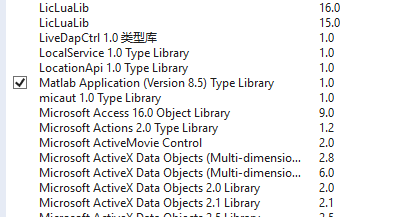


Picture 3. Import unity package



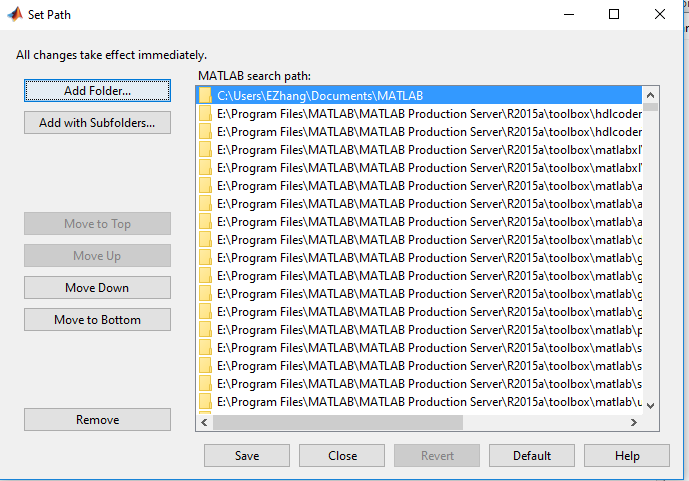
Picture 4. Select Demo scene and change resolution in game tab

1. User C# IDE to create new solution or project, copy the project or the code in “program.cs”, make sure you have or add the Matlab COM preference. In visual studio, from ‘Project’ menu, select ‘Add Reference’, select ‘COM’ tab, select ‘Matlab Application’



Picture 5. Add preference

1. Put MatLab function and .mat file in the Matlab Path folder, you can check your Matlab Path in the ‘Set Path’.



Picture 6. Set Matlab path

1. Start by playing Unity first, and then the C# program. Check Youtube Link: <https://youtu.be/khAgGfqcwUI> for the full process.

# System Introduction

Simulation – Unity3D

Use Unity3D to provide simulation scenario, which has an UAV operating at the airport with multiple bus and truck interference.

* Unity3D has Abundant **materials and Asset**
* Unity3D is Suitable for **quick development**
* Unity3D is **Multi-platform** (especially for .NET )

Data Transmission – Named Pipe

Use Named Pipe to complete data transmission

* **Named pipe** (also known as a FIFO for its behavior) is an extension to the traditional pipe concept
* **Inter-process communication** - to transfer data between two processes on one workstation
* Named pipe provide Provides a separate conduit for client/server communication (**Two - way**)

Image Process - Matlab

Use MatLab to process Image data and return feedback

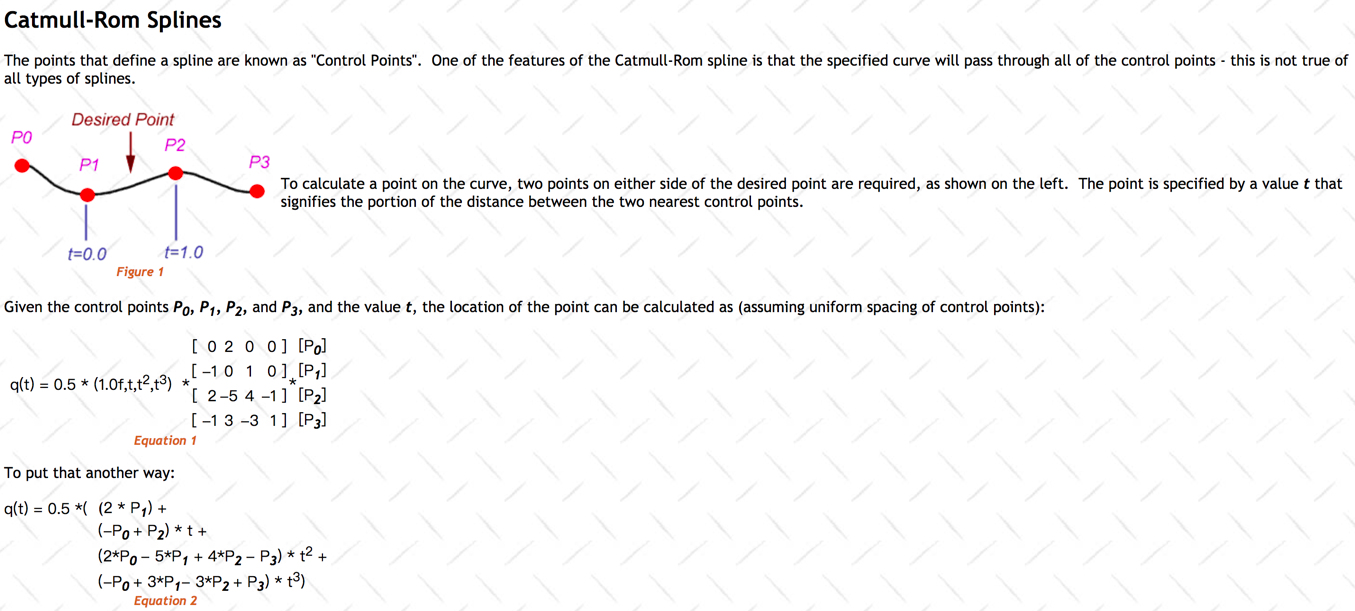
* A very large (and growing) database of **built-in algorithms and Toolbox** for image processing and computer vision applications
* The ability to read in a wide variety of both **common and domain-specific image formats**
* The ability to process both still images and video
* MatLab ensures the **image processing steps used are completely documented**, and hence can be replicated
* The source code for all **image processing functions are accessible** for scrutiny and test

# System Realization

**Unity3D Simulation**

1. ***Path setting & Generation***

* In the simulation, we need to set a path for every UAV and vehicles, which is generated by the waypoints we provided. But it's too unreal and robotic if we just simply connect every path points with a straight line. In order to generate smooth path base on the waypoints we provided, I used Catmull-Rom Splines theory.



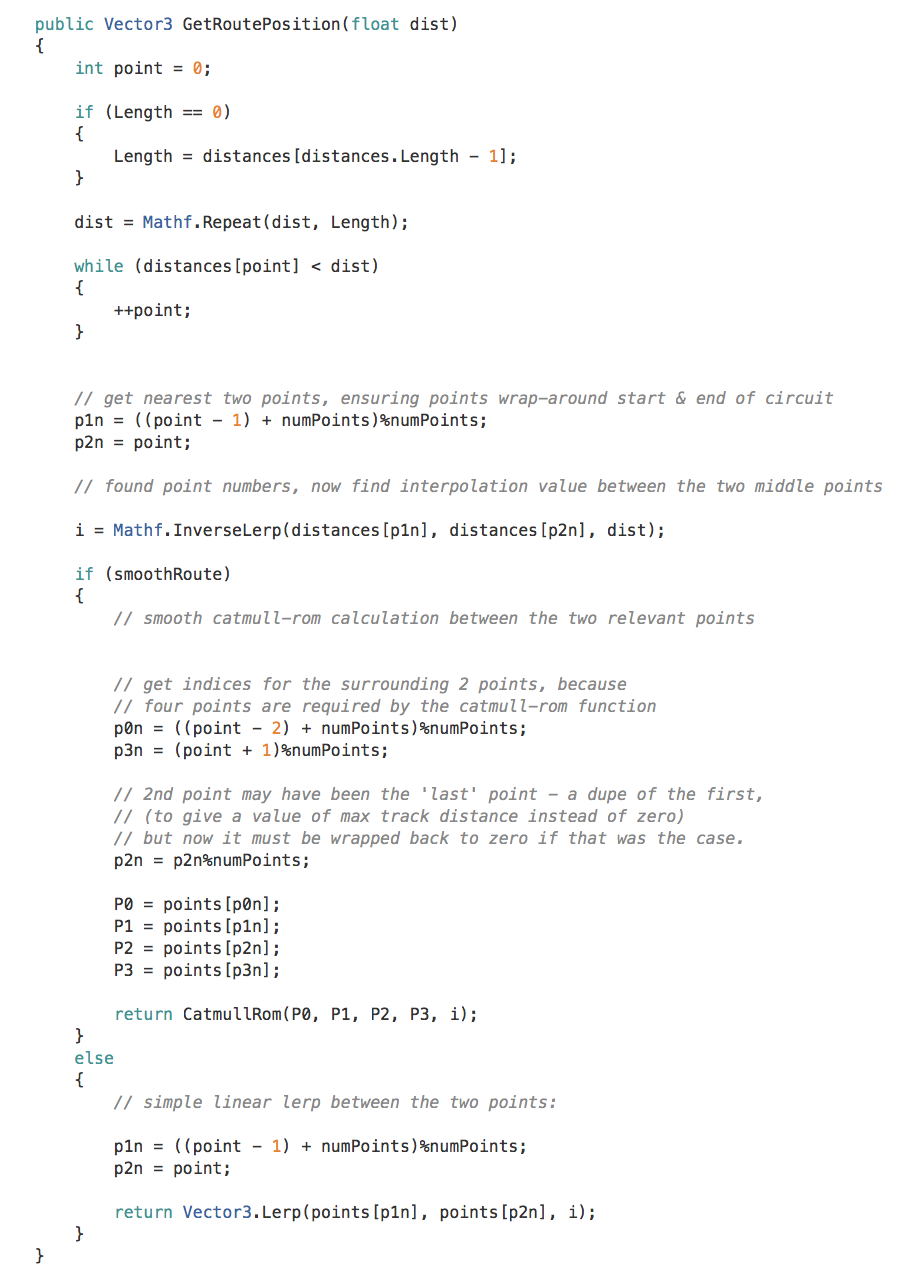
Picture 7. Catmull-Rom Splines Theory

* In order to smooth the route, the process is as follows:

**Input:** Waypoints position **Output:** Smooth path waypoint

1. Get every point position & distance to the start point;
2. Divide total distance into several parts based on smooth parameter (bigger = smoother)
3. For each part distance (equal to each path waypoint)
   1. Get two nearest setting waypoint
   2. Find interpolations value between two points
   3. Get two more surrounding points
   4. Calculate smooth path waypoint
4. Return all the path waypoints.

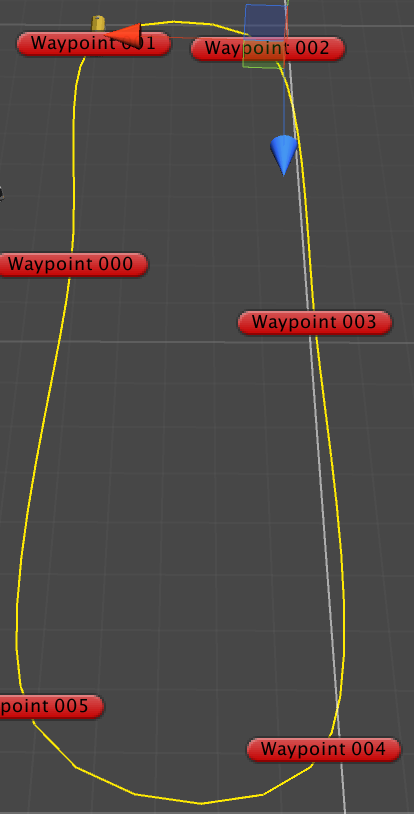
* Result:



Picture 8. Get Smooth waypoint function code



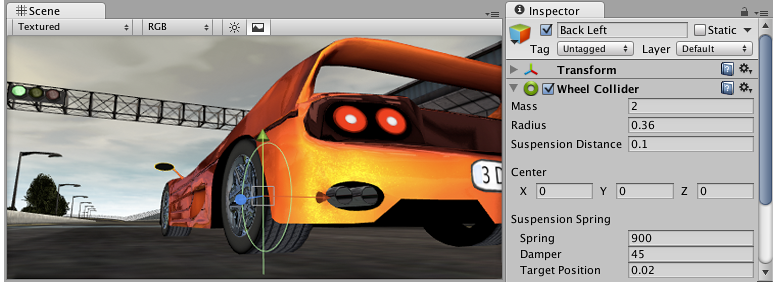
Picture 9. Path without smoothing



Picture 10. Path with smoothing

1. ***UAV & Vehicle Control***

* To know about vehicle control, we first need to know wheel collider in Unity3D. The **Wheel Collider** is a special collider for grounded vehicles. It has built-in collision detection, wheel physics, and a slip-based tire friction model. It can be used for objects other than wheels, but it is specifically designed for vehicles with wheels.



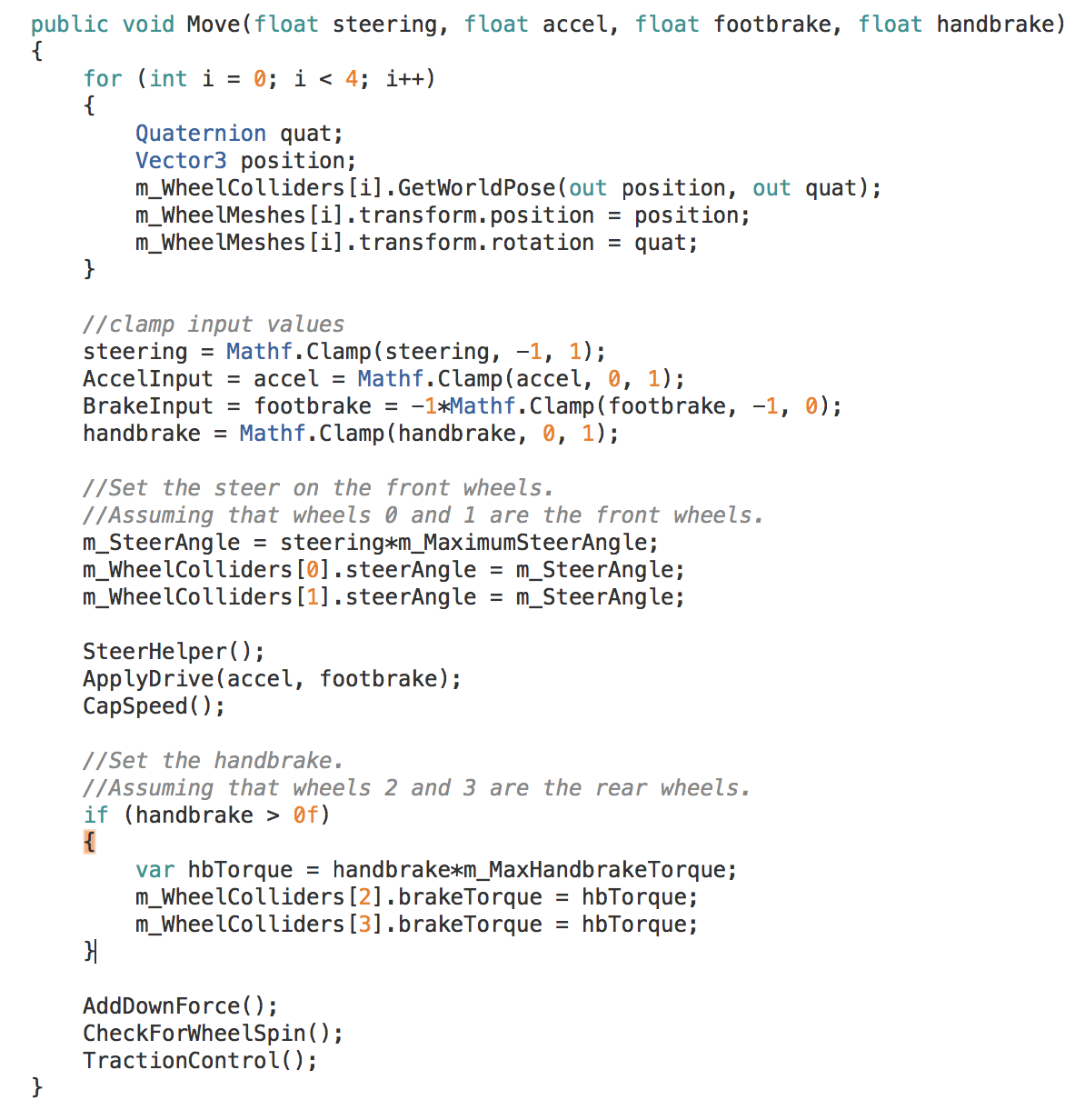
Picture 11. Wheel Collider example

* After introduced wheel collider, the vehicle control change to wheel control, the process is as follows:

**Input:** value of steer, acceleration, footbrake, handbrake **Output:** movement

1. Set the steer on the front wheel
   1. Check if the wheel is on the ground
   2. Fix gimbal lock (suddenly shift direction) by changing car body’s moving direction
2. Apply driving behavior
   1. Add thrust torque to the drive wheel
   2. Speed checking (keep under the Top speed)
3. Traction control
   1. Check for driving wheels’ tire slip
   2. IF slip > threshold value & torque > 0
      1. Then torque reduce
      2. Else torque increase

* Result:



Picture 12. Control function code

1. ***UAV & Vehicle AI***

* According to first two part, we get the path and the method to control the vehicle. The last is about how to control the vehicle and make it follow the path.
* The method I use in this project is to first set a **path tracker** point which will move along the path, then the vehicle will keep moving towards the path tracker. This means that I set a moving destination for the vehicle, so the problem will be simplified to how to **control vehicle move to the certain destination**.
* As follows is the process of set a path tracker on the input path.

**Input:** generated path **Output:** target movement

1. Get controlled object move speed & progress distance
2. For each frame
   1. Generate new position according to preset-ahead distance and move speed
   2. Set target to new position
3. Update current progress distance and position

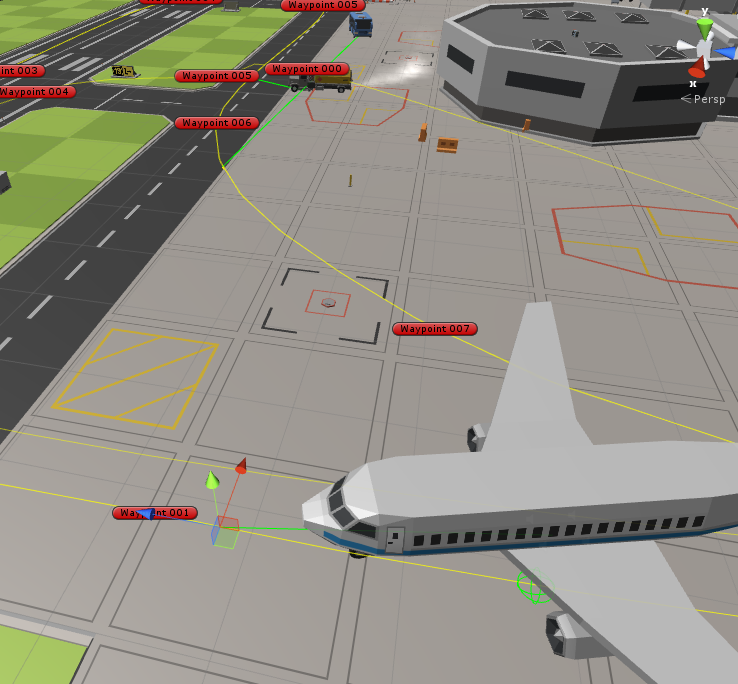
* Next is about controlling vehicle move to the target and slow down when it come close to the target.

**Input:** Control movement & target (waypoint tracker) **Output:** movement control

1. Get distance and angle between target and current position
2. Calculate cautiousness parameter according to distance and angle which decide the desired speed
3. Generate random data to make target to slightly off original position (wander across path-to-target)
4. Calculate steering, acceleration, brake parameter based on angle and distance
   1. Steering angle aim for the target
   2. Acceleration & brake depend on comparison between desire speed and current speed
5. Apply movement control

* Result:

As you can see in the picture, the yellow line is a path, for each path, there is a green line connected vehicle and tracker position.



Picture 13. Tracker and move behavior

**Named Pipe Transmission**

In this project, we use the [NamedPipeServerStream](https://msdn.microsoft.com/en-us/library/system.io.pipes.namedpipeserverstream(v=vs.110).aspx) and [NamedPipeClientStream](https://msdn.microsoft.com/en-us/library/system.io.pipes.namedpipeclientstream(v=vs.110).aspx) classes in .NET Framework to implement name pipe.

1. ***Pipe Server***
2. Create pipe server & while pipe is connected
   1. Create pipe server with name and transmission mode (InOut)
3. While (pipe is connected)
   1. Get image data
      1. Read every pixel’s texture data of every camera in RGB format before rendering
      2. Get raw image data
      3. Change data structure, separate R/G/B into three array
   2. Transfer data
      1. Transfer the size of picture (NamedPipeServerStream.writeByte())
      2. Transfer the R/ G/ B array (NamedPipeServerStream.write())
   3. Get distance data from client
      1. Read bytes from pipe client (.ReadByte())
4. End & pipe.close()
5. ***Pipe Client***

* In Pipe Client, we need use MatLab COM object as a reference to create a MatLab application interface which can access MatLab’s function and workspace.

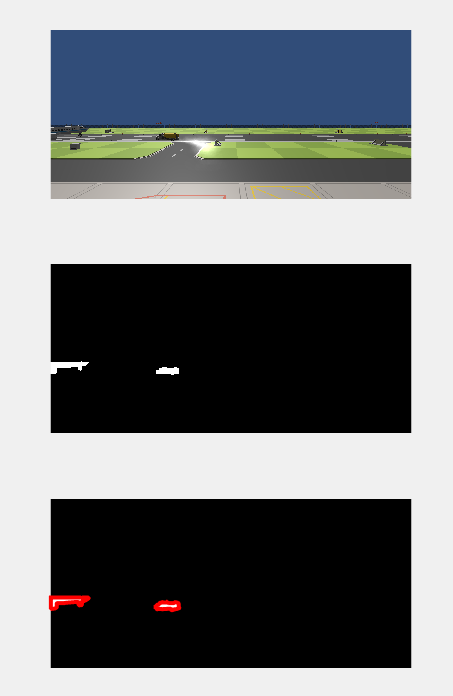
1. Create pipe client & connect to pipe server based on pipe name
2. Create Matlab COM interface load local data (background)
3. While (pipe is connected)
   1. Receive data from pipe server
      1. Get image size first
      2. Get image R/ G/ B array
      3. Check if the stream is null or correct
   2. Use Matlab function
      1. Invoke Matlab function to process image data
   3. Send process result back to server
4. End & pipe.close()

**MatLab Image Process**

* In MatLab, we chose to use image process toolbox to process image data.
* In previous pipe client section, the image data I sent to MatLab is three byte type array which contains red, green, blue color data in reverse order. Because when we try to extract picture data in unity3d, it will read every pixel’s color data and put them in a stack in R|G|B order. So we need to combine and reshape this three arrays into an image matrix first.

1. Reshape R/G/B array into matrix [width, height] based on the picture size
2. Reverse the R/G/B matrix and combine to full image matrix
3. Use rgb2gray convert RGB image to gray image
4. Process, get objects in the image
   1. Background – front
   2. Im2bw change to binary image
   3. Dilate and erode process (fulfill the hole)
   4. Select objects based on area
   5. Draw boundary
5. Calculate distance based on objects’ boundaries
6. Return distance

* Result



Picture 14. Image process result

# Conclusion and Future work

* This project provides a vision-based surveillance method, and use simulation to prove the feasibility of this method. In this project, we use unity3D as simulation tool and MatLab as image data process tool. Meanwhile, we use name pipe to provide data transmission function between unity3d and MatLab process. As a result, the project fulfills it functions requirements as a vision-based operation system.
* On the other hand, we also find some limitation in this project:
  + MatLab (Image process)
    - Multiple objects detection and accuracy: this project is not focused on image processing, so sometimes the distance result will be wrong if there are more than three objects in one camera.
    - Object identification: this project doesn’t include other environment interference and when there are multiple UAV in the scene, we need identification to provide feedback control for different objects.
  + Named pipe
    - Multiple thread management: If you want to create multiple pipes to transfer data, you will need to manage your thread pool to prevent data loss or process crash.
    - Network socket transmission: In real life situation, we will need to transfer data between devices rather than processes. So we can choose to use socket which allows data transmission based on IP address.