אוניברסיטת בן-גוריון בנגב

Ben-Gurion University of the Negev

הפקולטה למדעי ההנדסה

המחלקה להנדסת חשמל ומחשבים

Faculty of Engineering Science

Dept. of Electrical and Computer Engineering

פרויקט הנדסי שנה ד'

Fourth Year Engineering Project

דו"ח התקדמות

Progress Report

טיסה וניווט אוטונומיים של רחפנים ללא מערכת איכון גלובלית

Autonomous flight and navigation of UAV without GPS

Project number:	**p-2016-001**		**מספר הפרויקט:**
Students (Name & ID):	Asaf Sarid 301465258 Ohad Cohen 200654606	אסף שריד 301465258 אוהד כהן 200654606	**סטודנטים (שם ו ת.ז.):**
Supervisors:	Prof. Guterman Hugo Dr. Zohar Ilan	פרופ' הוגו גוטרמן ד"ר אילן זוהר	**מנחים:**
Sponsors:			**תומכים:**
Submitting date:	03/04/2016		**תאריך הגשה:**

Table of Contents

1. Introduction

Our project goal is to allow UAV to navigate from a known starting point to a given target print and avoiding from obstacles, by using optical flow technique and distance sensors. We have already implemented flight controller, which receive the desired coordinates and fly the UAV to this target point. The feedback of the measured location and angles are now come from analyzing the physics of the UAV, but in real world this is not enough since we have mechanic deviation and drift error. In addition, we designed and implemented optical flow algorithm, that process the stream from the camera and estimate out location. This information is sent to the controller as feedback instead the data from the GPS.
For the optical flow processing to be accurate, we need to supply it the real angles of capturing and the distance from the ground. This information is read from APM device- which include internal IMU and added Ultrasonic sensor.

Our next steps are:

- Integrate sonar measurements into the optical flow algorithm.
- Implement Kalman filter for sonar and optical flow filtering.
- Add GPS reading infrastructure to our code.
- Compile and run the program on the mini-pc and assert it is working without UAV.
- Optimize the program for maximal accuracy with minimum CPU utilization.
- Perform tests and simulate more complex scenarios to make sure that our controller can handle all of them.
- Tests with the UAV (changing and upgrading the controller during this step).

2. Optical Flow

Few methods of optical flow were tested, and after tests and analyzing we chose the 'Farneback' method. This method spans a grid on the captured frame, and calculate the distance each point in the grid has traveled between every two following frames [1]. We as well used [2] to compare between different optical flow algorithms.
The advantage of this method is the fact that it takes many points, in a fixed location- this reduce the processing effort (other algorithms finds different points each iteration) and is more suitable to measuring dynamic surface (and not cars or peoples that are moving in a static frame)- like our situation. [1]
The accuracy of the optical flow algorithm depends on many parameters and conditions:

- The camera must be calibrated.
- We need to know the exact angles of the capturing.
- The captured surface pattern and height consistency.
- The distance from the surface.
- The method selected.
- The density of the processed points.
- Etc.

We want to maximize the accuracy (with minimal effort from the controller), hence we need to get the current angles and distance from external device and transfer it into the algorithm.

2.1. From Camera coordinates to World coordinates

Our camera is installed on the UAV, and the angles of the camera changing according the UAV. In order to know our location using the camera, we need to transform the original captures frames from the UAV plane to a plane that is parallel to the surface (Pitch and Roll angles or zero).

We get the current angles of the capturing (and the UAV, since the device and camera will be connected to it) from the APM (see section 3). Then we use this angles to change the perspective of the optical flow processing, by using OpenCV function called- ‘warpPerspective’ [3] that apply a perspective transformation to an image. This allows us to keep the inaccuracy low, and take into account the angles of the UAV at any given moment. The function only changes the perspective of the image, so we created another function that limits the processing of the optical flow to the projection of the image. In measurements we did we can see that the accuracy of the location increases when using the angles of the UAV to the calculations.

2.2. Distance traveled calculation

In order to know what is our real location, we must transfer the data from the optical flow algorithm into distance in meters. This calculation influenced by the height of the UAV, the resolution of the camera and other parameters.

Note: The Alpha parameter was determined by our experiments with the camera. We set the camera within 1 meter from a screen, and measured what is the length and the width that it captures on the screen.

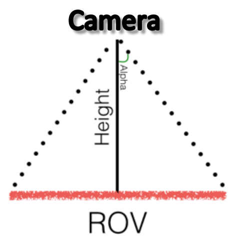


Figure 1- Range Of View

This is the calculation process:

2.2.1. Input data: Distance of each point in pixels

The optical flow algorithm outputs flow image with the distance each point traveled in pixels (X and Y axes). We are calculating the average of the frame in both axes.

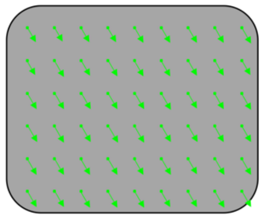


Figure 2- Stage 1

2.2.2. Input data: Distance of the frame in pixels

With use of trigonometry equations, we transfer this data to distance in meters, these are the two equations:

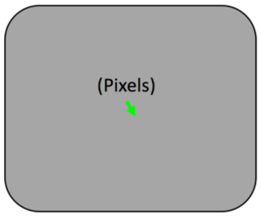


Figure 3- Stage 2

$$Range\ Of\ View: \quad ROV(m) = 2 \cdot \tan(\alpha) \cdot Height\ (m) \tag{1}$$

$$X(m) = X(pixels) \cdot \left(\frac{ROV(m)}{PixelsOfCamera(pixels)} \right) \tag{2}$$

2.2.3. Input data: Distance of the frame in meters

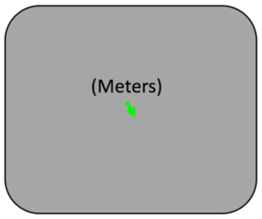


Figure 4- Stage 3

3. APM IMU and Range Finder.

APM (ArduPilot Mega) is a professional quality IMU autopilot that is based on the Arduino Mega platform. This autopilot can control fixed-wing aircraft, multi-rotor helicopters, as well as traditional helicopters [4]. This device supplies real-time angles of the camera (and UAV) to the optical flow algorithm. It has 3 axis gyros, 3 axis accelerometers, altimeter and a 5 Hertz GPS module (externally). The device contains 2 processors, internal flash memory, and support a set that includes telemetry cable, micro USB cable, DF13 6 position connector for the power module and GPS connector cable.

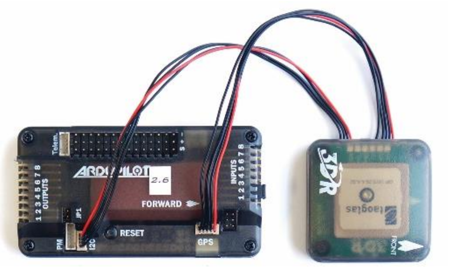


Figure 5- ArduPilot Mega, 3DR GPS

We will attach the feedback package to the UAV, so the data from the device will be the exact data of the camera and the UAV, and will help in processing the feedback to the flight controller. The device sends interrupts with different message ID: HW status, system time, attitude, range-finder info, barometer info, etc. The messages are sent periodically via MAVLINK protocol. [5][6]

3.1. Euler angles and speed

At first we burned the FW code, then configured the device to send messages of the information. We created thread that reads the message data from the device, and when identifying the right message ID, it parses that data into Euler angles and distance. The information is fusion of both sensors type, reducing the noise and deviation in the measurement of Euler angles (and its derivatives). We are using this information both to change the perspective of the frame and act as feedback to the flight control- input of the real angles and angular velocity of the UAV.

3.2. Active Sonar Sensor

The internal barometer (height sensor) is not accurate enough for our purposes, hence we connected active sonar range finder to the APM- MaxBotix LV-MaxSonar- EZ0 [7]. This sensor has 1-inch resolution and a maximum range of 254 inches (about 6.45 meters). When configuring the APM, we added request for reading from the analog input where the sensor is connected (A0). The device extends the list of messages its send, including the information from the sensor. This information helps us in two issues: transferring the distance a pixel has traveled in the frame to the distance this point traveled in reality (this trigonometry calculation depends on the distance from the surface we are capturing). The second issue is feedback to the Z axe in the flight controller. Further work will be to transfer the measurements threw a Kalman filter that we will implement in order to minimize the noise and get more accurate height distance.

4. System Configuration

Figure 2 shows an illustration of our system configuration, all the processing described above will run on an Odroid U3. It is a mini PC with 1.7GHz Quad-Core processor and 2GByte RAM that runs Lubuntu OS (light weight Ubuntu). [8]

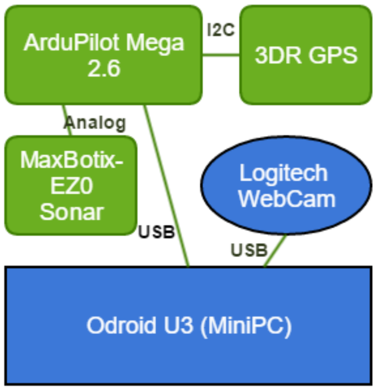


Figure 6 – The System Configuration

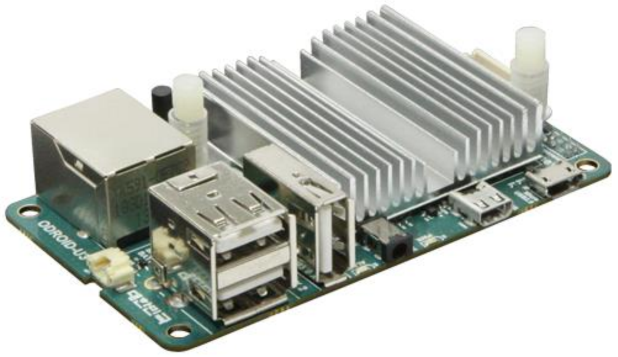


Figure 7 – Odroid U3

5. Feedback Code

Our code is written in C++, running on Linux OS (Using OpenCV and Mavlink libraries). The project refers to many other auxiliary libraries and ports. We needed to link the OpenCV files with our project, as well as to the Mavlink protocol files. So we invested several hours in order to create a suitable Makefile that will compile our project properly.

In Addition, in order to be able to work simultaneously and to maintain version control we added our project to github: https://github.com/asafsarid/Feedback .

The code of the program is distributed into few files/sections:

- Main (feedback.cpp) – Initializing and starting communication with all sensors, starts a new thread for updating global variable for Euler angles and Sonar distance. Starts the optical flow function and plots the information at the end of the execution.
- Sensors update (sensors.cpp) – Using MAVLINK protocol for reading messages from the APM. Every time we get a relevant message ID (Attitude / Range Finder) [3], the

relevant global variable is update with the current measurements. This part is been running in a separate thread.

- Optical Flow (opticalflow.cpp) – Reading frames from camera, applying projection of the frame with the current Euler angles, calculating optical flow using OpenCV Farneback algorithm [9], updating location (x,y) and drawing location plot in real-time.
- Plots (locationPlot.cpp) – In order to have save our experiments results and show them on some graphs we've made few functions that can draw and save relevant plots. We currently draw the plots using OpenCV but we are working on changing it to Qt plots (which should give better preview and performance).

6. Conclusions

We have made significant progress in the last months. Currently, we are very close to be able to test our project virtually. The code we wrote navigates the UAV to desired point, while the location and angles of the UAV are passed from processing data from various sources (sensors and camera), and process it.

There are some issues that we haven't solved or handled yet, and are now with the highest priority (like optimizing our code, calibrate again the camera, etc.). But many other important and heavy issues were solved: We finish reading the sensors and embedded the data from them into our code of optical flow.

One of the toughest parts were to transfer the flight controller we created in Matlab and Simulink into environment that can run on a micro-controller. There was also option of transferring all files into Matlab, but we had troubles with manipulation OpenCV functions and classes in Matlab, hence we decided to transfer our code into runnable configuration.

Another issue was to use the angles of the capturing to change perspective and processing of the optical flow algorithm, it took us a lot of time because many functions didn't fit our requirements or were deprecated, so we wrote our own functions or mixed few together.

Another issue we must take care of is the special condition we have in our project. For example, we may have GPS communication once in a while- this can reset our error. But at the same time, we may suffer from strong wind because the UAV flying outdoor- and this can increase the drift error over time. We need to adjust the final controller to these conditions.

7. References

[1] G. Farneback, “Two-Frame Motion Estimation Based on Polynomial Expansion”, 2004.

[2] J. de Boer, M. Kalksma, “Choosing between optical flow algorithms for UAV position change measurement”, 2015.

[3] OpenCV Warpperspective Function. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#warpperspective. Accessed Mar. 31, 2016.

[4] ArduPilot Mega. Available: http://www.ardupilot.co.uk/. Accessed Mar. 31, 2016.

[5] MAVLINK Protocol – Common Message Set. Available: https://pixhawk.ethz.ch/mavlink/ . Accessed Apr. 1, 2016.

[6] MAVLINK Tutorial. Available: http://dev.ardupilot.com/wp-content/uploads/sites/6/2015/05/MAVLINK_FOR_DUMMIESPart1_v.1.1.pdf . Accessed Apr. 1, 2016.

[7] Maxbotix Analog Sonar. Available: http://ardupilot.org/copter/docs/common-rangefinder-maxbotix-analog.html . Accessed Apr. 1, 2016.

[8] Odroid U3. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=g138745696275 . Accessed Mar. 31, 2016.

[9] OpenCV Farneback Function. Available: http://docs.opencv.org/3.0-beta/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowfarneback Accessed Mar. 31, 2016.

8. Appendix

8.1. Experiments

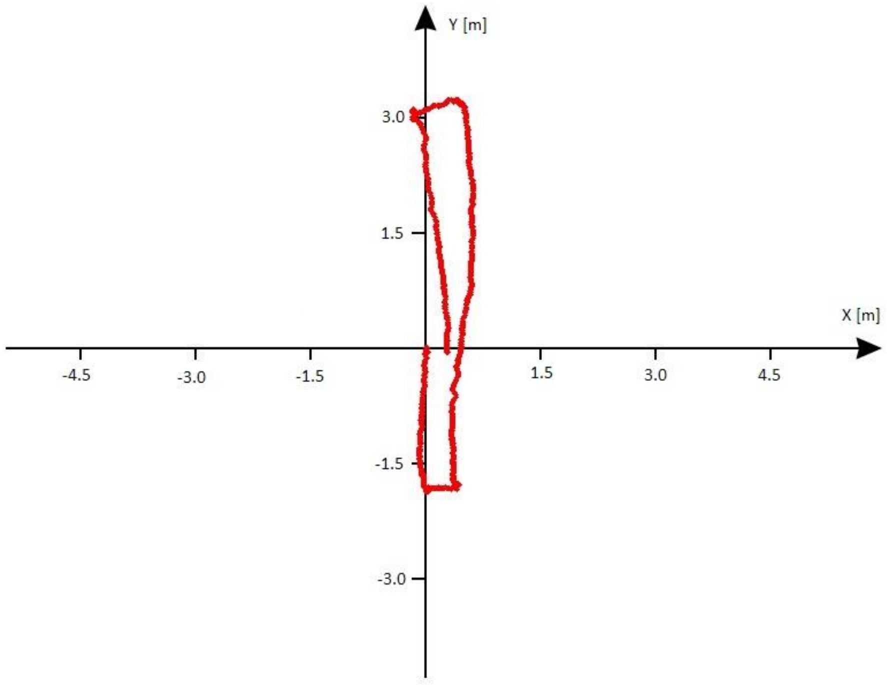
The project in currently in stage that we can move the feedback platform (sensors and camera) by hand, and then receive the location of the platform (UAV) at any given time. In order to check if the algorithm and code works we did some experiments in the hallway of building 33.

The first experiment was to take the platform in the hallway in a rectangle. We tried to avoid changes in roll angle, but we changed the pitch. When arriving to the end of the corridor we turned so there were also changes in the yaw.
We can see that the location was pretty accurate, and except small error in the yaw of the lest turn, we moved in straight lines and 90 degrees turns. We almost returned to the same place where we started.

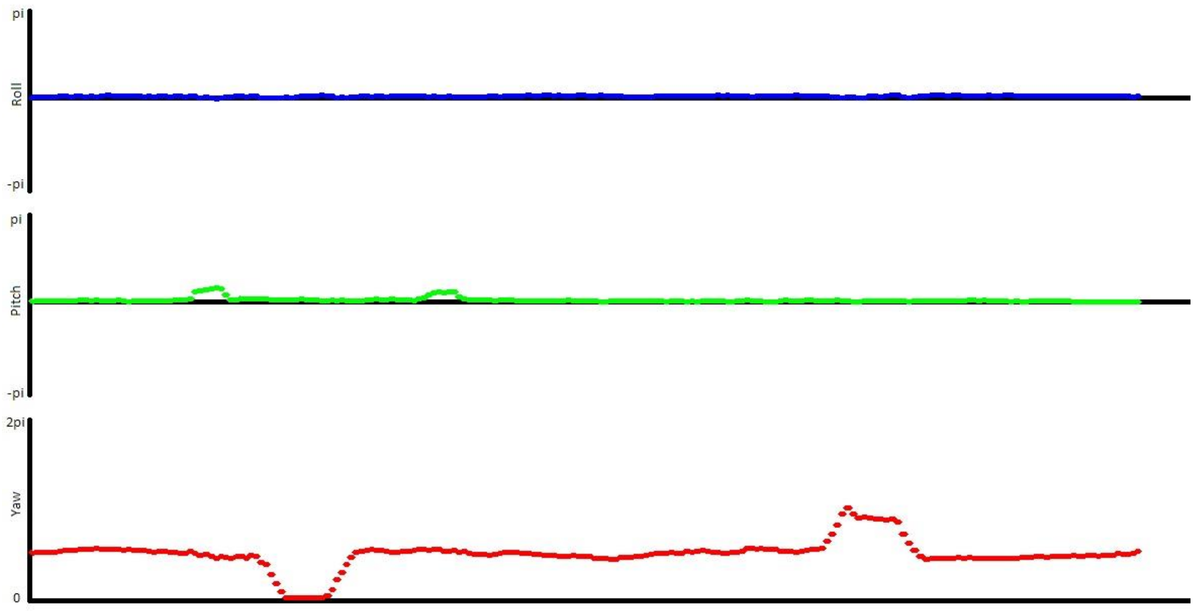
In the second experiment we put more effort on the changes in the angles. We rotate all angles. Started with the pitch, then the roll and finally the yaw. During this we keep tracking the location, and the location was as we expected.

Results:
X axis is north-south
Y axis is east-west

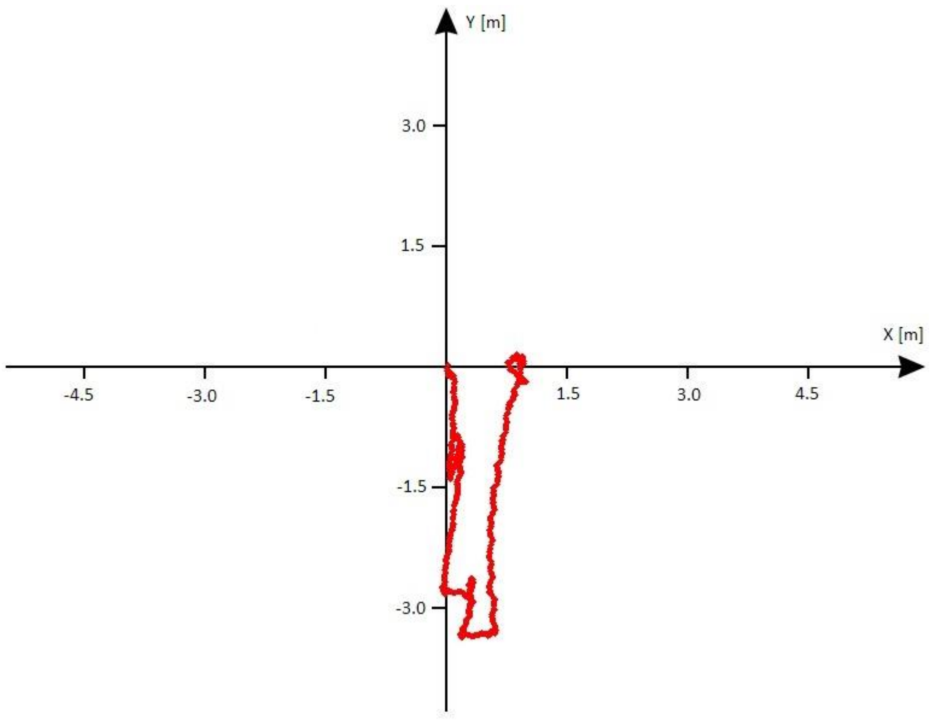
Experiment 1:
Location-



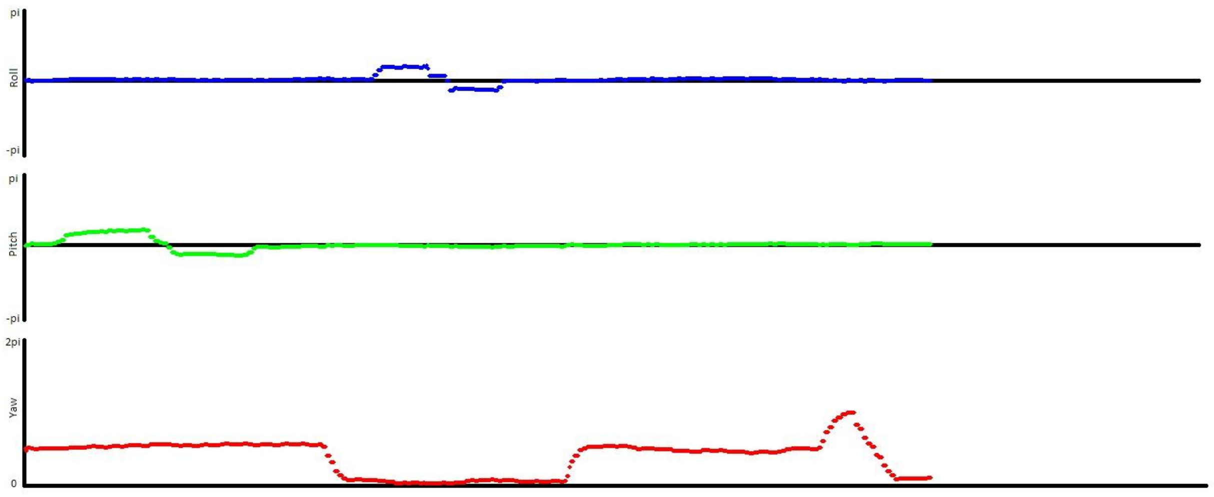
Angles-



Experiment 2:
Location-



Angles-



8.2. Gantt Table

Start Date	**End Date**	**Cohen Ohad**	**Sarid Asaf**	**Task**	**Status**	**Adviser Approved**	**Adviser Notes**
25/10/2015	20/11/2015	yes	yes	Basics: Rigid Body, QuadCopter, etc.	Done	no	
26/10/2015	20/11/2015	yes	yes	Literature Survey	Done	no	
26/10/2015	27/11/2015	yes	yes	Simulator: Installation and Activation	Done	no	
30/11/2015	01/01/2016	no	yes	Simulator: Deeper Understanding and Implementation	Done	no	
30/11/2015	28/12/2015	yes	no	Designing and Implementing the controller (Selecting Controller Method)	Done	no	
29/12/2015	15/01/2016	yes	yes	Testing the controller on the UAV	Done	no	
18/01/2016	04/02/2016	yes	yes	Learning Optical Flow navigation technique	Done	no	
5/02/2016	04/04/2016	yes	yes	Designing navigation algorithm	In Progress	no	
5/04/2016	12/05/2016	yes	yes	Integration of the systems	Not Done	no	
13/05/2016	21/06/2016	yes	yes	Tests and bug fixing	Not Done	no	

8.3. Grading

המלצת ציון לדו"ח מכין

<u>אם יש צורך, לכל סטודנט/ית בנפרד</u>

מספר הפרויקט: P-20____-_____

שם הפרויקט:

שם המנחה החיצוני:

שם המנחה מהמחלקה:

שם הסטודנט/ית: ת.ז.:

%		חלש 64-55	בינוני 65-74	טוב 84-75	ט"מ 94-85	מצוין 100-95	
25	הערכת ההתקדמות בעבודה בפועל בהשוואה לתכנית העבודה						
25	מציאת פתרונות לבעיות שהתגלו ויישומם						
20	גילוי יוזמה וחריצות						
30	מקוריות ותרומה אישית						

הערות: