Geospatial Object Detection using Machine Learning-Aviation Case Study

Abstract— **This paper presents the application of computer vision and machine learning to autonomous approach and landing and taxiing for an air vehicle. Recently, there has been growing interest in developing unmanned aircraft systems (UAS). We present a system and method that uses pattern recognition which aids the landing of a UAS and enhances the manned air vehicle landing. Auto-landing systems based on the Instrument Landing System (ILS) have already proven their importance through decades. The auto-land systems work in conjunction with a radio altimeter, ILS, MLS, or GNSS. Closer to the runway, both under VFR and IFR, pilots are expected to rely on visual references for landing. Modern systems like HUD or CVS allows a trained pilot to manually fly the aircraft using guidance cues from the flight guidance system. Notwithstanding the type of landing and instruments used, typically Pilots are expected to have the runway threshold markings, aiming point, displacement arrows, and touch down markings/lights insight before Minimum Decision Altitude (MDA). Imaging sensors are the basic common equipment in manned and unmanned aerial vehicles that are widely used during the landing maneuver. In this method, a dataset of visual objects from satellite images is subjected to training for pattern recognition. This trained system learns and then identifies and locates important visual references from imaging sensors and could help in decision making during landing and taxiing.**

Keywords—auto-land, air vehicle, UAV, UAS, Machine Learning, Computer Vision, landing, taxi, Runways

I. Introduction

Compared to other phases ,landing is the most demanding flight phase for both manned and unmanned aerial vehicles .It is also the most accident-prone stage for manned, remotely piloted and autonomous aircrafts. Auto-landing systems were introduced to reduce the workload on pilot during this critical phase. These systems based on the Instrument Landing System (ILS) have already proven their importance through decades. In an IFR auto-land systems were designed to make landing possible in visibility too poor to permit any form of visual landing, although they can be used at any level of visibility. They are usually used when visibility is less than 600 meters' runway visual range and/or in adverse weather conditions, although limitations do apply for most aircraft. The auto-land systems work in conjunction with radio altimeter, Inertial Navigation System (INS), ILS, MLS or Global Navigation Satellite System (GNSS). Closer to the runway, both under VFR or IFR, pilots are expected to rely on the visual references for landing. Modern systems like HUD or Combined Vision System (CVS) allows a trained pilot to manually fly the aircraft using guidance cues from the flight guidance system. This significantly reduces the cost of operating in very low visibility, and allows aircraft that are not equipped for automatic landings to make a manual landing safely at lower levels of look ahead visibility or runway visual range (RVR).

Notwithstanding the type of landing and instruments used, typically Pilots are expected to have the runway threshold markings, aiming point, displacement arrows , touch down point, helipad markings/lights in sight before Minimum Decision altitude (MDA).

Modern imaging sensors are integrated with displays to provide the pilots with better images than unaided human vision. With benefits provided by such systems like Enhanced Flight Vision Systems (EFVS) and Combined Vision System (CVS), there has been a growing interest in making use of vision as important apparatus for autonomous landing of UAS. Weiwei Kong et al [1] presents the main research groups involved in the development of vision-based autonomous landing systems. In these problems, different types of visual features were considered including geometric model of the target, points, corners of the runway, binormalized Plücker coordinates, the three parallel lines of the runway (at left, center and right sides) and the two parallel lines of the runway along with the horizon line and the vanishing point. Rather than points, lines or other features, researchers from IST/TU Lisbon [2] demonstrated the dense information could also be applied into vision-based landing system. Cesetti [3] et al. proposed a vision-based guide system for UAV navigation and safe landing using natural landmarks. This method might not be effective when the real-time image was much different from the reference image and its performance depends on matching algorithm.

II. OBJECTIVE

Object detection in machine learning makes use of Deep Learning for object classification, detection and instance segmentation. Deep learning is a subset of machine learning in Artificial Intelligence (AI) that has networks which are capable of learning by training on annotated data. Almost all the value today of deep learning is through supervised learning or learning from labeled data [Andrew Ng]. The success of an object detection depends on building representative labeled dataset; training; and evaluation of the model.

The objective of the current project is to build training dataset of annotated objects acquired from overhead perspective. It is useful for training a deep neural network to learn to detect, count specific airport objects.

This paper details the procedure and parameters used to create training dataset for running convolutional neural networks (CNNs) on a set of aerial images for efficient and automated object recognition.

This part of activity is precursor to the overall research objective of pattern detection for autonomous decision-making. The deep learning model created then identifies and locates important visual references from imaging sensors and could help in decision making during landing and taxiing.

III. Scope

This study limits the scope to provide an input for landing and taxiing system. Specifically, the study is restricted to below mentioned aspects:

- Creation of dataset from satellite images
- Apply *transfer learning* technique of Machine Learning on these images and create object detection inference model
- Verify if this model is suitable to detect airport objects from images acquired from imaging sensors

This study does not intend to cover:

- System of autonomous landing

- Aero dynamics of landing and mechanism of autonomous taxiing

IV. Data Used

QGIS-Open Layers plugin is used to access high resolution satellite and aerial imagery base map. Polygon boundaries of the objects of interest are generated from the vector data available at FAA website .Negative training set is created from the declassified New York state GIS data clearing house link .The dataset has the following attributes:

1) Data from overhead at about 1 meter per pixel resolution at ground.
2) Data from 1025 distinct locations in United States and Europe.
3) Intentional selection of hard negative examples.
4) Extra testing scenes for use after validation

V. Methodology

Transfer learning or inductive transfer is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

Faster R-CNN is an object detection framework based on deep convolutional networks, which includes a Region Proposal Network (RPN) and an Object Detection Network. Residual neural networks, or ResNets (Deep Residual Learning for Image Recognition), are a technique Microsoft introduced in 2015. The ResNet technique allows deeper neural networks to be effectively trained.

faster_rcnn_resnet101_pets is configuration that is trained on Oxford-IIIT Pet Dataset [4] for an algorithm to distinguish dogs from cats. We intend to use ***transfer learning*** method on this configuration to create training dataset for specific airport object detection.

VI. Study Area

We originally focus on the aviation domain, namely airport feature detection and classification. We view this problem as a compelling use case that also provides insights into semi-automatic feature extraction in sub meter resolution satellite imagery. Therefore, data from 1025 distinct locations in United States and Europe from over 400 airports is used in this study.

VII. Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

VIII. PREPARING GEOSPATIAL DATA FOR MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access data and use it learn for tasks like object classification, clustering, pattern recognition, localization, object detection and segmentation. The process of learning starts from providing relevant annotated data to ML algorithms, which find relationships, develop understanding, make decisions, and evaluate their confidence from the training data they're given. Such trained models have ability to generalize to previously unseen data. It is critical that we feed them the right data for the problem we want to solve. The success of any ML algorithm depends on the quality and quantity of training data. The process for providing data ready for object detection task using ML algorithms can be summarized as below sequence of steps:

A. Data Collection

This step is concerned with selecting the available data that we will be working with. For the purposes of this project, we could not get the dataset from public sources directly in image format. QGIS-Open Layers plugin is used to access high resolution satellite and aerial imagery base map. The data is free for academic purpose and research. Polygon boundaries of the specific objects of interest are generated from the vector data available at *Federal Aviation Administration - AIS Open Data* website. Additionally, several locations are annually digitized to achieve desired variability expected by ML algorithms.

For threshold classification purpose, we created a repository of two sets of 2595 images of 375X356 pixel at 1:500 and 1:1200 scales [Figure 1 & Figure 2].

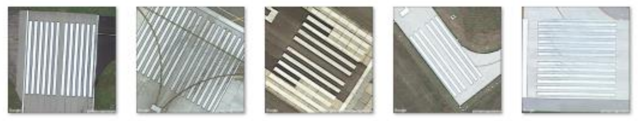
Figure 1 Snapshot of the training data at 1:500 scale (QGIS Open Layers)

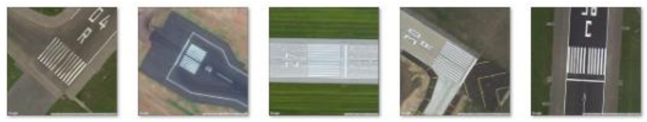
Figure 2 Snapshot of the training data at 1:1200 scale (QGIS Open Layers)

B. Data Preprocessing

This preprocessing step involves data cleaning, sampling, labeling, and splitting into training and test sets. If the data is collected from diverse source and in various formats, it is required to convert them into standard image format like JPEG or PNG depending on the ML library being used. In this case, we used RGB images in JPEG format. It is very likely that the machine learning tools we use on the data will influence the preprocessing we are required to perform. For example, Tensorflow Object Detection accepts labels for classification in PASCAL VOC XML format whereas Darknet's YOLO expects the same in CSV format.

Data cleaning requires removing some images that are not representative of selected pattern in terms of visual appearance though their geographical location is correct.

Sampling is the process of selecting representative subset of the available dataset. It involved visually inspecting each sample and weeding out objects that are too similar. The exercise is essential to avoid overfit of the training dataset. Our sampling process followed data augmentation guidelines prescribed to prevent overfitting . Thus, images that have good mixture of varied backgrounds, rotations, lighting, colors, textures, occlusions etc. are part of the sampled data [Figure 3].

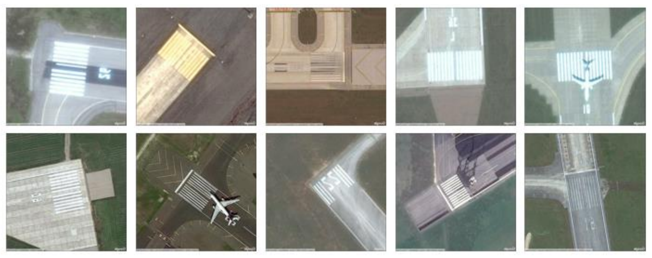
Figure 3 Snapshot of sample representativeness

Labeling is the process of annotating objects of interest for classification training. Since Object Detection not only classifies the object but detects the location of an object in the image, we need to pass it a bounding box identifying where the object is in the image and the label associated with that bounding box. To generate the bounding boxes for our images we used an annotation tool called LabelImg. LabelImg lets us hand label images and returns an xml file for each image with the bounding box and associated label. Annotation can be done for single class or multiple classes for which object detection is sought. In this study, as the objective is to detect a threshold, we used single class classification.

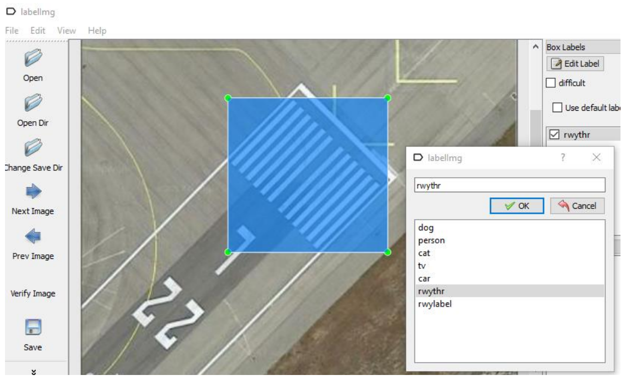

Figure 4 Annotation with LabelImg

To ensure robustness data is split into training and testing sets. Training data is used to train an algorithm. Generally, training data is a certain percentage of an overall dataset along with testing set. As a rule, the better the training data, the better the algorithm or classifier performs. Once a model is trained on a training set, it's usually evaluated on a test set to check if there is overtraining or model is generalized well.

To explore the transfer learning and ascertain the confidence in implementing the method for the airport object detection, we took a smaller representative sample of the selected data. Larger datasets often result in longer running times for algorithms, longer computational and memory requirements.

C. *Data Formatting*

In machine learning, we typically obtain the data and ensure that it is formatted before starting the training process. Tensorflow Object Detection API is used for this study, which expect the images and annotation in a binary format called TFRecord. All the images and annotations are converted into this format. Most of the batch operations aren't done directly from images, rather they are converted into a single tfrecord file (images which are Numpy arrays and labels which are a list of strings).

IX. Object Detection and Classification

Equipped with the data and format that is suitable for training using Tensorflow, the experiment for object detection and classification is carried out under two different data sample and hardware configurations. Phase 1 involved training and evaluating the results on a CPU. We used a GPU with a larger dataset in the Phase2 and observed remarkable increase in speed of training larger datasets.

A. *Phase1*

This phase is aimed at examining the suitability of the Faster-RCNN with faster_rcnn_resnet101_pets configuration for the data under study. The data split into training-80% and testing-20% datasets. Tensorflow returns object detection results at different accuracies, which can be configured by setting up a threshold Figure 5.

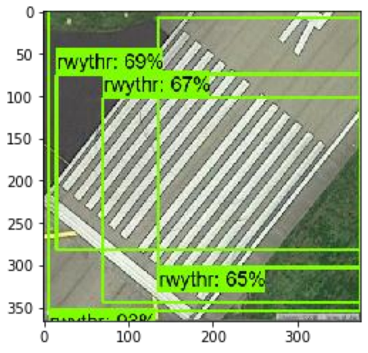

Figure 5 Object detection at different accuracies

To train 200 samples, it took more than 2.5 hours for training this data on a CPU. We noticed that for the given data, object detection is providing true positives in 60% of the cases when the confidence threshold is set at 85% and above-Figure 6.

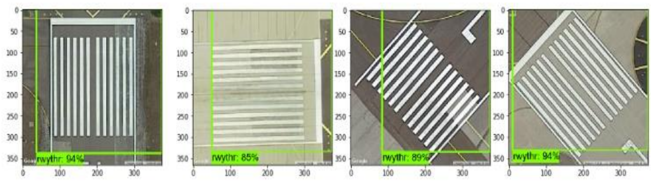

Figure 6 Successful object detection cases (CPU)

It is observed that although object detection is decent at a given scale, for images at coarser scales, the model is correct in detection but must improve in localization. Table *IX-1* summarizes the results obtained in Phase1 with hardware and software configuration details.

Table IX-1 Phase1 Configuration and Results

	SUMMARY OF PHASE1 CONFIGURATION AND RESULTS
System Configuration	E5470 - Intel HD Graphics 520. Core i5 6300U - 8 GB RAM - 256 GB SSD RAM, 2.4ghz, Windows 10 Enterprise 64 Bit
Model used	Transfer learning with Faster R-CNN ResNet
Training time	2.5 hours
Dataset Size	200 images
Train Test ratio	80% & 20%

	SUMMARY OF PHASE1 CONFIGURATION AND RESULTS
Iterations	200
Confidence threshold chosen	>85%
True Positives Observed	>60%
Failure	Localization requires improvement

B. *Phase2*

Phase 1 results support the suitability of the Faster R-CNN algorithm and indicated the processing time it takes to train on a CPU only system. Increase in size of the dataset, variability and number of iterations increase the object detection and localization. Among processing input data; training the deep learning model; storing the model; and deployment of the model, training is computationally more intensive task [H]. As training of more data requires enormous amount of processing power, Phase2 made use of GPU with higher number of data samples and training iterations.

The data set contained 1025 samples which are split into training-80% and test-20%. Training these samples on GPU has taken about 33 minutes for 2000 iterations. For five-fold increase in data and twenty-times increase in iterations, it is a significant increase when compared to processing on a CPU only system.

Increasing the sample size improved the result in terms of both object detection and localization. While localization failed in many cases in Phase1, we received correct result in Phase2.Object detection accuracies improved from 85% to 97% and above with least being 94%. We achieved, significantly higher number of samples that have accuracies between 97% and 99% Figure *7*.

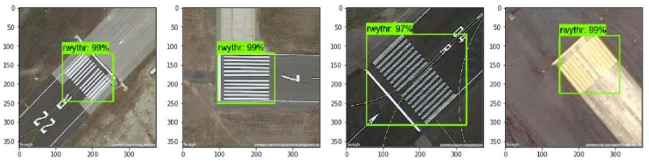

Figure 7 Improved accuracies with more samples (GPU)

With more samples and increasing the variety, we noticed accuracies above 95% even in the case of occlusion - Figure *8*.

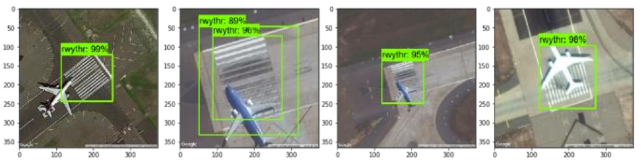

Figure 8 Object detection in occlusion

Similar level of true positives detected in case of highly unclear ambiguous images -Figure *9*.

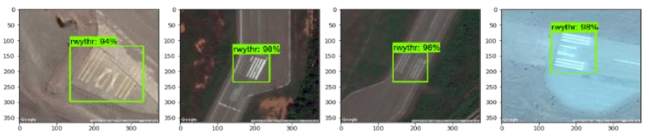

Figure 9 Object detection in ambiguous scenarios

TABLE *IX-2* summarizes the results obtained in Phase1 with hardware and software configuration details.

TABLE IX-2 PHASE2 CONFIGURATION AND RESULTS

	SUMMARY OF PHASE2 CONFIGURATION AND RESULTS
System Configuration	Dell Intel64 Family 6 Model ~2200 Mhz, Windows 10 Enterprise 64 Bit, Precision Tower 5810, Microsoft Windows 10 Enterprise, 64GB RAM
Model used	Transfer learning with Faster R-CNN ResNet
Training time	~33 minutes
Dataset Size	1025 images
Train Test ratio	80% & 20%
Iterations	2000
Confidence threshold chosen	>94%
True Positives Observed	>92%
Failure	Localization requires improvement

C. *Detecting objects captured from aviation imaging sensors*

The Faster-RCNN model that is retrained for this custom data is now applied on airport images acquired from imaging sensors for inference. Notice that these images are snapshots from publicly available aircraft landing videos, typically that are shot from onboard imaging sensors.

We selected several samples from landing videos that has runway threshold visible. Figure *10* illustrates the output of the inferences made by the model developed to detect this specific pattern. To test the model inference, we selected perspective images that have high variability in terms of pavement type, surroundings, distance of the pattern, visibility and wear and tear. While the bounding boxes are detected correctly, they are not tight as attained in Phase2.

As this model, can detect the trained pattern from perspective images, our hypothesis is validated.

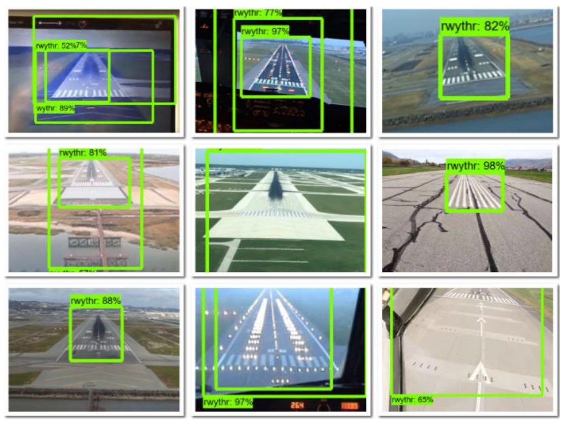

Figure 10 Object detection from aircraft landing/taxiing videos (YouTube)

FUTURE WORK

Encouraged with the results obtained, our future research intends to expand the study to include multiple object detection; Improve the model to obtain tight boundaries in the perspective images; Create a dataset of airport objects that comprises of images, annotations and masks.

ACKNOWLEDGMENT

We thank Gang He. Fellow, Honeywell Aerospace for assistance with his expert review and comments that greatly improved the manuscript.

REFERENCES

[1] Weiwei Kong , Dianle Zhou , Daibing Zhang and Jianwei Zhang, "Vision-based autonomous landing system for unmanned aerial vehicle: A survey",2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI).

[2] T. Goncalves, J. Azinheira, and P. Rives, "Vision-Based Automatic Approach and Landing of Fixed-Wing Aircraft Using a Dense Visual Tracking," , Informatics in Control Automation and Robotics, pp. 269-282..

[3] Cesetti, A., Frontoni, E., Mancini, A. et al. J Intell Robot Syst (2010) 57: 233. https://doi.org/10.1007/s10846-009-9373-3

[4] O. M. Parkhi, A. Vedaldi, A. Zisserman, C. V. Jawahar ;Cats and Dogs; IEEE Conference on Computer Vision and Pattern Recognition, 2012