UAS Simulator: A Laboratory Set-Up

Abhishek Kasana
Department of Aerospace Engg.
Defence Institute of Advanced Technology
Pune, India
abhikasana1997@gmail.com

Ajay Misra
Department of Aerospace Engg
Defence Institute of Advanced Technology
Pune, India
ajaymisra@diat.ac.in

VPS Naidu
Multi-Sensor Data Fusion Lab
CSIR – National Aerospace laboratories
Bangalore-17, India
vpsnaidu@gmail.com

2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.402

***Abstract*— This work describes the procedure for laboratory setup for Unmanned Aerial System (UAS) Simulator. UAS has three major components viz., UAV (or drone), Pilot and the system in place that connects both of them or interface. This simulator not only helps pilot for training purposes, designers for testing new models, mission planners in planning missions in a different environment but also helps in testing and development of the Synthetic Vision System (SVS). SVS generates a rendered image or 3D image of the flying environment and aware operator using an onboard database of terrain, obstacles, and relevant cultural features.**

Keywords—Flight Dynamics Model, Unmanned Aerial System (UAS), Visual Simulation, Synthetic Vision System. Introduction

I. INTRODUCTION

The contribution of Unmanned Aerial Vehicles (UAV's) continues to increase. In upcoming years, the increase of UAVs uses will not only be seen in defence sector but also in Retail, Transportation, Entertainment, Agriculture, Construction, Search & rescue, Home security, and many others. Software and hardware bugs could provoke a crash, leading to the loss of UAV. Although application testing is one of the most important aspects of the simulators, there is one more purpose of this Simulator, and that is testing and development of the Synthetic Vision System (SVS) for Visual Simulation. When accidents occur in avionics, it is mostly because of reduced situational awareness of the outside world, especially poor visibility. The darkness, fog, and rain are the primary enemies of flight. To overcome these problems, SVS is helpful.

For these reasons, UAS Simulator is useful to verify the performance and behaviour of the UAV before make it run on-board. The pilot must be trained properly and should familiar with the different environmental conditions before flying the real UAV. Thus, a laboratory-based setup of Unmanned Aerial System (UAS) Simulator has been developed and demonstrated at MSDF Lab. With the help of this simulator, the cost of development can also be minimized.

Unmanned Aerial System (UAS) Simulator has three major components (as shown in Fig. 1):

1. Visual Simulation
2. Interface
3. Flight Dynamic Model of UAV

II. FLIGHT DYNAMICS MODEL

Aircraft modelling and simulation have a long history in the aviation industry, and it has become an important part of the aviation industry. A 6-DOF simulation model has been used to test new aircraft designs or for any changes to present

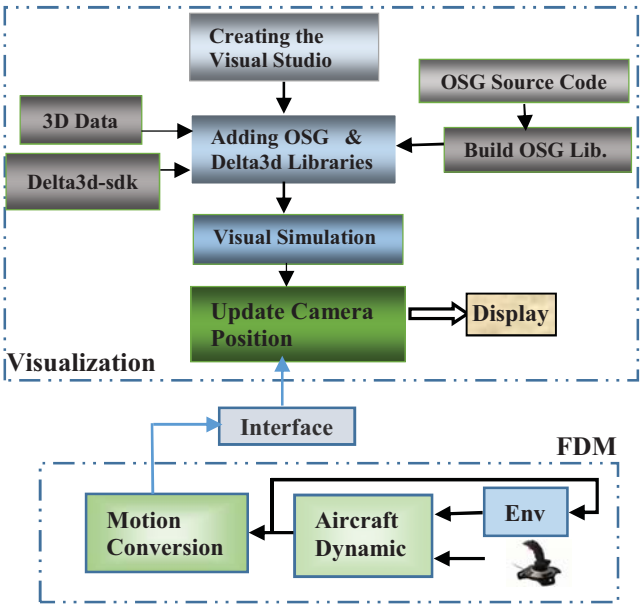

Fig. 1. Block diagram of Unmanned Aerial System

ones in a simulation environment prior to flight tests and this is very helpful when it comes to saving money and time [1]. Fig. 2 shows the information flow diagram of the Simulink model.

Joystick input consists of control surface deflection and rpm of the engine. Pitch input is for the deflection of an elevator, and Yaw input is for the deflection of a rudder and Roll input is for the deflection of an aileron. Flight Dynamic Model (FDM) of any aircraft is based on a set of mathematical equations called a mathematical model.

A. Mathematical Model

The mathematical model of the aircraft includes the Dynamic model, which is based on the set of math equations used to calculate the required state variables and Environmental model, which is based on the set of math equations required to calculate the variations in air temperature, pressure, density, gravity, turbulence and wind shear with respect to the change in altitude [2]. Position coordinates for Visuals should be in terms of longitude,

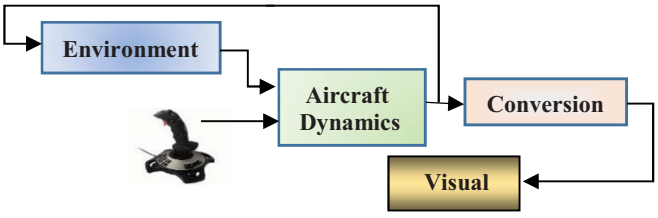

Fig. 2 Block diagram for FDM

978-1-7281-4142-8/$31.00 ©2020 IEEE

latitude, and height. So, a conversion is needed to convert the dimension of X & Y position coordinates from meters to a degree (ref. to Fig. 2). Flight-Gear and Animation tools are present in the Simulink library, which can also be used for Visual.

In the Mathematical model, Newton's equations of motion have been derived relative to a fixed-inertial reference frame. [3]. The total force on the airframe is the sum of gravity force, aerodynamic forces, and propulsion force. Neglect the other forces for simplicity.

i. Gravitational forces

Motion is easily expressed in a body-fixed frame, and gravitational force in body frame is given by,

$$f_g^b = R_i^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} -mgsin\theta \\ mg\,cos\theta\,sin\phi \\ mg\,cos\theta\,cos\phi \end{pmatrix} \quad (1)$$

Since gravity acts through the center of mass of the aircraft, there are no moments produced by gravity.

ii. Aerodynamic forces and moments

The aerodynamic forces and moments are divided into two groups.

a. Longitudinal forces and moment

It includes the forces in the i^b and k^b directions and the moment about j^b axis.

$$F_{lift} = \frac{1}{2}\rho V_a^2 S C_Z(\alpha, q, \delta_e) \quad (2)$$

$$F_{drag} = \frac{1}{2}\rho V_a^2 S C_X(\alpha, q, \delta_e) \quad (3)$$

$$m = \frac{1}{2}\rho V_a^2 S c C_m(\alpha, q, \delta_e) \quad (4)$$

b. Lateral Forces and Moment

It includes the force in the j^b direction and moments about i^b and k^b axes.

$$F_y = \frac{1}{2}\rho V_a^2 S C_Y(\beta, p, r, \delta_a, \delta_r) \quad (5)$$

$$l = \frac{1}{2}\rho V_a^2 S b C_l(\beta, p, r, \delta_a, \delta_r) \quad (6)$$

$$n = \frac{1}{2}\rho V_a^2 S b C_n(\beta, p, r, \delta_a, \delta_r) \quad (7)$$

In longitudinal & lateral forces and moments equation's, the coefficient term can be written as

$$C_X = C_{X0} + C_{X_\alpha}\alpha + C_{X_q}\frac{c}{2V_a}q + C_{X_{\delta_e}}\delta_e + C_{other} \quad (8)$$

$$C_Y = C_{Y0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r + C_{other} \quad (9)$$

$$C_Z = C_{Z0} + C_{Z_\alpha}\alpha + C_{Z_q}q\frac{c}{2V_a} + C_{Z_{\delta_e}}\delta_e + C_{other} \quad (10)$$

$$C_l = C_{l0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r + C_{other} \quad (11)$$

$$C_m = C_{m0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e + C_{other} \quad (12)$$

$$C_n = C_{n0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{other} \quad (13)$$

Where C_{other} is the other coefficients such as engine coefficient, fuselage coefficient, etc. for simplicity neglect other effects (take C_{other} zero).

One can use any notation to represent the coefficients. While lift acts in Z-direction and drags acts in X-direction. So, it's better to take C_X and C_Z notation for drag and lift coefficient. These aerodynamic coefficients are defined in the aerodynamic database of a particular airplane which is obtained from a combination of flight testing, wind tunnel tests, and CFD analysis [4]. Fig. 3 shows the block diagram of Total Forces & Moments calculation. The total coefficient of the body is calculated with the help of the above equations (8-13). Aerodynamic forces and moments are calculated with the help of equations (2-7). Gravity force is calculated with the help of equation (1). This paper does not deal with the Engine model, method for thrust calculation will be discussed later. To derive the dynamic equations of the aircraft (6DOF), newton's second law will be applied first to the translational degrees of freedom and then to the rotational degrees of freedom [2]. The environment model consists of the Atmospheric model, Gravity model, and Wind model. Fig. 4 shows the block diagram of the Environment model.

Atmosphere model, since most of the airplane's flight envelopes are constrained to the troposphere, which is found up to an altitude of about 11,000 m. The International Standard Atmosphere (ISA) model is used to determine the variation in airflow properties with height within the troposphere [1].

Gravity model, gravity or acceleration due to gravity (g) changes with respect to altitude, and so does the Gravity force [5].

Wind model, in a real scenario, wind turbulence, wind shear and gust will always present in the environment [3].

B. Simulink Flight Dynamics Model

The assumption for solving the above Mathematical model in Simulink are:

- Aircraft is a rigid body and having constant mass.
- Earth is flat & non-rotating.
- The X-Z plane is a plane of symmetry.
- The thrust line passes through the C.G point.

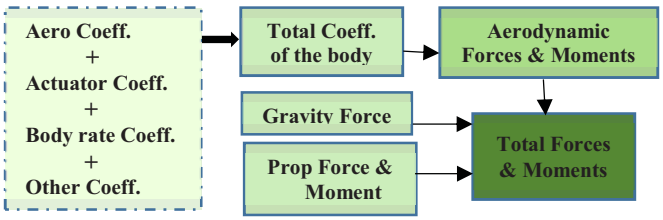

Fig. 3 Total Forces & Moments Calculation Blockset

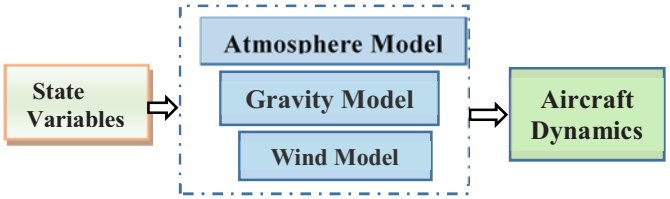

Fig. 4 Environmental model Sub-blocks

Matlab 2015a Simulink library is used to make FDM of Cessna 182 Skylane. Many of the Blocksets are available in Aerospace Blockset of Simulink library, which is used to calculate the dynamics of the plane [6].

Fig. 5 shows the working of the Simulink model. Here, P is the pilot input for calculating Actuator Coefficient (At.C) and Thrust Force (T.F). F.P is flight parameter calculation such as dynamic pressure, incidence, sideslip, and airspeed, which are used to calculate Aerodynamic Coefficient (A.C) and Body Rate Coefficient (BR.C). T.C is the Total Coefficient which is used to calculate Aerodynamic Forces and Moments (refer to Fig. 3). E is the Environment calculation (refer to Fig. 4). F&M is the Total Forces and Moments calculation. 6DOF is solving the 6 Degree of Freedom equations to get the state variables. Then, these state variables feed into E to calculate the required environmental variables and this simulation cycle keep going on until simulation stops. Blocksets used in Simulink FDM are as follows:

- Pilot input: Pilot Joystick Blockset under Animation in Aerospace Blockset.
- Atmosphere: ISA Atmosphere Model Blockset under Environment in Aerospace Blockset.
- Gravity: WGS84 Gravity Model Blockset under Environment in Aerospace Blockset.
- Wind: Wind Shear model, Dryden Wind Turbulence (continuous (+q +r)) model and Discrete Wind Gust model Blocksets under Environment in Aerospace Blockset.
- Flight Parameters: Dynamic pressure and Incidence, Sideslip & airspeed Blocksets under Flight Parameters in Aerospace Blockset.
- Signal Control: Bus creator & Bus selector Blocksets under HDL Coder in Simulink are used for controlling the input and output signals of Blocksets.
- Forces & Moments: Aerodynamic Forces & Moments Blockset is used to calculate Total A/D forces & Moments.
- For thrust calculation, Lookup Table is used, which defines the thrust value as a function of throttle input.
- Dynamic Equations: 6DOF(Quaternion) Blockset is used to calculate the State Variables.
- Conversion: FlatEarth to LLA Blockset is used to convert the dimension of X & Y coordinates.

The database of many aircraft is available on the internet, and the basic workflow is the same, which is discussed above for making the FDM of any aircraft.

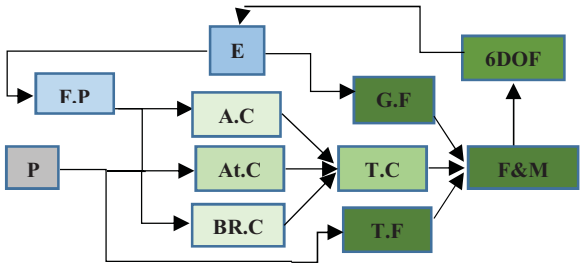

Fig. 5 Workflow of Simulink FDM

III. Visual Simulation

Nowadays, Aviation has reached a remarkable level of safety, and it is just because of technology improvement and constant efforts of pilots and crews. To reduce the accidents due to lack of situational awareness, current technology is beginning to provide solutions. Among the latest technologies, Integrated Enhanced Synthetic Vision System (IESVS) is the most popular, and it is an integration of EVS (Enhanced Vision System) and SVS (Synthetic Vision System). EVS creates real-time images with the help of camera sensors (e.g. near-infrared cameras, millimetre-wave radar) to provide vision in poor visibility conditions [7]. SVS creates a rendered image of the external visual environment using an onboard database of terrain, obstacles, and relevant cultural features in a manner comparable to how it would seem to flight crew if the visibility would not be poor [8, 9].

In this work, Synthetic image will be used for rendering. Database rendering requires the current position of the UAV, which is described by the six-state variables (Lat, Long, Height, Theta, Psi & Phi). Database rendering is done with the help of Delta3d and OpenSceneGraph (OSG) libraries. Fig. 6 shows the workflow to obtain visual simulation.

OSG is a high-level open-source 3D graphics toolkit. It is written in C++ and uses OpenGL as its underlying rendering API [10, 11]. Open Graphics Library (OpenGL) is a graphics API, it allows user to access graphics possessing unit (GPU) which is basically the graphics card of the system. OpenGL is one of many available API, such as Direct3D, Vulkan, Metal, etc. [12]. In simple language, to render a pixel on a screen, one needs to communicate with the GPU. To do so, one needs a medium, and this medium is called OpenGL.

Delta3d is basically a terrain rendering engine, and OSG is a support library for delta3d to add features in database rendering software [13]. So to use these libraries, a systematic sequence of steps should be followed. These steps are as follows (refer to Fig. 6)

- Download OpenSceneGraph 3.2.0 (OSG) source code [14] and delta3d 2.8.0 sdk [15].
- Download CMake. Then, open Cmake-gui.exe and browse the source code folder and set the build folder. Then hit configure and then generate [16].
- Go to the build folder and search for .sln file and open it. First built the ALLBUILD and then build INSTALL [17].
- Open the Microsoft visual studio 2010 and create a new project [18].

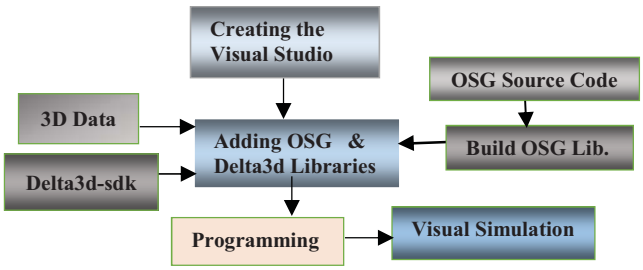

Fig. 6 Visual Simulation Overview

- Link the include folder of OSG Built and delta3d sdk under Additional Include Directories in the project properties.
- Link the lib folder of OSG Built and delta3d under Additional Library Directories in the project properties.
- Add Additional Dependencies in Input, which are the name of the libraries called during debugging.

Steps for loading 3d data with the help of libraries which are discussed above for visual simulation are as follows

- osgDB::readNodeFile is the function that handles the loading of the 3d data with the help of the plugin.
- The above function returns a pointer to a top node of the model's scene graph.
- Write a program to create the root of the scene graph by creating a group object and then add the loaded 3d data as a child to that group.
- Write the program to create nodes for the elements such as light source, textures, objects, effects, rain, etc. and then add them to the root.
- Write the program to create the viewer which is responsible for displaying the scene graph and associate it to a scene graph.
- Write the programming for rendering loop which will draw a new frame on the screen.

IV. Set-up

A flight simulator is exactly what it sounds like, it is a software built to simulate the real experience of flying an aircraft by using a joystick [1]. The flight simulator is divided into two main parts, as shown in Fig. 7.

A. Interface

The interface between the FDM Model of Simulink and Visual Simulation is done by UDP socket communication (refer Fig. 9). Although the interface is wireless in real life but this is a laboratory setup, as both FDM and visual simulation are running in the same system which can be conveniently connected to each other through UDP socket communication and this interface can easily be replaced by the wireless interface with the help of transmitter and receiver [19]. User Datagram Protocol (UDP) is a communication protocol that facilitates the exchange of messages between computing devices in a network [20]. The reason for choosing UDP is, it is faster as compared to TCP and there is no error checking for packets.

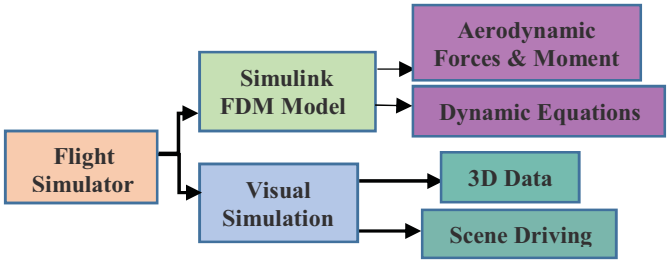

Fig. 7 Major parts of Flight Simulator

Steps for developing the Interface are:

- Put the starting position as Lat, Long and Height during conversion in FlatEarth to LLA Blockset (see Fig. 8).
- Send the Navigational data (Lat, Long, Height, Theta, Psi & Phi) to a particular IP address and IP port with the help of UDP Send Blockset (see Fig. 8).
- UDP send Simulink blockset byte order is Little Endian [21].
- This sending data is in World Geodetic System (WGS) coordinate system, and the 3D data is created by using North East Up (NEU) coordinates. So, write C++ programming for conversion.
- Conversion from a geodetic coordinate system to the ENU coordinate system is divided into two parts [22].
- First, convert geodetic coordinates (latitude φ, longitude λ, height h) to ECEF coordinates from the following formulae.

$$X = (N(\phi) + h)\cos\phi\cos\lambda \quad (14)$$

$$Y = (N(\phi) + h)\cos\phi\sin\lambda \quad (15)$$

$$Z = \left(\frac{b^2}{a^2}N(\phi) + h\right)\sin\phi \quad (16)$$

Where, $$N(\phi) = \frac{a^2}{\sqrt{a^2\cos^2\phi + b^2\sin^2\phi}} \quad (17)$$

- Where a and b are the equatorial radius and the polar radius. The normal $N(\phi)$ is the distance between the surface and the Z-axis along the ellipsoid normal.
- Second, ECEF coordinates to ENU coordinates. To transform from the ECEF coordinate system to ENU coordinate system, a reference point is taken so that if the database is located at $\{X_r, Y_r, Z_r\}$ and the runway at $\{X_p, Y_p, Z_p\}$ then the vector pointing towards the runway from the database in the ENU coordinate frame is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin\lambda_r & \cos\lambda_r & 0 \\ -\sin\phi_r\cos\lambda_r & -\sin\phi_r\sin\lambda_r & \cos\phi_r \\ \cos\phi_r\cos\lambda_r & \cos\phi_r\sin\lambda_r & \sin\phi_r \end{bmatrix} \begin{bmatrix} X_p - X_r \\ Y_p - Y_r \\ Z_p - Z_r \end{bmatrix} \quad (18)$$

- Write the UDP receive socket programming in C++ to receive the Navigational data from the Simulink model (see Fig. 9) and convert the little-endian byte order.
- Create a thread to receive the data continuously. Update the camera position or frame according to the navigational data, OpenGL is recommended.

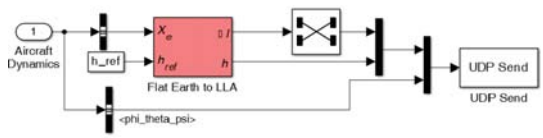

Fig. 8 Simulink side Interface

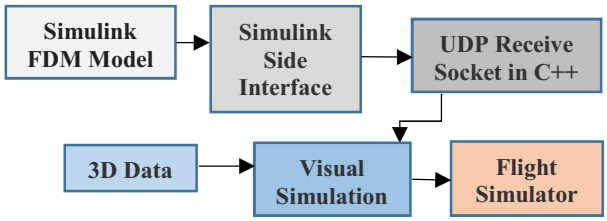

Fig. 9 Interface of Flight Simulator

B. *Software and Hardware Required*

- Processor: Intel i5 CPU 870 @2.40GHz
- RAM: 4 GB
- System type: 64-bit Windows 7 OS
- OpenGL 4.0
- OpenSceneGraph 3.2.0
- Delta3d 2.8.0
- CMake 3.15.6
- 3D data
- Microsoft Visual Studio 2010
- Aerospace Blockset
- Microsoft Joystick
- Matlab Simulink 2015a

First, run the visual simulation code and wait for the visual window to appear and UDP socket connection to form. Then run the Simulink model.

V. Results & Discussion

The camera's dynamic in visual stimulation is the same as the Simulink model (Cessna 182) dynamic's, and both are controlled using the joystick. The starting point of visual simulation is defined in the Simulink model, the starting coordinates of XYZ runway is taken as the starting position of the visual simulation, as shown in Fig. 10. OpenSceneGraph and delta3d are efficient tools for data rendering and Visual Simulation. The followings would be the advantages of using this simulator. The visual simulation is comparable to the Synthetic Vision System of aircraft, this gives the real-time experience. The tunnel in the sky approach can also be designed and developed in this visual simulation. The auto-land monitoring concept can also be designed and developed using this simulator. This simulator gives a free hand to design and test any FDM. Control laws such as Autopilot can also be tested in this simulator. A pilot can be trained for poor visibility conditions with the help of Synthetic Visual Simulation of this Simulator.

VI. Conclusion

In this paper, the Unmanned Aerial System (UAS) Simulator has been developed and implemented. Cessna 182 Skylane Flight Dynamics Model (FDM) has been used to test UAS simulator to demonstrate 3D Synthetic Visualization.

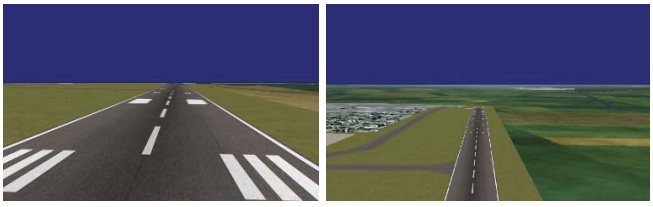

Fig. 10 Starting point &Take-Off view of XYZ runway

Complete steps in detail have been discussed for developing a Simulink model and creating a Visual simulation of 3D data. This simulator is very simple and can be used to demonstrated UAV auto-land monitoring, SVS studies and educational purposes.

VII. References

[1] David Allerton "Principles of Flight Simulation", John Wiley & Sons, 2009.

[2] M.V Cook., "Flight Dynamics Principles", Elsevier Linacre House, Burington, MA, 2007.

[3] Randal W. Beard & Timothy W. Mclain, "Small Unmanned Aircraft", Princeton University Press, New Jersy, 2012.

[4] Marcello R. Napolitano, "Aircraft Dynamics from Modeling to Simulation", John Wiley & Sons, Inc, Hoboken, 2012.

[5] Howard Curtis, "Orbital Mechanics for Engineering Students", Elsevier Butterworth-Heinemann, Burlington, 2005.

[6] "Aerospace Blockset user guide for use with Simulink". ,https://edulab.unitn.it/nfs/Manualistica/Software/MathWorks%20Guide/aeroblks/aeroblks.pdf accessed on 27th December 2019.

[7] Xiaoyan Hu, Sun Bo, Zhao Huiqin, Xie Bing, Wu-Hao, Hu Ge, "Visual Simulation System for Flight Simulation Based on OSG". College of Information Science and Technology Beijing Normal University, Beijing, China, 2010.

[8] Parrish, Russell V. "Aspects of synthetic vision display systems and the best practices of NASA's SVS project". NASA-TP-2008-215130 May 2008.

[9] "Synthetic vision system – from sandbox to reality", Military of Embedded system, September 18th, 2007.

[10] Rui Wang & Xuelei Qian, "OpenSceneGraph 3.0 Beginner's Guide". Packt (2010).

[11] Rui Wang & Xuelei Qian, " OpenSceneGraph 3 Cookbook". Packt 2012.

[12] Dave Shriner, Graham Sellers & John Kessenich, "OpenGL Programming Guide". Pearson Education, Inc. New Jersey, 2013.

[13] Pery McDowell & Rudolph Darken, "Delta3d: A complete open-source game & simulation Engine for building military training system". Defense & modelling Simulation, July 2006.

[14] Download the OpenSceneGraph source code from http://www.openscenegraph.org/index.php/download-section/stable-releases/146-openscenegraph-3-2-release-downloads accessed on 27th December-2019.

[15] Download the delta3d source code from http://sourceforge.net/projects/delta3d//files/delta3d/2.8.0/ accessed on 27th December-2019.

[16] John Lamp, "Cmake Tutorial". Johnlamp.net, 2017. https://www.johnlamp.net/files/CMakeTutorial.pdf accessed on 27th December 2019.

[17] Michael Halvorson, "Microsoft Visual Basic 2010". Microsoft Press, Washington, 2010.

[18] OpenSceneGraph website for building the source code: http://www.openscenegraph.org/index.php/33-openscenegraph accessed on 27th December-2019.

[19] Srikanth A & VPS Naidu, "MAS Simulator: A Laboratory Set-Up". International Conference on Cognitive Computing and Information Processing(CCIP), March Noida, 2015.

[20] Kameswari Chebrolu, "Socket Programming". Dept. of Electrical Engineering, IIT Kanpur, 2012. http://home.iitk.ac.in/~chebrolu/ee673-f06/sockets.pdf accessed on 27th December 2019.

[21] About UDP Send Blockset: http://www.mathworks.com/help/dsp/ref/udpsend.html accessed on 27th December-2019.

[22] Geodetic to ECEF coordinate conversion, http://en.wikipedia.org/wiki/Geographic_coordinate_conversion , accessed on 27th December-2019.