An Attentional Architecture for CNN-Based Visual Detection and Tracking

Tiago Filipe Figueira Cerveira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Alexandre José Malheiro Bernardino
Prof. Jorge dos Santos Salvador Marques

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Alexandre José Malheiro Bernardino
Members of the Committee: Prof. Rodrigo Martins de Matos Ventura

November 2017
Acknowledgments

It is with great delight that I write this Section to acknowledge all people that have assisted me in any significant way so that this final year thesis could be a success.

I wish to acknowledge the help provided by Alexandre Bernardino, my supervisor, for his constructive suggestions and valuable input given in the planning and development stages of this work. I would also like to thank all the time he has generously given to provide useful critiques. This effort did not go unnoticed.

The help provided by my second supervisor, Jorge Marques, is also acknowledged as he always made the time to meet with me to discuss various details of this work.

I am specially grateful for the assistance given by Ricardo Ferreira across all stages of this thesis. More particularly, I would like to thank his patient guidance, keen encouragement and insightful suggestions on this research work. I could not fail to remember all afternoons spent guiding me through the building blocks of this work as well as his very watchful final remarks on this document.

Assistance provided by Gonçalo Cruz and Jorge Matos was greatly appreciated in using the detector system and fully understanding implementation details of the correlation computations, respectively. As this work extends theirs in some ways, I would like to thank both for quickly introducing me to the subject.

I would like to acknowledge the support and encouragement provided by my family and close friends during the preparation of my final year thesis.

Finally, I would like to thank the computer and robot vision group (VISLAB) at the Institute for Systems and Robotics (ISR/IST Lisboa) and the Portuguese Air Force Research Center for their assistance with the collection of the dataset used in this work and their help in offering all the necessary resources in the laboratory.
Abstract

In this work an end-to-end system for the task of visual detection and tracking of vessels in the maritime setting is presented. Such setting presents challenging difficulties such as sun reflections hitting the recording equipment, wake patterns caused by the vessel’s motion, and fast motion of the subjects of interest. In order to handle this kind of challenges, additional considerations for robustness need to be made so state of the art performance is achieved. The tracking system uses correlation filtering with image features derived from a CNN, and the detector uses another CNN for the bounding box regression task. The proposed system uses an attention model to better manage computational resources and combine two tasks, tracking, and detection, that have different performance requirements, so they work in synchrony and achieve real-time performance. The proposed system is evaluated with a benchmark and the results discussed. Results show that using a attention model based on a Bayesian filter has clear advantages against the remaining alternatives, specially in settings with a high density of maritime vessels. The dataset used to perform the evaluations on real sequences was obtained under the SEAGULL project.

Keywords

(Attention Model; CNN; Correlation; KCF; Neural Network)
Resumo

Neste trabalho apresenta-se um sistema completo para a tarefa de detecção e seguimento visuais de embarcações no cenário marítimo. Este tipo de cenário apresenta dificuldades particulares que requerem atenção adicional, como a reflexão do sol atingir o equipamento de gravação, o padrão de arrasto das embarcações causado pelo seu movimento, e o movimento rápido dos objetos de interesse. Por forma a lidar com este tipo de desafios, considerações adicionais são necessárias. Estas considerações aumentam a robustez do sistema final e permitem atingir desempenho de estado da arte. O sistema de seguimento usa um filtro de correlação com features visuais extraídas a partir de uma CNN e o detetor usa outra CNN para a tarefa de indicar a localização de cada embarcação marítima. O sistema proposto utiliza um modelo de atenção de forma a gerir melhor os recursos computacionais e combinar as duas tarefas, detecção e seguimento, que tem requisitos computacionais distintos, de maneira a que seja possível trabalharem em sincronia e atinjam desempenho de tempo real. O sistema proposto é avaliado numa suite de testes própria e os resultados obtidos são discutidos. Os resultados mostram que utilizar um modelo de atenção baseado num filtro Bayesiano traz vantagens relativamente às abordagens a modelos de atenção restantes, especialmente nos cenários com uma grande densidade de embarcações marítimas. O conjunto de dados utilizado nas avaliações em sequências reais foram obtidas no âmbito do projeto SEAGULL.

Palavras Chave

(CNN; Correlação; KCF; Modelo de Atenção; Rede Neuronal)
## Contents

1 Introduction 1
   1.1 Challenges ....................................................... 4
   1.2 Goals ............................................................ 4
   1.3 Contributions .................................................. 5
   1.4 Outline .......................................................... 6

2 Related Work 7
   2.1 Image Detection Approaches .................................... 9
      2.1.1 Feature-Based Object Detection .......................... 9
      2.1.2 Template-based Object Detection ......................... 10
   2.2 Tracking Based on Deep Learning Techniques .............. 10
   2.3 Detection and Tracking in the Maritime Setting .......... 11

3 Overall Architecture 13

4 Technical Background 17
   4.1 Correlation Filter ............................................... 19
   4.2 Image Features .................................................. 21
      4.2.1 Introduction to Neural Networks ......................... 21
      4.2.2 Convolutional Neural Networks .......................... 25
         4.2.2.A CNN Architecture Considerations .................. 25
         4.2.2.B Types of layers on a CNN ........................... 26
         4.2.2.C Googlenet Network .................................. 27

5 Used Methods 31
   5.1 Kernelized Correlation Filter ................................. 33
      5.1.1 Applying the Kernel Trick ............................... 33
      5.1.2 Preprocessing of Input Images ........................... 35
      5.1.3 Filter Training ............................................. 35
      5.1.4 Re-detection ................................................. 36
   5.2 Transfer Learning with CNNs ................................ 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Optical Flow</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>Spatial Attention Model</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>Discretization of Space</td>
<td>41</td>
</tr>
<tr>
<td>6.2</td>
<td>Attention Model Strategies</td>
<td>42</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Random Sampling Attention Model</td>
<td>42</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Greedy Attention Model with a Bayesian Filter</td>
<td>43</td>
</tr>
<tr>
<td>6.2.2.A</td>
<td>Encoding Optical Flow in motion model</td>
<td>44</td>
</tr>
<tr>
<td>6.2.2.B</td>
<td>Performing the Measurement Update</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation</td>
<td>47</td>
</tr>
<tr>
<td>7.1</td>
<td>Defining the Cost Function</td>
<td>49</td>
</tr>
<tr>
<td>7.2</td>
<td>Results on Real Data Sequences</td>
<td>50</td>
</tr>
<tr>
<td>7.3</td>
<td>Simulator Characterization</td>
<td>51</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Characterize Complete System</td>
<td>53</td>
</tr>
<tr>
<td>7.3.1.A</td>
<td>Detector System</td>
<td>53</td>
</tr>
<tr>
<td>7.3.1.B</td>
<td>Tracker System</td>
<td>53</td>
</tr>
<tr>
<td>7.4</td>
<td>Results on Synthetic Data</td>
<td>54</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Performance With Changing MV Density</td>
<td>54</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Results With Changing UAV Speed</td>
<td>56</td>
</tr>
<tr>
<td>8</td>
<td>Conclusion</td>
<td>59</td>
</tr>
<tr>
<td>8.1</td>
<td>Conclusions</td>
<td>61</td>
</tr>
<tr>
<td>8.2</td>
<td>Future Work</td>
<td>61</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 UAV used to capture data for the SEAGULL project equipped with optical sensors. ........ 4

2.1 Wake pattern generated by a small boat. ......................................................... 12

3.1 Diagram of the final system’s building blocks and main interfaces. ......................... 15

3.2 Representation of blocks the simulator replaces. .............................................. 16

4.1 Graphical representation of a 2D Gaussian signal with zero mean and unit variance on a 20x20 sample image. ................................................................. 19

4.2 Example representation of Rosenblatt’s perceptron. ........................................... 22

4.3 Example of two different activation functions. (a) represents a sigmoid activation, \( y = \frac{1}{1+e^{-x}} \) and (b) represents a ReLU activation, \( y = \max(0,x) \). ................. 22

4.4 Example of a MLP with 2 hidden layers. Each circle represents a unit of the type indicated in Figure 4.2. ................................................................. 23

4.5 Example of a simple CNN architecture. ........................................................... 26

4.6 Architecture of Google’s Inception network, version 3. ..................................... 28

4.7 Naive version of the inception module. ............................................................. 28

4.8 Full version of the inception module. ............................................................... 29

5.1 Example of explicitly applying a kernel function. .............................................. 34

5.2 Graphical representation of a 2D Hanning window on a 30x30 sample image. ........ 36

6.1 Discretization of space on a frame with FULL HD resolution. Each blue square represents an AB. ................................................................. 42

6.2 Example of the computed movement of an AB. .............................................. 45

6.3 Graphical representation of the binary entropy function. ................................... 45

7.1 Results of the cost function on the Random Sampling Attention Model for real data sequences. ................................................................. 51
7.2 Results of the cost function on the Greedy Bayesian Attention Model for real data sequences. .................................................. 52

7.3 Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.1252$. ........................................... 54

7.4 Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.0731$. ........................................... 55

7.5 Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.0301$. ........................................... 55

7.6 Results of the cost function on the Greedy Bayesian Attention Model with $P(MV) = 0.0731$ and high speed. .................................................. 57

7.7 Results of the cost function on the Random Sampling Attention Model with $P(MV) = 0.0731$ and high speed. .................................................. 57
# List of Algorithms

<table>
<thead>
<tr>
<th>Section</th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Stochastic Gradient Descent</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Training KCF Tracker</td>
<td>36</td>
</tr>
<tr>
<td>5.2</td>
<td>Detection Step</td>
<td>37</td>
</tr>
<tr>
<td>6.1</td>
<td>Grid Localization</td>
<td>44</td>
</tr>
</tbody>
</table>
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Attention Box</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>FAST</td>
<td>Features from Accelerated Segment Test</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>FPS</td>
<td>Frames per Second</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>GD</td>
<td>Gradient Descent</td>
</tr>
<tr>
<td>HoG</td>
<td>Histogram of Gradients</td>
</tr>
<tr>
<td>IR</td>
<td>Infra-Red</td>
</tr>
<tr>
<td>KCF</td>
<td>Kernelized Correlation Filter</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi Layer Perceptron</td>
</tr>
<tr>
<td>MOT</td>
<td>Multiple Object Tracking</td>
</tr>
<tr>
<td>MSER</td>
<td>Maximally Stable Extremal Regions</td>
</tr>
<tr>
<td>MTT</td>
<td>Multiple Target Tracking</td>
</tr>
<tr>
<td>MV</td>
<td>Maritime Vessels</td>
</tr>
<tr>
<td>OF</td>
<td>Optical Flow</td>
</tr>
<tr>
<td>Short Form</td>
<td>Full Form</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>OTB</td>
<td>Object Tracking Benchmark</td>
</tr>
<tr>
<td>PSR</td>
<td>Peak to Side-lobe Ratio</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RNN</td>
<td>Recursive Neural Network</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, and Blue</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SIFT</td>
<td>Scale-Invariant Feature Transform</td>
</tr>
<tr>
<td>SURF</td>
<td>Speeded Up Robust Features</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>VOT</td>
<td>Visual Object Tracking</td>
</tr>
<tr>
<td>WITAS</td>
<td>Wallenberg Laboratory for Information Technology and Autonomous Systems</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Challenges ......................................................... 4
1.2 Goals ............................................................... 4
1.3 Contributions ................................................... 5
1.4 Outline ............................................................. 6
Earth’s surface is around 70% covered with water and, as most global trade is done using maritime routes, there are strong economical and social motivations to ensure the safety of people, ecosystems, and goods that use this environment. As Portugal has one of the vastest exclusive economic zones in the world it also has a great interest in systems that can provide the aforementioned safety.

This maritime setting must be secured so illegal activities can be prevented, such as drug smuggling, and pirate attacks, as well as for external border protection. There are systems already deployed that tackle the issue of managing the movement of people, goods, services, and capital, these systems work under the FRONTEX agency in the European Union. Such systems rely on costly equipment and need to be closely monitored by a trained human workforce, which is scarce. The recent migrant crisis in Europe exposed such difficulties.

Under the SEAGULL project [1], a maritime surveillance system was developed using Unmanned Aerial Vehicle (UAV)s, such as the one found in Figure 1.1, equipped with cheap optical sensors, such as Go Pro action cameras. This system tackles some of the issues present in other already deployed approaches because it is affordable, easy to deploy and with few infrastructure requirements. In this project, a fleet of fixed wing UAVs are equipped with on-board computers capable of automatic detection and tracking of Maritime Vessels (MV), whose location is relayed to a coastal ground station. The developed approaches still face some challenges such as sun reflections, breaking waves and boat wakes.

This work focuses on detection and tracking of MV in videos, captured in a maritime setting. This task involves verifying the presence of a specific object in an image sequence and locating it so the tracking step can initialize. Object tracking is the task where an object’s spatial and temporal changes are monitored during a video sequence, including its presence, position, size, and shape. This is done by solving a temporal correspondence problem, the problem of matching the target region in successive frames of a sequence of images taken at closely-spaced time intervals. These two processes, detection and tracking, are closely related [2, 3] because tracking usually starts with detecting objects, while detecting an object repeatedly in subsequent images is often necessary to verify tracking, as this helps adjust the confidence in the tracking step.

Multiple video sequences were captured during this project to create a dataset for research in the field of maritime surveillance. This dataset is later used for benchmarking the results presented in this document.

The following document works towards a more robust system to autonomously detect and track maritime vessels working on challenging settings.

\[1\text{http://labs.thebureauinvestigates.com/is-frontex-bordering-on-chaos/}\]
1.1 Challenges

In computer vision, more specifically related with tasks on object tracking, the main performance challenges are illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, background clutters, and low resolution. Those are all the distinct settings evaluated with the Object Tracking Benchmark (OTB) [4], a widely used benchmark framework for single object tracking.

Airborne maritime images also face all of these challenges to some extent. The most noticeable are illumination variation, that appears as sun reflections on the ocean hit the recording equipment, scale variation, as some video sequences are recorded with very high Field of View (FOV) cameras, so MV change size very noticeably from the moment they enter the frame, from one of the borders on the image, until they reach the middle of the frame, in-plane-rotation, as the UAV travels around the MV, and fast motion, mostly caused by the navigation of the UAV that has the recording equipment attached to it.

The performance requirements the system resulting from this work will be considered since as many computer vision systems fail to deliver real-time performance, so focus will also be directed at this subject.

1.2 Goals

The goal for this work is to further leverage the use of a Kernelized Correlation Filter (KCF) in the tracking step of this system, as described in [5] and reviewed in Chapter 5. The visual features delivered into this filter are based on a specific Convolutional Neural Network (CNN), also explained in Chapter 5. The details on how this kind of system works for any general purpose CNN is presented in Chapter 4 and the decision for this specific CNN design is made clear in Chapter 5.
The final system is then combined with a learnt neural network system used for autonomously detecting new MV and for adjusting the belief on already known ones, described in [6].

Both systems will be integrated into a single coherent new one that will use one of the proposed attention models as described in Chapter 6 to efficiently allocate resources in order to keep computing requirements low, so real-time performance can be achieved. Multiple approaches at an attention model will be considered and thoroughly tested so the resulting system can provide end to end autonomous tracking of MV.

This new framework should also be robust to difficult settings that tracking on maritime environments will include, from which one can highlight illumination variation, scale variation, in-plane rotation, and fast motion.

### 1.3 Contributions

Key contributions of this thesis are presented below.

1. The KCF tracker hinges on the work in [5] with the implementation based in [7], in which improvements were introduced in the correlation computation, which greatly improved the computation time for each frame, resulting in higher Frames per Second (FPS). A tracking failure measure was also added, known as Peak to Side-lobe Ratio (PSR).

2. All image features fed into the KCF tracker are produced from a different Artificial Neural Network (ANN) from the one used in [7], the VGG-Net [8]. Now, the tracker will use GoogleNet’s Inception V3 [9] network on the TensorFlow [10] framework, that resulted in comparable performance concerning time and accuracy, compared to the VGG-Net [8], but lowered memory requirements, an important detail given the limited amount of Graphics Processing Unit (GPU) memory available on the UAV, the location the ANN will be loaded onto.

3. The detector described in [6] used in this work was tested, its results were validated and effort was put into integrating it with other components of this work, establishing new communication protocols.

4. The attention model component is novel and key in this work as two considerably different networks, with different computing requirements are combined, so they work in synchrony. The tracker is substantially faster than the detector, for reasons discussed in Chapter 6, and evaluation will be performed in two different approaches that aim at maintaining consistency across the execution.

5. A system for simulating the tracker, detector, and MV behavior was developed so different attention model approaches could be tested in a more controlled environment. The biggest asset this novel
simulator provides is that it augments video sequences so a more comprehensive study on these approaches as well as the complete system’s overall performance can be obtained.

1.4 Outline

This document is organized as follows:

• Chapter 2 considers the state of the art in the tasks of tracking and image detection. This Chapter covers some of the most popular approaches in these tasks in association with research highlights relevant to this work.

• Chapter 3 describes all core components of the final system, as building blocks. It familiarizes the reader with the nomenclature, what each block is called, and also the way they interface with one another.

• Chapter 4 covers some background on technical details that will be useful when the actual methods used in this work are introduced.

• Chapter 5 covers the theoretical aspects of all methods directly used in this work. It is explained how to use a novel approach at correlation tracking for a single target, how to re-purpose a CNN developed for the task of image classification into a visual feature extractor, and details for the used approach for optical flow computation are provided.

• Chapter 6 proposes combining the two systems, the detector and the tracker into a single coherent one, using an attention model.

• Chapter 7 defines a cost function to benchmark the results obtained for real, and synthetic data sequences. The final system is then evaluated against this benchmark and its results are discussed.

• Chapter 8 presents final remarks on this work and discusses ways in which it can be extended.
2

Related Work

Contents

2.1 Image Detection Approaches ........................................ 9
2.2 Tracking Based on Deep Learning Techniques .................... 10
2.3 Detection and Tracking in the Maritime Setting .................. 11
Traditionally, Multiple Target Tracking (MTT) algorithms have been tailored for scenarios with multiple remote objects that are far away from the sensor and from each other. MTT based on small objects is a highly complex problem due to sensor noise, missed detections, clutter detections, the abrupt appearance of objects of interest in the frame, severe object occlusions, and an unknown and time-varying number of targets.

Situations such as people [11], players in sports [12], smart rooms [13], and, in this case, MV are some examples of settings where MTT algorithms have great relevance. In recent years, the increasing use of ANNs in the field of object tracking specifically, has allowed for the advance of performance in dealing with such challenges.

Object detection and tracking remains a very active research problem to this day, as every year the boundaries on performance are taken further. An example of this effort is seen in the Visual Object Tracking (VOT) Challenge [14], focused on evaluating single object tracking approaches, and the Multiple Object Tracking (MOT) Challenge [15], focused on evaluating multiple object tracking approaches, both held on an yearly basis. The challenge for creating a robust, accurate, and high performance approach still remains. The difficulty increases as the number of visual features required to observe also increase.

Most challenges arise from the image variability of video because of the motion of objects in frame, whether because the object itself is moving or the recording equipment is. As an object moves through the field of view of a camera, the images of the object change dramatically. The specific difficulties are recognized, and separately evaluated on the OTB benchmark, for instance.

\section{Image Detection Approaches}

The following Subsections cover some of the most popular approaches for the task of image detection in association with research highlights relevant to this work.

\subsection{Feature-Based Object Detection}

In feature-based object detection, it is important to find invariant image features. Such features are recognizable in consecutive frames even when there is significant change in illumination, scale, rotation or translation. The goal is to model objects of interest based on these extracted features rather than in raw pixels. This concept is key in understanding how image is typically processed. First, start with the most basic information on an image, the raw pixels, and try to extract more meaningful information out of it, the features.

Detection using this approach usually comprehends two steps. As a first step, the specific features are computed in two or more consecutive frames. The feature extraction will both reduce the amount of information to be processed, and provide a higher level of understanding of the scene. The second step
matches features between frames. Some of the most widely used visual features are Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Maximally Stable Extremal Regions (MSER), Histogram of Gradients (HoG), and, more recently, CNN features.

This two step approach is often used in optical flow computation. The work presented in [16] finds image motion using a edge detector, and the image motion is found at these edges. The constraints in this work make the assumption that the image flow is smoothly varying across the image.

2.1.2 Template-based Object Detection

Another popular approach in object tracking is known as template-based object detection. A template is an image patch that contains the object of interest, the object to detect. This approach can only be instituted when such template is available. In this setting, one can understand the process of object detection as matching features from the template to the image sequence.

One of the techniques used for detection based on templates is image subtraction. In this sense, a candidate position for the template location is found by minimizing the distance function between it and various positions in the image. This is a rudimentary approach that only performs well in restricted environments where imaging conditions, such as image intensity and viewing angles remain the same. An example where this approach is used is in the astro-photography setting, with the goal to search for asteroids \(^1\). In this work this technique was not readily applicable.

As an improvement to the image subtraction template-based detection, another technique uses correlation. Matching by correlation uses the position of the correlation peak between a template and an image to locate the best match. This technique is generally robust to noise and illumination effects in the images, and by, applying the kernel trick described in [5], the high computational complexity caused by summations over the entire template is greatly mitigated. The template can be updated for each new frame so the system can adapt to changes in the appearance model of the object being tracked. The downside of this approach is that, with time, the template is being updated with noise and the tracking result starts to drift from the target. Often times, a ground truth adjustment, provided by another system, is desirable.

2.2 Tracking Based on Deep Learning Techniques

The following Section overviews some research highlights on visual object tracking approaches that use ANNs.

\(^1\)http://brucegary.net/A86279/ISfor6128.html
MDNET [17] proposes a tracking algorithm based on a CNN trained in a multi-domain learning framework. This tracking algorithm learns domain independent representations from pretraining, and captures domain-specific information through online learning during tracking. The entire network is pretrained offline, and the fully connected layers including a single domain-specific layer are fine-tuned online.

The online update is conducted to model long-term and short-term appearance variations of a target in order to increase how robust and adaptive the system is and a hard negative mining technique is incorporated in the learning procedure to decrease false positive occurrence, as introduced in [18]. In this article, in order to avoid possible misclassification of faces, the authors obtain a comprehensive sample of face-like patterns and explicitly model their distribution. This approach won the 2015 VOT Challenge.

The work in [19] uses a tracking by detection approach that makes use of a Recursive Neural Network (RNN) that is applied on the source data over a temporal window. In addition to the appearance model using the mentioned RNN, this approach also implements a motion model and an interaction model, the latter takes into account not only the objects motion but also the behavior of nearby objects. The final system is then trained and tested in the MOT framework yielding an improvement in state of the art performance. This method runs on-line without the need to see future frames, making it a causal system.

### 2.3 Detection and Tracking in the Maritime Setting

Despite all the ground covered on the subject of visual detection and tracking, not a lot of approaches address the maritime setting, more specifically, the detection and tracking of MV. There are challenges in processing image in such conditions that are particular to this setting.

On the side of the MV, the waves and wakes caused by its motion create background clutter in an otherwise texture poor environment as seen in Figure 2.1. The sea also reflects the sun lighting and it is common for this reflection to hit the camera, causing very high variation in illumination. Multiple approaches on detection and tracking were also developed to be used on a motionless camera, in this setting the camera is attached to an UAV, and so arises the issue of the fast motion of the image sequence, caused by the UAV navigation, as well as in plane rotation as the UAV trajectory is such that it circumvents a particular MV.

As the goal of this work is to have an end-to-end system for detection and tracking of MV performing on on-board hardware of the UAV, there are additional concerns that take into account the resource limitations, both on computation time and memory consumption, so real time performance can be achieved.

With this in mind, the work presented in [20] relies on simple color blob analysis. The idea is that

---

2Edmont - Own work, CC BY-SA 3.0
the presence of a vessel in the ocean modifies the image texture and color, suggesting that texture and color features may provide useful cues to discriminate the vessel from the ocean surface, one of the assumptions being that the vessels have bright pixels represented by high Red, Green, and Blue (RGB) values. This system is capable of real-time operation on the UAV.

The work presented in [21] also addresses tracking of MV and uses an Extended Kalman Filter (EKF) for the estimation of the MV states and for the prediction of vessel trajectories, and uses an ANN for the detection step of the system.

Analyzing the current state of the art brings the conclusion that top performing systems often come at the expense of higher computational cost and requirements, as seen in the results published under the VOT Challenge for 2016 [14]. On longer sequences, the tracking tends to drift, and approaches based on key-points fail due to the typical low resolution and lack of texture of the target.
3

Overall Architecture
This Chapter will briefly describe all core components of the final system, as building blocks. The goal for this Section is to familiarize the reader with the nomenclature, what each block is called, and also the way they interface with one another. For a visual representation of these building blocks, refer to Figure 3.1.

The image stream is used as input for both the detector and tracker systems. These systems interface with the attention model that acts as a handler for how they should process the image stream information in the most efficient way. The final result, the location of every MV in the scene, is then acquired as the output.

**Figure 3.1:** Diagram of the final system’s building blocks and main interfaces.

1. Detector

   Is the system that first detects new or known MV in the scene. The detection step, if successful, will then start an instance of the tracker that will follow the target across the frame.

   This block runs in a separate process from the rest of the system, it receives the area where detection will be performed from the process running the attention model and, in case of a new detection, will communicate with a handler that will launch a tracker in the appropriate location.

   The final system will have a single instance of this MV detector.

2. Tracker

   Running independently from the detector one can find the trackers. After the initial bounding box where an MV is located is produced by the detector, its job is to follow the designated target in the frame, confirming its presence in following frames.

   The finished system deploys an instance of a tracker for each detected MV. This means that each tracker running at any given point is responsible for one and only one MV.

3. Attention Model
The final system will have a single instance of this block, the attention model. This block runs on a separate process from every other block and is responsible for choosing the area where the detector will perform detection.

This process directly communicates with the detector at every frame of the sequence to relay the necessary information.

4. Maritime Vessel and UAV Simulators

The relevance of this block is specially leading since adequately annotated real video sequences are scarce. In order to augment the dataset with different sequences from the real data already acquired, a separate system was designed specifically to better test attention model approaches. In this new setting, longer sequences can be defined, the speed of the UAV can be changed, the disposition of MV can be set in different ways. This simulator provides more granular control over the environment where the final system will perform and give more insight on how this approach at a attention model is expected to perform in more scenarios than the ones in the real sequences acquired.

When dealing with real data this block is absent, its purpose is only to generate new environments for the full system to perform on. In order to better understand what building blocks this simulator aims at replicating, refer to Figure 3.2. All blocks inside the green area can be simulated.
4

Technical Background

Contents

4.1 Correlation Filter ................................................................. 19
4.2 Image Features ................................................................. 21
The following Chapter covers some background on technical details that will be useful when the actual methods used in this work are introduced. Chapter 5 will then build on top of the subjects covered in this Chapter discussing all important methods that are used directly later.

## 4.1 Correlation Filter

A correlation filter is a template, or cropped image, that is correlated with the frames in the video sequence. This template is also referred to as simply the filter. The correlation of this template with the next frame in the sequence produces a response map, and the goal is to find the location of the maximum of this response. This represents the target location in the next frame.

The response map, $y$, is computed as follows:

$$ y(n) = x(n) \ast h(-n) $$

In Equation 4.1, $x$ refers to the new frame, $h$ refers to the filter, the $\ast$ operator refers to the convolution operation, and $n$ refers to a pixel. When referring to a 2D image one can vectorize it and apply the expression as mentioned.

In order to find a suitable filter for the task ahead, one must define their ideal response map. It is usual to define the output as a Gaussian signal with small variance, defined in two dimensions. One example of such response map can be found in Figure 4.1.

![Gaussian Function](image)

**Figure 4.1:** Graphical representation of a 2D Gaussian signal with zero mean and unit variance on a 20x20 sample image.

Taking the Fourier transform on both sides of Equation 4.1 and applying the convolution theorem results in:
\[ Y = X \odot H \] (4.2)

Where \( H \) represents the Fourier transform of the signal \( h(-n) \) and the multiplication operator on Equation 4.2 refers to an element-wise operation.

And then, the computation of an exact filter could then be defined as the solution of Equation 4.2 with respect to \( H \).

The work in [22] proved this filter, the exact filter, not to be very robust to transformations such as translation, scaling, or rotations, for instance, so they proposed a new algorithm that averages exact filters that are trained for different transformations of the object of interest. The justification behind this approach was that averaging emphasizes common features to a class of images and suppresses those that are image-specific, in an attempt to prevent or at least mitigate over-fitting the filter to the image.

The new filter, \( H \), becomes:

\[ H = \frac{1}{N} \sum_{i=1}^{N} H_i \] (4.3)

Where \( N \) is the number of training patterns used to define the filter.

Later, the work in [23] further improved this concept by defining the final filter \( H \) as the solution to an optimization problem that minimizes the sum of squared errors between the actual output of the convolution and the desired output of the convolution. The problem is formulated as:

\[ \hat{H} = \min_{H} \sum_{i} |X_i \odot H - Y_i|^2 \] (4.4)

This is a linear least squares problem with a known closed form solution.

This new approach at tracking with correlation filters improved robustness to variations in lighting, scale, pose, and non-rigid deformations while operating at very high FPS, and using less images to train the filter.

This work also introduces a measure of confidence in the detection step, referred to as the PSR. To compute the PSR, the correlation output \( y \) is split into the peak which is the maximum value and the side-lobe which is the rest of the pixels excluding an \( 11 \times 11 \) window around the peak. The PSR is then defined as:

\[ PSR = \frac{y_{max} - \mu_{sl}}{\sigma_{sl}} \] (4.5)

where \( y_{max} \) is the peak value and \( \mu_{sl} \) and \( \sigma_{sl} \) are the mean and standard deviation of the side-lobe. This measure of confidence instructs the algorithm to stop updating the filter if the value drops below a designated threshold and helps it recover from severe occlusion scenarios, should the object of interest...
resurface with a similar appearance.

4.2 Image Features

Visual information and content can be understood as a hierarchy of levels of abstraction. At the most basic level one can find the raw pixels that encode color or brightness information. Applying simple filters yields higher levels of abstraction such as edges, corners or color regions. Further processing interprets the previous features as objects to which attributes can be assigned to. The highest level of abstraction provides understanding concepts that identify multiple objects and interprets the relationships between them.

The topic of image feature extraction refers to methods that aim at computing these abstractions of image information at every pixel of the image. The resulting features will be subsets of the image domain. Correlation Filters can be extended to perform on any sort of dense image features that preserve spatial information. As such they can work on raw image pixels or other commonly used visual features, such as HoG features or CNN features. Given the recent rise in popularity in using Deep Neural Network (DNN)s in recent years, with demonstrated performance improvements, in this work, CNN features are used.

For the case case of using CNN features the correlation computations made are different from the raw image pixels case in the sense that now multiple feature channels are taken into account whereas for the previous case, the 3 usual color channels, Red, Green, and Blue, are converted to an intensity level, or gray-scale. This difference in computation is addressed later in this Chapter. Convolutional Neural Networks are a type of Artificial Neural Network widely used in image processing and so, the next Subsection will introduce the basic concepts of a Neural Network, followed by a more in-depth consideration on the specifics of a CNN.

4.2.1 Introduction to Neural Networks

As the name implies, neural network systems take its cue from the human brain by emulating its structure. The beginning of Neurocomputing is often taken to be the research article of McCulloch and Pitts, in [24], which showed that even simple neural networks could, in principle, perform any arithmetic or logical function. That work was followed in 1957 by Frank Rosenblatt’s perceptron described in [25] which was a linear classifier, or the simplest kind of feed forward neural network, as shown in Figure 4.2. A brief explanation of the core components of a simple feed forward neural network is provided below.

In the example depicted in Figure 4.2, \(x_1, x_2, x_n\) represent the input values fed into the network, \(b\) represent a bias term, or a constant, \(w_1, w_2, w_n\) represent the weights, or importance of each connection, and the activation function adds a non linearity to the perceptron and is usually one of the types found in Figure 4.3, some sigmoid function or Rectified Linear Unit (ReLU).
Two major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient. First, let’s recall the definition of a ReLU. The output of the ReLU function is defined as:

$$y = \max(0, a)$$  \hspace{1cm} (4.6)

Where:

$$a = Wx + b$$  \hspace{1cm} (4.7)

The first benefit is the reduced likelihood of a vanishing gradient. This arises when $a > 0$. In this regime, the gradient has a constant value. In contrast, the gradient of sigmoids becomes increasingly small as the absolute value of $x$ increases. The constant gradient of ReLUs have showed results in faster learning [26].

The other benefit of ReLUs is sparsity. Sparsity arises when $a \leq 0$. The more such units exist in a layer the more sparse the resulting network is, because now those connections would be valued as $y = 0$. Sigmoids on the other hand are always likely to generate some non-zero value resulting in dense representations. Sparse representations seem to be more beneficial, performance wise, than denser ones, as well as being more biological inspired, as is explained in [27].
A feed forward network, often referred to as a Multi Layer Perceptron (MLP), combines multiple instances of these units found in Figure 4.2, usually arranged into layers, as seen on Figure 4.4. Each of these units form a weighted sum of its inputs, to which a constant term is added. This sum is then passed through its activation function. Being feed forward means that the connections between units across layers do not form any loops.

This type of network system is designated as a supervised learning algorithm as it is assumed a training set is available, a set of input patterns for which the correct output, the label, is known. In this sense, the training of a MLP is done using a measure of the error between the network output, and the label, for instance, the mean squared error. It involves a backward propagation through a network similar to the one being trained and for this reason the training algorithm is usually called backpropagation. Training an artificial neural network involves adjusting its internal weights so that it gives the desired response when presented with particular inputs.

![Figure 4.4: Example of a MLP with 2 hidden layers. Each circle represents a unit of the type indicated in Figure 4.2.](image)

There are several mathematical methods for minimizing the error measure. Given the high number of parameters that today’s neural networks have, and how simple the method is, one of the most common is the Gradient Descent (GD). Other optimization methods, such as Newton’s or any second order methods, could also be used to train ANNs, and would do it in less iterations. However, numerical methods for computing the second derivative require a lot of computation, especially in settings with a high number of variables making the computational cost per iteration prohibitively high for training.

The simplest version of the GD algorithm consists of iteratively taking steps, in the weights space, proportional to the opposite direction of the gradient of the function being minimized. A single step of this algorithm is seen on Equation 4.8.
\[ w^{n+1} = w^n - \eta \nabla E \] (4.8)

In this case, \( \nabla E \) represents the gradient of \( E \), the error measure, for instance, the mean squared error, relative to the weights \( w \). There are other variations of the GD method that improve both robustness and convergence time such as including a momentum term or adjusting the term \( \eta \) in execution time, also known as the learning rate. Further explanation on both these improvements is found on the book in [28], section C1.2.

When dealing with very large training sets, even the standard GD method with the improvements above mentioned can take too long to train since, for each iteration of training, the method needs to compute the error measure and gradient for every training pattern, usually known as an epoch, in order to take a step. To circumvent this problem, the Stochastic Gradient Descent (SGD) method is widely used. The SGD method works in the same way as the standard GD but instead of sweeping through the whole training set for each iteration, or weight update, one can select a batch of training patterns, with size as small as one, and as large as the whole set, and perform the iteration of SGD on it. Since the stochastic algorithm does not need to remember which examples were visited during the previous iterations, it can process examples on the fly in a deployed system. In order to guarantee that SGD converges, two constraints must be placed on the learning rate, \( \eta \).

1. \( \eta(n) \rightarrow 0 \)
2. \( \sum_{n=1}^{\infty} \eta(n) = \infty \)

To summarize the training process the algorithm below is provided.

**Algorithm 4.1: Stochastic Gradient Descent**

1. Propagate each pattern in the batch forward in the network
2. Compute the cost function derivatives
3. Propagate backwards and compute its internal variables
4. Compute the gradient components
5. Update the weights

This process needs a criterion for when to stop training, when the values for the weights should be fixed. There are 2 main stopping criteria used across the literature, one can select a maximum number of epochs for training or select the maximum amount of error allowed, and then train the network until this measure is not exceeded.

Other improvements suggest using a separate set of patterns for a validation step which showed good results in improving generalization. For a better explanation in how this validation set can improve generalization refer to [28], section C1.2.
4.2.2 Convolutional Neural Networks

Convolutional Neural Networks are a specific type of ANN widely used in image processing. Some of the goals of such systems can be object recognition, object localization, scene classification, or bounding box regression, for instance.

The most important annual challenge on image recognition is the ImageNet Large Scale Visual Recognition Challenge for which, in the year of 2015, researchers reported that software exceeded human ability in this task, at around 5% error rate. The challenge is to classify an image as belonging to 1 of 1000 classes.

ImageNet [29] is a database of about 1.2 million of images, all hand labeled. For any given word, the database contains several hundred images. The annual ImageNet contest encourages researchers to compete and measure their progress in getting computers to recognize and label images automatically, among other challenges. In 2010 the winning system could correctly label an image 72% of the time. In 2012 one team achieved a significant jump in accuracy to 85%, the winner for that year used an ANN and the results obtained were a drastic improvement over previous years systems. The year of 2012 started the rise in popularity of visual recognition systems to use CNNs to perform visual object identification and performance improvements are achieved every year.

CNNs share a lot of features with the MLP discussed in Subsection 4.2.1 as they are made of neurons that have learnable weights and are also arranged in a feed forward manner. The network expresses a single differentiable score function across the whole network, from the raw image pixels on one end to scores at the other, and all that was discussed before, related to training, still applies as well. The advantage of using a CNN instead of a MLP where each pixel would enter the network on its own input neuron is that one can leverage knowing that the input is an image and encode some properties into its architecture that both improve performance and greatly reduce the memory overhead.

4.2.2.A CNN Architecture Considerations

As discussed in Subsection 4.2.1, a MLP receives an input pattern and modifies it through its internal neurons, these neurons are usually stacked as layers so they are usually referred to as hidden layers. In this layer configuration, every neuron from one layer is connected to every neuron on the following layer, as seen on Figure 4.4. This means that neurons on the same layer are treated as completely independent, the connections are not shared across the same layer. This full connectivity leads to an overwhelmingly high number of parameters that do not scale well as the image size increases.

CNNs arrange the neurons in a different way, more suitable for when images are used as input patterns. The convolutional layers are arranged in a 3 dimensional block, with width, height, and depth, as seen in Figure 4.5. The neurons on a given layer will only connect to a selected area on the layer that precedes it and not to every one, as seen on a MLP and these layers are usually arranged in a
pyramidal shape, so the width, and height decrease, and the depth increases as the ANN goes forward. It is usual, in classification tasks, to have the final layer with width, and height set to 1 and the depth set to the number of classes in the task and so the output would the result of a softmax function on that final layer, as the one found on Equation 4.9.

\[
y(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}
\]  

(4.9)

This softmax function takes as input a vector of generic size \( K \) where every element is an arbitrary real value \( z \) and transforms it into another vector with dimension \( K \) of real values in the range \( [0, 1] \) that add to 1.

The final vector now can be used to represent a categorical distribution — meaning, a probability distribution over \( K \) different possible outcomes.

In summary, a CNN is composed of layers that transform a 3D volume into another 3D volume with different dimensions with the use of a differentiable function.

4.2.2.B Types of layers on a CNN

Most CNNs are composed of 3 types of layers, the convolutional layer, the pooling layer, and the fully connected layer. A brief description on how the convolutional, and pooling layers work is provided below, as the fully connected layer was already presented.

The convolutional layer is the core layer of a CNN. This layer consists of a set of learnable filters, in the analogy with the MLP one could refer to them as weights but, for reasons that will soon be explained, the term filter better suits the computation these weights do. Every one of these filters is small spatially, along the width, and height dimensions, the size of the filter is called the patch size, and its depth is as big as the depth of the layer the filter is in.

During the forward pass on the CNN the filter is slid over the width and height of the input volume and produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. The training stage of this ANN will implicitly adjust these filters to be activated so that the final score function is maximized. It happens that, as training is done, that filters recognize a number of visual
features, such as gradients or blobs of color found at the early layers of the CNN up to more high level visual features later down the network.

This configuration of the network constrains the neurons in each depth slice to use the same weights and bias for the whole slide of the filter, making the learned filters insensitive to translation. It should also be noted that if all neurons in a single depth slice are using the same weight value, then the forward pass of the filter in each depth slice can be computed as a convolution of the neuron's weights with the input volume, hence the name convolutional layer. This is why it is common to refer to the sets of weights as a filter that is convolved with the input.

It is common practice to insert a max pooling layer between successive convolutional layers in a CNN, the periodicity of the appearance of these layers is an architectural choice. The goal of this layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and to also control over fitting. The pooling layer operates independently on every depth slice of the input and resizes it spatially, using the 'max' operation.

The most common form for this layer is a pooling layer with filters of size $2 \times 2$ which downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every 'max' operation would in this case be taking a max over 4 numbers. In the past, other networks chose to use other function in the pooling layer, one could choose average pooling or L2 norm pooling but more recent results show that the max pooling operation yields better results [30].

The network used in this work, the Googlenet, breaks the usual pattern found on CNNs architecture, that is convolutional and pooling layers in alternation until the final fully connected layer to output the final scores. This network features a more intricate and different connectivity structure detailed below.

### 4.2.2.C Googlenet Network

The ANN used in the scope of this work is the one described in [9]. In that work, it was proposed a deep CNN architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Challenge, the first version in 2014. The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by the introduction of the inception module that allows for increasing the depth and width of the network while keeping the computational budget constant.

The number of parameters was also kept much smaller than comparable ANNs, such as the VGG-Net. This means that the network is much more suitable for applications where GPU memory is more limited, such as on board the UAV used in this work or another mobile application.

This approach resulted in one of the first CNN architectures that significantly strayed from the general approach of stacking convolutional and pooling layers on top of each other in a sequential structure. This new model places notable consideration on memory, the number of learnable weights is much smaller
than conventional CNNs, and power usage because simply stacking all layers and adding large numbers of filters has a computational and memory cost, as well as an increased chance of overfitting.

At each layer of a traditional CNN, a choice must be made whether to have a pooling operation or a convolution operation, as well as the filter size used in such convolution. What an inception module allows is to perform all of these operations in parallel. The naive idea of how this module would look like can be found in Figure 4.7.

This implementation would be infeasible considering it causes a very high number of outputs, the depth channel of the output volume would be too great and so would be the computational cost. In order to counter this difficulty a $1 \times 1$ convolution is used and implemented as seen in Figure 4.8. This convolution serves two purposes:

1. Reduces the dimensions inside the inception module.
2. Adds more non-linearity by having ReLU activations immediately after every $1 \times 1$ convolution.

It can be seen from Figure 4.8, that $1 \times 1$ convolutions (in yellow), are used before $3 \times 3$ and $5 \times 5$ convolutions to reduce the input dimensions. It should be noted that a two step convolution operation can
always be combined into one, but in this case and in most other deep learning networks, convolutions are
followed by non-linear activation, usually a ReLU, and hence convolutions are no longer linear operators
and cannot be combined into one convolution.

![Figure 4.8: Full version of the inception module.](image)

This module acts on multiple convolution filter inputs, that are processed on the same input. It
also does pooling at the same time. All the results are then concatenated. This allows the model to
take advantage of multi-level feature extraction from each input while still remaining computationally
considerate.
# Used Methods

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Kernelized Correlation Filter</td>
<td>33</td>
</tr>
<tr>
<td>5.2</td>
<td>Transfer Learning with CNNs</td>
<td>37</td>
</tr>
<tr>
<td>5.3</td>
<td>Optical Flow</td>
<td>37</td>
</tr>
</tbody>
</table>
The following Chapter covers the theoretical aspects of all methods directly used in this work. Section 5.1 explains how to use this novel approach at correlation tracking for a single target, Section 5.2 explains how to re-purpose a CNN developed for the task of image classification into a visual feature extractor, and Section 5.3 details the used approach for optical flow computation.

5.1 Kernelized Correlation Filter

Despite the success of the aforementioned approaches at correlation based tracking systems, they only perform linear operations on the input images. Representing nonlinear transformations of the object of interest could only be achieved by increasing the number of training images for the filter, which would result in slower rate of processing. The work in [5] proposes using the kernel trick to allow the use of nonlinear operations as linear, achieving solution in cases where it was previously not practical to do so.

In this work, the problem is also formulated as an optimization problem. However, instead of the Least Squares approach taken in [23], here, this problem is formulated as a Ridge Regression problem, as follows:

\[
\min_w \sum_i \left( y_i - w^T x_i \right)^2 + \lambda \|w\|^2
\] (5.1)

Where \( x_i \) represents a sample image, \( y_i \) represents the desired output and \( \lambda \) is a regularization factor designed to prevent over-fitting. Both the sample images and desired output are vectorized.

The choice for the formulation of the problem as Ridge Regression is stated to be this way because a closed-form solution is known, and it performs similarly as other more sophisticated methods, such as Support Vector Machine (SVM). The solution for this problem is:

\[
w = \left( \lambda I + XX^T \right)^{-1} yX
\] (5.2)

Where \( X \) has one sample per row \( x_i \), and each element of \( y \) is a regression target \( y_i \).

5.1.1 Applying the Kernel Trick

The goal of the kernel trick is to find a mapping, or kernel function, \( \varphi(x) \), that enables operations on data points to be performed in a higher dimensional space without ever having to explicitly perform such mapping on the data, meaning not having to find the new space coordinates for every data point in the dataset in this new space.

In most cases, the need for this higher dimensional feature space arises with the goal to have linearly separated data. As most times this separation of data does not occur in the regular coordinate space, a mapping can be defined into another space, commonly referred to as the feature space, where data is
now linearly separable, meaning that a hyper-plane can be defined to perfectly separate all data points from each class. An example of such mapping can be found in Figure 5.1. In this example the data points are transformed from a 2 dimensional space into another 2 dimensional space. The difference here is that data points from different classes, represented by white and black dots, that were not linearly separable in the first space are now linearly separable in the second.

![Figure 5.1: Example of explicitly applying a kernel function.](image)

Applying this trick begins by replacing all sample images with their feature vector:

\[ x_i \rightarrow \varphi_i = \varphi(x_i) \]  \hspace{1cm} (5.3)

For this mapping, the number of dimensions of the feature space can be infinitely higher than the sample images, so it is important to assert not needing this explicit mapping.

The solution for this problem is given by:

\[ w = \sum_{i}^{N} \alpha_i \varphi(x_i) \]  \hspace{1cm} (5.4)

In this sense, the optimization problem is no longer done with respect to \( w \) but rather the coefficients \( \alpha \in \mathbb{R}^N \).

The algorithm is now defined in terms of the inner-products \( \varphi^T(x)\varphi(x') = k(x, x') \), computed using the kernel function and the solution for this version of the Ridge Regression problem, in the dual space, is now, in accordance with [5]:

\[ \alpha = (K + \lambda I)^{-1} y \]  \hspace{1cm} (5.5)

Where \( \alpha \) is a vector of coefficients and \( K \) is the kernel matrix that contains all the pairs of samples used in the linear combination in Equation 5.4. Having access to the kernel is the only thing necessary to compute the coefficients, not the explicit mapping.

The aforementioned work in [5] proves that the matrix \( K \) is circulant for a dataset composed of cyclic shifts if the kernel function satisfies the condition \( k(x, x') = k(Mx, Mx') \) for any permutation matrix \( M \).
Most commonly used kernel functions such as Gaussian, linear, polynomial, among others, satisfy this condition.

Because of this property one can obtain a fast solution for the regression problem in the nonlinear case as:

$$\alpha = \frac{Y}{k^{xx} + \lambda}$$

(5.6)

Where $k^{xx}$ is the first row of the kernel matrix $K$, and can be seen as the correlation of $x$ with itself, in the Fourier domain, and the fraction denotes an element-wise operation.

### 5.1.2 Preprocessing of Input Images

Due to the fact that images present discontinuities in their borders it is necessary to apply a window, or filter, to diminish artifacts that appear in the Fourier transform. Any type of window that goes smoothly to zero on the borders of the image could work on this scope and so, a commonly used window in digital signal processing was chosen, the Hanning window.

The one dimensional Hanning window is defined as follows:

$$H_{\text{ann}}(n) = \frac{1}{2} \times \left(1 - \cos \left(\frac{2\pi n}{N - 1}\right)\right)$$

(5.7)

Where $N$ defines the size of this window.

In order to get a two dimensional $(M \times N)$ Hanning window one can compute its outer product:

$$H_{\text{ann}}_{M,N} = H_{\text{ann}}_M \times H_{\text{ann}}_N^T$$

(5.8)

A graphical representation of how this window looks with a sample size of $30 \times 30$ is presented in Figure 5.2.

### 5.1.3 Filter Training

In order to train the correlation filter the filter coefficients $\alpha$ need to be computed. To do that the DFT of the desired output $y$ and the kernel auto-correlation $k^{xx}$ of the input image $x$ are required. The computation of $k^{xx}$ was covered in Subsection 5.1.1 and depends on the type of kernel function being used.

Another important aspect of correlation filters, that was not mentioned so far, is how to update the filter with the new information acquired with the new detections. Usually, the running average is the chosen method. In the KCF the coefficients $\alpha$ are updated as follows:
\[
\alpha_t = \eta \frac{Y}{k_{xx} + \lambda} + (1 - \eta) \alpha_{t-1}
\] (5.9)

Where \( \eta \) determines how much value is put in the previous filter coefficients. Note that if one chooses to entirely disregard previous information, the computation in Equation 5.9 is the same as the result provided in Equation 5.6

**Algorithm 5.1: Training KCF Tracker**

**Input:** \( x \) image patch, \( y \) desired response map, \( \alpha_{t-1} \) previous filter coefficients

**Output:** \( \alpha_t \) new filter coefficients

1. Weight \( x \) with a Hanning window
2. Compute Kernel \( K \)
3. Compute coefficients \( \alpha_t \)

For a step by step list of this segment of the algorithm, refer to Algorithm 5.1

### 5.1.4 Re-detection

At this point, the new filter coefficients \( \alpha \) have been computed, and so, this filter is ready to be used to re-detect the target in the following frame. Taking into account the dynamics of the environment it is assumed that the target is in the neighboring region of its last location, so a region around the target to perform this detection step is defined.

The translation from one frame to the other is the difference between maxima of this correlation response in two consecutive frames. In Algorithm 5.2 all the steps in this detection process are presented.
Algorithm 5.2: Detection Step

**Input:** \( x \) current model, \( z \) new image patch, \( \alpha_{t-1} \) filter coefficients from previous time frame  
**Output:** \( s \) translation  
1. Weight \( z \) with a Hanning window  
2. Compute Kernel  
3. Compute response map  
4. Determine maximum of response map and compute \( s \)

### 5.2 Transfer Learning with CNNs

The goal when using the aforementioned network is in the scope of visual feature extraction instead of image classification, the task for which the network was trained to perform upon, so the topic of transfer learning applied to CNNs will be considered.

The work presented in [31–33] suggests that one can take the output of any layer of a CNN before the fully connected part and use it as a fixed visual feature extractor for the new dataset being used. The complexity of the learned filters increase with how much of the network is kept.

The work in [31] states that on the first layer it is usual to find that the learned filters resemble Gabor filters or color blobs. However, as the propagation on the network occurs, higher level features are extracted up to the point where the final classification is made. [32] states that the results obtained strongly suggest that features obtained from deep learning with CNNs should be the primary candidate in most visual recognition tasks and many computer vision tasks have shown performance improvements by doing so.

The activations of the convolutional layers can be used as multiple channel features for the correlation filter. To extract these features the image is fed to the CNN and the activation of selected convolutional hidden layers is used as features.

Regarding the CNN used in this work, the Inception V3 [9], effort was put into using the outputs at the layer inception (4d) with an image size of \( 14 \times 14 \). This size proved to be too small for this work and so the decision was made to use the output at the layer inception (3b), therefore using a shorter network, and now dealing with an image size of \( 28 \times 28 \). The specific locations of the mentioned layers can be found in the work cited above.

### 5.3 Optical Flow

Optical Flow (OF) is the apparent motion of image pixels or regions from one frame to the next, so, while being a tool that provides a lot of information about the dynamics of a scene, it is not a 3D analysis, nor does it require any special equipment, it is computed solely with the image frames. It compares frames and tries to map or match every pixel across time.
The OF applied in this work is the one described by Gunner Farneback in [34]. A limitation on several approaches is that the estimation of the spatio-temporal orientation tensors requires the motion field to be temporally consistent, which is difficult for camera equipment mounted on UAVs or helicopters. To mitigate this problem, the motion field is estimated using only two frames at a time and try to compensate for the background motion, the high frequency vibrations of the vehicle. Since the above mentioned approach performed well for the sequences of the Wallenberg Laboratory for Information Technology and Autonomous Systems (WITAS) project \(^2\), captured with a camera mounted on an helicopter, it was the chosen method in this work.

In order to have a smoother result on the OF while simultaneously achieving faster computations, the image frame fed here is sub-sampled to a quarter of its size using bilinear interpolation.

Other approaches on calculating the OF are based on a sparse representation of the frame rather than dense, as is the one used in this work. These approaches, like the Lukas-Kanade, compute the OF by matching key-points across time, these key-points could be extracted using the Shi-Tomasi corner detection, for instance. While approaches based on a sparse computation of the optical flow are computationally less expensive than the dense options, in this case, the low texture of the environment being recorded resulted in poor results for the OF computation and so the choice was made to use the dense option.

\(^2\)http://www.ida.liu.se/ext/witas/
6

Spatial Attention Model

Contents

6.1 Discretization of Space ............................................. 41
6.2 Attention Model Strategies ........................................ 42
The following Chapter will combine the two systems, the detector and the tracker, as introduced in the Chapter 3, into a single coherent one, using an attention model. The tracker uses less computational resources than the detector, making it perform at a faster rate than the detector, mostly because its network is shorter, has less convolutional layers, and deals with much smaller images. The tracker handles images cropped around an MV and the detector uses the full frame. For this reason, in order to have both systems running simultaneously in synchrony, it was decided to have the detector run only on a fraction of a frame, in a way that would make both performance requirements similar, this fraction of frame will be designated as an Attention Box (AB). The attention model has the goal of, for each new incoming frame, select an area on that frame to perform detection upon.

Other options regarding combining both the detector and tracker could be to skip enough frames in the buffer for the detector in a way that would make the time requirements for them similar, for instance, if the detector were 4 times slower than the tracker one could make the detector only perform every 4 frames to satisfy the time-performance needs. This approach would, in general, delay both the detection of new MV entering the frame and re-detecting known MV. For these reasons this approach was not considered.

The detection on an AB will serve two purposes; it will adjust the bounding box on known instances of MV, adjust its centers and bounding box's size, and search for new MV that may be entering the frame. Two different approaches to model this attention will be considered and the way these attention boxes are selected is explained in this Chapter.

The first attention model relies on randomly sampling ABs in the frame and checking for MV in those areas, a decision threshold is then implemented that switches from one of the two purposes above mentioned for this model, and the second attention uses a Bayesian filter to select the AB with the highest entropy, the one the least of which is known.

### 6.1 Discretization of Space

Each frame is considered to be divided in fundamental units that will referred to as AB. In this case a rectangle of \((M \times N)\) pixels was used with no overlap between each AB. However, one can use a rectangle with any desired dimensions as long as the assumption that the objects of interest are smaller than the AB defined still holds true.

These ABs are time invariant and are all of the same size. At each time frame, the detector will receive the selected AB and perform detection on it. A representation of this division for a sequence recorded at the resolution of \(1920 \times 1080\) pixels, Full HD, divided in 18 ABs is found in Figure 6.1. All tests were performed with the same amount of ABs.

The assumptions made here are, as follows:
Performing detection on an AB has equivalent performance requirements as the tracking system. This means that the time necessary to perform one step of detection on this smaller subject is equivalent to performing one step of tracking.

The MV found in the sequences do not exceed the size of the AB, otherwise the detection step would never be successful since it would only be dealing with detecting on partial objects.

The previously made assumptions were tested to be accurate as, when dealing with a frame as described in Figure 6.1 the detection step and the tracking step take approximately the same time, and the test dataset does not include MV larger than the size of the AB.

An overlap between the ABs could be defined as way to prevent, or at least mitigate the likelihood of, a MV to not be fully within an AB space domain but, as most recorded MV are small enough for this phenomenon not to happen for an extended period of time and, if the tracking has been initialized for that subject, the tracker can keep performing on that subject without much loss on accuracy, the decision was made to not add more AB, so as not to increase the computational time it takes to select an AB for detection. Adding more overlapping ABs is expected to improve performance results at the expense of increased computational complexity.

6.2 Attention Model Strategies

The following Subsections aim at explaining how the attention model strategies we devised in detail.

6.2.1 Random Sampling Attention Model

As stated, the detection part of this system aims at essentially two objectives. The first is to adjust the bounding box on known instances of MV such that the drift that the tracker accumulates as hundreds of frames go by is mitigated and, when a difference is detected between the tracker and detector with
high confidence, the bounding box is reset. The second goal is to find new MV that may have entered the frame.

The first approach at an attention model was to use the discretization of space defined in the Section 6.1 and, for each frame, select one of the ABs at random. The added functionality to this model was to have an adjustable setting that would select between the first or second task, implemented as a probability test.

When the first task is selected, the center of the AB is centered on the respective AB. When the second task is selected, one of the available ABs is drawn at random. This task selector will henceforth be referred to as the decision threshold, a value between 0 and 1. This means that if the decision threshold is set to, for instance, 0.1, 10% of the selected ABs will try to re-detect known MV and the other 90% will focus on finding new MV. If there are more than one known MV one of them is chosen randomly when re-detecting.

Results for this approach as a function of the density of MV, speed of the UAV and the aforementioned adjustable setting are presented in Chapter 7.

### 6.2.2 Greedy Attention Model with a Bayesian Filter

In this approach, the goal is to find an AB of interest, the one the least of is known, so that entropy can be minimized. The intuitive source for this lack of knowledge, or entropy, originates from new information that has arrived at the scene and have not been used for detection or an area where a lot of frames go by without detection being performed.

Consider the event:

\[ X_{i,t} \in \{0, 1\} = AB^i \text{ has a MV at time } t \]

The implementation of the attention model closely resembles the Grid Localization algorithm from [35]. This means that a probability \( P(X_{i,t}) \) is defined for each AB and the posterior probability is approximated for each of them. These ABs are arranged in the same way as described in Section 6.1.

This approach, at each step, or, for each new incoming image frame, selects the AB with the highest entropy at that point. The evolution of the probability distributions is estimated for 1 step, hence the name Greedy.

The sets of all ABs mentioned, in addition with on extra set with an AB that includes all exterior information, the AB where all outside information is collapsed onto, are defined, and identified as:

\[
\text{Set } S_1 = \{ \text{all inside ABs} \} \\
\text{Set } S_2 = \{ \text{outside AB} \}
\]
The pseudo-code implementation of the Grid Localization algorithm can be found on Algorithm 6.1. This approach receives the probability values for each AB, $P(X_{i,t-1})$, as well as the optical flow computation, as described in Section 5.3, used in the motion model step.

**Algorithm 6.1: Grid Localization**

**Input:** Probability Distribution ($P(X_{i,t-1})$), Optical Flow  
**Output:** New Probability Distribution ($P(X_{i,t})$)

1. for $i \in AB$ do  
2. $P(X_{i,t}) = \text{motion\_model}(\text{optical\_flow}, X_{i,t-1})$;  
3. $P(X_{i,t}) = \text{measurement\_update}(X_{i,t-1})$;

At the beginning of each video sequence the probability distribution of the event has maximum entropy.

This happens because no prior information on the behavior of the scene is given, and so the distribution is uniform. This level represents the probability of having a MV in the cell. The AB used to encode outside information is always kept at maximum entropy, its information is always unknown.

$$P(X_{i,t=0}) = 0.5 \forall_{i \in \{S_1 \cup S_2\}}$$  \hspace{1cm} (6.1)

The loop in Algorithm 6.1 iterates through all cells of the set $S_1$. Line 2 of this algorithm implements the motion model update, sometimes referred to as the prediction step, and line 3 the measurement update. The way the motion model update is computed is explained in Subsection 6.2.2.A and the measurement update in Subsection 6.2.2.B.

### 6.2.2.A Encoding Optical Flow in motion model

The way the OF is computed for each new frame was covered in Section 5.3 so the goal now is to explain how that information shapes the probability transition, or motion update of the Algorithm 6.1.

For each of the ABs, the average of the OF within its area is computed. The tensor this computation provides reveals how the contents moved from the previous frame to the current, as if masses are being transported across the scene.

The transition of mass is proportional to the area that that AB overlaps with its neighbors as in Figure 6.2, that shows an example of how the top-left AB moved from one frame to the next. With this in mind, $P(X_{i,t+1})$, the new probability value for $AB^i$, is given by this ratio of overlap of the ABs, a value designated as $w(j,i) \in [0,1]$, that indicates how much of $AB^j$ overlaps with $AB^i$, times the previous value for the probability of said AB, divided by a normalization factor. This computation is provided below.

$$P(X_{i,t+1}) = \frac{\sum_{j \in \{S_1 \cup S_2\}} w(j,i)P(X_{j,t})}{\sum_{j \in \{S_1 \cup S_2\}} w(j,i)}$$  \hspace{1cm} (6.2)
This computation is performed for every AB, as described in Algorithm 6.1.

### 6.2.2.B Performing the Measurement Update

At each time frame the detector begins by performing detection on a single AB that is chosen as the one with the highest entropy, that provides one of two outcomes; finding or not finding a MV, after which the appropriate \( P(X^{i,t}) \) will be set to 1 or 0, respectively.

For all other frames that were not chosen for detection, its probability \( P(X^{i,t}) \) gets closer to 0.5 following an uniform step. The size of this step translates how much the entropy is modeled to increase for each frame not selected for measurement. The greater the value, the higher this entropy increases.

The binary entropy is defined as:

\[
H_b(p) = -p \log_2(p) - (1-p) \log_2 (1-p)
\] (6.3)

For which the graphical representation can be seen in Figure 6.3.

If the value for the update is too small, the selected ABs tend to be ones at the edges of the frame.
because new information into the frame is treated with higher priority than already detected upon ABs, the uncertainty of every other AB increases too slowly for them to be selected. If the value for this step is too great, new information into the frame has less priority. It is expected for the optimal value of this parameter to be related with how fast new information arrives at the scene.
7 Evaluation

Contents

7.1 Defining the Cost Function .................................................. 49
7.2 Results on Real Data Sequences .......................................... 50
7.3 Simulator Characterization .................................................. 51
7.4 Results on Synthetic Data .................................................. 54
Any of the attention model approaches considered have a cost associated with it, both in terms of implementation effort and performance. This Chapter explores a more comprehensive study and comparison between the aforementioned attention model strategies. The goal is to evaluate their performance in the SEAGULL [1] dataset and then on a simulated test suite with synthetic data. The two attention model approaches vary from a simple to implement Random Sampling strategy to a more involved greedy attention model based on a Bayesian filter as described in Chapter 6.

7.1 Defining the Cost Function

The first step towards having a method of comparing attention model approaches is to define a cost that works as a function of the reported detections of every MV in the frame.

The instantaneous loss function, or the error per frame, used to compare these attention models is, as follows:

$$ J(d_i^t, b_i^t, \hat{l}_i^t, t) = \alpha \sum_{i=1}^{X} I(d_i^t = 0 \land b_i^t = 1) + \gamma \sum_{j=1}^{B^t} \text{dist}(\hat{l}_j^t, l_j^t) $$

(7.1)

- The $\alpha$ is a scalar parameter that is used to adjust how important it is not to have false negatives patterns and, in all tests, this parameter was set to 345, the mean value of the edge size of an AB. In testing having false negatives is valued the same way as the false positives, hence the choice for this parameter.
- The value $X$ refers to number of ABs of the system.
- The values for $d_i^t$ and $b_i^t$ refer to the detection of a MV, when the system verifies that it has found an MV, or the actual existence of one at that AB at time $t$, respectively.
- The $\gamma$ is a scalar parameter that is used to adjust the importance of drift in the tracking step as well as the false positive rate. In all tests this value was set to 1.
- The value $B^t$ refers to number of trackers deployed at time $t$, meaning how many trackers are in current use.
- The $I(\cdot)$ function is defined as follows:

$$ I(d_i = 0 \land b_i = 1) = \begin{cases} 
1 & \text{if there is a MV } b_i \text{ but not a detection } d_i \\
0 & \text{if there are both a MV } b_i \text{ and a detection } d_i \\
0 & \text{if there are not a MV } b_i \text{ neither a detection } d_i \\
0 & \text{if there is not a MV } b_i \text{ but there is a detection } d_i 
\end{cases} $$

(7.2)
The symbols $\hat{l}_j$ and $l_j$ refer to the estimated position of the $MV_j$ and its real position, respectively.

The $dist(\cdot)$ function is defined as follows:

$$
\text{dist}(\hat{l}_j, l_j) = \begin{cases} 
    \| \hat{l}_j - l_j \| & \text{if there’s a tracker } j \text{ and } \| \hat{l}_j - l_j \| < D \\
    D & \text{if there’s a tracker } j \text{ and } \| \hat{l}_j - l_j \| > D \text{ or there’s a tracker } j \text{ but not a boat } j 
\end{cases}
$$

(7.3)

The value of $D$ is defined as the mean of the edge sizes of an AB.

The decision for this new metric could be done to separately evaluate the tasks of detection and tracking, therefore using the first term of the Equation 7.1 to penalize errors in detection, and the second term to penalize errors in tracking. This solution was avoided because it would penalize the case of missed detection twice, first in the $I(\cdot)$ and then again in the $\text{dist}(\cdot)$ function. For this reason, the decision was made so the $I(\cdot)$ does not penalize missed detections.

In order to get average loss value for the entire sequence:

$$
C = \frac{1}{N} \sum_{i=1}^{N} J
$$

(7.4)

Where $N$ is defined to be the number of frames in the sequence.

### 7.2 Results on Real Data Sequences

The tests on real data were performed on four real sequences acquired at the coast of Portimão, Portugal, and the recorded probability of an AB having a MV, $P(MV)$, was 0.0375.

In this case, two variables were taken into account for each attention model approach. For the Random Sampling approach, the decision threshold means the frequency at which the attention model tries to re-detect known MV or search for new ones, for instance, a value of 0.1 indicates the 10% of the ABs selected for detection should be used to re-detect known MV. For the Greedy Bayesian approach, the uniform step means how much not detecting on any given AB affects its entropy, how big the uniform step is.

The detector setting for both attention model approaches determines at which level of confidence a detection is accepted, values were evaluated from 0.5 to 0.97, detections with confidence smaller than 50% were disregarded. The top value is set at 0.97 because no detection ever received a higher score than 0.98 in this dataset.

The tests were performed five times for each of the sequences, for each attention model proposed, and the results can be found in Figure 7.1 and Figure 7.2.
The best recorded result for the Random Sampling and the Greedy Bayesian strategies are 17.95 and 18.23, respectively. In this case the Random Sampling strategy outperformed the Greedy Bayesian strategy, even if only getting a result 1.54% lower than the alternative.

In both tests it is observed that the highest score is achieved when the detector setting is used at 0.97. This means that the optimal setting for this variable is to only accept a detection with 97% confidence or higher. What is not clear from the performed tests is how the decision threshold for the Random Sampling strategy or the update step for the Greedy Bayesian strategy influence the defined cost function. This phenomenon is attributed to the fact that, not only is the density of MV low, all data sequences have mostly just a single MV and the trajectory of the UAV is defined around that MV.

This means that the probability of new MV entering or leaving the frame is too low on this data, the scenario where the Greedy Bayesian attention model shows the greatest advantage against the Random Sampling approach. For this reason, a simulator for the data and the complete tracking and detection system was developed. This novel simulator will augment our dataset allowing the definition of arbitrarily long sequences with varying degrees of MV that should provide better insight at how these two attention models compare with one another.

### 7.3 Simulator Characterization

This MV simulator was developed in MATLAB and assumes a top-down view on the boats, instead of being at a 45° angle relative to the ground, the setting used under the SEAGULL project. This should
have little to no influence in the results obtained as the decision to have a camera at 45° was made so the detector could have a better appearance model of a MV, it would not affect the attention model performance.

In this simulation, the MV are represented as a single point and so, no distinction is made between bigger or smaller MV. This assumption is made for simplicity reasons. In addition to this constraint, the MV are also considered to be stationary. From the gathered real data sequences, it was observed that the UAV motion in these sequences is much more significant than the movement from the MV motion and so, this simplification was also made.

To generate a UAV trajectory, random samples from a Gaussian distribution with zero mean and unit variance are generated for the acceleration of the UAV that are then integrated to velocity and finally position values. The trajectory is then recorded and used for all simulations in the Chapter 7.3.

In summary, the user of this simulator has at his disposal several tuning knobs designed to allow for experimentation on attention model approaches, the ones already included or additional approaches, from which it is highlighted:

- One can change the world size to any value desired;
- The distribution of MV on the world can be generated on-the-fly, following a uniform distribution, or an already made list of MV can be loaded onto the simulator;
- The length of the sequence is adjustable;
- The UAV trajectory can be generated on the fly or a predefined trajectory can be loaded;

*Figure 7.2:* Results of the cost function on the Greedy Bayesian Attention Model for real data sequences.
• The settings for both the detector and tracker can be changed to any desired value;
• The simulator already computes the error measure, as described above in this Chapter.

With this in mind, the following Subsection characterizes the tracking and detection system from a probabilistic point of view, so their behavior can be simulated.

7.3.1 Characterize Complete System

In this Subsection the system is characterized, the detector and tracker building blocks, from a probabilistic point of view, in order to accurately represent it in the simulations that follow. This characterization will be useful when simulating it against synthetic data, the results of which can be found in the Section below.

The attention models were completely ported to this new framework, there is no need to characterize them from a probabilistic perspective.

7.3.1.A Detector System

A detection is considered when a positive match with a confidence higher than 0.97 is received, this was the configuration that yielded the best results in the real data. In this setting, the probability of receiving a detection in the case when there was a MV present and the case where there was not a MV present was measured. The measured probability values are found in Equation 7.5 and Equation 7.6.

\[ P(detection|MV) = 0.91 \]  \hspace{1cm} (7.5)

\[ P(detection|\overline{MV}) = 0.034 \]  \hspace{1cm} (7.6)

These parameters will be used to simulate the detector for the synthetic data.

7.3.1.B Tracker System

The tracker is modeled using a random walk. This is the process by which randomly-moving objects wander away from where they started. The expected distance the target moves from its starting point for the standard random walk process is \( \sqrt{N} \), where \( N \) refers to the number of steps taken, but one can change the size of these steps to more accurately model how the tracking step behaves.

This process is used to model how the tracking tends to drift from the target as frames go by. Tests show that, on average, the tracking drifted 38 pixels per 100 frames and so the random walk process was also defined with this behavior.
7.4 Results on Synthetic Data

The tracker and detector have their behavior simulated using the characterization of Subsection 7.3.1 and the simulator is as described in Section 7.3. The Subsections below evaluate the attention models when changes to the environment are applied, the first set of evaluations deal with changes in the MV density, and the second set of evaluations handle changes in the UAV speed.

7.4.1 Performance With Changing MV Density

For the first batch of simulations, three scenarios were taken into consideration. The only difference between these scenarios is the density of MV, and the results will be presented in decreasing order of density.

The density variation is implemented as more or less MV are deployed in what is referred to as the world frame. A world was defined to have 6000, 3000 and 1500 MV, for the high, medium, and low MV density scenarios. The scenario with the fewer boats represents an MV density close to what is found in real data sequences.

The idea behind the metrics applied in this setting is to understand not only how the two attention models compare with one another but also how their performance evolves with changes in the environment so that the future user can better understand what he stands to gain by choosing one attention model over the other.

For the Greedy Bayesian Attention Model the only variable used is how big the uniform step is, as discussed in Chapter 7.2. In the case of the Random Sampling Attention Model the only variable is the decision threshold, also discussed in Chapter 7.2. For all Figures in this Chapter, the blue circles represent recorded data, and the orange dashed lines represent interpolated data.

![Figure 7.3: Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with \( P(MV) = 0.1252 \).](image-url)
The first setting occurs in a simulated world with the probability of an AB having a MV of \( P(MV) = 0.1252 \) and the performance of both attention models can be seen in Figure 7.3. As seen, the Greedy Bayesian Attention Model performs better than the Random Sampling Attention Model. The worst case scenario for the Greedy Bayesian Attention Model still gets a result 41.07% better than the alternative. If one takes the minimum cost of each model, the Greedy Bayesian Attention Model gets a cost 45.57% lower than the alternative.

![Cost Function for Random Sampling Attention Model](a) ![Cost Function for Greedy Bayesian Attention Model](b)

**Figure 7.4:** Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with \( P(MV) = 0.0731 \).

The second setting has \( P(MV) = 0.0731 \) and, as the Greedy Bayesian Attention Model remains the highest performing, if one compares each model at their lowest cost one gets that the Greedy Bayesian Attention Model gets a cost 33.27% lower than the other model. The graph that represents their performance can be seen in Figure 7.4.

![Cost Function for Random Sampling Attention Model](a) ![Cost Function for Greedy Bayesian Attention Model](b)

**Figure 7.5:** Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with \( P(MV) = 0.0301 \).
The third and last setting for the density has $P(MV) = 0.0301$ and again the Greedy Bayesian Attention Model performs better than the Random Sampling Attention Model achieving 15.64% lower cost. The graphs for this performance difference can be seen in Figure 7.5.

Regarding the Greedy Bayesian Attention Model the lowest performance is always achieved when the update step is selected to be 0.09, this showed to be the one that translates the importance of new information into the frame with already detected ABs for the selected speed of the UAV. As mentioned in Section 6.2.2.B, this parameter is expected to vary more with the UAV speed rather than the density of MV in the scene.

With respect to the Random Sampling Attention Model in the first two scenarios, the scenarios with the highest density of MV, the optimal value for the decision threshold is 0.1, this means that 10% of the ABs selected should be used to re-detect known instances of MV. When the density of MV decreases, the third scenario in the tests, the optimal value for the decision threshold is 0.2. The increase in value for this parameter makes intuitive sense since, as the density of MV decreases, greater focus should be shifted to re-detecting known MV rather than look for new ones.

Even though the Greedy Bayesian Attention Model outperformed the Random Sampling Attention Model in all evaluated settings, one can see that there are diminishing returns as the density of MV decreases. This finding also makes intuitive sense as the greatest asset of the first attention model is in how quickly it can anticipate other MV entering the frame, by using the optical flow information, and this advantage decreases as the world is less dense in MV.

The absolute results obtained in the third scenario are close to the ones in the real sequences, these had, approximately, the same density of MV. This result shows that the simplifications assumed when simulating the real world behavior were fair and did not have a severe impact in the recorded loss function. The results show that the probabilistic characterization of the system is accurate as well.

### 7.4.2 Results With Changing UAV Speed

The second batch of simulations tries to understand the effect the speed of the UAV has on the performance of the Greedy Bayesian Attention Model, with the results in Figure 7.6 and in the Random Sampling Attention Model, in Figure 7.7. The MV density was kept at the lowest density level.

In the case of the Greedy Bayesian strategy, the increased speed of the UAV resulted in the optimal value for the uniform step variable to be at 0.05, a smaller value than before. The higher the value, the higher the entropy in ABs that are not selected increases. The smaller this variable is set, the more the attention model focuses on ABs near the border, as new information coming into the scene, so it makes intuitive sense that the optimal value for the variable decreases as the speed of the UAV increases.

In the case of the Random Sampling Attention Model, the optimal value for the decision threshold stayed the same. The density of MV appears to be a more deciding factor when tuning this parameter.
Figure 7.6: Results of the cost function on the Greedy Bayesian Attention Model with $P(MV) = 0.0731$ and high speed.

Figure 7.7: Results of the cost function on the Random Sampling Attention Model with $P(MV) = 0.0731$ and high speed.

The absolute values for the cost also increased, in comparison with the results for the first batch. This change is attributed to the higher speed of the UAV that made the task of detecting and tracking MV more difficult.
8 Conclusion

Contents

8.1 Conclusions ......................................................... 61
8.2 Future Work ....................................................... 61
8.1 Conclusions

In this work, a new system that encompasses both tasks, tracking, and detection, working in synchrony is proposed. This system aims at overcoming the main challenges that vision algorithms faced in the maritime setting. Tracking robustness is assured with the kernelized correlation filter acting on CNN features and detection robustness relates to the choice of CNN for bounding box regression.

Two approaches at an attention model to fuse these two visual tasks, tracking, and detection are presented. The first is based on randomly sampling AB across the scene, and the second uses Bayesian filter to estimate the evolution of entropy for each AB with the intent to minimize it. Results show that, for scenarios with low MV density, both approaches have the same performance. However, as the MV density increases, the approach that uses the Bayesian filter shows increasingly better results.

The main limitations are:

- Failure when following small targets. The low resolution of the target in these cases makes it so it can be lost.
- Results were shown for images in the visible spectrum and surveillance in the maritime setting also includes the need for following MV during night time, using Infra-Red (IR) cameras. Results for this setting were not produced.

8.2 Future Work

Finally, some research topics that could extend the system presented in this document are identified as follows:

- Use the UAV telemetry information to predict where new maritime vessels may appear.

  In addition to the OF information gathered strictly from the images, one could leverage accessing telemetry data from the UAV to better predict the most likely entry points for newer MV entering the scene.

- Have different models for different heights.

  One could experiment having different appearance models for different heights and so attempt improvements on following even small objects with low texture.

In conclusion, a challenging problem with challenging dataset from a real world scenario was presented and a solution provided. The solution provided fuses both tasks, tracking and detection, with the advantages of being able to determine tracking failure, and having the ability at multi-target tracking.
Bibliography


