ON DATA AUGMENTATION TECHNIQUES FOR THE AUTOMATIC DETECTION OF MOSQUITO BREEDING GROUNDS USING VIDEOS

Bettina D'Avila Barros

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Sergio Lima Netto
Eduardo Antônio Barros da Silva

Rio de Janeiro
Fevereiro de 2019

ON DATA AUGMENTATION TECHNIQUES FOR THE AUTOMATIC DETECTION OF MOSQUITO BREEDING GROUNDS USING VIDEOS

Bettina D'Avila Barros

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Sergio Lima Netto, Ph.D.

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Claudio Esperança, Ph.D.

Prof. Flavio Codeço Coelho, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2019

D'Avila Barros, Bettina

On data augmentation techniques for the automatic detection of mosquito breeding grounds using videos/Bettina D'Avila Barros. – Rio de Janeiro: UFRJ/COPPE, 2019.

XII, 63 p.: il.; 29, 7cm.

Orientadores: Sergio Lima Netto

Eduardo Antônio Barros da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 58 – 63.

1. Mosquito control. 2. Computer vision. 3. Machine learning. I. Lima Netto, Sergio *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Agradeço primeiramente à minha família: tudo que sou devo a vocês. Especialmente à minha mãe, agradeço por estar sempre presente. Sua dedicação a tudo que faz serve de modelo para eu melhorar a cada dia. Agradeço a cada bom dia, conversa e suco de laranja (foram muitos). Ao meu pai, agradeço por todo apoio e incentivo. Em cada etapa da minha vida você contribuiu, seja com conselhos ou opiniões com visões diferentes. Sua maneira de ver o mundo mudou a minha para melhor.

Agradeço à minha avó paterna por servir de exemplo para mim. Sinto muito orgulho a cada aluno que te para nas ruas esbordando gratidão. Juntamente com meu avô, vocês passaram muito respeito aos filhos e netos, formando a família paterna inimaginável que tenho hoje. Obrigada a todos pelo carinho.

Agradeço também à minha família materna, em especial à minha tia Teresinha. Adoro seu jeito de ser trazendo muitas risadas e seu carinho me torna muito sortuda: eu tenho amor de duas mães. Agradeço aos meus avós maternos que infelizmente não tive a oportunidade de conhecer, mas os levo no coração por cada história contada.

Aos meus orientadores, agradeço imensamente por cada momento de atenção. Aprendi muito nesses anos e o contato com vocês foi o principal motivo de eu ter optado pelo doutorado. Espero que tenha sido tão gratificante para vocês quanto foi para mim.

A todos os amigos próximos, agradeço simplesmente por serem amigos: torceram pelo meu melhor e disponibilizaram os ouvidos quando precisei. Agradeço também pelos momentos de descontração e peço desculpas se não foram tantos assim.

A todos os funcionários e professores do laboratório, agradeço por manterem um ótimo ambiente. Agradeço também aos colegas por me acolherem tão bem. Tive a grande sorte de alguns terem virado bons amigos que quero levar para toda vida.

Agradeço à equipe do CEFET por terem iniciado este projeto e por toda cooperação ao longo de 2018, sem a qual este trabalho não seria possível. Agradeço também aos colegas do grupo de visão computacional, em especial ao Rafael Padilla, pela grande contribuição no trabalho desenvolvido.

Por fim, agradeço ao CNPq e à FAPERJ pelo apoio financeiro nesses últimos dois anos. Desejo que toda ajuda se multiplique em retorno para o desenvolvimento do país e espero ter o imenso prazer de contribuir.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ACERCA DE TÉCNICAS DE AUMENTO DE DADOS PARA A DETECÇÃO AUTOMÁTICA DE FOCOS DE MOSQUITO USANDO VÍDEOS

Bettina D'Avila Barros

Fevereiro/2019

Orientadores: Sergio Lima Netto
Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

Esta dissertação discute sobre técnicas de aumento de dados para detectar potenciais focos de mosquito usando vídeos gravados por um drone. Primeiramente, um estudo sobre doenças transmitidas por mosquitos é apresentado para propor um sistema de visão computacional capaz de automaticamente detectar objetos associados a criadouros. Uma base de dados composta por seis vídeos aéreos contendo objetos como caixas d'água, pneus e garrafas é desenvolvida, desde a etapa de planejamento até a de execução (gravação e anotação). Entretando, devido a dificuldade de obter uma base extensiva de cenários reais, técnicas de aumento artificial de dados são apresentadas. Esse trabalho contempla três métodos para inserir imagens de objetos em vídeos a fim de aumentar o número de objetos do conjunto de treino. Por fim, uma rede neural convolucional é proposta para avaliar essas técnicas, indicando que o aumento artificial de dados reduz o sobreajuste, melhorando a capacidade da rede de detectar os objetos de interesse.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ON DATA AUGMENTATION TECHNIQUES FOR THE AUTOMATIC DETECTION OF MOSQUITO BREEDING GROUNDS USING VIDEOS

Bettina D'Avila Barros

February/2019

Advisors: Sergio Lima Netto
Eduardo Antônio Barros da Silva

Department: Electrical Engineering

This work discusses data augmentation techniques for detecting mosquito breeding grounds using videos recorded by a drone. Firstly, a study regarding mosquito-related diseases is presented in order to propose a computer vision system capable of automatically detecting disease-related objects, such as water tanks, tires, and bottles. A database composed of six aerial videos containing breeding-related objects is devised, including its planning and execution (recording and annotation) stages. However, due to the difficulty of obtaining extensive records of real scenarios, artificial data augmentation techniques are presented. This work addresses three methods of inserting images of the objects into videos in order to increase the number of objects in the training set. Finally, a convolutional neural network detector is used to evaluate these techniques, indicating that artificial data augmentation reduces overfitting, improving the overall detection performance by the proposed network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Dengue, chikungunya, and zika altogether affect millions of people each year and are considered a threat for people living in tropical countries like Brazil. The main vector of these diseases is a mosquito called *Aedes aegypti*, which reproduces mostly in man made containers with accumulated clean water. Therefore, as a control method, health agents conduct searches for mosquito breeding grounds.

In order to assist these agents, this dissertation proposes a system for automatically detecting objects associated with diseases transmitted by mosquitoes. By using a drone to map an area and computer-vision techniques to detect critical objects, the system tends to accelerate and make the agents' work more effective.

1.2 Contributions

The contributions of this dissertation are five-fold:

- An investigation of the practical problem and a complete solution proposal.
- The construction of a video database containing objects associated with high mosquito breeding potential, such as tires, bottles, and water tanks.
- The construction of an image database of tires to apply artificial data augmentation techniques on the recorded video database.
- Development and application of data warping and merging methods to generate additional training video samples.
- Use of a state-of-the-art neural network object detector to perform tire classification using the constructed databases.

1.3 Dissertation organization

This dissertation is organized as follows: Chapter 2 describes the awareness situation regarding mosquito-borne diseases and discusses the current prevention and control strategies. Also, Chapter 2 shows how computer vision can be used in this context by providing a literature review and proposing a new system to automatically detect mosquito breeding-related objects.

In Chapter 3, a database containing mosquito breeding grounds, such as water tanks, tires, and bottles, is constructed by using a drone to record the videos. The current database version comprises six aerial videos three of which are annotated frame-by-frame using bounding boxes.

In Chapter 4, artificial data augmentation techniques are presented to increase the number of objects in the database. In Chapter 5, the components of a convolutional neural network are explained and the architecture of the YOLO neural network is described. This network is then used to evaluate the data augmentation techniques of Chapter 4.

Finally, Chapter 6 presents conclusions and potential future works.

Chapter 2

The fight against mosquitoes

According to the World Health Organization (WHO), mosquitoes spread diseases that affect 700 million people and cause about one million deaths every year [8]. The mosquitoes adaptability to a large range of environments and their spread velocity are worryingly fascinating. Therefore, to understand their behavior and highlight the importance of control strategies constitute the main objectives of this chapter.

Section 2.1 shows the characteristics of the mosquitoes and the burden of associated diseases. Section 2.2 presents the current control methods of the health authority of the city of Rio de Janeiro, Brazil. Finally, Section 2.3 shows how computer vision can be used to combat mosquito-borne diseases, reviewing the literature and also proposing a new system.

2.1 Characteristics and threats of the mosquitoes

The mosquito life cycle includes four stages: egg, larva, pupa, and adult. Depending on the species and the climate, the duration of each stage varies widely, but an entire cycle usually takes two weeks.

The main reason mosquitoes survived for millions of years is their ability to reproduce: a female lays up to 500 eggs spread over a distance of up to 10 km and a very small puddle on natural or manmade containers is enough to serve as breeding ground. In the words of the famous entomologist Harold Oldroyd, "It is the mosquitoes that breed in temporary water that show the real ingenuity in making the most of what they can find". Mosquitoes deposit their eggs directly on or adequately above stagnant water surfaces. Consequently, rainfalls increase the chance of a mosquito finding breeding grounds and if the water level rises where the eggs were deposited, they can rapidly hatch into larvae.

Although the climate interferes on the mosquito habits and reproduction, the studies in [9] reveal that it has rarely been the main factor of an epidemic. That source indicates that human activities, such as the rapid population and urbaniza-

tion increase, the movement of infected people, and the degradation of the health infrastructure, among others, have generally been much more significant for disease outbreaks. In this way, the investment in basic sanitation facilities, the vector control, and population awareness to eliminate natural and manmade containers across the cities are extremely important to combat mosquito-borne diseases.

The adult mosquitoes typically feed from nectar and plant juices. However, the female is responsible for reproduction and most urban species require specific proteins encountered in vertebrate blood, preferably human blood, in order to develop their eggs [10]. In this feeding process, if the person (or animal) is infected with a mosquito-borne illness, the female can contract the disease pathogen and transmit it to other people during subsequent bites through its saliva, which also has an anesthetic so the person does not feel the bite. Therefore, the use of insect repellent is important to avoid infecting other mosquitoes as much as to not contracting illnesses.

There are about 3,500 mosquito species, but only the females of about a hundred species can carry viruses that are transmitted to humans. Some species of genus *Anopheles* are vectors of malaria, responsible for half of the deaths caused by mosquitoes. The genus *Aedes*, especially the *Aedes aegypti*, is the one that affects more people each year, causing dengue, chikungunya, zika, and recently also yellow fever.

Table 2.1: Comparison of symptoms for dengue, chikungunya, zika, and yellow fever. The symbol "+" represents the presence of the symptom and its quantity shows the symptom severity. The symbol "−" appears when the symptom is highly uncommon to that disease. Adapted from [5] also including the symptoms of yellow fever from [6].

Disease	Dengue	Chikungunya	Zika	Yellow fever
Fever	++++	+++	+++	++++
Myalgia/arthralgia	+++	++++	++	++
Oedema in limbs	−	−	++	−
Maculopapular exanthema	++	++	+++	−
Head/retro-orbital pain	++	+	++	++
Conjunctivitis	−	+	+++	+
Lymphadenopathy	++	++	+	++
Hepatomegaly	−	+++	−	++
Bleeding	+	−	−	++
Jaundice	−	−	−	+++

Dengue, chikungunya, zika, and yellow fever have many common symptoms. A comparison for these diseases is presented in Table 2.1 using [5, 6] as references. The dengue has been the most common arboviruse in the world for years and its treatment was estimated to cost 2.1 billion dollars per year in the Americas [11]. Its

virus is divided into five types, which have similar symptoms, but can vary on their severity, a second contraction of the disease increasing the chance of severe complications. While dengue and yellow fever can cause death through bleeding or liver failure, respectively, people who contract chikungunya commonly feel arthralgia, a strong joint pain, for years, and the zika virus is highly associated with microcephaly in babies infected during pregnancy [12]. Currently, yellow fever is the only *Aedes*-related disease with an effective vaccine.

There are also a few isolated cases that one person is coinfected, for example by dengue and chikungunya [13], and even though mosquito coinfections are known to be very rare, [14] shows that *Aedes aegypti* triple infection and transmission (dengue, chikungunya, and zika) is possible. The chances of this resulting in a multiple disease outbreak are still estimated as insignificant, but investigation beyond current threats are important to prevent devastating epidemics.

2.2 Current prevention and control strategies

This section focus on the current mosquito control strategies of the health authority of the Rio de Janeiro city, Brazil. They follow an international protocol to combat arboviruses which includes routine actions, such as daily visits and emergency actions, when a case is reported. The department goal is to keep vector control below 1%, i.e., if more than 1 out of 100 searched houses of a location have the presence of infected mosquitoes, control actions should be prioritized and personalized in the nearby region.

One health agent visits an average of 25 properties per day. In a visitation, the agent looks for possible breeding grounds to be removed or to have insecticides applied; collects samples of found mosquitoes traces, as eggs and larvae; and orients the people on how to combat the mosquito-borne diseases with simple actions. Also, if the place has a persistent alarming breeding ground, it is marked as a strategic location and then visitations are made every 15 days.

When a case of mosquito-borne disease is reported and the conditions of an epidemic are favorable, emergency action is taken. Up to a maximum of 48 hours after the reported case, the health agents try to block the spread of the infected mosquitoes by setting a radius of about 300 m around the infected patient home and work neighborhoods, which requires an action duration of 2 to 3 hours. Visitations on the surrounding properties are organized from outside in, looking for the disease sources. If there are many reported cases in the region, fog is used as the last option.

Fogging of insecticides was widely used a decade ago in Brazil, but nowadays the specialists alert about the ravages of their continuous use: mosquitoes can develop resistance to the insecticides contained in the fog. Also, since they feed from human

blood, usually 90% of the female adults are inside the houses when they are not depositing eggs, making it difficult for the fog to reach all flying mosquitoes.

A relative recent technique is the use of the bacteria *wolbachia* as a biological instrument of *Aedes aegypti* control. In this approach, scientists infect hundreds of mosquitoes with the bacteria, which creates dengue immunology, and release them into a region in order to compete with the infected insects. Many tests were put into practice and this technique seems effective in reducing the index of infected mosquitoes [15–17].

To sum up and complement, the health department strategies are divided into five control methods:

- Mechanical: displacement of objects which can serve as breeding grounds to covered places, protected from rainfalls; sealing of water tanks, among others.
- Chemical: use of larvicide and insecticide/fog to kill the aquatic and flying stages of the mosquitoes, respectively.
- Biological: use of larvae that eat the eggs deposited on the water and use of the bacteria *wolbachia* to block the infection on mosquitoes.
- Legal: support of justice to visit abandoned or blocked property with suspicions of containing mosquito breeding grounds.
- Integrated: involvement of other departments to provide social mobilization.

In the absence of an efficient vaccine, almost all control methods involve the detection of mosquito breeding grounds. The guidelines of WHO or any other health organization always warn about the risks of letting objects with accumulated water. Table 2.3 shows, from [7], the most crucial objects that the agents of the health department of Rio look for when visiting a property. The objects of interest are divided into five groups detailed in Table 2.2 and, except for the natural deposits (group E), all items are manmade containers.

Table 2.2: Description of mosquito breeding ground groups and subgroups. Source: [7].

Group	Group description	Subgroup	Subgroup description
A	Water deposits for human consumption	A1	Connected to the grid
		A2	Deposits at ground level
B	Mobile containers	-	-
C	Fixed deposits	-	-
D	Containers capable of being removed	D1	Rolling materials
		D2	Garbage
E	Natural deposits	-	-

Table 2.3: Objects that can serve as mosquito breeding grounds. Source: [7]. Codes are detailed in Table 2.2.

Code	Objects
A1	High tanks
A2	Barrel, tub, drum, tank, well
B	Vases/jars, plates, drippings, drinking fountains
C	Tanks, gutters, slabs
D1	Tires and other rolling materials
D2	Plastic containers, bottles, cans, scraps
E	Bromeliads, bark, tree holes

It is known through a local study that it is possible to identify containers of higher mosquito incidence, and treating only them is almost as effective as treating all existing ones [18], drastically reducing the potential for epidemic development. In the case of Nova Iguaçu, a town located in the state of Rio de Janeiro, the containers listed with high potential for an emergency are, according to [19], water tanks, glass and plastic bottles, buckets, discarded tires, and external drains.

Lastly, as all monitoring and control actions, the mosquito control can benefit from the growth and development of technology, as discussed in the next section.

2.3 Computer vision meets mosquito control

2.3.1 Related works

As mosquitoes reproduce in stagnant water, water-detecting systems, which are already common in other applications, can be used to detect breeding grounds [2, 3, 20]. The authors of both [2, 3], through community-sourced geotagged images, propose systems that generate a geographical heat map, showing where their systems indicate potential breeding sites, as illustrated in Figure 2.1, generated from the web-site in [1]. This information could be used by the population and health organizations to take preventive actions.

The dataset of [3] is composed of Google images containing water puddles, flowing water, open tires, tires attached to a vehicle, among others. Figure 2.2 shows a few examples. Their method first uses SIFT (scale-invariant feature transform) [21] to extract key points and 128-vector descriptors of all images. Then, it creates a codebook of visual words by using K-means to cluster the descriptors, as illustrated in Figure 2.3. The feature vector of each image is composed of a histogram indicating the frequency of features belonging to the clusters. Lastly, an SVM (support-vector machine) algorithm is trained to indicate the probability of an image containing a mosquito breeding site, resulting in 82% of classification accuracy.

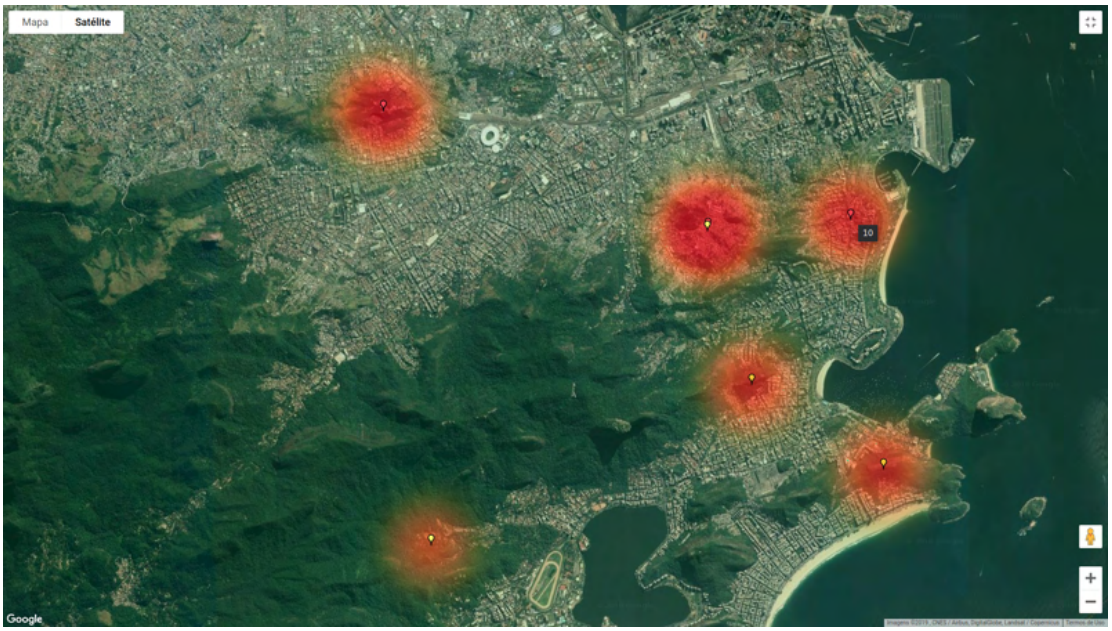


Figure 2.1: Heat map indicating regions with potential breeding sites. Source: [1, 2].

(a) Stagnant water. (b) Running water. (c) Open tires.

Figure 2.2: Images used in [3], captured from Google images.

In [2], the use of RGB and thermal images together, as illustrated in Figure 2.4, results in water puddle average classification accuracy of 90%. The feature extraction follows a similar bag-of-words approach as reference [3]. However, instead of using SIFT, key-points and descriptors are extracted using SURF (speeded-up robust features) [22] and dimensionality is halved by PCA (principal component analysis) [23]. Also, clustering is not performed; instead, the vectors are reduced into one by calculating the components average through key points, resulting in a 64-bit vector for each image.

Still in [2], an ensemble of Bayesian classifiers is used to identify if the image contains a puddle or not. Also, to improve classification results, a boosting step is added to the classification procedure using the Adaboost algorithm. These final steps improved accuracy from 82% to 90% when compared to using a single SVM classifier.

Unlike [2, 3], [20] proposes the use of a drone to inspect areas of difficult access in

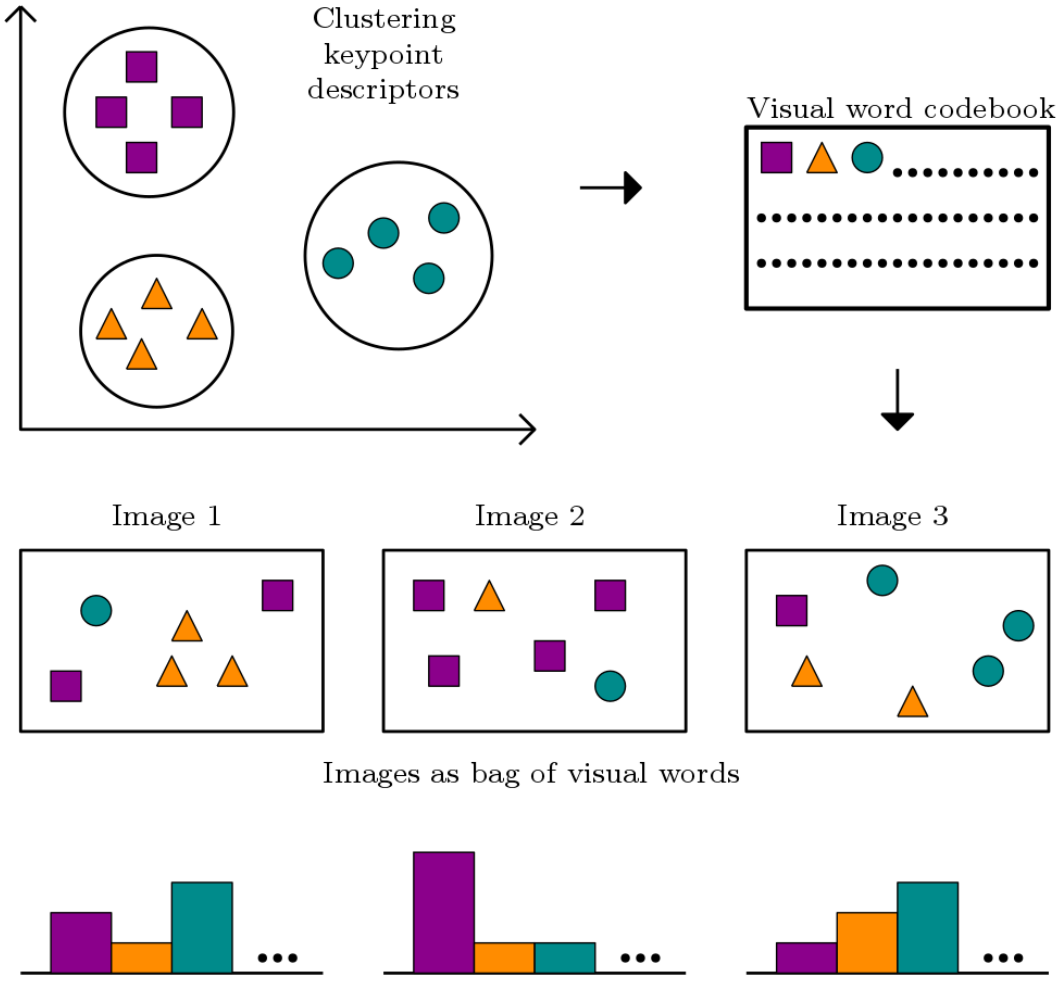
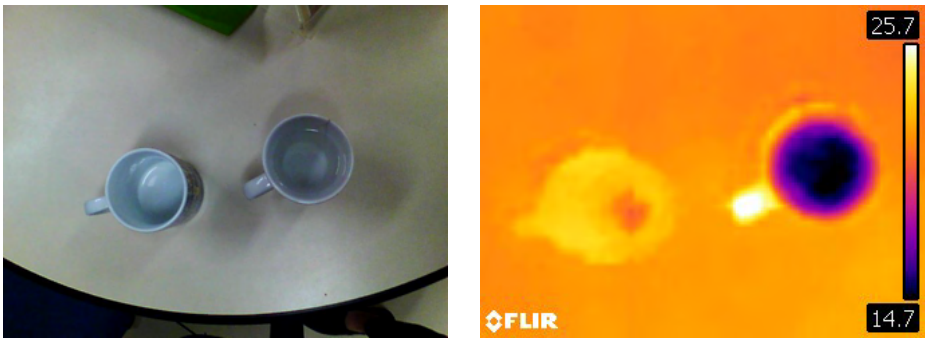


Figure 2.3: Feature extraction using bag of visual words model. The keypoint descriptors are the 128-vector extracted from images using SIFT. Based on: [3].



(a) Standard RGB. (b) Thermal image.

Figure 2.4: Illustration of the advantage of using thermal images for detecting water. In (a) both objects are reflecting and the water is as not evident as in (b).

order to automatically detect stagnant water patches in videos. As the final problem is to identify if the area contains or not water puddles, the authors are concerned about detecting at least a part of the puddles and avoiding false positives.

The detector used in [20] is a combination of (i) the score of an SVM-based method using saturation and intensity as features; (ii) the magnitude of a modified optical flow, that captures the water mirroring property. More precisely, the optical flow measures the apparent motion of objects from subsequent frames. As stagnant water reflects other objects, the puddles seem at a greater depth than the surroundings and, consequently, in a slower motion.

As one can easily note, the literature on this subject is not extensive. This indicates that the application of computer vision for detecting mosquito breeding grounds is still in the beginning and therefore there is a lot to be explored.

2.3.2 System proposal

This dissertation presents a system inspired mostly on the control actions of the Rio de Janeiro health department described in Section 2.2, but also by the related works of Subsection 2.3.1. This proposal is the result of a collaboration with the Federal Center for Technological Education of Rio de Janeiro (CEFET-RJ) and is an extension of the works presented in [24, 25].

The desired system should be capable of, by scanning a predefined area with a drone, generating a map that indicates the geographic positions of possible mosquito breeding grounds, following the object coding displayed in Table 2.3. The goal is to provide the health agents a decision support system, that can speed up their preventive actions. The main advantages for the health department are in the daily visits and emergency actions, where it can prioritize the houses to be checked.

Unlike the related works in the literature, the proposed system uses aerial images where the breeding grounds are usually very small. Although the authors of [20] also use drones, they normally fly at low heights and, as a consequence, the water puddles of the database are mostly large. Also, the proposed system focuses on detecting objects, i.e., instead of just classifying an image as containing or not a breeding ground, the object is detected with a tight bounding box. Moreover, like the one in [3], the proposed system aims at detecting all object types associated with mosquito-borne diseases and not directly water puddles. Since mosquitoes can reproduce in very small puddles that form inside objects, detecting just water may not cover a wide range of breeding grounds.

Figure 2.5 shows how the desired system works. The block diagram is divided into three parts: data preparation, system development, and system usage. With the GPS coordinates of an area as input, the first step of the data preparation is to record videos of the defined area using a drone. Then camera calibration is performed to remove distortions on the video frames and finally the bounding boxes are annotated indicating the mosquito breeding grounds. The data preparation

should be processed for many different areas to result in a diverse database. In this dissertation, the video database is fully described on Chapter 3.

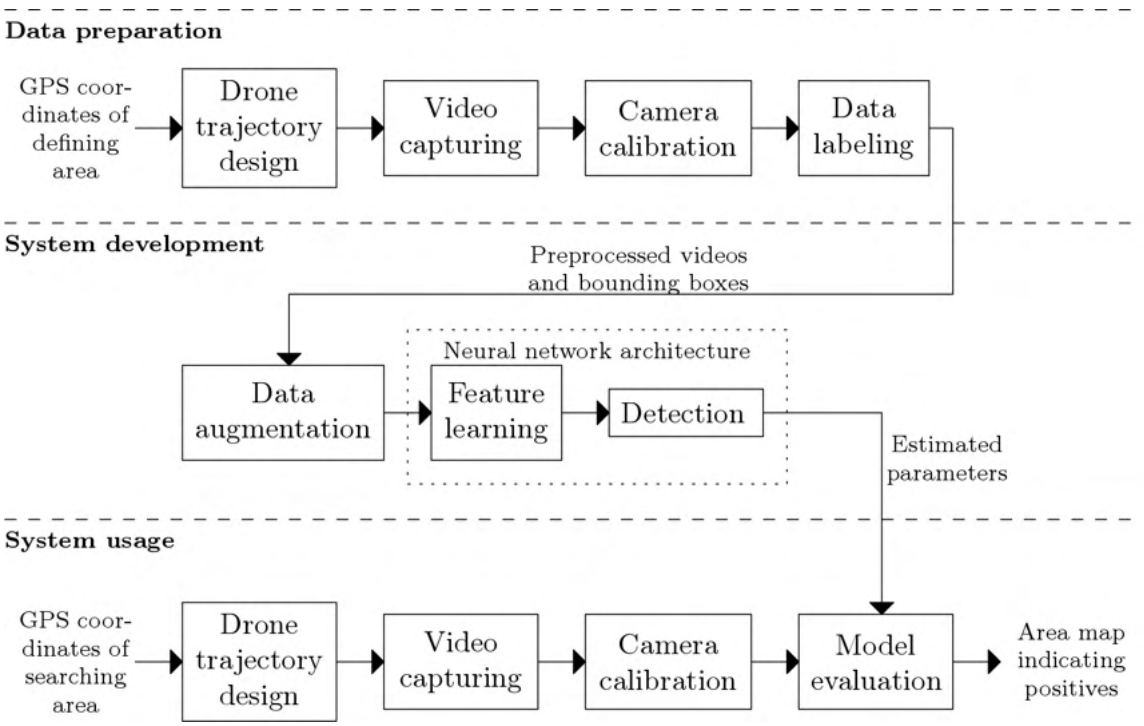


Figure 2.5: Block diagram of the desired system.

The system development stage receives the processed videos and bounding boxes from the data-preparation stage and performs data augmentation, feature learning, and object detection. Data augmentation is a technique for increasing the database in order to achieve lower generalization error. It usually involves only transformations of the original database, but data augmentation can also comprise the use of a complementary database, as done in Chapter 4. Feature learning and detection are performed by a convolutional neural network detector, described in Chapter 5.

Finally, the system usage shows how to locate the possible mosquito breeding grounds of a searching area. The same first steps of data preparation are performed: drone trajectory design, video capturing, and camera calibration. After that, using the estimated parameters of the system development, the model is evaluated and a map is constructed containing the locations of potential breeding grounds, called positives. The desired output of the system usage stage is similar to the heat map illustrated in Figure 2.1 but with higher location precision.

To sum up, this system is designed to search for breeding sites in predefined areas, resulting in maps that serve as a decision support system regarding mosquito control. The next chapters discuss more about requirements and functionalities of the system.

Chapter 3

Video database

This chapter presents a complete description of the "Mosquito Breeding Grounds" (MBG) video database devised for detecting mosquito breeding grounds in aerial videos captured by a drone. There are many databases to work with stagnant water detection, like in [2, 3, 20]. However, to the best of our knowledge, [3] is the only that works with the detection of objects associated with mosquito-borne diseases. The MBG video database contributes with more extensive types of objects and includes the bounding-boxes of these objects and the telemetry of the drone, allowing the localization of each object of interest.

The main goal of this work is to support health agents, helping them to work more efficiently in the fight against mosquito-borne diseases. As reported in Chapter 2, mosquitoes reproduce in clean stagnant water, so objects such as water tanks, abandoned tires, and bottles are commonly found by the health agents containing larvae. In this way, a proper database for the problem of supervised learning to detect and classify objects in aerial videos should have (i) control of the maximum number of parameters or at least their measurement; (ii) an expressive number of samples and representativeness of each object class; (iii) variability of the background, objects, luminosity, and height; (iv) no camera distortions; and (v) reliable annotation.

Since drones have many flight restrictions, the main difficulty to construct such a database is to obtain access to extensive types of real scenarios. Even if drone access is available to several locations, it is also necessary to know where the disease-related objects are. In this way, the solution adopted in this work is to alter manually the locations where foot access is available, inserting objects in the scene. Although the first recordings are not yet perfectly representing a real urban scenario, they are an adequate proxy. In addition, the developed methodology allows one to construct an adequate database.

To introduce the MBG video database, this chapter is organized as follows: Section 3.1 presents details about the drone employed in this work. The design and

recording process of the proposed database are described in Section 3.2. Section 3.3 presents the camera calibration process used to remove distortions, as well as the annotation process of the objects of interest that allows a supervised learning approach. Finally, a description of the MBG video database in its version 1.0 (as of January 2019) is presented in Section 3.5.

3.1 Phantom 4 Pro details

To detail the database development methodology, this section describes the drone model used to generate the "Mosquito Breeding Grounds" video database. Since almost all parameters cited in this section are common to any drone, one can adapt them to any model.

The Phantom 4 Pro (P4P) is a top-2018-market drone, which has a high-quality 4K camera, a gimble that makes pitch and roll axis (see Figure 3.1) rotational movements imperceptible on video records, and good aircraft stability and relative location, even with moderate wind and low GPS signal. From the DJI website page [26], one can find the P4P specifications as given in Table 3.1.

Table 3.1: Phantom 4 Pro specifications.

Aircraft	Weight	1388 g
	Diagonal size	350 mm
	Max speed	From 50 to 72 km/h
	Max wind speed resistance	10 m/s
	Max flight time	Approx. 30 minutes
	Vertical hover accuracy range	±0.1 m (with vision positioning) ±0.5 m (with GPS positioning)
	Horizontal hover accuracy range	±0.3 m (with vision positioning) ±1.5 m (with GPS Positioning)
Camera	Lens	FOV 84° 8.8 mm/24 mm (35 mm format equivalent) f/2.8 - f/11 Auto focus at 1 m - inf
	Image size	3:2 aspect ratio: 5472 × 3648 4:3 aspect ratio: 4864 × 3648 16:9 aspect ratio: 5472 × 3078
	Video recording modes	H.265 4K 24/25/30 p H.264 4K 24/25/30/48/50/60 p
	Max video bitrate	100 Mbps
Gimbal	Stabilization	3-axis (pitch, roll, yaw) Figure 3.1
	Controllable range	Pitch: -90° to +30°

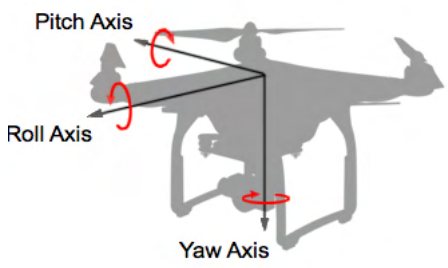


Figure 3.1: Drone pitch, roll, and yaw axis. Drone's front is pointed to the roll axis.

3.1.1 Camera lens field of view

The so-called field of view (FOV) measures the maximum diagonal of the image in the projection plan. In this drone's specifications, it is associated with the 3:2 aspect ratio (5472 × 3648). For this work, the 16:9 aspect ratio (5472 × 3078) is adopted.

Let $image\_length$, $image\_width$, where $image\_length \geq image\_width$, be the dimensions in pixels of the image in the camera projection plane. Figure 3.2 shows the geometry when projecting the camera focal length and lens FOV, FOV_{diagonal}, $FOV_{\text{left,right}}$, and $FOV_{\text{front,rear}}$.

In terms of the aspect ratio $image\_length : image\_width$, the resolution rate is

$$resolution\_rate = \frac{image\_length}{image\_width}. \tag{3.1}$$

From Figure 3.2

$$\tan\frac{FOV_{\text{left,right}}}{2} = \frac{image\_length}{2.focal\_length}, \tag{3.2}$$

$$\tan\frac{FOV_{\text{front,rear}}}{2} = \frac{image\_width}{2.focal\_length}, \tag{3.3}$$

$$\tan\frac{FOV_{\text{diagonal}}}{2} = \frac{\sqrt{image\_length^2 + image\_width^2}}{2.focal\_length}. \tag{3.4}$$

Therefore, dividing Equations (3.2) and (3.3) by (3.4) and substituting in (3.1)

$$\begin{aligned}\tan\frac{FOV_{\text{left,right}}}{2} &= \tan\frac{FOV_{\text{diagonal}}}{2}\frac{image\_length}{\sqrt{image\_length^2 + image\_width^2}} \\ &= \tan\frac{FOV_{\text{diagonal}}}{2}\frac{1}{\sqrt{1 + resolution\_rate^2}}\end{aligned} \tag{3.5}$$

and

$$\tan\frac{FOV_{\text{front,rear}}}{2} = \tan\frac{FOV_{\text{diagonal}}}{2}\frac{image\_width}{\sqrt{image\_length^2 + image\_width^2}}$$
$$= \tan\frac{FOV_{\text{diagonal}}}{2}\frac{1}{\sqrt{1 + resolution\_rate^{-2}}}. \tag{3.6}$$

Substituting maximum $FOV_{\text{diagonal}} = 84°$ and the related aspect ratio 3:2 in Equations (3.5) and (3.6)

$$\text{maximum } FOV_{\text{left,right}} \approx 73.68° \text{ and maximum } FOV_{\text{front,rear}} \approx 53.08°.$$

Considering the 4K (3.840 × 2.160) resolution that has an aspect ratio of 16:9, the entire length (left-right) of sensor area is used, but not the width (front-rear). Thus the used FOV angles are

$$FOV_{\text{left,right}} \approx 73.68° \text{ and } FOV_{\text{front,rear}} \approx 45.70°. \tag{3.7}$$

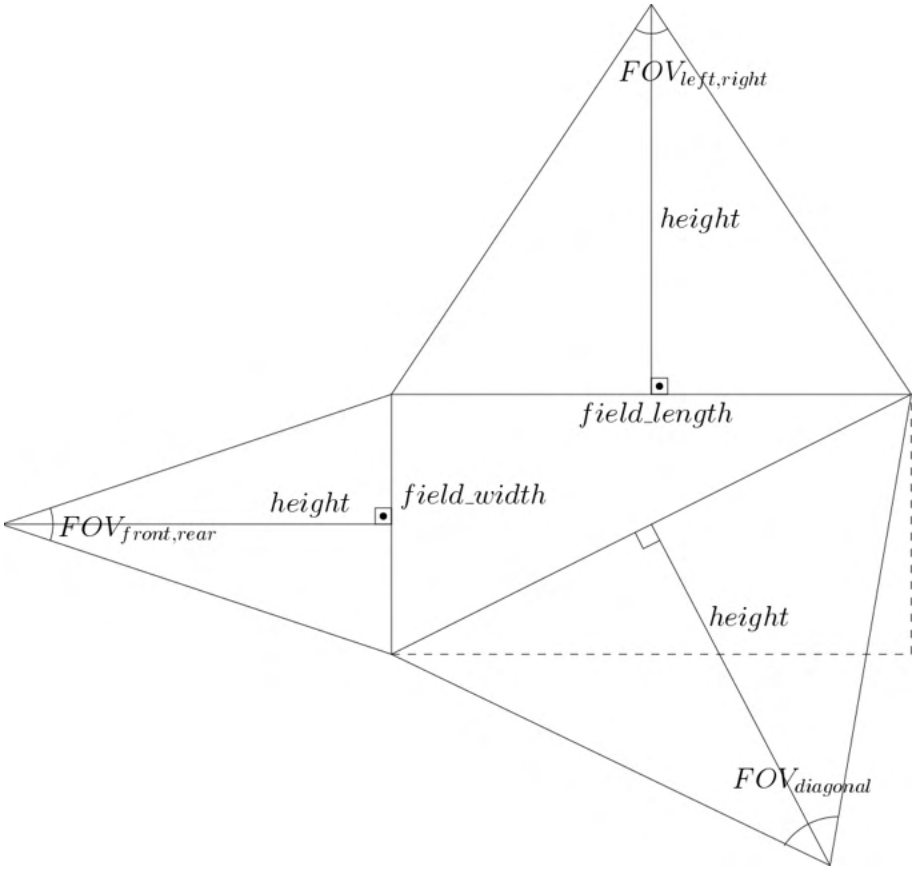


Figure 3.2: Projections of the camera center showing the relationship over the FOV angles.

These calculation results are important to understand the amplitude of the drone view in Section 3.2.

3.2 Recording setup

In order to reduce the time spent in each recording and to have more control over the drone trajectory and its associated parameters, a methodology for planning an automatic flight was designed as a mission of the Litchi app [27]. The official DJI app for the P4P drone does not provide the telemetry of the drone, neither allows one to design an automatic flight. Therefore, the Litchi app was considered, which provides almost all our needs.

First, the drone trajectory is designed as a function of several important parameters, such as the points of interest and drone height and speed. After some tests to define a range of the height and speed of the drone and include the aircraft positioning error in the model are presented.

3.2.1 Drone trajectory

The design of the drone trajectory prior to the flight has many benefits. First, it prevents wasting time before and during the flight. Without such design, each recording session takes about ten minutes of setting parameters and another ten minutes of recording, supposing everything goes right at first, what seldom happens. With a proper flight plan, however, the same area can be covered in about two minutes by reducing the overlap between successive drone laps.

When setting the flight plan manually, common mistakes are to overlap areas more than the ideal or to do not overlap them at all, especially when changing the height of each flight. In this work, the drone trajectory depends on the drone height, so one can easily calculate the trajectories for various heights and save profiles to decide later which one to use, given the conditions of the area, like trees height for example.

Moreover, using a flight plan, the drone tends to be more stable regarding speed, changes in movement directions, and flying in straight lines. Improving these conditions facilitates the last step of labeling the database.

Due to the nature of our problem, the downward vision system is used, which means the camera always points directly to the floor. In this way, the videos have the lowest projection distortion and minimum occlusion effects. Also, the drone height and speed are set as constant during a flight.

For the flight planning strategy, we consider the points of interest, like the corners of an area, the drone height and speed, and some drone parameters that only depends on the drone model. The trajectory (i) covers a rectangular area, defined as the rectangle with minimum area that surrounds the given points; (ii) maps the rectangular area in an inverted "boustrophedon" way, i.e., from bottom to top and from top to bottom in parallel lines of alternate directions; and (iii) has an overlap that

allows all objects of interest with size smaller than a maximum to be represented in its entirety in at least one pass, as shown in Figure 3.3.

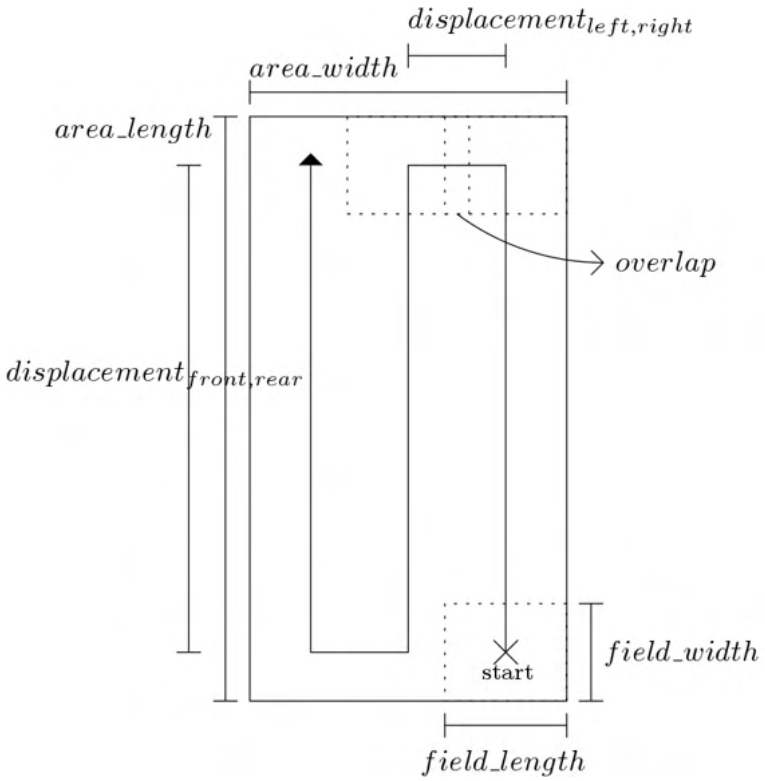


Figure 3.3: Structure of the designed drone trajectory.

Let the given n points of interest be $P_i = (lat_i, lon_i), i = 1, \ldots, n$ in GPS coordinates, where lat_i, lon_i denote the latitude and longitude of point i, respectively. The initial step is to convert these values to the Universal Transverse Mercator (UTM) [28] coordinates, $p_i = (x_i, y_i, zn_i, zl_i), i = 1, \ldots, n$, where x_i, y_i are the easting and northing coordinates and zn_i, zl_i the zone number and letter of the point i. Zone's number and letter appear because UTM divides the globe into six zones and needs these references to disambiguate the geolocation position between zones. Since UTM is a conformal projection formed with planar coordinates, the calculations can be performed using Euclidean geometry, which is not possible with the GPS coordinates.

To calculate the minimum-area rectangle, a mathematical entity called convex hull is used. The convex hull of a finite set of points $(x_i, y_i), i = 1, \ldots, n$ is defined by the intersection of all convex sets containing all points, where a convex set is a subset of the space that is closed under convex combinations. We use the algorithm described in [29] to find the convex hull. The algorithm sorts the points lexicographically in $O(n \log n)$ time and then builds the upper and lower hulls (see Figure 3.4) by using two monotone chains in $O(n)$ time. Thus, the time complexity of this algorithm is $O(n \log n)$. Since finding the convex hull is the most consuming

step, the mission flight plan algorithm has also $O(n \log n)$ complexity.

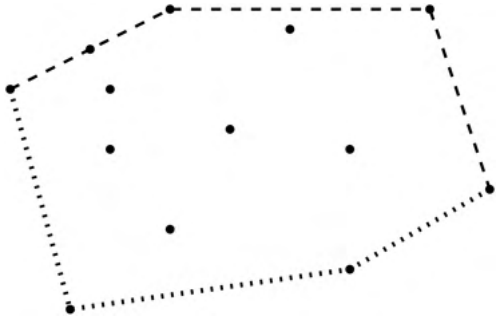


Figure 3.4: Upper hull (dashed line) and lower hull (dotted line) forming the convex hull of a set of points.

Let $A \subset \{(x_i, y_i), i = 1, \ldots, n\}$ be a sorted subset that represents the convex hull of the given points. Two consecutive points of the convex hull define an edge. So and so [30] proves that the minimum-area rectangle contains at least one of the edges of the convex hull A. So one can consider only the rectangles that have an edge coincident with the convex hull and thus the minimum-area rectangle problem becomes simple from here. One should use just find the farthest point from the line that contains the coincident edge to define the opposite edge support line. Find also the projections of all points into both of these lines, so the farthest combining projections in each line are the rectangle's vertices.

With a defined rectangle one can easily design the trajectory. Let $field\_length$, $field\_width$, where $field\_length \geq field\_width$, be the measures of the ground that is covered by the drone. Therefore, the starting point is set as a point inside the rectangle at a distance, parallel to the sides of the rectangle, of $\left(\frac{field\_length}{2}, \frac{field\_length}{2}\right)$ from the rectangle vertex closest to the home point (see Figure 3.3).

Let $area\_length, area\_width$, where $area\_length \geq area\_width$, be the rectangle side measures. Thus

$$displacement_{\text{front,rear}} = area\_length - field\_width$$

and

$$displacement_{\text{left,right}} = \frac{area\_width - field\_length}{passes}$$

with

$$passes = \left\lceil \frac{area\_width - field\_length}{field\_length - overlap} \right\rceil,$$

where $\lceil x \rceil$ denotes the minimum integer greater than or equal to x.

Defining the first way-point of the drone trajectory as the starting point, the

remaining way-points are calculated by a number of iterations consisting of alternate vector additions. One adds a vector of module $displacement_{\text{front,rear}}$ and front-rear direction to the last way-point. The other adds a vector of module $displacement_{\text{left,right}}$ and left-right direction. Both directions are directly calculated using the rectangle vertices.

3.2.2 Drone height and speed

In the proposed setup, drone height and speed are fixed for each flight. With a constant height, one does not have to deal with resolution or depth changes inside each video, and both constant height and speed accelerate the annotating process.

By reducing the drone height, the resolution of objects on the floor increases. However, by using a lower height one would spend more time covering the same area and larger objects such as a pool or water tank might not fit in a single frame. In general, there is a different minimum height restriction for each area due to physical limitations such as trees, cables, and light poles, besides the flight restrictions by ANAC Brazilian flight regulation agency [31].

We performed a test to analyze the object resolution as a function of drone height and the result can be found in Figure 3.5. The goal was to visualize which would be the higher altitude in which it is still possible to identify the smallest object of interest, a bottle for instance. Looking through a 4K resolution monitor, the bottle could be recognized well from up to 10m of height. Once it is the smallest object in our database, the others could also be identified from this height. Thus, when possible, the drone height is set to 10 m.

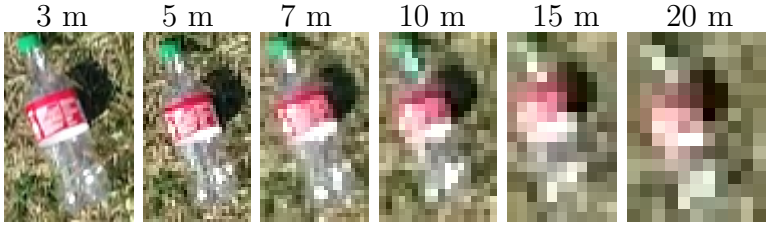


Figure 3.5: Regions of interest showing the same plastic bottle of 600ml recorded at different heights, indicated in the top of each image.

To prevent an object to appear only in a few frames, which may hinder the detection, a range for the drone speed was designed based on the number of frames that contain the largest object of interest. Figure 3.6 shows the problem's geometry.

One should recall that the drone camera is pointing directly to the floor and that the drone does not change its orientation, only its direction. Let $field\_length, field\_width$, where $field\_length \geq field\_width$, be the measures of

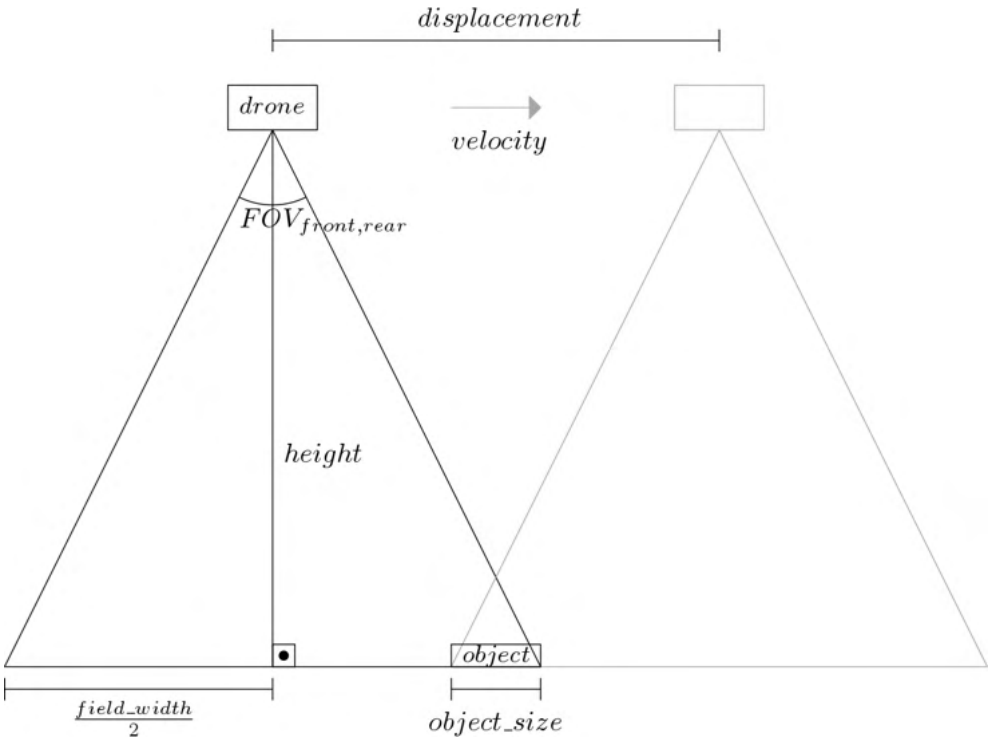


Figure 3.6: Geometry of a drone displacement at a given height and speed that covers a whole static object. Drone is moving forward, so the calculations consider the field width and front-rear FOV angle.

the covered ground. Consider also the constant speed and height of the drone, an object with size *object_size*, the front-rear field of view $FOV_{\text{front,rear}}$ as calculated in Equation (3.7), and the frame rate. Therefore the displacement of the drone is

$$\begin{aligned} displacement &= field\_width - object\_size \\ &= 2.height.tan\frac{FOV_{\text{front,rear}}}{2} - object\_size, \end{aligned} \tag{3.8}$$

and the quantity of frames containing the whole object becomes

$$\begin{aligned} frames &= frame\_rate.\frac{displacement}{speed} \\ &= \frac{frame\_rate}{speed}\left(2.height.tan\frac{FOV_{\text{front,rear}}}{2} - object\_size\right) \end{aligned} \tag{3.9}$$

When using frame rate equal to 30 fps and object size of 1 m, we have the results shown in Table 3.2, from which we conclude that from a height of 10 m it is sufficient to use the maximum speed tested. Speeds higher than 20 km/h are not recommended and the speed of 15 km/h is suggested as higher values make the annotation process hard.

Table 3.2: Number of frames that contain a whole object of size 1 m as a combination of the drone height and speed, using 30 fps frame rate.

speed \ height	3 m	5 m	7 m	10 m	12 m	15 m	20 m
5 km/h	33	69	105	160	196	251	342
6 km/h	27	57	88	133	164	209	285
7 km/h	23	49	75	114	140	179	244
8 km/h	20	43	66	100	123	157	214
9 km/h	18	38	58	89	109	139	190
10 km/h	16	34	52	80	98	125	171
11 km/h	15	31	48	72	89	114	155
12 km/h	13	28	44	66	82	104	142
13 km/h	12	26	40	61	75	96	131
14 km/h	11	24	37	57	70	89	122
15 km/h	11	23	35	53	65	83	114
16 km/h	10	21	33	50	61	78	107
17 km/h	9	20	31	47	57	73	100
18 km/h	9	19	29	44	54	69	95
19 km/h	8	18	27	42	51	66	90
20 km/h	8	17	26	40	49	62	85

3.2.3 Aircraft positioning error

As specified in Table 3.1, the P4P has four types of measures regarding the aircraft hover accuracy: vertical or horizontal and vision or GPS positioning.

Horizontal GPS positioning accuracy is considered by expanding the rectangular area in order to be sure it covers all the given points of interest. The expansion is made like in Figure 3.7. GPS error is considered only in the first vertex, as the drone has a relative positioning system besides the GPS system.

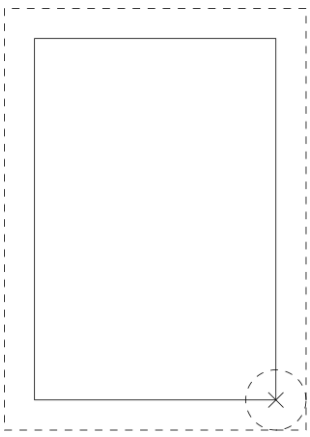


Figure 3.7: The rectangle with a continuous line represents the area to be covered by the drone. The dashed circle shows the uncertainty of the horizontal GPS positioning. The dashed rectangle is the extended rectangle to be used in the trajectory, ensuring the drone covers all the area.

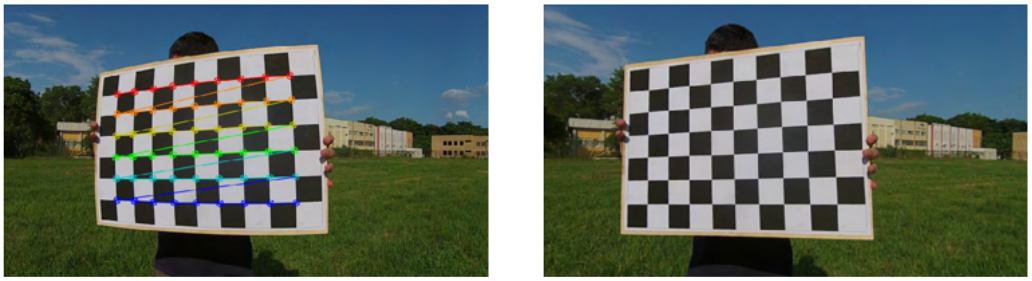
We also consider the horizontal and vertical vision positioning accuracy when calculating the trajectory overlap. With these positioning errors, the overlap becomes

$$overlap = object\_size + vision\_horizontal\_accuracy + 2.vision\_vertical\_accuracy.tan\left(\frac{FOV_{\text{left,right}}}{2}\right).$$

With these considerations on the drone positioning error, the assumptions of covering the entire area and of the objects not being cut in at least one drone lap are guaranteed.

3.3 Camera calibration and data labeling

Although the P4P camera does not seem to have relevant radial distortions, it was calibrated so as the lens distortion could be removed [25]. A common and accurate method comes from Zhang [32], where several images of a chessboard in different positions and angles are used to estimate the intrinsic camera parameters and then undistort the images, as illustrated in Figure 3.8 with a drone camera different from the one used.



(a) P2V original frame.

(b) P2V undistorted frame.

Figure 3.8: Example of undistorted frame by using Zhang's method to calibrate the Phantom 2 Vision drone camera. Radial distortion can be visualized.

The bounding-box frame-by-frame annotation of all recorded videos has been done by the Bodetronic team at CEFET/RJ using the Zframer system, as illustrated in Figure 3.9. The Zframer software uses a linear interpolation over the interval of 10 frames, so one does not need to select bounding-boxes in every video frame, that is very time-consuming.

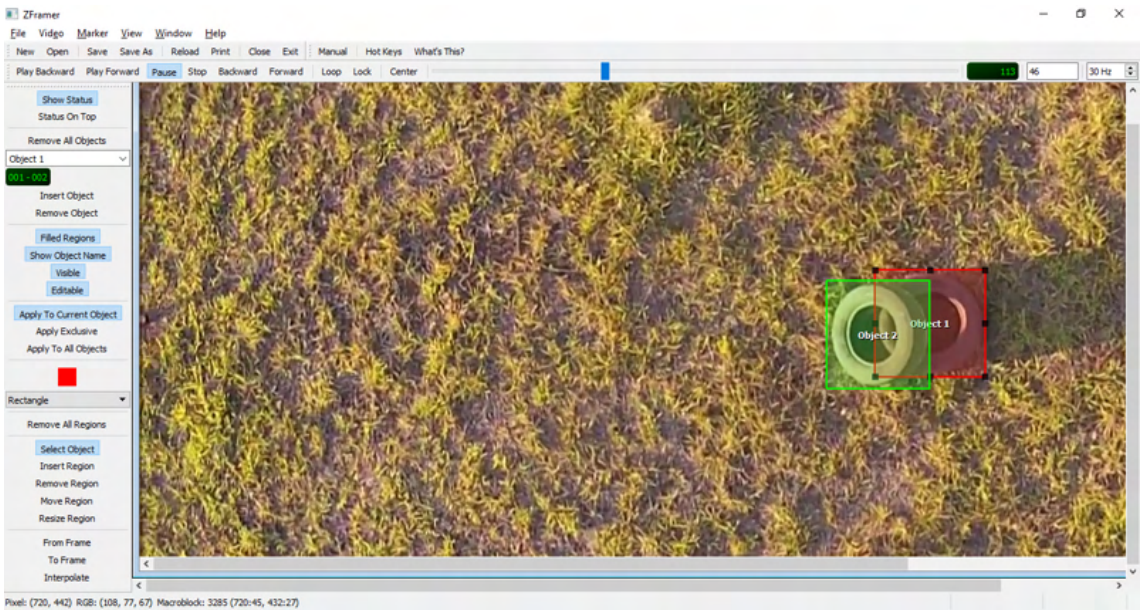


Figure 3.9: A print screen of tire annotation examples using the Zframer software.

3.4 Recording script

To facilitate the recording process, a recording script is set up as follows:

1. Select the points of interest on Google Maps or Litchi Mission Hub [33].
2. Define drone height (desired 10 m) and speed (suggested 15 km/h).
3. Set the flight plan remotely as described in this section.
4. Import the output of item 3 to Litchi Mission Hub and save it after verifying general parameters:
    - Finish Action: RTH (Return To Home).
    - Path Mode: Straight Lines.
    - Default Gimbal Pitch Mode: Disabled.
5. Arrange the objects in the area, if necessary.
6. Open the Litchi app and verify other parameters such as:
    - Resolution: 4K (3.840 x 2.160).
    - Focus at infinity (do it manually by touching the screen on the horizon), with the care that an unfocused video, that ruins the objects details, is avoided.
    - White balance as sunny or cloudy, depending on the weather.

- Set ISO, the sensitivity of camera sensors, to manual and change it to get a reasonable lighting.

7. Record video for camera calibration.

8. Open the saved mission on the Litchi app.

9. Press play.

3.5 "Mosquito Breeding Grounds" video database

The MBG video database is composed of videos captured by a drone as described in previous chapters. Each video has an annotation file, containing the bounding box of the objects of interest for each frame, and a telemetry file, that stores the values of several drone parameters during the recording. The database comprises an approximate total of 27 minutes of video divided in 12 parts. Figure 3.10 illustrates the different sites and Table 3.3 describes the attributes of the respective recording videos.

Table 3.3: Attributes of the recordings made at each site. BLI3, CCMN, and FAU had not annotated yet when this dissertation was written. Column object layouts has the number of arrangements of the objects manually inserted in the scene.

site name	**ground type**	**drone height**	**marked objects**	**video duration**	**covered area**	**object layouts**
Gremio	low grass	10 m	18	212 s	4.7 $\times 10^3$ m^2	3
BLI1	high grass	10 m	14	97 s	2.4 $\times 10^3$ m^2	2
BLI2	street and low grass	20 m	16	23 s	1.2 $\times 10^3$ m^2	2
BLI3	building	40 m	0	221 s	39.9 $\times 10^3$ m^2	1
CCMN	wasteland	15 m	0	75 s	4.0 $\times 10^3$ m^2	2
FAU	wasteland	20 m	0	187 s	13.4 $\times 10^3$ m^2	2
Total	**-**	**-**	**48**	**27 min**	**65.6 $\times$ 10^3 m^2**	**12**

The proposed database has more than six types of breeding-related objects, such as tires, bottles, water tanks, and other containers that can accumulate clean water. Also, objects not associated with mosquito-borne diseases, called confusion objects, were used to render the detection task more real. Figures 3.11 and 3.12 illustrate the variability of these objects in a recorded video at "Gremio" (first row of Table 3.3).

Lastly, this is the first version of the MBG video database, it should and can be continuously improved.

(a) (b)

(c) (d)

(e) (f)

Figure 3.10: Frames to illustrate each recording site: (a) Gremio, (b) BLI1, (c) BLI2, (d) BLI3, (e) CCMN, and (f) FAU.

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.11: Examples of objects of interest.

(a) (b) (c) (d) (e)

Figure 3.12: Examples of objects that should not be detected.

Chapter 4

Artificial data augmentation

State-of-the-art image classification and object detection systems usually require large training sets. Despite all efforts to construct an extensive and diverse database, the process of capturing data and manually labeling instances is exhaustive. Hence, to increase the performance of such systems, artificial data augmentation is being widely used [34–39].

A successful application of data augmentation for image classification is on digit recognition by using affine transformations on the input images, such as rotation, scaling and horizontally flipping, and even elastic deformations to add variability in the training data [34, 40]. These kinds of transformations are known as traditional data augmentation and are still widely used combining with other techniques.

Recently, many are embracing neural networks for data augmentation. Works like [35] synthesize a new image using Generative Adversarial Networks (GANs) [41] to transfer style between two input images. Other works [39, 42] study Generative Latent Optimization (GLO) [43] to create synthetic objects. These new samples can be used to increase the number of instances in the training set of an object detector. In [39], the authors exploit stingrays detection on aerial videos captured by a drone and a generative network based on GLO is used for creating images similar to stingrays under water, enhancing detection.

Reference [36] presents a data augmentation technique that cuts and pastes object images on background scenes using traditional transformations, without concerns about realism. The authors measure the performance of their approach in a database for detecting specific small objects in photos of house rooms and office environments and the usage of the augmented database improves performance by 21%, going beyond state-of-the-art for instance detection. Following this simple idea, [37] adds scene context information and extends the framework to databases with different environments as background scenes.

On the one hand, using neural networks to perform data augmentation usually yields non-realistic but impressive visually-appealing images with minimum effort.

On the other hand, a simple approach that just cuts and pastes images produces remarkable results, where the object is realistic but the way it is integrated in the scene is not. Although both techniques proved to be promising, for applying data augmentation on the problem of detecting disease-related objects, this chapter is inspired by [36], that introduces the "cut-and-paste" framework.

This chapter is organized in four sections as follows. Section 4.1 presents a tire image database which is further used to apply data augmentation in the MBG video database. Section 4.2 and Section 4.3 propose methods for transforming and pasting object images on background scenes. Then, the framework is extended in Section 4.4 to insert an object image into a sequence of video frames, considering visual consistency. Finally, Section 4.5 specifies the augmented MBG database by applying the proposed framework.

4.1 Tire image database

In this section, an image database composed of photos of different open tires is presented and two steps are treated: data capture and image segmentation. A Moto G5S Plus camera, with 13-MP resolution, was used to capture indoor photos of six tires. The photo set has many tire arrangements regarding orientation and also combining multiple tires. The proposed database comprises 62 images and some examples are shown in Figure 4.1.

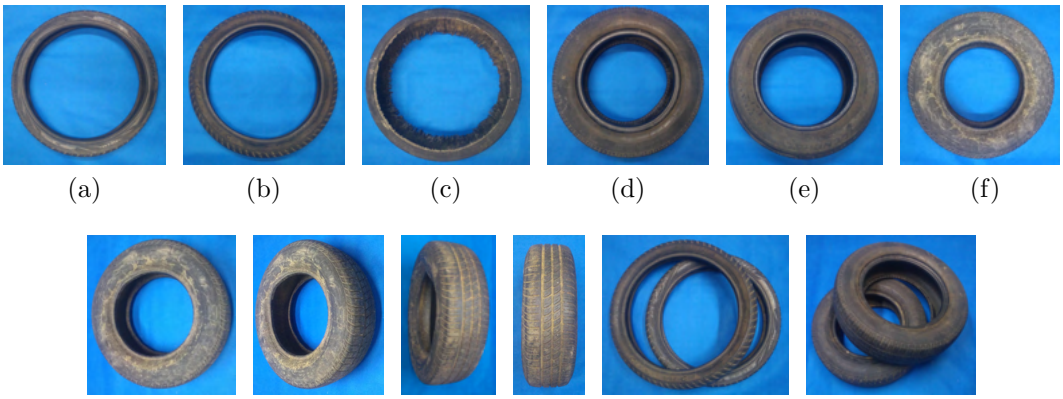
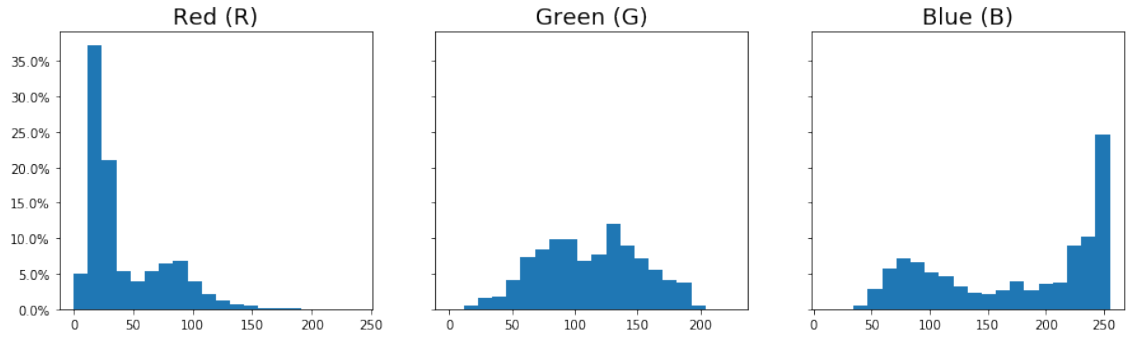


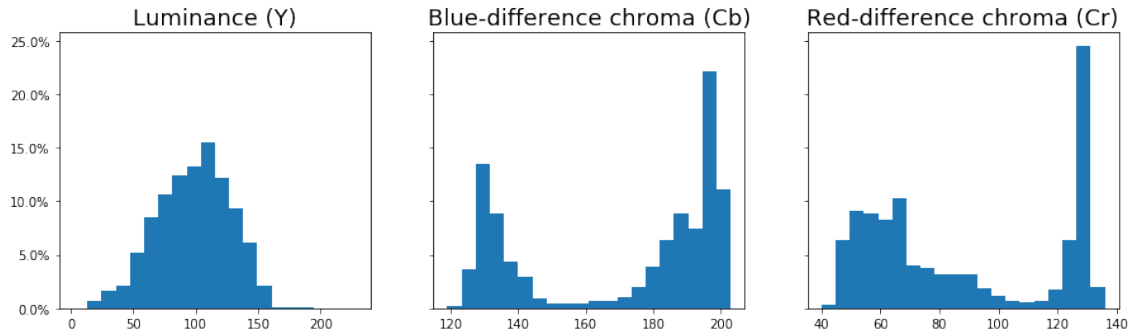
Figure 4.1: Subfigures (a)-(f) show the six different tires, (f)-(j) the orientations chosen for a tire, and (k)-(l) examples of multiple tire combinations.

A blue canvas was used as a background for the photos in such a manner that a simple image segmentation can be applied to remove the background and generate a sharp segmentation mask. Figure 4.2 shows the RGB and YCbCr color-space histograms of Subfigure 4.1(l). The RGB color space represents the three components red, green, and blue, while YCbCr represents the luminance, blue different chroma,

and red different chroma. All these six components are integers that vary from 0 to 255.



(a) RGB histograms.



(b) YCbCr histograms.

Figure 4.2: Percentage histograms of each component of the RGB and YCbCr color spaces for image 4.1l.

The histograms of the YCbCr color space have clear separations in the last two components, much more enhanced than in the histograms of RGB color space. Therefore, the segmentation is performed on the YCbCr color space resulting in Figure 4.3. The same procedure is adopted to all photos with slight differences in the threshold values.

(a) Original image. (b) Segmented image.

Figure 4.3: Segmentation of an image by using thresholds on the YCbCr color space.

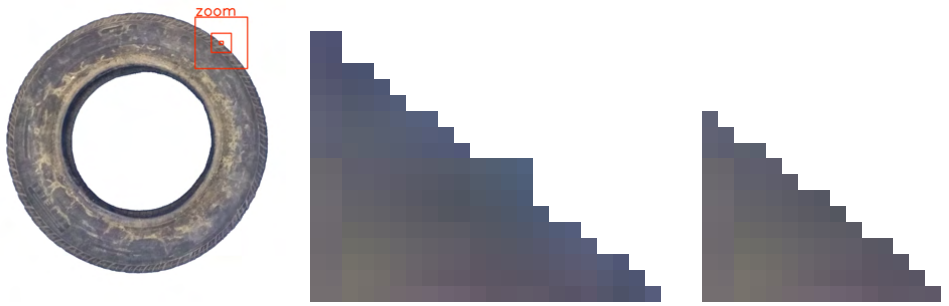
A white canvas was also tested with the segmentation performed on the luminance component of the YCbCr color space. However, since tires have a majority of

gray tones and shadows on the white canvas are also gray, there are cases in which a simple threshold is not capable of separating the tire from the white background, as illustrated in Figure 4.4.

(a) Original image. (b) Segmented image. (c) Zoom on the segmented image.

Figure 4.4: Example of an image segmentation that uses a threshold on the luminance.

Although using a blue canvas makes the segmentation simpler, it yields in a complication at the borders of the tires: the pixels of the object border are bluish. To mitigate this problem, a morphological erosion operation [44] can be applied to the image masks. For this database, a 3×3 circular structuring element is used in five erosion iterations and results are illustrated in Figure 4.5.



(a) Segmented image. (b) Zoom. (c) Zoom after erosion.

Figure 4.5: Example of applying erosion on the mask of an segmented image.

When the mask of a tire image is defined, the bounding box is automatically calculated. However, for multiple-tire images, the bounding boxes are manually annotated.

4.2 Image warping

Image warping is defined in this work as manipulations over an image to produce new instances. Four actions are executed: introduce an arbitrary luminance gain that belongs to a predefined interval, horizontally flip the image with 50% probability,

rotate it with an arbitrary angle, and resize it. Figures 4.6 and 4.7 show examples of rotating and flipping tires.

(a) 0° (b) 45° (c) 90° (d) 135° (e) 180°

Figure 4.6: Examples of rotating Subfigure (a) with different angles.

(a) Original. (b) Flipped.

Figure 4.7: Example of horizontal flip. Note: vertical flip is not required as it is a combination of horizontal flip and 180°rotation.

The interval of luminance gain is experimentally defined. Figure 4.8 shows shifted luminance averages over a segmented tire image and one can see how the synthetic object becomes unrealistic when shifting too much the luminance. Hence, since the photos were all taken at the same lighting environment, the shifted luminance average remains within $[50, 150]$.

(a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250

Figure 4.8: Example of a tire image that originally has luminance average of approximately 116. The Subfigures represent the image with shifted luminance average. The luminance is truncated at 0 and 255. Blue and red chroma (Cb and Cr) are not altered.

Before merging, the image is resized based on a range that considers the drone height and the possible measures of a tire. The tire sizes used in Section 4.1 vary approximately from 0.6 m to 0.7 m. Therefore, to avoid deviating too much from

reality and knowing the drone height, the image is resized to appear within the range [0.5, 0.8] in meters.

When rotating and resizing the image, the mask is returned as a grayscale image and thus a threshold of 127 is set to maintain the values either 0 or 255, i.e., pure black and white. Following these procedures to add variability on the 62 tire images, merging methods can be applied to perform data augmentation on the video frames of the MBG database.

4.3 Merging methods

Given an object image with its mask and a video frame with an indicated position on the background scene, a merging method is defined in this work as a technique to insert the image on the background. Three different merging methods are presented and subjectively compared in this chapter and also evaluated within a detector in the next chapter.

4.3.1 Paste

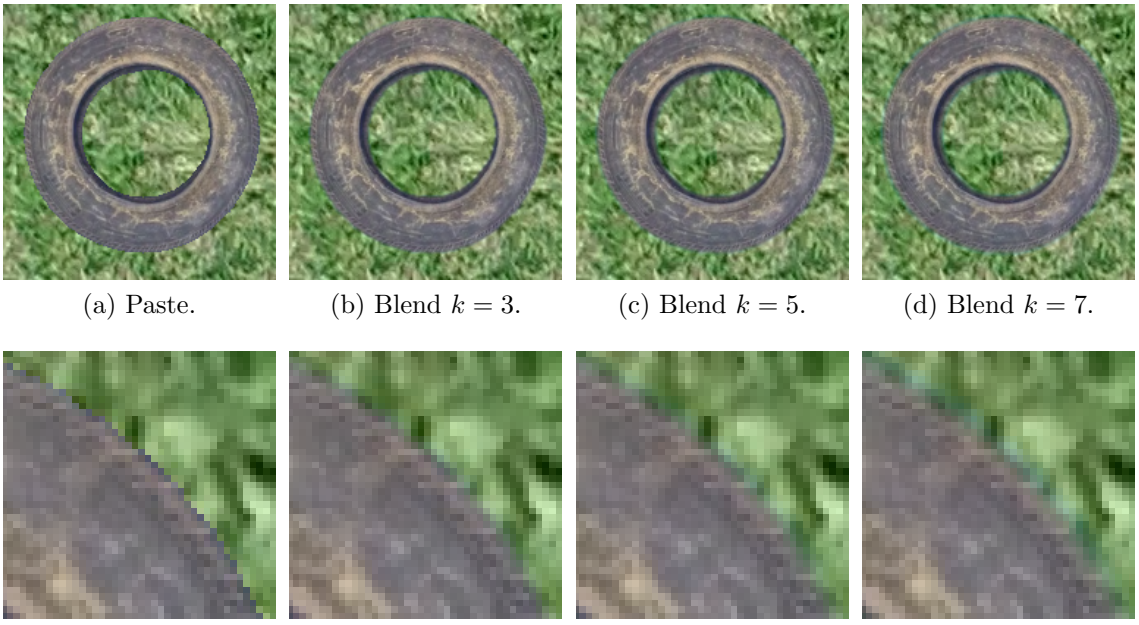
The first merging method only pastes the image on the background, with no other consideration. Subfigures 4.9(a) and 4.9(e) contain an example of this method. Instead of increasing the performance of the detector, this method introduces boundary artifacts that tend to hinder the detection results. Therefore, the merging methods presented in the sequel carry out procedures to make the transition between object and background smoother when pasting the object on the background.

4.3.2 Blend

The second merging method, similarly to the one in [36], blends the border of the object with the background when pasting the image. The blending is done by using the object mask after applying to it a Gaussian filter to smooth the object border. Figure 4.9 shows the result of blending using different kernel sizes. The size $k = 3$ is chosen for looking more natural.

4.3.3 Blur and blend

Although the second method explores blending to mitigate the boundary artifacts introduced by the first merging method, in many cases, the objects still visually stand out from background, looking too artificial. This can facilitate the detection and then bias the learning process. The detector features should not emphasize details of a specific object or the traces of the merging process. Instead, it should



(a) Paste. (b) Blend $k = 3$. (c) Blend $k = 5$. (d) Blend $k = 7$.

(e) Paste. (f) Blend $k = 3$. (g) Blend $k = 5$. (h) Blend $k = 7$.

Figure 4.9: Blend merging method using different Gaussian kernels.

extract characteristics inherent to the object class, like form and color in the case of the tire class. Therefore, the last method studies blurring to obtain a merged image with blurring closer to that of the background.

A two-dimensional Gaussian filter with the same variance in both directions is applied to blur the objects and a Laplacian of Gaussian (LoG) filter is used in order to evaluate the blur. The LoG has attractive scaling properties and is commonly used to detect borders [45] due to its band-pass characteristic of selecting high frequencies while reducing noise. Meanwhile, the variance of the LoG is applied in many works [46, 47] to measure the blur level.

The Laplacian of Gaussian

The continuous two-dimensional Gaussian function is given by

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{4.1}$$

where σ^2 is the variance. Consider the two-dimensional Laplacian operator L, defined as the second derivative of a given function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$$\begin{aligned} L(x, y) &= \nabla^2 f(x, y) \\ &= \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}. \end{aligned} \tag{4.2}$$

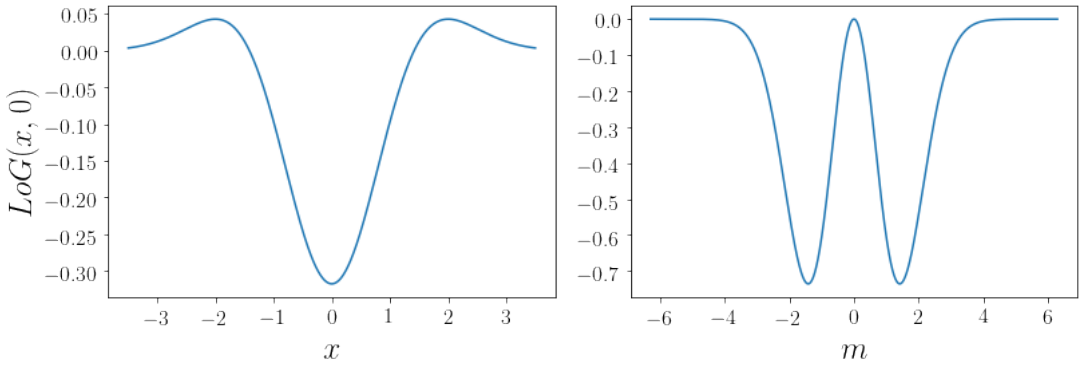
Then, using Equation 4.1,

$$\begin{aligned} LoG(x,y) &= \nabla^2 G(x,y) \\ &= -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right). \end{aligned} \tag{4.3}$$

The Fourier transform of the LoG is calculated as

$$\mathcal{F}(LoG)(m,n) = -(m^2+n^2)\exp\left(-\frac{\sigma^2(m^2+n^2)}{2}\right) \tag{4.4}$$

where m, n are the spatial frequencies and an illustration is given in Figure 4.10.



(a) Spatial response at $y = 0$.

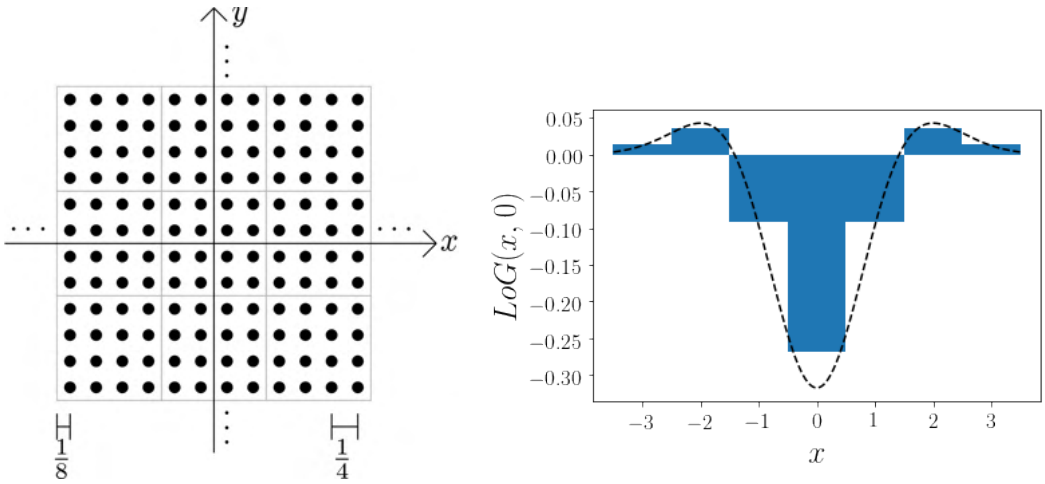
(b) Frequency response at $n = 0$.

Figure 4.10: Cross-section of the bidimensional LoG function and its Fourier transform using $\sigma = 1$.

One can note that, even though the Gaussian is a separable function, the LoG is non-separable, i.e., it can not be written as a product of two unidimensional functions. Hence, calculations to design and apply a LoG filter must be done on the two-dimensional space.

Discretization

The first step is to discretize the functions. Samples are symmetrically and uniformly selected with a $\frac{1}{n}$ step and a half-sample offset. Considering a squared grid of 1 unit squared and centered at zero, the functions are evaluated in each sample and each grid value is set as the mean of the n^2 inside samples values. The parameter n is chosen according to the precision of the estimation one requires and for this work $n = 100$. Figure 4.11 illustrates the sampling process. Also, the standard deviation σ has to be greater or equal to 0.5, otherwise, it has almost or no filtering effect.



(a) Sample grid when using $n = 4$.

(b) Cross-section at $y = 0$ of a discrete LoG with $\sigma = 1$ and $n = 100$.

Figure 4.11: Scheme of the discretization process in Subfigure (a) and an example in Subfigure (b).

Truncation

For the one dimensional Gaussian function, it is well-known that truncating at one, two, and three standard deviations cover approximately 68.3%, 95.5%, and 99.7% of the values. Thus, for the two-dimensional Gaussian function, if truncating in both dimensions separately, the coverage becomes approximately 46.6%, 91.2%, and 99.4%, the quadratic of the percentages on the one-dimensional domain. Then, the kernel size is defined as

$$k_{Gaussian}(\sigma) = 2 \; . \; \lceil 3\sigma - 0.5 \rceil + 1, \tag{4.5}$$

ensuring truncation greater or equal to 3σ and odd kernel size of at least 3.

For the LoG function, since it is not a probability function, it is more reasonable to look at the energy error. From [45], three and four standard deviations contain 86.41% and 99.27% of the energy, respectively. Then, in this work, the LoG kernel size is set as

$$k_{Laplacian}(\sigma) = 2 \; . \; \lceil 4\sigma - 0.5 \rceil + 1. \tag{4.6}$$

In this way, the truncation of the LoG is made after 4σ. The frequency responses of the generated kernels are analyzed in Figure 4.12 to guarantee adequate estimations.

Consistency analysis

Discretizing and truncating the functions can end up modifying the filters regarding image gain. The Gaussian filters must sum one to avoid introducing gain on the

image brightness, while the LoG filters must sum zero to prevent changes in the signal average (the DC level). Hence, for the Gaussian filter, all coefficients are positive and then a simple scaling is applied. To the LoG filter, a subtraction of the filter average value is applied to all coefficients in order to correct the sum.

These operations, including the discretization, may disturb the frequency response, so a verification is made in Figure 4.12, showing that, for the cases applied in this work, these procedures are sufficient to generate adequate filters.

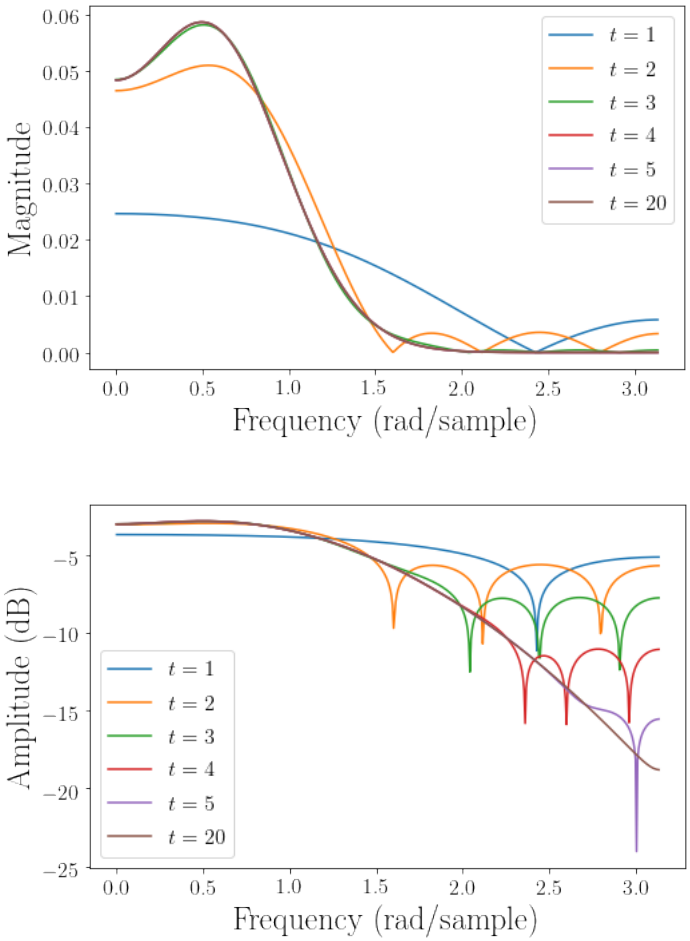


Figure 4.12: Magnitude of the frequency response after discretizing and truncating the Laplacian of Gaussian function with $\sigma = 2$. The variable t represents how many standard deviations are used for truncation.

LoG filter

The LoG filter is used to capture the blurring of the image scene. Therefore, to maintain consistency, the standard deviation value should be scaled based on the drone height.

For $\sigma = 0.5$, the LoG filter is given by

$$LoG_{\sigma=0.5} = \begin{bmatrix} 0. & 0.004 & 0.013 & 0.004 & 0. \\ 0.004 & 0.109 & 0.121 & 0.109 & 0.004 \\ 0.013 & 0.121 & -1. & 0.121 & 0.013 \\ 0.004 & 0.109 & 0.121 & 0.109 & 0.004 \\ 0. & 0.004 & 0.013 & 0.004 & 0. \end{bmatrix},$$

so the filter basically compares how the central region is different from its neighborhood.

A tire with 60 cm of diameter has the following approximate measures when captured from different heights:

- 10 m: 152 pixels
- 15 m: 101 pixels
- 20 m: 76 pixels
- 40 m: 38 pixels

In this way, the standard deviation values are set as

- 10 m: $\sigma = 2.0$
- 15 m: $\sigma = 1.5$
- 20 m: $\sigma = 1.0$
- 40 m: $\sigma = 0.5$

Method outline and comparison

To evaluate the blurring, a blur measure is defined by the variance of the image convolved with the LoG filter. Then, the greater the blur measure, the lower the blurring.

Figure 4.13 shows how the method works. First, the LoG filter is defined based on the drone height and the blur measure of the background τ^2_{ref} is calculated and stored as a reference value. Then, iterations for blurring the object are performed to obtain a merged image with blurring closer to that of the background.

The variance of the Gaussian filter used to blur the object is initialized as $\sigma_0^2 = 0.5$ and updated by $\Delta\sigma = 0.1$ in each iteration. This process stops when the blur measure of the blended image τ_i^2 is smaller than the reference value τ^2_{ref}, resulting in a merged image with equalized blur measure.

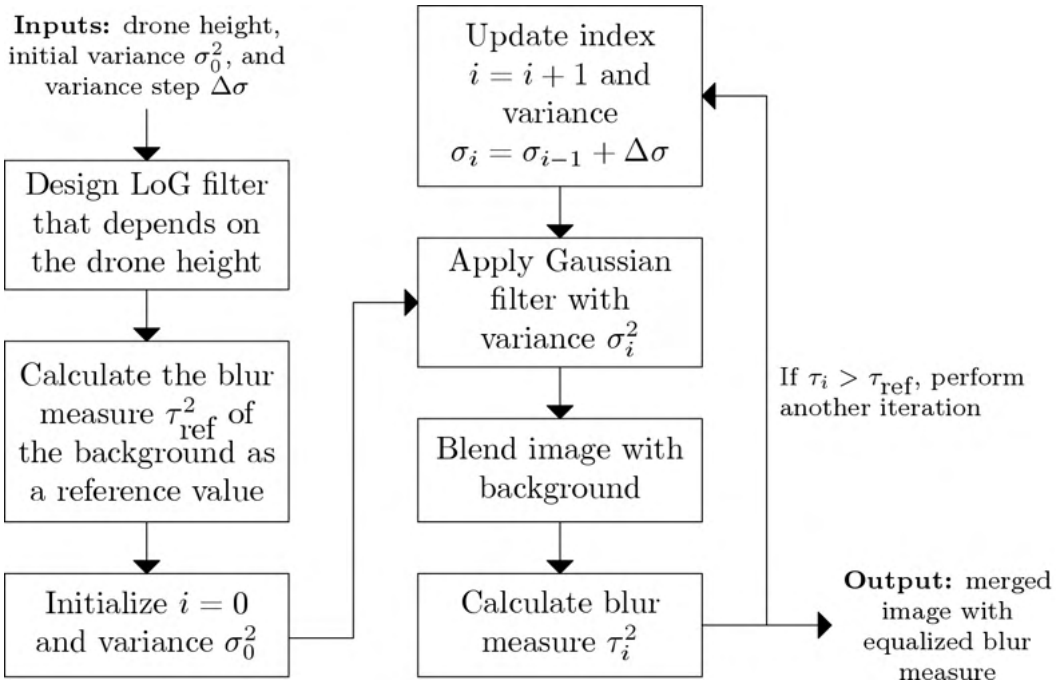
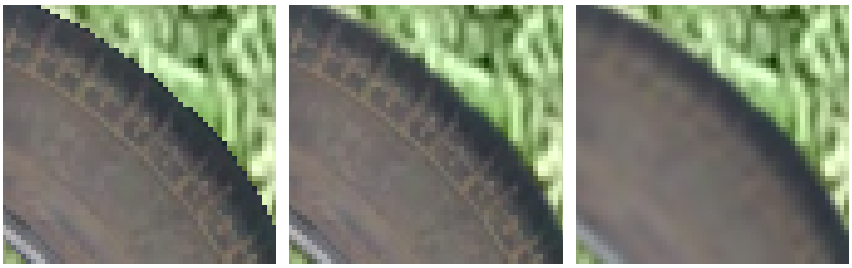


Figure 4.13: Block diagram of the blur and blend merging method.

Figure 4.14 shows the results when applying the three merging methods. The paste merging method introduces boundary artifacts while the blend merging method solves this problem. In addition, the blur and blend merging method blurs the artificial object image to obtain a blur measure that is closer to the background, which could help detect real objects.

(a) Paste. (b) Blend. (c) Blur and blend.



(d) Paste. (e) Blend. (f) Blur and blend.

Figure 4.14: Comparison of the merging methods.

4.4 Video data augmentation

The previous section presented a methodology to insert a given object image into a scene. In this section, the framework is extended to insert an object into a video instead of over an image background. The main difference is that using videos the subsequent frames should maintain consistency regarding object position and luminosity on the ground.

To paste an object in the same ground position, instead of using the GPS coordinates, which can have significant measuring errors, a visual approach is considered. As the designed trajectories have constant height and the drone do not rotate and have a stable gimbal, the movements can be considered pure translations and hence phase correlation technique [48] is adequate.

Considering two subsequent frames f_0, f_1 on grayscale and the translation movement represented by a vector (t_x, t_y) in pixels, the displaced frame is

$$f_1(x, y) = f_0(x - t_x, y - t_y). \tag{4.7}$$

Let F_0, F_1 be the discrete Fourier transform of f_0, f_1, respectively, and m, n the spatial frequencies. From Equation 4.7 and the shifting theorem

$$F_1(m, n) = F_0(m, n) \ . \ e^{-2\pi i(mt_x + nt_y)}. \tag{4.8}$$

The cross-correlation $c(x, y)$ between images is defined as the circular convolution of the functions $f_0(m, n), f_1(-m, -n)$. Then, applying the discrete Fourier transform

$$\begin{aligned} C(m, n) &= \frac{F_0 \ . \ F_1^*}{|F_0 \ . \ F_1^*|} \\ &= \frac{F_0 \ . \ F_0^*}{|F_0 \ . \ F_0^*|} \ . \ e^{2\pi i(mt_x + nt_y)} \end{aligned} \tag{4.9}$$

where $*$ indicates the complex conjugate.

Since the phase of $F_0 \ . \ F_0^*$ is always zero, taking the inverse Fourier transform, the normalized cross-correlation becomes

$$c(x, y) = \delta(x - t_x, y - t_y), \tag{4.10}$$

a delta function indicating the translational displacement.

The method used in this work is based on a extended version of the phase correlation [49] that considers subpixel registration. Also, in real applications, the movements are not perfectly pure translations and then the peak of the cross-correlation function is used as the estimated displacement. Figure 4.15 illustrates an example of video data augmentation using phase correlation.

Figure 4.15: Frames stepped every 10 of an augmented video fragment using phase correlation. The frames are cut to appropriately utilize the page space.

From a Litchi telemetry file corresponded to a video of the MBG database, one can approximately recover the drone trajectory with the GPS coordinates. Figure 4.16 shows the approximated drone trajectory and the evolution of the displacement components of the phase correlation over time.

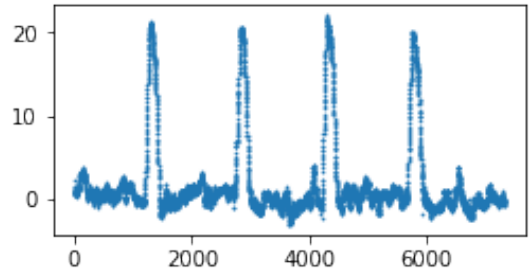
4.5 The augmented MBG video database

Finally, after presenting methods to create synthetic objects and adding them into videos, this section details the augmented MBG video database. The augmentation uses the image database presented in Section 4.1 composed of tires, which are the target class for the detector in Chapter 5.

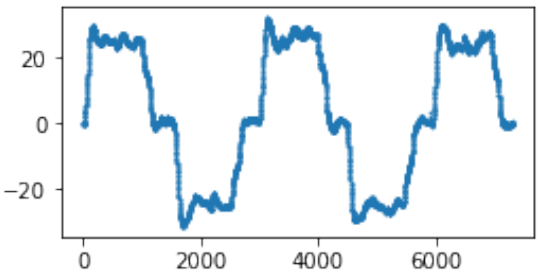
In contrast to increasing the number of objects in the database, maintaining part of the regions untouched is important to ensure real environments in the training set. An approximate proportion of one object per 20 m^2 is held and thus the augmented database has an approximate average of 20 synthetic objects per frame when the drone flew 20 m high. Also, inserted objects do not occlude another object, especially an annotated one, being it a tire or any other type.

The locations are randomly chosen and the three merging methods use the same configuration. Table 4.1 details the augmented video database and Figure 4.17 shows a frame as an example.

(a) GPS coordinates.



(b) x-direction.



(c) y-direction.

Figure 4.16: GPS coordinates from Litchi file illustrating the drone trajectory and the displacement in pixels estimated using phase correlation.

Table 4.1: Details of the augmented video database.

site name	**ground type**	**marked real tires**	**synthetic tires**	**covered area**
Gremio	low grass	6	235	4.7 $\times 10^3$ m^2
BLI1	high grass	6	120	2.4 $\times 10^3$ m^2
BLI2	street and low grass	6	60	1.2 $\times 10^3$ m^2
BLI3	street and building	0	425	8.5 $\times 10^3$ m^2
CCMN	wasteland	0	200	4.0 $\times 10^3$ m^2
FAU	wasteland	0	670	13.4 $\times 10^3$ m^2
Total	**-**	**18**	**1,330**	**26.6 $\times$ 10^3 m^2**

(a) From video Gremio (10 m).

(b) From video BLI2 (15 m).

Figure 4.17: Frames of the augmented MBG video database. Real tires in blue and artificial ones in red.

Chapter 5

Object detector

Object detection involves both localization and classification of objects. Localization can be done as a general mask, identifying for each image pixel if it is or not part of an object. However, segmentation masks often require many parameters and, if a supervised approach is taken, they make the annotation task very laborious. Instead of using masks, bounding boxes reduce each object label to only five components: the two coordinates of the central point, the object size as height and width, and the class label.

Current state-of-the-art object detectors are typically based on deep convolutional neural networks (CNNs) [50]. In traditional machine learning systems, feature extraction is usually the most challenging step, especially when dealing with multidimensional data. CNNs solve this problem by automatically learning features from images. In contrast, deep learning means that dozens of convolutional layers are used, thus requiring an extensive database to learn the many parameters without overfitting. This shows why data augmentation is so desirable in the deep learning context.

The chapter is concerned with evaluating the data augmentation approaches developed in Chapter 4. To simplify our comparisons, an image object detector is constructed as an initial strategy.

The most commonly used CNNs for object detection are based on R-CNN or its variants Fast and Faster R-CNN [51, 52]. These networks have the state-of-the-art results in terms of accuracy. In contrast, they are not simple to use and can be very time consuming to train properly. Another well-known CNN is the so-called YOLO neural network [53], that is much simpler and faster than the R-CNN and its variants, while still achieving near state-of-the-art detection performance. On the one hand, YOLO showed higher localization error than the Fast R-CNN, whereas, on the other hand, YOLO predicts fewer background detections, i.e., it tends to detect object where there is none [53]. As the application of this dissertation does not require perfect localization, YOLO is chosen to construct an object detector.

This chapter is organized as follows: Section 5.1 details the CNN components and presents the architecture and functionality of the YOLO network. Section 5.2 shows how images are extracted from the MBG video database in order to use an image object detector. Finally, Section 5.3 presents the achieved results: evidence that artificial data augmentation techniques are valuable for avoiding over training.

5.1 Convolutional neural networks

5.1.1 Fundamental structure

There are many types of CNNs but all have a fundamental structure in common. This subsection, based on [50], aims to briefly explain the main structure components.

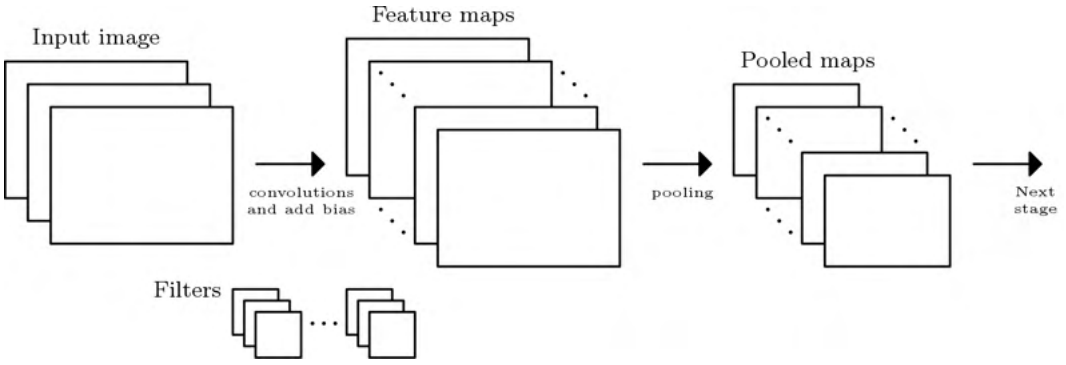


Figure 5.1: First stage of a CNN. The input image is convolved with a filter of same depth resulting in a feature map. Using n filters result in n feature maps. Then, each pooled map is a subsampling of a feature map.

Figures 5.1 illustrates the first stage of a CNN. The in-between stages have the same framework of the first, except for the input that becomes the output of the previous stage. In the end, the maps are converted into a one-dimensional vector that associates the image features to class probabilities. Each structure component shown in Figure 5.1 is detailed below.

Input maps

The input images usually consist of three channels, typically the red, green, and blue of the RGB color space. These channels are the input maps of the first stage of the network. On the next stages, the input maps can be either feature or pooled maps, as discussed in the next items.

Convolutional layers and feature maps

A convolutional layer performs convolutions of the input with filters of the same depth as the input. It adds bias after each convolution, generating the feature maps. Each feature map is responsible for saving different information about the signals. The network training is focused on estimating the parameters of these convolutional layers in order to recognize patterns for each object class.

Pooling layers and pooled maps

The pooling layers subsample the feature maps leading to a deeper level of abstraction, but these layers are not necessarily in every stage. The most common pooling layer is a $2 \times 2/2$ maxpool, which calculates the maximum of each sliding window of size 2×2 with stride 2.

Softmax function

The softmax function is used after the final stage of the network to normalize the final output into a probability distribution. Basically, it maps the network outputs into multi-class probabilities, allowing the classification of the network inputs.

5.1.2 YOLO neural network

The simplicity of YOLO neural networks is that "you only look once" at each input image. With a single convolutional network, it allows joint training of the entire network, directly optimizing detection performance [53].

YOLO detection networks can have different number of layers and output sizes at each stage. As an example, Table 5.1 describes the framework of a YOLO network that has 19 convolutional layers and five maxpooling layers. In sequence, a global average pooling layer and a softmax function are applied to calculate the class probabilities. Table 5.1 also shows the number of filters for each convolutional layer, as well as the size of filters and outputs at each layer.

To predict the bounding boxes within a single CNN, YOLO divides an input image into an $S \times S$ grid and uses the concept of anchors boxes: regions with predefined shape and size. Using the architecture as shown in Table 5.1, the parameter S is set as 26 and the input images have 832×832 resolution. If an input has a different resolution, it is automatically resized before going through the network.

Each grid cell predicts C conditional class probabilities and a fixed number B of bounding boxes based on these anchors and with respective confidence scores. The parameter B is set as 5 and C represents the number of classes which depends on the dataset. Figure 5.2 illustrates this process.

Table 5.1: YOLOv2 architecture. Adapted from [4].

Layer type	Number of filters	Filters size and stride	Output size
Convolutional	32	3×3	832×832
Maxpool		$2 \times 2/2$	416×416
Convolutional	64	3×3	416×416
Maxpool		$2 \times 2/2$	208×208
Convolutional	128	3×3	208×208
Convolutional	64	1×1	208×208
Convolutional	128	3×3	208×208
Maxpool		$2 \times 2/2$	104×104
Convolutional	256	3×3	104×104
Convolutional	128	1×1	104×104
Convolutional	256	3×3	104×104
Maxpool		$2 \times 2/2$	52×52
Convolutional	512	3×3	52×52
Convolutional	256	1×1	52×52
Convolutional	512	3×3	52×52
Convolutional	256	1×1	52×52
Convolutional	512	3×3	52×52
Maxpool		$2 \times 2/2$	26×26
Convolutional	1024	3×3	26×26
Convolutional	512	1×1	26×26
Convolutional	1024	3×3	26×26
Convolutional	512	1×1	26×26
Convolutional	1024	3×3	26×26
Convolutional	Number of classes	1×1	26×26
Avgpool		Global	Number of classes
Softmax			

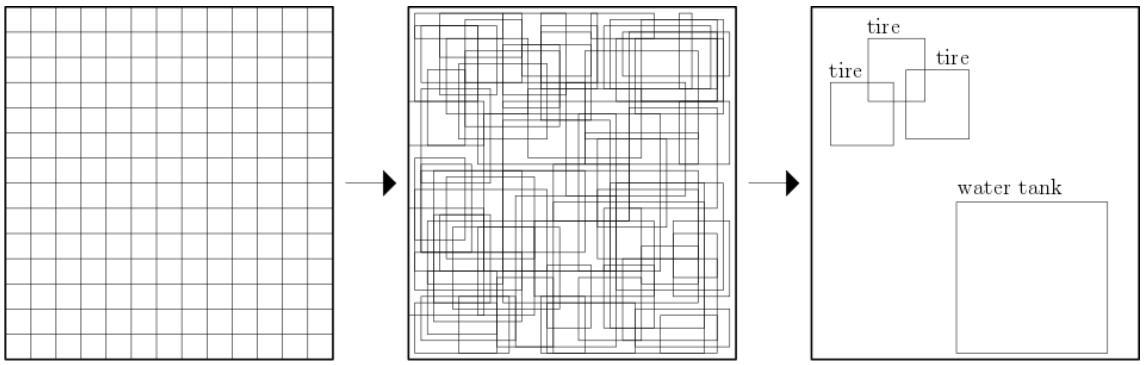


Figure 5.2: YOLO divides the image into grid cells and predicts bounding boxes with confidence scores and class probabilities. Based on: [4].

The box confidence score is given by $\Pr(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$, where $\Pr(\text{Object})$ is the probability that the box contains an object and $\text{IOU}_{\text{pred}}^{\text{truth}}$ is the intersection over union between the ground truth (x_i, y_i, w_i, h_i) and the predicted box $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$.

The loss function to be optimized during the training comprises three loss compo-

nents: localization, confidence, and classification loss; and the full expression given by

$$L = L_{\text{localization}} + L_{\text{confidence}} + L_{\text{classification}}, \tag{5.1}$$

where

$$\begin{aligned}
L_{\text{localization}} =& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \delta_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \delta_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right], \\
L_{\text{confidence}} =& \sum_{i=0}^{S^2} \sum_{j=0}^{B} \delta_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \delta_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2, \\
L_{\text{classification}} =& \sum_{i=0}^{S^2} \delta_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2,
\end{aligned}$$

and each additional parameter is defined bellow.

The functions δ_i^{obj}, δ_{ij}^{obj}, and $\delta_{ij}^{\text{noobj}}$ can only assume values 0 or 1: δ_i^{obj} assumes 1 when the object appears in cell i and 0 otherwise. A grid cell is said to be responsible for detecting an object if the center of this object falls within the grid cell. The function δ_{ij}^{obj} assumes 1 if the jth bounding box predictor in cell i is responsible for that prediction, otherwise, assumes 0. Finally, $\delta_{ij}^{\text{noobj}}$ is the complement of δ_{ij}^{obj}, i.e., $\delta_{ij}^{\text{noobj}} = 1 - \delta_{ij}^{\text{obj}}$.

The term $L_{\text{localization}}$ in Equation 5.1 represents the localization loss, the error between the predicted box and the ground truth. The parameter λ_{coord} is used to emphasize the localization loss ahead of the other loss components when optimizing the loss function during the training stage. The term $L_{\text{confidence}}$ in Equation 5.1 is the confidence loss. If an object is not detected in the box, the confidence error is multiplied by λ_{noobj} to deal with class imbalances. Finally, the term $L_{\text{classification}}$ in Equation 5.1 is the classification loss, where $p_i(c)$ is the conditional class probability of class c in cell i.

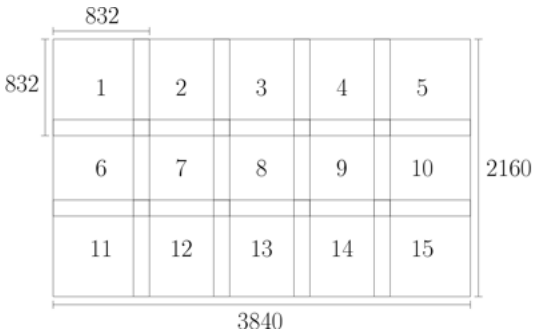
With a designed architecture and a defined loss function, the YOLO network is ready to be trained.

5.2 Image extraction from the MBG video database

In order to use an image object detector, images should be extracted from the MBG video database. Subsequent frames are very similar and may not have additional information in terms of object detection. Therefore, frames are selected based on phase correlation, a technique explained in Section 4.4 that calculates the translational displacement between frames. In our case, frames are selected every 720 pixels displacement, three times less than the front-rear resolution, in such a way that each object appears approximately three times on the extracted image set.

Since the application of this dissertation uses aerial videos, the objects tend to have low resolution. In fact, the annotated videos of the original MBG database have the maximum size of the object bounding boxes varying from approximately 20^2 to 330^2 squared pixels area. A common reference of evaluation metrics is described in [54] for the COCO dataset [55] where objects are split into three scales:

- Small objects: area $< 32^2$.
- Medium objects: $32^2 <$ area $< 96^2$.
- Large objects: area $> 96^2$.



(a) The sliding window used for cutting frames of size $3,840 \times 2,160$ into images of size 832×832.

(b) Original frame.

(c) Images extracted.

Figure 5.3: Cutting smaller images from the database frames.

Following this classification, the MBG database, in full resolution, contains objects in all scales. Therefore, in order to avoid reducing object resolutions, the extracted frames are cut into images of lower resolution. A sliding window is used with the same size as the network inputs, 832×832, and the largest possible stride while still covering the entire frame of size $3,840 \times 2,160$, as illustrated in Figure 5.3.

5.3 Tire detection

As a partial solution, this dissertation tackles the problem of detecting abandoned tires of the MBG database.

The first step is to split the MBG video database into training and test sets. In order to avoid contaminating the test set, an entire video is chosen to be in only one set. The current MBG database version contains only three annotated videos which are used as training set. Other three non-annotated videos are allocated in the test set. Table 5.2 has details of the split sets.

Table 5.2: Attributes of the recordings made at each local splited into trainning and test sets. FAU, BLI3, and CCMN are not annotated yet.

site name	**ground type**	**drone height**	**marked tires**	**video duration**	**covered area**	**set flag**
Gremio	low grass	10 m	6	212 s	4.7 $\times 10^3$ m^2	train
BLI1	high grass	10 m	6	97 s	2.4 $\times 10^3$ m^2	train
BLI2	street and low grass	20 m	6	23 s	1.2 $\times 10^3$ m^2	train
Train	**-**	**-**	**18**	**5.53 min**	**8.3 $\times$ 10^3 m^2**	**train**
FAU	wasteland	20 m	0	187 s	13.4 $\times 10^3$ m^2	test
BLI3	building	40 m	0	47 s	8.5 $\times 10^3$ m^2	test
CCMN	wasteland	15 m	0	75 s	4.0 $\times 10^3$ m^2	test
Test	**-**	**-**	**0**	**5.15 min**	**25.9 $\times$ 10^3 m^2**	**test**

From Table 5.2, the original database contains 18 annotated tires. After extracting images from the video as explained in Section 5.2, the training set has only 54 images containing bounding boxes of real tires.

A suitable way to deal with insufficient data when training a CNN is to use pre-trained weights. There are many available pre-trained weights based on large datasets such as COCO [55], ImageNet [56], PascalVOC [57], Open Images [58], and Tiny Images [59]. Using these weights tends to accelerate the training of custom objects and also tends to avoid overfitting.

The proposed tire detector uses the pre-trained weights of PascalVOC on YOLOv2 architecture, as described in Table 5.1. The VOC2007 is composed of natural images collected from the flickr photo-sharing web-site. There are 24,640

annotated objects of 20 classes, such as car, bicycle, person, dog, bottle, sofa, and table.

Problems associated with insufficient data can also be prevented by using data augmentation. By introducing image scaling and translations, by varying exposure and saturation, and many other transformations, one diversifies the data, reducing the chance of overfitting. YOLO already considers these data transformations to improve performance. One can note that these transformations are applied to the entire image, unlike the data augmentation methods presented in Chapter 4, that are used to add objects into image or video background scenes. These data modifications are respectively entitled from now on as YOLO data augmentation and merging data augmentation.

In this work, the YOLO training is executed considering the following scenarios:

- Case 1: No data augmentation.
- Case 2: YOLO data augmentation.
- Case 3: Merging data augmentation.
- Case 4: Both merging and YOLO data augmentation.

As mentioned in Section 4.5, the augmented MBG database has three versions, one for each merging method. For a fair comparison of these three approaches, the same parameters for tire insertion are used for the three methods: when randomly choosing the tire, warping parameters, and tire location. From the videos of the training set, 330 frames are extracted resulting in 4,950 images with 18 real and 1,237 artificial tires. In the test set, 450 images are evaluated containing 13 real tires.

Using 0.001 of learning rate and 1,000 iterations with batch 64, YOLO achieves the results presented in Table 5.3 and Table 5.4 when thresholding the class probability on 50%. Table 5.3 shows the total number of false positive bounding boxes predicted by YOLO in the test set. The results of training the network without augmentation present a large number of false positives.

Table 5.4 shows the number of real tires detected by YOLO. Since all videos of the test set are non-annotated, the tire detections are visually counted. An object appears in three images in average and if a tire is detected in at least one, it counts as a detected tire. Otherwise, if a tire is not detected in any of the images it appears, then it counts as a non-detected tire. Figure 5.4 shows an example of an object that is no longer detected when occlusion occurs, but still counting as a detected tire.

Table 5.3: Number of true and false positive bounding boxes (TP_b and FP_b) with probability greater than 0.5 of class tire in a total of 450 images.

	CCMN		FAU		BLI3		Total	
	TP_b	FP_b	TP_b	FP_b	TP_b	FP_b	TP_b	FP_b
Case 1	0	27	8	27	7	71	15	125
Case 2	0	11	5	13	3	60	8	84
Case 3 (Paste)	0	0	3	4	4	6	7	10
Case 3 (Blend)	0	0	2	2	5	3	7	5
Case 3 (Blur+Blend)	0	1	10	2	7	7	17	10
Case 4	0	0	7	3	5	4	12	8

Table 5.4: Number of true and false positive objects (TP_o and FN_o) using a 0.5 threshold.

	CCMN		FAU		BLI3		Total	
	TP_o	FN_o	TP_o	FN_o	TP_o	FN_o	TP_o	FN_o
Case 1	0	4	3	3	3	0	6	7
Case 2	0	4	2	4	2	1	4	9
Case 3 (Paste)	0	4	1	5	2	1	3	10
Case 3 (Blend)	0	4	1	5	2	1	3	10
Case 3 (Blur+Blend)	0	4	4	2	3	0	7	6
Case 4	0	4	3	3	3	0	6	7

Figure 5.4: Example of occlusion that harms detection.

One can note that none of the networks detected a tire from video CCMN. A possible reason is that this video is much darker than the ones used for training,

as shown in Subfigure 5.5(a), making it difficult to detect. In contrast, all the six networks are capable of detecting the at least two of three tires on video BLI3. Figure 5.6 presents the three detections.

(a) Tire (bottom right).

(b) Container.

(c) Bottle.

(d) People.

Figure 5.5: Examples of YOLO (without augmentation) detections on video CCMN.

Figure 5.5 shows four images tested using YOLO trained without augmentation. The network does not detect the tires but many false positives occur, such as a container, a bottle, and people. This suggests that the network did not learn the tire class, but rather background discrepancies.

Moreover, a bounding box location bias is visually noted on the tire detections using YOLO without augmentation, as shown in Figure 5.7. By looking through the tire annotations in the original database, one can note that the bounding boxes

(a) (b)

(c)

Figure 5.6: Examples of YOLO (blur and blend) detections on video BLI3. The tires of Subfigures (b) and (c) are detected by the six networks.

are not fitting perfectly the objects. Figure 5.7 gives a couple of examples. These errors may occur due to the linear interpolation used to accelerate the annotation process. Further investigation is necessary to fairly compare the results. However, another advantage of merging data augmentation comes up: the annotation of the artificial tires are perfectly fit by construction.

(a) (b) (c) (d) (e)

Figure 5.7: Errors on bounding box annotations (a)-(c) resulting in detection bias (d),(e).

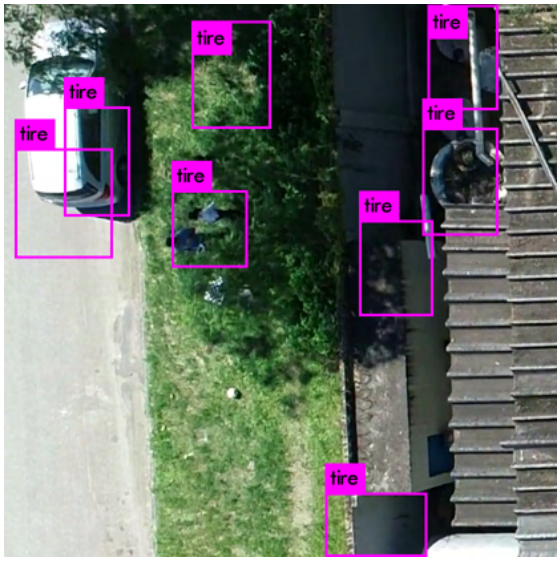
The experimental results show that, without data augmentation, the network tends to detect many false positives, as illustrated in Figure 5.5 and Figure 5.8. Using YOLO augmentation, the number of false positives is much lower, but also the number of tire detections decreases. A more extensive study regarding the network parameters may encounter an adequate compromise between true and false positives.

Comparing the merging methods, the blur and blend method reveals an advantage: the number of false positives is very low and presents the higher number of tire detections on videos FAU and BLI3. Figure 5.9 gives two examples: one which only blur and blend method detected and other which only the network trained without augmentation detected.

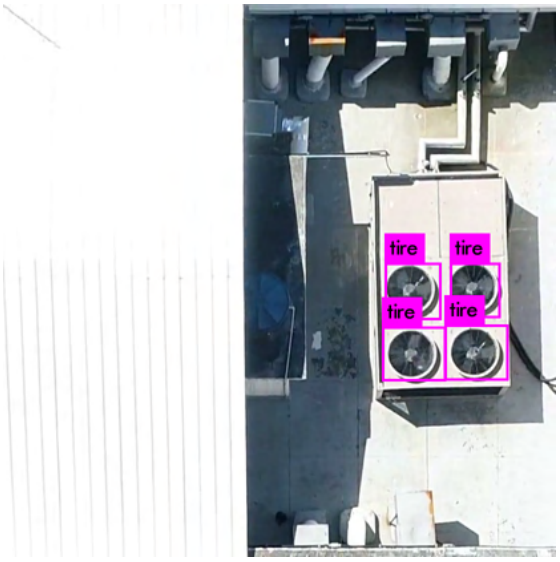
(a) Tire detected only by the network without data augmentation.

(b) Tire detected only by the network that uses blur and blend method.

Figure 5.9: Tires that are rarely detected by YOLO networks.



(a) Frame with the largest number of false positive detections using network without data augmentation.



(b) Fans detected by YOLO using blur and blend method.

(c) A shirt as a common false positive.

(d) Buckets as common false positives.

Figure 5.8: YOLO false detections.

These results indicate that artificial data augmentation techniques are very useful for reducing false positives. Moreover, in this initial investigation, the blur and blend method holds the highest detection performance. This indicates that approaches that attempt to hinder the detection of objects during training may be promising to increase overall detection accuracy.

Chapter 6

Conclusion

This dissertation presents a framework for detecting mosquito breeding-related objects. Starting by understanding the problem that health organizations are facing regarding mosquito-borne diseases, Chapter 2 proposes a system that uses computer vision in order to automatically detect these objects of interest.

The MBG video database presented in Chapter 3 is recorded by a drone and the current version is composed of six videos three of which are annotated frame-by-frame. Since state-of-the-art object detection systems usually require large training sets, data augmentation techniques are the focus of this dissertation and two strategies are analyzed in Chapter 5: (i) the insertion of artificial objects into the videos; (ii) transformations over the entire frames, such as introduce scaling, translations, and vary exposure and saturation.

The second strategy is widely used and the available detectors like YOLO already include these parameters for training the network. However, the first strategy requires many stages to be applied. Chapter 4 devises an image database composed of segmented tires and techniques of merging these images into the videos of the MBG database are presented.

Lastly, Chapter 5 uses a convolutional neural network detector to evaluate the different proposed methods for augmenting the database. The results indicate that artificial data augmentation reduces overfitting, improving the overall detection performance by the proposed network.

6.1 Future work

There is a lot of work to be done. First, further investigation is necessary to affirm and compare the benefits of each artificial data augmentation technique. Explore the data augmentation parameters such as the variability range of the transformations and the number of artificial insertions and also vary the detector parameters such as the number of training iterations and the probability thresholds are required.

However, in order to adequately evaluate the techniques, the extension of the video database and annotations is strongly necessary.

A possible direction is to extend the data augmentation approach to different objects of interest, such as containers, buckets, and bottles, which are already enclosed in the MBG database. Also, a dataset with objects of confusion can be used as distractions to the detector, which tends to reduce even more the false positives.

Moreover, the use of a video detector instead of an image detector is expected. The time information may benefit the detection by applying a temporal consistency analysis regarding standing objects.

Bibliography

[1] "Zika virus monitoring: Image analysis for identifying mosquito breeding grounds". `http://zika-virus-monitoring.weebly.com/`. Accessed: 2018.

[2] MEHRA, M., BAGRI, A., JIANG, X., et al. "Image analysis for identifying mosquito breeding grounds". In: *Proc. IEEE International Conference on Sensing, Communication and Networking*, pp. 1–6, 2016.

[3] AGARWAL, A., CHAUDHURI, U., CHAUDHURI, S., et al. "Detection of potential mosquito breeding sites based on community sourced geotagged images". In: *Proc. Geospatial InfoFusion and Video Analytics IV; and Motion Imagery for ISR and Situational Awareness II*, v. 9089, p. 90890M, 2014.

[4] REDMON, J., FARHADI, A. "YOLO9000: Better, faster, stronger", *arXiv preprint arXiv:1612.08242*, 2016.

[5] IOOS, S., MALLET, H.-P., GOFFART, I. L., et al. "Current Zika virus epidemiology and recent epidemics", *Medecine et Maladies Infectieuses*, v. 44, n. 7, pp. 302–307, 2014.

[6] MONATH, T. P. "Yellow fever: an update", *The Lancet Infectious Diseases*, v. 1, n. 1, pp. 11–20, 2001.

[7] "Levantamento rápido de índices para Aedes aegypti para vigilância etimológica do Aedes aegypti no Brasil". `http://bvsms.saude.gov.br/bvs/publicacoes/levantamento_rapido_indices_aedes_aegypti.pdf`, 2013. Accessed: 2018.

[8] CARABALLO, H., KING, K. "Emergency department management of mosquito-borne illness: malaria, dengue, and West Nile virus", *Emergency Medicine Practice*, v. 16, n. 5, pp. 1–23, 2014.

[9] REITER, P. "Climate change and mosquito-borne disease", *Environmental Health Perspectives*, v. 109, n. Suppl 1, pp. 141, 2001.

[10] HARRINGTON, L. C., EDMAN, J. D., SCOTT, T. W. "Why do female Aedes aegypti (Diptera: Culicidae) feed preferentially and frequently on human blood?" *Journal of Medical Entomology*, v. 38, n. 3, pp. 411–422, 2001.

[11] SHEPARD, D. S., COUDEVILLE, L., HALASA, Y. A., et al. "Economic impact of dengue illness in the Americas", *The American Journal of Tropical Medicine and Hygiene*, v. 84, n. 2, pp. 200–207, 2011.

[12] MLAKAR, J., KORVA, M., TUL, N., et al. "Zika virus associated with microcephaly", *New England Journal of Medicine*, v. 374, n. 10, pp. 951–958, 2016.

[13] CARON, M., PAUPY, C., GRARD, G., et al. "Recent introduction and rapid dissemination of Chikungunya virus and Dengue virus serotype 2 associated with human and mosquito coinfections in Gabon, central Africa", *Clinical Infectious Diseases*, v. 55, n. 6, pp. e45–e53, 2012.

[14] RÜCKERT, C., WEGER-LUCARELLI, J., GARCIA-LUNA, S. M., et al. "Impact of simultaneous exposure to arboviruses on infection and transmission by Aedes aegypti mosquitoes", *Nature Communications*, v. 8, pp. 15412, 2017.

[15] MOREIRA, L. A., ITURBE-ORMAETXE, I., JEFFERY, J. A., et al. "A Wolbachia symbiont in Aedes aegypti limits infection with dengue, Chikungunya, and Plasmodium", *Cell*, v. 139, n. 7, pp. 1268–1278, 2009.

[16] MCMENIMAN, C. J., LANE, R. V., CASS, B. N., et al. "Stable introduction of a life-shortening Wolbachia infection into the mosquito Aedes aegypti", *Science*, v. 323, n. 5910, pp. 141–144, 2009.

[17] DUTRA, H. L. C., ROCHA, M. N., DIAS, F. B. S., et al. "Wolbachia blocks currently circulating Zika virus isolates in Brazilian Aedes aegypti mosquitoes", *Cell host & microbe*, v. 19, n. 6, pp. 771–774, 2016.

[18] TUN-LIN, W., LENHART, A., NAM, V. S., et al. "Reducing costs and operational constraints of dengue vector control by targeting productive breeding places: a multi-country non-inferiority cluster randomized trial", *Tropical Medicine & International Health*, v. 14, n. 9, pp. 1143–1153, 2009.

[19] LAGROTTA, M. T. F. *Geoprocessamento de indicadores entomológicos na identificação de áreas, imóveis e recipientes (chaves) no controle do Aedes aegypti*. Tese de Doutorado, 2006.

[20] PRASAD, M. G., CHAKRABORTY, A., CHALASANI, R., et al. "Quadcopter-based stagnant water identification". In: *Proc. National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, pp. 1–4, 2015.

[21] LOWE, D. G. "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, v. 60, n. 2, pp. 91–110, 2004.

[22] BAY, H., ESS, A., TUYTELAARS, T., et al. "Speeded-up robust features (SURF)", *Computer Vision and Image Understanding*, v. 110, n. 3, pp. 346–359, 2008.

[23] JOLLIFFE, I. "Principal component analysis". In: *International Encyclopedia of Statistical Science*, pp. 1094–1096, 2011.

[24] DIAS, T., ALVES, V., ALVES, H., et al. "Autonomous detection of mosquito-breeding habitats using an unmanned aerial vehicle". In: *Proc. Latin American Robotic Symposium*, pp. 351–356, 2018.

[25] PASSOS, W. L., DIAS, T. M., JUNIOR, H. A., et al. "Acerca da deteção automática de focos do mosquito Aedes aegypti". In: *Anais do Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 2018.

[26] "DJI Phantom 4 Pro specifications". `http://www.dji.com/phantom-4-pro/info`. Accessed: 2018.

[27] "Litchi App". `http://www.flylitchi.com`, . Accessed: 2018.

[28] SNYDER, J. P. *Map projections – a working manual*, v. 1395. US Government Printing Office, 1987.

[29] ANDREW, A. M. "Another efficient algorithm for convex hulls in two dimensions", *Information Processing Letters*, v. 9, n. 5, pp. 216–219, 1979.

[30] FREEMAN, H., SHAPIRA, R. "Determining the minimum-area encasing rectangle for an arbitrary closed curve", *Communications of the ACM*, v. 18, n. 7, pp. 409–413, 1975.

[31] AGÊNCIA NACIONAL DE AVIAÇÃO CIVIL. "Regulamento Brasileiro de Aviação Civil Especial – E no 94". 2017.

[32] ZHANG, Z. "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 11, pp. 1330–1334, 2000.

[33] "Litchi Mission Hub". `http://www.flylitchi.com/hub`, . Accessed: 2018.

[34] SIMARD, P. Y., STEINKRAUS, D., PLATT, J. C. "Best practices for convolutional neural networks applied to visual document analysis". In: *Proc. International Conference on Document Analysis and Recognition*, p. 958, 2003.

[35] PEREZ, L., WANG, J. "The effectiveness of data augmentation in image classification using deep learning", *arXiv preprint arXiv:1712.04621*, 2017.

[36] DWIBEDI, D., MISRA, I., HEBERT, M. "Cut, paste and learn: surprisingly easy synthesis for instance detection". In: *Proc. IEEE International Conference on Computer Vision*, pp. 1310–1319, 2017.

[37] DVORNIK, N., MAIRAL, J., SCHMID, C. "On the importance of visual context for data augmentation in scene understanding", *arXiv preprint arXiv:1809.02492*, 2018.

[38] TREMBLAY, J., PRAKASH, A., ACUNA, D., et al. "Training deep networks with synthetic data: bridging the reality gap by domain randomization". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977, 2018.

[39] CHOU, Y.-M., CHEN, C.-H., LIU, K.-H., et al. "Changing background to foreground: an augmentation method based on conditional generative network for stingray detection". In: *Proc. IEEE International Conference on Image Processing*, pp. 2740–2744, 2018.

[40] CIREŞAN, D. C., MEIER, U., GAMBARDELLA, L. M., et al. "Deep, big, simple neural nets for handwritten digit recognition", *Neural Computation*, v. 22, n. 12, pp. 3207–3220, 2010.

[41] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., et al. "Generative adversarial nets". In: *Proc. Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[42] LEDIG, C., WANG, Z., SHI, W., et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 105–114, 2017.

[43] BOJANOWSKI, P., JOULIN, A., LOPEZ-PAZ, D., et al. "Optimizing the latent space of generative networks", *arXiv preprint arXiv:1707.05776*, 2017.

[44] SOILLE, P. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[45] GUNN, S. R. "On the discrete representation of the Laplacian of Gaussian", *Pattern Recognition*, v. 32, n. 8, pp. 1463–1472, 1999.

[46] PECH-PACHECO, J. L., CRISTÓBAL, G., CHAMORRO-MARTINEZ, J., et al. "Diatom autofocusing in brightfield microscopy: a comparative study". In: *Proc. IEEE International Conference on Pattern Recognition*, v. 3, pp. 314–317, 2000.

[47] JUNIOR, A. K., COSTA, M. G., COSTA FILHO, C. F., et al. "Evaluation of autofocus functions of conventional sputum smear microscopy for tuberculosis". In: *PRoc. International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3041–3044, 2010.

[48] PEARSON, D. "Image Processing (Essex Series in Telecommunication and Information Systems)". 1991.

[49] FOROOSH, H., ZERUBIA, J. B., BERTHOD, M. "Extension of phase correlation to subpixel registration", *IEEE Transactions on Image Processing*, v. 11, n. 3, pp. 188–200, 2002.

[50] GONZALEZ, R. C. "Deep convolutional neural networks", *IEEE Signal Processing Magazine*, v. 35, n. 6, pp. 79–87, 2018.

[51] GIRSHICK, R. "Fast r-cnn". In: *Proc. IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[52] REN, S., HE, K., GIRSHICK, R., et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Proc. Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[53] REDMON, J., DIVVALA, S., GIRSHICK, R., et al. "You only look once: Unified, real-time object detection". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[54] "COCO: Common objects in context". `http://cocodataset.org/#detection-eval`. Accessed: 2018.

[55] LIN, T.-Y., MAIRE, M., BELONGIE, S., et al. "Microsoft coco: Common objects in context". In: *Proc. European Conference on Computer Vision*, pp. 740–755, 2014.

[56] DENG, J., DONG, W., SOCHER, R., et al. "Imagenet: A large-scale hierarchical image database". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[57] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K., et al. "The pascal visual object classes (voc) challenge", *International Journal of Computer Vision*, v. 88, n. 2, pp. 303–338, 2010.

[58] KUZNETSOVA, A., ROM, H., ALLDRIN, N., et al. "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale", *arXiv preprint arXiv:1811.00982*, 2018.

[59] TORRALBA, A., FERGUS, R., FREEMAN, W. T. "80 million tiny images: A large data set for nonparametric object and scene recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 30, n. 11, pp. 1958–1970, 2008.