RHODES UNIVERSITY
Where leaders learn

LITERATURE REVIEW OF THE TRACKING AND DETECTION OF HUMAN FACIAL FEATURES AND PROJECT REVISION

J L GOUWS
Supervisor: MR. J CONNAN
Computer Science Department, Rhodes University

May 3, 2022

Abstract

Tracking objects in video streams is a powerful tool in computer vision. This research will investigate the recognition and tracking of multiple faces in settings where the faces are concurrently visible. The study will implement a tracking system that after, initialising with minimal input data, detects faces and tracks their motion in a video stream. The task of tracking starts by initialising the tracker with bounding boxes that define the faces of the targets in images. The initialised tracker will be able to detect if a target face is present in or absent from an arbitrary video stream. The tracker will identify any visible target and follow its motion. If a target face disappears and later reappears in the video stream, the tracker will be able to identify and track it again, provided the target remains in the field of view.

1 Introduction

This document is a literature review for the associated Honours project. This paper also serves to revise the proposal. Below is an amended introduction to the project.

Consider a continuous video stream where a set of faces appears, and each face might appear at different times. The individual faces might move and change orientation in the video stream. Imagine that some subset of these faces is of interest–the target faces. A set of pictures of these faces must exist; these pictures may be unrelated to the video or frames within it. This research aims to develop a system that solves the problem of automatic identification and tracking of target faces in the video stream.

The initial input given to the system is images that have uniquely labelled regions of interest or bounding boxes which define individual faces. The input images must contain at least one instance of each target face, but there is no maximum limit. This process constitutes the initialisation of operation, after which the system functions autonomously.

Given an arbitrary video, the system searches for the presence of any target faces. The tracker proceeds to label detected target faces with the label provided in the initialisation of the system. Once a target face is labelled, the tracker follows its motion, and all other targets appearing in the current frame. This process constitutes the running phase of the system, where the system identifies and tracks faces in the supplied video stream.

While the system is running, it determines information about the target faces. It can, hence, extract the number of times each target face appears, the amount of time for which each target appears and the trajectory of each face while it is apparent in the video. This process is the output stage, which concludes the system's operation.

The system operates with minimal input data supplied in the initialisation stage. With this constraint, the system should use all the data it can access. Thus, the system uses the video to learn more about each target face in the running phase. In this way, the system can identify and track faces with better accuracy as the video progresses.

The next section of this document reviews the formal research statement. Following this, three sections introduce works that are related to this research. This section serves as the literature review of the project. The literature review is followed by a section discussing the project's methodology. Finally, the conclusion summarises the literature review findings and details of the project.

2 Reseach Statement

This section revises of the problem statement stated in the project proposal. The problem statement has been restated for clarity, but the content of the statement is the same.

- The design and implementation of a long-term tracking system–given minimal input data, the system can count the number and measure the duration of appearances of multiple target faces in a single video stream–can use machine learning and computer vision techniques.

The central computer vision technique that is tested is the Tracking, Learning, and Detection framework–discussed in Section 5.4. This framework allows for long-term tracking with minimal input data.

Long-term tracking(LT) is tracking of objects that can undergo partial oclusions, change appearance, and disappear and reappear from the field of view. This is opposed to short-term tracking(ST) where the object remains fully in the field of view for the whole duration of tracking. ST can be used as a basis for LT, as is done in the case of TLD.

The definition of minimal input data, in the context of this research, is a single image for each face that is required to be tracked, where each image includes at least one bounding box. The bounding box defines the location of the face in the image and a label for the face. The goal of this research is to implement a working system that meets the conditions specified by the Research Statement.

3 Facial Recognition

Facial recognition is a standard element of computer vision with numerous applications. The goal of facial recognition is to label a face that is present in an image.

3.1 Eigenfaces

Turk et al. [29] describe how to use principle component analysis(PCA) to determine features for facial recognition. PCA is used to determine eigenvectors, referred to as "eigenfaces", that form a basis for the faces of concern. Any face in a given set of faces can be decomposed into a linear combination of these basis vectors, as an example, this is equivalent to mathematical eigenvectors in a Euclidean space. The compononents of the decomposition can be used to recognize faces.

This usage of eigenfaces allows for a more compact representation of a face. PCA finds a set of features that account for the largest amount of variation in some set of faces. This allowed Turk et al. to acheive a method for recognizing faces that is "fast, relatively simple, and has been shown to work well in a constrained environment [29]."

Bartlett et al. [1] suggest the use of Independent Component Analysis(ICA) instead of PCA. ICA is a generalization of PCA that takes the relationship of distant pixels into account. This allows ICA to encode more information, in comparison to PCA, in the eigenvectors.

A machine learning model can be trained on this set of features in order to identify a given face. One option is to use a neural network which offers high accuracy and quick recognition [29, 1]. Another option is to use a naive bayes classifier which is amenable to online learning [22].

3.2 Multi-Pose Face Recognition

In a realistic video stream it is atypical for all the faces to be facing the camera at a given time. The faces might change pose from frontal to profile. It is thus key for a detector to recognize faces that both from a frontal and portrait pose, if it is going to used on real world data.

Pentland et al. [26] suggest two solutions to the problem of Multi-Pose Face Recognition using eigenfaces. First, a single high dimensional eigenface space can be used. In this face-space a basis vector contains information about the face and its orientation. Second, different face-spaces can be used for different orientations. Each face-space is defined by using PCA on images of all the faces taken a given viewpoint, for example 10 degrees left of frontal view.

Nair et al. [23] describe ways to recognise and track a face that takes on multiple poses. The system proposed by Nair et al. has three components: Haar Cascades based face detection, weighted modular PCA based face recognition and Kalman tracker.

4 Learning

Consider a continious video stream that is being analyzed by a pre-trained tracking system. This video stream is a constant flux of information, and some of the information might not have been encountered the tracking system's history. The information contained in the video stream can, thus, be used to improve the tracking system. Improving the system requires learning mechanisms, specifically online learning–learning where information becomes available while the system is in operation.

4.1 Online Learning

Online learning is learning done as data becomes available. This form of training machine learning models is required to handle streaming data, for example stock-prices and videos. Online learning can also be used to train models offline, for example when a dataset is too big to load into a computer's memory.

One option for training a classifier online is Stochastic Gradient Descent(SGD). SGD aims to minimize a cost function:

$$J(\boldsymbol{\theta}) = J^* \left(f(\boldsymbol{\theta}, \boldsymbol{z}_1), f(\boldsymbol{\theta}, \boldsymbol{z}_2), \dots, f(\boldsymbol{\theta}, \mathbf{z}_N) \right))$$

by changing the model parameters $\boldsymbol{\theta}$ [22]. The $f(\boldsymbol{\theta}, \boldsymbol{z}_i)$ are functions that give the cost of some data point $\boldsymbol{z}_i$ for the parameters $\boldsymbol{\theta}$ The parameters are updated in sequential steps of the form:

$$\boldsymbol{\theta}_{k+1} = G(\boldsymbol{\theta}_k, \nabla f(\boldsymbol{\theta}, z_k))$$

The gradient descent can be implemented in many ways, and can be improved with adaptive step sizes [6].

Another option for an online learning classifier uses Bayesian inference. The posterior probability of the Bayesian classifier can be updated according to:

$$p(\boldsymbol{\theta} | \mathscr{D}_{1:k}) \propto p(\mathscr{D}_k | \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathscr{D}_{1:k-1})$$

Where $\mathscr{D}_{1:n}$ indicates that the classifier has been given data points 1 to n. Bayes classifiers, in most cases, can be trained faster than classifiers using SGD [22].

4.2 Reinforcement Learning

Schrittwieser et al. [28] propose a new online reinforcement learning method that can be used to train models with minimal input data. Schrittwieser et al. describe the *Reanalyse* algorithm. Given a state of a machine learning model the *Reanalyse* algorithm generates training targets for the model from some input data. When the model has improved by training, the *Reanalyse* algorithm generates more training targets based on the new state of the model, the already seen input data, and any new input data. The algorithm allows the available training data to be cycled–this allows the algorithm to extract most of the information from a limited dataset.

4.3 Updating Template Trackers

Template tracking [21] assumes that the appearace of the target object does not undergo changes. The results in simplistic tracking and the tracker will fail if the target undergoes a change in orientation or a change of view. Matthews et al. propose solutions to this, and discuss the problems around the solutions. The problems are a result of what is known as the stability-plasticity dilema [9].

Everytime a tracker template is updated, some error in the template is introduced. This causes the tracker to drift, and evetually cause the tracker to fail [21].

Suppose that $\mathbf{x}$ is the coordinate vector of a pixel in the n^{th} frame $I_n(\mathbf{x})$ of a video. Let $T(\mathbf{x})$ be the template of the target image, and $T_n(\mathbf{x})$ be the template of the object in the n^{th} frame of a video sequence. The warp of the image $\mathbf{W}(\mathbf{x};\mathbf{p})$ represents the allowed deformations of the template given a set of parameters $\mathbf{p}$ which define a deformation. The warp maps a pixel from the template frame to the coordinates of the video frame $I_n(\mathbf{x})$.

Given these defintions, the problem of tracking formally reduces to computing the parameters for the deformation of the object:

$$\mathbf{p}_n = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in T_n} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x})]^2 \tag{1}$$

And then updating the tracking template based on the warp of the n^{th} frame, for example a naive update is [21]:

$$\forall n \geq 1, T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$$

Implementing this requires a gradient descent algorithm for non-linear optimizations. Equation 1 now becomes:

$$\mathbf{p}_n^* = \mathrm{gd}\min_{\mathbf{p}=\mathbf{p}_{n-1}} \sum_{\mathbf{x}\in T_n} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x})]^2 \tag{2}$$

With $\mathrm{gd}\min_{\mathbf{p}}$ indicating a gradient descent minimization starting from the warp parameters of the $(n-1)^{\text{th}}$ frame.

Using Equation 2, Matthews et al. suggest a template update with drift correction given by:

$$\begin{array}{l} \text{If } \|\mathbf{p_n}^* - \mathbf{p_n}\| \leq \varepsilon \text{ Then } T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*)) \\ \text{else } T_{n+1}(\mathbf{x}) = T_n(\mathbf{x}) \end{array}$$

Given some small threshold $\varepsilon > 0$. This updates the template if retaining the template would cause tracker drift, otherwise the template is not updated.

4.4 P-N Learning

Training a tracker on a video stream is, in effect, bootstrapping a classifier online. Kalal et al. [13] offers a solution to training a binary classifier in a stable way. It is important that the tracker's learning is stable, so that tracker does not drift as the video progresses.

P-N learning consists of two components that evaluate the errors of the classifier every instant that new data become available, for example every frame of a video. The first component is the P-expert which attempts to recognize the false negatives that classifier makes. The second component is the N-expert which attempts to recognize false positives of the classifier. These components make errors, otherwise the components would make a perfect classifer by themselves. The errors of one component, however, compensate for the errors of the other component leading to stable learning [14].

The P-N learning system can be modelled as a two dimensional dynamical system [13] [14]. This model can be used to determine the stability of the learning.

5 Tracking

Tracking is a versatile form of computer vision that is present in almost all forms of video analysis. The task of tracking is to determine the trajectory of an object in a video stream.

Tracking is typically broken into two categories: short-term and long-term. Long-term tracking(LT) is tracking of objects that can undergo partial oclusions, change appearance, and disappear and reappear from the field of view. This is opposed to short-term tracking(ST) where the object remains fully in the field of view for the whole duration of tracking. ST can be used as a basis for LT, as is done in the case of TLD.

5.1 General Comparison of Trackers

The Visual Object Tracking Challenge(VOT) is a challenge that benchmarks various trackers every year [18] [16]. VOT investigates both ST and LT. In recent years, VOT has also introduced a real time challenge [16].

In the VOT 2020 challenges the majority of trackers use deep features, instead of hand-picked features which were predominant in the past. In the short term tracking challenge 86 % of trackers were seen to use deep features. Deep discriminative correlation filters(DCF) have seen a lot of recent success [5], 15 of short-term trackers in VOT2020 also used deep DCF.

All of the long-term trackers in VOT 2020 used convolutional neural networks. Several of trackers seen in the long term challenge are based on MDNet [24] to detect for object presence. One of the long-term trackers was based on deep DCF, and another used siamese convolutional neural networks.

5.2 Convolutional Neural Networks

Convolutional Neural networks(CNN) have been prominent in the field of Computer vision in recent years. CNNs have been used in field of computer vision in order to classify images with high accuracy and speed [27]. CNNs come at the cost of requiring long times to train, in most cases using GPUs [19]. This makes standard use of CNNs impractical for tracking unknown objects, training a CNN online results in impairment of the system's speed [2].

5.2.1 Crowd Segmentation with CNNs

A crucial part of a tracking is the ability to distinguish background from objects of interest. Kang et al. [15] propose fast fully convolutional neural networks(FCNN) to segment crowds. The FCNN allows for a whole image to be searched in one pass of forward propagration. This allows faster and more accurate segmentation in comparison to sliding window techniques.

The FCNN has no fully connected layers in the CNN, instead it uses a 1×1 convolution kernel layer to predict labels. This removes the translation dependence of CNNs that is due to the fully connected layers. The translation invariance of the network allow the single pass of the image.

This technique uses appearance to segment the crowd–this reduces the false positives which occur when motion cues are used to segment crowds. Another advantage of scanning the whole input image once is the availability of more contextual appearance information which can be used in segmentation [29, 7]. Once an image of a crowd is segmented, more specific analiyisis can be done, for example identification of people in the crowd.

5.2.2 Multi-domain CNN

Nam et al. [24] use a CNN as the basis of a tracker(MDNet) that they implement. The CNN is trained on a large, labelled dataset of videos to obtain generic target representations. Training is done on one separate domain, cars for example, at a time. All the domains are worked through in an iterative process. This provides a CNN that can distinguish between a target, from any of the trained domains, and background.

The output of this CNN is branches into input for domain-specific layers, see Figure 1. These output layers are binary classifiers that use the CNN's generic representation to track a target in a specific domain. These layers are trained online using the video stream as input, in order to improve the tracker's accuracy as the video progresses.

This design allowed Nam et al. to get very high accuracy in VOT challenges. The high accuracy comes at the cost of real time execution–the binary classifers of the network require a

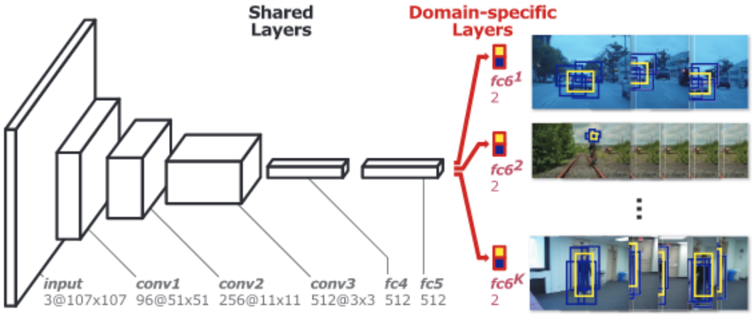

Figure 1: The architecture of the Multi-Domain Network, from [24]

lot of time to train online. MDNet was only able to process one frame per second in the VOT challenges [2], this makes in unsuitable for real-time tracking.

5.2.3 Convolutional Siamese Neural Networks

Bertinetto et al. [2] offer a solution to the problem of tracking an unknown object with Convolutional Neural networks. The solution given by Bertinetto et al. is to train, offline, a CNN that solves the more general similarity problem. A similartiy function $f(z, x)$ is learned, the funtion compares images x and z and returns values that estimate how similar the images are. This function can be considered as a composition of two other functions $f(z, x) = g(\phi(z), \phi(x))$. In this composition, ϕ is an embedding that does the convolution and g is a metric. A siamese neural network, two identical neural networks conjoined at the output node [4], is used to learn the similarity relation.

A *fully-convolutional*, translation invariant, siamese network is suggested by Bertinetto et al. The input to the network is an image centered on the target's previous location. The network outputs a score map represented as a grid of numbers, Figure 2.

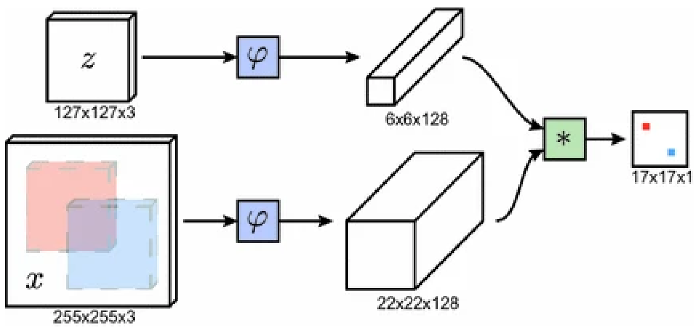

Figure 2: The architecture of the Fully Convolutional Siamese Neural Network, from [2]

This architecture can be seen as an advanced template tracker. The CNN is equivalent to the preprocessing needed to create a tracking template. The metric g compares the template and a search window–the sum of squares metric is often used in template trackers.

5.3 Tracking with Correlation Filters

Henriques et al. [11] propose Kernelized Correlation Filters(KCF) and the novel Dual Correlaion filter(DCF). Both KCF and DCF use circulant matrices and the kernel Trick. The implementation of KCF by Henriques et al. uses a Gaussian Kernel, whereas the DCF implementation uses a

Algorithm	feature	Mean precision	Mean FPS
KCF	HOG	73.2%	172
DCF	HOG	72.8%	292
KCF	Raw pixels	56.0%	154
DCF	Raw pixels	45.1%	278
TLD		60.8%	28
Struck[10]		65.6%	20
MOSSE[3]		43.1%	615

Table 1: Comparison of various trackers, adapted from [11]

linear kernel. The calculations involved with the linear kernel are less computationally complex than KCF. DCF can, hence, be processed faster, but, at the cost of some tracking precision.

Work by Galoogahi et al. [8] allows KCF and DCF to be applied to modern and useful feature descriptors. Henriques et al. show that KCF and DCF be be applied to Histogram of Oriented Gradient(HOG) features to track and detect objects in a video stream with lower computation times and better accuracy. KCF and DCF applied to HOG features are shown to outperfrom many tracking systems Table 1. The results shown by Table 1 are obtained from running the algorithms on a standard four core desktop processor from 2014.

The system implemented by Henriques et al. does not, however, encorporate a failure recovery mechanism–section 8 of [11]. In other words Henriques et al. only explore KCF in the domain of ST. This is in contrast to the original TLD system which provides a failure recovery mechanism in the detection component [14]. The ST using KCF and DCF done by Henriques et al. can be used in a TLD framework for LT.

Ma et al. [20] investigate the problem of single object LT using correlation tracking. Ma et al. use two Gaussian ridge regression [22] models for tracking. One model uses the relative change in background and target as time progresses, the other model tracks by using the target's appearance. The first model is used to track the object's trajectory through fast motion and occlusions, and the second is used for scale change. Using both tracking models they train an online detector that is both flexible(from first tracker model) and stable(from second tracker model).

Ma et al. train a random fern classifier [25, 14] online in order to handle tracker failure. This solves the LT problem in a similar way to Kalal [12].

5.4 Tracking, Learning, Detection

In 2011 Kalal et al. invented the Tracking, Learning and Detection(TLD) framework for the longterm tracking of objects in a video stream. Kalal's original implementation uses a median flow tracker, P-N learning, and a random forrest and nearest neighbour based detector [12]. These three components give the respective tracking, learning and detection components of the system.

The learning compenent of TLD forms the backbone of the system, governing the interaction between the detector and tracker. The three components exchange information as shown in Figure 3, this allows the tracker to improve it's performance as time progresses [14]. By the nature of TLD, online learning is required for the learning component. Kalal developed the P-N Learning paradigm [13], a semi-supervised bootstrapping model [22], tailored to the needs of TLD.

The tracker of TLD outputs a bounding box for the target object in every frame. A second bounding box for the target object is also produced by the detector. The P-experts and N-experts

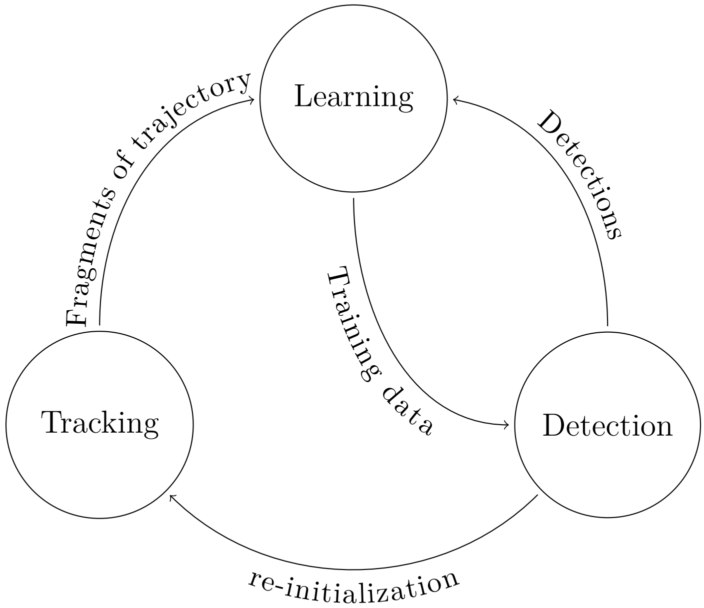

Figure 3: The interaction between tracking, learning and detection in TLD. Figure from [14]

of the learning component use these bounding boxes to determine the false positives and the false negatives of the detector. This is used to update the detector, as descibed in Section 4.4. An integrator is then used to combine the bounding boxes given by the tracker and detector [14].

6 Methodology

6.1 Methodology Overview

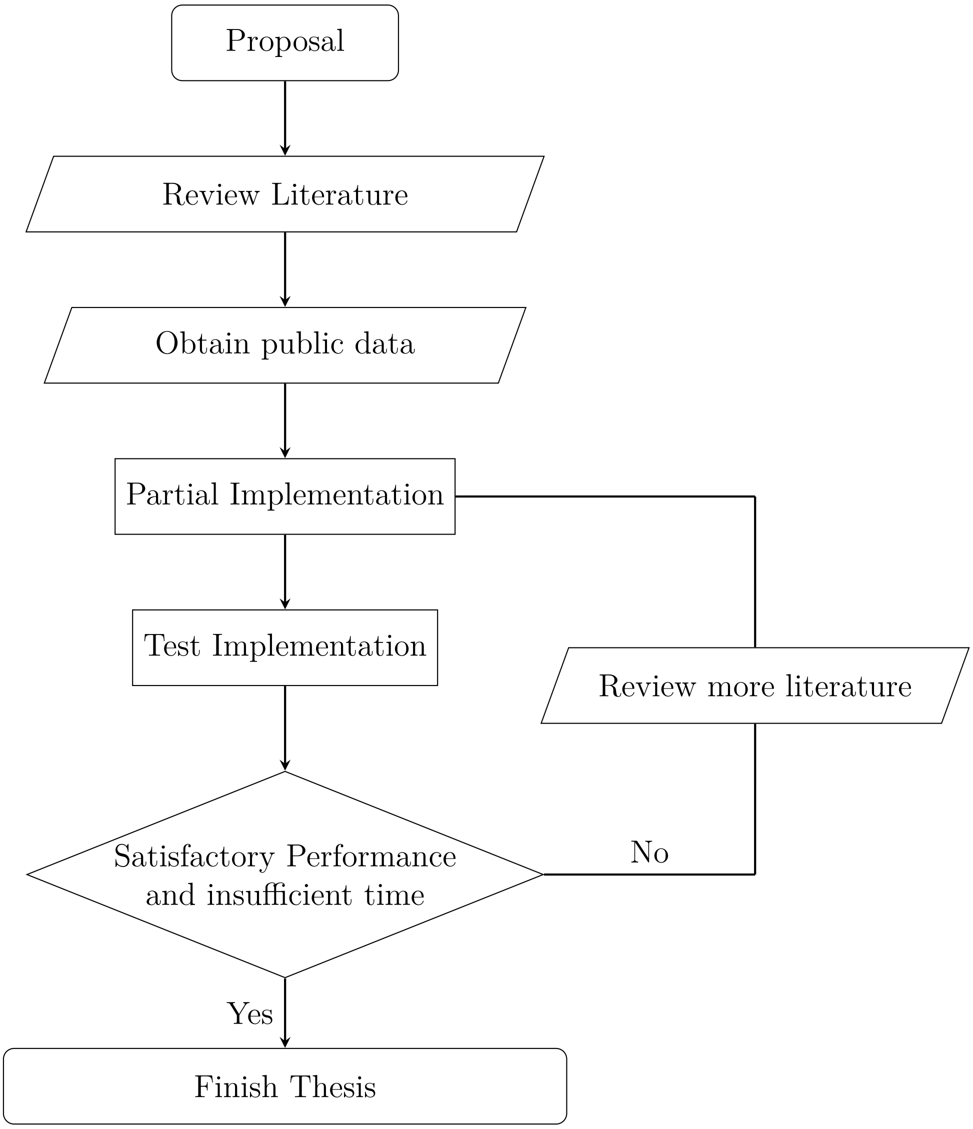

Figure 4: Conceptual overview of design methodology

6.2 Approach to Research

The first phase of this research will consist of in-depth reading of literature and further literature reviews. The literature reviews will start by reviewing Kalal's work on TLD. After a thorough review of Kalal's work, works pertaining to other trackers will be reviewed. Following this, there will be a further review of the literature discussing the online training of classifiers. The implementation of the system requires data for testing–a brief phase of data acquisition from public sources will provide data for developmental and testing purposes. This should suffice for the literature review and data collection.

The second stage of this research will relate to the practicalities of the system's implementation. The system will be implemented in C++, this requires proficient understanding of the C++ language. Investigation of the openCV C++ library will be done, and the available utilities will be surveyed.

The third phase consists of a functional reimplementation of the original TLD. Testing of this base system is required at this point, so that problems do not occur in later stages of the full system implementation. Following satisfactory performance of the TLD reimplementation, the next stage of research will commence.

At this point further improvements to the base TLD model will be made. The focus of the improvement will be on the tracking and detection components of the system. This involves either keeping the base model and improving the individual components or restructuring the model to improve model performance. This will constitute the fourth stage of the research.

Following this, an investigation of extending the system to track multiple object simultaneously will be made. There are naive ways to implement a multiple object tracking system–for example creating many different single target trackers to detect and track each object. This stage, therefore, requires significant planning, research and reviewing the current implementation in order to acheive good model performance and efficiency. This will complete the implementation of the system.

The final stage of this research involves three things. First, a full review of the implementation will be carried out. If improvements to the implementation are required and time permits, a return to stage four will be made. Second is a testing phase, where the complete system will be tested with videos obtained from the initial phase of the research. Third, a thesis will give a description of the implementation and specifications of the system. This completes the research project.

6.3 Timeline

The timeline for this project has been revised since the Proposal version. Most of the implementation deadlines now fall in the June and July Holiday, this is on account of exams and busy term times.

Time	Deliverable
30 March 2022	Seminar 1: Presentation of project
11 April 2022	Draft proposal
19 April 2022	Final proposal
6 May 2022	Literature review
20 May 2022	Obtaining suitable public videos
28 May 2022	Functional re-implementation of TLD
28 June 2022	Using KCF as Tracking stage of tracker
30 June 2022	Implementing DCF and comparing to KCF
6 July 2022	Reviewing VOT for better trackers
11 July 2022	Investingating random fern detectors
11-13 July 2022	Seminar 2: Progress Presentation
25 July 2022	Final decision on Tracking and Detection stages.
12 August 2022	Extension of system to multiple targets
19 August 2022	Progress Report
26 August 2022	Test and make small improvements the system
3 October 2022	First Draft of thesis
10 Octorber 2022	Completion of implementation
14 October 2022	Short ACM-style paper
17-19 October 2022	Seminar 3: Final Oral presentation
28 October 2022	Final project submission

7 Conclusion

7.1 Conclusion of the Literature Review

The Visual Object Tracking Challenge(VOT) ranks many current tracking systems every year [18, 17]. The challenge ranks trackers in different categories, the two main categories are long-term and short-term tracking. VOT ranks the trackers based on various metrics that indicate the accuracy of the tracker.

The VOT 2017 challenge introduced a real-time tracking challenge, which uses the short-term tracking dataset and performance measures [17]. VOT does not, however, require long-term trackers to run at real-time frame rates, some trackers run at less than one frame a second. This means that many of the tracking systems ranked in VOT are not amenable to this research.

There are, at present, two main approaches to solve the problem of long-term tracking in real time. The first approach uses correlation tracking and training a classifier online [20, 11, 14]. These methods require little set-up time, and can track single unkown objects in a video stream.

Most modern correlation trackers, seen in VOT, use deep features and neural networks. Correlation trackers dominate the VOT short term challenge, and are also submitted for the long-term challenge.

The second approach uses CNNs [2]. This approach requires an offline training stage, but the offline training stage need not be repeated. CNNs allow for high accuracy tracking [24, 2], and provided that online learning is not required, the trackers can process videos at high frame rates [17].

From a user's point of view, the operation of the CNN approach and the correlation approach seem identical. After the CNN has undergone offline learning, the CNN tracking systems requires a single image with a bounding box to track an unkown object [24, 2].

7.2 Revised Project Conclusion

This research uses the Tracking, Learning and Detection(TLD) framework to implement a system that can track and detect multiple faces simultaneously. TLD is used in the research to develop a long-term tracker that can recognize human faces and follow the trajectory of the faces in a video stream.

The system that the research implements is tested on public data. Testing is done by having the system extract information about faces that appear in a given video stream. The information collected by the implemented system is the number and duration of appearances for each target face.

The research is limited to the tracking of faces, and is not aimed at tracking general objects or other human features. Owing to the time limitations of an honours project, this project has substantial dependence on research done by other people and available tools.

References

[1] M.S. Bartlett et al. 2002. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13, 6, 1450–1464. DOI: 10.1109/TNN.2002.804287.

[2] Luca Bertinetto et al. 2016. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*. Springer, 850–865.

[3] David S. Bolme et al. 2010. Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2544–2550.

[4] Jane Bromley et al. 1993. Signature verification using a ”siamese” time delay neural network. *Advances in neural information processing systems*, 6.

[5] Martin Danelljan et al. 2019. Atom: accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4660–4669.

[6] John Duchi et al. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, (July 2011), 2121–2159.

[7] Clement Farabet et al. 2012. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35, 8, 1915–1929.

[8] Hamed Kiani Galoogahi et al. 2013. Multi-channel correlation filters. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* (December 2013).

[9] Stephen Grossberg. 1987. Competitive learning: from interactive activation to adaptive resonance. *Cognitive science*, 11, 1, 23–63.

[10] Sam Hare et al. 2011. Struck: structured output tracking with kernels. In *2011 International Conference on Computer Vision*, 263–270. DOI: `10.1109/ICCV.2011.6126251`.

[11] João F. Henriques et al. 2014. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37, 583–596, 3.

[12] Zdenek Kalal. 2011. Tracking learning detection. `http://www.ee.surrey.ac.uk/CVSSP/Publications/papers/Kalal-PhD_Thesis-2011.pdf`.

[13] Zdenek Kalal et al. 2010. P-n learning: bootstrapping binary classifiers by structural constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 49–56. DOI: `10.1109/CVPR.2010.5540231`.

[14] Zdenek Kalal et al. 2011. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34, 1409–1422, 7.

[15] Kai Kang et al. 2014. Fully convolutional neural networks for crowd segmentation. *arXiv preprint arXiv:1411.4464*.

[16] Matej Kristan et al. 2020. The eighth visual object tracking vot2020 challenge results. In *Computer Vision – ECCV 2020 Workshops*. Adrien Bartoli et al., editors. Springer International Publishing, Cham, 547–601.

[17] Matej Kristan et al. 2020. The eighth visual object tracking vot2020 challenge results. (2020).

[18] Matej Kristan et al. 2017. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. (October 2017).

[19] Alex Krizhevsky et al. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[20] Chao Ma et al. 2015. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (June 2015).

[21] L. Matthews et al. 2004. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 6, 810–815. DOI: `10.1109/TPAMI.2004.16`.

[22] K.P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning series.* MIT Press. ISBN: 9780262018029.

[23] Binu Nair et al. 2011. Multi-pose face recognition and tracking system. *Procedia CS*, 6, (December 2011), 381–386. DOI: `10.1016/j.procs.2011.08.070`.

[24] Hyeonseob Nam et al. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (June 2016).

[25] Mustafa Ozuysal et al. 2007. Fast keypoint recognition in ten lines of code. In (June 2007). DOI: 10.1109/CVPR.2007.383123.

[26] Pentland et al. 1994. View-based and modular eigenspaces for face recognition. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 84–91. DOI: 10.1109/CVPR.1994.323814.

[27] Ali Sharif Razavian et al. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 512–519.

[28] Julian Schrittwieser et al. 2021. Online and offline reinforcement learning by planning with a learned model. In *Advances in Neural Information Processing Systems*. M. Ranzato et al., editors. Volume 34. Curran Associates, Inc., 27580–27591. https://proceedings.neurips.cc/paper/2021/file/e8258e5140317ff36c7f8225a3bf9590-Paper.pdf.

[29] Matthew Turk et al. 1991. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3, 1, (January 1991), 71–86. ISSN: 0898-929X. DOI: 10.1162/jocn.1991.3.1.71.