Runway Detection and Localization in Aerial Images Using Deep Learning

Javeria Akbar[1], Muhammad Shahzad[1,2], Muhammad Imran Malik[1,2], Adnan Ul-Hasan[2], Fasial Shafait[1,2]

[1]School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST), Islamabad, Pakistan.
{jakbar.mscs15seecs, muhammad.shehzad, malik.imran, faisal.shafait}@seecs.edu.pk

[2]Deep Learning Lab (DLL), National Center of Artificial Intelligence (NCAI), Islamabad, Pakistan

Abstract— **Landing is the most difficult phase of the flight for any airborne platform. Due to lack of efficient systems, there have been numerous landing accidents resulting in the damage of onboard hardware. Vision based systems provides low cost solution to detect landing sites by providing rich textual information. To this end, this research focuses on accurate detection and localization of runways in aerial images with untidy terrains which would consequently help aerial platforms especially Unmanned Aerial Vehicles (commonly referred to as Drones) to detect landing targets (i.e., runways) to aid automatic landing. Most of the prior work regarding runway detection is based on simple image processing algorithms with lot of assumptions and constraints about precise position of runway in a particular image. First part of this research is to develop runway detection algorithm based on state-of-the-art deep learning architectures while the second part is runway localization using both deep learning and non-deep learning based methods. The proposed runway detection approach is two-stage modular where in the first stage the aerial image classification is achieved to find the existence of runway in that particular image. Later, in the second stage, the identified runways are localized using both conventional line detection algorithms and more recent deep learning models. The runway classification has been achieved with an accuracy of around 97% whereas the runways have been localized with mean Intersection-over-Union (IoU) score of 0.8.**

Keywords—runway, detection, localization, deep learning

I. Introduction

Unmanned Aerial Vehicles (UAVs) have become very popular during the last few years due to their use in tasks that are too dangerous to be performed by manned aerial vehicles. They have been used for various purposes other than military such as urban planning, inspection, monitoring, surveying, search and rescue, precision agriculture and many more using a variety of onboard sensors including optical images, laser scanners or even synthetic aperture radars [1][2].

For successful operation of UAVs without hardware damages, a safe landing operation is essential. To achieve this, vision based approaches for UAV landing have been an active topic in research particularly owing to their ability to provide rich textual information at relatively much less relative cost making them much more suitable for automatic landing problem compared to other sensors. These vision-based approaches can be integrated with traditional control techniques for a robust landing approach.

Runway detection can be defined as a method of identifying runway in an image. Localization means to find the exact location of runway in the image. Runway detection and localization is an important task for UAVs as they can use it for landing as well as self-localizing and navigation. Rotary wing UAVs are simple to land as they can hover and land vertically. But fixed wing UAVs require runway to land. Vision based UAV navigation is typically controlled by processing images taken from onboard cameras. The classified and detected runways in those images allow to extract additional information from them such as relative position and orientation, which in turn can be used to align fixed-wing UAVs to runway and hence guiding it in landing.

Vision based approach for fixed-wing UAV landing includes vision-based step for runway detection, alignment of UAV to runway and a controller to guide UAV accordingly. This research only focuses on the first step that is the runway detection using single onboard camera while preparing for landing.

II. Literature Review

This section presents different approaches previously being used for runway detection for UAV landing and for other purposes such as urban planning etc. Majority of these approaches are based on template matching, Hough transform, Active Contours and Machine Learning algorithms. Broadly, these approaches can be categorized into following two main categories: template based and feature based.

A. Template Based Approaches

Template based approaches use a model of the object to be detected in the query image. This model is stored in an image called a template. This template is compared with the query image on pixel by pixel basis to find matches. Template based methods are not too common and mostly they are used alongside other feature based techniques. In [3], delta correlation has been used for matching process in proposed template based runway recognition method. In [4], EVS and SVS have been used to generate templates and chamfer matching has been used for matching process.

B. Feature Based Approaches

These approaches use features of runway like intensity edges, high contrast corners, texture primitives and other similar image components. Feature based approaches can be further divided into two more categories; one that uses geometric shape of the runway for detection, and the other that uses machine learning based approaches.

1) Geometric Shape Based Approaches: In [5], they have proposed a method which combines segmentation and minimization of an energy function. In [6], they have employed sobel operators to get edges and used heuristic search to extract lines. Runways edges have been determined

978-1-7281-3857-2/19/$31.00 ©2019 IEEE

based on parallel constraint. In [7], the authors used canny edge detection with morphology opening to detect runway in Aviation Reconnaissance Images. In [7], the runways using primitive sobel and robert edge detectors have been used for detection. In [8], they have used hough transform after broadening runway axis based on certain criteria to extract runway edges in remote sensing images. In **Error! Reference source not found.**, they have proposed a sensor fusion algorithm based on hough transform and discrete wavelet transform to detect runways. In [9], they have proposed a combination of ICCBET and hough transform to localize two parallel lines of runway.

2) Machine Learning Based Aproaches: In [12], they have used a SVM based classifier to detect runway in ROCs formed by grouping of SIFT key points in IKONOS images. In [13], they have proposed a texture based method which uses adaboost algorithm with features such as Fourier power spectrum, Gabor filters etc. to detect runway. In [14], they have used hough transform and graph based saliency model to find ROIs based on runway existence and then have used HDR to classify extracted SIFT features to detect airport region. In [15], they have used an improved K-means clustering algorithm to classify potential straights found by rotating projection algorithm to extract runway edges. In [16], they have proposed a combination of SVM classifier and parallel line constraint to discriminate runways. In [17], they have used Bayesian classifier to classify runway pixels in POLSAR image. Real runway has been found by using morphology filtering and topological properties. In [18], the authors proposed airport detection method using deep end-to-end CNN architecture with hard example mining. In [19], they have proposed a runway detection method based on LBP (Local Binary Pattern) Cascade Classifier. In [20], they have proposed a runway detection method based on a multi-channel pulse coupled neural network (MPCNN).

Some research has been based on deep learning. Here we mention two such papers. In [21], they proposed an airport detection based on Faster R-CNN. First, a CNN has been used to identify potential airport regions. Then another CNN has been used to detect airport by using some improvements based on runway features. In [22], they have used LSD to find potential airport regions based on runways. Then they have used AlexNet to classify these regions as airport.

It can be inferred from above discussion that majority of the runway detection methods are not based on machine learning. And those that are based on machine learning are mostly airport detection methods. This calls for more research on machine learning based techniques for runway detection. It has been known that machine learning based methods can give more accurate results. Usually, they are avoided in real time object detection methods, but with the recent advancement in hardware, now it is possible to even use deep learning for real time object detection.

III. Methodology

In machine learning domain, it has been established that CNNs give us more accurate feature representation as compared to other methods. For runway detection, first land has been classified to know whether there is runway in the image or not using CNNs. After confirming that runway is present in the image, it has been localized using both CNN based methods and non-CNN based methods. This research can be divided into two modules; runway detection and runway localization.

A. Runway Detection (Land Classification)

Image classification is the most common task in computer vision. In image classification, algorithm processes image and classifies the object present in the image. For land classification, an image is processed to identify the area represented by that image. Land consists of multiple areas like runways, roads, forests, buildings, seas, mountains, deserts and many more. This classification is performed to find whether a runway exists in the image or not. In this module, CNN models have been used for classification purpose.

1) Dataset: For binary classification, land areas other than runway like roads, forests, mountains, deserts etc. would have been treated as one class which would have been a less suitable approach. So, a remote sensing dataset [23] with multiple classes has been used for this purpose. This dataset is the largest available dataset, which is variant enough to apply CNN models and consists of satellite images downloaded from google earth, collected by experts. There are 45 classes with each class containing 700 images.

2) Feature Extraction: CNN classification models VGG16 [25], Resnet50, Resnet152 [26] and Densenet161 [27] trained on ImageNet dataset [24] have been used to extract features from images. For each classification model, input images have been resized to 224x224 and the only preprocessing performed is mean normalization. Keras models with backend as TensorFlow have been used for feature extraction.

a) VGGNet: For VGGNet, pre-trained VGG-16 model has been taken and last FC layer has been removed to extract 4096- dimensional feature set from images.

b) ResNet: Two models of Resnet have been used; Resnet50 and Resnet152. For both models, classification layer has been removed to extract 2048-dimensional feature set from images.

c) DenseNet: For densenet, Densenet-161 model has been taken and classification layer has been removed to extract 2208-dimensional feature set from images.

3) Classifier & Training: After extracting features, a softmax classifier has been trained on these features. This classifier has been implemented using TensorFlow. It works as follows: Weight matrix is initialized using random values based on normal distribution and biases are initialized to zero. Inputs (extracted features) are multiplied with weight matrices and biases are added. Training labels are first converted into one hot encoding sequence (representation of labels as binary vectors). Then loss is calculated by computing cross-entropy and taking average of it across all training examples. The minimum loss is found using gradient descent optimizer. Three classifiers for each model have been trained with different training and testing data and their mean accuracy has been reported in results section for different training ratios. Cross entropy has the form:

$$L_i = -\log\left(\frac{e^{fyi}}{\sum_j e^{fj}}\right) \quad (1)$$

Where fj is the j[th] element of the class scores vector f. The full loss of dataset is average of L_i across all training examples. Table I shows parameters used for training.

TABLE I. PARAMETERS USER FOR TRAINING

Parameters	*Values*
Learning Rate	0.05
Regularization Parameter	0.01
Batch Size	128
Number of Steps	10000

4) Fine-tuning: As the selected dataset is like ImageNet dataset so finetuning the classification model could improve results. Only the Model with the best performance from the above-mentioned models has been finetuned. ResNet50 has been selected as the best model based on results presented in Results section. Keras based implementation of Resnet50 trained on ImageNet has been used for finetuning. Softmax classifier of Resnet model has been replaced with a customized softmax classifier. For finetuning, input images have been resized into 224x224 and have been randomly divided into 80% training data, 10% validation and 10% test data. Parameters have been finetuned manually based on validation accuracy. Table II shows parameters used for finetuning.

TABLE II. PARAMETERS USED FOR FINETUNING RESNET50

Parameters	*Values*
Learning Rate	0.001
Momentum	0.9
Batch Size	16
Number of Epochs	50

B. Runway Localization

Runway detection is used for only finding out that whether runway exists in the image or not. To find the exact location of the runway in the image, the runway has been localized using both line detection algorithms and deep learning CNN models. Same dataset which has been used for classification purpose has been used here for localization purpose. But here only one class of that dataset that is runway is used. The purpose is to localize only runway in the image.

1) Line Detection Techniques: As the runway structure is composed of straight lines, line detection algorithms can be employed to localize the runway in the image. Simple Hough transform, probabilistic hough transform and LSD [28] based approaches have been used for localizing runway in this section.

a) Hough Transform (HT): In this approach, firstly runway images from the selected dataset have been converted into grayscale images. It can be clearly seen from Figure 1 that gray values of runway area are much different than those of background. Then Canny edge detection has been applied to extract edges. Canny algorithm with hysteresis threshold ratio of 1:3 has been used for detecting edges in the image. Then hough transform has been applied on edge image with ρ accuracy of 1 and different θ accuracy and threshold parameters. It returns lines in parametric form (ρ, θ). This parametric form has been used to get endpoints of returned lines as follows:

- $x_1 = x_0 + n * (-b)$
- $y_1 = y_0 + n * (a)$
- $x_2 = x_0 - n * (-b)$
- $y_2 = y_0 - n * (a)$

where $a = cos\theta$, $b = sin\theta$ and $x_0 = a * rho$, $y_0 = b * rho$ and n is the number of rows in the image. For each line detected in the image, its distance and angle is compared to every other line detected in the image. The distance $distance_{(i,j)}$ between two lines is calculated as follows:

$$\frac{abs[(y_{j2}-y_{j1})*x_{i1}-(x_{j2}-x_{j1})*y_{i1}+x_{j2}*y_{j1}-y_{j2}*x_{j1}]}{\sqrt{(y_{j2}-y_{j1})^2+(x_{j2}-x_{j1})^2}} \quad (2)$$

The angle of a line i to horizontal axis is calculated as follows:

$$angle_i = \tan^{-1}\frac{y_{i2}-y_{i1}}{x_{i2}-x_{i1}} \quad (3)$$

To find the correct values for angles, their sign is checked. If there is a minus sign, the value of angle is added to 180 degrees so that angles of two lines can be correctly compared. Now, the final two lines to identify runway are chosen based on two conditions:

I. $distance_{(i,j)} > \max distance$

II. $abs(angle_{(i)} - angle_{(j)}) < threshold$

Secondly, probabilistic Hough transform has been applied to see that if there is any improvement in results. It takes two extra parameters alongside with image, ρ accuracy, θ accuracy and threshold. First is the minimum length of a line, lines shorter than this length are rejected and the second one is the maximum gap allowed between two lines to consider them as one. After detecting the lines, same above mentioned procedure has been used to identify two boundaries of runway.

b) Line Segment Detector (LSD): In this approach, firstly runway images from the selected dataset have been converted into grayscale images. After detecting all line segments in the image, their lengths have been calculated. It is known that runways are elongated structures and they have long boundaries, so some threshold has been set and lines are filtered based on their length. That is, those lines are selected which have length greater than some threshold. LSD algorithm returns the two endpoints (x_1, y_1), (x_2, y_2) of a line segment so the length of a line segment cab be calculated as follows:

$$length = \sqrt{(x1-x2)^2 + (y1-y2)^2} \quad (4)$$

Finally, the constraints (I, II) have been used to find the final two lines representing runway boundaries.

2) CNN: The objective is to localize the runway such that the runway can be extracted with its boundaries. Bounding boxes give us subset of the image which includes the required object. We still cannot extract the exact the object as it is. To extract the object with its boundaries, a segmentation algorithm is needed. In segmentation, each pixel is assigned to a class. For each pixel, it is decided that whether it belongs to a particular class or not, so it can also be called as pixel level classification. A pixel wise mask for the required object is generated and the model learns by finding the difference

between the predicted and the ground truth mask. First bounding boxes can be found to reduce the image area on which pixel wise classification is to be applied. The model used for this purpose is Mask R-CNN [29]. Keras based implementation of Mask R-CNN pre-trained on COCO dataset [30] has been used as a segemntation algorithm. Its backbone architecture is Resnet50. As it has been seen in land classification that Resnet50 learned runway features very well compared to other models so it is a suitable choice for this purpose. This pre-trained model has been finetuned on selected dataset and custom-made dataset.

a) Dataset: 700 images of class runway from the selected dataset have been labeled using LabelMe[1]. Figure 1 shows some samples of runway masks. Runways have been labeled by drawing closed polygons around the runway. Instead of labeling the whole runway including curved areas, only that part of the runway has been labeled which is used for landing. This part of runway is covered with white lanes. Along with other runway features, these white lanes are the major features which model learns.

b) Experiments: Experiments have been conducted using train, validation and test sets. Following ratio have been observed between these sets. Train: Validation: Test = 70: 10: 20. Table III shows parameters used for finetuning mask segmentation model. Experiments have also been conducted on self-made customized dataset using train and validation sets. This has been done to indentify that how model behaves on runway images which have been taken from more height as compared to selected dataset. Some 100 images downloaded from google earth have been taken and have been split into 16 images each to form a dataset similar to the selected dataset. The customized dataset has been divided into train and validation sets. Out of 457 images, 381 have been used for training and 76 have been used for validation purposes.

Fig. 1. Some samples of runway masks

TABLE III. PARAMETERS USED FOR TRAINING

Parameters	***Values***
Learning Rate	0.0001
Momentum	0.9
Decay	0.0001
Batch Size	1
Number of Epochs	10

IV. RESULTS AND DISCUSSION

A. Land Classification

As each class in the dataset has equal number of images so accuracy can be used to measure the correctness of the model. With unequal images in each class, accuracy is biased but in this case, when there are equal number of images in each class, accuracy will not be biased. Accuracy is calculated simply by counting the number of instances where predicted class is same as true class and dividing it by the total number of samples in the dataset. Accuracy is calculated as follows:

$$\frac{sum(predicted\ label == true\ label)}{total\ number\ of\ samples} \times 100 \qquad (5)$$

1) Feature Extraction: Graph in Figure 2 shows comparison of four CNN models used for extracting features from images. It can be seen that increasing training ratio has resulted in increased accuracy. According to graph, Resnet50 and Resnet152 have almost same performance based on their accuracies. But as shown in Table IV Resnet50 is faster so it has been chosen for finetuning. Table V shows average time of image read operation and average time of classifying extracted features for resnet architectures.

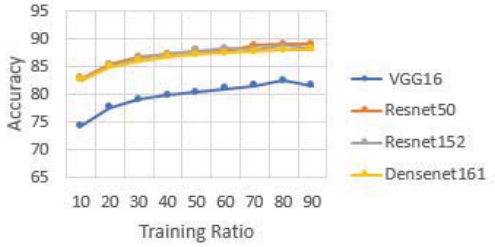

Fig. 2. Comparison of CNN models used for feature extractionmparison of processing time

	Feature Extraction	
	cpu (sec/image)	***gpu(sec/image)***
VGG16	0.56	0.024
Resnet50	0.27	0.028
Resnet152	0.76	0.056
Densenet161	0.75	0.078

TABLE IV. PROCESSING TIME OF IMAGE READ AND RESNET

Average time of image read operation	***gpu (sec/image)***	***cpu (sec/image)***
	0.75	0.52
Average time of classifying extracted deep features	***Resnet50***	***Resnet152***
	0.038	0.038

2) Finetuning: CNN model Resnet50 has been finetuned on selected dataset. Model has been initialized with pretrained weights of ImageNet dataset. Validation accuracy on training ratio of 80% has been reported to be 97.33% and test accuracy on training ratio of 80% has been reported to be

[1] LabelMe, http://labelme2.csail.mit.edu/Release3.0/index.php

96.63%. To evaluate the model on target class runway, precision and recall has been calculated for class runway. For training ratio of 80%, precision and recall has been calculated as 94.44 % and 97.14% respectively. This means that capability of model to correctly classify a runway image as runway is a little more as compared to capability of a model to not classify a non-runway image as runway. For comparison with previous research, model has been finetuned with 10% and 20% training ratio too. Table VI compares these results.

TABLE V. COMPARISON WITH PREVIOUS RESULTS

	Model used	*Without fine-tuning*		*With fine-tuning*	
		10%	*20%*	*10%*	*20%*
Existing results	VGG16	76.47± 0.18	79.79± 0.15	87.15± 0.45	90.36± 0.18
Proposed Approach	Resnet50	82.80± 0.20	85.33± 0.06	88.47	90.6

3) Customized Dataset Results: Here, only one class runway has been considered. Every other class is treated as a negative sample for runway. Evaluation metrics accuracy, precision and recall has been used for evaluating resnet50 on customized dataset. As there are equal number of positive and negative samples, so accuracy will be unbiased. Accuracy of resnet50 without finetuning on customized dataset for class runway has been found to be 88.88%. The model correctly predicted the runway class with a precision of 86.36% and recall of 92.34%. Difference between the recall and precision tells us that the model has a better true positive rate for class runway. Accuracy of finetuned resnet50 on customized dataset for class runway has been found to be 90.73%. The finetuned model correctly predicted the runway class with a precision of 89.03% and recall of 92.89%. With finetuning, accuracy of runway class has increased by almost 2%.

B. Runway Localization

1) Hough Transform: For evaluation purpose, 460 images with different properties have been selected from 700 runway images of selected dataset. Whether runway has been successfully localized or not, it has been evaluated by inspection. Runway is considered successfully localized if two detected lines are almost same as the real two boundaries of the runway. All such images have been manually counted, and accuracy has been reported. Table VII shows accuracy results for simple hough transform based approach and Table VIII shows accuracy results for probablistic hough transform based approach. Figures 3 and 4 shows stepwise results of HT based approach and PHT based approach respectively.

TABLE VI. ACCURACY OF HT BASED APPROACH

ρ	θ	*vote threshold*	*Accuracy*
1	$\pi/150$	100	74.13%
1	$\pi/180$	100	70%

TABLE VII. ACCURACY OF PHT BASED APPROACH

ρ	θ	*vote threshold*	*min length*	*max gap*	*Accuracy*
1	$\pi/150$	100	100	10	70.65%
1	$\pi/180$	100	100	90	74.50%

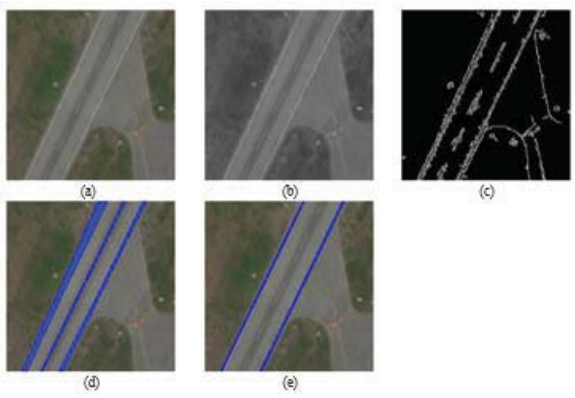

Fig. 3. Stepwise results of HT based approach (a) Original image (b) Grayscale image (c) Result of canny edge detection (d) Result of applying HT (e) Result of applying constraints I, II

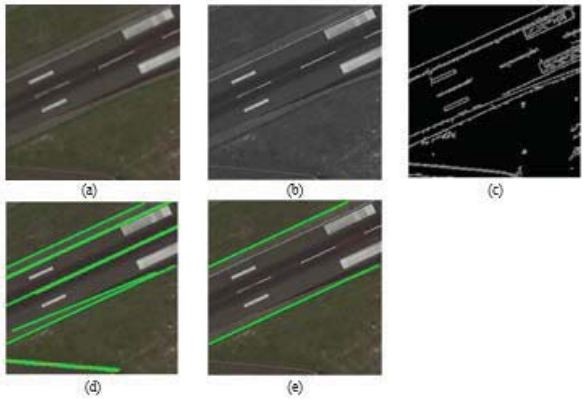

Fig. 4. Stepwise results of PHT based approach (a) Original image (b) Grayscale image (c) Result of canny edge detection (d) Result of applying PHT (e) Result of applying constraints I, II

2) Line Segment Detector: OpenCV based implementation of LSD has been used with default parameterization as it showed satisfying results except for number of bins. Number of bins have been selected based on the dataset used. Same set of images have been used as in above method and runway has been correctly localized in almost 76.5% of the total images. Figure 5 shows stepwsie results of LSD based approach.

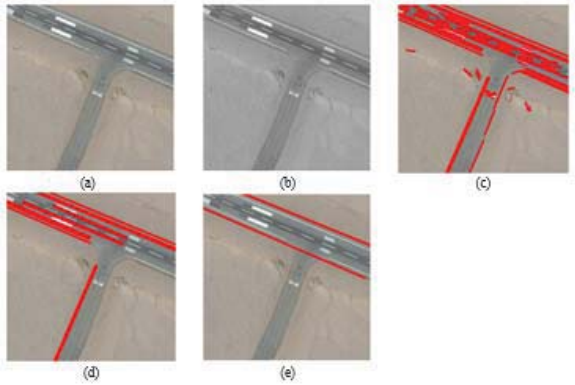

Fig. 5. Stepwise results of LSD based approach (a) Original image (b) Grayscale image (c) Result of applying LSD (d) Result of applying length constraint (e) Result of applying constraints I, II

3) CNN: Both selected dataset and novel customized dataset has been used for experiments. In both cases, weights

have been initialized with pre-trained weights of COCO dataset for finetuning. Parameters used have been finetuned manually based on evaluation metrics. Evaluation metrics used for evaluating the model are; IOU, Pixel wise evaluation and average precision.

a) Intersection over union (IOU): As the name implies, IOU is a fraction with a numerator which gets the area of overlap between the predicted and ground truth mask and denominator gets the area of union of both predicted and ground truth masks. In mathematical form, IOU can be depicted as:

$$IOU = \frac{\sum m_i^g . m_i^p}{\sum m_i^g + m_i^p} \quad (6)$$

b) Pixel wise evaluation: For each image, pixelwise accuracy, precision and recall calculated. For these metrics, binary classification is considered that is whether a pixel belongs to runway (class 1) or background (class 0).

*c) Average Precision: IOU is compared with thresholds in range of 0.5 to 1.*0 with standard deviation of 0.05. Based on this comparison, for each threshold, precision is calculated. Then average precision is calculated over all thresholds for a single image.

d) Selected Dataset Results: As seen from Figure 6, model has been successful in its ability to accurately detect runways masks for test dataset. For training purpose, masks generated for runways mostly included straight parts of runway. This property is depicted in the last image of above figure where only that part of the runway is detected which has been marked as runway in the ground truth image. To evaluate these experiments, all three evaluation metrics discussed above have been used. Intersection over union has been calculated using OR and AND operations on images. Mean IOU (masks) for validation set is found to be 0.80 and Average IOU (masks) for test set is found to be 0.76. For each image in val/test dataset, pixelwise accuracy, precision and recall has been calculated. For these metrics, binary classification is considered that is whether a pixel belongs to runway (class 1) or background (class 0). Mean pixelwise accuracy, precision and recall over whole val/test dataset is reported in Table IX. There are two classes runway and background. For each image, IOU has been calculated. Then iou is compared with thresholds in range of 0.5 to 1.0 with standard deviation of 0.05. Based on this comparison, for each threshold, precision is calculated. Then average precision is calculated over all thresholds for a single image. Finally mean average precision is calculated over all images. Table X shows mean average precision for different thresholds.

TABLE VIII. PIXEL WISE EVALUATION

	Validation	***Test***
Mean pixelwise accuracy	0.93	0.88
Mean pixelwise precision	0.90	0.82
Mean pixelwise recall	0.84	0.79

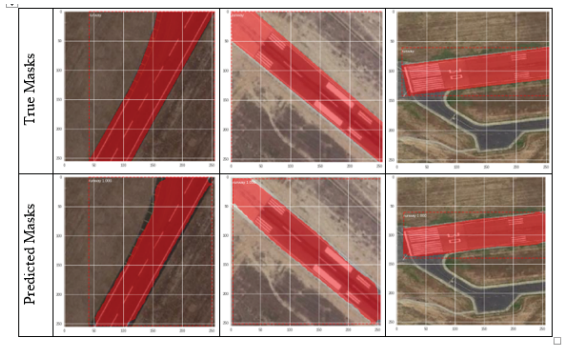

Fig. 6. Mask R-CNN results on selected dataset. Above row shows true masks and lower row shows predicted masks.

TABLE IX. MEAN AVERAVE PRECISION

Threshold	***mAP***
0.5 – 0.6	0.94
0.6 – 0.7	0.90
0.7 – 0.8	0.85
0.8 – 0.9	0.75
0.9 – 1.0	0.37

e) Customized Dataset Results: The customized dataset is different from the selected dataset in the sense that it has narrower runways. Runway images of selected dataset have mostly broader runways. This dataset has been used to test that how model behaves when there are narrow runways in images. As seen from Figure 7, model is able to correctly detect these narrow runways. Intersection over union for validation set is found to be 0.73. Below tables shows the pixel wise accuracy, precision and recall for validation set. Mean average precision for threshold of range (0.5 to 1.0) is found to be 0.75. Mean test execution time for predicting masks is found to be 0.26 s per image.

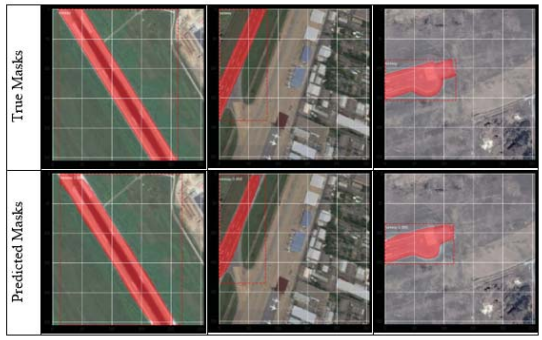

Fig. 7. Mask R-CNN results on customized dataset.. Above row shows true masks and lower row shows predicted masks.

V. CONCLUSION

This paper presents a method to perform the runway detection using aerial images acquired from onboard vision sensor. The work presented in this paper is the initial step in UAV landing that includes the detection and localization of runways.

This research has been conducted to find an accurate runway detection model. Previous research has been mostly based on non-machine learning based methods with lot of assumptions about position of runway in the image. Use of deep learning in runway detection allows to detect runways without explicitly extract hand crafted features. The proposed

runway detection model has been validated on two datasets including a custom-made runway detection dataset and a public remote sensing dataset for aerial image classification which shows that this model can detect any shape of runway with the appropriate training data.

This research includes two modules, first land has been classified to find out if there exists a runway or not, then runway detection model has been applied to extract runway from image. Combination of these two approaches increase accuracy. With right hardware, this can be implemented in landing of UAVs. After successful extraction of runway, this extracted runway can be used to align UAV with the runway. The proposed runway detection model achieved reasonable IOU of 0.8 which validates its efficacy.

REFERENCES

[1] M. Schmitt, M. Shahzad, X. X. Zhu, "Reconstruction of Individual Trees from Multi-Aspect TomoSAR Data", *Remote Sensing of Environment*, vol. 165, pp. 175-185, 2015.

[2] M. Shahzad, M. Schmitt, X. X. Zhu, "Segmentation and crown parameter extraction of individual trees in an airborne TomoSAR point cloud", *Proceedings of the PIA15+HRIGI15 – Joint ISPRS conference, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XL-3/W2, 2015, Munich, Germany.

[3] D. Meng, C. Y.-feng, G. Lin, "A method to recognize and track runway in the image sequences based on template matching," *2006 1st International Symposium on Systems and Control in Aerospace and Astronautics*, Harbin, 2006, 4 pp.-1221.

[4] J. Wang, Y. Cheng, J. Xie, W. Niu, "A Real-Time Sensor Guided Runway Detection Method for Forward-Looking Aerial Images." *2015 11th International Conference on Computational Intelligence and Security (CIS)*, 2015, pp. 150-153.

[5] K. A. Jbara, W. Alheadary, G. Sundaramorthi, C. Claudel, "A robust vision-based runway detection and tracking algorithm for automatic UAV landing." *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, Denver, CO, 2015, pp. 1148-1157.

[6] Y. Dong, B. Yuan, H. Wang, Z. Shi, "A runway recognition algorithm based on heuristic line extraction," *2011 International Conference on Image Analysis and Signal Processing*, Hubei, 2011, pp. 292-296.

[7] Z. Fengjing, S. Wenbang, G. Yongsheng, "Airport runway extraction method in aviation reconnaissance image," *2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, Toronto, ON, 2013, pp. 152-154.

[8] W. Yang, Z. Li, Z. Shen, "Recognizing and tracking airport runway target in infrared images," *Proceedings of the IEEE 1997 National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, USA, 1997, pp. 1045-1051.

[9] P. T. G. Jackson, C. J. Nelson, J. Schiefele, B. Obara, "Runway detection in High Resolution remote sensing data," *2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA)*, Zagreb, 2015, pp. 170-175.

[10] A. F. Fadhil, R. Kanneganti, L. Gupta, R. Vaidyanathan, Fusion of Enhanced and Synthetic Vision System Images for Runway and Horizon Detection, Sensors (Basel), 19(17), 2019.

[11] N. Di, M. Zhu, Y. Wang, "Real time method for airport runway detection in aerial images," *2008 International Conference on Audio, Language and Image Processing*, Shanghai, 2008, pp. 563-567.

[12] C. Tao, Y. Tan, H. Cai and J. Tian, "Airport Detection from Large IKONOS Images Using Clustered SIFT Keypoints and Region Information," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 1, pp. 128-132, Jan. 2011.

[13] Ö. Aytekin, U. Zöngür and U. Halici, "Texture-Based Airport Runway Detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 3, pp. 471-475, May 2013.

[14] X. Wang, Q. Lv, B. Wang, L. Zhang, "Airport detection in remote sensing images: A method based on saliency map", *Cognitive Neurodynamics*, 7(2), 2013.

[15] Z. Guan, L. Jie, Y. Huan. "Runway Extraction Method Based on Rotating Projection for UAV" *Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation*. Atlantis Press, 2016.

[16] Y. Qu, C. Li, N. Zheng, "Airport Detection Base on Support Vector Machine from A Single Image," *2005 5th International Conference on Information Communications & Signal Processing*, Bangkok, 2005, pp. 546-549.

[17] P. Han, L. Chang, Q. Shi, J. Qu, "Runways detection based on scattering similarity and structural characteristics", *2015 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, Herdon, VA, 2015, pp. H2-1-H2-8.

[18] B. Cai, Z. Jiang, H. Zhang, D. Zhao, Y. Yao, "Airport Detection Using End-to-End Convolutional Neural Network with Hard Example Mining", *Remote Sensing*, 9, 1198, 2017.

[19] J. Cruz, E. Shiguemori and L. Guimaraes, "Concrete and Asphalt Runway Detection in High Resolution Images Using LBP Cascade Classifier," *2013 BRICS Congress on Computational Intelligence & 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC)*, Ipojuca, 2013 pp. 465-469.

[20] H. Zhuang, K. S. Low, "Real time runway detection in satellite images using multi-channel PCNN," *2014 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou, 2014, pp. 253-257.

[21] F. Chen, R. Ren, T. V. Voorde, W. Xu, G. Zhou, Y. Zhou, "Fast Automatic Airport Detection in Remote Sensing Images Using Convolutional Neural Networks", *Remote Sensing*, 10(3), 2018.

[22] P. Zhang, X. Niu, Y. Dou and F. Xia, "Airport Detection on Optical Satellite Images Using Deep Convolutional Neural Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1183-1187, Aug. 2017.

[23] G. Cheng, J. Han, X. Lu. "Remote sensing image scene classification: Benchmark and state of the art." *Proceedings of the IEEE*, vol. 105, issue 10, 2017.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge", *Inyernational Journal of Computer Vision,* vol. 115, issue 3, pp 211-252, 2015.

[25] S. Karen, and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[26] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition" *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[27] G. Huang, Z. Liu, L. von der Maaten, K. Q. Weinberger, "Densely connected convolutional networks" *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708.

[28] R. Grompone von Gioi, J. Jakubowicz, J. Morel, G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722-732, April 2010.

[29] K. He, G. Gkioxari, P. Dollar, R. Girshick, "Mask R-CNN", *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961-2967.

[30] T-Y. Lin et al. "Microsoft COCO: Common objects in context." *Proceedings of the European Conference on Computer Vision*, Springer, Cham, 2014, pp 740-755.