An Attentional Architecture for CNN-Based Visual Detection and Tracking

Tiago Filipe Figueira Cerveira
tiagocerveira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2017

Abstract

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 (Convolutional Neural Network), 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

1. Introduction

The maritime setting must be secured so illegal activities can be prevented. There are systems already deployed that tackle the issue of managing the movement of people, goods, services, and capital that rely on costly equipment and need to be closely monitored by a trained human workforce, which is scarce.

Under the SEAGULL project [1], a maritime surveillance system was developed using UAVs (Unmanned Aerial Vehicles) equipped with cheap optical sensors. 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.

This work focuses on detection and tracking of MV (Maritime Vessel) in videos, captured in a maritime setting. 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.

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.

1.1. Contributions

Key contributions of this thesis are presented below.

1. The KCF (Kernelized Correlation Filter) tracker hinges on the work in [4] with the implementation based in [5], in which improvements were introduced in the correlation computation, which greatly improved the computation time for each frame, resulting in higher FPS. A tracking failure measure was also added, known as PSR (Peak to Side-lobe Ratio).

2. All image features fed into the KCF tracker are produced GoogleNet's Inception V3 [6] network on the TensorFlow framework, that resulted in comparable performance concerning time and accuracy, compared to the VGG-Net [7], but lowered memory requirements.

3. The detector described in [8] 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. 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.

2. Related Work

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 VOT Challenge [9] and the MOT Challenge [10]. The challenge for creating a robust, accurate, and high performance approach still remains.

2.1. 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.

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 [11] relies on simple color blob analysis. The idea is that 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. This system is capable of real-time operation on the UAV.

The work presented in [12] also addresses tracking of MV and uses an EKF for the estimation of the MV states and for the prediction of vessel trajectories.

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. 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

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 a visual representation of these building blocks, refer to Figure 1.

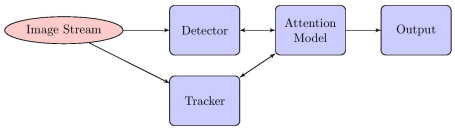

Figure 1: Diagram of the final system's building blocks and main interfaces.

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.

   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.

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. This simulator provides more granular control over the environment where the final system will perform and give

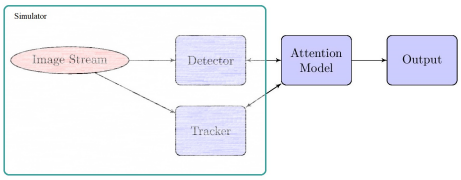

Figure 2: Representation of blocks the simulator replaces.

more insight on how this approach at a attention model is expected to perform.

When dealing with real data this block is absent, its purpose is only to generate new environments for the the full system to perform on. In order to better understand what building blocks this simulator aims at replicating, refer to Figure 2. All blocks inside the green area can be simulated.

4. Used Methods

The following Section covers the theoretical aspects of all methods directly used in this work.

4.1. Kernelized Correlation Filter

Despite the success of the aforementioned approaches at correlation based tracking systems, they only perform linear operations on the input images. The work in [4] proposes using the kernel trick to allow the use of nonlinear operations as linear.

In this work, the problem is formulated as a Ridge Regression problem, as follows:

$$\min_{w} \sum_{i} \left(y_i - w^T x_i\right)^2 + \lambda ||w||^2 \tag{1}$$

Where x_i represents a sample image, y_i represents the desired output, both vectorized, and λ is a regularization factor designed to prevent overfitting.

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 SVM (Support Vector Machine). The solution for this problem is:

$$w = \left(\lambda I + XX^T\right)^{-1} yX \tag{2}$$

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

4.1.1 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.

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

$$x_i \rightarrow \varphi_i = \varphi(x_i) \tag{3}$$

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:

$$w = \sum_{i}^{N} \alpha_i \varphi(x_i) \tag{4}$$

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 [4]:

$$\alpha = (K + \lambda I)^{-1} y \tag{5}$$

Where α 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 4. Having access to the kernel is the only thing necessary to compute the coefficients, not the explicit mapping.

The aforementioned work in [4] 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 satisfy this condition.

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

$$\alpha = \frac{Y}{k^{xx} + \lambda} \tag{6}$$

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.

4.2. 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 he topic of transfer learning applied to CNNs will be considered.

The work presented in [13, 14, 15] 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 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 [6], effort was put into using the outputs at the layer inception (4d) with an image size of 14×14. 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 28×28. The specific locations of the mentioned layers can be found in the work cited above.

4.3. Optical Flow

OF is the apparent motion of image pixels or regions from one frame to the next. It compares frames and tries to map or match every pixel across time.

The OF (Optical Flow) applied in this work is the one described by Gunner Farneback in [16]. Since the above mentioned approach performed well for the sequences of the WITAS project [1], 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.

5. Spatial Attention Model

The tracker uses less computational resources than the detector, making it perform at a faster rate than the detector. 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 AB (Attention Box). The attention model has the goal of, for each new incoming frame, select an area on that frame to perform detection upon.

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. 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.

[1] http://www.ida.liu.se/ext/witas/

5.1. 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.

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 1920×1080 pixels, divided in 18 ABs is found in Figure 3.

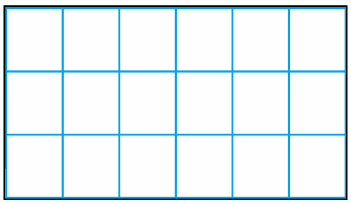

Figure 3: Discretization of space on a frame with FULL HD resolution. Each blue square represents an AB.

The assumptions made here are, as follows:

- Performing detection on an AB has equivalent performance requirements as the tracking system.
- the MV found in the sequences do not exceed the size of the AB.

The previously made assumptions were tested to be accurate.

5.2. Attention Model Strategies

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

5.2.1 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 reseted. 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 5.1 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.

5.2.2 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:

$$X^{i,t} \in \{0,1\} = \text{AB}^i \text{ has a MV at time } t$$

The implementation of the attention model closely resembles the Grid Localization algorithm from [17]. 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 5.1.

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 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 4.3, used in the motion model step.

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.

Input: Probability Distribution ($\text{P}(\text{X}^{\text{i, t-1}})$), Optical Flow
Output: New Probability Distribution ($\text{P}(\text{X}^{\text{i, t}})$)
for $i \in AB$ **do**
  $P(X^{\text{i, t}}) =$
   $motion\_model(optical\_flow, X^{\text{i, t-1}})$;
  $P(X^{\text{i, t}}) = measurement\_update(X^{\text{i, t-1}})$;
end

Algorithm 1: Grid Localization

$$P(X^{i,t=0}) = 0.5 \forall_{i \in \{S_1 \cup S_2\}} \tag{7}$$

The loop in Algorithm 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.

5.2.3 Encoding Optical Flow in motion model

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.

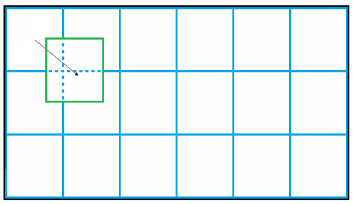

Figure 4: Example of the computed movement of an AB.

The transition of mass is proportional to the area that that AB overlaps with its neighbors as in Figure 4, 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.

$$P(X^{i,t+1}) = \frac{\sum_{j \in \{S_1 \cup S_2\}} w(j,i) P(X^{j,t})}{\sum_{j \in \{S_1 \cup S_2\}} w(j,i)} \tag{8}$$

This computation is performed for every AB, as described in Algorithm 1

5.2.4 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:

$$H_b(p) = -p\log_2(p) - (1-p)\log_2(1-p) \qquad (9)$$

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.

6. Evaluation

This Section 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.

6.1. 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:

$$J(d_i^t, b_i^t, \hat{l}^t, t) = \alpha \sum_{i=1}^{X} I(d_i^t = 0 \wedge b_i^t = 1) + \gamma \sum_{j=1}^{B^t} dist(\hat{l}_j^t, l_j^t) \qquad (10)$$

- The α 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 γ 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:
  - 1 if there is a MV b_i, not a detection d_i.
  - 0 if there is a MV b_i and a detection d_i.
  - 0 if there are not a MV b_i neither a detection d_i.
  - 0 if there is not a MV b_i but there is a detection d_i.
- The symbols $\hat{l}_j^t$ and l_j^t refer to the estimated position of the MV^j and its real position, respectively.
- The $dist(\cdot)$ function is defined as follows:
  - $\parallel \hat{l_j} - l_j \parallel$ if there's a tracker j and the expression $\parallel \hat{l_j} - l_j \parallel < D$ holds true.
  - D if there's a tracker j and $\parallel \hat{l_j} - l_j \parallel > D$ or there's a tracker j but not a boat j
- The value of D is defined as the mean of the edge sizes of an AB.

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

$$C = \frac{1}{N}\sum_{i=1}^{N} J \qquad (11)$$

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

6.2. 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

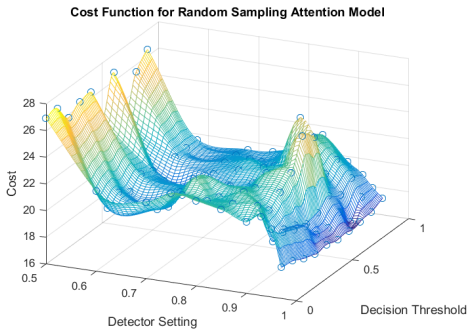

Figure 5: Results of the cost function on the Random Sampling Attention Model for real data sequences.

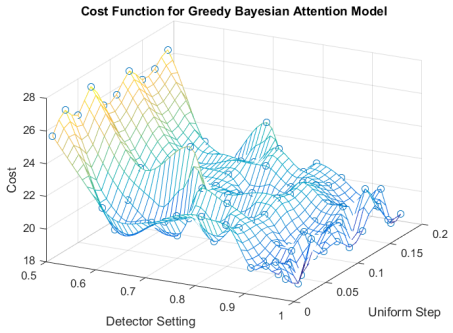

Figure 6: Results of the cost function on the Greedy Bayesian Attention Model for real data sequences.

the frequency at which the attention model tries to re-detect known MV or search for new ones. For the Greedy Bayesian approach, the uniform step means how much not detecting on any given AB affects its entropy.

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 tests were performed five times for each of the sequences, for each attention model proposed, and the results can be found in Figure 5 and Figure 6

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.

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.

6.3. Simulator Characterization

This MV simulator was developed in MATLAB and assumes a top-down view on the boats.

In this simulation, the MV are represented as a single point and so, no distinction is made between bigger or smaller MV. In addition to this constraint, the MV are also considered to be stationary.

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.

In summary, the user of this simulator has at his disposal several tuning knobs designed to allow for experimentation on attention model 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 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;
- 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 Section.

The following Section characterizes the tracking and detection system from a probabilistic point of view, so their behavior can be simulated.

6.3.1 Characterize Complete System

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.

- 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 12 and Equation 13.

$$P(detection|MV) = 0.91 \tag{12}$$

$$P(detection|\overline{MV}) = 0.034 \tag{13}$$

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

- Tracker System

The tracker is modeled using a random walk. 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 showed that, on average, the tracking drifted 38 pixels per 100 frames and so the random walk process was also defined with this behavior.

6.4. Results on Synthetic Data

The Sections 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.

6.4.1 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 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 Section 6.2. In the case of the Random Sampling Attention Model the only variable is the decision threshold, also discussed in Section 6.2.

For all Figures in this Section, the blue circles represent recorded data, and the orange dashed lines represent interpolated data.

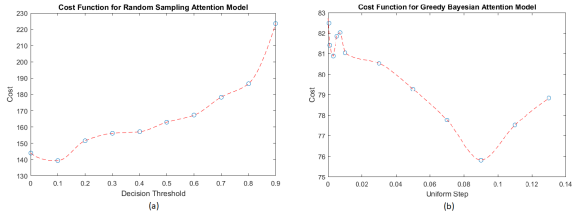

Figure 7: Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.1252$.

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. If one takes the minimum cost of each model, the Greedy Bayesian Attention Model gets a cost 45.57% lower than the alternative.

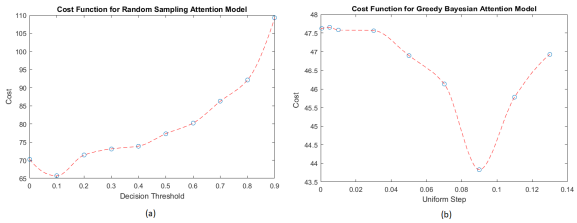

Figure 8: Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.0731$.

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 8.

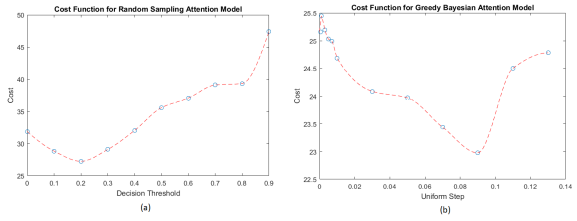

Figure 9: Results of the cost function on (a) Random Attention Model and (b) Greedy Bayesian Attention Model with $P(MV) = 0.0301$.

The third and last setting for the density has $P(MV) = 0.0301$, the same value for the real se-

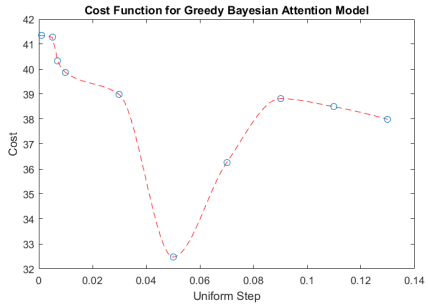

Figure 10: Results of the cost function on the Greedy Bayesian Attention Model with $P(MV) = 0.0731$ and high speed.

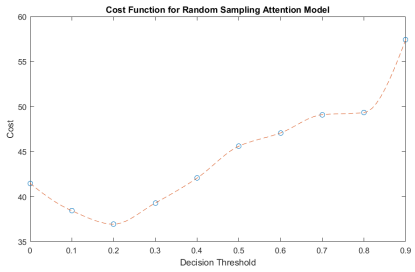

Figure 11: Results of the cost function on the Random Sampling Attention Model with $P(MV) = 0.0731$ and high speed.

quences, 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 9.

Regarding the Greedy Bayesian Attention Model the lowest performance is always achieved when the update step is selected to be 0.09.

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.

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.

6.4.2 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 10 and in the Random Sampling Attention Model, in Figure 11. 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.

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

7. 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.

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 is the failure when following small targets. The low resolution of the target in these cases makes it so it can be lost.

7.1. Future Work

One could 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.

One could have different models for different heights and 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.

Acknowledgements

I wish to acknowledge the help provided by Alexandre Bernardino, and Jorge Marques for their constructive suggestions and valuable input given in the planning and development stages 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.

Assistance provided by Gonalo Cruz and Jorge Matos was greatly appreciated in using the detector system and fully understanding implementation details of the correlation computations, respectively.

I would also like to acknowledge the help and assistance given by my family and close friends in making this final year thesis.

References

[1] Mario Monteiro Marques and Pedro Dias. Unmanned aircraft systems in maritime operations: Challenges addressed in the scope of the SEAGULL project. 2015.

[2] Andrei Barbu, Siddhart Naraynanaswamy, Aaaron Michaux, and Jeffrey Siskind. Simultaneous object detection, tracking, and event recognition. *Advances in Cognitive Systems*, 2:203–220, 2012.

[3] Longxin Lin, Weiwei Lin, and Sibin Huang. Group object detection and tracking by combining RPCA and fractal analysis, 2016.

[4] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[5] Ricardo Ribeiro Jorge Matos, Alexandre Bernardino. Robust Tracking of Vessels in Oceanographic Airborne Images. 2016.

[6] Christian Szegedy and Wei Liu. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 1–9, 2015.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Iclr*, pages 1–14.

[8] Alexandre Bernardino Gonalo Cruz. Improving Detection on Maritime Airborne Video Sequences. 2017.

[9] The visual object tracking VOT2016 challenge results. volume 9914 LNCS, pages 777–823, 2016.

[10] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, March 2016. arXiv: 1603.00831.

[11] Jorge S. Marques, Alexandre Bernardino, and Gonçalo Cruz. An algorithm for the detection of vessels in aerial images. pages 295–300, 2014.

[12] Lokukaluge P. Perera, Paulo Oliveira, and C. Guedes Soares. Maritime Traffic Monitoring Based on Vessel Detection, Tracking, State Estimation, and Trajectory Prediction, 2012.

[13] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems 27 (Proceedings of NIPS)*, 27:1–9, 2014.

[14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. 2014.

[15] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *Icml*, 32:647–655, 2014.

[16] Gunnar Farneb. Two-Frame Motion Estimation Based on Polynomial Expansion. *Lecture Notes in Computer Science*, 2749(1):363–370, 2003.

[17] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. 2005.