Dissertation Professional presented to the Instituto Tecnológico da Aeronáutica, in partial fulfillment of the requirements for the degree of Master of Engineering of the Professional Master's Course in Aeronautical and Mechanical Engineering.

Victor André Lima

AIRPORT RUNWAY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

Dissertation Professional approved in its final version by signatories below:

Marcos Ricardo Omena de Albuquerque Maximo

Prof. Dr. Marcos Ricardo Omena de Albuquerque Maximo

Advisor

[illegible]

Prof. Dr. Ney Rafael Sêcco

Co-advisor

[illegible]

MSc. Eng. Ney Ricardo Moscati

Co-advisor

Prof^{a}. Dr^{a}. Emilia Villani

Dean of Graduate Studies

Campo Montenegro
São José dos Campos, SP - Brazil
2022

Cataloging-in Publication Data
Documentation and Information Division

Lima, Victor André
AIRPORT RUNWAY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS / Victor André Lima.
São José dos Campos, 2022.
69f.

Master's Dissertation – Professional Master's Course in Aeronautical Engineering. Area of Aeronautical Systems – Instituto Tecnológico de Aeronáutica, 2022. Advisor: Prof. Dr. Marcos Ricardo Omena de Albuquerque Maximo. Co-advisors: Prof. Dr. Ney Rafael Sêcco and MSc. Eng. Ney Ricardo Moscati.

1. Redes neurais. 2. Processamento de imagens. 3. Pistas (de pouso e decolagem). 4. Simulação computadorizada. 5. Banco de dados de imagens. 6. Segurança de voo. 7. Computação. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

LIMA, Victor André. **AIRPORT RUNWAY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**. 2022. 69f. Master's Dissertation in Aeronautical and Mechanical Engineering – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Victor André Lima
Rua Corinto, 512
05586-060 – São Paulo–SP

AIRPORT RUNWAY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

Victor André Lima

Acknowledgments

First, I thank God for all the strenght passed on to me, and all the resilience that is still to come. I thank the company and spirited cheers of my wife Thayna, to whom I am in eternal debt. I thank the hard work and support of my beloved mother, reason why I was today able to contribute with my research to the world. I thank the relentless advices of my father, who warns and guides me through the dangers of the world. I thank my co-advisor, Cap. Ney Secco, who accepted my invitation to embark in this periculous journey through computer vision. I thank my advisor, Dr. Marcos Maximo for the brilliant guidance through this mystical world of deep learning. I also thank all my classmates from Embraer's PEE28 for enduring together through the program, and I extend these thanks to my industrial co-advisor, Eng. Ney Moscati. Lastly, I thank the random variables of this universe. Who would know what would have happened otherwise?

"I have fought the good fight, I have finished the race, I have kept the faith. Now there is in store for me the crown of righteousness, which the Lord, the righteous Judge, will award to me on that day, and not only to me, but also to all who have longed for his appearing."

— 2 Timothy 4:7-9

Resumo

Os recentes avanços em poder computacional permitiram ao mundo o uso mais extensivo de algoritmos de aprendizado profundo. O processamento de imagens em si é uma tarefa que exige grandes quantidades de dados, mas se demonstrou de extrema importância em muitos campos de conhecimento. Neste trabalho, um método de detecção de pista de pouso para aeronaves durante a fase de pouso, baseado em redes neurais convolucionais, é proposto com o objetivo de reduzir a carga de trabalho do piloto, aumentar segurança de voo durante pouso, e dar novos passos em direção a voos comerciais de piloto único ou até autônomos. Os resultados mostram que a rede neural pôde convergir suas funções de perda nos dados de treino e de validação com sucesso, e a inferência em imagens dos dados de teste e imagens reais produziu resultados satisfatórios para *keypoints* e *bounding box*. Os resultados numéricos indicam que imagens virtuais de pistas de pouso podem ser utilizadas para pré-treinar uma rede a fim de detectar pistas de pouso em imagens reais, então o trabalho contribui para uma abordagem mais barato e rápida para desenvolvimento de tal tecnologia. Análises nas estimativas para pose da câmera obtidas pelos *keypoints* inferidos indicam que o projeto é promissor sendo que, mesmo sendo um trabalho inicial, os resultados proveram histogramas de erros que podem facilmente ser controlados pelo uso de filtros para remover *outliers* e pela fusão de dados com outros sensores em aplicações reais. Em suma, contribui-se com resultados indicativos de que o treinamento virtual, com uso de técnicas de aumento de dados para enriquecimento do banco de imagens, beneficia a detecção em imagens reais e que os *keypoints* inferidos podem ser usados em conjunto a algoritmos para estimar pose a fim de estimar a pose de aeronaves em relação à pista de pouso.

Abstract

The recent advancement in computational power allowed the world to give more use of deep learning algorithms. The image processing task is on itself data-hungry, but showed to be of extreme importance in many fields of knowledge. In this work, a runway detection method, for aircraft during landing phase, based on convolutional neural network is made with the motivation to reduce pilot workload, increase flight safety during landing and take new steps towards single-piloted or even unmanned commercial flights. With the results, the network could successfully converge its loss function on the training dataset and validation dataset, and inference on several images from test dataset and real runway images gave good results for keypoint and bounding boxes predictions. Our numerical results indicate that virtual runway images can be used in order to pretrain a network to detect real runway images, so we contribute to a cheaper and faster approach on the development of such technology. Further analysis on estimatives for orientation and position of the aircraft camera from predicted keypoint indicate the work to be promising. Even being an initial work, the results gave a robust error histogram which could be easily controlled by filtering outliers and by data fusion with other sensors in real application. Overall, we contribute with results indicating that virtual training, using data augmentation methods for dataset enrichment, benefits real detection and that the keypoint predictions may be used together with pose estimation algorithm to give estimates of aircraft pose related to runway.

List of Figures

FIGURE 1.1 – Example of visual differences during landing phase. In this case, there are differences in terms of height and descent path angle between images (Extracted from Federal Aviation Administration (2016)). 17

FIGURE 1.2 – Optimal glide path in the landing phase. The GS signal gives the vertical reference (light blue) and LOC signal gives the horizontal reference (Adapted from StudyFlying (2019)). 19

FIGURE 1.3 – Satellite based systems' accuracy per coverage (Adapted from European Space Agency (2011)). 20

FIGURE 1.4 – The functional placement of positional modules in aircraft navigation (Adapted from International Virtual Aviation Organisation (2022)). 22

FIGURE 2.1 – Example with a linear regression. The learned parameters give an estimate of the relationship between car pricing and weight. 25

FIGURE 2.2 – A higher degree polynomial function could make the learning algorithm more costly in computational power. 26

FIGURE 2.3 – A basic neural network, or shallow neural network, comprises some neurons distributed in layers. 26

FIGURE 2.4 – The representation of input features are in two-dimensional format (pixels). 29

FIGURE 2.5 – Outputs from different filters are concatenated, forming the third dimension (depth or channels). 29

FIGURE 2.6 – Padding area being added to an input block. 30

FIGURE 2.7 – A stride of two would make the output block shrink by roughly a factor of two. 31

FIGURE 2.8 – Max-pooling layer takes the greatest number among all inside the filter as output. 32
FIGURE 2.9 – Bypass added from layer l to layer l+1's output. 32
FIGURE 2.10 –Inception layer made by different sized filters. 33
FIGURE 2.11 –Example of object detection using Detectron2 (DETECTRON2, 2021b). 34
FIGURE 2.12 –Example of landmarking in human bodies, with the detection of joints as keypoints (DETECTRON2, 2021b). 34
FIGURE 2.13 –Added region of interest feature to CNNs' architecture (Adapted from Girshick *et al.* (2014)). 36
FIGURE 2.14 –By the added branch, the network simultaneously works on masking and bounding box prediction (HE *et al.*, 2017). 36

FIGURE 4.1 – Illustration of the generated vector cone with respect to the runway. 45
FIGURE 4.2 – Example screenshot taken from the Flight Gear simulator. 46
FIGURE 4.3 – Annotated image pair for the example in Figure 4.2. 47
FIGURE 4.4 – The two steps performed by the annotator: differentiation between clean and annotated image, followed by the extraction of the center point of the four difference zones. 47
FIGURE 4.5 – Block diagram of image exclusion from the dataset due to the keypoints' visibility. 48
FIGURE 4.6 – The world coordinate axes placement on SBLO runway. 50

FIGURE 5.1 – Total loss evolution for training and validation datasets. 52
FIGURE 5.2 – Keypoint loss evolution for training and validation datasets. 52
FIGURE 5.3 – Total loss evolution for training and real datasets. 53
FIGURE 5.4 – Keypoint loss evolution for training and real datasets. 53
FIGURE 5.5 – Good detection results on the test dataset. 54
FIGURE 5.6 – Wrong results on the test dataset. 55
FIGURE 5.7 – Image with bad unfiltered annotation on the training dataset. 56
FIGURE 5.8 – Detection with different epochs models (2000). 56
FIGURE 5.9 – Detection with different epochs models (71000). 56
FIGURE 5.10 –Detection with different epochs models (2000). 57

FIGURE 5.11 –Detection with different epochs models (71000). 57

FIGURE 5.12 –Detection with different epochs models (2000). 57

FIGURE 5.13 –Detection with different epochs models (71000). 57

FIGURE 5.14 –Detection with different epochs models (2000). 57

FIGURE 5.15 –Detection with different epochs models (71000). 57

FIGURE 5.16 –Detection with different epochs models (2000). 58

FIGURE 5.17 –Detection with different epochs models (71000). 58

FIGURE 5.18 –Histogram of Euclidean distance absolute errors. 59

FIGURE 5.19 –Histogram of Euclidean distance absolute errors (filtered). 60

FIGURE 5.20 –Per-axis errors distribution related to distance from runway (filtered). 61

FIGURE 5.21 –Euclidean distance errors distribution related to distance from runway (filtered). 61

FIGURE 5.22 –Angles absolute errors' histogram (unfiltered). 63

List of Tables

TABLE 1.1 – Landing categories according to accuracy restrictions on systems (Adapted from Volpe (2001)). 18

TABLE 3.1 – Computer vision timeline. 39

TABLE 4.1 – Dataset Flight Gear airports' conditions used to generate the dataset with FG. 43

TABLE 4.2 – Dataset Flight Gear airports' orientation data used to generate the dataset with FG. 43

TABLE 4.3 – Detectron2 configuration. 49

TABLE 4.4 – Data augmentation used in training. 49

TABLE 4.5 – Arguments passed to PNP algorithm. 50

TABLE 5.1 – Datasets' composition 51

TABLE 5.2 – Evaluation metrics from prediction on the test dataset. 53

TABLE 5.3 – Error metrics for pose estimation. 58

TABLE 5.4 – Error metrics for pose estimation (filtered) 59

List of Abbreviations and Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CVS	Computer Vision System
DL	Deep Learning
DNN	Deep Neural Network
D2	Detectron2
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Administration
FG	FlightGear
GPS	Global Positioning System
GPU	Graphics Processing Unit
GS	Glideslope
ILS	Instrument Landing System
INS	Inertial Navigation System
IoU	Intersection-over-Union
LOC	Localizer
ML	Machine Learning
NN	Neural Network
OKS	Object Keypoint Similarity
RGB	Red, Green, and Blue
R-CNN	Region proposal Convolutional Neural Network
UAV	Unmanned Air Vehicle

List of Symbols

α	Learning rate
ϕ	Horizontal angle for landing cone
θ	Vertical angle for landing cone
x	Coordinate, in pixels, in the horizontal axis on the image
y	Coordinate, in pixels, in the vertical axis on the image
v_n	Visibility of keypoint n
dL	Derivative of loss function
dW_n	Derivative of learnable parameter n

Contents

1 Introduction 16

1.1 Motivation 16

1.1.1 The landing phase 17

1.1.2 Instrument Landing System 18

1.1.3 Global Navigation Satellite System 19

1.1.4 Computer Vision System 20

1.1.5 Systemic overview 21

1.2 Objective 22

1.3 Contribution 23

1.4 Dissertation overview 23

2 Knowledge Background 24

2.1 Machine learning and neural networks 24

2.2 Convolutional neural networks 28

2.2.1 Architecture description 28

2.2.2 Layer types 31

2.2.3 CNN tasks 33

2.2.4 CNN usage 35

2.3 Region-proposal CNN evolution line 35

2.3.1 Mask R-CNN 36

2.4 Evaluation metrics 37

2.4.1 Intersection-over-union 37

2.4.2 Object Keypoint Similarity 37

3 Literature Review 38

3.1 Timeline 38

4 Methodology 41

4.1 Project workflow 42

4.2 Aircraft positioning 42

4.3 Python interface 43

4.4 Simulation control and modding 44

4.5 Screenshot generation 45

4.6 Annotation 46

4.6.1 Detectron2 dataset format 46

4.6.2 Differential annotator 46

4.7 Dataset registering 48

4.8 Training and evaluation 49

4.9 Pose estimator 50

5 Results 51

5.1 Loss functions 51

5.1.1 Virtual-virtual analysis 51

5.1.2 Virtual-real analysis 52

5.2 Evaluation 53

5.2.1 Test dataset 53

5.2.2 Real dataset 56

5.3 Pose estimation 58

6 Conclusion 64

Bibliography 66

1 Introduction

This chapter will approach the motivation to give a better understanding of its contribution to society, as well as the objectives of this Master thesis' project in progress.

1.1 Motivation

With the advances in electronics and general computation, many aspects of human life were altered, made faster, or more efficient. The industry automatization, which is the face of a vast group of changes made by technology, has taken roots inside the aviation industry. With such, the embedding of technologies has solidified in the industry, now being in a steep climb towards the presence in manufactured products (Aerocorner, 2022a).

There are two main scenarios running that support our motivation. The first is a crescent necessity of a more connected world; This necessity, in terms of transportation, heavily relies on safe flight routes between countries, provinces and cities, and one of the means to achieve this goal is the better technology onboard the aircraft. Airline companies are in a constant need of better, more efficient planes that can reach more places in safety (Aerocorner, 2022a).

The second scenario analyzed is Brazillian's National Defense Strategy (Brazil Defense Ministry, 2022) with its directives to supply the country with modern armed forces, well equipped, trained and in permanent readiness state, capable of discouraging threats and agressions (AED-8); And also the Blue Amazon (Amazônia Azul®), Brazil's vast territorial sea with over 5.7 million squared kilometers, which must be guarded and defended in order to maintain sovereignty. The whole defense concept also leads to better, more efficient air-naval equipment and vehicles in order to be efficiently executed.

A regular airplane – a winged structure, commanded by a human which in the past solely used its visual sensoric aspects – became a massive platform of systems and automatic machinery built to relief the workload to the pilots and make the flight more economically efficient (CROUCH, 2022).

The advances made flight travel safer, cheaper and allowed the flying vessels to become carriers of persuasive power or accurate surveillance systems for military purposes. Thus, the addition of assistance systems is a task that must be continuous to guarantee reliable products, safer operation, and quality of life. The insertion of more landing assistance systems would (ROSCOE; GRIEVE, 1986):

- Give pilots less workload and stress, reducing human error probability.
- Get precise navigation information if augmented with data fusion from other sensors.
- Further open the path to single-pilot operations for commercial aviation, or unmanned operations for other areas.

1.1.1 The landing phase

The landing operations is a flight phase in which the pilot has increased workload and the attention requirement is higher (KUGLER *et al.*, 2019). This is mainly due to the phase's piloting aspects:

1. Controlled air speed loss to achieve touchdown speed.
2. Maintain leveled wings.
3. Adjusting lift according to speed loss, either with angle of attack or flap angle.
4. Controlled and fixed path angle.

As seen in Figure 1.1, information about orientation and position can be naturally extracted from visual differences in terms of path angle, roll angle or distance to runway.

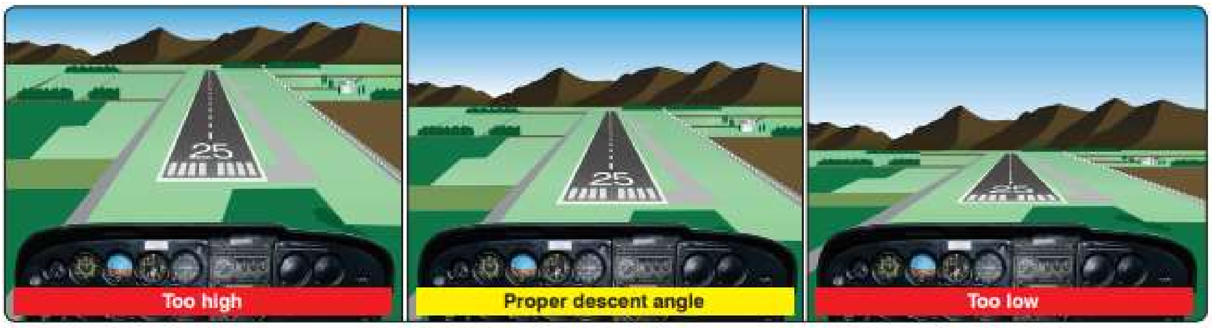


FIGURE 1.1 – Example of visual differences during landing phase. In this case, there are differences in terms of height and descent path angle between images (Extracted from Federal Aviation Administration (2016)).

As said before, human errors in this flight phase can lead to catastrophic events. Statistically, it is known that 45% of the accidents involved in general aviation happened during take-off or landing phases, and more than 90% of them were directly caused by human error. (Federal Aviation Administration, 2016). Notice that before the widespread use of Global Navigation Satellite System (GNSS), Instrument Landing System (ILS) and Inertial Navigation System (INS) modules, visual navigation was used during landing. The electronics advancements of those modules provided safety to landing procedures as it made "blind" landing possible (Aircraft Systems Tech, 2022).

The operational performance requirements for precision approach and landing are provided in Table 1.1. These are based on FAA and International Civil Aviation Organization (ICAO) requirements for use of a Global Navigation Satellite System (GNSS).

TABLE 1.1 – Landing categories according to accuracy restrictions on systems (Adapted from Volpe (2001)).

Operation	**Accuracy [m]**
Cat. I	16 (H) and 4.0 to 6.0 (V)
Cat. II	6.9 (H) and 2 (V)
Cat. III	6.2 (H) and 2 (V)

It means that, as example, to be certified for landing in CAT III (which is the most strict category for low visibility operation), an aircraft navigation system must have a positional accuracy of at least 2 meters on the local vertical, and 6.2 meters on horizontal plane.

1.1.2 Instrument Landing System

The most used system nowadays for pilot guidance during the landing phase is the Instrument Landing System (ILS). It comprises two land-positioned frequency emitters, one for vertical guidance, called Glide Slope (GS), and one for horizontal guidance, called Localizer (LOC). Depending on the received signal, the onboard system detects whether the path is higher, lower, to the left or the right compared to a central glide path, as seen in Figure 1.2.

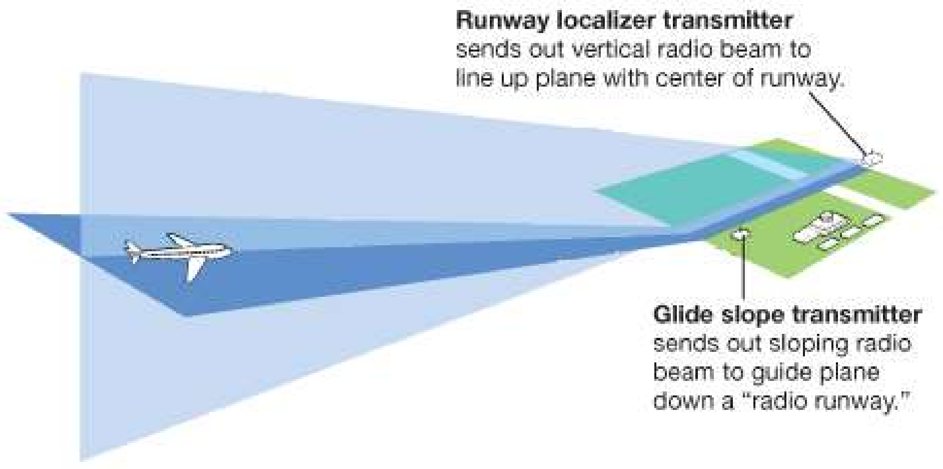

Runway localizer transmitter
sends out vertical radio beam to line up plane with center of runway.
Glide slope transmitter
sends out sloping radio beam to guide plane down a "radio runway."

FIGURE 1.2 – Optimal glide path in the landing phase. The GS signal gives the vertical reference (light blue) and LOC signal gives the horizontal reference (Adapted from StudyFlying (2019)).

The system is simple but efficient, with the analog signaling further contributing for its robustness. By itself, it is possible to reach the requirements for CAT III landing. Its only major downside is the need of an external ground structure for signal emitting and associated maintenance tasks.

1.1.3 Global Navigation Satellite System

GNSS uses orbital satellites to provide positioning to receivers around the globe. It is necessary at least 4 satellites signaling for the receiver on Earth to sucessfully determine four unknown variables: three with respect to positioning and one with respect to time delay (GARMIN, 2022). GNSS has a huge coverage and is currently in major applications worldwide, but it is susceptible to some effects, such as (VOLPE, 2001):

- Unintentional:
  - Ionospheric interference and solar flares.
  - Interference from radiofrequency (RF) emitters near or on planet surface.
- Intentional:
  - Shutdown due to conflicts.
  - Jamming, spoofing, and meaconing.

On top of that, GNSS signal has some accuracy restrictions (European Space Agency, 2011). Figure 1.3 shows satellite-based navigation technologies according to their positional accuracy versus coverage distance. Note that Ground Based Augmentation Systems

(GBAS) have a maximum coverage of around 20 kilometers, but that does not mean the navigation systems can no longer be used farther away; Instead, they lose the augmentation benefits and fall back into normal GNSS conditions. The same applies to Satellite Based Augmentation Systems (SBAS), although their accuracy is not enough, by itself, to fulfill requirements for CAT III landing procedure. Further information can be found at European Space Agency (2011).

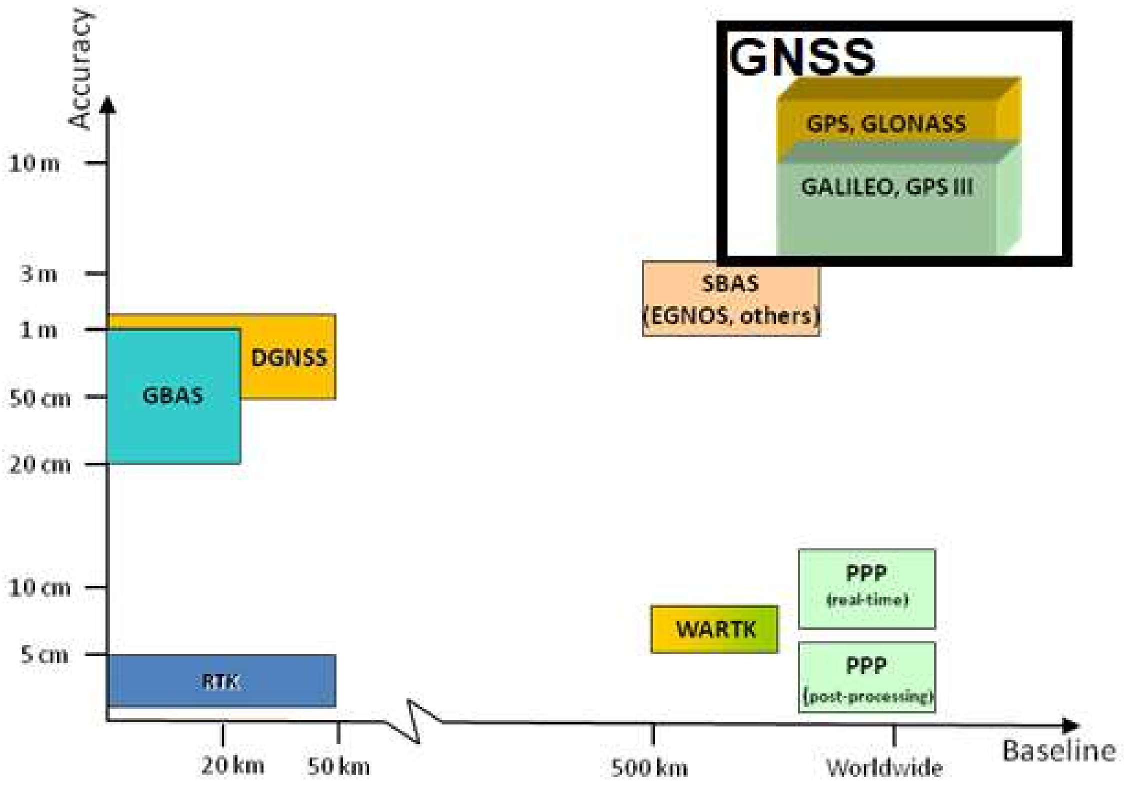


FIGURE 1.3 – Satellite based systems' accuracy per coverage (Adapted from European Space Agency (2011)).

GNSS without augmentation as good as GBAS or better cannot be used in a safe landing for CAT III, and GBAS falls into the same downside as ILS in terms of necessity of infrastructure. That reinforces the need of data fusion with other navigation system, preferably not affected by the vulnerabilities the GNSS has.

1.1.4 Computer Vision System

We define a Computer Vision System (CVS) as future systems using cameras to extract navigational information, with images as stimuli. Cameras in aircraft can act as the pilot's eyes for manned vehicles, or be visual sensors for the Flight Control System (FCS) for unmanned vehicles (Phase 1 Technology Corporation, 2021). As said before, visual aspects of the landing phase can provide information in terms of aircraft pose related to runway (see

Figure 1.1). As example, the aircraft roll angle has a clean visual effect, given its effect of rotation in a camera image. With CVS implemented, the flight safety could be higher in approaches without ILS guidance, providing a broader coverage of airports for commercial flight while possibly fulfilling safety requirements when used as augmentation for GNSS or INS. Regarding cost-oriented markets, which are the commercial and executive aviation, the main benefit of such technology would be the safer single pilot operation, the reduction in landing risks due to errors or mistakes (Phase 1 Technology Corporation, 2021), and the ability to land in airports that do not have an installed ILS.

For the military sector, the main outcome of such technology would be:

- Independence from external systems, like satellite positioning systems or land signaling.
- Smarter autonomous aircraft for surveillance or other military missions.

The safety effect of auxiliary systems in this case would be higher, given that most of military aviation missions are single-piloted or even unmanned. As the system depends on visual clearance, the following downsides also may apply (HO; CHAKRAVARTY, 2014):

- Low or no efficiency if target occlusion is present due to weather condition, terrain, and man-made structures.
- Reduced efficiency due to lack of maintenance procedures on camera and lenses.
- Reduced efficiency due to ice formation on lenses and light refraction, ice formation on nearby aircraft surfaces, light reflection, and lack of illumination.
- Reduced efficiency due to sunlight flare.

Globally, there are organisations taking effort in order to estimate aircraft pose from runway images during landing phase. Some recent work with similar objectives to a CVS will be explained in Chapter 3.

1.1.5 Systemic overview

The role of modules such as GPS, ILS, INS, or future CVS is: to provide, to the pilot or Flight Control System, aircraft position with respect to a desired reference (Aerocorner, 2022b). As an example, Figure 1.4 illustrates the system placement for modules in a automatic landing flight phase. Positional information is fed to the FCS, which will correct its course to another referential (generally, the optimal approach path according to landing category).

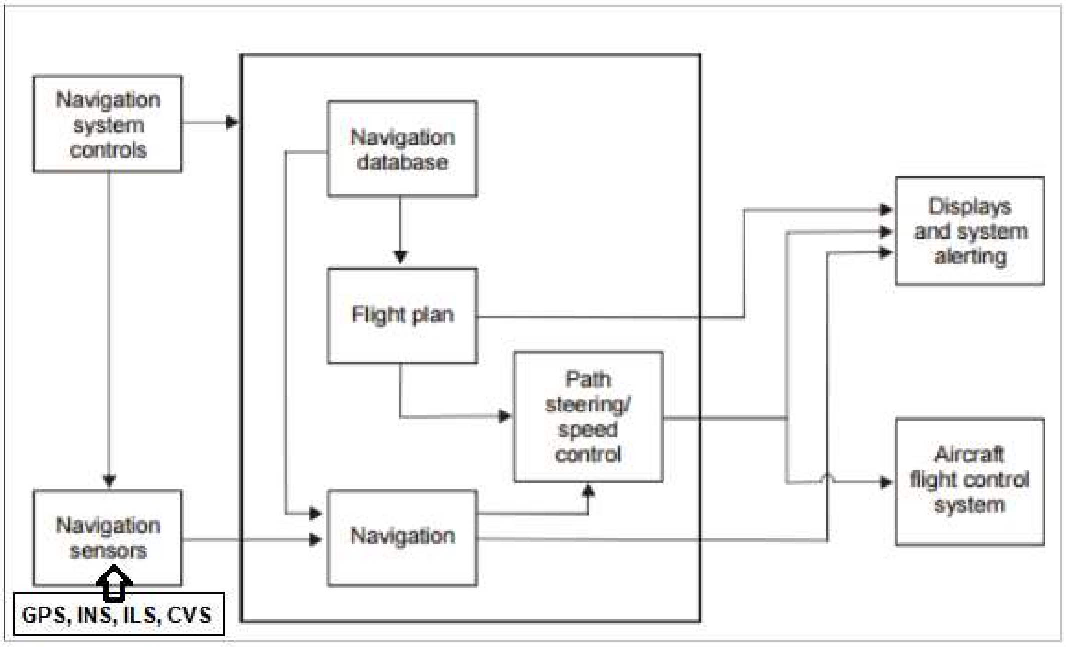

Navigation system controls
Navigation database
Flight plan
Path steering/ speed control
Navigation
Navigation sensors
GPS, INS, ILS, CVS
Displays and system alerting
Aircraft flight control system

FIGURE 1.4 – The functional placement of positional modules in aircraft navigation (Adapted from International Virtual Aviation Organisation (2022)).

Notice that each of the mentioned modules provide information with respect to different references: the GPS and INS with respect to Earth geoid; the ILS with respect to glideslope and localizer planes; and CVS, in landing procedures, with respect to the desired runway. Nonetheless, the addition of a certified CVS to an aircraft would contribute to existing navigation systems, covering downsides on GNSS and ILS, while providing a different technology (optical) as backup, further strengthening the aircraft's navigation system's integrity and availability.

1.2 Objective

The main objective of this work is to verify the feasibility of using convolutional neural networks (CNN) to detect airport runways and their corners as keypoints, in virtual pictures simulating the point of view of an aircraft during its landing flight phase. We then evaluate: results of pose estimation algorithm using keypoints; and detection technique on real runway images. We outline the following specific objectives to reach the main goal:

1. Generate the virtual image dataset on airports.
2. Build an automatic annotation tool.
3. Train and validate using the generated dataset.

4. Evaluate the trained network on a test dataset.
5. Evaluate inference on real runway images.
6. Extract pose estimates from data provided by inference.

1.3 Contribution

To the best of our knowledge, this is the first work to use deep learning keypoint detection method to identify and localize runway features in virtual images simulated as if the aircraft was in landing phase, use virtual runway images to train and analyze training effects on real runway images and use the keypoint information in order to estimate aircraft position.

1.4 Dissertation overview

From here on, the document will detail how we successfully trained a network to detect runway in pictures by simulating camera angles similar to that of a landing airplane.

- Chapter 2 explains the prior technical background that led to current computer vision systems.
- Chapter 3 denotes related work according to a timeline of computer vision and vision landing system evolution.
- Chapter 4 explains how the project was built with its main components and characteristics.
- Chapter 5 will show the obtained results on the airport simulated sceneries, as well as the usage of data to estimate aircraft position; Additionaly, will show qualitative analysis regarding runway detection on a real dataset.
- Chapter 6 will conclude our work and explain about future steps.

2 Knowledge Background

The present chapter approaches the knowledge needed for better understand the technical aspects of the project.

2.1 Machine learning and neural networks

Machine Learning is often defined as the process of making a computer effective at doing a determined task without explicitly programming it to do so (BURNS, 2022). With access to a reasonable amount of data, the program can learn by itself by iterating over learning phases in each data.

A Machine Learning problem can be described according to Tom Mitchell's formalism (MITCHELL, 1997): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

A learning task can be of two forms (BURNS, 2022):

- Supervised: when the algorithm is fed with information of input data as well as information of what the output, or the right answer, to that data should be. It is the most basic type of learning as if you are training the machine by repeating and strengthening. Example: Feeding an algorithm with a thousand different pictures of a bonfire, and, for each of them, actually telling (in computational means) that the picture is a bonfire.
- Unsupervised: when all that is fed into the algorithm is the input data. In this case, it can learn to detect patterns in the data or segment it into groups by similarity, although not actually knowing what each of them represents. Example: Feeding an algorithm with 500 pictures of a bonfire and 500 pictures of an ice cube. As an ideal outcome, it is expected that the program will divide the thousand pictures into two groups by similarity, although not knowing what they represent. Group A might be assigned to 500 pictures of a bonfire, and group B to the ice cubes.

When presented to a new image of an oven fire, it could assign it to group A due to similarity or assign it to a newly created group C and accommodate this new class found (depending on training quality, hyperparameters, and other variables not discussed).

Supervised learning is the most used and can be very powerful depending on the training data it relies on. As a very simple example, a computer program can learn to estimate the price of cars in the market, according to their weight, by making it learn a pricing formula from data samples (HORNIK *et al.*, 1989). In this case, the data would be a collection of pairs of information from collected, real values: the weight of a given car and its price.

The learning algorithm will basically try to learn the right parameters to best fit a regression for the data given, as seen in Figure 2.1.

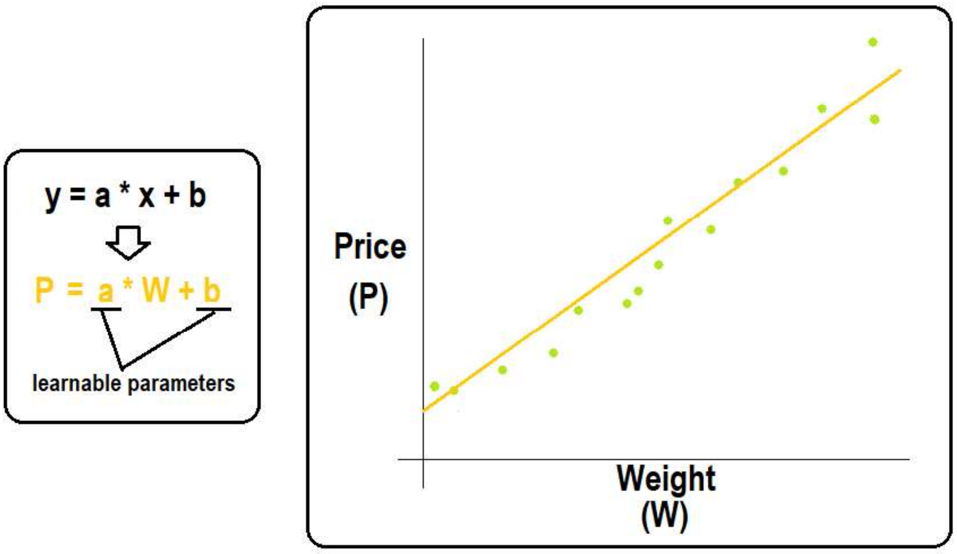


FIGURE 2.1 – Example with a linear regression. The learned parameters give an estimate of the relationship between car pricing and weight.

A linear regression, with two parameters to learn, might be enough in a small number of applications. Some applications may require a quadratic function, a higher degree polynomial, or any learnable function (Figure 2.2), and that could make the learning algorithm more costly in terms of computational power.

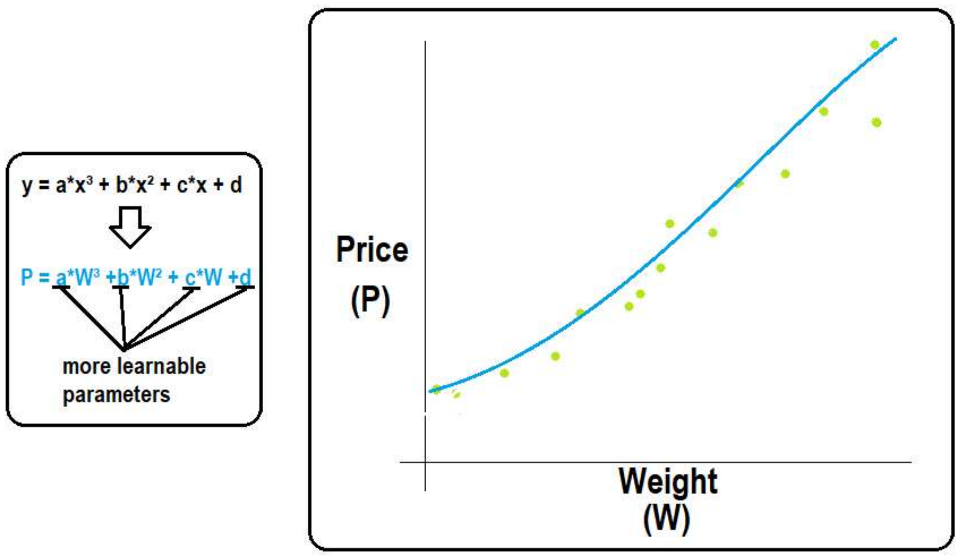


FIGURE 2.2 – A higher degree polynomial function could make the learning algorithm more costly in computational power.

Assume that the learnable function is called a “neuron” that takes its input and estimates the cost. As we advance in complexity, some applications might want to stack, horizontally or vertically, these neurons for even more complex functions to be learned. To do that, a non-linear function can be applied at the output of each neuron, called the activation function. When that is done, the most basic neural network is created (Figure 2.3) and new definitions are made:

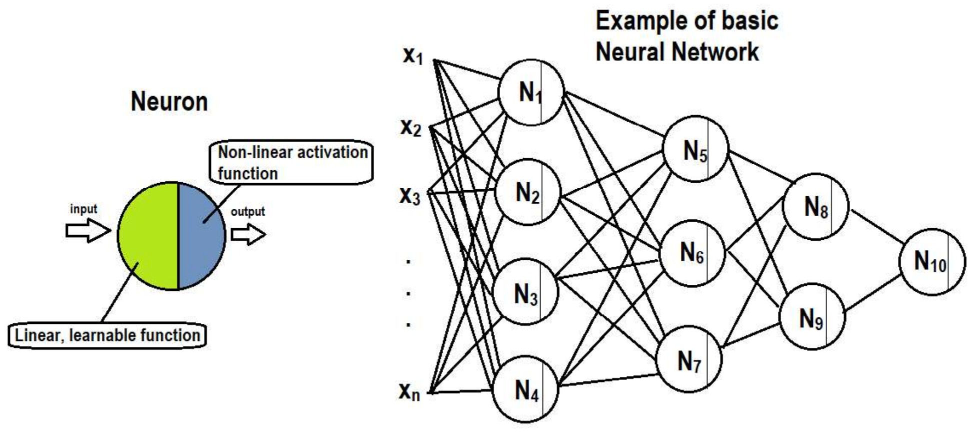

Neuron
Non-linear activation function
input
output
Linear, learnable function
Example of basic Neural Network
X1
X2
X3
Xn
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10

FIGURE 2.3 – A basic neural network, or shallow neural network, comprises some neurons distributed in layers.

- The functions that take the input are the first layer of computation.
- The functions that take as input the outputs of the first layer compose the second layer.
- Then, the third layer takes the outputs from the second layer, and so on.

Each function block is defined as a neuron, and the combination modes of different number of neurons per layer or number of layers in the whole network can achieve a number of functions to be mapped and learned. An artificial neural network (ANN) can approximate itself to a desired function with some tolerance, provided that it has a minimal number of neurons. Mathematically, this means that ANN are universal approximators (HORNIK *et al.*, 1989). This can be achieved by the non-linear activation functions added to each linear function inside the neurons, as seen in Figure 2.3. without the non-linearity, the whole network would still be a linear function and would not be able to map a huge amount of different functions. This property of a neural network is surely why it is so powerful. In fact, given a complex application and a reasonable amount of data to train on (KRIZHEVSKY *et al.*, 2012), the network might learn a complex function even when the person supervising its training does not have any clue of what it looks like, or what it should be.

Learning process

One of the main mechanics that allows a model to learn is the backpropagation algorithm (GOODFELLOW *et al.*, 2016). Its first step is the reverse calculation of how much the loss function varies (derivatives) with each of the learnable parameters in a network. The second is to make each of the parameters converge according to the calculated derivatives. As example, let us assume a learnable parameter W and a loss function L, which, simply put, is a function that measures how far from the fitting points the trained model is at the moment. After dL/dW is calculated, W assumes a new value

$$W_n = W_{n-1} - \alpha \cdot \frac{dL}{dW_{n-1}},$$

where α is called the learning rate. This process is one step of a gradient descent, and is iterated a number of times for each learnable parameter and each data in the training set, until the human supervisor decides it has learned enough or an automatic stopping condition is reached.

A successful learning algorithm presents a loss function decay at mostly all the training session. The descending function denotes that the network is fitting each time better to the training set.

2.2 Convolutional neural networks

A problem emerged once the amount of input data began to rise. In deep neural networks, the number of parameters to be trained is very depending on the number of inputs to the network and the number of neurons (GOODFELLOW *et al.*, 2016). With that, some types of data (images, for example) have a considerable amount of features compared to other applications. A 640x480 picture in the Red-Green-Blue (RGB) format, as an example, would have 921,600 input variables, and, in a network with 1000 neurons in the first layer, the number of learnable parameters will reach about 921 million. And this refers only to the first layer. In image applications, it is known that the function needed to get reasonable inference has to be complex, given the virtually infinite combinations of pixels it can have (LECUN *et al.*, 1998).

So, instead of fully connected layers, as seen in deep neural networks, a new implementation arises: the convolution layer. Basically, it makes the operation simpler by reducing the number of learnable parameters per cell, which relates to the number of parameters in the filter (analogous to neurons). The convolution works in a two-dimensional shape, and its computational efficiency was proven in images due to its mechanics of only connecting data from one layer to another if they are in proximity (GOODFELLOW *et al.*, 2016). A Convolutional Neural Network (CNN) can be trained for different tasks, further explained later.

2.2.1 Architecture description

As CNNs are mostly used for image applications (GOODFELLOW *et al.*, 2016), the input layer (and subsequent layers as well) is no longer represented as a column vector of input features. Instead, due to the mechanics of convolution, the data is represented in two-dimensional space, representing the pixels in the image for the input layer, or a feature detection in mid-layers, as seen in Figure 2.4. There is also a third dimension – depth – which, in the first layer represents the red, green, and blue channels, while for other layers is the dimension in which the concatenation of convolution results are made for different filters. Now the three-dimensional data cluster in each layer is comprised of a block with height, width (forming a surface area - image), and the number of channels (depth).

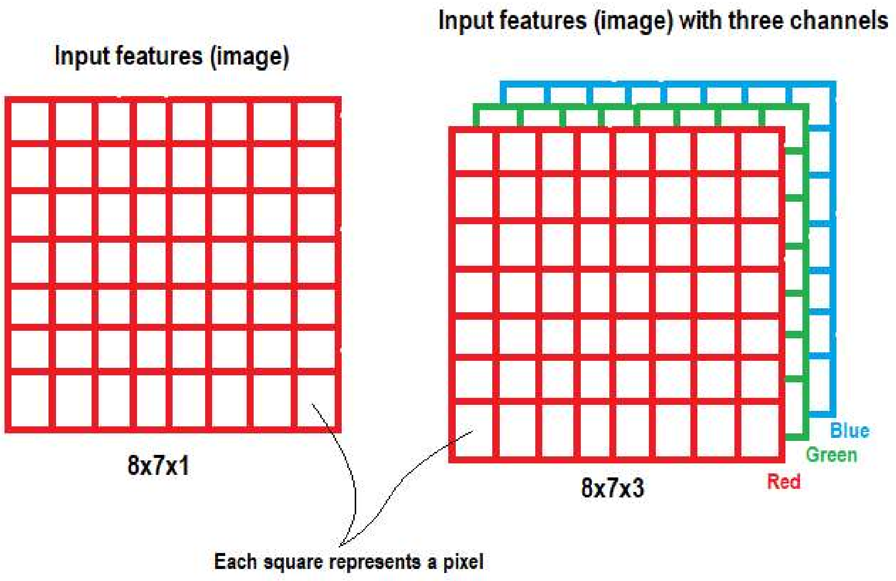

Input features (image)
Input features (image) with three channels
8x7x1
8x7x3
Blue
Green
Red
Each square represents a pixel

FIGURE 2.4 – The representation of input features are in two-dimensional format (pixels).

Filters

The filters are the operational cells in the CNNs and have the trainable variables for the network. A convolution in a layer is made by sweeping the filters, one by one, through the input block, and concatenating the results depthwise (see Figure 2.5).

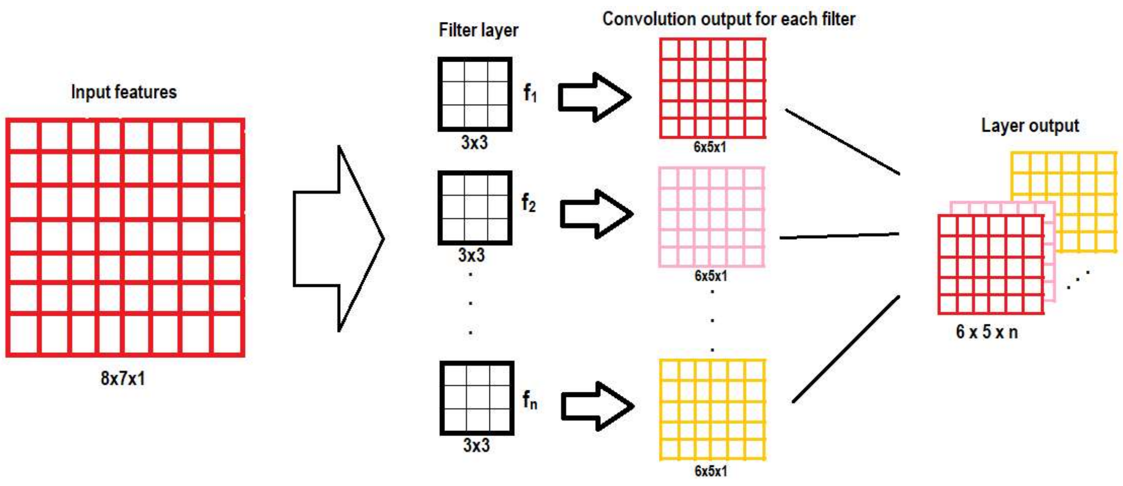


FIGURE 2.5 – Outputs from different filters are concatenated, forming the third dimension (depth or channels).

Padding

When a convolution is done, the image area shrinks according to the filter size. That is the most common type of convolution – the valid convolution. If one needs to maintain the block area, padding is needed: basically, an extension in the borders of the input volume sufficient to make the output and input have the same area size (Figure 2.6). This type of convolution is called the "same-convolution" (see Section 2.2.2).

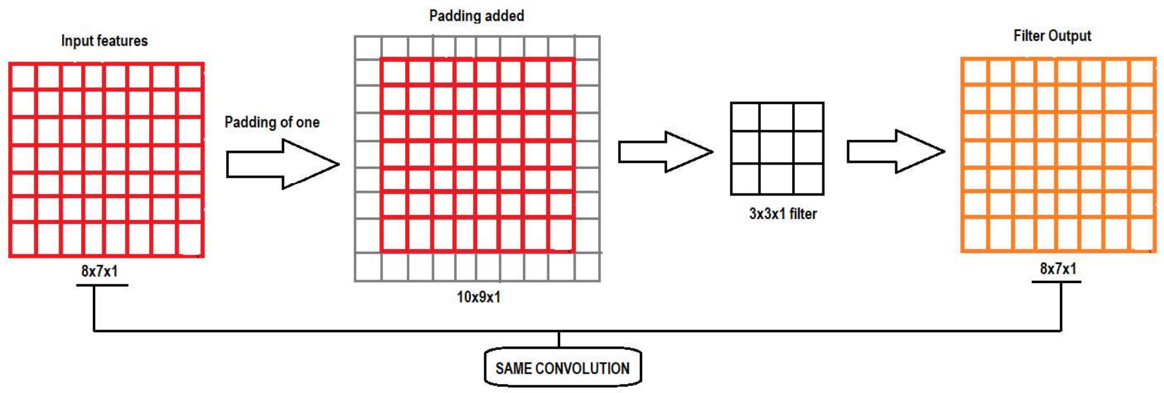


FIGURE 2.6 – Padding area being added to an input block.

Stride

The main mechanics of a convolution uses a one-element step sweep through the input block. If needed, the step size can be modified to any positive integer value at the cost of output block area size (see Figure 2.7)

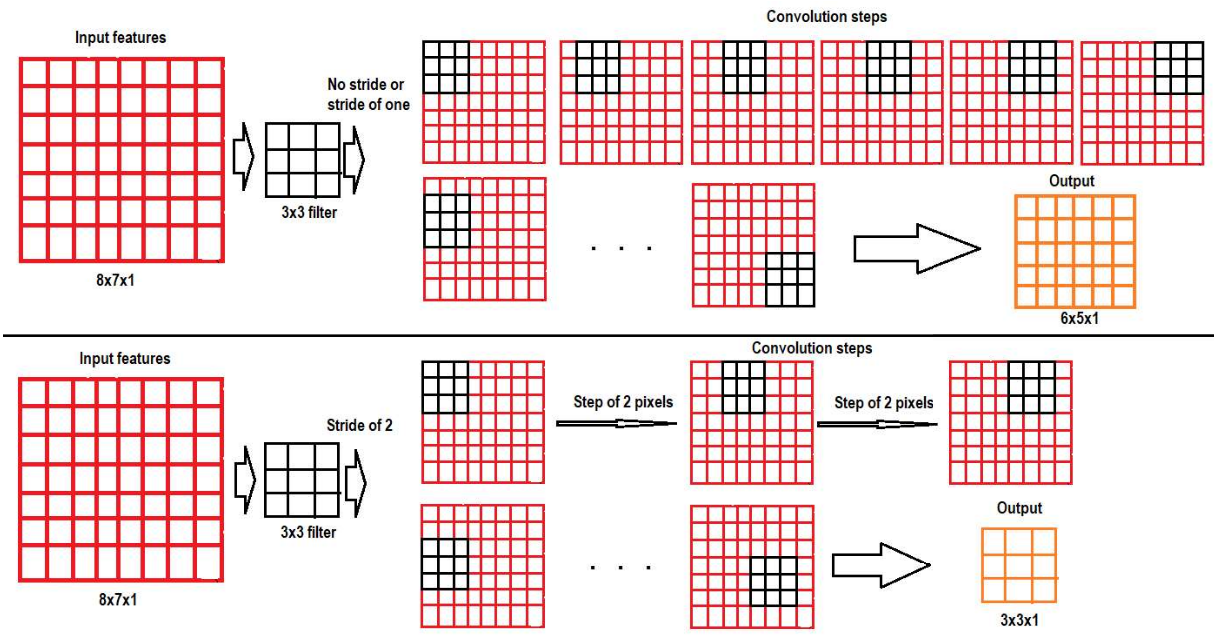


FIGURE 2.7 – A stride of two would make the output block shrink by roughly a factor of two.

2.2.2 Layer types

Convolution layer

The most basic type of layer, comprising a group of filters being convoluted over an input block. It can be a same-convolution (padding is added in order to equalize the output features' size to the input features' size) or valid-convolution (no padding is added and the convolution shrinks the input block area). This layer's purpose is, according to the combination of filters used, to extract low-level features and transform them into higher-level features. For example, one layer could take the pixels as input and output the likelihood of containing corners and edges in each region (ZEILER; FERGUS, 2014).

Pooling layer

The pooling operation uses a filter to shrink the input block according to some logic. The most used type is the max-pooling layer (GOODFELLOW *et al.*, 2016), in which, for each convolution step, the result will be the greatest number among the ones being evaluated by the filter (see Figure 2.8). The average-pooling layer, which outputs the average all input elements can also be used in the architecture, among other types of pooling layers.

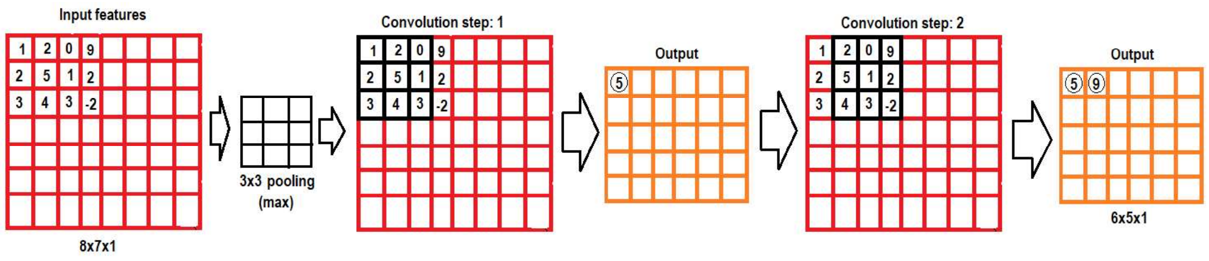


FIGURE 2.8 – Max-pooling layer takes the greatest number among all inside the filter as output.

Residual layer

Introduced by the ResNet manuscript (HE *et al.*, 2015), the residual layer is a basic convolution layer with a bypass connection added towards the output of two layers ahead (see Figure 2.9). This gives the network the ability to achieve the identity function over the bypassed layers, and mitigates the problem of vanishing and exploding gradients, allowing implementation of deeper networks.

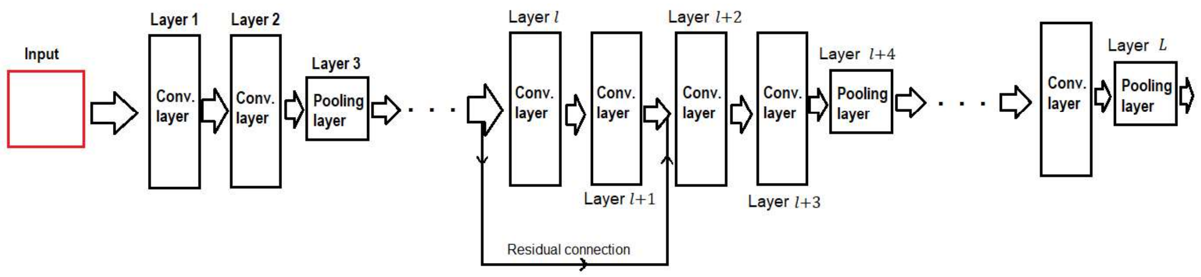


FIGURE 2.9 – Bypass added from layer l to layer l+1's output.

Inception layer

Introduced by the Inception Network manuscript (SZEGEDY *et al.*, 2014), the inception layer is a convolution layer with mixed-up filter sizes. With it, it is possible, through same-convolutions, to concatenate the results of 2x2, 3x3, 7x7, or filters with other sizes (see Figure 2.10).

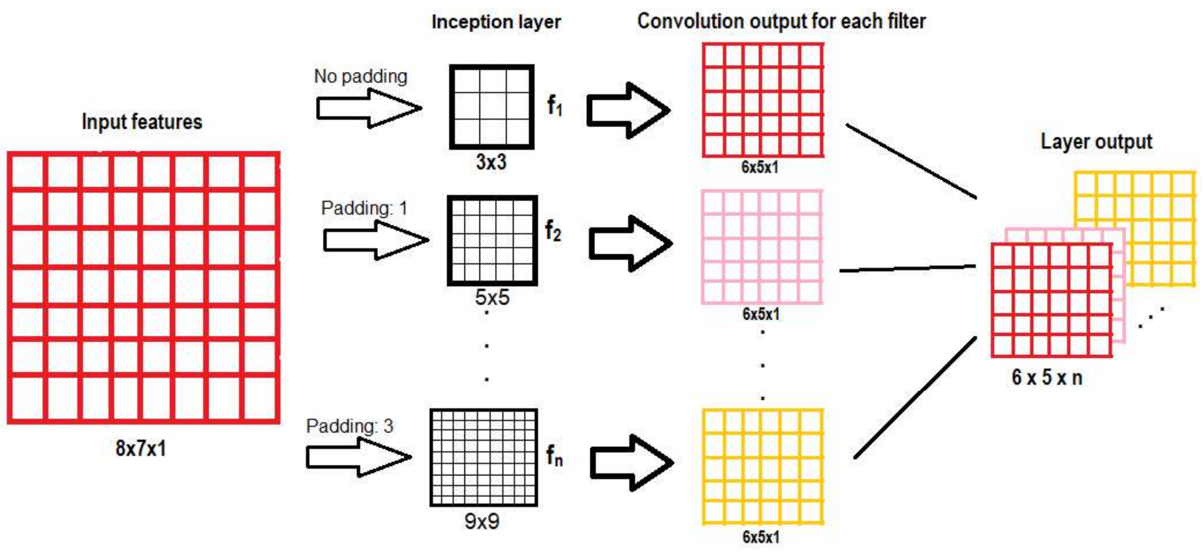


FIGURE 2.10 – Inception layer made by different sized filters.

2.2.3 CNN tasks

The most common uses for CNN are:

Classification

The inference on a given image gives a boolean result for classification purposes, as in recognizing a cat, a dog, a person, or common objects in a picture. (KRIZHEVSKY *et al.*, 2012).

Object detection

The detection task from inference gives the pixel-wise location of an object's center if detected in an image (AKTAS, 2022). Some networks can also infer bounding boxes for the detected objects (see Figure 2.11).

Landmarking or keypoint detection

In addition to object detection, the inference gives detailed keypoints of a given object. For example, locating a car and its wheels and windows as keypoints; Or a person and their limbs and head position (LAVANYA S., 2021), as seen in Figure 2.12.

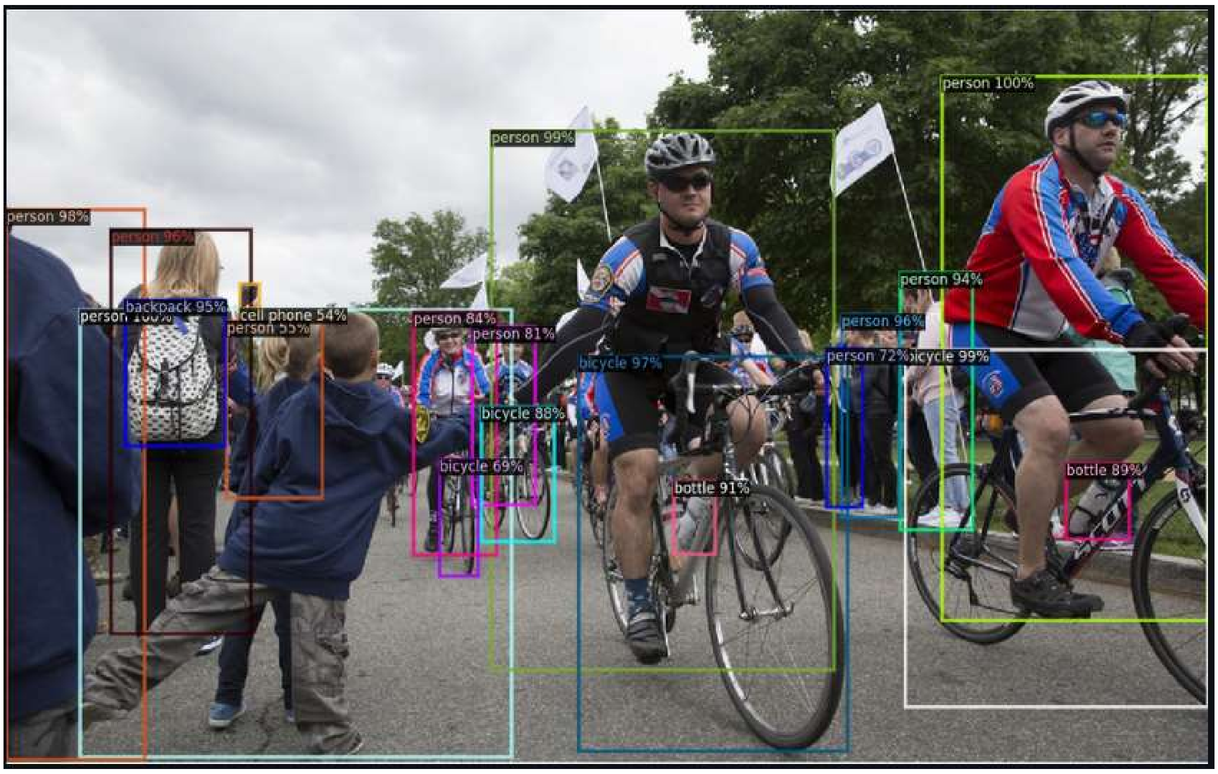


FIGURE 2.11 – Example of object detection using Detectron2 (DETECTRON2, 2021b).

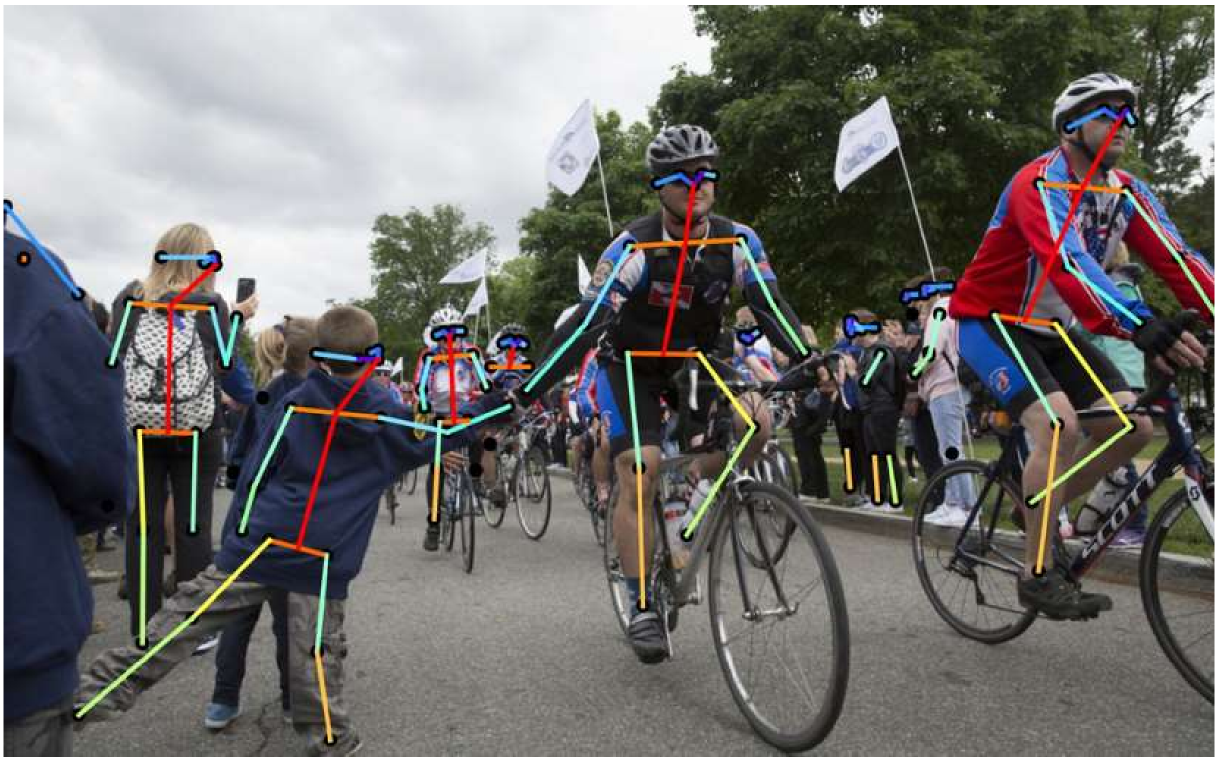


FIGURE 2.12 – Example of landmarking in human bodies, with the detection of joints as keypoints (DETECTRON2, 2021b).

Segmentation

The segmentation task at first combines Classification and Object Detection to give multiple detections in the image, as well as the localization information about them (KUMAR, 2020). Additional information can come with:

- Instance segmentation: the network detects, classifies, localizes, and estimates pixels belonging to each detected object.
- Semantic segmentation: the network classifies each pixel in the image as belonging to a class (e.g. pavement, road, car, sky), so the whole image will be divided into zones.
- Panoptic segmentation: combines instance and semantic segmentation to detect multiple objects and divide the image into zones according to each object.

2.2.4 CNN usage

Some CNNs in use today are:

- LeNet: one of the most basics CNNs used to detect and classify handwritten numbers (LECUN *et al.*, 1998).
- ResNet: as neural networks grew deeper, the problem of overfitting became greater. ResNet took the approach of adding the possibility of later layers of a network to be assigned the identity function. As result, a very deep network might use the latest layers if needed, and, if not, it can virtually reduce its depth by learning the identity function (HE *et al.*, 2015).
- Inception Network: uses a combination of filters sizes in the same layer, so the network will have more types of convolutional results to use and train on (SZEGEDY *et al.*, 2014).

2.3 Region-proposal CNN evolution line

In 2013, a new approach to reduce computational costs of CNNs was made by introducing the idea of "region of interest (RoI)" CNNs (R-CNNs). With that, the network makes a prior search for regions (Figure 2.13) where there might be objects in the image, thus reducing the area being convoluted on the subsequent layers (GIRSHICK *et al.*, 2014).

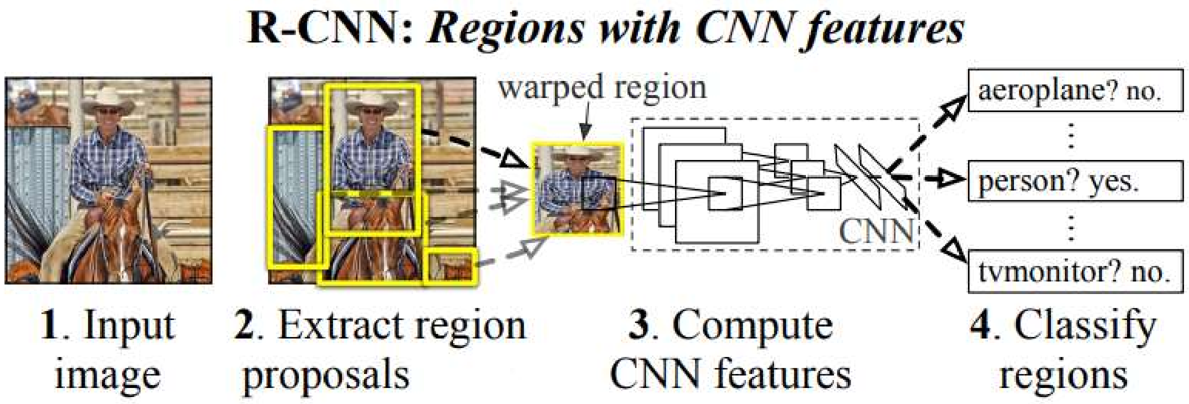


FIGURE 2.13 – Added region of interest feature to CNNs' architecture (Adapted from Girshick *et al.* (2014)).

The Fast R-CNN was created by using RoIPool to achieve faster and more accurate RoI from feature maps. Then, Faster R-CNN granted an extension by adding a region proposal network (RPN) and learning about regions of interest (HE *et al.*, 2017).

2.3.1 Mask R-CNN

As an evolution to Faster R-CNN, and with a slight increase in computational cost, the Mask R-CNN adds a parallel branch (see Figure 2.14) to the architecture which predicts objects instancing masks. Its results were the efficient division of different objects in the same image, by using the given predicted masks (HE *et al.*, 2017).

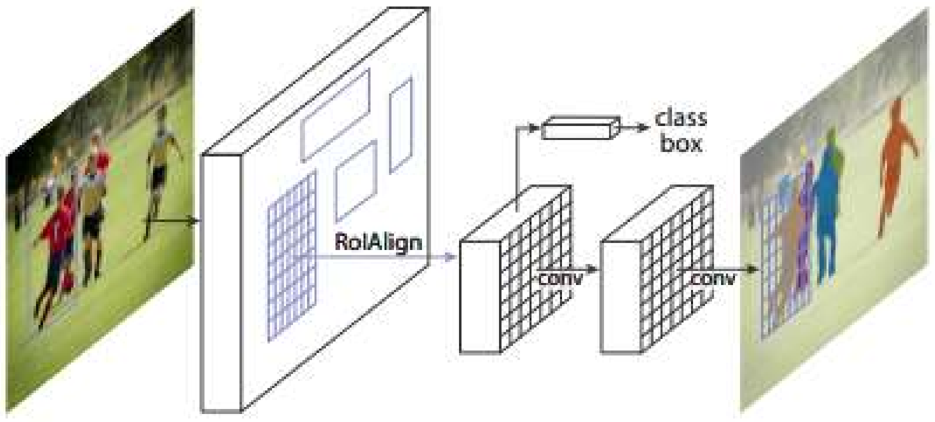


FIGURE 2.14 – By the added branch, the network simultaneously works on masking and bounding box prediction (HE *et al.*, 2017).

Currently, Detectron2's project in Github (DETECTRON2, 2021b) is based on Mask R-CNN and has the full capacity of all prior explained tasks: instance, semantic, landmark/keypoint, and panoptic segmentation. This makes Detectron2's framework capable of covering the majority of applications for developers and enthusiasts.

2.4 Evaluation metrics

Metrics are calculations used to measure the performance of an object, keypoint or segmentation detector. In this project, metrics may apply for bounding box prediction and keypoint prediction. On both cases, the Average Precision (AP) is the primary metric and can be calculated: for bounding boxes using Intersect-over-union (IoU) concept and for keypoints using Object Keypoint Similarity (OKS) concept. Further explanation about the metrics is given by Lin *et al.* (2014). Some auxiliary metrics are:

- AP75: Average Precision for IoU/OKS greater than 0.75.
- APs, APl, APm: Average Precision calculated taking into account for objects that are small (area lesser than 32 pixels squared), large (area greater than 96 pixels squared) or medium (other cases), respectively.

2.4.1 Intersection-over-union

The intersect-over-union is the raw metric used for bounding box evaluation. It consists of a calculation involving the areas of the predicted bounding box (A_p) and the ground truth bounding box (A_{gt}). Mathematically, it is defined as

$$IoU = \frac{A_{gt} \cap A_p}{A_{gt} \cup A_p}.$$

2.4.2 Object Keypoint Similarity

Analogous to IoU, Object Keypoint Similarity is the most commonly used metric for keypoint evaluation. It is obtained, for each keypoint in each instance detected, by the following formula on all labeled and visible keypoints:

$$OKS = \frac{\sum_i^n e^{\frac{-d_i^2}{2s^2k_i^2}}}{n}, \tag{2.1}$$

where d_i is the Euclidean distance between predicted position and groundtruth position for the i^{th} keypoint, s is the object scale related to its area in pixels, k_i is the keypoint weight, and n is the number of keypoints.

3 Literature Review

This chapter's objective is to give information about related works in computer vision focused in landing assistance context, as well as the time scaling and evolution of techniques used for different applications.

3.1 Timeline

To define a time when computer vision was born is a difficult task, but it is known to be used since the primordial decades of computer science. Given the exponential rise of computer technology, only the latest technologies in the past decade will be approached.

Table 3.1 shows some related works from 1995 onwards. Around year 2000, literature research still marks the use of classic computer vision, which was made solely on arithmetic computation and pattern-seeking programs, with a heavy need for human workload and preprocessing/feature extraction.

TABLE 3.1 – Computer vision timeline.

Project	**Approach**	**Year**
LeNet (CNN)	DL/NN	1998
Zongur *et al.*	Classic	2009
Krizhevsky *et al.*	DL/NN	2012
R-CNN	DL/NN	2013
Inception	DL/NN	2014
ResNet	DL/NN	2015
Lee *et al.*	Classic	2016
Mask R-CNN	DL/NN	2017
Abu-Jbara *et al.*	DL/NN	2018
Kügler *et al.*	DL/NN	2019
Airbus	–	2021
Daedalean	DL/NN	2021
Daedalean and EASA	DL/NN	2021
Daedalean and FAA*et al.*	DL/NN	2022

The idea of making the program "learn" by itself was not largely implemented by then. Near the year of 2000, the publication of one of the first convolutional neural networks, the LeNet (LECUN *et al.*, 1998), happened and a new stream in computer vision began to rise. As the computational power grew due to technological advances, practical applications became viable for learning algorithms, and the human need for preprocessing and feature extraction came into fall.

Zongur *et al.* (2009) revealed a work to detect airports in aerial images by pattern recognition and image processing followed by the Adaboost learning algorithm. By this approach, a set of possibilities for runway locations in images are obtained, although in a coarse representation. Still in the classic approach, (LEE *et al.*, 2016) published a work of a runway detection algorithm for safe landing assist in aircraft. The approach is of geometrical processing and extracts lines and angles from infrared cameras' projection to detect the runway, and further improvements possibly include fusing data with the exact orientation and velocity of the aircraft. By using a classical computer vision approach, the project might not be adequade if camera parameters are modified or some parts of the runway are not visible. And the aerial image dataset explored cannot be used to guide aircraft during landing.

Kugler *et al.* (2019) published a work using a system called C2Land integrated with other aircraft components to better achieve autolanding safety. The vision system demonstrates an accurate outline around the runway. It is unknown whether C2Land uses classical computer vision or machine learning.

Abu-Jbara *et al.* (2018) published a work in which they implemented a real-time program for runway edge detection. Their algorithm seems to solely use semantic segmentation as output. They analyze the positional estimates taken from inference under a Kalman filter in video streaming from UAV landing. Their conclusion is that the vision system implemented can be efficient in assisting unmanned aircraft during landing conditions. Being an edge detector, the project might have reduced efficiency if the runway is partially visible.

The Wayfinder project (AIRBUS, 2021) is in its middle stage, and comprises an artificial intelligence capable of detecting bounding boxes of runway main features: threshold bars, aiming markings, and sidebars. Airbus reveals that the project is enforcing the concept of Autonomous Taxi, Take-Off, and Landing (ATTOL) in their commercial products. The network implemented has as output the distance from the runway as well as a virtually-created ILS plane for LOC and GS. The downside of this approach might be that some runways do not have all markings, so the network efficiency might decrease on these cases where only sidebars are detected, as example.

A company named Daedalean also works on a CVS for landing assistance (DAEDALEAN, 2021a). In the promotion video, we can see the system beginning as object detector as the bounding box is drawn. When the plane gets closer, the system begins to estimate the three parallel lines comprising the runway: sidebars and centerline.

EASA recently published a document, a joint work with Daedalean, entitled Concepts of Design Assurance for Neural Networks (Daedalean and EASA, 2021), currently on version 2. In the document, some standardization for airborne machine learning/neural network-based systems, with respect to trustworthiness and performance assurance, is made. The work made by these agencies approaches the subjects listed above by the W-shaped process, which is based on traditional system development and has the steps: Requirements, Data, Learning process, Model training, Validation, Implementation, Inference verification and integration, and Independent data. As a use case, the document analyzes a visual air traffic control that uses object detection.

Daedalean and FAA, following Daedalean and EASA's work, published a report of analysis on Daedalean's visual land assistance (DAEDALEAN, 2021b), with the objectives to evaluate the use case, verify certification assurance, generate and inform policies for regulatory framework compatibility, validate visual-based landing assistance as a backup role for aircraft guidance, among other minor objectives. These government entities' directives are to be used as guidance for future work, as they will be provide certification requirements for such application. Note that due to many of the related projects being industrial applications, more technical information could not be obtained.

4 Methodology

For a successful deep learning project, one of the most crucial aspects tends to be the dataset. The dataset generation for runway images has practical restrictions, such as:

- Costly flight hours, as we would need a set of runway pictures at the landing phase of flight.
- Dataset generation process would take a long period for a solid amount of pictures.
- Annotation process would also take a long period of work.

We decided to take the project on simulated images for training and evaluation. Considering our problem, the most usual approach would be to use software capable of rendering runways in simulated scenery, such as:

- X-Plane (X-PLANE, 2021).
- Unity (UNITY, 2021).
- Unreal Engine (UNREAL, 2021).
- Flight Gear (FLIGHTGEAR, 2021).
- Microsoft Flight Simulator (MICROSOFT, 2021).

Some aspects were taken into consideration when deciding the simulator:

- Readiness: whether the simulator already has the objects and functions needed to generate the dataset.
- Modding: capability of modifying aspects of the program to better suit the dataset needs.
- Render quality: will affect screenshot quality, and has great importance to computer vision applications.

- Price.

With those aspects in mind, the Flight Gear (FG) software became a priority due to the ease of execution, vast support community, being open-source, being free to use, and having the capability of interfacing with Python for data collection.

4.1 Project workflow

The steps taken in order to obtain the results are:

1. Decide aircraft positioning and orientation with respect to runway.
2. Clean screenshot generation: un-annotated, raw images of runways.
3. Annotated screenshot generation: annotated, marked images of runways.
4. Dictionary building: combination of both images types to generate the bounding box and keypoint data in Detectron2's format.
5. Training session: obtain total and keypoint loss curves for training, validation and real datasets.
6. Evaluation (virtual): obtain metrics (AP) on test set to avail the prediction confidence of the trained model.
7. Evaluation (real): qualitative evaluation of trained model's prediction on some real runway images.
8. Pose estimation evaluation: obtain metrics on the position and orientation estimation algorithm.

4.2 Aircraft positioning

Flight Gear is an open-source flight simulation program with an active community and forums which updates and adds new features. With it, you can fly any created airplane model through any available world scenario, given they are correctly configured for the program to read and execute. Nine airports, presented in Tables 4.1 and 4.2, were chosen to cover multiple characteristics in terms of number of runways and neighborhood, and provide a more robust training set. In Table 4.1, "Runways" column denotes how many runways are in the airport and their positional characteristic related to each other. "Nearby" indicates characteristics of the landscape around the scenery, and "Dataset" indicates to which dataset a given airport's pictures goes to. In Table 4.2, for a correct generation of random positions related to each runway (to be explained in Section 4.4), we provide information on runways being annotated, their designated runway number, true heading in degrees, the magnetic-to-true heading conversion constant in its position, and the number of screenshots to be generated.

TABLE 4.1 – Dataset Flight Gear airports' conditions used to generate the dataset with FG.

ICAO	**Location**	**Runways**	**Nearby**	**Dataset**
BIKF	Keflavik, Iceland	2, cross	city, water	Training, validation
SDRJ	Rio de Janeiro, Brazil	2, parallel	city, water, mountain	Training, validation
SBEG	Manaus, Brazil	1	forest	Training, validation
LFPG	Paris, France	2, parallel	city	Training, validation
HECA	Cairo, Egypt	1	isolated	Training, validation
SBGL	Rio de Janeiro, Brazil	2, afar	city, water	Training, validation
VVTS	Ho Chi Minh, Vietnam	2, parallel	city	Training, validation
SBGP	Gavião Peixoto, Brazil	1	farmland	Training, validation
SBLO	Londrina, Brazil	1	city, forest	Test

TABLE 4.2 – Dataset Flight Gear airports' orientation data used to generate the dataset with FG.

ICAO	**Runway**	**True heading**	**Magnetic to true**	**Screenshots**
BIKF	28	270	-10	1000
SDRJ	02R/L	0	-22	1000
SBEG	28	270	-10	1000
LFPG	26R/L	265	+5	1000
HECA	23L	229	-1	1000
SBGL	28	260	-22	1000
VVTS	25R/L	250	0	1000
SBGP	20	180	-22	1000
SBLO	31	290	-20	2000

4.3 Python interface

First of all, we needed a way to easily control the position and orientation of the camera inside the simulation. We used the FlightGear.py repository, which can establish a TELNET stream to the simulator and alter, in runtime, any listed "internal properties" listed. For our goal, the relevant internal properties are:

- `/position/longitude-deg`: variable for longitude positioning in degrees.
- `/position/latitude-deg`: variable for latitude positioning in degrees.
- `/position/altitude-ft`: variable for altitude (MSL) positioning in feet.
- `/orientation/heading-deg`: variable for heading orientation in degrees.
- `/orientation/pitch-deg`: variable for pitch orientation in degrees.
- `/orientation/roll-deg`: variable for roll orientation in degrees.

With those, we can control the camera successfully to take pictures of the runway in any possible pose.

4.4 Simulation control and modding

Some angles and distances would have no value for training, as the picture would not cover all runway corners depending on them, so we limited the camera position to be inside a closed conic region with its vertex in the middle of the threshold bars (runway head). The cone has an elliptical base due to the vertical angle being different from the horizontal angle. We created a Python module to generate a uniform random distribution in spherical coordinates (r = distance to the runway head, ϕ – horizontal angle, and θ – vertical angle) inside the limiting cone (see Figure 4.1). With the distribution vector, another module transforms from spherical to latitude, longitude, and altitude vectors, using the airport's runway head's coordinates as input.

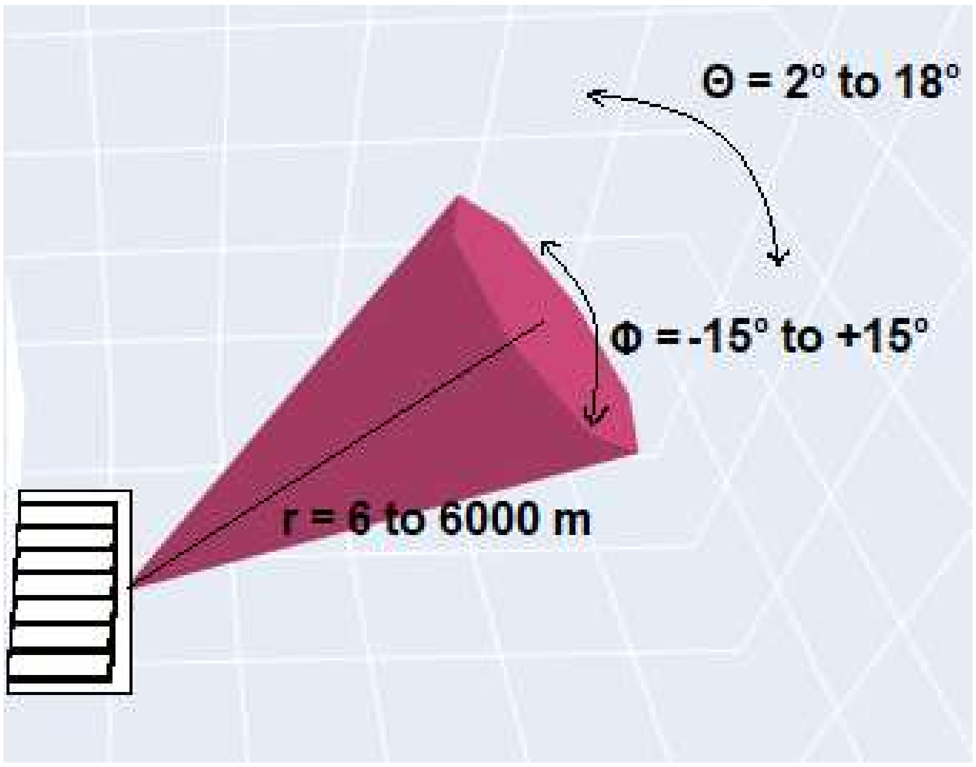


FIGURE 4.1 – Illustration of the generated vector cone with respect to the runway.

Heading and pitch angles were calculated from each given latitude, longitude and altitude, in order to point at the runway and reduce the number of screenshots containing it partially. Roll angles were generated randomly from −20 to 20 degrees.

4.5 Screenshot generation

We created a screenshot module, which will go through all indexes of the previously explained coordinates vector, and, for each position, will take a screenshot (see Figure 4.2) and save it to a reserved folder in the system. Given that each screenshot takes approximately a second to be taken, at this point, we can have a theoretically unlimited number of different un-annotated pictures in the database if enough time can be expended on the screenshot sequences.

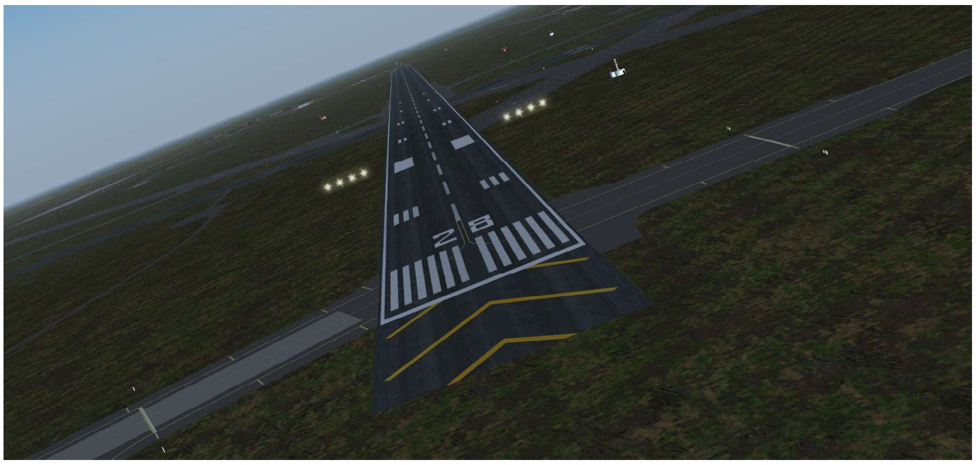


FIGURE 4.2 – Example screenshot taken from the Flight Gear simulator.

4.6 Annotation

The manual annotation for keypoint detection in a huge number of images would greatly limit the work's efficiency due to the time and effort needed to manually annotate an image. As such, one of the goals of this work is to make an automatic annotator.

4.6.1 Detectron2 dataset format

As seen on Detectron2 Documentation (DETECTRON2, 2021a), the annotation for keypoints consists on a list of floats with the following data for each picture:

$$[x_1\ y_1\ v_1\ x_2\ y_2\ v_2\ ...\ x_i\ y_i\ v_i],$$

where x_i, y_i and v_i are the width coordinate, height coordinate and the visibility of the i^{th} keypoint on the image (visibility 0 equals to not visible on the image). Each runway in an image has its own keypoints list.

The main purpose of the annotator is to extract the keypoints data: in our case, the keypoints will be the four corners of a runway. Then, the annotation code needs to somehow extract the (x, y) coordinates of each corner of the runway, for each image, to label the dataset.

4.6.2 Differential annotator

First, in the FG environment, we take all screenshots from the given airport as described in the screenshot generation section. Now, given that the random camera position vector used to generate those screenshots is saved, a new screenshot sequence will be taken. This time, using the FG Nasal console, we put a green sphere in each of the four corners of the runway (see Figure 4.3) as an artificial object in the FG simulated environment, and take screenshots once again.

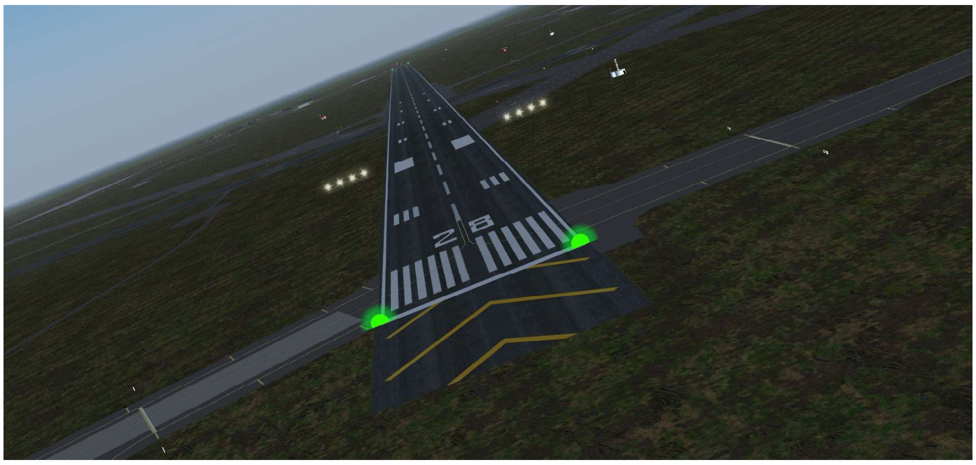


FIGURE 4.3 – Annotated image pair for the example in Figure 4.2.

To correctly extract the corners, we used the process indicated in Figure 4.4. The differential annotator is a function based on a work posted by Rosebrock (2017) that takes the differences between each pair of clean and annotated screenshots and returns the center of each of the differences (which will be the center of the objects put in the corners), thus constructing the keypoint dictionary needed for Detectron2.

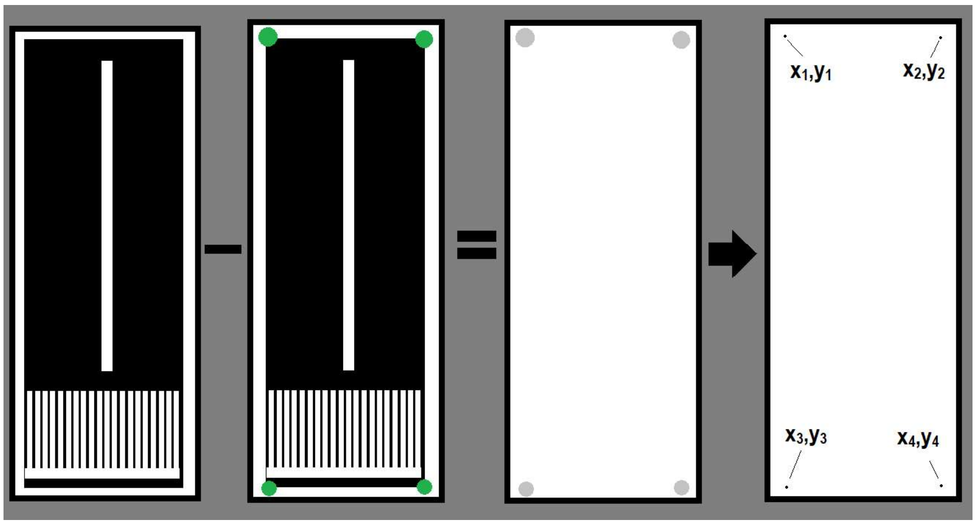


FIGURE 4.4 – The two steps performed by the annotator: differentiation between clean and annotated image, followed by the extraction of the center point of the four difference zones.

After the extraction, a keypoint sorter is used to efficiently put the keypoint data in the right order: bottom-left, bottom-right, top-left, top-right corners.

Without the construction of this automatic annotation code, the feasibility of the project would be severely affected, as the manual annotation for a decent amount of images, to train effectively, could take a high amount of effort and time.

4.7 Dataset registering

With the annotator, it is possible to register the training, validation, test and real datasets. In Detectron2's default configuration, it is divided into Dataset and Metadata Catalogs.

- Dataset Catalog: contains the images' location, their size in pixels and their bounding box, keypoints or segmentation annotation data.
- Metadata Catalog: contains common information and parameters about the dataset. In this case, the number and names of classes, the number, and names of keypoints, flip map (which keypoints are swapped in case of an image flip for data augmentation), and connection rules (which keypoints must be connected by lines).

For ease of training, we decided to only use images with runways having all four corners visible. For that, a dataset filter was implemented, in order to only register images with a number of keypoints multiple of 4. The process is illustrated in Figure 4.5.

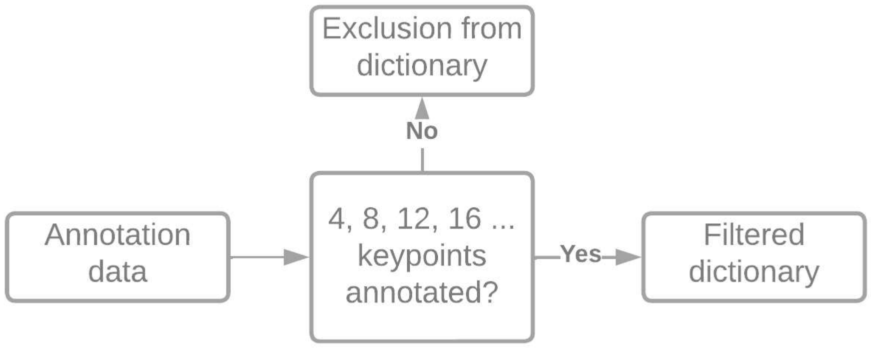

Exclusion from dictionary
No
Annotation data
4, 8, 12, 16 ... keypoints annotated?
Yes
Filtered dictionary

FIGURE 4.5 – Block diagram of image exclusion from the dataset due to the keypoints' visibility.

This process, using the airport images, generates the training, validation and test datasets. Lastly, there is the real dataset with 27 images taken from the internet and manually annotated. Its size is not enough for a trustable statistical outcome, but it may be used to check the existence of benefits from virtual training, as well as virtual overfitting.

4.8 Training and evaluation

With the dataset and annotation information, the training may proceed, using, at first, the default parameters from Detectron2 trainer engine tutorial, with some changes shown in Table 4.3 and data augmentation in Table 4.4

TABLE 4.3 – Detectron2 configuration.

Attribute	**Value**
`MODEL.MASK_ON`	False
`MODEL.KEYPOINT_ON`	True
`SOLVER_IMS_PER_BATCH`	1
`TEST.DETECTION_PER_IMAGE`	2
`MODEL.ROI_KEYPOINT_HEAD.NUM_KEYPOINTS`	4
`SOLVER.BASE_LR`	0.00025
`SOLVER.MAX_ITER`	100,000
`MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE`	64
`MODEL.ROI_HEADS.NUM_CLASSES`	1
Architecture and starting weights	COCO R-CNN 50 layers

TABLE 4.4 – Data augmentation used in training.

Augmentation	**Properties**
Random Brightness	0.6 to 1.4
Random Flip	40% probability, horizontal only
Random Contrast	0.7 to 1.3

Other augmentations, like Random Cropping, were excluded due to the chance of removing keypoints from the image in the process. No learning rate decay was applied. The total loss and keypoint loss function curves will be evaluated on training and validation sets on every 20 epochs. Next, a second training session will be made with the losses being evaluated on the real dataset.

The next step is to load the trained model obtained on early-stopping and to use the COCO Evaluator, obtaining AP metrics for bounding boxes and keypoint prediction on the Test dataset. In order to correctly calculate OKS, the keypoint weights are `TEST.KEYPOINT_OKS_SIGMAS` $= [1\ \ 1\ \ 1\ \ 1]$, which means all four keypoints have full weight on metrics.

4.9 Pose estimator

On the test dataset, we are going to use a Perspective N-point (PNP) algorithm to estimate aircraft camera coordinates and orientation related to the world coordinates. For a correct analysis, the cone coordinates in Section 4.4 were stored and to be compared against the estimate of the algorithm. The world coordinates used will have its origin at the bottom-right corner of the runway; The axes x, y and z are aligned to the Earth longitude, Earth latitude and the local vertical, respectively, as seen in Figure 4.6.

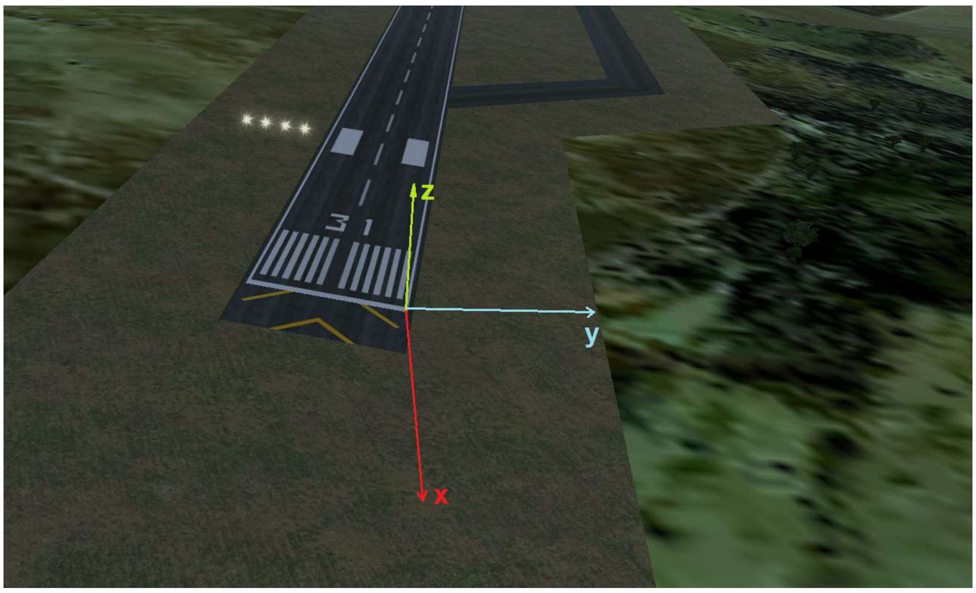


FIGURE 4.6 – The world coordinate axes placement on SBLO runway.

The algorithm uses the prediction of keypoints in the test images, indirectly contributing with other particular metrics for keypoints other an AP_{OKS}. As example, positional errors, obtained by substracting the ground truth of x, y, z from the estimative provided by the algorithm; and orientation, analogous to positional error, using heading, pitch and yaw angles, predicted and ground truth.

The framework used for PNP was OpenCV, specifically `solvePnP` and `Rodrigues` functions. Table 4.5 shows used data for algorithm.

TABLE 4.5 – Arguments passed to PNP algorithm.

Argument	**Value**	**Obs.**
Camera Field of View	55 [deg]	Diagonal
Image size	1900 x 892	W x H
Distortion coefficients	[0000]	–

5 Results

The results to be presented were obtained using the fully annotated datasets (exactly 4 or 8 visible keypoints) taken around the airports' scenery. Table 5.2 presents how the data was split between the datasets. Before filtering out images (explained in Section 4.7), we had 8000 images (excluding SBLO airport). 7330 of those compose the training set, while the remaining 670 became the validation set by random selection. The two thousand SBLO pictures compose the test dataset, and 27 real runway images make the real dataset.

TABLE 5.1 – Datasets' composition

Dataset	**Before**	**Excluded**	**After**
Training	7330	920	6410
Validation	670	84	586
Test	2000	14	1986
Real	27	1	26

After filtering to keep images with exactly 4 or 8 visible keypoints, as explained in Section 4.7, the training, validation, test and real sets lost 920, 84, 14 and 1 image(s), respectively. The total training time until 71,000 epochs, which was the early-stopping trigger, was approximately 1 day 17 hours and 38 minutes using a Tesla P100 GPU on Google Colab.

5.1 Loss functions

5.1.1 Virtual-virtual analysis

During training and for every 20 epochs, total and keypoint losses were calculated, and results are shown in Figures 5.1 and 5.2, respectively. We can see that their behavior is very similar, and, at the point around 70,000 epochs, further iterations would not give considerable benefits towards validation dataset losses, indicating proximity of the overfitting point.

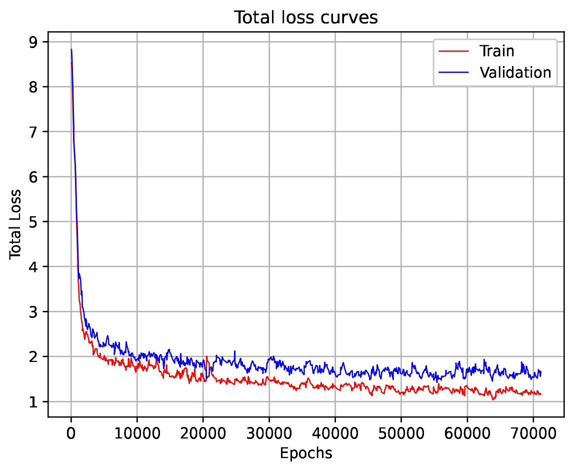


FIGURE 5.1 – Total loss evolution for training and validation datasets.

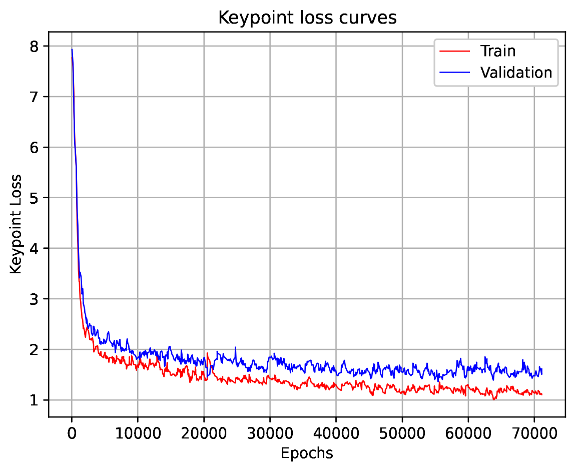


FIGURE 5.2 – Keypoint loss evolution for training and validation datasets.

5.1.2 Virtual-real analysis

Following the same methodology, total and keypoint losses on the real dataset were also calculated every 20 epochs, and its results are shown on Figures 5.3 and 5.4, respectively, comparing to the training dataset losses. As expected, the training had a very clear overfitting point which happens around 2,000 epochs. Prior to that, there is a decrease in both keypoint and total losses for the real dataset, indicating that a virtual dataset may be used in a pretraining process in order to reduce costs and increase agility in development; After virtual overfitting, the loss functions for the real dataset increase slowly. That behavior remains during the 71,000 epochs. Qualitative differences between 2000 and 71000 epochs trained models on real runways will be shown later.

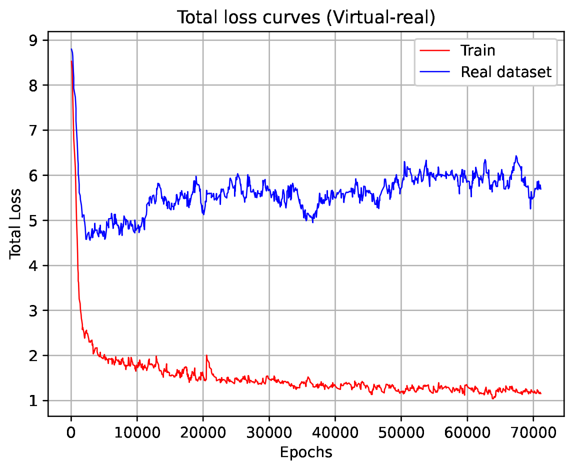


FIGURE 5.3 – Total loss evolution for training and real datasets.

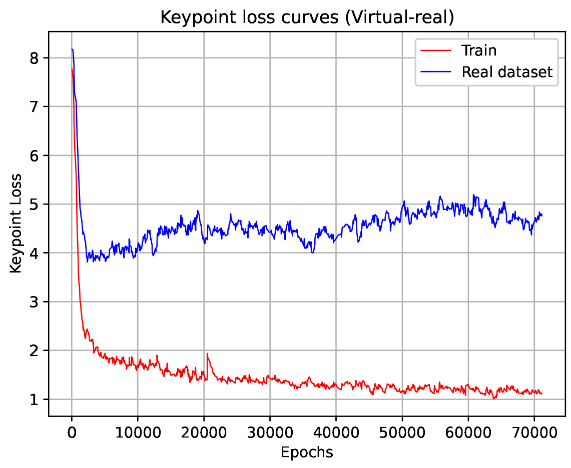


FIGURE 5.4 – Keypoint loss evolution for training and real datasets.

5.2 Evaluation

5.2.1 Test dataset

Using the final model weights, the COCO Evaluator provided AP metrics for bounding boxes and keypoints shown in Table 5.2. These metrics are considered standard metrics for evaluating object detection and keypoint detection models in the academy.

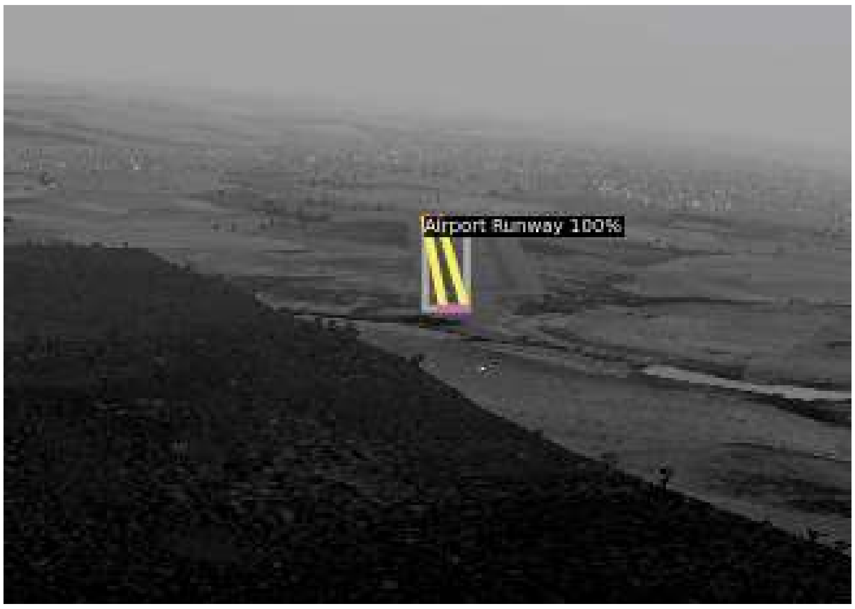
TABLE 5.2 – Evaluation metrics from prediction on the test dataset.

Bounding boxes	
Metric	**Value**
AP	81.5
AP50	93.6
AP75	89.6
APs	37.8
APm	71.1
APl	86.5

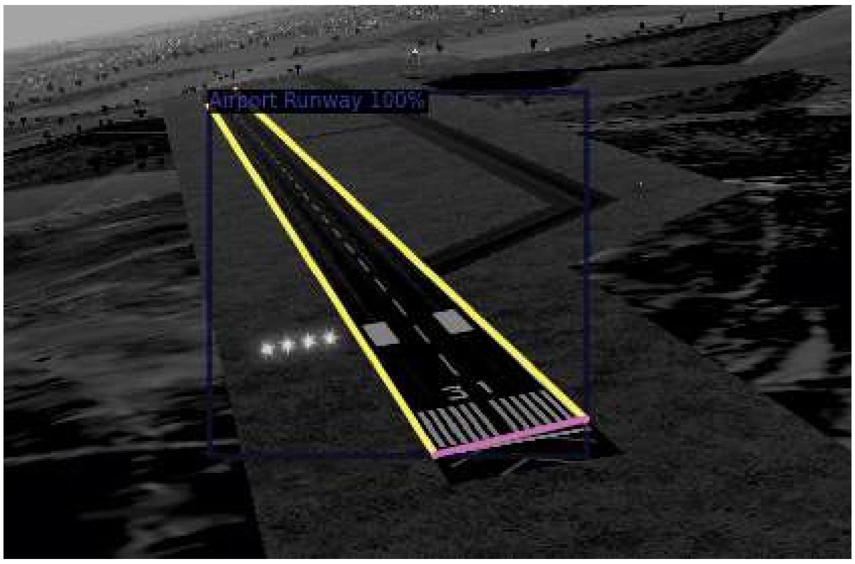
Keypoints	
Metric	**Value**
AP	94.1
AP50	96
AP75	94.7
APm	89.3
APl	96.6

The results indicate that the network has successfully trained over virtual runway detection, with an average precision of 81 % for bounding box and 94 % for keypoints. The high AP metrics when compared to current benchmarks can be explained by the fact that our network is trained for only one class (airport runway). Nonetheless, as examples seen in Figure 5.5, inference provided solid results for near and far images. The keypoints' placement was assertive and bounding box correctly contained the runway in the image.

Some images came out with peculiar detection results, as seen in Figure 5.6. The example (c) can be explained by erroneus annotation on training set, which trained the network to detect a close shot of threshold bars as runways. Figure 5.7 shows an example of a bad annotation which could have led to these individual cases. All four runway corners were annotated around runway threshold bars.

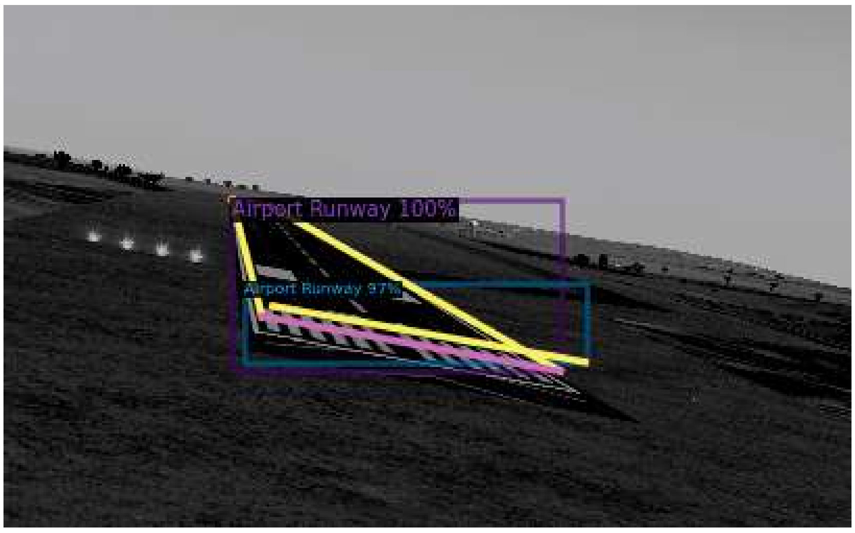


(a) Valid result by inference on the test dataset runway image from distance.

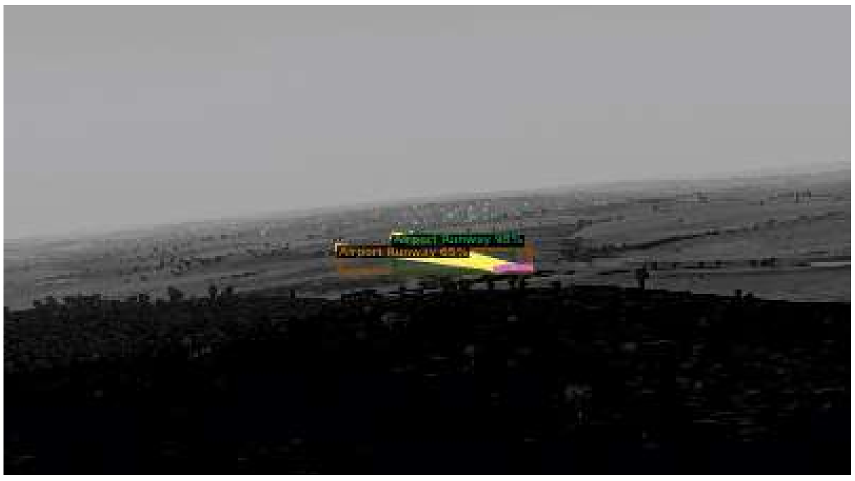


(b) Valid result by inference on the test dataset runway image closely.

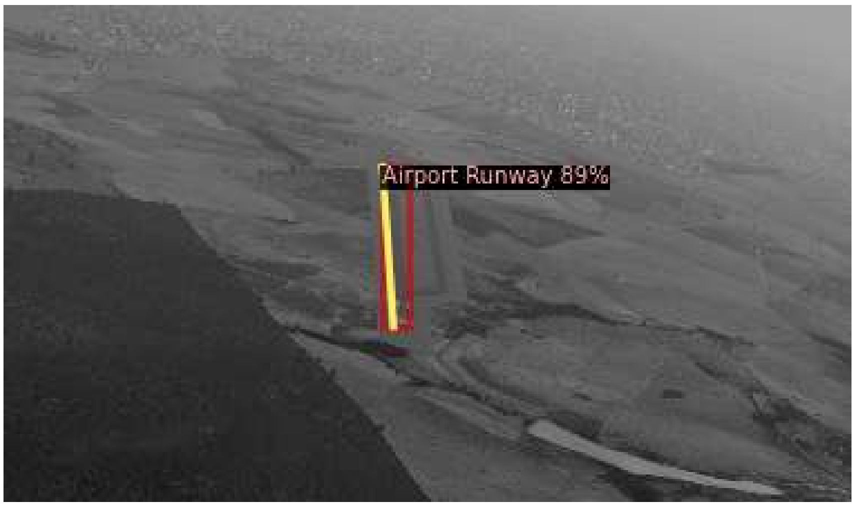
FIGURE 5.5 – Good detection results on the test dataset.



(a) Invalid result on the test dataset showing wrong detection.



(b) Invalid result on the test dataset showing wrong detection.



(c) Invalid result on the test dataset showing wrong keypoint placement.

FIGURE 5.6 – Wrong results on the test dataset.

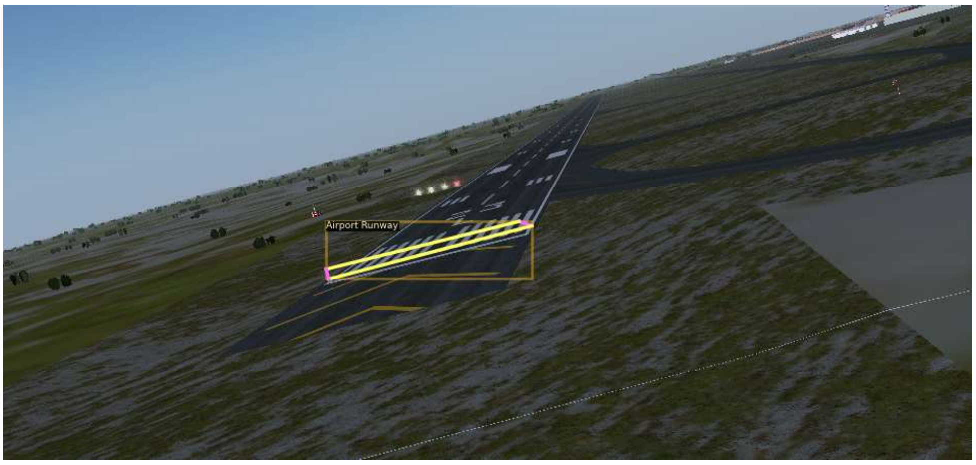


FIGURE 5.7 – Image with bad unfiltered annotation on the training dataset.

Aside the wrong detections, a visual analysis on many images, reinforced by the AP metrics, show that even when keypoints are misplaced or undetected, bounding box predictions are valid and well placed. If this characteristic maintains itself in more advanced works involving real images, it could help aircraft guidance when approaching runway and until it can have a clear enough view of the runway to begin valid keypoint inference. This hypothesis also reinforces the explained idea of data fusion with other guidance systems, as they can be used to properly guide the aircraft towards the runway until the CVS is close enough to output quality results.

5.2.2 Real dataset

Inference on the real dataset was made with the 2000 and 71000 epochs models. As seen in Figures 5.8 to 5.17.

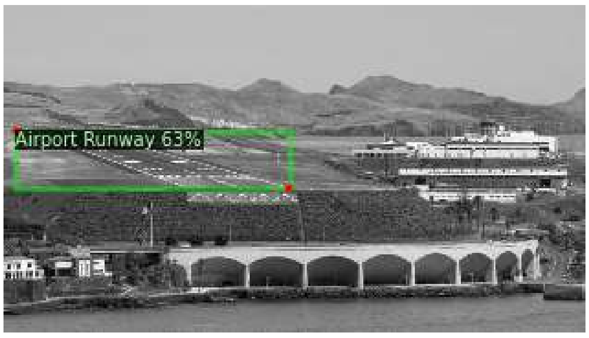


FIGURE 5.8 – Detection with different epochs models (2000).

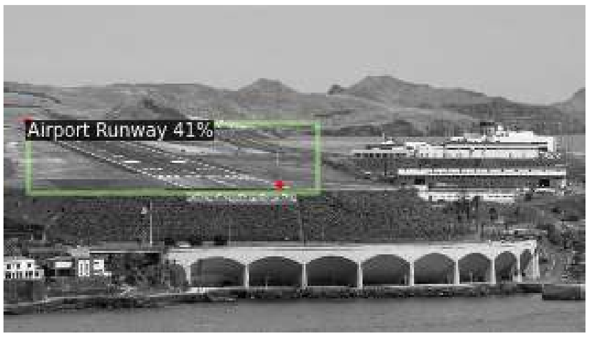


FIGURE 5.9 – Detection with different epochs models (71000).

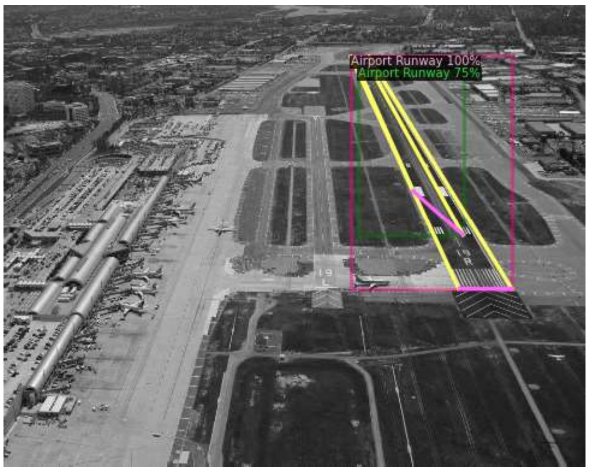


FIGURE 5.10 – Detection with different epochs models (2000).

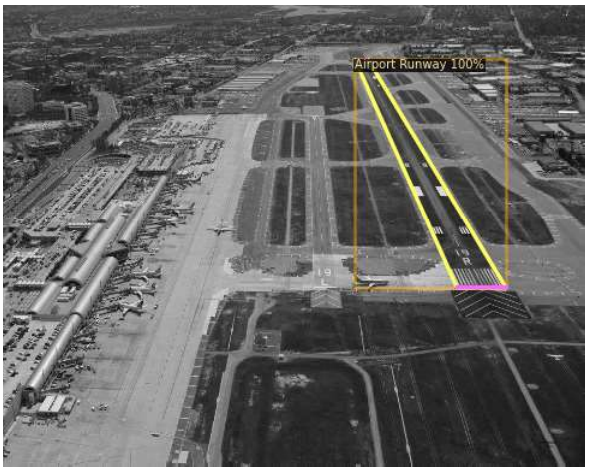


FIGURE 5.11 – Detection with different epochs models (71000).

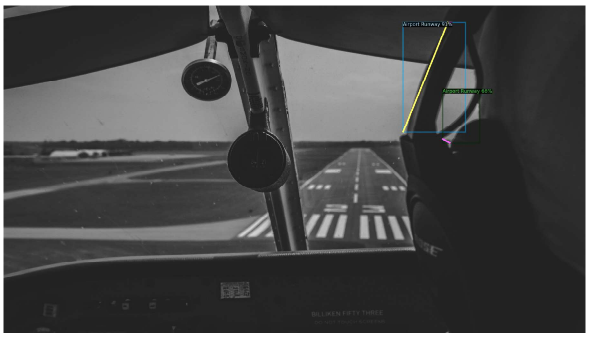


FIGURE 5.12 – Detection with different epochs models (2000).

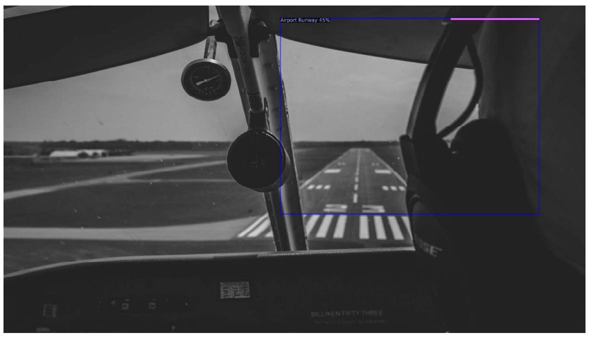


FIGURE 5.13 – Detection with different epochs models (71000).

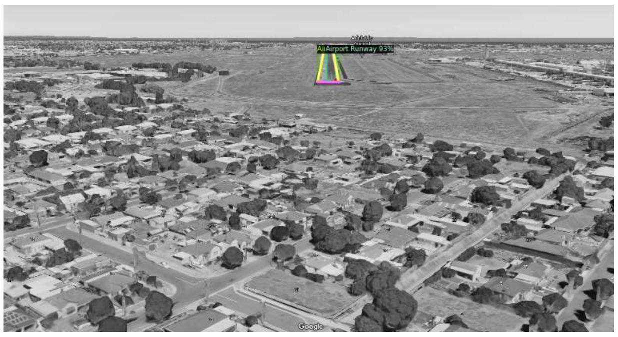


FIGURE 5.14 – Detection with different epochs models (2000).

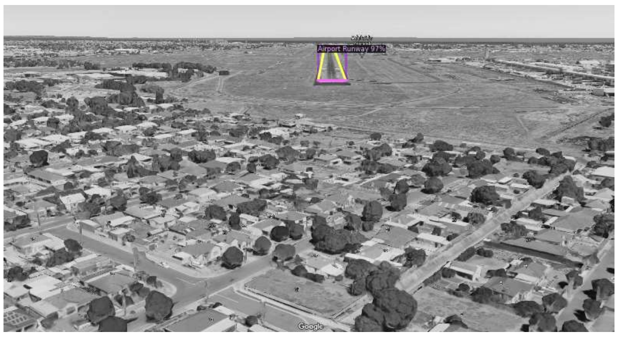


FIGURE 5.15 – Detection with different epochs models (71000).

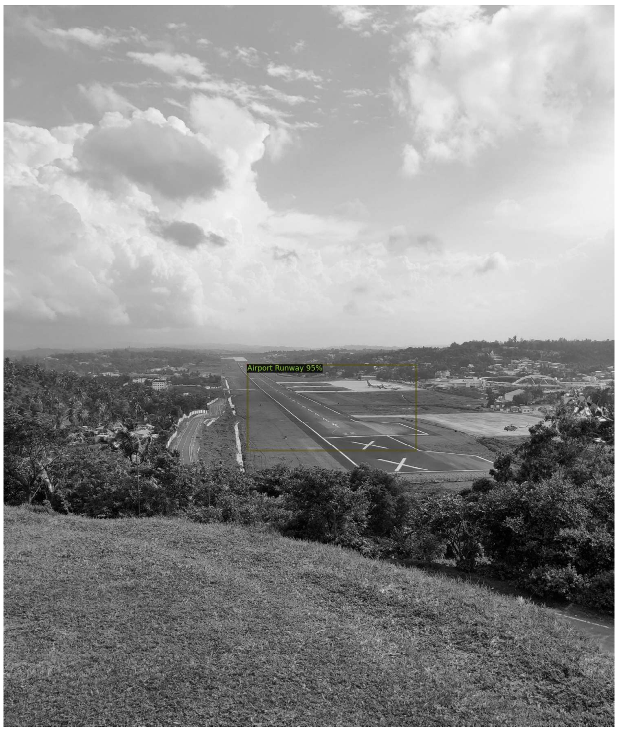


FIGURE 5.16 – Detection with different epochs models (2000).

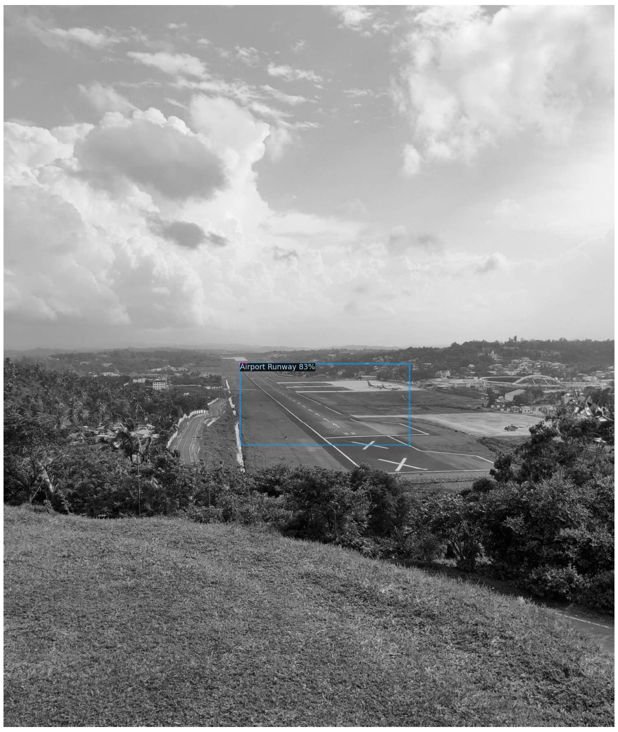


FIGURE 5.17 – Detection with different epochs models (71000).

For both models in a detection threshold of 0.2, runways were detected in 14 of 26 dataset images. As we can see, the accuracy of keypoints' placement for the most trained model is better on those images. But, among those images with detections, some wrong detections cases occurred in both bounding box and keypoints, as seen in Figures 5.12 and 5.13. Especially in Figures 5.16 and 5.17, the lack of keypoints' placement is probably explained by the absence of runway markings. The network could have trained itself to predict keypoint placement from threshold bars and/or aiming markings.

5.3 Pose estimation

The PNP algorithm used the keypoints' inference results for the whole test dataset. The metrics for positional and orientation predictions are shown on Table 5.3

TABLE 5.3 – Error metrics for pose estimation.

Metric	**x [m]**	**y [m]**	**z [m]**	**Hdg [deg]**	**Pitch [deg]**	**Roll [deg]**
Mean abs. error	$2.91 \cdot 10^6$	$3.13 \cdot 10^6$	$6.56 \cdot 10^6$	11.58	2.10	12.35
Std. deviation	$73.6 \cdot 10^6$	$94.6 \cdot 10^6$	$222 \cdot 10^6$	40.03	4.93	35.50

These values were not expected, as they are far superior to the generated cone maximum distance (6000 meters). A histogram of Euclidean distance error was plotted (see

Figure 5.18), as many outliers could contribute to those unreliable metrics.

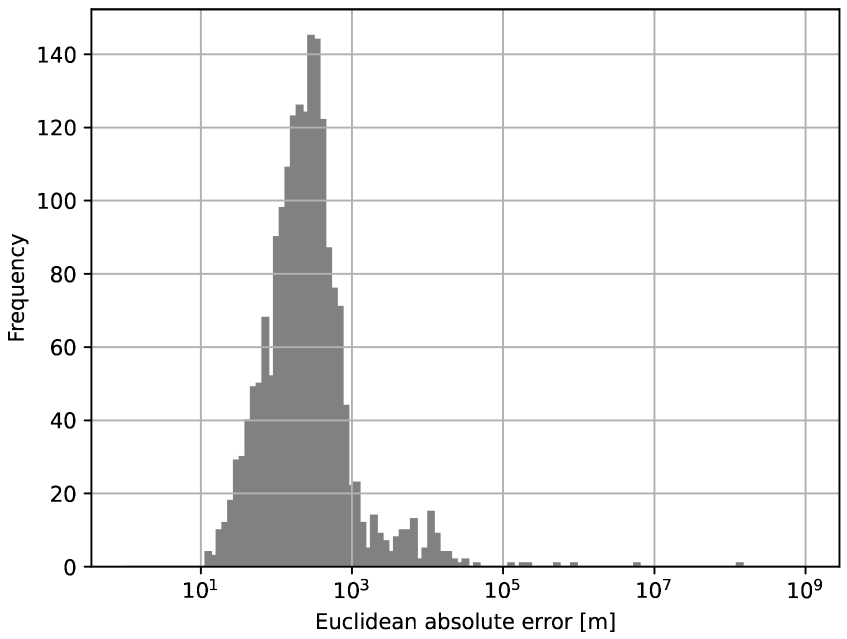


FIGURE 5.18 – Histogram of Euclidean distance absolute errors.

Dealing with the outliers was a must to proceed with analysis. We then removed errors above one tenth of the generated cone maximum distance (600 m) from the population, as these errors could be automatically filtered out by data fusion with other sensors in real application. The filtering process changed the metrics and the histogram, as shown in Table 5.4 and Figure 5.19.

TABLE 5.4 – Error metrics for pose estimation (filtered)

Metric	**x [m]**	**y [m]**	**z [m]**	**Hdg [deg]**	**Pitch [deg]**	**Roll [deg]**
Mean absolute error	173.50	54.75	37.55	1.68	1.09	2.80
Standard deviation	121.44	42.20	31.94	1.08	0.56	1.91

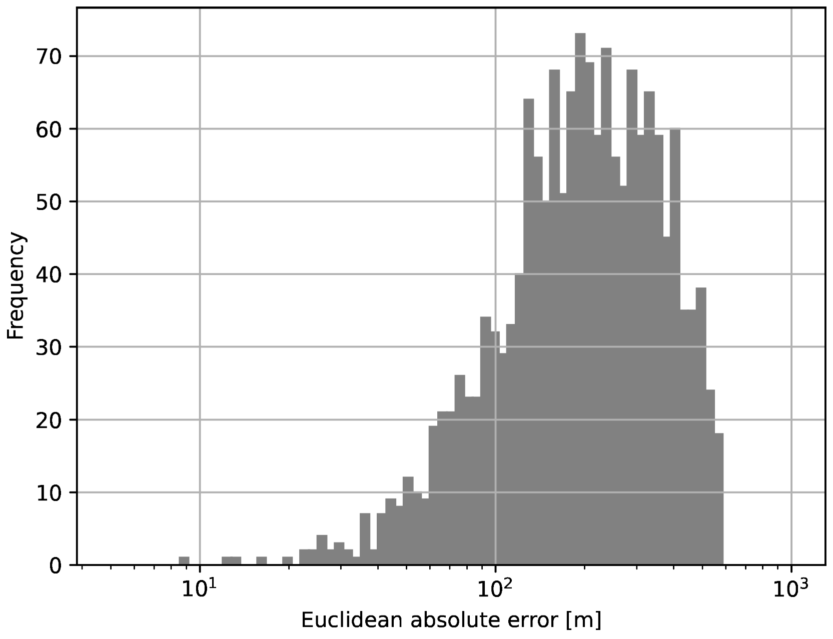


FIGURE 5.19 – Histogram of Euclidean distance absolute errors (filtered).

A mean error on axis of around 200 meters can be acceptable for a proof of concept work, meaning that further development could push values to a better result. With that in mind, Figure 5.20 shows the per-axis absolute errors distribution according to the ground truth Euclidean distance from the runway; And, by combining the positional errors, Figure 5.21 shows the Euclidean distance errors related to ground truth runway distance.

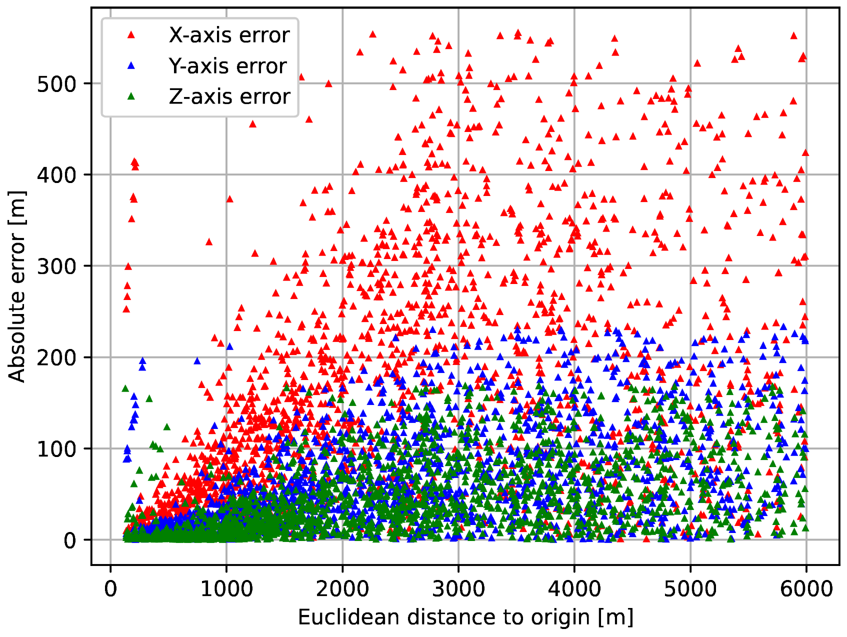


FIGURE 5.20 – Per-axis errors distribution related to distance from runway (filtered).

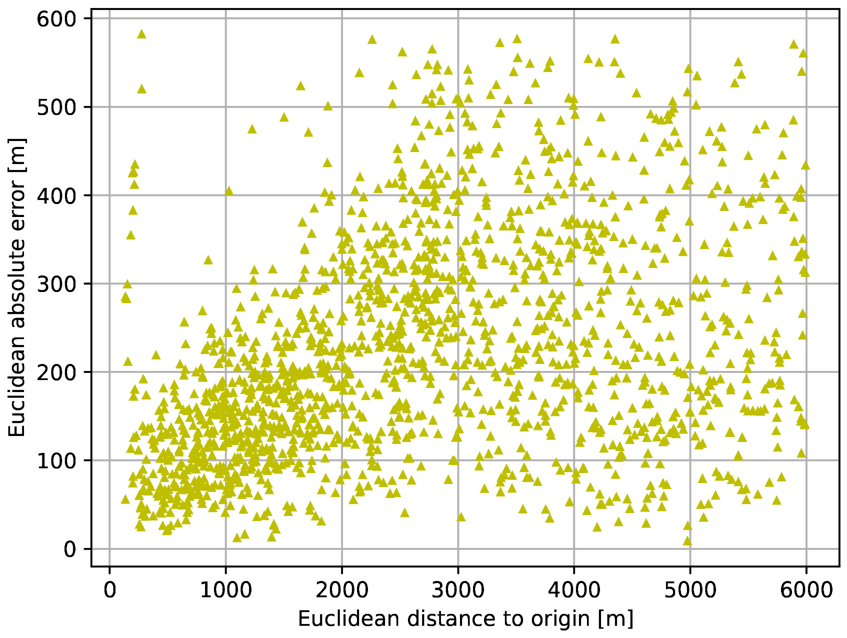


FIGURE 5.21 – Euclidean distance errors distribution related to distance from runway (filtered).

The data reveals that the network becomes more robust as the aircraft gets closer, as all three axis' errors increase with distance to the runway. Also, the prediction for height seems more accurate than latitude and longitude predictions. In terms of orientations, Figure 5.22 shows the histogram for the heading, pitch and roll predictions' distribution, respectively. It seems that the orientation inference is more accurate than positioning, with most of all three distributions concentrating below 5 degrees. We conclude that a keypoint prediction network may very well supply pose estimation algorithms in practical applications.

Bibliography

ABU-JBARA, K.; SUNDARAMORTHI, G.; CLAUDEL, C. Fusing vision and inertial sensors for robust runway detection and tracking. **Journal of Guidance, Control, and Dynamics**, v. 41, n. 9, p. 1929–1946, 2018. Available from Internet: <https://doi.org/10.2514/1.G002898>.

Aerocorner. **How Airplanes Changed The World on Every Level**. 2022. Available from Internet: <https://aerocorner.com/blog/how-airplanes-know-where-to-go/>. Date of access: 11 jul. 2022.

Aerocorner. **How Airplanes Know Where To Go (History of Aviation Navigation)**. 2022. Available from Internet: <https://aerocorner.com/blog/how-airplanes-know-where-to-go/>. Date of access: 11 jul. 2022.

AIRBUS. **Developing autonomous flight and machine learning solutions for the next generation of aircraft**. 2021. Available from Internet: <https://acubed.airbus.com/projects/wayfinder/>. Date of access: 19 jul. 2021.

Aircraft Systems Tech. **History of Avionics**. 2022. Available from Internet: <https://www.aircraftsystemstech.com/2017/05/communication-and-navigation.html>. Date of access: 11 jul. 2022.

AKTAS, Y. C. **Object Detection with Convolutional Neural Networks**. 2022. Available from Internet: <https://towardsdatascience.com/object-detection-with-convolutional-neural-networks-c9d729eedc18>. Date of access: 11 jul. 2022.

Brazil Defense Ministry. **Brazil National Defense Strategy**. 2022. Available from Internet: <https://www.gov.br/defesa/pt-br/assuntos/copy_of_estado-e-defesa-/estrategia-nacional-de-defesa>. Date of access: 24 jun. 2022.

BURNS, E. **Machine Learning**. 2022. Available from Internet: <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>. Date of access: 11 jul. 2022.

CROUCH, T. D. **Avionics, passenger support, and safety**. 2022. Available from Internet: <https://www.britannica.com/technology/history-of-flight/Avionics-passenger-support-and-safety>. Date of access: 11 jul. 2022.

DAEDALEAN. **Land safely! Daedalean's Visual Landing System**. 2021. Available from Internet: <https://daedalean-ai.cdn.ampproject.org/c/s/daedalean.ai/tpost-/inxg7y7671-land-safely-daedaleans-visual-landing-sy>. Date of access: 19 jul. 2021.

DAEDALEAN. **Neural Network Based Runway Landing Guidance for General Aviation Autoland**. [S.l.], 2021.

Daedalean and EASA. **Concepts of Design Assurance for Neural Networks (CoDANN) II**. [S.l.], 2021. 112 p.

DETECTRON2. **Detectron2 Documentation**. 2021. Available from Internet: <https://detectron2.readthedocs.io/en/latest/>. Date of access: 19 jul. 2021.

DETECTRON2. **Detectron2's Github Repository**. 2021. Available from Internet: <https://github.com/facebookresearch/detectron2>. Date of access: 19 jul. 2021.

European Space Agency. **GNSS Augmentation**. 2011. Available from Internet: <https://gssc.esa.int/navipedia/index.php/GNSS_Augmentation>. Date of access: 11 jul. 2022.

Federal Aviation Administration. **Airplane Flying Handbook**. [S.l.], 2016. 348 p.

FLIGHTGEAR. **Flight Gear Main Website**. 2021. Available from Internet: <https://www.flightgear.org>. Date of access: 19 jul. 2021.

GARMIN. **What is GPS?** 2022. Available from Internet: <https://www.garmin.com/en-US/aboutgps/>. Date of access: 11 jul. 2022.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2014.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`.

HE, K.; GKIOXARI, G.; DOLLAR, P.; GIRSHICK, R. Mask r-cnn. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. **CoRR**, abs/1512.03385, 2015. Available from Internet: <http://arxiv.org/abs/1512.03385>.

HO, N.; CHAKRAVARTY, P. Localization on freeways using the horizon line signature. In: . [S.l.: s.n.], 2014.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080. Available from Internet: <https://www.sciencedirect.com/science/article/pii/0893608089900208>.

International Virtual Aviation Organisation. **Area Navigation - RNAV - Systems**. 2022. Available from Internet: <https://mediawiki.ivao.aero/index.php?title=Area_Navigation_-_RNAV_-_Systems>. Date of access: 11 jul. 2022.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2012. v. 25. Available from Internet: <https://proceedings-.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

KUGLER, M.; MUMM, N.; HOLZAPFEL, F.; SCHWITHAL, A.; ANGERMANN, M. Vision-augmented automatic landing of a general aviation fly-by-wire demonstrator. In: . [S.l.: s.n.], 2019.

KUMAR, A. **Semantic Image Segmentation using Fully Convolutional Networks**. 2020. Available from Internet: <https://towardsdatascience.com/semantic-image-segmentation-using-fully-convolutional-networks-bf0189fa3eb8>. Date of access: 11 jul. 2022.

LAVANYA S. **Face Key-point Recognition Using CNN**. 2021. Available from Internet: <https://www.analyticsvidhya.com/blog/2021/07/face-key-point-recognition-using-cnn-/>. Date of access: 11 jul. 2022.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. In: **Proceedings of the IEEE**. [S.l.: s.n.], 1998. p. 2278–2324.

LEE, C.-h.; LEE, H.-H.; LEE, M.-j. Runway detection based on geometric constraints of runways for safe landing. **MATEC Web of Conferences**, v. 54, p. 08005, 01 2016.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; BOURDEV, L.; GIRSHICK, R.; HAYS, J.; PERONA, P.; RAMANAN, D.; ZITNICK, C. L.; DOLLÃ¡R, P. **Microsoft COCO: Common Objects in Context**. arXiv, 2014. Available from Internet: <https://arxiv.org/abs/1405.0312>.

MICROSOFT. **Flight Simulator on Microsoft Store**. 2021. Available from Internet: <https://www.xbox.com/pt-BR/games/microsoft-flight-simulator>. Date of access: 19 jul. 2021.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

Phase 1 Technology Corporation. **Overcoming Aviation Challenges Using Computer Vision**. 2021. Available from Internet: <https://www.phase1vision.com/blog-/overcoming-aviation-challenges-using-computer-vision>. Date of access: 11 jul. 2022.

ROSCOE, A. H.; GRIEVE, B. S. The impact of new technology on pilot workload. **SAE Transactions**, SAE International, v. 95, p. 1179–1186, 1986. ISSN 0096736X. Available from Internet: <http://www.jstor.org/stable/44470625>.

ROSEBROCK, A. **Image Difference with OpenCV and Python**. 2017. Available from Internet: <https://www.pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python>. Date of access: 19 jul. 2021.

STUDYFLYING. **Instrument Landing System**. 2019. Available from Internet: <https://studyflying.com/wp-content/uploads/2019/03/localizer-antenna-location.gif>. Date of access: 19 jul. 2021.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Available from Internet: <http://arxiv.org/abs/1409.4842>.

UNITY. **Unity Main Website**. 2021. Available from Internet: <https://unity3d.com>. Date of access: 19 jul. 2021.

UNREAL. **Unreal Engine Main Website**. 2021. Available from Internet: <https://www.unrealengine.com>. Date of access: 19 jul. 2021.

VOLPE, J. A. Vulnerability assessment of the transportation infrastructure relying on the global positioning syst. In: . [S.l.: s.n.], 2001.

X-PLANE. **X-Plane 11 Main Website**. 2021. Available from Internet: <https://www.x-plane.com>. Date of access: 19 jul. 2021.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: FLEET, D.; PAJDLA, T.; SCHIELE, B.; TUYTELAARS, T. (Ed.). **Computer Vision – ECCV 2014**. Cham: Springer International Publishing, 2014. p. 818–833. ISBN 978-3-319-10590-1.

ZONGUR, U.; HALICI, U.; AYTEKIN, O.; ULUSOY, I. Airport runway detection in satellite images by adaboost learning. **Proceedings of SPIE - The International Society for Optical Engineering**, v. 7477, 09 2009.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DP	2. DATA 23 de agosto de 2022	3. DOCUMENTO Nº DCTA/ITA/DP-006/2022	4. Nº DE PÁGINAS 69

5. TÍTULO E SUBTÍTULO:

AIRPORT RUNWAY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

6. AUTOR(ES):

Victor André Lima

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Neural; Rede; Runway; Virtual; Overfitting; Landing; Pose

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Redes neurais; Processamento de imagens; Pistas (de pouso e decolagem); Simulação computadorizada; Banco de dados de imagens; Segurança de voo; Computação.

10. APRESENTAÇÃO: **(X) Nacional** **( ) Internacional**

ITA, São José dos Campos. Mestrado Profissional em Engenharia Aeronáutica e Mecânica. Orientador: Prof. Dr. Marcos Ricardo de Omena Albuquerque Maximo. Coorientadores: Prof. Dr. Ney Rafael Sêcco; Msc. Ney Ricardo Moscati. Defesa em 27/07/2022. Publicada em 2022.

11. RESUMO:

The recent advancement in computational power allowed the world to give more use of deep learning algorithms. The image processing task is on itself data-hungry, but showed to be of extreme importance in many fields of knowledge. In this work, a runway detection method, for aircraft during landing phase, based on convolutional neural network is made with the motivation to reduce pilot workload, increase flight safety during landing and take new steps towards single-piloted or even unmanned commercial flights. With the results, the network could successfully converge its loss function on the training dataset and validation dataset, and inference on several images from test dataset and real runway images gave good results for keypoint and bounding boxes predictions. Our numerical results indicate that virtual runway images can be used in order to pretrain a network to detect real runway images, so we contribute to a cheaper and faster approach on the development of such technology. Further analysis on estimatives for orientation and position of the aircraft camera from predicted keypoint indicate the work to be promising. Even being an initial work, the results gave a robust error histogram which could be easily controlled by filtering outliers and by data fusion with other sensors in real application. Overall, we contribute with results indicating that virtual training, using data augmentation methods for dataset enrichment, benefits real detection and that the keypoint predictions may be used together with pose estimation algorithm to give estimates of aircraft pose related to runway.

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) SECRETO**