TÉCNICO LISBOA

Tracking and Following a Moving Person Onboard a Pocket Drone

Tomás Vasconcelos Ferreira de Ávila Duro

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. José Raul Carreira Azinheira

Examination Committee

Chairperson: Prof. João Manuel Lage de Miranda Lemos
Supervisor: Prof. José Raul Carreira Azinheira
Member of the Committee:

October 2016

Resumo

Pilotar um drone é uma tarefa complicada, sendo normalmente necessário um piloto treinado. Esta dissertação descreve uma estratégia baseada em visão computacional para fazer seguimento automático de uma pessoa por um Micro-Aerial-Vehicle (MAV). Com o sistema apresentado, é possível usar um drone para filmar ou tirar fotografias de sítios previamente inacessíveis, sem a necessidade de um piloto humano a controlar a aeronave. Composto por dois componentes principais, um tracker e um sistema de controlo, o tracker executa a função de estimar a posição da pessoa que está a ser seguida, enquanto que o sistema de controlo cumpre a função de colocar o drone perto da pessoa. Limitado em peso, consumo de energia e capacidade de processamento, o sistema resulta de um delicado balanço entre tais parâmetros. Os contributos principais deste trabalho são a análise de dois trackers visuais do estado da arte, Struck e KCF, o sistema de controlo que usa o output do tracker para executar a tarefa de seguimento, e um novo tracker, desenvolvido para ser computacionalmente leve, de modo a conseguir correr a bordo do pocket drone. Este novo tracker é baseado em extracção de HOG features e faz uso de regressão logística para treinar um detector no aspecto de uma pessoa. Os resultados do KCF e do Struck mostraram que estes são demasiado exigentes para correr a bordo do drone, enquanto que o novo tracker consegue, produzindo resultados promissores.

Abstract

Flying a drone is not an easy task, usually requiring a trained pilot. In this dissertation a vision based strategy is presented, designed to work fully onboard a small pocket drone, for autonomously tracking and following a person. With the presented system it is possible to use a drone for filming or taking pictures from previously inaccessible places without the need for a person controlling the aircraft. The system is comprised by two main components, a tracker and a control system. The tracker has the function of estimating the position of the person to be followed, while the control system gets the drone near that person. Limited by payload weight, power consumption and processing power, the system results in a delicate balance between these constraints. The main contributions of this paper are the comparison between two state-of-the-art visual trackers running on paparazzi, Struck and KCF, the control system that uses the output location of the tracker to perform the person following task, and a new tracker, developed to be as computationally light as possible, so that it can run onboard a small pocket drone, based on HOG feature extraction, it uses logistic regression to train a detector on the appearance of a person. Both KCF and Struck proved too demanding to run onboard the drone, while the new tracker runs fully onboard, producing very promising results.

Contents

Chapter 1

Introduction

In the last few years, drones have become relatively inexpensive, leading to a growing number of applications for them. Their agility and ability to reach previously inaccessible places makes them particularly attractive for applications such as gathering aerial footage, for cinema and television, and in particular for sports coverage. These tasks usually consist of following a moving target, either a person or an object, and consistently pointing a camera at them. However, drones are notoriously difficult to control, requiring highly skilled trained pilots for their safe operation, so it is not straightforward to use them for the purpose of target-following.

An alternative to manually piloting the drone is having it fly autonomously. This is a very active field of research, and in the context of following a moving target, the main challenges are detecting and tracking the position of the target, and guiding the drone towards it. Recently, commercial products have been developed for this task, like the Lily Drone[1], the AirDog[2], or the Hexo+[3]. These rely on GPS markers on the target which transmit their position back to the drone. Although this solution works mostly fine outdoors, it does not work in GPS-denied environments, making it unsuitable for indoor applications, among others.

To overcome this limitation, a vision-based system can be used instead. Object detection and tracking is an extensively studied problem in the computer vision literature, and many different methods have been developed to detect the in-frame position of the person or object that is being tracked.

Vision-based tracking is a very complex task, dealing with a number of different problems [4][5], namely illumination changes, occlusions, changes to the appearance of the object, and the motion of both camera and target. By imposing constraints on the tracking problem, such as assuming that the movement of the target is smooth, with small displacements, or that a person has mostly an upright position, tracking strategies can be simplified. In cases where the camera has a fixed position, subtracting the background from the image can greatly simplify the problem, as it becomes much easier to identify moving objects in the scene.

Some trackers continuously update the model of the target online, resulting in a greater accuracy, due to the increased robustness to changes in the scene, as well as in the appearance of the target itself. Failure detection and redetection capabilities also add greatly to tracking robustness as it allows

the tracker to reinitialise when the task deteriorates below a certain threshold. Though often used strategies, they may be too demanding to allow for smooth onboard running of the algorithm.

But visual trackers are, in general, computationally demanding, and when the purpose is to run it onboard a MAV, special care has to be taken, due to its limited memory, processing power, and energy consumption. A delicate balance has to be maintained between these constraints, and the accuracy of the tracker. A commonly employed solution to this problem is sending the images from the camera to a ground-station, e.g. a personal computer, where the heavy calculations are performed, and then send the steering controls back to the drone. However, the use of a ground station is often impractical due to its reduced portability, eliminating one of the main virtues of using a drone for this task.

As far as we are aware of, no previous vision-based tracker implementation has managed to work fully onboard an inexpensive drone, without the assistance of a special feature to identify the target that is being followed, and distinguish it from others. In this work, we aim at a system which does not require such special identifiers on the target, and that works fully onboard a small pocket drone, highly limited in both memory and processing power. We ported two state of the art trackers into the Paparazzi UAV[6][7] autopilot system, namely Struck[8] and KCF[9], and tested them on a Parrot Bebop drone. Both trackers are discriminative, and update the model of the target online with each new frame, making them very robust, but computationally unsuitable for onboard use. We then develop a simple tracking algorithm, designed to be computationally efficient, so that it can run onboard a pocket drone. This tracker is based on tracking by detection, and uses HOG features to describe the model of the person, a logistic regression classifier is trained offline with a few hundred images of both positive and negative examples of people.

The remainder of the dissertation is organised as follows. In chapter 2 the related work is presented and in chapter 3 an overview of the different stages of the tracking and following process is given. Chapter 4 gives a detailed explanation of the visual trackers and chapter 5 explains the control system design. In chapter 6 we have the experimental results and finally in chapter 7 the conclusions and future work.

Chapter 2

Related Work

There have been many previous vision-based approaches to the task of using a drone for person and object following. Most works use very similar control policies, achieving good results with a simple PID controller. What distinguishes the different systems is the tracker, used to estimate the position of the target. A recent survey by Smeulders et al [4] reviews and makes a detailed comparison between 19 of the most used and cited trackers in the literature. All the trackers are evaluated under the same test conditions, using 12 different performance metrics. In the following review we focus on the simplest trackers and on the ones that perform best.

Some of the simpler tracking strategies are based on colour thresholding, images are filtered in order to identify the target by its colour. The methods are not very robust, having difficulties with tracking targets with similar colours to those of the background. Dang et al [10] use colour thresholding to track and follow a small red object from above. Even though such a system should be capable of running entirely onboard the AR Drone they used, they actually perform the computations offboard, using a ground station.

The similar problem of tracking an object in the ground was also explored by Teuliere et al[11], where a red car is tracked using a particle filter with a simple velocity-based model, and an appearance model based on colour histograms. The tracker can handle large displacements, and both partial and full occlusions. When the tracker fails, a redetection strategy is employed, based on running several Camshift [12] searches, and using the best among the candidate positions to reinitialise the tracker. Calculations are performed offboard, on a ground station.

Mondragón et al[13] also use the Camshift algorithm to track a specific colour across frames. In each frame, both the centre of the colour region, as well as its circumference, are calculated. Often used in tracking, the Camshift algorithm only uses colour and as such suffers from the same problems as most colour based strategies. Factors that influence colour in the image, such as dim illumination or too much illumination, as well as different coloured lighting, e.g. lightbulbs with different illumination colours, tend to damage the performance of this kind of methods.

In recent work, Smedt et al [14] develop a system to track and follow a walking person, onboard a UAV, using a colour-based particle filter as a tracker, and the Aggregate Channel Feature (ACF) [15]

detector. The detector is used to reinitialise the tracker whenever its confidence level drops below a certain threshold, increasing the overall robustness of the system. To increase the speed of the detector, the ground plane is estimated, and used to reduce the region over which the target is searched.

More sophisticated methods have been developed to deal with the problems of the previously proposed colour based techniques. Viola and Jones [16] propose a framework for face detection, making use of integral images to quickly calculate the Haar-like features, which they introduce. AdaBoost [17] is then used for feature selection, and to train a classifier.

The two trackers that performed best in the survey by Smeulders were Struck [8] and TLD [18].

The Struck tracker was developed by Hare et al[8]. It makes use of a support vector machine (SVM), trained on the appearance of each training example, as well as on the translation from the previous detection. By incorporating translations on the learning phase, the algorithm is able to directly output the position of the new detection, unlike traditional trackers, which output a score corresponding to the likelihood of each tested location. We used Struck in some of our experiments, a more detailed description is given in section 4.1.

More recently, Kalal et al developed the TLD tracking framework, consisting of 3 components: a Tracker, a Learning component, and a Detector. The main purpose of the tracker is estimating the motion of the target between successive frames. However, should the target move too much, possibly even getting out of sight, the detector then starts scanning each new frame independently, in order to find the target again and reinitialise the tracker. The learning component trains the detector based on the performance of both the tracker and the detector, discriminating the appearance of the target against the background, through the use of positive and negative training examples.

However, the TLD algorithm is computationally very demanding, and it is currently unfeasible to run it onboard a consumer drone. In the work by Pestana et al[19], a ground station is used to run the TLD algorithm on the images sent by an AR Drone 2.0. The control commands are then generated by a PD controller, and sent back to the drone. Similar work has been performed by Bart et al[20], again using a ground station for offboard computing of the TLD tracker.

Many trackers are based on the Histogram of Oriented Gradients (HOG) feature descriptor, which was introduced by Dalal and Trigs [21], and originally used to train a SVM for pedestrian detection. As we make use of HOG features through out this work, a detailed explanation on how to calculate them is presented in appendix A.1. It has since been widely used in various branches of computer vision, for object and people detection. Danelljan et al [22] worked on a person-following framework, composed by an Adaptive Color Tracker [23], and a SVM trained on HOG features to recover from tracking failures. All computations are performed on a ground station. In recent work by Imamura et al [24], a linear SVM is trained on HOG features for human detection, running offboard, with the system being validated using an AR Drone.

In Haag et al[25], two correlation filter-based trackers are used for person pursuit, using a low cost quadcopter, and the algorithms themselves running offboard. They extend the DSST [26] and KCF [9] tracking algorithms with target loss detection and redetection capabilities. Both trackers make use of HOG features for improved results, when compared with simple pixel intensity values. KCF is one of the

trackers used in our work, and a detailed explanation of the algorithm is given in section 4.2.

The problem of recovering the 3D trajectory of a moving person by a UAV was researched by Lim et al [27]. They use a monocular camera mounted on a drone to autonomously follow a person, while recovering its trajectory. The Struck algorithm is used to track the person, and a detector trained with AdaBoost to recover from tracking failures.

From the wide variety of approaches used by the various works in the literature, we observe that no solution has yet emerged as the single best strategy for the task of target-following using a drone. Most of the systems make use of a ground station to perform the intensive computations, sacrificing mobility, and the ones that do manage to perform the calculations onboard require powerful specialized hardware, usually not found in small inexpensive drones.

Chapter 3

System Overview

In this chapter, an overview of the tracking and following process is given, we go over the several stages of the algorithms in more detail, as to give a clear understanding of how each of the system components fits in the general pipeline of the process.

Our aim is to figure out what strategy enables us to implement a person following system onboard a pocket drone. The parrot Bebop was used as a platform to test two state of the art algorithms, KCF and Struck. Both trackers achieve some of the best results among the current trackers and as such they were used in a preliminary phase to help identify the computational constraints of the hardware.

3.1 Parrot Bebop

The Bebop, shown in Fig. 3.1, is a commercially available platform. It uses a Parrot P7 dual-core CPU Cortex A9, a Quad core GPU and 8 Gb of flash memory. The camera has a Fisheye lens 180°1/2,3 with 1920x1080p pixel definition, that works up to 30 fps.

Figure 3.1: Parrot Bebop.

3.2 Pocket Drone and Stereo Camera

Then we made use of a Pocket drone equipped with a stereo camera to implement a simpler tracker, capable of running with lower computational resources. The pocket drone is a very small and light aircraft, it only weighs 43 grams when equipped with a stereo-vision camera, seen in Fig. 3.2. The algorithms run directly on the stereoboard, the board that performs the processing of the images, which is limited to 196kB of memory with a processor speed of 168MHz. The camera has an horizontal field of view of 57,4 °and can handle a maximum frame rate of 30 frames per second. The maximum pixel resolution of the images is 660×492 though we use a smaller resolution image of 128×96.

Figure 3.2: Pocket drone with the stereo camera attached.

3.3 Tracking and Following Process

As it can be seen in figure 3.3, each cycle of the system starts with a new frame. The way a visual tracker works, it receives as inputs the sequential frames from a camera, each frame is analyzed and used to estimate the position of the target being tracked. Some trackers, as discussed in chapter 1, also use the information in each frame to update the internal model of the target. Following the analysis of the frame, the tracker outputs the predicted location of the target in the format of a bounding box.

Figure 3.3: General overview of the tracking and following process.

Most trackers, including the three we used in this work, make use of a bounding box to describe the output predicted location for the target, Fig. 3.4 shows an example of the output for a single frame. Four parameters fully describe a bounding box, two for location and two for size, respectively the x and y pixel position for the center of the box, P_c, and the width, BB_w, and height, BB_h for the size.

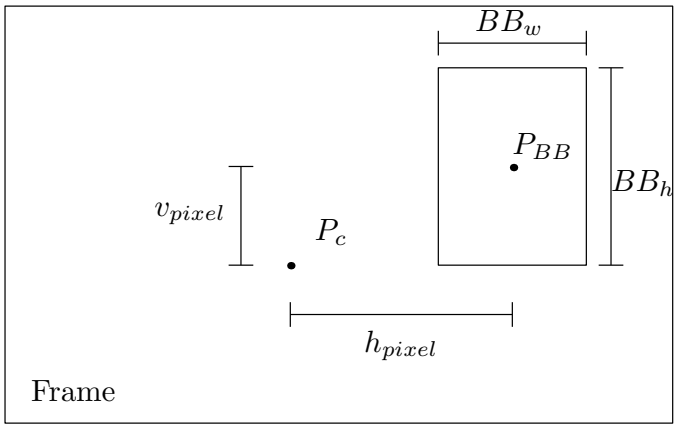


Figure 3.4: Bounding Box

Having the estimated position of the target calculated, we need to steer the aircraft controls to get it close to the target, the following task. This is done using the bounding box's four parameters to calculate references which are then fed to paparazzi internal control loops. The calculation of these references is explained in detail in chapter 5.

Chapter 4

Visual Trackers

In this chapter, a detailed explanation of each of the trackers we used in our experiments is given, namely the Struck [8] and KCF [9] trackers, and the Pocket Tracker, that we developed to work onboard the Pocket Drone.

Both Struck and KCF are adaptive trackers, using information provided in the first and following frames to build a model of the target, with no need for a pre-trained detector. The pocket tracker works differently, it was trained offline in the task of people detection, and is consequently less robust than the other two.

4.1 Struck

The authors of Struck [8] make a few comments on what they see as shortcomings of traditional adaptive tracking by detection approaches. Learning is usually decoupled from the information on the translation from the previous detection, based only on a binary label given to each training example. Moreover, examples are equally weighted, independent of the level of overlap with the bounding box of the tracker. Small inaccuracies during tracking can lead to poorly labeled examples, which worsens the tracker in the training phase, in turn leading to cyclic deterioration of the tracker. Usual strategies to generate training samples are based on fixed rules, such as considering the predicted location in the current frame as a positive training sample, and locations that are a certain distance from this locations as negative samples. This is an error prone strategy, as there is no control on what the negative samples look like, these samples may result in deterioration of the performance of the tracker.

The framework of Struck is built to deal with these issues, instead of learning a classifier, its prediction function directly estimates the objects transformation between each frame. The output space becomes all possible object transformations inside a search region, instead of binary labels ± 1.

The prediction function is learned in a structured output SVM [28] [29]. SVMs keep both positive and negative samples from previous frames, called support vectors, a frame for which support vectors have been kept is called a support pattern.

In the optimisation process of the prediction function, a loss function that depends on the level of

overlap with the current bounding box is incorporated. By doing so, they manage to differentiate negative samples that have significant overlap with the bounding box from those that have little or no overlap, addressing their comment on giving the same weight to all negative samples.

Three main optimisation stages perform the maintenance of the support vectors, namely $ProcessNew$, $ProcessOld$ and $Optimize$. $ProcessNew$ processes a new example, having the ability to add both positive and negative support vectors, while $ProcessOld$, as the name indicates, goes back to a previous support pattern, only having the capacity to add negative support vector or change the coefficients associated with the existing support vectors. Unlike $ProcessNew$ and $ProcessOld$, the $Optimize$ stage cannot add support vectors, only change the coefficients associated with a existing support pattern, being as such the less expensive operation among the three. Each full cycle of Struck runs $ProcessNew$ once, and multiple instances of $ProcessOld$ and $Optimize$.

The last big component of the tracker is its budget maintenance system. In SVMs the number of support vectors is not bounded, in a task like tracking they can grow indefinitely if a mechanism is not incorporated to deal with the maintenance of support vectors. With more support vectors, storage and computational costs grow, $ProcessNew$ and $ProcessOld$ also become more costly operations. In Struck, the budget maintenance process is run every time a new support vector may have been added, it checks whether or not the number of SVs has exceeded the budget limit, and if it has, the support vector that causes the smallest change to the norm of the weight vector is removed, causing little impact on the performance of the tracker.

As in support vector machine algorithms, by using different kernel function they can implement different features as well as a combination of more than one. In their experiments they make use of features like raw pixel intensities, Haar features and intensity histogram features, obtaining the best results for the combination of Haar features with intensity histograms. The overall structure of Struck, incorporating translations directly on the learning phase, together with their support vectors optimisation and budgeting mechanisms made for some of the best results among current adaptive trackers.

4.2 KCF

The tracker proposed by Henriques [9] deals with what they consider to be the main factor inhibiting performance in trackers, the undersampling, especially of negative samples. In most modern trackers, incorporating more samples comes at both a computational and memory cost, and as such they usually try to add as many samples as possible, while maintaining both costs viable.

KCF manages to incorporate thousands of samples, and even though it is based on Kernel Ridge Regression as proposed by [30], it manages to break the "Curse of Kernelization". The samples are generated by a specific translation model, circulant matrices, which in the Fourier domain allows for a fast learning algorithm. The tracker can also use multiple features, e.g. HOG features, which results in an increment to its performance.

Training samples are generated by a base sample that is used as a positive example, then multiple negative samples result from translations of that base sample. The aforementioned circulant matrix

results from making each of its lines the base sample with increasing translations. What makes the use of circulant matrices so appealing is that these matrices become diagonal by discrete Fourier transforming them.

Working in the Fourier domain with training data in the format of circular shifts allows for the simplification of linear regression, matrices become diagonal and all operations become element-wise on the diagonal elements.

Powerful non-linear regression functions can be used with the kernel trick, the optimisation problem maintains its linearity, in the dual space, with the downside that now the complexity of the resulting regression function grows with the number of samples, the "curse of kernelization". By proving that the kernel matrix is circulant for certain kernel functions, including Gaussian, Linear and Polynomial, among others, and working in the Fourier domain the regression problem is greatly accelerated.

The detection phase is also significantly accelerated. By modelling the patches that are to be tested by cyclic shifts, the entire detection response can be computed very fast, again in the Fourier domain.

The dual formulation of the problem allows for easy implementation of multiple channels, like HOG features, by simply summing the results for each channel in the Fourier domain. As referred before, the use of HOG features gives a good performance enhancement. Modelling samples with translations and making the resulting data and kernel matrices circulant allowed KFC to be one of the fastest state-of-the-art trackers, with speeds of hundreds of frames per second when compared to the low frame rates of other state-of-the-art trackers, such as Struck.

4.3 Pocket Tracker

The stereoboard is extremely limited in terms of processing power and memory capacity, as shown in section 3.2. Both Struck and KCF are too demanding to run onboard the stereo camera, simpler less expensive strategies have to be taken. Adaptive trackers, which require online training, tend to be expensive in the updating of the model, and for this reason we chose not to update our person model online and use a pre-trained detector. The Pocket Tracker applies a traditional tracking by detection approach based on HOG feature extraction.

4.3.1 Detector

To train our detector we make use of logistic regression, a binary classification algorithm, often used in machine learning problems. The function we are going to use to represent our hypothesis is:

$$h_\theta(x) = g(\theta^T x) \tag{4.1}$$

where x is an input example, theta are the parameters that describe our model and g is the sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

The sigmoid function outputs values between 0 and 1, and $h_\theta(\theta^T x)$ can be seen as the probability of x belonging to the positive class. By thresholding the output of Eq. 4.1 we can separate positive from negative predictions, a typical value is to consider a prediction positive for $h_\theta(x) \geq 0.5$ and negative otherwise.

The training of our logistic regression classifier consists in finding the parameters θ which minimise the cost function in Eq. 4.3 across all training examples.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1-y^{(i)}) \log\left(1-h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{4.3}$$

where m represents the number of training examples and n the number of features, $y^{(i)}$ is the ground truth label for training sample i and λ a regularization parameter to prevent overfitting. Usually used in logistic regression, this cost function penalises false positives and false negatives in the training dataset. For false positives we have $h_\theta(x^{(i)}) \geq 0.5$ and $y^{(i)} = 0$, making the term $(1-y^{(i)}) \log\left(1-h_\theta(x^{(i)})\right)$ large. A similar situation happens with false negatives, $h_\theta(x^{(i)}) \leq 0.5$ and $y^{(i)} = 1$, with $y^{(i)} \log\left(h_\theta(x^{(i)})\right)$ also becoming large. In the limit, for completely wrong predictions, i.e. $y^{(i)} = 0$ with $h_\theta(x^{(i)}) = 1$ or $y^{(i)} = 1$ with $h_\theta(x^{(i)}) = 0$, the cost function would go to infinity.

To find the set of parameters θ that produced the best results we used Matlab optimisation function $fminunc$.

As we are limited in both memory and speed, our detector is run for a fixed scale, a 28×24 pixel sized window was chosen, which at 2 meters away from the camera includes the head, shoulders and a small portion of background. The size of the area that is going to be used to search for our tracker is also an important factor influencing the speed of the algorithm. We could run our search in the entire picture, but that would make the tracker slower, without the need for it. The movement of our target, a human, should be mostly horizontal, a human head maintains a relatively constant altitude. To give emphasis to this motion constraint of the target, we chose a region of interest (ROI) of 96×56, used to search for the target.

Various HOG features configurations were tested and the results are presented in section 6.3, using 2×2 pixel cells without the use of blocks and 4 bins was the chosen configuration. This configuration produced some of the best results as well as being one of the fastest, it uses a small number of bins, reducing the number of computations, and as it does not use blocks there is no need for the extra step of calculating them. Running the stereoboard with this configuration gives us a frame rate of 1.68. Although this is a low frame rate, unlike adaptive trackers like KCF and Struck, this should not be a problem for our tracker as long as the target remains inside the region of interest.

4.3.2 Pipeline of the Tracker

Fig. 4.1 depicts the working steps of the tracker:

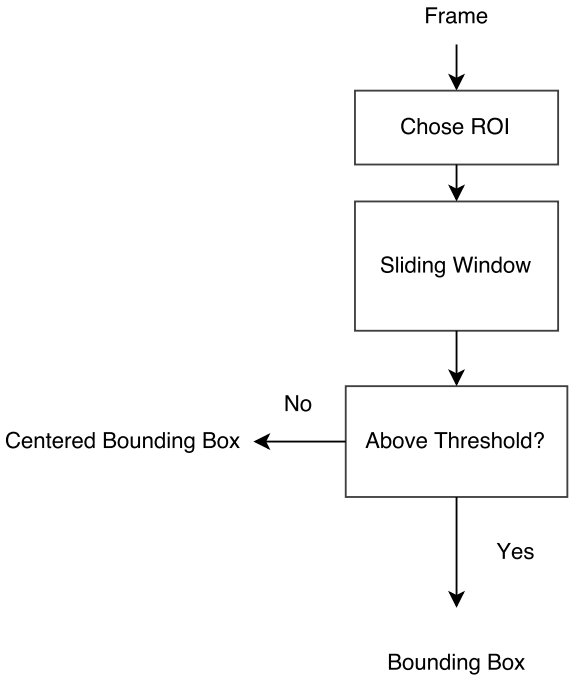


Figure 4.1: Pocket Tracker pipeline

In the first frame we don't know the location of our target, as we don't use the full image to run the detector, we start by selecting the ROI in the centre of the frame. Our aim is to find the patch, represented by its respective x features, inside the ROI that has the highest likelihood of being our target, meaning, the patch that produces the highest output to $h_\theta(x)$. The patch that produces the highest $h_\theta(x)$ is the same as the patch that produces the highest $\theta^T x$, as such we evaluate patches directly by this measure. The detector searches inside the ROI is using an extensive sliding window approach, a window of fixed size slides in the ROI, evaluating every possible location, while looking for the patch that produces the highest result. To prevent the detector from selecting patches that have low probability from being selected as the output of the detector's search, i.e. a patch that has a low $\theta^T x$ but still the highest in the image, which is especially problematic when no human is present in the scene, we introduce a detection threshold. Whenever a frame fails to produce a detection above this threshold, the system considers that no detection has occurred and enters its redetection mode. During this redetection mode the output of the tracker is a perfectly centred bounding box, as to result in the drone just hovering in place, while waiting for the next successful detection. Following a frame with a successful detection, the ROI is selected centered in the position of the previous detection, where the detector searches for a new target position.

Chapter 5

Control System Design

We could say that in a person following task, the perfect controller is the one that can maintain the person in the centre of the camera image, at a fixed distance from him or her. Most trackers, including the three used during this work, Struck [8], KCF [9] and the Pocket Tracker, output their location in the format of a bounding box as described in section 3.3. As such, the control system was built to be easily implemented with any tracker whose output is also a bounding box, depending essentially on the camera that is being used. Using the field of view and pixel resolution of the camera, it is possible to calculate angles in the image and use them in turn to calculate distances.

The implemented control scheme makes use of the control loops of paparazzi, by analysing the behaviour of the output bounding box, the position of the person relative to the drone is calculated and used to generate the references that are fed to the control loops. Yaw and Altitude control have a very similar approach while the control in the horizontal plane is slightly different.

5.1 Yaw and Altitude Control

The reference point is the centre of the frame, centring the person in that point can be divided into two control problems, yaw and altitude control. By dividing the field of view of the camera, *fov*, by the number of corresponding pixels, n_{pixels}, we get the approximate angle per pixel, let's call it $\text{angle}_{\text{step}}$:

$$\text{angle}_{\text{step}} = \frac{fov}{n_{\text{pixels}}} \tag{5.1}$$

When a detection is made, the centre of the bounding box has a horizontal and vertical pixel displacement, respectively h_{pixel} and v_{pixel}, from the centre of the frame, as can be seen in Fig. 5.1(a). These displacements can be used to calculate the associated horizontal and vertical angle displacements, h_{angle} and v_{angle}, which can be seen in Fig. 5.1(b) and 5.1(c), using the following equation:

$$\text{angle}_{\text{disp}} = \text{pixel}_{\text{disp}} \times \text{angle}_{\text{step}} \tag{5.2}$$

The yaw reference, h_{angle}, can be directly calculated using Eq. 5.2, in which $\text{pixel}_{\text{disp}}$ is h_{pixel}, v_{angle}

can be calculated in an analogous way.

To calculate the altitude reference, an extra information is needed, the distance between the target and the camera, d_{target}, which is calculated for the Horizontal control, and as such explained in section 5.2.Through the use of trigonometry the vertical distance displacement, v_{dist} is calculated with d_{target} and v_{angle}.

$$v_{\text{dist}} = d_{\text{target}} \times \tan(v_{\text{angle}}) \tag{5.3}$$

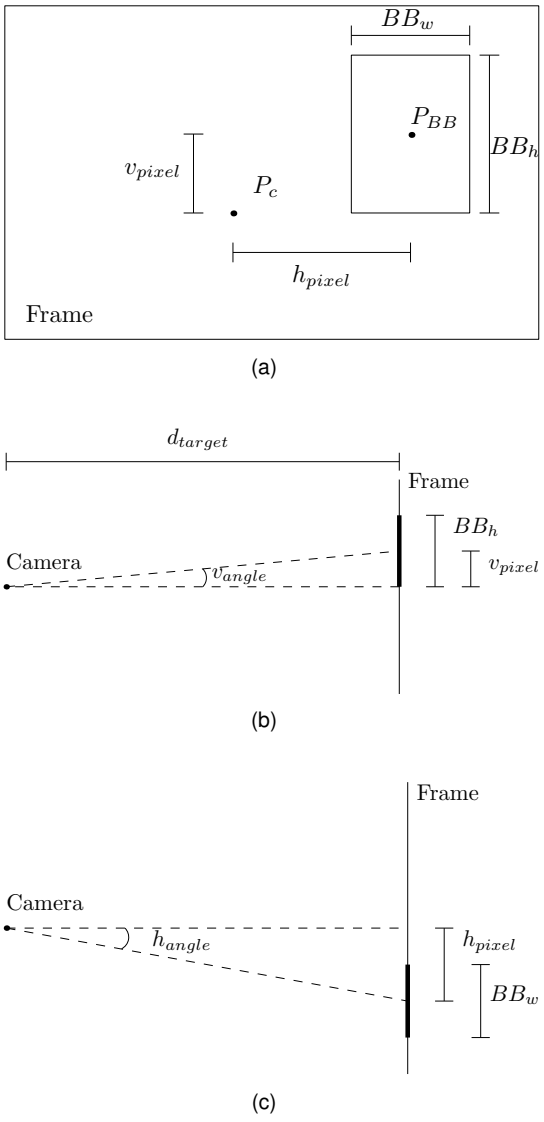


Figure 5.1: These three figures represent three views of a frame with a detection on it, the drawn bounding box, and the three dimensional position of the camera in space. From top to bottom we have the frontal view, then the side view and lastly the top view. The various parameters used to calculate the references for the control system are depicted in the figures. The bounding box height and width are respectively BB_h and BB_w, the centre of the bounding box is P_{BB} and the centre of the frame is P_c. The rest of the parameters are explained in sections 5.1 and 5.2.

5.2 Horizontal Control

5.2.1 Scale Adaptive Trackers

The horizontal control works on the position along the axis that is pointing towards the front of the aircraft, which is also the camera axis. References are generated for the aircraft to move forth and back in this direction, while trying to maintain a fixed distance from the person. The distance we want to keep from the person, d_{desired} is a parameter of the system and the error, d_{error} we want to minimize comes from the subtracting from this distance the calculated distance from the target:

$$d_{\text{error}} = d_{\text{desired}} - d_{\text{target}} \tag{5.4}$$

The tracker is initialised at a fixed distance and with a fixed box size, as such we know the real height, $\text{height}_{\text{real}}$, corresponding to the bounding box pixel height. If the person gets closer or further away from the camera, the pixel size of the bounding box changes, but the equivalent real size is kept constant, if tracking is successful of course. Through the use of Eq. 5.2 we can calculate the vertical angle associated with the pixel height of the bounding box, $\text{height}_{\text{angle}}$. Using $\text{height}_{\text{real}}$ and $\text{height}_{\text{angle}}$ we can calculate d_{target} with the following trigonometric relation:

$$d_{\text{target}} = \frac{\text{height}_{\text{real}}}{\tan(\text{height}_{\text{angle}})} \tag{5.5}$$

5.2.2 Fixed Scale Trackers

The detector used in the pocket tracker doesn't adapt to scale, as mentioned before, making the strategy used for the other trackers unfeasible. Working with a stereo camera gives us access to the disparity map corresponding to the images that we are using for tracking. With a pair of images from a stereo system, by matching the pixels in one of the images with the corresponding pixels in the other image, we can calculate the displacements between each pair of pixels, the image formed by all these displacements is the disparity map. The disparity map can in turn be used to calculate the depth map, which contains an estimate of the distance at which each pixel is from the camera. As it will be seen in the experimental results section 6.3, successful detections are very accurate, always with the head of the person it is tracking in the centre of the detection window. Matching pixels from the centre of the detection window with the corresponding pixels in the depth map and averaging over them gives us an estimate of the distance between the drone and the target.

5.3 Implementation Details

The order in which the reference values are calculated is important, the altitude calculations have to come after the horizontal distance calculation because they depend on the distance between the aircraft and the target. The initialisation of the system also needs to follow a few steps to ensure that the tracker is properly initialised on the target. The aircraft starts by taking off and stabilising its flight. The tracker

is then initialised by command, when the person to be tracked is correctly positioned in front of the drone. Upon initialisation, the control system is changed from hover mode to person following with the references calculated as explained in the previous sections.

Chapter 6

Results

In this section we present and discuss the results for the three visual trackers, as well as for the control scheme. Both the KCF and Struck visual trackers had C++ open source implementations by the respective authors, which were adapted to work with paparazzi on the Bebop. Due to the large field of view of the camera of the Bebop, only a crop of the full image was used, a 124×96 pixel image, with a field of view of 60 degrees. We first discuss the results for the two state-of-the-art trackers, then we present the experiments that led to the Pocket Tracker tuned parameters and the respective performance of the tracker associated with the different tracker configurations. Finally, the results to validate the reference generation are presented.

6.1 Struck

When run on a normal computer, Struck presents very good results, it easily handles changes in appearance, e.g. a person rotating, abrupt movements like quick changes in the direction of movement are also not a problem and adapts very smoothly the size of the bounding box to the size of the person. Though very promising, this performance comes at a cost, when faced with weaker hardware, the speed at which the tracker can run decreases, which results in deteriorated performance. Though it can still run fully onboard the bebop, the frame rate drops dramatically to 0.3 frames per second, becoming impossible to obtain any kind of tracking past the first frame.

Figure 6.1: Detection results for the KCF tracker onboard the bebop. The images were recovered from a wireless communication link, hence the low quality.

6.2 KCF

Similarly to Struck, KCF shows impressive results when tested at higher speeds. With its much faster framework, when compared to Struck, it is able to run at 3.5 frames per second onboard the Bebop. Tracking is achieved but the relative movement of the target with regard to the image plane that the tracker can handle becomes very limited. A few examples of successful tracking can be seen in Fig. 6.1. The capacity to adapt to scale also suffers from the low frame rate, performing very crudely in its adaption to objects coming closer or further away from the camera. For a slow moving object the tracker can handle the task, but for the normal speed of a person, it drifts aways from the target very quickly.

6.3 Pocket Tracker

In this section we present the results obtained for the detector of the Pocket Tracker, its main component, trained and tested for different HOG features configuration and for different regularisation parameters for the logistic regression training phase. The dataset used to train the detector was made using the stereocamera, ground truth positive samples were manually annotated and negative samples extracted from the background. We used 764 positive samples of a person with different orientation angles, i.e. samples from all 360 degrees of rotation, and 1000 negative samples extracted from the background. The test sequence used to evaluate the results of the various detector configurations was made of 167 images, with a section where tracking presented reasonable to excellent results depending on the configuration in place, and a section in which the pixel values of the background were very similar to those of the person and as such the performance of the detector deteriorated for all configurations. Examples of those two sections can be seen in Fig. 6.2. We used always the same training dataset and test sequence in order to ensure comparable results.

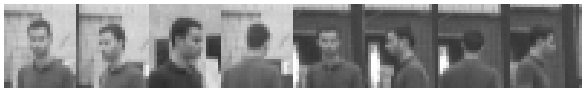


Figure 6.2: Detection results for the pocket tracker, the four images on the left correspond to positive detections by the Pocket Tracker, while the images on the right correspond to difficult situations where the pocket tracker failed to produce positive detections. The images were recovered from the stereo camera.

A first phase of tests consisted in running our test sequence for different block sizes and block overlap pairs, for a fixed number of HOG bins, across a wide range of λ (see Eq. 4.3) values. Then, for the λ that showed the best result in each of the three best configurations, similar tests were run, now for a fixed λ, block size and and block overlap, while varying the number of HOG bins.

To evaluate the performance of the detector we used three different metrics, the percentage of failed detections, the Object Tracking Performance (OTP)[31] and the Average Tracking Accuracy (ATA)[32]. For the first one we use the PASCAL criterion [33]:

$$\frac{\left|T^i \cap GT^i\right|}{\left|T^i \cup GT^i\right|} \geq 0.5 \tag{6.1}$$

where T^i and GT^i represent respectively the detected bounding box in frame i and the ground truth bounding box in frame i. Whenever the Pascal criterion is met, i.e. the intersection area of the two boxes divided by the union area is greater than 0.5, a detection is considered as successful, if bellow this threshold we consider it to be a failed detection. The OTP gives us a measure of how accurate the detector is for the cases where positive detections were achieved, discarding the frames where Eq. 6.1 is not met. OTP is given by:

$$OTP = \frac{1}{|M_s|} \sum_{i \in M_s} \frac{\left|T^i \cap GT^i\right|}{\left|T^i \cup GT^i\right|} \tag{6.2}$$

where M_s denotes that set of frames with positive detections. Finally ATA, very similar to OTP, calculates the average accuracy across all frames including the ones with true positives and the ones with false positives:

$$ATA = \frac{1}{N_{frames}} \sum_{i} \frac{\left|T^i \cap GT^i\right|}{\left|T^i \cup GT^i\right|} \tag{6.3}$$

The ATA is the metric that gives us the best evaluation on the overall performance of the detector, penalised by both the inaccuracy of the detector on positive detections and on failed detections.

6.3.1 Block Configuration

As the bounding box in our detector is of fixed size, not all HOG configurations fit well into the size of the box, as an example, if we had a 5×5 pixel window and used 2×2 pixel cells, we would not be able to fit an integer number of cells into such window.

A total number of 6 different configurations were tested for, each identified by a letter in the legend of Fig. 6.3(c). The correspondence between that letter and the configuration specifications can be found in Table 6.1:

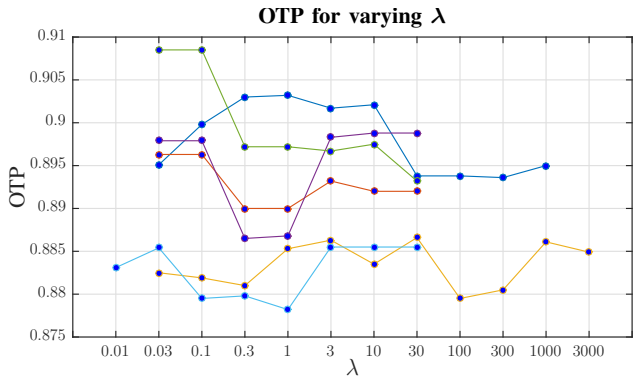
Table 6.1: HOG configurations.

Configuration	Block Size [cells]	Block Overlap [fraction of block]
A	1×1	0
B	2×2	0
C	2×2	1/2
D	3×3	2/3
E	4×4	3/4
F	4×4	1/2

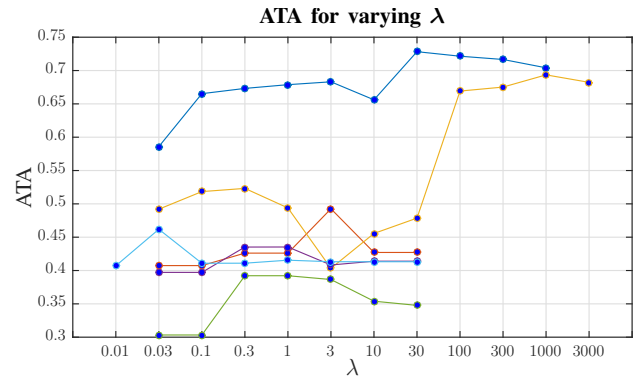
All configuration were tested for different regularisation values, ranging from 0.03 up to 30, in steps of roughly three times the previous value,and whenever necessary, lower and higher values were tested for, until convergence was achieved. In this testing phase we used 9 bins, which was the number of bins that gave the best equilibrium between speed and accuracy in the original work by Dalal and Trigs [21].

The OTP, Fig. 6.3(a), was very high for all tested configuration and regularisation values, with values always close 0.9, it reflects how well the output bounding box matches with the ground truth results for

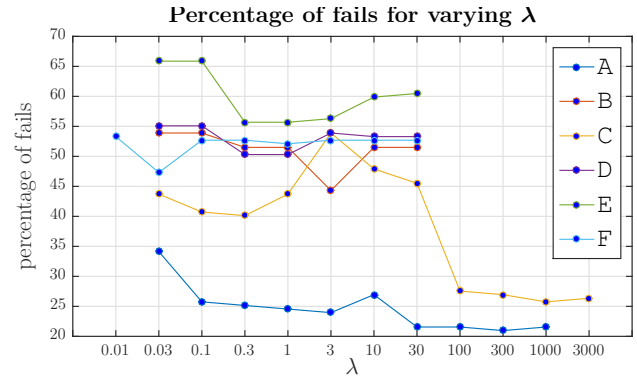
positive detections. It's the ATA, Fig. 6.3(b), and the percentage of fails, Fig. 6.3(c), that clearly shows us that two of the configurations have much better results than the others, respectively the 1×1 and 2×2 cells per block configuration without overlap. Block overlap turns out not to be beneficial for the detectors performance, in contrary to what was to be expected.



(a)



(b)

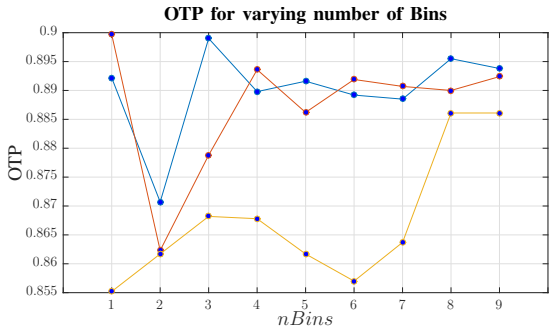


(c)

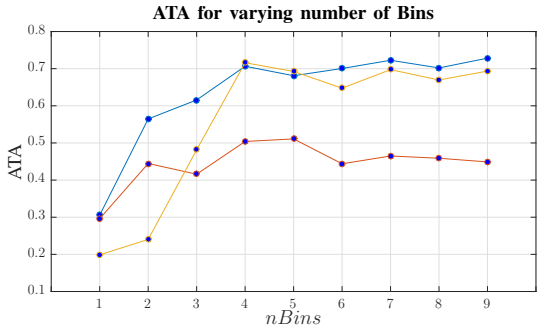
Figure 6.3: Performance results for the different HOG feature configuration, while varying λ (see Eq. 4.3), evaluated from top to bottom on OTP, ATA and percentage of fails respectively.

6.3.2 Bin configuration

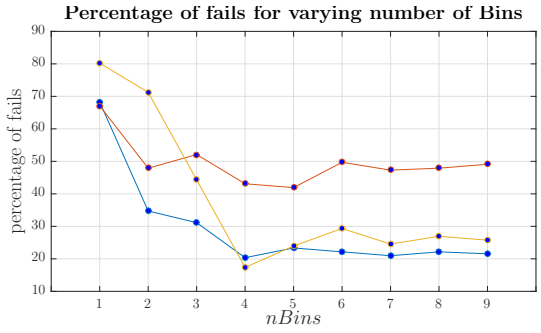
Configuration A, B and C were the ones that achieved the best results, even though A and C clearly surpass the performance of B and the others, we still tested the three of them for different numbers of bins, using the regularisation values that produced the best results within each configuration. Starting with one bin and going up to 9, the lower numbers already presented impressive results, even with 1 bin some correct detections were achieved, the intensity of the gradients alone gives us a hint at how good the HOG descriptor is. Increasing the number of bins quickly gets the results near the ones we get for 9 bins, with 4 bins sometimes getting better results than any other number. OTP, Fig. 6.4(a), is again very high to all tested configurations, while ATA, Fig. 6.4(b), and the percentage of fails, Fig. 6.4(c), gives very close results for the A and C configurations.



(a)



(b)



(c)

Figure 6.4: Performance results for fixed λ, while varying the number of bins, nbins, performed on the three best configurations from the 6.3, evaluated from top to bottom on OTP, ATA and percentage of fails respectively.

As 4 bins already produced very good results, and the difference in performance between using 1×1 and 2×2 cells per block is very small, we chose to use configuration A with 4 bins, both fast and with some of the most accurate results, also benefits from not needing to calculate blocks and the step between tested image patches is smaller, 1 cell instead of 2. The frame rate for configuration A, for varying number of bins is presented in Table 6.2.

Table 6.2: Frame rates with the number of bins for configuration A.

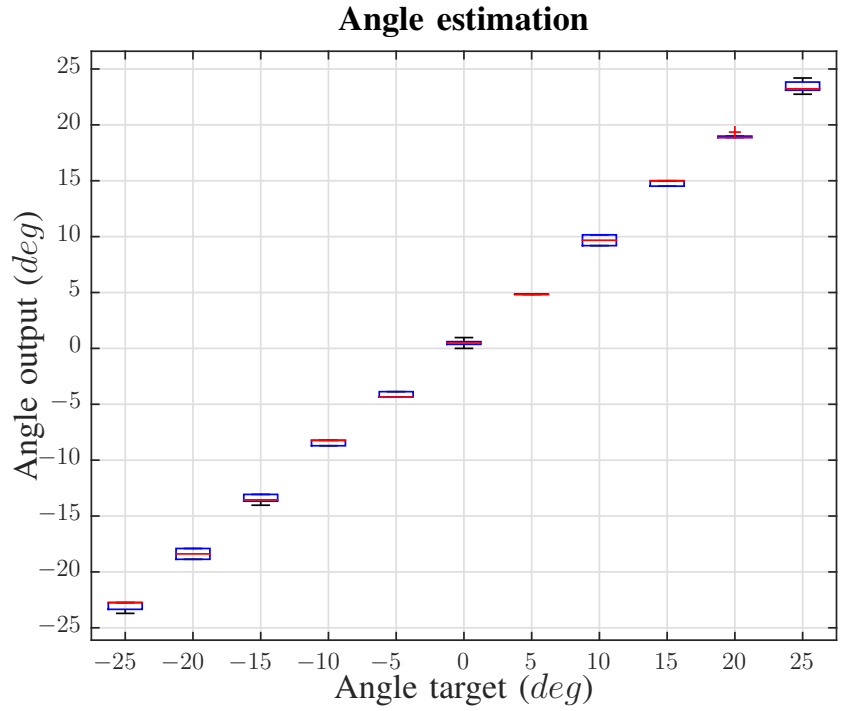
nBins	1	2	3	4	5	6	7	8	9
fps	2.56	2.09	1.89	1.68	1.49	1.39	1.27	1.20	1.15

6.4 Reference generation

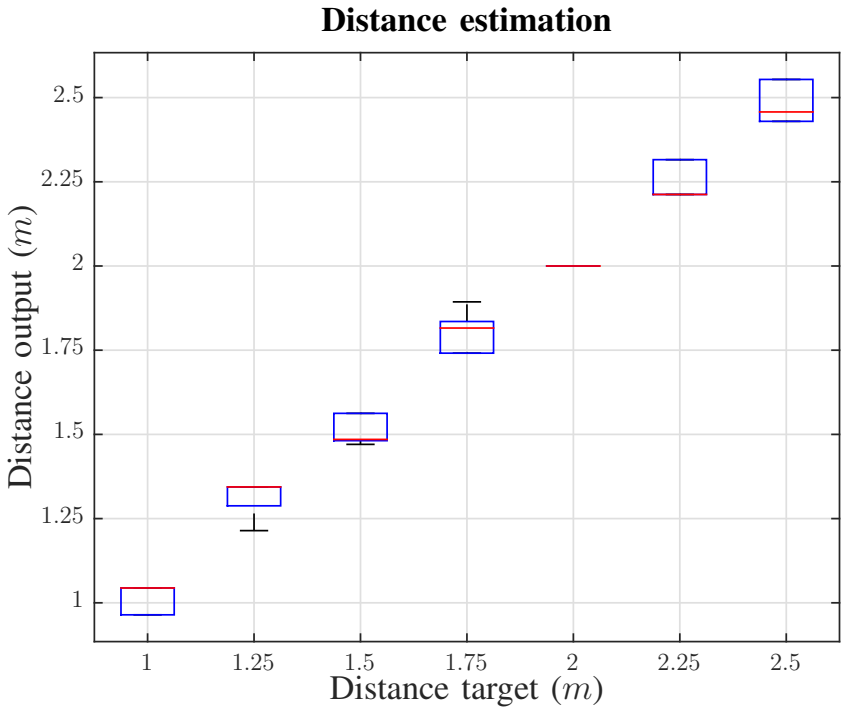
As we make use of the control loops of paparazzi, our evaluations go into the accuracy of the references we are generating. There are two main measurements influencing the references, angle measurements based on pixel displacements and distances based on the adaptive size of the bounding box.

To evaluate the angle measurements we compared for different angle positions, going for -25° to 25°, in 5° steps with 0° being the centre of the frame where the KFC tracker was initialised on the bebop. These measurements can be seen as direct results for yaw reference. For distance measurements the camera of a macbook air was used because of the aforementioned low frame rates when on board the drone, resulting in the tracker not adapting well to scale, which is the main factor influencing distance calculations. Initialised with the target at a distance of 2 meters from the camera we performed 0.25 meter steps from 1 meter to 2.5 meters away from the camera, for higher distances the results get too deviated due to the trackers performance, and as such we stopped the tests there, for closer distances the target gets too close to the camera.

For each experiment 5 repetitions were made, and the results plotted in box plots. In Fig. 6.5(a) we have the box plot for the yaw angle experiments, and in Fig. 6.5(b) for the distance calculation experiments. Both results are very positive, angle measurements present a very small variance, note that the results for the distance equal to 2 meters are perfect, this is because it is the distance at which the the tracker was initialised. The references can be concluded to be well calculated, depending mainly on the performance of the tracker that is generating the bounding box.



(a)



(b)

Figure 6.5: Accuracy tests on angle estimation, above, and distance estimation, below, calculated from the detected bounding box, compared against ground truth results.

Chapter 7

Conclusions

7.1 Achievements

In this paper, a vision based strategy for tracking and following a person onboard a MAV is presented, not needing external information on the location of the person, it retrieves an estimation of the position through the processing of the frames of the camera. We first tested the performance of two adaptive state-of-the-art trackers, KCF and Struck, onboard the Parrot Bebop drone. The overall result for both trackers tells us that these kinds of adaptive trackers work extremely well if they can run at normal frame rates, but with the low frames per second imposed by the hardware of the drone, makes tracking very difficult for a person moving at normal speed.

As our objective was to implement a person following system onboard a Pocket Drone, which has very limited hardware when compared with the Bebop, it was clear that a different strategy had to be implemented, we chose a tracking by detection framework, based on a logistic regression classifier, trained on HOG features. Our detector performs very well for scenes where the darkness of the background is different from the darkness of the person that is being tracked, but when there is very little difference between the intensity values of the pixels of the hair and those of the background, the detector has a lower performance when predicting the correct location of the person, as seen in Fig. 6.2.

7.2 Future Work

The Pocket Tracker shows great results for such a lightweight tracker, though some work still has to be done to improve its robustness, one of the main features that would increase the performance would be the introduction of a penalisation factor to give more emphasis to samples which are more close to the last detection, making it more robust to the presence of multiple people in the scene.

Tests should be performed on the detector for different template window sizes, and make a detailed evaluation into the accuracy of the depth map recovered from the stereocamera.

Extensive flight tests need to be performed in order to evaluate the performance of our strategy in a real environment.

Finally investigation should be done into more specialised control schemes for the person following problem, as it stands our strategy tries to make the distance between the drone and the person constant, with no regard for the flight path, other strategies, such as trying to follow the same path of the person, or maintaining a fixed heading while following could be interesting flight modes.

All in all, we are always faced with the same problem when working with tracking algorithms onboard drones, the computational capacities are limited, resulting in low frame rates, which inhibits the full performance of the trackers. With more powerful hardware, that should be available in the years to come, increasingly more powerful and robust methods should be able to run onboard drones.

Bibliography

[1] Lily drone. information can be found at: `https://www.lily.camera`.

[2] Airdog. information can be found at: `https://www.airdog.com`.

[3] Hexo+. information can be found at: `https://hexoplus.com`.

[4] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7): 1442–1468, 2014.

[5] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38 (4):13, 2006.

[6] M. Mueller and A. Drouin. Paparazzi—the free autopilot. build your own uav. In *24th Chaos Communication Congress, Berliner Congress Center, Dec*, pages 27–30, 2007.

[7] B. Remes, P. Esden-Tempski, F. Van Tienen, E. Smeur, C. De Wagter, and G. De Croon. Lisa-s 2.8 g autopilot for gps-based flight of mavs. In *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014*. Delft University of Technology, 2014.

[8] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270. IEEE, 2011.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(3):583–596, 2015.

[10] C.-T. Dang, H.-T. Pham, T.-B. Pham, and N.-V. Truong. Vision based ground object tracking using ar. drone quadrotor. In *Control, Automation and Information Sciences (ICCAIS), 2013 International Conference on*, pages 146–151. IEEE, 2013.

[11] C. Teuliere, L. Eck, and E. Marchand. Chasing a moving target from a flying uav. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4929–4934. IEEE, 2011.

[12] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[13] I. F. Mondragón, P. Campoy, M. A. Olivares-Mendez, and C. Martinez. 3d object following based on visual information for unmanned aerial vehicles. In *Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC)*, pages 1–7. IEEE, 2011.

[14] F. De Smedt, D. Hulens, and T. Goedemé. On-board real-time tracking of pedestrians on a uav. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2015.

[15] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[16] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[17] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[18] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[19] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli. Vision based gps-denied object tracking and following for unmanned aerial vehicles. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2013.

[20] R. Bart, A. Vykovsk, et al. Any object tracking and following by a flying drone. In *2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 35–41. IEEE, 2015.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[22] M. Danelljan, F. S. Khan, M. Felsberg, K. Granström, F. Heintz, P. Rudol, M. Wzorek, J. Kvarnström, and P. Doherty. A low-level active vision framework for collaborative unmanned aircraft systems. In *Workshop at the European Conference on Computer Vision*, pages 223–237. Springer, 2014.

[23] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[24] Y. Imamura, S. Okamoto, and J. H. Lee. Human tracking by a multi-rotor drone using hog features and linear svm on images captured by a monocular camera. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2016.

[25] K. Haag, S. Dotenco, and F. Gallwitz. Correlation filter based visual trackers for person pursuit using a low-cost quadrotor. In *Innovations for Community Services (I4CS), 2015 15th International Conference on*, pages 1–8. IEEE, 2015.

[26] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[27] H. Lim and S. N. Sinha. Monocular localization of a moving person onboard a quadrotor mav. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2182–2189. IEEE, 2015.

[28] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *European conference on computer vision*, pages 2–15. Springer, 2008.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484, 2005.

[30] R. Rifkin, G. Yeo, and T. Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

[31] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Transactions on Image Processing*, 21(10):4334–4348, 2012.

[32] B. Karasulu and S. Korukoglu. A software for performance evaluation and comparison of people detection and tracking methods in video processing. *Multimedia Tools and Applications*, 55(3): 677–723, 2011.

[33] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

Appendix A

HOG Features

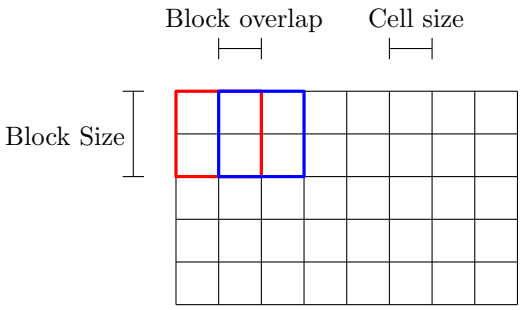


Figure A.1: HOG features representation in a n image, in the picture, each black square represents a cell, two blocks of 2×2 cells are drawn, red and blue, with a block overlap of two cells, the ones belonging to both blocks.

A.1 HOG Features

Histograms of Oriented Gradients is a feature descriptor introduced in 2005 by Dalal and Triggs[21], often used for object detection, as showed by the multiple works in literature that make use of this descriptor, including our work. The image that we want to describe is divided into small squared regions, the cells, and for each of these cells a histogram of gradient orientations is calculated, cells can be grouped in blocks. The HOG feature descriptor results from the concatenation of these cells. In the following sections a detailed explanation on how the HOG features are calculated is presented.

A.1.1 Gradient Calculation

The first step is to calculate the gradient values, in the x and y directions, for each pixel belonging to a cell. Those gradients are calculated using the kernel filters in Eq. A.1.

$$[-1, 0, 1] \qquad [-1, 0, 1]^T \tag{A.1}$$

A.1.2 Orientation binning

The histogram that we are calculating for each cell is the sum of each individual pixel histograms. It can be signed or unsigned, meaning that its range can go from 0 to 360 degrees or from 0 to 180 degrees respectively, and the number of bins determines how fine the histogram is, as an example, an unsigned histogram of 4 bins means that we have 180 degrees divided into four channels. The orientation of the gradient defines in which channels its contribution is going to be accounted for, and the magnitude of the gradient defines the weight of this contribution.

A.1.3 Blocks

The use of blocks in HOG features is also very common, characterised by block size and overlap, they usually result in an increased performance. The block size is the number of cells that form each block, a 2×2 block is a block with 4 cells in it, and the block overlap allows for each cell to contribute to more than one block. A representation of the block size and overlap can be seen in Fig. A.1. Blocks can be locally normalised, to account for illumination changes, also adding to better performances, in the original work four different normalisation factors were tested and all showed good improvements in performance. The full descriptor would then result from the concatenation of these blocks.