1. **Introduction**

Visual object tracking is a long standing challenge in the field of image processing and computer vision for quite some time. This is due to the fact that it is a fundamental requirement for efficient functionality of humans and smart machines – if it is to achieve a close to human level of performance in major areas of endeavour. Detecting the presence of a predefined object of interest in an environment, and effectively tracking it over successive frames is a recurring theme in security and surveillance, business (stock and market price action prediction), medical imaging, transportation, robotics, to name a few[[1-3](#_ENREF_1)].

**1.1 Study Background**

It is without saying that there has been tremendous progress in this field in the past few years, especially with the introduction of Deep learning architectures to computer vision. The convolutional neural network (CNN) in particular has been at the helm of recent object tracking researches, and are being presented in almost every computer vision journal and conference [[4-6](#_ENREF_4)]. This is as a result of the CNN’s ability to accommodate large training dataset, and come up with elegant feature representation which can be used for future tracking. As profound as the results have been, the strength of the CNN also happens to be its weakness, in that it largely dependent on the abundance of data and in turn, requires large computational capabilities. Also, CNN is not primarily designed to operate as a tracker where there are obstacles (although I believe it can achieve this due to its high level of computation and classification ability, if it is able to detect and match the emerging object to a previous detection).

State of the art tracking methods in the presence of occlusions and obstacles utilizes real time state estimation filters in the mould of Kalman filters [[7](#_ENREF_7), [8](#_ENREF_8)], correlation filters [[9](#_ENREF_9), [10](#_ENREF_10)], and of course, ensembles of deep networks and other optimization algorithms [[11-13](#_ENREF_11)], but these models come with a prior understanding of the physics of the environment that facilitates incorporation of an observer in the algorithm. However, the problem of uncertainty remains with existing traditional approaches.

This work explores the possibilities inherent in the re – emerging reinforcement learning system – which has gained some attention in the past few years – to solve the problem of tracking an object around obstacles and occlusions. Given an object of interest in a frame of a video, the goal is to successfully detect the object, and find its location in subsequent image frames. This task is made more challenging with the introduction of obstacles and occlusions which are commonly found in the typical human environment, and successfully designing a system that solves this problem with a higher degree of accuracy than existing traditional models might be the key to unveiling further life changing applications of autonomous object tracking in addition to the already existing applications in autonomous vehicle navigation, unmanned aerial vehicles (UAVs), stock price prediction, missile trajectory optimization, etc.

To deal with the issues outlined above, we propose a reinforcement learning system which combines the strength of the deep network for dimensionality reduction and pre – processing, and the flexibility of the model free reinforcement policies to recreate the path of an occluded object behind the obstacle, and anticipate its emergence with the goal of predicting future positions for the object of interest.

**1.2 Significance**

In lieu of the aforementioned challenges inherent in the accomplishment of successful object(s) tracking, this work is focused on the recreation of missing tracks when objects move in an occluded environment.

A successful path recreation would help anticipate the re-emergence of the object, and localize the region of search. This would assist the navigation of autonomous vehicles in environments with parked large vehicles, and also aid the search and quick recovery of missing airplanes when the work is extended to trajectory prediction. Another area of interest would be anomaly detection, when object motions do not behave as expected in a given environment.

**1.3 Aim of this work**

The aim of this research is to design a model for single object tracking.

**1.4 Objectives**

The objectives of this work are to:

1. Have an overview of existing object tracking algorithms.
2. Select a top performing algorithm.
3. Identify issues hampering its efficiency that remain unsolved.
4. Come up with suitable modification to enhance the performance.
5. **Related work**

**2.1 Visual object tracking**

Object tracking is the location of a single or multiple objects over a given period. As [[13-17](#_ENREF_13)] pointed out in their reviews, quite a number of efficient trackers have been developed to tackle this fundamental issue. Earlier object tracking models [[18-21](#_ENREF_18)] used the tracking by detection method. This isolates the target object using a background subtraction technique, and detects the identified object across various frames. It is a discriminative model that was updated to form an online tracking system.

With the boost of faster computational capabilities in personal computers, deep learning models (CNN in particular) have gained widespread acceptance in tracking systems [[22-24](#_ENREF_22)]. A number of trackers have employed CNN in feature extraction and data representation, while many more other tracking systems utilize pre – trained CNNs in large scale image classification problems [[4](#_ENREF_4), [6](#_ENREF_6), [25-27](#_ENREF_25)]. Most of these systems are trained with enormously large video datasets in other to have a wealth of features extracted for the system to work with and in spite of these, some of the models still struggle with occlusion of speeding objects.

**2.2 Deep Reinforcement Learning**

Reinforcement learning is a branch of machine learning in which one or more objects (known as agents) learn to act in an environment, in a way that maximizes some form of cumulative reward function [[28](#_ENREF_28)]. It is a machine learning strategy that is based on trial and error, where an agent samples a number of actions in a known or mostly, unknown environment, and receives a positive or negative reward.

Based on this principle, Ba et al [[29](#_ENREF_29)] and Mnih et al [[30](#_ENREF_30)] trained a reinforcement learner on spatial attention policies for image classification, [[31](#_ENREF_31)] designed a reinforcement learner that uses CNN for feature representation, then applied the recurrent neural network to the top level frame before running the reinforcement learning algorithm for offline tracking. In 2016, Silver et al showed that pre – training a policy network with supervised learning before applying the policy gradient improves the performance of the reinforcement learner[[32](#_ENREF_32)]. Similarly, [[33](#_ENREF_33)] took advantage of the technique to propose a tracker that is pre – trained using a 3 – layer CNN with 4 more layers that are concatenated with the action dynamics of an action based decision network to predict the action probabilities and state confidence score, which was validated to achieve a competitive performance with much reduced computation complexities.

Ren et al [[34](#_ENREF_34)] proposed an offline deep prediction – decision network model for multiple object tracking. It considers each object as an agent, detects and predicts the objects’ trajectories. Subsequently, it attempts to track the predictions (hence, the objects) via deep reinforcement learning, that takes advantage of the interactions between the agents and the environment to link object to predictions over successive frames.

**2.3 Occlusion handling**

As established in the reviews, it is clear that occlusion remains a persisting challenge of single and multi object tracking. Hence, various trackers are being proposed, with varying degrees of success on Single and Multi Object tracking benchmarks [[35-37](#_ENREF_35)].

Adaptive and correlative filters have also been used to a large extent in the quest to obtain efficient object tracking. [[9](#_ENREF_9), [10](#_ENREF_10), [38-40](#_ENREF_38)] have been profound in the use of correlation filters as highlighted in the performance. This approach trains the correlation filter with Fourier analysis to optimize computational requirement, hence, obtain computational efficiency. They worked on different extensions of the concept, and integrated other filters. In fact, Danelljan et al [[11](#_ENREF_11)] went as far as integrating deep convolution networks for feature representation, this further improved the performance of the correlation filter tracker, although, the deep network took a toll on the system run time.

Further research efforts towards occlusion handling considered particle filters and image processing tools; the tracking model developed by Sardari et al [[41](#_ENREF_41)] uses a heuristic approach in tracking detected objects over several frames, and when occlusions are detected, the search algorithm is modelled to increase its number of iterations in the next frames until the object re emerges. Cheong et al [[42](#_ENREF_42)] applied an image processing technique in which foreground objects are segmented, and a colour tracking algorithm is used to track the objects based on colour, and dimensions. However, this model is not robust enough to handle objects of same colour and/or similar dimensions, and was mostly applied to objects with plain colours and shapes. [[43](#_ENREF_43)] proposed a discriminative classifier which categorizes each image frame into one of being without occlusion, partially occluded, or fully occluded. It then processes the frames without occlusions with structured kernels, and estimates an occlusion threshold from each frame after extracting the target patch, the target distance and occlusion threshold are then used to discriminate the occlusion. [[44](#_ENREF_44), [45](#_ENREF_45)] both used moving cameras in their object tracking models, which are able to perform background estimations, and learn the image background from different viewing angles for possible occlusions.

As identified above, previous efforts towards tracking of single or multiple objects in crowded and occluded environments have been focused on the identification of new tracks as either newly detected objects on the scene, or re-emerging objects. This work focuses on the real – time prediction of a region for possible object re-emergence.

1. **Methodology**

The first phase of this research work implements a tracking-by-detection model in the tracking of a single moving object over successive frames in a video sequence. This model leverages on the feature representation strength of convolutional neural networks, the precision of detections using the Non-max suppression algorithm, and <my contribution>

* 1. **The YOLO algorithm**

The YOLO algorithm is a modified convolutional implementation of the sliding window method of detecting object location in an image which predicts multiple bounding boxes and class probabilities for these boxes[[46](#_ENREF_46)]. The sliding window involves defining a window size, and sliding it across an image of interest with the purpose of matching it to a region with interesting set of pixel values, which can be further investigated for the presence of possible objects or mere background. Depending on the stride used for the sliding, the system iterates over lots of pixels which do not contain significant details, hence, using up a lot of computational resources.

Without a prior knowledge of the content of images, it is not uncommon to have quite a number of sliding windows return different portions of the same image. To address this, this algorithm splits each image into grids and examines each grid once, for the presence of notable details. Where objects are detected, the window returns the dimensions of the identified object relative to the grid cell in which it was detected.

(3.1)

Every grid cell in which detection has been recorded, predicts bounding boxes for every object detected, and confidence score for those boxes. Grids without any detection are expected to have a confidence score of zero. The predicted bounding box is an array with 5 values: x, y, w, h, and the confidence score. The (x, y) coordinates represent the centre of the box, w is the width of the box, and h represents its height.

(3.2)

**3.1.1 Intersection over Union (IoU)**

In the object detection task, detected objects are expected to be localized, but window grid cells mostly do not fit images perfectly, and multiple objects could share the same detection. Hence, the IoU is a technique that calculate the ration between the intersection between two boxes, and the union of the two boxes. That is, objects detected are assigned to grid cells that have an IoU greater than a predefined threshold.

**3.1.2 Non – max suppression**

With a high chance of having multiple grids exceed the IoU threshold, especially when a fine definition of grid cells is used, the non-max suppression function identifies the bounding box with the highest confidence of detection, and suppresses every bounding box with a high IoU to the identified bounding box. This in turn, ensures that objects have just a single detection bounding box.

**3.1.3 Anchor boxes**

While it is common to have multiple grid cells detect the same object, it is not unusual to have a single grid return multiple detections. With the idea of anchor boxes, different anchor boxes of shapes corresponding to the expected detections are predefined. As a result, detected objects in a grid are compared to the defined anchor boxes, and assigned to the anchor box that has the higher IoU with the image ground truth information.

However, two issues persist with the use of anchor boxes for localizing detections; the first case occurs when the number of objects detected in a grid exceed the number of predefined anchor boxes, and the second case is when multiple object assigned to the same grid share a similar anchor box.

The YOLO architecture has 24 convolutional layers, followed by 2 fully connected layers. Each convolution layer is connected to a batch normalization layer, and activated using the leaky ReLU activation function.

**3.2 Batch Normalization**

Batch normalization is a pre – processing technique, which normalizes the input values for each layer of the deep learning architecture such that it has a zero mean and variance in other to speed up the back propagation process.

Batch statistics equation:

Batch mean (3.3)

Batch variance (3.4)

Normalization of layer inputs using the previously calculated batch statistics:

(3.5)

Scaling and shifting in order to obtain the output of the layer:

(3.6)

**3.3 Activation Function**

When building a neural network, one of the important decisions to be made is the choice of activation function to be used in the hidden and output layers. Earlier implementations of neural networks employed the sigmoid activation function which goes between 0 and 1. An alternative to the sigmoid function is the tanh function, which is a shifted version of the sigmoid function going between -1 and +1.

A major disadvantage of the sigmoid and the tanh activation functions is that it yields a small gradient when its input is either very large or very small (close to zero), which can make the training of a large neural network slow. The Rectified linear unit (ReLU) activation function was introduced to address the near zero gradient problem of the sigmoid and tanh function, but it also outputs a zero gradient when the input is less than zero. A modified version of the ReLU is the Leaky ReLU, which we will be using in this work.

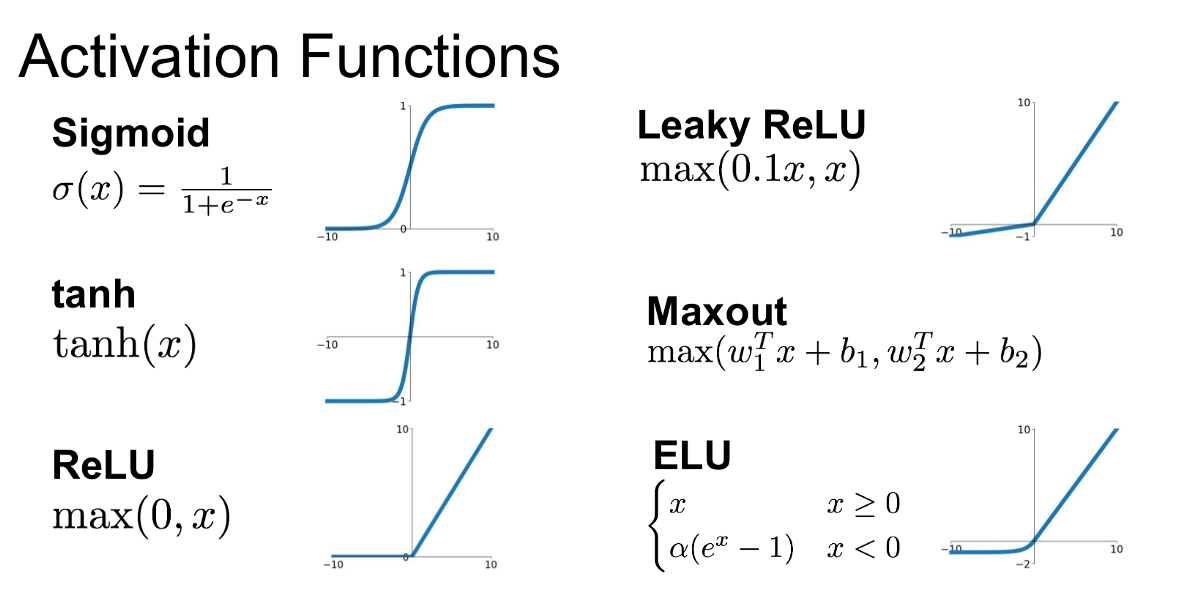


Fig 3.1: Commonly used activation functions

The choice of leaky ReLU as the activation function comes as a solution to the diminishing gradient problem – in which the gradient of a function is so close to zero that it slows down the optimization of a model – activation functions have when the network gets so large. Negative input values into the ReLU activation function are immediately turned into zero which could decrease the ability of the model to train and fit properly as the negative values could be influential (no matter how minimal) in the output. The leaky ReLU function addressed this issue by assigning a small non zero value to the input values of zero.

**3.4 Max Pooling**

Another technique introduced to the architecture is the max pooling function, which has proven useful in reducing the dimension of its input without hurting the performance of the network. The max pooling function outputs the maximum value of a number of given grid cells (depending on the intended dimension).

An alternative to the max pooling is the average pooling, which computes the average of the given grid values instead of the maximum value.

1. **Experiments**

**4.1 Training**

This work implements the YOLO object detection algorithm[[46](#_ENREF_46)], and trained the system on a number of object classes in the TinyTLP dataset[[47](#_ENREF_47)]. Test carried out showed good results on a few epochs, and gave a promise of better achievements if trained for longer period and with larger dataset. So weights are loaded from a pre – trained YOLO algorithm on the Microsoft Common Object in Context (COCO) 80 – class object detection dataset[[48](#_ENREF_48)]. This gave good improvements on the result.

The choice of the YOLO model is due to its precision and speed of detection which has – along with the R-CNN – a better performance when compared to other object detection algorithms; and considering the proposed model, precision and speed are essential in the pre – training phase of the work.

The YOLO model has a convolutional layer which is combined with batch normalization and Leaky ReLU layers. The output is then connected to a max pooling layer to reduce the dimensions to 32 The next layer follows a similar pattern, with the output nodes increasing, and the object dimensions reducing in height and width.

Being a supervised learning model, network parameters are trained by first generating training samples which consist of image patches, class labels, x and y coordinates of the image’s top left position, and the height and width of the image.

The TinyTLP dataset consists of 50 videos of single or multiple moving objects, with 600 frames per video, and also provides the ground truth position and size of a particular object of interest to be tracked. The COCO dataset

**4.2** **Results**

Below are the results obtained from carrying out a number of experiments using the YOLO algorithm:

**4.2.1. Detection**

Given that the YOLO algorithm is an object detection dataset, the first task in this work was to ensure its functioning; hence, the algorithm was implemented to detect objects in the TinyTLP test set.

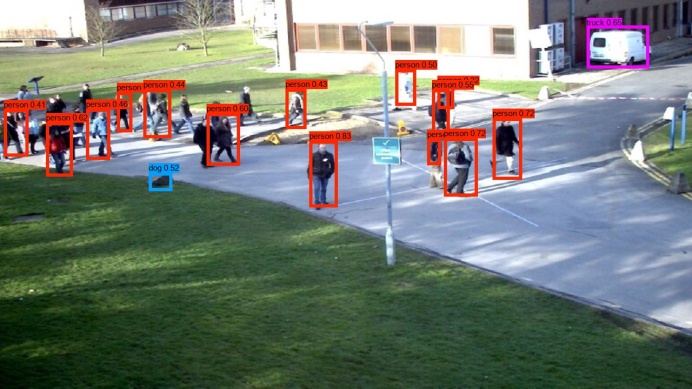
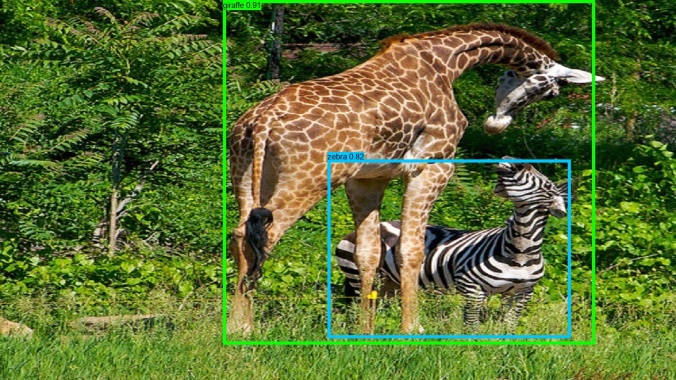
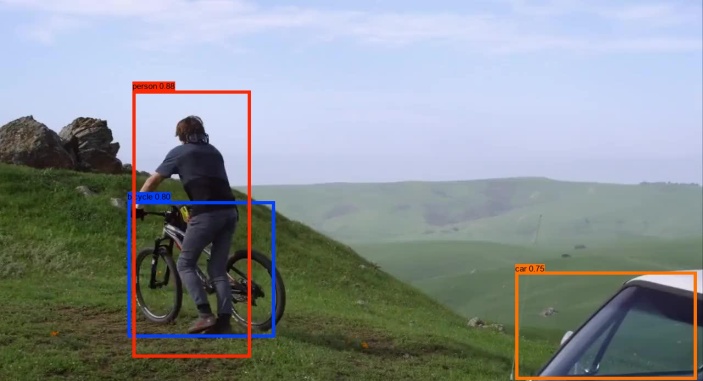
 

Fig. 4.1: Qualitative results on natural images from the 3DMOT2015 and the TinyTLP datasets. Its detections are mostly accurate, although it misses a number of persons in the first test image.

**4.2.2. Tracking**

Following the success in the object detection, the model was converted to perform tracking. In doing this, the fully connected (fc) layer which was incorporated to predict objects within 80 predefined classes was changed to a single class fc layer which only returns the presence of a detected object (or lack of it), and the dimensions of the localized detection (x and y coordinates of the top left position, the width, and the height of the detected object).

Also, the IoU threshold which estimates the probability that a detected object belongs to a grid cell or another was reduced to 0.2 to accommodate more detections since objects could be partially occluded by other objects, while the score threshold which is the measure of confidence that a detected object actually belongs to the class for which it was predicted was increased to 0.6 so as to get rid of false classifications.

As a result of this, it was possible to effectively track a single object over successive frames in a video sequence with minimal overlap by other objects it interacts with.

Fig. 4.2: Single object tracking on the Car chase test set from the TinyTLP dataset

**4.2.3. Noise**

Most of the test datasets available over the internet have been coupled for specific purposes, and hence, are usually fit to size with little or no additive noise present. Since the proposed object tracking model will be used as a real time application on video sequences obtained from a widely varying camera models, some form of noise was added to the test set to observe how the model would react to the change.

Fig 4.3: Tracking results on the same image with different noise types

Top: Gaussian, poisson, Bottom: salt and pepper, and speckle noise (From L – R)

1. **Conclusion**

The YOLO algorithm has proven to be a reliable model for object detection, and modifications made in this work has shown that its capabilities can be extended to object tracking for instances where the object remains visible.

The algorithm has been used largely on controlled test sets with reduced or no noise, but when ran on noisy images, there has been a significant drift in the localization of detected objects especially when the speckle and salt and pepper noise are introduced. To correct this, we could either employ data augmentation in the training phase, whereby the dataset for training is duplicated, but with different noise functions added while maintaining the same class labels. Alternatively, a de – noising function can be introduced, which would detect the presence of noise, classify the noise type, and filter the image before tracking.

Future work will also include a prediction function for estimating the object’s likely position in the next frame given a number of previous detected frames. To achieve this, we hope to use reinforcement learning, as this is a more flexible model, and mimics the human’s ability to work in an unsupervised environment [[49](#_ENREF_49)] as opposed to using a filter model which has also been shown to give good results in object tracking[[50-52](#_ENREF_50)].

References

1. Jeon, S., B. Hong, and V. Chang, *Pattern graph tracking-based stock price prediction using big data.* Future Generation Computer Systems, 2018. **80**: p. 171-187.

2. Ojha, S. and S. Sakhare. *Image processing techniques for object tracking in video surveillance-A survey*. in *Pervasive Computing (ICPC), 2015 International Conference on*. 2015. IEEE.

3. Perera, A.A., et al. *Multi-object tracking through simultaneous long occlusions and split-merge conditions*. in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. 2006. IEEE.

4. Wang, L., et al. *Visual tracking with fully convolutional networks*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

5. Son, J., et al. *Multi-object tracking with quadruplet convolutional neural networks*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

6. Hong, S., et al. *Online tracking by learning discriminative saliency map with convolutional neural network*. in *International Conference on Machine Learning*. 2015.

7. Kim, D.Y. and M. Jeon, *Data fusion of radar and image measurements for multi-object tracking via Kalman filtering.* Information Sciences, 2014. **278**: p. 641-652.

8. Hamid, K.R., A. Talukder, and A.E. Islam, *Implementation of Fuzzy Aided Kalman Filter for Tracking a Moving Object in Two-Dimensional Space.* International Journal of Fuzzy Logic and Intelligent Systems, 2018. **18**(2): p. 85-96.

9. Henriques, J.F., et al., *High-speed tracking with kernelized correlation filters.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015. **37**(3): p. 583-596.

10. Bolme, D.S., et al. *Visual object tracking using adaptive correlation filters*. in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010. IEEE.

11. Danelljan, M., et al. *Beyond correlation filters: Learning continuous convolution operators for visual tracking*. in *European Conference on Computer Vision*. 2016. Springer.

12. Danelljan, M., et al. *Convolutional features for correlation filter based visual tracking*. in *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015.

13. Wang, X., et al., *A survey of recent advances in particle filters and remaining challenges for multitarget tracking.* Sensors, 2017. **17**(12): p. 2707.

14. Fiaz, M., A. Mahmood, and S.K. Jung, *Tracking Noisy Targets: A Review of Recent Object Tracking Approaches.* arXiv preprint arXiv:1802.03098, 2018.

15. Yang, H., et al., *Recent advances and trends in visual tracking: A review.* Neurocomputing, 2011. **74**(18): p. 3823-3831.

16. Yazdi, M. and T. Bouwmans, *New trends on moving object detection in video images captured by a moving camera: A survey.* Computer Science Review, 2018. **28**: p. 157-177.

17. Smeulders, A.W., et al., *Visual tracking: An experimental survey.* IEEE Transactions on Pattern Analysis & Machine Intelligence, 2013(1): p. 1.

18. Babenko, B., M.-H. Yang, and S. Belongie, *Robust object tracking with online multiple instance learning.* IEEE transactions on pattern analysis and machine intelligence, 2011. **33**(8): p. 1619-1632.

19. Kalal, Z., K. Mikolajczyk, and J. Matas, *Tracking-learning-detection.* IEEE transactions on pattern analysis and machine intelligence, 2012. **34**(7): p. 1409.

20. Grabner, H., C. Leistner, and H. Bischof. *Semi-supervised on-line boosting for robust tracking*. in *European conference on computer vision*. 2008. Springer.

21. Grabner, H., M. Grabner, and H. Bischof. *Real-time tracking via on-line boosting*. in *Bmvc*. 2006.

22. Chetlur, S., et al., *cudnn: Efficient primitives for deep learning.* arXiv preprint arXiv:1410.0759, 2014.

23. Otterness, N., et al. *An evaluation of the NVIDIA TX1 for supporting real-time computer-vision workloads*. in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE*. 2017. IEEE.

24. Gawande, N.A., et al., *Scaling deep learning workloads: NVIDIA DGX-1/Pascal and intel knights landing.* Future Generation Computer Systems, 2018.

25. Held, D., S. Thrun, and S. Savarese. *Learning to track at 100 fps with deep regression networks*. in *European Conference on Computer Vision*. 2016. Springer.

26. Nam, H. and B. Han. *Learning multi-domain convolutional neural networks for visual tracking*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

27. Wang, N., et al., *Transferring rich feature hierarchies for robust visual tracking.* arXiv preprint arXiv:1501.04587, 2015.

28. Sutton, R.S., A.G. Barto, and F. Bach, *Reinforcement learning: An introduction*. 1998: MIT press.

29. Ba, J., V. Mnih, and K. Kavukcuoglu, *Multiple object recognition with visual attention.* arXiv preprint arXiv:1412.7755, 2014.

30. Mnih, V., N. Heess, and A. Graves. *Recurrent models of visual attention*. in *Advances in neural information processing systems*. 2014.

31. Zhang, D., et al., *Deep reinforcement learning for visual object tracking in videos.* arXiv preprint arXiv:1701.08936, 2017.

32. Silver, D., et al., *Mastering the game of Go with deep neural networks and tree search.* nature, 2016. **529**(7587): p. 484.

33. Yoo, S., et al. *Action-decision networks for visual tracking with deep reinforcement learning*. 2017. CVPR.

34. Ren, L., et al. *Collaborative Deep Reinforcement Learning for Multi-object Tracking*. in *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

35. Valmadre, J., et al., *Long-term Tracking in the Wild: A Benchmark.* arXiv preprint arXiv:1803.09502, 2018.

36. Milan, A., et al., *MOT16: A benchmark for multi-object tracking.* arXiv preprint arXiv:1603.00831, 2016.

37. Leal-Taixé, L., et al., *Motchallenge 2015: Towards a benchmark for multi-target tracking.* arXiv preprint arXiv:1504.01942, 2015.

38. Choi, J., et al. *Visual tracking using attention-modulated disintegration and integration*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

39. Danelljan, M., et al. *Accurate scale estimation for robust visual tracking*. in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. 2014. BMVA Press.

40. Hong, Z., et al. *Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

41. Sardari, F. and M.E. Moghaddam, *A hybrid occlusion free object tracking method using particle filter and modified galaxy based search meta-heuristic algorithm.* Applied Soft Computing, 2017. **50**: p. 280-299.

42. Cheong, Y.Z. and W.J. Chew. *The Application of Image Processing to Solve Occlusion Issue in Object Tracking*. in *MATEC Web of Conferences*. 2018. EDP Sciences.

43. Dong, X., et al., *Occlusion-aware real-time object tracking.* IEEE Transactions on Multimedia, 2017. **19**(4): p. 763-771.

44. Zhao, S., S. Zhang, and L. Zhang, *Towards Occlusion Handling: Object Tracking With Background Estimation.* IEEE transactions on cybernetics, 2017.

45. Hou, L., et al., *Robust human tracking based on DPM constrained multiple-kernel from a moving camera.* Journal of Signal Processing Systems, 2017. **86**(1): p. 27-39.

46. Redmon, J., et al. *You only look once: Unified, real-time object detection*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

47. Moudgil, A. and V. Gandhi, *Long-term visual object tracking benchmark.* arXiv preprint arXiv:1712.01358, 2017.

48. Lin, T.-Y., et al. *Microsoft coco: Common objects in context*. in *European conference on computer vision*. 2014. Springer.

49. Kong, X., et al. *Collaborative deep reinforcement learning for joint object search*. in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

50. Liu, C., P. Shui, and S. Li, *Unscented extended Kalman filter for target tracking.* Journal of Systems Engineering and Electronics, 2011. **22**(2): p. 188-192.

51. Xu, S. and A. Chang, *Robust object tracking using Kalman filters with dynamic covariance.* Cornell University, 2014: p. 1-5.

52. Jatoth, R.K., S. Gopisetty, and M. Hussain, *Performance analysis of alpha beta filter, kalman filter and meanshift for object tracking in video sequences.* International Journal of Image, Graphics and Signal Processing, 2015. **7**(3): p. 24.