International Journal of Engineering Research and Reviews ISSN 2348-697X (Online)
Vol. 3, Issue 4, pp: (50-66), Month: October - December 2015, Available at: www.researchpublish.com

Vision-Aided Landing Using Flight Management System for Fixed Wing Unmanned Aerial Vehicle

[1]Engin Esin, [2]Asst.Prof.Dr. Ali Türker KUTAY

[1, 2] Department of Aerospace Engineering, Middle East Technical University, Ankara, Turkey

Abstract: **This article focuses on vision based autonomous landing of a fixed wing unmanned aerial vehicle (UAV). Computer vision algorithms are used as a feedback sensor in control loops. Design of flight management system (FMS) algorithms and control structure of UAV is described. FMS provides to apply specific missions like keeping trajectory points between two coordinates with keeping specific altitude and speed conditions. To be able to manage this condition, waypoints include latitude, longitude, heading, altitude and speed specifications. Therefore, to be able to process these waypoints; roll, pitch, altitude, heading and speed controller are designed. To be able to generate, load waypoints and activate designed algorithms to UAV, a ground control station (GCS) interface is designed. On the vision algorithms side, position of aircraft is detected with respect to known runway by using image processing algorithms supported by OpenCv library. Position information obtained from image processing is used by FMS. System has been successfully tested in flight simulation environment under several different wind and turbulence condition with different initial orientation of the UAV.**

Keywords: **Autonomous Landing, Unmanned Aerial Vehicles, Vision-based Control, Vision-based Navigation, Automatic Flight Control Systems, Flight Management System.**

1. INTRODUCTION

The integration of robotic technologies into UAVs suggest new solutions to problems which must be overcome. Landing is one of the most critical phases of the operation for UAVs. Operate a UAV under non-optimal weather conditions is highly dependent on user ability to land safely. Moreover, even optimal weather conditions, with a small error in guidance or control could cause system loss of damages. Therefore, autonomous landing is a must for a UAVs and aim of this paper is to propose an alternative way to manage autonomous landing with image processing.

In this paper, implementation of flight management system which supply hold references to auto pilot is represented. Roll, pitch, altitude, heading and speed of the UAV must be hold to predefined references to be able to achieve a safe landing. Designed FMS stores waypoints which have information about latitude, longitude, course, altitude and speed information of target coordinate. Between two waypoints, related references are held with respect to interpolation of target and previous waypoints information. With the line of sight satisfied to the runway, references is supplied by image processing algorithm so that while landing, GPS latitude and longitude information is not a must for getting position information of the aircraft.

Despite the fact that the landing is often most critical phase in the whole mission, few publications about automatic landing systems for fixed wing UAVs are available. Heron 1 [1] and MQ-1B Predator [2] have automatic landing ability in military grade but no detail information about these systems are available. For rotary wing UAVs, many work has been done in automatic landing like [3], [4] and [5]. Some way of vision based automatic landing for fixed wing UAVs have been published like [6] which can get precise results by the method of template matching, however, this method has the limitation of adjusting template size continuously according to runway view change during approaching. Literature [7] proposes that its method can recognize the runway by placing red marker on corners, which is useful, but not practical. Literature [8] recognize runway by extracting SIFT (Scale-invariant feature transform) features to perform image registration against a stack of images in which location of the runway is known, but extracting the SIFT features requires high computational cost.

This paper suggest to use runway light bulbs to detect runway. Many runway have runway edge lights to be used to outline the edges of runways during periods of darkness or restricted visibility conditions. Because these bulbs are one of the brightest objects, it is very cost efficient to detect these bulbs in the image. After obtain runway edges in the image, pose estimation of the aircraft are done with the runway known information. When runway edges captured, flight management system (FMS) uses calculated aircraft position instead of using GPS.

2. SYSTEM OVERVIEW

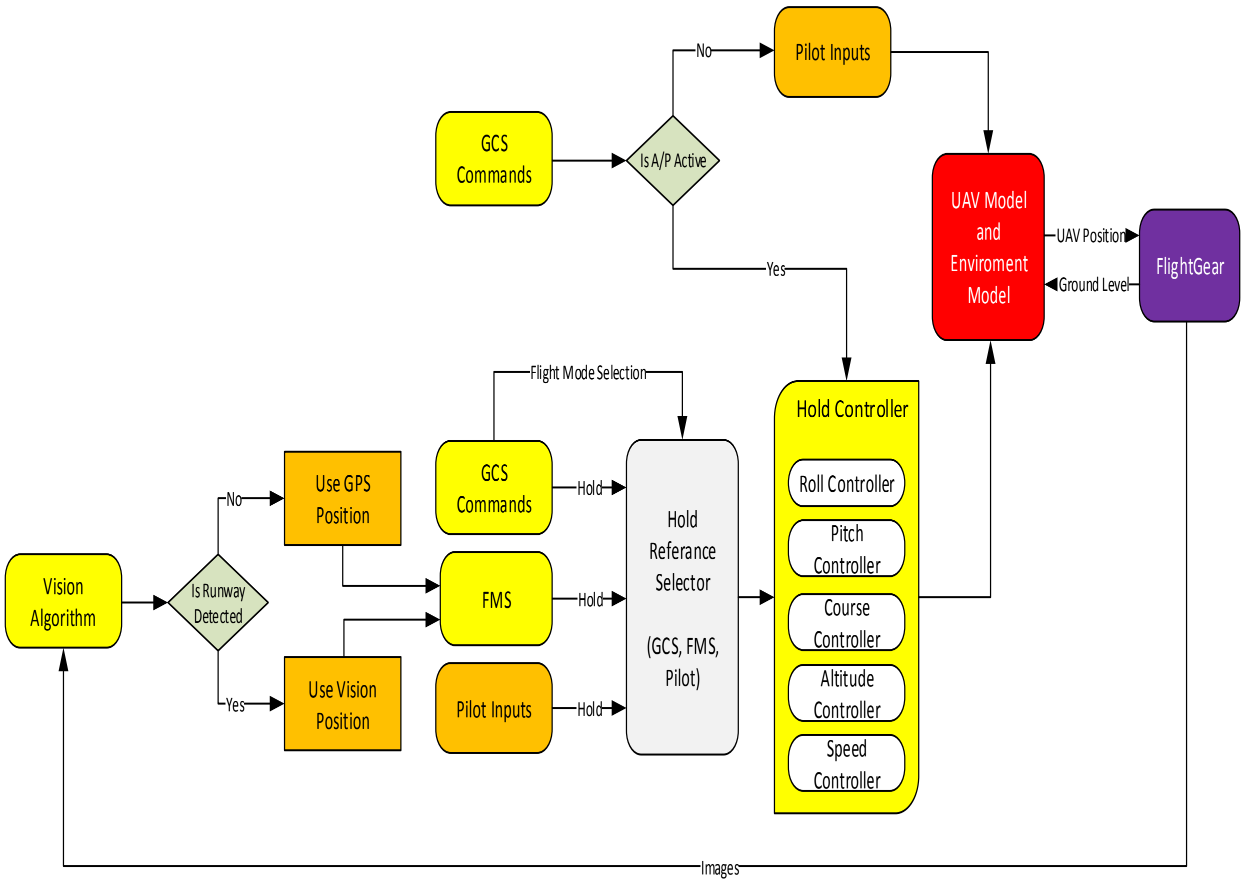
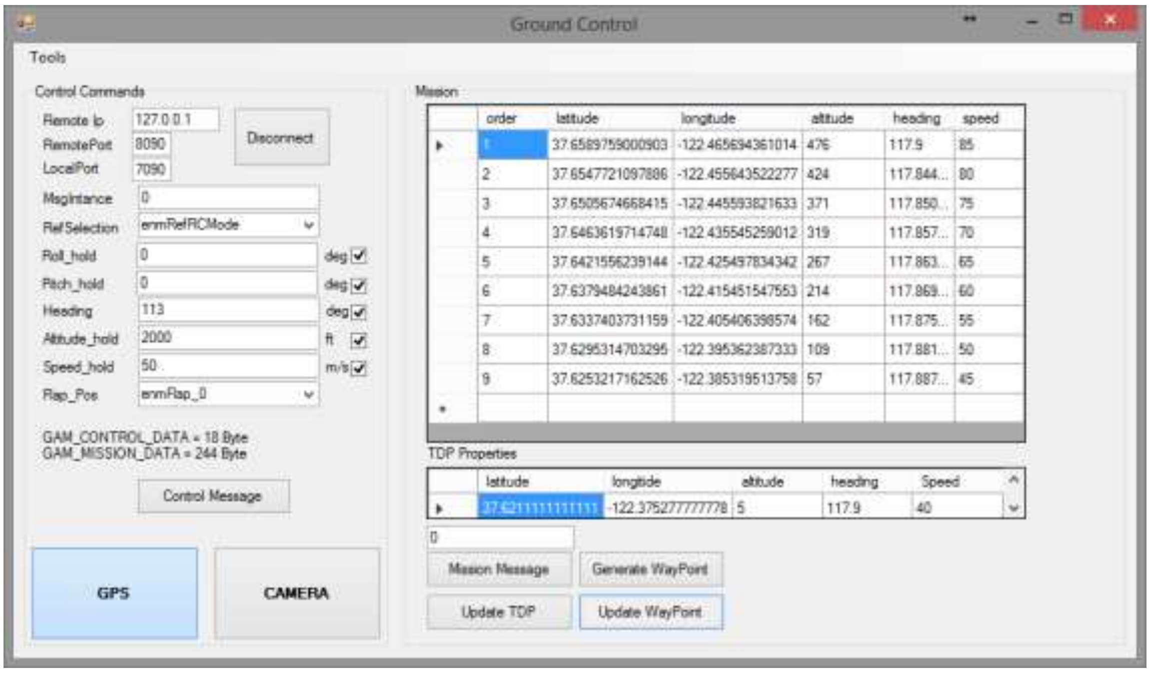


Figure 1 System Overview

The functionality and general algorithm of the system of the UAV is described in Figure 1. The system components can be divided into four parts. Control loops, flight management system, ground station interface, and runway detector. UAV and environmental model is obtained from Matlab/Simulink demo of “Fly the DeHavilland Beaver” [9]. The DeHavilland Beaver model includes the airframe dynamics, aerodynamics and wind profiles. Controller and FMS are constructed on that demo. Developed ground control interface and image processing algorithm application communicates with demo Simulink project via UDP protocol. Flightgear simulation tool is used as an image generator for image processing part. Also ground altitude level is obtained from Flightgear simulation tool. Flightgear [10] is a free and open source flight simulator.

Ground control station (GCS), which is a C# application, send commands to Simulink project via UDP. GCS command control flight mode of the UAV such as Pilot (free) mode, Hold mode in which UAV holds references comes from GCS command and FMS mode in which UAV holds list of waypoint data defined from GCS. Image processing algorithm captures Flightgear screen and process this image to find runway. If Vision application success to find runway, it sends corresponding information to Simulink project with healthy status of that information.

3. GROUND CONTROL STATION (GCS)



Remote Ip	127.0.0.1
RemotePort	8090
LocalPort	7090
MsgIntance	0
RefSelection	enmRefRCMode
Roll_hold	0 deg ✓
Pitch_hold	0 deg ✓
Heading	113 deg ✓
Altitude_hold	2000 ft ✓
Speed_hold	50 m/s ✓
Flap_Pos	enmFlap_0

GAM_CONTROL_DATA = 18 Byte
GAM_MISSION_DATA = 244 Byte

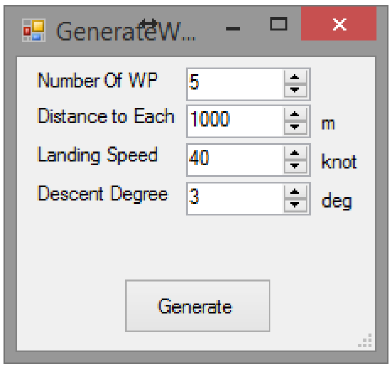
order	latitude	longitude	altitude	heading	speed
1	37.6589759000903	-122.465694361014	476	117.9	85
2	37.6547721097886	-122.455643522277	424	117.844...	80
3	37.6505674668415	-122.445593821633	371	117.850...	75
4	37.6463619714748	-122.435545259012	319	117.857...	70
5	37.6421556239144	-122.425497834342	267	117.863...	65
6	37.6379484243861	-122.415451547553	214	117.869...	60
7	37.6337403731159	-122.405406398574	162	117.875...	55
8	37.6295314703295	-122.395362387333	109	117.881...	50
9	37.6253217162526	-122.385319513758	57	117.887...	45

latitude	longitide	altitude	heading	Speed
37.6211111111111	-122.375277777778	5	117.9	40

0

Figure 2: Ground Control Station Interface

GCS interface developed with C# language. GCS communicates with Simulink model via UDP protocol. By using GCS two message can be sent to UAV named as GAM_CONTROL_DATA and GAM_MISSION_DATA. GAM_CONTROL_DATA sends flight mode (RC, Hold, and FMS), hold values for hold flight mode (roll, pitch, heading, altitude, airspeed, flap position). Also for FMS mode, Position source (GPS or Camera) is set by this message. GAM_MISSION_DATA can send up to ten waypoint with information of latitude, longitude, altitude, heading and speed of each. Generated waypoints and TDP could be saved by Update TDP and Update Waypoint buttons with XML format to be able to use again. Waypoints could be modified manually or could be generated by Generate Waypoint button and new GUI for waypoint generation created as can be seen from Figure 3.



Number Of WP	5	
Distance to Each	1000	m
Landing Speed	40	knot
Descent Degree	3	deg

Figure 3: Waypoint Generator

4. RUNWAY DETECTION ALGORITHM

Vision based positioning information generated by C++ application which is using OpenCV library that is free and open source computer vision library. Image processing application searches the Flightgear window by using Windows APIs. If Flightgear window is found, image of that window captured to processing as illustrated at Figure 4 (a). Runway is selected to be San Francisco international airport (KSFO) from Flightgear. On captured image, thresholding is applied to detect bright objects which are candidates of runway light bulbs at Figure 4 (b). Main runway has white bulbs and bulbs without white color eliminated at Figure 4 (c). Bulbs are positioned near the runway edges so by detection of bulbs which are in a line order gives possible runway edges. By using RANSAC [11] algorithm, points which can create a line are detected and grouped at Figure 4 (d). Runways have two edges so combinations out of two groups give runway candidates but if two lines crosses each other they are eliminated from this combination because runway lines must be parallel to each other. Combination results can be seen at Figure 4 (e). For elimination of runway candidates to find out the real runway needs a cost function. It is assumed that nose view of UAV is looking to runway which is aimed to land. So true runway would be close to center of view. Also length of two edges of the runway must be close to each other. On the other hand, as can be seen from Figure 4, there is two true runway in the image. It is assumed that aim of the UAV is to land on left runway so all runway candidates tagged with an order number starting from left to right. In the light of foregoing,

$$\begin{aligned} Cost\ Function = {} & abs\big(\,Lenght(LeftEdge)\ -\ Lenght(RightEdge)\big) * (-10) \\ & + distance(CenterofRunwayCandidate, CenterofwholeImages\ )\ * (-10) \\ & + Order * (-100) \end{aligned}$$

Finally, ordering this cost functions, runway which has biggest cost function is set to true runway as can be seen from Figure 4 (f). When target runway detected, runway width in term of pixels and angle of two edges are calculated. Flightgear image has filed of view (FOV) 55° and focal length of view can be calculated as fallows,

$$Focal\ Lenght = \frac{\left(\frac{\text{Image.cols}}{2}\right)}{tan\left(\frac{\text{FOV}}{2} * \frac{\text{PI}}{180}\right)}$$

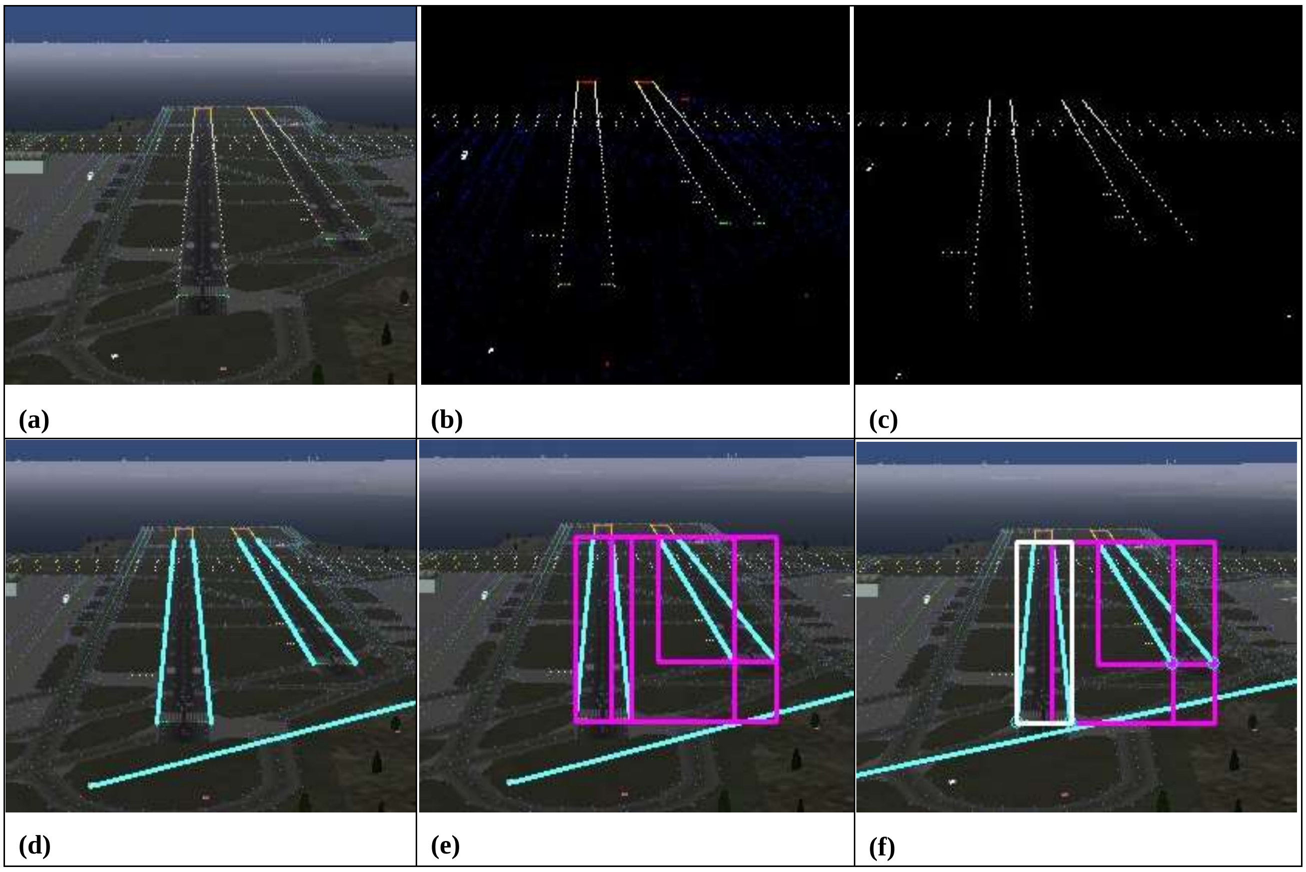


Figure 4: (a) Captured Image from Flightgear Screen. (b) Threshold Image. (c) Selection of White Bulbs. (d) Line Detection Algorithm Result. (e) Output of Candidates Runways Algorithm. (f) Elimination of Candidate Runways.

5. HOLD AUTOPILOT CONTROLLER

[1] Roll Hold:

For a fixed-wing UAV, while flying at a constant altitude, a roll angle command is often used for heading control. So to be able to control heading rate, it is necessary to control roll angle. However, while landing phase, roll angle must be close to zero degree otherwise wings can be touch to ground which probably will cause damages or loss of the vehicle.

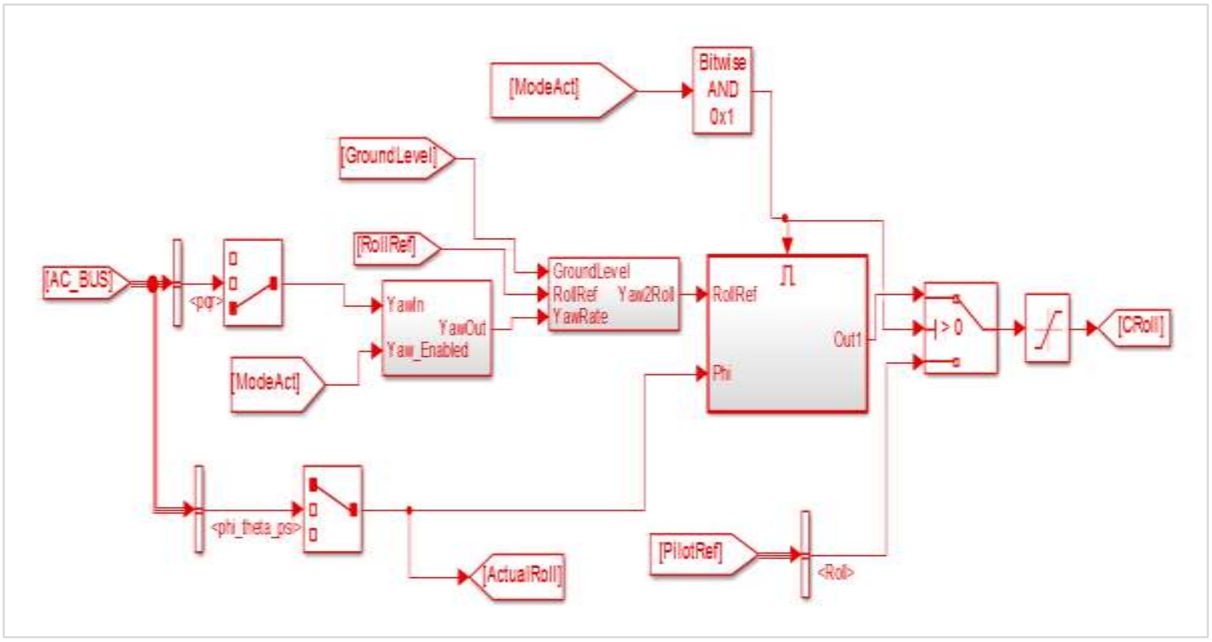


Figure 5: Roll Hold Controller

Ground control station can disable or enable control modes individually. When course control disabled, effect of course control to roll angle also disabled at the roll control algorithm. Also whatever the yaw effect is, if altitude is lower than predefined value than roll controller try to keep zero bank angle. PID controller is used to keep roll angle to roll reference value by controlling aileron actuator angle. Aileron actuator angle is limited between [-20 20] degree and back-calculation anti-windup method is applied to discharge the integrator on block saturation.

Roll control can be disabled by GCS command. If roll control or autopilot is disabled for controlling UAV by pilot command, aileron actuator angle is set directly by pilot inputs.

[2] Course Hold:

Course control is very important while landing phase because UAV must be in the range of runway width. Proposed course hold algorithm suggest a coupled control with bank angle and yaw angle while altitude is higher than predefined value. At later stage, when altitude is lower than predefined level, heading control is done with controlling yaw actuator with zero bank angle so that back wheels are touch the ground together to prevent any jump or crash of vehicle.

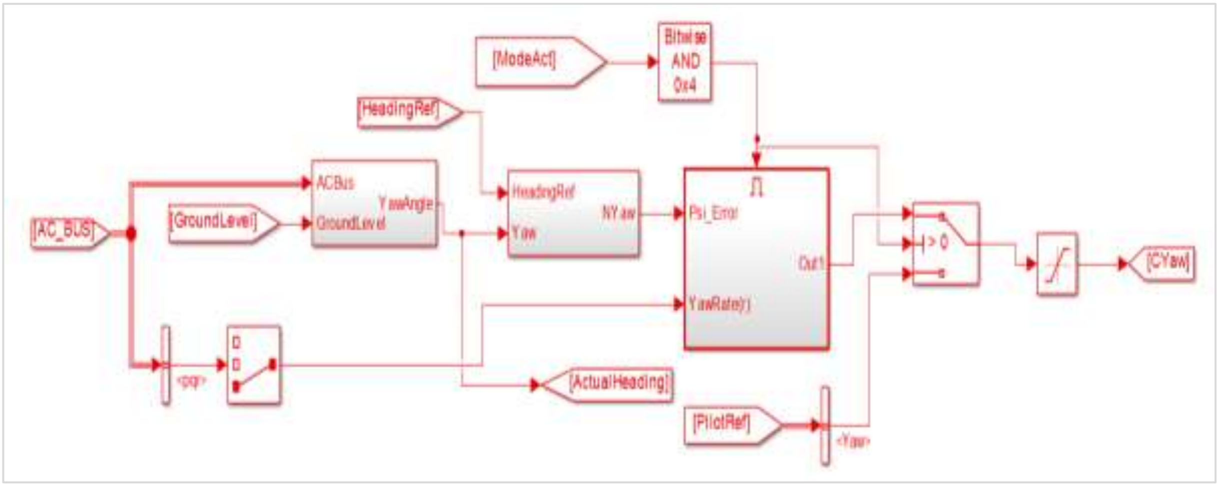


Figure 6: Course Controller

Weather conditions have very important effect on landing phase. Especially side winds must be taken into account while try to keep cross track error minimum with respect to runway position. Similar to roll effect on course control, up to a predefined altitude, side wind is taken into account.

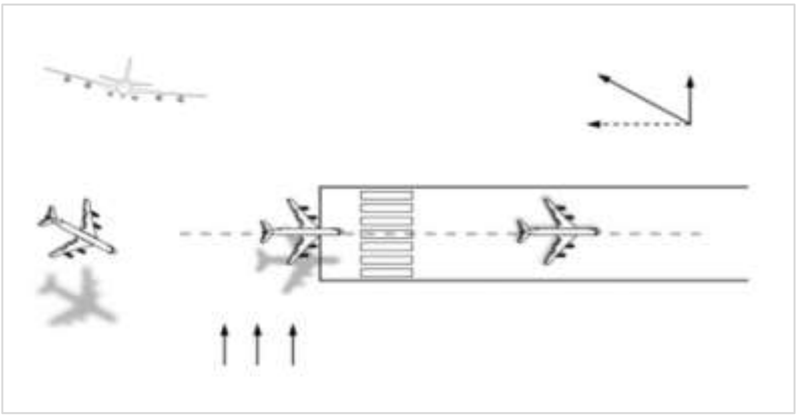


Figure 7: Sideslip

As can be seen from Figure 7, when side wind is existing, course direction differs from actual heading of the plane which nose of the plane points. To be able to hold cross track, earth speed direction must be controlled up to predefined altitude. As aim of this paper manage a successful landing without using GPS position, from air speed of the UAV and wind speed of the wind, sideslip angle is tried to keep zero. After reaching predefined altitude level, UAV body speed direction is controlled using IMU heading information.

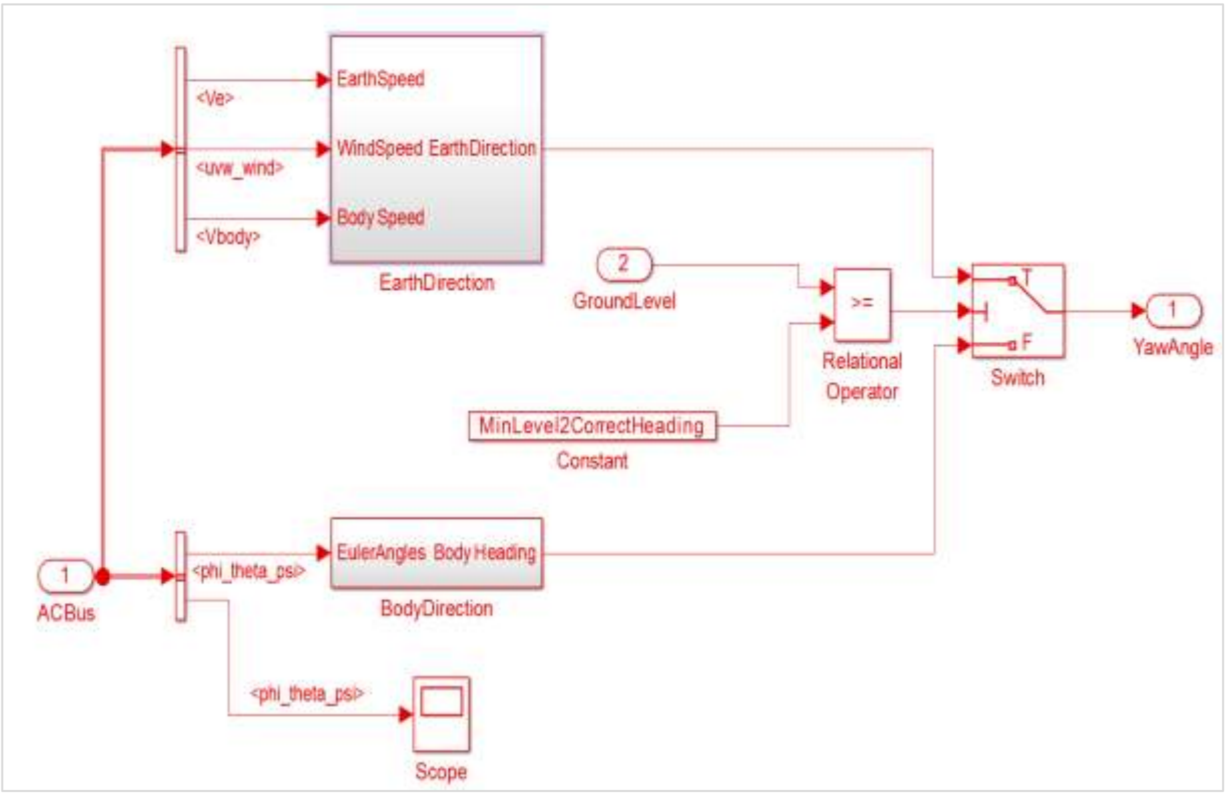


Figure 8: Sideslip Angle Calculation and Touch Down State Algorithm (Touch2Ground Block)

PID controller is used for course hold to a cross track reference value by controlling yaw actuator angle. Yaw actuator angle is limited between [-15 15] degree and back-calculation anti-windup method is applied to discharge the integrator on block saturation. Course controller can be disabled by GCS command. If course controller or autopilot is disabled for controlling UAV by pilot command, yaw actuator angle is set directly by pilot inputs.

[3] Pitch & Altitude Hold:

The purpose of pitch hold autopilot is to make the pitch angle constant at certain value. It is rarely used alone but used to control speed or altitude rather. Altitude hold is built on the pitch hold controller. The main objective of the altitude hold autopilot is to keep the aircraft flying at an altitude set point.

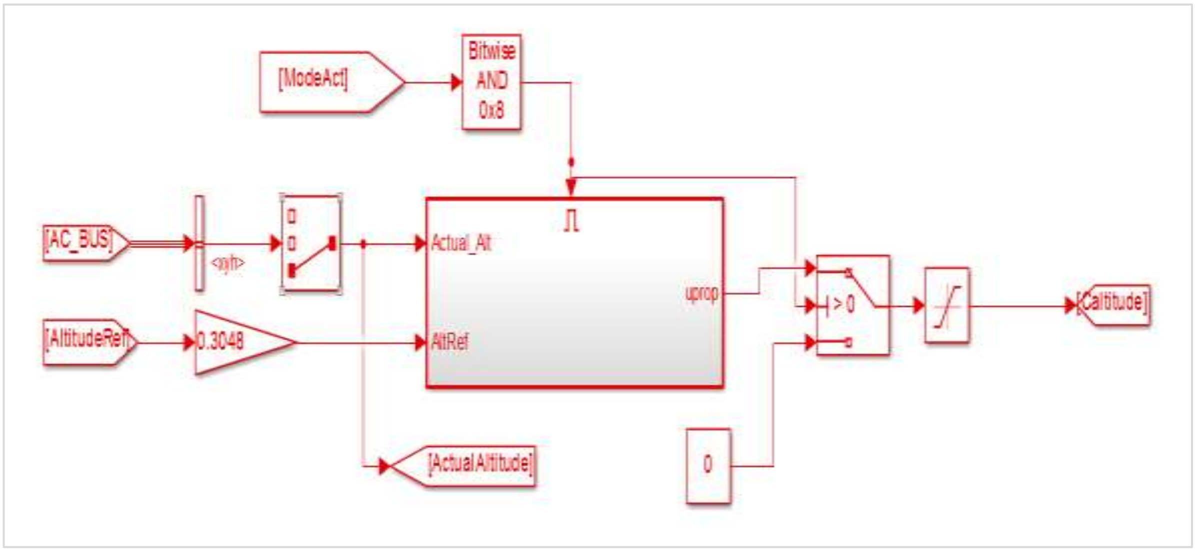


Figure 9: Altitude Hold Controller

Altitude controller needs some extra works due to its nature. Big changes on altitude reference cause big integration accumulations because for big changes on altitude reference, it would take time to reach destination altitude due maximum climb rate and maximum descend rate of the UAV which is limited to [8 -10] degree. To solve this problem, altitude error saturated between [-100 100] meters as can be seen from Figure 10. So that speed of accumulation of the integration part get slower and this limits is sufficient to reach limit rates reasonably.

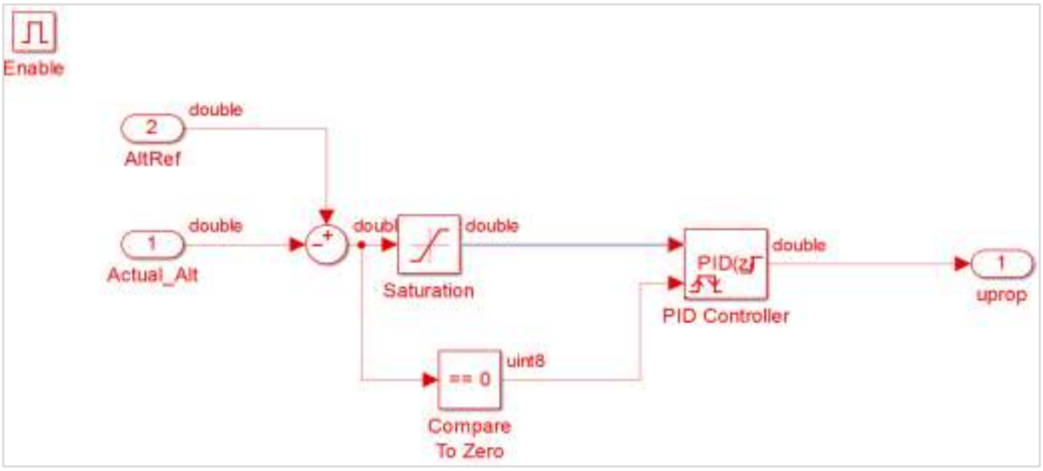


Figure 10: PID Block of Altitude Controller

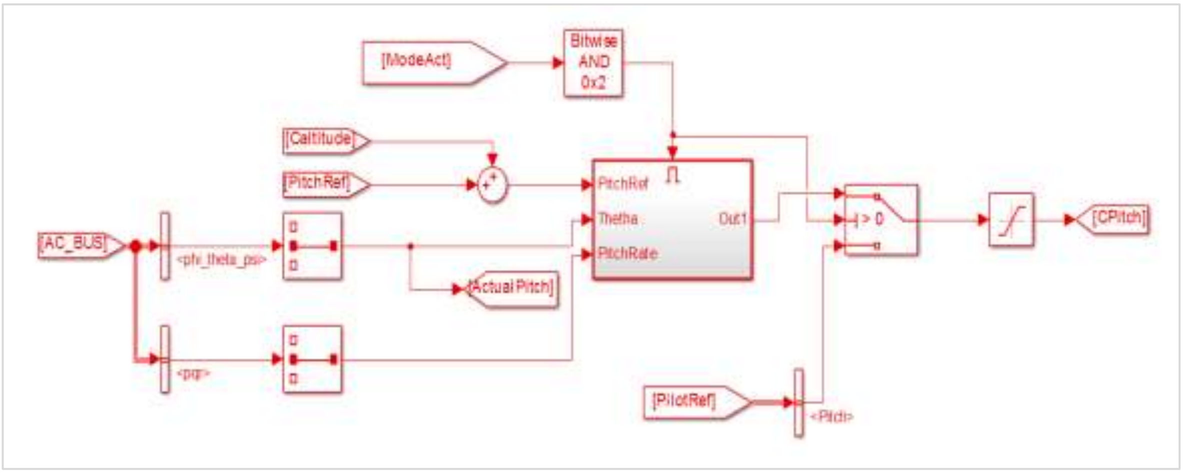


Figure 11: Pitch Hold Controller

GCS can set pitch and altitude references or disable these controllers individually. As mentioned above, altitude controller is built on pitch controller. Therefore, if pitch controller is disabled, altitude controller would be useless. If altitude controller disabled only, pitch angle keeping at pitch reference value comes from GCS. If pitch controller is disabled, pilot inputs applied to rudders whatever if altitude controller is disabled or vice versa.

PID controller is used to keep pitch angle and altitude to related reference values by controlling rudder actuator angle. Rudder actuator angle is limited between [-15 15] degree and back-calculation anti-windup method is applied to discharge the integrator on block saturation.

[4] Speed Hold:

The main objective of the airspeed hold autopilot is to keep the aircraft flying at a constant airspeed defined by GCS airspeed reference. The airspeed hold can be built in two ways. It can be either build on the pitch hold autopilot or by changing throttle value. In this design, altitude controller is built on pitch controller and while landing phase, altitude must be controlled precisely to prevent any damages caused by absurd altitude level changes, second way is preferred. The final airspeed autopilot block diagram is shown in Figure 12 and Figure 13.

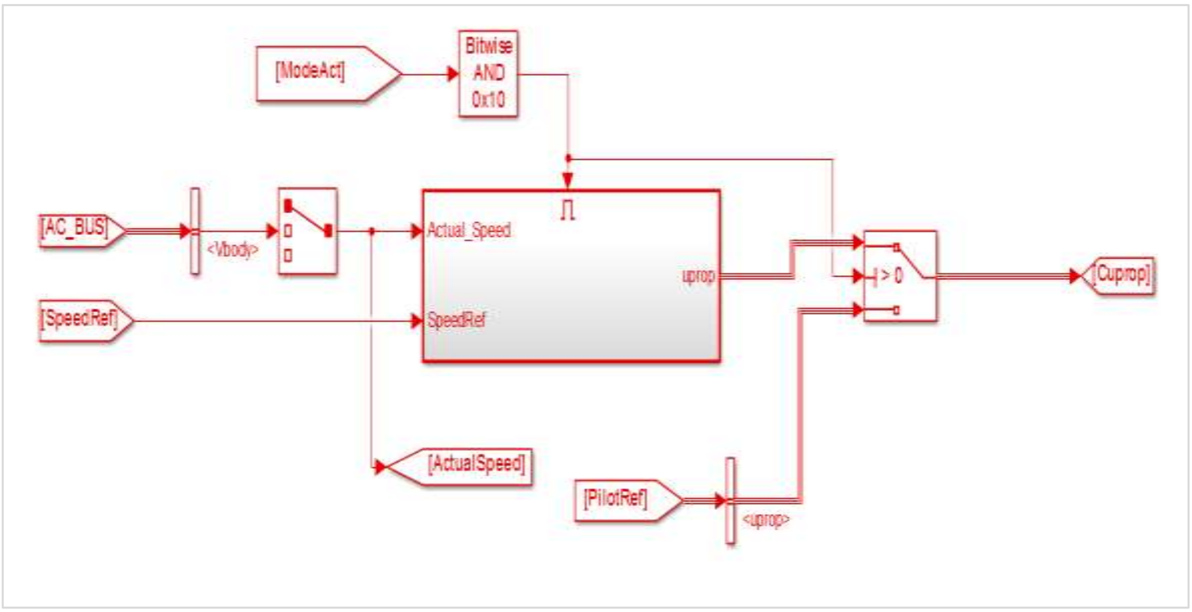


Figure 12: Air Speed Controller

Speed controller has crucial role while landing. Air speed is needed to choose a value that is above the stall speed of the UAV, but low enough that the aircraft is able to lose altitude. If speed of the UAV is high while landing, jumps could be occurred while touch down phase or. On the other hands, if speed of the UAV is below the stall speed, altitude of the UAV cannot be hold as planned by FMS and hard landing would occur.

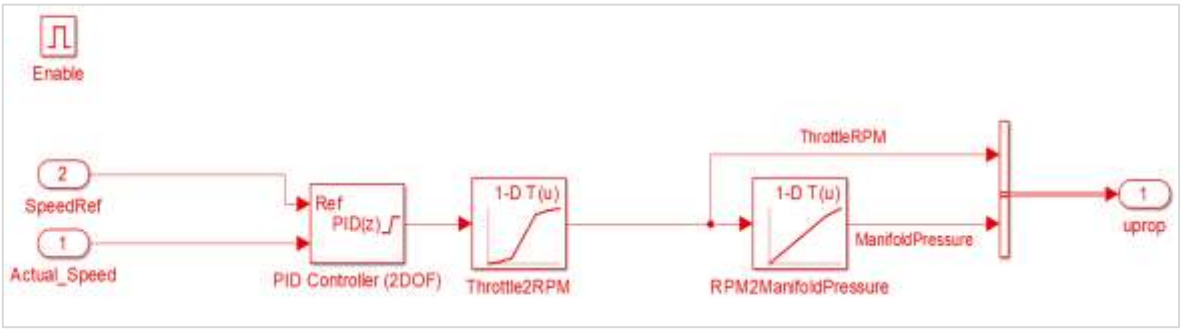


Figure 13: Air Speed Controller and Motor Model

PID controller is used to keep air speed to reference speed value by controlling throttle level. Throttle range is [0 1] and this value converted into motor RPM which is used for calculation of trust vector. Air speed controller can be disabled by GCS command. If air speed controller or autopilot is disabled, throttle is set directly by pilot inputs.

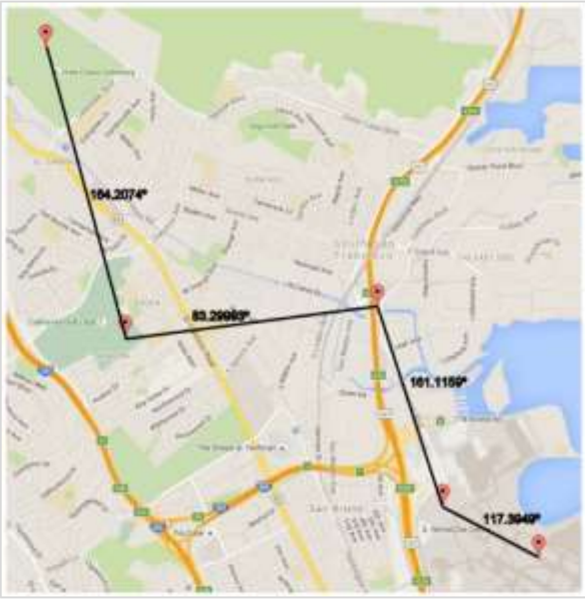
6. FLIGHT MANAGEMENT SYSTEM

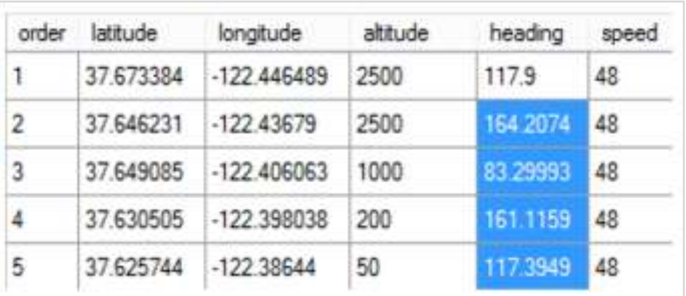
Flight management system (FMS) is a fundamental component of avionics which automates many in-flight tasks, reducing workload on the crew. From this point of view, FMS is a must for a UAV to be able to apply missions automatically. FMS needs many sensors such as GPS, air data sensors and needs a database to store flight plan which is generally determined on the ground. In mentioned design, flight plan can be generated or manually entered up to 10 waypoint including touch down point.

In this design, FMS has six parameters as fallows,

- Order: Order information of the waypoint. Execution order of the waypoints are managed with this information.
- Latitude: Target latitude information of the corresponding waypoint.
- Longitude: Target longitude information of the corresponding waypoint.
- Altitude: Target altitude information of the corresponding waypoint.
- Speed: Target speed information of the corresponding waypoint.

GCS load up to ten waypoints to UAV via UDP protocol and when FMS mode is set active, FMS algorithm start to execute this waypoints with respect to this order. FMS is interpolating target and previous set points to calculate current target value of altitude and speed. For example if previous waypoint speed value is 60 knot, target is 50 knot and UAV is on the midway of these two waypoints, current speed value that must be hold is set to 55 knot. Similarly, this calculation is done for altitude level. Heading values are calculated from GCS interface with respect to consecutive waypoints as can be seen from Figure 14.





order	latitude	longitude	altitude	heading	speed
1	37.673384	-122.446489	2500	117.9	48
2	37.646231	-122.43679	2500	164.2074	48
3	37.649085	-122.406063	1000	83.29993	48
4	37.630505	-122.398038	200	161.1159	48
5	37.625744	-122.38644	50	117.3949	48

Figure 14: Waypoints and Calculated Headings in Degree

Once in flight, UAV's position must be supplied to FMS so that FMS supplies up to date references to controller to be able to execute waypoint missions. To solve this problem, GPS and vision based positioning is supplied to FMS. If runway can be detected by image processing algorithm, information of runway with validity flag is send to FMS algorithm. With this condition if GCS sends command to use vision positioning system, FMS starts to use vision positioning information instead of GPS information. If validity of vision positioning information is not valid, FMS would ignore the command which comes from GCS for positioning information source is set.

[5] GPS Based Positioning:

Once GPS is selected for positioning system from GCS, FMS would use GPS information to determine UAV position. FMS calculates direction from UAV current position to next waypoint position and try to change this heading value to minimize error (Bearing Error) between calculated heading value and heading value between previous to target waypoint as illustrated from Figure 15. For the initial run of FMS, due to no previous waypoint exist at this time, UAV is directed towards first waypoint. However if only two waypoint is set which consist of one target point and one TDP, first waypoint is assumed to be at the same direction with runway so runway direction is accepted as cross track and bearing error calculated with respect to this assumption. Purpose of that algorithm is to set position of UAV to reasonable to manage landing.

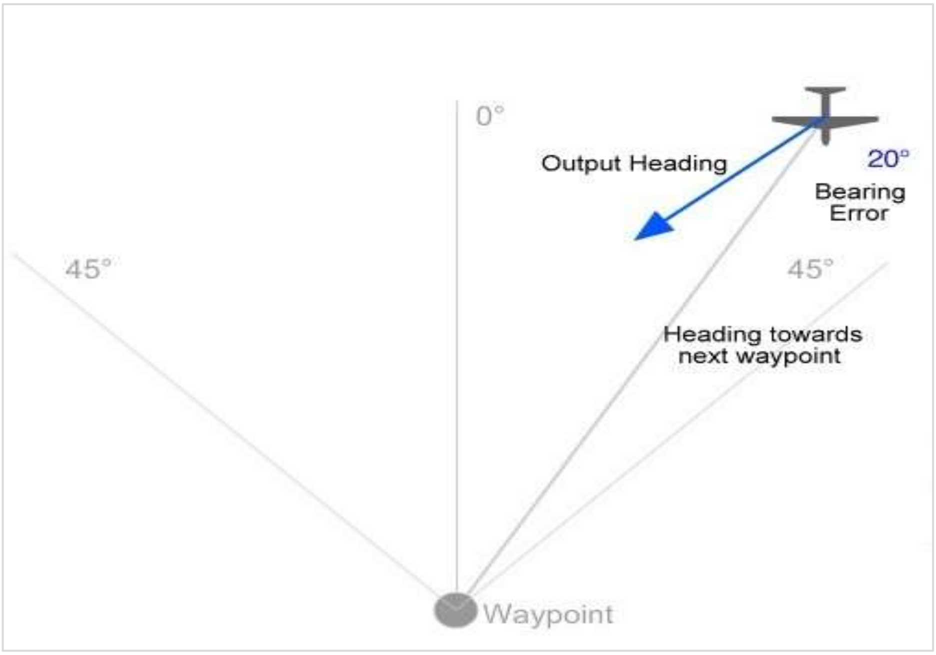


Figure 15: Bearing Error and Line Fit Algorithm

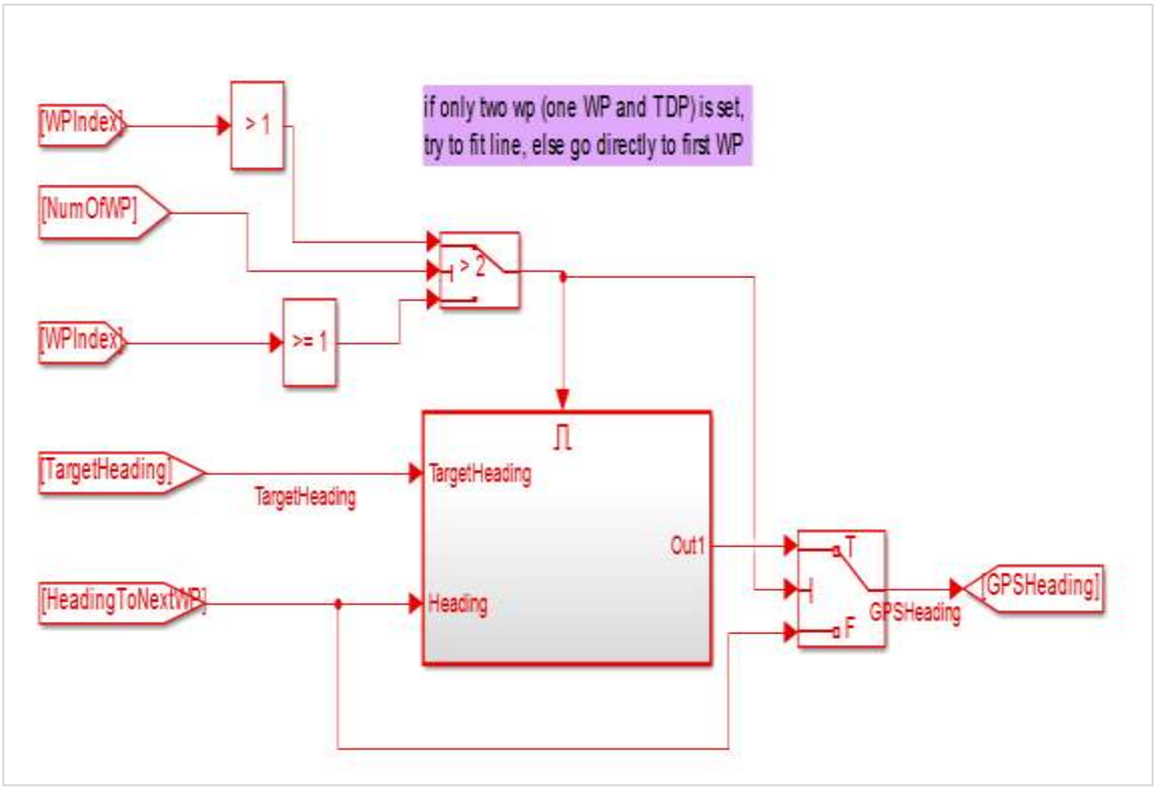


Figure 16: Cross Track Controller

Calculated heading is feed to heading controller reference which is mention at previous section. PI controller is used for minimizing bearing error. Controller output is deduced from heading to next waypoint with a controller limitation between [-15 15] deg. Details can be seen from Figure 17.

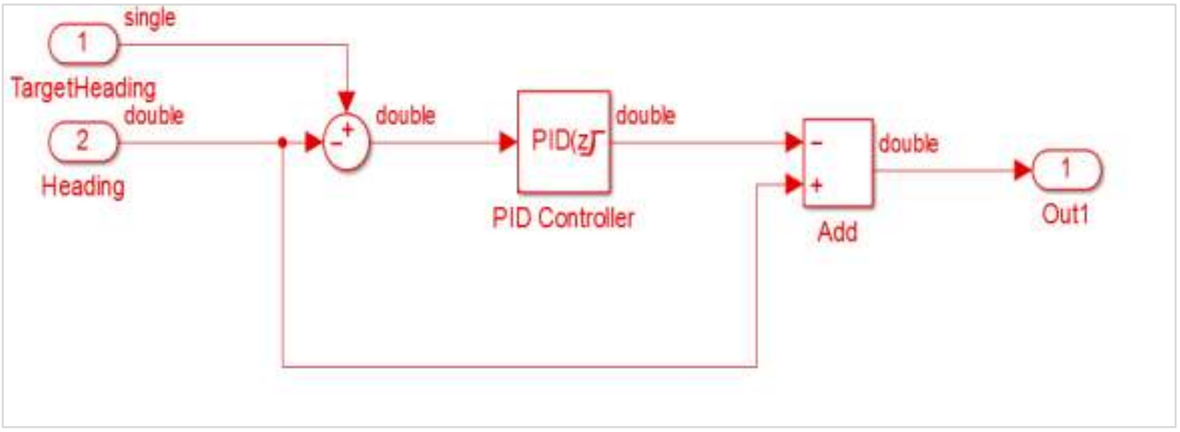


Figure 17: Cross Track Controller Algorithm

Execution of waypoints consecutively is decided with calculation of distance to next waypoint. Narrowing square algorithm is used for making decision of jumping next waypoint. If distance of UAV to target waypoint is in the range of generated sphere, it is accepted that next waypoint can be target waypoint. This sphere radius is get closer as waypoint execution continues. For the first waypoint, it is enough to be in 200 meter in range to change target to next waypoint. For the last waypoint this radius is reduced to 50 meter. Sphere radius for intermediate waypoint is calculated with interpolation of total waypoint number and execution order of the corresponding waypoint. Related algorithm can be investigated from Figure 17 whose output is “WPIndex”. After execution of all waypoints, “Validity” output is set to false which means FMS has done its jobs. After “Validity” is set to false, it is assumed that UAV touches the ground so altitude controller is disabled.

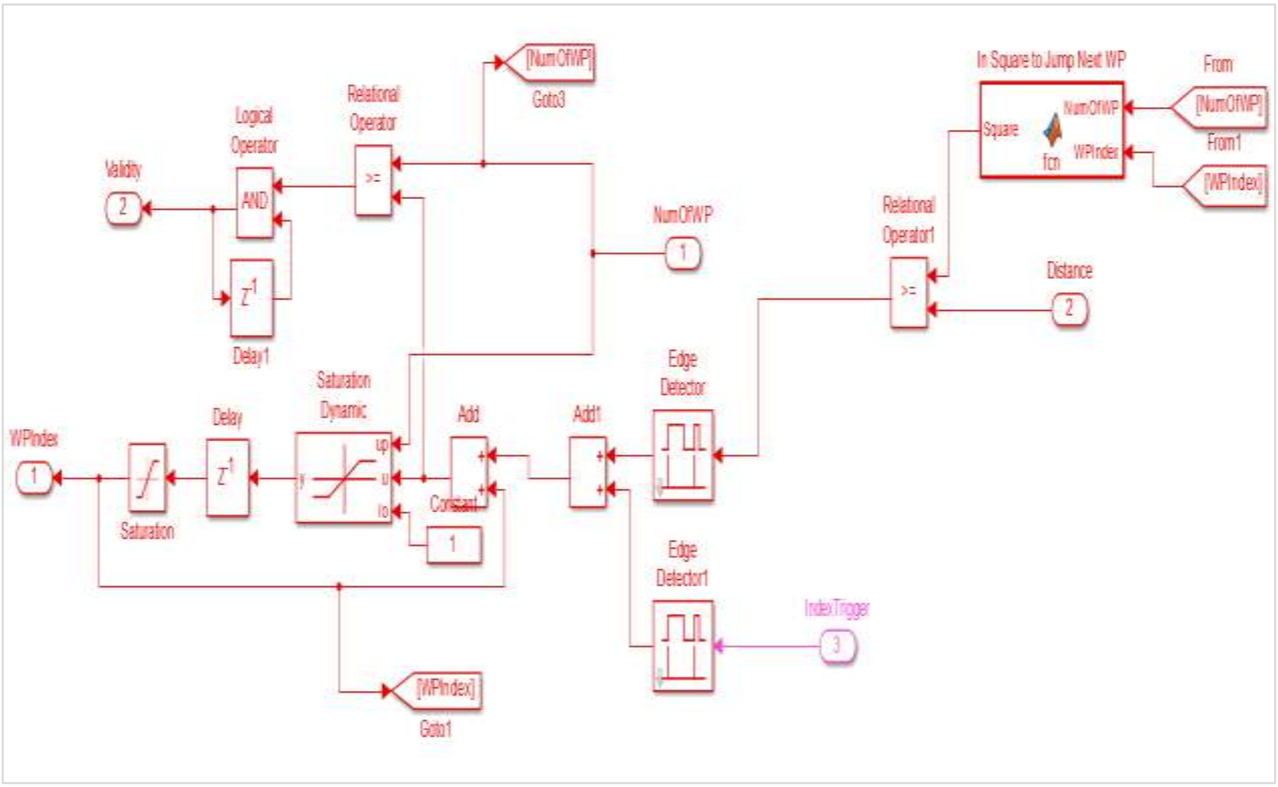
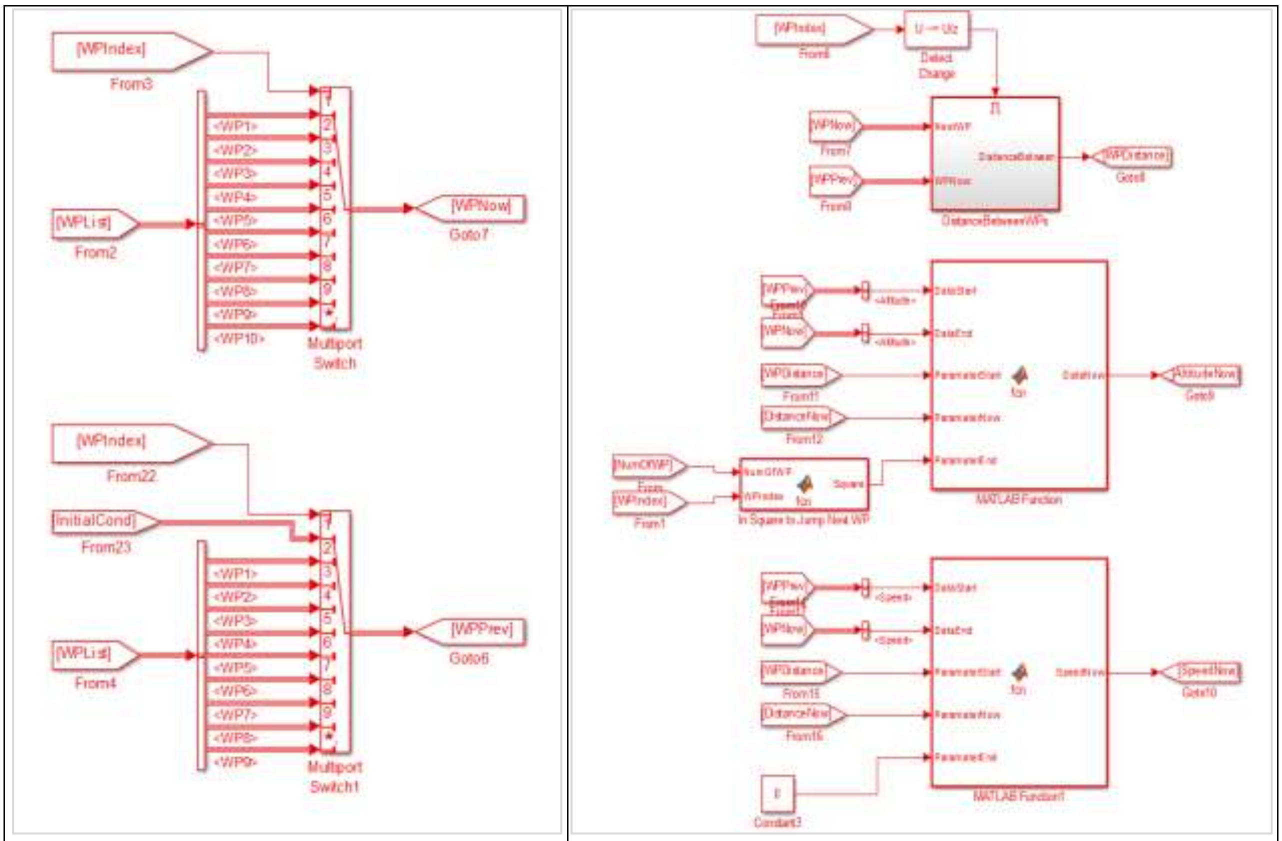


Figure 18: Decision of Set Target to Next Waypoint

When UAV position between two waypoint, altitude and speed interpolate with respect to initial and target values and how much of the distance is covered. Purpose of this algorithm is to manage smooth transition between waypoints. Otherwise, jumps occurs at FMS outputs which is also controller inputs. Big changes of controller inputs cause uncomfortable flight conditions which is not acceptable especially for a UAV which is going to land. This algorithms can be seen from Figure 18.



(a) (b)

Figure 19: (a) Target and Previous Waypoint Selectors, (b) Interpolation Algorithm to Smooth Transition between Waypoints

[6] Vision Based Positioning:

Once Camera is selected for positioning system from GCS, FMS would use vision information to determine UAV position. Position of UAV calculated by vision information is differential position with respect to touch down point (TDP). Runway width in pixel and angle of edges in degree are obtained from image processing algorithm. In reality, runway width is 61 m and from corresponding data comes from image processing, distance can be calculated by following formula,

$$Distance = \frac{(FocalLenght * RunwayActualWidth)}{RunwayPixelWidth} + DistanceTDPtoRunwayStartingPoint$$

Minimum distance that can be measured by vision information is limited because while approaching to the runway, runway gets bigger and at a point image width and runway width get equal. After that point runway width extends beyond the image width. Minimum distance that could be calculated by vision data is,

$$MinimumDistance = \frac{(FocalLenght * RunwayActualWidth)}{VisionSensorHorizontalLimit\_pix} + DistanceTDPtoRunwayStartingPoint$$

Due to distance calculation has a lower bound, altitude calculation from vision data also has a lower bound. Minimum altitude can be calculated by assuming that glideslope angle would be 3º. So with some trigonometry,

$$MinimumAltitude = MinimumDistance * \tan\left(\frac{GlideSlope * \pi}{180}\right)$$

When GCS selects camera for FMS position source, distance of UAV from vision calculation and altitude of UAV from barometric sensor is taken and set for initial condition of UAV. With the reduction of distance to TDP, target altitude value is calculated by making interpolation between UAV initial altitude level and minimum altitude value which is calculated above. After minimum altitude level is reached, altitude started to be decreased with glideslope angle by using airspeed of the UAV. Related algorithms can be seen from Figure 20.

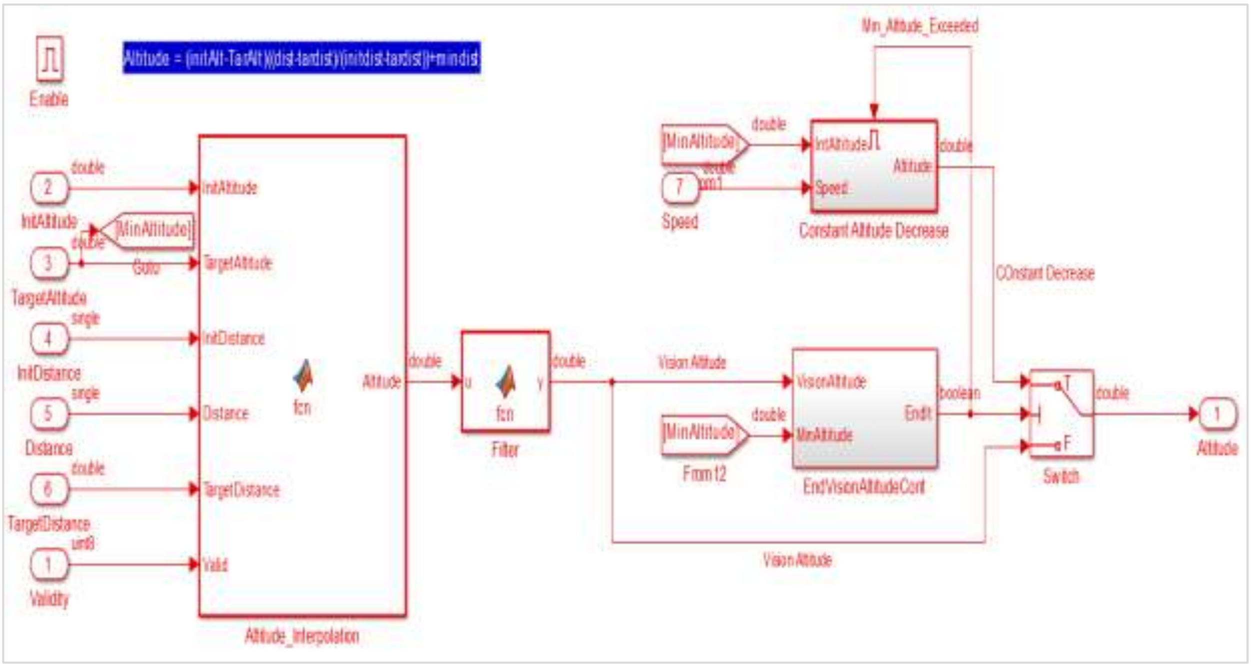


Figure 20: Vision Altitude Calculator Algorithm

Heading of UAV is calculated from angle of two edges of the runway like [12]. When UAV in front of the runway, angle difference of left and right angles equals to zero. However if position of UAV is at the left or right side of the runway, this difference would be nonzero. So by a PI controller, heading reference position is generated to feed heading controller which is mentioned above. Due to resolution issues of generated images and due to image processing is done using pixels, data comes from image processing has a discrete behaviour. Because of that fact, calculated altitude value has been filtered with a moving average filter with length 400. This value is determined by the trial and error method.

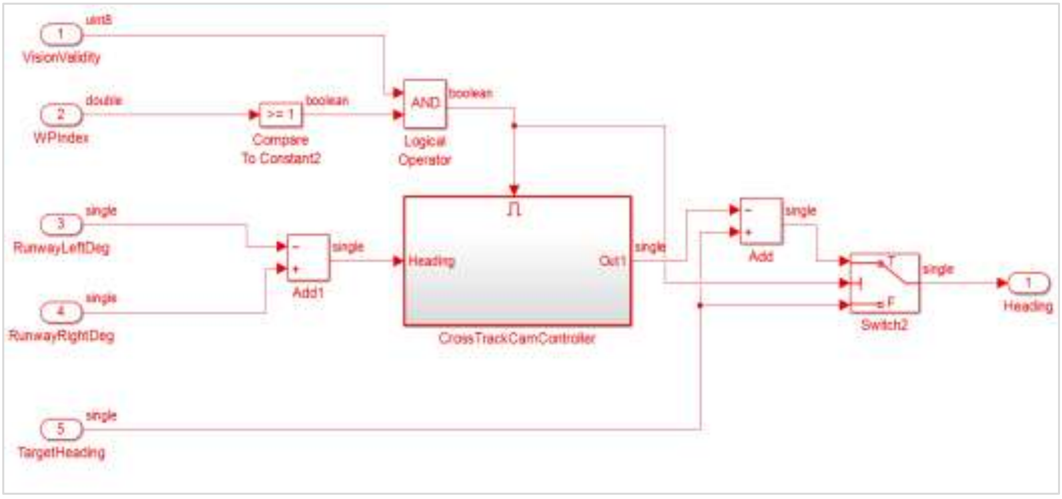
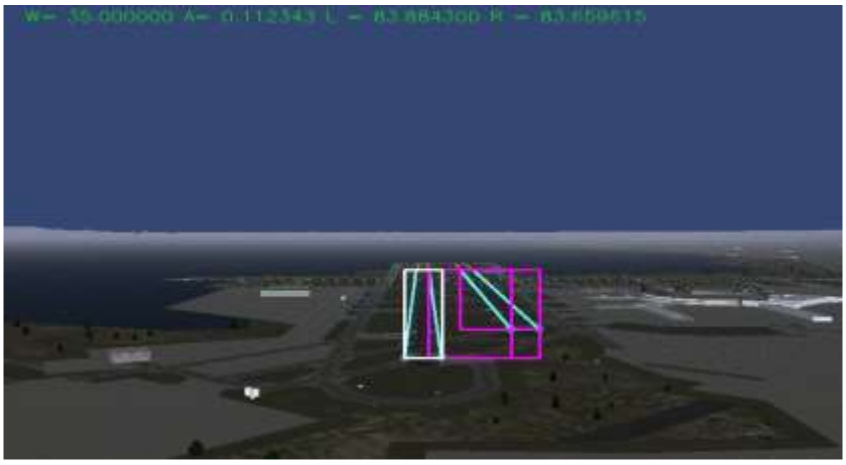


Figure 21: Vision Heading Reference Controller

7. SIMULATION AND RESULTS

The vision based landing algorithm proposed above have been developed and tested in a simulation environment where controller and FMS algorithms are implemented in Matlab/Simulink. Aircraft and atmospheric model are used from demo of “Fly the DeHavilland Beaver” in Matlab/Simulink. Image processing algorithms developed in C/C++, GCS interface developed in C#, the simulated image is generated which is set to 800x640 pixel by the Flightgear flight simulator and JMapView open source software is used to see and track UAV position on the map. Stall speed of the aircraft model is 40 knots, 110 knots of maximum speed. The chosen airport scenario was the San Francisco international airport (KSFO). Simulation has started with a random position of UAV. Mission data created by using GCS with 3º glideslope for airspeed of 45 knots.

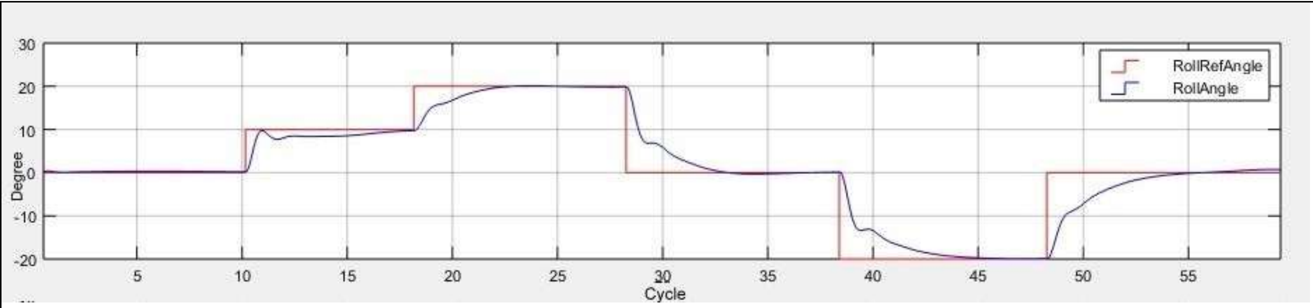
Image processing algorithm always works to find runway and send related information with success situation of the result to Matlab/Simulink model. An instance of measurement as can be seen Figure 22 at which runway width is 35 pixels, left side angle of the runway is 83.88 degree and right side of the runway is 83.65 degree.



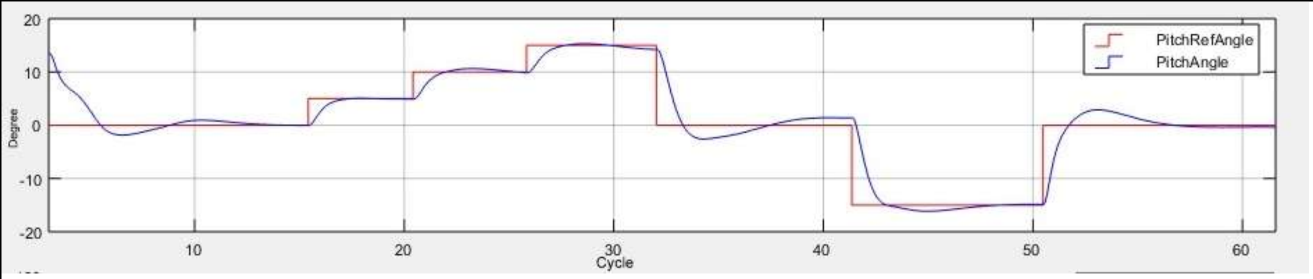
```
W= 35.000000 A= 0.112343 L = 83.884300 R = 83.659615
```

Figure 22: Outputs of Image Processing Algorithm

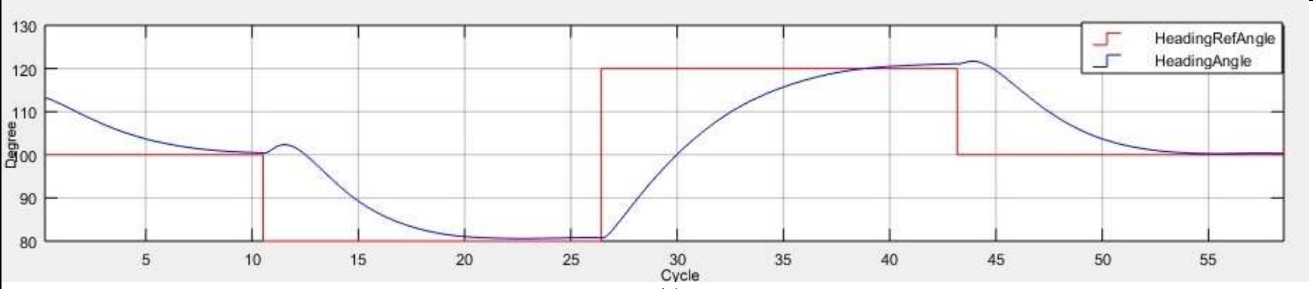
On the hold autopilot controller side, step inputs are supplied to test performance of the roll, pitch and heading controller. Altitude and speed controller tested indirectly while monitoring FMS outputs and response of UAV on these axes. For performance testing, flight mode selected as Hold Mode and reference values of axes are changed; corresponding results plotted at Matlab/Simulink model. Matlab/Simulink model runs with solver options of type fixed step size, solver of ode3 and with fixed step size of 20 msec. So every cycle value means 20 milliseconds at Figure 23.



(a)



(b)



(c)

Figure 23: (a) Roll Reference and Roll Angle, (b) Pitch Reference and Pitch Angle, (c) Heading Reference and Heading Angle

For FMS test, outputs of vision and GPS sourced altitude, heading, speed references are monitored. Result of simulation and test data can be investigated from Figure 24.

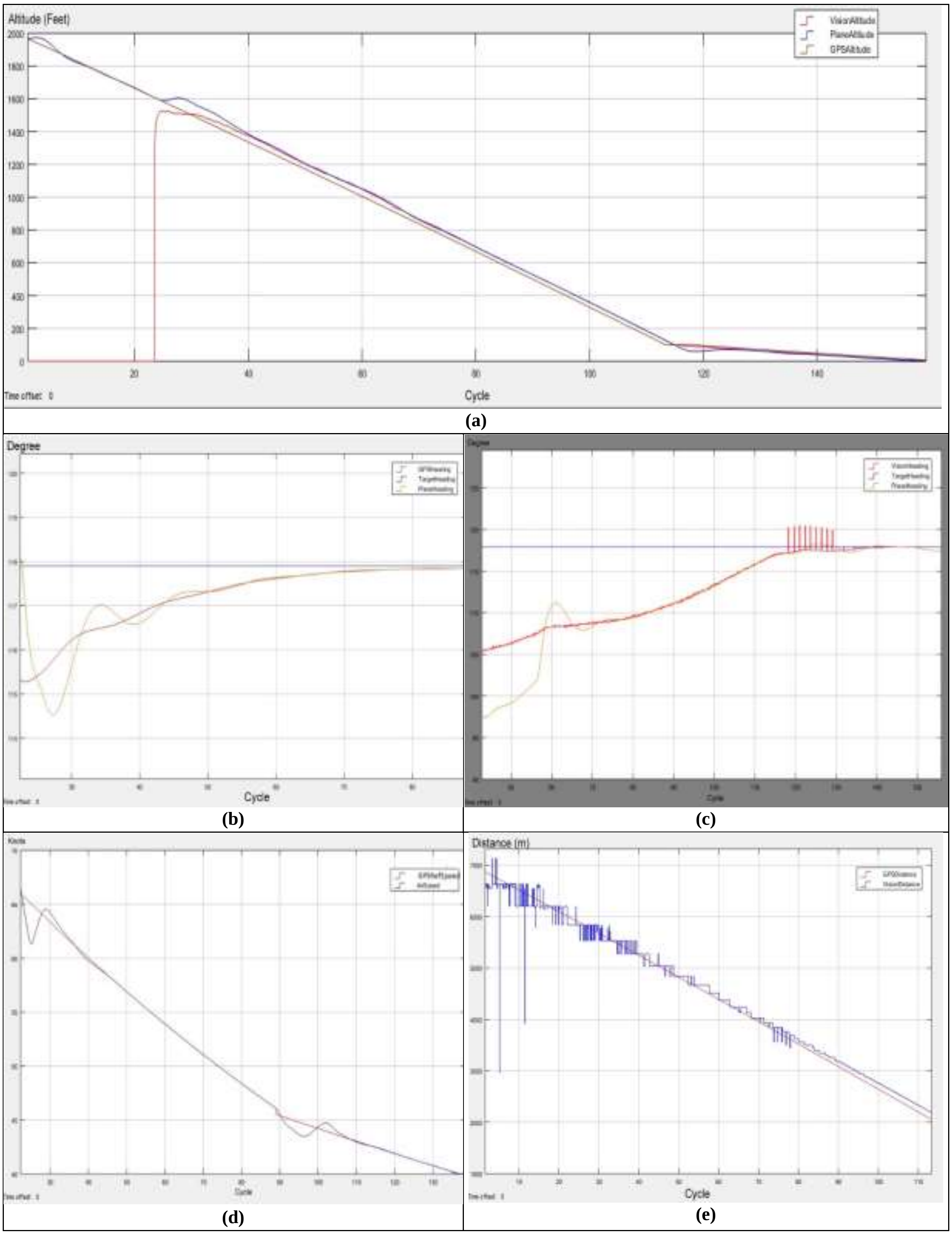


Figure 24: (a) Altitude Reference Generation by Using GPS and Vision Algorithm, (b) Heading Reference Generation by Using GPS, (c) Heading Reference Generation by Using Vision Algorithm, (d) Airspeed Reference Generation, (e) Distance Calculation to TDP by Using GPS and Vision Algorithm

Several simulation scenarios with different waypoint lists are applied such as like Figure 25 to be able to verify system reliability. Dotted line shows the way of UAV tracks while execution of 5 waypoints.

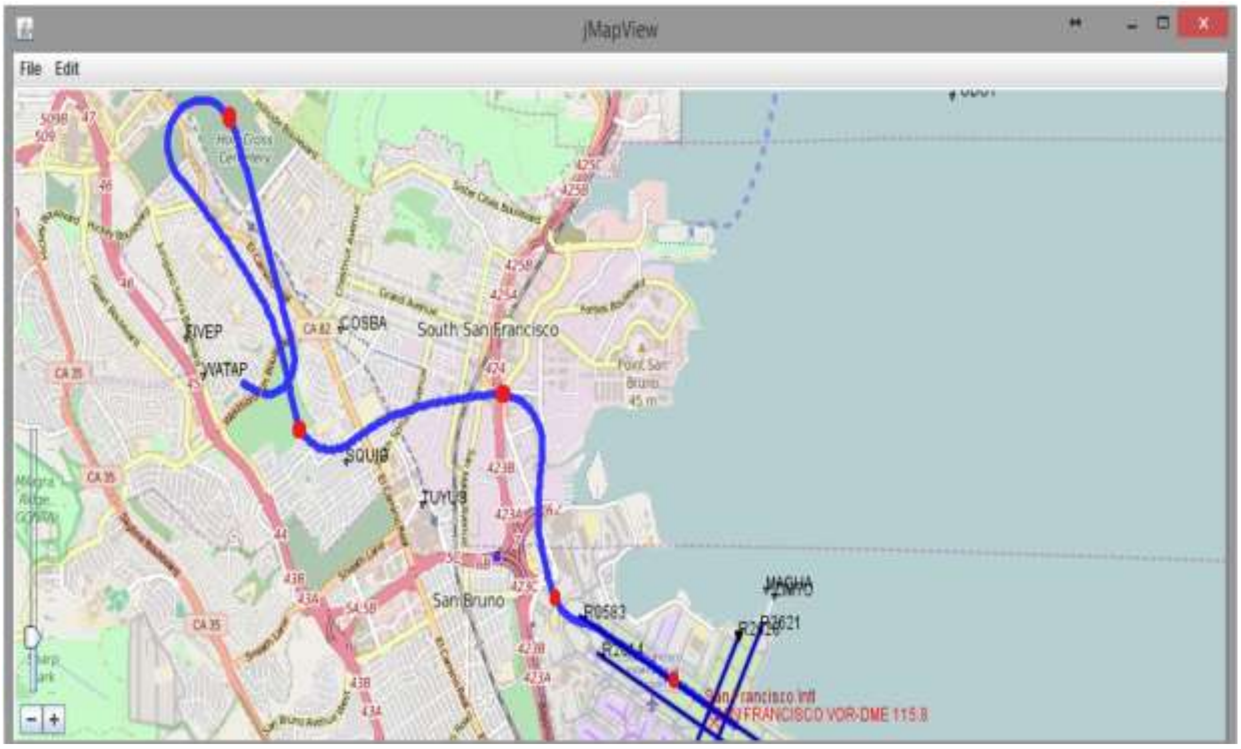


Figure 25: One of the Extreme Waypoint List Test Case for FMS

All simulation runs on an ultra-book with Core i7 2.4 GHz processor and 4 GB ram without any performance issue. Most of the resource is spent by Flightgear which is used to simulated image. Communication of Flightgear and Simulink model is via UDP so Flightgear can be run on another computer at the same network for a solution to lower powered computers. Image processing algorithm runs averagely every 60 milliseconds which is very fast because image processing algorithm waits for new image frame from Flightgear which is generating images about 20 fps (one image per 50 milliseconds). GCS sends aperiodic messages so for GCS performance consumption is not an issue.

8. CONCLUSION

In the present paper, a set of algorithms suggested for vision based auto landing which is applicable for fixed wing UAVs. As an output, a basic UAV design is developed with ground control station interface, hold autopilot controller, flight management system which use GPS or camera for position detection, airspeed sensors for detection of UAV speed, side slip sensor for measuring beta, and barometric sensor for altitude determination of UAV when GPS is not used. Although successive landing managed by vision data, due to resolution of generated image, resolution of calculated data from vision algorithm is also not as high as near positions to runway. For future work, camera zoom may be applied to increase resolution and algorithm can be used on a small fixed wing UAV.

REFERENCES

[7] "Israel Aerospace Industries", http://www.iai.co.il

[8] "General Atomics Aeronautical", http://www.ga-asi.com/

[9] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on, vol. 3. IEEE, 2002, pp. 2799–2804.

[10] S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, vol. 2. IEEE, 2001, pp. 1720–1727.

[11] Ufuk Suat Aydin, Engin Esin, "Development of vision based position estimation system and its use on autonomous takeoff and landing of rotary aerial vehicles," SIU 2014, pp. 176-179

[12] Andrew Miller, Mubarak Shah, Don Harper. Landing a UAV on a Runway using image registration. IEEE International Conference on Robotics and Automation. Pasadena, 2008:182-187

[13] Alison A.Proctor, Eric N.Johnson. Vision-only Approach and Landing [J]. AIAA . 2005:1-10.

[14] Andrew Miller, Mubarak Shah, Don Harper. Landing a UAV on a Runway using image registration. IEEE International Conference on Robotics and Automation. Pasadena, 2008:182-187

[15] "Matlab Examples", http://www.mathworks.com/examples/aeroblks/1005-fly-the-dehavilland-beaver

[16] Flightgear, http://www.flightgear.org/about/

[17] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[18] Sara S, Gomi H, Ninomiya T, et al.. Position and attitude estimation using image processing of runway. AIAA Aerospace Sciences Meeting and Exhibit. Reno, NV.2000:1-10.