Automated Landing, Take-off and Surveillance by UAV using Computer Vision

Dhruv Sharma (B14CS015) Mohit Mehta (B14CS043)
Mentor: Dr. Chiranjoy Chattopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur

April 24, 2018

1 Acknowledgement

We would like to express our special thanks of gratitude to Dr. Suril V. Shah, Department of Mechanical Engineering, IIT Jodhpur for providing us the opportunity to use the facilities of the Robotics Lab. We would also like to thank all the Lab Assistants, Kartik Ramachandruni (4th Year, ME), Dhagash Desai (3rd Year, ME) and Sudhir Pratap Yadav (4th Year, EE) for helping us in the entire process of ROS simulations.

2 Motivation

Autonomous surveillance finds its application in many fields. It is a critical requirement for any search and rescue scenario under an emergency or radiation leak. The authorities may not want to send a human in such scenarios as it can be catastrophic. An Unmanned aerial vehicle can detect radiations and send proper feed to the base station so that appropriate steps can be taken. In search and rescue operation, an aerial vehicle can send live video feed and its current location to the base station so as to obtain the exact location and situation of the affected region. At such sites also, human intervention is not always possible. So, we need an autonomously operated drone, which can place itself safely at the operation site and report the conditions in the neighbouring areas.

3 Objective

The objective of our project is to provide a Proof of Concept for simulating an Unmanned Autonomous Vehicle (UAV) to achieve the 3 major tasks - take-off (Sec. 3.1), surveillance (Sec. 3.2) and the landing (Sec. 3.3), using computer vision. The objective is described in a pictorial format in the Fig. 1 We are using a P3-DX (Sec. 4) for the simulation purpose, with a camera mounted over it.

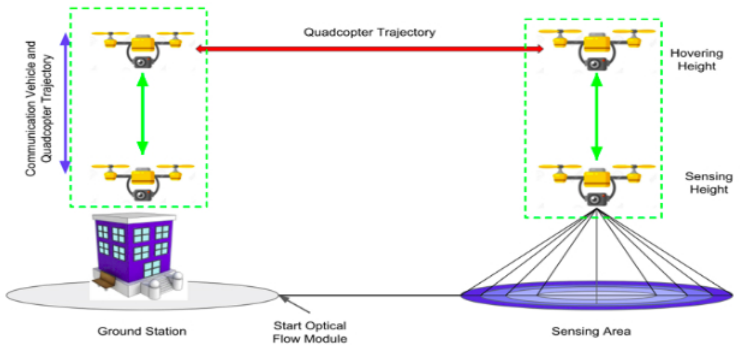


Figure 1: Problem Statement

3.1 Taking off

The UAV will be sent to the destination region from a base station which will be communicating with it continuously. For this, the UAV has to take-off from its base location. When any vehicle, either manned or unmanned, the main task is to stabilize its velocity in all the 3 coordinate directions so as to control the abrupt movements in the beginning. In order to stabilize the UAV, the angle which the UAV axis is making with the horizon should converges to zero.

3.2 Surveillance

The next task is to send the drone from the base station to the destination region. Multiple UAVs have to be sent to analyze the situation. In order to avoid the continuous message exchange over the network for path following, first some service drones have to be sent. The service drones follow different paths to the destination location and continuously send the video feed, of whatever the camera mounted on them sees in the downward direction, to the base station at a particular frequency. The path traced by the drones has to be stitched at the base station and the center of each frame input are to be mapped onto the world map coordinate and stored for the use of the sensing drones. The best and the safest path coordinates are to be chosen and appropriate signals are to be given to the sensing drones so that they follow the chosen path. For situations where continuous surveillance is to be done over a region, these stitched images provide an excellent idea of what all has changed in that region over the interval between which the drone passed over that it.

3.3 Landing

When the UAV reaches the target region, it has to come down from the surveillance height in order to better sense the situation and send the feed. For that, downward velocity has to be sent to the UAV from the base station. But the main difficulty here is to prevent the vehicle from crashing into the ground. In order to do so, optimum downward velocity has to be calculated for the height at which the vehicle is hovering. Likewise, when the UAV comes back to the base station, it has to autonomously land at a particular position. So, using vision, the height and the optimum velocity has to be calculated and appropriate control signals are to be sent the UAV so that when it is very close to the ground, the velocity becomes so low that it does not crash into the ground.

4 Simulation Environment

In the actual scenario, all the modules of our project are to be run on an aerial vehicle. But in this project, in order to provide the Proof of Concept, we have used Pioneer P3-DX which benefits from a complete Robot Operating System (ROS) for the purposes of programming. The detailed descriptions about ROS and P3-DX are given in the subsequent sections.

4.1 ROS

Robot Operating System (ROS) is a collection of software frameworks for robot software development. The single executable in our normal programming language is called a node here which uses ROS to communicate with other nodes via some messages. A node can publish a message to a topic which can be subscribed by other nodes using the topic name or can subscribe to the topics published by some other nodes.

4.2 Simulation using P3-DX

Pioneer P3-DX is a research mobile robot which can be communicated with using ROS. For the purpose of this project, one of the P3-DX available in the Robotics Lab of our Institute was used. A USB camera was mounted on the P3-DX which published ROS images on the topic usb_cam. The P3-DX had an in-built ROS installed. The P3-DX and our machine were connected with the same hotspot. By exporting the ROS_IP of the P3-DX, the topics published by the nodes connected over that hotspot were visible to each other. The P3-DX uses a package called RosAria to subscribe to velocity over the topic cmd_vel. The message has a data type Twist which is a pair of velocities (v, w) representing linear and angular velocity respectively. Both v and w are set of three values (v_x, v_y, v_z) and (w_x, w_y, w_z) respectively representing velocities in the three coordinate axis. The architecture of the complete system is depicted in the Fig. 2

4.2.1 Landing

For the landing module, we simulated it by moving the P3-DX in the horizontal direction towards a board. The camera was facing the board which published the frames at a frequency of 30 Hz on the topic usb_cam. The processing of these frames is explained in the Sec. 5.3 which published the optimum velocity for the P3-DX at a particular distance from the board. The P3-DX subscribed to this velocity and moved accordingly.

4.2.2 Surveillance

For the Surveillance module, we simulated it by moving the P3-DX in a defined path on top of a poster. The camera was facing the poster which published the

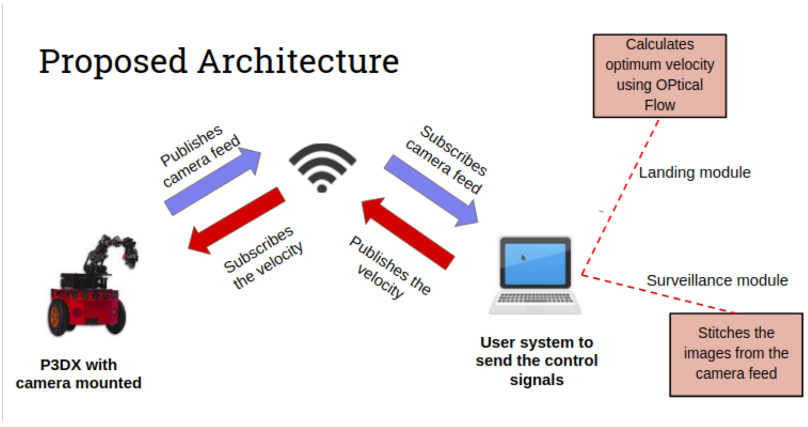


Figure 2: Architecture of the entire system

frames at a frequency of 30 Hz on the topic usb_cam. The processing of these frames is explained in the Sec. 5.2. The node on the user side continuously published velocity so that the P3-DX moved on the path decided.

5 Methodology/ Algorithm

In all the different steps described below, images were subscribed from the Camera mounted on P3DX in the form of ROS Image and were converted into OpenCV image using the function imgmsg_to_cv2() of CVBridge class.

5.1 Taking off

The main aim of this step was to stabilize the camera angle with the horizon so that it converges to zero. The detailed steps of this module are described below:

- Edges were obtained from the image obtained in the previous step using the function cv2.Canny(). The threshold1 and threshold2 were selected to be 10 and 30. The function returns a binary image for the input image.
- The binary image was sent as an input to the function cv2.HoughLines() with threshold value as 200. Threshold here is the minimum vote it should get for it to be considered as a line. The output of this step is the tuple (ρ, θ) which represent the distance of the line from the image origin and angle which the line makes with the X-Axis.
- Out of the lines obtained in the previous step, the line with the maximum length was selected with the assumption that the horizon is the longest line among all in the frame. This assumption holds good because this module is to be used only at the base station and the environment can be controlled accordingly.
- The angle corresponding to the line selected in the previous step was displayed. The negative of the angle should be sent to the UAV controller so as to make the angle with the horizon become zero.

5.2 Surveillance

A major task of the project is to recreate the scene that the camera captures below, while hovering above the terrain on its way to the target site. This recreated scene can be helpful for various use cases as described in Sec. 3.2. To carry out the surveillance module, a USB camera was mounted over the P3-DX, which continuously provided the camera feed. The assumption here made was that the aerial vehicle shall fly over a plainer terrain at a constant height. Moreover, the scene creation was done only after the camera feed was stopped. This entire task of scene stitching was carried out using the following steps:

- The surveillance algorithm can be broken down in basic steps- retrieving the image feed, pre-processing it, finding features for matching between two consecutive frames, orienting with respect to the matches, and finally stitching the two images. This process is carried out with the new generated stitched image and the incoming frame until the last frame is captured.
- The stitched result till the current step were stored in a global variable. The key features were located for the result till now and the incoming frame using SIFT algorithm.
- The key points detected from the previous step were then matched using the brute-force algorithm. The best matches of the calculated ones were then picked out using the KNN-match algorithm. This list of matches was then used to computer the homography matrix(H). H is used to capture the orientation of the new frame with respect to the current result.
- The H, thus produced, is then altered to realign the centre of the new frame from origin to the centre of the canvas. This is done by pre-multiplying H with translation matrix. This step provided the flexibility of stitching the new frame in any direction with respect to the result.
- After updating the H, the orientation of the new frame is calculated and pasted on the blank canvas. The previous result is stitched over then to get the desired stitched image. This obtained result is then defined as the result and the above set of steps is repeated with the updated results and the new frame.

The above method generated a good approximation of the scene below. But it generated a very blurry image of the initial frames as shown in the Fig. 9. This can be explained due to continuous Homography matrix multiplication and interpolation of the image points at each stitching steps. This led to a lot of distortion in the final results.

To overcome this issue, the above stated algorithm was changed a bit, as shown in Fig. 3. The fact that the stitching is done after all the frames are captured, was used to get better results. The improvised algorithm worked as follows:

- Let there be N frames in total. Then the frames were divided into buckets of size 25 each. This generated N/25 buckets in all. The main idea behind this was to reduce the excessive Homography matrix multiplication to the frames.

- For each bucket, the above stated algorithm was applied separately. This generated stitched patches of the 25 frames in there corresponding bucket. The patches, thus generated, were of much better quality as compared to the previous results.
- After this, each patch generated was treated as a single image. This meant that there were N/25 images to stitch now. These new images were then stitched as per the initial algorithm.

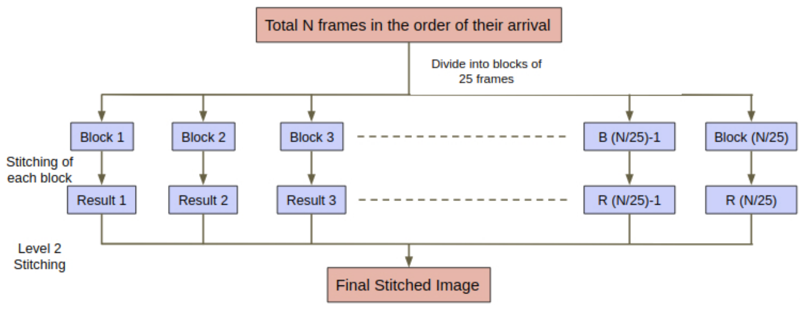


Figure 3: Improvised algorithm for scene stitching

5.3 Landing

In this module, we used a board which had a marker drawn on it as shown in Fig. 4 The main aim of this step was to calculate the optimum velocity for the P3DX so that it does not crash into the board while moving towards it. The detailed steps of this module are described below:

- In this module, we had the assumption that the initial distance of the P3-DX from the board was known and the initial velocity was zero. In the real world scenario ,i.e, with the UAV, the former will be valid as long as the environmental factors of wind pressure, terrain etc. are relaxed. The latter can be relaxed without the loss of generality but the still the initial velocity cannot be too high.
- If the subscribed frame was the first one, Region of Interest (ROI) was selected which was basically the region around the marker on the ground.
- From the ROI, points were extracted using the function cv2.goodFeaturesToTrack(). These points were tracked in the subsequent frame until the P3-DX had reached very close to the board.
- For subsequent frames, the selected points were tracked in the adjacent frames using the function cv2.calcOpticalFlowPyrLK(). If the points could not be located in either of the two frames, such points were discarded.
- Axis was selected as the vertical line, V passing through the point $x = \frac{x_{min}+x_{max}}{2}$ where x_{min} and x_{max} were the x-coordinates of the leftmost and rightmost points that were being tracked.

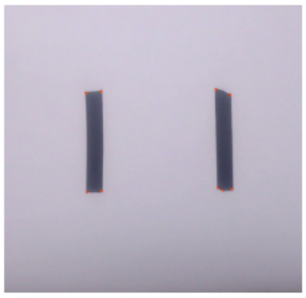


Figure 4: The marker used for the Landing Module

- Two vector sums, S_{left} and S_r of the vectors, S_{Left} and S_{Left} to the left and right of the line V.
- 5 cases arose for the length of the vectors, L_{Left} and L_{Right} as described below -
  - The P3-DX was considered to be moving towards the board if the vectors V_{Left} and V_{Right} were in the opposite direction and pointing towards the line V.
  - The P3-DX was considered to be moving away from the board if the vectors V_{Left} and V_{Right} were in the opposite direction and pointing away from the line V.
  - The P3-DX was considered to be moving towards the board if the vectors V_{Left} and V_{Right} were pointing towards the left with respect to the frame coordinate and the value of $abs(l_{Left} - l_{Right})$ was more than some threshold T and if the length l_{Left} was more than the lengthl_{Right}.
  - The P3-DX was considered to be moving away from the board if the vectors V_{Left} and V_{Right} were pointing towards the right with respect to the frame coordinate and the value of $abs(l_{Left} - l_{Right})$ was more than T and if l_{Left} was less than l_{Right}.
  - The P3-DX was considered to be static if the value of $abs(l_{Left} - l_{Right})$ was less than T.
- The divergence value was calculated for the tracked points in the adjacent frames using the formula given in [3]

  $$D = \frac{1}{n\Delta t} \star \sum_{i=1}^{n}[\frac{d_{(t-\Delta t,i)} - d_{t,i}}{d_{(t-\Delta t,i)}}],$$

  where $d_{(t-\Delta t,i)}$ and $d_{(t,i)}$ are the image distance between every two tracked points in the first and the second frame respectively, Δt is the time interval between the two frames and n is the total number of tracked points.
- The P3-DX was assumed to be moving towards the board if the divergence value was negative. It was assumed to be moving away if divergence value

was positive and static if the abs(value) was less than 0.1. In ideal case this threshold should be zero.

- The distance (Z) at i^{th} iteration was calculated using the formula as given in [3]:

$$Z_i = \frac{V_{Z,i-1}}{D_i}$$

- The velocity ($V_{Z,i}$) at the i^{th} iteration calculated using the formula as given in [3]:

$$V_{Z,i} = D_i \times Z_i$$

- The calculated velocity was published to be subscribed by the P3-DX.
- The process was repeated until the distance of the P3-DX from the board was less than a threshold height $Z_{threshold}$
- If the distance was less than $Z_{threshold}$, zero velocity was published.

6 Results

This section covers the results generated as per the above discussed methods.

6.1 Stitching Results

Figure 5 refers to the uni-directional image stitching, in which the camera is moving downwards. Since the total frames in this case were very less, we did not go for two level stitching in this case which was described in Sec. 3.2.
Figure 6 captures the image stitching results of moving the camera in a V-shape over a wall. In this case, two level stitching was used with a block size of 15.

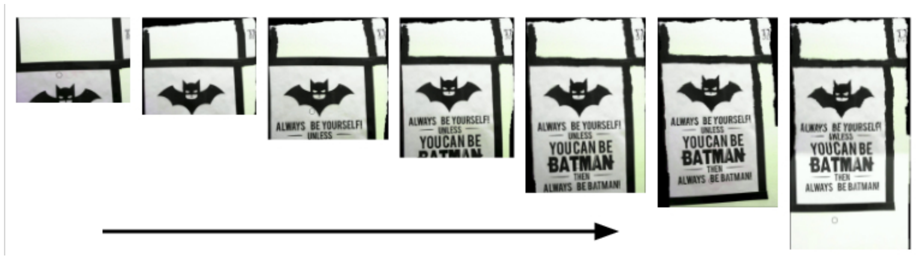


Figure 5: Stitching result with unidirectional movement of the camera

6.2 Landing Results

As discussed in Sec. 5.3, the P3-DX was placed at an initial distance of 1.9 m from the marker. The initial velocity was taken to be 0 m/s. The estimated optimal velocity first increased and then started to decrease as can be observed from the Fig. 7. The threshold distance was set to be 0.3 m. So, the P3-DX stopped when the distance was reached because zero velocity was published.

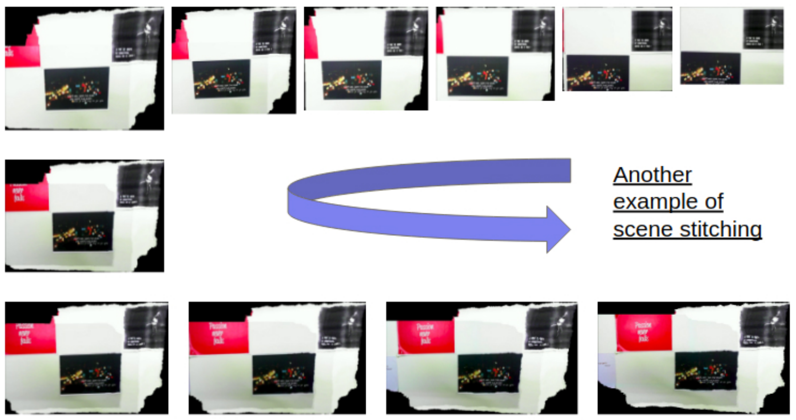
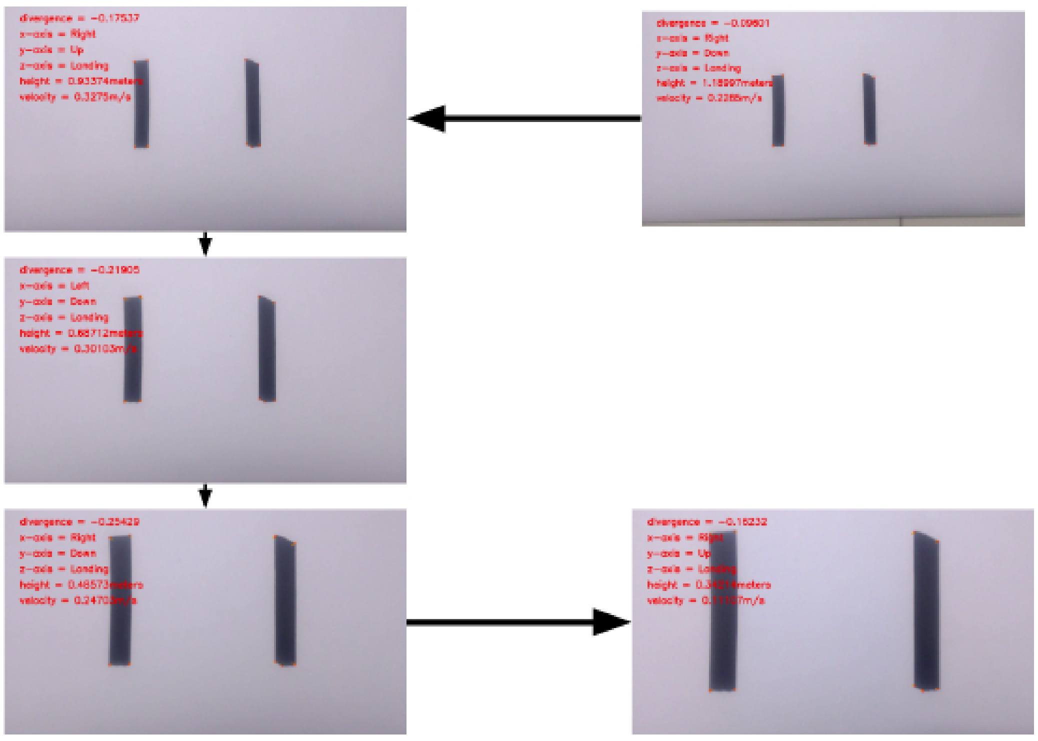


Figure 6: Stitching result with a V-shaped movement of the camera



```
divergence = −0.09601
x−axis = Right
y−axis = Down
z−axis = Landing
height = 1.18897meters
velocity = 0.2288m/s
```

```
divergence = −0.17537
x−axis = Right
y−axis = Up
z−axis = Landing
height = 0.93374meters
velocity = 0.3275m/s
```

```
divergence = −0.21905
x−axis = Left
y−axis = Down
z−axis = Landing
height = 0.68712meters
velocity = 0.30103m/s
```

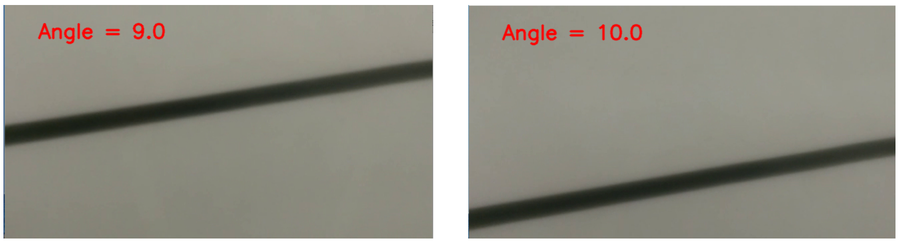
```
divergence = −0.25429
x−axis = Right
y−axis = Down
z−axis = Landing
height = 0.48573meters
velocity = 0.24703m/s
```

```
divergence = −0.18232
x−axis = Right
y−axis = Up
z−axis = Landing
height = 0.34214meters
velocity = 0.11107m/s
```

Figure 7: Simulation of Landing, display of Height and Velocity

6.3 Take-off Results

As discussed in the section 3.1, the angle of the drone with the horizon was calculated. The figure 8 displays the angle of the drone, considering the black line as the horizon. This angle calculated can be used to send appropriate control signals to the drone to align itself with the horizon until the angle is zero.



(a) Frame 1 during take-off (b) Frame 2 during take-off

Figure 8: Take-off results displaying the angle of the UAv with the horizon

7 Future Scope

In future, following work can be done in this project:

- The Stitching module as explained in the Sec. 5.2 can be made more robust and efficient. As it can be observed in the Fig. 9, due to continuous multiplication with the homography matrix, the initial frames have become blurred. We have proposed a solution for this problem in the Sec. 5.2 but more research can be done in this field to completely remove the blurriness or distortion and retain the original sharpness of the image.

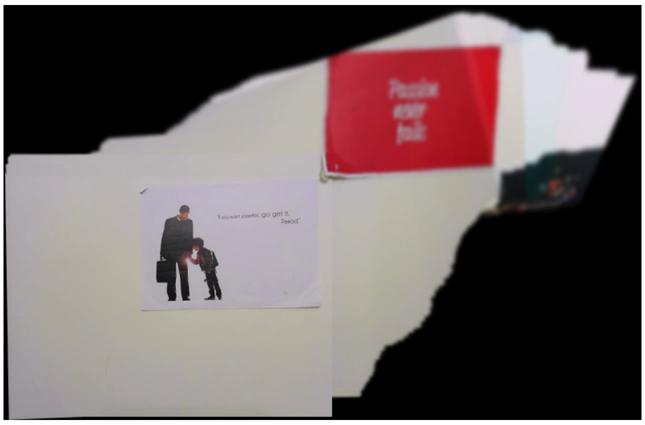


Figure 9: Blurred image generated due to initial algorithm described in Sec. 5.2

- The stitching of the frames is being done after the video feed from the P3-DX has stopped. In future, real time stitching can be implemented in which the frames will be stitched at the base station as and when they arrive.

- The project work is a Proof of Concept and works as a base for the future implementation on an actual UAV. The code can be tested in real scenario with an on-board camera on the aerial vehicle which would send live video feed to the user machine where all the modules would be run.
- In the real world scenario, due to wind pressure, uneven terrains and other environmental factors, the height of the UAV from the ground will change. This factor has to be incorporated while calculating the height of the UAV so that it becomes more robust.

References

[1] OpenCV Optical Flow Documentation, `https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html` 2017

[2] Hough Transform Documentation, `https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html` 2014

[3] Hann Woei Ho and Guido CHE de Croon and Qiping Chu, Distance and velocity estimation using optical flow from a monocular camera, International Journal of Micro Air Vehicles, 2017, 03, 198-208

[4] Cesetti, Andrea and Frontoni, Emanuele and Mancini, Adriano and Zingaretti, Primo and Longhi, S, Vision-based autonomous navigation and landing of an unmanned aerial vehicle using natural landmarks, Mediterranean Conference on Control and Automation, 2009, 06, 910-915

[5] Tiago Gomes Carreira, Quadcopter Automatic Landing on a Docking Station, 2013

[6] Merz, Torsten and Duranti, Simone and Conte, Gianpaolo, Ang, Marcelo H. and Khatib, Oussama, Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing, Experimental Robotics IX, 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 343–352

[7] Basic Image Stitching Code, `https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching` 2017

[8] Scale Invariant Feature Transform (Sift) Documentation, `https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html` 2015

[9] Basic Image Stitching Code, `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html` 2014

[10] Ros Tutorials, `http://wiki.ros.org/ROS/Tutorials`