# RSNN: Recurrent Siamese Neural Network based Target Tracking

Akshay Sharma*[1], Mandeep Singh*[1], Praful Kumar*[2], and Mangal Kothari[2]

***Abstract*— This paper presents the challenges faced by the current state-of-the-art single object trackers in the challenging visual datasets which includes occlusions and similar objects in the frame. We also describe the proposed Recurrent Siamese Neural Network (RSNN) which is more robust in visually challenging scenario and is potentially better solution for active target tracking.**

## I. INTRODUCTION

Object detection and tracking is an important aspect of modern robotics, which nowadays are required to perform complex actions, ranging from an interactive humanoid robot to self-driving cars. With an ever-increasing extend of robotics into human life, from a fully-autonomous delivery drone to an robotic security bot, the need for robots that can see like we do. Dynamic Object tracking using autonomous agents is a very active and important field of research as it has wide ranging applications especially in the field of urban surveillance. The challenge is to develop a tracker that is not only accurate but also capable of yielding real time results. Deep Neural Networks, especially CNNs with the capability of extracting rich features have been proven extremely capable at image classification, greatly improving the tracking performances. This project aims to address the challenge of tracking a dynamic object moving randomly in a known environment using autonomous agents taking the help of vision based inputs.

## II. RELATED WORK

There are a lot of research work related to the object detection and tracking over the last few decades. Object tracking is the task to estimate the location and motion of the target objects using camera feeds. Various techniques are in the literature to solve the problem of object tracking (classified into single object tracking and multi object tracking). Early successful works which are the most popular trackers (optical flow) directly work on estimating the flow vector of the dense image pixels (Dense Optical Flow) or the features detected from the images (Sparse Optical Flow like Kanade Lucas Tomasi feature-tracker). Many have explored the idea of Kalman Filtering using some motion model and clustering algorithms like Meanshift, Camshift to predict the location of the moving object in the image which to the extend have shown great results for single object tracking. Recently, many single object trackers have become popular to run in real time and good tracking and failures and is included in the OpenCV library. These include BOOSTING (based on AdaBoost) which categories the image patches into positive and negative instances; Multiple Instance Learning (MIL) which is similar to the concept of Boosting with the addition of using bag of positive instances so that it would contain one instance whose centroid matches with that of the object; KCF (Kernelised Correlation Filter) built over the idea of MIL, TLD (Tracking, Learning and Detection) which includes first tracking the object frame to frame and then detector corrects its location and the detectors error is learned. Recently, after the success of Neural Networks (NN) soon after the ImageNet Challenge in major computer vision tasks, it has become a notion to test NN in all types of vision tasks. Many have implemented the idea of using convolutional neural network for the visual tracking problem. Among them is the fully connected Siamese Neural Network which has proven to perform best for the face recognition related tasks and raise interest in the tracking community for their simplicity and competitive performance. Also, recently Siamese network is implemented to perform single object tracking (SiamFC: Siamese Fully Conv-net) and its improved version CFNet which is the winner of VOT-2017 Challenge.

## III. EXISTING TRACKER ANALYSIS

We have tested various trackers in the OpenCV library: Boosting, KCF, MIL, TLD, along with the SiamFC, winner of VOT-2017 Challenge over different hard datasets which includes full occlusion, drastic change of instances. Some of the instances where all the popular object tracker failed is showcased in the following figures. BOOSTING, KCF and MIL tracker does not adapt the variation in the size of the object being tracked while TLD can adapt by changing the size of the bounding box (fig. 1). It is also found that BOOSTING and MIL track false objects during full occlusion and BOOSTING recovers after full occlusion but MIL fails to do so (fig. 2).

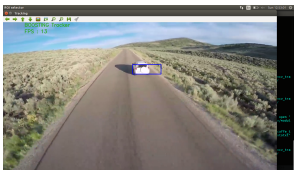

Fig. 1: BOOSTING fails to adapt bounding box size

Fig. 2: MIL fails after full occlusion

*This work was done in the partial fulfillment of AE640A: Autonomous Navigation course

[1]Akshay Sharma, Mandeep Singh is with the Department of Mechanical Engineering, Indian Institute of Technology (IIT) Kanpur, INDIA {akshay, mandeeps}@iitk.ac.in

[2]Praful Kumar, Dr. Mangal Kothari is with the Department of Aerospace Engineering, Indian Institute of Technology (IIT) Kanpur, INDIA {praful, mangal}@iitk.ac.in

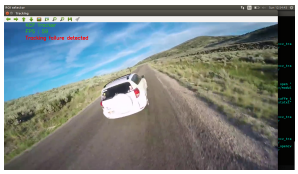

Fig. 3: KCF fails to adapt in changing viewpoint

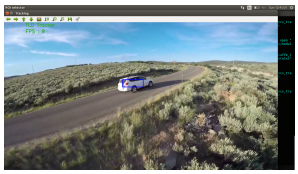

Fig. 4: TLD not robust in scale change

Also, KCF gives tracking failure in cases of change in the viewpoint of the camera in addition to the full occlusion (fig. 3. However, TLD is shown to perform much better in all the above mentioned cases. The object is still tracked even if the viewpoint is changed. Also, it shows correct tracking failure under full occlusion and it does recover after full occlusion. It also reports failure when object is not in the frame but in not robust in the scale change of the bounding box (fig. 4).

After the analysis of SiamFC, it is found that the performance of SiamFC is not robust in cases when the target object is in the environment with similar looking objects. It usually switches to the similar looking object leaving the target (fig 5). As it does not keep the past instances in memory it started tracking the new object as the actual target is occluded for some time (fig 6). So, saving different instances for future comparisons can potentially improve the performance.

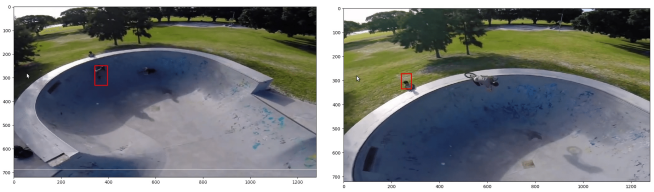

Fig. 5: SiamFC switch to similar looking person in BMX4 dataset

Fig. 6: SiamFC failed to track toyplane after occlusion

## IV. SYSTEM OVERVIEW

We will use the SiamFC tracker as our base, which uses a fully convolutional siamese neural network, and propose possible improvements in the same. This tracker was the winner of VOT-2017 Real-time Visual Tracking challenge, supporting frame rates up to 80 hz on a desktop and accuracy comparable to that of the state-of-art trackers. In this tracking approach a deep convolutional network is pre trained for a similarity learning problem and then is used to perform object tracking using the learned similarity functions on an exemplar image and the next frame. The basic working of the tracker involves the use of a learned similarity function f(z,x) which compares a candidate image x, with the target image z. Based on the score returned by this similarity function a prediction is made, such that a high score indicates more similarity between x and z. The candidate image x is taken from all possible locations in the new frame and the one with the highest score indicates the location of the target in the new frame.

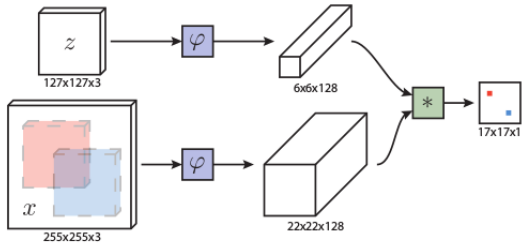


Fig. 7: Reference SiamFC model

The above figure. 7 represents the basic framework of the tracker. The target image z and the candidate image x are both passed through identical transformations Y and then those transformed images are passed through the similarity function g, such that $f(z,x) = g(\varphi(z), \varphi(x))$. The transformation $\varphi$ represents the embedding of the images. The function $g(\varphi(z), \varphi(x))$ actually produces a convolution of the embeddings of $z$ and $x$ for all possible $x$ in the new frame.

$$g(\varphi(z), \varphi(x)) = \varphi(z) * \varphi(x) + b1$$

, where $b1$ denotes a signal which takes value $b \varepsilon R$ in every location. This produces a score map on the whole frame indicating the similarity values for each candidate image. Then using the target image and the image with the max. Similarity score we can further measure the translation of the target object. To obtain good frame rates while tracking the tracker has been kept simple in the sense that it does not keep a memory of the past appearances and also avoids the use of techniques like optical flow or colour histograms. Below is an explanation of each component of the basic framework

### A. *Fully-convolutional Siamese network*

A fully convolutional Siamese network takes two input images $z^{'}$ and $x^{'}$, which are basically the training image and the test image respectively. The image $z^{'}$ defines the target object that is to be tracked and the image $x^{'}$ represents the next video frame and can be treated as a large search area in which we have to pick out patches that match the target object represented by the image $z^{'}$. Both the imaged are passed through a pre-trained CNN to generate feature maps.

These maps are used to generate a cross co-relation map by the following formula:

$$g_p(z^{'}, x^{'}) = f(z^{'}) * f(x^{'})$$

. This cross co-relation actually equates the exhaustive searching for a patch in $x^{'}$ which matches the target image $z^{'}$.

### B. Training with large search images

The siamese network operates on an exemplar image and a large search image and generate s a score map. This score map is used to calculate losses corresponding to each target and exemplar image pair. For training the following loss function has been used,

$$l(y, v) = log(1 + \exp(-yv))$$

where $v$ is the score of a exemplar-candidate pair and $y\varepsilon{+1}, -1$ is the ground truth label. This ground truth label descries whether a match was found for the exemplar image in the candidate image or not. As we pass a large search area along with the exemplar image we get a score value for all the pairs of the exemplar image and the candidates that are formed from the large search image. This gives us a loss value for all those pairs which are then used to find the mean loss for the pair of exemplar image and the large search image. The mean loss is defined as,

$$L(y, v) = \frac{1}{D} \sum_{u \varepsilon D} l(y[u], v[u])$$

. Now to obtain the parameters ($\theta$) of the convolutional network we apply Stochastic Gradient Descent (SGD) to minimize the loss function,

$$argmin_\theta E_{z,x,y} L(y, f(z, x, ; \theta))$$

The search area is taken from the center of the previous estimated location of the object in the image. The classification of an example as positive is made if the elements of the score map are within radius R of the center. $y[u] = +1$, if $k||u - c|| <= R$, otherwise $y[u] = -1$

### C. Tracking

To accomplish the tracking task the feature representations of both the target image and the search image are both passed through the Siamese network. For every new image frame a search area equal to four times the size of the target object is taken. This search area is centered at the estimated position of the object in the previous image frame. This search area forms the search image $x^{'}$. Now the position with the highest score value from the cross co-relation function gives us the position of the target object in the current image frame. The target image $z^{'}$ is also not static but is a weighted average of all the previous instances of the target object. When the object moves its appearance changes as its relative position with respect to the camera changes along with the viewing angles. The moving average part of the tracker takes this into account by updating the target image with a weighted new instance of the target object.

## V. Our Approach

While checking out the existing iteration of the tracker we noticed that at some places the tracker started following wrong but similar looking objects and also was not able to properly detect partial or full occlusion. On going in depth into the working of the tracker we came up with some possible additions in the tracker that can possible improve the overall functioning of the tracker. The existing tracker was taking a moving average of all the instances of the target object without considering whether the detected object was correct or not. This problem of not checking whether the detected object is correct or not create issues whenever the target gets partially or fully occluded. In these cases the tracker updates the exemplar image of the target with false objects which leads to it failing in some cases later on even when the object comes out of occlusion.

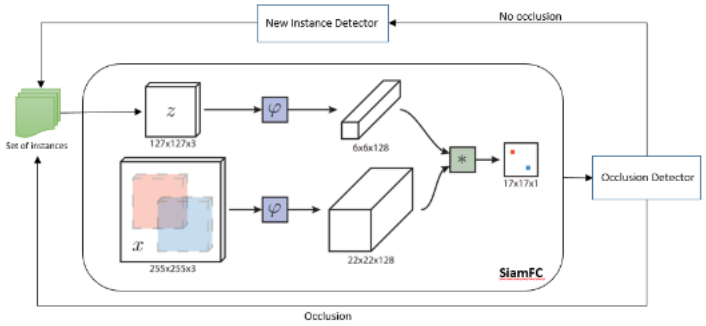


Fig. 8: RSNN model

### A. Score Thresholding

To account for the above problem we started storing the previous instances of the object. The criteria of storing the instances will be discussed below. Now we used the score map to generate a new score called max score. The score map earlier was being used to generate the location of the center of the new bounding box for the object. But now we used this location to calculate an average score within a window of one-tenth of the size of the bounding box. This score was generated for each stored instance and then these scores were normalized and were used as weights for generating a weighted mean representation of the target instances based on the stored instances. For every stored instance, the value of max score was calculated as:

$$maxscore_k = \frac{\sum_{x-\frac{w}{10}}^{x+\frac{w}{10}} \sum_{y-\frac{h}{10}}^{y+\frac{h}{10}} \left( score[i][j] \right)}{2(x+y)}$$

, here $k = k^{th}$ instance
x,y = represent the center of the predicted bounding box w,h = width and height of the bounding box

This max score for corresponding to every instance was saved in a array corresponding to every k. The array can be represented as $W$, such that

$$W[k] = maxscore[k]$$

This array was then normalized before calculating the mean template,

$$W = \frac{W}{||W||}$$

The mean template was then calculated using these weights and the stored instances,

$$template_{mean} = \sum_{k}^{n} (W[k] * template_k)$$

,where $n$ = Total no. of templates stored so far
$template_{mean}$ = mean template
$template_k$ = $k^{th}$ template

Now using the maximum of the max scores a new template was generated from the current image frame. This template was used to calculate cosine similarity with the mean template. This value was then used to decide whether to add this new template to the list of stored templates or not. The usefulness of these separately stored instances shows up when we detect the scores going to low values meaning there is an ambiguity amongst multiple objects. At this point instead of just checking with the earlier exemplar image we generate scores for all pairs of exemplar templates saved and the search area and the maximum of the maximum score generated is used to update the target location.

### B. Occlusion/Similarity Detection

In order to implement changes in model functioning in case of occlusion or subject change, it becomes nescessary to identify the situation. We plotted the heat maps of max score matrices generated by Score thresholding and it was observed that the model shows high probability of losing target when it's score matrix shows a diversion from single, dominant maxima to multiple local maximas. The absolute maximum score value in this case drops as the scores are redistributed.
Running the model on different sets, we identified a threshold value for absolute maximum score below which the score map begins to disassociate, signifying a case of occlusion or subject change (Fig.10 vs Fig. 11)

### C. LSTM

We decided to train a neural network on the given ground truth bounding boxes of the train dataset to extract a kind of motion model. To train this model we used LSTMs as they are used to learn data which is sequential in nature. The bounding box data that we have has a sequence in it as in a way this data stores the motion model of the target object.

LSTM takes a sequential input and predicts the next state after each input and passes the output to the next time step along with the next input to allow the sequential information to get incorporated. Here a series of bounding boxes is given as the sequential input through the LSTM model. The bounding box data is taken from the groundtruth values of the training data and is passed as a set of 5 timesteps and is ran for 30 epochs. The trained model is used along with

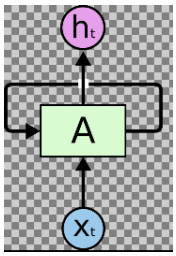


Fig. 9: Folded LSTM Cell

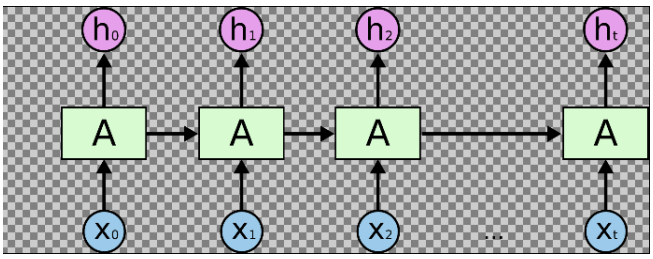


Fig. 10: Unfolded LSTM Cell

the above previous model. Whenever we detect partial or full occlusion the framework switches to the LSTM model and starts passing a series of previously estimated values of bounding boxes through the learned model. This model then predicts a new bounding box from the given input data and then this bounding box is used to track the object. This keeps happening until the maximum score values of the score map again gets within a given threshold indicating that the object has come into view once again.

### D. LSTM Motion Model Architecture

The LSTM system architecture used for predicting the motion of the target object is:

*Layer 1: LSTM layer with 32 hidden units*

*Layer 2: 0.5 Dropout layer*

*Layer 3: Dense layer with 16 neurons*

*Layer 4: Activation layer with softmax as activation function*

*Layer 5: Dense layer with 4 neurons*

## VI. RESULTS

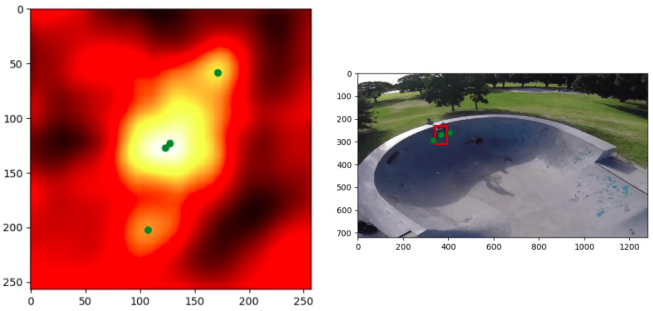


Fig. 11: SiamFC failed to track toyplane after occlusion

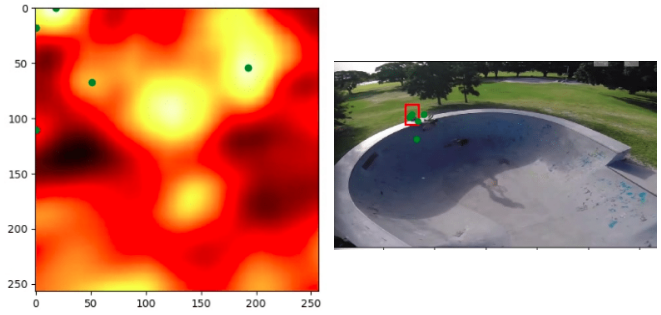

Fig. 12: SiamFC failed to track toyplane after occlusion

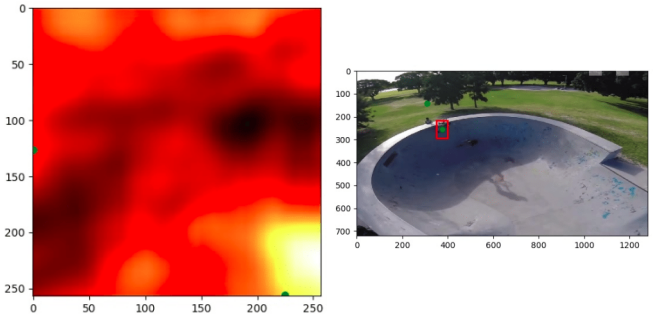

Fig. 13: SiamFC failed to track toyplane after occlusion

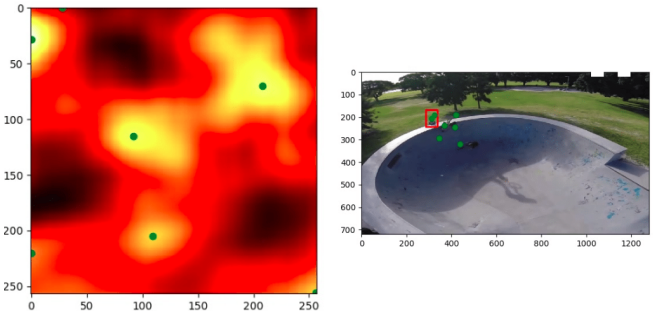

Fig. 14: SiamFC failed to track toyplane after occlusion

The above figures show the results of running our framework on the BMX4 dataset. The heat maps correspond to the score maps generated on passing the exemplar image and the candidate search area through the siamese network. In this particular dataset the original tracker started tracking the person standing by the wall when the cyclist passes closed to it and did not recover after that. Our framework tries to correct that wrong detection. In the first pair of heat map and image it can be seen that the correct object (cyclist) is being tracked and the corresponding heat map shows the score maxima to be on the cyclists which can be seen as the green dots. Now in the next figure 12 as the target approaches the person by the wall we see a spread in the heat map. The local maximas which are represented by the green dots spread in a larger area indicating multiple objects being closely matched with the exemplar image. This can be seen in the form of the bounding box being shifted to the person by the wall. Now unlike the original framework our model tries to correct this mistake. In the next figure 13 the tracker shifts back to the actual target. This happens because when the tracker detects the local maximas to be distributed it switches to the trained LSTM model which based on the previous positions of the bounding box, predicts the correct location of the target. Although the LSTM framework is working to an extent but we see that the tracker starts tracking the wrong object once again. The main reason behind this is that our LSTM model is not trained on enough amount of data due to lack of computing resources. Due to minimal computing power we were not able to train the model properly.

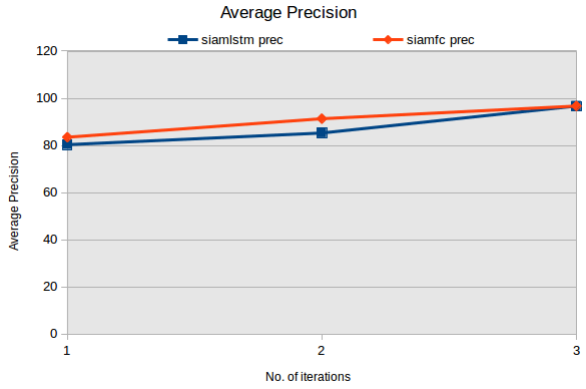


Fig. 15: Average Precision comparison

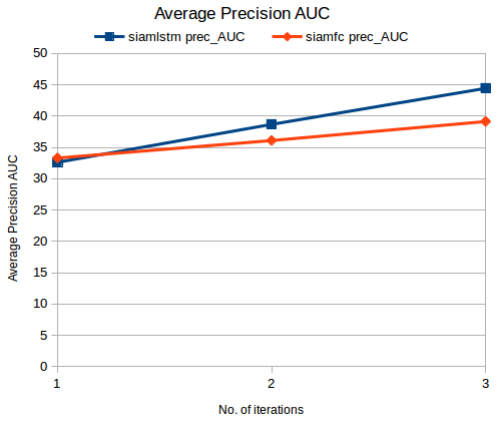


Fig. 16: Average Precision AUC comparison

The proposed RSNN (named as *siamlstm*) in the results is run on the small HKUST object tracking dataset. Also, the baseline SiamFC is used for comparison of the average precision (fig. 15), average precision AUC (fig. 16), and average IOU (fig. 17). The comparison is based on three different number of iterations. We found that the average precision is slightly less than the baseline SiamFC as the amount of video dataset used for comparison is a subset of the total dataset. Also, due to the less computational resources available to experimentally verify the proposed model, the

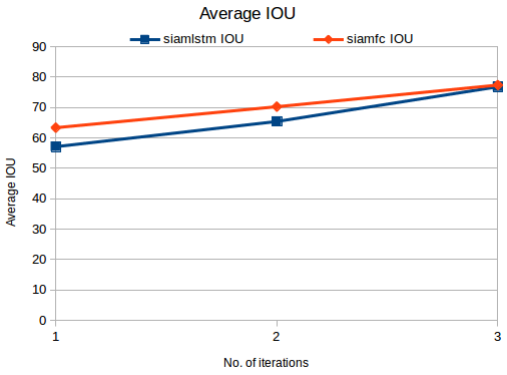


Fig. 17: Average IOU comparison

model is not well tuned. The computational speed of the system is by far 60 times less than the authors' hardware capabilities of SiamFC baseline which leads to slightly lower precision. Hence, RSNN can potentially outperform the baseline SiamFC with the adequate computational resources (with Nvidia GPUs) to train the model better.

## VII. CONCLUSIONS

The Siamese Neural Net is a state-of-the-art tracking framework but it has some flaws. We were able to come up with possible solutions after careful evaluation of the framework, and were able to implement those solutions in the existing framework to a certain extent. We were limited by the limited computing resources we had at our disposal. Even with the limited resources we were able to see some improvements that have been explained in the report, and we assume that if we are able to train our model on a larger dataset using better computing resources our model can perform significantly better than its current state.

## REFERENCES

[1] S. Salti, A. Cavallaro and L. Di Stefano, "Adaptive Appearance Modeling for Video Tracking: Survey and Evaluation," in IEEE Transactions on Image Processing, vol. 21, no. 10, pp. 4334-4348, Oct. 2012. doi: 10.1109/TIP.2012.2206035

[2] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, A. van den Hengel, "A Survey of Appearance Models in Visual Object Tracking," in ACM Transactions on Intelligent Systems and Technology (TIST), Sep. 2013.

[3] Bertinetto, Luca, et al. "Fully-convolutional siamese networks for object tracking." European conference on computer vision. Springer, Cham, 2016.

[4] Z. Kalal, K. Mikolajczyk and J. Matas, "Tracking-Learning-Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1409-1422, July 2012.

[5] Li, Siyi and Yeung, Dit-Yan *Visual Object Tracking for Unmanned Aerial Vehicles: A Benchmark and New Motion Models.* AAAI, pages(4140-4146), 2017

[6] Fu, Changhong and Carrio, Adrian and Olivares-Mendez, Miguel A and Suarez-Fernandez, Ramon and Campoy, Pascual *Robust real-time vision-based aircraft tracking from unmanned aerial vehicles* Robotics and Automation (ICRA), 2014 IEEE International Conference on pages (5441-5446), 2014