VISION-AIDED LANDING FOR FIXED WING UNMANNED AERIAL VEHICLE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ENGİN ESİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

JUNE 2016

Approval of the thesis:

VISION-AIDED LANDING FOR FIXED WING UNMANNED AERIAL VEHICLE

Submitted by **ENGİN ESİN** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ozan Tekinalp _____
Head of Department, **Aerospace Engineering**

Asst. Prof. Dr. Ali Türker Kutay _____
Supervisor, **Aerospace Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Ozan Tekinalp _____
Aerospace Engineering Dept., METU

Asst. Prof. Dr. Ali Türker Kutay _____
Aerospace Engineering Dept., METU

Prof. Dr. Aydan Erkmen _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. DR. Yusuf Sahillioğlu _____
Computer Engineering Dept., METU

Asst. Prof. Dr. Ali Ruhşen Çete _____
Aerospace Engineering Dept., GAZU

Date: 03.06.2016

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : ENGİN ESİN

Signature :

ABSTRACT

VISION-AIDED LANDING FOR FIXED WING UNMANNED AERIAL VEHICLE

Esin, Engin

M.S., Department of Aerospace Engineering

Supervisor: Asst. Prof. Dr. Ali Türker Kutay

June 2016, 163 pages

The aim of this thesis is to design an autoland system for fixed wing unmanned aerial vehicle (UAV) to make auto landing by using position information calculated by image processing algorithms. With this ability, even if GPS is not available to be used, UAV still could make a safe automatic landing. Landing autopilot is aimed to keep UAV on a straight line with a constant flight path angle. Therefore, landing autopilot and computer vision methods are studied within the scope of this thesis. Also, to test designed system by sending control messages to landing autopilot, ground control station (GCS) software is developed. By using GCS interface, one can send commands to landing autopilot to analyze performance of the landing autopilot, activate or deactivate functions of the landing autopilot, change position data source as visual positioning system (VPS) or global positioning system (GPS) and change flight mode of the UAV. Besides testing and analyzing the system, GCS is used to prepare flight plans for landing. Waypoints of the prepared flight plan is applied by landing autopilot to keep trajectory points between two coordinates with keeping altitude and speed requests. To be able to manage that mission, waypoints include latitude, longitude, heading, altitude and speed specifications. Therefore, to be able to execute these waypoints; roll, pitch, altitude, heading and speed controllers

are designed. On the image processing side, position of aircraft is detected with respect to a known sized runway. This differential position information, which is obtained by image processing, is used instead of GPS information by landing autopilot to make a safe landing. Developed system has been successfully tested in flight simulation environment under several different wind and turbulence conditions with different initial orientations of the UAV.

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

ÖZ

SABİT KANATLI INSANSIZ HAVA ARAÇLARI İÇİN GÖRÜNTÜ İŞLEME İLE OTOMATİK İNİŞ

Esin, Engin
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ali Türker Kutay

Haziran 2016, 163 sayfa

Bu tezin amacı; görüntü işleme algoritmaları tarafından hesaplanan konum bilgilerini kullanarak sabit kanatlı insansız bir hava aracı (İHA) için otomatik iniş sistemi tasarlamaktır. Bu kabiliyet sayesinde GPS bilgisi mevcut olmasa bile, İHA yine de güvenli olarak otomatik iniş yapabilecektir. İniş otopilotu sabit uçuş yolu açısı ile düz bir yörünge üzerinde hava aracını yönlendirmeyi amaçlanmaktadır. Bu nedenle bu tez kapsamında, iniş otopilotu ve görüntü işleme yöntemleri incelenmiştir. Ayrıca, tasarlanan iniş otopilot sistemini kontrol mesajları göndererek test edebilmek için yer kontrol istasyonu (YKI) yazılım geliştirilmiştir. YKI arabirimini kullanarak, iniş otopilot performansını analiz etmek, otopilot fonksiyonlarını devre dışı bırakmak veya etkinleştirmek, konum veri kaynağını görüntü konumlandırma sistemi veya küresel konumlandırma sistemi olarak değiştirmek ve uçuş modlarını değiştirmek için iniş otopilotuna komut göndermek için kullanılabilir. Sistem testi ve analizinin yanı sıra, YKI arabirimi iniş için uçuş planları hazırlamak için de kullanılır. İniş otopilotu hazırlanan bu uçuş planındaki koordinatlar arasındaki yörünge noktalarını irtifa ve hız gereksinimlerini de karşılayarak tutmaya çalışmaktadır. İniş otopilotunun bu görevi yönetebilmesi için YKI tarafından yüklenen hedef noktaları enlem, boylam, uçuş başı, yükseklik ve hız gereksinimlerini içermektedir. Bu nedenle iniş otopilotu kapsamında yanal, yunuslama, uçuş başı, yükseklik ve hız kontrolcüsü

tasarlanmıştır. Görüntü işleme ile hava taşıtının diferansiyel konumu boyutları bilinen bir piste göre hesaplanır. Görüntü işleme ile elde edilen bu diferansiyel konum bilgisi, güvenli bir iniş yapmak için iniş otopilotu tarafından GPS bilgisi yerine kullanılabilir. Geliştirilen sistem uçuş simülatörü ortamında farklı rüzgar ve türbülans koşulları altında ve IHA'nın farklı ilk konumlarını içerecek şekilde başarı ile test edilmiştir.

Anahtar Kelimeler: Otomatik İniş, İnsansız Hava Aracı, Görüntü İşleme İle Kontrol, Görüntü İşleme İle Seyrüsefer, Otomatik Uçuş Kontrol Sistemleri, Güdüm Ve Seyrüsefer Sistemi.

To my parents...

ACKNOWLEDGMENTS

First, I would like to thank meaning of my life Tutku Mısırlıoğlu for her unlimited support and understanding. She always encouraged me for my studies and without her support, none of this would be possible. I would like to special thank my loving mother, Nuran Esin, and my dear father, Ahmet Şahin Esin, for their unconditional support during my life, and for inspiring me to follow my dreams.

I would like to give special thanks to my supervisor, Asst. Prof. Dr. Ali Türker Kutay, for his guidance, encouragement, and great support throughout my M.S. study. He always there to listen and advice when I need. I am very grateful to him for his patience during the research meetings and productive discussions. Also, I would like to thank Prof. Dr. Aydan Erkmen for productive discussions, which carried me forward to this day both technically and morally.

I am grateful to Ercüment Türkoğlu, Hakan Tiftikçi, Seçkin Arıbal, Yavuz Öztürk and Devrim Korkmaz at TAI for their helpful advices and discussions that guide me through. I would like to extend my thanks to Serdar Topaloğlu, Fahri Çaycı, Emrah Hışır, Adnan Özkart, Mustafa Kemal Işıklar, Engin Çelen, Ufuk Suat Aydın and Ahmet Gürbüz for their valuable friendship and support during thesis.

TABLE OF CONTENTS

ABSTRACT ..... **V**

ÖZ ..... **VII**

ACKNOWLEDGMENTS ..... **X**

TABLE OF CONTENTS ..... **XIII**

LIST OF TABLES ..... **XVI**

LIST OF FIGURES ..... **XVII**

LIST OF SYMBOLS ..... **XXV**

CHAPTERS

1 INTRODUCTION **1**

1.1 Background and Motivation 1

1.2 Design Goal 4

1.3 Literature Survey 6

1.4 Organization of the Thesis 8

2 AIRCRAFT DYNAMICS **9**

2.1 Platform 9

2.2 Aerodynamic Model 11

2.3 Reference Frames 14

2.4 Nonlinear Equations of Motion 15

2.5 Linear Equations of Motion and Analysis 17

2.5.1 Longitudinal Dynamics 18

2.5.2 Lateral Dynamics 19

3 RUNWAY DETECTOR ALGORITHM **21**

3.1 Image Capturing and Light Bulbs Detection 22

3.1.1 Capturing FlightGear Window 22

3.1.2 Thresholding Method 23

3.1.3 Morphological Method 25

3.2 POINTS TO LINE CONVERTER (RANSAC) 30
3.3 DETERMINATION OF RUNWAY CANDIDATES 31
3.4 RUNWAY CANDIDATE FILTER 33
3.5 EXTRACT INFORMATION OF TARGET RUNWAY 38

4 LANDING AUTOPILOT 41

4.1 CONTROLLER REQUIREMENTS 43
4.2 PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) CONTROLLER 44
4.2.1 Back-Calculation Anti-Windup Method 47
4.2.2 Clamping Anti-Windup Method 49
4.2.3 Time Domain Performance 51
4.3 INNER LOOP 51
4.3.1 Lateral Controller 52
Roll Controller 52
Heading Controller 56
4.3.2 Longitudinal Controller 59
Pitch & Altitude Controller 59
Airspeed Controller 66
4.4 OUTER LOOP 68
4.4.1 Altitude and Airspeed Reference Calculation 73
Distance Calculation by Using GPS 75
Distance Calculation by Using VPS 76
4.4.2 Course Controller 83
Course Control by Using GPS 84
Course Control by Using VPS 89

5 SIMULATION AND RESULTS 95

5.1 OVERVIEW 95
5.2 CASE 1: NO WIND AND TURBULENCE 96
5.2.1 GPS Based Landing 96
5.2.2 VPS Based Landing 101
5.3 CASE 2: 15 KTS CROSS WIND CONDITION 105
5.3.1 GPS Based Landing 106

5.3.2 VPS Based Landing 111
5.4 CASE 3: 15 KTS CROSS WIND WITH CROSS TRACK ERROR 117
5.4.1 GPS Based Landing 117
5.4.2 VPS Based Landing 123
5.5 CASE 4: 15 KTS CROSS WIND + TURBULENCE 128
5.5.1 GPS Based Landing 130
5.5.2 VPS Based Landing 135
5.6 CASE 5: 20 KTS NOSE WIND 140
5.7 CASE 6: 6 KTS TAIL WIND 143

6 CONCLUSION 147

6.1 SUMMARY AND CONCLUSIONS 147
6.2 FUTURE WORKS 148

REFERENCES 151

APPENDIX

A. GROUND CONTROL STATION (GCS) 155

A.1 FUNCTION AND INTERFACE OF GCS 155
A.2 CONTROL COMMANDS 160
A.3 MISSION COMMANDS 161

LIST OF TABLES

TABLES

Table 1-1 ILS Performance Categories 2
Table 2-1 General Aircraft Data of the Beaver [4] 10
Table 2-2 Aircraft data on which the aerodynamic model is based [4] 11
Table 2-3 Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-rang [4] 13
Table 2-4 Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-rang [4] 13
Table 2-5 Longitudinal mode characteristics 18
Table 2-6 Lateral Mode Characteristics 19
Table 3-1 Message Table of VISION_DATA 40
Table 4-1 Control Surfaces 42
Table 4-2 Landing Autopilot Requirements 43
Table 4-3 Environmental Requirements for Landing 44
Table 4-4 Actuator Limit Constraints 44
Table 4-5 Inner Loop Autopilot Capabilities 52
Table 4-6 Gain Values and Time Domain Performance of Roll Controller 56
Table 4-7 Gain Values and Time Domain Performance of Heading Controller 58
Table 4-8 Gain Values and Time Domain Performance of Pitch Controller 61
Table 4-9 Gain Values and Time Domain Performance of Altitude Controller 64
Table 4-10 Gain Values and Time Domain Performance of Speed Controller 68
Table 4-11 Example of a Flight Plan Instance 73
Table 5-1 Waypoints Generated for Simulations 95
Table 5-2 Parameters of Configured Von Karman Wind Turbulence Model 129
Table A - 1 Message Table of GAM_CONTROL_DATA 161
Table A - 2 Message Table of GAM_MISSION_DATA 163
Table A - 3 Message Table of GAM_Mission_WayPoint Composition 163

LIST OF FIGURES

FIGURES

Figure 1-1 ILS Performance Categories [4] 2

Figure 1-2 System Overview 5

Figure 2-1 de Havilland Canada DHC-2 Beaver 9

Figure 2-2 Earth and Body Axes 14

Figure 2-3 Definition of Inertia Coefficients [25] 16

Figure 2-4 Initial conditions for V = 45 m/s, H = 6000 ft obtained with the trim routine ACTRIM command [4]. 17

Figure 3-1 Runway Detection Algorithm 21

Figure 3-2 FlightGear Simulator Interface 22

Figure 3-3 Captured Image from Flightgear Screen 23

Figure 3-4 Threshold Result of Red, Green, and Blue Pixels for Light Bulbs Detection 24

Figure 3-5 All White Pixels 25

Figure 3-6 Dilation Example: (a) Set A. (b) Square Structuring Element (Dot is the Center), (c) Dilation of A by B, Shown Shaded. (d) Elongated Structuring Element. (e) Dilation of A Using This Element [27] 26

Figure 3-7 Erosion Example: (a) Set A. (b) Square Structuring Element (Dot is the Center). (c) Erosion of A by B Shown Shaded. (d) Elongated Structuring Element. (e) Erosion of A Using This Element [27] 27

Figure 3-8 Structuring element B rolling along the inner boundary of A (the dot indicates the origin of B). The heavy line is the outer boundary of the opening. Final output is shaded. [27] 28

Figure 3-9 Result of Opening Method 28

Figure 3-10 Selection of White Bulbs. Original Image Subtracted From Opening Image 29

Figure 3-11 RANSAC Algorithm Steps 30

Figure 3-12 Line Detection Algorithm Result 31

Figure 3-13 Vanishing Point Illustration 32

Figure 3-14 Output of Runway Candidates Algorithm. 4 Lines Results with 6 Candidates 33

Figure 3-15 Elimination of Candidate Runways 37

Figure 3-16 Focal Length and FOV Relation 39

Figure 3-17 Runway Edges 39
Figure 4-1 Inner Loop Control System Representation 41
Figure 4-2 Control Surface on DeHavilland Beaver 42
Figure 4-3 Block Diagram of PID Controller 45
Figure 4-4 Illustration of integral windup where output is y, set point is y_{sp}, control signal is u and integral part is I 47
Figure 4-5 Back- Calculation Anti-Windup Method 48
Figure 4-6 Illustration of Back-calculation method effects on controller 49
Figure 4-7 Integrator clamping $et \cdot ut > 0$ 50
Figure 4-8 Clamping Anti-Windup Effect 50
Figure 4-9 Characteristics of closed-loop response to step in reference [32] 51
Figure 4-10 Roll Controller 53
Figure 4-11 “RollRefConstructor” Block. Sets Roll Reference with respect to Requirements stated in Table 4-2 54
Figure 4-12 Roll Controller 54
Figure 4-13 PID Tuning and Step Response Performance Analysis for Roll Controller 55
Figure 4-14 Step Response of Roll Controller 56
Figure 4-15 Heading Controller Algorithm 57
Figure 4-16 “Heading” Block 57
Figure 4-17 PID Tuning and Step Response Performance Analysis for Heading Controller 58
Figure 4-18 Step Response of Heading Controller 59
Figure 4-19: Pitch Controller 60
Figure 4-20 “Pitch” Block 60
Figure 4-21 PID Tuning and Step Response Performance Analysis for Pitch Controller 61
Figure 4-22 Step Response of Pitch Controller 62
Figure 4-23 Altitude Hold Controller 63
Figure 4-24: PID Block of Altitude Controller 63
Figure 4-25 PID Tuning and Step Response Performance Analysis for Altitude Controller 64
Figure 4-26 Step Response of Altitude Hold Controller 64
Figure 4-27 Altitude Controller Tracks Landing Trajectory Successfully 65
Figure 4-28 Air Speed Controller 66

Figure 4-29: Air Speed Controller and Motor Model 66
Figure 4-30 PID Tuning and Step Response Performance Analysis for Speed Controller 67
Figure 4-31 Step Response of Speed Hold Controller 68
Figure 4-32 Outer Loop Feeds Inner Loop with Altitude, Airspeed and Heading References 69
Figure 4-33 Piecewise linear interpolation of x= [1 2 3 4 5] and y= [10 9 4 4 0] 70
Figure 4-34 Piecewise cubic interpolation 71
Figure 4-35 Position Source Selection Algorithm 73
Figure 4-36 Altitude References with respect to Table 4-11 Generated by Using PCHIP and Piecewise Linear Interpolation 74
Figure 4-37 Airspeed References with respect to Table 4-11 Generated by Using PCHIP and Piecewise Linear Interpolation 74
Figure 4-38 Distance Calculation by Using VPS Data 76
Figure 4-39 Error rate in Distance Calculation for 1 Pixel Error While Altitude of 500 Meters 77
Figure 4-40 Error Value in Distance Calculation for 1 Pixel Error While Altitude of 500 Meters 78
Figure 4-41 Distance Calculation Case 1 78
Figure 4-42 Distance Calculation Case 2 79
Figure 4-43 Distance Calculation Case 3 79
Figure 4-44 Calculated Distance to TDP by GPS and VPS 81
Figure 4-45 Unfiltered Altitude Reference Generated Using VPS 82
Figure 4-46 Filtered Altitude Reference Generated Using VPS 82
Figure 4-47: Landing maneuver sequence While Crosswind is Exist 83
Figure 4-48 Course (Track) Hold While Crosswind is Exist 84
Figure 4-49 Bearing Error and Line Fit Algorithm 85
Figure 4-50: GPS Cross Track Controller 86
Figure 4-51: Cross Track Controller Algorithm 87
Figure 4-52 Step Response of Cross Track Controller 88
Figure 4-53 GPS Decrab Maneuver under 15 kts Side Wind 89
Figure 4-54 Runway Orientations. L is Angle of Left Edge, R is Angle of Right Edge 90
Figure 4-55 Three Condition To Calculate Heading Reference. (a) UAV Is Right Side of the Runway, (b) UAV Is In Front of the Runway, (C) UAV Is Left Side of the Runway 90

Figure 4-56 Effect of Heading Change on (Right - Left) Angle of the Runway at 100m and 3400m away from Runway Threshold 91

Figure 4-57: Vision Cross Track Controller 92

Figure 4-58 Step Response of Cross Track Controller of Vision Based Positioning Algorithm 93

Figure 4-59 VPS Decrab Maneuver under 15 kts Side Wind 94

Figure 5-1 Tracking of the UAV Position While Landing for Case 1: No Wind and Turbulence GPS Based Landing 96

Figure 5-2 Vehicle Position and Orientation for Case 1: No Wind and Turbulence GPS Based Landing. Left is Pilot Cam, Right is View from Back 97

Figure 5-3 Landing Trajectory Path and Altitude of the UAV for Case 1: No Wind and Turbulence GPS Based Landing 97

Figure 5-4 Ground Shock at Touchdown for Case 1: No Wind and Turbulence GPS Based Landing 98

Figure 5-5 h for Case 1: No Wind and Turbulence GPS Based Landing 99

Figure 5-6 Variation of Theta for Case 1: No Wind and Turbulence GPS Based Landing 99

Figure 5-7 Variation of Speed for Case 1: No Wind and Turbulence GPS Based Landing 100

Figure 5-8 Tracking of the UAV Position While Landing for Case 1: No Wind and Turbulence VPS Based Landing 101

Figure 5-9 Vehicle Position and Orientation for Case 1: No Wind and Turbulence VPS Based Landing. Left is Pilot Cam, Right is View from Back 101

Figure 5-10 Landing Trajectory Path and Altitude of the UAV for Case 1: No Wind and Turbulence VPS Based Landing 102

Figure 5-11 Ground Shock at Touchdown for Case 1: No Wind and Turbulence VPS Based Landing 103

Figure 5-12 h for Case 1: No Wind and Turbulence Case 1: No Wind and Turbulence VPS Based Landing 104

Figure 5-13 Variation of Theta for Case 1: No Wind and Turbulence VPS Based Landing 104

Figure 5-14 Variation of Speed for Case 1: No Wind and Turbulence VPS Based Landing 105

Figure 5-15 Tracking of the UAV Position While Landing for Case 2: 15 kts Cross Wind Condition GPS Based Landing 106

Figure 5-16 Vehicle Position and Orientation for Case 2: 15 kts Cross Wind Condition GPS Based Landing. Left is Pilot Cam, Right is View from Back 106

Figure 5-17 Cross Track Performance for Case 2: 15 kts Cross Wind Condition GPS Based Landing 107

Figure 5-18 Roll Angle of the UAV for Case 2: 15 kts Cross Wind Condition GPS Based Landing..... 108

Figure 5-19 Landing Trajectory Path and Altitude of the UAV for Case 2: 15 kts Cross Wind Condition GPS Based Landing 109

Figure 5-20 Ground Shock at Touchdown for Case 2: 15 kts Cross Wind Condition GPS Based Landing 109

Figure 5-21 Variation of Theta for Case 2: 15 kts Cross Wind Condition GPS Based Landing 110

Figure 5-22 h for Case 2: 15 kts Cross Wind Condition GPS Based Landing 110

Figure 5-23 Variation of Speed for Case 2: 15 kts Cross Wind Condition GPS Based Landing 111

Figure 5-24 Tracking of the UAV Position While Landing for Case 2: 15 kts Cross Wind Condition VPS Based Landing 112

Figure 5-25 Vehicle Position and Orientation for Case 2: 15 kts Cross Wind Condition VPS Based Landing. Left is Pilot Cam, Right is View from Back..... 112

Figure 5-26 Cross Track Performance for Case 2: 15 kts Cross Wind Condition VPS Based Landing..... 113

Figure 5-27 Roll Angle of the UAV for Case 2: 15 kts Cross Wind Condition VPS Based Landing..... 113

Figure 5-28 Landing Trajectory Path and Altitude of the UAV for Case 2: 15 kts Cross Wind Condition VPS Based Landing 114

Figure 5-29 Ground Shock at Touchdown for Case 2: 15 kts Cross Wind Condition VPS Based Landing 115

Figure 5-30 Variation of Theta for Case 2: 15 kts Cross Wind Condition VPS Based Landing 115

Figure 5-31 h for Case 1: No Wind and Turbulence Case 2: 15 kts Cross Wind Condition VPS Based Landing 116

Figure 5-32 Variation of the Speed for Case 2: 15 kts Cross Wind Condition VPS Based Landing..... 116

Figure 5-33 Tracking of the UAV Position While Landing for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing 118

Figure 5-34 Vehicle Position and Orientation for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing. Left is Pilot Cam, Right is View from Back 118

Figure 5-35 Cross Track Performance for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing..... 119

Figure 5-36 Roll Angle of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing..... 120

Figure 5-37 Landing Trajectory Path and Altitude of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing 120

Figure 5-38 Variation of Theta for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing 121

Figure 5-39 Ground Shock at Touchdown for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing..... 121

Figure 5-40 h for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing..... 122

Figure 5-41 Variation of Speed for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing 122

Figure 5-42 Tracking of the UAV Position While Landing for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 123

Figure 5-43 Vehicle Position and Orientation for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing. Left is Pilot Cam, Right is View from Back 123

Figure 5-44 Cross Track Performance for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 124

Figure 5-45 Initial Position of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 125

Figure 5-46 Roll Angle of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing 125

Figure 5-47 Landing Trajectory Path and Altitude of the Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing 126

Figure 5-48 Variation of Theta for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing 126

Figure 5-49 Ground Shock at Touchdown for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 127

Figure 5-50 h for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 127

Figure 5-51 Variation of the Speed for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing 128

Figure 5-52 Wind Profile for Case 4: 15 kts Cross Wind + Turbulence..... 129

Figure 5-53 Tracking of the UAV Position While Landing for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing 130

Figure 5-54 Vehicle Position and Orientation for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing. Left is Pilot Cam, Right is View from Back..... 130

Figure 5-55 Cross Track Performance for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing 131

Figure 5-56 Roll Angle of the UAV for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing 132

Figure 5-57 Landing Trajectory Path and Altitude of the UAV for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing 133

Figure 5-58 Variation of Theta for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing..... 133

Figure 5-59 Ground Shock at Touchdown for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing 134

Figure 5-60 h for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing... 134

Figure 5-61 Variation of Speed for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing..... 135

Figure 5-62 Tracking of the UAV Position While Landing for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing 135

Figure 5-63 Vehicle Position and Orientation for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing. Left is Pilot Cam, Right is View from Back 136

Figure 5-64 Cross Track Performance for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing 137

Figure 5-65 Roll Angle of the UAV for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing 137

Figure 5-66 Landing Trajectory Path and Altitude of the UAV for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing..... 138

Figure 5-67 Variation of Theta for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing..... 139

Figure 5-68 Ground Shock at Touchdown for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing 139

Figure 5-69 h for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing... 140

Figure 5-70 Variation of the Speed for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing..... 140

Figure 5-71 Landing Trajectory Path and Altitude of the UAV for Case 5: 20 kts Nose Wind VPS Based Landing 141

Figure 5-72 Variation of Theta for Case 5: 20 kts Nose Wind VPS Based Landing 142

Figure 5-73 Ground Shock at Touchdown for Case 5: 20 kts Nose Wind VPS Based Landing 142

Figure 5-74 h for Case 5: 20 kts Nose Wind VPS Based Landing..... 143

Figure 5-75 Variation of the Speed for Case 5: 20 kts Nose Wind VPS Based Landing 143

Figure 5-76 Landing Trajectory Path and Altitude of the UAV for Case 6: 6 kts Tail Wind VPS Based Landing 144

Figure 5-77 Variation of Theta for Case 6: 6 kts Tail Wind VPS Based Landing .. 145

Figure 5-78 Ground Shock at Touchdown for Case 6: 6 kts Tail Wind VPS Based Landing 145

Figure 5-79 h for Case 6: 6 kts Tail Wind VPS Based Landing..... 146

Figure 5-80 Variation of the Speed for Case 6: 6 kts Tail Wind VPS Based Landing 146
Figure 6-1 Result of Developed Vision Algorithms on Real World Frames 150
Figure A- 1 Ground Control Station Interface 156
Figure A- 2 Altitude Generation Example 157
Figure A- 3 Altitude Calculation Algorithm 158
Figure A- 4 Waypoint Generator 159
Figure A- 5 Waypoints and Calculated Headings in Degree 162

LIST OF SYMBOLS

Symbol	Description	Units
AR	*Aspect ratio*	-
fov	*Camera filed-of-view angle in the image*	*deg*
ϕ,θ,ψ	*Euler angles (in roll, pitch and yaw planes)*	*deg*
u, v, w	*Velocity components in body fixed reference*	*mps*
p, q, r	*Components of angular velocity in the body fixed frame with respect to the earth fixed*	*deg/s*
α	*Angle of attack*	*deg*
β	*β side slip angle*	*deg*
C_x	*Drag force coefficient*	
Cy	*Side force coefficient*	
Cz	*Lift force coefficient*	
Cl	*Rolling moment coefficient*	
Cm	*Pitching moment coefficient*	
C_n	*Yawing moment coefficient*	
I	*Inertia matrix*	
I_{xy} , I_{xz} , I_{yz}	*Cross inertia terms*	
Ω	*Angular velocity*	*rad/sec*

Abbreviations

2-D	Two-Dimensional
3-D	Three-Dimensional
UAV	Unmanned Air Vehicle
GAM	Ground to Air Message
FOV	Field-of-View
GPS	Global Positioning System
VPS	Visual Positioning System

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The integration of robotic technologies into UAVs suggest new solutions to problems such as landing of UAVs. Landing is one of the most critical phases of the operation for UAVs. Operating a UAV under high winds, turbulence or wind shear weather conditions is highly dependent on user ability to land safely. Moreover, even in no wind conditions, a small error in guidance or control could cause system loss or damages. At this point the importance of an automatic landing system comes out such that both safety of flight is increased and workload of the pilot is significantly reduced during landing phase. Automatic landing systems increase wind limits of the aircrafts for safe landing.

First automatic landing was made by British Airways during a commercial flight in June 1965 using automatic flare controller in pitch axis [1] and in 1996, landing with full control in three axes was made. For landing phase of an aircraft, position information of the vehicle is needed. Since 1920, landing aids for guiding pilots to down aircraft to flare height have been developed [2]. In 1930, H. Diamon and F.W. Dunmore presented a landing aid based on radio navigation similar to today's Instrumented Landing System (ILS) [3]. Today's commercial autoland systems use GPS and ILS to lower the aircraft with -3º glide path angle while keeping the aircraft's track aligned with runway centerline.

Auto landing equipment is certificated to CAT I, CAT II or CAT III, which is categorized with respect to minimum altitude above touchdown zone that the automatic landing system could be used as presented in Figure 1-1. This minimum altitude is named as decision height and at decision height, the pilot must have runway in sight and be able to see as far down the runway. This distance is named as runway visual range.

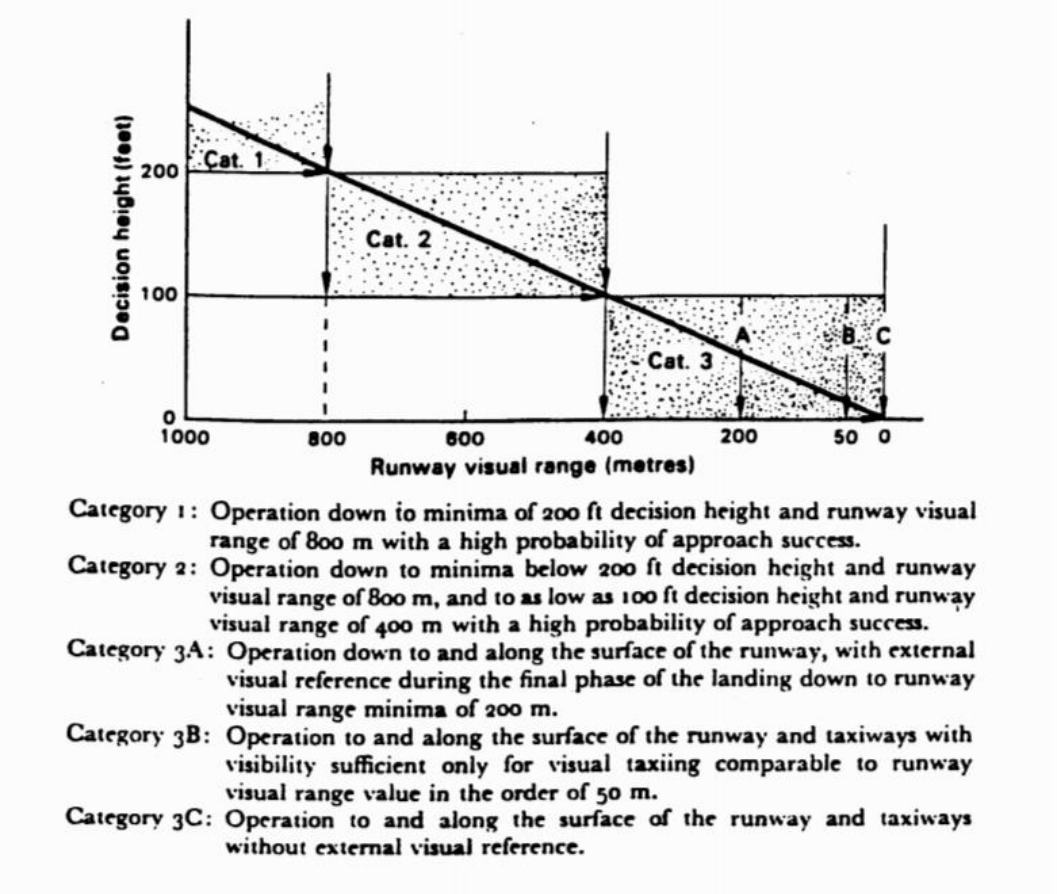


Figure 1-1 ILS Performance Categories [4]

Category	**Decision Height (feet)**	**Runway Visual Range (meters)**
CAT I	≥ 200	≥ 550
CAT II	200 > DH ≥ 100	≥ 350
CAT III A	< 100 or no DH	≥ 200
CAT III B	< 50 or no DH	200 > RVR ≥ 50
CAT III C	No minima	No minima

Table 1-1 ILS Performance Categories

Commercial aircraft commonly have CAT III certificated equipment but general aviation aircraft do not typically have due to high costs. Furthermore, certificated equipment is needed at aircraft, runways also must have ground-based equipment. At Turkey, only runway 3 of Esenboğa (Ankara) and runway 5 of Atatürk (İstanbul) airports have CAT III systems and many of the runways do not have ILS system at all.

In 1994, automatic landing is managed using only GPS (DGPS) for positioning information with a Boeing 757 [5]. DGPS uses local ground stations installed at the airport to transmit correction for the raw GPS signal received from satellite. Disadvantages of this system is that it is highly dependent on quality of wireless link between ground station and aircraft. Any delay or link loss cause high position error. Source of GPS service is also very important. Accuracy in position calculation is highly dependent on number of satellites sources and this highly dependent on weather conditions. DGPS service may be unavailable due to dense cloud cover, solar flares, and permanent obstructions such as trees and buildings.

Common disadvantages of DGPS and ILS system is high cost. Also, in hostile environment, none of this system could be used due to their ground-based equipment requirements. In addition to that, due to dependence of radio communications, easiness of jamming is a big problem. In USA, National Space-Based Positioning, Navigation, and Timing Advisory Board reports [6] that GPS has a national security threat due to its weakness to jamming. GPS is open to attacks from devices capable of jamming as little as 30 US Dollars [7]. For this reason, an alternative method for especially military UAV navigation is essential to be developed. In addition to the jamming problem, most commonly-used GPS units do not meet performance requirements for high resolution needed navigation tasks such as UAV landing on runways, or on helipads. This Thesis demonstrates a proof-of-concept of a method that can be modified to meet these requirements with image processing which is more difficult to jam than GPS or ILS.

Vision-based state estimation for robotic platforms is an attractive method due to its passive nature. This passive nature makes optical sensing much more robust against jamming techniques than electromagnetic waves used in other positioning devices (radar, lidar, magnetometers, etc.). Cameras also generally discard radiation from outside of their field of view, unlike omnidirectional GPS antennas that can be jammed from any aspect. Digital cameras are easy to obtain, inexpensive, lightweight, low powered, and more robust to jamming than electromagnetic interfaces as mentioned. With development of powerful single board computers, vision algorithms is applicable to use on mobile platforms more widely.

1.2 Design Goal

The aim of this thesis is to design an autoland system to manage auto landing with the position information calculated by image processing. So that, process of landing can continue even when GPS information does not exist. Autoland controller is aimed to keep UAV on a straight line with a constant flight path angle even if a crosswind is present. Therefore, landing autopilot and computer vision methods are studied within the scope of this thesis.

National Space-Based Positioning, Navigation, And Timing (PNT) Advisory Board reports [6] that GPS has a national security threat due to its weakness to jam. GPS is open to attacks from as little as 30 US Dollars devices [7]. In addition to the jamming problem, most commonly-used GPS units do not meet performance requirements for high resolution needed navigation tasks such as UAV landing on runways, or on helipads. For this reason, an alternative method for especially military UAV navigation is essential to be developed. This Thesis demonstrates a proof-of-concept of a method that if GPS information is not suitable to use for any reason, position information of the aircraft is possible to be calculated by image processing for a safe landing.

The statistics about flight accidents shows that 67% of the accidents are due to human factors as the primary cause and 5% are attributed to weather factors. With respect to the flight phases, 47% accidents occur during the final approach or landing of aircrafts [8]. Due to this fact, landing algorithms have been developed to decrease accident possibility.

Landing autopilots usually consist of an inner loop which controls the faster rotational dynamics and an outer loop which controls the position and velocities. A challenging part of the landing phases is the decrab maneuver which is necessary to align the aircraft heading with the runway just before touchdown and the flare maneuver which is necessary to reduce the aircraft vertical speed that is acceptable for touchdown. During decrab maneuver, roll and yaw angles are commanded directly instead of position or velocity. With the flare maneuver, touching to ground is managed with positive theta angle.

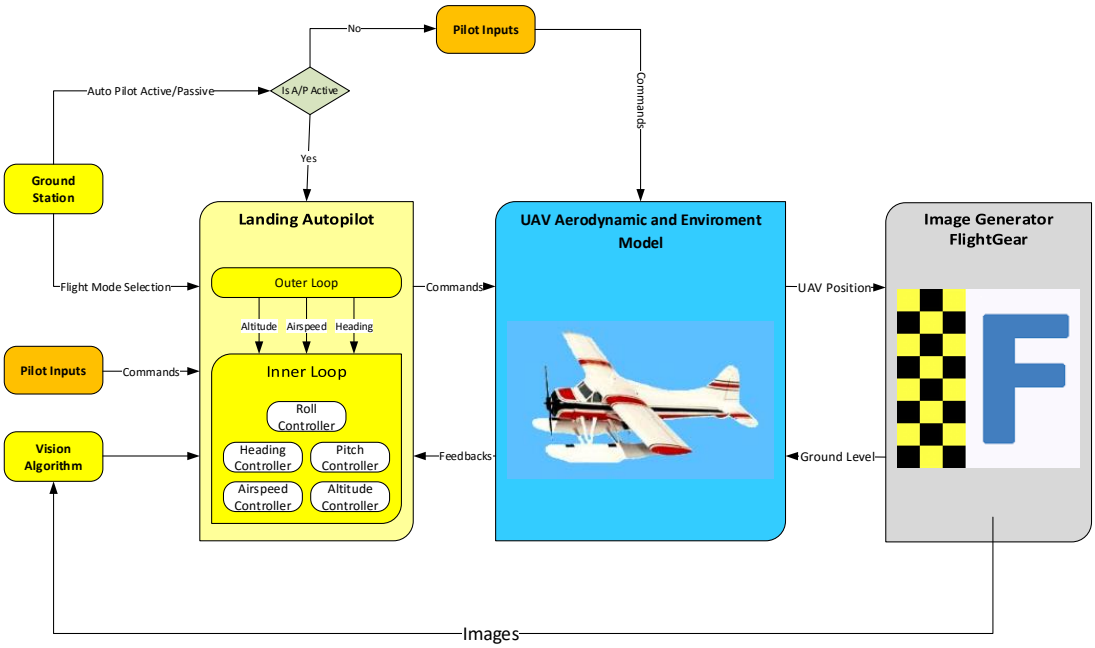


Figure 1-2 System Overview

The functionality and general algorithm of the developed UAV system is described in Figure 1-2. The system components can be divided into three parts. Landing autopilot, ground station interface (GCS), 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 which is obtained from M.O. Rauw thesis work [4]. Developed ground control interface and application of image processing algorithm communicates with Matlab/Simulink project via UDP protocol. FlightGear [10] simulation tool is used as an image generator for image processing part. Also altitude with respect to ground level of the aircraft is obtained from this tool. FlightGear is a free and open source flight simulator.

1.3 Literature Survey

There are many works in the literature related to automatic landing. These works can be separated into two groups, traditional control methods, and modern control methods. Kargin [11] works on the design of a lateral and longitudinal autopilot for a UAV. Aim of that work is to design an autopilot that is capable to land the aircraft under severe weather conditions within predefined limits. Cho et al. [12] worked on Linear Quadratic Regulator (LQR) Controller for automatic landing with only differential GPS.

Development of the automatic landing systems was based on the experience collection in many years in HERON. The automatic landing of the Heron UAV under crosswinds is investigated in Attar et. al. [13].This platform uses DGPS augmented by a radar altimeter and a laser tracker for measuring altitude, heading and distance of UAV. With the combination of the data generated from UAV and ground based sensors mentioned above, lateral and vertical position and orientation are calculated to control glide slope and flight path for landing until UAV stops completely.

When the aircraft transitions from the glide path to the flare path, the UAV tends to an unstable region [14]. This article investigates the instability that occurs during the change in landing phase. The article formulates a blending function to overcome instability during transition and concludes with simulation results that show a smooth stable transition from the glide slope to flare phase. Implementation of a blending function for a stable transition from glide slope to flare path was considered in this project. This problem is resolved by using exponential flare calculation with respect to time as in [15], [11] and by using again exponential flare calculation but with distance to touchdown point as in [16], [17].

For rotary wing UAVs, many work has been done in automatic landing by using image processing like [18], [19] and [20] , however, these algorithms all rely on a coplanar assumption and a customized landing pad, and are not necessarily suitable for use with a fixed-wing vehicle. Andrew [21] 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. Some way of vision based automatic landing for fixed wing UAVs have been published like [22] 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 [23] proposes that its method can recognize the runway by placing red marker on corners, which is useful, but not practical. Cesetti et al [24] took a different approach and developed a system that uses natural landmarks and SIFT (Scale-Invariant Feature Transform) features in combination with satellite imagery to estimate position in all three axes as well as heading. However, this approach only works if the image is taken normal to the ground plane (i.e., pitch and roll are both constant), which certainly cannot be guaranteed on an airplane during landing. It could be used, however, to provide a navigation solution during steady level flight if GPS is unavailable.

1.4 Organization of the Thesis

This thesis is organized as follows: In Chapter 2, analysis about dynamics of air vehicle is introduced. Chapter 3 presents runway detector algorithm which are used to detect position of UAV and output of applied methods, which is named as Visual Positioning System (VPS) data is described. Chapter 4 mentioned about design of landing autopilot controller. Details of inner loop which aims to control the faster rotational dynamics and outer loop controller which works for guidance are introduced. Outer loop is designed to generate references to inner loop to execute waypoints in correct order and capable to work with GPS or VPS data. In Chapter 5, simulation results are presented with different wind scenarios and different initial condition of the UAV. Chapter 6 presents conclusions and future work recommendations. Finally, Chapter A describes the GCS abilities, communication with the UAV model, detailed message structures, purpose of these messages, and generation of waypoints.

CHAPTER 2

AIRCRAFT DYNAMICS

2.1 Platform

Figure 2-1 de Havilland Canada DHC-2 Beaver

The purpose of this thesis is to design an auto landing system that can operate with position information obtained from image processing for de Havilland Canada DHC-2 Beaver as can be seen in Figure 2-1. Beaver is a high wing, a single-engined propeller driven aircraft developed by de Havilland Canada. Braver has a wide range usage such as cargo and passenger transportation, crop dusting, and adopted by armed forces as a utility aircraft. General aircraft data of the Beaver is presented in Table 2-1.

Specification Name	Value
Manufacturer	The **De** Havilland Aircraft of Canada Ltd.
Serial no.	**1244**
Type	Single engine, high-wing, seven seat, all metal aircraft.
Wing span (b)	**14.63** m
Wing area	**23.23** m2
Mean aerodynamic chord (F)	**1.5875** m
Wing sweep	**0"**
Wing dihedral	**1"**
Wing profile	NACA **64** A **416**
Fuselage length	**9.22** m
Max. take-off weight	**2315 kgf** = **22800** N
Empty weight	**1520** kgf = **14970** N
Engine	Pratt and Whitney Wasp Jr. **R-985**
Max. power	**450** Hp at **n** = **2300** RPM,***p,*** = **26"Hg**
Airscrew	Hamilton Standard, two-bladed metal regulator propeller
Diameter of propeller	**2.59** m
Total contents of fuel tanks	**521 1**
Fuselage front tank	**131** ***1***
Fuselage center tank	**131 1**
Fuselage rear tank	**95** ***1***
Wing tiptanks	**2 x 8 2 1**
Most forward admissible c.g. position	**17.36%**c a t **1725** kgf = **16989** N 29.92%c a t 2315 kgf = 22800 N
Most backward admissible c.g.position	**40.24%c**

Table 2-1 General Aircraft Data of the Beaver [4]

2.2 Aerodynamic Model

The aerodynamic model expresses the aerodynamic force and moment coefficients in terms of non-linear polynomial functions of the state variables and aerodynamic control inputs. The aerodynamic model of the 'Beaver' can be written in terms of dimensionless body-axes force and moment coefficients [25]:

Parameters	Value	Unit
$X_{c.g.}$	0.5996	m
$Y_{c.g.}$	0	m
$Z_{c.g.}$	-0.8851	m
I_x	5368.39	kgm^2
I_Y	6928.93	kgm^2
I_z	11158.75	kgm^2
J_{xy}	0	kgm^2
J_{xz}	117.64	kgm^2
J_{yz}	0	kgm^2
m	2288.231	kg
h	1828.8	m
P	1.024	kg/m^3

Table 2-2 Aircraft data on which the aerodynamic model is based [4]

$$C_D = C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_{\alpha^3}}\alpha^3 + C_{X_q}\frac{q\bar{c}}{V} + C_{X_{\delta_r}}\delta_r + C_{X_{\delta_f}}\delta_f + C_{X_{\alpha\delta_f}}\alpha\delta_f$$

$$C_S = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r + C_{Y_{\alpha\delta_r}}\alpha\delta_r$$

$$C_L = C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\alpha^3}}\alpha^3 + C_{Z_q}\frac{q\bar{\bar{c}}}{V} + C_{Z_{\delta_e}}\delta_e + C_{X_{\delta_e\beta^2}}\delta_e\beta^2 + C_{Z_{\delta_f}}\delta_f + C_{Z_{\alpha\delta_f}}\alpha\delta_f$$

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r + C_{l_{\alpha\delta_a}}\alpha\delta_a$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\alpha^2}}\alpha^2 + C_{m_q}\frac{q\bar{c}}{V} + C_{m_{\delta_e}}\delta_e + C_{m_{\beta^2}}\beta^2 + C_{m_r}\frac{rb}{2V} + C_{m_{\delta_f}}\delta_f$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{n_q}\frac{q\bar{c}}{V} + C_{n_{\beta^3}}\beta^3$$

The subscripts a, e, r of the δ terms denote the aileron, elevator and rudder deflections, respectively. The dimensional force and moment coefficients can be made non-dimensional using the following equations:

- $X_a = C_{X_a} * \frac{1}{2}pV^2S$
- $Y_a = C_{Y_a} * \frac{1}{2}pV^2S$
- $Z_a = C_{Z_a} * \frac{1}{2}pV^2S$

and:

- $L_a = C_{l_a} * \frac{1}{2}pV^2Sb$
- $M_a = C_{Y_a} * \frac{1}{2}pV^2S\overline{c}$
- $N_a = C_{Z_a} * \frac{1}{2}pV^2S\mathrm{b}$

C_x		C_y		C_z	
Parameter	Value	Parameter	Value	Parameter	Value
0	-0.03554	0	-0.002226	0	-0.05504
$\propto$	0.00292	β	-0.7678	$\propto$	-5.578
$\propto^2$	5.459	$\frac{pb}{2V}$	-0.124	$\propto^3$	3.442
$\propto^3$	-5.162	$\frac{rb}{2V}$	0.3666	$\frac{q\bar{c}}{V}$	-2.988
$\frac{q\bar{c}}{V}$	-0.6748	δ_α	-0.02956	δ_e	-0.398
δ_r	0.03412	δ_r	0.1158	$\delta_e\beta^2$	-15.93
δ_f	-0.09447	$\delta_r\alpha$	0.5238	δ_f	-1.377
$\propto \delta_f$	1.10600	$\frac{\dot{\beta}b}{2V}$	0.3666	$\propto \delta_f$	-1.261

Table 2-3 Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-rang [4]

C_l		C_m		C_n	
Parameter	Value	Parameter	Value	Parameter	Value
0	5.91e-04	0	0.09448	0	-0.003117
β	-0.06180	$\propto$	-0.6028	β	0.006719
$\frac{pb}{2V}$	-0.50450	$\propto^2$	-2.140	$\frac{pb}{2V}$	-0.1585
$\frac{rb}{2V}$	0.16950	$\frac{q\bar{c}}{V}$	-15.560	$\frac{rb}{2V}$	-0.1112
δ_α	-0.09917	δ_e	-1.921	δ_α	-0.003026
δ_r	0.006934	β^2	0.6921	δ_r	-0.08265
$\delta_\alpha\alpha$	-0.08269	$\frac{rb}{2V}$	-0.3118	$\frac{q\bar{c}}{V}$	0.1595
		δ_f	0.4072	β^3	0.1373

Table 2-4 Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-rang [4]

2.3 Reference Frames

In the Beaver model, equations of the motion are written in body axis frame and this frame rotates with the vehicle and its origin is at the center of mass of the aircraft. The aircraft velocity and the angular rates are represent in this axis. Aircraft x, y and z positions are represented in inertial axis. Inertial reference axes are represented by X, Y and Z. X axis directed north, Y axis directed east and z axis points downwards to earth's center. The inertial reference frame, which is assumed to be fixed to the ground, is the non-accelerating, non-rotating frame. Also earth is assumed to be non-rotating and flat. Earth and body reference frames are presented in Figure 2-2.

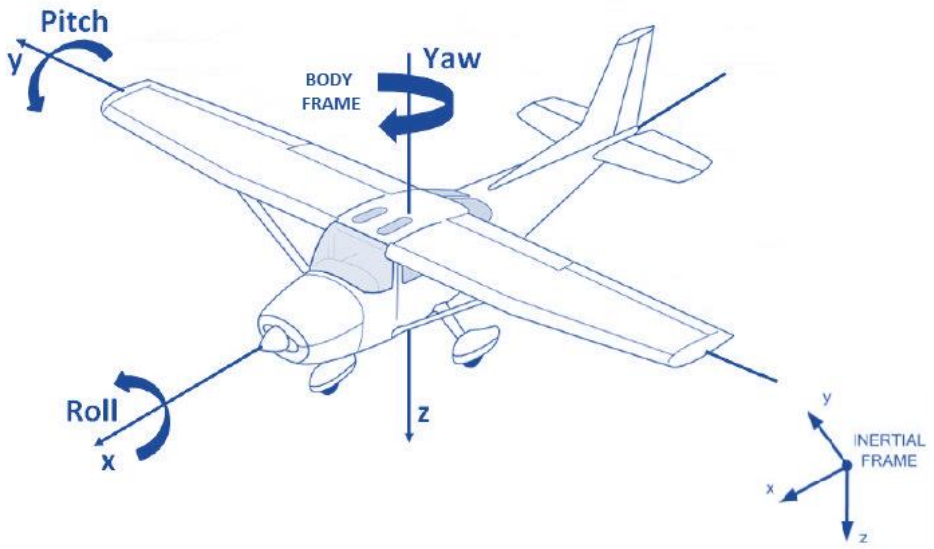


Figure 2-2 Earth and Body Axes

2.4 Nonlinear Equations of Motion

By using Newtonian mechanics, the aircraft equations of motion can be derived. Due to the fact that mass and moment of inertia changes more slowly when compared with the translational and rotational velocities, during the derivation of translational and rotational dynamic equations of motion, mass and inertia terms may be taken as constant.

Following equality is used for derivation of translational dynamic equations,

$$F = m\left(\frac{\partial V}{\partial t} + \ \Omega \, x \, V \ \right)$$

$F = \left[F_x \, F_y \, F_z\right]^T$ is total force that consist of aerodynamic forces, gravity forces, engine trust, ground reaction and wind disturbances. $V = [u \, v \, w]^T$ is total aircraft velocity vector. $\Omega = [p \, q \, r]^T$ is the angular velocity vector about the center of gravity.

Similarly, rotational dynamic equation can be derived by,

$$M = \frac{\partial (I.\Omega)}{\partial t} + \ \Omega \, x \, (I.\Omega)$$

$M = [L \, M \, N]^T$ is sum of all moments vector consist of engine, landing gear. I is the inertia which is defined as

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

By using translational and rotational dynamics,

$$u_e = F_X - qw_e + rv_e$$

$$v_e = \frac{F_y}{m} + pw_e - ru_e$$

$$w_e = \frac{F_z}{m} - pv_e + qu_e$$

$$\dot{p} = P_{pp}p^2 + P_{pq}pq + P_{pr}pr + P_{qq}q^2 + P_{qr}qr + P_{rr}r^2 + P_l L + P_m M + P_n N$$

$$\dot{q} = Q_{pp}p^2 + Q_{pq}pq + Q_{pr}pr + Q_{qq}q^2 + Q_{qr}qr + Q_{rr}r^2 + Q_l L + Q_m M + Q_n N$$

$$\dot{r} = R_{pp}p^2 + R_{pq}pq + R_{pr}pr + R_{qq}q^2 + R_{qr}qr + R_{rr}r^2 + R_l L + R_m M + R_n N$$

Where P_{pp}, P_{pq}, … R_n are inertial coefficients derived from matrix multiplications which have been given in Figure 2-3.

symbol	definition
$\|I\|$	$I_{xx}I_{yy}I_{zz} - 2J_{xy}J_{xz}J_{yz} - I_{xx}{J_{yz}}^2 - I_{yy}{J_{xz}}^2 - I_{zz}{J_{xy}}^2$
I_1	$I_{yy}I_{zz} - {J_{yz}}^2$
I_2	$J_{xy}I_{zz} + J_{yz}J_{xz}$
I_3	$J_{xy}J_{yz} + I_{yy}J_{xz}$
I_4	$I_{xx}I_{zz} - {J_{xz}}^2$
I_5	$I_{xx}J_{yz} + J_{xy}J_{xz}$
I_6	$I_{xx}I_{yy} - {J_{xy}}^2$
P_l	$I_1 \ / \ \|I\|$
P_m	$I_2 \ / \ \|I\|$
P_n	$I_3 \ / \ \|I\|$
P_{pp}	$-(J_{xz}I_2 - J_{xy}I_3) \ / \ \|I\|$
P_{pq}	$(J_{xz}I_1 - J_{yz}I_2 - (I_{yy} - I_{xx})I_3) \ / \ \|I\|$
P_{pr}	$-(J_{xy}I_1 + (I_{xx} - I_{zz})I_2 - J_{yz}I_3) \ / \ \|I\|$
P_{qq}	$(J_{yz}I_1 - J_{xy}I_3) \ / \ \|I\|$
P_{qr}	$-((I_{zz} - I_{yy})I_1 - J_{xy}I_2 + J_{xz}I_3) \ / \ \|I\|$
P_{rr}	$-(J_{yz}I_1 - J_{xz}I_2) \ / \ \|I\|$
Q_l	$I_2 \ / \ \|I\|$
Q_m	$I_4 \ / \ \|I\|$
Q_n	$I_5 \ / \ \|I\|$
Q_{pp}	$-(J_{xz}I_4 - J_{xy}I_5) \ / \ \|I\|$
Q_{pq}	$(J_{xz}I_2 - J_{yz}I_4 - (I_{yy} - I_{xx})I_5) \ / \ \|I\|$
Q_{pr}	$-(J_{xy}I_2 + (I_{xx} - I_{zz})I_4 - J_{yz}I_5) \ / \ \|I\|$
Q_{qq}	$(J_{yz}I_2 - J_{xy}I_5) \ / \ \|I\|$
Q_{qr}	$-((I_{zz} - I_{yy})I_2 - J_{xy}I_4 + J_{xz}I_5) \ / \ \|I\|$
Q_{rr}	$-(J_{yz}I_2 - J_{xz}I_4) \ / \ \|I\|$
R_l	$I_3 \ / \ \|I\|$
R_m	$I_5 \ / \ \|I\|$
R_n	$I_6 \ / \ \|I\|$
R_{pp}	$-(J_{xz}I_5 - J_{xy}I_6) \ / \ \|I\|$
R_{pq}	$(J_{xz}I_3 - J_{yz}I_5 - (I_{yy} - I_{xx})I_6) \ / \ \|I\|$
R_{pr}	$-(J_{xy}I_3 + (I_{xx} - I_{zz})I_5 - J_{yz}I_6) \ / \ \|I\|$
R_{qq}	$(J_{yz}I_3 - J_{xy}I_6) \ / \ \|I\|$
R_{qr}	$-((I_{zz} - I_{yy})I_3 - J_{xy}I_5 + J_{xz}I_6) \ / \ \|I\|$
R_{rr}	$-(J_{yz}I_3 - J_{xz}I_5) \ / \ \|I\|$

Figure 2-3 Definition of Inertia Coefficients [25]

2.5 Linear Equations of Motion and Analysis

To be able to analyze the stability characteristics of the A/C, the equations of motion are linearized around the trim values. This information is also used in the controller design. Detailed information about the linearization procedure is given in [4]. Trim values of the Beaver is calculated for airspeed V = 45 m/s, altitude h=1828 m above sea level.

Variable:	Value at t = 0 sec:	
V	45	m/s
α	0.1444	rad
β	-0.0147	rad
p	0	rad/s
q	0	rad/s
r	0	rad/s
ψ	0	rad
θ	0.1444	rad
φ	0	rad
x_e	0	m
y_e	0	m
H	6000 ft = 1828.8 m	
δ_e	-0.0425	rad
δ_a	0.0091	rad
δ_r	-0.0460	rad
δ_f	0	rad
p_z	21.072	"Hg
n	1800	RPM

Figure 2-4 Initial conditions for V = 45 m/s, H = 6000 ft obtained with the trim routine ACTRIM command [4].

In modeling and simulation of the aircraft, the state space representation is used for making analyzes.

- $\dot{x} = Ax + Bu$
- $y = Cx + Du$

2.5.1 Longitudinal Dynamics

The longitudinal states are,

- $x = [V\ \alpha\ q\ \theta]^T$

- $A_{long} = \begin{bmatrix} -3.8930 * 10^{-2} & 5.4535 & -4.0758.10^{-1} & -9.800 \\ -8.3760 * 10^{-3} & -1.285 & 9.7640 * 10^{-1} & -1.0893 * 10^{-6} \\ 1.3905 * 10^{-2} & -6.7369 & -3.0292 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

The longitudinal motion has two oscillatory modes, phugoid and short period mode. Information of this modes can be obtained from eigenvalues of the A_{long} matrix.

	Root Location	**Natural Frequency ω_n(rad/s)**	**Damping Ratio**	**Time to Half Amplitude(s)**	**Period (s)**
Short Period	-2.160 ± 2.4112i	2.411	0.667	0.3	2.60
Phugoid	-0.01614 ± 0.2631i	0.2631	0.0612	42.94	23.8

Table 2-5 Longitudinal mode characteristics

As can be seen in Table 2-5, both short period and phugoid mode are stable because real parts of their roots are negative. The short period mode has a higher damping ratio therefore, it has a shorter time to half and period which are 0.3s and 2.6s,

respectively. Roots of the phugoid mode is closer to the origin so, as expected, longer half value and period exist with respect to short period mode 42.94s and 23.8, respectively.

2.5.2 Lateral Dynamics

The lateral states are,

- $x = [\beta\ p\ r\ \varphi]^T$
- $A_{lat} = \begin{bmatrix} -1.8068 * 10^{-1} & 1.4002 * 10^{-1} & -9.8159 * 10^{-1} & 2.1682 * 10^{-1} \\ -4.0528 & -5.4022 & 1.7965 & 0 \\ 1.7227 * 10^{-1} & -8.7058 * 10^{-1} & -5.5189 * 10^{-1} & 0 \\ 0 & 1 & 1.4540 * 10^{-1} & 0 \end{bmatrix}$

Eigenvalues of the A_{lat} matrix contain information about lateral modes named as roll mode, spiral mode and Dutch-roll mode.

	Root Location	**Natural Frequency ω_n(rad/s)**	**Damping Ratio**	**Time to Half Amplitude(s)**	**Period (s)**
Dutch-roll	-0.477 ± 0.9763i	0.9763	0.439	1.453	1.0866
Roll	-5.1351	-	-	0.135	-
Spiral	-0.04573	-	-	15.15	

Table 2-6 Lateral Mode Characteristics

As can be seen in Table 2-6, Dutch-roll mode has relatively large damping and the spiral mode is stable since the real parts of their roots are negative.

CHAPTER 3

RUNWAY DETECTOR ALGORITHM

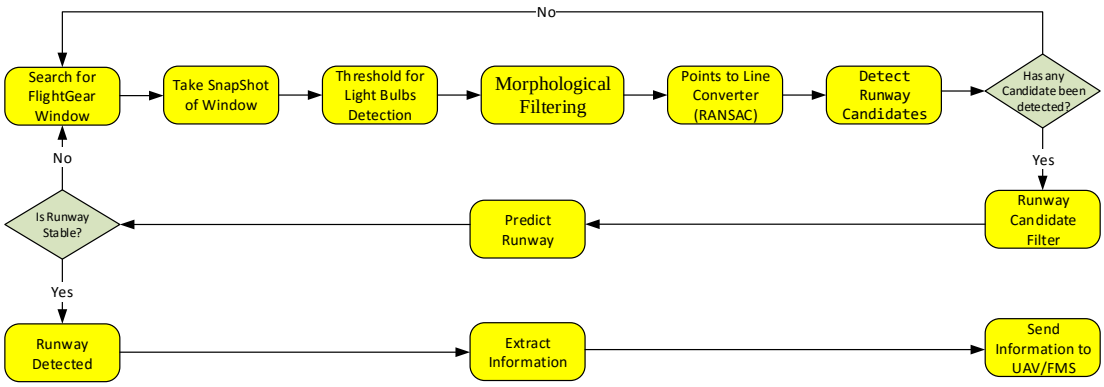


Figure 3-1 Runway Detection Algorithm

In this chapter, runway detection algorithm will be presented. To calculate position of the aircraft with respect to a known position, image processing methods are applied with C++ language. OpenCV computer vision library, which is free and open source, is used to implement image processing methods in C++. Besides explanation of used image processing methods, information obtained from these methods are filtered to obtain durable results. Reasons and results of filtering methods, decision mechanisms, and information extraction from result of overall process will be presented. General way of working of the runway detection algorithm can be investigated in Figure 3-1.

3.1 Image Capturing and Light Bulbs Detection

3.1.1 Capturing FlightGear Window

Image processing application searches FlightGear simulator window on the screen by using Windows APIs (Application Program Interface). If FlightGear window is found, image bitmap of that window is captured to process. An instance of FlightGear screen and captured image can be seen in Figure 3-2 and Figure 3-3. To be eliminate header of FlightGear window and FlightGear menu on Figure 3-2, cropping is done on captured image to select only simulator output bitmaps.

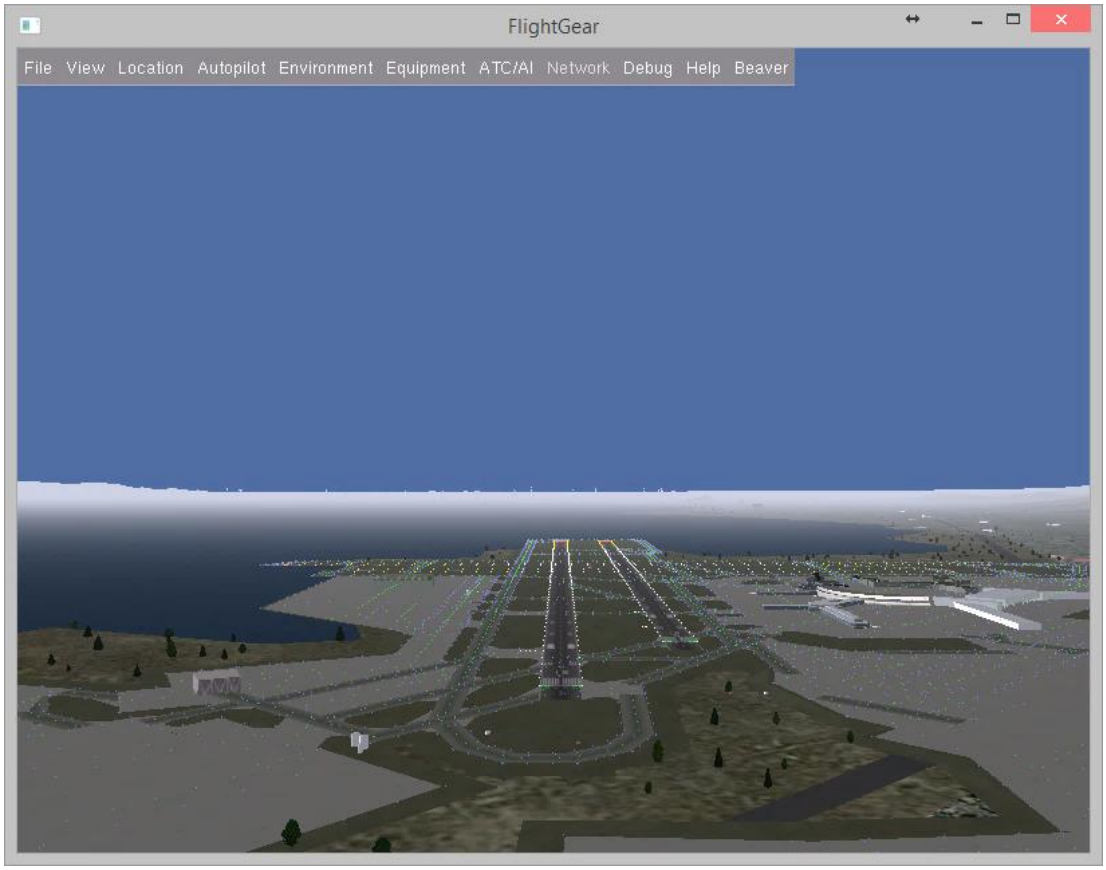


Figure 3-2 FlightGear Simulator Interface

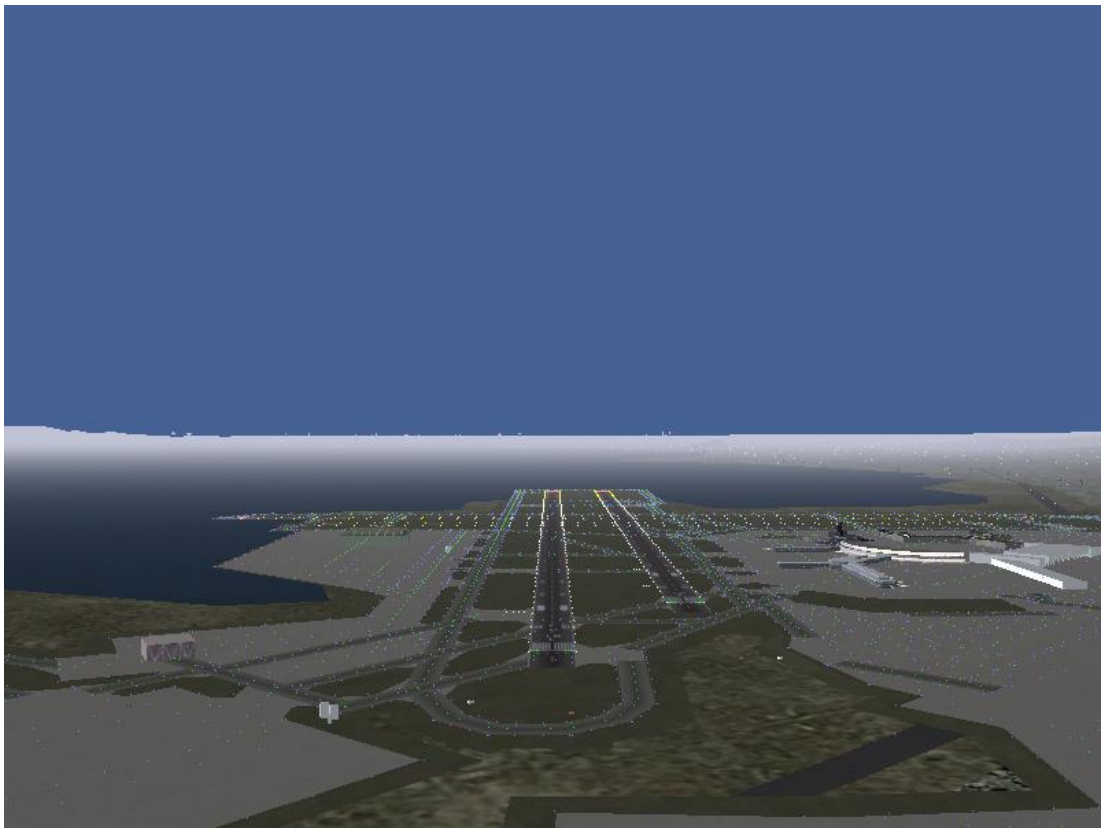


Figure 3-3 Captured Image from Flightgear Screen

3.1.2 Thresholding Method

Main purpose of image processing is to detect runway. As can be seen in Figure 3-3, runway has light bulbs on the left and right side. These light bulbs give valuable information about borders of the runway so by detecting of these bulbs, runway can be detected. Thresholding is the simplest segmentation method. Thresholding method needs a gray scaled one dimensioned image which can be easily supplied in Figure 3-3. Captured FlightGear image is split to its red, green, and blue components and thresholding is applied to these images. Processed images are 8-bits/pixels images and notation of them as follows:

- Black pixel: in grayscale values for an 8 bits/pixel indexed image and its value is 0,
- White pixel: in grayscale values for an 8 bits/pixel indexed image and its value is 255.

Threshold method obtains pixel values and if a pixel value is greater than selected threshold value, this pixel value is assigned to 255 (White), else it is assigned to 0 value (Black). For designed project, this threshold value is selected as 185 over 255 (8-bit). After three thresholding operation, three image again merged to get full color RGB image. Red, green and blue images after thresholding can be seen in Figure 3-4 respectively.

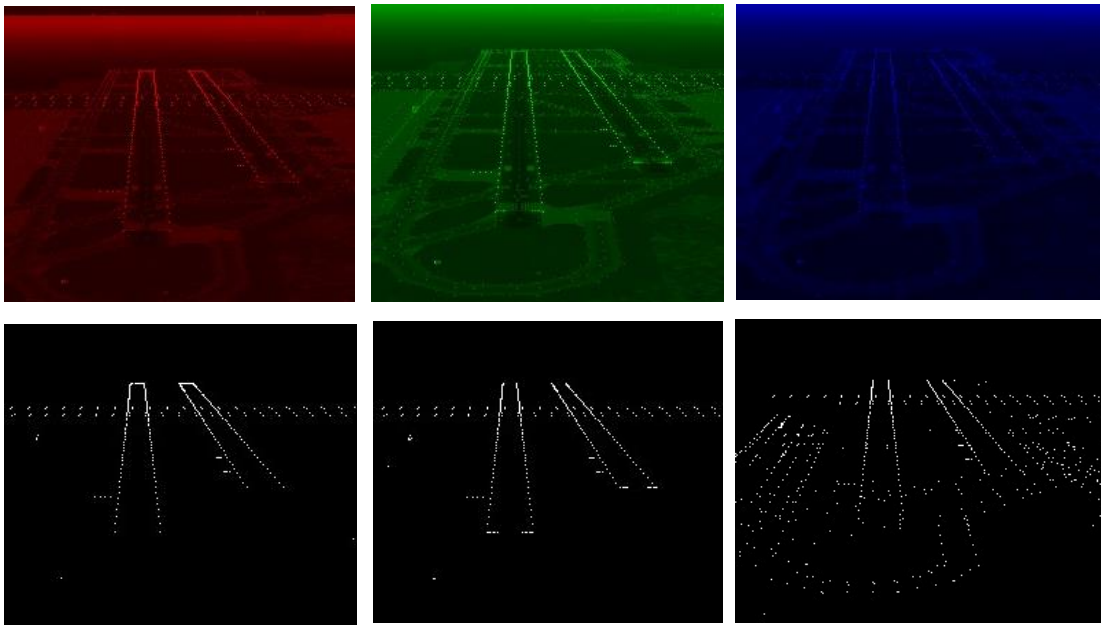


Figure 3-4 Threshold Result of Red, Green, and Blue Pixels for Light Bulbs Detection

Main runways have white bulbs but auxiliary runways have different colored light bulbs. So bulbs without white color must be eliminated in Figure 3-3. White pixels have values [255 255 255] respectively red, green and blue value. To obtain white pixels, following operation is done and output is as in Figure 3-5.

$$WhitePix = RedPixels \,\&\, GreenPixels \,\&\, BluePixels$$

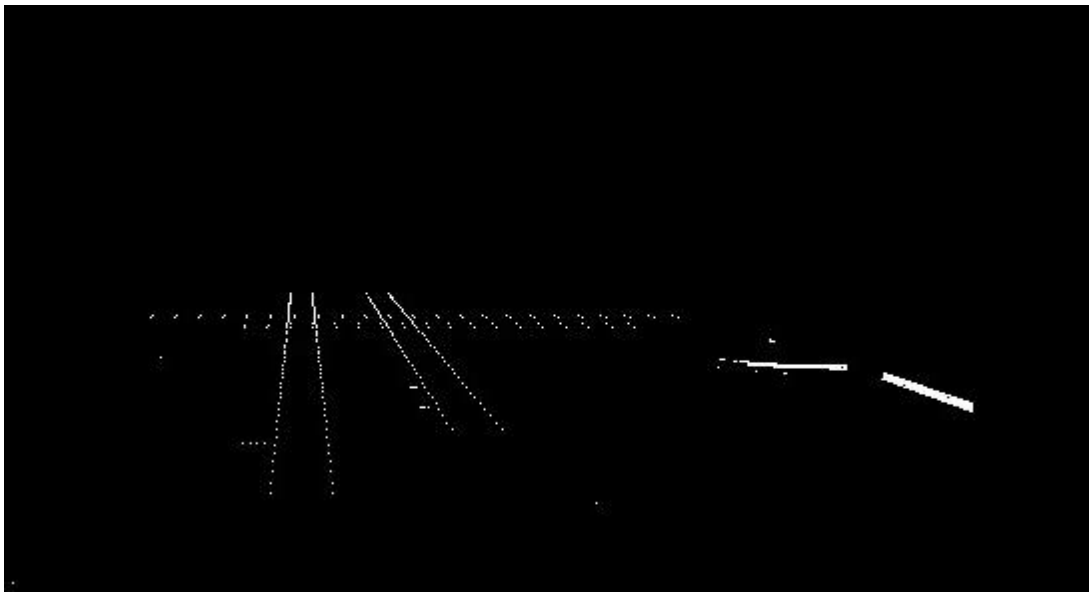


Figure 3-5 All White Pixels

3.1.3 Morphological Method

Thresholding operation outputs all white objects which can be a bulb or a building or a white cloud as in Figure 3-5. However, we need to obtain only white bulbs. So a filtering operation is needed. The common feature of the bulbs is that they are small objects. On the other hand, building, clouds or any other object appears on thresholded image is relatively bigger than a bulb size. So by filtering big objects, many non-interest white object can be eliminated.

To eliminate big objects, many methods can be used. In this thesis, opening morphological method is used. Morphological operation methods offer an attitude to the processing of digital images which is based on shape. Appropriately used, these operations tend to simplify image data preserving their essential shape characteristics and eliminating noises [26].

Dilation and erosion are basic morphological processing operations. Both dilation and erosion are produced by the interaction of a set called a structuring element with a set of pixels of interest in the image. The structuring element has both a shape and an origin. Let A be a set of pixels and let B be a structuring element. Let $\hat{B}$ be the reflection of B about its origin and followed by a shift by s [27].

Dilation, written $A \oplus B$, is the set of all shifts that satisfy the following:

$$A \oplus B = \{s|(\widehat{B})_s \cap A \neq \emptyset\}$$

Equivalently,

$$A \oplus B = \{s|\left((\widehat{B})_s \cap A\right) \subseteq A\}$$

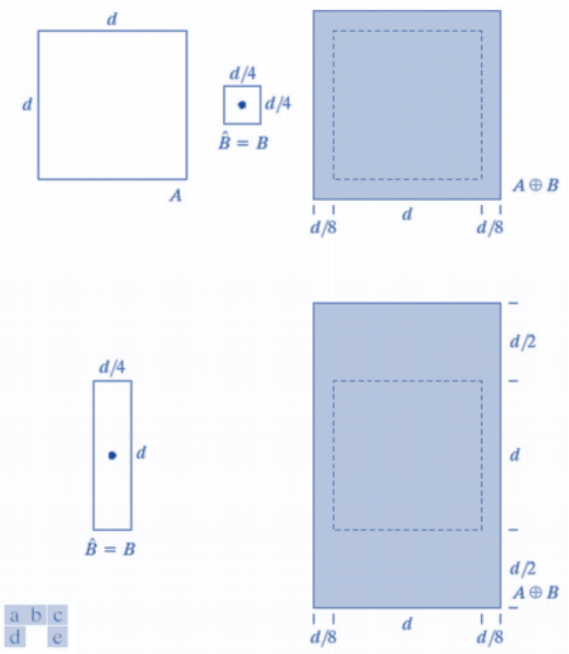


Figure 3-6 Dilation Example: (a) Set A. (b) Square Structuring Element (Dot is the Center), (c) Dilation of A by B, Shown Shaded. (d) Elongated Structuring Element. (e) Dilation of A Using This Element [27]

So dilation increases the region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So after erosion, we dilate it since noise is gone and they won't come back, but our object area increases to nearly its original size. It is also useful in joining broken parts of an object.

The basic idea of erosion is eating away the boundaries of foreground object. So kernel slides through the image as in 2D convolution. A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). So what happens is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises.

$$A \ominus B = \{s|(\widehat{B})s \subseteq A\}$$

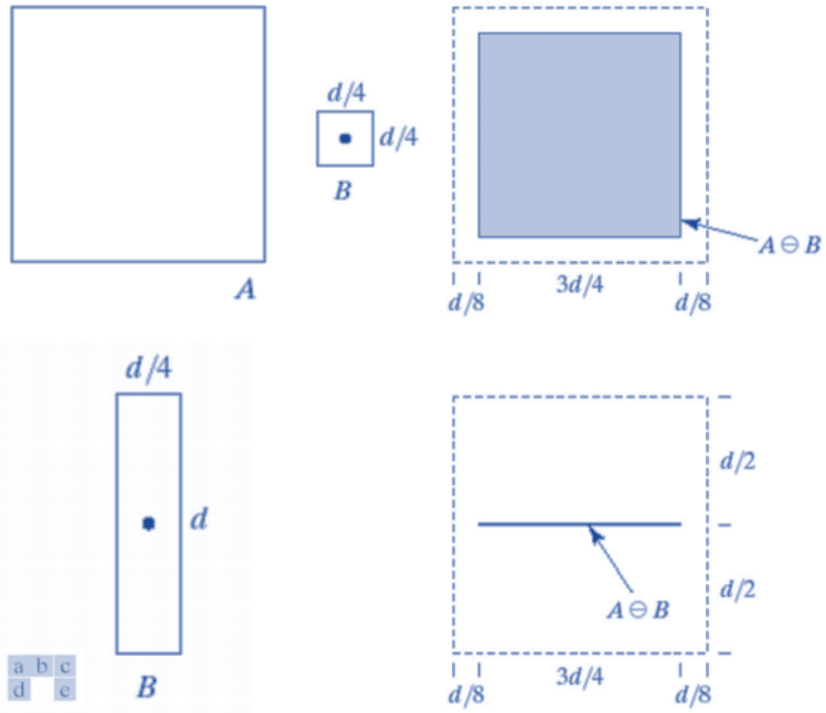


Figure 3-7 Erosion Example: (a) Set A. (b) Square Structuring Element (Dot is the Center). (c) Erosion of A by B Shown Shaded. (d) Elongated Structuring Element. (e) Erosion of A Using This Element [27]

Opening, which is used here, is the dilation of the erosion of a set A by a structuring element B. it removes small objects, makes object smoother.

$$A \circ B = (A \ominus B) \oplus B$$

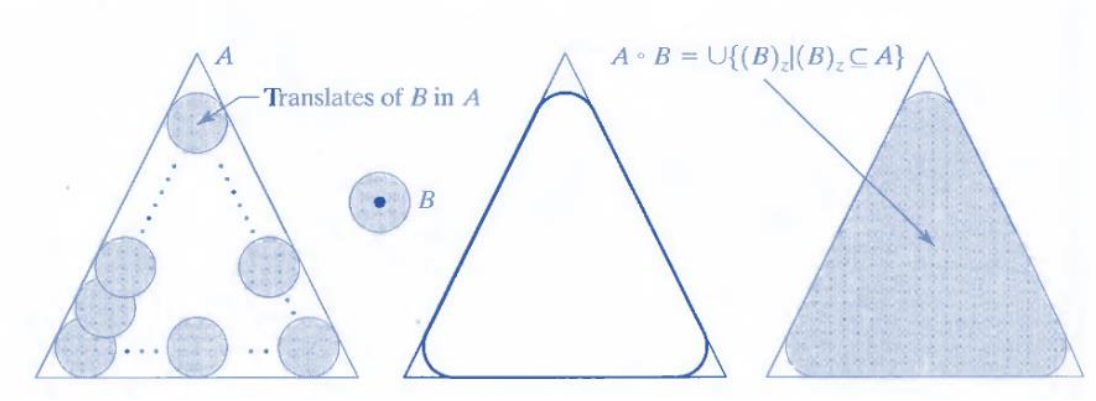


Figure 3-8 Structuring element B rolling along the inner boundary of A (the dot indicates the origin of B). The heavy line is the outer boundary of the opening. Final output is shaded. [27]

Briefly, with opening method, small objects are eliminated such as light bulbs in current situation. So big objects can be detected in the image. Result of related operation can be seen in Figure 3-9.

Figure 3-9 Result of Opening Method

Here, white object out of light bulbs are found. If opening image Figure 3-9 is subtracted from thresholded image Figure 3-5, resulted image gives only small light bulbs as in the figure of Figure 3-10.

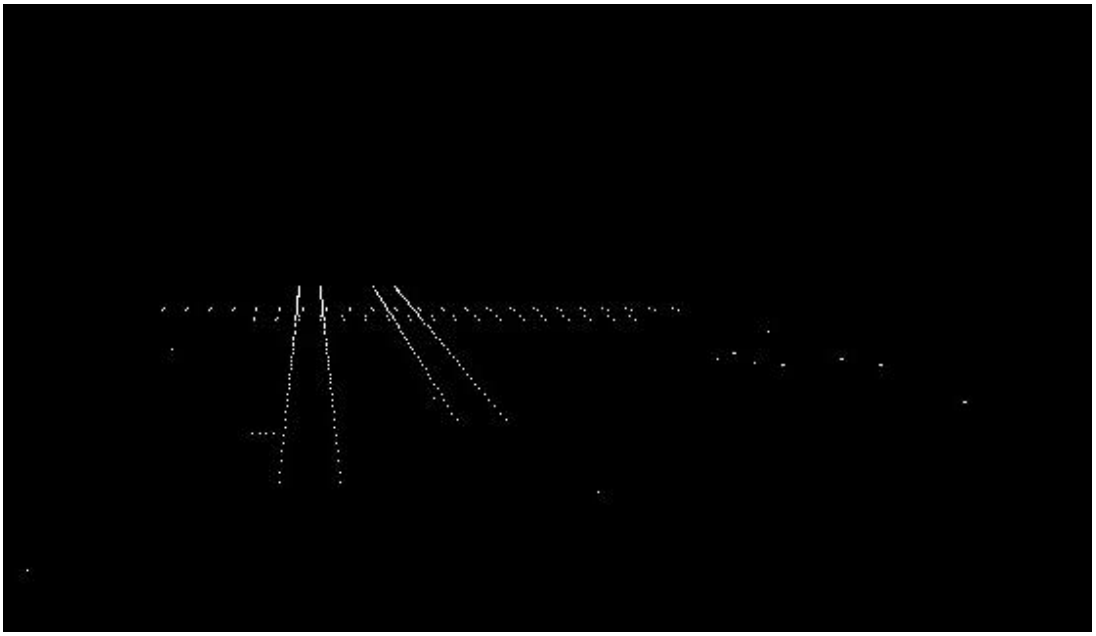


Figure 3-10 Selection of White Bulbs. Original Image Subtracted From Opening Image

With implementation of the explained methods, we start from original captured image Figure 3-3 and obtain image seen in Figure 3-10. Main purpose of combination of the image processing methods is to obtain small light bulb objects but as can be seen in Figure 3-10, some noisy pixels still exist which is belong to big object which are eliminated. These regions are corner and sharp regions of eliminated objects. Although it seems that they break the elimination logic, however, as will be described in the next sections, important subject is that they are not in a line order and even if they are in a line order, they do not belong to a runway structure of order which will be investigated also.

3.2 Points to Line Converter (RANSAC)

Bulbs are positioned near the runway edges so by detection of bulbs which are in a line order gives possible runway edges. By using RANdom SAmple Consensus (RANSAC) [28] algorithm, points which can be fitted to a line can be detected. Algorithm can be explained as follows:

- Randomly select a sample of s data points from data set S and instantiate the model from this subset.
- Determine the set of data points S_i which are within a distance threshold T of the model. The set S_i is the consensus set of samples and defines the inliers of S.
- If the subset of Si is greater than some threshold T, reestimate the model using all the points in S_i and terminate.
- If the size of S_i is less than T, select a new subset and repeat the above.
- After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i.

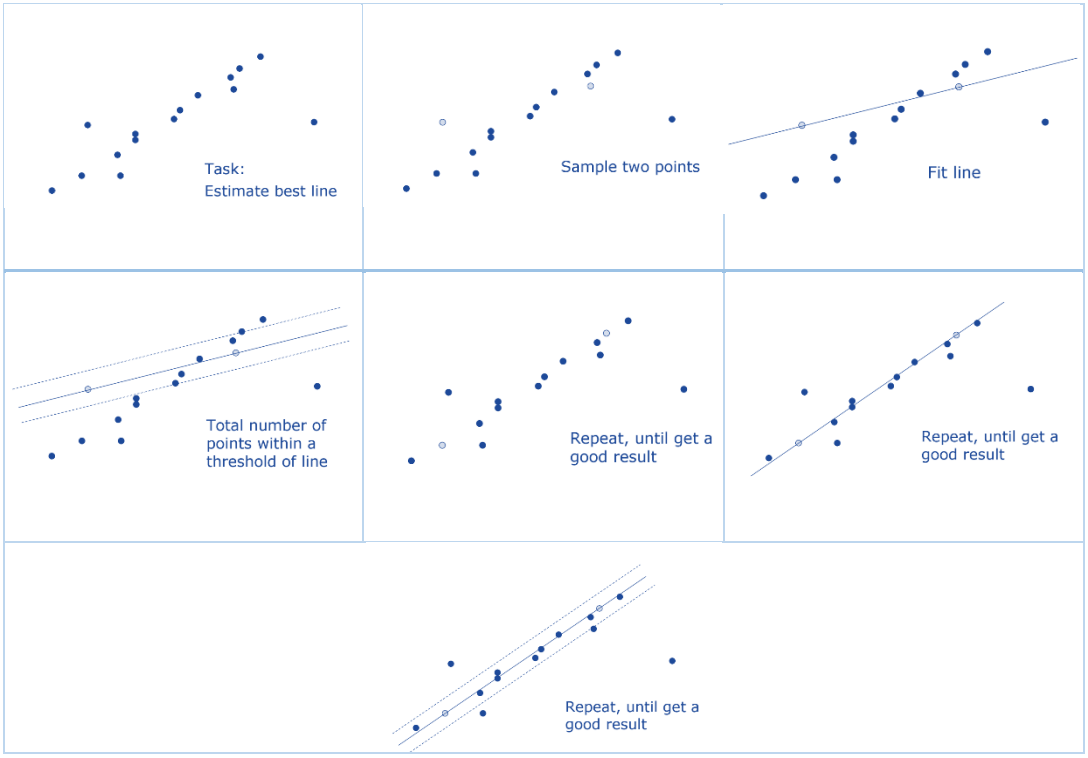


Figure 3-11 RANSAC Algorithm Steps

Here, all pixel values of Figure 3-10 are scanned and coordinates of white pixels stored in a 2-D coordinate vector which is data set to RANSAC algorithm. After RANSAC algorithm is applied, results are filtered such that slope of the detected line must be within [-85 85] degree at which 0 degree would be a vertical line. Reason of this elimination is that there is horizontal runways with white bulbs which are not interested to be found. Filtered results are projected on original colored image that can be seen in Figure 3-12.

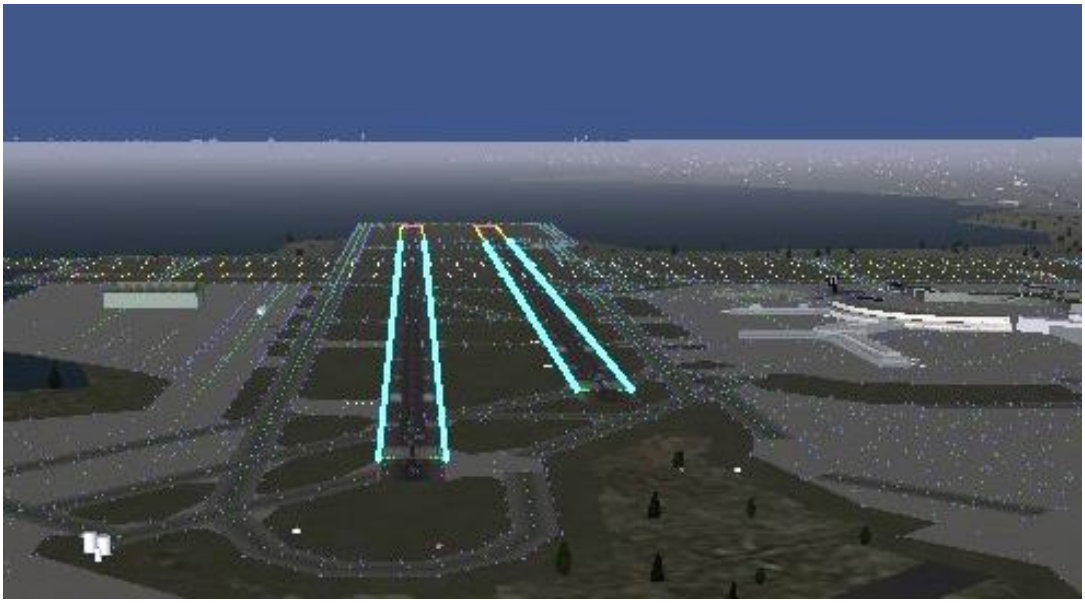


Figure 3-12 Line Detection Algorithm Result

3.3 Determination of Runway Candidates

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. In graphical perspective, parallel lines have an intersection point named as vanishing point as can be investigated in Figure 3-13. From previous section, we obtain lines and coefficients of these lines. Therefore, known coefficients and known point set of these lines, intersection point can be found by using basic linear algebra as follows:

$$A_1x_1 + B_1y_1 + C_1 = A_2x_2 + B_2y_2 + C_2$$

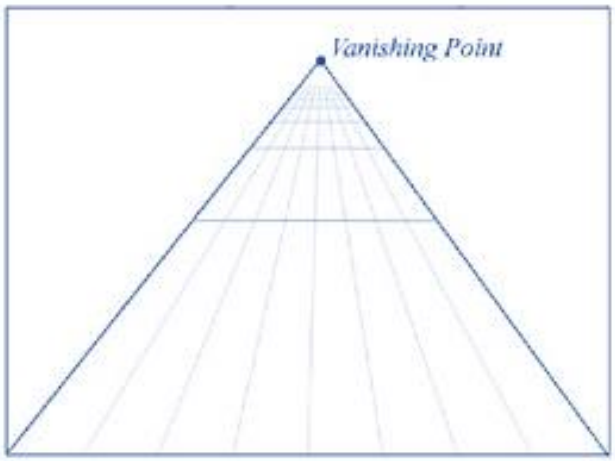


Figure 3-13 Vanishing Point Illustration

If two lines not cross each other, their intersection point has higher coordinate at y-axes than all points belong to these lines. So, after determination of vanishing point and checking of the y-coordinates, lines are marked as parallel or not. If related lines are parallel, a bounding rectangle is created to mark runway candidates and set of information is stored in a vector whose components are as follows:

- Coefficients of two lines belong to that runway candidate,
- Slope of these lines,
- Length of these lines,
- Coordinates of light bulbs belong to lines,
- Bounding rectangle information.

Result of runway candidate determination can be investigated in Figure 3-14. Main purpose of the runway candidate detector algorithm is to obtain line sets which belong to a runway. If no candidate is determined, this means that input line vectors do not belong to a runway but may be part of any other building or earth surface. Under this condition, in other word if no candidate is found, it is unnecessary to continue with rest of the algorithm. Therefore, if no runway candidates are detected, runway detection algorithm send data to autopilot of the UAV after setting validity

variable as false in Table 3-1 Message Table of VISION_DATA. With this flag autopilot will know that received data is not reliable. Action of autopilot to this situation will be expressed in chapter 4. After reporting the fail situation, runway detector algorithm starts over to capture new image from Flightgear for new scene and searches runway candidates again.

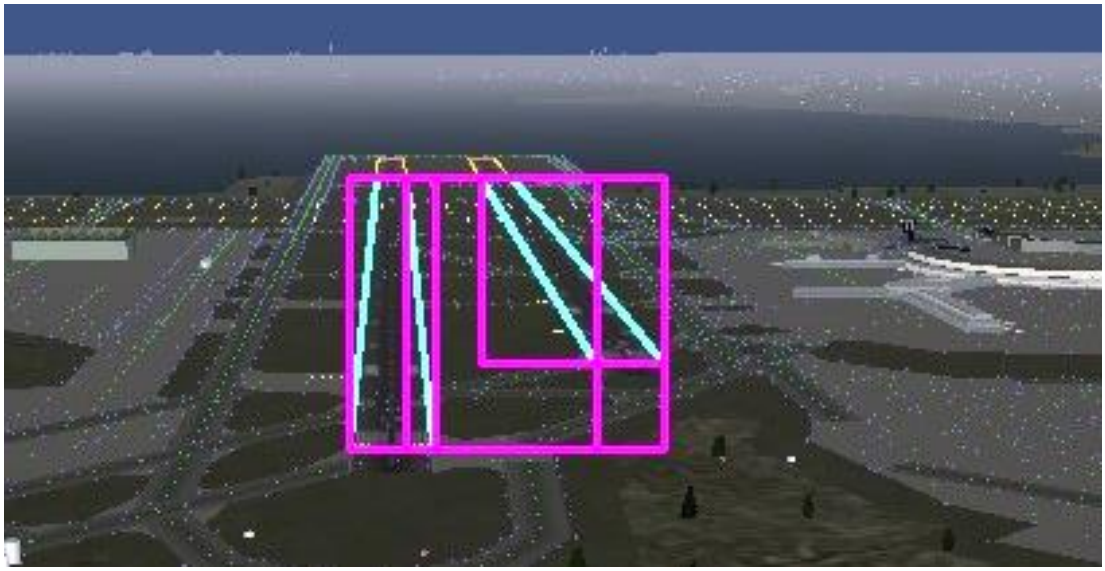


Figure 3-14 Output of Runway Candidates Algorithm. 4 Lines Results with 6 Candidates

3.4 Runway Candidate Filter

From previous section, we have runway candidates but one of them has to be selected as target runway. In the previous section case, detected line number was four so runway candidate number was combination of them out of two groups which is $\binom{4}{2} = 6$. For elimination of runway candidates to find out the real runway, we need a cost function. To design cost function, parameters are selected as follows:

- It is assumed that nose view of UAV is looking to target runway which is aimed to land. So true runway would be close to center of view.
- Length of two edges of the runway must be close to each other. On the other hand.

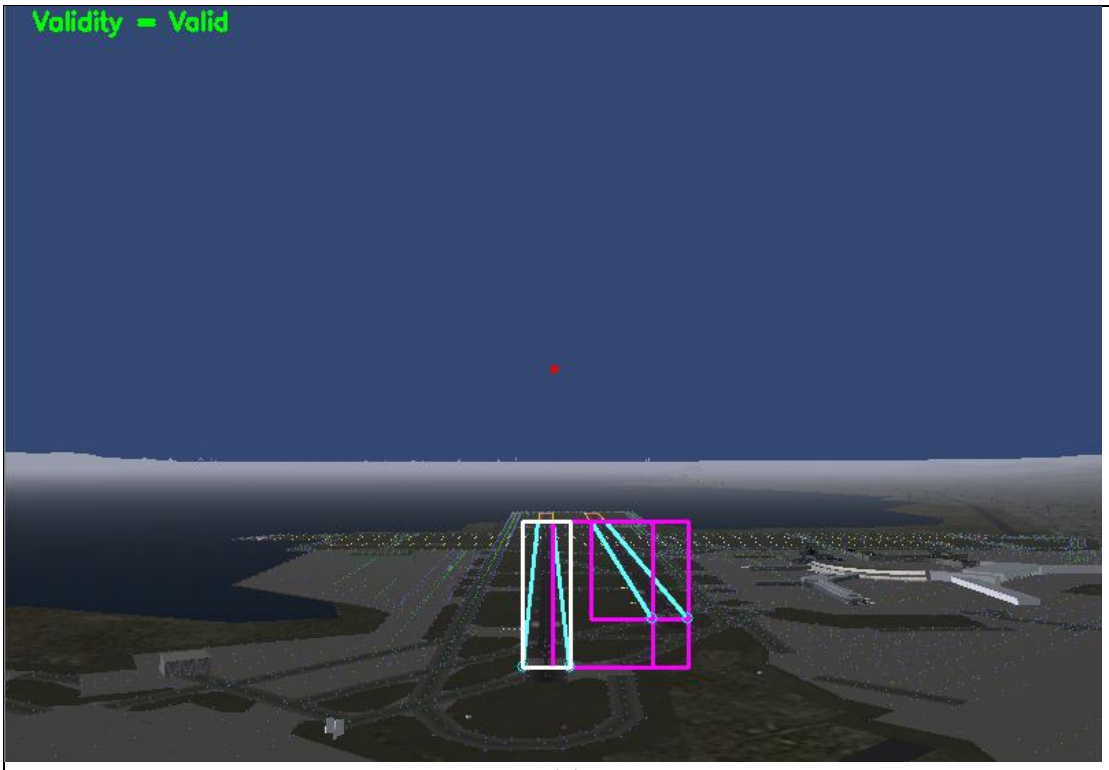
- As can be seen in Figure 3-14 , there is two runways in the image. It is assumed that aim of the UAV is to land on left runway so all runway candidates tagged with an order number starting from left to right.

In the light of foregoing,

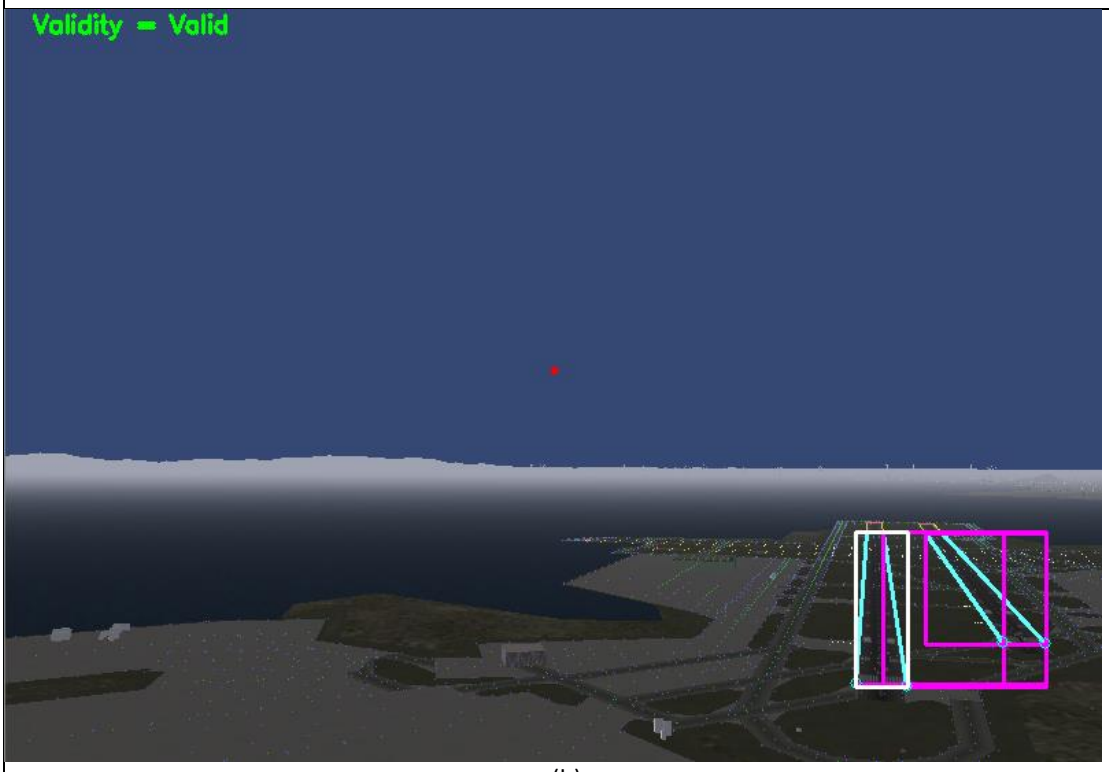
$$Cost\ Function = abs\big(\ Lenght(LeftEdge)\ -\ Lenght(RightEdge)\big) * (-10) + distance(CenterofRunwayCandidate, CenterofwholeImage\ )\ * (-10) + \text{LeftToRight}Order * (-100)$$

From previous section, length of edges, information of bounding rectangle were stored. Also runway candidates are stored in an order such that left most runway candidate is first element and right most runway candidate is last element. Finally, after calculation and ordering of these cost functions, runway which has biggest cost function is selected as true runway. Different instances of this algorithm can be seen in Figure 3-15. White rectangle is selected as target runway to land out of other rectangles.

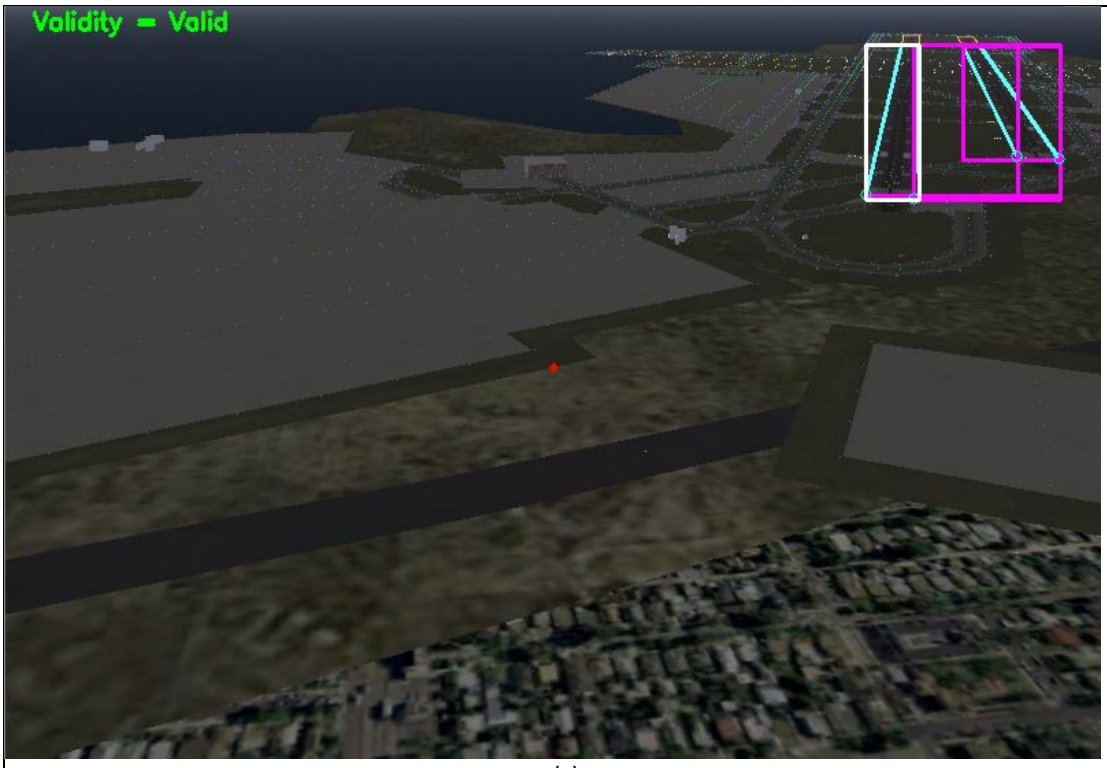
In Figure 3-15, many conditions are created to test reliability of the developed algorithm and it is validated that developed algorithm is functional under different orientation state. For these cases, line difference and left to right ordering of the runways are dominant parameters for the cost function because distance to center of the view is more or the less close to each other for runway candidates. However, since main purpose is landing to target runway, as UAV continues to operate for landing, target runway will move closer to center of the view by time.



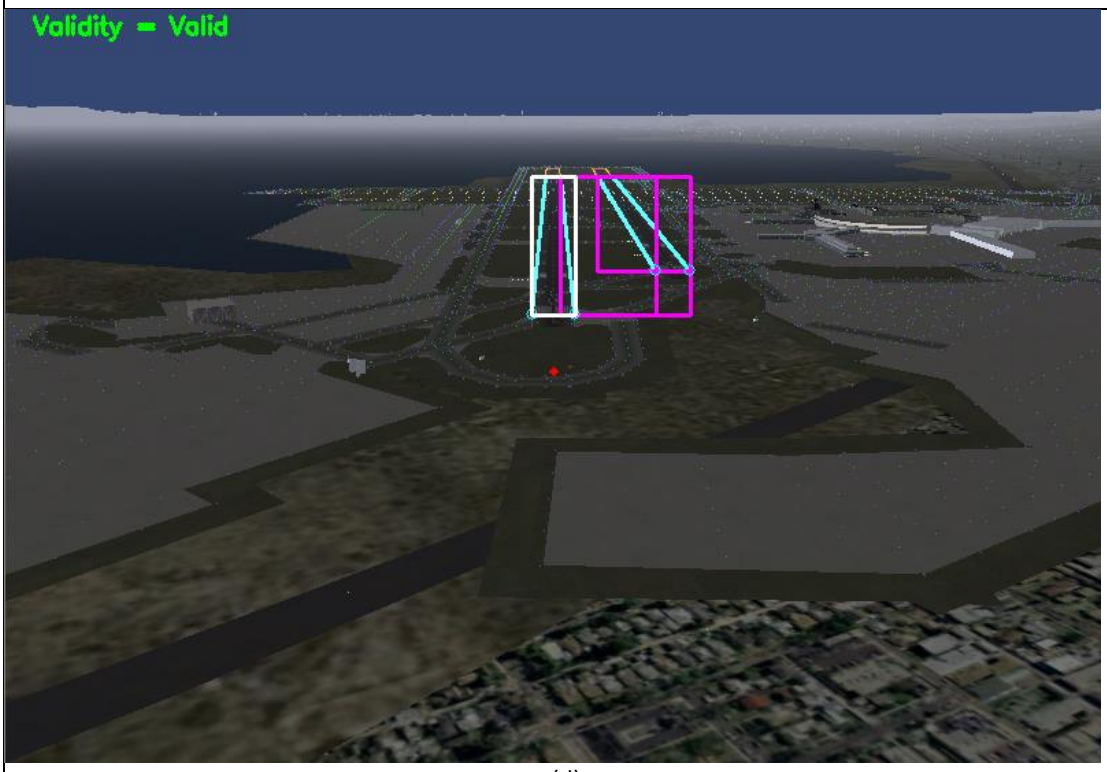
(a)



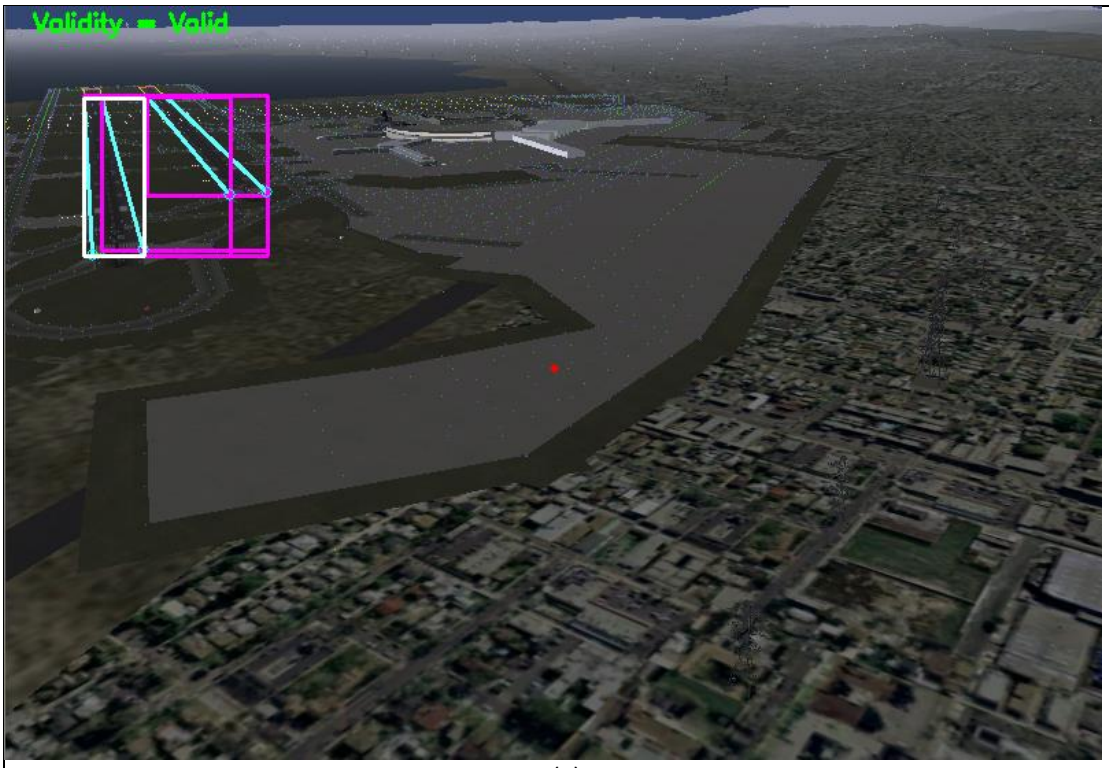
(b)



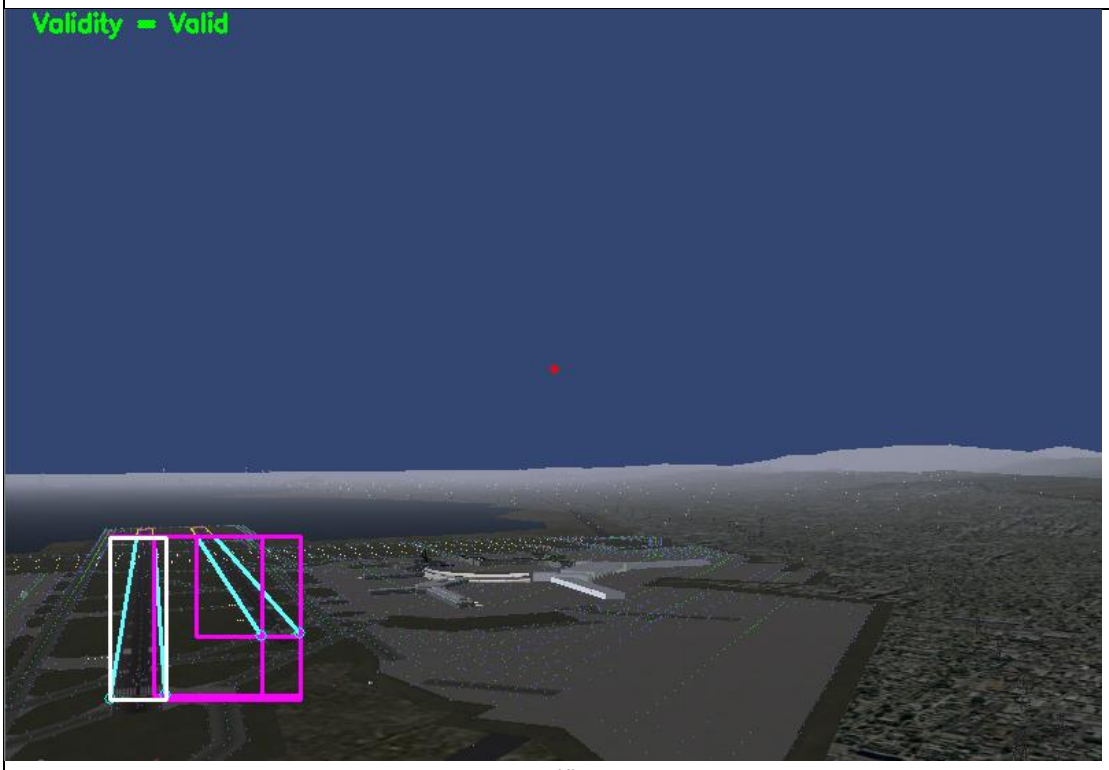
(c)



(d)



(e)



(f)

Figure 3-15 Elimination of Candidate Runways

3.5 Extract Information of Target Runway

After detection of target runway, information about runway are extracted to send landing autopilot of UAV. Also validity of algorithm result and focal length of the camera are sent to UAV which will be used in calculation. Information within VISION_DATA can be listed as follows:

- **Validity:** After determination of target runway as a result of image processing method and filtering mechanisms, Validity variable set to true. If no target runway detected, then it is set to false.
- **FocalLength:** The focal length of the camera is the distance between the lens and the image sensor. We know captured image size and field of view (FOV) parameter of FlightGear. To calculate distance to known sized objects, focal length must be determined. Detail of calculation is presented in Table 3-1.
- **RunwayWidth:** Number of pixels between bottom of the left and right edge of the runway.
- **LeftEdgeSlope:** Angle between left edge of the runway and runway center.
- **RightEdgeSlope:** Angle between right edge of the runway and runway center.

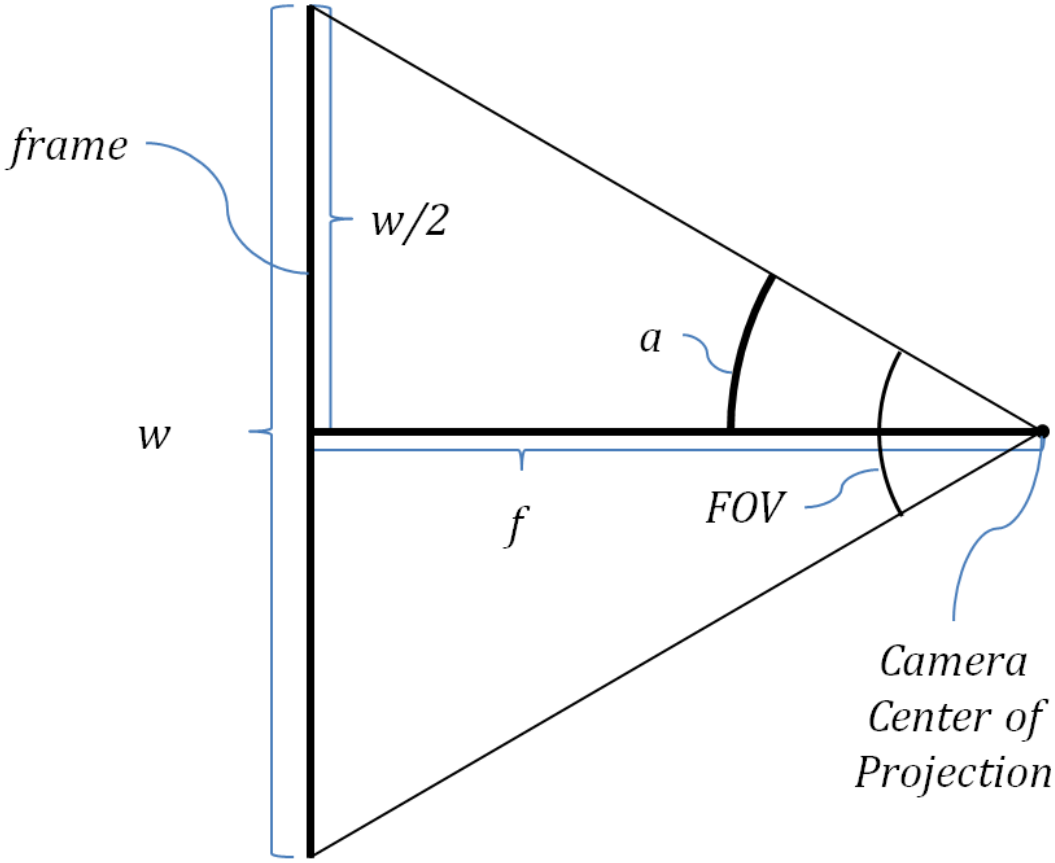


Figure 3-16 Focal Length and FOV Relation

By using right and left edge angles, autopilot determines the course correction value and distance to runway is calculated with the help of runway width and focal length [29] as described in chapter 4.

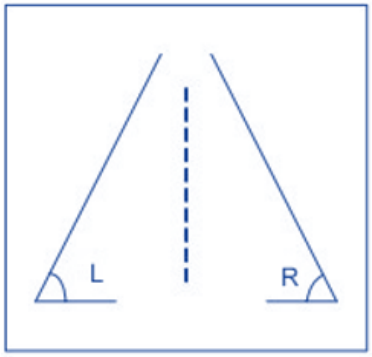


Figure 3-17 Runway Edges

Briefly, detail of VISION_DATA can be seen in Table 3-1.

Name	Type	Unit	Description
Validity	Bool	N/A	
FocalLength	Integer	N/A	$Focal\ Lenght = \frac{\left(\frac{\text{Image. cols}}{2}\right)}{tan\left(\frac{\text{FOV}}{2} * \frac{\text{PI}}{180}\right)}$
RunwayWidth	Integer	Pixel	Width of bottom of the target runway
LeftEdgeSlope	Float	degree	Slope of left edge of the target runway
RightEngeSlope	Float	degree	Slope of right edge of the target runway

Table 3-1 Message Table of VISION_DATA

CHAPTER 4

LANDING AUTOPILOT

Hold autopilot is designed in two steps: inner and outer loop controller is design to control dynamics of the UAV. The inner loop is designed to generate lateral controller commands for yaw and roll angle of the UAV and longitudinal controller commands for pitch angle and airspeed of the UAV. Rudder is used for yaw angle control and ailerons are used for roll (bank) angle of the UAV by the lateral controller. Longitudinal controller uses the elevator and engine thrust to control pitch angle and airspeed of the UAV. It is assumed that sensor data have no noise.

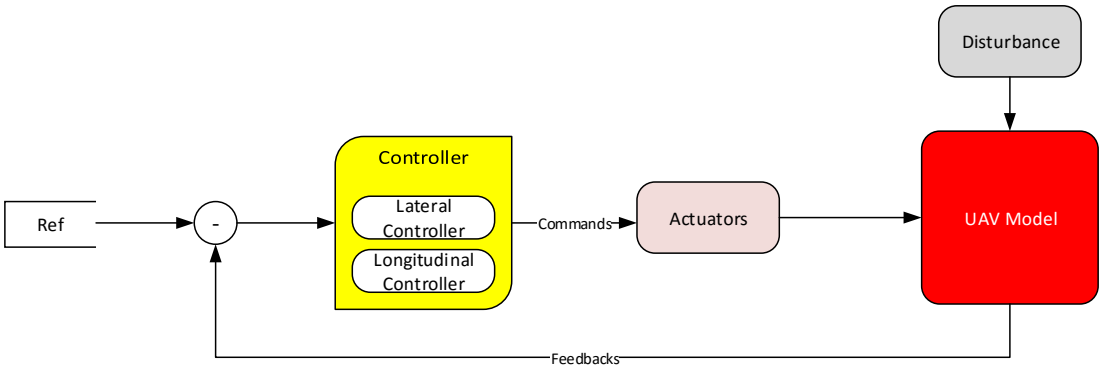


Figure 4-1 Inner Loop Control System Representation

Outer loop is designed to control velocity and position of the UAV. Also for more realistic usage, by using GCS interface, developed under this thesis, aimed velocity and speed of the outer loop controller may be modified. While controlling position of the UAV, position source may be changed with GPS or visual positioning system (VPS) by GCS. Details about GCS interface is presented at Appendix A.

In this design, landing autopilot commands to DeHavilland Beaver aircraft model with five control surface shown in Table 4-1 and presented in Figure 4-2.

Control Surface	Action
Engine Thrust	Speed
Aileron	Roll (Bank) Angle
Yaw	Heading Angle
Elevator	Pitch Angle
Flap	increase lift & drag

Table 4-1 Control Surfaces

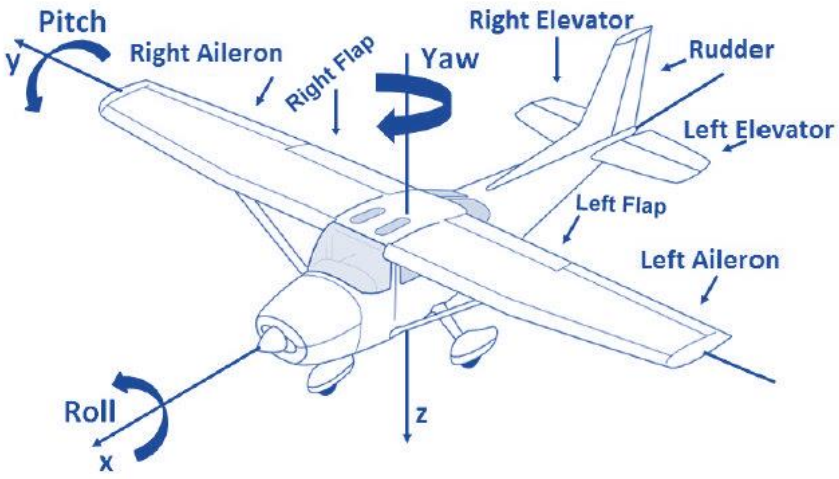


Figure 4-2 Control Surface on DeHavilland Beaver

4.1 Controller Requirements

Before designing an autopilot, requirements must be clearly defined before the design. Thus, performance of the design process can then be tested according to the requirements. In this thesis, requirements are grouped into two category. First category requirements are about behavior of the UAV before touchdown and second requirements mention about at touchdown phase of the flight. Controller requirements can be listed as in Table 4-2. Also, landing operation may continue under environmental requirements listed in Table 4-3.

	In Air		**At Touchdown**	
	Min	**Max**	**Min**	**Max**
Roll Angle	-20°	20°	-10°	10°
Pitch Angle	-15°	15°	1°	7°
$\dot{h}$	-	-	-	-3 (f/s)
Decrab Angle	-12°	12°	-4°	4°
Air Speed	80 kts	110 kts	79 kts	82 kts

Table 4-2 Landing Autopilot Requirements

"At Touchdown" requirements have narrower range than "In Air" requirements due to extra interaction with ground. Therefore, roll angle range must be limited to prevent wing edges to touch to the ground or apply unbalance force to landing gears. Similarly, due to ground interaction, decrab angle must be small values. Otherwise, force applied to landing gear may cause damages. Pitch angle must be positive value to slow down vertical velocity $\dot{h}$ and as a result, ensures that the rear wheels touch to the ground first. Beaver speed ranges from 70 kts to 110 kts as stated in [4]. Landing speed is accepted to be 20% higher than stall speed which is 66 kts so minimum landing speed obtained as $66 * 1.2 \cong 79$ kts.

Wind Types	Wind Magnitudes
Side Wind	±15 kts
Nose Wind	20 kts
Tail Wind	6 kts

Table 4-3 Environmental Requirements for Landing

In real life, actuators have operation limits. Although there is stated range for actuator limits, for this thesis actuator limits are accepted as in Table 4-4.

Actuators	Ranges
Aileron	±20°
Elevator	±15°
Rudder	±15°
Throttle	0% - 100%
Flaps	0° - 10° - 20°

Table 4-4 Actuator Limit Constraints

4.2 Proportional-Integral-Derivative (PID) Controller

Purpose of a control system is to obtain pre-defined system responses. Feedback control is the most widely used control structure whose aim is to make error signal, which is the difference between output and input signal, as small as possible [30].

$$\lim_{t\to\infty} error = \lim_{t\to\infty} (Output - Input) = 0$$

Proportional-Integral-Derivative (PID) controller has several important properties like removing steady state offsets by integration, and prediction of the future by derivation. Mathematically, PID controller can be described as:

$$\tau = \mathrm{K_p} e(t) + K_i \int_0^t e(\tau)dt + K_d \frac{d}{dt} e(t)$$

Where K_p, K_i and K_d are proportional, integral and derivative gains respectively and τ is control output. K_p is simply multiplied control error to minimize error signal smaller, however, too high proportional gain K_p can cause instability and proportional gain only itself cannot eliminate steady state error.

The integration term can eliminate steady state error. Integration property of controller deals with past values of error. It can cause overshoot because it reacts to accumulated error from the past up to current situation.

Overshoots can be solved by adding the derivative term. Derivative action deals with the prediction of the future values of error. However, derivative term can cause instability because derivative term is sensitive to noise in the error signal which could be caused by a sensor or disturbance comes from environment.

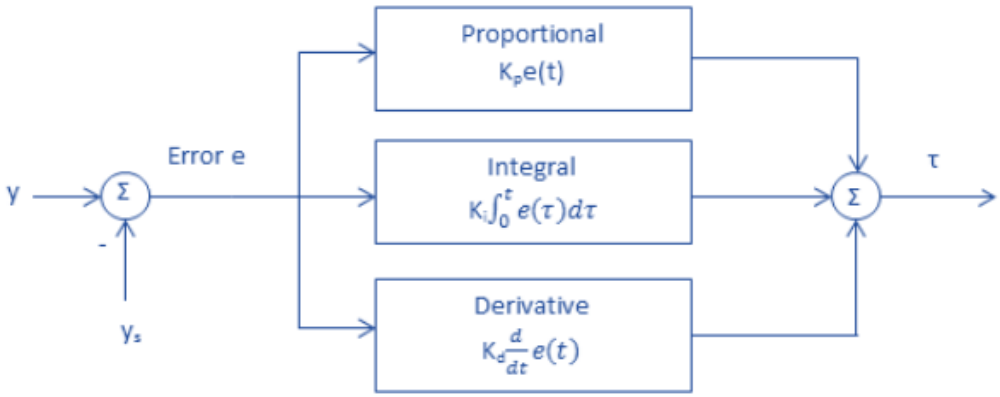


Figure 4-3 Block Diagram of PID Controller

To control a nonlinear system, gain scheduling is an approach to match pre-defined requirements. This approach also named as divide and conquer because with this method nonlinear design decomposes into different operating points of the system.

As stated in [4], most dominant factor for Beaver is velocity and in this thesis operational speed is not much varied due to landing purpose.

In real life, some other nonlinear phenomena must be taken into consideration. There are limitations in actuators; motors have limited throttle input or speed, etc. For a control system which must be operated in wide range of operational situations, control outputs can reach the actuator limits. Under this conditions, feedback loop is broken and system turns into an open loop control system because after actuators are saturated, whatever the output of the controller output, actuators remains constant at their limit value. To be able to survive from this situation, error must change sign up to integrator winds down which may cause to large transients. Here, anti-windup methods can be used to solve this problem [31].

In Figure 4-4, anti-windup phenomenon is illustrated on PI controller. As can be seen, reference signal is changed so large and actuator signal is saturated. Accumulation of integral signal continues up to at time 10 because error is positive. After time 10 to 20, although error is negative, output of the controller remains saturated due to high integral term and does not leave saturation until the error has been negative for a sufficiently long time. Notice that the control signal bounces between its limits.

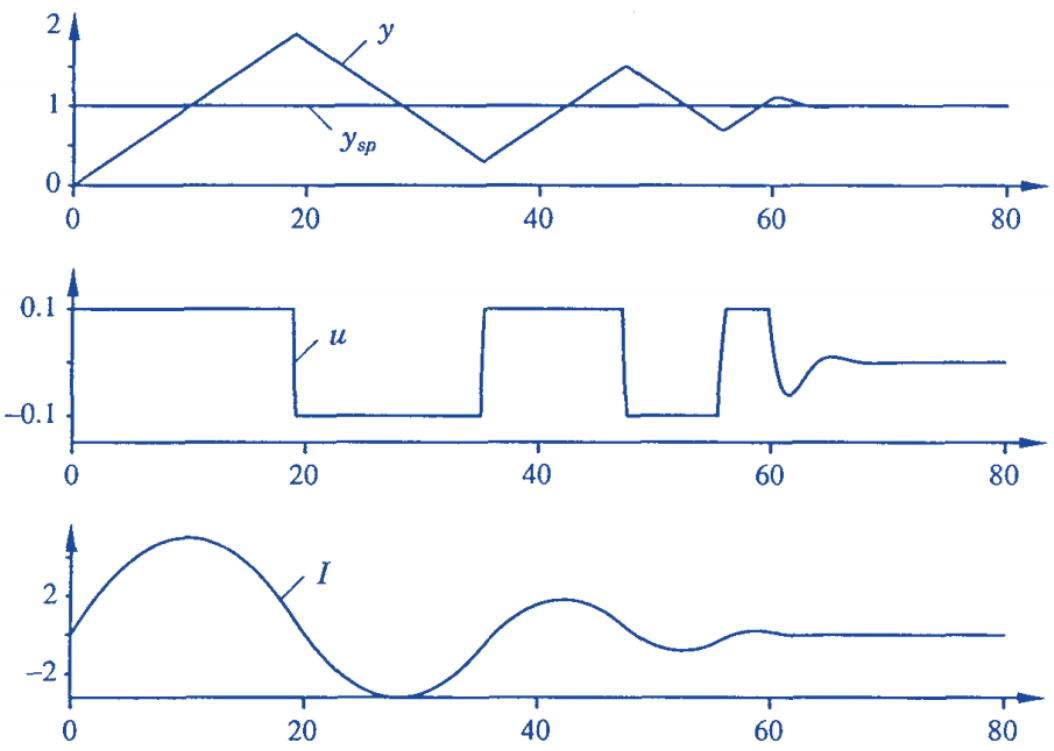


Figure 4-4 Illustration of integral windup where output is y, set point is y_{sp}, control signal is u and integral part is I

There are ways to solve windup problem. In this thesis, back-calculation and clamping anti-windup methods are used.

4.2.1 Back-Calculation Anti-Windup Method

Back-calculation method adds an extra feedback path to the control system that is forming an error signal (e_s) by measuring difference between the actual actuator output and controller output. The signal e_s is fed to the input of the integrator through gain $1/T_t$. When no saturation exist, e_s is equal to zero and this extra feedback loop has no effect on the system. This feedback loop attempts to make e_s equal to zero. Therefore, controller output is kept close to saturation limit. Back-calculation method is illustrated in Figure 4-5.

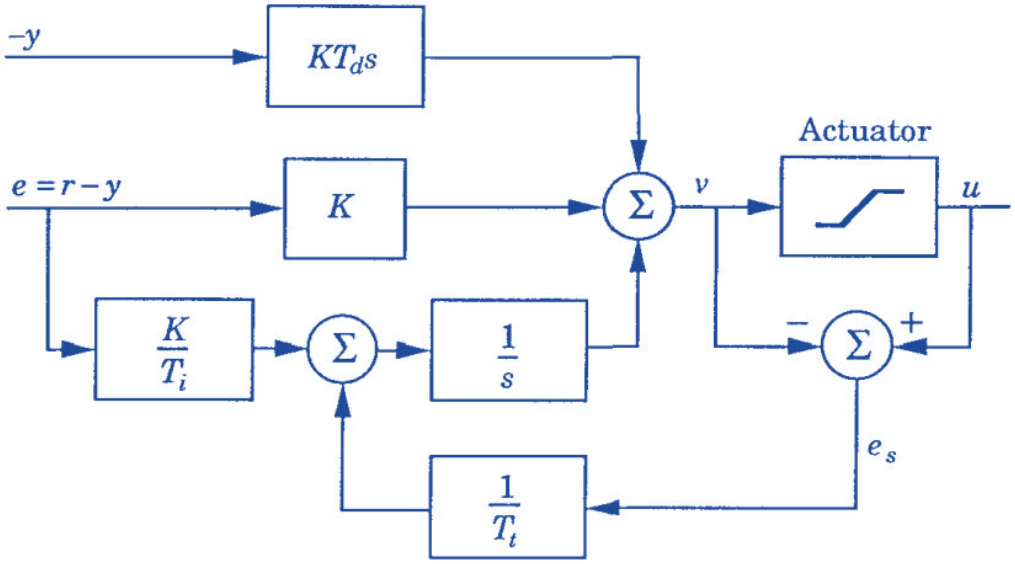


Figure 4-5 Back- Calculation Anti-Windup Method

Result of Back-calculation method can be investigated in Figure 4-6 and can be compared with Figure 4-4, which does not have anti-windup algorithm.

The integral input is

$$\frac{1}{T_t}e_s + \frac{K}{T_i}e$$

where e is the control error. Therefore,

$$e_S = -\frac{KT_t}{T_i}e$$

in steady state. Since $e_S = u - v$, it follows that

$$v = u_{lim} + \frac{KT_t}{T_i}e$$

where u_{lim} is the saturating value of the control variable. Since the signals e and u_{lim} have the same sign, it fallows that v is always larger than u_{lim} in magnitude.

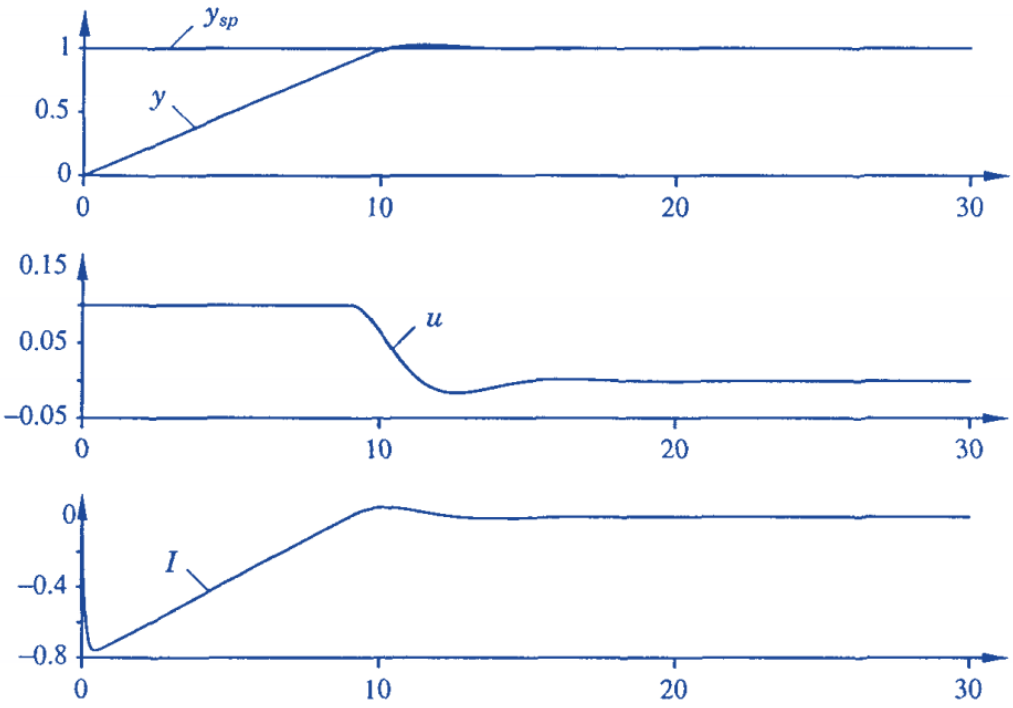


Figure 4-6 Illustration of Back-calculation method effects on controller

4.2.2 Clamping Anti-Windup Method

Clamping anti-windup method is so called conditional integration. It consist of switching off the integration when the controller variable saturates and output and the controller error and control variable have the same sign ($e(t) \cdot u(t) > 0$) [31]. Clamping anti-windup method is illustrated in Figure 4-7 and effect of it is presented in Figure 4-8.

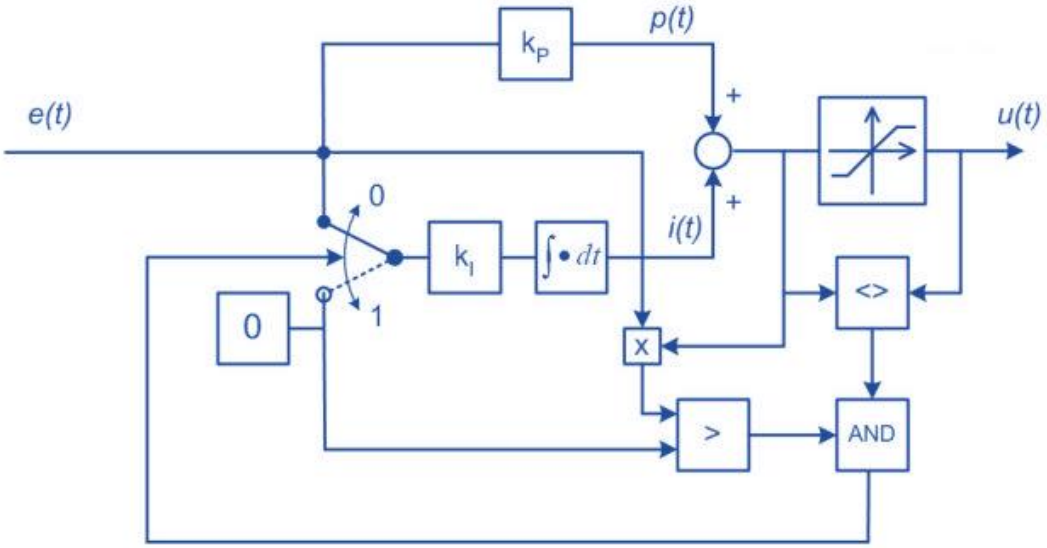


Figure 4-7 Integrator clamping $(e(t) \cdot u(t) > 0)$

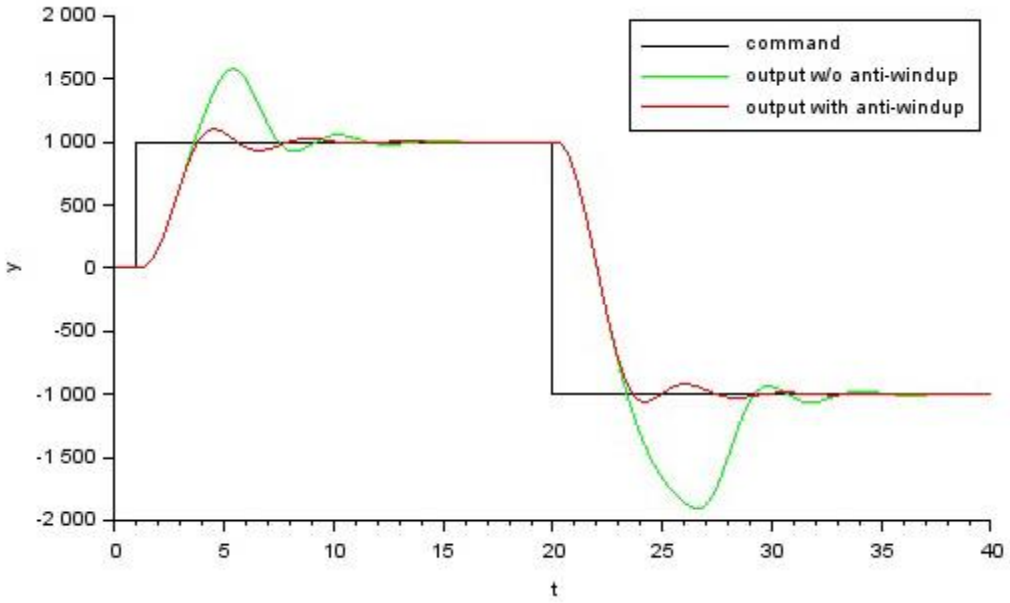


Figure 4-8 Clamping Anti-Windup Effect

4.2.3 Time Domain Performance

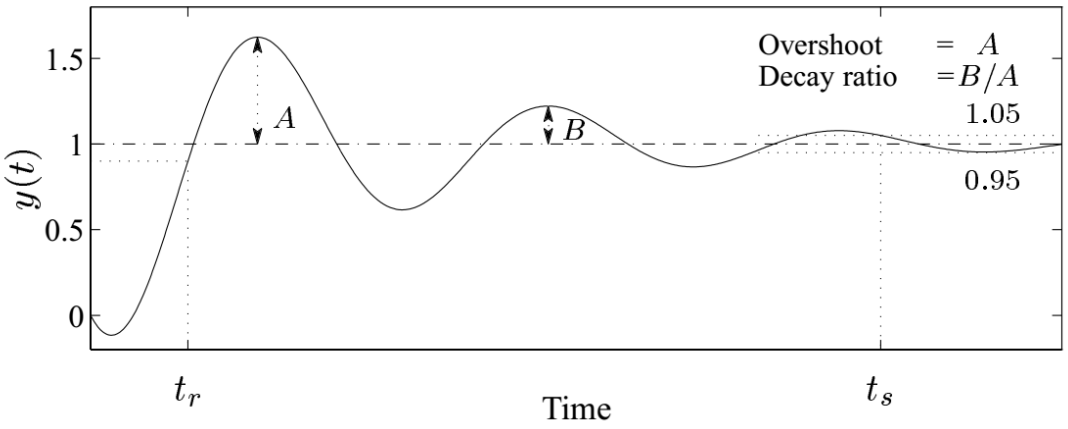


Figure 4-9 Characteristics of closed-loop response to step in reference [32]

Performance of the designed controllers will be evaluated with time domain performance criteria as follows:

- **Rise Time (t_r):** The time it takes for the output to first reach 90% of its final value. Which is usually required to be small.
- **Settling Time (t_s):** The time after which the output remains within 5% of its final value, which is usually required to be small.
- **Overshoot:** the peak value divided by the final value, which should typically be 1.2 (20%) or less.
- **Decay ratio:** the ratio of the second and first peaks, which should typically be 0.3 or less.
- **Steady-state offset:** the difference between the final value and the desired final value, which is usually required to be small.

4.3 Inner Loop

Aim of the inner loop controller is to increase the safety of flight and improving wind limits of the UAV to provide safe landing even under strong wind conditions. For this purpose, inner loop controller is designed as at Table 4-5.

Longitudinal Hold	Lateral Hold
• Pitch Controller • Speed Controller • Altitude Controller	• Roll Controller • Heading Controller

Table 4-5 Inner Loop Autopilot Capabilities

4.3.1 Lateral Controller

Lateral controller controls aileron and rudder control surfaces to be able to hold bank angle and heading of the UAV. For navigating through waypoints, heading must be controlled to direct UAV to target waypoint or to make direction error between UAV and runway heading as small as possible. For safe landing bank angle and heading of UAV must be controlled also.

Roll Controller

For a fixed-wing UAV, while flying at a constant altitude, roll angle command is often used for heading control. Therefore, 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 with respect to Table 4-2, otherwise wings can touch the ground which probably will cause damages or loss of the vehicle.

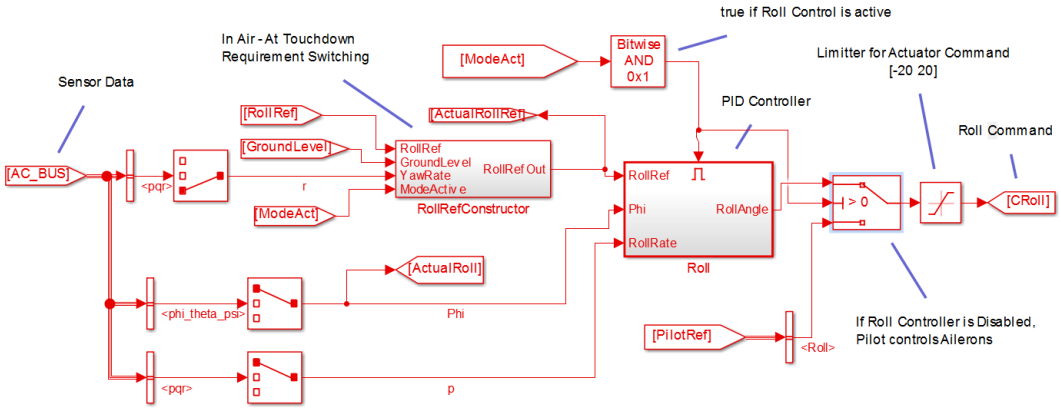


Figure 4-10 Roll Controller

PID controller is used to keep roll angle to roll reference value by controlling aileron actuator angle. Ground control station can disable or enable control modes individually. If roll controller or all autopilot is disabled, aileron actuator angle is set directly by pilot inputs. This enable/disable signal comes to hold autopilot model with “ModeAct” signal by the first bit of “ControlEnableSelections” variable in Table A - 1 Message Table of GAM_CONTROL_DATA. All structure of roll controller is presented in Figure 4-10.

Roll controller has an effect on heading control. While heading control works for holding heading of UAV, roll controller reference is manipulated to make UAV rolling to direction of turning so yaw rate used as a damper. As a result of that, UAV can reach target heading value faster and loss of airspeed is minimized. However, for landing phase, whatever the yaw effect is, if altitude is lower than predefined value, then roll controller try to keep zero bank angle to match “At Touchdown” requirement. Reason of this predefined altitude level is that while approaching the runway, controller must satisfy both “In Air” and “On Ground” requirements as both detailed in Table 4-2 and for a successful switching between these requirements, there must be a time interval given. By keeping in mind the other requirements, this value has been determined to be 5 m (MinLevel2CorrectHeading variable in Figure 4-11). Related logic can be investigated in Figure 4-10 and Figure 4-11.

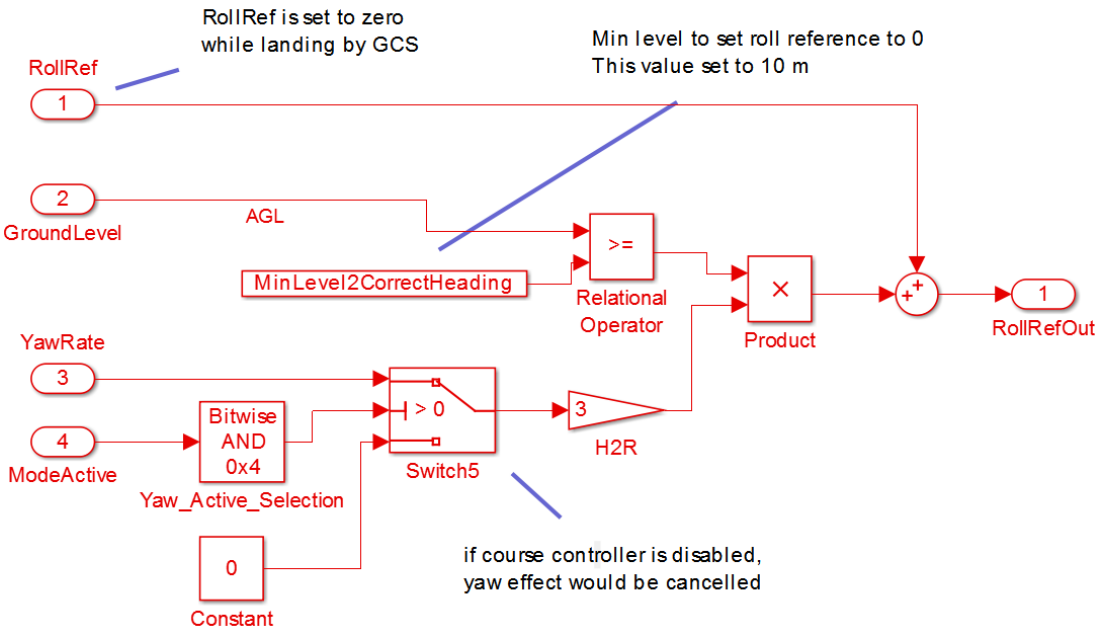


Figure 4-11 "RollRefConstructor" Block. Sets Roll Reference with respect to Requirements stated in Table 4-2

Aileron actuator angle is limited between [-20 20] degree with respect to requirements stated in Table 4-4 and roll command is limited with respect to this requirement. Clamping anti-windup method is applied to limit the integrator effect on block saturation [33] to handle this nonlinear outcome.

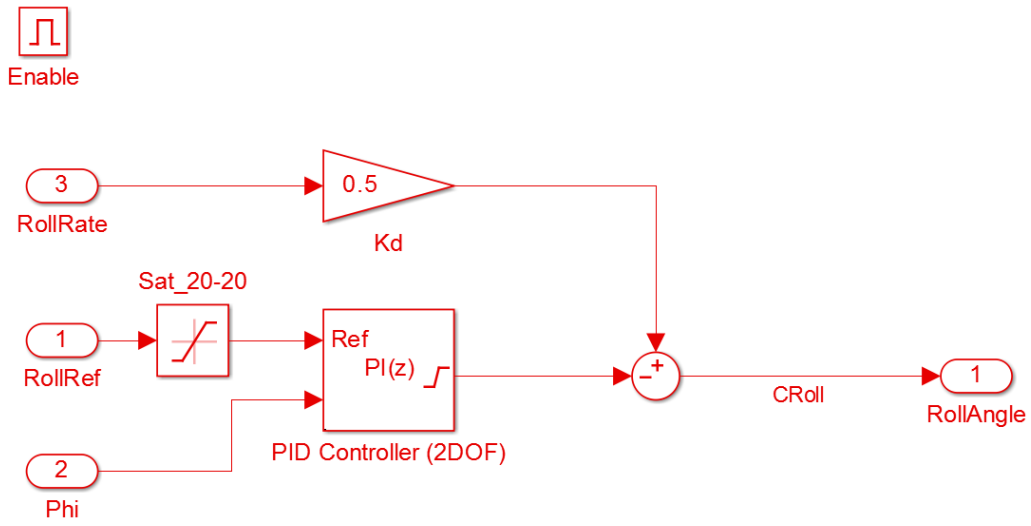


Figure 4-12 Roll Controller

"Roll" block where main PID controller exist in can be investigated in Figure 4-12. To match "In Air" requirements, roll reference is limited between ±20. "At Touchdown" requirement handled in "RollRefConstructor" block as presented in Figure 4-11. Under 5 m altitude, roll reference is set to zero.

	P	I	D	Tr	Ts	OvShoot	Decratio	peak_1	Peak_2	Step
P	8	2	1	1.13	1.43	2.9	0	10.29	10	10
	6	2	1	1.17	1.61	4.2	0	10.42	10	10
	4	2	1	1.23	5.63	6.8	0	10.68	10	10
	2	2	1	1.23	5.76	18.9	0	11.89	10	10
	1	2	1	1.25	13.4	37.7	0	13.77	10	10
I	4	1	1	1.39	6.44	6.5	0	10.65	10	10
	4	2	1	1.23	5.63	6.8	0	10.68	10	10
	4	4	1	1.23	4.56	6.9	0	10.69	10	10
	4	6	1	1.23	3.64	6.9	0	10.69	10	10
	4	8	1	1.05	2.42	8.4	0	10.84	10	10
D	4	2	0	0.6	1.74	7.5	0.573333	10.75	10.43	10
	4	2	0.5	0.9	4.04	6	0	10.6	10	10
	4	2	1	1.23	5.63	6.8	0	10.68	10	10
	4	2	1.5	1.63	5.04	9.3	0	10.93	10	10
Fine-Tune	6	1	1	1.21	2.12	3.4	0	10.34	10	10
	6	2	0.5	0.89	1	4.5	0	10.45	10	10
	6	2	1	1.17	1.61	6	0	10.6	10	10
	6	4	1	1.2	1.4	6.5	0	10.65	10	10

Figure 4-13 PID Tuning and Step Response Performance Analysis for Roll Controller

PID gains of the roll controller are tuned manually and parameters were changed systematically for analyzing the controller step response performance and understand result behavior of the nonlinear aircraft model. Result of this process is presented in Figure 4-13. As a result of this analysis, P parameter is selected as 6 to decrease overshoot ratio and settling time. Also this makes steady state error reasonably small. I parameter is selected as 2 to decrease steady state error. Further increase of I variable increases overshoot ratio. D parameter is selected as 0.5 to decrease overshoot and oscillations. Further increase in D cause increase in settling time, rise time and overshoot ration. Time domain performance of the roll controller is resulted as in Table 4-6 and step response is shown in Figure 4-14.

P	I	D	T_r	T_s	Overshoot
6	2	0.5	0.89	1	4.5

Table 4-6 Gain Values and Time Domain Performance of Roll Controller

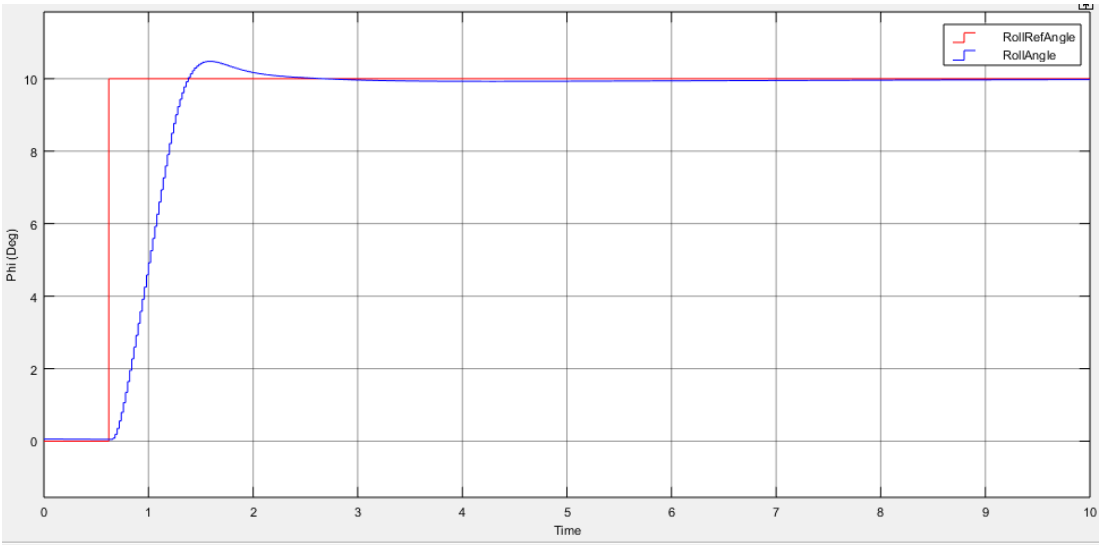


Figure 4-14 Step Response of Roll Controller

Heading Controller

Heading controller aims to keep heading angle of UAV to reference value. Heading controller commands to rudder actuators to change heading. If altitude is higher than 5 m, as mentioned at Roll Controller section, change in yaw rate has an effect on roll controller. Thus, UAV can reach target heading value faster and loss of airspeed is minimized. If altitude is lower than 5 m, to match “At Touchdown” requirements listed in Table 4-2, this roll effect is canceled to protect wings and landing gears. General algorithm is presented in Figure 4-15 and Figure 4-16.

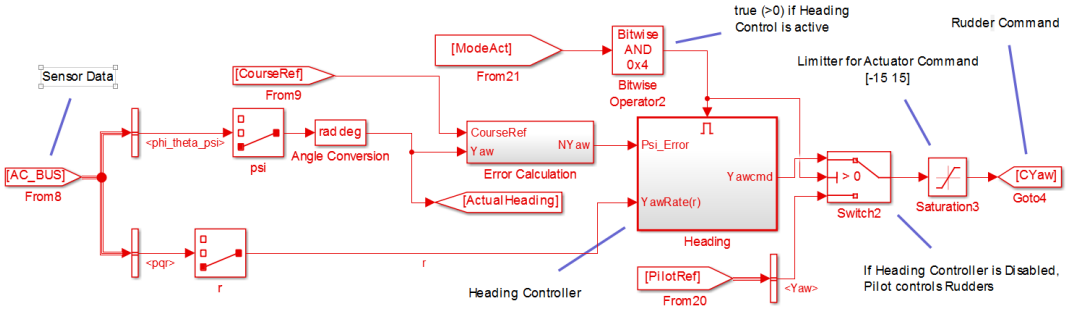


Figure 4-15 Heading Controller Algorithm

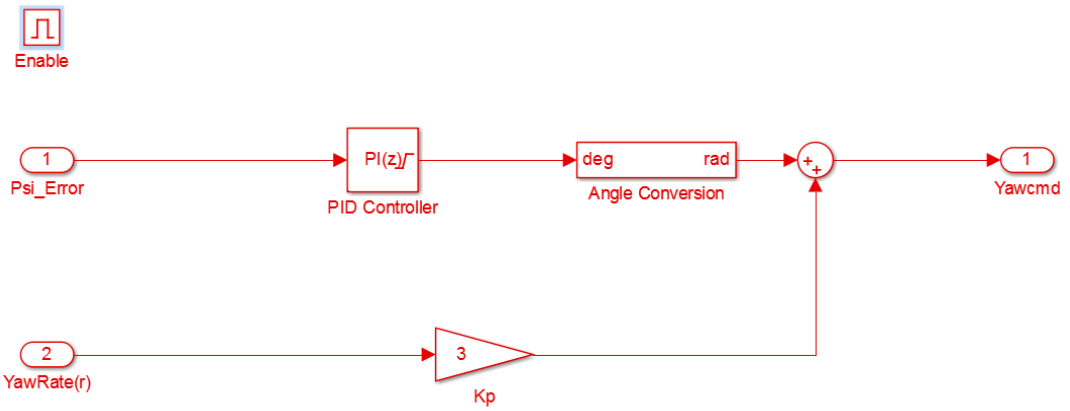


Figure 4-16 “Heading” Block

Ground control station can disable or enable control modes individually. This enable/disable signal comes to hold autopilot model with “ModeAct” signal by the third bit of “ControlEnableSelections” variable in Table A - 1 Message Table of GAM_CONTROL_DATA. If heading controller or autopilot is disabled, rudder actuator angle is set directly by pilot inputs.

Rudder actuator angle is limited between [-15 15] degree with respect to requirements stated in Table 4-4 and yaw command is limited with respect to this requirement. Back-calculation anti-windup method is applied to limit the integrator effect on block saturation [33] to handle this nonlinear outcome.

	P	I	D	Tr	Ts	OvShoot	peak_1	Peak_2	Step
P	2	1	3	2.45	8.94	21	12.1	10	10
	3	1	3	2.7	3.05	7.7	10.77	10	10
	4	1	3	3.31	4.26	0.2	10.02	10	10
	5	1	3	4.7	7.16	0.1	10.01	10	10
I	4	0.5	3	6.45	9.54	0.1	10.01	10	10
	4	1	3	3.31	4.26	0.2	10.02	10	10
	4	1.5	3	2.68	3.08	4.1	10.41	10	10
	4	2	3	2.43	6.26	7	10.7	10	10
D	4	1	1	1.27	3.99	0.1	10.01	10	10
	4	1	2	2.73	4.12	0.1	10.01	10	10
	4	1	3	3.31	4.26	0.2	10.02	10	10
	4	1	4	3.97	4.73	1.5	10.15	10	10

Figure 4-17 PID Tuning and Step Response Performance Analysis for Heading Controller

Time domain performance analysis is presented in Figure 4-17. Heading controller time domain analysis is done to investigate for matching "At Touchdown" requirements. Therefore, roll controller is commanded to hold 0 degree throughout this analysis. As a result, P parameter is selected as 4 to decrease overshoot ratio and settling time. Also this makes steady state error reasonably small. Integral parameter is selected as 1 to decrease steady state error. Further increase of I variable increases overshoot ratio. D parameter is selected as 3 to decrease overshoot and oscillations. Further increase in D cause increase in settling time, rise time and overshoot ration. Time domain performance of the heading controller is resulted as in Table 4-7 and Figure 4-17.

P	**I**	**D**	**T_r**	**T_s**	**Overshoot**
4	1	3	3.31	4.26	0.2

Table 4-7 Gain Values and Time Domain Performance of Heading Controller

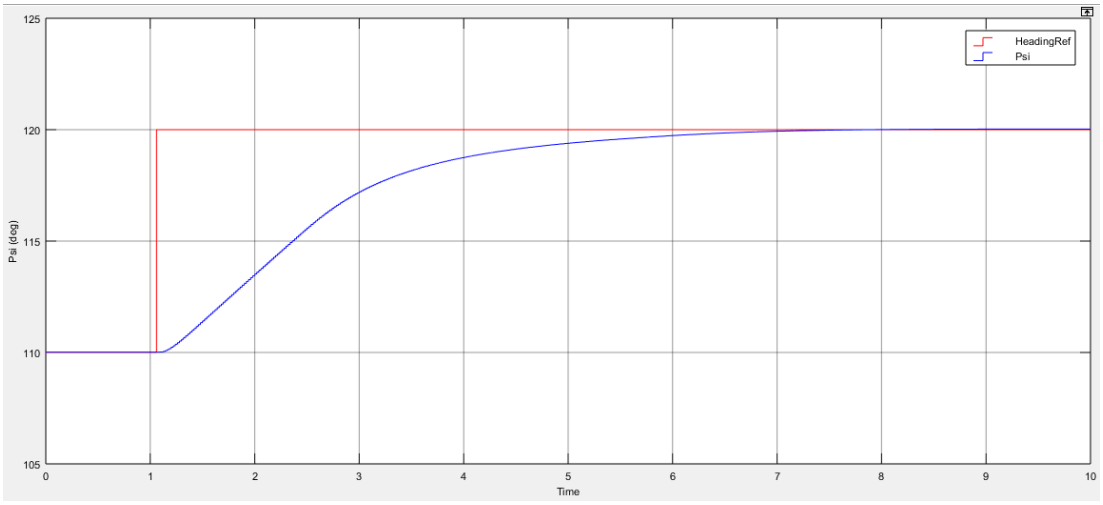


Figure 4-18 Step Response of Heading Controller

4.3.2 Longitudinal Controller

Longitudinal controller controls elevator and engine thrust control surfaces to be able to hold pitch angle and airspeed of the UAV. For navigating through waypoints and safe landing, altitude and speed of UAV must be controlled. In this design, altitude is controlled by elevator and speed is controlled by engine thrust.

Pitch & Altitude Controller

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

Ground control station can disable or enable control modes individually. If pitch controller or autopilot is disabled, elevator actuator angle is set directly by pilot inputs. This enable/disable command reaches to the controller with the second bit of

"ControlEnableSelections" variable of "ModeAct" signal in Table A - 1 Message Table of GAM_CONTROL_DATA.

PID controller is used to keep pitch angle to pitch angle reference value by controlling elevator actuator angle. Elevator actuator angle is limited between [-15 15] degree with respect to requirements stated in Table 4-4 and elevator command is limited with respect to this requirement. Clamping anti-windup method is applied to limit the integrator effect on block saturation [33] to handle this nonlinear outcome. General pitch controller algorithm can be investigated in Figure 4-19 and Figure 4-20.

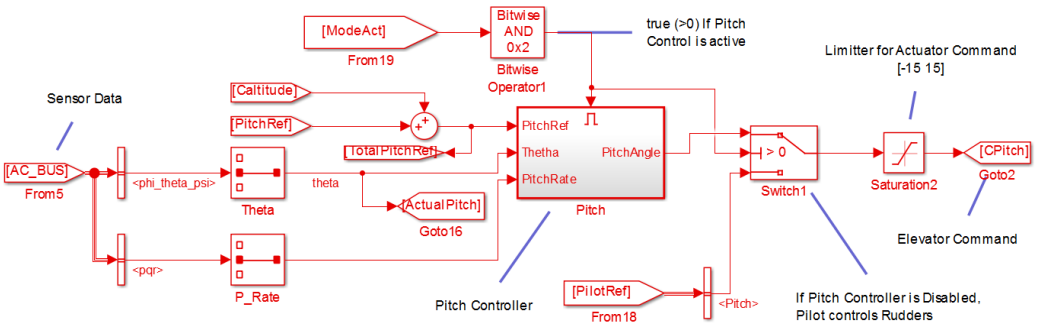


Figure 4-19: Pitch Controller

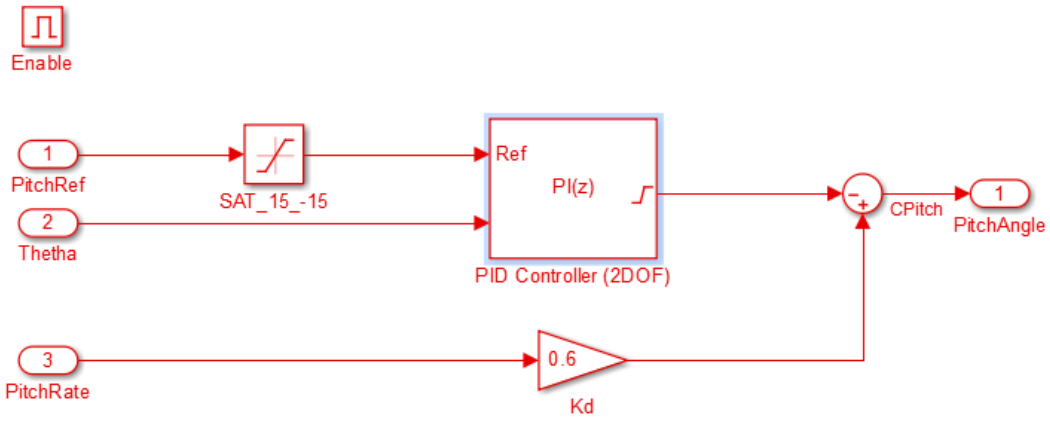


Figure 4-20 "Pitch" Block

Pitch controller is tested by setting “PitchHoldRef_deg” variable and disabling altitude controller by setting “ControlEnableSelections” in Table A - 1 Message Table of GAM_CONTROL_DATA. Altitude controller output (CAltitude) and “PitchHoldRef_deg” is summed before entering Pitch controller in Figure 4-19 so by disabling altitude control, step input can be set without altitude controller contribution. Time domain performance analysis is presented in Figure 4-21. As a result, P parameter is selected as 4 to decrease overshoot ratio, settling time and to make steady state error reasonably small. Integral parameter is selected as 1 to decrease steady state error. Further increase of I variable increases overshoot ratio. D parameter is selected as 3 to decrease overshoot and oscillations. Further increase in D cause increase in settling time, rise time and overshoot ration. Time domain performance of the heading controller is presented in Table 4-8 and Figure 4-22.

	P	I	D	Tr	Ts	OvShoot	Decratio	peak_1	Peak_2	Step
P	2	1	0.6	0.95	4.89	8.6	0	10.86	10	10
	3	1	0.6	0.95	1.27	2	0	10.2	10	10
	4	1	0.6	0.75	1.12	0.4	0	10.04	10	10
	5	1	0.6	0.55	1.09	0.01	0	10.001	10	10
	6	1	0.6	0.5	0.55	1.2	0	10.12	10	10
I	3	0.5	0.6	0.98	1.42	0.2	0	10.02	10	10
	3	1	0.6	0.95	1.27	2	0	10.2	10	10
	3	2	0.6	0.75	0.94	4.9	0	10.49	10	10
	3	3	0.6	0.69	2.49	7	0	10.7	10	10
	3	4	0.6	0.61	2.25	7.9	0	10.79	10	10
D	3	1	0.1	0.4	1.15	11.5	5.217391	11.15	10.06	10
	3	1	0.6	0.95	1.27	2	0	10.2	10	10
	3	1	1	1.13	1.39	4	0	10.4	10	10
	3	1	2	1.64	5.89	7.8	0	10.78	10	10
Fine-Tune	4	1	0.6	0.78	1.18	0.3	0	10.03	10	10
	4	1	1	1	1.32	1.2	0	10.12	10	10
	4	1	0.6	0.75	1.12	0.4	0	10.04	10	10
	4	2	0.6	0.69	0.99	1.7	0	10.17	10	10

Figure 4-21 PID Tuning and Step Response Performance Analysis for Pitch Controller

P	I	D	T_r	T_s	Overshoot
4	1	0.6	0.78	1.18	0.3

Table 4-8 Gain Values and Time Domain Performance of Pitch Controller

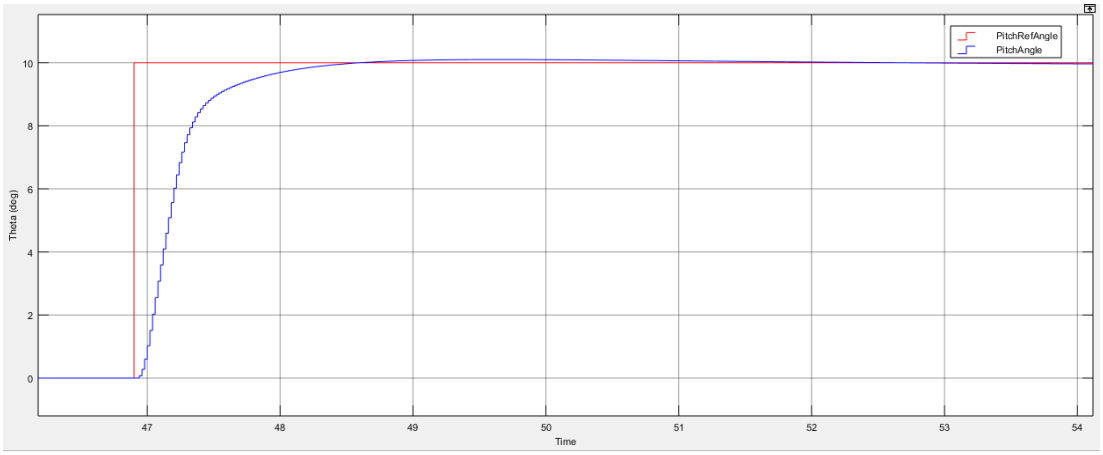


Figure 4-22 Step Response of Pitch Controller

Altitude controller developed on pitch controller by using PID controller is shown in Figure 4-23. Altitude controller work on keeping error as small as possible between altitude of the UAV and altitude reference value. Output of altitude controller contributes to Pitch reference which is set by “AltitudeHoldRef_ft” variable of GCS control command in Table A - 1.

As stated before, elevator actuator angle is limited between [-15 15] degree with respect to requirements stated in Table 4-4 and elevator command is limited with respect to this requirement. There would be major changes in reference of the altitude controller. In addition to this cases, it would take time to reach destination altitude due maximum climb rate and maximum descend rate of the UAV. For such cases, integrator factor of the PID controller cause overflow in time and this situation cause big overshoots at the output. For these reasons, back-calculation windup method used to discharge saturation of integration action. General algorithm can be investigated in Figure 4-23 and Figure 4-24.

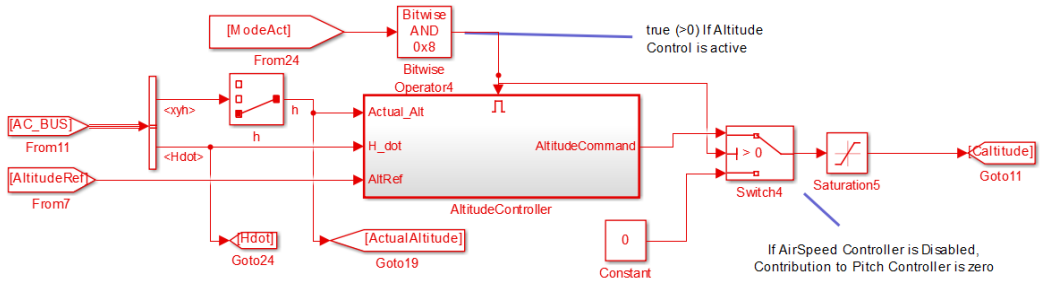


Figure 4-23 Altitude Hold Controller

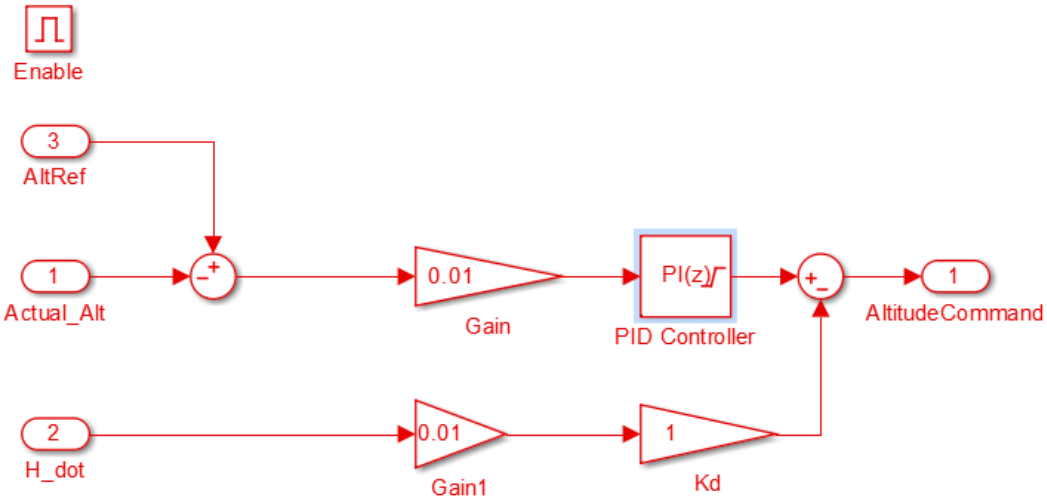


Figure 4-24: PID Block of Altitude Controller

Step response of altitude controller is obtained by giving step input with GCS control command. For altitude controller both angle of descent and climb is important for auto landing capable air vehicle. So altitude controller is tested by setting altitude reference from 1300 feet to 1200 feet altitude difference and vice versa. Result is presented in Figure 4-26 and Table 4-9. Also time domain performance analysis is done for various PID gains to understand system nonlinearities.

	P	I	D	Tr	Ts	OvShoot	Decratio	peak_1	Peak_2	Step
P	1	0.4	1	3.61	19.15	45.82	4.037538	145.82	101.85	100
	2	0.4	1	4.66	5.4	0.001	0	100.001	100	100
	3	0.4	1	12.6	17.1	0.001	0	100.001	100	100
	4	0.4	1	18.96	25.37	0.001	0	100.001	100	100
I	2	0.2	1	3.75	16.2	35.63	0	135.63	100	100
	2	0.4	1	4.68	5.51	0.001	0	100.001	100	100
	2	0.6	1	3.9	7.68	6.96	0	106.96	100	100
	2	1	1	3.5	7.83	18.8	2.5	118.8	100.47	100
	2	1.5	1	3.49	10.61	26.34	14.80638	126.34	103.9	100
D	2	0.4	0	3.57	8.95	-0.09	0	99.91	100	100
	2	0.4	1	4.68	5.51	0.001	0	100.001	100	100
	2	0.4	3	6.98	19.47	13.1	1.526718	113.1	100.2	100
	2	0.4	5	10.311	27.53	18.6	4.354839	118.6	100.81	100

Figure 4-25 PID Tuning and Step Response Performance Analysis for Altitude Controller

P	I	D	T_r	T_s	Overshoot
2	0.4	1	4.68	5.51	0.001

Table 4-9 Gain Values and Time Domain Performance of Altitude Controller

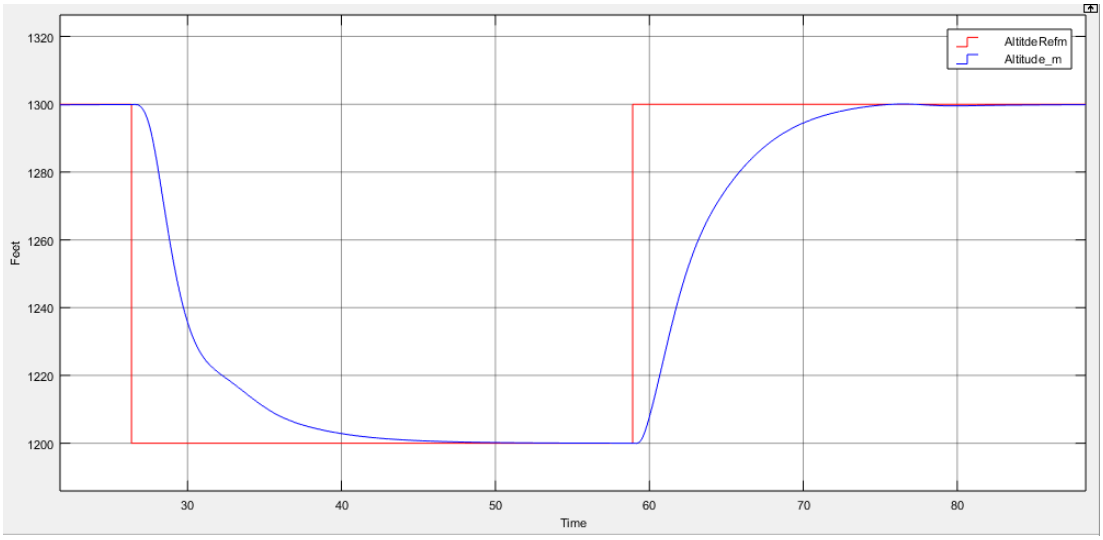


Figure 4-26 Step Response of Altitude Hold Controller

For tracking performance test of altitude controller, outer loop is used to generate landing trajectory path and detail of generation of landing trajectory will be explained at section 4.4. As can be investigated in Figure 4-27, tracking performance is successful.

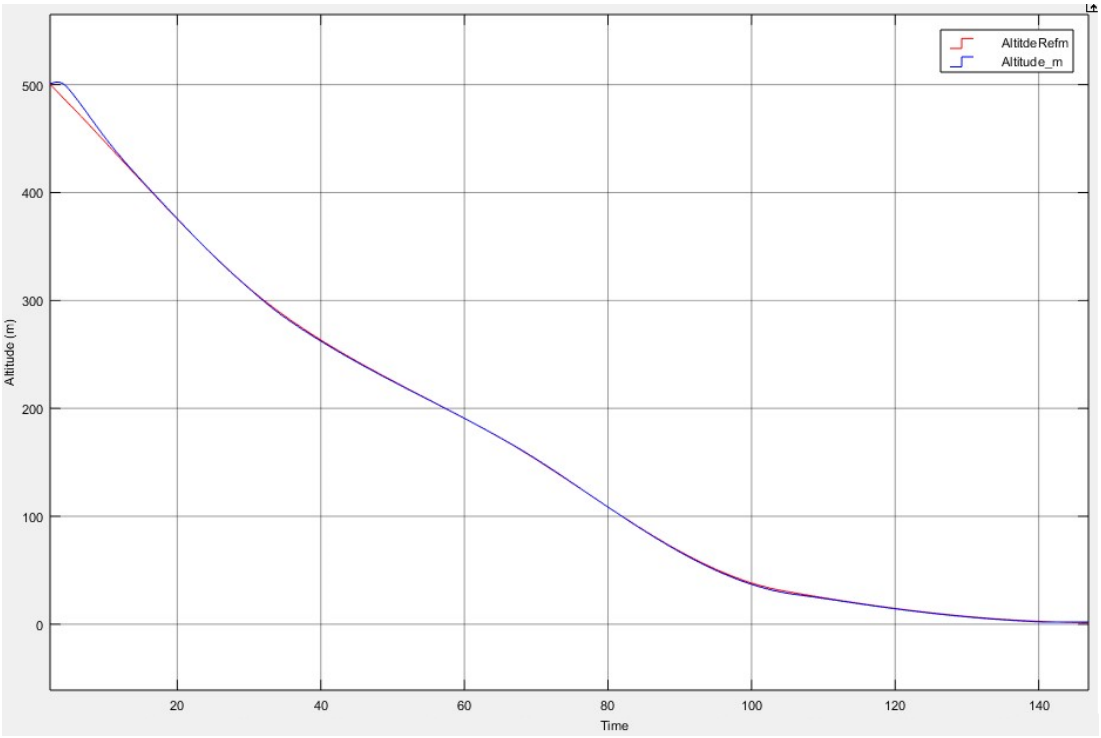


Figure 4-27 Altitude Controller Tracks Landing Trajectory Successfully

As mentioned before, GCS can set pitch and altitude references, or disable these controllers individually. Altitude controller is built on pitch controller, therefore, if pitch controller is disabled, altitude controller would be useless and pilot inputs applied to elevators whatever the altitude controller output is. If altitude controller is disabled only, pitch angle is kept at pitch reference value that comes from GCS and contribution of altitude controller to pitch controller would be zero degree.

Airspeed Controller

The main objective of the airspeed controller is to keep airspeed of the UAV at constant value. Airspeed controller can be designed in two ways. It can be either built on the pitch controller or by changing throttle value. In this design, airspeed controller controls throttle value. The final airspeed autopilot block diagram is shown in Figure 4-28 and Figure 4-29.

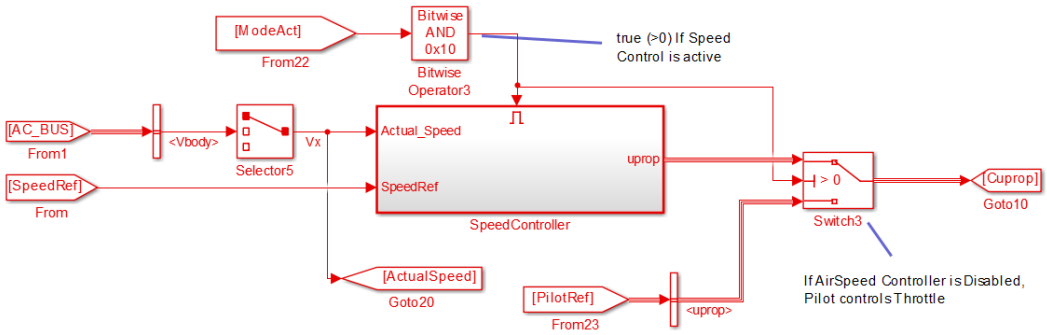


Figure 4-28 Air Speed Controller

Speed controller has crucial role while landing. Air speed is needed to be chosen 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. On the other hands, if speed of the UAV is below the stall speed, altitude of the UAV cannot be held as planned by outer loop and hard landing would occur. Airspeed requirement is stated in Table 4-2.

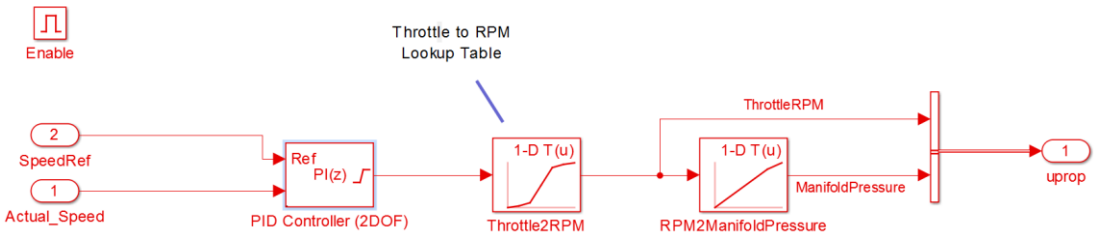


Figure 4-29: Air Speed Controller and Motor Model

PI controller is used to keep air speed to reference speed value by controlling throttle level and clamping anti-windup method is applied to limit the integrator effect due to nonlinear engine model and throttle limits. In contrast to other previous sections, PID is not preferred for airspeed hold controller because under turbulence air conditions, airspeed measurement could have high frequency components which is not suitable for derivative action. Throttle command range is [0 1] (which corresponds to 0% - 100% in Table 4-4) and this value converted into motor RPM (Revolutions per Minute) which is used for calculation of trust vector.

Airspeed controller can be disabled by GCS command. If air speed controller or autopilot is disabled, throttle is set directly by pilot inputs. This enable/disable command reaches to the controller with the fifth bit of "ControlEnableSelections" variable of "ModeAct" signal in Table A - 1.

Airspeed controller is tested by setting "SpeedHoldRef_deg" variable with 20° flap. During these tests, altitude controller is active. Time domain performance analysis is presented in Figure 4-30. P parameter is selected as 0.5 to decrease overshoots and settling time. Further increase in P parameter may cause frequent changes in throttle value. This situation cause oscillations on UAV due to change in forward trust force. Parameter I is selected 0.2 to eliminate steady state errors. Applied PI parameters and performance of the controller is presented in Table 4-10 and Figure 4-31

	P	I	D	Tr	Ts	OvShoot	Decratio	peak_1	Peak_2	Step
P	0.2	0.2	0	9	15.16	13.3	6.015038	11.33	10.08	10
	0.3	0.2	0	10.3	14.97	9.7	0	10.97	10	10
	0.4	0.2	0	9.6	12.81	6.3	0	10.63	10	10
	0.5	0.2	0	9	9.6	3.4	0	10.34	10	10
I	0.5	0.1	0	9.03	9.67	2	0	10.2	10	10
	0.5	0.2	0	9	9.6	3.4	0	10.34	10	10
	0.5	0.3	0	8.98	11.75	5.1	0	10.51	10	10
	0.5	0.4	0	9.43	12	5.8	0	10.58	10	10

Figure 4-30 PID Tuning and Step Response Performance Analysis for Speed Controller

P	I	T_r	T_s	Overshoot
0.5	0.2	9	9.6	3.4

Table 4-10 Gain Values and Time Domain Performance of Speed Controller

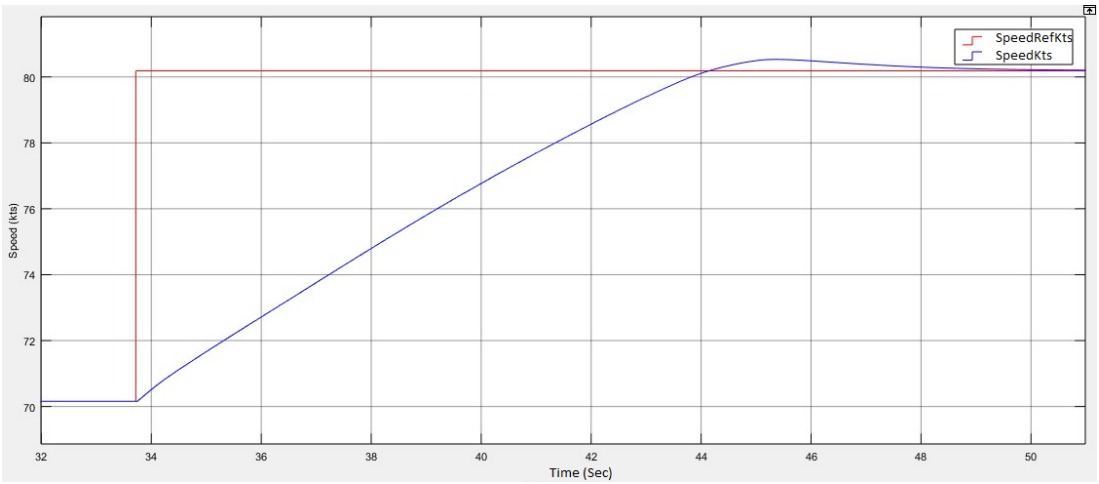


Figure 4-31 Step Response of Speed Hold Controller

4.4 Outer Loop

Outer loop is responsible for landing trajectory generation and it manages required maneuvers for a safe landing and reduces workload on the crew. Outer loop needs position information of the UAV and needs a database to store flight plan besides other air data information which is used by inner loop. Generally, position information is obtained from GPS. Moreover, in landing phase, INS information may also be used. In this thesis, outer loop is designed to use Visual Positioning System (VPS) in addition to GPS. So that, auto landing may continue even when GPS information is not reliable, available or GPS data is jammed.

In this design, outer loop requires flight plan for generating landing trajectory. Designed flight plan database consist of six parameters as follows:

- 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.

This parameters are stored in GAM_Mission_WayPoint composition in Table A - 3.

GCS can load up to ten waypoints to outer loop flight plan database with GAM_MISSION_DATA in Table A - 2 via UDP protocol. When outer loop mode is selected as FMS_Mode by “ReferanceSelection” value of GAM_CONTROL_DATA in Table A - 1, outer loop starts to execute this waypoints with respect to their order value. With respect to flight plan database, outer loop generate altitude, heading and speed reference as presented in Figure 4-32.

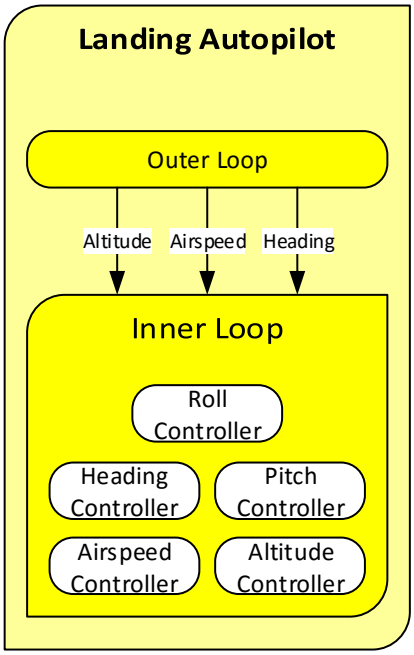


Figure 4-32 Outer Loop Feeds Inner Loop with Altitude, Airspeed and Heading References

Execution of the flight plan starts with choosing the flight mode as FMS_Mode by GCS. Outer loop generate altitude, airspeed and course reference to assure that the UAV follows the flight path. To make flight plan easier to manage, it is aimed to make a feasible path. This is an optimization problem therefore, to avoid instantaneous changes in position/angle, speed/rotation or acceleration/torsion, flight path needs to be continuous [34].

The most basic way to generate a path is to plot waypoints and raw straight lines between them. This kind of flight paths is sufficient for vehicles with medium or low demand of precision. However, big problem is that this flight paths may not be feasible or flyable because waypoints may have an instantaneous change of direction when going from one to another such as in Figure 4-33.

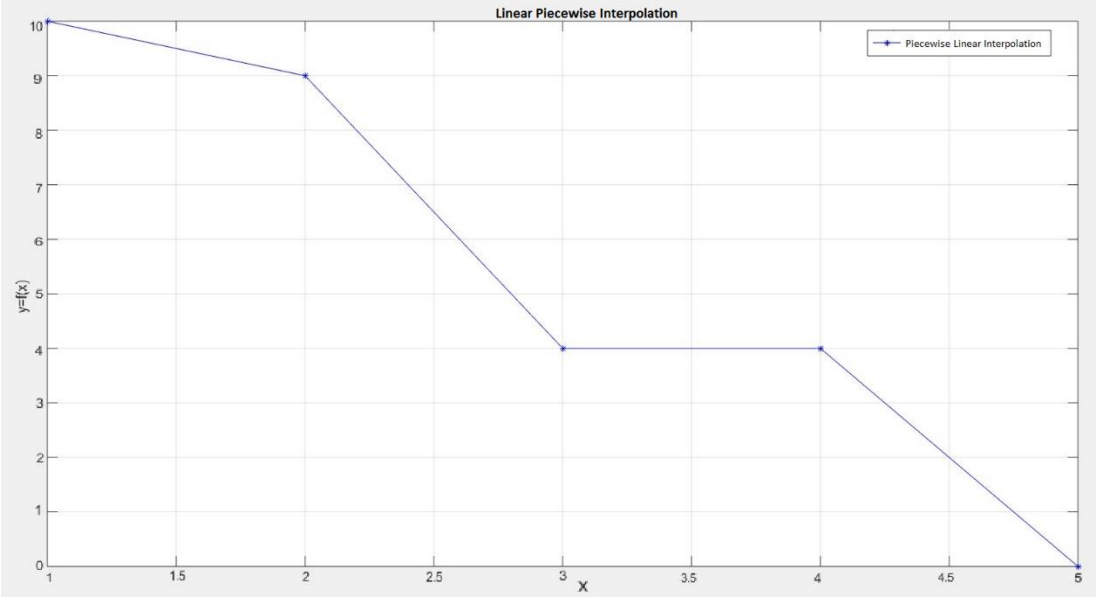


Figure 4-33 Piecewise linear interpolation of x= [1 2 3 4 5] and y= [10 9 4 4 0]

The derivative of the resulting curve in Figure 4-33 is not continuous. In order to fix this drawback, one can use Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) method [35] [36]. This interpolation is also called Hermite cubic interpolation. This interpolation is a method to ensure monotonicity of the resulting Hermite spline. By meaningfully selecting the interpolating curves, overshoot is prevented, meaning that the monotonicity of the path is not violated. This is of

special interest when there are obstacles present, and an overshoot can have fatal consequences as UAV may be hit them such as ground or a building.

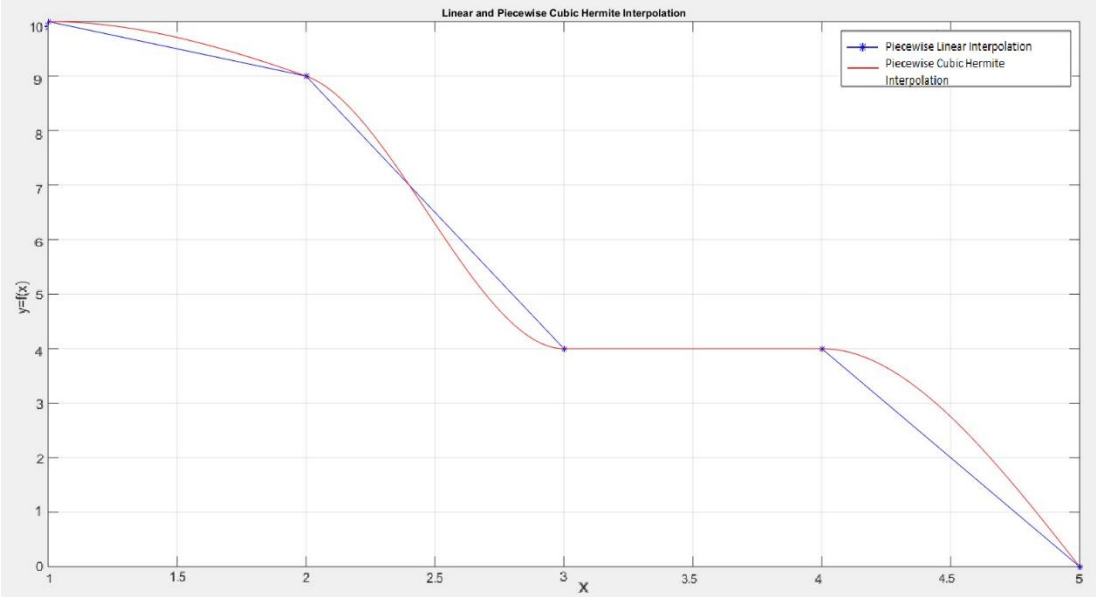


Figure 4-34 Piecewise cubic interpolation

Let h_k denote the length of the k_{th} subinterval:

$$h_k = x_{k+1} - x_k$$

$$\delta_k = \frac{y_{k+1} - y_k}{h_k}$$

Let d_k denote the slope of the interpolation at x_k:

$$d_k = P'(x_k)$$

For the piecewise linear interpolation, $d_k = \delta_k - 1$ or δ_k, but this is not necessarily true for higher order interpolation. The key idea is to determine the slopes d_k so that the function values do not overshoot the data values, at least locally. If δ_k and δ_{k-1} have opposite signs or if either of them is zero, then x_k is a discrete local minimum or maximum, so we set

$$d_k = 0.$$

This can be easily investigated result of an example interpolation in Figure 4-34. If δ_k and δ_{k-1} have the same sign, then d_k is a weighted harmonic mean, with weights determined by the lengths of the two intervals:

$$\frac{w_1 + w_2}{d_k} = \frac{w_1}{\delta_{k-1}} + \frac{w_2}{\delta_k}$$

Where

$$w_1 = 2h_k + h_{k-1}, \quad w_2 = h_k + 2h_{k-1}$$

If two intervals have the same length,

$$\frac{1}{d_k} = \frac{1}{2}\left(\frac{1}{\delta_{k-1}} + \frac{1}{\delta_k}\right)$$

Another advantage of this interpolation, which can be also seen from formulation above, is that moving a point do not affect all the curve. Consequently, in this design, generation of altitude and speed references are done by using PCHIP method. Course reference is generated with the calculation of current UAV situation and target direction.

Position source selection is done via "PositionSource" variable of GAM_CONTROL_DATA message of GCS presented in Table A - 1. If position source is selected as "Use_Camera", outer loop starts to use VPS information which is presented in Table 3-1 Message Table of VISION_DATA and this information obtained by runway detection algorithm which is mentioned at chapter 3. VPS has also validity information such that if runway can be detected by runway detection algorithm, this validity flag is set to true otherwise set to false. If validity is VPS is true, outer loop update position information relatively. However, if validity is false, outer loop keep last generated altitude, speed and heading reference up to valid VPS data comes. While the valid VPS information comes, reference values are also continued to update. Withdraw of using VPS is left to operator/pilot as a precaution. Operator/pilot may switch to GPS to continue landing or may cancel landing. Related algorithm can be investigated in Figure 4-35.

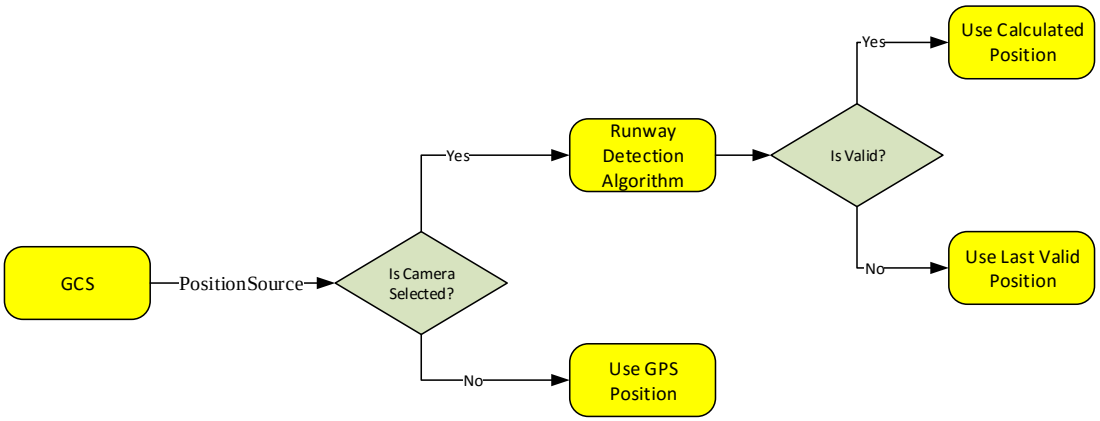


Figure 4-35 Position Source Selection Algorithm

4.4.1 Altitude and Airspeed Reference Calculation

After loading flight plan by GCS to the landing autopilot, outer loop generate altitude and airspeed reference values obtained by using PCHIP method. Interpolation is done with respect to distance to touch down point (TDP) and related altitude of airspeed value of the waypoints. To illustrate the mechanism, a flight plan is prepared for altitude and airspeed as in Table 4-11 to implement PCHIP and piecewise linear interpolation. Result of interpolations are presented in Figure 4-36 and Figure 4-37.

Order	Distance to TDP (m)	Altitude (ft)	Speed (kts)
WP_1	-6000	1300 ~396 m	90
WP_2	-4500	800 ~244 m	88
WP_3	-3000	600 ~183 m	82
WP_4	-1500	150 ~46 m	80
TDP	0	5 ~2 m	79

Table 4-11 Example of a Flight Plan Instance

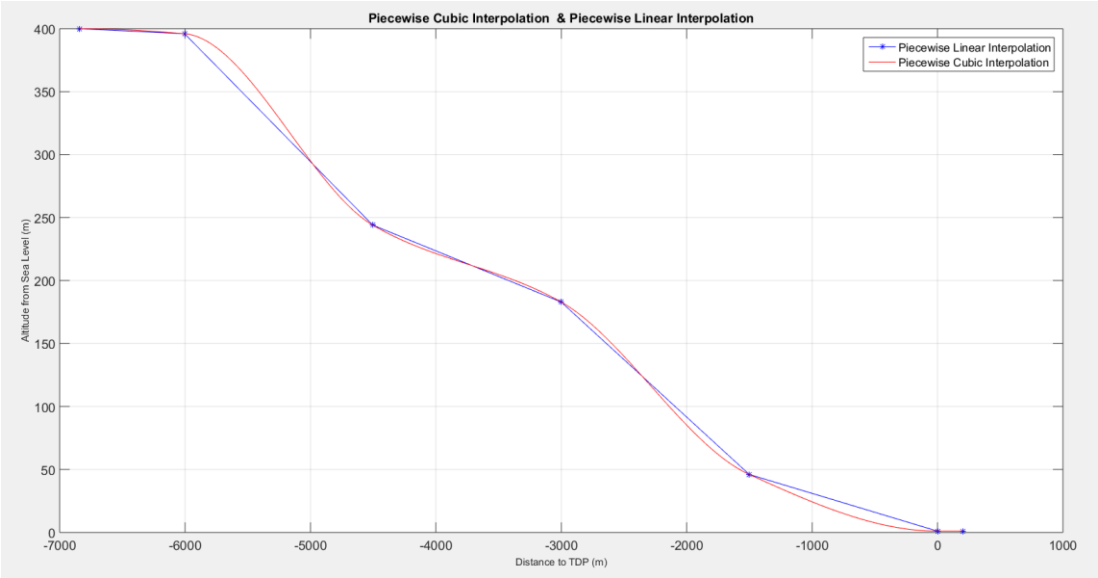


Figure 4-36 Altitude References with respect to Table 4-11 Generated by Using PCHIP and Piecewise Linear Interpolation

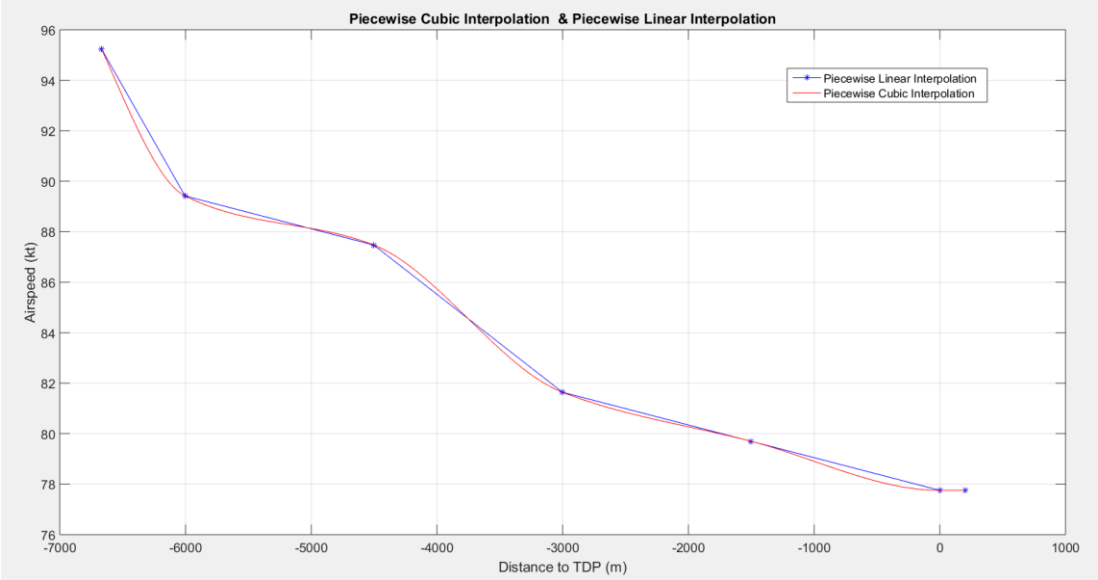


Figure 4-37 Airspeed References with respect to Table 4-11 Generated by Using PCHIP and Piecewise Linear Interpolation

Here, advantages of the PCHIP over piecewise linear interpolation is understandable more clearly. Smooth transition from one waypoint to another ensures a flyable, feasible target references and overshoots of the output of the inner loop will be minimized. An imaginary waypoint is added to avoid ambiguity in the references

through last waypoint for both altitude and airspeed interpolations. This imaginary waypoint has same properties with the last waypoint except its position. It is positioned beyond the TDP towards positive direction. It does not matter how far the imaginary point from TDB is because this point has same altitude and airspeed value. Therefore, δ_{k+1}= 0 which means that distance of imaginary waypoint to TDP has no effect on interpolation result. Addition of imaginary waypoint to flight plan is an internal property of the outer loop and this logic does not have any effect on user or GCS.

Another reason to add an imaginary waypoint to flight plan is to land the UAV on the runway such that the back landing gears touch the ground first, with a low speed and the low vertical velocity at the end of the landing phase. Rate of change of altitude reference, which directly has an effect on $\dot{h}$, decreases while approaching to the TDP to make $d_{TDP} = 0$.

Calculation of altitude and airspeed references are interpolated by using related waypoint values and distance to waypoint from TDP. Here, position of the UAV must be identified. Designed landing autopilot may obtain position of the UAV from GPS and VPS and references are fed to inner loop with respect to this distance value.

Distance Calculation by Using GPS

Distance between two coordinates can be easily calculated by using Haversine formula [37] as follows:

$$\Delta\varphi = \varphi_1 - \varphi_2$$

$$\Delta\lambda = \lambda_1 - \lambda_2$$

$$a = \sin^2\frac{\Delta\varphi}{2} + \cos\varphi_1 * \cos\varphi_2 * \sin^2\frac{\Delta\lambda}{2}$$

$$c = 2 * tan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right)$$

$$distance = R * c$$

Where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6371000 m)

Distance Calculation by Using VPS

Distance is calculated with respect to runway start line by VPS. Therefore, distance from runway starting point to TDP must be added to calculation as illustrated in Figure 4-38. Runway width in pixel, angle of edges in degree and focal length are obtained from image processing algorithm as mentioned at chapter 3. Selected runway for landing is 10L of San Francisco International Airport (KSFO) whose width is 61 m.

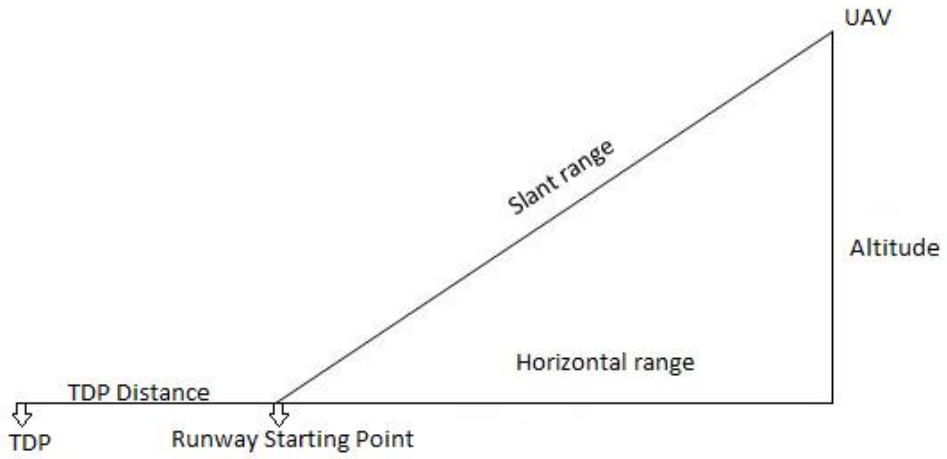


Figure 4-38 Distance Calculation by Using VPS Data

By using VPS data comes from image processing algorithm, distance can be calculated by following formula:

$$\text{Slant Distance} = \left(\frac{(FocalLenght * RunwayActualWidth)}{RunwayPixelWidth}\right)$$

$$\text{Horizontal Range} = \sqrt{(\text{Slant Distance})^2 - Altitude^2}$$

$$Distance = \text{Horizontal Range} + (TDP\ Distance)$$

Measured distance from image processing algorithm is hypotenuse distance from the UAV to runway starting point. Therefore, by using barometric altitude data and altitude of the TDP, horizontal range is calculated with basic triangle length rules.

Error ratio in distance measurement from runway is dependent on actual distance to runway. Error due to camera resolution is accepted to be same but while approaching to runway, error rate of camera resolution is decreasing. In Figure 4-39, error of distance measurement in percentage to one pixel error measurement of runway width is presented. As can be seen in Figure 4-40 also, when runway width is 10 pixel, 1 pixel error result 6.56% or 435.5 m error for distance calculation. However, when runway width is 128 pixel, 1 pixel error result 0.14% or 3.17 m error for distance calculation.

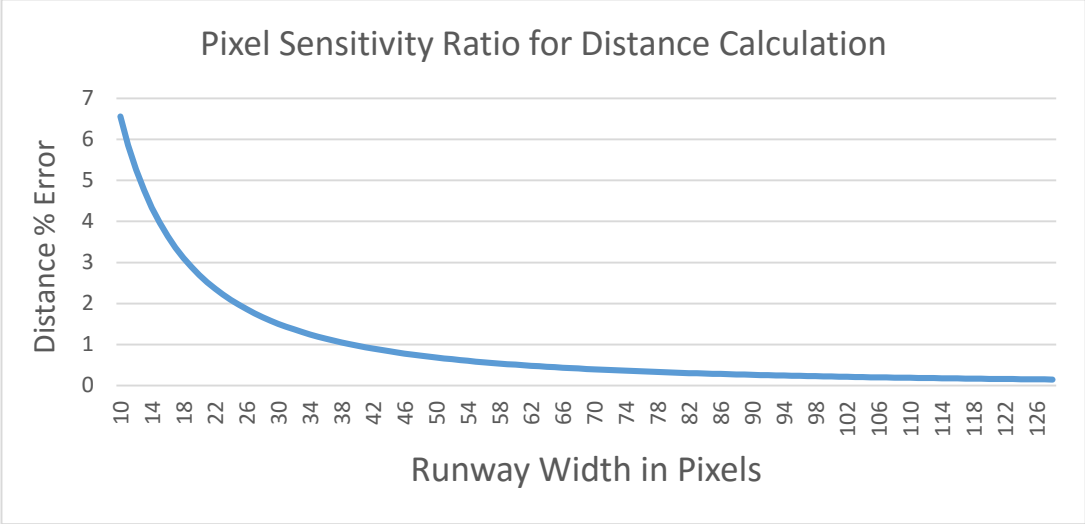


Figure 4-39 Error rate in Distance Calculation for 1 Pixel Error While Altitude of 500 Meters

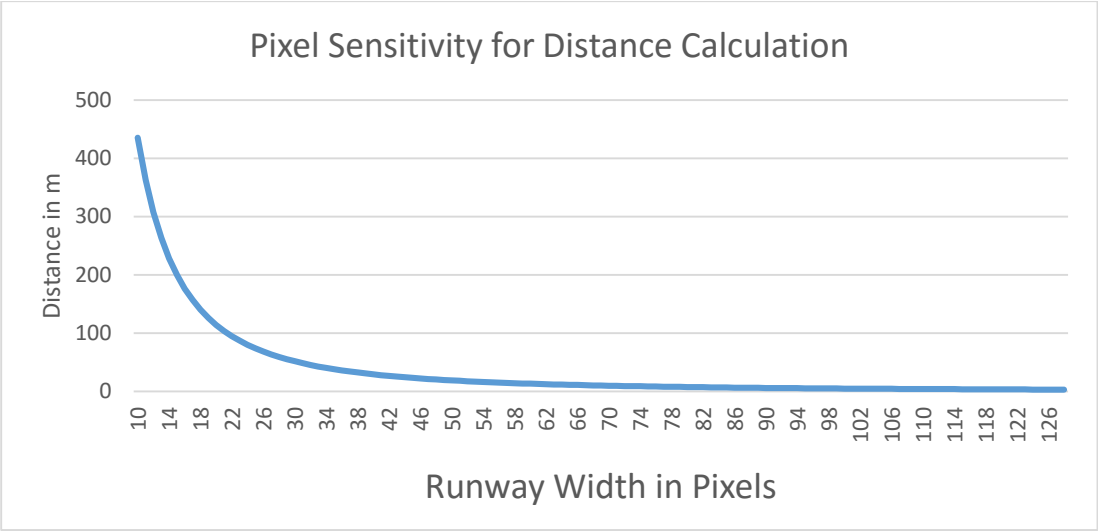


Figure 4-40 Error Value in Distance Calculation for 1 Pixel Error While Altitude of 500 Meters

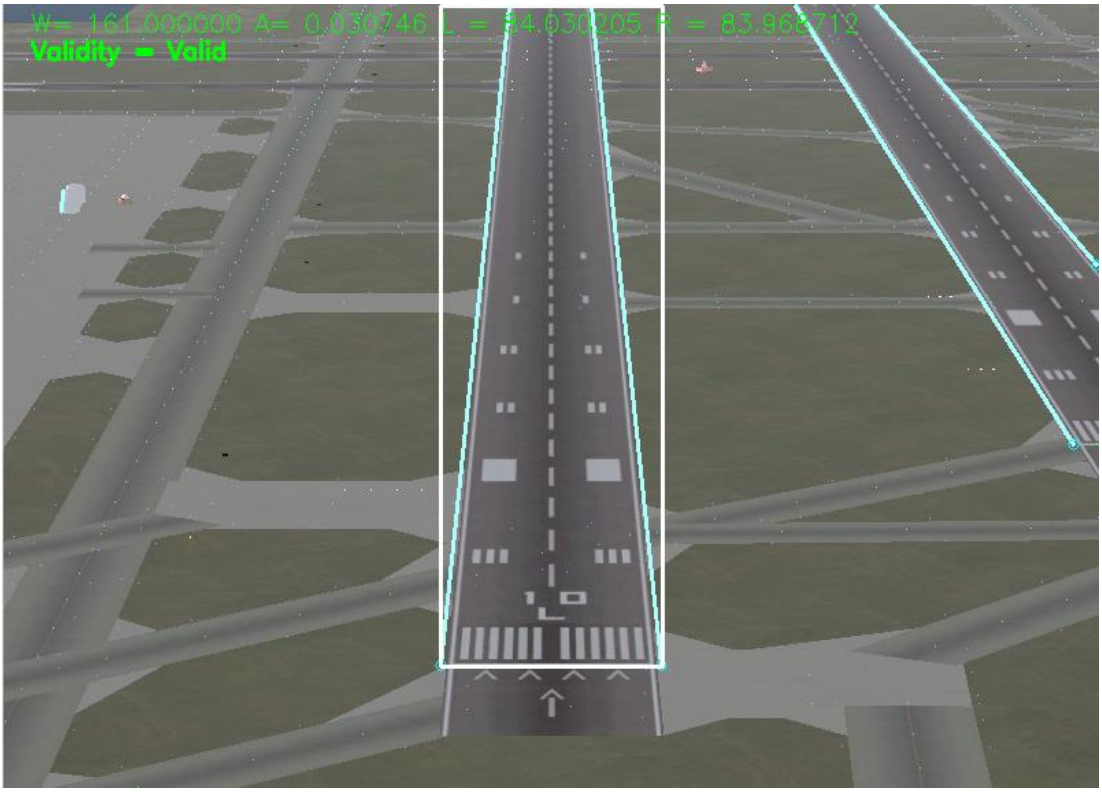


Figure 4-41 Distance Calculation Case 1

Figure 4-42 Distance Calculation Case 2

Figure 4-43 Distance Calculation Case 3

VPS is designed to calculate distance from starting line of the runway to UAV. Therefore, there is a restriction in the calculation of distance due to limited camera field of view. This restriction is illustrated in Figure 4-42. When bottom of the runway is out of camera vision, calculated distance represent distance between UAV and remaining runway in camera frame. However, we are concerned with actual distance between UAV and runway starting line as illustrated in Figure 4-41. Due to this constraint, while creating a flight plan, last waypoint before TDP must be selected such that position of it must be longer than the distance between TDP and runway starting line.

Although VPS has minimum distance constraint, UAV distance to TDP must be calculated for landing. Therefore, after last waypoint is passed, distance to TDP is calculated by using air speed and IMU rotation information of the UAV as follows:

$$Distance = BodyV_x * \cos(Theta) * SimulationCycleTime$$

Where $SimulationCycleTime$ is autopilot cycle time which is 20 milliseconds. This mechanism is used if runway starting line is within the 10% of bottom pixels height of the frame while altitude is lower than the altitude $1.1 * (TDP\ Distance) * \tan(3°)$ or width of the runway overflows from the image frame as illustrated in Figure 4-43.

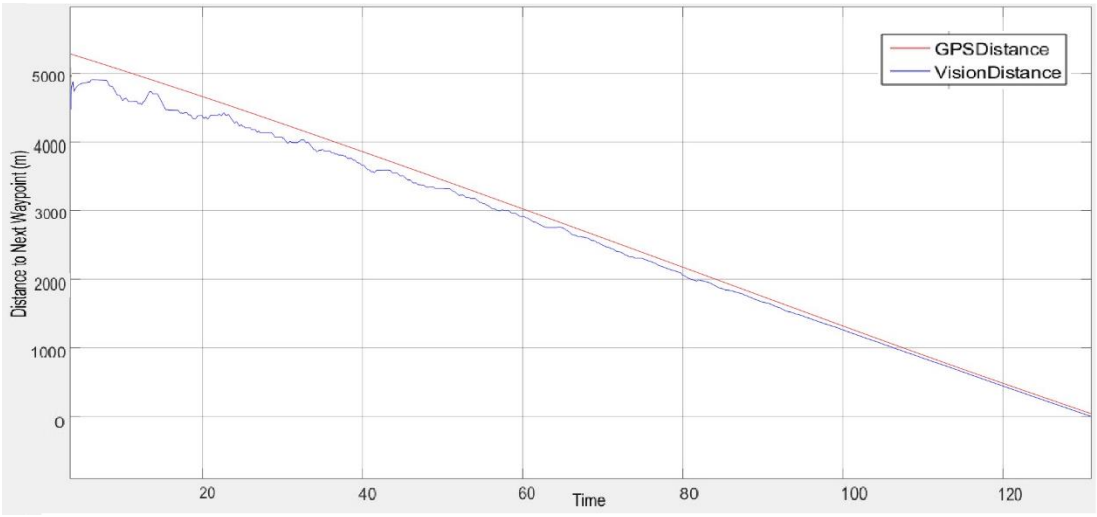


Figure 4-44 Calculated Distance to TDP by GPS and VPS

As can be investigated in Figure 4-44, VPS and GPS distances are close to each other. Also after minimum distance exceeds, distance to TDP is calculated by airspeed but this is not noticeable because this period is very short as $(TDP\ Distance) = 140\ m$ and altitude is 26 feet above ground level. Meaning of close distance measurements of VPS and GPS is that, interpolation results would be close while generating altitude and airspeed references to inner loop. Also while switching to GPS to VPS or vice versa, jumps or discontinuity in these references would be small.

VPS send width of the runway in pixel and this information is used to calculate distance to TDP. Due to resolution of the camera, when UAV far away from the runway, calculated distance error over pixel is very high as seen in Figure 4-44. For example if runway width is 10 pixel, then 1 pixel error cause high distance error. This effect can be seen on altitude reference generation in Figure 4-45.

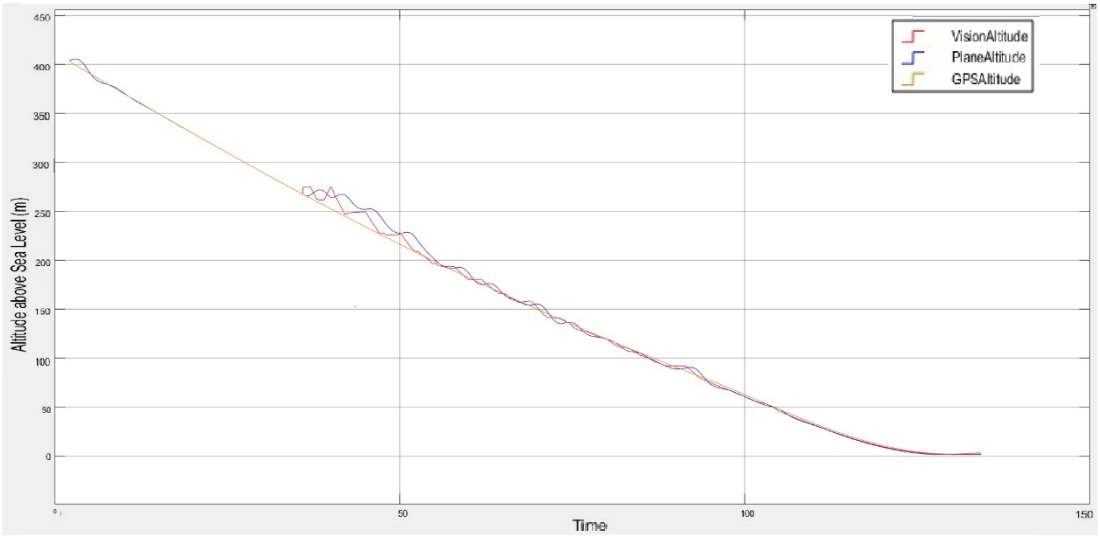


Figure 4-45 Unfiltered Altitude Reference Generated Using VPS

Due to previously explained error rate change behavior according to distance to the runway, generated altitude and speed references by using PCHIP are applied average sum filter. Result is represented in Figure 4-46.

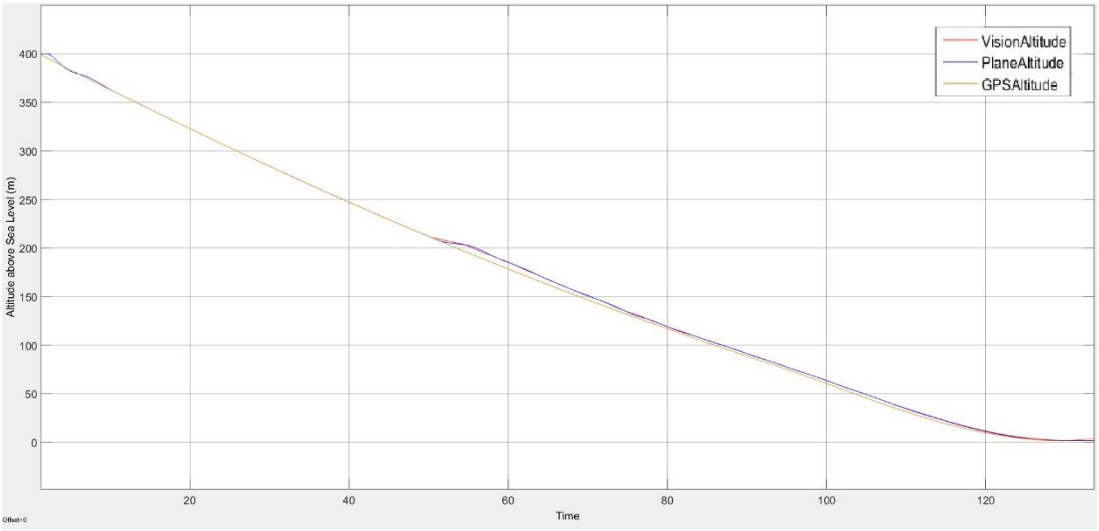


Figure 4-46 Filtered Altitude Reference Generated Using VPS

To generate airspeed reference, exactly same algorithm is applied which is used for altitude reference calculation. For speed reference calculation, altitude values is interchanged with speed values of waypoints.

4.4.2 Course Controller

Course controller calculate heading reference to feed inner loop heading controller and aims to keep cross track error of the UAV minimum which is set by loaded flight plan. Besides keeping cross track error minimum, course controller must keep crap angle within the required range as stated in Table 4-2 “At Touchdown” requirements. Aim of this requirement is to avoid large lateral movements when wheel of the UAV touches the ground and not damage the landing gears also.

Under no crosswind condition course direction is equal to body heading value of the UAV, however, under crosswind condition, these values can be different to each other. When crosswinds exist, tracking direction and heading values are different from each other. For example, when crosswind is blowing in y direction, to be able to hold tracking, UAV heading must be directed towards to wind direction as illustrated in Figure 4-48. However, before touch down, error between UAV heading and track angle must be as small as possible because big differences can cause crashes when tires touch to the ground. This maneuver is called as decrab maneuver and sequence of the maneuver is illustrated in Figure 4-47.

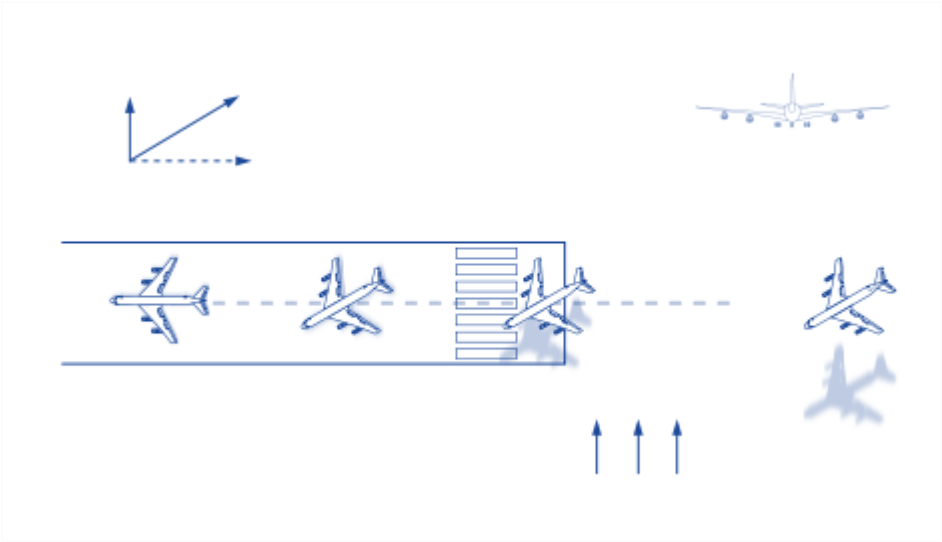


Figure 4-47: Landing maneuver sequence While Crosswind is Exist

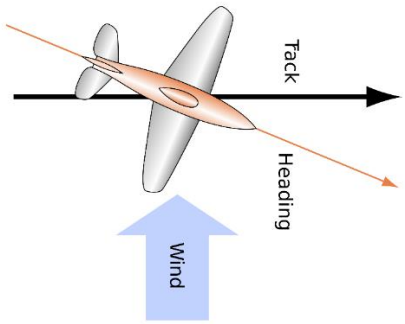


Figure 4-48 Course (Track) Hold While Crosswind is Exist

If position source is selected as GPS, cross track error can be identified by using coordinates of the UAV and target waypoint and course controller aims to decrease cross track error. However, if VPS is selected as position source, then cross track error is measured by angle of two edges of the runway.

Course Control by Using GPS

Once GPS is selected for position data source from GCS interface in Table A - 1, outer loop would use GPS information to determine course output. Outer loop calculates direction from current position of the UAV to next waypoint position and try to change heading value to minimize bearing error as illustrated in Figure 4-49.

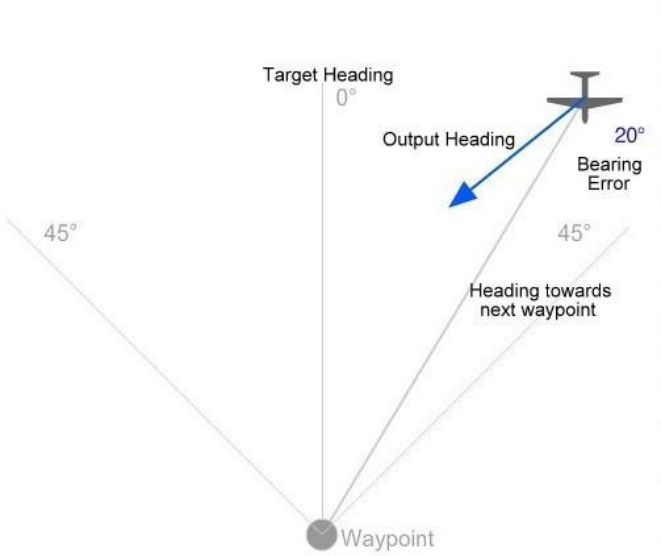


Figure 4-49 Bearing Error and Line Fit Algorithm

For the initial run of outer loop, 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, course controller works to make bearing error as small as possible with respect to heading value from first waypoint to TDP. In other word, 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 align with runway to manage safe landing.

Related logic is presented in Figure 4-50. As can be seen here, if number of waypoint is bigger than two, course controller starts to work after reaching the first waypoint. Until then, heading to first waypoint is feed to inner loop. After reaching to first waypoint, direction from last reached waypoint to target waypoint is calculated and controller works to minimize difference between target direction and current direction of the UAV. Target directions comes with flight plan and calculated by GCS interface. Heading calculation between two coordinates is done with following rules:

- If latitudes and longitudes are same, heading is zero.
- If longitudes is same and latitude of first waypoint is bigger than second one, heading is 180°, else heading is zero.

- If latitudes and longitudes are different from each other, then heading value is:

$$Var1 = acos\Big(sin(lat2) * sin(lat1) + cos(lat2) * cos(lat1) * cos\big((lon2 - lon1)\big)\Big)$$

$$Heading = asin\left(cos(lat2) * \frac{sin\big((lon2 - lon1)\big)}{sin(Var1)}\right)$$

Same algorithm is used for GCS interface to calculate target heading values that is described at Appendix A.

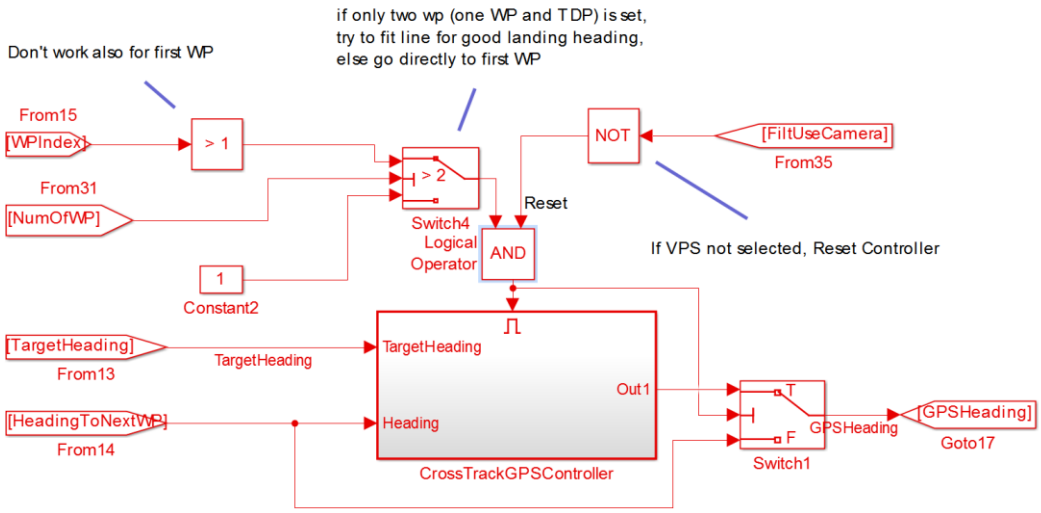


Figure 4-50: GPS Cross Track Controller

Calculated heading is feed to inner loop controller reference. PI controller is used to obtain correction value for calculation of output heading by using cross track error value. Calculated correction value subtracted from current heading to next waypoint to minimize bearing error as soon as possible. Details can be seen in Figure 4-50 and Figure 4-51.

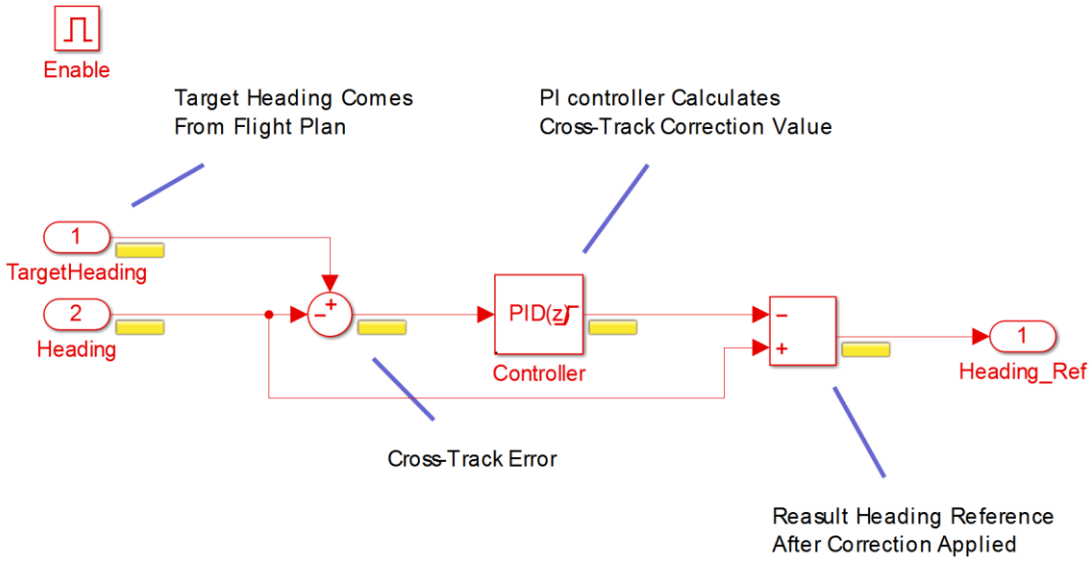


Figure 4-51: Cross Track Controller Algorithm

Cross track controller is test with following initial conditions,

- Target Heading is 119.9 degree,
- Heading to next waypoint is 134.2 degree, so cross-track error is 16.3 degree.
- Distance to next waypoint is 4788 m.

Result of course controller step response is presented in Figure 4-52. With respect to the result, Course has about %12 overshoot which corresponds to about 2 degree which is acceptable. Settling is established where 1894 meter and 42 second before to reach target waypoint. Cross track performance is sufficient for a landing purpose mission.

In Figure 4-52, “CalculatedHeadingRef” is output of cross track controller which is feeding to Heading Controller of inner loop, “Cross Track” is target cross track which is obtained from flight plan, “PlaneHeading” is heading of UAV (psi), and “Heading2NextWP” is direction of UAV coordinate to target waypoint coordinate.

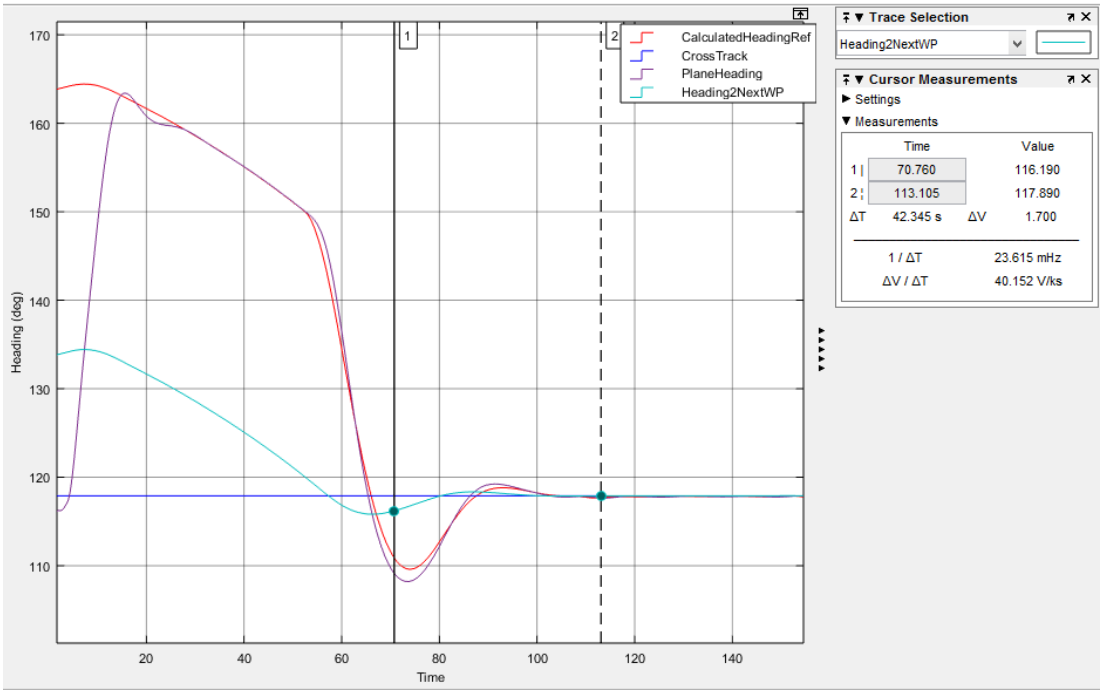


Figure 4-52 Step Response of Cross Track Controller

Under crosswind conditions, although cross-track angle is zero, heading value may differ from cross-track direction as illustrated in Figure 4-48. However, with respect to “At Touchdown” requirement in Table 4-2, decrab maneuver is needed whatever the crosswind is. Decrab maneuver is obtained such that, after 5 m altitude, inner loop is feed by target heading value which is obtained from flight plan and this value is equal to runway heading value. This switch can be investigated at the end of the simulation under 15 kts side wind condition presented in Figure 4-53. At the end of the decrab maneuver, decrab angle is 0.36 degree and so requirements in Table 4-2 is satisfied successfully.

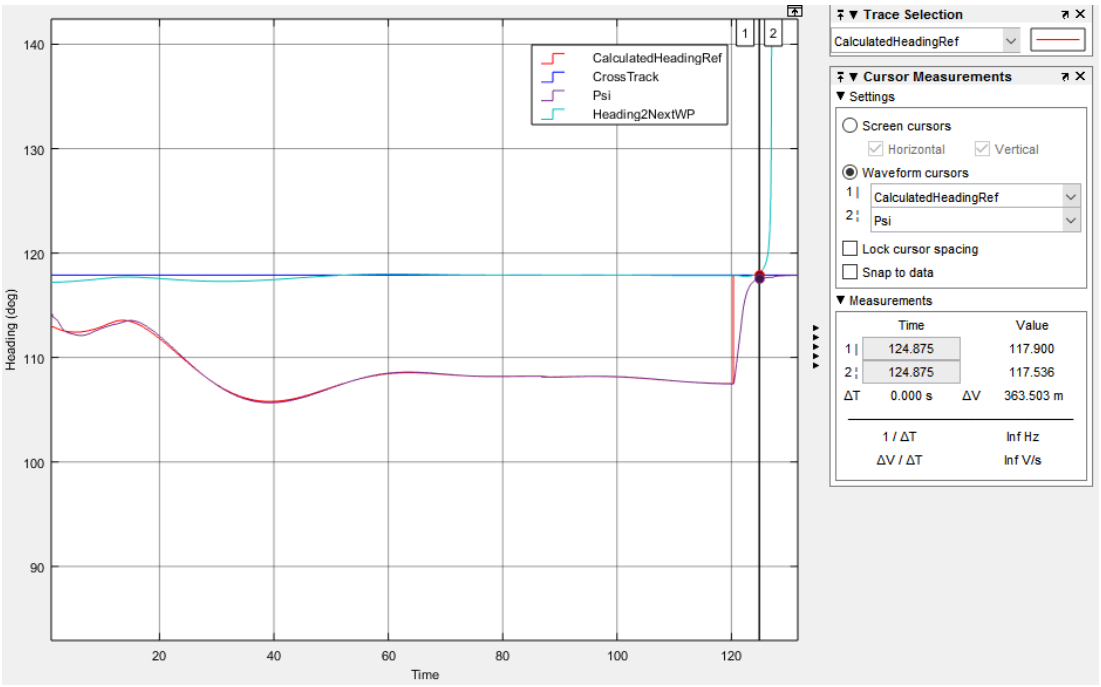
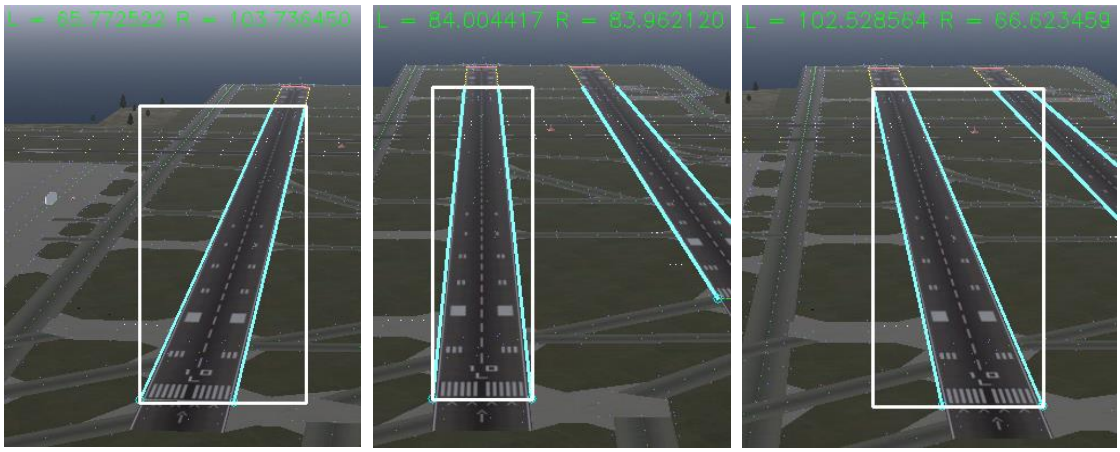


Figure 4-53 GPS Decrab Maneuver under 15 kts Side Wind

Course Control by Using VPS

Once VPS is selected for position information source, cross track error of UAV is calculated from angle of two edges of the runway like [38]. When UAV aligned with the runway, angle difference between 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 as can be investigated in Figure 4-54 and Figure 4-55. So by a PI controller, heading reference position is generated to feed Heading Controller of inner loop.

Figure 4-54 Runway Orientations. L is Angle of Left Edge, R is Angle of Right Edge



(a) (b) (c)

Figure 4-55 Three Condition To Calculate Heading Reference. (a) UAV Is Right Side of the Runway, (b) UAV Is In Front of the Runway, (C) UAV Is Left Side of the Runway

As can be seen in Figure 4-55, for condition,

(a) UAV is right side of the runway and L > R. So UAV must turn to left,

(b) UAV is in front of the center of the runway and L = R. So UAV must flight straight,

(c) UAV is left side of the runway and L < R. UAV must turn to right.

Difference between left and right angle of the runway edges may occur when UAV is misaligned with the runway or when heading of the UAV is different than runway heading direction. By using error between right and left angle of the runway, both

misalignment in position and heading orientation of the UAV are corrected. Effect of the heading on runway edge difference is presented in Figure 4-56.

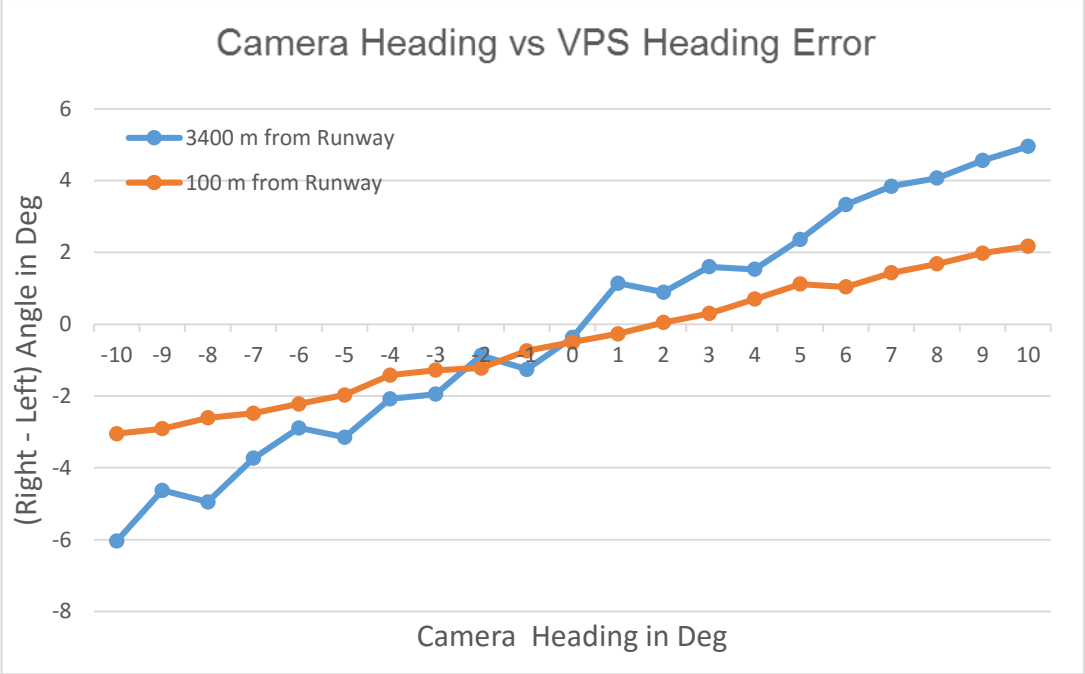


Figure 4-56 Effect of Heading Change on (Right - Left) Angle of the Runway at 100m and 3400m away from Runway Threshold

As can be seen in Figure 4-56, error due to body heading change is bounded and range of the resulted error gets smaller while approaching to runway. Also effect of misalignment is always much more dominant than body heading while calculating error between right and left angle difference of the runway. Also, due to faster change in body heading, error due to this parameter is eliminated faster. Briefly, whatever the body heading direction is, if right and left angle is close to each other, it is guarantee that UAV will be land on runway.

Same noisy data encounter in distance calculation emerges here also. Angle of runway edges comes from VPS have noisy characteristics so need to be filtered before using. Therefore, an average sum filter is used to smooth data. As mentioned at Chapter 3, runway detection algorithm supplies VPS data with a validity flag.

Therefore, if validity of the VPS data is false, last valid heading output state is kept until valid VPS data comes. Course controller use angles of runway edges so it has no limitation as distance calculation has. Course controller can work until runway edges detected. However while approaching to ground, runway starts to overflow from the image frame and merge with horizon line. For this reason, if runway width in pixel is higher than 90% of image frame such as Figure 4-43, last valid state is kept. This condition is accepted such that UAV is about to land so keeping last valid state up to touchdown is reasonable. PI controller is used to generate course reference to Heading Controller of the inner loop controller and this reference angle is limited between [-15 15] degree from target cross track angle. Clamping anti-windup method is applied to limit the integrator effect on block saturation. Controller algorithm is presented in Figure 4-57.

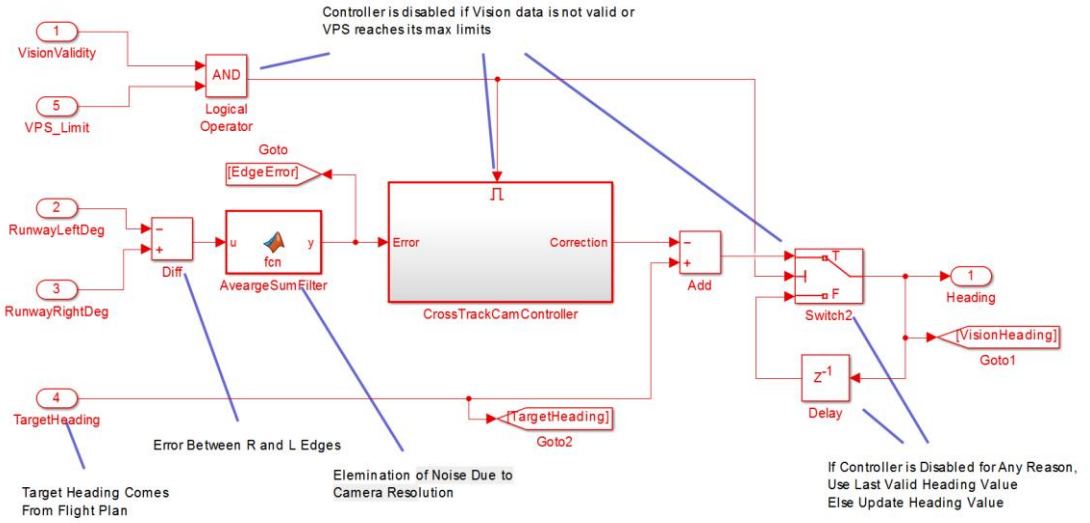


Figure 4-57: Vision Cross Track Controller

Step response of vision cross track algorithm can be investigated in Figure 4-58. Algorithm is start to work with -74 degree edge difference error. There is no significant overshoot or oscillation in error or controller output.

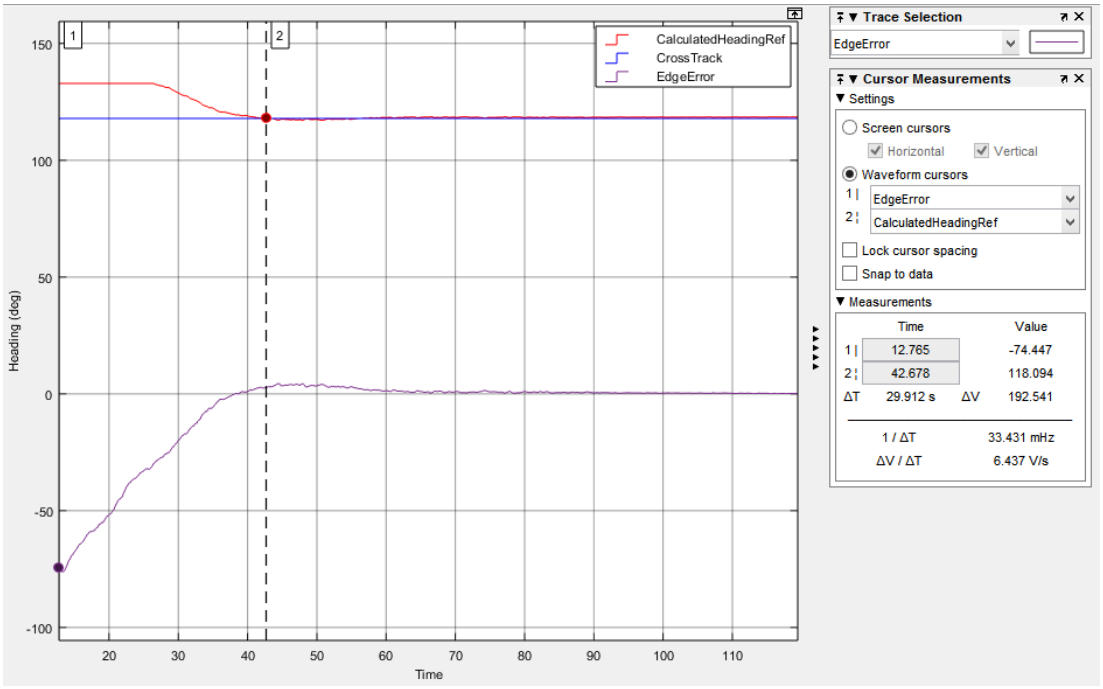


Figure 4-58 Step Response of Cross Track Controller of Vision Based Positioning Algorithm

As in GPS case, under crosswind condition, heading value may differ from cross track direction although angles of right and left runway edges do not differs much. To make decrab maneuver, after altitude of the UAV get lower than 5 meters, heading reference of the UAV is set to cross track value to touching to ground safely. As presented in Figure 4-59, “At Touchdown” requirements in Table 4-2 is satisfied successfully with 0.44 degree decrab angle under 15 kts side wind.

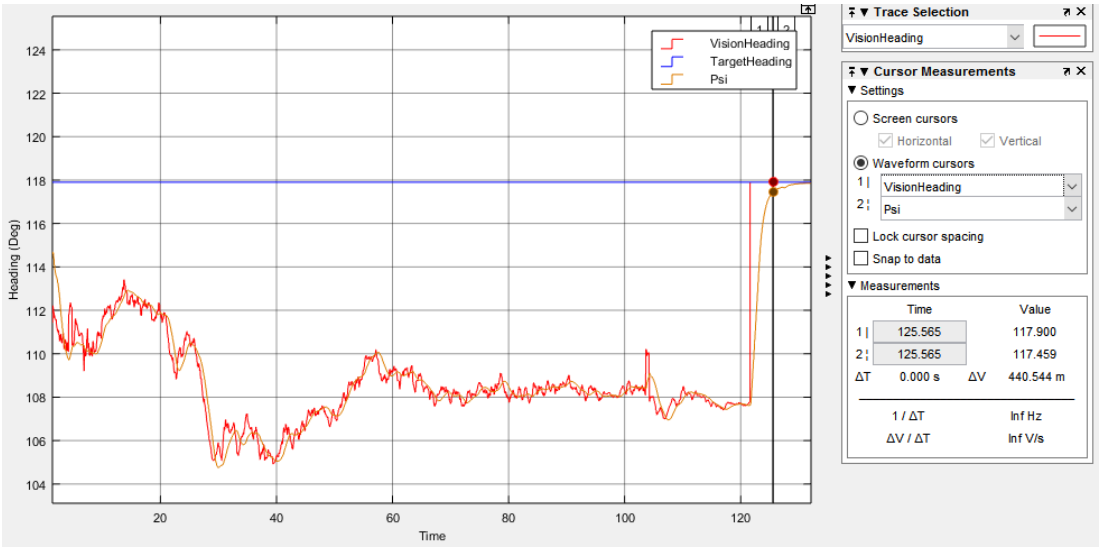


Figure 4-59 VPS Decrab Maneuver under 15 kts Side Wind

CHAPTER 5

SIMULATION AND RESULTS

5.1 Overview

The vision based landing algorithm proposed above has been developed and tested in a simulation environment where controller algorithm is implemented in Matlab/Simulink; aircraft and atmospheric model are imported from demo of "Fly the DeHavilland Beaver" [9] in Matlab/Simulink; image processing algorithms developed in C/C++; GCS user interface developed in C#; the simulated image is generated by the FlightGear flight simulator, which has 800x640 pixel resolution, and JMapView [39] open source software is used to see and track UAV position on the map. Stall speed of the aircraft model is 66 kts and the chosen airport scenario was the San Francisco international airport (KSFO). Simulation has started with a random position of the UAV.

Several simulation scenarios are applied with different weather with respect to waypoint information in Table 5-1. Aim of these simulation is to clarify that if Table 4-2 Landing Autopilot Requirements and Table 4-3 Environmental Requirements for Landing are satisfied or not.

order	Latitude	Longitude	Altitude (ft)	Heading	Speed
1	37.63381558835	-122.40539765887	266	117.9	85
2 (TDP)	37.6275	-122.39033333333	7	117.9	80

Table 5-1 Waypoints Generated for Simulations

5.2 Case 1: No Wind and Turbulence

For this case system behavior is obtained for no wind and turbulence condition. There is no cross wind exists and initial position of the UAV is on the cross track direction.

5.2.1 GPS Based Landing

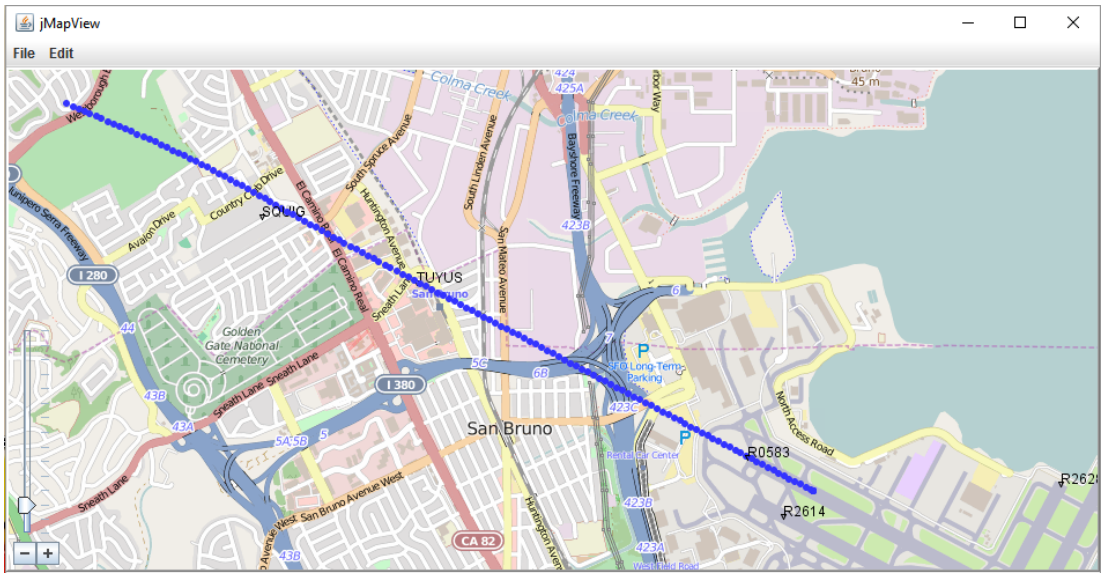


Figure 5-1 Tracking of the UAV Position While Landing for Case 1: No Wind and Turbulence GPS Based Landing

Under no wind and turbulence conditions, inner loop of the autopilot has no difficulty to keep outer loop references. As can be seen in Figure 5-1 and Figure 5-2, cross track is kept and UAV is aligned with the runway.

Figure 5-2 Vehicle Position and Orientation for Case 1: No Wind and Turbulence GPS Based Landing. Left is Pilot Cam, Right is View from Back

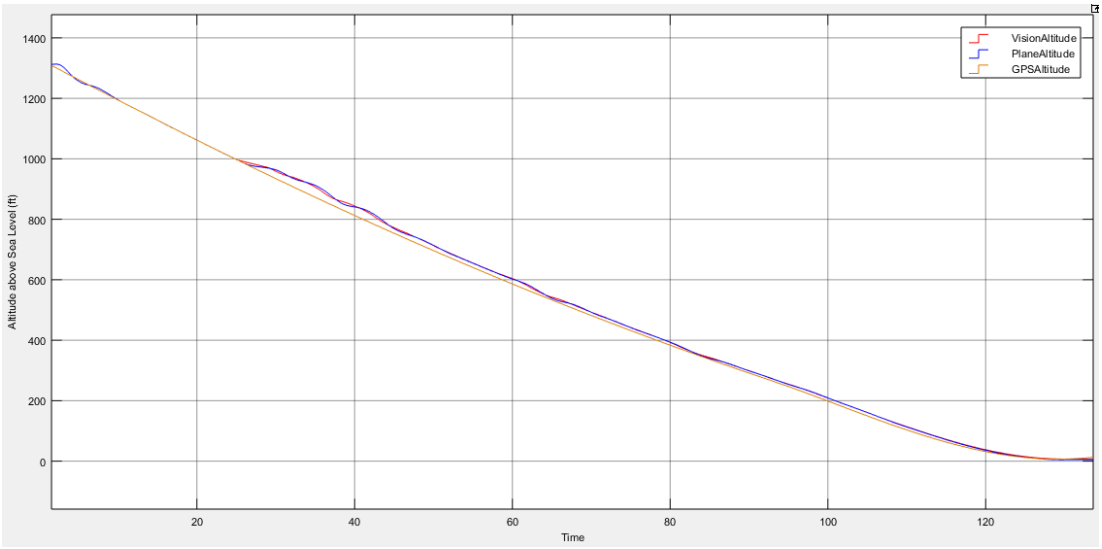


Figure 5-3 Landing Trajectory Path and Altitude of the UAV for Case 1: No Wind and Turbulence GPS Based Landing

At the beginning of the simulation in Figure 5-3, there is an oscillations in altitude of UAV. At the beginning of simulation, flight mode of the UAV is RC_MODE so no autopilot is working. During the change of the flight mode to FMS_Mode, altitude of UAV is varied depend on orientation of UAV. After enabling of FMS_Mode, altitude reference is successfully tracked by Pitch & Altitude Controller. Also rate of change of altitude is decreasing towards last waypoint which is TDP. So that flare maneuver is done to reduce vertical velocity to touch the ground softly. Shock comes

from landing gear can be investigated in Figure 5-4. Flare effect can be seen after 105 seconds. At second 126, ground interaction is established. Due to flare effect, shock comes from ground interaction is small which is essential for landing gears and avionics equipment.

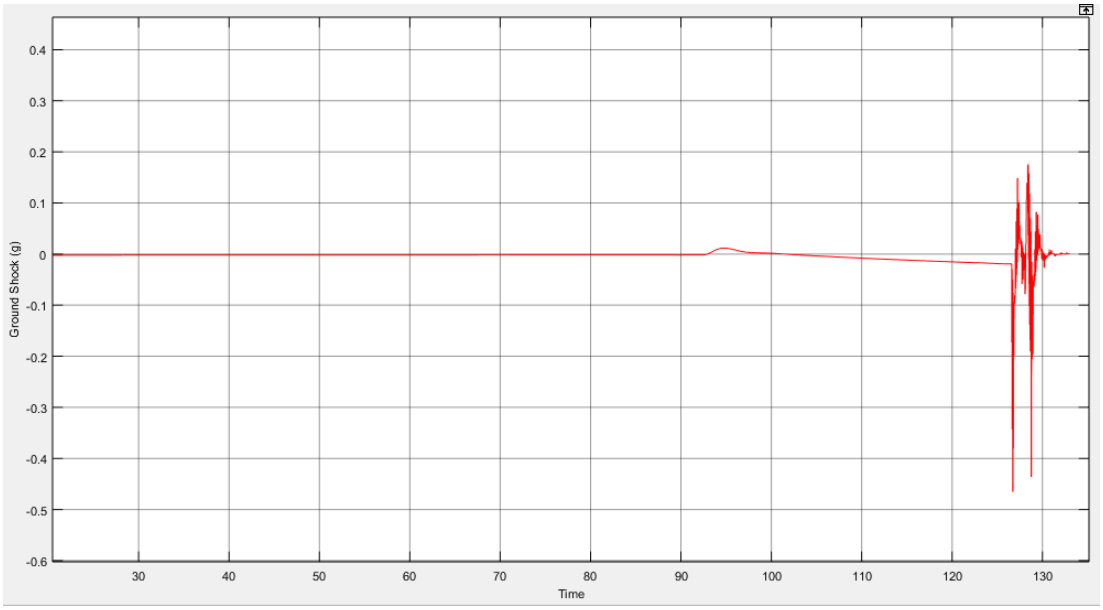


Figure 5-4 Ground Shock at Touchdown for Case 1: No Wind and Turbulence GPS Based Landing

In Figure 5-6, pitch angle variation is presented. While approaching to runway, pitch angle is command to negative values for decreasing the altitude. While approaching to TDP, pitch angle starts to increase to positive values due to flare maneuver to decrease vertical velocity and $\dot{h}$. As mentioned at previous chapter, this flare maneuver is obtained with imaginary waypoint added after TDP so that slope of the last part of the flight plan is zero. By using the property of the PCHIP, exponential flare trajectory is obtained. $\dot{h}$ is presented in Figure 5-5 and it is clearly identified that $\dot{h}$ is decreasing with the flare maneuver and "At Touchdown" requirement is satisfied by $\dot{h}$ = 1.7 ft/s which may be max -3 ft/s with respect to Table 4-2 Landing Autopilot Requirements.

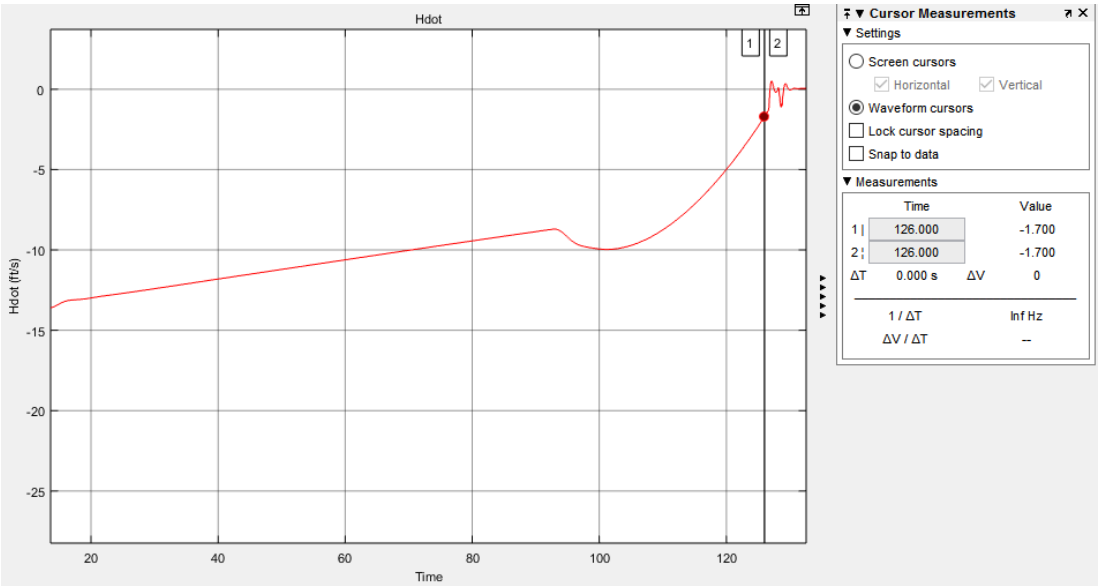


Figure 5-5 $\dot{h}$ for Case 1: No Wind and Turbulence GPS Based Landing

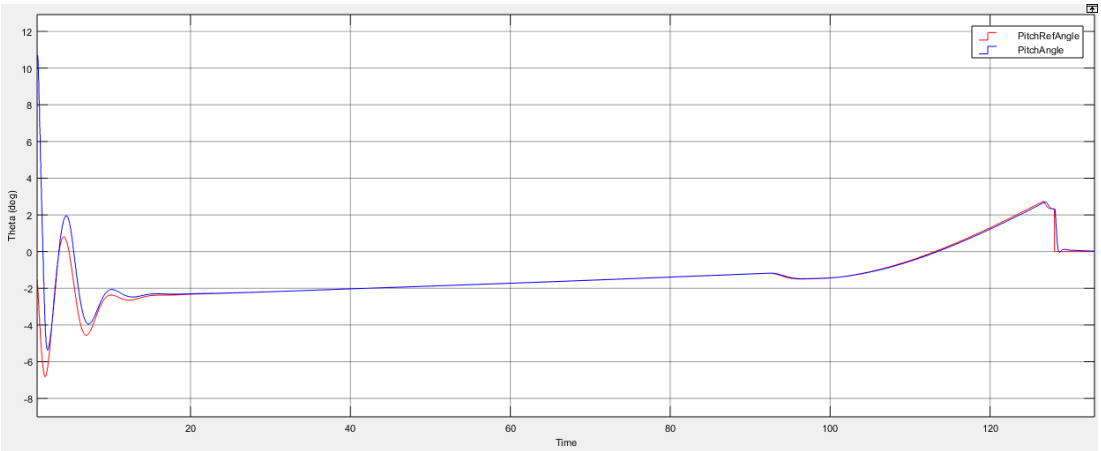


Figure 5-6 Variation of Theta for Case 1: No Wind and Turbulence GPS Based Landing

While getting closer to TDP, theta gets positive value due to flare maneuver as expected and requirements are satisfied with respect to Table 4-2 Landing Autopilot Requirements.

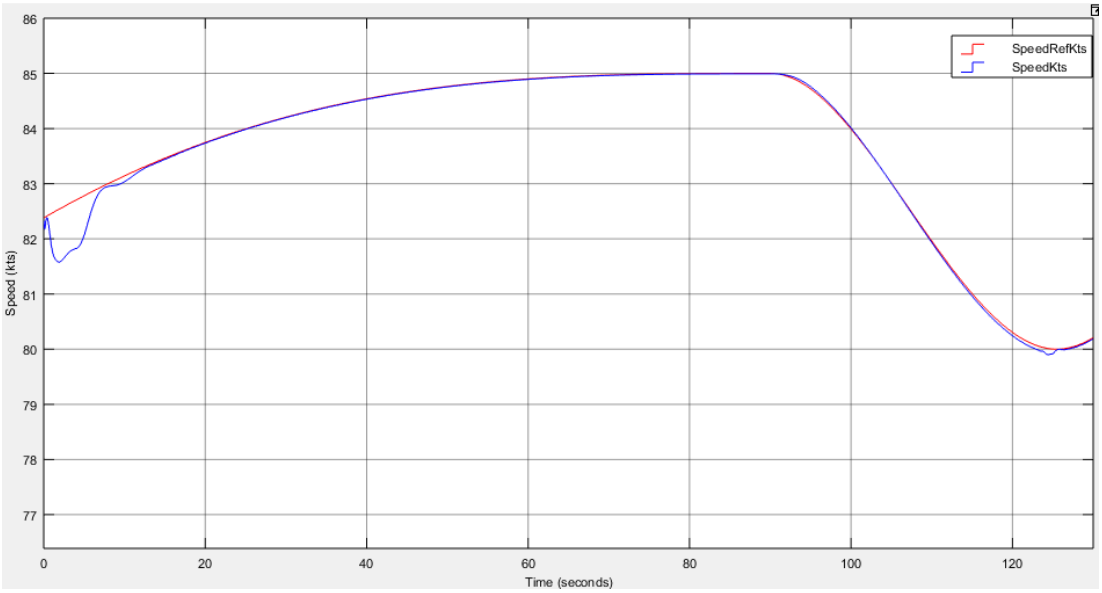


Figure 5-7 Variation of Speed for Case 1: No Wind and Turbulence GPS Based Landing

In Figure 5-7, speed of the UAV is presented. As can be seen from that figure, speed tracks flight plan and increase to 85 kts. After first waypoint is managed, speed is decrease to reach TDP speed value which is 79 kts. Flare maneuver helps to reach final speed value faster. Again Table 4-2 Landing Autopilot Requirements is satisfied successfully.

5.2.2 VPS Based Landing

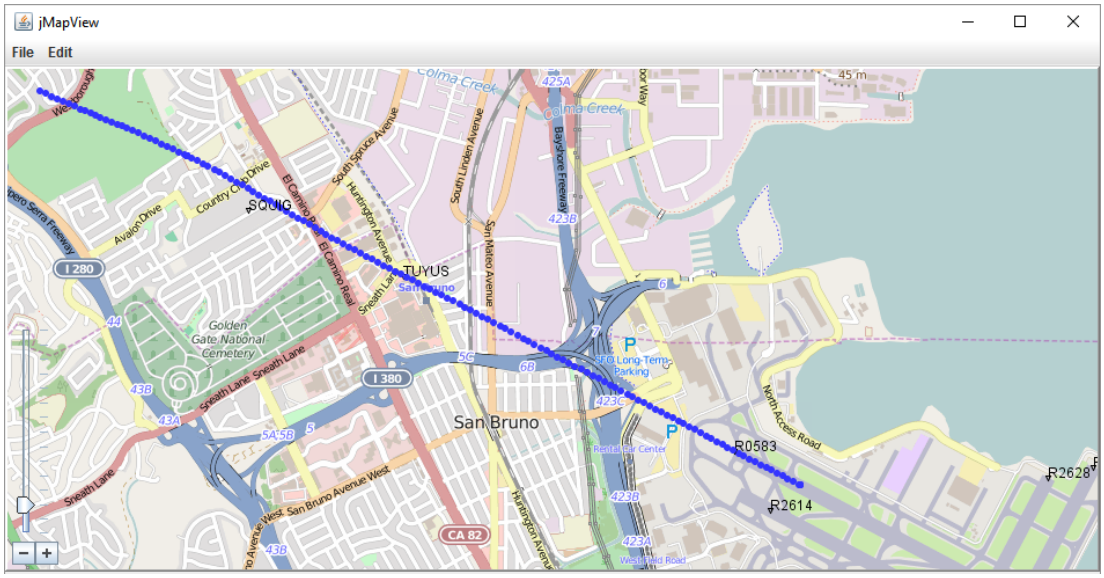


Figure 5-8 Tracking of the UAV Position While Landing for Case 1: No Wind and Turbulence VPS Based Landing

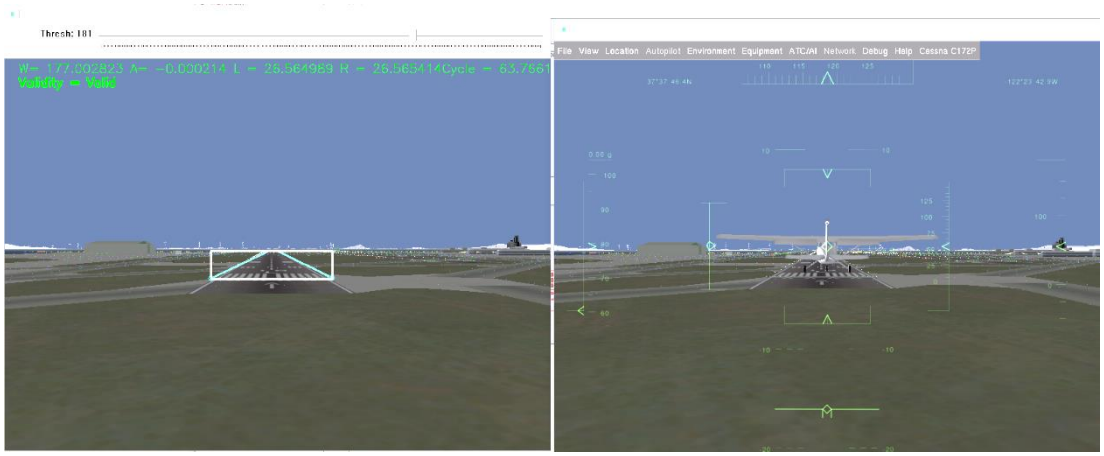


Figure 5-9 Vehicle Position and Orientation for Case 1: No Wind and Turbulence VPS Based Landing. Left is Pilot Cam, Right is View from Back

First comparison between GPS and VPS can be done by comparison of Figure 5-2, Figure 5-9 and Figure 5-1, Figure 5-6. Position and orientation of the UAV are similar both GPS and VPS active cases. Therefore, not much difference is faced with while landing compared to GPS based landing.

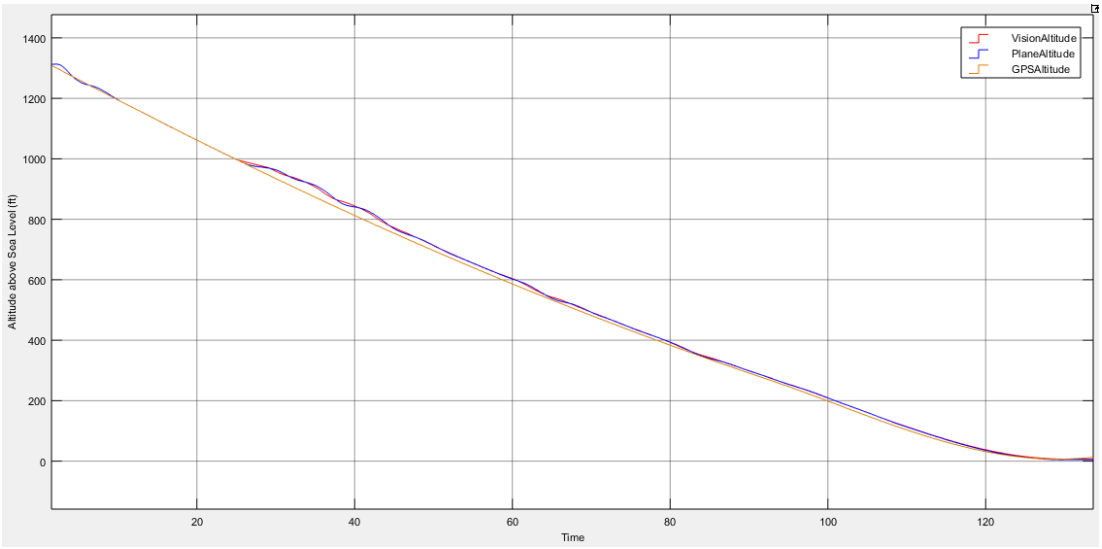


Figure 5-10 Landing Trajectory Path and Altitude of the UAV for Case 1: No Wind and Turbulence VPS Based Landing

As mentioned at Distance Calculation by Using VPS, as getting closer to runway, measurement error is decreased in distance therefore in altitude also. VPS is activated after reaching 1000 ft as can be seen in Figure 5-10 and as getting closer to TDP, error between altitude interpolation of GPS and VPS is getting smaller. Same logic is applied for flare maneuver and flare behavior at VPS is very close to at GPS case. Therefore, shock effect value is nearly same compared to GPS case. However during flight with VPS, due to measurement errors, there are oscillations in altitude reference and this effects $\dot{h}$ and its derivative which is the source of the shock actually. However, as getting closer to runway, sensitivity of error ration to measurement is getting smaller and oscillations are decreasing as presented in Figure 5-11 and Figure 5-12.

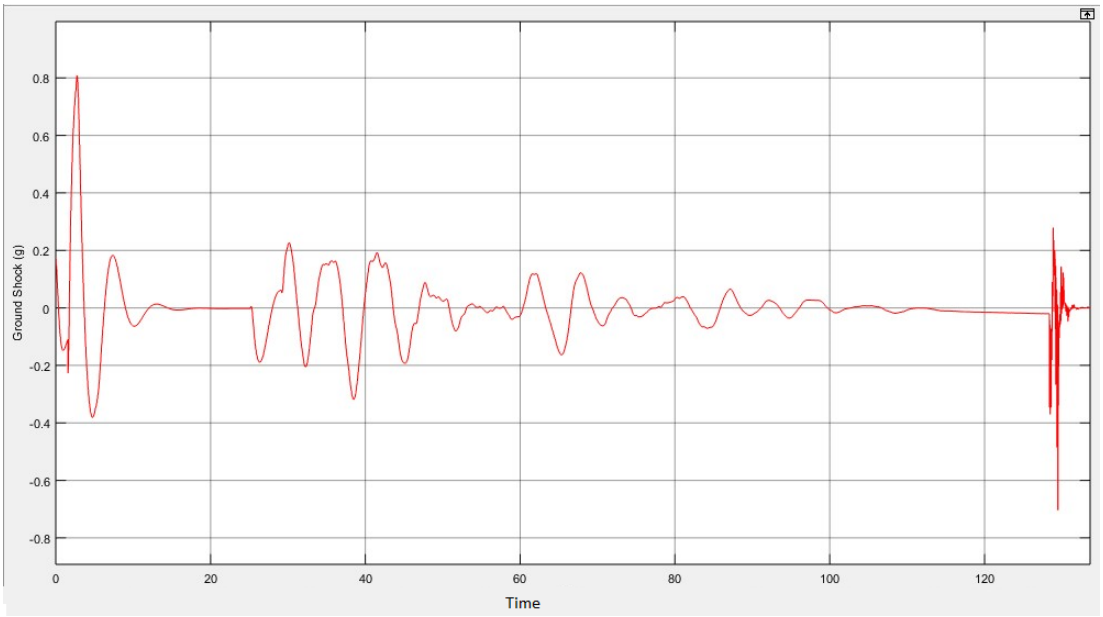


Figure 5-11 Ground Shock at Touchdown for Case 1: No Wind and Turbulence VPS Based Landing

With the flare maneuver, $\dot{h}$ is getting smaller because rate of change of altitude reference gets smaller by approaching to TDP, whose slope is zero and parallel to ground. Therefore exponential flare trajectory is generated and "At Touchdown" requirement is satisfied by $\dot{h} = 1.4$ ft/s which may be max -3 ft/s with respect to Table 4-2 Landing Autopilot Requirements.

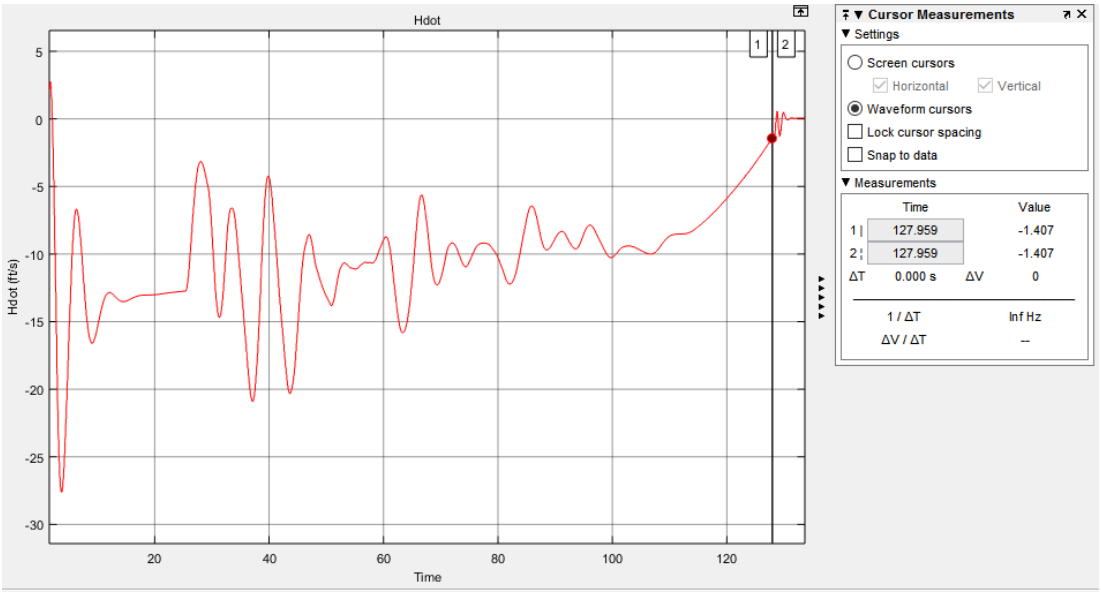


Figure 5-12 $\dot{h}$ for Case 1: No Wind and Turbulence Case 1: No Wind and Turbulence VPS Based Landing

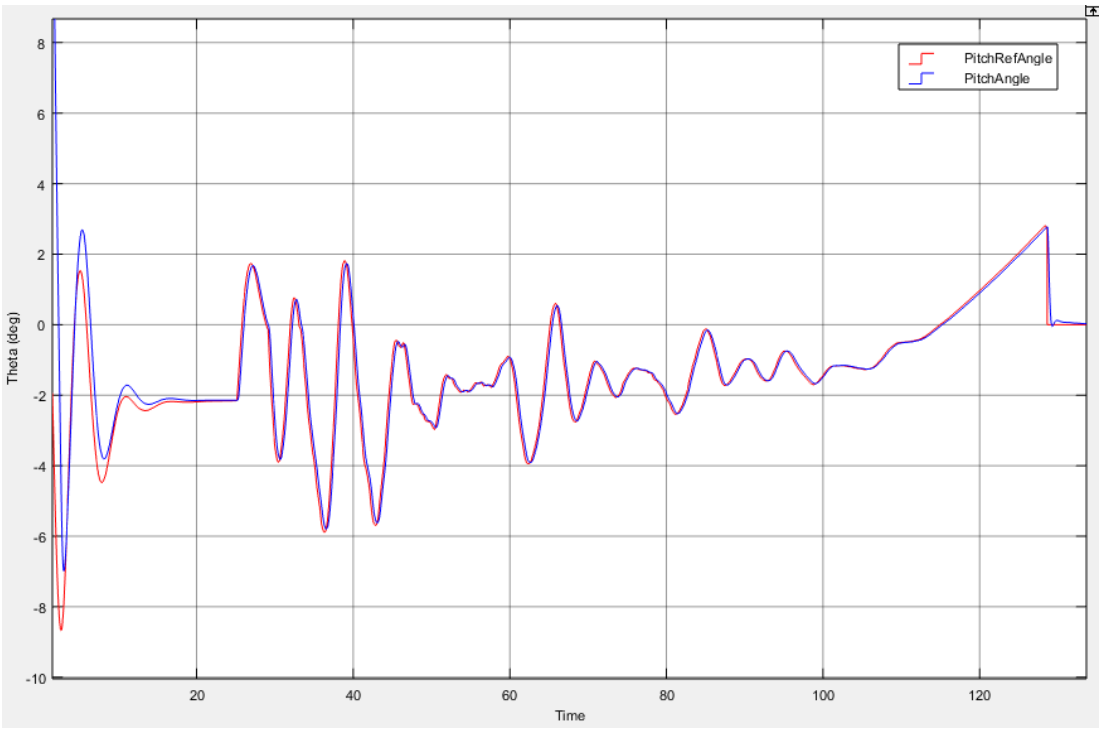


Figure 5-13 Variation of Theta for Case 1: No Wind and Turbulence VPS Based Landing

Similar sensitivity effect can be seen at pitch angle in Figure 5-13 and speed of the UAV in Figure 5-14. Due to resolution issues, as getting closer, oscillations are decreased. While approaching to TDP, quality of the pitch reference and speed reference are increasing.

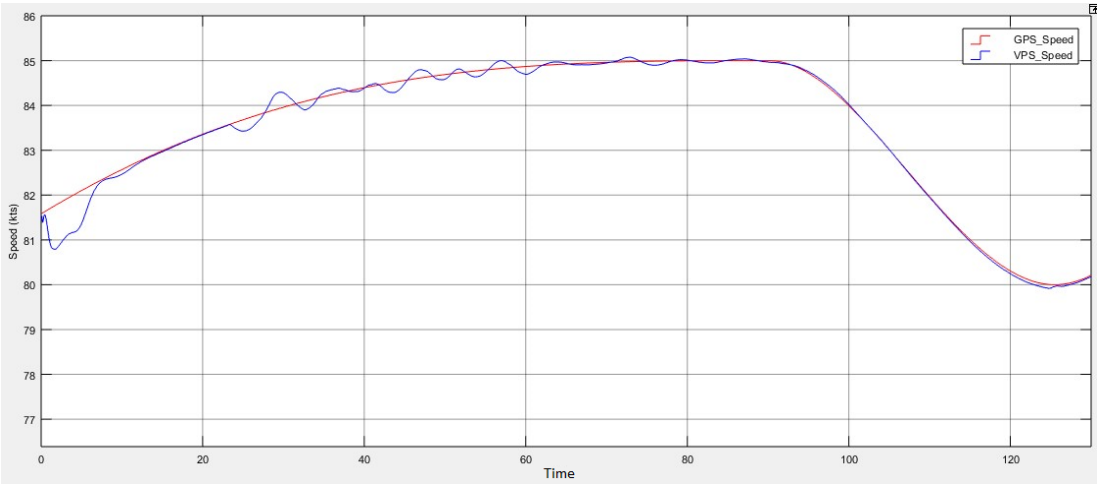


Figure 5-14 Variation of Speed for Case 1: No Wind and Turbulence VPS Based Landing

Although there is oscillations in generated references, these oscillations are decreased while approaching to runway and all requirements are satisfied in Table 4-2 Landing Autopilot Requirements without using GPS but using VPS.

5.3 Case 2: 15 kts Cross Wind Condition

For this case system behavior is obtained under 15 kts cross wind condition. Initial position of the UAV is on the cross track direction.

5.3.1 GPS Based Landing

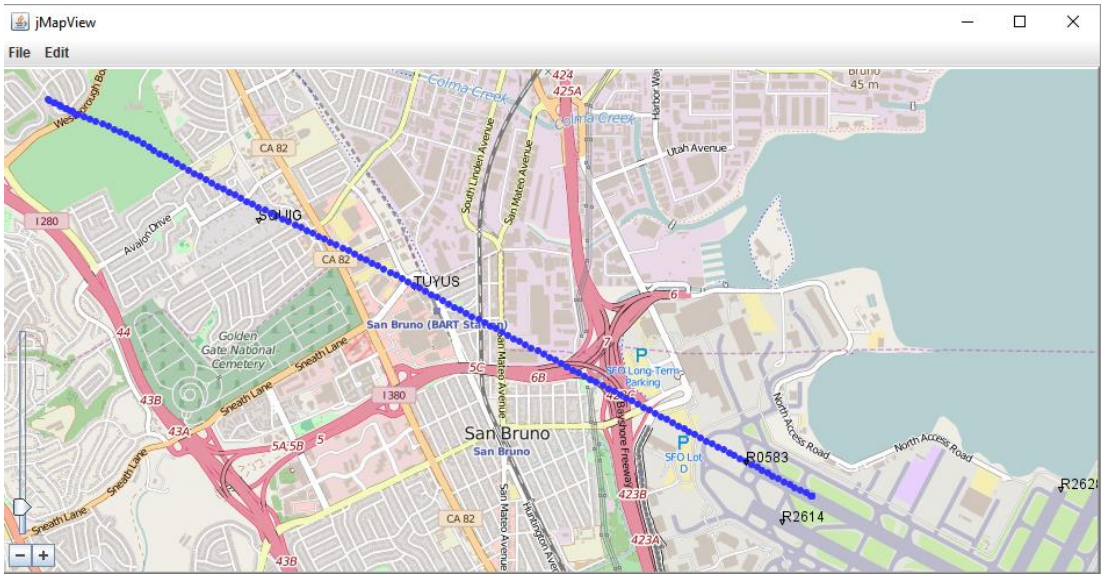


Figure 5-15 Tracking of the UAV Position While Landing for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Under 15 kts crosswind condition, Course Controller of outer loop and Heading Controller of inner loop keep cross track successfully. No significant effect observed on cross track position of the UAV compared to Case 1: No Wind and Turbulence in Figure 5-1.

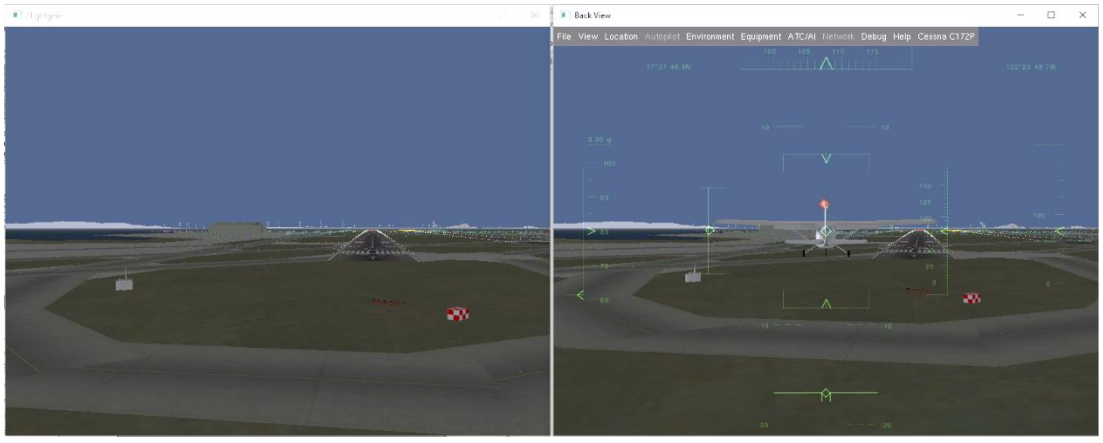


Figure 5-16 Vehicle Position and Orientation for Case 2: 15 kts Cross Wind Condition GPS Based Landing. Left is Pilot Cam, Right is View from Back

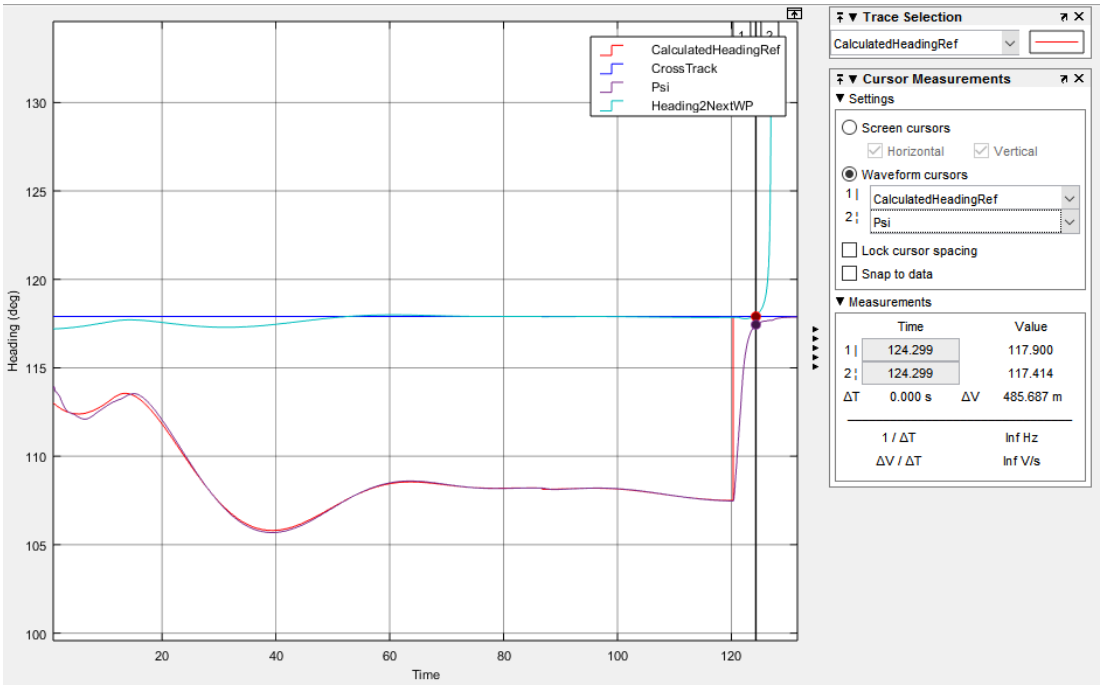


Figure 5-17 Cross Track Performance for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Here, effect of cross wind can be observed in Figure 5-17 and Figure 5-16. Although generated reference is about 107 degree, movement of the UAV is on the cross track which is exactly as expected. To keep cross track, heading of the UAV differs from cross track direction under cross wind exist. After time at 120, decrab maneuver is applied and UAV corrects its heading to movement direction for touching to ground. At time 124.3, ground interaction is occur and decrab angle range requirement is satisfied in Table 4-2 Landing Autopilot Requirements. Heading to next waypoint is increased by 180 degree after touchdown because TDP stays behind after that time. However, outer loop keeps runway heading value as cross track after touchdown.

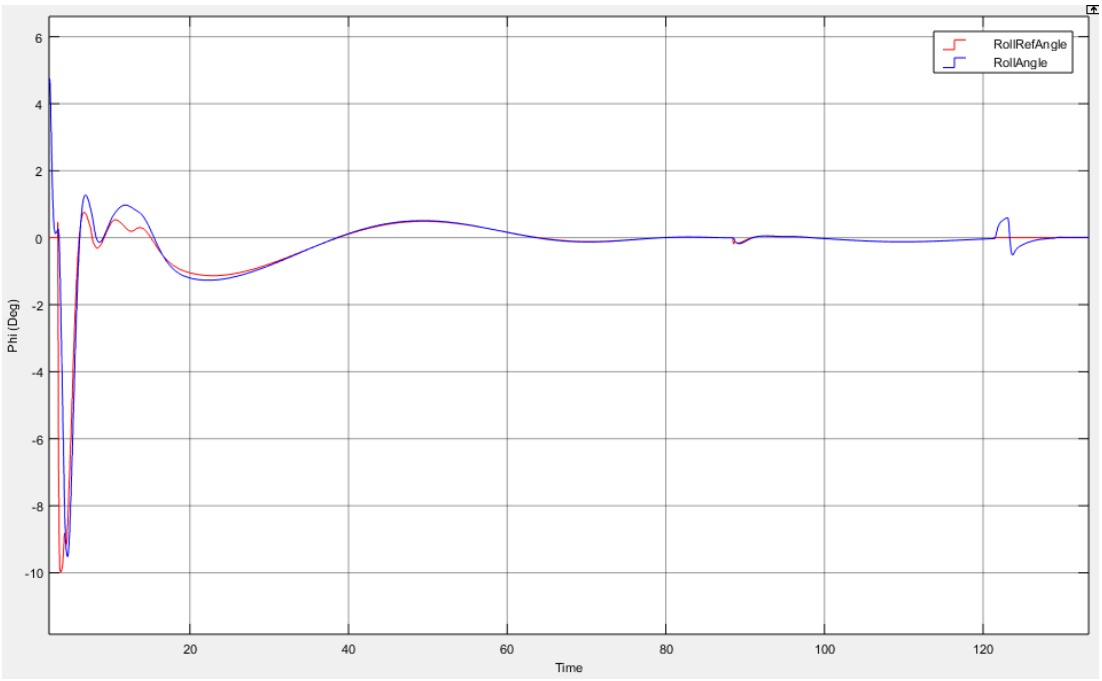


Figure 5-18 Roll Angle of the UAV for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Decrab Maneuver is managed by using rudders and this fast response changes yaw rate. However, as explained in Roll Controller section of inner loop, to be able to satisfy Table 4-2 Landing Autopilot Requirements, not to apply unbalance force to landing gear and not to hit the edge of the wings, roll reference is set to zero. In Figure 5-18, after time at 124.3, decrab maneuver is applied and this maneuver affects roll angle of the UAV but roll controller handles this effect.

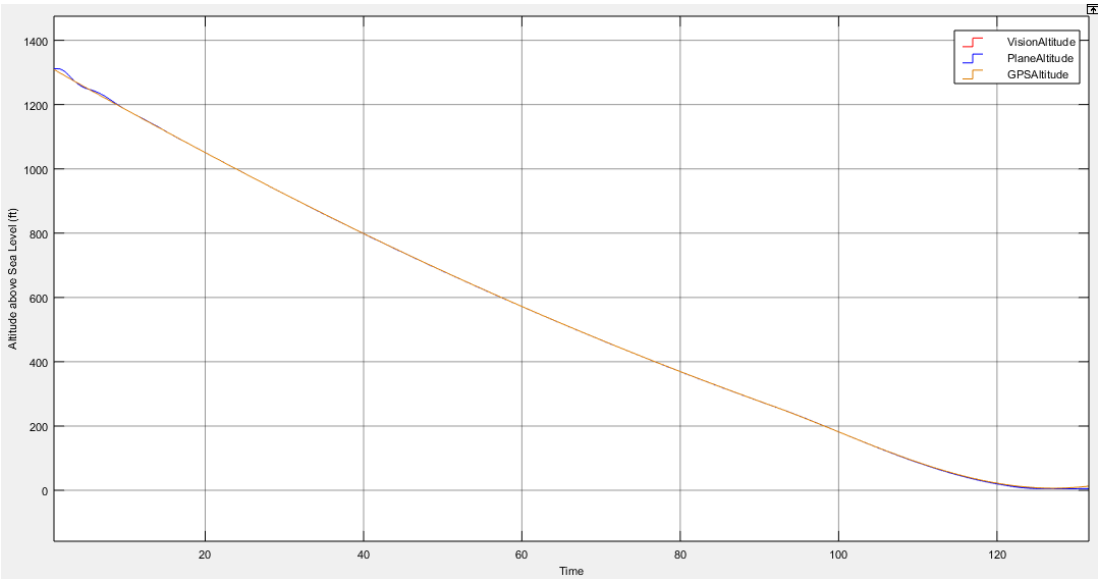


Figure 5-19 Landing Trajectory Path and Altitude of the UAV for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Altitude controller works almost with the same performance as in the Case 1: No Wind and Turbulence. There is no effect observed due to side winds compared to Case 1: No Wind and Turbulence. Altitude trajectory of the UAV and flare maneuver is presented in Figure 5-19. Ground interaction effect is similar to no wind and turbulence case also which is presented in Figure 5-20.

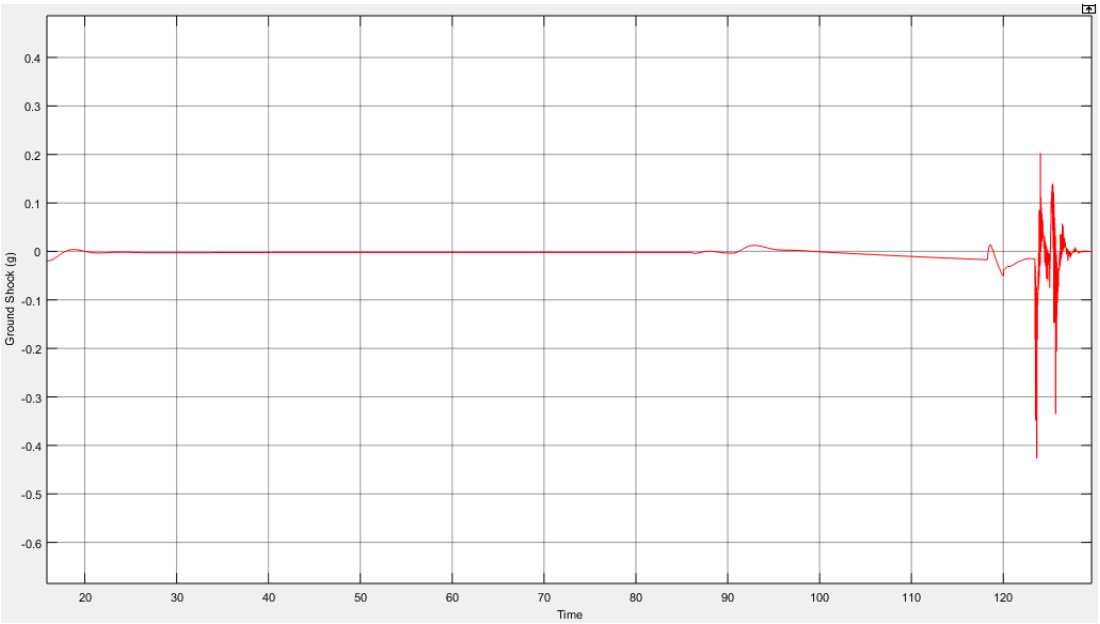


Figure 5-20 Ground Shock at Touchdown for Case 2: 15 kts Cross Wind Condition GPS Based Landing

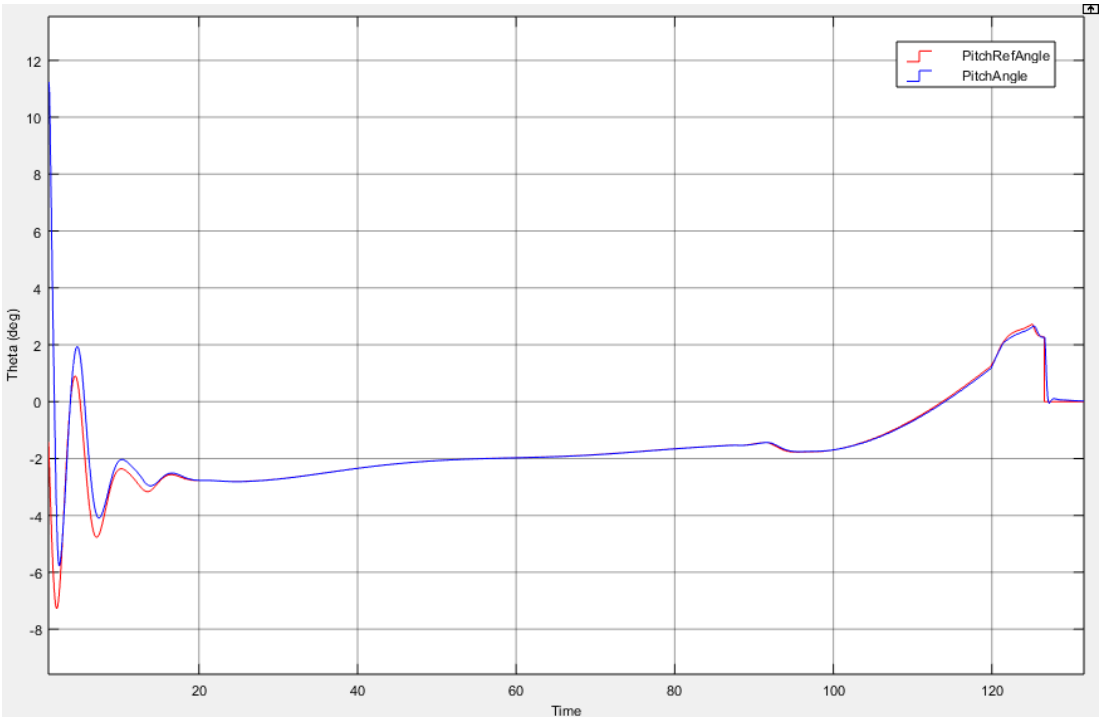


Figure 5-21 Variation of Theta for Case 2: 15 kts Cross Wind Condition GPS Based Landing

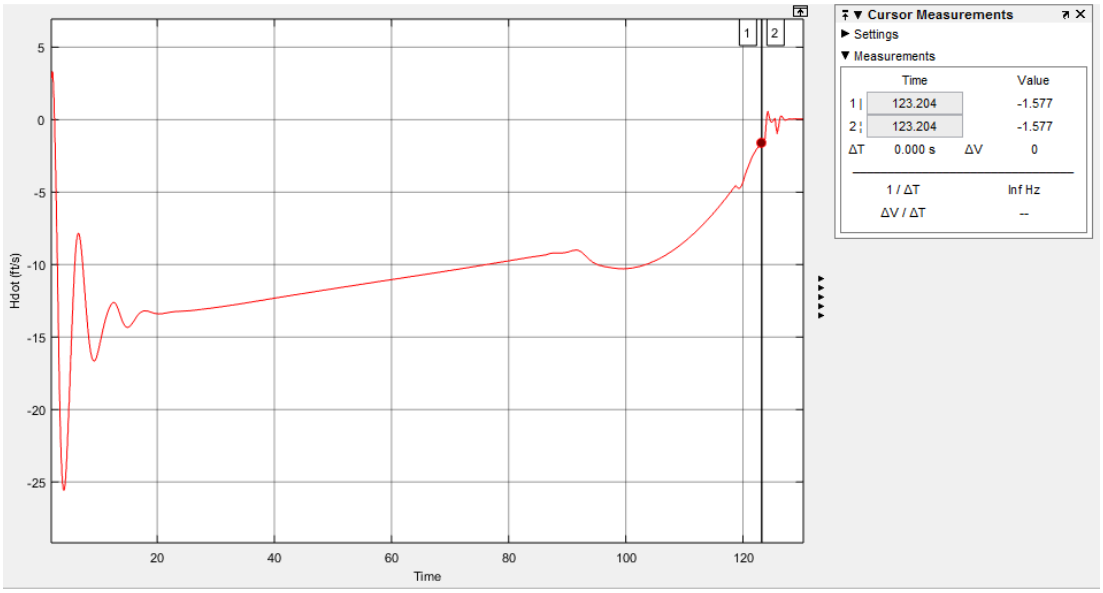


Figure 5-22 $\dot{h}$ for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Pitch angle requirement in Table 4-2 Landing Autopilot Requirements is satisfied as can be seen in Figure 5-21. Also resulted $\dot{h}$ is within the range of requirements as seen in Figure 5-22.

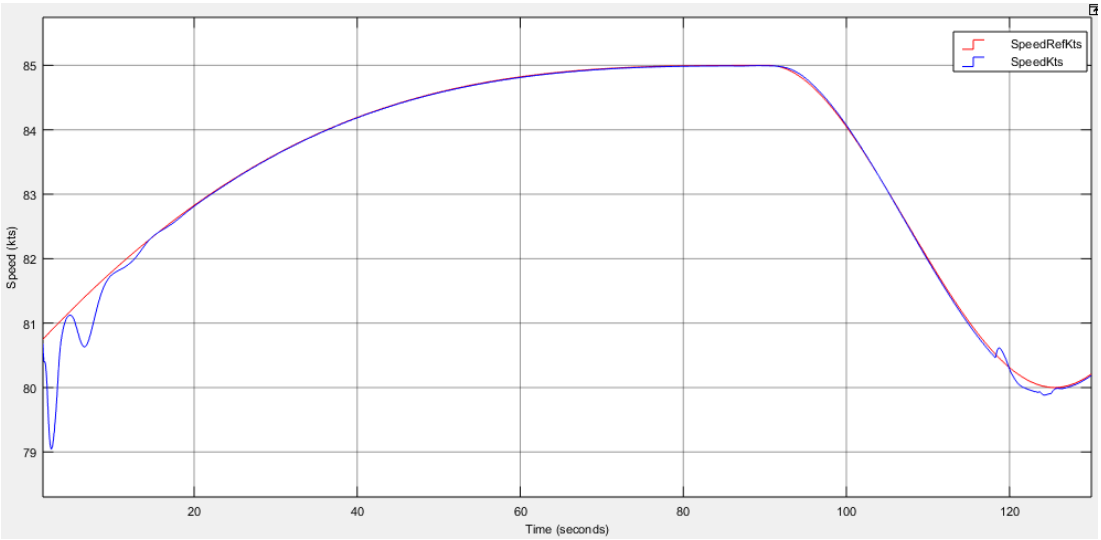


Figure 5-23 Variation of Speed for Case 2: 15 kts Cross Wind Condition GPS Based Landing

Speed controller is affected from side wind due to decrab maneuver. During decrab maneuver, velocity component redistributed between x and y axis while changing heading instantly so this effect is understandable.

5.3.2 VPS Based Landing

Under cross wind condition, vision algorithms give very successful results as can be seen from change of UAV position in Figure 5-24.

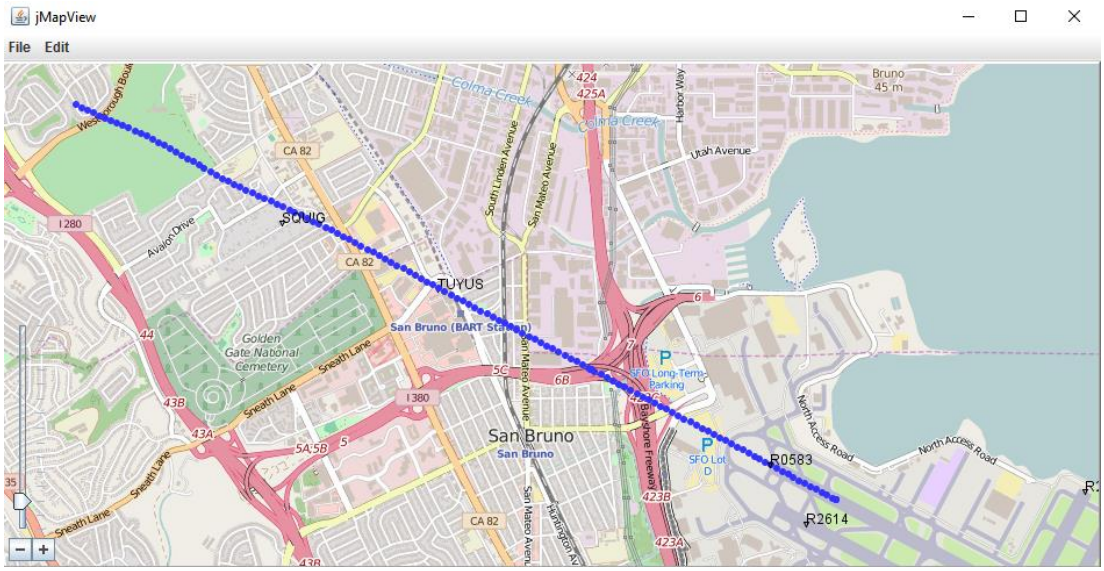


Figure 5-24 Tracking of the UAV Position While Landing for Case 2: 15 kts Cross Wind Condition VPS Based Landing

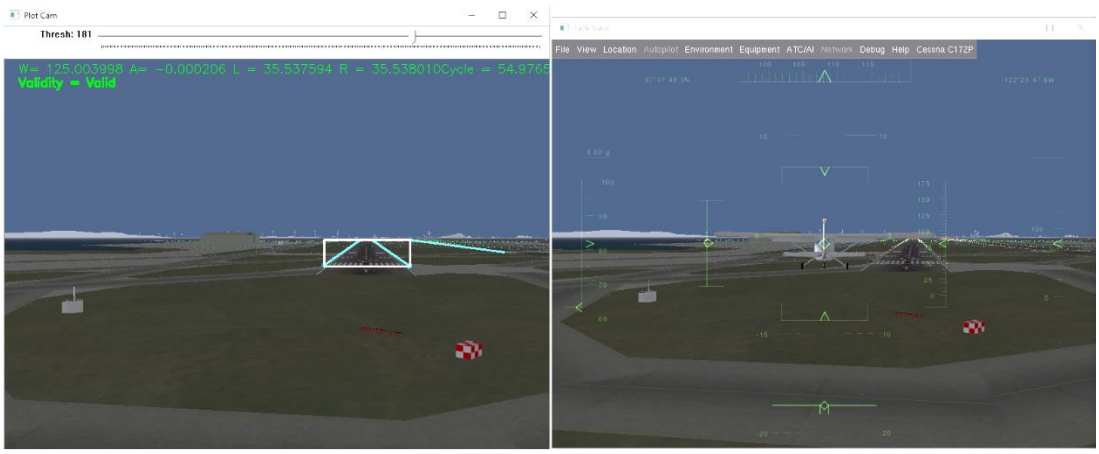


Figure 5-25 Vehicle Position and Orientation for Case 2: 15 kts Cross Wind Condition VPS Based Landing. Left is Pilot Cam, Right is View from Back

Although, position source of VPS and GPS cross track controller is different, resulted reference to inner loop is similar except VPS cross controller has noisier characteristic but as getting closer to runway, this noisy characteristic decreases. As can be seen in Figure 5-25, cross track angle and body heading angle are different, similar to GPS case, as expected. Similar to GPS case, decrab maneuver is applied and resulted with 0.14 degree decrab angle is satisfactory to match Table 4-2 Landing Autopilot Requirements. Course controller performance is presented in

Figure 5-26 and roll controller is effected by course controller noisy outputs due to change in yaw rate. This noisy effect is decreasing while approaching to runway as presented in Figure 5-27. Also decrab effect is can be investigated which is shown at t = 120 and as GPS case, decrab effect on roll angle is handled successfully to match Table 4-2 Landing Autopilot Requirements.

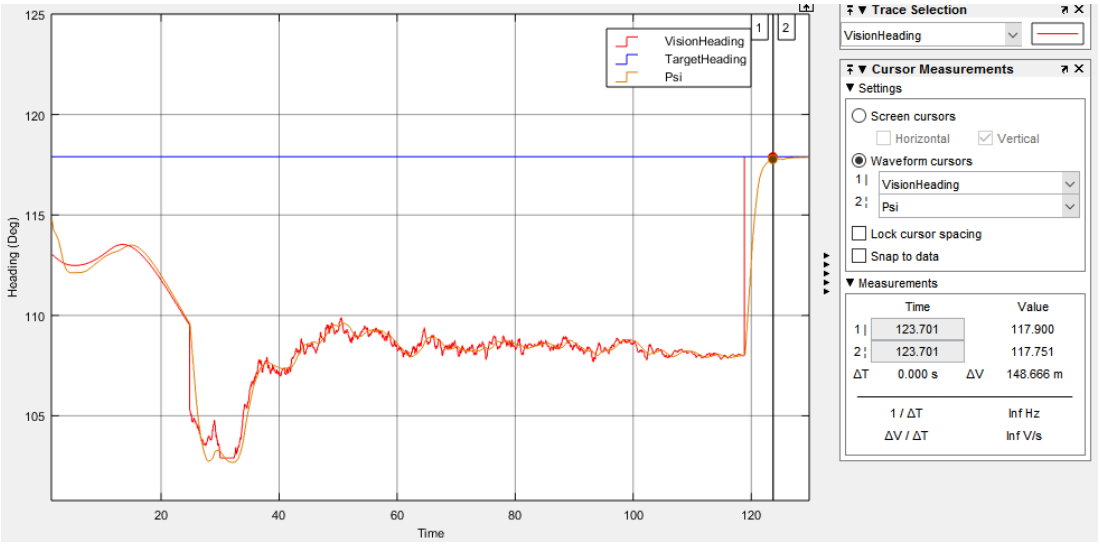


Figure 5-26 Cross Track Performance for Case 2: 15 kts Cross Wind Condition VPS Based Landing

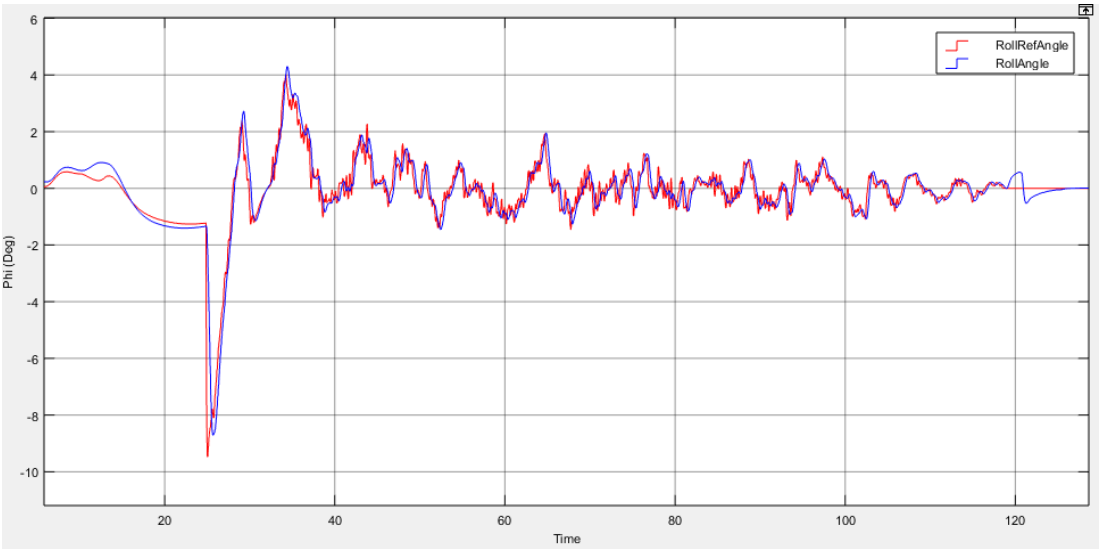


Figure 5-27 Roll Angle of the UAV for Case 2: 15 kts Cross Wind Condition VPS Based Landing

There is a difference in altitude references at distant positions from runway. Especially after activation of vision algorithm, there is a transition region exist. However, after this transition, performance of vision algorithm is sufficient compared to trajectory path generated by using GPS. They are overlapped with time and getting closer to runway. Altitude trajectory of the UAV and flare maneuver is presented in Figure 5-28. Ground interaction effect is similar to no wind and turbulence case, which is presented in Figure 5-29. Pitch angle Theta is varying correspondingly to altitude trajectory and similar noisy characteristic exist again. Theta is presented in Figure 5-30 and with respect to this result, pitch angle requirement is satisfied which is stated in Table 4-2 Landing Autopilot Requirements.

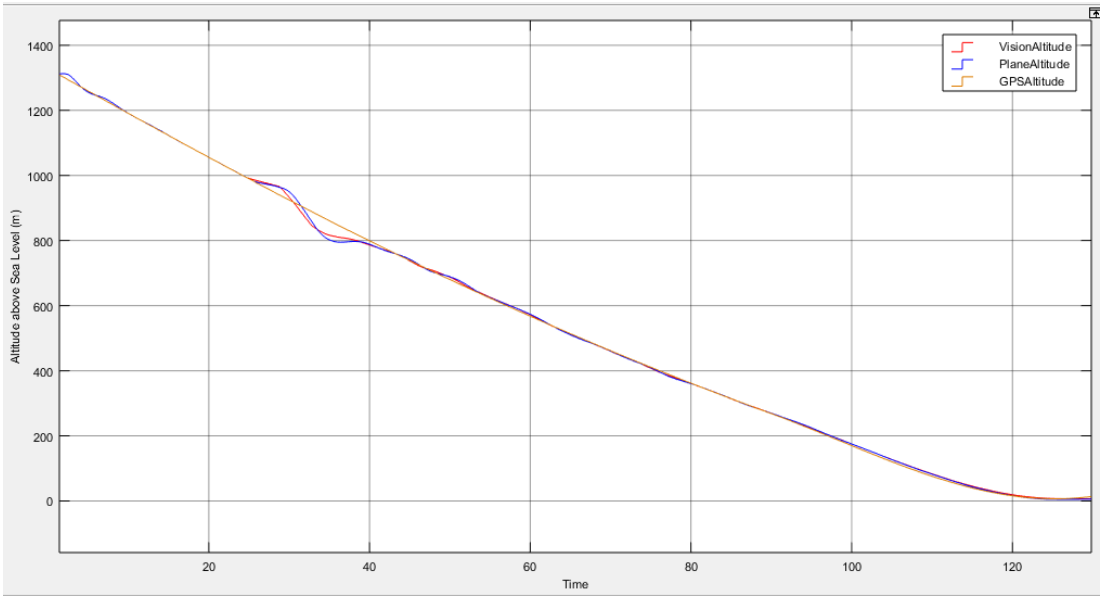


Figure 5-28 Landing Trajectory Path and Altitude of the UAV for Case 2: 15 kts Cross Wind Condition VPS Based Landing

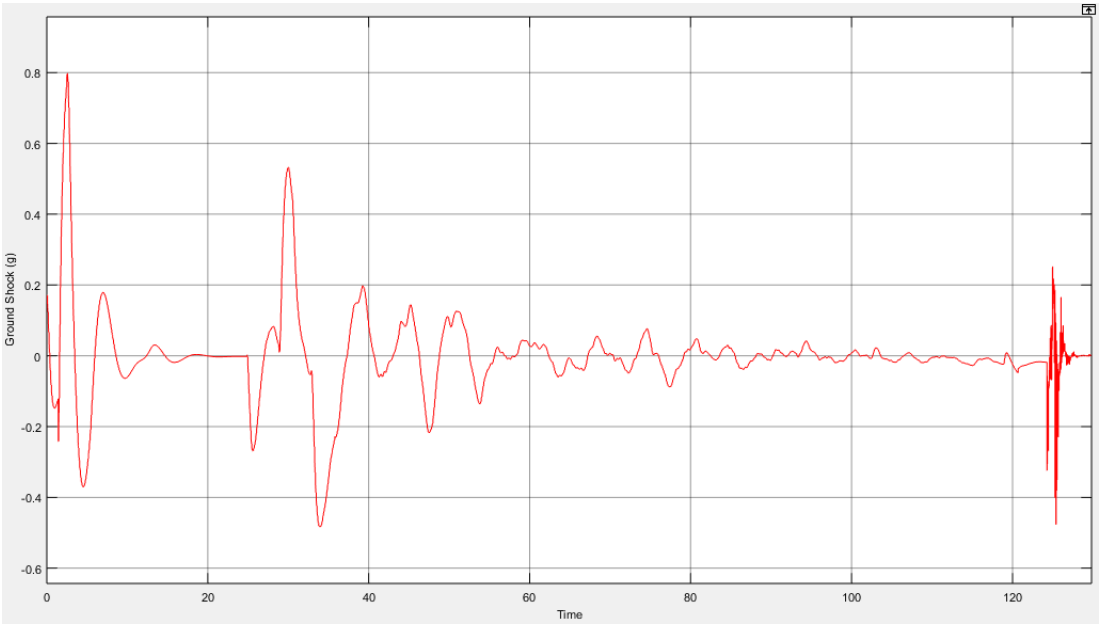


Figure 5-29 Ground Shock at Touchdown for Case 2: 15 kts Cross Wind Condition VPS Based Landing

Ground impact is similar to no wind case of VPS and GPS so we can easily say that smooth landing is managed with VPS under 15 kts side wind. Resulted $\dot{h}$ is within the range of requirements as seen in Figure 5-31.

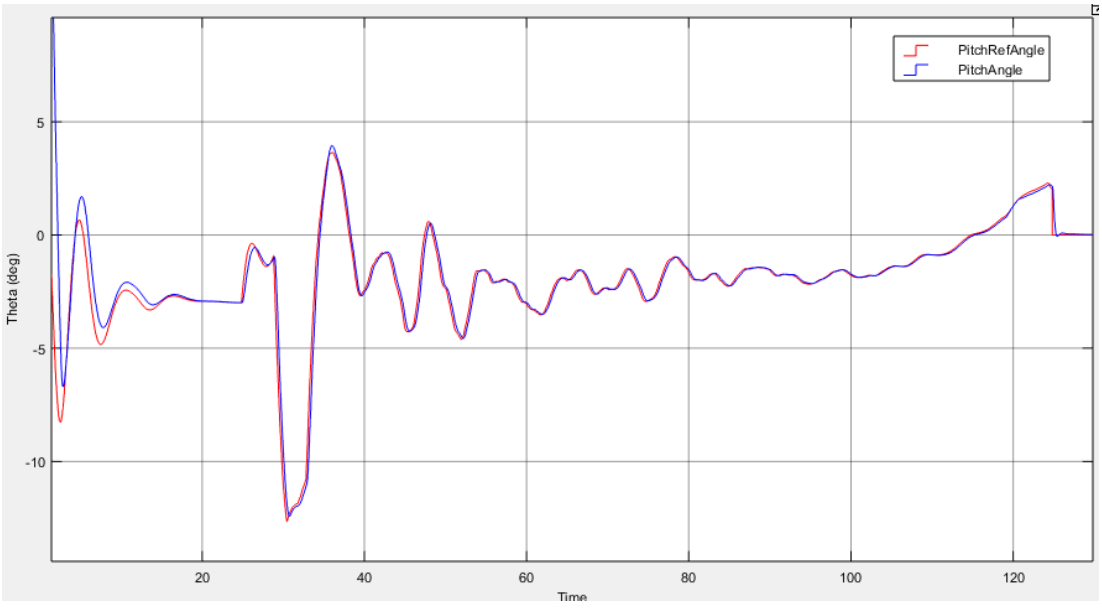


Figure 5-30 Variation of Theta for Case 2: 15 kts Cross Wind Condition VPS Based Landing

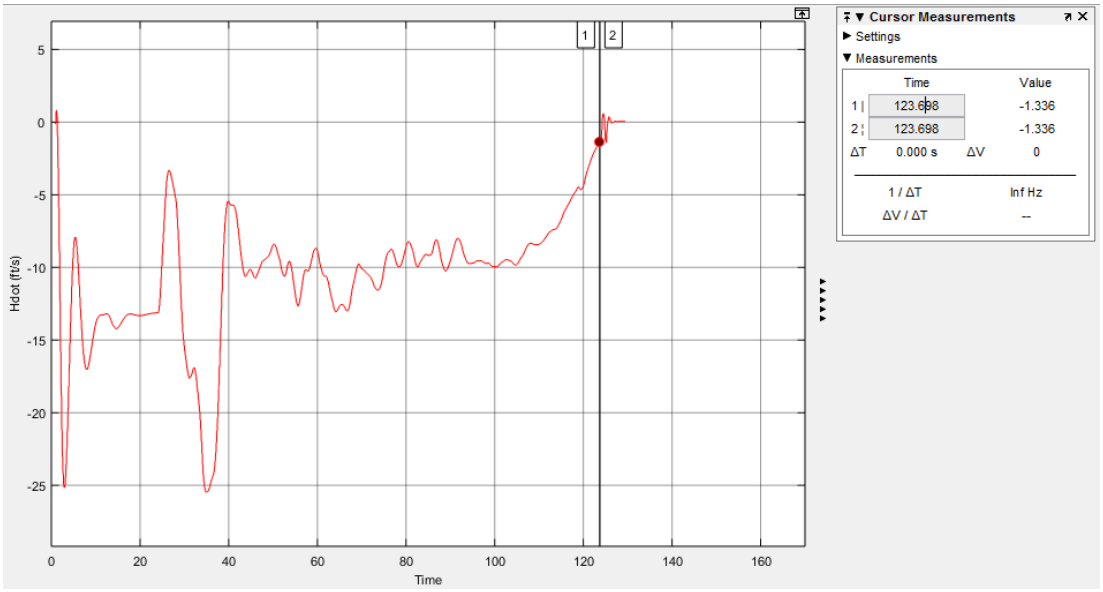


Figure 5-31 $\dot{h}$ for Case 1: No Wind and Turbulence Case 2: 15 kts Cross Wind Condition VPS Based Landing

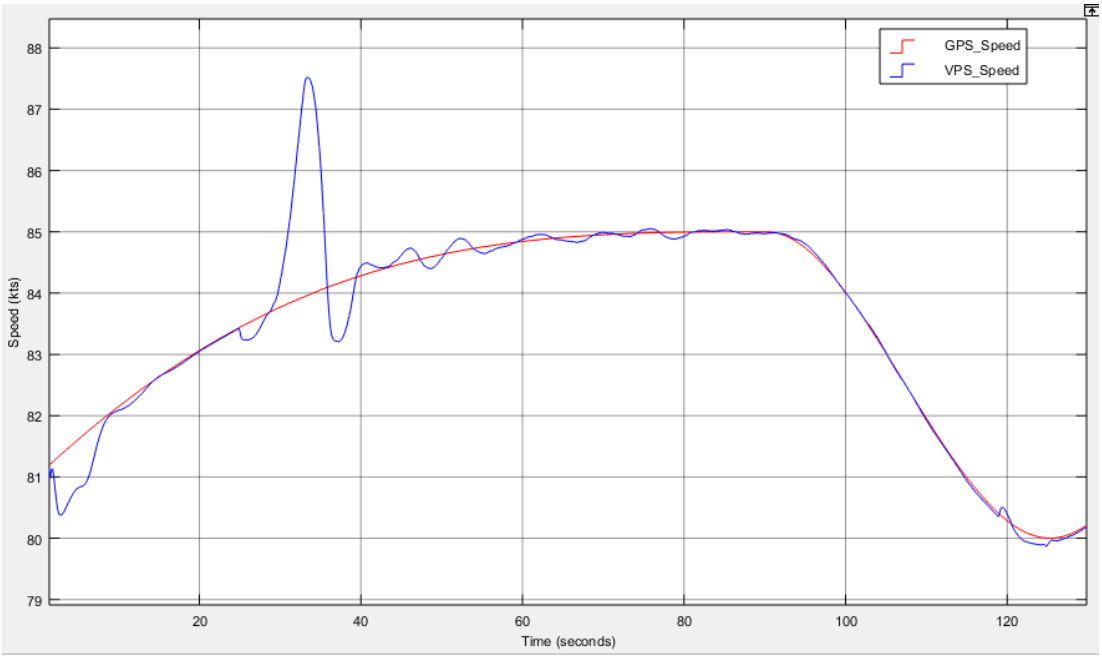


Figure 5-32 Variation of the Speed for Case 2: 15 kts Cross Wind Condition VPS Based Landing

Speed variation is understandable due to high altitude reference change. Aircraft model does not have air brakes so if pitch down condition occurs, although no trust

exist, speed of the aircraft may increase. At t = 120, decrab effect is observable as GPS case. Variation of the speed is introduced in Figure 5-32.

Briefly, some oscillations are faced with cross wind conditions however these oscillations are extinguished with getting closer to runway and successful landing is managed under these conditions.

5.4 Case 3: 15 kts Cross Wind with Cross Track Error

For this case system behavior is obtained under 15 kts cross wind condition and initial position of the UAV is not on the cross track direction. After reaching cross track direction, rest of the simulation turn into Case 2: 15 kts Cross Wind Condition.

5.4.1 GPS Based Landing

As can be investigated from position history of the UAV in Figure 5-33, initial position of the UAV is not on the cross track direction. Course Controller of Outer Loop makes this error as small as possible. As can be seen in Figure 4-33, cross track is kept highly before landing occurs.

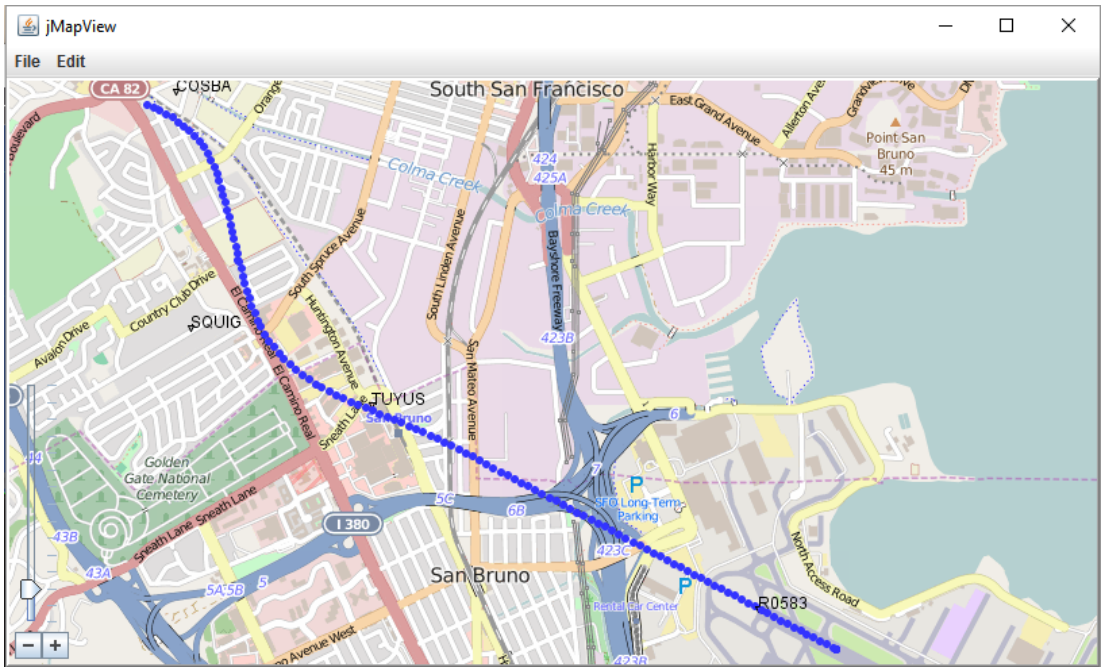


Figure 5-33 Tracking of the UAV Position While Landing for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

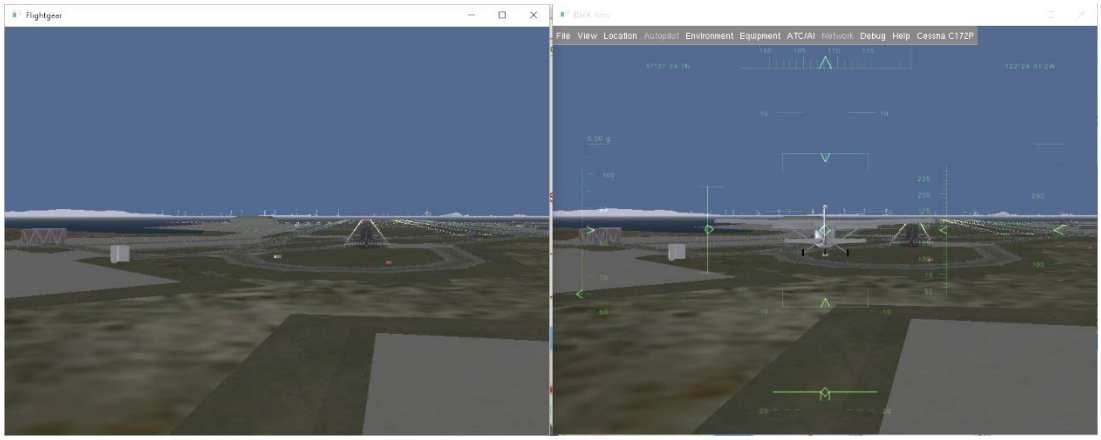


Figure 5-34 Vehicle Position and Orientation for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing. Left is Pilot Cam, Right is View from Back

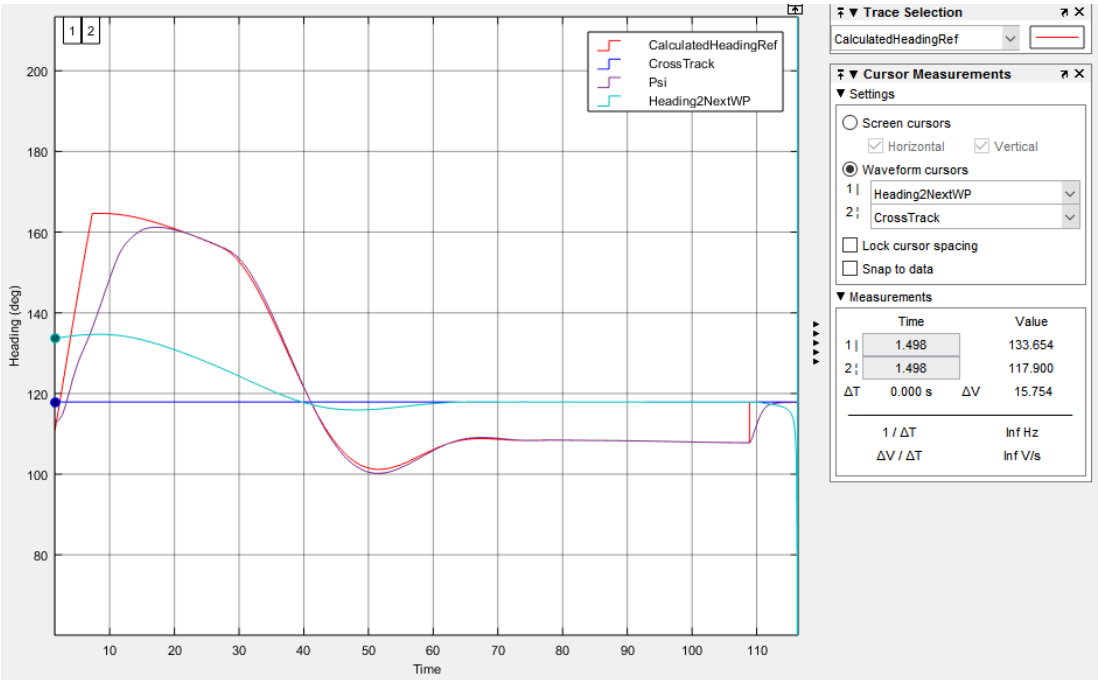


Figure 5-35 Cross Track Performance for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

At the beginning of the simulation, cross-track error is 15.75°. Course controller output increases and saturates which is presented with “CalculatedHeadingRef”. Initial heading of the UAV is 113° and heading controller changes it to fit outer loop heading reference. Cross track error becomes smaller with time but reference value of outer loop does not decreased to cross track value due to crosswind. After t = 110, decrab maneuver can be investigated also. After keeping cross track, results would be similar to Case 2: 15 kts Cross Wind Condition, therefore, “At Touchdown” requirements is accepted to be satisfied.

Roll angle is affected from cross track error because while changing heading of the UAV, roll controller also participates in that process and roll angle changes more than Case 2: 15 kts Cross Wind Condition. As can be seen in Figure 5-36, roll angle does not exceed allowed range defined in Table 4-2 Landing Autopilot Requirements while on ground and at touchdown.

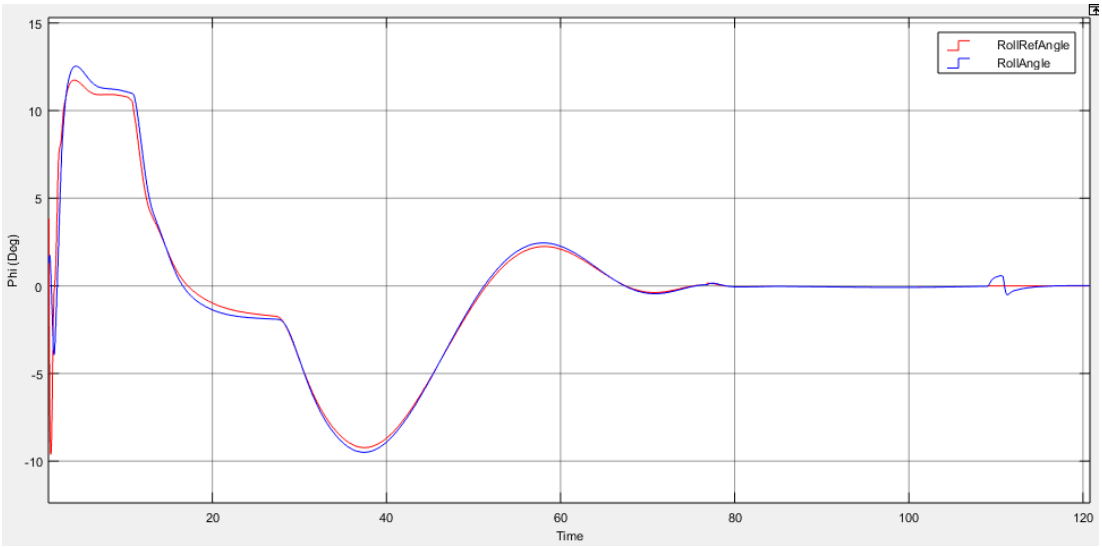


Figure 5-36 Roll Angle of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

There is no significant change in altitude trajectory, presented in Figure 5-37, with respect to previous cases. Therefore, pitch angle in Figure 5-38, ground shock in Figure 5-39, $\dot{h}$ in Figure 5-40 and speed in Figure 5-41 have similar characteristics with Case 2: 15 kts Cross Wind Condition.

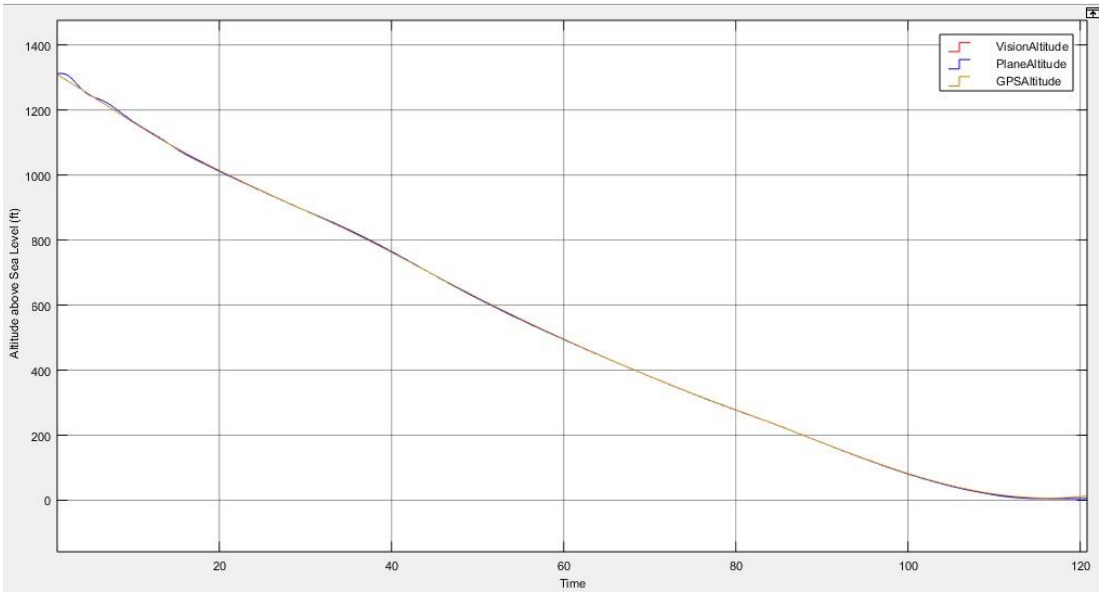


Figure 5-37 Landing Trajectory Path and Altitude of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

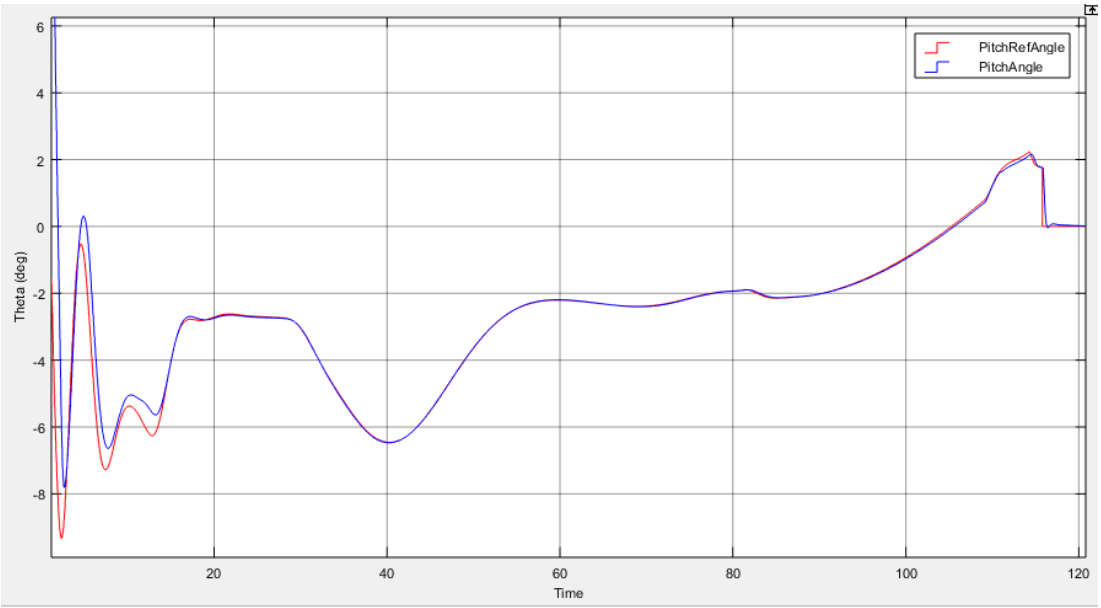


Figure 5-38 Variation of Theta for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

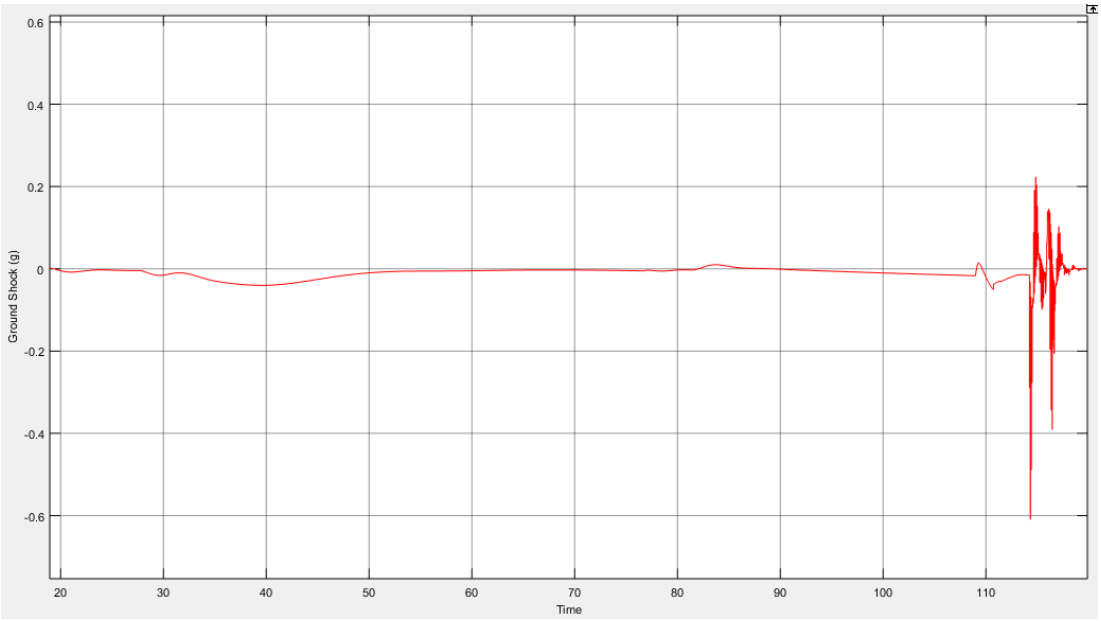


Figure 5-39 Ground Shock at Touchdown for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

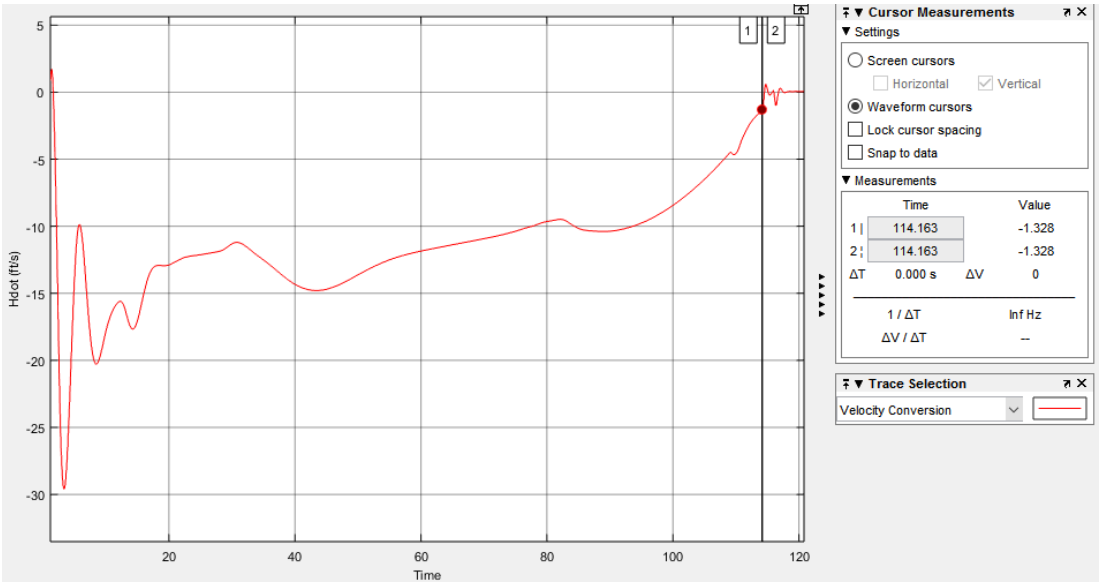


Figure 5-40 $\dot{h}$ for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

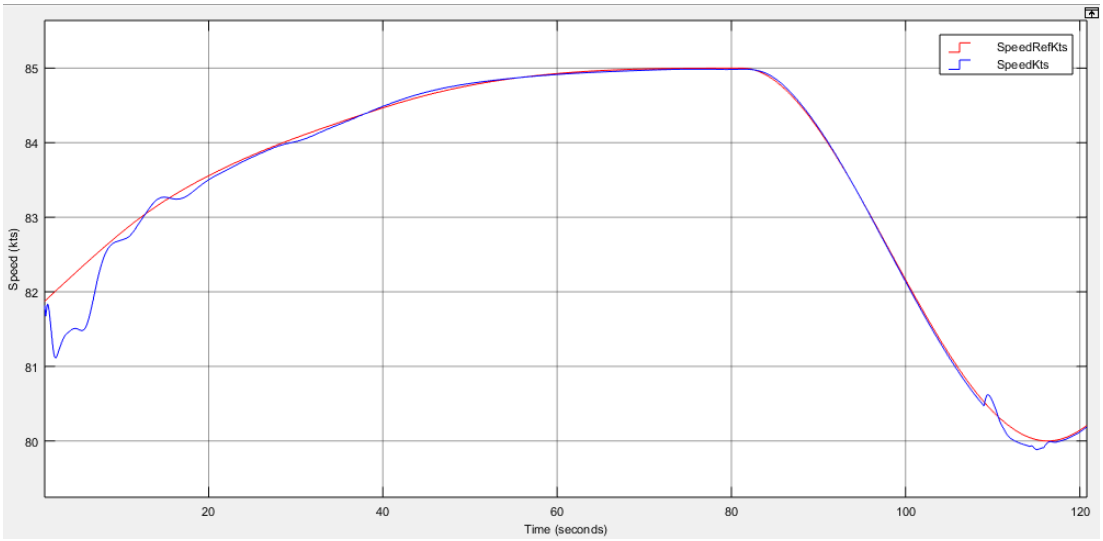


Figure 5-41 Variation of Speed for Case 3: 15 kts Cross Wind with Cross Track Error GPS Based Landing

Briefly, all requirements stated in Table 4-2 Landing Autopilot Requirements are satisfied successfully.

5.4.2 VPS Based Landing

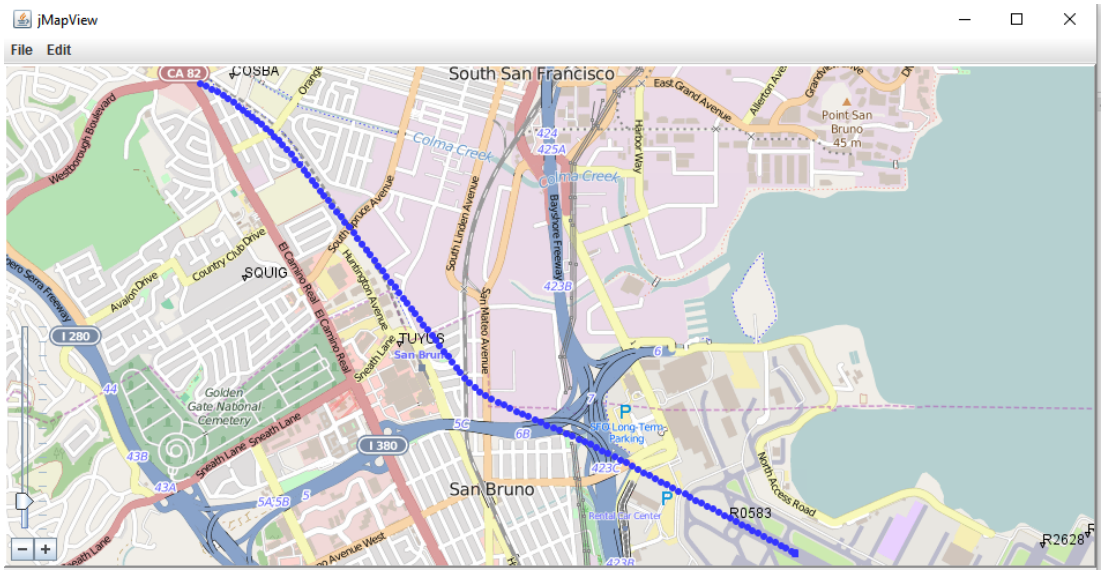


Figure 5-42 Tracking of the UAV Position While Landing for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

As can be seen in Figure 5-42, cross track error is extinguished by using VPS and UAV could reach TDP successfully.

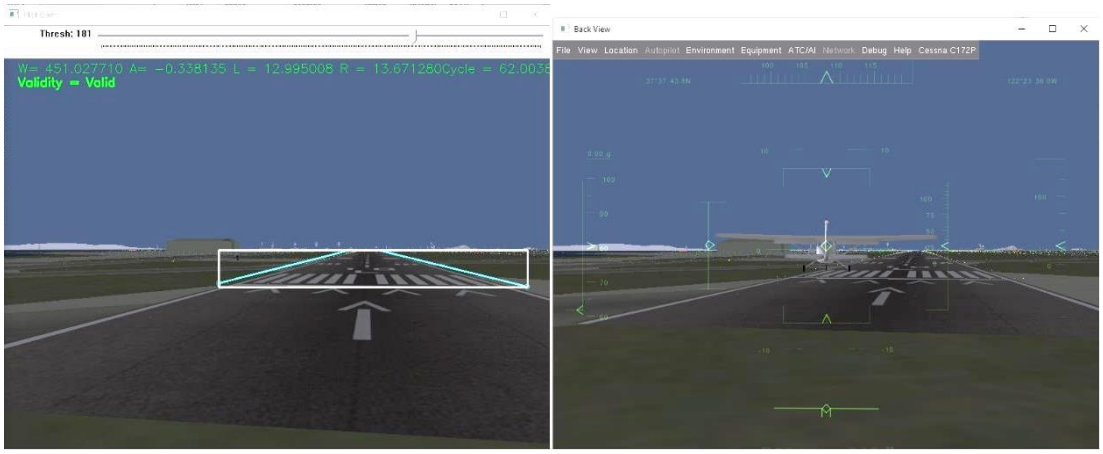


Figure 5-43 Vehicle Position and Orientation for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing. Left is Pilot Cam, Right is View from Back

To keep runway in the vision frame, cross track controller output of VPS is limited within [-15 15] degree. Initial position of the UAV is at left side of the runway and

crosswind is blowing from left of the UAV. So, to be able to decrease cross track error, firstly, heading value of the UAV is bigger than cross track angle. After time is equal to 60, cross track error changes sign so heading value of the UAV gets smaller than cross track angle to keep bearing error small. Decrab angle value is 0.13 which is very close to Case 2: 15 kts Cross Wind Condition as presented in Figure 5-44.

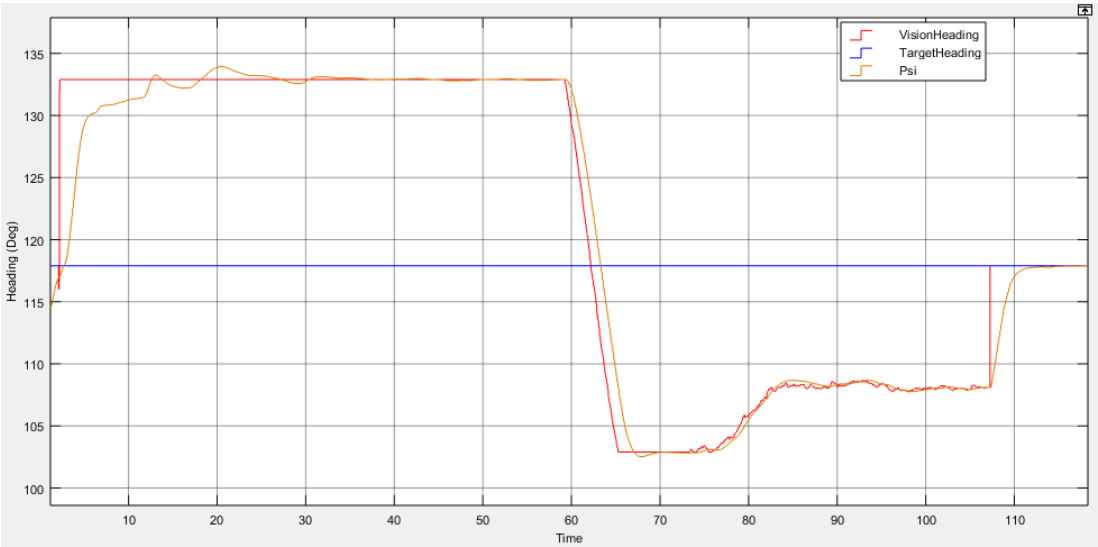


Figure 5-44 Cross Track Performance for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

Variation of the Roll angle is similar to Case 2: 15 kts Cross Wind Condition except instant heading reference change effect. Between 160-170 seconds, course controller affects roll angle via heading controller architecture of inner loop which is an expected result. Although fast change in heading reference occurs, roll angle is within the range which is defined in Table 4-2 Landing Autopilot Requirements. After instant change in heading values, bearing error is minimized and rest of the simulation results is similar to results of Case 2: 15 kts Cross Wind Condition.

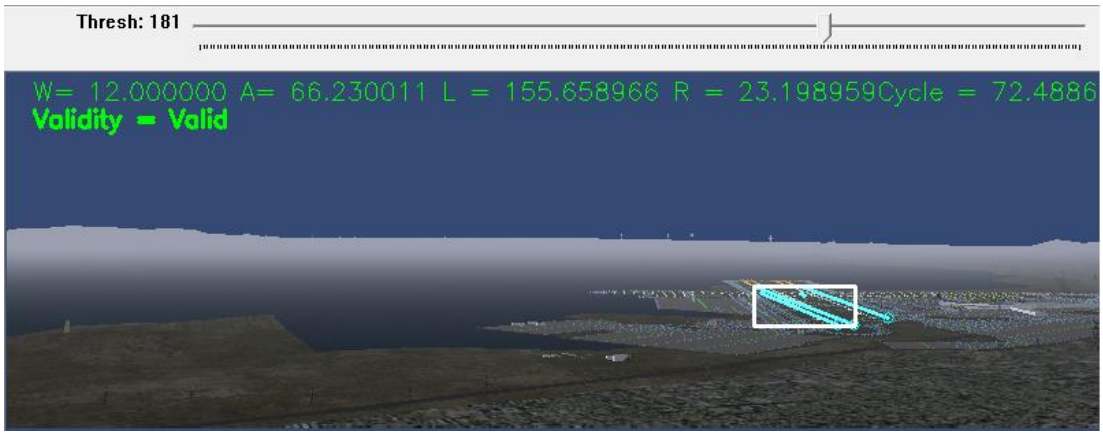


Figure 5-45 Initial Position of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

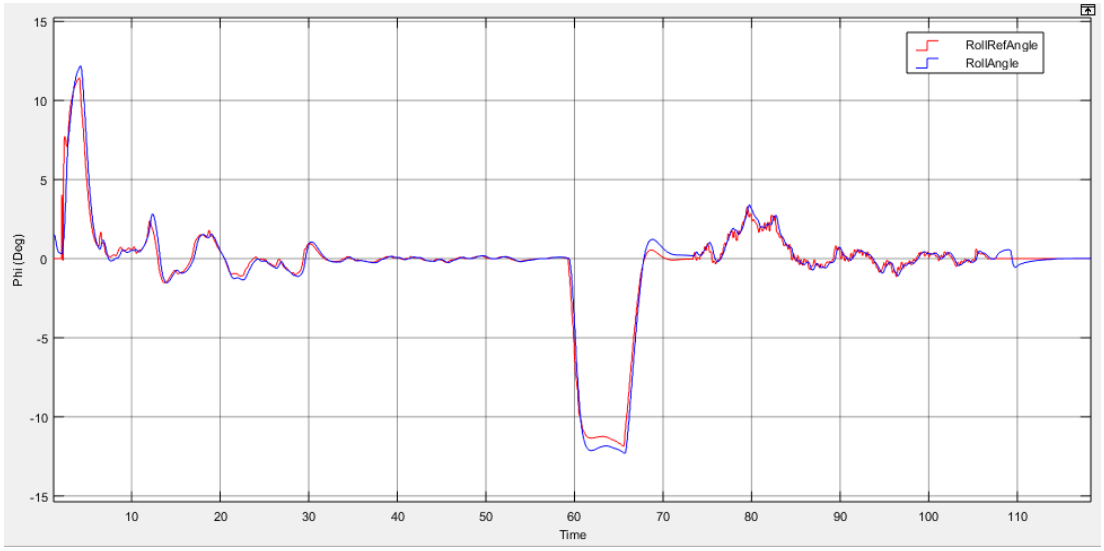


Figure 5-46 Roll Angle of the UAV for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

Initial alignment error of UAV is very high as can be seen in Figure 5-45 and as presented in this figure, edges of the runway is very close to each other. During minimizing the bearing error, measured runway width changes very fast has very noisy characteristic. Therefore, effect of that noisy characteristic is reflected to altitude and speed interpolations. However, similar to previous cases, this noisy characteristic of VPS is turn into smoother data as can be seen from variation of altitude trajectory in Figure 5-47, variation of pitch angle in Figure 5-48, variation of

$\dot{h}$ in Figure 5-50 and variation of speed in Figure 5-51. Also ground shock value is very close to value at Case 2: 15 kts Cross Wind Condition.

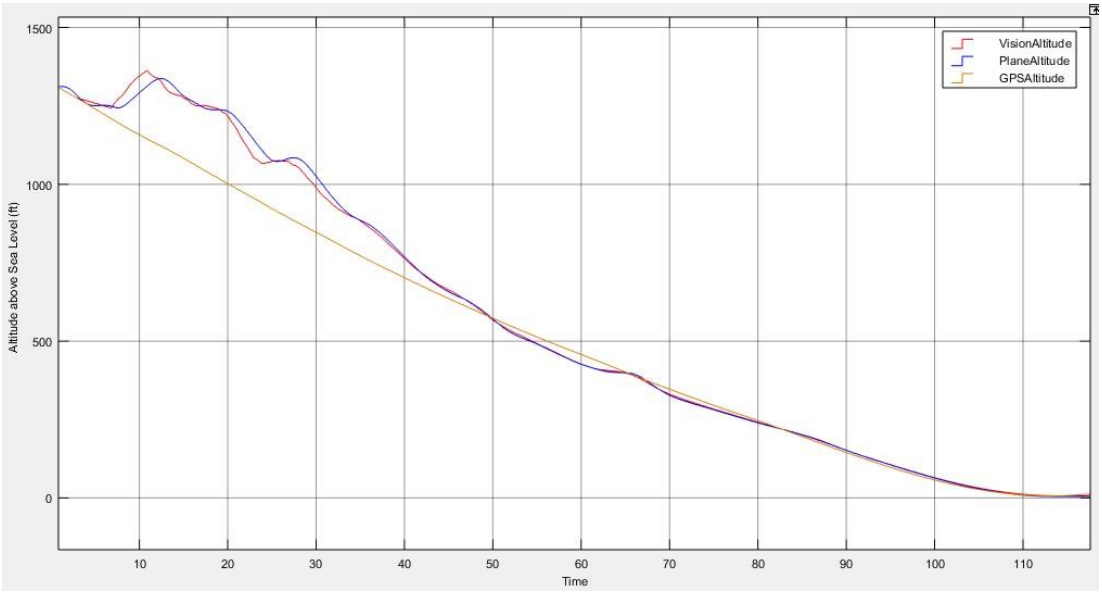


Figure 5-47 Landing Trajectory Path and Altitude of the Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

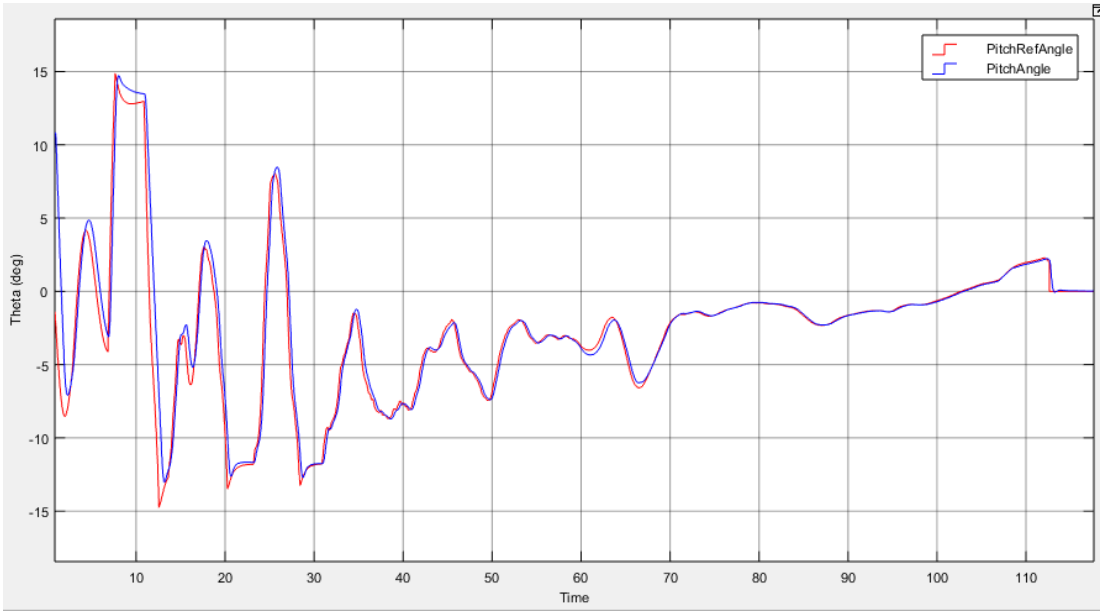


Figure 5-48 Variation of Theta for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

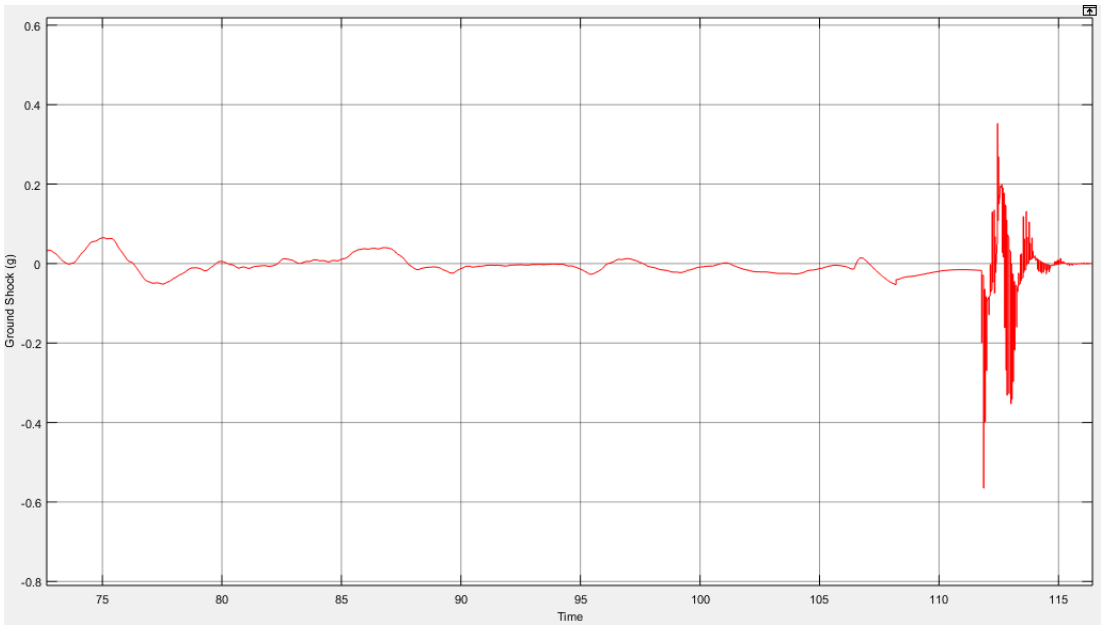


Figure 5-49 Ground Shock at Touchdown for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

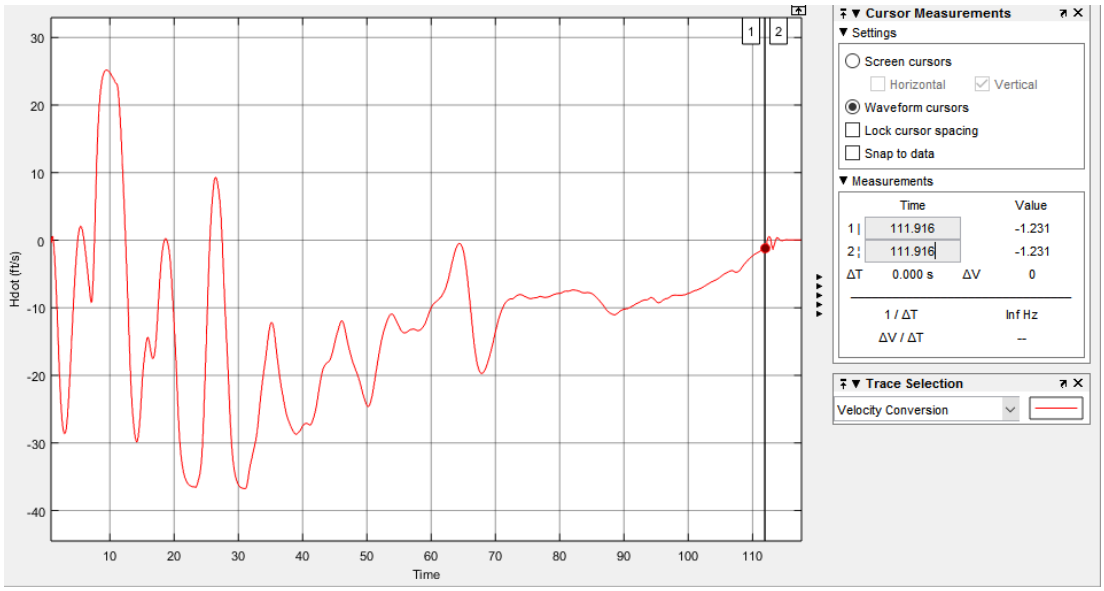


Figure 5-50 $\dot{h}$ for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

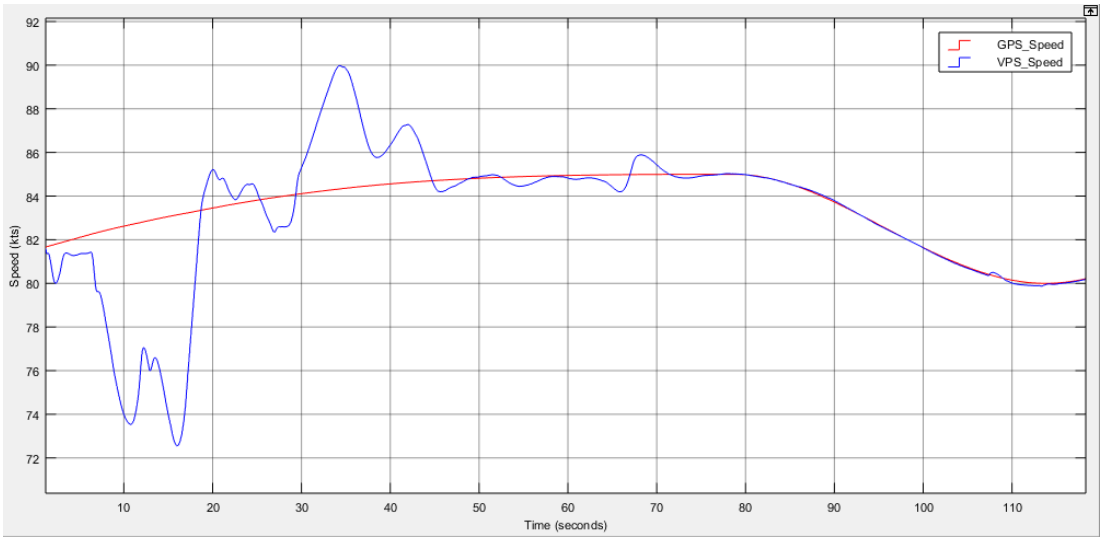


Figure 5-51 Variation of the Speed for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

For this case, VPS lateral performance is worse than no cross track error case. However, as can be seen in figures, this oscillations are temporary and extinguishes while approaching to runway. Longitudinal performance was not affected so much because it is work differential by looking right and left edges. So noises in slopes can eliminated easily because same noise exist for both edges. However, we have no chance to eliminate noise in measurements of runway width.

5.5 Case 4: 15 kts Cross Wind + Turbulence

In this case, by using wind model which is exist in Beaver demo of Matlab, turbulence is adjusted to be highest and 15 kts cross-wind at y axis are applied. Von Karman Wind Turbulence Model in Matlab/Simulink Aerospace Blockset is used to generate wind conditions in Beaver demo, which implements the mathematical representation in the Military Specification MIL-F-8785C. Configuration of Dryden Wind Turbulence Model is presented in Table 5-2 and resulted wind condition can be investigated in Figure 5-52.

Property	Value
Units	Metric
Specification	MIL-F-8785C
Model Type	Discrete Dryden (+q -r)
Wind speed at 6 m defines the low-altitude intensity (m/s)	5
Wind direction at 6 m (degrees clockwise from north):	0
Probability of exceedance of high-altitude intensity:	10^{-6}
Scale length at medium/high altitudes (m):	533.4
Wingspan (m):	14
Band limited noise and discrete filter sample time (s)	0.1
Noise seeds [ug vg wg pg]:	[23341 23342 23343 23344]

Table 5-2 Parameters of Configured Von Karman Wind Turbulence Model

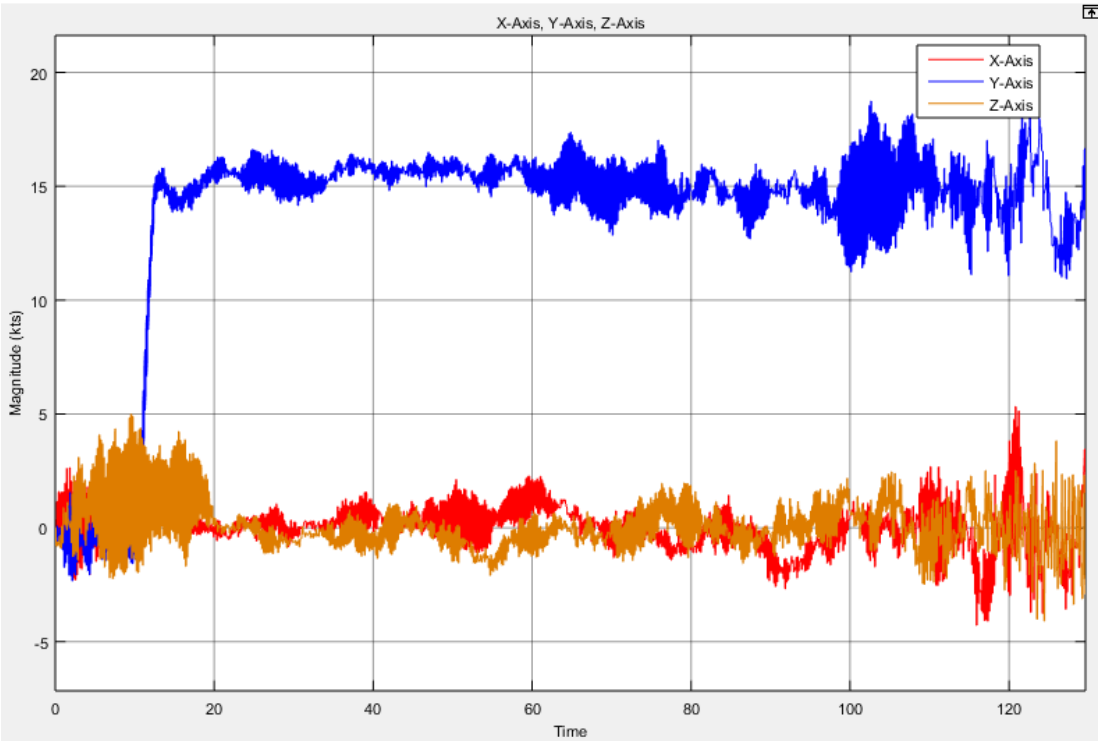


Figure 5-52 Wind Profile for Case 4: 15 kts Cross Wind + Turbulence

5.5.1 GPS Based Landing

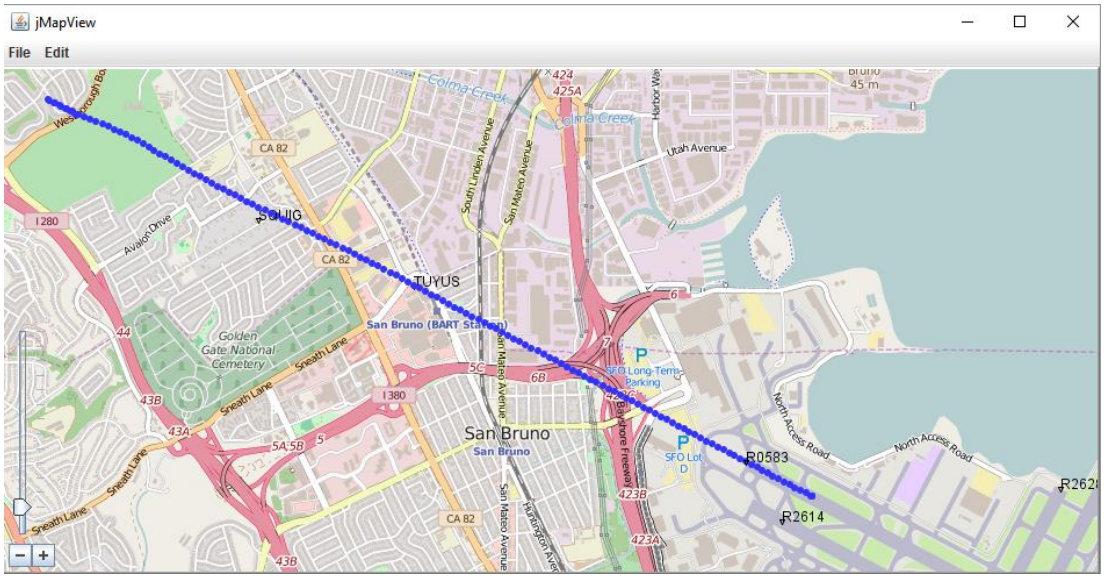


Figure 5-53 Tracking of the UAV Position While Landing for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

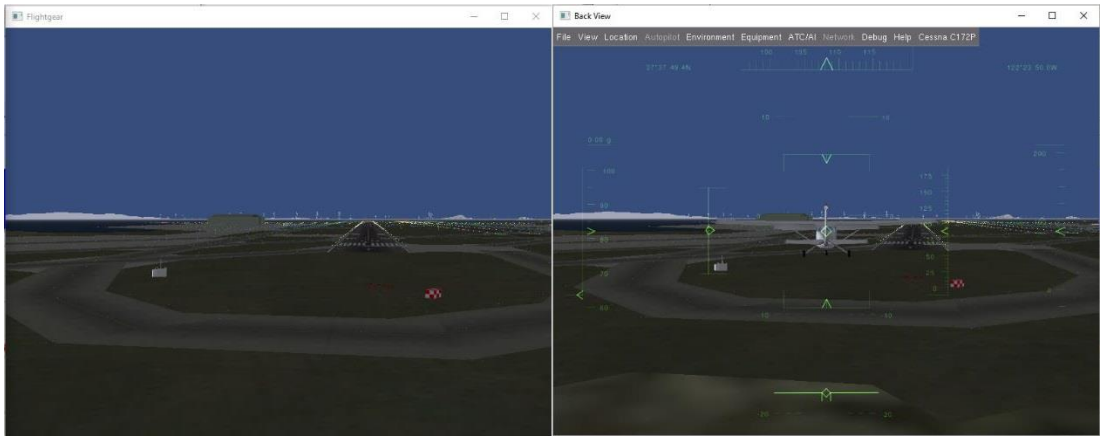


Figure 5-54 Vehicle Position and Orientation for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing. Left is Pilot Cam, Right is View from Back

Cross track is kept successfully by using GPS under turbulence and side wind condition. Decrab angle is obtain nearly zero but it is absolute that this is not due to controller performs better but due to turbulence effect. It may be worse also but generally course controller and heading controller satisfy Table 4-2 Landing

Autopilot Requirements. Cross track performance is introduced in Figure 5-53, Figure 5-54 and Figure 5-55.

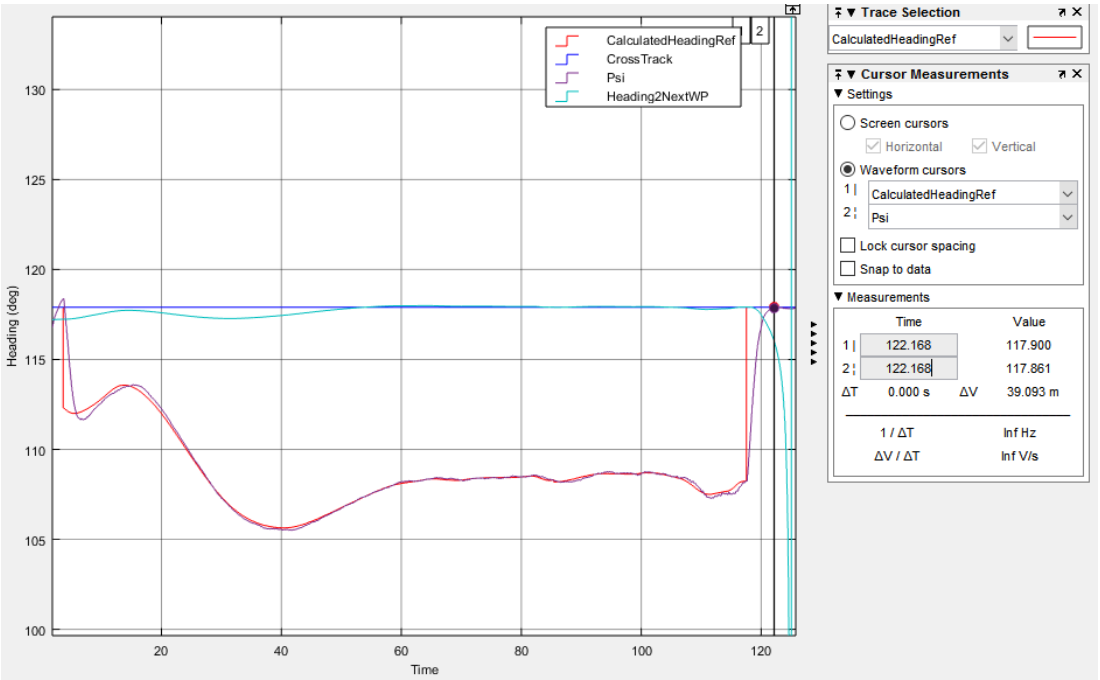


Figure 5-55 Cross Track Performance for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

Roll angle is affected from turbulence but this effect does not exceed to 1 degree. Here, important thing is roll angle at touchdown and as can be seen in Figure 5-56, roll angle is within the range of Table 4-2 Landing Autopilot Requirements.

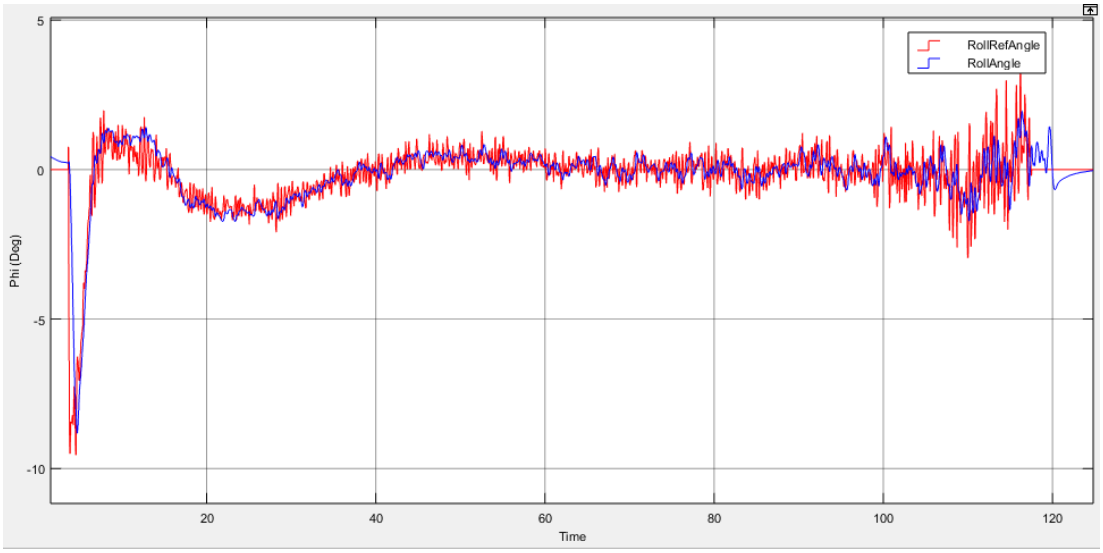


Figure 5-56 Roll Angle of the UAV for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

Effect of turbulence on altitude of the UAV is limited. Small oscilations are observed in altitude but in general altitude controller generally works well and UAV is landed safely. However, that is not the mean that pitch angle or $\dot{h}$ is not effected. They have a oscilation characteristic to correct turbulence effect so that, altitude can be kept successfully. Pitch angle variation is within allowed range and presented in Figure 5-58. Due to turbulence $\dot{h}$ is higher than other cases at touchdown. It is nearly border of the requirements but still satisfies Table 4-2 Landing Autopilot Requirements. $\dot{h}$ is introduced in Figure 5-60.

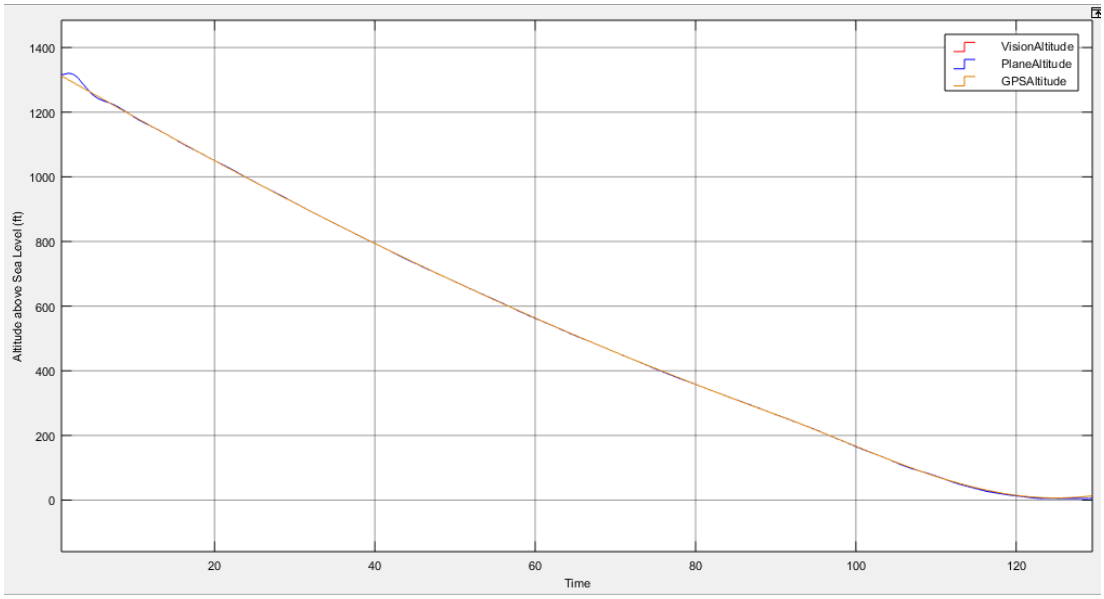


Figure 5-57 Landing Trajectory Path and Altitude of the UAV for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

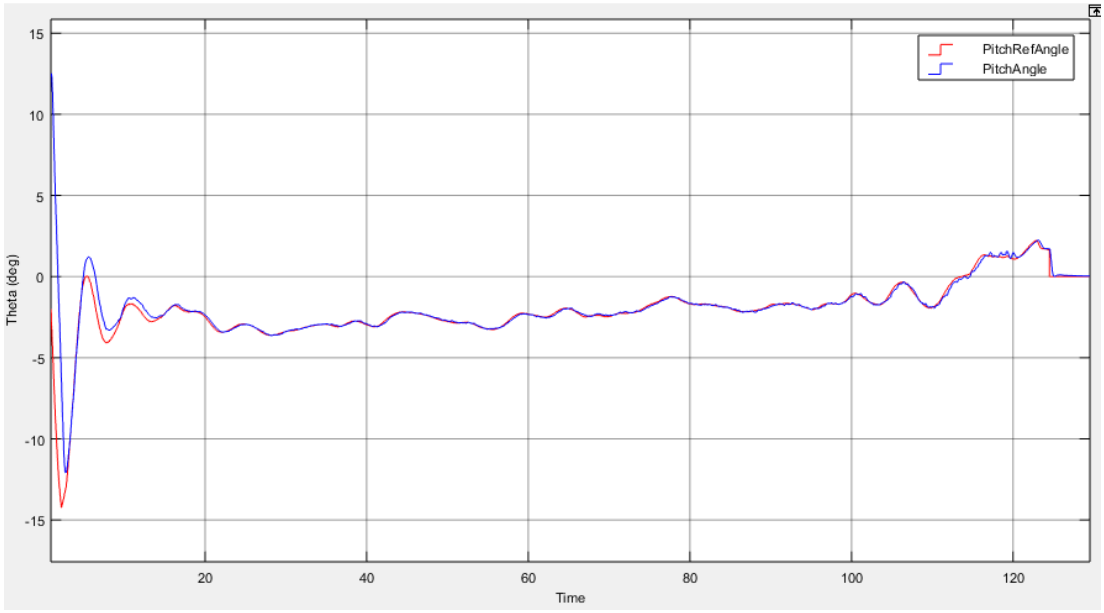


Figure 5-58 Variation of Theta for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

Turbulence affects vertical acceleration during flight so ground impact is not distinguishable as previous cases. However, highest value after t=120 shows that hardest landing is occurred with -0.85 g which is a similar result with $\dot{h}$.

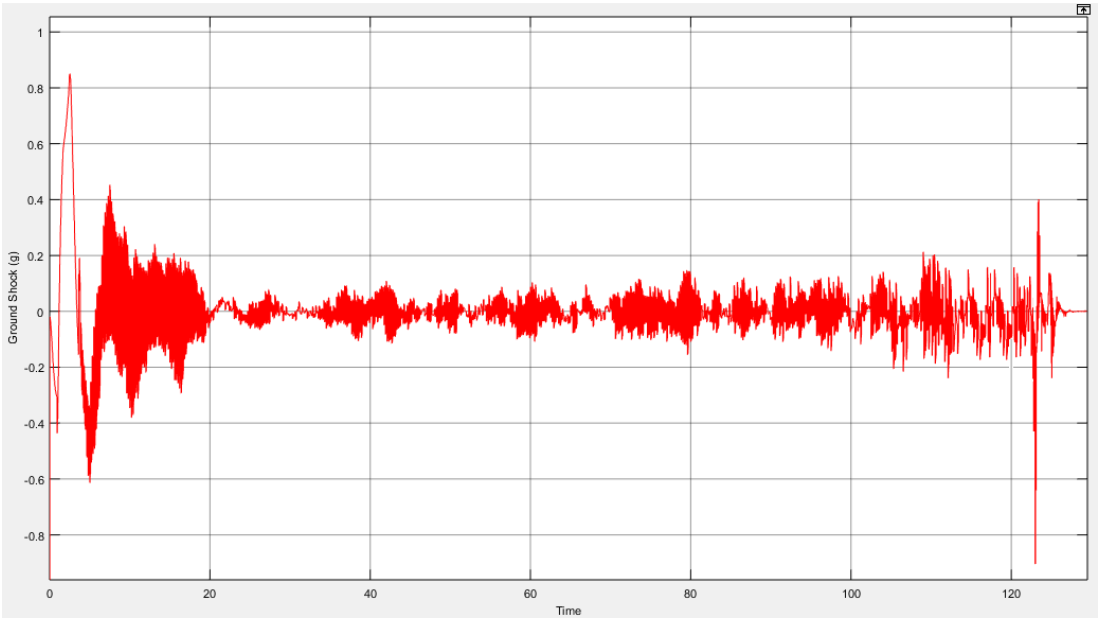


Figure 5-59 Ground Shock at Touchdown for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

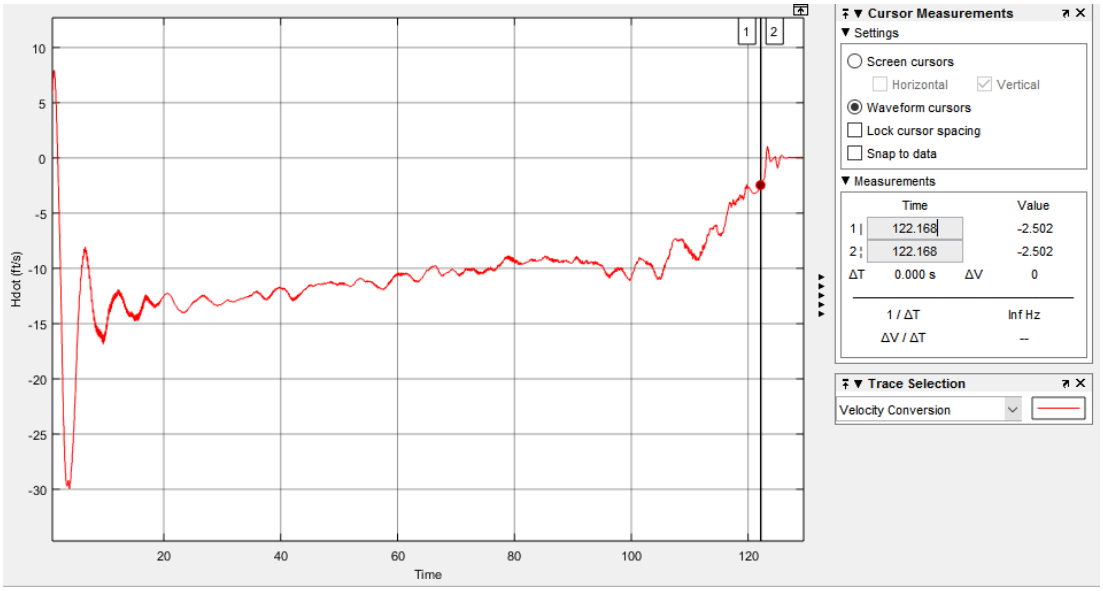


Figure 5-60 $\dot{h}$ for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

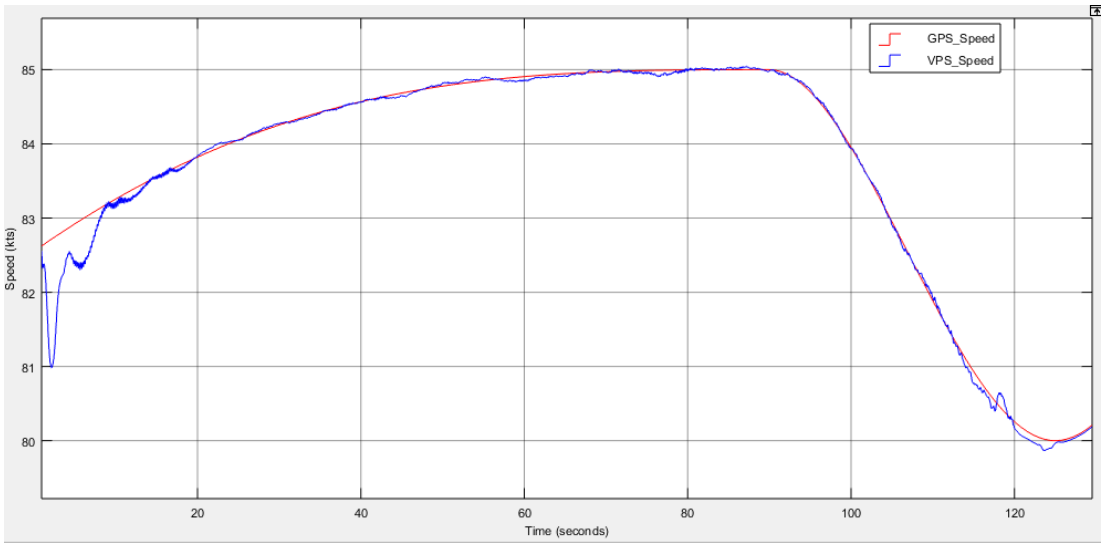


Figure 5-61 Variation of Speed for Case 4: 15 kts Cross Wind + Turbulence GPS Based Landing

Again, oscillations are observed in speed due to turbulence. Generally, turbulence more effective than side wind and it has a considerable effect on $\dot{h}$.

5.5.2 VPS Based Landing

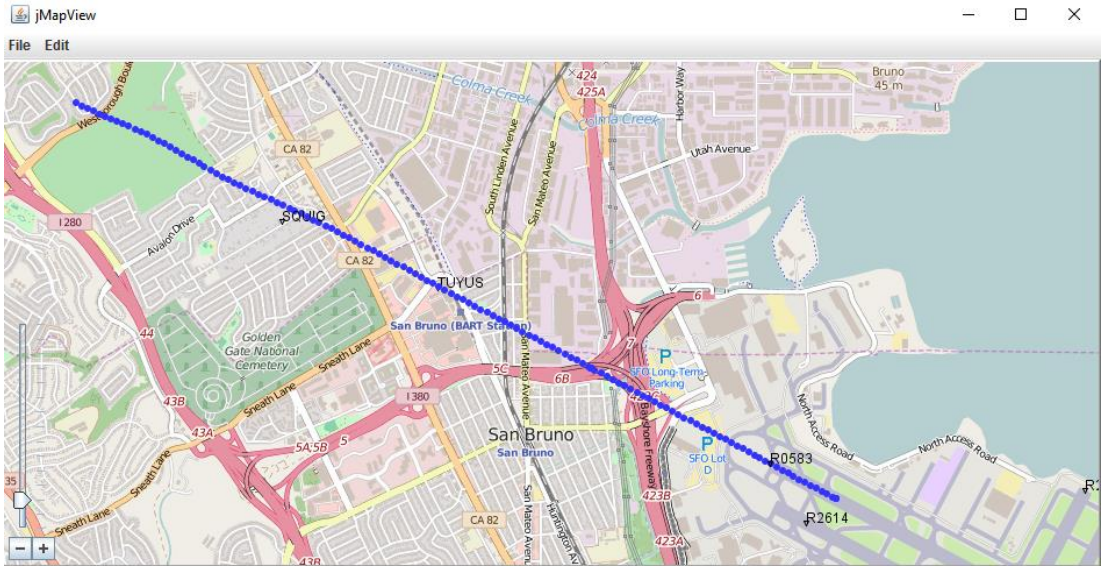


Figure 5-62 Tracking of the UAV Position While Landing for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

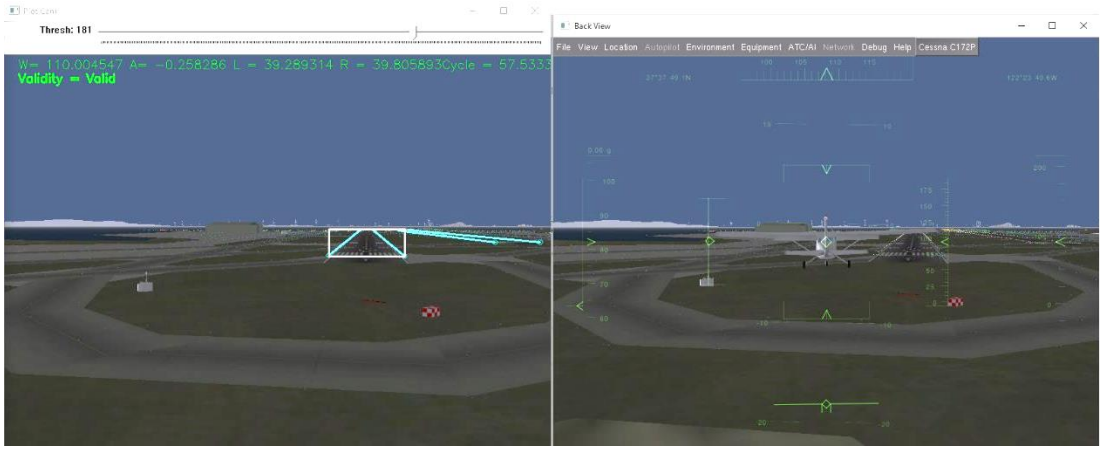


Figure 5-63 Vehicle Position and Orientation for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing. Left is Pilot Cam, Right is View from Back

Under turbulence and cross wind condition, landing autopilot successfully land the vehicle by using VPS. Cross track path is presented in Figure 5-62 and position of the UAV before landing is presented in Figure 5-63. Again, runway heading is 117.9° but due to cross wind, VPS course controller controls heading of the UAV to be less than runway heading value to keep cross track. As can be seen in Figure 5-63, runway detection algorithm detects runway successfully and angle of edges is very close to each other, which shows that UAV aligned with the runway center line. Course controller performance is presented in Figure 5-64. Decrab angle is within the range at touchdown which is defined in Table 4-2 Landing Autopilot Requirements.

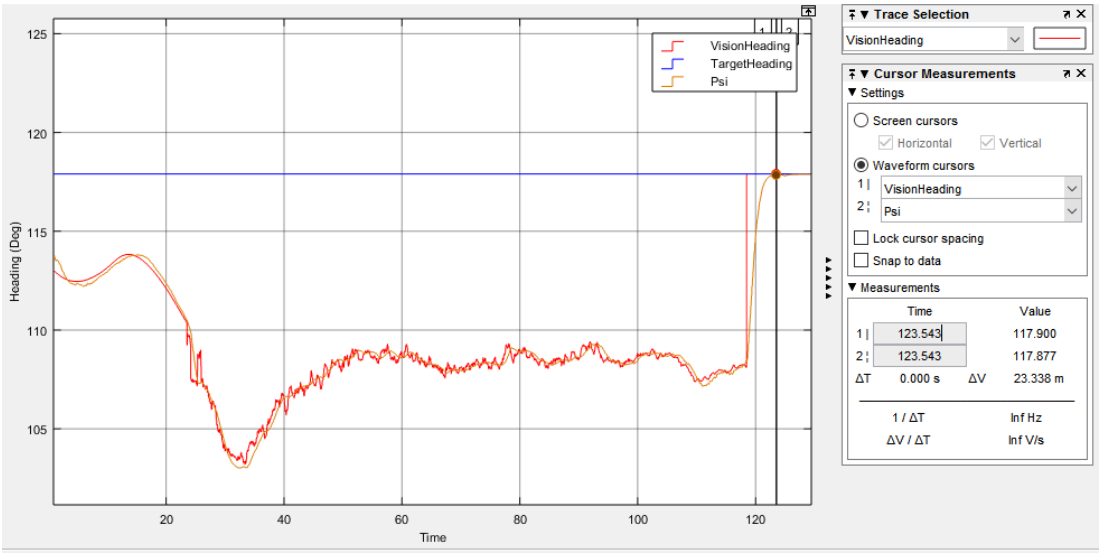


Figure 5-64 Cross Track Performance for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

Roll angle has a noisy characteristic due to sensitivity issues of VPS while away from the runway. With turbulence, this noisy characteristic is amplified more. After altitude is lower than 5 m, roll controller commanded to keep zero degree for decrab maneuver as can be investigated at the end of the simulation result in Figure 5-65.

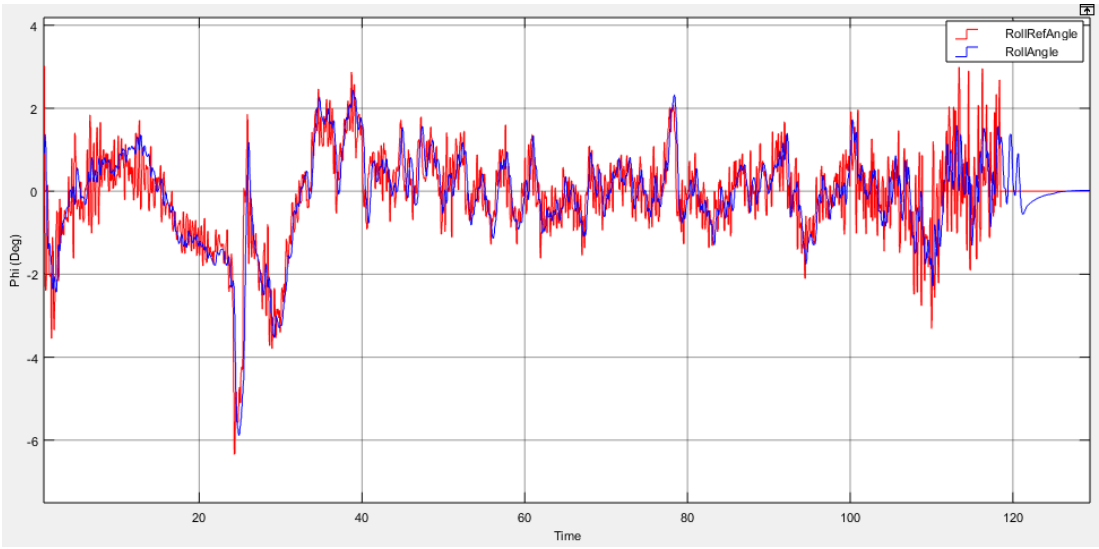


Figure 5-65 Roll Angle of the UAV for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

Although some small oscillations observed due to turbulence, general altitude trajectory tracking performance in Figure 5-66 is similar to previous cases. Under turbulence and crosswind conditions, UAV can land safely. Flare maneuver decrease $\dot{h}$ in Figure 5-67 and ground shock magnitude in Figure 5-68. Pitch angle in Figure 5-69 increase towards positive values due to flare maneuver. Compared to GPS Based Landing in case 4 context, ground shock and $\dot{h}$ are smaller. Rapid changes in pitch angle affects the speed output as expected. However, requirements stated in Table 4-2 Landing Autopilot Requirements is satisfied.

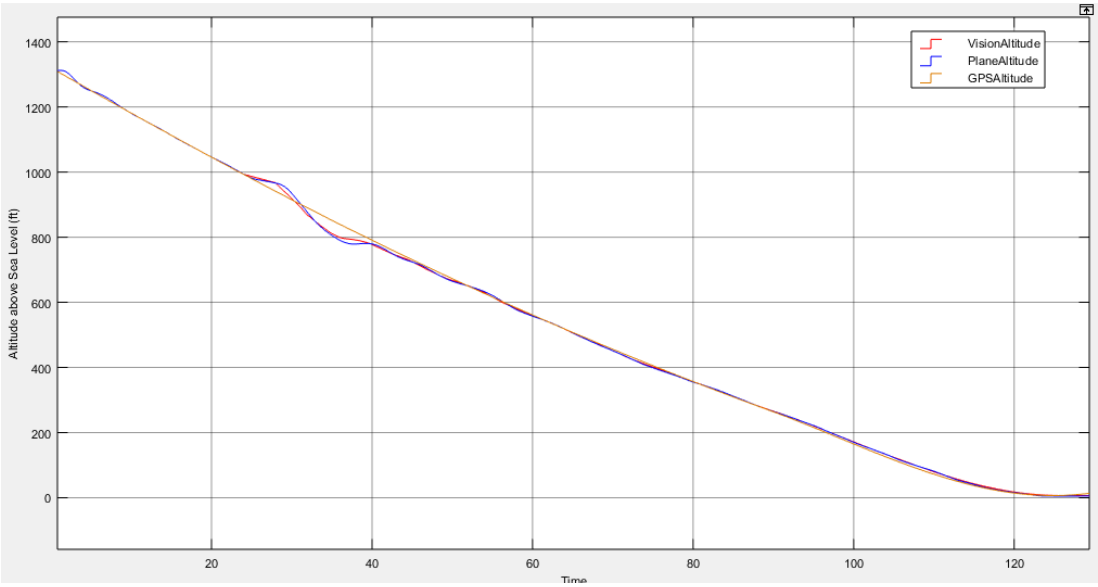


Figure 5-66 Landing Trajectory Path and Altitude of the UAV for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

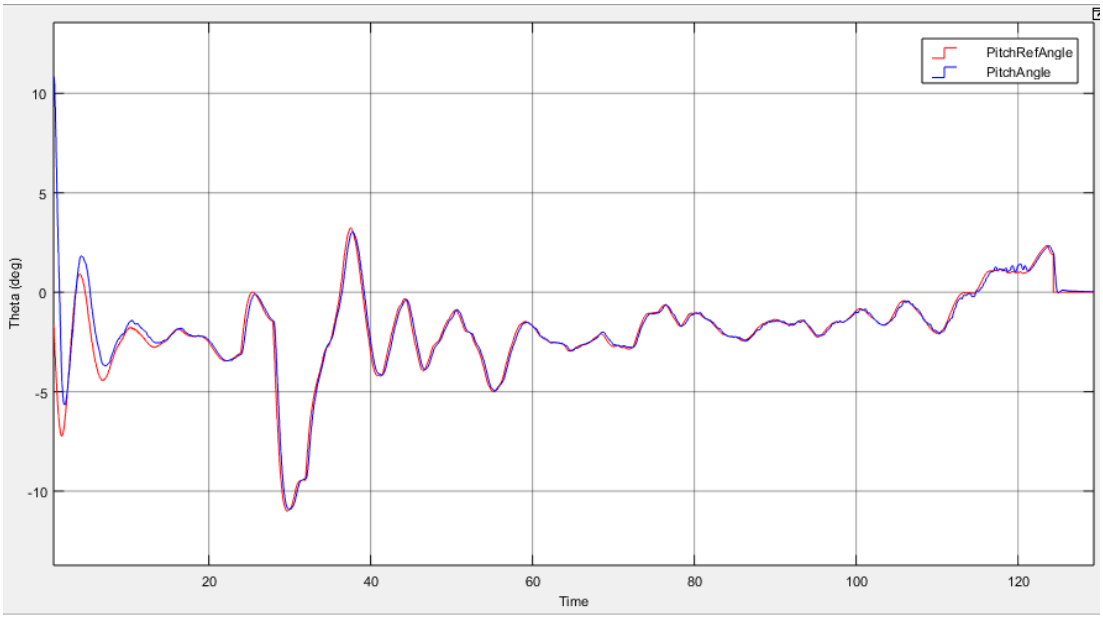


Figure 5-67 Variation of Theta for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

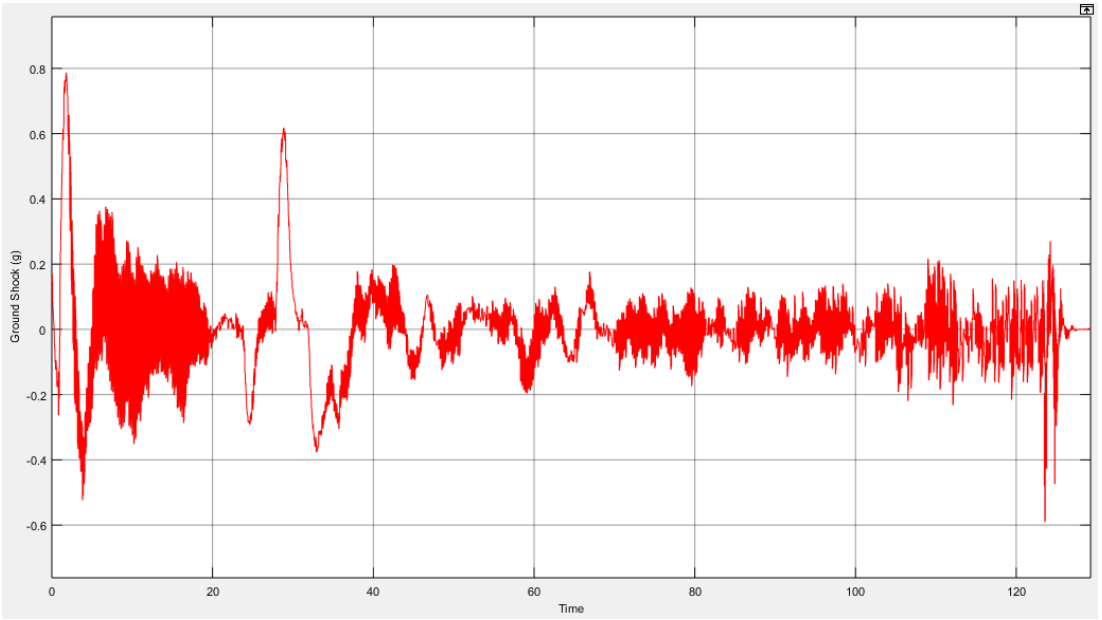


Figure 5-68 Ground Shock at Touchdown for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

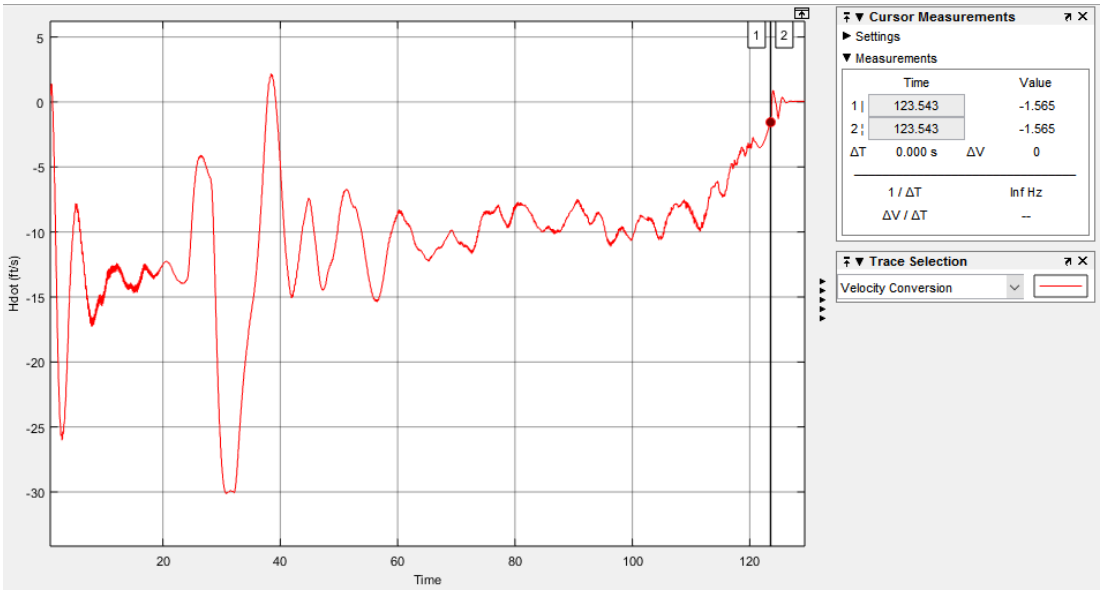


Figure 5-69 $\dot{h}$ for Case 4: 15 kts Cross Wind + Turbulence VPS Based Landing

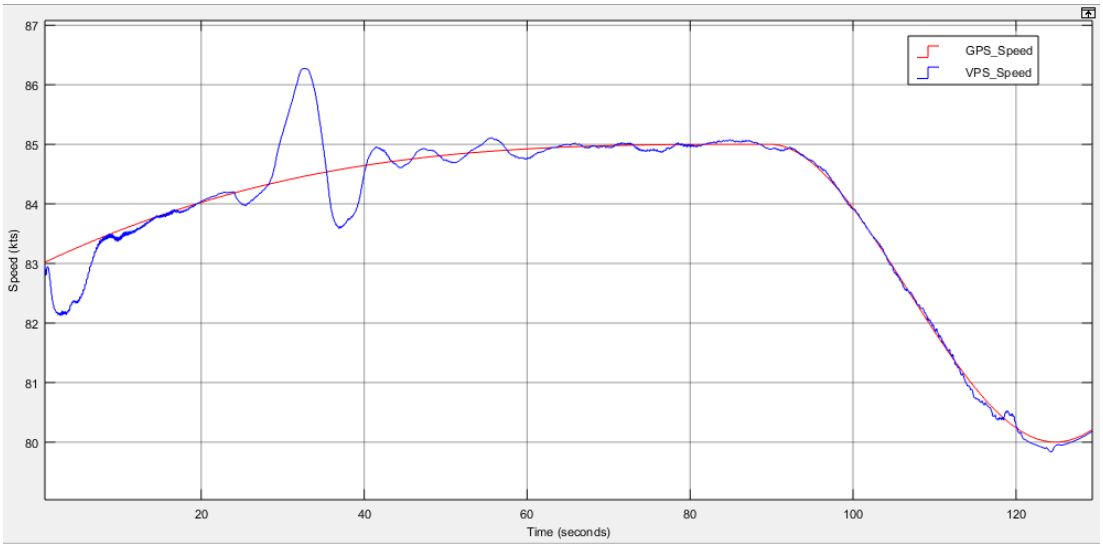


Figure 5-70 Variation of the Speed for Case 3: 15 kts Cross Wind with Cross Track Error VPS Based Landing

5.6 Case 5: 20 kts Nose Wind

If we take account previous cases, for this cases only VPS based landing is investigated because VPS has noisy characteristic and we are testing if landing autopilot can handle Table 4-3 Environmental Requirements for Landing. Side wind

was tested at case 2, 3 and 4. For this case, 20 kts nose wind requirement is tested. It is assumed that nose wind does not affect course controller so longitudinal performance is investigated.

As can be seen result below, altitude and speed reverences are kept successfully. Speed controller keeps airspeed so nose wind effect handled easily. However, as mention at Distance Calculation by Using VPS section at Chapter 4, VPS has limitation constraints while calculating the distance to TDP. After this constraints exceed, distance to TDP is calculated with the help of airspeed. To test this constraint, after altitude is lower than 2.5 m, nose wind canceled. As a result, pitch angle and $\dot{h}$ is affected because change in airspeed is also affects altitude interpolation. Therefore, we can say that, when UAV is out of operational region of VPS, altitude and speed references are highly dependent on airspeed value. Explained effect can be investigated towards the end of the simulation result presented in Figure 5-71, Figure 5-72, Figure 5-74 and Figure 5-75. Ground shock is relatively high with respect to other cases as presented in Figure 5-73. Overall, speed of UAV, $\dot{h}$ *and* pitch angle still satisfy requirements mentioned in Table 4-2 Landing Autopilot Requirements.

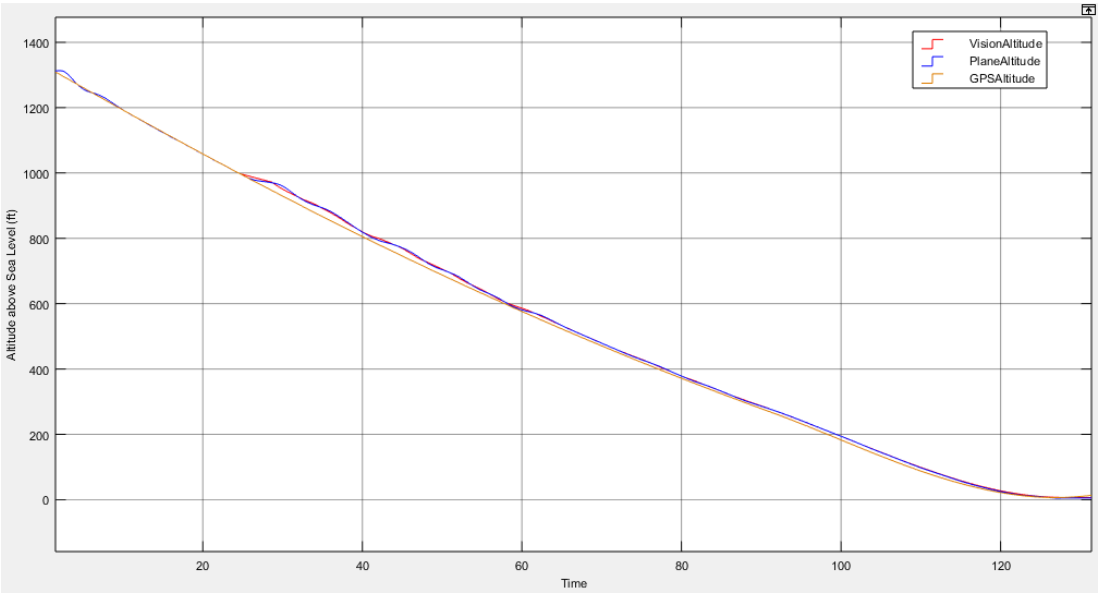


Figure 5-71 Landing Trajectory Path and Altitude of the UAV for Case 5: 20 kts Nose Wind VPS Based Landing

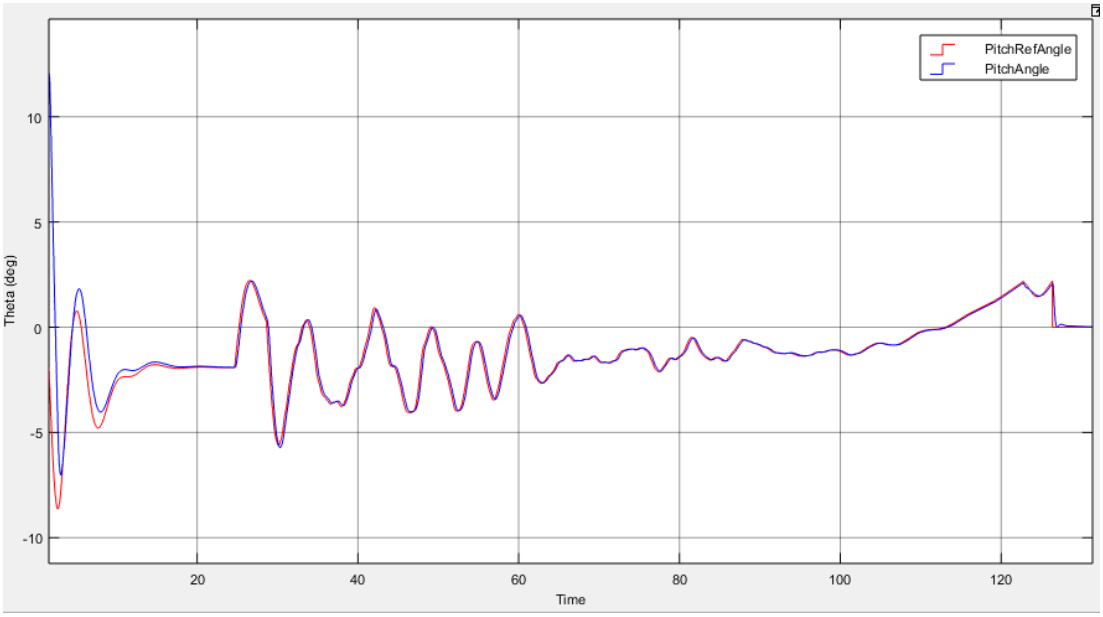


Figure 5-72 Variation of Theta for Case 5: 20 kts Nose Wind VPS Based Landing

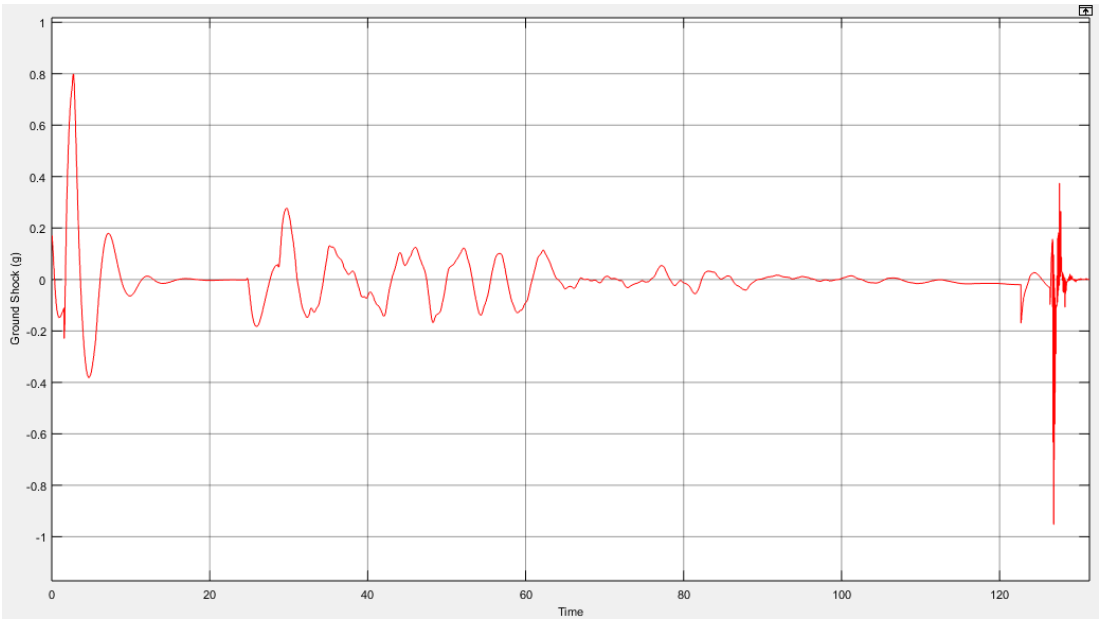


Figure 5-73 Ground Shock at Touchdown for Case 5: 20 kts Nose Wind VPS Based Landing

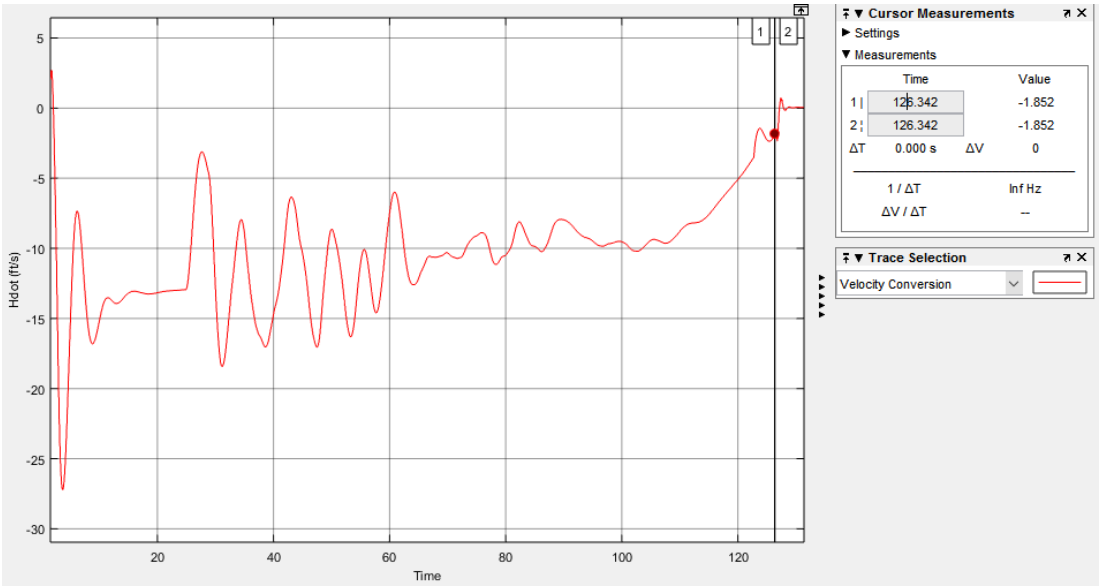


Figure 5-74 $\dot{h}$ for Case 5: 20 kts Nose Wind VPS Based Landing

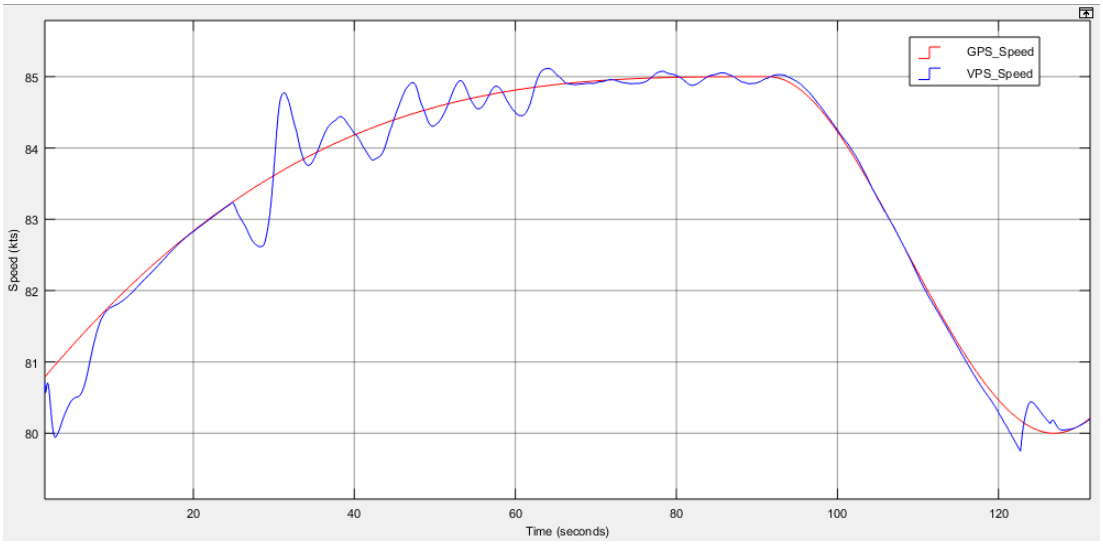


Figure 5-75 Variation of the Speed for Case 5: 20 kts Nose Wind VPS Based Landing

5.7 Case 6: 6 kts Tail Wind

Similar to Case 5, for this case VPS is used for position data source. Tail wind may decrease lift so it must be considered while developing controller. Our controller

keeps airspeed so if tail wind exist, vehicle speed with respect to earth will be higher than no tail wind case with amount of tail wind. This means that UAV will reach to TDP with higher earth speed. Simulation results show us that landing autopilot is works well to handle this environmental condition. Again, to test landing performance while out of VPS operation region, tail wind is cancelled after altitude is lower than 2.5 m. this time discontinuity effect of the wind disturbance is negligibly small because wind magnitude is lower than case 5. As can be seen figures below, all requirements are satisfied.

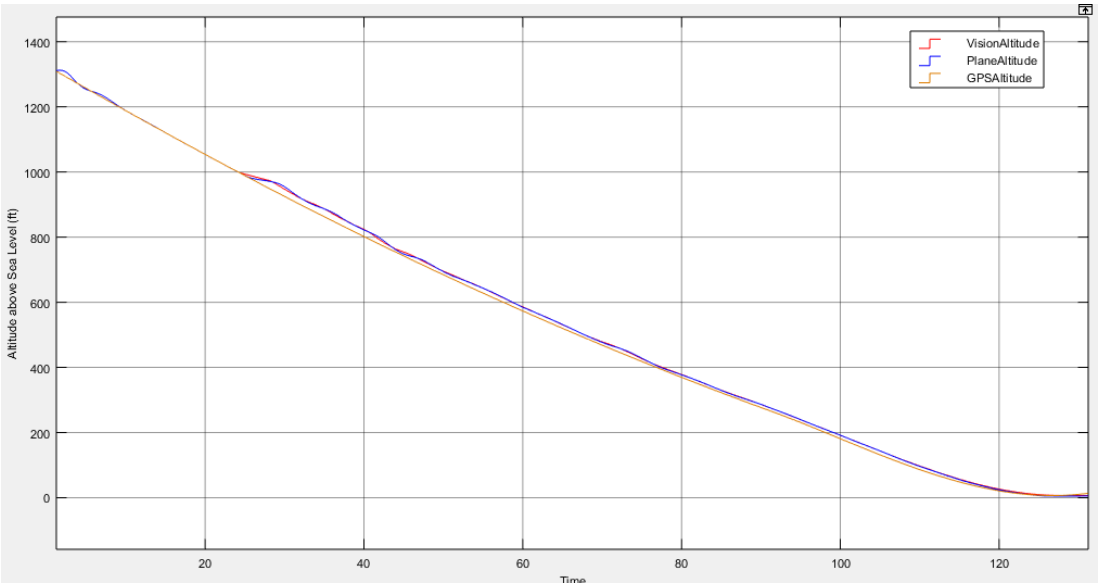


Figure 5-76 Landing Trajectory Path and Altitude of the UAV for Case 6: 6 kts Tail Wind VPS Based Landing

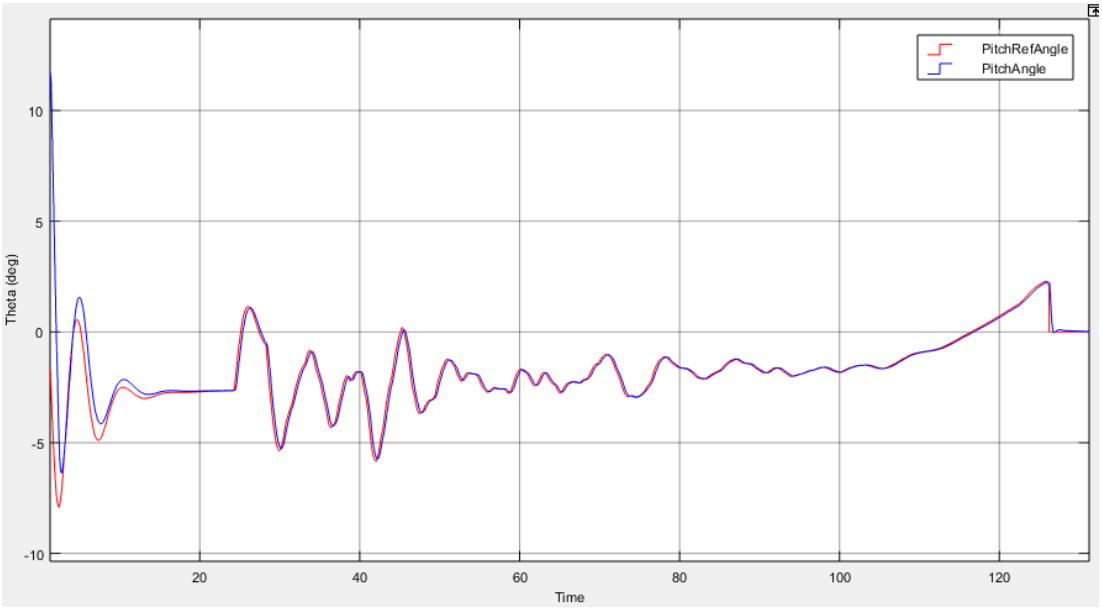


Figure 5-77 Variation of Theta for Case 6: 6 kts Tail Wind VPS Based Landing

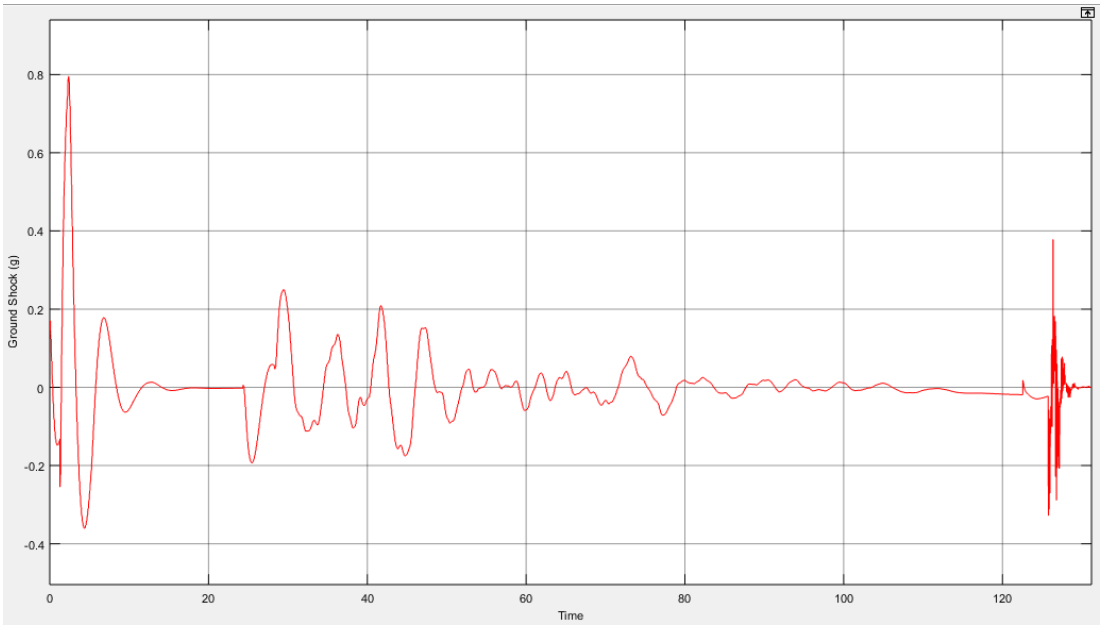


Figure 5-78 Ground Shock at Touchdown for Case 6: 6 kts Tail Wind VPS Based Landing

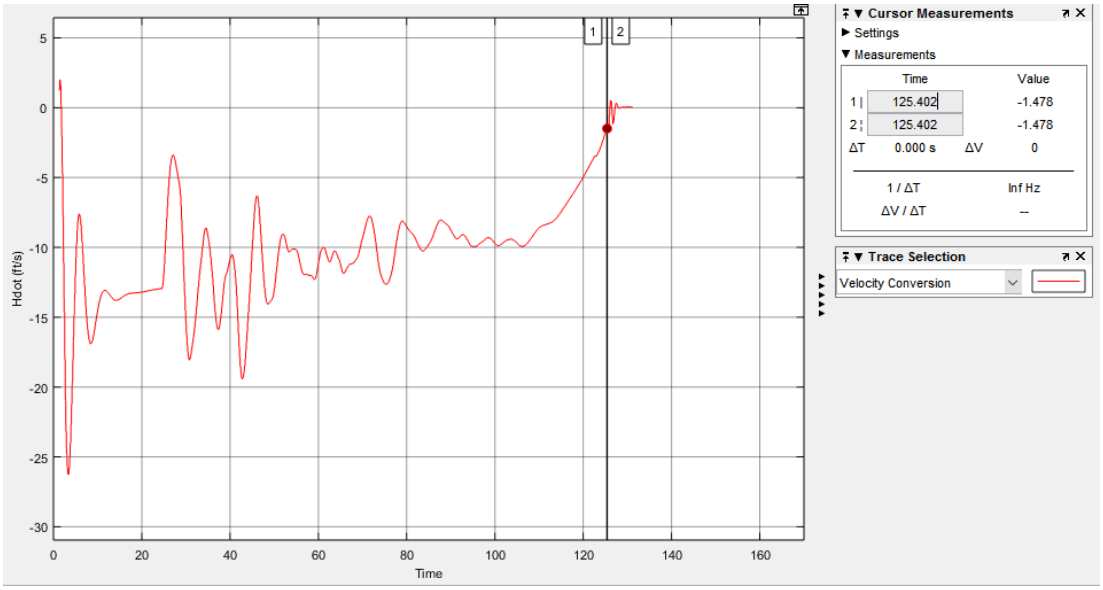


Figure 5-79 $\dot{h}$ for Case 6: 6 kts Tail Wind VPS Based Landing

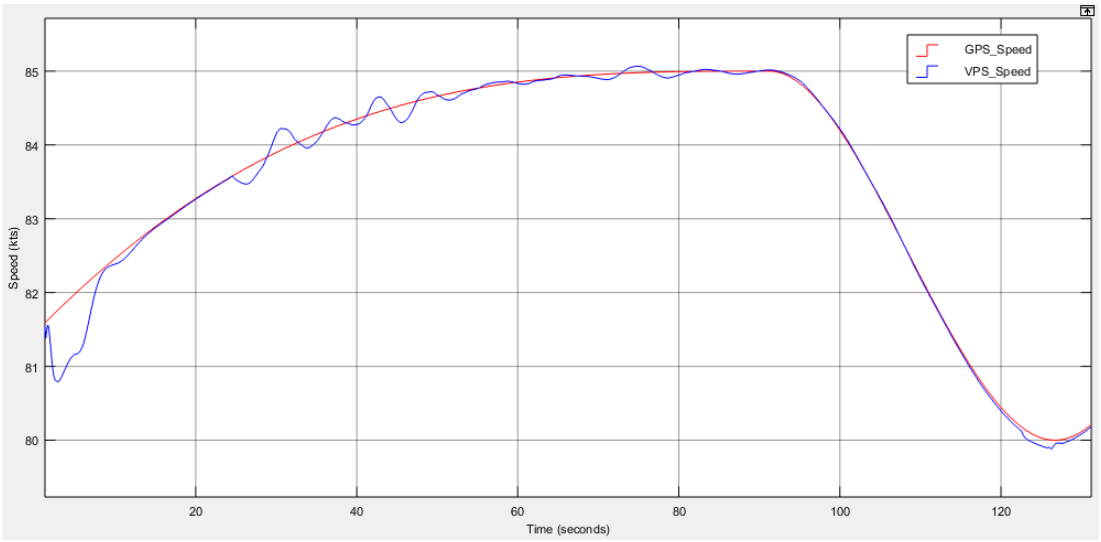


Figure 5-80 Variation of the Speed for Case 6: 6 kts Tail Wind VPS Based Landing

CHAPTER 6

CONCLUSION

6.1 Summary and Conclusions

In this thesis, main purpose was to proof that by image processing methods, position of the UAV can be calculated with respect to a known position and with that information, auto landing algorithms can be developed without needing GPS data. Therefore, a set of algorithms were suggested to auto land by using vision based positioning techniques alternative to GPS. As an output, a basic UAV controller is developed with many subsystems and relations between these systems are described. Ground control station interface is a user interface to control flight mode, flight parameters and create waypoints to navigation and guidance. Landing autopilot a designed to control the UAV by changing related control surfaces. Also it uses GPS or camera for position detection and gets generated waypoint list from GCS interface to manage guidance. Airspeed sensor, pitot tube, is used for detection of UAV speed, and barometric sensor is used for altitude determination of UAV for this project. Also for position determination, GPS receiver and camera connected to the UAV to obtain position information for control loops. Several simulations are done with different initial and different weather conditions with respect to predefined requirements.

For this design, Matlab/Simulink is used to design of landing autopilot; OpenCV libraries in C++ is used to developed image processing algorithms to detect runway from image frame and extract related information for calculations of position data; C# is used to developed GCS interface, which communicates with Simulink model

like runway detection application and sends control and mission messages to control subsystems running on UAV, generates and loads mission data for landing autopilot and activate camera or GPS for positioning source. FlightGear [10] is a free and open source flight simulator application and it is used to generate image frames to runway detection application. Also it supplies ground altitude level of the UAV to Matlab/Simulink. All aerodynamic and weather models run at Matlab/Simulink and obtained from Aerospace Blockset. These models is outputs of Rauw's work [4]. JMapView [39] application is an open source application, which is communicating with FlightGear and used for monitoring position of the UAV on map to test of the landing autopilot. FlightGear sends GPS information of the UAV to JMapView and this positions are shown on the map. It is used just analyzing position history of the UAV.

Systems developed for this thesis has multidisciplinary areas such as control theory, image processing, and software developing. All subsystems are developed as decoupled as possible to provide a development environment independently. With developed platform for this thesis, runway detector, landing autopilot and GCS interface can be developed further or different methods can be implemented as recommended in the next section.

6.2 Future Works

In this study, although it was a proof of concept work, accuracy of the runway detection algorithms and its input quality can be improved. One basic solution is increasing resolution of the FlightGear frames but this solution is not drastic one. As mentioned above, FlightGear is an open source project and to improve frame quality while away from the runway, a patch is coded into FlightGear source to control zoom and direction of the camera, however, this solution is left for future improvement due to limited time period and complexity of the project. By applying this method, runway can be detected without any resolution problem.

Also, while altitude level is lowering, runway width is starting to overflow from image frame which is an important problem. To solve this issue, runway lines may be detected to alignment of UAV with runway or stereo vision can be used so that field of view would be increase and also by using two camera, distance calculation can be done without the information of runway sizes. Also, sensor fusion algorithms can be implemented to detect position and orientation of the UAV while VPS is out of operation region. VPS is works until 140 m to touchdown and 26 feet above ground level therefore, up to touch down, error accumulation of the sensor fusion would be very small.

This work is a proof of concept, therefore, all simulations and test are done at simulator environment and therefore runway detection algorithm is designed for generated image frames from this simulators. With modification of runway detection algorithm and by using a cheap and easy to set up flying platform, auto landing algorithms can be applied in real world environment. In Figure 6-1 (a), night landing images from pilot camera of an aircraft obtained from web and vision algorithms are applied to it. Result of these algorithms to real world frame can be investigated in Figure 6-1 (b).

(a)

(b)

Figure 6-1 Result of Developed Vision Algorithms on Real World Frames

Also autonomous takeoff and modern control methods can be applied to current design architecture; an emergency flight plan can be developed in order to cancel landing if landing autopilot cannot manage to execute loaded waypoints for any reason such as weather conditions or impossible waypoint configurations encountered due to user fault. For environmental condition tests, profiled wind model may be applied to developed controller.

REFERENCES

[1] "Technology review: Britain’s blind landing begins to pay off," *New Scientist,* vol. 68, p. 398, 1975.

[2] R. Mola, "History of aircraft landing aids," [Online]. Available: http://www.centennialofflight.gov/essay/Government_Role/landing_nav/POL14.htm.

[3] H. D. a. F. W. Dunmore, "A radio system for blind landing of aircraft in fog," *National Bureau of standards,* p. 678–685, September 19, 1930.

[4] M. Rauw, A SIMULINK environment for Flight Dynamics and Control analysis - application to the DHC-2 'Beaver', TUDelft, 1993.

[5] D. Hughes, "United complete GPS Autoland Tests," *Aviation Week and Space Technology,* vol. 141, p. 68, 1994.

[6] "Jamming the Global Positioning System - A National Security Threat: Recent Events and Potential Cures," National Space-Based Positioning, Navigation, and Timing Advisory Board, 2010.

[7] D. Hambling, "GPS Chaos: How a $30 Box Can Jam Your Life," *New Scientist,* 4 March 2011.

[8] B. L. K. C. J. Juang, "Intelligent Fuzzy Systems for Aircraft Landing Control," in *Lecture Notes in Computer Science,* Berlin Heidelberg, Springer-Verlag, 2005, p. 851 – 860.

[9] "“Matlab Examples”," MathWorks, [Online]. Available: http://www.mathworks.com/examples/aeroblks/1005-fly-the-dehavilland-beaver. [Accessed 25 05 2016].

[10] "FlightGear Flight Simulator," [Online]. Available: http://www.flightgear.org/about/.

[11] Kargın V., Design of Autonomous Landing Control Algorithm for a Fixed Wing UAV, M.S. thesis, Aerospace Eng. Dept., Middle East Technical University, 2007.

[12] Cho A. et al., "Fully Automatic Taxiing, Takeoff and Landing of a VAV Using a Single-Antenna GPS Receiver Only," *International Conference on Control, Automation and Systems*, pp. 821-825, 2007.

[13] Attar, M., Wahnon, E., Chaimovitz, D, "Advanced Flight Control Technologies for UAVs," in *2nd AIAA "Unmanned Unlimited" Systems, Technologies and Operations*, 2003.

[14] C. S. R. a. J. S. K. Senthil Kumar, "Design and Simulation of Blending Function for Landing Phase of a UAV," *Defence Science Journal*, vol. 58, no. 3, pp. 315-326, May 2008.

[15] S. ARIBAL, EVELOPMENT OF AN AUTOPILOT FOR AUTOMATIC LANDING OF AN UNMANNED AERIAL VEHICLE, MS.Thesis, ELECTRICAL AND ELECTRONICS ENGINEERING, MIDDLE EAST TECHNICAL UNIVERSITY, 2011.

[16] V. K. O. E.O.Kovalevskiy, "AIRCRAFT LANDING FLARE," *3.2.10 PROCEEDINGS, THE SIXTH WORLD CONGRESS, "AVIATION IN THE XXI-st CENTURY", ,* vol. 2, p. 3.2.10, September 23-25, 2014.

[17] S. Singh and R. Padhi, "Automatic Path Planning and Control Design for Autonomous Landing of UAVs using Dynamic Inversion," in *American Control Conference, St. Louis, MO, 2009, pp. 2409-2414.*, 2009.

[18] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," *IEEE International Conference on Robotics and Automation*, vol. 3, no. ICRA'02, p. 2799–2804, 2002.

[19] 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, 2014.

[20] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," *IEEE International Conference on Robotics and Automation*, vol. 2, no. ICRA '01, p. 1720–1727, 2001.

[21] Andrew Miller, Mubarak Shah, Don Harper, "Landing a UAV on a Runway

using image registration," *IEEE International Conference on Robotics and Automation*, pp. 182-187, 2008.

[22] C.-f. Ding Meng, "A Method to Recognize and Track Runway in the Image Sequences Based on Template Matching," *IEEE*, pp. 1218-1221, 2006.

[23] Alison A.Proctor, Eric N.Johnson, " Vision-only Approach and Landing," *AIAA Aerospace Sciences Meeting and Exhibit*, pp. 1-10, 2005.

[24] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, S. Longhi, "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 233 - 257, 2010.

[25] M. Rauw, FDC 1.2 - A Simulink Toolbox for Flight Dynamics and Control Analysis, Delft University of Technology, 2nd Edition, 2001.

[26] Haralick, R.M., STANLEY R. STERNBERG, AND XINHUA ZHUANG, "Image Analysis Using Mathematical Morphology," *Pattern Analysis and Machine Intelligence*, Vols. PAMI-9, no. 4, pp. 532 - 550, 1987.

[27] R. E. W. R. C. Gonzalez, Digital Image Processing. 2nd Ed., New Jersey: Prentice-Hall, Inc, 2002 .

[28] 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*, vol. 24, no. 6, p. 381–395, 1981.

[29] D.-q. T. N. S. Fei Lia, "Vision-based pose estimation of UAV from line correspondences," *Procedia Engineering* , vol. 15, pp. 578-584, 2011.

[30] A. Visioli, "Practical PID Control," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2217-2218, 2006.

[31] K. J. Å. a. T. Hägglund, PID Controllers: Theory, Design, and Tunning, NC 27709: ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park , 1995.

[32] I. P. Sigurd Skogestad, MULTIVARIABLE FEEDBACK CONTROL Analysis and design, JOHN WILEY & SONS, August 29, 2001.

[33] D. McLean, Automatic Flight Control Systems, Prentice Hall Inc., 1990..

[34] M. Frølich, Automatic Ship Landing System for Fixed-Wing UAV, Master of Science in Cybernetics and Robotics, Norwegian University of Science and Technology, 2015.

[35] G. W. a. I. Alfy, "Monotonic Cubic Spline Interpolation," in *In Proceedings of the International Conference on Computer Graphics (CGI '99)*, Washington, DC, USA, 1999.

[36] C. Moler, Numerical Computing with MATLAB, Natick, Massachusets: The MathWorks, Inc., 2004.

[37] "Movable Type Scripts," [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html.

[38] Jiajia Shang, Zhongke Shi , "Vision-based Runway Recognition for UAV Autonomous Landing," *IJCSNS International Journal of Computer Science and Network Security,* vol. 7, no. 3, 2007.

[39] "JMapView," [Online]. Available: http://jmapview.sourceforge.net/.

[40] "WindTriangle," [Online]. Available: http://www.delphiforfun.org/programs/math_topics/WindTriangle.htm. [Accessed 18 10 2015].

[41] "AIRCRAFT LANDING FLARE," *PROCEEDINGS, THE SIXTH WORLD CONGRESS, "AVIATION IN THE XXI-st CENTURY"*.

APPENDIX A

A. GROUND CONTROL STATION (GCS)

UAV has a complex architecture and operates autonomously in dynamically changing environment, flight missions, and requirements. To be able to satisfy flight safety and flight requirements, controlling cooperation of different subsystems on the UAV is required. Here, importance of GCS is arises. GCS sets flight mode, flight mission, and controls way of working of the UAV autopilot controller. Flight mode is needed to set if control of the UAV is wanted to be managed manually by user (pilot on the ground) RC_Mode must be selected. To activate autopilot to hold references comes from pilot joystick, then Sticks_Mode must be selected. Similar to Sticks_Mode, if UAV is wanted to be controlled by setting references entered to GCS interface, then GCS_Mode must be selected. This mode is especially used for the testing of autopilot. Therefore, any step input can be applied individually for all channels. To activate landing autopilot, FMS_Mode must be selected. If this mode is selected, then waypoints, which are generated and loaded previously by GCS, is started to be applied by one by.

In this chapter, function of GCS will be described by introducing user interface and functionality of that interface. Messages that is sent to Matlab/Simulink will be described as well. Finally, generation of waypoints will be introduced.

A.1 Function and interface of GCS

GCS user interface is developed by using C# language and user interface can be seen in Figure A- 1. GCS user interface communicates with Simulink model via UDP

protocol. By using GCS user interface, two messages can be sent to UAV which are named as GAM_CONTROL_DATA and GAM_MISSION_DATA. GCS user interface accepts inputs to fill related messages. Messages sent from GCS user interface are aperiodic and sending operation is triggered by pressing “Control Message” and “Mission Message” buttons. Waypoints can be saved with XML format or previously saved waypoints can be loaded to GCS user interface for GAM_MISSION DATA message.

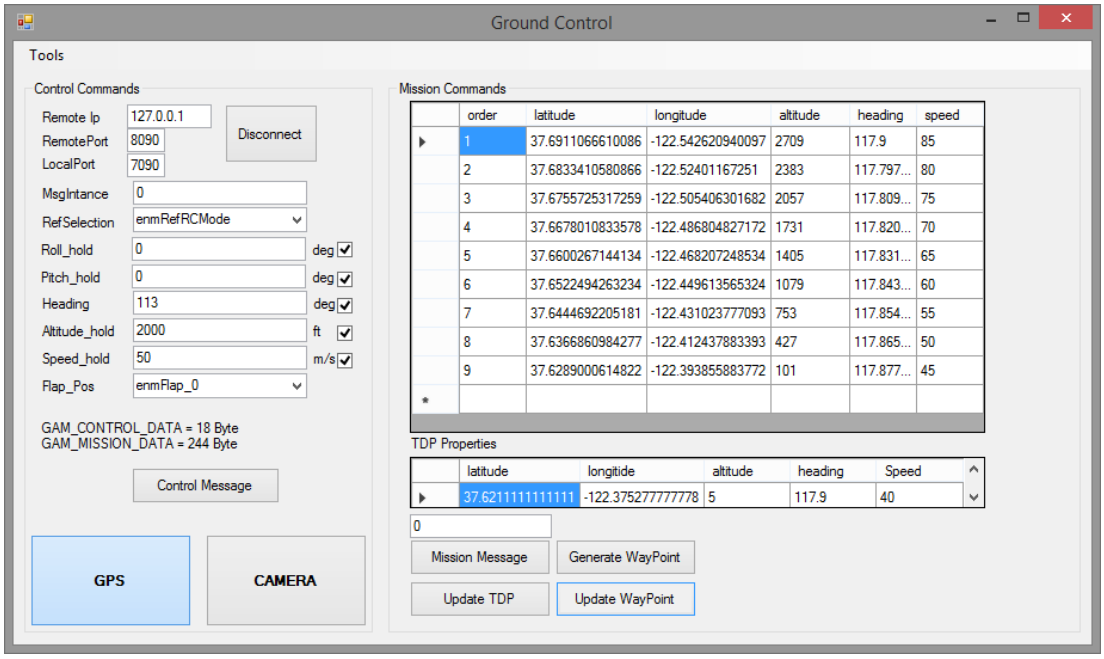


Figure A- 1 Ground Control Station Interface

Waypoint generator can be used for generation of up to ten waypoint in the same direction. Waypoint generation is done by pressing “Generate Waypoint” button. After pressing this button, “Generate Waypoint” interface in Figure A- 4 is popping-up. By waypoint generator interface, user sets the parameters as follows:

- Waypoint number that will be generated,
- Distance of each successive waypoint
- Speed of UAV when touch down occurs,
- Altitude descent rate in degree.

Modification by selecting related waypoint is allowed and after modification, waypoint list can be saved for using that list in the future. Waypoints are generated with respect to TDP properties and generated waypoints are shown at the GCS user interface. Speed of the UAV is decreased 5 mps for every waypoints to satisfy the target speed while touching down the runway. Also altitudes are calculated with respect to descent degree and distance of successive waypoints. Whatever selected descent degree is, last target altitude of the waypoint comes before TDP is calculated with respect to 3 degree to keep glide slope angle in safe region for landing.

Altitude generation result with respect to 10 waypoint (includes TDP), 1850 meter distance to each, and 10° descent degree can be investigated in Figure A- 1 and Figure A- 2. Altitude descents with 10° between waypoints 1-9 but descent degree between waypoints 9-10 is 3°.

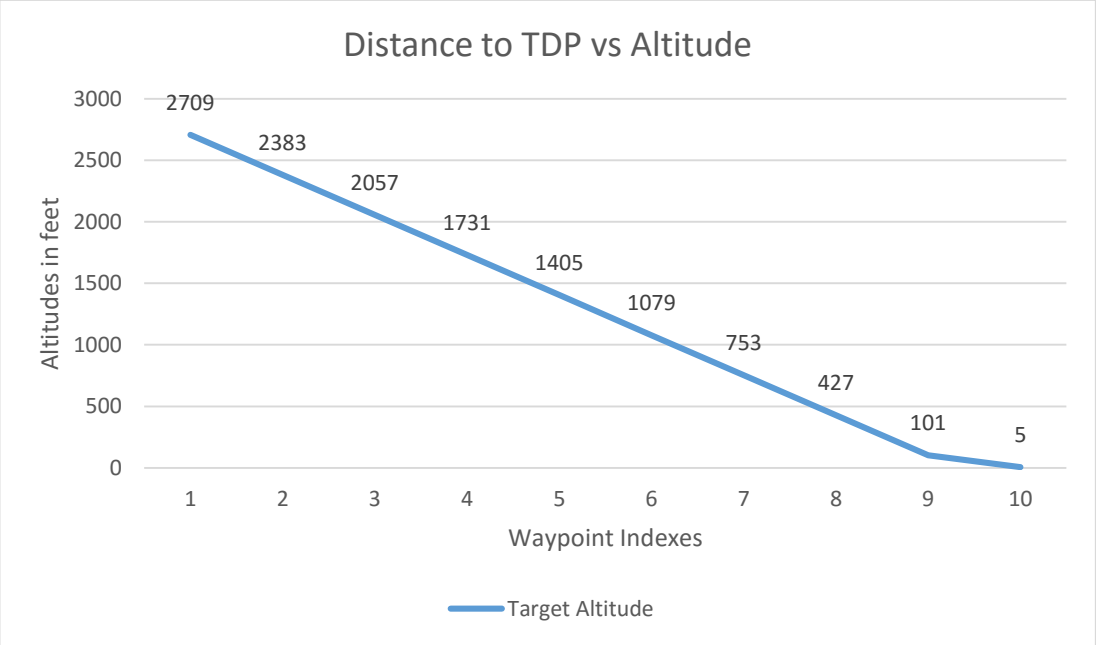


Figure A- 2 Altitude Generation Example

Altitude calculation algorithm and calculation details can be seen in Figure A- 3.

- Altitude of the first waypoint before TDP,

$$glideSlopeAltitude = tan\big(deg2rad(3°)\big) * Distance2Each$$

- Altitude of the other waypoints,

$$Altitude = tan(deg2rad(RunwayDescentAngle)) * Distance2Each * (NumOfWP - index\ -\ 1) + glideSlopeAltitude$$

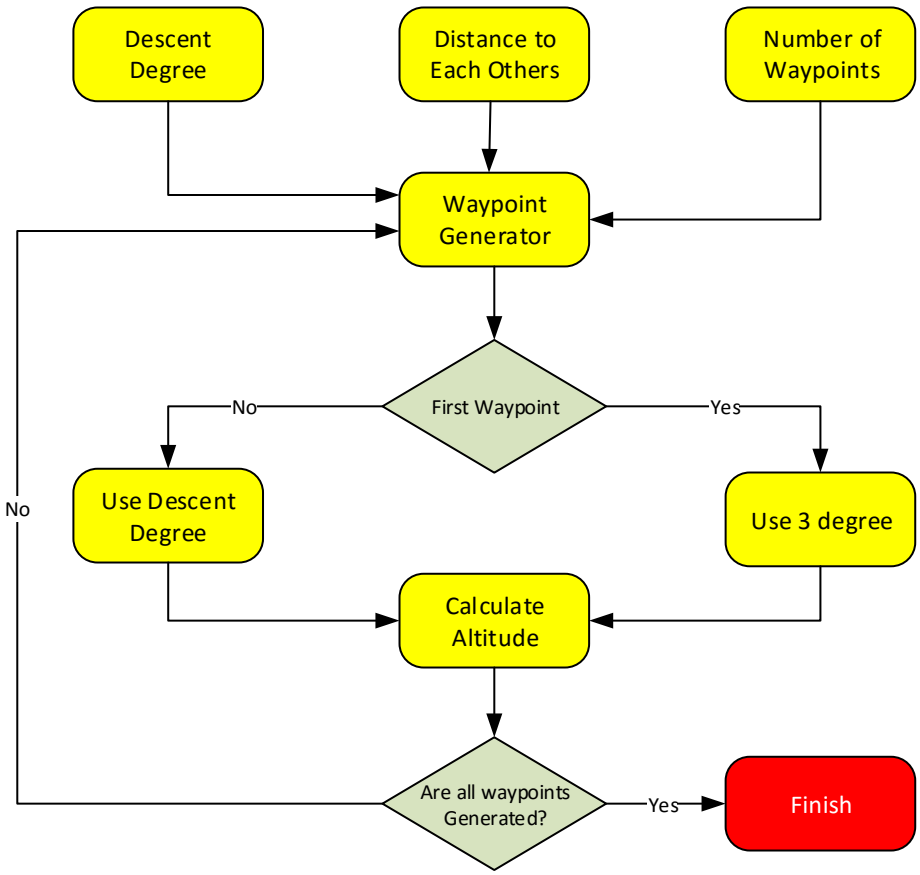


Figure A- 3 Altitude Calculation Algorithm

Headings are calculated with respect to two successive waypoint direction by using their latitude and longitude information. Generation example results for nine waypoints plus TDP point, which is totally ten waypoints, can be seen in Figure A- 2

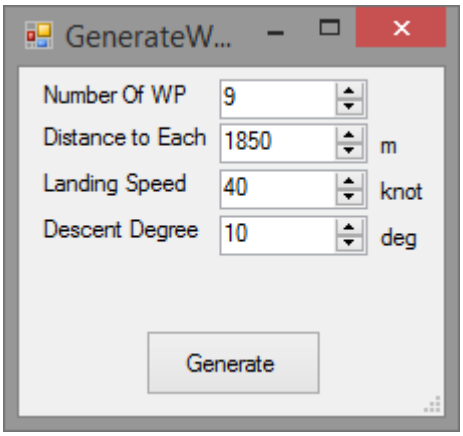


Figure A- 4 Waypoint Generator

Heading calculation is done between [Lat1 Lon1] and [Lat2 Lon2] by following:

$$Var = Acos\Big(Sin(lat2) * Sin(lat1) + Cos(lat2) * Cos(lat1) * Cos\big((lon2 - lon1)\big)\Big)$$

$$Heading = Asin\left(Cos(lat2) * \frac{Sin\big((lon2 - lon1)\big)}{Sin(Var)}\right)$$

Latitude and longitude of the waypoints are generated with respect to runway heading and distance of waypoint to each other as follows:

$$\varphi 2 = \mathrm{asin}(\sin(\varphi 1) * \cos(\delta) + \cos(\varphi 1) * \sin(\delta) * \cos(\theta))$$

$$\lambda 2 = \lambda 1 + atan2(\sin(\theta) * \sin(\delta) * \cos(\varphi 1), \cos(\delta) - \sin(\varphi 1) * \sin(\varphi 2))$$

Where φ is latitude, λ is longitude; θ is the bearing (clockwise from north); δ is the angular distance d/R; d being the distance travelled, R the earth's radius [37].

A.2 Control Commands

GAM_CONTROL_DATA includes flight mode information, hold reference values for GCS mode, and position source (GPS or Camera) for FMS_Mode. With the checkboxes at the interface in Figure A- 1, related controllers can be activated or deactivated. If controller of an axes is deactivated, control of this axes left to pilot inputs directly. This feature is used for testing purposes especially while validate performance of landing autopilot to disturbances while trying to track generated trajectories and testing coupling effect of controllers. Message table of GAM_CONTROL_DATA can be seen in Table A - 2 Message Table of GAM_MISSION_DATA.

Name	**Type**	**Unit**	**Description**
MessageInstance	integer	N/A	Increase one by one with message sent operation
Label	integer	N/A	Label gives information of discrimination against other messages. Label for Control Command is 1
ControlEnableSelections	Integer	N/A	1st bit -> On/Off Roll Controller 2nd bit -> On/Off Pitch Controller 3rd bit -> On/Off Course Controller 4th bit -> On/Off Altitude Controller 5th bit -> On/Off Speed Controller
ReferanceSelection	Integer	N/A	0. RC_Mode 1. Sticks_Mode 2. GCS_Mode 3. FMS_Mode
RollHoldRef_deg	Float	deg	Roll reference command to hold autopilot. Used at GCS mode
PitchHoldRef_deg	Float	deg	Pitch reference command to hold autopilot. Used at GCS_Mode

HeadingHoldRef_deg	Float	deg	Heading reference command to hold autopilot. Used at GCS_Mode
AltitudeHoldRef_ft	Integer	feet	Altitude reference command to hold autopilot. Used at GCS_Mode
SpeedHoldRef_mps	Integer	mps	Speed reference command to hold autopilot. Used at GCS_Mode
FlapPosition	Integer	deg	0. 0 degree 10. 10 degree 20. 20 degree
PositionSource	Integer	N/A	0. Use_GPS 1. Use_Camera

Table A - 1 Message Table of GAM_CONTROL_DATA

By ReferanceSelection variable, following flight modes can be selected:

- RC_Mode: Control of the UAV is left to pilot inputs. No autopilot is running.
- Sticks_Mode: Hold autopilot is running to keep pilot inputs.
- GCS_Mode: Hold autopilot is running to keep reference values come from GAM_CONTROL_DATA. Related references set from user by GCS user interface.
- FMS_Mode: Inner loop of the landing autopilot is running to keep references supplied by outer loop of the landing autopilot. Outer loop generate references with respect to waypoint inputs which are coming by GAM_MISSION_DATA message.

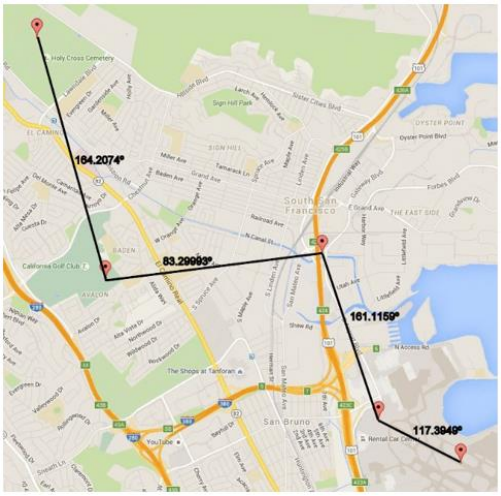
A.3 Mission Commands

GAM_MISSION_DATA includes up to ten waypoints with information of

- Latitude,
- Longitude,
- Altitude,
- Heading,
- Speed of UAV.

Generated waypoints and entered TDP could be saved by “Update TDP” and “Update Waypoint” buttons (which can be seen in Figure A- 1) with XML format to be able to use again. Waypoints could be modified, entered manually or could be generated by Generate Waypoint button and new GUI for waypoint generation is created as can be seen in Figure A- 4.

Heading values are calculated from GCS user interface with respect to consecutive waypoints latitude and longitude information as can be seen as in Figure A- 5. While landing autopilot executes waypoint requirements one by one, cross track is keeping with respect to these heading directions. Since GCS does not have information of current location of UAV, first heading is set to runway heading value.



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 A- 5 Waypoints and Calculated Headings in Degree

Details of GAM_MISSION_DATA is presented at Table A - 2. GAM_MISSION_DATA includes size of waypoint list so that landing autopilot execute that much waypoints.

Name	Type	Unit	Description
MessageInstance	integer	N/A	Increase one by one with message sent operation
Label	integer	N/A	Label gives information of discrimination against other messages. Label for Mission Command is 2
NumOfWaypoint2Sent	Integer	N/A	NumOfWaypoint2Sent gives information to landing autopilot about how many waypoints will be executed.
GAM_Mission_WayPoint[]	Composition Array with size 10	N/A	Size of GAM_Mission_WayPoint composition array is 10. Details of the composition can be seen in Table A - 3

Table A - 2 Message Table of GAM_MISSION_DATA

Name	Type	Unit	Description
Order	byte	N/A	Generation order of the waypoints
Longitude	double	N/A	Target longitude of waypoint
Latitude	double	N/A	Target latitude of waypoint
Altitude	UInt16	Feet	Target Altitude of waypoint
Heading	Single	Degree	Target Heading of waypoint to next waypoint
Speed	byte	mps	Target speed of waypoint

Table A - 3 Message Table of GAM_Mission_WayPoint Composition