MSc Autonomous Vehicle Dynamics and Control

Master Thesis Report:

SENSE AND AVOID USING HYBRID CONVOLUTIONAL AND RECURRENT NEURAL NETWORKS

Author:
Daniel Vidal Navarro

Supervisors:
Dr. Changhun Lee
Prof. Antonios Tsourdos

College Road, Cranfield University, Cranfield, Bedfordshire, MK43 0AL
August, 2018

Acknowledgements

Achievement of this project would not have been possible without the support of my supervisors, sponsors, family, friends and many others.

First of all, I would like to express my entire gratitude to my main supervisor Dr. Changhun Lee for his consistent guidance and support during the course. He was always available when needed, providing good advice and helping in everything he could.

Secondly, I would also like to thank my second supervisor Prof. Antonios Tsourdos. His ideas about where to redirect the thesis when doubts appeared were key.

I am grateful also to Prof. Nick Colosimo from BAE Systems, not only for giving me the opportunity to work on this fascinating topic but also for his time and support to set the requirements and for helping to find solutions to the specified problems.

Many thanks also go to my friends, who allowed me to forget about the hard work for some time so I could keep motivated for the whole duration of the thesis.

Of course, thanks to my family because they are always there in the background, sending all their energy so I can get the best results.

Finally, I also want to express my gratitude to Cranfield University for its valuable services and infrastructure, it has been a year full of learnings and experiences that eventually lead to this thesis result.

Abstract

A Sense and Avoid technique has been developed in this master thesis. A special method for small UAVs which use only an electro-optical camera as sensor has been considered. This method is based on a sophisticated processing solution using hybrid Convolutional and Recurrent Neural Networks. The aim is to study the feasibility of this kind of neural networks in Sense and Avoid applications.

First, the detection and tracking part of the algorithm is presented. Two models were used for this purpose: a Convolutional Neural Network called YOLO and a hybrid Convolutional and Recurrent Neural Network called Re^3.

After that, the collision avoidance part was designed. This consisted of the obstacle relative range estimation using an Extended Kalman Filter, the conflict probability calculation using an analytical approach and the geometric avoidance manoeuvre generation.

Both parts were assessed separately by videos and simulations respectively, and then an experimental test was carried out to integrate them. Measurement noise was experimentally tested and simulations were performed again to check that collisions were avoided with the considered detection and tracking approach.

Results showed that the considered approach can track objects faster than the most common computer vision methods based on neural networks. Furthermore, conflict was successfully avoided with the proposed technique. Design parameters were allowed to adjust speed and manoeuvres accordingly to the expected environment or the required level of safety.

The main conclusion was that this kind of neural network could be successfully applied to Sense and Avoid systems.

List of Figures

Contents

Chapter 1

Introduction

1.1 Background and motivation

Nowadays, Unmanned Aerial Vehicles (UAVs) are becoming a reality and they are starting to be introduced into daily life. One of their most critical points is safety. Since there are no human pilots inside UAVs, it is very important that they are capable to ensure safe flights even in unexpected situations such as when an obstacle appears in the trajectory. In manned vehicles, this is called See and Avoid so it has been called Sense and Avoid in unmanned vehicles. Sense and Avoid systems must be capable of detecting objects or dangerous situations and generating suitable actions in order to avoid any conflict or collision. UAVs must always be equipped with Sense and Avoid systems.

There are several sensors that are commonly used to sense the surrounding environment around a vehicle. These sensors include radars, lidars, electro-optical cameras and cooperative solutions such as TCAS or ADS-B. Usually, large vehicles have some of these sensors onboard to ensure a proper environment perception. However, there are also small UAVs that could be used for delivery or supply purposes, or as sacrificial decoys in a defence context. In this kind of vehicles, there exist additional limitations called SWaP limitations which are about Size, Weight and Power. As a result, compact electro-optical camera systems combined with a sophisticated processing solution is a potential candidate for a low SWaP and effective solution. With such systems, these vehicles should be capable not only to avoid other aircraft but also obstacles close to the surface where they are likely to be operating most of the time.

In parallel, neural networks have been experiencing a fast development in the past years. Due to advancements in processing units capabilities, neural networks are showing to be very useful in areas such as computer vision or speech recognition. Hence, it is

time to explore the capabilities that deep learning techniques could have in Sense and Avoid applications, what will be done in this project. In this case, a special kind of neural network will be studied: hybrid Convolutional and Recurrent Neural Networks. Even though Convolutional Neural Networks and Recurrent Neural Networks have been deeply studied separately, this hybrid architecture is very new and it could use the best characteristics of the other two to be successfully applied to Sense and Avoid systems.

1.2 Aims and objectives

A Sense and Avoid system considering an electro-optical camera to capture the environment and hybrid Convolutional and Recurrent Neural Networks as post-processing solution will be designed. The complete system will consist of the obstacle detection and tracking, a conflict evaluation procedure and an avoidance manoeuvre generation. Due to the novelty of the considered processing approach, the main aims of the project were to analyse the feasibility of using this new kind of neural networks in Sense and Avoid applications for UAVs, to identify possible issues that might occur and to suggest future directions in its research. In order to achieve these aims, the following objectives were considered:

- Developing a detection and tracking algorithm with hybrid Convolutional and Recurrent Neural Networks.
- Assessing the situation of conflict with any detected obstacle.
- Proposing an avoidance manoeuvre to use in case it is required.
- Assessing the performance of the designed technique.
- Discussing advantages, limitations and research gaps found.

1.3 Thesis layout

The structure of this report is organised as follows. First, a literature review covering Sense and Avoid systems and subsystems and deep learning post-processing solutions is included. After that, the designed method is explained. It was divided into two main parts: object detection and tracking, and collision avoidance. Then, first the object detection and tracking algorithm will be described including a discussion of the obtained results. Following that, the collision avoidance technique will be considered, covering the performance assessment as well by means of simulations. A experimental

test was carried out too so its description and results are shown next. Finally, the closing remarks with the extracted conclusions and proposals for future research are included.

Chapter 2

Literature review

2.1 Sense and Avoid requirements

UAV Sense and Avoid is the artificial system equivalent of a human pilot detecting and avoiding hazard situations (mid-air collisions, controlled flight into terrain, flightpath obstacles, and clouds) following both the rules of the air, See and Avoid aircraft collision avoidance rules and Visual Flight Rules (VFR) where appropriate [15]. In See and Avoid circumstances, the pilot must be able to avoid collisions even without the help of other existing infrastructure, air traffic services or onboard surveillance equipment [16]. UAV operation requires an Equivalent Level of Safety (ELOS) to existing manned air vehicles, hence, Sense and Avoid systems must have at least the same See and Avoid capabilities.

In fact, Sense and Avoid is required not only to avoid collisions but also to ensure the self-separation so the conflict does not occur [17]. Self-separation reduces the possibility of conflict and collision avoidance is executed when a safe separation can no longer be sustained. The Collision Volume (CV) is usually defined as a cylindrical volume centred on the vehicle with 500 feet of radius and 200 feet of height. Any object inside this volume is considered as a Near Mid-Air Collision (NMAC). Objects outside the Collision Volume may not be treated as NMAC but may do as threats due to having some effect on the UAV such as the vehicle ownship. Hence, that is the reason to consider a self-separation distance as well as a collision avoidance [17].

Sense and Avoid can usually be divided into three phases [15]. The first one is the strategic sense and avoid, which involves long range planning to avoid potential conflicts. The second phase consists of conflict resolution advisories and it would provide recommendations to the pilot in the case that there is one. The last phase would be the autonomous collision avoidance in which the UAV should be capable to avoid collisions

by its own. In this project, completely autonomous vehicles are considered so the last phase will be designed.

The fully Sense and Avoid procedure can also be dissected into different functions which have commonly been Detect, Decide and Act (DAA) or Observe, Orient, Decide and Act (OODA) [16]. However, for collision avoidance techniques, the commonly three phases are referred as Detection (Detect or Observe and Orient), Avoidance (Decide and Act) and Resolution. In the next sections, each one of these functions will be developed.

2.2 Object detection and tracking

The trajectory of any vehicle would always be safe if it does not encounter any external agent. However, it is possible that the predefined trajectory crosses an unexpected structure or that two trajectories from different agents such as other vehicles, birds or any other object intersect. Hence, sensing equipment is required in every vehicle in order to detect any kind of object and avoid any collision.

2.2.1 Types of sensors

Air vehicles have usually got many different sensors onboard in order to be aware of the surrounding environment. These sensors can be divided into cooperative or non-cooperative [18]. Cooperative sensors include common systems included in civil aircraft such as TCAS (Traffic alert and Collision Avoidance System) and ADS-B (Automatic Dependent Surveillance-Broadcast). These systems work by the communication between vehicles to exchange information and suggest possible trajectories to avoid collision [19]. However, some vehicles may not be fitted with this kind of equipment so they require non-cooperative sensors in order to know about their surroundings.

Nowadays, there are a wide range of non-cooperative sensors that are used not only in air vehicles but also in ground and maritime vehicles. For example, active sensors such as radars, lidars and sonars can measure object distances very accurately by emitting waves and calculating the time to receive the reflection [20]. However, these systems are relatively big and expensive so they may not be desired in some kind of vehicles [19].

The recent development in powerful processing units have brought a new possibility to sense the environment around a UAV and this is the usage of cameras. These passive sensors have not been usually used just on their own because of the limiting information that they are able to extract directly [21]. However, technology advancements are al-

lowing to use more powerful post-processing techniques of the information provided by cameras so they can be used for collision avoidance applications.

2.2.2 Cameras

Even when it is known that cameras want to be used, there are different possible solutions that could be used. Electro-optical cameras are usually used to extract colour information of the environment. Stereo cameras can be used to extract images from different locations and, thus, to obtain more robust information. However, this last approach would introduce more weight and cost to the vehicle. Since the main advantages of electro-optical sensors with respect to waves sensors are the lower size, weight, cost and power consumption and the faster scan rate, these aspects want to be optimized [22]. In this project, only an electro-optical camera will be used along with a sophisticated post-processing technique in order to overcome these limitations.

One of the most important technical parameters of a camera is the Field Of View (FOV). That is the reason why an expected FOV for Sense and Avoid systems has been quoted, being this of $\pm 110^\circ$ in azimuth and $\pm 15^\circ$ in elevation [23] as specified by the White Paper issued by Air Combat Command (ACC) entitled "Sense and Avoid (SAA) Requirement for Remotely Operated Aircraft (ROA)". In general, increasing the FOV reduces the resolution, what also reduces the detection range. In fact, the azimuth FOV could be reduced to $\pm 90^\circ$ in vehicles much faster than the encountered objects [24]. For that reason, an array of cameras, wide angle cameras or spherical cameras may be required in order to fill the whole FOV keeping enough resolution [16].

However, visual band cameras have usually got limitations in order to perform properly on low light conditions or on difficult weather conditions such as rainy, cloudy or fog presence. Therefore, an additional infrared camera may be needed so the environment can be sensed even when any of these hard visibility conditions occur. An additional disadvantage of this type of sensors is the lack of range information so sophisticated techniques such as filters are also needed to extract this information [22].

2.2.3 Post-processing solutions

Electro-optical cameras are very promising to detect and track objects but the information extracted from them is limited so advanced algorithms are required in order to be able to use this kind of sensors in Sense and Avoid systems alone, with no help of any other sensor. Since these algorithms provide the detection and tracking capability, their performance is very important for the subsequent avoidance and resolution func-

tions [16]. If an object cannot be accurately located, it will not be possible to avoid the collision with enough guarantee.

Morphological plus temporal filtering approaches

One of the most common approaches has been using morphological based methods in the spatial filtering stage [25] [26] [27] [28]. However, this kind of method has not been used alone, so additional stages were needed to consider temporal evolution such as temporal filtering [27], dynamic programming [26] or support vector machines (SVM) [25].

Morphological based techniques

Several techniques have been tested such as closed-minus open (CMO), preserved-sign, bottom-hat, adaptive contour morphological filtering and morphological reinforcement but the ones giving the best performance have been CMO and bottom-hat approaches [28].

The way spatial filters work is by extracting point-like features from the image frame so objects can be detected when these features are passed through a temporal filtering approach [26]. The two fundamental operations to do that are known as dilation and erosion.

Temporal filtering

As mentioned above, temporal filtering is needed when using morphological based methods. The two most widely used approaches have been Hidden Markov Models (HMM) and dynamic programming. HMM filter basically consists of a probabilistic model of all the different motions that an object can adopt [28]. Transition probabilities, initial probabilities and measurement probabilities are recursively propagated to obtain a detection [27]. Dynamic programming detects an object after multiple outputs of the morphological filter. It averages them along possible target trajectories and makes a decision about the presence of the object or not [26].

Bio-inspired methods

Optical flow

A more recent methodology for object detection has been optical flow, which is a technique inspired from biology. Since some animals such as insects do not have enough distance between both eyes, they cannot use stereo-vision to orientate themselves so they

have developed a different kind of motion detector based on the optic flow [29]. When an insect is flying, it can sense the environment according to the apparent movement of texture in the visual field relative to the insect's velocity [30]. Hence, it is possible to detect a closer object if it is moving faster that other further away objects. Furthermore, it can know if the object is on a collision course if it does not move but is increasing in size [31]. Several algorithms can be used to compute optic flow such as differential methods, correlation approaches and frequency-based methods [29]. However, optical flow is very sensitive to some background effects such as aggressive attitude manoeuvres, scene illumination and weather conditions and the ground when it is behind the target [22].

Saliency method

The saliency method is another bio-inspired technique based on the human vision system which is able to identify salient objects in complex environments by using the inherent visual attention mechanisms [32]. By this kind of algorithm, the vehicle can detect salient obstacles from the background.

Deep learning solutions

Recently, the use of neural networks for object detection has been increasing due to improvements in graphic processing units and datasets [33]. Despite existing several types of neural networks, Convolutional Neural Networks (CNN) have been the most widely used for these purposes [34]. Due to the advancements in CNN, they have already been applied in sensing processes from UAVs and aerial vehicles [35] [36] [37].

2.2.4 Range estimation

Usually, the location of a detected object in an image frame is given relatively to the image size or as the pixel locations. From this data, it is possible to calculate the relative angular measurements (bearing and elevation) with respect to the vehicle. To do that, camera information such as the focal length or the FOV is required so the measured angles can be calculated by simple expressions [27]. However, relative range and velocity cannot be measured from a single 2D image so passive ranging techniques are needed [38].

One method to estimate the target relative range is by static triangulation. If the vehicle is performing a manoeuvre that changes the line of sight (LOS), geometric calculations would give the range as the function of the different measurements [38].

A more extended and reliable technique has been to filter the measurements making use of Kalman filters (KF). Common KF used for this purpose have been Modified Spherical (MS-EKF), Range-Parametrised (RP-EKF), Log-Polar Coordinate (LPC-EKF) and Unscented (UKF) [16]. This kind of filters are optimal recursive data processing algorithms that take into account measurement outputs, prior knowledge about the system and measuring errors to minimise the error statistically [39]. The main problem with bearing-only measurements filters is the non-linearity issue which, along with the required target manoeuvre for range observability, requires equation linearisation or noise terms increment [40]. Several KF for this kind of measurements have been designed and compared, although practical results do not usually match the theoretical expectations [41].

One additional problem is the observability. Depending on the relative motion between vehicle and target, it is possible that the range cannot be estimated from the LOS measurement. For example, if the measured LOS is constant, it will not be possible to know about the relative range [16]. It has been shown that the range and the range rate are only observable when the vehicle performs a manoeuvre with an acceleration component perpendicular to the LOS vector [38].

2.3 Conflict detection

Once an object around the UAV has been detected, it is required to identify if it will generate a conflict or not. Since it is expected to have sensors capable to detect objects when the collision is not yet imminent, the future trajectory of both the object and the UAV must be estimated in order to assess the probability that a future conflict exists. According to the way to estimate the trajectory, a different conflict probability calculation approach can be considered.

2.3.1 Trajectory prediction

The most simple approach is the nominal method. In this kind of technique, a unique trajectory is estimated for each agent without considering any uncertainty on it. Hence, object manoeuvre is not taken into account and this method can only be used in short ranges or short periods of time since it becomes inaccurate otherwise [18].

The worst-case prediction, however, calculates every possible trajectory of the object and checks which ones of them would generate a conflict with the UAV. This is a very inefficient and computationally expensive method but it ensures that no collision will occur [42].

A kind of trade-off between the two previous techniques would be the probabilistic method. In this case, the trajectory is predicted considering uncertain factors that describe the risk probability [19].

An additional approach can be the flight plan exchange, although this would be available only in UAVs with cooperative sensors. By sharing route and dynamics information between vehicles, accurate information would be considered [18].

2.3.2 Conflict decision

When the future trajectories have been estimated, a decision of whether there is the risk of conflict or not must be taken. There are two ways to make this decision. The first and most simple one is by a deterministic method. In this case, certain distance threshold around the vehicle is considered and a conflict is detected if the trajectory of the object intersects this threshold at some point [43]. Therefore, this would be the method used with either the nominal trajectory prediction approach or the shared flight information one.

A more realistic technique would be the probabilistic method. Predicting the trajectories by a worst-case or a probabilistic approach, a conflict probability could be calculated and a corresponding manoeuvre would be executed if this probability is greater than a threshold value. Using a probabilistic trajectory prediction, there could be two different ways to approximate the conflict probability since the analytical expression cannot be calculated [44]. The most accurate one would be by Monte-Carlo sampling using a large number of samples. This approach, however, would have a high computational cost so it would not be appropriate for real time applications such as collision avoidance [45]. A more simple approach would be approximating the conflict probability with an analytical expression that could be quickly computed. Some accuracy would be lost but the value would be quickly obtained, allowing longer available time to execute an evasion manoeuvre.

The most widely analytical approximation used is [14] [46] [47]:

$$P(conflict) = \frac{1}{2} erf\left(\frac{R + \bar{r}_f}{\sqrt{2}\sigma_{r_f}}\right) + \frac{1}{2} erf\left(\frac{R - \bar{r}_f}{\sqrt{2}\sigma_{r_f}}\right) \tag{2.1}$$

where R is the required separation between vehicle and obstacle, $\bar{r}_f$ is the miss distance vector, σ_{r_f} is its variance and the Gaussian error function is:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du \tag{2.2}$$

However, different estimations of the parameters have been studied since the uncertainties can be modelled in several ways. To evaluate whether the obtained expressions work correctly, usually direct simulations are performed to check that the conflict can be avoided and also the obtained conflict probabilities are compared with Monte-Carlo results so the accuracy can be validated [45].

2.4 Manoeuvre resolution

When a conflict is predicted, an evasion manoeuvre is needed in order to avoid a possible collision. As expected, there are several ways to execute an appropriate manoeuvre as well.

2.4.1 Rule-based methods

Sets of rules would be predetermined during the system design phase considering some predicted situations. This simple approach would allow for a very efficient and fast manoeuvre calculation without any need to perform additional computation and giving more time to avoid a collision. However, it has less effectiveness than others and is not optimal [18] [19].

2.4.2 Geometric approach

This kind of method consists of generating a geometric trajectory according to any geometric requirement as could be the separation distance. It is also a simple approach but its online calculation can make it more appropriate to the encountered situation. However, it could become more complex if the geometric trajectory wants to be optimized [18] [42].

2.4.3 Optimized trajectory

This approach is strongly related to the previous one since it also relies on a geometric trajectory but, in this case, the generated path is optimized according to some constraint

or cost. The trajectory could be optimized by using different factors such as the least-aggressive manoeuvre, increasing the complexity [18] [42] [19].

2.4.4 Force field methods

Potential fields have widely been used for path planning in many different applications. The target location is treated as an attractive force while obstacles are represented by repulsive forces. By tuning correctly the characteristic parameters, the vehicle will finally reach the target destination avoiding the obstacles. However, it could get stacked at some point with no possibility to get to the destination. Furthermore, the field must be updated every time the current configuration is updated so it can be computationally expensive [19] [42] [18].

2.4.5 Game theory

Game theory methods rely on inter-agent communication so a decentralized solution is achieved when all the agents look to avoid the collision [18] [19].

2.4.6 Bearing angle based approach

This method is relatively new and uses visual sensors. The objective is to keep the object locations at a safe position within the FOV. However, it is not very accurate and does not take into account the object distance to the UAV [42].

2.5 Deep learning post-processing

In this project, a deep learning approach has been considered to detect objects from the captured frames from the camera. Due to recent improvements in processing units, their importance has increased in the last years.

Neural networks can be categorised into supervised, unsupervised and reinforcement learning according to their learning process [34]. In supervised learning, neural networks are provided with a ground truth dataset so they can learn from it and, then, generalise to other sets of data later. Unsupervised learning, however, does not use ground truth data and the objective of this kind of neural networks is to extract key features from the data so ways to classify them can be found. Finally, reinforcement learning makes

use of interaction between the Artificial Intelligence (AI) agent and a real or simulated environment to provide feedback and to improve the performance of the activity carried out.

Supervised learning neural networks have been used in this project. In this kind of neural networks, the training process can be performed online or offline. Online training refers to learning from the real experience while the neural networks is operating whereas offline training refers to a training process previous to operation where a prepared training dataset is provided and the neural network does not learn while operating so they are usually faster.

The most common training process is offline since networks are able to operate faster what is important in applications where speed is a critical factor. In the scenario considered in this project, low computational power is required in order to make the UAV lighter so offline training neural networks have been considered. For this training process, a set of ground truth data is provided so there is certain output from every input in this dataset. Hence, the inputs of the training dataset are propagated through every layer of the neural network and an output is obtained, which is compared with the ground truth output. An error is then calculated and it is backpropagated so the neural network weights are adjusted. It is an iterative process highly dependent on the dataset that should be large and diverse.

There are several kinds of neural networks that meet the previously commented requirements. However, two of them - Convolutional Neural Networks and Recurrent Neural Networks - have been used due to their characteristics. In fact, a more advanced and new kind of neural network involving both of them has also been used called hybrid Convolutional and Recurrent Neural Networks. The literature review of each one of them will be presented in the next sections.

2.5.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is the most common deep learning approach for computer vision. They can perform pattern recognition tasks in images so they are used for object detection purposes [1]. Their input, hence, is a matrix with the pixel values of an image that has three spatial dimensions: height, width and depth. The basic CNN architecture is as depicted in Figure 2.1. There are three types of layers: convolutional layers, pooling layers and fully connected layers.

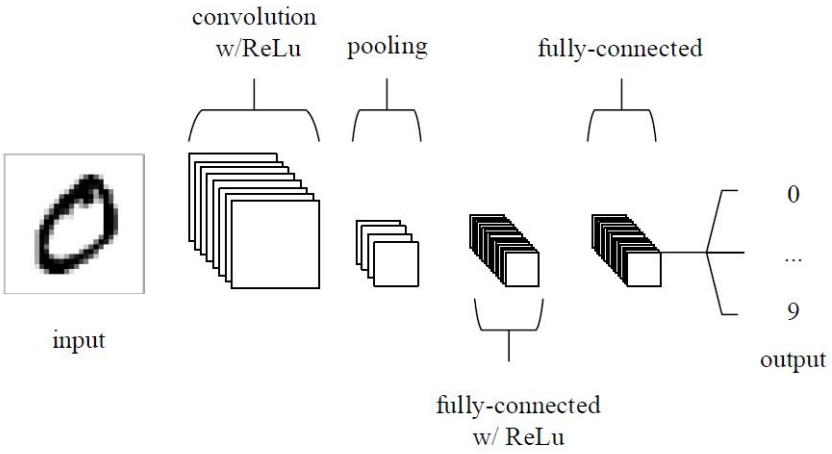


Figure 2.1: Simplified CNN architecture for MNIST classification [1].

Convolutional layers are responsible to extract features from the image. Each convolutional layer applies several filters to the image matrix executing the convolution calculation with each one of them. Hence, a convolutional map from each filter is obtained so as many maps as filters applied are extracted from each layer. Furthermore, each filter represents a different image feature so features can be extracted from different locations in the image. One advantage of this kind of neural networks is the parameter sharing since each filter is applied to different regions of the image but its parameters are the same so the parameter optimization during the training process is reduced with respect to other neural networks. Commonly, a rectified linear unit (ReLu) is also applied at the end as activation function.

Pooling layers are used to reduce the size of the obtained matrices. There are different ways to define them but the most common ones are averaging the values around certain position or just using the maximum value [48]. Usually, convolutional and pooling layers are successively used until the final fully connected layers so the inputs of the next convolutional layers are reduced due to the pooling layers action. Moreover, it is possible to extract deeper features by using deeper neural networks, what refers to using more convolutional and pooling layers. This is common to every neural network but there is a problem: higher computational load will be required and overfitting may appear, what refers to fit very well to the training dataset but failing at generalizing to external data.

Finally, fully connected layers are usually placed at the end of the neural network in order to analyse the extracted features and combine the obtained information to make a decision. Usually, a CNN is designed to be able to detect several classes in an image so the output matrix would be a vector with a size of that amount of classes. Hence, the

output of the fully connected layers would be a vector with the detection probability of each one of the classes.

Popular CNNs

CNNs development has been motivated by competitions such as ImageNet Large Scale Visual Recognition Challenge [49], PASCAL VOC Challenge [50] or COCO Challenge [51]. Furthermore, larger and diverse datasets were also developed for these challenges so they had a key role on CNN technology. Some of the most important CNN architectures developed so far will be described next.

AlexNet

AlexNet [2] was the first deep neural network to increase significantly the accuracy, winning the ImageNet LSVRC-2010. They used a simple architecture with 5 convolutional layers followed by 3 fully connected layers as shown in Figure 2.2. They were the first ones to introduce the ReLu activation function instead of the tanh or sigmoid which were the most common ones until then. With this new activation function, training was faster and the probability of having the vanishing gradient problem was reduced. An additional advancement was to add a DropOut layer after every fully connected layer in order to reduce overfitting.

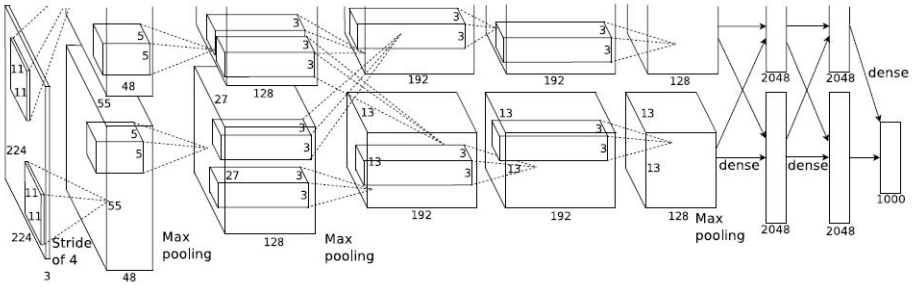


Figure 2.2: AlexNet architecture [2].

VGG16

VGG16 [52] won the localization task of the ImageNet LSCRC-2014. In their paper, results for several depths of the neural networks are shown so they prove that higher depth (higher number of layers) is beneficial, obtaining the best results with 16 convolutional layers. Furthermore, they decreased the size of the filter used in the convolutional layers with respect to the ones used in AlexNet, extracting more complex features and at a lower cost.

GoogLeNet/Inception

The problem with VGG16 was the high computational requirements. GoogLeNet [3] defends that all output channels in a convolution operation are not connected to all input channels so there are some connections that can be spared. GoogLeNet aims to spare these unnecessary or redundant connections. To do so, they designed the Inception module as shown in Figure 2.3 to reduce a common sparse CNN in a more effective way where convolutions of different sizes are used in order to capture details at varied scales. The 1x1 convolution is known as bottleneck layer and it is introduced to significantly reduce the computation requirements. In addition, the final fully connected layers are replaced by a global average pooling layer. In AlexNet, the fully connected layers parameters represent around the 90% of the total parameters so, using this replacement the number of parameters is drastically reduced.

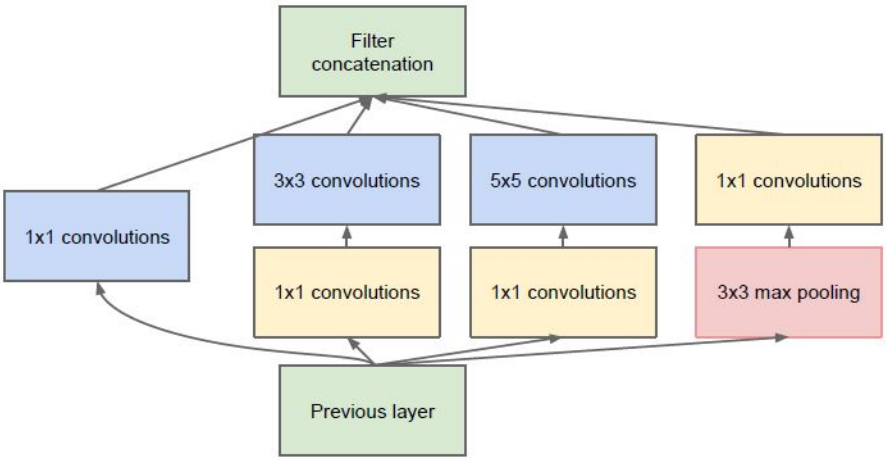


Figure 2.3: Inception module [3].

ResNet

According to previously comments, deeper networks will increase accuracy if overfitting does not occur. However, there are some other problems that may appear such as vanishing gradient (earlier layers are almost negligible learnt) and problems coming from the optimization of a huge parameters space. ResNet [4] reduced these problems making use of a method called degradation as in Figure 2.4, making it possible to keep using very deep networks. ResNet won the classification, detection and localization tasks of ImageNet LSCRC-2015 and the detection and segmentation tasks of COCO 2015.

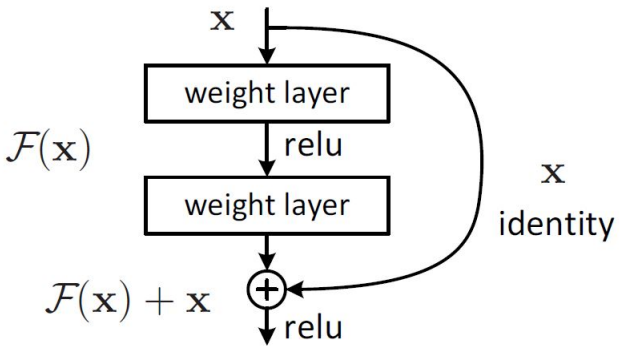


Figure 2.4: Residual learning: a building block [4].

CNNs for object localization

The most traditional CNNs as the ones explained before can detect and classify objects in an image. However, they cannot locate them. To obtain the location of the objects within the image, more complex networks are required. The output of this kind of neural networks usually includes a bounding box around the detected object. Some of the state-of-the-art methods are explained now.

Faster R-CNN

This method was originally called R-CNN when it was published in 2014 because it was based on Regions with CNN features (Region-Convolutional Neural Network) [53]. It showed greater performance than other similar approaches so it was a great advancement. That is why further research was carried out and it was developed, leading to more sophisticated algorithms such as Fast R-CNN [54] and Faster R-CNN, which is the last version [5].

The working principle of this kind of methods is a sliding window along an image so when an object is detected, its location is known by the position of the window when the detection occurred. The improvement in Fast R-CNN was that Regions Of Interest (ROI) were proposed so speed was greatly improved since there was no need to use the sliding window across the whole image. The further improvement of Faster R-CNN was to use an additional network for this purpose. Hence, Faster R-CNN has two networks: one generating regions proposals and another one detecting objects in those regions. This reduces the time cost again. The network ranks region boxes called anchors and proposes the ones with the highest probability of containing an object. Another important advantage is the ROI Pooling. Since the proposed regions will probably be of different sizes, the CNN feature maps will also have different sizes, what is not efficient. By using ROI Pooling, the output has always the same size independently of the input

size. In Figure 2.5, a simplified scheme of the mechanism is shown. Used with VGG16 in a mix of COCO and PASCAL VOC datasets, a precision of 75.9% mAP (mean Average Precision) at 5 fps (frames per second) can be achieved.

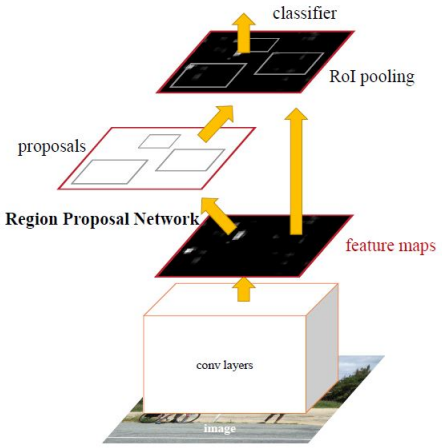


Figure 2.5: Faster R-CNN working scheme [5].

SSD

Although Faster R-CNN got good results, recently more research is being conducted towards different methods that are expected to achieve faster speeds in the future. In these new methods, it is possible to look into the whole image for the objects locations instead of performing a region proposal first, so everything is done at once and not in two steps as with Faster R-CNN. One of these methods is Single Shot Multibox Detector (SSD) [6].

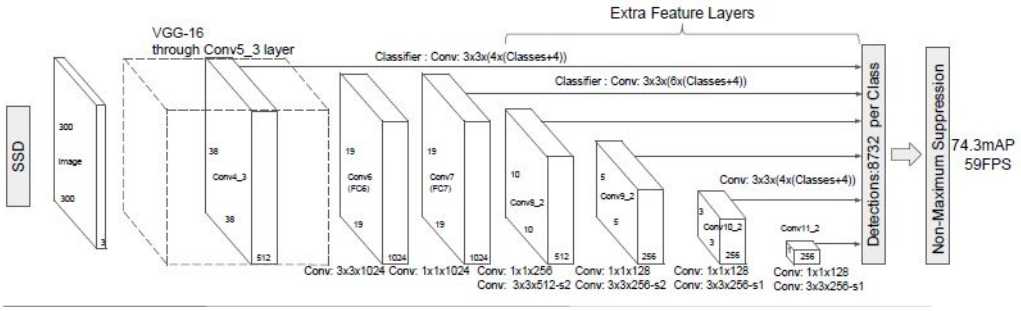


Figure 2.6: SSD architecture [6].

SSD scored over 74% mAP at 59 fps on standard sets such as PASCAL VOC and COCO and it showed that faster object locators were possible. SSD's architecture builds on the VGG16 architecture but discarding the last fully connected layers and adding convolutional layers instead as in Figure 2.6. In this case, the bounding boxes proposal

is performed by a method called Multibox in which a convolutional network with an Inception style is used. Hence, every map cell is associated with a set of default bounding boxes of different dimensions and aspect ratios and a location loss is computed in order to select the correct ones. In the original paper, some additional observations are provided such as that more default boxes results on higher accuracy but lower speeds so there is a trade-off between both, and that 80% of the time is spent on VGG16 so faster speeds could be achieved with more efficient networks.

YOLO

YOLO is the abbreviation of You Only Look Once, what is a brief explanation of its essence: it consists of a single network that gets the object locations. As with Faster R-CNN the first version was released in 2016 [11] but an updated version was published in 2017 as YOLO9000 [55] and the most recent one is YOLOv3 which was presented in 2018 [56].

YOLO claims to be as accurate as the best detectors but much faster. For example, comparing with SSD, they get the same accuracy but at a three times faster speed. In this case, they use their own CNN network called darknet. To find the bounding boxes, the input image is divided into multiple regions and certain number of bounding boxes are proposed in each one. Taking into account some parameters such as confidence probability and Intersection over Union (IoU), the final bounding boxes are selected.

2.5.2 Recurrent Neural Networks

A different type of neural networks are the Recurrent Neural Networks (RNN). The main feature of this kind of neural networks is that they are able to learn temporal information [57]. Due to this capability, their main applications have been handwriting recognition [58] [59] [60] and generation [61], language modeling [62] and translation [63], acoustic modeling of speech [64] and speech synthesis [65].

The structure of a simple RNN is given in Figure 2.7. In this case, only the hidden layer considers past information using the feedback loops, that are recurrent cycles over time or sequence. However, these feedback loops can also be applied to the output or input layers. The main working principles are the same as of any supervised Artificial Neural Networks (ANN) so the goal of the training process is to get the same outputs from the network than the ground truth.

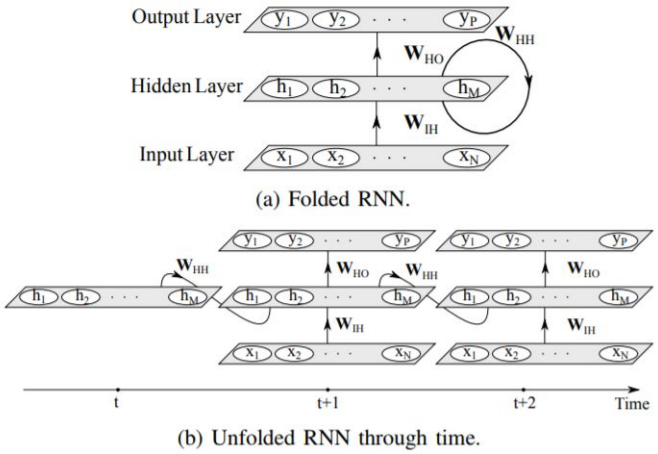


Figure 2.7: Simple Recurrent Neural Network structure [7].

One of the main drawbacks of this kind of neural networks was the difficulties to train them, so long times were required [7]. However, advancements in GPUs as well as the development of backpropagation using Gradient Descent (GD) has helped in this training process. However, there are still some additional problems when modeling long-term dependencies, which are the vanishing and exploding gradient problems [66]. To solve this issue, a special kind of RNN were developed, which are Long Short-Term Memory (LSTM) RNNs.

Long Short-Term Memory

LSTM reparametrizes the RNN so the gradient cannot vanish [66]. The structure of a LSTM cell can be seen in Figure 2.8. However, the main disadvantage is the increment in complexity, as can be easily spotted. The introduced non-linear gating units regulate the information flow into and out of the cell [8]. These gates are the input gate, the output gate and the forget gate, so they make it possible to decide what information remains in consideration and what information can be forgotten.

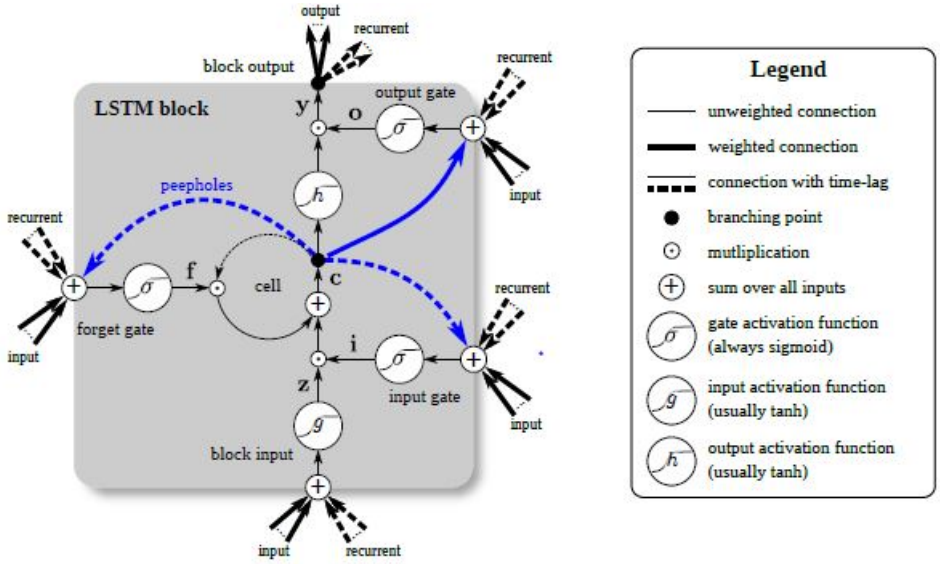


Figure 2.8: Long Short-Term Memory cell structure [8].

Computer Vision Applications

As commented above, computer vision has not been one of the main applications of RNN. Other applications such as language modeling or speech recognition have been more used. However, this kind of neural networks has some characteristics that can also be helpful for computer vision. That is why some research in this area has been carried out in the last years.

Visual Attention

Since CNN can be very computationally expensive, this technique relies on using visual attention to make the object detection process faster [67] [68] [69]. This method is based on Recurrent Attention Models (RAMs) which are inspired by the way humans perform visual recognition tasks. Basically, they process a multi-resolution crop of the input image called glimpse and use deep RNNs to find additional objects or features in the surroundings taking into account the previous searched area. Hence, they pay attention to a particular region in the image (glimpse) and detect objects by considering a sequence of features. However, many of these networks still use convolutional layers to extract features from the image.

Action recognition

A further step for object recognition in images is to use RNNs in videos due to their temporal information consideration. Hence, many projects have focused on finding a

neural network that detects actions in videos as CNNs detect objects in images. For these purposes, commonly LSTM is used since long-term memory is usually required [70]. However, more complex networks have been developed from simple LSTMs to derive on solutions such as deep differential recurrent neural networks [71] [72]. One of the main problems of this application is the poor available datasets. It is difficult to get huge datasets for videos as well as to label them with ground truth. Therefore, the results are promising with the datasets used but their generalisation into the real world is still an open issue.

Tracking

One of the most promising RNN applications is tracking. The main characteristics of the two previous applications would be used in this last one since an attention model would be used to track a specific object in the image but it would be tracked in videos. Although there are other ways to perform tracking, RNN would also take into account image features information, what could be a great improvement. Several research [73] [74] has already shown good results, allowing to track sized objects accurately and in real time.

2.5.3 Hybrid Convolutional and Recurrent Neural Networks

The main characteristics of Convolutional and Recurrent Neural Networks have already been outlined in previous sections. Hence, it is possible to see that both of them have very attractive behaviours. The interest on these kind of neural networks has increased in the last years so research has been conducted separately. However, after some time researching about them independently, currently there is an increased interest on studying the behaviour of neural networks considering the principles of both of them. This new kind of neural networks has been called hybrid Convolutional and Recurrent Neural Networks (CRNN).

The main applications have been focused on the temporal feature of RNNs to structures of data were CNNs are advantageous. A brief overview of the work done for different applications is given next.

Audio classification

CNNs can also be used to extract features from audio recordings. Hence, it is straightforward to think that their combination with RNNs would suppose an advantage since they are basically a sequence of sounds. That is why some researchers [75] [76] have developed CRNN for music feature extraction and feature summarisation. This type of

networks showed better performance than simple CNNs due to the previous information consideration but the speed was decreased, since the neural networks were deeper.

Action recognition

In computer vision, action recognition in videos has also been one of the main applications as was with RNN. Due to the available datasets, it is easier to test the performance and to compare it with other methods tested with the same dataset, so it becomes a good option to assess the performance on new kind of neural networks in early stages.

In [77], the option of using 3D CNN was studied. 3D CNNs work as common CNNs but convolution is applied in three dimensions instead of two, the third dimension being the time. Their CRNN proposed solution consisted of adding a hidden layer with 50 LSTM cells after the 3D CNN. They used KTH dataset to validate the algorithm and showed that the performance can be increased by the 3% adding LSTM, specially with long sequence cases. They also compared the feature extractor previous to the LSTM with other common methods and using a deep learning technique such as 3D CNN showed to get better performance.

In [9], a CRNN with a structure as the one in Figure 2.9 was designed. They also used LSTM and their model was end-to-end trainable. They showed to work correctly not only on activity recognition tasks but also on image and video descriptions generation. They tested different configurations with the LSTM after the first or the second fully connected layer of the CNN. They used UCF-101, Flickr30k, COCO2015 and TACoS datasets to test the models in the different applications and showed that the performance was improved when including sequential dynamics learning.

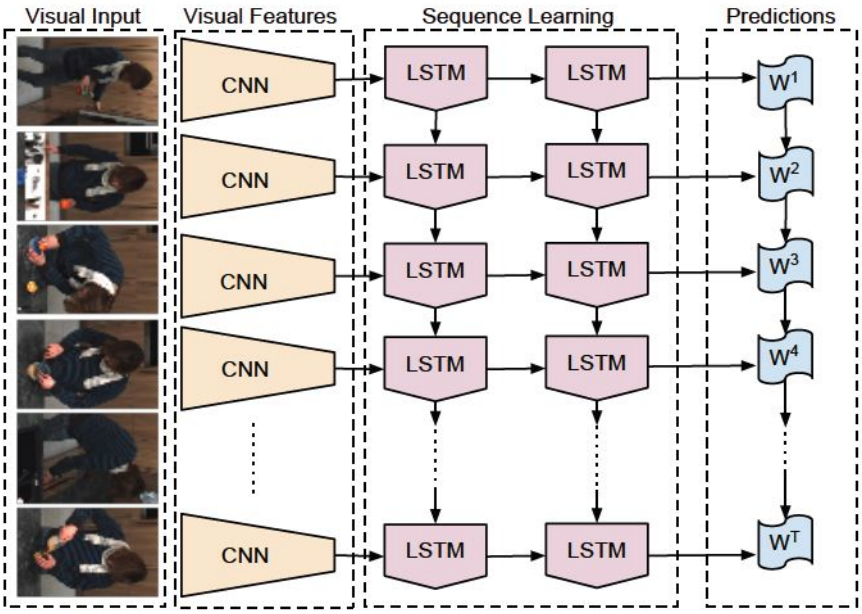


Figure 2.9: Example of a CRNN structure [9].

A different approach was considered in [78] since two CNNs were used before the LSTM: one processing the input frame and the second one processing the stacked optical flow. The best performance to date was achieved with 91.3% in UCF-101 and 83.5% in Columbia Consumer Videos (CCV). However, more neural networks were needed to be run so speed was penalised.

Some of the characteristics from the previously commented works were used together in [79]. They also used two CNNs: one for the raw images and another one for the optical flow information. Then, two different approaches were evaluated: one using just feature pooling and another one using LSTM with the outputs of the previous CNNs. They used UFC-101 and Sports 1 million datasets to validate the results and showed that using optical flow information is not always beneficial except when it is used with LSTM, so the best performance was achieved using LSTMs on both imaeg frames and optical flow.

Tracking

Object tracking has been one of the most important and active research areas in the field of computer vision [80]. The obvious application of CRNN to this area has lead to the development of some models to face this purpose.

In [81], an end-to-end trainable CRNN which is trained offline to track anonymous objects in noisy environments is proposed. It uses the past predictions and the corre-

sponding visual features extracted with the convolutional network. They tested it on cluttered scenarios with occlusions and other objects distractions and obtained robust performance.

A complete network to detect and track objects was developed in [82] so instead of performing the tracking after the detection stage, both actions were performed at once. Their model can be divided into two parts: a detection part based on Fast-RCNN and RNN to initialize an object trajectory and a tracking part based on a Markov decision process which performs the motion prior and appearance comparison.

A different open source code was used in [10]. In this case, YOLO is used for the detection and LSTM is added to track the location making a kind of recurrent YOLO so the new model was called ROLO. A simplified scheme of the system and the tracking procedure is shown in Figure 2.10. In this algorithm, the extracted features were transmitted to the LSTM before the fully connected layers so the class detection was omitted in the temporal propagation. After assessing it in challenging benchmark tracking datasets, this method showed to be more accurate and robust that the state of the art while maintaining a low computational cost. Other works [83] also compared this solution with other combinations such as SSD and LSTM showing that ROLO got the best results.

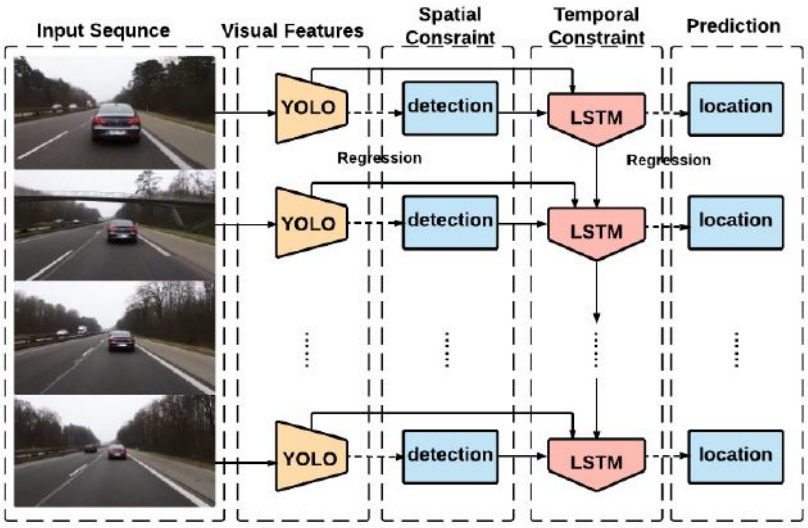


Figure 2.10: Simplified architecture and working scheme of ROLO [10].

A hierarchical attentive recurrent model was developed in [84]. Hybrid CRNN were used to focus the attention on some part of the image depending on past information. The evaluation was performed with pedestrian tracking on the KTH dataset and with the KITTI object tracking dataset, where state of the art performance was obtained at a reasonable computational cost.

A new CRNN tracker was designed in [12]. This algorithm can track any generic object with a predefined bounding box around it but does not perform object detection. It considers the features of the objects by a CNN and takes into account their evolution over the time thanks to two LSTM so it is robust to rotations and occlusions. They were able to achieve speeds of 150 fps in a GPU so this comes at a very low computational cost, as was represented in the model name: Re^3: Real-time Recurrent Regression Network. They evaluated the performance in datasets such as VOT 2014, VOT2016 and ILSVRC 2016, where they showed to outperform other real-time trackers and to be competitive with other deep approaches.

Finally, [85] proposed a recurrent filter learning (RFL) algorithm using convolutional LSTMs. They achieved improved and competitive results on large-scale tracking datasets such as OTB100, VOT2016 and VOT2017 while also improving the training performance with respect to other CNN-based methods.

Chapter 3

Object detection and tracking

The most important part of a Sense and Avoid technique is likely to be the object detection and tracking. If an obstacle is not detected, it will not be possible to avoid any conflict with it. Hence, a correct performance of this part has key importance. Furthermore, the innovative contribution of this thesis comes from this section since a new technique that has never before been used in this application has been tested. One of the main limitations for the study was the available time for carrying it out. Neural networks usually require long time to design and to train, specially when they include RNNs. Since it was needed to develop a complete Sense and Avoid algorithm, the decision to use pretrained open source codes of the required neural networks was assumed.

After a deep literature review of the state of the art work and its open availability, some options were selected. Since the objective was to use hybrid Convolutional and Recurrent Neural Networks, open source solutions were searched. However, since these neural networks are a very new trend, there were not many of them. After considering some candidates, trying them out and rejecting them because of finding important issues, a network was finally selected. The CRNN considered was Re^3, which was a very recent code released just a couple of months before. This project showed to be serious and user-friendly. Furthermore, very good results were reported since it was possible to track any object at very high speeds. Re^3 was chosen as the CRNN for the Sense and Avoid application. Nevertheless, Re^3 was a tracker and not an object detector. It could track any object with a bounding box around it but it could not generate this bounding box, so that was taken as an initial input. Hence, a previous CNN was considered to generate the initial bounding box and, then, Re^3 would be used to track the detected object.

Due to the previous decision, a CNN with object localisation had to be selected as well. In this case, there has been more research about this kind of neural networks and the most advanced ones were open source. As commented, YOLO looked to obtain the best results in main of the experimental tests. However, YOLO is implemented in a platform called Darknet that was not widely used and did not have as many capabilities as other options. Moreover, Re^3 was implemented in Python, which was considered a better platform than Darknet, so a Python code was desired. After the consideration of using Faster R-CNN or SSD, an implementation of YOLO in Python using Tensorflow was found. This extension was called darkflow and allowed to use YOLO in Python. Although the most recent version was YOLOv3, in darkflow only YOLOv1 and YOLOv2 were available. However, YOLOv2 was tested and better results than with Faster R-CNN and SSD were obtained specially in relation to speed, what is important for Sense and Avoid applications and was also taken into account because of the limited computing power available for the project. Therefore, YOLO was selected as the object detector method for the algorithm.

A new deep learning technique was created from the two previous models. Since Re^3 cannot detect objects and YOLO cannot track them efficiently, a new model called ReLO was designed in which both of them were integrated in order to obtain a suitable performance for Sense and Avoid applications. In the design process, the inputs and outputs of the previous models were combined and some additional design parameters were introduced in order to adapt the behaviour to different scenarios so the best performance was always achieved.

In the next sections, the used open source codes will be explained in more detail, as well as their implementation and configuration. Then, the designed detector and tracker algorithm called ReLO will be introduced. Finally, ReLO's capabilities and advantages will be presented and its performance will be assessed.

3.1 YOLO

The main advantage of YOLO (You Only Look Once) with respect to other approaches such as Faster R-CNN is that the whole process is performed in a single neural network, what is usually faster and easier to optimize since there is no need of training different networks separately. In this case, a unified neural network is responsible of proposing bounding boxes and selecting the final ones.

3.1.1 Prediction

What YOLO does to predict bounding boxes is to divide the input image into an $S \times S$ grid of cells. The cell containing the centre of an object present in the image is responsible for detecting that object. Each grid cell predicts a prefixed number of bounding boxes B. The prediction of each one of these bounding boxes consists of 5 components: x, y, w, h and $confidence$. The coordinates (x, y) represent the central point of the bounding box, (w, h) represent the width and the height of the bounding box and $confidence$ is the probability that the predicted bounding box actually exists. Although B bounding boxes are predicted in each grid cell, they can actually be larger than the cell's size since objects can be larger than cells and there is only one cell responsible for detecting each object. Furthermore, the coordinates (x, y) and the magnitudes (w, h) are normalized to be between 0 and 1. The last component of the prediction vector is the confidence score. It reflects how confident the model is that the predicted bounding box contains an object and how accurate the bounding box is. It is calculated as $Pr(object) \times IOU$: the product of the probability that there is an object and the Intersection over Union (IOU) parameter. Hence, the confidence score should be 0 if there is not any object associated to that cell and it should be the IOU parameter if there is an object. The IOU gives the ratio between the overlapped area and the unified area of two bounding boxes as given in Figure 3.1. In this case, it is calculated with the area of the predicted bounding box and the area of any ground truth box.

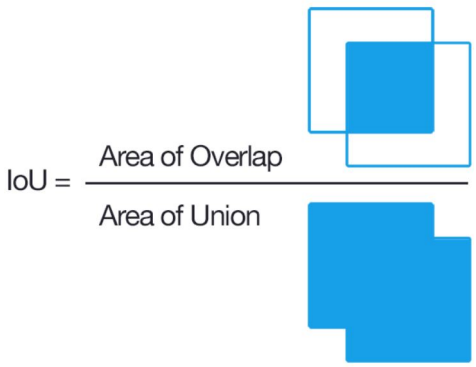


Figure 3.1: Intersection over Union (IOU) representation.

As can be seen, the bounding boxes are predicted independently of the object class. The class information is included by adding to the prediction vector C class probabilities. These values are conditional class probabilities $Pr(class|object)$ so they depend on the probability of having an object associated to that cell. In this way, the class probabilities will be low if the object detection probability is low. Only one set of class probabilities is predicted independently of the number of predicted bounding boxes B.

Therefore, there are $S \times S \times B \times 5$ outputs related to bounding box predictions and $S \times S \times C$ outputs related to class predictions, what makes a final output tensor of $S \times S \times (B \times 5 + C)$.

3.1.2 Testing

In order to obtain a final detection score, the conditional class probabilities are multiplied by the bounding box confidence prediction, which gives a class-specific confidence score for each bounding box.

$$Pr(class|object) \times Pr(object) \times IOU = Pr(class) \times IOU \tag{3.1}$$

Hence, this probability reflects the class confidence and how well the bounding box fits the object. Since multiple bounding boxes will be predicted in the image, only the ones with the probability over certain threshold are finally kept.

3.1.3 YOLO network architecture

The network structure looks like a CNN with 24 convolutional layers, 4 max pooling layers and 2 fully connected layers. The sequences of 1×1 reduction layers followed by 3×3 convolutional layers were inspired by the inception module used by GoogLeNet. In the example of Figure 3.2, the output tensor is of $7 \times 7 \times (2 \times 5 + 20)$ since the grid size was 7×7, 2 bounding boxes per cell were considered and the PASCAL VOC dataset was used for training, which contains 20 different classes.

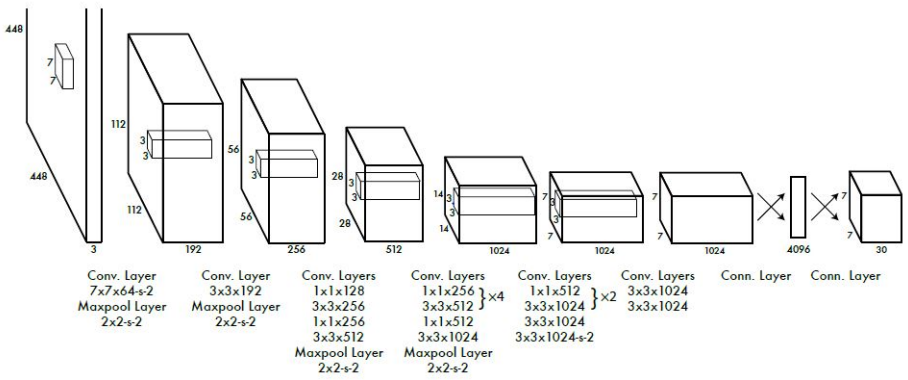


Figure 3.2: YOLO network structure [11].

3.1.4 YOLOv2

Now that the working basics of YOLO have been explained with its first created version, the improvements introduced in its second version YOLOv2 will be commented, since it was the used one in the project. The first version showed to classify very accurately but with some localization errors and low recall. Therefore, the aim of the second version was to improve the recall and the localization while maintaining the accuracy.

In order to improve the localization performance, anchor boxes were used. The last fully connected layers were eliminated and anchor boxes were used to predict the bounding boxes instead so the offsets were predicted instead of the coordinates. The reason is that predicting offsets simplifies the problem and makes it easier for the network to learn so accuracy is improved.

An additional improvement with the first version was the improved grid resolution since the feature map was of 13×13 instead of 7×7. By this refinement, smaller object could be detected. To do that, they used something similar to ResNet by adding a passthrough layer from the $26 \times 26 \times 512$ feature map, which turns to $13 \times 13 \times 2048$, so lower and higher features were concatenated.

3.1.5 Detection process

The version of YOLO used was trained with the COCO dataset, which can detect 80 different classes such as cars, people, aircraft or birds. First, when an input image was run through the network, it was divided into a grid of 13×13 cells as in Figure 3.3.

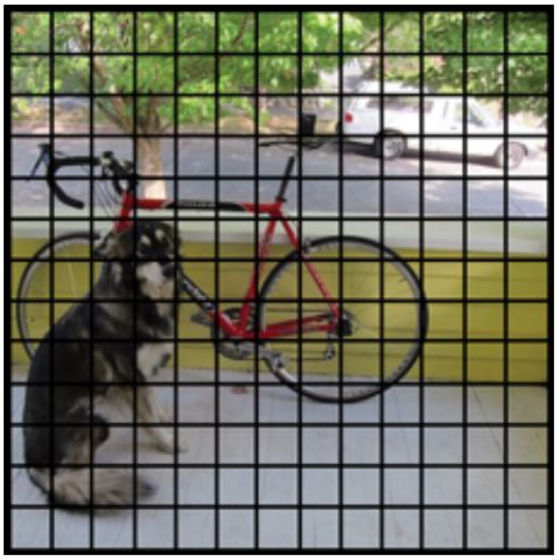


Figure 3.3: Input image grid division with YOLO.

5 bounding boxes were predicted in each cell. In Figure 3.4, all the predicted bounding boxes are shown. As commented before, in addition to their position and size, also a confidence score was obtained. In this case, higher confidence scores are represented by wider lines of the bounding boxes.

Figure 3.4: Predicted bounding boxes with YOLO.

As mentioned before, each output tensor consists of the bounding box information and of the classes probabilities information. Then, for each bounding box, the probability that each class has been detected was obtained. Therefore, the higher probability class

was assigned to that bounding box. That is represented in Figure 3.5. In this case, both the bounding box and class information were combined as in Equation 3.1 so a final confidence score for each bounding box representing the class confidence and the bounding box accuracy was given.

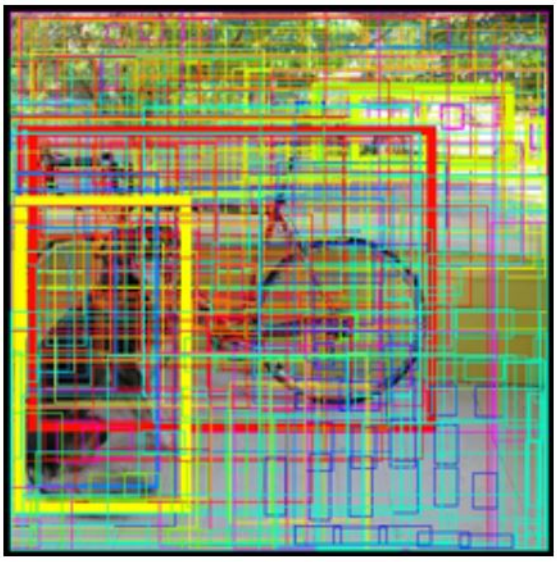


Figure 3.5: Predicted bounding boxes and associated class with YOLO.

As it can be seen, a lot of bounding box were predicted. Indeed, $13 \times 13 \times 5 = 845$ bounding boxes were predicted. However, most of them had very low final confidence scores so they could be rejected. To do that, a threshold value was selected and could be adjusted for each image. In this example, a threshold value of 30% was used so only bounding boxes with confidence scores over the 30% including class probability were kept giving a result as in the Figure 3.6. With higher threshold value, more false alarms would be avoided but the probability of not detecting a true object would also increase. Hence, a trade-off value was needed.

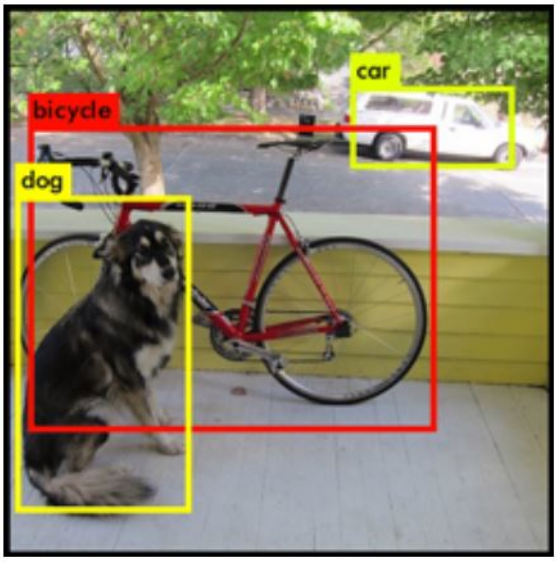


Figure 3.6: Final output image with YOLO.

3.2 Re^3

Re^3 is a hybrid Convolutional and Recurrent Neural Network capable to track any random object even if the neural network was not trained with that particular object. It is implemented with Python language using Tensorflow and the publication paper was released on February of 2018 so it is a very up-to-date method. The overall structure can be seen in Figure 3.7.

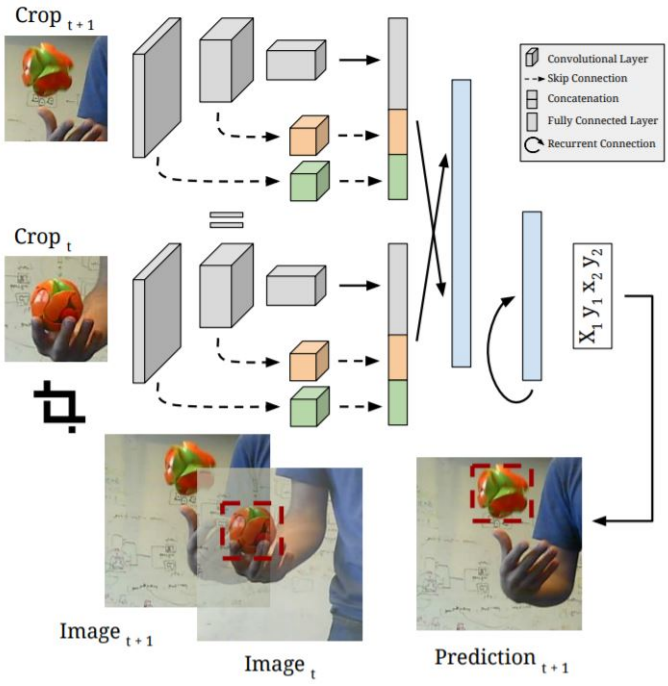


Figure 3.7: Re63 structure [12].

3.2.1 Object features extraction

This neural network worked as a tracker so it needed a detected object to be tracked. Hence, the input to this neural network was a bounding box around an object. However, in order to track that object, its features must be extracted so the same features could be detected in subsequent frames and the object could be tracked. Furthermore, these features might change over time so they must be updated. The feature extraction of Re^3 was performed by a CNN.

The way features' evolution was considered is as follows. The neural network is fed with two crops: one centred at the object location in the previous frame and the second in the same location but in the current frame. Furthermore, both crops have the same size, which is the double of the input bounding box in the previous frame so surrounding context is also considered. By comparing both crops, it is possible to learn how motion affects features evolution. However, this method did not guarantee that the object will be inside the second crop since, if it moves more than 1.5 times its width and height in a single frame, it would be out of the crop. It was considered that it is unlikely that the object has such a high speed. Moreover, the method is based on a trade-off between speed, resolution and search region size. If the search region was made larger, it would allow to track faster objects but the algorithm speed would decrease since it would need to process a higher amount of data at each frame. Both crops are concatenated at the

end of the convolutional pipeline instead that at the beginning in order to differentiate clearly their specific features.

Different feature-levels were considered as well. As more convolutional layers were used, deeper features were extracted from the image. In order to have as much information as possible, not only the output of the last layer was taken into account in the temporal evolution, but also features extracted at intermediate levels. To do so, skip connections at intermediate points were used so their outputs were concatenated with the output of the CNN to feed through a final fully connected layer to reduce dimensionality before feeding into the recurrent pipeline.

The CNN architecture was based on a similar model to AlexNet called CaffeNet [86] as the one shown in Figure 3.8. Skip connections were taken after the first, the second and the fifth convolutional layers with 16, 32 and 64 channels respectively. Moreover, each skip connection has a PReLU nonlinearity.

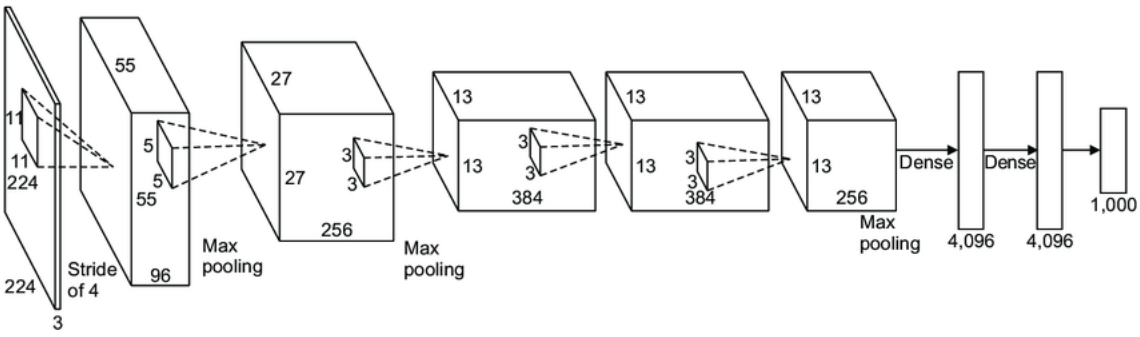


Figure 3.8: CaffeNet architecture [13].

3.2.2 Recurrent stage

One of the main complexities in RNNs are the difficulties for training since it usually takes more iterations to converge and the selection of hyperparameters is more demanding. Hence, the design of this stage is fairly important. The approach used to consider temporal information was to use a two layer LSTM since it is possible to capture more complex transformation and for longer times than a single layer LSTM.

The LSTM input was taken from the output of the CNN which had 2048 units. Hence, each LSTM layer consisted of 1024 units and contained an input gate, a forget gate, a cell gate and an output gate in order to manage temporal features. The outputs of the LSTM were fed into a fully connected layer to obtain only 4 parameters consisting of the top left and bottom right corners of the bounding box, giving its size and position. During training, many times the ground truth outputs were fed into the inputs of the

next stage in order to generate predictions. However, in this case the ground truth was given at the first iterations but the own predictions were fed later so the model was less prone to diverge.

3.2.3 Training remarks

The network was trained with large datasets combining real and synthetic data so it could adapt to any object. The object tracking datasets used for training were ILSVRC 2016 and Amsterdam Library of Ordinary Videos 300++ (ALOV). The Imagenet Video dataset is the largest one for video data with 3862 training videos while 307 videos of ALOV were used. To generate the synthetic dataset, images of Imagenet Object Detection dataset were used. Random tracks with Gaussian noise were generated for different objects in some of those images. This gave diversity to the model since classes presented in the image dataset but not in the video dataset could also be considered.

Another remark is that training was carried out with sequences with a maximum length of 32 frames. Then, during test time, LSTM parameters were reset every 32 iterations since they were prone to diverge otherwise. However, instead of resetting them to zero, they were reset at the values of the first iteration in order to maintain some information of the object.

The optimization method used to train the network was the ADAM gradient optimizer with the default momentum and weight decay and a learning rate starting at 10^{-5} and ending at 10^{-6}. The training process took approximately 200000 iterations and one week in an Intel Xeon CPU E5-2696 v4 @ 2.20GHz and an Nvidia Titan X (Pascal).

3.2.4 Working procedure

Unlike other methods, in this case training was not performed online in addition to the offline one. The network was not updated itself but the recurrent parameters were adjusted to model the object temporal evolution. With this procedure, the network could track objects while they changed over time without requiring the additional computational cost of online training.

Since this was a tracker, the input was a frame with a bounding box around certain object. Two crops of the image at two different time steps were processed by the CNN as commented. The output of the CNN was not a class probability because Re^3 was not a classifier either so it only output the extracted features from the crop. However, not only the deepest features were taken into account, some skip connections were added

at intermediate layers of the CNN in order to consider features at different levels. The obtained features were then fed into a two layer LSTM network. This recurrent network considered the evolution of the features over time to obtain an accurate prediction of the new bounding box in the new frame. Then, the next frame was released with the new bounding box and the same process was carried out again. Due to the consideration of past information, this model was capable of keeping tracking objects even when they were occluded for some period of time (CNNs would not detect the object) or they were rotating or changing some features (CNNs could not detect them because of their different features).

3.3 ReLO

Once both models were tested to work correctly, their integration was carried out in order to create the algorithm that would detect and track objects in Sense and Avoid applications. The methodology to do that was to start from the most simple case and to check that it worked correctly. If it did, the algorithm was improved to expand its applicability to more complex scenarios. The resulting model using a CNN such as YOLO and a hybrid CRNN such as Re^3 was called ReLO, since it was based on a combination of Re^3 and YOLO.

3.3.1 Single object

The first algorithm was designed to track a single simple object moving steadily or remaining static. It was assumed that the object always remained within the FOV limits. With these specifications, the object could be tracked using just Re^3. However, an initial bounding box around the object was needed. Considering that the object could be detected from the beginning, running YOLO at the first frame and Re^3 at the rest of frames would be enough. However, the UAV would probably fly without encountering any object most of the time so, in the case that YOLO was run at the first frame when there was no object and Re^3 in the rest of frames, a new incoming object could not be detected. The considered solution was to run YOLO in the UAV as a basis if no objects were detected. Once an object was detected with YOLO, Re^3 would take the generated bounding box and would keep tracking the object for the next frames. As will be showed in the next section, YOLO run much slower than Re^3. That fact, in addition to CRNN characteristics such as the use of temporal information, made that the optimum behaviour was achieved minimizing the use of YOLO and maximizing the use of Re^3, but YOLO was needed to detect objects.

In terms of code, the previous explanation was implemented by introducing a YOLO-flag. Hence, in YOLO-mode, the YOLO-flag was activated so YOLO was used in each frame to detect objects. After every time YOLO was run, it was checked whether an object had been detected or not. If there was no object, YOLO-flag kept activated and, if an object was detected, YOLO-flag deactivated so Re^3 was run for the next frames. The inputs Re^3 took from YOLO outputs were the class of the detected object and 4 coordinates of the bounding box, which were its x and y coordinates of the top left and bottom right corners.

3.3.2 Multiple objects

Re^3 had different codes for single and multiple objects. Thus, the algorithm must be modified in order to be able to detect more than one object, which is essential in Sense and Avoid applications because of safety issues. In addition to using the Re^3 version considering multiple objects, some other features were added. One issue was that each object was identified in each frame only by its class. Hence, if multiple objects of the same class were detected, different objects could be tracked as the other one at some point. To solve this, a different name was assigned to each object when they were detected with YOLO. This name consisted of the detected class and a reference number. Therefore, at each frame it was generated a bounding box around the object with its identity name so it could be checked that every object was correctly tracked over the whole time period.

3.3.3 Refresh parameter

The previous algorithms were very basic and were assuming non-realistic scenarios where the objects remained within the FOV for an infinite time. The truth was that the scenario would not be the same and would be changing from time to time. For example, once a conflict had been avoided with a detected object, it would disappear from the FOV of the UAV. However, a bounding box would remain in the frames because Re^3 did not know when an object was not there any more so it kept generating its bounding box. Therefore, an additional term called 'refresh parameter' was introduced in order to deal with this problem.

The refresh parameter stopped running Re^3 and run YOLO every defined number of frames. Hence, every certain number of frames, YOLO was run so the detected objects were refreshed. That means that if any object was not in the FOV any more, ReLO would not track it any more, or that if any additional object had appeared during that time period, ReLO would keep tracking the previous objects and the new one or,

basically, every object detected in the FOV with YOLO. Thus, the refresh parameter was the number of frames at which the tracking reset so YOLO was run, the objects in the FOV at that frame were detected and they were tracked for the next frames with Re^3 until YOLO was run again. It must be remarked that the YOLO-flag kept working so if at some point YOLO was run and no object was detected, YOLO would keep running for the next frames until a new object was detected.

Therefore, the introduction of the refresh parameter allowed ReLO to work in any kind of environment with more confidence. Nevertheless, the main drawback was the speed penalty. YOLO was slower than Re^3 so running YOLO more frequently made the overall speed lower than if it would have been run only once at the first frame. Anyway, the refresh parameter was a design parameter so it could be tuned accordingly to the expected environment. If the UAV is expected to fly at high altitude in the countryside, it will probably not find many objects so the refresh parameter could be tuned larger because the probability that an additional object appears while tracking a previous one is low. If the UAV will fly in a complex and crowded environment, thus a lower refresh parameter will be required in order to take into account new objects coming in and out of the FOV. Furthermore, another characteristic to consider when tuning the refresh parameter would be the class of the detected object. It will not be the same to track an aircraft with constant speed than a bird with different speeds and flying directions. Hence, the refresh parameter should be tuned accordingly.

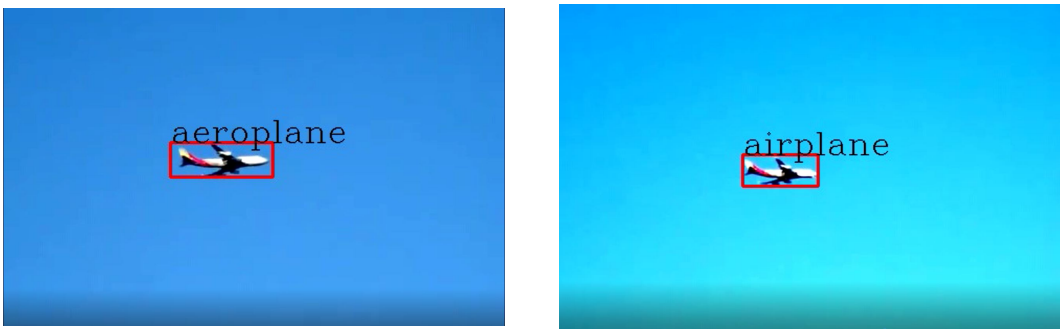
3.4 Performance Assessment

Once the algorithm was designed, its performance should be assessed in order to check that its behaviour was acceptable. The way to do this was by assessing videos qualitatively so it could be checked that the objects were tracked and the bounding boxes correctly generated. The relative speed between different versions could also be assessed. Several videos were tested and special attention was given to the objects appearing in those videos since they had to be of one of the classes detected by COCO, the dataset used for YOLO training. In this case, since YOLO was the responsible to detect new objects, only the classes considered in its training could be detected. Since pre-trained weights were used in this project, only those classes could be detected. However, in future applications YOLO could be trained with whatever are the desired objects.

3.4.1 YOLO and Re^3

First, YOLO performance was compared to Re^3 in a simple video with a single object as shown in Figure 3.9. In the case of YOLO, each frame was taken by separate and YOLO

was directly applied to each one of them. Re^3 could not detect any object since it only tracked. Therefore, YOLO was applied only in the first frame to generate the bounding box and Re^3 was applied in the rest of them so the object was tracked. It was checked that the object was correctly tracked in both cases but there were some differences. The first one was the assigned class to the object. In the Re^3 case, the class was detected by YOLO in the first frame and then it was propagated to track the object. Hence, the class did not change during the tracking process. In YOLO, however, the detection and class assignment process was carried out at every frame so there were some frames where the object was detected as a bird instead of an aircraft as was the correct one. However, if a wrong class would be detected in the first frame, it would be propagated for the whole video with Re^3 and would lead to higher error. The second difference was the processing speed. In a common CPU with Intel i7 and 8 GB RAM, YOLO worked at 1.3 fps while Re^3 worked at 12.2 fps. Re^3 was capable of tracking an object at a speed almost 10 times greater than YOLO. That was one of the most important advantages of using Re^3 instead of YOLO since it would allow to track objects at real time. It must be remarked that the official paper where Re^3 was published claimed that the algorithm worked at 150 fps with an Intel Xeon CPU E5-2696 v4 @ 2.20GHz and an Nvidia Titan X (Pascal). Since real time cameras commonly work at around 30 fps, it was considered that this algorithm could work in real time applications. Moreover, a factor of around 12 between the achieved speed with the used processing unit and the more powerful ones was assumed from now on in order to get conclusions about speed.



(a) YOLO performance.

(b) Re^3 performance.

Figure 3.9: YOLO and Re^3 comparison.

3.4.2 Refresh parameter

The refresh parameter was introduced in order to get a trade-off between the performance characteristics of both networks. By introducing YOLO more frequently, the probability of propagating a wrong class was reduced because most of the frames were correctly detected by YOLO. Furthermore, multiple objects coming in and out from the FOV could be detected as well. However, it worked faster than running YOLO at every single frame.

In Figure 3.10, an example of a complex environment with multiple objects moving fast and coming in and out from the FOV is shown. In such complex environments, low values of the refresh parameter were needed so the detected objects could be updated frequently. In fact, if the refresh parameter was set at too high values, YOLO was run every longer time intervals, what had two misdetecting consequences. The first one was that birds flying into the FOV during that period were not detected until the next time YOLO was run. Hence, higher the refresh parameter, longer the time undetected objects were in the FOV. The second was that bounding boxes were kept in the image even when the corresponding objects had already left the image. The reason was that Re^3 could not know if an object was not in the FOV, it just tracked the object. Therefore, when the object went out, there was a bounding box present in the image with no object in it until YOLO was run again. That would be a problem since the collision avoidance part of the algorithm might consider that an avoidance manoeuvre was needed when there was not any object actually. In those cases, higher the refresh parameter, longer the time that an empty bounding box would be tracked in the image.

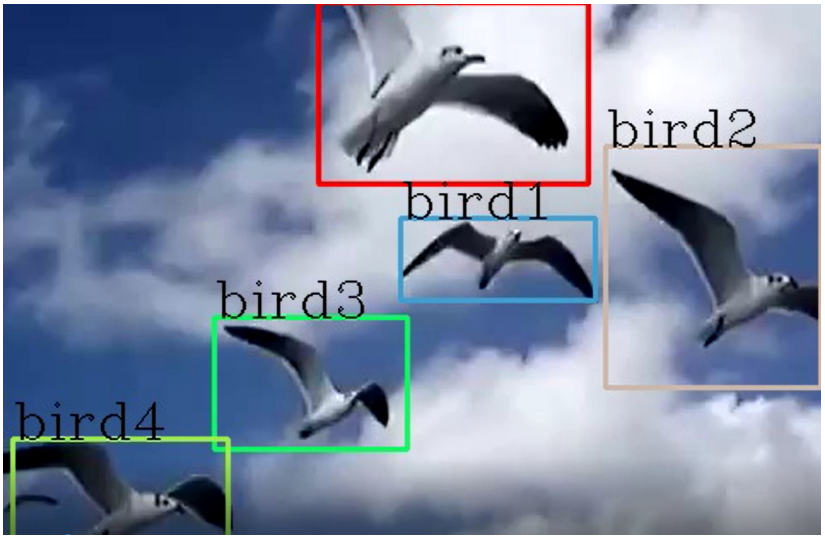


Figure 3.10: Frame capture of a video applying ReLO in a complex environment.

Another important reason to introduce the refresh parameter was speed. In Figure 3.11, a graph is presented showing the trend of the mean tracking speed at different refresh parameters for the same complex example as before. It can be seen that, in general terms, the higher the refresh parameter, the higher the speed since YOLO was run less frequently. In any case, from around 30 frames, the speed did not increased so much while the detection performance decreased so it did not make much sense to use refresh parameters higher than that. The reason of that stop in the increment was that, since YOLO was run every longer time intervals, the objects detected when it was run varied. Hence, at 60 frames YOLO was run at frames were there were only two objects so for the next interval only two objects were tracked while at 70 frames,

maybe 5 objects were detected so there were more objects to track for the next interval. Thus, the decrement in speed due to a higher number of objects might be greater than the increment in speed because of running YOLO less frequently. However, despite the fact that speed increased, the detection performance was badly affected. Therefore, a trade-off was needed. From 50 frames, serious misdetection and empty boxes problems appeared so, in this case, the optimum refresh parameter was found around 30 frames. At those values, the detection performance was fine since there were no misdetected objects or empty bounding boxes for very long and the mean tracking speed was of 4.2 fps, what could be around 48 fps in a more powerful processing unit, what would allow to work in real time. Furthermore, it must be remarked that this was just an example of a very complex environment that a UAV would probably never find. Thus, ReLO would be usually allowed to work at lower refresh parameters, improving speed and maintaining the detection performance.

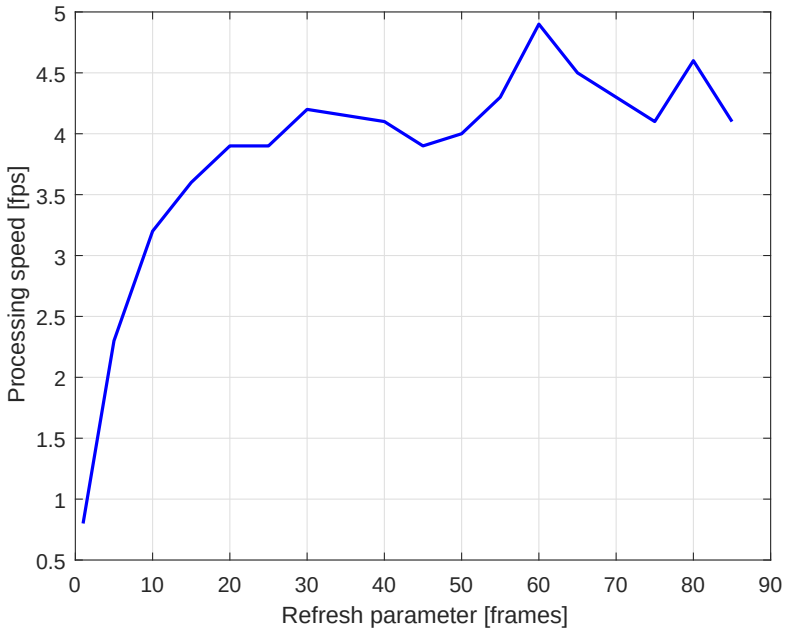


Figure 3.11: Tracking speed for different refresh parameter values.

Chapter 4

Collision avoidance

The second part of the Sense and Avoid algorithm is the collision avoidance. In this part, it was assumed that an object had been initially detected by ReLO. Then, the whole process to refine the measurement, to decide if an avoidance manoeuvre was needed and to generate such an avoidance trajectory was designed. The process was the following: first, the angle measurement was extracted from the pixel location from the camera. Then, the relative range between the object and the UAV was estimated with a Kalman filter. Once, the object position had been obtained, the probability of a future conflict was calculated. Finally, the avoidance trajectory required to avoid the conflict was also proposed. In Figure 4.1, a simple scheme of the considered collision avoidance procedure is shown.

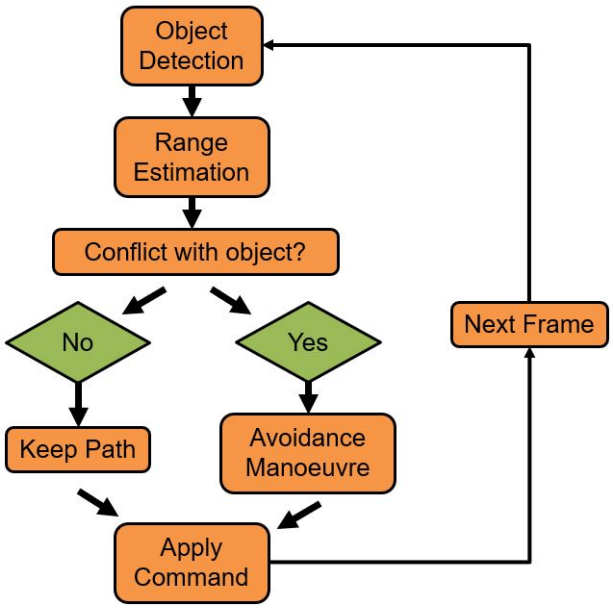


Figure 4.1: Basic scheme of the collision avoidance methodology.

4.1 Angle measurements

The output of ReLO when an object had been detected consisted of the bounding box coordinates around the object. These bounding box coordinates were given in pixels. The position of the object was considered the centre point of the bounding box. Thus, the object position was given as a pixel location in a 2D image. The first objective was to translate the available information into angle measurements, what would be a bearing angle in the horizontal direction and an elevation angle in the vertical direction. To perform this translation was straightforward if the camera specifications were available.

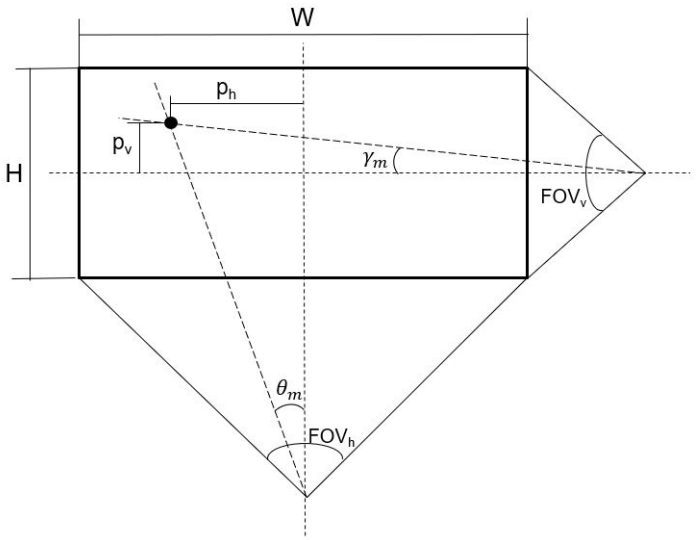


Figure 4.2: Basic graphical scheme for angle measurements extraction.

The required information from the camera were the resolution and the FOV. The resolution is a specification from cameras given by the manufacturer while the FOV is usually given too but it can also be obtained from a camera calibration. Hence, the situation would be as depicted in Figure 4.2. The object location was considered as a point and the camera features were guessed as knowns. Therefore, the bearing and elevation angles could be easily calculated by the following expressions:

$$\theta_m = \frac{p_h FOV_h}{W} \tag{4.1}$$

$$\gamma_m = \frac{p_v FOV_v}{H} \tag{4.2}$$

4.2 Range estimation

In the event of a collision, the relative range between the object and the UAV would decrease until it was zero. Hence, it was important to monitor the relative range in order to know the distance between the UAV and the object once this had been detected. However, angle measurements lacked this information. Thus, other methods were needed in order to extract range information. Although additional sensors could be used, in this case it was desired to use only a single camera, which only took angle measurements. Therefore, some kind of estimator must be used in order to calculate this range. The most extended method has been with filters such as Kalman filters or the particle filter. Since range estimation from angle measurements was a non-linear process, non-linear estimators were needed. According to the literature, the Extended and the Unscented were the most commonly used Kalman filters. In this case, the Extended Kalman Filter (EKF) was selected to estimate the relative range. The EKF designed in [40] was taken as a template and it was extended to the 3D since a more basic problem was approached in that article because it was based on 2D bearing-only measurements.

The working principle of an EKF is to linearise the measurement equation so it can be used as a linear problem. It is assumed that the behaviour of the system will be close to the linear assumption so it is an approximation that will be more accurate as the assumption is hold.

4.2.1 Definitions

The state vector of the obstacle was defined by the three coordinates of its 3D position and the three components of its velocity as:

$$X^t = \begin{bmatrix} x^t & y^t & z^t & \dot{x}^t & \dot{y}^t & \dot{z}^t \end{bmatrix}^T \tag{4.3}$$

The state vector of the UAV was defined in the same basis:

$$X^o = \begin{bmatrix} x^o & y^o & z^o & \dot{x}^o & \dot{y}^o & \dot{z}^o \end{bmatrix}^T \tag{4.4}$$

The desired parameter that wanted to be estimated was the relative range, so the true state vector of the EKF was defined by the relative coordinates taking into account both obstacle and UAV kinematics:

$$X = X^t - X^o = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{4.5}$$

Translating the previous Cartesian coordinates into polar coordinates, the bearing angle θ, the elevation angle γ and the relative range r could be calculated as:

$$\theta = f_1(X) = \arctan\left(\frac{y}{x}\right) \tag{4.6}$$

$$\gamma = f_2(X) = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \tag{4.7}$$

$$r = \sqrt{x^2 + y^2 + z^2} \tag{4.8}$$

The angle measurements at each time step k were β_k and ϕ_k which were not assumed perfect so they had some associated noise modelled as Gaussian. Hence, the measurement noise covariance matrix R was defined by the covariances of both measurements noises:

$$R = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} \tag{4.9}$$

4.2.2 Initialization values

Before running a Kalman filter, the initial states and covariances must be initialized. The initial values are usually guessed according to empirical predictions, intuitions or other methods. In this case, the initial position coordinates were initialized randomly while the initial velocity components were initialized as 0 so the obstacle was initially guessed static. However, as will be discussed later in this report, the initial guesses could be adjusted depending on the class of the detected object.

The covariance matrix, however, was not initialized directly randomly but previous parameters were used for this purpose. In fact, the error covariance matrix was initialized as:

$$P_{1|1} = \begin{bmatrix} C_{xyz} & 0_{3x3} \\ 0_{3x3} & C_{\dot{x}\dot{y}\dot{z}} \end{bmatrix} \tag{4.10}$$

where C_{xyz} was the covariance matrix of x, y and z which was obtained by simple rotation:

$$C_{xyz} = B_2 B_1 C_{x'y'z'} B_1^T B_2^T \tag{4.11}$$

where $C_{x'y'z'}$ was the covariance matrix in its native coordinate system and could be approximated assuming $\sigma_{rr} \ll 1$:

$$C_{x'y'z'} = (r_{1|1})^2 \begin{bmatrix} \sigma_{rr}^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\gamma^2 \end{bmatrix} \tag{4.12}$$

The corresponding rotation matrices were given as:

$$B_1 = \begin{bmatrix} \cos(f_1(X_{1|1})) & -\sin(f_1(X_{1|1})) & 0 \\ \sin(f_1(X_{1|1})) & \cos(f_1(X_{1|1})) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

$$B_2 = \begin{bmatrix} \cos(f_2(X_{1|1})) & 0 & \sin(f_2(X_{1|1})) \\ 0 & 1 & 0 \\ -\sin(f_2(X_{1|1})) & 0 & \cos(f_2(X_{1|1})) \end{bmatrix} \tag{4.14}$$

The initial covariances of the velocity components were given as:

$$C_{\dot{x}\dot{y}\dot{z}} = \begin{bmatrix} \sigma_{v_x}^2 & 0 & 0 \\ 0 & \sigma_{v_y}^2 & 0 \\ 0 & 0 & \sigma_{v_z}^2 \end{bmatrix} \tag{4.15}$$

4.2.3 Prediction

The first stage of the EKF was the prediction. In this stage, the state and covariance were predicted before taking into account the measurement at the corresponding time step. The predicted state could be calculated as:

$$X_{k|k-1} = F X_{k-1|k-1} - U_k \tag{4.16}$$

where F was the state transition matrix that modelled the process. Thus, considering constant velocity kinematics:

$$F = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

where T was the time interval between time steps.

U_k was the input vector, which inserted the known manoeuvres from the UAV into the equation:

$$U_k = \begin{bmatrix} (x_k^o - x_{k-1}^o) - T\dot{x}_{k-1}^o \\ (y_k^o - y_{k-1}^o) - T\dot{y}_{k-1}^o \\ (z_k^o - z_{k-1}^o) - T\dot{z}_{k-1}^o \\ \dot{x}_k^o - \dot{x}_{k-1}^o \\ \dot{y}_k^o - \dot{y}_{k-1}^o \\ \dot{z}_k^o - \dot{z}_{k-1}^o \end{bmatrix} \tag{4.18}$$

The covariance matrix must also be predicted. The equation for that prediction was:

$$P_{k|k-1} = FP_{k-1|k-1}F^T + GQG^T \tag{4.19}$$

where Q was the process noise covariance so q must be tuned accordingly to the process characteristics:

$$Q = \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & q \end{bmatrix} \tag{4.20}$$

and G was defined as:

$$G = \begin{bmatrix} \frac{T^2}{2} & 0 & 0 & T & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 & 0 & T & 0 \\ 0 & 0 & \frac{T^2}{2} & 0 & 0 & T \end{bmatrix}^T \tag{4.21}$$

4.2.4 Update

After predicting the state and covariance in each time step, the measurements were introduced and the final estimated states and covariances were adjusted. The state estimate equation was given by:

$$X_{k|k} = X_{k|k-1} + K_k v_k \tag{4.22}$$

In the previous equation, v_k represents the innovation, which is the difference between the measured angle and the predicted angle, that was obtained from the predicted state:

$$v_k = \begin{bmatrix} \beta_k - f_1(X_{k|k-1}) \\ \phi_k - f_2(X_{k|k-1}) \end{bmatrix} = \begin{bmatrix} \beta_k - \arctan\left(\frac{y}{x}\right) \\ \phi_k - \arctan\left(\frac{z}{\sqrt{x^2+y^2}}\right) \end{bmatrix} \tag{4.23}$$

The matrix K_k was the Kalman gain, what was the responsible to give more or less importance to the measured values or to the estimated ones. The equation to calculate the Kalman gain was:

$$K_k = P_{k|k-1} H_{k|k-1}^T \times [H_{k|k-1} P_{k|k-1} H_{k|k-1}^T + R]^{-1} \tag{4.24}$$

where H was the observation matrix and, in an EKF, is the Jacobian Matrix of the predicted state $X_{k|k-1}$. Since it tends to divergence and gives wrong results when expressed in Cartesian coordinates, in this case it was translated to polar coordinates as suggested in [40]. Hence, the observation matrix ended as:

$$H_{k|k-1} = \left[\frac{\partial f}{\partial X_{k|k-1}}\right] = \begin{bmatrix} \frac{\cos(f_1(X_{k|k-1}) + \frac{\pi}{2} + \frac{\sigma_\theta}{2})}{\sqrt{x_{k|k-1}^2 + y_{k|k-1}^2}} & \frac{\sin(f_1(X_{k|k-1}) + \frac{\pi}{2} + \frac{\sigma_\theta}{2})}{\sqrt{x_{k|k-1}^2 + y_{k|k-1}^2}} & 0 & 0 & 0 & 0 \\ \frac{\cos(f_2(X_{k|k-1}) + \frac{\pi}{2} + \frac{\sigma_\phi}{2})}{r_{k|k-1}} & 0 & \frac{\sin(f_2(X_{k|k-1}) + \frac{\pi}{2} + \frac{\sigma_\phi}{2})}{\sqrt{x_{k|k-1}^2 + y_{k|k-1}^2}} & 0 & 0 & 0 \end{bmatrix} \tag{4.25}$$

With this information, the covariance matrix could also be updated:

$$P_{k|k} = [I - K_k H_{k|k-1}]P_{k|k-1} + P^s_{k|k} \tag{4.26}$$

where $P^s_{k|k}$ was an added stabilizing noise matrix that was obtained as:

$$P^s_{k|k} = \begin{bmatrix} C_{xy} & 0_{2x4} \\ 0_{4x3} & 0_{4x3} \end{bmatrix} \tag{4.27}$$

where C_{xy} was the covariance matrix of x and y which was obtained by rotation:

$$C_{xy} = B_1 C_{x'y'} B_1^T \tag{4.28}$$

where B_1 is given in Equation 4.13 and $C_{x'y'}$ was the covariance matrix and could be approximated as:

$$C_{x'y'} = (r_{k|k})^2 \begin{bmatrix} (1 - \cos(\sigma_\theta))^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.29}$$

4.3 Conflict detection

Despite the several methods to assess the presence of conflict, a probabilistic technique was considered in this case. When approaching the problem from a stochastic point of view, Monte Carlo approximations are the most accurate method. However, a high number of samples is needed in order to achieve this good accuracy so it takes too much time and computational cost to be used in real time applications. The expression to calculate the conflict probability is NP-hard but it can be approximated by some analytical expressions. The processing requirements to execute this kind of expressions is minimal compared to Monte Carlo samplings and they have been proven to be very accurate as well. In Sense and Avoid applications, processing speed is critical so later a conflict is detected, lower the available time to avoid that conflict. Therefore, it is important to obtain a method that can assess the conflict probability quickly so there is enough time to execute an avoidance manoeuvre if needed. For this reason, the analytical approximation from a probabilistic point of view was selected.

In this case, the considered approach was based on the methodology of [14], who proposed a simple method to obtain the conflict probability analytically. All the analytical expressions proposed after this one are very similar but modifying the noise modelling or some formulation.

4.3.1 Kinematic parameters calculation

In order to obtain some of the parameters, some kinematics must be calculated. The considered situation is illustrated in Figure 4.3 where A represents the UAV and B is the obstacle. At each time step, it was assumed that the relative velocity between both elements would keep constant in the future to assess the probability of conflict, so the state propagation was linear.

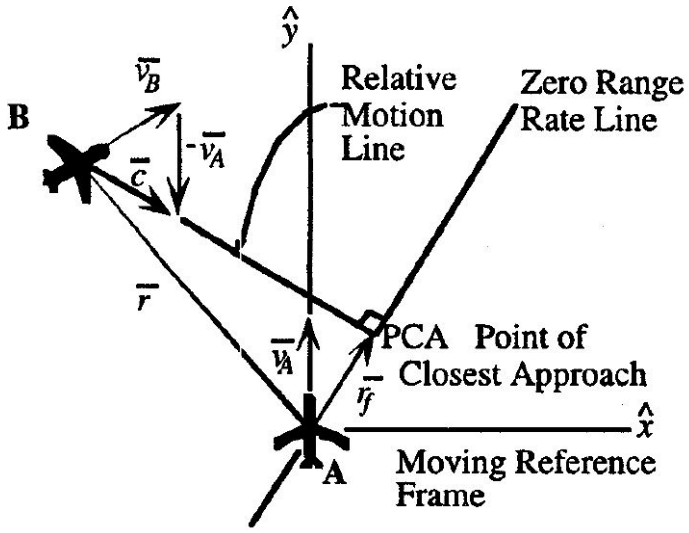


Figure 4.3: Relative motion scheme in 2D [14].

The miss distance is the minimum distance between the UAV and the obstacle when both trajectories are propagated through the future space. By definition, it was calculated as:

$$\bar{r}_f = \hat{c} \times (\bar{r} \times \hat{c}) \tag{4.30}$$

where $\bar{r}$ was the relative position vector of the obstacle with respect to the UAV, that is, the relative range vector; and $\hat{c}$ was the unitary vector in the direction of the relative velocity between both where $\bar{c} = \bar{v}_B - \bar{v}_A$:

$$\hat{c} = \frac{\bar{c}}{\|\bar{c}\|} \tag{4.31}$$

The Point of Closest Approach (PCA) is the point where the UAV will be the closest to the obstacle according to the linear state propagation. In that point, the miss distance vector and relative velocity vector are orthogonal:

$$\bar{r}_f \cdot \bar{c} = 0 \tag{4.32}$$

The time-to-closest-approach was defined by τ. Hence, assuming linear state propagation at the relative velocity $\bar{c}$, the miss distance vector could be expressed as:

$$\bar{r}_f = \bar{r} + \bar{c} \cdot \tau \tag{4.33}$$

Solving Equations 4.32 and 4.33, the time-to-closest-approach τ could be calculated:

$$\tau = -\frac{\bar{r} \cdot \bar{c}}{\bar{c} \cdot \bar{c}} \tag{4.34}$$

The time-to-closest approach was positive when approaching the obstacle and negative otherwise, so the conflict probability was automatically set to 0 when negative τ were obtained. An important parameter in order to estimate the conflict probability is the variance on the miss distance vector estimation. In this case, a simplistic approach was considered by propagating the variance according to the time-of-closest-approach:

$$Var(r_f) = \sigma_x^2 \sigma_y^2 \sigma_z^2 + \sigma_{v_x}^2 \sigma_{v_y}^2 \sigma_{v_z}^2 \tau^2 \tag{4.35}$$

where σ_x, σ_y and σ_z were the standard deviations of the three components of the position and the same with σ_{v_x}, σ_{v_y} and σ_{v_z} with the velocity. Instead of guessing them, those values were directly taken from the diagonal of the covariance matrix P obtained from the Kalman filter at each time step.

4.3.2 Conflict probability

The conflict was defined as the situation when the distance between the UAV and the obstacle was below certain value R. This parameter R should be set accordingly to the

required distance that should be kept in order to ensure safety. This requirement was modelled as a sphere around the object with radius R. Therefore, having previously calculated the miss distance vector and its variance, the conflict probability can be obtained by integrating the area under the probability distribution function between the locations that correspond to R and $-R$:

$$P(conflict) = \frac{1}{\sigma_{r_f}\sqrt{2\pi}} \int_{-R}^{R} e^{-\frac{(x-\bar{r}_f)^2}{2\sigma_{r_f}}} dx \tag{4.36}$$

Finally, using a change of variable $u = \frac{x-\bar{r}_f}{\sqrt{2}\sigma_{r_f}}$ and the Gaussian error function:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-u^2} du \tag{4.37}$$

The conflict probability was calculated as:

$$P(conflict) = \frac{1}{2} erf\left(\frac{R+\bar{r}_f}{\sqrt{2}\sigma_{r_f}}\right) + \frac{1}{2} erf\left(\frac{R-\bar{r}_f}{\sqrt{2}\sigma_{r_f}}\right) \tag{4.38}$$

4.4 Avoidance manoeuvre

When the probability of conflict is over certain threshold, an avoidance manoeuvre should be executed in order to avoid a collision. It was assumed that the UAV was following certain trajectory which only should be modified in case of a conflict. Hence, only the required deviation from the predefined trajectory was designed. A geometric approach was considered in this case. The proposed technique was based on the solution from [87]. In that case, it was designed to be used with a deterministic conflict assessment and with two aircraft that could execute a cooperative avoidance trajectory. However, it was adjusted to be used in this new application where the UAV was the unique responsible to avoid the collision.

The avoidance technique consisted of using the same safe distance R_{safe} introduced in the previous section about conflict detection. If the conflict probability was high, that meant that the UAV was very likely to cross that distance with respect to the obstacle. Therefore, the proposed avoidance trajectory consisted of modifying the UAV velocity direction so it went through the surface of the safe sphere and, thus, going further from the obstacle. This technique can be visualized in Figure 4.4.

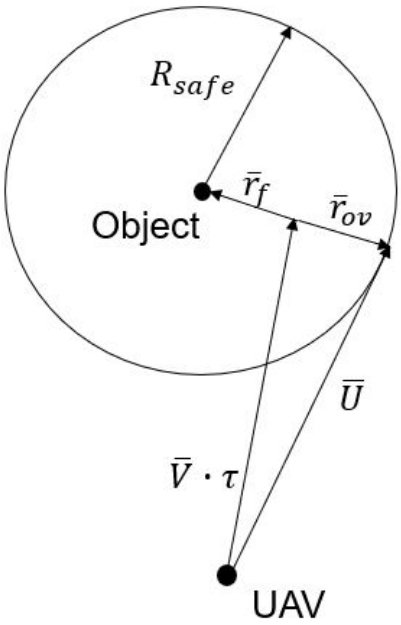


Figure 4.4: Avoidance manoeuvre technique.

The UAV would be flying with a velocity $\bar{V}$. Propagating that velocity until the time-to-closest-approach τ, it gave the miss distance vector $\bar{r}_f$ as discussed before. The distance vector for obstacle avoidance $\bar{r}_{ov}$ could be obtained by simple geometry as:

$$\bar{r}_{ov} = (R_{safe} - r_f) \cdot \left(-\frac{\bar{r}_f}{\|\bar{r}_f\|} \right) \tag{4.39}$$

Hence, the unitary vector for the new velocity direction to point through the safe circumference could be expressed as:

$$\bar{U} = \frac{\bar{V} \cdot \tau + \bar{r}_{ov}}{\|\bar{V} \cdot \tau + \bar{r}_{ov}\|} \tag{4.40}$$

Once the desired velocity vector was obtained, it was only needed to generate the required control commands in order to execute this new velocity. The avoidance manoeuvre consisted only of changing the velocity direction while maintaining the speed, so no thrust control was considered as first approach. It was assumed that changing only azimuth and/or elevation angles would generate the desired velocity direction. Hence, the desired trajectory angles could be obtained from the desired director vector of the new velocity:

$$\alpha = \arctan\left(\frac{U_y}{U_x}\right) \tag{4.41}$$

$$\psi = \arctan\left(\frac{U_z}{\sqrt{U_x^2 + U_y^2}}\right) \tag{4.42}$$

Therefore, executing this manoeuvre ensured that the safe distance R_{safe} was kept. It would be generated by applying the corresponding azimuth and elevation angles to change the velocity direction and it would be generated only when the conflict probability was over a threshold. Once the UAV got further from the obstacle, the conflict probability would decrease and the avoidance manoeuvre would not be generated any more, returning the UAV to its previous trajectory.

4.5 Performance assessment

The collision avoidance technique was assessed by simulation. MATLAB was the selected platform and an environment was designed in order to test the vehicle ability to avoid obstacles with this technique, once the objects had been detected. The simulation was first developed in 2D and expanded to 3D later.

4.5.1 Scenario

The simulated scenario was a cubed volume of $12 \times 12 \times 12$ km^3, but these magnitudes could be set at any values. The obstacle location was generated randomly around the centre of the volume so each component was between 4 and 8 km. It could be defined as static or dynamic. In case of choosing dynamic obstacle, it was possible to set the velocity. As standard velocity, simulations with moving obstacles were carried out at a speed of 5 m/s in the x direction. The UAV was initially located very close to the origin and with random heading and pitch angles between 0 and 45 deg. The UAV speed was set at 20 m/s and its minimum turn radius at 200 m. Its target point was generated near the (12, 12, 12) km point. This setup was designed with the goal of achieving a different situation at each simulation and being able to assess the behaviour at any scenario, but ensuring that the conflict would exist so a collision avoidance manoeuvre would be required. However, those were design parameters that could be changed at any time. In Figure 4.5, an example is shown. The animation was updated after each iteration so it was possible to see the result on real time. This Figure shows the final image after a

simulation. The UAV and the obstacle were represented by red and black geometrical shapes correspondingly and the safe distance around the obstacle was represented by 3 blue circles since the whole sphere would not allow to see the obstacle. The green line represented the past trajectory of the UAV and the cyan line represented the past and present estimated position of the obstacle as output from the EKF.

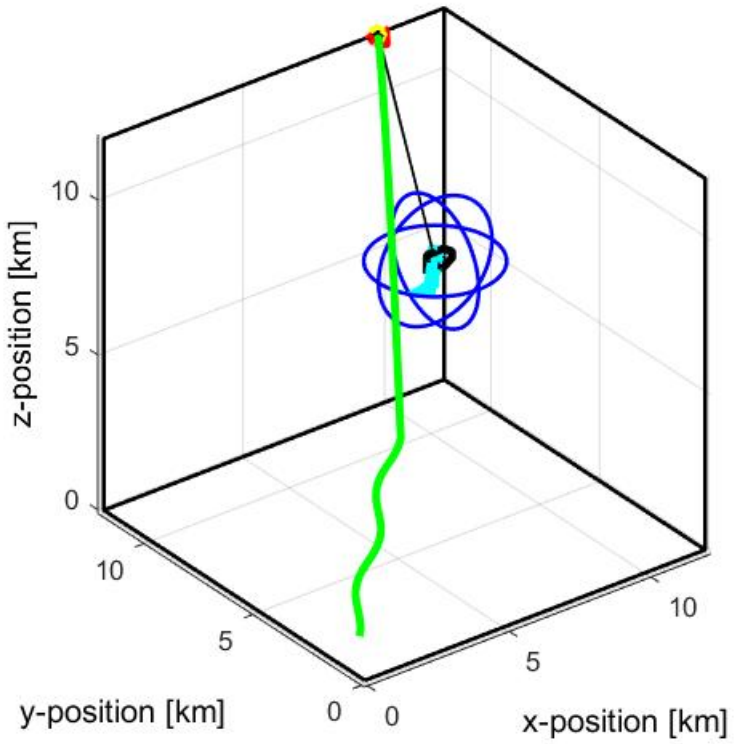


Figure 4.5: 3D simulation scenario.

The simulation consisted of a loop simulating each time step, so a sampling time 0.5 s was considered between them. The initial values of the Kalman filter had been initialized before the loop. In each loop, the range and velocity errors were calculated and the UAV movement was generated with the Lyapunov Vector Field Guidance (LVFG) according to the desired heading and pitch angles, that were set according to the direct range to the target point and modified if that was required to avoid a conflict. Then, bearing and elevation measurements were generated by taking the ground truth angles and adding Gaussian noise according to the specified measurements noise in the matrix R of the EKF. After that, the Kalman filter was executed so the relative range at that iteration loop was estimated. With that estimation, the conflict probability was calculated and the desired heading and pitch angles were generated taking into account whether that

probability was above or below a previously defined threshold. To calculate the conflict probability, a safe distance of 2 km between the obstacle and the UAV was considered.

In terms of trajectory, at the beginning there was some time when the UAV performed a sinusoidal trajectory and did not head towards the target point. The reason to set that trajectory was to ensure range observability. During this phase, bearing and elevation measurements were taken and EKF started working to estimate the range. The reason to use a sinusoidal trajectory was that the required manoeuvre to ensure range observability must have an acceleration component perpendicular to the LOS so the obstacle should have a sinusoidal trajectory too in the same direction that the UAV in order to avoid range observability, what was considered to be very unlikely. However, the sinusoidal trajectory was not the optimal one for this purpose. Active research is being carried out in this area and it was not the aim of this work to study the optimal trajectory to ensure observability, that would probably depend on the obstacle movement. Therefore, a sinusoidal trajectory in the heading direction was considered. The duration of this phase was set at 5 minutes and after that, the UAV guidance was activated so it went towards the target point.

4.5.2 Parametric study

There were a lot of design parameters in the simulation that would affect the collision avoidance performance with the developed technique. Hence, once the simulation had been checked to work correctly, it was desired to perform a parametric study to check the effect of some of these parameters on the avoidance performance, so they could be tuned accordingly in order to obtain the best performance as possible. For the study of each parameter, the same values for the rest of parameters were considered. Thus, the same object and UAV initial locations and target position were fixed instead of being generated randomly, in order to ensure the same conditions in all the simulations and that only the corresponding parameter influence was considered. The only random values allowed were the measurements generated at each time step; the mean value was constant but the measurements were randomly generated with normal distribution.

Conflict probability threshold

First, the conflict probability threshold at which the UAV must perform an avoidance manoeuvre was studied. Depending on this threshold, the UAV could perform the manoeuvre too late or too soon, so the trajectory could lose safety or be too conservative. The object was considered static and the UAV flying at a speed of 20 m/s. The UAV started at the origin and the target was at the opposite corner while the obstacle was

placed at the centre of the volume. Three different thresholds are shown in Figure 4.6: 10%, 50% and 90%. It can be seen that when the conflict probability threshold was high, the safe distance could not be met so their relative range was below the safe value for some period of time. The reason is that the UAV executed an avoidance manoeuvre too late, when its manoeuvrability was not enough to avoid the conflict. On the other hand, when the threshold was 10%, the safe range was not even reached but the trajectory would deviate too much from the direct one. With the threshold at 50%, the safe distance was respected and the original trajectory was not modified so much. From the conflict probabilities graph, it can be seen that higher probabilities were achieved when higher thresholds were considered. The reason is that at lower thresholds, the UAV executed an avoidance manoeuvre at an earlier stage so it did not allow that the probability reached so high. However, when the threshold was high, no manoeuvre was executed until there was an actual risk of conflict so the probability kept increasing due to the UAV approaching the obstacle with no avoidance manoeuvre.

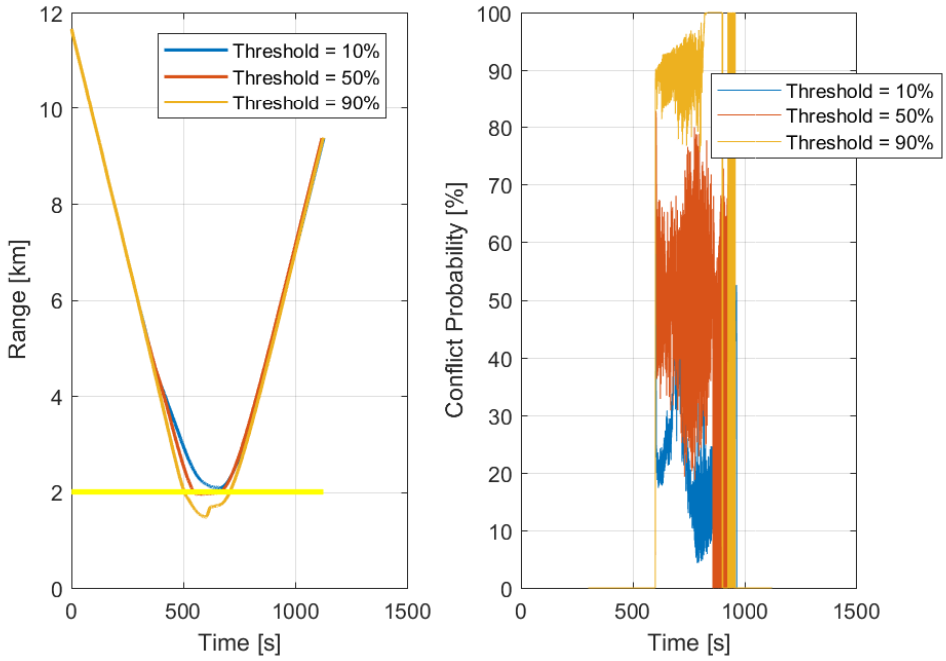


Figure 4.6: Relative range (left) and conflict probability (right) at different conflict probability thresholds.

An additional consideration is that at very low thresholds, the UAV would perform an avoidance manoeuvre when its future state propagation would not even be inside the safe sphere. Due to the geometric manoeuvre defined, the velocity vector would be modified so it was tangent to that sphere, thus approaching the obstacle and decreasing safety instead that increasing it. Therefore, low conflict probability threshold values would also have a penalty on safety.

Object and UAV speeds

The conflict probability threshold was set at 50% and different speeds were considered for the UAV and the obstacle. In Figure 4.7, the relative range and the conflict probability for different UAV speeds are shown. The main differences were about the required time to reach the target and the time the conflict was detected. At lower speeds, the conflict was detected later because the UAV approached the obstacle later and its probability remained high for longer since it took more time to avoid that conflict. When the speed was increased, this process was done faster but the conflict could not be avoided at 40 m/s as the safe distance limit was crossed. The reason is that at such high speed there was short time between the object detection and the conflict so when the range could be properly estimated, the UAV had already left the obstacle behind and the conflict probability was null. That would be a risk when flying at high speeds since some time is needed to estimate the relative range from the moment the object is detected. One solution could be to slow down the vehicle when an object has been detected so the range can be correctly estimated and then increase the speed once the conflict has already been avoided.

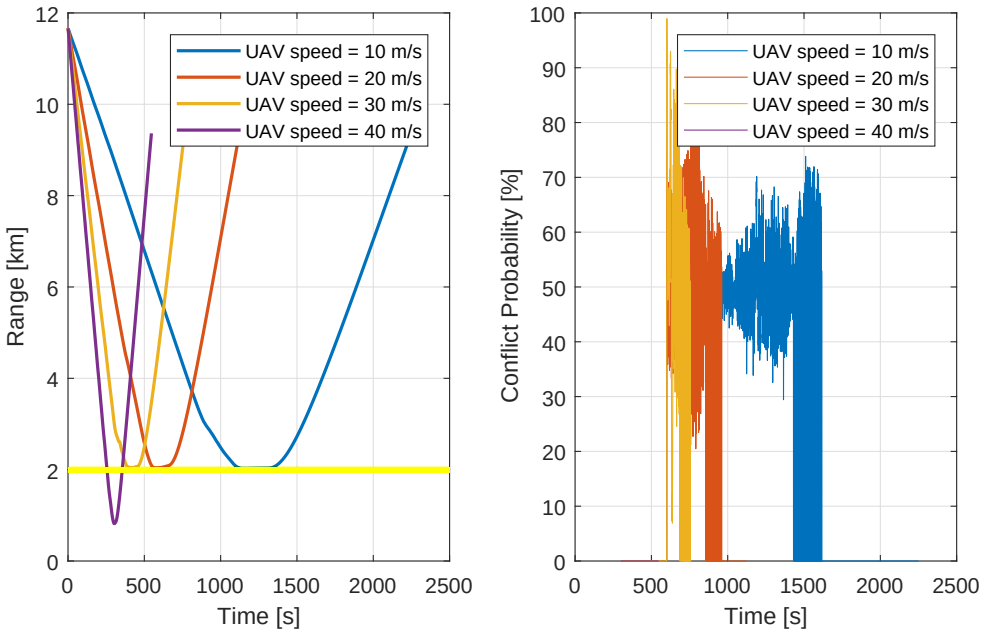


Figure 4.7: Relative range (left) and conflict probability (right) at different UAV speeds.

When a moving object was assumed, also its speed was considered. The estimated range error can be seen in Figure 4.8. When the object was moving, it was more complex to estimate the range since the UAV had also some movement and that complexity could be seen from the range error. Higher the speed, higher the error. Therefore, faster objects

would be more difficult to avoid due to the increased complexity in range estimation. A longer initial phase with a prescribed manoeuvre for range observability would help to decrease this error.

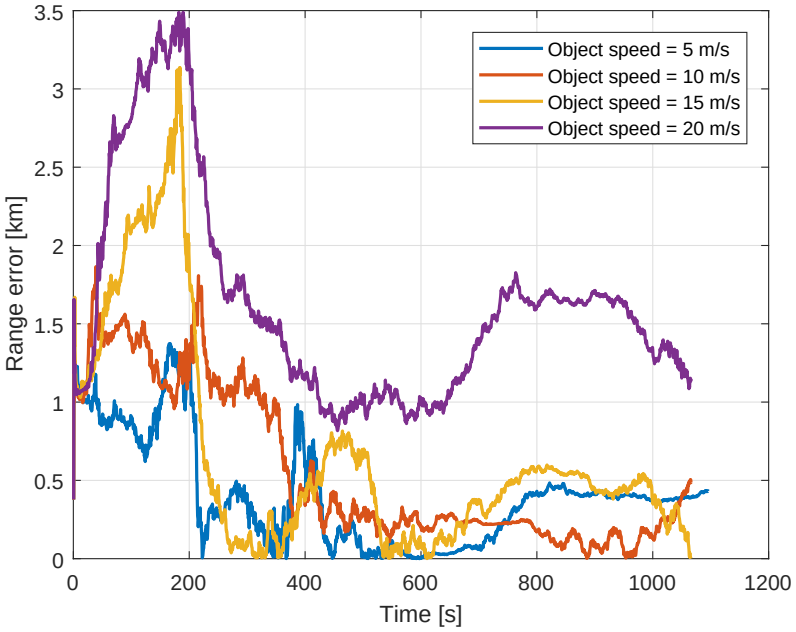


Figure 4.8: Estimated range error with different object speeds.

EKF noise

Focusing on the EKF, one of the keys for a good performance is a correct tuning of noise. In this application, the measurement noise would be given by the detection part of the algorithm with ReLO. However, it was desired to check how the EKF would work with different measurement and process noises. The rest of parameters were left as before, with a conflict probability threshold at the 50%, static object and UAV flying at 20 m/s. First, the process noise was set at 0.01 of standard deviation and the measurement noise was varied. The same noise was considered for both bearing and elevation measurements. In Figure 4.9, the estimated range error and the Kalman gain of the y component of the relative range as example are plotted for three different standard deviations of the measurements: 1, 5 and 10 degrees. When the measurement was of bad quality, that is, with high standard deviation, obviously the errors in the estimated range were greater than the ones with lower noise. However, with very low standard deviations, the estimated range could diverge as happened with 1 deg. That is because the process noise was fixed at 0.01 and if the measurement noise changes, thus the process noise should also be modified accordingly to achieve the best results. The optimum behaviour would happen with a standard deviation around 5 deg since the estimated range error

was low and decreased over time, when more measurements were available. Looking at the Kalman gain, it can be seen that higher values were reached when lower the standard deviations were. The reason is that, in such cases, the EKF trusted more the measurements than the predictions due to their low standard deviation so the Kalman gain was higher and the estimated range had more influence in the measurements when they were of higher quality. Anyway, it can be seen that their behaviour was the common one with any standard deviation so the Kalman gain was high at the beginning when there was almost no information about the process and tended to zero afterwards.

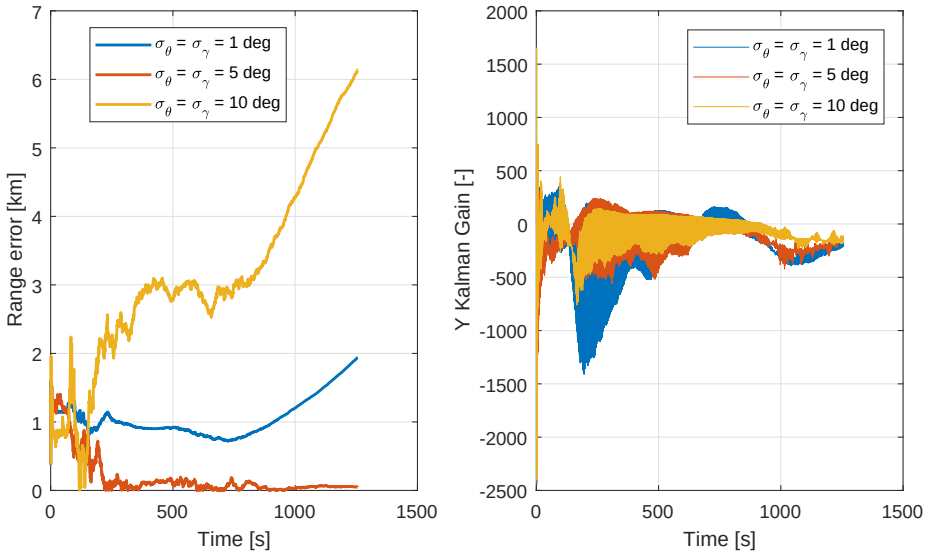


Figure 4.9: Estimated range error and Kalman gain of the y component of the relative range for different values of the standard deviation of both angle measurements.

To analyse the effect of the process noise, the standard deviation of the measurements were set at 5 deg and the process noise was modified. The results of 3 cases with different orders of magnitude of the process noise are shown in Figure 4.10. These results are similar to the measurement noise ones. Higher noises lead to higher errors but very low noises could cause the divergence as well as shown in the graph so it was important to find the optimum one. With respect to the Kalman gain, all cases followed the typical trend of Kalman gain decrement but the higher noise one had a peak after some time due to its divergence. It is also remarkable the practically constant and equal to zero value of the steady Kalman gain when the process noise was very low. The reason is that the EKF finally trusted completely the predicted states while the measurements were neglected.

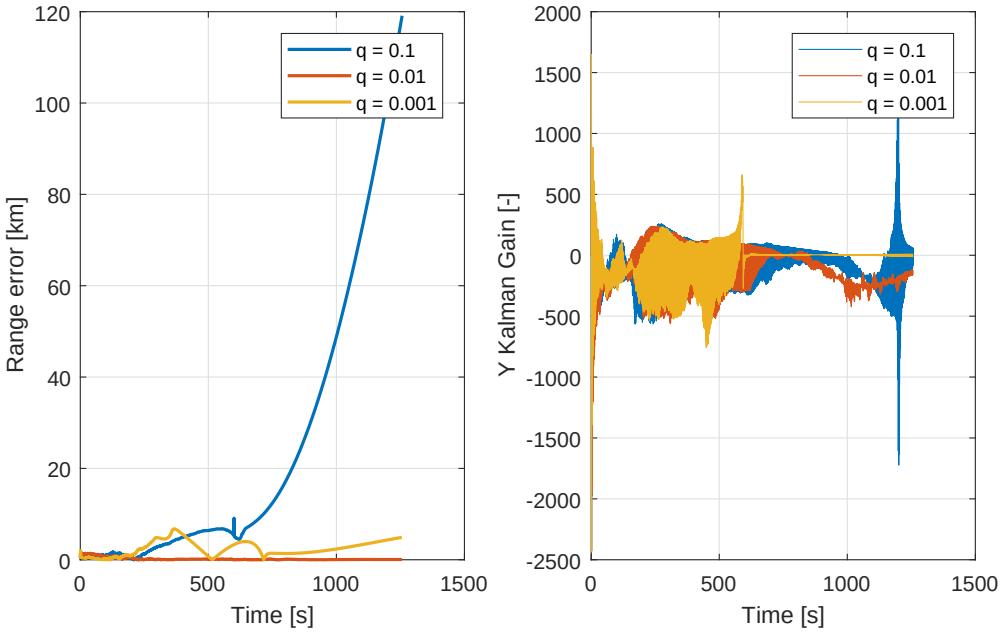


Figure 4.10: Estimated range error and Kalman gain of the y component of the relative range for different values of the standard deviation of the process.

4.5.3 Collision avoidance performance

After performing multiple simulations with different scenarios with random vehicle and object locations, target points and vehicle orientations and performing a parametric study of some of the most influencing parameters, it could be concluded that the conflict could be avoided with a correct tuning of the Kalman filter. In fact, the most important part of this method was the EKF. If the relative range was accurately estimated, the collision could be successfully avoided. Even if the EKF parameters were tuned correctly, some external parameters such as the object speed could make the range estimating task more difficult. Hence, the collision avoidance was highly dependent on the EKF. The conflict detection and resolution showed to be very reliable and even if it was desired to give increased safety, higher safe distances or lower conflict probability thresholds could help on that.

Moreover, there were some advantages of using this collision avoidance method with a detection and tracking technique based on neural networks. One was that the neural network gave a bounding box around the object so that gave some information about its size. According to this information, the safe distance could be set accordingly since it was defined from the centre point of the object. With that information, the safe distance could be calculated from the boundaries of the bounding box and not from the object centre. An additional advantage is that neural networks also provided class information

about the object. For example, the dynamics of a bird and an aircraft would be different so different noises would be needed to tune the EKF because an aircraft would be known to fly at an approximately constant and linear speed while birds' movements are more chaotic and the magnitudes of their speeds are different. With other methods, the class would not be identified so general initial states, covariances or process noises would be given. However, with this kind of technique, those parameters could be adjusted according to the detected class, improving the estimation performance.

Chapter 5

Experimental test

5.1 Objective

The Sense and Avoid algorithm was divided into two different parts: the sensing part and the avoiding part. Each one of them was designed separately and assessed in different ways. The object detection and tracking algorithm was assessed with pre-recorded videos whereas the collision avoidance algorithm was tested with simulations. The aim of this experimental test was to integrate both of them so information of the first part was used in the second one.

When developing the simulation of the collision avoidance part, guessed values of the measurement and process noises were considered in the EKF. Actually, measurement noise is defined by the sensor technique while process noise is a design parameter to optimize the filter behaviour. Therefore, the objective was to obtain the measurement noise from the detection and tracking part of the algorithm. Once the real noise of the measurement process was experimentally calculated, it was used to test the EKF again but using this last value, since the measurement noise had been guessed in the previous demonstration. By this method, it could be checked whether the collision avoidance technique could be successful with the considered deep learning measuring process.

5.2 Test scenario

The test was carried out in the flying lab at Cranfield University. This room was equipped with a VICON navigation system as shown in Figure 5.1. That system consisted of multiple high-resolution cameras surrounding the capture volume. The VICON system

can locate certain retroreflective markers with very high accuracy (a few millimetres) and very low latency (1.5 ms). Hence, putting those markers on the objects in the capture volume, it was possible to obtain a very accurate detection and tracking of them. Due to its high accuracy, VICON measurements were considered as ground truth. That was the reason why the experimental test was carried out in that location, so ground truth was available.

Figure 5.1: Flying lab at Cranfield University equipped with a VICON navigation system.

A vehicle as the one in Figure 5.2 was placed in the volume area. Since the objective was to check the object detection and range estimation, a rover was used in this test. Hence, 2D dynamics were considered instead of 3D, although the results could be extrapolated to the 3D easily later. The vehicle was equipped with an optical camera onboard that would record the data that would be processed by the developed algorithm. The camera was fixed to the vehicle and as aligned as possible with the vehicle orientation. The camera model was a GoPro Hero 5 Black. Data was recorded at 4k resolution (3840×2160), at its wide FOV and at 30 fps.

Figure 5.2: Rover vehicle used at the flying lab at Cranfield University.

The detected object working as an obstacle was a desktop computer keyboard. Prior tests were carried out with the GoPro camera and several objects in order to check which was the most convenient since, due to the training of the neural networks, some objects are easier to detect than others. Finally, due to its simplicity of handling and transportation and the good performance with ReLO, a keyboard was used.

Therefore, the keyboard would be placed statically in a corner of the squared capture area on the floor, which had a size of around 4×4 m^2. The vehicle would also be placed inside the capture area with the obstacle within the camera FOV. Both the vehicle and the obstacle had some markers attached around them so they could be detected by the VICON system. The vehicle would move around the capture area while the GoPro camera onboard and the VICON system were recording data. The VICON system recorded position and orientation data from both the vehicle and the obstacle so information from the vehicle was considered as known data for the measurements processing while obstacle information was used to calculate the ground truth. The onboard camera would detect and track the obstacle getting only bearing measurements since elevation measurements were discarded due to the 2D simplification. Both VICON and camera measurements were post-processed offline once the experimental test was over.

5.2.1 Test cases

5 different test cases were considered: 1 static and 4 dynamic. The first test case consisted of the vehicle remaining static while recording the obstacle with the onboard camera during 30 s. The objective of this test case was to detect any measurement offset due to any misalignment between the camera and the vehicle or between the location of the markers and the true orientations. The rest of test cases consisted of different trajectories of the dynamic vehicle. Different trajectories were considered so noise information could be extracted from different vehicle behaviours and, then, it could be more robust. However, there were some limitations about available space and vehicle manoeuvrability so only very simple trajectories were possible to perform. In any case, four different trajectories were designed in order to calculate the measurement noise with each one of them. The performed trajectories are shown in Figure 5.3. Each one of the trajectories was repeated 3 times trying to be as similar as possible. The reason to do that was to have more robust information about noise. Thus, in total, 13 tests were carried out.

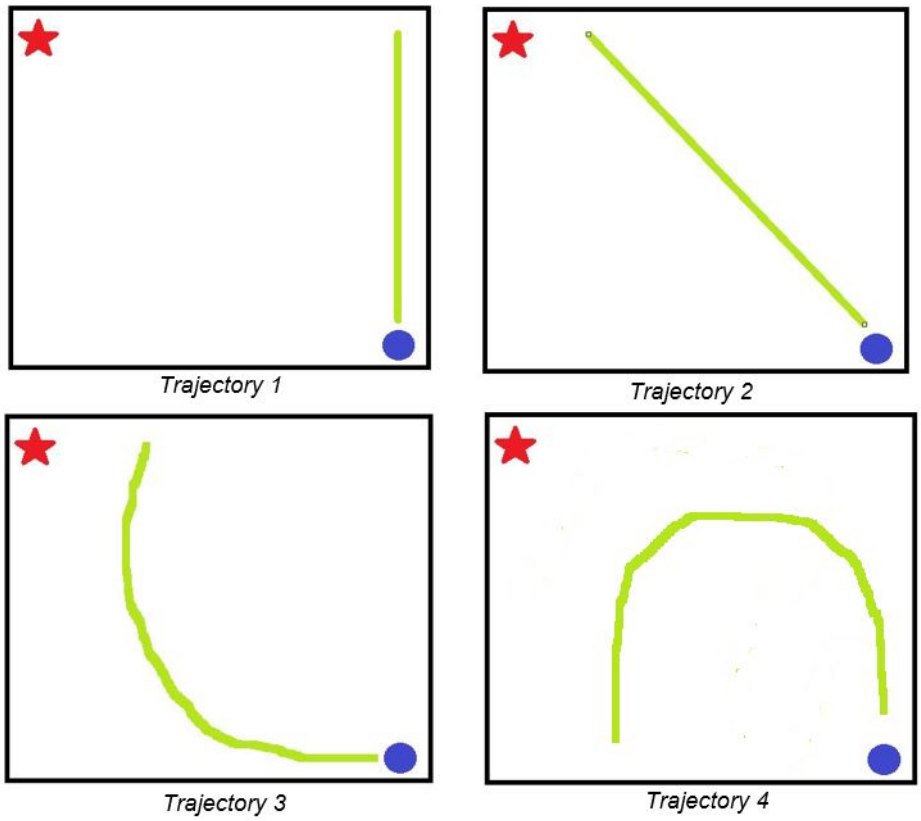


Figure 5.3: Tested vehicle trajectories.

5.2.2 Limitations

The experimental tests carried out were very limited by the available scenario. With respect to the test cases design, there were some considerations that limited the testing possibilities. For example, the capture area was small so it did not allow complex manoeuvres because there was not enough space. Furthermore, the vehicle manoeuvrability was very limited. It was very sensitive to accelerations so it was difficult to make it accelerate slowly and it was complex to perform proper curves with the controls. Although more complex trajectories were tried, they were impossible to perform so eventually very basic trajectories were considered.

An additional important limitation was about the bearing measurement. The rover did not have a specific platform to fix the camera so it had to be fixed with simple materials such as tape. Although that was done as strongly as possible, due to the fast accelerations of the rover, it was possible that the camera could rotate some millimetres modifying the detected object bearing. Furthermore, the rover's attitude taken from VICON data was considered to convert the measured bearing from the camera into the relative bearing independent of the rover orientation. Although that data is very

accurate in terms of position, the attitude error can be a bit higher since it depends on the markers location. They were put as symmetrically as possible but small error could be present.

5.3 Camera calibration

In order to obtain measurements as accurate as possible, a calibration process was performed before handling camera data. Camera calibration was important in order to obtain distorsion coefficients and intrinsic parameters such as the focal length and the optical centre.

There are two types of distorsion: radial and tangential. The radial distorsion is represented as:

$$x_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{5.1}$$

$$y_{distorted} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{5.2}$$

While the tangential distorsion is represented as:

$$x_{distorted} = x + [2 \cdot p_1 \cdot x \cdot y + p_2(r^2 + 2 \cdot x^2)] \tag{5.3}$$

$$y_{distorted} = y + [p_1(r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y] \tag{5.4}$$

Therefore, there were 5 distorsion coefficients that were determined by calibration, which were k_1, k_2, p_1, p_2 and k_3. Also, the intrinsic parameters were obtained. These parameters are the focal lengths and the optical centre and are represented by the camera matrix:

$$camera \quad matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

The camera calibration was carried out making use of OpenCV 3.4.1. In order to obtain all the required parameters, a set of images with a defined pattern were provided. There were two possible patterns: a chess board or a circular grid. In this case, a chess board was used so it was printed and attached to a rigid body before recording a video with the GoPro Hero 5 Black moving the chess board around, reaching the limits of the FOV and rotating it. Then, images were taken from that video and an OpenCV function detected the corner points between squares as shown in Figure 5.4.

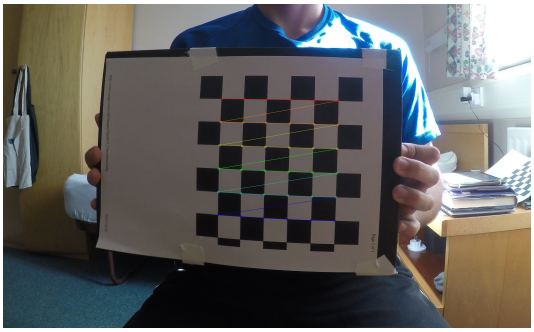


Figure 5.4: Example of corner detection for camera calibration using OpenCV 3.4.1.

With the data gathered from the corners detection, there was an additional OpenCV function that calculated directly the distorsion coefficients and the camera matrix. The obtained distorsion coefficients were:

$$
\begin{aligned}
k_1 &= -0.276 \\
k_2 &= 0.0924 \\
p_1 &= 0.000391 \\
p_2 &= -0.001136 \\
k_3 &= 0.01086
\end{aligned}
$$

From these coefficients, the frames taken from the camera could be undistorted before processing them so data would be more accurate. The parameters of the camera matrix in pixels were:

$$
\begin{aligned}
f_x &= 1782.9 \\
f_y &= 1782.4 \\
c_x &= 1943.8 \\
c_y &= 1034.8
\end{aligned}
$$

With these values, some additional extrinsic parameters from the camera dependent on the resolution and the sensor size could be determined. In this case, 4k resolution corresponding to 3840×2160 pixels was used and the sensor size was 6.17×4.55 mm^2. The additional parameters were the FOV in both horizontal and vertical directions, the total focal length, the principal point and the aspect ratio, which were given as:

$$
\begin{aligned}
FOV_x &= 94.2 \quad \text{deg} \\
FOV_y &= 62.4 \quad \text{deg} \\
f &= 16.16 \quad \text{mm} \\
c &= (17.62, 12.29) \quad \text{mm} \\
AR &= 0.99975
\end{aligned}
$$

5.4 Results

Once the data had been gathered and the camera calibrated, a post-processing method was performed. First, the camera measurements were extracted by running the recorded videos through the designed algorithm. Then, measurement noise was obtained by comparing the measured results with the ground truth data. After that, the process noise of the EKF was accordingly tuned and it was checked that collisions could be avoided with the noise associated to the considered detection and tracking mechanism.

5.4.1 Camera measurements

The recorded videos were run through ReLO in order to generate the bounding box around the keyboard and to track it. The first step was to check that the keyboard could be detected. Despite the previous checking with the camera, the different orientations or light conditions could affect the image features so it was possible that the keyboard

could not be detected in the flying lab. However, it was checked that it was correctly detected and tracked as shown in Figure 5.5.

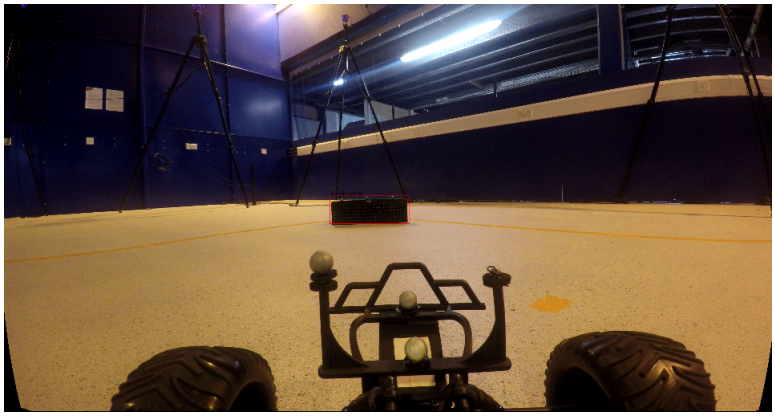


Figure 5.5: Keyboard detection from experimental test using ReLO.

Once the bounding box had been generated, the angle measurements were obtained as explained before so only bearing measurements were taken into account due to the 2D assumption. Since the EKF considered the relative bearing without taking into account the vehicle attitude, the directly measured bearing was corrected with the current azimuth angle of the rover.

5.4.2 Static test case

First, the static test case was processed. This case was used to detect any offset between the measurement and the ground truth, so it was important to compare results later. The rover did not move in this test case, so the relative bearing should be constant. In Figure 5.6, both the measured and ground truth bearings are represented as well as the absolute error between them. Hence, it could be seen that the approximated offset between measurement and ground truth was about 8.25 deg so this modification would be added to the dynamic measurements in order to have comparable magnitudes. This error is mainly due to two misalignment errors: one due to the camera position with respect to the true rover orientation and the other due to the chosen markers position that give an orientation with some error with respect to the true rover orientation. If the errors were in the same direction, they could compensate each other but, if they are in opposite directions, the error is magnified. The markers position on the keyboard could also influence this error since the ground truth could be slightly different to the mid-point of the measured bounding box, and small errors could give bigger angle errors at long distances.

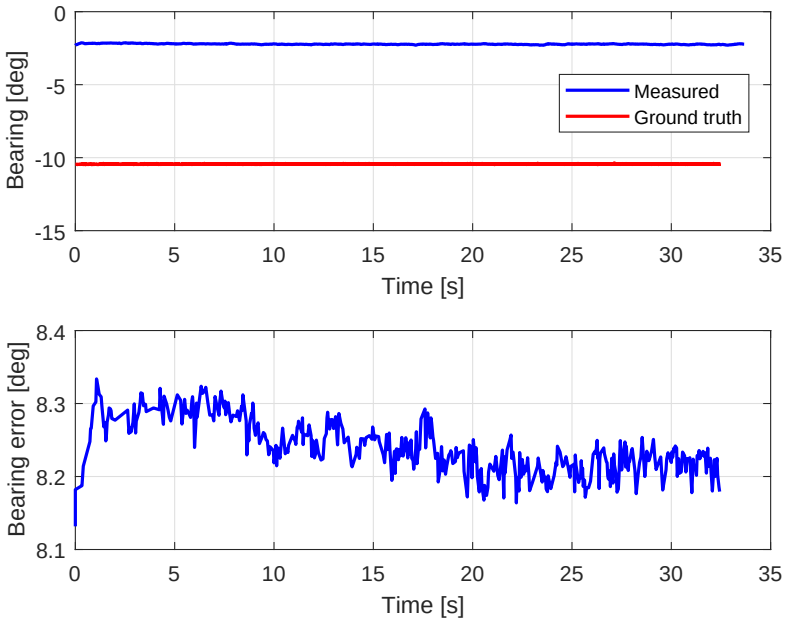


Figure 5.6: Measured and ground truth bearings (up) and absolute bearing error (down) of the static test case.

5.4.3 Measurement noise

All the dynamic test cases were performed in order to compare the measured bearing angles with the ground truth. The 8.25 deg offset measured from the static case was added to all the dynamic tests measurements. In Figure 5.7, an example of the result obtained from a test case of the Trajectory 4 is shown. Both the ground truth obtained with the VICON system and the relative bearing obtained from the angle measurement are represented as well as the angle error between them. It can be seen that the measurement noise using the considered deep learning technique is much greater that the one using a high precision sensor system such as VICON.

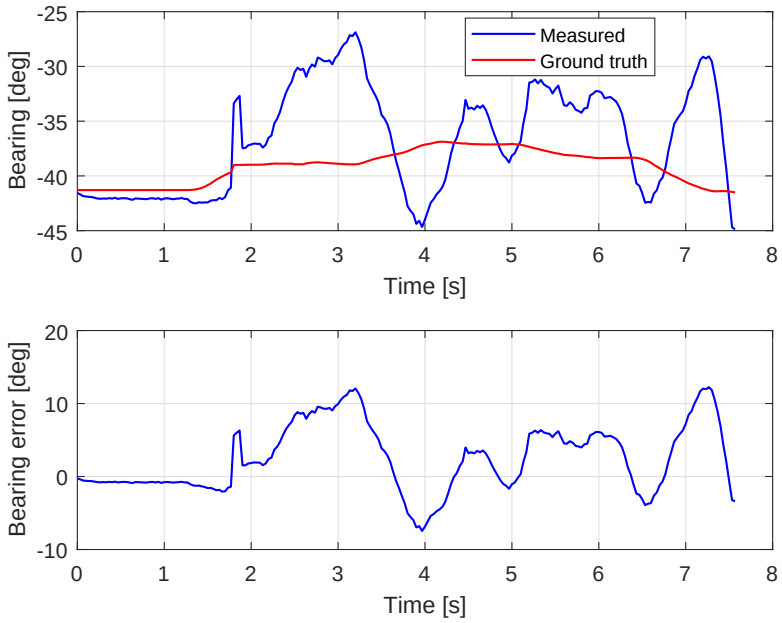


Figure 5.7: Measured and ground truth bearings (up) and absolute bearing error (down) of the Trajectory 4.

An additional consideration was given in some test cases such as in the Trajectory 1 as shown in Figure 5.8. It can be seen that the error was higher at the end of the recording time. The reason was that the object was near the limit of the camera FOV so only part of the object was inside the frame. However, ReLO was capable to keep tracking it although the generated bounding box was not around the whole object but around the visible part of the object. Since the measured angle was obtained from the centre point of the bounding box, it did not match with the real location of the whole object. Therefore, the error of the angle measurement was higher when only part of the object was inside the frame and, hence, the measurement accuracy was reduced when the object was near the FOV limits. However, this effect only happened in some of the test cases since, in some others, the object could not be detected if it was not entirely inside the frame so this error did not appear but the object was tracked for shorter time. In the cases where this effect happened, the measurements were subsampled so this last part was not considered in order to obtain its standard deviation.

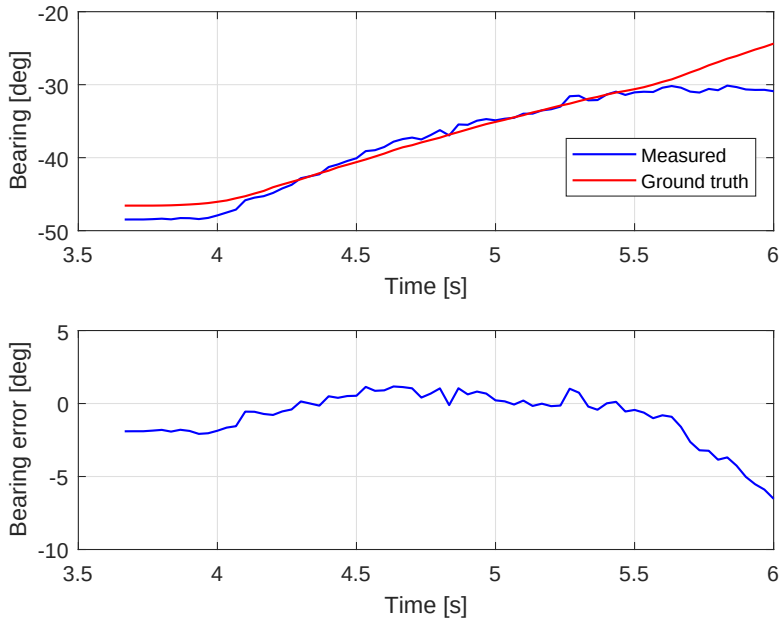


Figure 5.8: Measured and ground truth bearings (up) and absolute bearing error (down) of the Trajectory 1.

Therefore, the standard deviations of the bearing errors of each of the dynamic test cases were obtained, giving the results of Table 5.1. It can be seen that there was some dependency on the trajectory. For example, the measurements extracted from the test cases of the Trajectory 3 were the noisiest. The reason is that the manoeuvre in those cases were more complex so the bearing angle measured by the camera changed quickly. Thus, ReLO accuracy was affected since the considered past measurements were more different between them. Furthermore, due to the manoeuvrability of the rover, turns were very aggressive so the camera could experience a slight rotation. The standard deviation associated to the measurement process was calculated as the average of all the standard deviations obtained from the test cases. Hence, the result was a standard deviation of $\sigma_\theta = 4.97$ deg.

Table 5.1: Bearing measurement standard deviation for each test case.

Test	Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
Cases	1	2	3	4	5	6	7	8	9	10	11	12
Standard deviation (deg)	1.03	3.77	2.06	2.73	2.85	5.41	12.62	7.49	10.01	4.80	3.81	3.12

5.4.4 Process noise tuning

The measurement noise information obtained from the experimental tests was used to run again the designed EKF so the process noise could be tuned accordingly and the true behaviour associated to the real performance of the measuring process could be considered. Therefore, the 2D case was considered in this case since only bearing measurement noise was available. The standard deviation was directly introduced as 4.97 deg so the measurement matrix, that was a scalar number in this case due to the 2D assumption, was fixed. Then, different values of the process noise were tested in order to check which one gave the best results. The results obtained with a static obstacle are shown in Figure 5.9. It can be seen that the estimated range diverged when the process noise had a standard deviation of 1. With smaller standard deviations, the results were very similar and accurate from a standard deviation of 0.1. Therefore, that would be an acceptable value for the process noise.

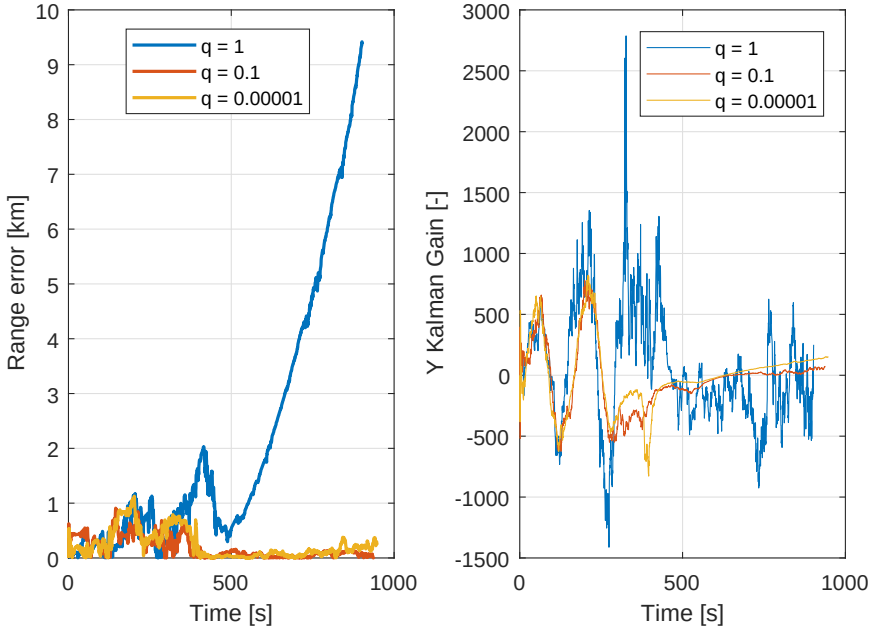


Figure 5.9: Range errors (left) and y position Kalman gains (right) with an static object for different process noise.

In Figure 5.10, the results of different process noises are shown for the case when the obstacle is moving with constant velocity. Similar results to the static case were obtained. However, it can be seen that the range error diverged after some time when the process noise was 0.1. Hence, it could be concluded that the very small process noise should be considered in such cases.

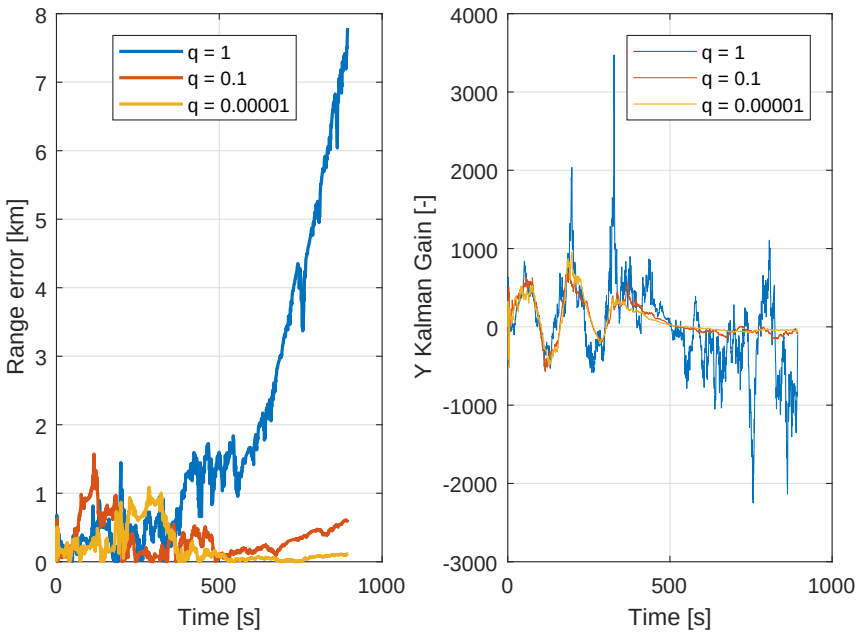


Figure 5.10: Range errors (left) and y position Kalman gains (right) with a constant velocity object for different process noise.

However, the previous scenarios were very simple. The behaviour of the EKF was also studied in a case when the velocity changed at some point. An additional simulation was carried out in which the obstacle started moving with constant velocity in the y direction and suddenly changed to the x direction. The results for three different process noises can be seen in Figure 5.11. For small process standard deviation such as 0.01, the EKF tried to follow the considered process so when it changed, as happened with the change of velocity direction, the range error increased and it also diverged later. When the process standard deviation was high such as 1, the EKF relied on the measurements so the range error did not increase when the velocity changed but it was noisier and it diverged. It can also be checked that the Kalman gain did not decrease completely to zero but it kept considering measurements so that is why the range error kept noisy. When the process noise was 0.1, however, the range error increased a bit when the velocity changed but it quickly decreased again, keeping small after that and not so noisy as the case with a standard deviation of 1.

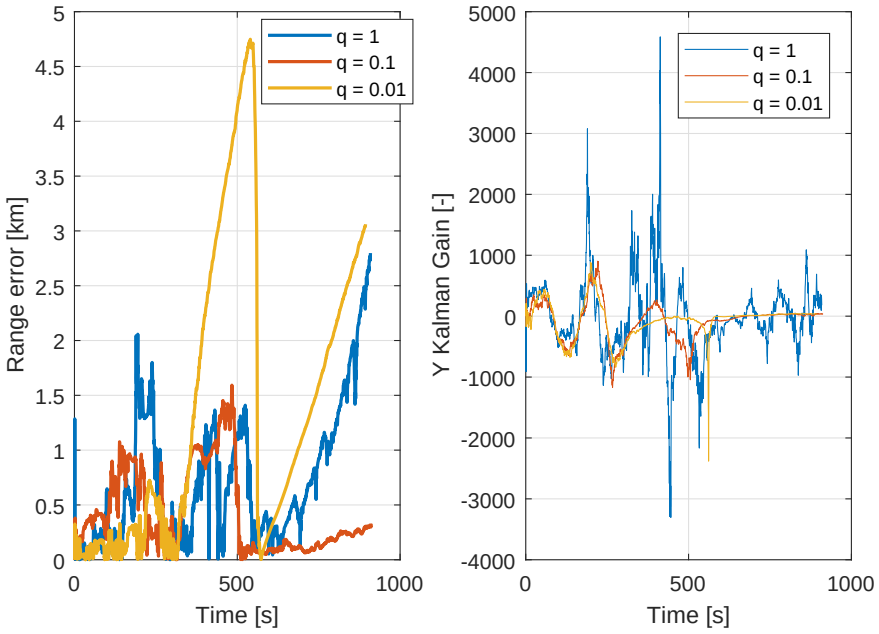


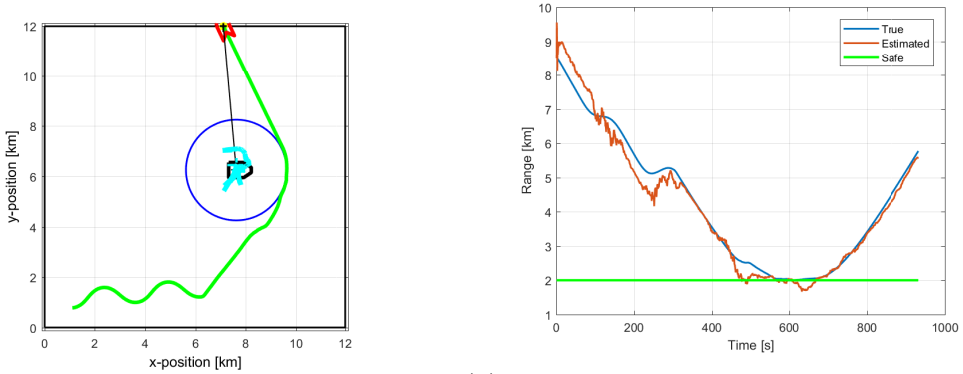
Figure 5.11: Range errors (left) and y position Kalman gains (right) with a varying velocity object for different process noise.

Therefore, according to the performed simulations, the best range estimation performance was achieved when the process noise was considered as 0.1. However, there can be some cases when different process noises can be more convenient, as happened with the constant velocity case when the smaller the process noise, the smaller the range error. One advantage of using neural networks for the detection part of the algorithm is that class information is available. Hence, it is known what kind of obstacle is being tracked. Therefore, the process noise could be considered as 0.1 as general basis but could be adjusted accordingly to the detected object. For example, if an aircraft in the high airspace was detected, it would be likely to fly at constant velocity for the time it remained in the UAV FOV. Thus, the process noise considered in the EKF in this case could be adjusted using smaller values so better range estimations could be obtained.

5.4.5 Collision avoidance check

Once the Kalman filter was tuned with the measurement noise associated to the detection and tracking process performed with ReLO, the rest of the collision avoidance technique was also simulated in order to check that conflicts could be avoided with this method. Measurement noise of 4.97 deg and process noise of 0.1 were considered. Several simulations were run with random conditions of the UAV and object initial positions, orientations and target positions. It was possible to confirm that the conflict was always

avoided when considering a conflict probability threshold of 50% as shown in Figure 5.12. Although the estimated range was below the safety distance for a short time, this conflict probability threshold allowed that the true range was always above this value so it could also help to enhance robustness to measurement or estimation noise. However, as it was explained before, this threshold could also be tuned to enhance safety.

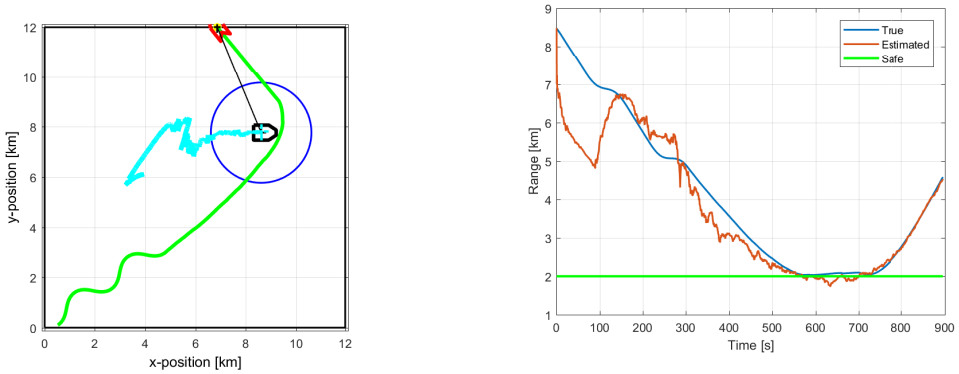


(a) Considered scenario.

(b) True and estimated relative range and safety distance.

Figure 5.12: Collision avoidance check with static object.

Collision avoidance was also checked for the case of an object with constant positive velocity in the x direction as shown in Figure 5.13. Therefore, the conflict was also avoided when the detected object was moving and results showed that the most important stage was the range estimation, since collision could always be avoided if range was correctly estimated.



(a) Considered scenario.

(b) True and estimated relative range and safety distance.

Figure 5.13: Collision avoidance check with dynamic object.

Chapter 6

Closing remarks

6.1 Conclusions

Hybrid Convolutional and Recurrent Neural Networks are an emerging kind of neural networks with very attractive characteristics for Sense and Avoid applications. Small UAVs that would be equipped with only a visual camera could take advantage of them. Therefore, a complete Sense and Avoid technique was developed making use of this kind of method in order to identify possible issues and assess feasibility.

A detection and tracking model was designed based on two open source codes: the Convolutional Neural Network YOLO to detect objects and the hybrid Convolutional and Recurrent Neural Network Re^3 to track them. The resulting algorithm was called ReLO and was capable to correctly detect and track multiple objects in complex environments. A refresh parameter was introduced in order to get a better trade-off between speed and detection performance. This parameter should be adjusted accordingly to the working environment. The detection capability depended on YOLO, which could have misdetection or wrong-class detections, whereas tracking relied on Re^3, which had tracking limitation with very fast objects. In terms of an important requirement in Sense and Avoid applications such speed is, ReLO showed to work more than 4 times faster than the fastest CNNs.

The collision avoidance technique was also designed in order to complete the Sense and Avoid algorithm. Angle measurements were obtained from the detection data extracted with the neural networks and they were used to estimate the relative range between the object and the UAV using an Extended Kalman Filter. Then, the conflict was assessed by an analytical probabilistic method and an avoidance trajectory based on a geometrical approach was generated. The resulting performance showed to be highly

dependent on the range estimation done by the Kalman filter, since the collision could be reliably avoided once the range had been estimated correctly. Therefore, the parameters in the EKF must be accurately tuned according to the measurements and the scenario. Furthermore, some parameters such as the safe distance for conflict and the conflict probability threshold could be adjusted according to the required safety level.

Experimental tests were also performed. Real measurement noise information associated to angle measurements extraction with ReLO was obtained. Then, the EKF was tuned according to this true measurement noise so the collision avoidance behaviour could be checked with this additional information. It was confirmed that collisions could be successfully avoided with the noise associated to ReLO but further simulations in different scenarios would be needed to check its robustness.

In general, hybrid Convolutional and Recurrent Neural Networks showed to be a promising tool in the considered context. Speed was increased with respect to the most traditional neural networks used for image processing while the detection performance was kept. Therefore, further research on this kind of neural networks is recommended since it was shown that it could be feasible to use them in Sense and Avoid applications. The most important dependencies found were the detection performance, which is a field where there is much research currently being developed, and the range estimation, where different filters or approaches could also be studied.

6.2 Further research

Since hybrid Convolutional and Recurrent Neural Networks are very new, its application to Sense and Avoid systems is in a very early stage. The aim of this project was to study the feasibility of this kind of technique. Therefore, the most important limitations have been identified and the corresponding research areas can be proposed.

With reference to the detection and tracking algorithm, further research should be carried out on hybrid CRNN. Recently, there has been much research in common neural networks such as CNN and RNN, specially CNN in computer vision. However, the capabilities of CRNN have not been widely explored yet. It was difficult to find available codes for this purpose while all the best CNNs are open source. If approaches such as ReLO would be preferred, special attention would be required in YOLO since tracking works fast and accurately but it depends on the initial bounding boxes provided by the CNN. Improving the detection performance of neural networks by increasing reliability and reducing noise would lead to much better results. Moreover, it will be needed to assess the detection performance so quantitative results can be provided in order to make

it easier to compare them. Temporal occlusions and rotations should also be considered so one of the most important advantages of hybrid CRNN can be evaluated.

The simulations performed with the collision avoidance technique showed that they could have a very good performance. However, the variety of simulations was limited. More simulations in different scenarios are required in order to assess performance in other situations such as with multiple objects with different dynamics, since in this case only a single object with linear velocity was considered. Furthermore, the optimal trajectory to ensure range observability should also be studied.

As with simulations, additional experimental tests would also be required. Before testing the whole algorithm, some part of it could be tested first. Experimental tests with a moving vehicle detecting and tracking an obstacle and estimating the relative range should be carried out in order to check the correct EKF performance. Different scenarios with different objects and vehicles trajectories and speeds should be considered. Then, multiple objects could be introduced.

It would also be convenient to assess the algorithm performance as a whole. Two different approaches are proposed. The first one is to test the complete Sense and Avoid technique experimentally. However, that should be done in a reliable way so there is no harm in case that the system fails. Therefore, first the algorithm should be tested in the same processing unit that the vehicle in order to check that it can work on real time. Then, during the test, additional accurate sensors should be used for safety reasons. The vehicle would trust only the algorithm results except when a safety-critical condition is detected with some of the other sensors, so safety is ensure in case that the proposed method is not working. Furthermore, the vehicle should be a UAV and the obstacles should be objects that a UAV could find while operating. The second approach to test the overall technique would be with Virtual Reality (VR). This would be safer since there would not be real equipment in danger. The camera on the UAV would sense the virtual environment and the corresponding control input would be applied when required to avoid a collision, so the virtual vehicle would perform the avoidance manoeuvre. However, this environment would need to have available multiple objects that the UAV could find in reality, as well as to be as close to reality as possible, so results could be generalised. Nevertheless, there are no open source Virtual Reality environments available that are advanced enough to carry out this kind of tests. In any case, further research will be needed previously to find the best methods to perform each of the stages in the Sense and Avoid algorithm. Once there is strong confidence about every stage, the complete algorithm should be tested. This process could take some years that will probably bring Virtual Reality developments.

Bibliography

[1] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," pp. 1–11, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[7] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent Advances in Recurrent Neural Networks," pp. 1–21, 2017.

[8] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[9] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[10] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking," 2016.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[12] D. Gordon, A. Farhadi, and D. Fox, "Re3 : Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects," pp. 1–8, 2017.

[13] J. H. Lee and K. G. Kim, "Applying Deep Learning in Medical Images : The Case of Bone Age Estimation," vol. 24, no. 1, pp. 86–92, 2018.

[14] J. Krozel and M. Peters, "Strategic conflict detection and resolution for free flight," , {*Proceedings*} *of the 36th* {*IEEE*} {*Conference*} *on* {*Decision*} *and* {*Control*}*, 1997*, vol. 2, no. December, pp. 1822—-1828 vol.2, 1997.

[15] T. Hutchings, S. Jeffryes, and S. J. Farmer, "Architecting UAV sense & avoid systems," in *Proc. IET Conf. Autonomous Systems*, no. 1, pp. 1–8, 2007.

[16] A. Mcfadyen and L. Mejias, "A survey of autonomous vision-based See and Avoid for Unmanned Aircraft Systems," *Progress in Aerospace Sciences*, vol. 80, pp. 1–17, 2016.

[17] J. Boskovic, J. A. Jackson, and R. Mehra, "Sensor and Tracker Requirements Development for Sense and Avoid Systems for Unmanned Aerial Vehicles," *AIAA Modeling and Simulation Technologies (MST) Conference*, pp. 1–19, 2013.

[18] Š. Kopřiva, D. Šišlák, and M. Pěchouček, "Sense and Avoid Concepts: Vehicle-Based SAA Systems (Vehicle-to-Vehicle)," *Sense and Avoid in UAS: Research and Applications*, pp. 143–173, 2012.

[19] B. M. Albaker and N. A. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles," *International Conference for Technical Postgraduates 2009, TECHPOS 2009*, no. January, 2009.

[20] A. Zhahir, A. Razali, and M. R. Mohd Ajir, "Current development of UAV sense and avoid system," *IOP Conference Series: Materials Science and Engineering*, vol. 152, no. 1, 2016.

[21] B. C. Karhoff, J. I. Limb, S. W. Oravsky, and A. D. Shephard, "Eyes in the domestic sky: An assessment of sense and avoid technology for the army's "warrior" unmanned aerial vehicle," *Proceedings of the 2006 IEEE Systems and Information Engineering Design Symposium, SIEDS'06*, pp. 36–42, 2007.

[22] G. Recchia, G. Fasano, D. Accardo, A. Moccia, and L. Paparone, "An optical flow based electro-optical see-and-avoid system for UAVs," *IEEE Aerospace Conference Proceedings*, pp. 1–9, 2007.

[23] X. Prats, L. Delgado, J. Ramírez, P. Royo, and E. Pastor, "Requirements, Issues, and Challenges for Sense and Avoid in Unmanned Aircraft Systems," *Journal of Aircraft*, vol. 49, no. 3, pp. 677–687, 2012.

[24] J. D. Griffith, M. J. Kochenderfer, and J. K. Kuchar, "Electro-Optical System Analysis for Sense and Avoid," *Proc of AIAA Guidance, Navigation and Control Conference and Exhibit*, no. August, pp. 18–21, 2008.

[25] C. Geyer, D. Dey, and S. Singh, "Prototype Sense-and-Avoid System for UAVs," *Report*, no. February, 2009.

[26] R. Carnie, R. Walker, and P. Corke, "Image processing algorithms for UAV "sense and avoid"," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 2848–2853, 2006.

[27] L. Mejias, J. J. Ford, and J. S. Lai, "Towards the implementation of vision-based UAS sense-and-avoid," *Proc of the 27th Int Congress of the Aeronautical Sciences*, no. September, pp. 19–24, 2010.

[28] L. Mejias, A. McFadyen, and J. J. Ford, "Sense and avoid technology developments at Queensland University of Technology," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 7, pp. 28–37, 2016.

[29] L. Muratet, S. Doncieux, Y. Briere, and J. A. Meyer, "A contribution to vision-based autonomous helicopter flight in urban environments," *Robotics and Autonomous Systems*, vol. 50, no. 4, pp. 195–209, 2005.

[30] W. Green, P. Oh, and G. Barrows, "Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pp. 2347–2352 Vol.3, 2004.

[31] Oh, "Closed Quarter Aerial Robot Prototype to Fly In and Around Buildings Paul Y . Oh , William E . Green and Geoffrey Barrows,"

[32] Z. Ma, T. Hu, L. Shen, W. Kong, and B. Zhao, "A detection and relative direction estimation method for UAV in sense-and-avoid," *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*, no. August, pp. 2677–2682, 2015.

[33] S. Hwang, J. Lee, H. Shin, S. Cho, and D. Shim, "Aircraft detection using deep convolutional neural network for small unmanned aircraft systems," *AIAA Information Systems-AIAA Infotech at Aerospace, 2018*, no. 209989, 2018.

[34] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *Journal of Sensors*, vol. 2017, 2017.

[35] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude UAV imagery with the faster R-CNN," *Journal of Advanced Transportation*, vol. 2017, 2017.

[36] S. Han, W. Shen, and Z. Liu, "Deep Drone: Object Detection and Tracking for Smart Drones on Embedded System," pp. 1–8, 2016.

[37] M. Radovic, O. Adarkwa, and Q. Wang, "Object Recognition in Aerial Images Using Convolutional Neural Networks," *Journal of Imaging*, vol. 3, no. 2, p. 21, 2017.

[38] O. Shakernia, W.-z. Chen, and M. V. M. Raska, "Passive Ranging for UAV Sense and Avoid Applications," *AIAA InfotechAerospace 2005*, no. September, pp. 1–10, 2005.

[39] A. Reshma, S. Anooja, and E. G. Deepa, "Bearing Only Tracking using Extended Kalman Filter," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 1140–1144, 2013.

[40] A. Ahmadian and A. M. Pezeshk, "A Modified Extended Kalman Filter for Bearings- Only Tracking," *The 22nd Iranian Conference on Electrical Engineering (ICEE 2014)*, no. Icee, pp. 1712–1716, 2014.

[41] A. Toloei and S. Niazi, "State Estimation for Target Tracking Problems with Nonlinear Kalman Filter Algorithms," *International Journal of Computer Applications*, vol. 98, no. 17, pp. 30–36, 2014.

[42] H. Pham, S. A. Smolka, S. D. Stoller, D. Phan, and J. Yang, "A survey on unmanned aerial vehicle collision avoidance systems," no. 1, 2015.

[43] D. Bareiss, J. Van Den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 4818–4825, 2015.

[44] P.-j. Nordlund and F. Gustafsson, "Probabilistic Conflict Detection for Piecewise Straight Paths," 2008.

[45] P. J. Nordlund and F. Gustafsson, "Probabilistic noncooperative near mid-air collision avoidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1265–1276, 2011.

[46] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A Probabilistic Approach to Aircraft Conflict Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–219, 2000.

[47] R. Irvine, "A geometrical approach to conflict probability estimation," *Encounter*, pp. 1–15, 2001.

[48] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6354 LNCS, no. PART 3, pp. 92–101, 2010.

[49] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[50] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2014.

[51] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[52] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[53] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[54] R. Girshick, "Full-Text," pp. 1440–1448.

[55] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017.

[56] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[57] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," pp. 1–38, 2015.

[58] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[59] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout Improves Recurrent Neural Networks for Handwriting Recognition," in *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2014.

[60] P. Doetsch, M. Kozielski, and H. Ney, "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition," in *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2014.

[61] A. Graves, "Generating Sequences With Recurrent Neural Networks," pp. 1–43, 2013.

[62] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *Iclr*, 2014.

[63] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the Rare Word Problem in Neural Machine Translation," 2014.

[64] F. Beaufays, H. Sak, and A. Senior, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has," *Interspeech*, 2014.

[65] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based Recurrent Neural Networks," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2014.

[66] A. Dosovitskiy, J. T. Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1538–1546, 2015.

[67] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple Object Recognition with Visual Attention," pp. 1–10, 2014.

[68] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent Models of Visual Attention," *Advances in Neural Information Processing Systems*, vol. 27, pp. 1–9, 2014.

[69] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," 2015.

[70] A. Grushin, D. D. Monner, J. A. Reggia, and A. Mishra, "Robust Human Action Recognition via Long Short-Term Memory," *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, 2013.

[71] V. Veeriah, N. Zhuang, and G.-J. Qi, "Differential Recurrent Neural Networks for Action Recognition," *International Conference on Computer Vision (ICCV)*, 2015.

[72] N. Zhuang, T. D. Kieu, G.-J. Qi, and K. A. Hua, "Deep Differential Recurrent Neural Networks," 2018.

[73] S. E. Kahou, V. Michalski, R. Memisevic, C. Pal, and P. Vincent, "RATM: Recurrent Attentive Tracking Model," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 1613–1622, 2017.

[74] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online Multi-Target Tracking Using Recurrent Neural Networks," *AAAI Conference on Artificial Intelligence*, pp. 4225–4232, 2017.

[75] G. Keren and B. Schuller, "Convolutional RNN: An enhanced model for extracting features from sequential data," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016-Octob, pp. 3412–3419, 2016.

[76] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional Recurrent Neural Networks for Music Classification," pp. 1–5, 2016.

[77] T. Guha, S. Member, and R. K. Ward, "Sequential deep learning for human action recognition," *Human Behavior Understanding*, no. October 2016, pp. 29—-39, 2011.

[78] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, "Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification," 2015.

[79] G. Yue-Hei Ng, Joe and Hausknecht, Matthew and Vijayanarasimhan, Sudheendra and Vinyals, Oriol and Monga, Rajat and Toderici, "Beyond Short Snippets : Deep Networks for Video Classification," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694—-4702, 2015.

[80] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[81] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, "First Step toward Model-Free, Anonymous Object Tracking with Recurrent Neural Networks," pp. 1–13, 2015.

[82] K. Fang, "Track-RNN: Joint Detection and Tracking Using Recurrent Neural Networks," no. Nips, 2016.

[83] M. P. Ghaemmaghami, "Tracking of Humans in Video Stream Using LSTM Recurrent Neural Net- work," 2017.

[84] A. R. Kosiorek, A. Bewley, and I. Posner, "Hierarchical Attentive Recurrent Tracking," no. Nips, 2017.

[85] T. Yang and A. B. Chan, "Recurrent Filter Learning for Visual Tracking," *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, vol. 2018-Janua, pp. 2010–2019, 2018.

[86] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," 2014.

[87] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "UAV collision avoidance based on geometric approach," *2008 SICE Annual Conference*, pp. 2122–2126, 2008.