21st International Congress on Modelling and Simulation, Gold Coast, Australia, 29 Nov to 4 Dec 2015
www.mssanz.org.au/modsim2015

A Quadrotor UAV Navigational Command and Control Aid: A Landing Pad Detection and Localisation System

Amir Parsa Anvar[a], Rahul Kalampattel[a], Caterine S. Oliveria[a] and Amir M. Anvar[a]

[a] School of Mechanical Engineering
The University of Adelaide, South Australia

Emails: amir.p.anvar@hotmail.com; rahul.kalampattel@gmail.com; caterine@live.de; amir.m.anvar@gmail.com

Abstract: Research and development of quadrotor UAV robots is a fast-growing area. There is currently a high R&D focus by many companies on UAVs for delivery purposes, swarm applications, farming applications, search and rescue missions, environmental studies and other advanced applications. GPS is a navigational system that can be used to guide a UAV toward a target destination. However, as GPS navigation is prone to local errors, there is a vital need for another sensor to assist with short-range navigation and finding the target landing position. In this case, the quadrotor's on-board camera could act as a useful, optical, sensor to collect real-time data about a destination area and the location of a landing pad. In this research work, successful landing pad detection was accomplished by employing the most ideal of a number of classifiers, which were trained using OpenCV. Two different types of classifier training were performed: these were Haar and LBP (Local binary patterns) training. Theory was also derived to localise the UAV within the earth-fixed coordinate system, based upon its position with respect to the landing pad within the captured image. This article briefly discusses the principles, procedures and results for landing pad detection, as well as image analysis, UAV localisation and command generation.

Keywords: *UAV, localisation, image processing, OpenCV, landing pad detection*

1. INTRODUCTION

Navigation by GPS is prone to local errors, yet over long enough distances such errors tend to cancel out. Long-range navigation is thus often more accurate than short-range navigation. To achieve accurate short-range navigation, digital image processing techniques may be used. This paper describes the basis for establishing a short-range navigation system, based on imagery, for accomplishing precision tasks such as the landing of a UAV.

The scenario of interest in this paper commences with the UAV flying from its home location to a region, where a landing pad is located. As illustrated in Figure 1, once the UAV has reached this region (i.e., it is within the dome of short-range navigation), it must photograph a landing pad, which marks the landing region, localise itself with respect to the landing location and navigate to a point directly above the landing location. It may then perform a gentle descent. Therefore, the major problems that must be addressed are firstly how to detect the landing pad, and secondly how to localise the UAV within the earth-fixed coordinate system. A third problem, navigation of the UAV to the landing pad by command, will only be covered in brief.

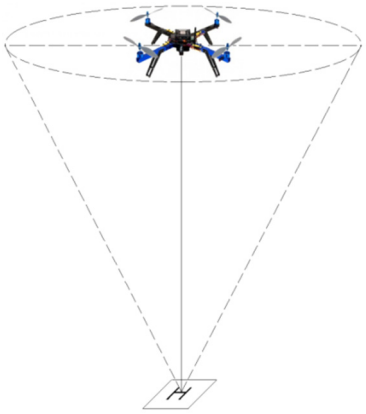

Figure 1. An illustration of the problem statement.

2. LANDING PAD DETECTION

After the long process of searching for a suitable method for landing pad detection, the method of classifier training in OpenCV was found (Docs.opencv.org, 2015a). A classifier is an object that behaves like an intelligent brain. The classifier can be trained to associate with certain input stimuli and disassociate with others. In this sense, it can learn how to tell the difference between a landing pad and a non-landing pad object.

For this research, four different classifiers were trained. For each classifier, two databases, a positive and a negative database, were required. Two types of landing pad were created (a **X** type and **H** type) from matt paper, as illustrated in Figure 2. Both of these pads feature a ring for enhanced classification accuracy.

Figure 2. Landing pad designs.

The landing pads were filmed and a number of positive image databases were assembled, by capturing frames from the videos. As the cropped images provided good contrast, they were not grayscaled or equalized. As illustrated in Table 1, three positive databases were created, the first containing images of both the **X** type and **H** type landing pads (4534 image), the second containing only **X** type images (2353 images), and the third only **H** type images (2185 images). A single negative images database, containing 3573 random non-landing pad images, was also created for use with all of the classifiers.

From the **H** type positive image database and the negative images database, a **H** type classifier was trained. Figure 3 displays images from the output video from the detection test for the **H** type classifier. The ROI (the

region of interest), frame centre and distance between them have been marked by using the output from OpenCV, which will further be discussed in Section 3.

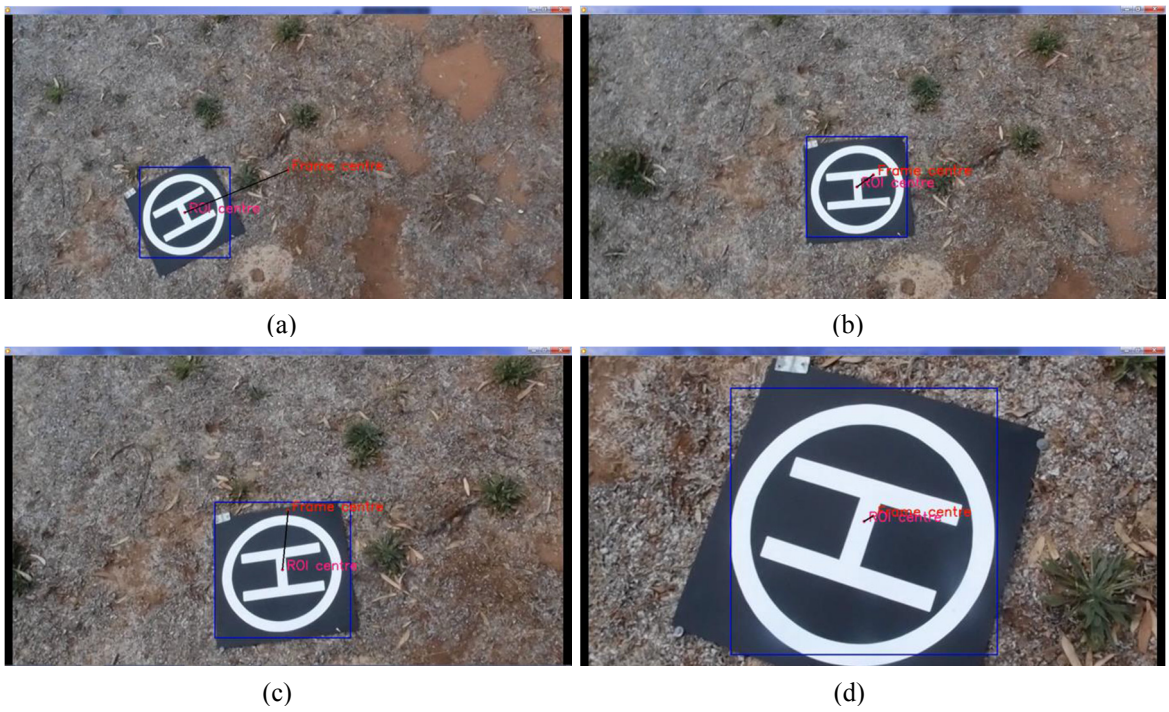

Figure 3. Frames from the video from the detection test using the **H** type classifier. (a) Detection at 6 seconds, (b) detection at 21 seconds, (c) detection at 33 seconds & (d) detection at 46 seconds.

A **X** type classifier and a **H/X** (combined) type classifier were similarly trained using the same negative images database as before, and the **X** and **H/X** positive image databases respectively. All three of the aforementioned classifiers are LBP classifiers, i.e. classifiers that were trained with OpenCV to look for local binary patterns, as described in Liao et al. (2007). The other available training method in OpenCV is the more accurate Haar method, described by Viola & Jones (2001). In order to observe differences in performance characteristics between Haar training and LBP training, a Haar trained **H/X** type classifier was also created. As listed in Table 1, the training time for the Haar trained combined classifier was 8 hours and 8 minutes, while the training time for the LBP trained combined classifier was 1 hour and 5 minutes. Therefore, it is clear that Haar training is much slower than LBP training, when applied to our training data sets, as was expected.

It can also be seen from Table 1 that the training time increases with the number of images used. All classifiers successfully detected both landing pads, with little observed difference in performance of any classifier between **H** type and **X** type landing pads. The reason for this is thought to be the fact that both landing pads designs are quite similar; the main feature is a circle, and the shape inside the circle has the same number of edges and corners in both cases. The **H/X** type classifier was observed to be a little more accurate in detections than the individual **H** type and **X** type classifiers, so the LBP **H/X** type classifier was used in subsequent work.

Table 1. Cascade classifier training times

Type of Landing Pad	Number of images in positive database (*totalPos*)	The number of positive images used per stage (*numPos*)	Number of images needed in negative database (*numNeg*)	Training method (*featureType*)	Time
H/X	4534	4140	2070	LBP	**1h 05m 49s**
H/X	4534	4140	2070	HAAR	**8h 08m 21s**

H	2185	1995	997	LBP	**12m 05s**
X	2353	2148	1074	LBP	**34m 18s**

The value of *numPos* (the number of positive images used per stage) was found according to (1) from Au.mathworks.com (2015).

$$numPos = \left\lfloor \frac{totalPos}{1 + (numStages - 1) \times (1 - minHitRate)} \right\rfloor \quad (1)$$

where $minHitRate$ denotes the minimum hit rate, which was assigned the default value (0.995),

and $numStages$ denotes the total number of training stages, which took a value of 20.

3. LOCALISATION

3.1 Overview

The imaging algorithm used to recognise the pad is based on examples in the OpenCV documentation for face detection (Docs.opencv.org, 2015b). If the landing pad is detected, a rectangle, known as the 'region of interest' or ROI, will enclose it. The dimensions of the ROI and its coordinates, with respect to the image frame, were obtained using OpenCV. Given this knowledge, the centre of the ROI can be found, which would ideally also be the centre of the landing pad. As shown in Figure 4, the dimensions of the image frame are denoted l' and w', and the coordinate system used within this frame is x', y', z'.

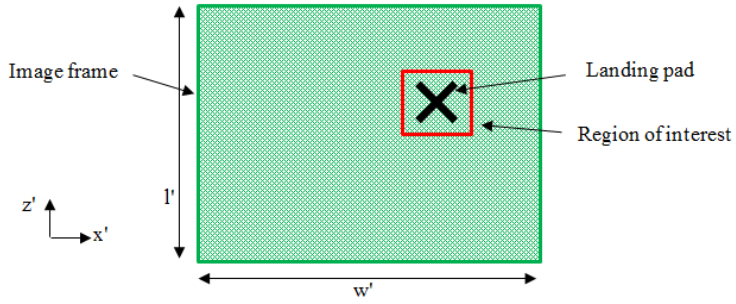

Figure 4. An illustration of the image frame.

3.2 Localisation (Simple Case)

If the UAV is in level flight (parallel to the ground), and the camera is located beneath the geometric centre of the UAV, then the centre of the image coincides with the relative on-ground position of the UAV. The heading of the UAV is determined by a vector from the centre of the image to the centre of the ROI (the 'relative vector', see Figure 5). The magnitude of this vector, denoted d', is the relative distance between the points. The actual distance (the distance in earth-coordinates) can be found using a scaling factor that will convert between image-frame-coordinates (x', y', z') and earth-coordinates (x, y, z).

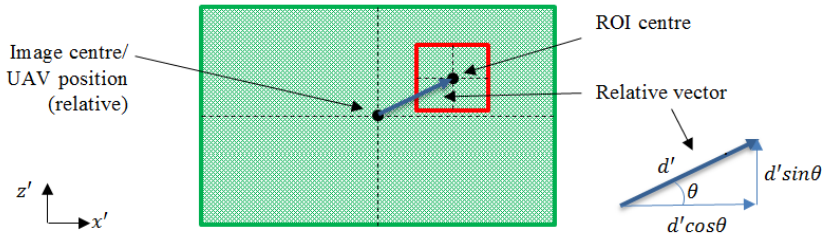

Figure 5. Location of the UAV within the image frame, as well as the relative vector (broken down into its x' and z' components).

The bearing of the landing pad relative to the UAV heading (θ), at the time of image capture, can be determined by finding the angle relative to the horizontal (2), evaluating its value against a set of criteria (3)

and then transforming it to a form that is relative to the heading of the UAV at the time of capture of the image (the positive z' axis).

$$\theta = \tan^{-1}\frac{LandingPadx - UAVx}{LandingPadz - UAVz} \quad (2)$$

$$\{if\ (LandingPadx > UAVx)\ \rightarrow \theta = 90 - \theta,$$
$$else\ \rightarrow \theta = 270 - \theta\} \quad (3)$$

The result is then added to the bearing of the UAV at the time of capture of the image (the bearing with respect to earth-fixed North) to find the absolute yaw angle (the yaw angle relative to North):

$$yaw = \theta + bearing \quad (4)$$

The scaling factor to convert between image-frame-coordinates (x', y', z') and earth-coordinates (x, y, z) is determined by the field of view of the camera and the altitude of the UAV. For a UAV at an altitude h, with a field of view α_x, the span of view w along the x axis is given by:

$$w = 2h\tan\frac{\alpha_x}{2} \quad (5)$$

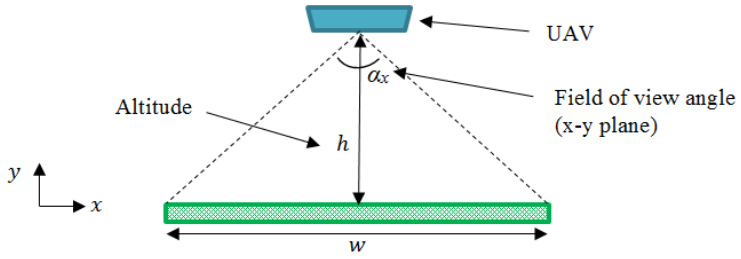

Figure 6. An illustration of the situation for the simplified case.

The scaling factor in this axis can then be determined by considering the difference in magnitudes between this distance and the width of the image frame; i.e., $c_x = \frac{w}{w'}$ and takes the following form:

$$c_x = \frac{2h\tan\frac{\alpha_x}{2}}{w'} \quad (6)$$

Similarly, along the z axis:

$$c_z = \frac{2h\tan\frac{\alpha_z}{2}}{l'} \quad (7)$$

The travelling distances (w' and l') can then be found accordingly.

3.3 Localisation (General Case)

In general, the UAV may not be parallel to the ground, as it may roll or pitch in order to move in the x or z directions, so the centre of the image will no longer correspond to the relative position of the UAV. A transformation is therefore necessary to find the location of the UAV.

The roll angle, shown in the x-y plane of Figure 7, is denoted β_z. The UAV is no longer at the centre of the image, so variables w_1 and w_2 are defined as the distances from the UAV position to the ends of the frame (previously, for the simple case, $w1$ and $w2$ were equal, and $w_1 + w_2 = w$). The relationships between these distances and the angles α_x and β_x can be found by observation of the resulting triangles as follows:

$$w_1 = h\tan\left(\frac{\alpha_x}{2} + \beta_z\right) \quad (8)$$

$$w_2 = h\tan\left(\frac{\alpha_x}{2} - \beta_z\right) \quad (9)$$

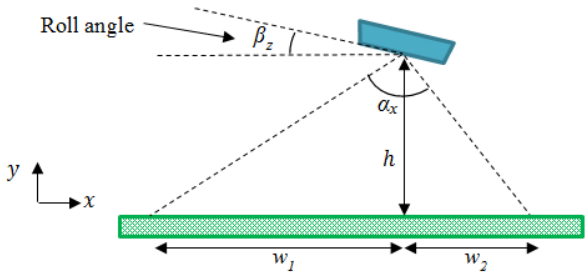

Figure 7. An illustration of the situation for the general case.

The position of the UAV within the image can now be found using the relationship $\frac{w'_1}{w'_2} = \frac{w_1}{w_2}$, and the relative vector can be drawn, as in Figure 8. The scaling factor will be $\frac{w_1+w_2}{w'_1+w'_2}$ where $w'_1 + w'_2$ is simply the frame width. Thus, the travelling distance (d) can be found; the angle θ is found again by trigonometry. It is worth noting that for small β_z, the results converge to what was established in the simple case (w_1=w_2 and $w_1 + w_2 = w$), i.e. the simple case is one particular limit of the general case. There are also upper limits to β_z. Firstly, β_z must be less than $\frac{\alpha_x}{2}$, otherwise w_2 will be zero or negative. β_z must also be less than $90 - \frac{\alpha_x}{2}$, otherwise w_1 tends to infinity. To summarise, the condition is that $\beta_z < \min[\frac{\alpha_x}{2}, 90 - \frac{\alpha_x}{2}]$. As with the simple case, all of these calculations can be performed again for UAV pitch (non-zero β_x), yielding similar results for different variables. When there is both pitch and roll present, the relative vector is found as above for each. The resultant relative vector is then found through vector addition.

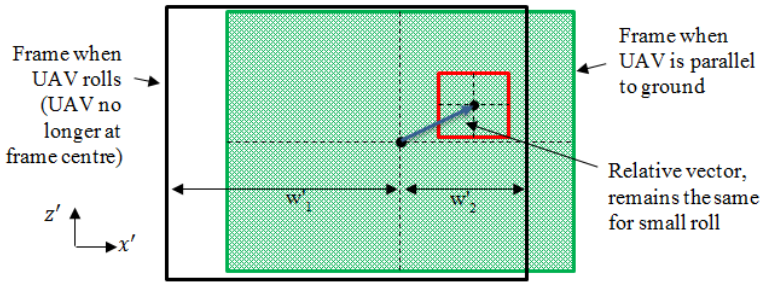

Figure 8. Frame shift due to UAV roll.

4. COMMAND

Once the values of the distance to the centre of the region of interest in earth-coordinates (d, the travelling distance) and the absolute yaw angle (the steering angle) are known, the UAV can navigate to the landing pad as follows:

a. Rotate on the spot (yaw) to the desired heading.

b. Proceed forward by the travelling distance (d), whilst maintaining altitude hold.

c. Repeat steps a and b until the UAV is above the landing pad ($d \approx 0$), and then perform a controlled descent.

GPS is error prone, so it cannot be used to navigate the UAV to the centre of the ROI; hence, the camera must be used. After a certain number of frames, the system reassesses the distance and angle to this position, corrects UAV heading and commands it to travel the reassessed distance. The rate of reassessment depends on distance, and it continuously increases as distance reduces; this closed-loop approaches prevents the UAV from overshooting the target location. When the distance is less than a certain limit, the system must enter the UAV into landing mode.

In general, the UAV images its surroundings, and this data, along with UAV sensor data, is received by the operator's computer. The computer uses an imaging algorithm to detect the landing pad, and then uses a localisation algorithm to determine directions to the landing pad. Based on these directions, control signals are generated and transmitted to the UAV, which then generates a set of new commands in response. This process, as illustrated in Figure 9, is repeated until the UAV lands.

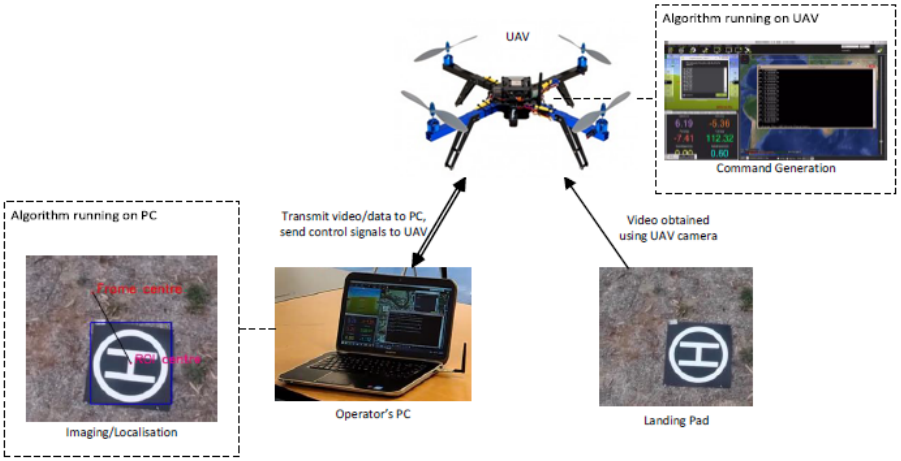

Figure 9. High level overview.

5. DISCUSSION AND CONCLUSIONS

This paper has discussed a basic procedure for navigating a UAV to a landing pad using a camera, landing pad detection using OpenCV, localisation of the UAV within the earth-fixed coordinate system and command generation for UAV navigation. The LBP **H/X** type classifier was found to be the most suitable for landing pad detection. The individual **H** type and **X** type classifiers performed similarly to one another, due to similar landing pad designs. They also exhibited performance similar to the **H/X** type classifier, but required a smaller database for training and lesser training time. In terms of localisation, two cases were covered; simple and general. Determination of the yaw angle, and of the scaling factors that allow for the determination of the travelling distance, were also demonstrated. In terms of command generation, the principles for navigating the UAV, using the results from the previous two steps, have been demonstrated. Overall, this paper provides good insight into basic UAV navigation by digital imaging.

6. FUTURE WORK

The process of navigating the UAV to the centre of the landing pad has been tested on a real UAV test-bed (an Arducopter) with its motors turned off; i.e., a camera has been placed underneath the UAV test-bed and the structure has been moved manually to test the implemented navigation system. The system was successfully able to report the travelling distance and yaw angle, for a variety of simulated UAV motions. Future work demands the testing of this system for a UAV test-bed for different real life scenarios.

ACKNOWLEDGMENTS

The authors wish to acknowledge that the research presented in this paper was part of a project funded by Boeing Research and Technology - Australia. The authors would also like to acknowledge and thank Dr Jason Armstrong and Mr Brendan Williams for their insightful research idea, support throughout the project and encouragement.

REFERENCES

Au.mathworks.com. (2015). *Train a Cascade Object Detector - MATLAB & Simulink*. Retrieved 10 September 2015, from http://au.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

Docs.opencv.org. (2015a). *Cascade Classifier Training — OpenCV 2.4.11.0 documentation*. Retrieved 5 September 2015, from http://docs.opencv.org/doc/user_guide/ug_traincascade.html

Docs.opencv.org. (2015b). *OpenCV: Face Detection using Haar Cascades*. Retrieved 5 September 2015, from http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0

Liao, S., Zhu, X., Lei, Z., Zhang, L., & Li, S. Z. (2007). Learning multi-scale block local binary patterns for face recognition. In Advances in Biometrics (pp. 828-837). Springer Berlin Heidelberg.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-511). IEEE.