Defence Science Journal, Vol. 75, No. 6, November 2025, pp. 734-741, DOI : 10.14429/dsj.20874

Modeling of Terrain Awareness and Warning System (TAWS) for Fixed-Wing Aircraft

Akshat Jani*, M. Sarath Chandra and S. Charulatha

School of Aeronautical Sciences, Hindustan Institute of Technology & Science, Chennai – 603 103, India
**E-mail: akshatjani27@gmail.com*

ABSTRACT

In this paper, the Terrain Awareness and Warning System (TAWS), a system responsible for guiding aircraft safely into terrain, has been modeled, analyzed, and simulated using MATLAB and Simulink. Firstly, the Virtual Reality (VR) environment has been created by generating terrain using the elevation values extracted from an open-source terrain database and adding an aircraft controllable through Simulink. Secondly, an algorithm to predict the aircraft's position after a specified time interval has been developed. It also extracts and returns the terrain elevation value at the predicted position and compares it with the aircraft's altitude. Finally, suitable aural and visual warnings have been generated and displayed in the animation window based on comparison. The model can further be integrated into a real-time flight simulation environment serving various purposes, such as design and development, concept demonstration, system assessment, test and validation, and so on.

Keywords: Terrain awareness and warning system; Enhanced ground proximity warning system; Controlled flight into terrain; Forward-looking terrain awareness; Collision avoidance; Simulink 3D animation; Flight simulation

NOMENCLATURE

TAWS	: Terrain Awareness and Warning System
VR	: Virtual Reality
CFIT	: Controlled Flight into Terrain
GPWS	: Ground Proximity Warning System
EGPWS	: Enhanced Ground Proximity Warning System
GPS	: Global Positioning System
FLTA	: Forward-Looking Terrain Awareness
6-DOF	: 6-Degrees of Freedom
DEM	: Digital Elevation Model
PNG	: Portable Network Graphics
TIFF	: Tagged Image File Format
MFD	: Multi-Functional Display
VRML	: Virtual Reality Modelling Language
USGS	: United States Geological Survey
WRL	: World
UAV	: Unmanned Aerial Vehicle

1. INTRODUCTION

In the history of aviation, Controlled Flight into Terrain (CFIT) has been one of the major causes of fatal accidents. In response to such incidents, the industry developed the Ground Proximity Warning System (GPWS), which automatically warns pilots when the aircraft dangerously approaches the terrain or the ground. When an aircraft is in danger of colliding with terrain, the GPWS produces visual and aural warnings. It does this by measuring the height of the aircraft above the ground using a radar altimeter. It calculates the height based on the time difference between the transmitted and the received pulse of radio frequency. Since 1974, the GPWS has been mandated for larger airplanes, resulting in a significant reduction in accidents pertaining to CFIT. In the year 2000, the requirements have been also mandated for smaller general aviation aircraft. Although the GPWS system has been very successful as proven by the marked reduction in the accident rate, it has a few limitations. Firstly, it can only detect the terrain right underneath the aircraft, meaning it has a small terrain-capturing range. Generally, it has a height span ranging from 20 feet to 1500 feet. Suppose there is a sharp change in terrain (for instance, an increase in terrain elevation), it cannot detect this until it is too late for the crew to react. Even during landing(s) & takeoff(s), the GPWS is unable to provide forward terrain awareness due to which pilots have to rely on visual navigation during those phases.

To overcome the limitations of GPWS, a more advanced technology, named Enhanced Ground Proximity Warning System (EGPWS), has been introduced. It makes use of the worldwide digital terrain database to produce a virtual 3D map of the terrain around the aircraft. It integrates the aircraft's navigation system (ideally, Global Positioning System - GPS) with the terrain database. The aircraft's instantaneous position provided by the GPS is compared with the earth's coordinates from the terrain database and, if any discrepancy is found after comparing, suitable cautions and warnings according to the nature of the hazard are provided to the pilots. The EGPWS is generically known as the Terrain Awareness and Warning System (TAWS). It extends the GPWS features by providing a wider range of focus as well as forward terrain awareness (conventionally known as Forward Looking Terrain Awareness

Received : 29 October 2024, Revised : 28 March 2025
Accepted : 30 April 2025, Online published : 04 November 2025

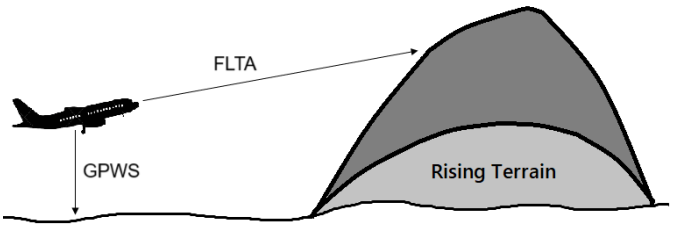

Figure 1. Schematic of FLTA in Comparison with GPWS.

as shown in Fig. 1), which facilitates pilots even during the critical flight phases, such as landing (it can provide warning coverage even up to runway threshold) and takeoff.

In the current study, a representative model of TAWS for a generic aircraft has been developed using MATLAB & Simulink. There are very few previous works that simulate various guidance and navigation systems for aircraft using the aforesaid tools. Hence, the aim is to model, analyze, test, and simulate the system by blending MATLAB code with simplified Simulink block diagrams. Along with section 1, which is the 'Introduction', this paper is divided into five main sections:

Section 2: It describes the steps involved in the generation of the terrain into MATLAB 3D World Editor using the Terrain-Elevation grid (sample values) taken from the Terrain Visualization example from the MATLAB library. It can be done in two different approaches. The first way is through the MATLAB code, and the other is through 3D World Editor by creating a virtual world using different nodes. Using the nodes, many features of the virtual world, such as appearance, geometry, sensors, navigational information, and so on, can be specified. This section also describes the process of integrating the aircraft model from the MATLAB library template with the terrain in the same editor window.

Section 3: This section emphasizes the ideas, logic, and processes behind the development of the algorithm for the prediction of aircraft's position and altitude and compares the same with the terrain's altitude. The output from the Algorithm Subsystem is later fetched into the display subsystem to generate appropriate warnings. This section is further divided into two sub-parts:

Section 3.1: It provides a brief description of the concept and steps involved in developing the code for estimating the predicted position of the aircraft based on the initial aircraft coordinates. It also highlights a few simplified formulas, which are used in the code.

Section 3.2: The predicted position coordinates of the aircraft are further used in the second part of the algorithm to fetch out terrain elevation data at that point. Once the terrain elevation is extracted, the aircraft's predicted altitude and terrain elevation are compared using the comparator block to return binary output (1-high for any discrepancy or 0-low for no discrepancy).

Section 4: The output, as described in Section 3.2 of the algorithm, is taken as an input to the display subsystem where multiple switches are used to activate various loops for aural and visual warnings, whenever the input is high (value is 1) in case of warnings. The output from the display system is fetched into the VR Sink block where the real-world dynamic simulations are produced after the computations. The suitable warnings are then displayed in the animation so that the viewer can get a concise idea about the system. The final output is the 3D animation of the real-world scenario where the aircraft flies, surrounded by terrain and intended aural and visual warnings, are generated based on the algorithm.

Section 5: This section provides a brief overview of the procedure to integrate the present Simulink Model into a real-time flight simulation environment for development and testing purposes.

The motivation to develop a Forward-Looking Terrain Awareness (FLTA) has been taken from Honeywell Inc. pilot's guide for EGPWS[1] and Xiao[2], *et al.*. The function used for generating the contour map described in Mullins[3], *et al.* has been slightly modified, and colormaps are generated. Moreover, Mullins[3], *et al.* represent the overview of TAWS, which has been taken as a reference (blueprint) to develop the block diagrams. The format of a terrain-elevation grid corresponding to the latitude/longitude of the region, as conceptualized by Gellerman[4], *et al.*, has been extended and implemented in MATLAB code in a more simplified way. Various terrain-generating algorithms, as described in Raj[5], *et al.*, represent the 3-dimensional maps generated using MATLAB. A similar idea is applicable in the MATLAB 3D World Editor, where the generated terrain is used for the animations.

This model can be considered as the experimental test bench, which provides an enhanced version of Mode-2 alerts[6] using the FLTA algorithm. This paper also forms the basis for developing other complex modes of EGPWS (up to Mode-6)[6] with necessary add-ons. From the discussion on suitability between cluttered and less-cluttered cockpit displays[7] and composite display design[8], the paper illustrates the integrated 3-dimensional maps with a miniature aircraft changing its color during warnings, as presented in Simulink Visualization Block resembling Multi-Functional Display (MFD) of the cockpit.

The results from this paper suggest the integration of this model into real-time aircraft systems or flight simulators with necessary programming, enhancing the processing/storage capacity and the computation time of the onboard flight computers. A similar example is the implementation of the Simulink Model into NASA's Vertical Motion Flight Simulator (VMS)[9]. This experimental test bench can be incorporated in such simulators for concept demonstration and testing, which can include assessments of handling qualities, flight control systems design, guidance and displays, and so on. The FLTA algorithm implemented in this model assumes a straight line ahead of the aircraft to predict the terrain and the aircraft's coordinates. However, in future studies, the authors plan to make it more resilient by modifying the boundary of the forward-looking alarm area such that it remains trapezoidal when the aircraft is flying straight and becomes pipe-shaped[10] when the aircraft is rolling left or right, thus making the algorithm more robust.

2. CREATING A VIRTUAL REALITY (VR) ENVIRONMENT

The data required for the terrain generation is primarily the elevation data corresponding to the 2-dimensional coordinates (latitude and longitude). In this case, the elevation values corresponding to the respective VRML (Virtual Reality Modelling Language) coordinates have been assigned. The sample elevation values have been taken from the Terrain Visualisation example by The MathWorks Inc. Moreover, according to one's choice, the elevation data for any region of the world can be found in one of the databases as mentioned in the Simulink Mapping Toolbox User Guide.

In this case, the sample elevation values taken from the MATLAB example are extracted from the United States Geological Survey (USGS) database whose specifications can be found in Table 1.

Table 1. Specifications of USGS-DEM File

Dataset	Digital Elevation Model (DEM)
Scale	1:24,000
Spatial resolution	1-arc sec. (~30 m)
Units of elevation	M
Region	Nearby San Francisco

The scale of 1:24,000 means for every 1-meter distance covered on the map generated, approximately 24,000 m are covered on the ground in a real-time scenario. The spatial resolution depicts the spacing of 30 m between the tiles (elevation values are indexed every 30 m in X and Z directions, starting from the origin). Further details about indexing can be found in Section 2.1.

Since the primary aim of this paper lies in the design and implementation of the algorithm and display system, for simplicity, the code for the terrain generation has been referenced from the MATLAB built-in example called "Terrain Visualization". The steps for importing and generating the terrain as implemented in this paper are as follows Algorithm 1.

Algorithm 1. Terrain Elevation Data Generation

1. Unzip (Graphical Zip File) the DEM data file to a temporary directory
2. Read every point of the 1:24,000 file in the standard format using the suitable MATLAB function.
3. Delete the temporarily created unzipped file.
4. Prepare the data for the creation of a Virtual World by modifying it.
5. Assign spacing of 30 m in X and Z directions.
6. Bring up the sample template from the MATLAB library and create a handle for the node that will contain the DEM data.
7. Create necessary node fields for configuring the shape, appearance, material, and so on of the terrain.
8. Using the appropriate MATLAB function, create a texture for the given elevation data and scale it if required.
9. Save the texture file in the .PNG format in the working directory.
10. Assign the texture to the appearance field of the terrain node.
11. Save the terrain file in ".wrl" format.

Once the code is completed, the file is saved with an extension of .wrl (Virtual Reality Modelling Language format). The file is opened and further modification is done in MATLAB 3D World Editor, an addon built specifically to create or edit the VR models to be used for simulations. The terrain is generated in the tessellated form (shape patterns that fit perfectly with each other without any gap).

2.1 Indexing of the Elevation Values

The indexing operation can be performed using the "ElevationGrid" node feature of the Editor. It is useful for a variety of purposes, such as creating terrain landscapes, building squared-off walls of rooms of the house, and so on. It maps a surface function of two sampled input values after taking the rectangular 2D array as an input (elevation values in this case). As stated in Brutzman and Daly (2007)[11], the Elevation Grid is sized by xDimension and zDimension fields. Hence, the elevation (height) field must always contain the (xDimension × zDimension) number of values. The values in the xSpacing and zSpacing determine how many m apart each point is placed in the X and Z directions. Thus, the overall footprint of the ElevationGrid node is (xDimension × xSpacing) by (zDimension × zSpacing) m squared.

2.2 Adding the Aircraft in the VR Environment

Algorithm 2. Add a Miniature Aircraft to the Map

1. Open the WRL file containing the terrain in 3D World Editor.
2. Import the suitable aircraft files from the inline models present in the MATLAB/Simulink library.
3. Store the aircraft data in a transformation node such that its sub-nodes contain data (geometry, appearance, material, and so on) for aircraft parts and rename it.
4. Scale up the aircraft by a factor of 30 in all three dimensions to make it visible in the terrain. (However, real-life scenarios may be different).
5. Add a viewpoint node as a sub-node to the parent node containing the aircraft data.
6. Tune the data for the viewpoint node such that it acts as a camera following the aircraft during simulation.
7. Look for sub-nodes containing the data of fuselage (geometry, appearance, and material) in the parent node.
8. Configure the fuselage appearance such that it changes its color on getting the intended input from the Simulink model (in case of warnings) (Fig. 6).
9. Close the Editor and save the file in ".wrl" format.

2.3 Input from User

Static input is given to the aircraft model using constant, moment, and sliderblocks, which provide constant forces and moments that allow the aircraft to fly as per the values provided. The flight parameters data, such as position, Euler orientation, velocity, and mass, are assigned in block parameters of

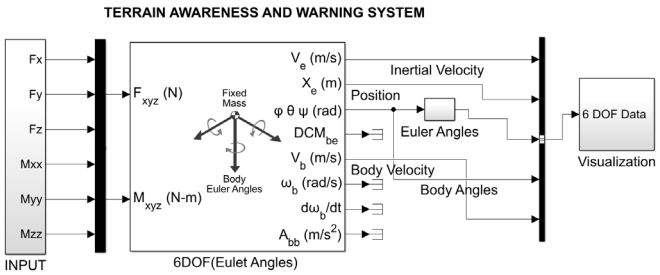

Figure 2. Simulink block diagram consisting of input, 6-DOF block, and visualisation (output) subsystem.

the 6-DOF (Euler angles) block, which remains the same throughout the computation. This simplifies the algorithm for predicted path calculation and interpolation of elevation data from the terrain. As shown in Fig. 2, the 6-DOF block takes the input from the INPUT subsystem and static input values from the data provided in the block parameters dialog box; a schematic of the same is shown in Table 2.

2.4 Six (6)-DOF Block with Euler Angles

The 6-DOF (Euler angles) block implements the Euler angle representation of the equations of motion for six degrees of freedom. 6-DOF (Euler angles) block takes into account the rotation of the body's fixed coordinate system (Xb, Yb, Zb) around the earth's flat reference system (Xe, Ye, Ze).

Table 2. Block parameters dialog box (tabular view) with input values

Block parameters	Value
Units	Metric (MKS)
Mass type	Fixed
Representation	Euler angles
Initial position in inertial axes [Xe, Ye, Ze]	[8500 0 -850]
Initial velocity in body axes [U,v,w]	[0 -75 0]
Initial Euler orientation [roll, pitch, yaw]:	[-0.05 0.1 -0.1]
Initial body rotation rates [p,q,r]	[0 0 0]
Initial mass	1.0
Inertia	Eye(3)

The origin of the body's fixed coordinate system is the center of gravity of the aircraft. Assuming the body is rigid, this assumption does not need consideration of the forces acting between the different mass elements. The flat earth reference frame is considered to be inertial. The 6-DOF block considers the forces/moments acting on the aircraft along with the flight parameters data as input and gives inertial velocity, body velocity, and Euler angles (psi, theta, phi) as output, which are stored in outport (6-DOF data) in the form of signals in the Simulink.

2.5 Transformation of Coordinates and Orientation

The Euler angles and position coordinates are converted into axis angles and VR coordinates (compatible with the created VR Environment) using *Euler to Axis Angle* and X_e *to* X_e *VR* blocks taken from MATLAB example of NASA HL-20 model.

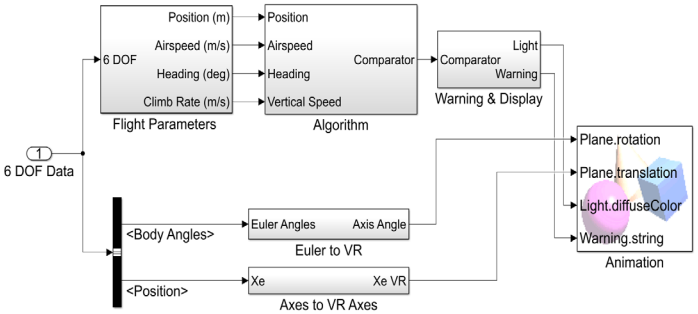

Figure 3. Overview of various subsystems forming the part of TAWS.

Euler to Axis angles: It transforms the 6-DOF position coordinates into VR coordinates.

X_e to X_e VR axes: It transforms the Euler angles suitably to VR axis angles.

VR Sink Block: It produces animation based on the inputs given from the Simulink model.

From the 6-DOF data, flight parameters such as climb/descent rate (for example, vertical speed), position, heading, and airspeed are computed. All the parameters are in the SI units. These values are stored in the flight parameters block and are given as input to the Algorithm Subsystem block, which contains the MATLAB function block, interpreted MATLAB function block, comparator, and so on. The algorithm subsystem calculates the aircraft's predicted position/altitude and the terrain's altitude (at the predicted aircraft's position). A detailed review of the Algorithm Subsystem and Warning/Display Subsystem is described briefly in the succeeding sections. The overview of the subsystems mentioned is shown in Fig. 3.

3. Developing the Algorithm

As stated in the introduction part, the algorithm is divided into two subsections. The first subsection contains detailed information related to the calculation for the aircraft's predicted position, while the second section describes briefly the logic and method to fetch the terrain's altitude from the terrain elevation grid at the points corresponding to the aircraft's predicted position coordinates. The 'MATLAB Function Block' and 'Interpreted Function Block' are used for integrating the MATLAB code into the Simulink model. Figure 4 represents the block diagram inside the Algorithm Subsystem block.

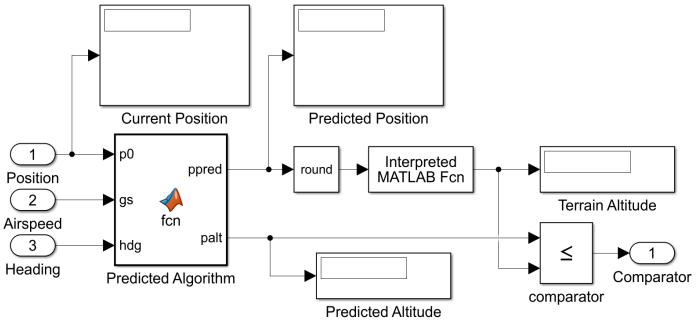

Figure 4. Block diagram of the algorithm subsystem.

3.1 Predicting the Aircraft's Position

Some specific flight parameters, such as position, airspeed, heading, and vertical speed, are extracted from the Flight Parameters block and used as input into the MATLAB function block. Before writing the code, it is necessary to provide the number of inputs and outputs information using the syntax**: function(output1, output2, output3, …., outputN) = fcn(input1, input2, input3, …., inputN).** Here, the inputs are the flight parameters as described earlier, and the output is the aircraft's predicted position and predicted altitude assigned to two variables named "ppred" and "palt". The predicted altitude is extracted by indexing the value at the third position from the "ppred" variable (predicted coordinates).

A few assumptions are taken in the calculation of the predicted position of the aircraft. One of them is the consideration of the aircraft to be a rigid mass so that the environmental and other external effects are neglected for the time being. The motion of the aircraft is assumed to be linear. Moreover, the flight parameters are assumed to be constant over the entire computation time. Hence, the initial velocity is taken to be 75 m/s. Also, it is assumed that the predicted position is calculated every 30 seconds (time interval). Applying the generic distance-velocity equations (distance = velocity*time), it is found that the predicted position is calculated for approximately 2250 m (2.25 km) ahead of the aircraft's initial position at every timestep. It has been found quite reasonable since the dimension of the terrain tile in the MATLAB 3D World Editor remains 13.89 km * 11.1 km. However, the size of the terrain tile and the look-ahead distance are scaled down compared to the real-life scenario to reduce the computation time and complexity in configuring the VR environment. The important formulas used for the predicted position calculations are as follows.

(i) $[Pfx, Pfy, Pfz] = [Pix, Piy, Piz] \pm [\Delta t \times (vx, vy, vz)]$
(ii) $[vx, vy, vz] = [gs \times cos\theta, gs \times sin\theta, vs]$
(iii) $Ppredicted = Pinitial \pm \Delta P$
(iv) $\Delta P = time \times [vx, vy, vz]$
where,
Pf = Final (Predicted) Position
Pi = Initial Position
Δt = Time Interval
gs = Ground Speed
vs = Vertical Speed
θ = 90 – Heading Angle
v_x, v_y, v_z = Inertial velocities of aircraft in x, y and z directions, respectively

NOTE: Suffixes *x, y* & denote the values in the *x*, *y*, and *z* directions, respectively.

The MATLAB code[12] in the form of an easily understandable flow chart along with steps is shown in Algorithm 3. Moreover, the code follows the syntax of MATLAB function block and MATLAB script. The longitudinal and lateral velocities are calculated by taking the components of ground speed in X and Z directions, respectively. The value of the heading angle stored in a variable named "hdg" is used for this purpose. The rate of change of altitude of the aircraft is calculated by multiplying the vertical component of the velocity with the time interval. The vertical velocity is the sine of the climb angle times the ground speed "gs".

Algorithm 3. Calculate the Aircraft's Predicted Position and Altitude

1. Read the inputs from SIMULINK in MATLAB Function Block.
   function [ppred, palt] = fcn(p0,gs,hdg,vs)
2. Calculate the angle made by the aircraft in the horizontal direction
   theta = 90-hdg;
3. Calculate the velocities in all the three directions.
   v = [-gs*cosd(theta), gs*sind(theta), vs];
4. Calculate the predicted position for the time interval of 30 sec.
   ppred = p0 - (30*v);
5. Calculate the predicted altitude from the initial altitude and vertical speed.
   palt = -p0(3) + (vs*30);

NOTE: The sign of the velocity vector is based on the predicted position in either direction from the current position.

3.2 Terrain Data Extraction

Once the terrain elevation data is imported into MATLAB, a separate grid for the X and Z coordinates is generated by taking a combination of suitable X and Z values so that the terrain data can be allocated to them in the later stages. The terrain generation code is further extended to accomplish this task. The X grid contains all the X coordinate values indexed in the horizontal sequence. Similarly, the Z grid contains all the Z-coordinate values. The arrays of X and Z coordinates are concatenated to form the coordinate combinations for the grid. Further, the elevation values are assigned to every combination, and, hence, the terrain elevation database is developed, which is adaptable with the 3D Editor coordinates. The MATLAB code developed for the said purpose is also briefly described in Algorithm 4.

3.2.1 Code for Terrain Data Extraction

Algorithm 4. Generating Terrain Database and Retrieving Elevation Values from It

1. Convert all the rows of the elevation grid into columns and assign them to a variable.
   E1 = Z(:);
2. Fill the missing values(NaN) in the grid with the constant value "0". Here, the missing value (NaN) represents that the terrain's elevation is the same as the Mean Sea Level (MSL). Hence, "0" is appropriate for this purpose.
   E1 = fillmissing(E1,'constant',0),
3. Generate the grid with the following formulations. The initial position to start the indexing of elevation points is assumed to be (0,0). Considering 30m spacing and the number of points equal to "nz" (in the case of X direction), the last point will be located at a distance of (nz-1)*30 from the origin. The same holds true for the Z direction by replacing "nz" with "nx". Further, the transposes of both

arrays are taken and assigned to new variables.

```
X1 = [0:30:(nz-1)*30];
Z1 = [0:30:(nx-1)*30];
X1 = X1'; Z1 = Z1';
```

4. Using repmat() & repelem() functions, the matrix and elements of X1 and Z1 matrices are repeated according to the number of points in the X and Z directions, respectively.

```
X1 = repmat(X1, 371, 1);
Z1 = repelem(Z1, 464, 1);
```

5. Convert all the elements of the rows into columns to facilitate concatenation at last.

```
X1 = X1(:);
Z1 = Z1(:);
```

6. Generate a terrain data grid F by concatenating all three arrays X1, Z1, and E1.

```
F = [X1, Z1, E1];
```

Since in the terrain data grid, only the points at 30 m spacing have elevation values, the "scatteredinterpolant()" function is used to interpolate and return the nearest elevation values at the points, which do not have the elevation data. It is used to interpolate and return the values in case a of two-dimensional or a three-dimensional grid. Further, the grid is saved in the ".mat" file extension, which is further called out using the Interpreted MATLAB Function block during the compilation in Simulink. There are several interpolation methods, which include "linear interpolation", "natural interpolation" and "nearest interpolation". As the values corresponding to the nearest neighbour can be easily retrieved, "nearest interpolation" has been found to serve the purpose. The code for defining the interpolation function and saving it is as follows.

```
f = scatteredInterpolant(X1, Z1, E1, 'nearest');
save('grid.mat');
```

Here, f is a scatteredInterpolant function containing the arrays X1, Z1, and E1. The predicted coordinate values are passed through the function, and interpolated terrain elevation value(s) corresponding to the predicted coordinates are returned. When the compilation (simulation) is started, the Interpreted MATLAB Function block initializes the function "f", which takes the predicted X ("xp") and Z ("zp") coordinates of the aircraft as an input and returns the corresponding elevation value ("ep") as the output. The terrain database can be used as a template to develop more complex databases as the obstacle system demonstrated in Yamamoto[13], *et al.*, which operates on stereo pair satellite images capturing the topological information and man-made obstacles.

3.2.2 Comparison Between the Aircraft's Altitude and Terrain's Elevation

Once the terrain elevation value ("ep") is returned from the Interpreted MATLAB Function block, it is fetched into the comparator block and compared with the aircraft's predicted altitude. It makes use of a relational operator and gives the output in the binary form. If the aircraft's predicted altitude is equal to or lower than the terrain's altitude, it returns the output as 1 (high), and, in other cases, it will return 0 (low). The output from the comparator block is further fed into the Warning and Display subsystem, which is discussed in the following section. For this purpose, the output is stored in the in-port.

4. WARNING AND DISPLAY SYSTEM

The output value from the signal of the comparator block that is stored in the in-port block is fed (from the out-port) as an input to the switches, which forms the Warning and Display sub-system, as depicted in Fig. 5.

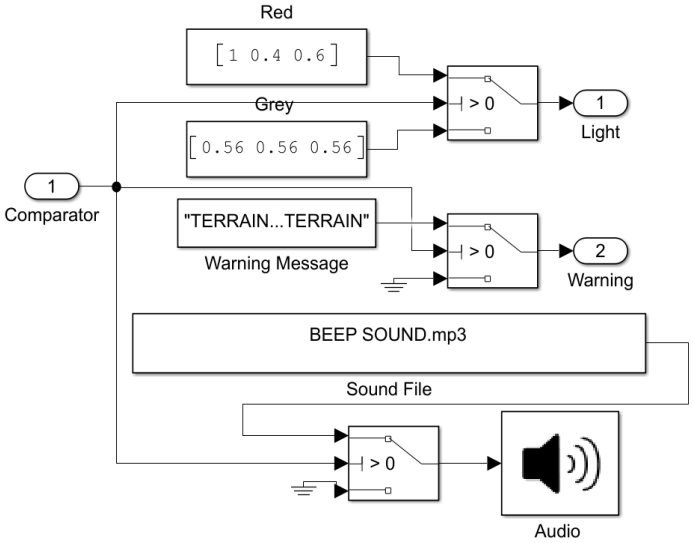

Figure 5. Block diagram of warning and display subsystem.

The output from the first switch is fed into the node, which contains the data for the fuselage geometry and appearance. If the input to the warning system turns high (value becomes 1), it activates the circuit that turns the color of the aircraft "red" as seen in the display window (Fig. 6). This indicates that the aircraft's altitude is unsafe relative to the terrain along its predicted path for the next 30 sec. If the input to the warning block is low (value becomes 0), the circuit with grey color (which is the original color of the fuselage) coding values is activated, keeping the color of the fuselage intact. The "constant" blocks contain the color code for the intended colors to be fetched into the switch when it gets activated.

The second switch is connected to the HUD text node of the VR Sink block. The HUD text node helps to display the text stored in the form of a string in the pre-defined region in the animation window. If the comparator block returns the value "1" (TRUE), the circuit with the string block containing the warning message "TERRAIN…TERRAIN" gets activated, and it further passes the signal to the VR Sink block to display the message in the VR animation window. If the output from the comparator block is "0", the circuit remains grounded. Hence, no message will be displayed as the aircraft is expected to fly safely above the terrain at the predicted position (in the upcoming 30 sec.).

The third switch contains the circuit for the audio warning. The Simulink block, named "From Multimedia File", is used to read multimedia files containing audio, video, and so on. As soon as the comparator returns the value "1", the

circuit containing the sound file is activated and transmits the signal to the Audio Device Writer, which produces the "Beep" sound through the system's built-in speakers or headphones signifying the potential discrepancy in aircraft's altitude in comparison to terrain. In all other cases, the comparator value will be "0", and there will not be any aural warnings since the circuit remains grounded.

The VR Sink block plays a very significant role when it comes to displaying the simulation of the system. It contains the data for the Virtual Reality (VR) World (terrain geometry and aircraft in this case). Moreover, it also contains the input ports connected to the nodes of various objects in VR World to perform multiple actions based on the data transmitted from the Simulink blocks in the form of signals. These systems and subsystems (of Simulink) are solely responsible for the control of the objects in the VR World using various static inputs given by the user or the inputs embedded beforehand. In this way, the aural and visual warnings are produced to represent the real-time terrain awareness scenario to the user. Figure 6 shows the animation representing the entire system that can be seen as the final output by the users. The same simulation can be depicted on the Multi-Functional Displays (MFDs) in the cockpit to make the pilots aware more quickly compared to the visual warnings with 2D maps.

Figure 6. Simulink 3D animation window simulating the system.

5. INTEGRATION OF THE SIMULINK MODEL INTO REAL-TIME FLIGHT SIMULATORS

Over the years, MathWorks products, such as MATLAB and Simulink, have been used extensively in the aerospace industry, allowing the testing of complex systems before the implementation of actual hardware. Hence, integrating the models directly into simulators or cockpits instead of reprogramming them can significantly reduce the programming errors and bring down the simulation implementation and validation time.

The process was developed for integrating the Simulink models into a real-time flight simulator environment by Lewis et al. (2012)[9]. This procedure makes use of Real-Time Workshop (RTW), a MathWorks utility that generates C code from the Simulink block diagrams with the external code resources being handled through MATLAB's S-function mechanism. This procedure has been proven to be very efficient, maintains the integrity of the function provided by the model, and gives flexibility to the researchers to continue the development work in the MATLAB environment.

6. CONCLUSION

The terrain awareness and warning system has been modeled, analyzed, and simulated using MATLAB and Simulink. The algorithm has been designed in such a way that it detects the terrain in the environment by extracting the elevation points in the grid along the aircraft's path and provides visual and aural warnings to the pilots to avoid accidents. The final simulation has been carried out in the Simulink 3D Animation window, which illustrates the working of this system in a real-time environment. This system can be used in a wide range of applications, such as UAVs, drones, and so on. The model can also be integrated with flight simulators for a variety of purposes. Moreover, the existing two-dimensional maps in the cockpit displays can be modified into 3-D maps along with improvised visual warnings, as discussed in the preceding sections. However, this paper mainly considers static input where flight parameters remain uniform with time. Also, it provides collision detection in a straight line along the flight path. Further improvisations have to be done to provide dynamic input options (such as control stick, yoke, and so on), collision detection along a region (for example, a trapezoidal boundary) in front of the aircraft instead of a straight line, and providing the capability for automated actions based on warnings (self-correcting the attitude) to avert the collision.

REFERENCES

1. Helicopter–Enhanced Ground Proximity Warning System Pilot's Guide. Rev. C ed., Honeywell.
2. Xiao, G., He, F. & Wu, J. Research on an EGPWS/TAWS simulator with forward-looking alerting function. In IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC). IEEE Xplore, 2014. doi:10.1109/DASC.2014.6979554.
3. Mullins, M., Foerster, K. & Kaabouch, N. Incorporating Terrain Avoidance into a Small UAS Sense and Avoid System. Infotech@Aerospace, 2012. doi:10.2514/6.2012-2596.
4. Gellerman, N., Kaabouch, N. & Semke, W. A Terrain Avoidance Algorithm Based on the Requirements of Terrain Awareness and Warning Systems. IEEE Aerospace, 2015. doi:10.1109/AERO.2015.7119101.
5. Raj, S., Rastogi, P. & Kushvah, B.S. Matrix-Based Method for Terrain Generation. Int. J. Comput. Sci. Inf. Technol., 2016, 7(3).
6. Lee, J. (2012). Modeling terrain awareness and warning systems for airspace and procedure design. 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), pp.2B2-9. doi: 10.1109/dasc.2012.6382283.
7. Nyenke, C. and Qu, S., 2003. Development of Advanced Terrain Awareness & Warning Display System.
8. Hui, H., 2024, September. Research on forward-looking warning algorithm and synthetic scene integrated simulation system of TAWS. In International Conference on Automation Control, Algorithm, and Intelligent Bionics (ACAIB 2024) (Vol. 13259, pp. 716-723). SPIE.
9. Lewis, E. and Vuong, N., 2012. Integration of MATLAB Simulink® Models with the Vertical Motion Simulator. In *AIAA Modeling and Simulation Technologies*

Conference (p. 4797).
10. Chen, R. and Zhao, L., 2022. A resilient forward-looking terrain avoidance warning method for helicopters. *Aerospace*, *9*(11), p.693.
11. Brutzman, D. & Daly, L. X3D: Extensible 3D Graphics for Web Authors. Elsevier, Amsterdam, Boston, 2007. doi:10.1016/B978-0-12-373676-3.X5000-4.
12. Rose, W. Controlled Flight Into Terrain (CFIT). [online] Available at: https://in.mathworks.com/matlabcentral/answers/1780440-controlled-flight-into-terrain-cfit#answer_1027805 [Accessed on 22 December 2022].
13. Yamamoto, H., Homma, K., Gomi, H., Kitagata, S., Kumasaka, K. and Oikawa, T., 2003, March. Geographical information system for flight safety. In Remote Sensing for Environmental Monitoring, GIS Applications, and Geology II, 4886, pp. 543-550. SPIE.
14. USGS - U.S. Geological Survey. EarthExplorer. [online] Available at: https://earthexplorer.usgs.gov/ [Accessed on 12 January 2023].

CONTRIBUTORS

Mr Akshat Jani is a postgraduate student at IIT-Kanpur in the Department of Aerospace Engineering, specializing in Aerodynamics. His area of research is Computational fluid dynamics and in flight guidance, navigation, and control domain. In the current study, he has conceptualized and developed MATLAB algorithms for terrain data generation and extraction and aircraft position prediction and comparisons with terrain elevation. Moreover, he has also contributed to designing the Simulink subsystems to make the algorithm work.

Mr M. Sarath Chandra is a postgraduate student in the Department of Aerospace Engineering at Embry-Riddle Aeronautical University, USA. His area of research include: Systems and controls.
In the current study, he has brainstormed many ideas to implement the feature of forward-looking terrain awareness. He has also conceptualized how the warning and display subsystem generates warnings in case of potential hazards. He has made a notable contribution to developing and testing various blocks in Simulink.

Dr S. Charulatha is an Associate Professor in the Department of Aeronautical Engineering at Hindustan Institute of Technology & Science, Chennai. Her area of specialization include: Avionics and satellite remote sensing.
In the current study, she has helped in implementing the entire collision avoidance algorithm through suitable mathematical formulations, concepts from avionics and aircraft systems, and so on. She has also helped to simplify the complex algorithm to load, generate, and extract the terrain elevation data.