Automatic Pose Estimation of Micro Unmanned Aerial Vehicle for Autonomous Landing

Manish Shrestha[1], Sanjeeb Prasad Panday[2(✉)], Basanta Joshi[2(✉)], Aman Shakya[2], and Rom Kant Pandey[3]

[1] Nepal College of Information Technology, Pokhara University, Lalitpur, Nepal
[2] Pulchowk Campus, Institute of Engineering, Tribhuvan University, Lalitpur, Nepal
{sanjeeb,basanta,aman.shakya}@ioe.edu.np
[3] Sanothimi Campus, Tribhuvan University, Bhaktapur, Nepal

Abstract. The guided navigation has enabled users with minimal amount of training to navigate and perform flight mission of micro unmanned aerial vehicle (MAV). In non-urban areas, where there are no other aerial traffic and congestion, MAV take-off & travel does not need much Global Positioning System (GPS) accuracy. The critical part seems to be during the landing of the MAV, where slight GPS inaccuracy can lead to landing of the vehicle in the dangerous spot, causing damage to the MAV. This paper aims to propose a low cost portable solution for the Autonomous landing of the MAV, using object detection and machine learning techniques. In this work, You Only Look Once (YOLO) has been used for object detection and corner detection algorithm along with projective transformation equation has been used for getting the position of MAV with respect to the landing spot has been devised. The experiments were carried with Raspberry Pi and the estimation shows up to 4% of error in height and 12.5% error in X, Y position.

Keywords: Micro unmanned aerial vehicle · UAV · GPS · Autonomous landing · Object detection · CNN · YOLO

1 Introduction

Micro Unmanned Aerial Vehicles (MAV) or drones has been using Global Positioning System (GPS) to execute flight missions easily. Even though there are some fluctuations in GPS readings from time to time even for the same spot, they are commonly being used in such missions. Instead of GPS, the landing at specified spot can also be done with the help of other sensors, like camera. Takeoff, hovering, moving forward and landing are some of the basic phases for autonomous flight of MAV. Among them, landing visually on a specified target is especially complex because it requires robust recognition of the landing pad and precise position control; and a slight offset of few meters can also cause crash landing of the vehicle.

Vision based approach was also used by Yang et al. [1] presented an on- board vision system that can detect a landing pad consisting of the letter "H" surrounded by a circle, from images captured by a monocular camera on a MAV and determine the 6

DOF pose of the MAV relative to the landing pad using projective geometry. The 5 DOF pose is estimated from the elliptic projection of the circle. The remaining geometric ambiguity is resolved by incorporating the gravity vector estimated by the inertial measurement unit (IMU). The last degree of freedom pose, yaw angle of the MAV, is estimated from the ellipse fitted from the letter "H". A neural network was used to initially detect the landing pad. The structure of the neural network is a multilayer perceptrons with 196 input units (one per pixel of patterns resized to 14×14), only one hidden layer consisting of 20 hidden units and three output units.

In another paper, Yang et al. [2] presented a solution for micro aerial vehicles (MAVs) to autonomously search for and land on an arbitrary landing site using real-time monocular vision. The autonomous MAV is provided with only one single reference image of the landing site with an unknown size before initiating this task. The autonomous navigation of MAV was achieved by implementing a constant-time monocular visual SLAM framework, while simultaneously detecting an arbitrarily textured landing site using ORB features, and estimating its global pose.

Daniel et al. [3] employed visual odometry techniques with feature-based methods to compute the aircraft motion and thereby allowing the position estimation in GPS denied environments. With regards to GPS inaccuracy, Stark et al. [4] showed that almost half (49.6%) of all ≈68,000 GPS points recorded with the Qstarz Q1000XT GPS units fell within 2.5 m of the expected location, 78.7% fell within 10 m and the median error was 2.9 m.

Traditional object detection systems are variants of the following pipeline: Firstly, find potential objects and their bounding boxes, then do feature ex- traction, and finally classify using a good classifier. Selective Search (SS) [5] enjoyed being the state-of-the-art for detection on PASCAL VOC etc. competitions. HOG [6] and SIFT [7] are the popular choices for feature extractions. A classifier is applied on image pyramid to overcome problems with scale.

The current state-of-the-art object detectors such as Fast R-CNN [8], YOLO [9], SSD [10] etc. are based on convolutional neural networks (CNN) and have outperformed the traditional techniques. The key to the success of CNNs is their ability to extract/learn generic features. Furthermore, the advancement in computational resources such as high-performance GPUs and its easy availability through the use of high-performance cloud computing platforms played an important role in the recent success of neural networks.

In this work, monocular vision based system has been proposed to localize the MAV position with respect to the landing spot. Detection of the landing spot has been carried out with more advanced and recent classifiers known as You Only Look Once (YOLO) [9]. A simpler projective transform with 3-DOF variables based on rectangular feature points of a simple landing spot has been used. The proposed system also aims to develop effective system using a simple camera (Raspberry Pi camera) instead of advanced camera (with global shutter).

2 Methodology

This work is divided into two phases: Learning Phase and Implementation phase.

2.1 Learning Phase

As shown in Fig. 1, the learning phase involves data collection, pre-processing and training and evaluating two classifiers, namely YOLO v3 and YOLO Tiny v3.

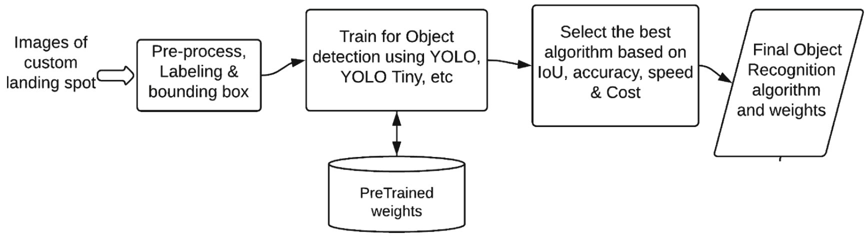

Images of custom landing spot
Pre-process, Labeling & bounding box
Train for Object detection using YOLO, YOLO Tiny, etc
Select the best algorithm based on IoU, accuracy, speed & Cost
Final Object Recognition algorithm and weights
PreTrained weights

Fig. 1. Learning phase diagram.

A custom landing pad of dimension 142 cm × 112 cm with 4 rectangular regions of color red, blue, white and black of equal size was designed. Images of the landing spot in various background and orientation is captured from different height, using simple web camera in a flying MAV, for object detection training. For pose estimation, images will be captured along with roll, pitch and yaw angle using a handheld MAV. Data augmentation technique like rotation, etc. is done to increase the samples of our captured data set for verifying the corner detection phase. For training the object detection classifier, the images has been classified to contain the landing spot and those images are also tagged with the bounding box that indicates the location of the landing spot within the image.

Using the pre-trained available weights and collected datasets, the neural networks YOLO v3 and YOLO tiny v3 have been trained to detect our custom landing spot. The YOLO Tiny v3 has been made to detect the landing spot in Raspberry Pi 3 hardware too.

2.2 Implementation Phase

During the real time application phase, the object detection of the landing spots will be followed by corner detection phase and then the pose estimation phase as shown in Fig. 2.

2.3 Object Detection

Whenever the MAV arrives near the final landing spot as reported by GPS, the task for object detection comes into action. The live images from the camera installed in the MAV are feed into the object detection classifier (YOLO). The classifier, using the

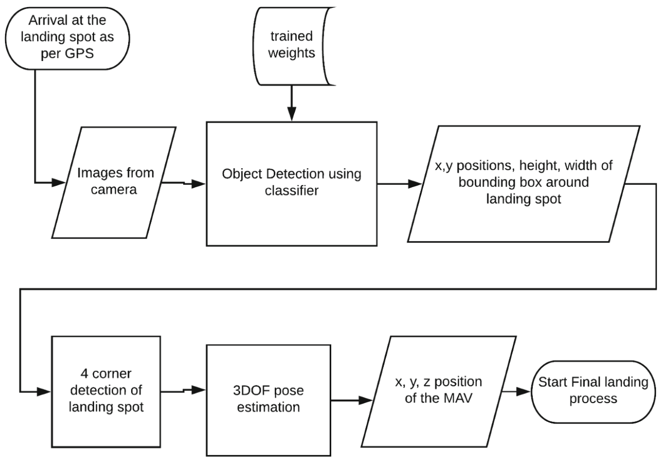

Arrival at the landing spot as per GPS
trained weights
Images from camera
Object Detection using classifier
x,y positions, height, width of bounding box around landing spot
4 corner detection of landing spot
3DOF pose estimation
x, y, z position of the MAV
Start Final landing process

Fig. 2. Implementation phase diagram of the system

already trained weights, calculates the bounding box position of the landing spot from the image. The bounding box position constitutes a rough approximate of x, y position of the landing spot in the image and the height and the width of the bounding box.

2.4 Corner Detection

The information about the bounding box and the image from the first phase would be feed into the Corner estimation algorithm. In this phase, the top left, top right, bottom left and bottom right corners of the landing spot will be identified in the image. The section of the image, which is slightly larger than the detected bounding box area and which encompasses the detected bounding box, should be chosen for the corner detection. Here, the width and height of the selected section of the image can be 1.5 times that of the detected bounding box. The steps for acquiring the corners from the selection section of the image is listed below:

1. Converting to gray scale image.
2. Application of Canny edge detection to get the edges of the landing spot.
3. Perform Hough lines detection to find the lines of the landing spot.
4. Augment the image with the detected lines. The lines to be augmented are chosen such that they are among the top 14 lengthiest lines among the detected lines.
5. Perform Harris corner detection on the augmented image.
6. Take the top leftmost, top rightmost, bottom leftmost, bottom rightmost as the four corners in the images as the corresponding points of the landing spot.

In order to calculate the effectiveness of both methods, the difference between the ground truth value and detected positions of each of the four corners would be calculated. Then mean error distance for each corner detected will be calculated, which is the average of the distance from ground truth calculated for each of the four points.

2.5 Pose Estimation

The position of the four corners detected in the image using the previous phase is used, to get the pose of the camera with respect to the landing spot. The pose information contains the x, y, z position and the rotation around x, y & z axis. Estimating of the pose of a calibrated camera, given a set of n 3D points in the world and their corresponding 2D projections in the image, is a Perspective- n-Point problem. Since the camera position is fixed with respect to the MAV frame, the pose of the camera also gives the pose of MAV with respect to the landing spot.

After solving the Perspective-n-Point problem using the 4 point correspondence between the detected four corners and the actual four corners in the world coordinate, more accurate estimation of the x, y, z positions of the MAV with respect to the landing spot is obtained. This position information would finally be used by the landing mechanism for landing the MAV into the landing spot. The equation governing this projective transformation is shown below:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Here, s is the scaling factor. fx and fy are the x and y focal length in pixels. x, y, z are the world coordinate of a given point and u, v are the x and y locations of the point in an image. The transformation in x, y, z direction is given by t1, t2, and t3.

Since obtaining the ground truth data for the flying MAV is difficult in absence of state-of-the-art object tracking laboratory, the ground truth data of the x, y, z position of the MAV with respect to the landing spot needs to be obtained using hand-held MAV. The difference in the x, y and z position of the calculated observed value and the ground truth value needs to be calculated. The error percentage in x, y and z positions would be given by the formula below:

$$Error\,percent\,in\,X\,position = \frac{(Xg - Xc) * 100\%}{Xg}$$

$$Error\,percent\,in\,Y\,position = \frac{(Yg - Yc) * 100\%}{Yg}$$

$$Error\,percent\,in\,Z\,position = \frac{(Zg - Zc) * 100\%}{Zg}$$

Here, Xg, Yg, Zg denotes the ground truth value of the measured distance between the MAV and landing spot. Xc, Yc, Zc, represents the calculated distance from the pose estimation phase.

3 Experimental Setup

3.1 Development of Experimental MAV for Handheld Experiment

Before making actual outdoor flights, the indoor testing was carried with an experimental setup up consisting of MAV, Raspberry Pi3, Raspberry Pi Camera as shown in the Fig. 3a. An existing MAV was fitted with Raspberry Pi3, and its camera. The existing MAV had 4 motors, 4ESC units, flight controller based on PixHawk, GPS, telemetry unit and LiPo battery. Also a landing pad of 1/4 th of the original size was also made for indoor and handheld experiments as shown in Fig. 3b.

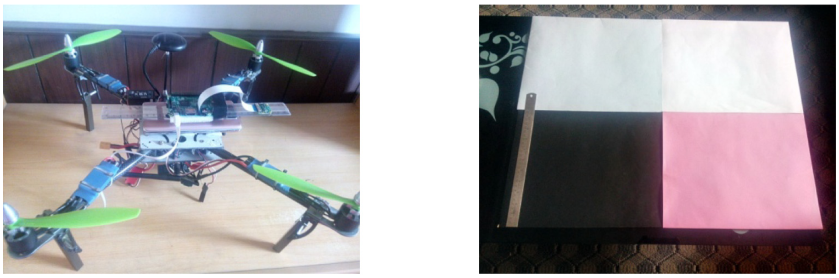

(a) MAV setup for experiments (b) Mini Landing pad

Fig. 3. MAV and landing pad for indoor and handheld experiments.

3.2 Software Setup

The Raspberry Pi3 was installed with the operating system Raspbian Stretch with desktop (release date 20180418). OpenCV 3.3 was compiled and installed into it. Also picamera module was also installed for accessing the Raspberry Pi Camera.

Since the object detection is based on YOLO, darknet system had to be compiled. It was compiled in Linux, Windows and Raspberry Pi. Also in order to take advantage of GPU based calculation, NVIDIA CUDA Deep Neural Network library (CuDNN) had to be installed in both Windows and Linux machine.

3.3 Camera Calibration

Before sending the recorded image through a set of image processing pipeline, a process to correct the image from deformations resulting from camera geometry, image plane placement, etc. needs to be done. For this camera calibration is done to determine extrinsic and intrinsic parameters [11]. The Raspberry Pi Camera was calibrated using a simple checkerboard pattern and the API provided by OpenCV.

3.4 Experiment Parameters

The experiments were performed in three separate environments with the specified parameters as shown in Table 1. The training was done in Environment 1 and

Environment 2. Due to low hardware resources, Raspberry Pi3 has been only used for testing purpose with trained weights from Environment 2. Since YOLO v3 network requires minimum of 4 GB of RAM, it could not be tested in Raspberry Pi3. The YOLO was only tested in Environment 1 with the Windows 10 machine with GTX 1060 Nvidia graphics card.

Table 1. Experiment environment and parameters

Parameters	Environment 1	Environment 2	Environment 3
Hardware	Laptop with 16 GB RAM	Alienware Ddsktop	Raspberry Pi3
Operating System	Windows 10	Ubuntu 16.04	Ubuntu 16.04
Graphics card	Nvidia GTX 1060	Nvidia GTX 1080 Ti	NA
Used for	Training/Testing	Training	Testing
Detection Type	YOLO v3	YOLO v3 Tiny	YOLO v3 Tiny
1. S. Yang	S. A. Scherer and A. Zell	"An onboard monocular vision system for autonomous takeoff	hovering and landing of a aerial vehicle
No of training Image	351	320	N/A
Network input size	416 by 416	448 by 448	448 by 448
Batch Iteration	3100	21000	N/A
Batch size	64	64	N/A
Training Time	9 h	4 h	N/A

4 Dataset Collection and Pre-processing

4.1 Images of Landing Spot for Object Detection

In order to train the neural network for recognizing the landing spot, images of the landing spot from different height was needed. First, video in mp4 format using GoPro Hero 3 camera was taken at the resolution of 1920 * 1080. Then the mp4 format video was converted to still images using YOLO mark tools and using OpenCV API. Then some images in the original size were used, while some of the images were down sized to 448 * 448, using OpenCV API. Then each of those images were labeled with class name and bounding box. Also in order to test the output of our pose estimation algorithm, images were captured from different height using a simple web camera at resolution of 680 $\times$ 480.

Labeling of Landing Spot for Yolo Training: The labeling of the landing spot in the captured images was done using an open source tool called Yolo mark. The YOLO mark was tool was obtained from and was compiled in Windows 10 machine.

4.2 Pre-trained Weights

In order to make the training with few amount of custom images, we use pre- trained weights, that had been created using thousands of images from standard image dataset. The pre-trained weights for YOLO v3 and YOLO tiny v3 were **darknet53.conv.74** and **yolov3-tiny.weights** and were taken from official site of darknet.

4.3 Images of Landing Spot for Pose Estimation

Since it is not possible to obtain the ground truth value of the x, y, z position of a flying MAV in a simple lab setup, the handheld MAV was used to capture the image of the mini landing spot from different height and angle. During the capture, the roll, pitch and yaw angles were also noted. The position, from where the images were taken, was also measured with the help of measuring tape. The images were taken from around 6, 10 and 16 m of height from a building. These images and the measured distances are used to validate the results of the object detection phase.

Marking the Landing Spot Corners in Images: In order to generate the ground truth data for the corner detection phase, all the images used for the verification the corner detection phase, was one by one marked with the x, y position in the image. A Python program was written in order to display the image, on which four corners can be clicked by mouse and then the clicked positions would be recorded. For each image, four points representing the top left, top right, bottom right and bottom left were to be clicked sequentially.

5 Results and Analysis

5.1 Training on Environment 1

YOLO version 3 was trained on Environment 1 with parameters as mentioned in Table 1 with pre-trained weight obtained from official site of YOLO. Images that were directly converted from the video of 1080p, with the resolution of 1920 * 1080 were used. It took around 9 h of training in the Windows machine. The average loss in the network was around 0.08 to 0.07 for about an hour, and hence the training was stopped.

While testing against the test image set, the Intersection over Union (IoU) was calculated. For different IoU detection thresholds, the resulting Average IoU% F1 score are tabulated as shown in Table 2. It can be observed that even for high IoU threshold like 0.95, the results are quite satisfactory with F1-score of 0.99 and False negative of only 1.

Table 2. Comparison of validation results while mapping the YOLO v3 trained network with the test set using different IoU thresholds.

S No	Threshold %	Average IoU %	True positive	False positive	False negative	F1-score
1	0.25	88.85	38	0	0	1
2	0.5	88.85	38	0	0	1
3	0.75	88.85	38	0	0	1
4	0.85	88.85	38	0	0	1
5	0.9	89.14	37	0	1	0.99
6	0.95	89.14	37	0	1	0.99
7	0.99	89.4	33	0	5	0.93

5.2 Training on Environment 2

YOLO Tiny version 3 was trained on Environment 2 with parameters as mentioned in Table 1. The final average loss was also around 0.08. In the training hardware, the trained network of YOLO v3 tiny was able to detect low resolution images (640 pixels * 480 pixels) taken from the web-camera at different height.

5.3 Object Detection

A comparison of the object detection using YOLO version 3 and YOLO Tiny version 3 is tabulated in Table 3.

Table 3. Object Detection Comparison of YOLO v3 versus YOLO Tiny v3.

S No	Detector type	Environment on	Detection time in seconds	Frames per seconds
1	Yolo v3	Environment 1	0.0451	22.172949
2	Yolo v3 tiny	Environment 3	10.901	0.0917347

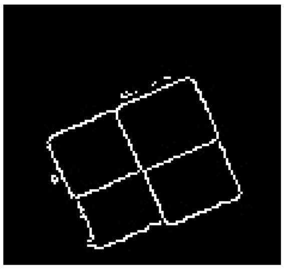
With Environment 1, Yolo V3 was able to detect the landing spot within 0.0451 s, resulting about 22 Frames per Seconds. A slight modification in the original YOLO v3 code was done in order to get the position of the detected bounding box and the size of the bounding box. Here 0.707 and 0.858 are the relative x and y position of the detected bounding box with respect to original image size. And similarly 0.258 and 0.271 are the width and height of the bounding box detected. An image depicting the bounding box is detected by the trained YOLO v3 in Environment 1.

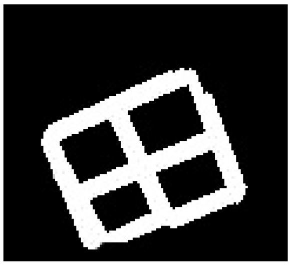
With Environment 3(Raspberry Pi3), Yolo Tiny version took 10.9 s to detect the landing spot resulting in 0.09 frames per seconds of speed. An image depicting the bounding box is detected by the YOLO tiny v3.

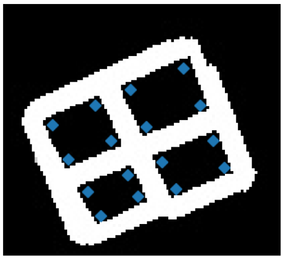
5.4 Corner Detection

After the object detection phase, the x, y position of the landing spot with its height and width is obtained. This gives rough location of the landing spot in the image. Then for the area that is 1.5 times of the indicated dimensions (by the object detection) is considered for the corner detection.

For estimating corners, Canny edge detection, Hough Transform and Harris Corner detection is used. The result of Canny edge detection is shown in Fig. 4a. After the lines are detected using Hough transform, those lines are super imposed on the edge detected image, as shown in Fig. 4b. Then applying Harris corner on this superimposed image, results in the 16 corners as shown in Fig. 4c. Finally, 4 corners obtained after this step is shown in Fig. 4d. The results gave mean error distance of 4 pixels and standard deviation of 2 pixels.


(a) Canny edge detection


(b) Lines detected from Hough transform super imposed on the probable area.


(c) Harris Corner detection in the super imposed image.

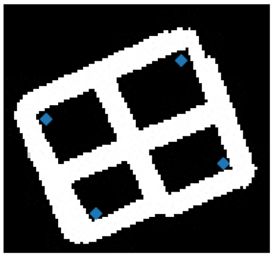

(d) Four Corner detection

Fig. 4. Corner detection for landing spot

5.5 Pose Estimation

Camera Calibration. The calibration of Raspberry Pi camera v2.1 was done by taking various images in 640 x 480 pixels and then detecting the corners in the checker board pattern using OpenCV API. The focal length in x & y direction are at 499 and 501. The optical center position in x & y are at 323 and 234 pixels, which sounds reasonable. The radial distortion parameters k1, k2 and k3 are 0.17, −0.27, −0.20 respectively. And tangential distortion coefficients are −0.0043 and 0.0006 respectively.

Pose Estimation Calculation. After the 4 corners in the 3D world coordinate of the landing spot and corresponding 4 corners in the 2D image has been found, the homogeneous matrix obtained was calculated. The homogenous matrix was decomposed to get the rotation and translation vector between the world coordinate and the camera coordinate. The obtained position of the camera (which also represents the position of the MAV) has been tabulated as shown in Table 4. The ground truth values can also be seen in the table. Here the errors in x, y, z is within reasonable boundary when the yaw angle from which the picture was taken is not much different than in the landing pad. Since a valid constraint, that Y axis MAV should be pointing to Y axis of the landing pad during landing, can be added, this error can be eliminated. From this table it can be concluded that

Table 4. Comparison of final x, y & z positions obtained from the pose estimation with the ground truth.

	Ground truth (in meter)			Calculated result (in meter)			Error percent (Error/GT * 100%)		
S N	Height	X	Y	Height	X	Y	Z	X	Y
1	10.85	3.6	2.3	11.38	1.72	0.78	4.88	52.22	66.09
2	16.1	0.6	0.85	14.05	4.19	4.48	12.73	598.33	427.06
3	6.65	3.65	0.05	6.72	3.78	0.06	1.05	3.56	20.00
4	10.85	3.6	2.3	9.99	4.7	3.26	7.93	30.56	41.74
5	10.85	3.6	2.3	10.28	4.3	2.16	5.25	19.44	6.09

– Rows 1, 2 show high percentage of error due to high Yaw Difference between the landing spot and MAV. This can be eliminated if the landing is done with Y axis of MAV pointing in the Y axis of the world coordinate.
– Up to 8% of error in height estimation, and up to 30% and 41% error in X, Y estimations are obtained in normal conditions without any correction.
– It can be seen that when the image is corrected by pitch angle of the MAV, the error in X, Y and Z position reduces from 30%, 40% and 8% to 19%, 6% and 5% respectively. The row 4 depicts the result after normal calculation and row 5 depicts result of the same calculation after image correction by pitch angle.

After the pitch angle correction is done, it can be seen that the average error across many images in x, y, z position of the MAV is 12%, 13% and 4% respectively, which should be practically acceptable for calculating the position of the MAV using low cost approach described here. Hence, the correction of the captured image, by the pitch angle of the camera or the MAV, is recommended before the pose estimation calculation is done, for better approximation.

6 Conclusion

There are challenges for landing of Micro Unmanned aerial vehicle (MAV) and robust recognition of the landing pad and precise position control is necessary. This work proposes a new visual based approach for MAV for estimating the approximate x, y, z positions of the MAV from the landing spot using recorded camera images, thereby assisting in the landing of MAV. You Only Look Once (YOLO) v3 has been used for object detection of the landing spot in the image, which indicates sub section in the image where the landing spot can be found. Then Harris corner detector has been applied around the subsection, in order to get the four corners of the landing spot in the image. Then after some pre- processing, the pose estimation of the MAV from the planar landing spot has been done by decomposing the homogeneous matrix obtained from 4 points correspondence. The experiments were carried with Raspberry Pi and the estimation shows up to 4% of error in height and 12.5% error in X, Y position. The present work doesn't analyze the performance in adverse lighting condition. The techniques for mitigating the effect of low light and very bright light while taking images from the low-cost camera can be studied in future. Also, the pose estimation can be improved using stereo camera instead of the single camera.

Acknowledgement. This work has been supported by the University Grants Commission, Nepal under a Collaborative Research Grant (UGC Award No. CRG-74/75-Engg-01) for the research project "Establishment of a Disaster Telecommunications Research and Educational Facility Advancing a Scientifically Sound Disaster Telecommunication Infrastructure and Processes in Nepal".

References

1. Yang, S., Scherer, S.A., Zell, A.: An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. J. Intell. Robot. Syst. **69**(1–4), 499–515 (2013)
2. Yang, S., Scherer, S.A., Schauwecker, K., Zell, A.: Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision. J. Intell. Robot. Syst. **74** (12), 27–43 (2014)
3. Villa, D.K., Brandao, A.S., Sarcinelli-Filho, M.: A survey on load transportation using multirotor UAVs. J. Intell. Robot. Sys. **98**, 267–296 (2019)
4. Schipperijn, J., Kerr, J., Duncan, S., Madsen, T., Klinker, C.D., Troelsen, J.: Dynamic accuracy of GPS receivers for use in health research: a novel method to assess GPS accuracy in real-world settings. Front. Pub. Health **2**, 21 (2014)
5. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**(2), 154–171 (2013)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2005)
7. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision (1999)
8. Girshick, R.: Fast R-CNN. In: Proceedings of IEEE International Conference on Computer Vision (2015)

9. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
10. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
11. Joshi, B., Ohmi, K., Nose, K.: Comparative study of camera calibration models for 3D particle tracking velocimetry. Int. J. Innov. Comput. Inf. Control **9**(5), 1971–1986 (2013)