A Robust UAV Landing Site Detection System Using Mid-level Discriminative Patches

Xufeng Guo, Simon Denman, Clinton Fookes, Sridha Sridharan
Image and Video Laboratory, Queensland University of Technology (QUT), Brisbane, Australia
Email: {felix.guo, s.denman, c.fookes, s.sridharan}@qut.edu.au

***Abstract*—The forced landing problem has become one of the main impediments to UAV's entering civilian airspace. Unfortunately there is no robust forced landing site detection system that will reliably detect a safe landing site. One of the main reasons for this is the difficulty in considering the various classes of surface, to determine whether they are safe or not. We propose a robust UAV landing site detection system using mid-level discriminative patches. The training and tuning process uses a dataset containing 1600 randomly selected Google map images with weak labels. We then show how the output from multiple mid-level discriminative patch detectors can be combined to indicate the level or danger for a given region. The proposed technique reliably detects safe landing areas in UAV imagery, and achieves improved performance over the state-of-the art. The proposed system outperforms the baseline system by 29.4% for completeness and 33.9% for correctness, and is invariant to the changes of illumination, sharpness and resolution of images.**

I. Introduction

Unmanned Aerial Vehicles (UAV) have become widely used, however there remains a safety issue that prevents UAVs entering urban areas. The lack of an automated UAV forced landing site detection system is identified as one of the key impediments to the wider deployment of UAVs [1]. When a malfunction occurs, to cater for all contingencies the UAV itself must be able to quickly locate a safe landing site to perform an automated forced landing from visual information only [2]. Previous research focuses on hand-crafted systems that provide the forced landing sites with extremely high accuracy rate under a known environmental condition [1], however such systems detect very few available landing sites [3] and lack robustness to unseen datasets.

In this article we develop a system that locates candidate landing sites for a forced landing using mid-level discriminative patches. Detected landing sites should be free of humans or man-made structures such as buildings, roads and cars in order to prevent loss of life and property; and if possible, the UAV should select a site that will avoid damage to the UAV. Hence we suggest a tiered safety ranking approach, where a region is classified as 'very dangerous' if the area is likely to contain people; 'not recommended' if the area is free of people and buildings, but may cause damage to the UAV; and 'safe' for an area that is recommended for landing.

Rather than use a hand-crafted and heuristic approach, in this paper we learn the features of dangerous areas directly from images, allowing the computer to learn what makes a landing site look dangerous. The proposed approach offers a number of benefits over existing heuristic approaches: 1) it uses a simple feature representation that can represent all classes; 2) it incorporates context by considering the region surrounding a pixel; and 3) it only requires coarsely labeled data, greatly reducing the annotation requirements for training.

The main contributions of this paper are: 1). Using mid-level discriminative patch detection with weakly labeled data to automatically learn what makes a region dangerous; 2) the system ranks each area in terms of its level of danger in the form of a heat map; and 3) we perform evaluations over a series of images captured by actual UAV flights, in which the resolution and illumination conditions are totally different. The contribution also includes the release of a database containing 1600 aerial images with weak labels to learn the appearance of 'very dangerous' and 'not recommended' areas, and another database containing three sequences of images captured from UAV flight with pixel level region labeling to evaluate the proposed approach.

II. Prior Research

Many existing approaches to the safe landing problem have been proposed, including a heuristic combination of an artificial neural network (ANN) classifier with a Canny edge detector [1], [2]; through applying a K-means model and edge features [4]; by searching for a predefined pattern using a heuristic process [5], [6]; by employing an optical flow model with Scale-Invariant Feature Transform (SIFT) to locate known landmarks [7]; and by fusing multiple block based features and using support vector machines (SVMs) to classify a site as safe or unsafe [3]. However, these methods all focus on low level local features such as the texture or color of a given pixel or a small patch. The broader context in which that feature occurs (i.e. is this patch of green part of a small garden, or a large field?) is not taken into account. For instance, by unilaterally considering colour or texture information, a flat gray coloured area that comes from a roof top appears similar to a region of flat ground.

At the other extreme are approaches that consider a whole image as a feature. Such approaches include inferring location from high resolution photographs [8], and scene recognition from down-sampled (32×32 pixels) imagery [9]. In spite of the outstanding performance of these systems for scene classification [8], [9], they require a large amount of training data (often millions of images) in order for the system to adequately model all possible environments. Additionally, for the application of UAV forced landing site detection, the system

Table I: Definition of the three classes of region: 'Very Dangerous', 'Not Recommended' and 'Safe'.

Class	Likely Content	Suggested UAV Action
'Very Dangerous'	Buildings, roads, car parks, people, and other areas that may contain people	Definitely Avoid
'Not Recommended'	River, sea/ocean or other area that may cause damage to the UAV	Fine to land, but not recommended
'Safe'	Grass, clear land, or other areas that will be suitable to perform a proper landing.	Ideal for landing

needs to produce several ranked candidate landing sites within a given view, instead of treating the image as a whole.

III. Proposed Approach

A. System Overview

The proposed approach is illustrated in Fig. 1. We train patch detectors for 'very dangerous' and 'not recommended' classes using 1,200 weakly labeled Google map images, and tune detection parameters on a further 400 weakly labeled Google map images. The tuned output from these detectors is then used to determine how dangerous each region in an image is, generating a 'heat map', which can subsequently be used to determine a safe landing site by applying a thresholding operation which could optionally consider other data from the UAV (i.e. fuel load, severity of fault). Note that, in this paper we use the term 'patch' to denote the sub-image of a given image. Specifically, the patch size in our system is between $80px \times 80px$ and $400px \times 400px$. The following sub-sections will explain each component of the system in detail.

B. Training and Tuning Process

To train the mid-level discriminative patches we follow the approach of [10]. We use weakly labeled training data, in which every image is labeled as 'Very Dangerous', 'Not Recommended', or 'Safe'; indicating the images suitability for a forced landing. Definitions and likely image content for each of these classifications are provided in Table I.

Initialization: The training set is partitioned into two sets and each set consists of positive and negative images. We denote the positive and negative images in each set as P_1, P_2, N_1 and N_2. The training process first extracts HOG features [11] from 8×8 cells from patches that are randomly sampled from P_1, and performs a k-means clustering in HOG space with a large number of clusters (one quarter of the total number of patches). Clusters with less than three patches are removed and the remaining clusters form our initial set.

Iterative Training: Using this initial set, linear SVMs are trained for each cluster. The patches within a particular cluster are positive and the patches that are randomly sampled from N_1 are negative samples. After applying these SVMs to perform detections on the second subset of positive images (P_2), high confidence detection patches results for each SVM form a new set of clusters. This set of clusters as well as patches sampled from negative images (N_2) is then used to train a new set of linear SVMs similarly, which then perform detections on P_1 to update the initial set of clusters. This process of training linear SVMs and using them to refine the set of clusters is repeated,

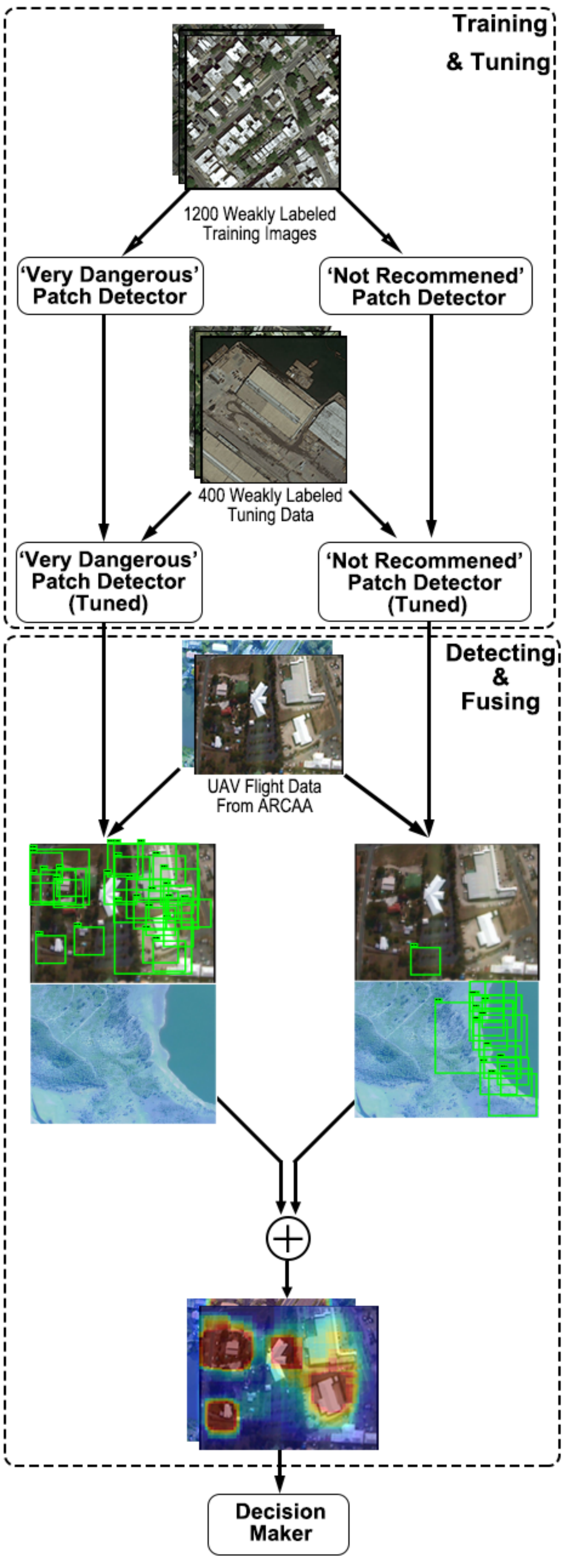

Training & Tuning
1200 Weakly Labeled Training Images
'Very Dangerous' Patch Detector
'Not Recommened' Patch Detector
400 Weakly Labeled Tuning Data
'Very Dangerous' Patch Detector (Tuned)
'Not Recommened' Patch Detector (Tuned)
Detecting & Fusing
UAV Flight Data From ARCAA
Decision Maker

Figure 1: System Overview: Detectors for 'very dangerous' and 'not recommended' areas are trained, and tuned on two sets of Google map data. The output on testing data from these detectors can then be combined to generate a 'heat map' that describes the level of danger for a given area. This 'heat map' can then be used to identify a safe landing site.

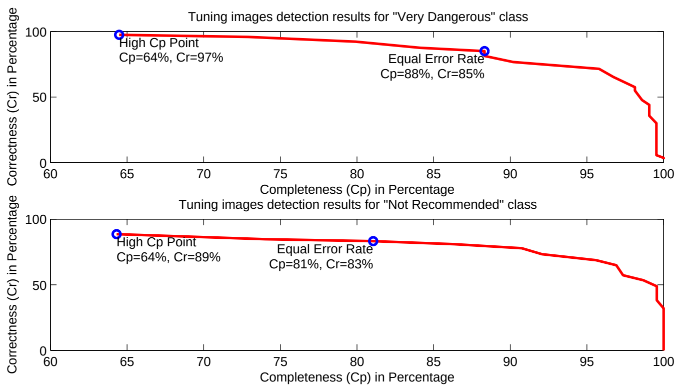


Figure 2: System performance with the tuning dataset as the parameters are varied. To generate the 'heat map', we select the parameter values that maximize the overall Correctness (Cr) and Completeness (Cp) as defined in Section IV-B

either for a given number of iterations or until convergence. For our approach, we set the number of iterations to 3 since in practice we find that the algorithm converges within 3 iterations for this problem.

The iterative training ensures both the purity and the discriminativeness of detected patches, as only patches that are strongly associated with the target class remain after several iterations. After the iterative training, the resulting SVMs form a detector for a particular class. We repeat the same process separately for two classes to get two detectors, one for 'very dangerous', and one for 'not recommended' areas.

Tuning Stage: The detectors will extract a large number of patches from every image, and give every patch a confidence score in the range of $[-1.0, 1.0]$, representing the detector confidence when classifying the patch (with 1.0 being certain the patch is 'very dangerous'/'not recommended').

In order to emphasise the contribution of high confidence patches and reduce the effect of low confidence patches, a sigmoid function is applied to the confidence value of each detected patch. The sigmoid function will significantly encourage the high confidence inputs, and restrict the low confidence inputs in a non-linear way.

In the tuning stage, the system takes different parameter combinations and tries to detect patches in the tuning set, comparing the results with the weak labels of the tuning set. The performance of detectors on the tuning set is shown in Fig. 2. We choose an operating point that outputs good overall results on the tuning set, and use the corresponding sigmoid function parameters to form two sigmoid functions f_d and f_{nr}.

Fig. 4 shows some examples of high confidence patches picked up by the system for each detector and Fig. 3 indicates the detected patch location in the input images. In both figures, Subfigure. (a) and Subfigure. (b) are patches picked up by the 'very dangerous' detector, demonstrating that the 'very dangerous' detector tends to pickup roads (Subfigure. (a)), buildings and car parks (Subfigure. (b)). Subfigure. (c) are patches detected by the 'not recommended' detector, which shows that the detector has determined that the presence of water (river or sea) renders the site safe, but not suitable for the UAV to perform a proper landing. Subfigure. (d) shows patches that are detected as 'safe'.

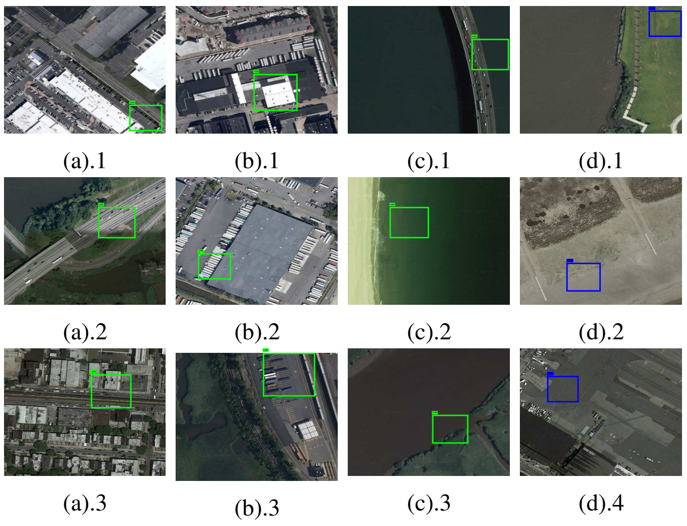
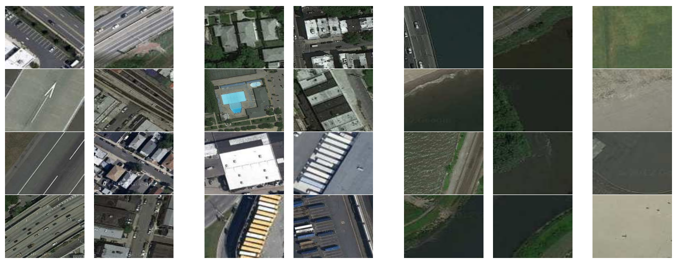


Figure 3: Patches automatically selected from weakly labeled Googlemap images: 'Very dangerous' patches selected by the system are shown in (a) and (b); 'Not recommended' patches are shown in (c); and 'safe' patches (i.e. patches that are rejected as being 'Very dangerous' or 'Not recommended') are shown in (d).



(a) 'Very dangerous' (b) 'Very dangerous' (c) 'Not recommended' (d) 'Safe'

Figure 4: Examples of Detected Patches from Training Data: 'Very dangerous' patches tend to be roads (a), buildings and car parks (b); 'Not recommended' patches tend to contain river and beaches (c); and 'safe' patches tend to be grass and flat ground (d).

The patches shown in Fig. 4 and Fig. 3 clearly indicate that: 1) the system is able to learn which significant visual elements make a site look 'very dangerous' or 'not recommended'; and 2) the system considers context: a plain, textureless area on a roof top is recognized as 'very dangerous' (Fig. 3 (b).1), while a similar textureless region from a white beach is considered to be 'safe' (Fig. 3 (c).2).

C. Detecting a Safe Landing Site

When the system is trained and tuned, it is used to detect safe landing sites. For each input image, the output is a risk assessment in the form of a heat-map, which describes the level of danger for each area.

Firstly, tuned ‘very dangerous’ and ‘not recommended’ patch detectors are used to detect patches in a given image. As illustrated in Fig. 1, the outcome of each discriminative patch detector is roughly between $0 - 50$ bounding boxes, each with a confidence score. Depending on the confidence score of each box, the box will contribute between 0 and 255 to the corresponding area in final heat map. The mapping from a confidence score to an intensity value is performed by the tuned sigmoid function mentioned above. The heat map generation is described in Equation. 1, where N is the number of dangerous patches that include pixel x, y, M is the number of ‘not recommended’ patches, f_d and f_{nr} are the tuned sigmoid functions, and $P_d(i)$ and $P_{nr}(i)$ are confidence value of the detected patch that include pixel x, y for the dangerous class and not recommended class respectively.

$$H(x,y) = min(255, \sum_{i=0}^{N} f_d(P_d(i)) + \sum_{i=0}^{M} f_{nr}(P_{nr}(i))) \quad (1)$$

The summed result forms a heat map which is used to decide on a landing site. The simplest method for doing this is to use a threshold, $thres$, to generate a binary image, such that pixels in the heat map that are higher than the threshold are converted to logical ‘0’, meaning ‘not safe’. In an actual UAV flight, this process should consider additional factors. For instance, if fuel is very low such that UAV must land now, then the threshold can be progressively lowered to take the best option available. Similarly, when fuel is plentiful, a more strict threshold can be used to locate an ideal landing site.

IV. Evaluation Protocol

A. Data and Ground Truth

We use four data sets in our experiments.Google maps images are used to train our patch detectors and tune the heat-map generation, and three sequences of images captured from a UAV [1] are used to test the proposed approach.

The training and tuning images are 1600 randomly selected locations downloaded from Google maps. Images are selected from a region to the south of New York city, bounded by $(40.36°N, -74.23°W)$ and $(40.96°N, -73.63°W)$, as shown in Fig. 5. The database covers a wide range of environments, including urban, suburban, rural and ocean under a variety of illumination conditions. The image size of each image is $512px \times 512px$ with a resolution of approximately $1.21m^2/px$. Out of the 1600 images, a randomly selected subset of 1200 images is used for training the two discriminative patch detectors, and the remaining 400 images are for tuning parameters such as the upper asymptote, the growth rate and the x-value of the mid point in the sigmoid function to optimise heat map generation. Example training and tuning images are shown in Fig. 3. The major advantage of our system is that we don’t need to give strong labels to the training data, hence all 1600 Google map images are only weakly labeled as either ‘very dangerous’, ‘not recommended’ or ‘safe’ (see Table I for classification descriptions) [1]. Three annotators are used to label images, and

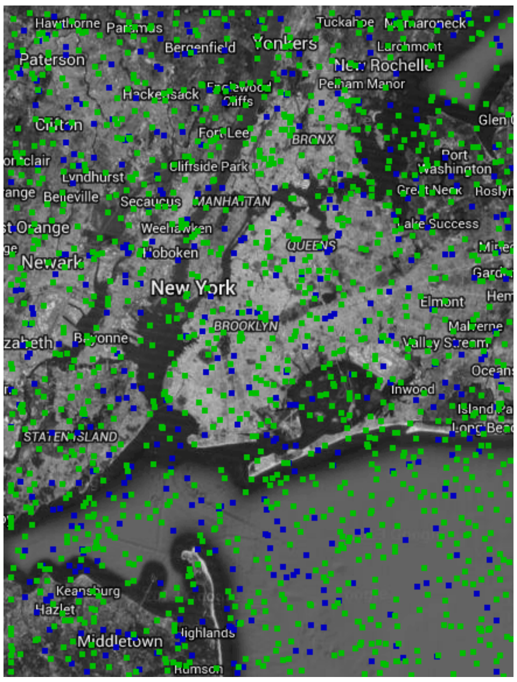


Figure 5: Aerial imagery was collected in 1600 randomly selected locations (shown by colored blocks) across the south New York Region. 1200 images (shown in green) are randomly selected as training data, and the remaining 400 images (shown in blue) are selected as the tuning data for system. [Image CC-BY-SA Googlemap Contributors and QUT]

Table II: Experimental data set. Note that the testing data is significantly different from the training data in terms of resolution and illumination condition (weather condition).

Data set	Whether	Resolution	Training	Tuning	Testing
Google map	Sunny	$1.21m^2/px$	1200	400	0
ARCAA - A [1]	cloudy	$0.07m^2/px$	0	0	700
ARCAA - B [1]	Sunny	$0.57m^2/px$	0	0	522
ARCAA - C [1]	Sunny	$0.07m^2/px$	0	0	8400

only images on which all three annotators agree are included in training and validation sets.

The three UAV testing datasets are more challenging than the Google map images, and contain a variety of altitude changes and illumination variations [1]. For the testing sequences, we manually label all pixels in the image as one of three classes ‘Very Dangerous’, ‘Not Recommended’ or ‘Safe’, based on definitions in [1], and as outlined in the Table I. Fig. 7 shows example images from this testing dataset with the corresponding hand annotated ground truth.

B. Performance Metrics

The completeness (Cp) of a system represents the percentage of correctly detected ‘safe’ landing areas out of all ‘safe’ areas, and is defined as

$$Cp = \frac{S_{cds}}{S_s} = \frac{S_{cds}}{S_{cds} + S_{fdu}} = \frac{TP}{TP + FN}, \quad (2)$$

where S_{cds} is the area of correctly detected ‘safe’ regions returned by the system; and S_s is the area of all ‘safe’ landing sites annotated in the ground truth. If there are no ‘safe’ landing

[1]Please contact the authors to obtain a copy of this data

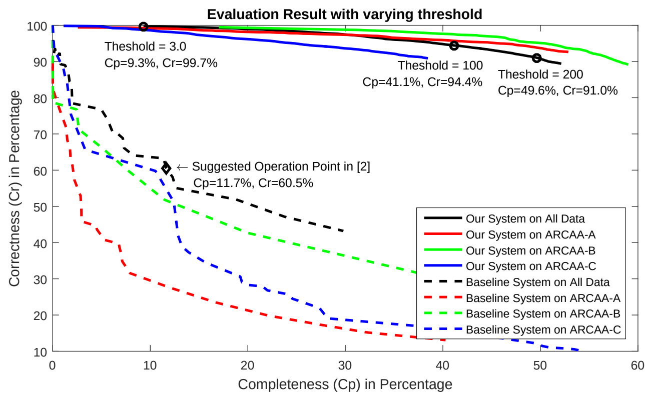


Figure 6: Evaluation Result Plot: Performance of the proposed approach is shown with solid line as the detection threshold is varied. Performance of the baseline system [1] is plotted with a dashed line. For each system, Black represents the average performance of three different datasets, while Red, Green, and Blue represents the ARCAA-A, ARCAA-B and ARCAA-C datasets respectively. Note that our system outperforms the baseline, and is more stable when conditions vary.

sites available in the groundtruth image, the completeness is assumed to be 100%.

The correctness (Cr) of a safe landing site detection result represents the percentage of the correctly detected 'unsafe' areas out of all 'unsafe' areas, and is defined as

$$Cr = \frac{S_{cdu}}{S_u} = \frac{S_{cds}}{S_{cdu} + S_{fds}} = \frac{TN}{TN + FP}, \tag{3}$$

where S_{cdu} is the area of correctly detected 'unsafe' regions, which is denoted by the black areas in the binary image output; and S_u is the area of all 'unsafe' landing sites marked by the ground truth. If there are no 'unsafe' landing sites in the ground truth image, the correctness is assumed to be 100%.

V. Experimental Results

We evaluate our system on three sequences captured from UAV flights (see Table. II). As previously mentioned, the heat map is only an intermediate output of the final UAV forced landing system, and a secondary process is responsible for using this, alongside other data such as the amount of fuel remaining, to decide where to land. However for the purposes of evaluation, we use a simple threshold to determine safe landing sites. We compare our approach to the state of the art forced landing site detection system in [1], using the same parameters as is recommended in their paper.

Fig. 6 plots the performance of our proposed approach (in terms of completeness and correctness) with solid lines as the threshold is varied from $0 - 255$. The performance of the baseline system [1] is also shown with dashed lines as the parameters in the baseline system vary. For each system, we present the performance for the dataset ARCAA-A (in red), ARCAA-B (in green), ARCAA-C (in blue) and the averaged performance across all three testing datasets (in black). It is clear from Fig. 6 that irrespective of the threshold and of the dataset type, the proposed system outperforms the baseline, achieves a correctness higher than 90%. In ARCAA-C, the correctness of our system drops off faster than for the other two datasets, since the majority images in the ARCAA-C dataset are taken in urban areas with dense residential or industrial buildings, and thus the 'safe' regions are far fewer and smaller. As such, it is hard for our patch based method to detect a 'safe' region without including any residential areas. For instance in Fig. 7 (b), the lawn with no obstacles or man-made structures is considered to be 'unsafe' due to it's proximity to other unsafe regions.

Despite the quicker drop off of our system on ARCAA-C, our approach is more stable than the baseline when resolution is reduced (dataset ARCAA-B), or when illumination conditions change (dataset ARCAA-A). The variation in terms of the correctness in our system is less than 10%; whilst the correctness of the baseline system is reduced dramatically in ARCAA-A and ARCAA-C. The heuristic nature of the baseline system makes changes in illumination highly problematic, leading to greater inconsistency [3].

The training data used by the proposed system contains a variety of illumination conditions, and the iterative training process ensures that the final system detects patches that are pure (the patches only contains the target class) and discriminative (the patches rarely exist in the non-target class) [10]. Thus, detected patches are robust to illumination conditions; and robust to resolution changes since the size of the patches vary. The invariance is demonstrated by the fact that the proposed system is trained and automatically tuned on Google map data, which has a different resolution, illumination conditions, image sharpness and geometric location than the testing data (Table II). This mismatch in training and testing conditions demonstrates that the proposed approach is highly robust to changes in conditions. In the averaged curve (Figure 6), we show three thresholds for the proposed approach: $thres = 3$, $thres = 100$ and $thres = 200$; representing situations where there is increasing urgency to find a landing site. As is shown, when a very conservative threshold is set (i.e. $thresh = 3$), the system can avoid 99.7% of dangerous areas (correctness) whilst still detecting 9.3% of the safe landing sites (completeness). When $thres = 100$ and $thres = 200$, the system performance is $Cp = 41.1\%, Cr = 94.4\%$ and $Cp = 49.6\%, Cr = 91.0\%$ respectively, illustrating how more suitable landing sites can be identified, with only a slight increase in the number of false sites found. If we compare the performance of our system on a typical threshold $thres = 100$ with that of the baseline sytem on the parameters suggested by [1], our system achieves an improvement of 29.4% in completeness and 33.9% in correctness.

Example images produced by the system are shown in Fig. 7, in which (a) are input images, (b) are the computed heat maps, (c), (d) and (e) are the binary output images for different thresholds ($thres = 100$ and $thres = 200$) and from the baseline system [1], and (f) are the corresponding ground truth images. Note that for the ground truth images, the red, yellow and blue colours represent 'very dangerous', 'not

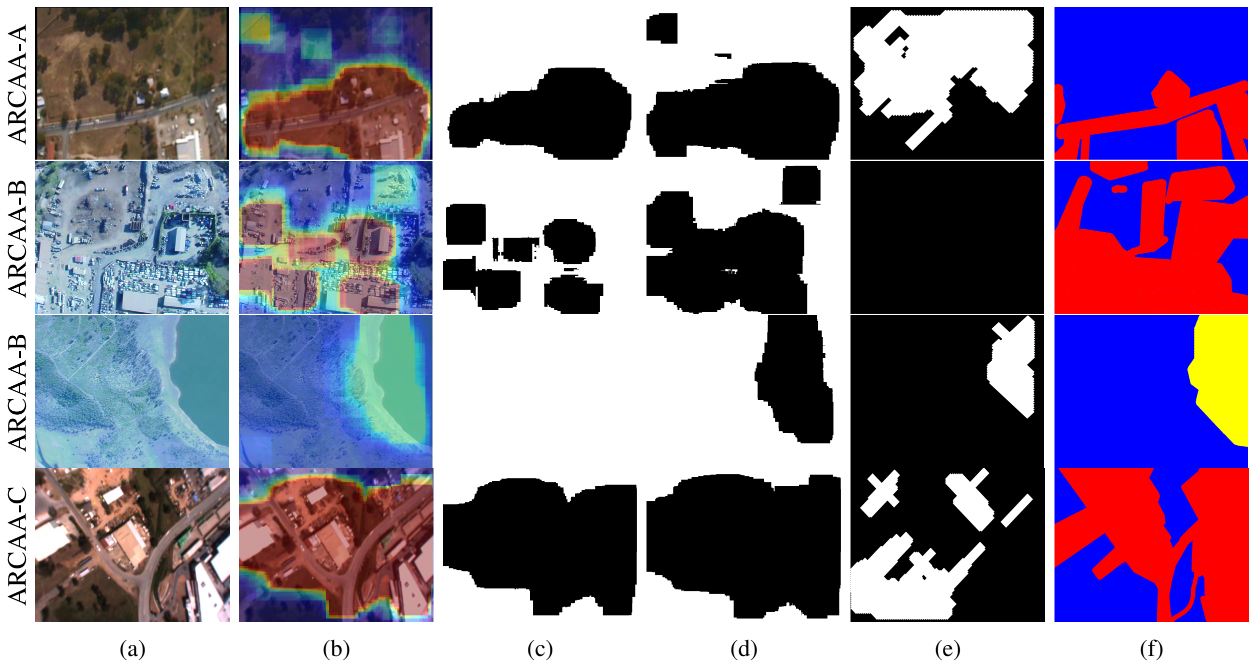


Figure 7: Examples of a):input images; b): system generated heat map, in which a warmer color indicates a greater level of danger for performing a forced landing; c): binary output for $thres = 100$ (white represent 'safe' and black represent 'unsafe'); d): binary output for $thres = 200$; e): binary output of the baseline system (Mejias [1]); and f): ground truth of the corresponding images, in which Red represent 'Very Dangerous' class, Yellow represent 'Not Recommended' class and Blue represent the 'Safe' class. All images CC-BY-SA ARCAA and QUT

recommended' and 'safe' respectively. The third row of images show an example of how the threshold affects the definition of some areas of water: when the UAV is allowed to fly for a short range to find a better landing site, the water area are is not recommended for landing (Fig. 7(d)), however, when the UAV needs to land immediately, the water areas are considered to be safe (Fig. 7(c)). Note that, the system is robust enough to prevent areas that look 'very dangerous' such as building and roads, from being considered 'safe'.

VI. Conclusion

This paper proposes a robust UAV landing site detection system using mid-level discriminative patches to automatically determine the key differences between dangerous and safe regions. We show how multiple mid-level discriminative patch detectors can be combined into a robust forced landing site detection system, that generates heat map of landing site danger levels. The systems is trained and tested on separate corpus, demonstrating it's robustness to changing image conditions such as those that may be encountered in the field. By generating the heat-map as an intermediate result, we are able to vary the operating point to reflect the need to find a landing site. This allows the other parts of the UAV control system to incorporate factors such as hardware condition and fuel load, such that the threshold reflects the urgency with which a site must be found. Importantly, we show that the system is able to avoid the vast majority of 'very dangerous' (i.e buildings and car parks) areas for all threshold selections. For a typical threshold ($thres = 100$), our system achieves an absolute performance improvement of 29.4% for completeness and 33.9% for correctness over the state of the art.

References

[1] L. Mejias, "Classifying natural aerial scenery for autonomous aircraft emergency landing," in *ICUAS*, May 2014, pp. 1236–1242.

[2] L. Mejias, D. L. Fitzgerald, P. C. Eng, and L. Xi, "Forced landing technologies for unmanned aerial vehicles towards safer operations," *Aerial Vehicles*, pp. 415–442, 2009.

[3] X. Guo, S. Denman, C. Fookes, L. Mejias, and S. Sridharan, "Automatic uav forced landing site detection using machine learning," in *DICTA*. IEEE, 2014, pp. 1–7.

[4] Y.-F. Shen, Z.-U. Rahman, D. Krusienski, and J. Li, "A vision-based automatic safe landing-site detection system," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 49, no. 1, pp. 294–311, 2013.

[5] C. Sharp, O. Shakernia, and S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *ICRA*, vol. 2, 2001, pp. 1720–1727 vol.2.

[6] S. Saripalli, J. Montgomery, and G. Sukhatme, "Visually guided landing of an unmanned aerial vehicle," *ICRA*, vol. 19, no. 3, pp. 371–380, 2003.

[7] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi, "A vision-based guidance system for uav navigation and safe landing using natural landmarks," in *2nd International Symposium on UAVs*. Springer, 2010, pp. 233–257.

[8] J. Hays and A. A. Efros, "im2gps: estimating geographic information from a single image," in *CVPR*.

[9] A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE T-PAMI*, vol. 30, no. 11, pp. 1958–1970, Nov 2008.

[10] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *ECCV*, 2012. [Online]. Available: http://arxiv.org/abs/1205.3137

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1. IEEE, 2005, pp. 886–893.