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 5, 2022

Abstract

Tracking objects in video streams is a powerful and general tool in computer vision. This research will investigate the recognition and tracking of multiple faces in settings where the faces are concurrently visible. The full study will implement a tracking system that detects faces and tracks their motion in a video stream. The tracker requires data that define the faces of the targets to initialise itself. The initialised tracker will detect if a target face is present or absent in a frame of a given 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. A thorough review of existing literature is required to implement such a system. This document presents works related to the topic.

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 automatically identifies and tracks target faces in a 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 as input, the system searches the video 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 the tracker identifies a target face, the tracker follows its motion. 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 Research Statement

This section revises the problem statement of the project proposal. The problem statement has changed for clarity, but the statement's content remains 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 primary computer vision technique 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) the is tracking of objects that can undergo partial occlusions, change appearance, and disappear and reappear from the field of view. This form of tracking is opposed to short-term tracking (ST), where the entire object is in the field of view for the whole duration of the video. ST can be used as a basis for LT, as is done in the case of TLD.

This research defines minimal input data as a single bounding box around each target face. The bounding box gives a target's name and location in an image. This study aims 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. Facial recognition aims to label a face present in an image.

3.1 Eigenfaces

Turk and Pentland (1991) describe how to use principal 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. PCA can decompose any face, from a given set, into a linear combination of basis vectors; this is equivalent to representing a vector in Euclidean space in an eigenvector basis. A facial recognition system can use the decomposition's components to identify faces.

This usage of eigenfaces allows for a more compact representation of a face. PCA finds a set of features that account for the most significant variation in a given collection of faces. This method allows Turk and Pentland (1991) to achieve facial recognition that is "fast, relatively simple, and has been shown to work well in a constrained environment (Turk and Pentland, 1991)."

Bartlett *et al.* (2002) suggest using Independent Component Analysis (ICA) instead of PCA. ICA is a generalisation of PCA that considers the relationship of distant pixels. This mechanism allows ICA to encode more information in the eigenvectors compared to PCA.

A machine learning model can use this set of features to identify a given face. One option is to use a neural network which offers high accuracy and quick recognition (Turk and Pentland, 1991; Bartlett *et al.*, 2002). Alternatively, a naive Bayes classifier which amenable to online learning (Murphy, 2012) can do the recognition.

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. Thus, facial recognition must identify faces from a frontal and profile pose for application to real-world data.

Pentland *et al.* (1994) suggest two solutions to the problem of Multi-Pose Face Recognition using eigenfaces. The first solution is using a single high dimensional eigenface space. This face space uses the basis vectors to represent information about both face and pose. Alternatively, different spaces can represent different viewpoints–PCA defines each face space by using all the images displaying a specific face angle. One face space uses photographs taken 10 degrees left of the frontal view; Another face space uses images of faces at an angle of 20 degrees right of the frontal view.

Nair *et al.* (2011) describe ways to recognise and track a face that takes on multiple poses. The system proposed by Nair *et al.* (2011) has three components: Haar Cascades based face detection, weighted modular PCA based face recognition and Kalman tracker.

4 Learning

Consider a continuous video stream that a pre-trained tracking system is analysing. This video stream is a constant flux of information, and the tracker might not have encountered some of this information in its history. The tracking system can, thus, use the video stream's content to improve itself. Improving the system requires learning mechanisms, specifically online learning–where information becomes available during operation.

4.1 Online Learning

Online learning is learning done as data becomes available. This form of model training is required to handle streaming data, for example, stock prices and videos. Offline learning can also use online learning methods. This technique is applicable, 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 minimise a cost function given by:

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

Changing the model's parameters $\boldsymbol{\theta}$ achieves minimisation (Murphy, 2012). The $f(\boldsymbol{\theta}, \boldsymbol{z}_i)$ are functions that give the cost of some data point $\boldsymbol{z}_i$ for the parameters $\boldsymbol{\theta}$ SGD updates the parameters in sequential steps of the form:

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

There are many ways to implement gradient descent, and adaptive step sizes can improve the optimisation results (Duchi *et al.*, 2011).

Another option for an online learning classifier uses Bayesian inference. Training the Bayesian classifier updates its posterior probability according to:

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

Where $\mathcal{D}_{1:n}$ 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 (Murphy, 2012).

4.2 Reinforcement Learning

Schrittwieser *et al.* (2021) propose a new online reinforcement learning method that can train models with minimal input data. In their paper, Schrittwieser *et al.* describe the *Reanalyse* algorithm. The *Reanalyse* algorithm uses a machine learning model's state and input data from the environment to generate training targets for the model. 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 cycles the available training data–this allows the algorithm to extract most of the information from a limited dataset.

4.3 Updating Template Trackers

Template tracking (Matthews *et al.*, 2004) assumes that the appearance of the target object does not change. This methodology results in simplistic tracking, and the tracker will fail if the orientation or view of the target changes. Matthews *et al.* (2004) propose solutions to this and discuss the concerned problems. The problems result from what is known as the stability-plasticity dilemma (Grossberg, 1987).

Every template update introduces some form of error in the template. This scheme causes the tracker to drift and eventually fail (Matthews *et al.*, 2004).

Suppose that $\mathbf{x}$ is a pixel's coordinate vector 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 object's template in the n^{th} frame of the 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 definitions, the problem of tracking formally reduces to computing the parameters for the deformation of the object:

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

And then updating the tracking template based on the warp of the n^{th} frame, for example a naive update is (Matthews *et al.*, 2004):

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

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

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

With $\text{gd}\min_{\mathbf{p}}$ indicating a gradient descent minimisation 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:

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

Where $\varepsilon > 0$ is some small threshold. This algorithm updates the template if template retention causes tracker drift; otherwise, the algorithm does not update the template.

4.4 P-N Learning

Training a tracker on a video stream is, in effect, bootstrapping a classifier online. Kalal *et al.* (2010) offer a solution to stably training a binary classifier. Stability in the tracker's learning is essential in avoiding tracker drift during the video progression.

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 recognise the false negatives made by the classifier; the second component is the N-expert which identifies the classifier's false positives. These experts make errors; otherwise, they would make a perfect classifier by themselves. One component's mistakes can, with proper implementation, compensate for the other's mistakes which leads to stable learning (Kalal *et al.*, 2011).

Two-dimensional dynamical systems can model the P-N learning system (Kalal *et al.*, 2010). This model can be used to determine the stability of the learning.

5 Tracking

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

5.1 General Comparison of Trackers

The Visual Object Tracking Challenge(VOT) is a challenge that benchmarks various trackers every year (Kristan *et al.*, 2017, 2020a). VOT investigates both ST and LT. VOT has also introduced a real-time challenge (Kristan *et al.*, 2020a).

In the VOT 2020 challenge, most trackers use deep features instead of handpicked features that were predominant in the past. In the short-term tracking challenge, 86 % of trackers used deep features. Deep discriminative correlation filters (DCF) (Danelljan *et al.*, 2019) have succeeded in many recent VOT challenges. Of the short-term trackers in VOT2020, 15 used deep DCF.

All of the long-term trackers in VOT 2020 used convolutional neural networks. Several long-term challenge trackers use MDNet (Nam and Han, 2016) as a basis for object presence detection. One of the long-term trackers used deep DCF; 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. Computer vision uses CNNs to classify images with high accuracy and speed (Razavian *et al.*, 2014). CNNs come at the cost of requiring long times to train, in most cases using GPUs (Krizhevsky *et al.*, 2012). Therefore, the standard use of CNNs is impractical for tracking unknown objects; training a CNN online impairs the system's speed (Bertinetto *et al.*, 2016).

5.2.1 Crowd Segmentation with CNNs

A crucial part of tracking is distinguishing background from objects of interest. Kang and Wang (2014) propose fast fully-convolutional neural networks(FCNN) to segment crowds. Fully convolutional means that the CNN is invariant under the translation of the input image. The FCNN searches a whole image in one pass of forward propagation through the network. This process allows faster and more accurate segmentation compared to sliding window techniques.

The CNN of FCNN has no fully connected layers; instead, it uses a 1×1 convolution kernel layer to predict labels. This design removes the translation dependence, due to fully connected

layers, from the network. The translation invariance of the network allows it to search a whole image in one pass.

The FCNN uses appearance to segment the crowd–reducing the false positives occurring in motion cue segmentation. Another advantage of scanning the whole input image at once is the availability of more contextual appearance information, which aids the segmentation (Turk and Pentland, 1991; Farabet *et al.*, 2012). Once an image of a crowd is segmented, another system can do a more specific analysis, such as identifying people.

5.2.2 Multi-domain CNN

Nam and Han (2016) use a CNN as the basis of their tracker(MDNet). MDNet trains on a large, labelled dataset of videos to obtain generic target representations. The CNN learns one domain, cars say, at a time; all the domains are worked through iteratively. This process provides a CNN that can distinguish between any target and background in the trained domains.

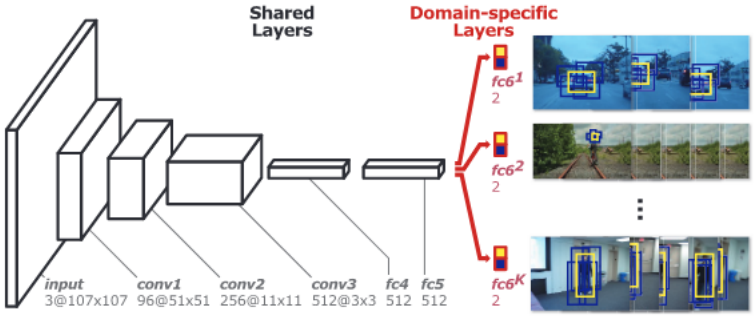

Figure 1: The architecture of the Multi-Domain Network, from (Nam and Han, 2016)

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

This design allowed Nam and Han to get very high accuracy in VOT challenges. The high accuracy comes at the cost of real-time execution–the binary classifiers of the network require a lot of time to train online. MDNet was only able to process one frame per second in the VOT challenges (Bertinetto *et al.*, 2016)–making it unsuitable for real-time tracking.

5.2.3 Convolutional Siamese Neural Networks

Bertinetto *et al.* (2016) offer another method of tracking an unknown object with Convolutional Neural networks. The method given by Bertinetto *et al.* is to train, offline, a CNN that solves the more general similarity problem. Their system learns a similarity function $f(z, x)$ that compares the images x and z; The system produces values that estimate how similar the images are. This function is equivalent to a composition of two other mappings: $f(z, x) = g(\phi(z), \phi(x))$. This composition consists of an embedding, ϕ, that does the convolution and a metric, g . A siamese neural network, two identical neural networks conjoined at the output node (Bromley *et al.*, 1993), is used to learn the similarity relation.

Bertinetto *et al.* suggest a *fully-convolutional* siamese network to learn the similarity relation. The input to the network is an image centred on the target's previous location. The network outputs a score map represented as a grid of numbers, Figure 2.

This architecture is in the form of 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–often, template trackers use the sum of squares metric (Matthews *et al.*, 2004).

5.3 Tracking with Correlation Filters

Henriques *et al.* (2014) propose Kernelized Correlation Filter(KCF) and the novel Dual Correlation Filter(DCF). Both KCF and DCF use circulant matrices and the kernel trick. The imple-

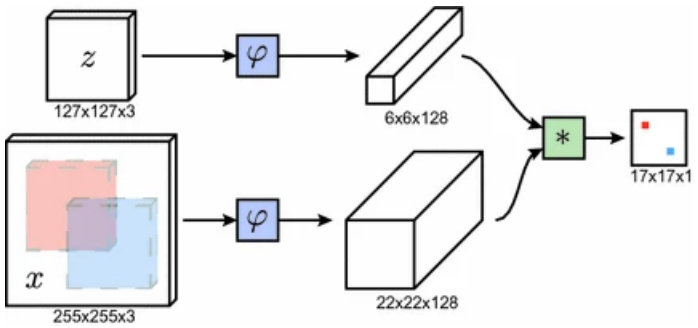

Figure 2: The architecture of the Fully Convolutional Siamese Neural Network, from (Bertinetto *et al.*, 2016)

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(Hare *et al.*, 2011)		65.6%	20
MOSSE(Bolme *et al.*, 2010)		43.1%	615

Table 1: Comparison of various trackers, adapted from (Henriques *et al.*, 2014)

mentation of KCF by Henriques *et al.* uses a Gaussian kernel, whereas the DCF implementation uses a linear kernel. The calculations involved with the linear kernel are less computationally complex than for the Gaussian kernel. Therefore, DCF can be processed faster at the cost of some tracking precision.

Work by Galoogahi *et al.* (2013) allows KCF and DCF to exploit modern and useful feature descriptors. Henriques *et al.* show that KCF and DCF can be applied to the Histogram of Oriented Gradients (HOG) descriptor to track and detect objects in a video stream with lower computation times and better accuracy. Table 1 shows that KCF and DCF applied to HOG features outperform many tracking systems. The results shown in Table 1 give the tracker's performance on a standard four core desktop processor from 2014.

The system implemented by Henriques *et al.* does not incorporate a failure recovery mechanism– section 8 of (Henriques *et al.*, 2014). In other words, Henriques *et al.* only explore KCF in the domain of ST. This approach contrasts the original TLD system that provides a failure recovery mechanism in the detection component (Kalal *et al.*, 2011). The TLD framework can employ Henriques *et al.*s' short-term KCF or DCF tracking system to build a long-term tracker.

Ma *et al.* (2015) investigate the problem of single object LT using correlation tracking. Ma *et al.* use two Gaussian ridge regression (Murphy, 2012) 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 tracks the object's trajectory through fast motion and occlusions, and the second tracks scale changes. Using both tracking models, they train an online detector that is both flexible(from the first tracker model) and stable(from the second tracker model).

Ma *et al.* train a random fern classifier (Ozuysal *et al.*, 2007; Kalal *et al.*, 2011) online to handle tracker failure. This solution for long-term tracking is similar to what Kalal does with TLD.

5.4 Tracking, Learning, Detection

Kalal *et al.* (2011) invented the Tracking, Learning and Detection(TLD) framework for the long-term tracking of objects in a video stream. Kalal's original implementation uses a median flow tracker, P-N learning, and a random forest and nearest neighbour based detector (Kalal, 2011).

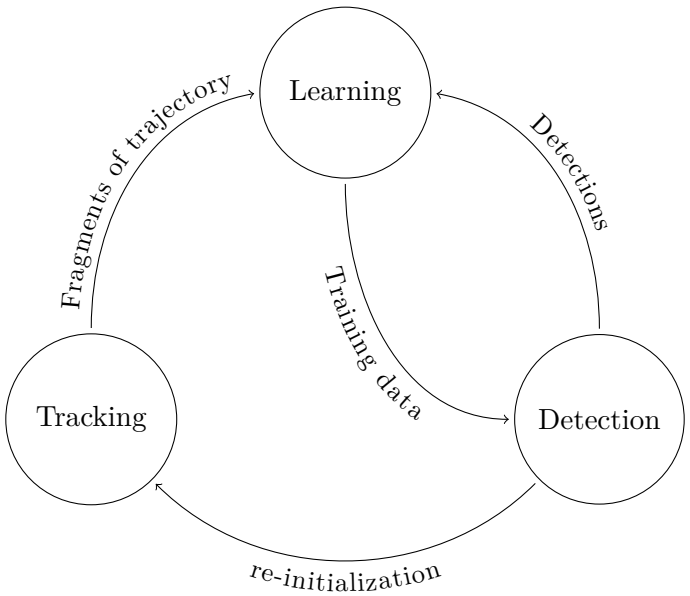

Figure 3: The interaction between tracking, learning and detection in TLD. Figure from (Kalal *et al.*, 2011)

These three components give the system its tracking, learning and detection capabilities in that order.

The learning component 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 its performance as time progresses (Kalal *et al.*, 2011). TLD's learning component requires online learning, owing to the nature of the framework. Kalal developed the P-N Learning paradigm (Kalal *et al.*, 2010), a semi-supervised bootstrapping model (Murphy, 2012), tailored to the needs of TLD.

The tracker of TLD outputs a bounding box for the target object in every frame; The detector produces a second bounding box for the target object. The P-experts and N-experts of the learning component use these bounding boxes to determine the detector's false positives and false negatives. The learning mechanism updates the detector with this error information, as described in Section 4.4. An integrator combines the bounding boxes given by the tracker and detector to provide better target location estimation (Kalal *et al.*, 2011).

6 Methodology

6.1 Methodology Overview

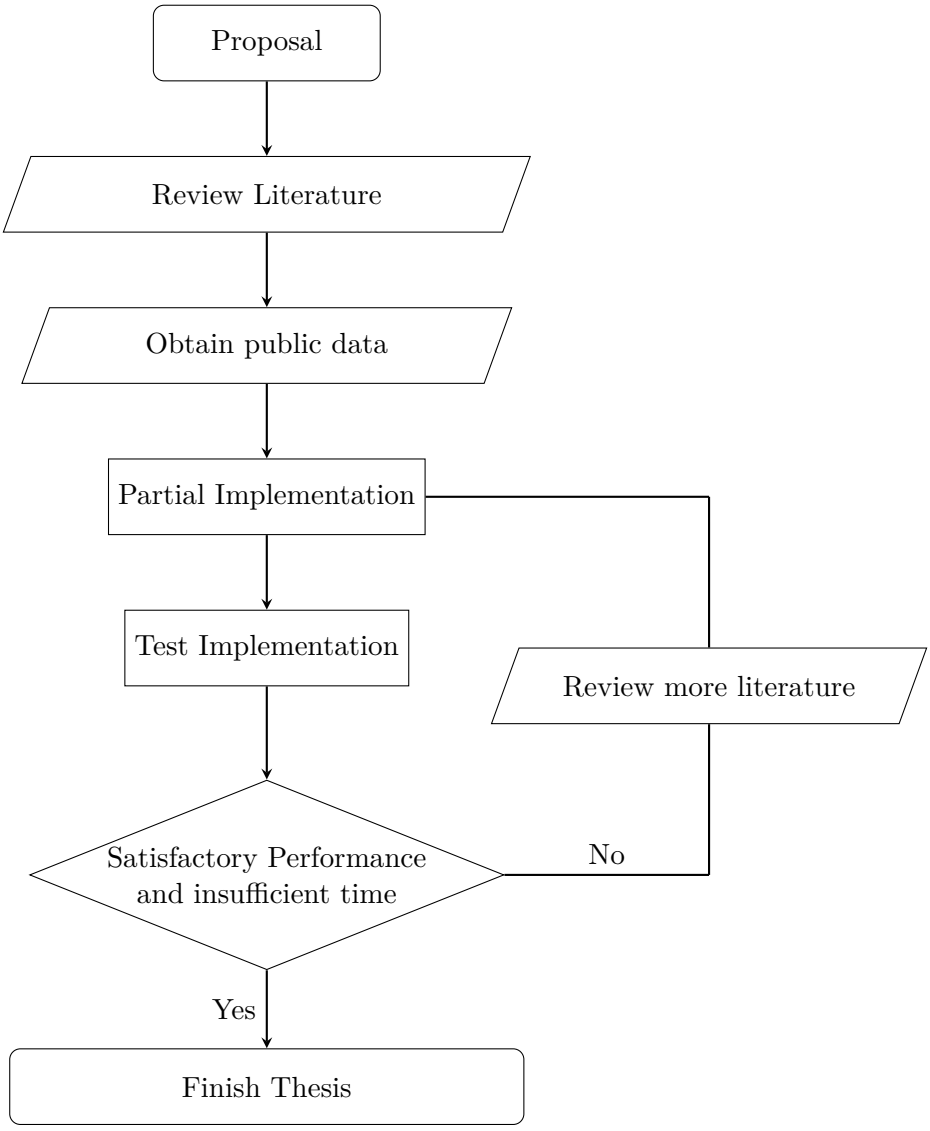

Figure 4: Conceptual overview of design methodology

6.2 Approach to Research

The first phase of this research will consist of an 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, the research requires a review of works describing trackers other than TLD. Following this, there will be a further review of the literature discussing the online training of classifiers. Implementing the system requires data for testing–a brief phase of data acquisition from public sources will provide data for developmental and testing purposes. This material accumulation 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 language of the system's development is C++; The implementation requires a proficient understanding of the C++ language. Implementation requires an Investigation of the OpenCV C++ library and other available utilities at this point of the approach.

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 the satisfactory performance of the TLD reimplementation, the next stage of research will commence.

At this point, the base TLD system requires further improvement. The improvement will

focus on the tracking and detection components of the system. These improvements involve keeping the base model and improving the individual tracker parts or restructuring the model to improve performance. This phase will constitute the fourth stage of the research.

Following this, the research investigates extending the system to track multiple objects simultaneously. There are naive ways to implement many object tracking systems, for example, creating many different single-target trackers to detect and track each object. This stage requires significant planning, research and reviewing of the current implementation to achieve good model performance and efficiency. This stage completes the implementation of the system.

The final stage of this research involves three things. First, the system requires a full review to find the strengths and faults of the design. If improvements to the implementation are required and time permits, development returns to stage four. Second, the system does test runs on videos obtained from the initial phase of the research for performance evaluation. Third, a thesis describes the implementation and specifications of the system. This step completes the research project.

6.3 Timeline

This section revises the proposal's timeline. Owing to exams and busy term times, most of the implementation deadlines now fall in the June-July Holiday.

Time	Deliverable
30 March 2022	Seminar 1: Presentation of the 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 it comparing to KCF
6 July 2022	Reviewing VOT for better trackers
11 July 2022	Investigating 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 to the system
3 October 2022	First Draft of thesis
10 October 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 (Kristan *et al.*, 2017, 2020b). The challenge ranks different categories of trackers–the two main categories are long-term and short-term tracking. VOT ranks the trackers based on various metrics that indicate the tracker's accuracy.

The VOT 2017 challenge introduced a real-time tracking challenge, which uses the short-term tracking dataset and performance measures (Kristan *et al.*, 2020b). 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. Thus, many of the tracking systems ranked in VOT are not amenable to this research.

There are, at present, two main approaches to solving the problem of long-term tracking in real-time. The first approach uses correlation tracking and training a classifier online (Ma *et al.*, 2015; Henriques *et al.*, 2014; Kalal *et al.*, 2011). These methods require little set-up time and can track single unknown 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 feature in the long-term challenge.

The second approach uses CNN's (Bertinetto *et al.*, 2016). This approach requires an offline training stage, but the offline training stage need not be repeated. CNN's allow for high accuracy tracking (Nam and Han, 2016; Bertinetto *et al.*, 2016), and provided that online learning is not required, the trackers can process videos at high frame rates (Kristan *et al.*, 2020b).

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 require a single image with a bounding box to track an unknown object (Nam and Han, 2016; Bertinetto *et al.*, 2016).

7.2 Revised Project Conclusion

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

The research tests the implemented system on public data. Tests consist of 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 does not aim to track 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

Bartlett, M., Movellan, J., and Sejnowski, T. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, 2002. doi:10.1109/TNN.2002.804287.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

Danelljan, M., Bhat, G., Khan, F. S., and Felsberg, M. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4660–4669. 2019.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

Galoogahi, H. K., Sim, T., and Lucey, S. Multi-channel correlation filters. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. December 2013.

Grossberg, S. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

Hare, S., Saffari, A., and Torr, P. H. S. Struck: Structured output tracking with kernels. In *2011 International Conference on Computer Vision*, pages 263–270. 2011. doi:10.1109/ICCV.2011.6126251.

Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37:583–596, 2014.

Kalal, Z. Tracking Learning Detection. Ph.D. thesis, University Of Surrey, 2011. URL http://www.ee.surrey.ac.uk/CVSSP/Publications/papers/Kalal-PhD_Thesis-2011.pdf

Kalal, Z., Matas, J., and Mikolajczyk, K. P-n learning: Bootstrapping binary classifiers by structural constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 49–56. 2010. doi:10.1109/CVPR.2010.5540231.

Kalal, Z., Mikolajczyk, K., and Matas, J. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34:1409–1422, 2011.

Kang, K. and Wang, X. Fully convolutional neural networks for crowd segmentation. *arXiv preprint arXiv:1411.4464*, 2014.

Kristan, M. *et al.* The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. Oct 2017.

Kristan, M. *et al.* The eighth visual object tracking vot2020 challenge results. In **Bartoli, A. and Fusiello, A.**, editors, *Computer Vision – ECCV 2020 Workshops*, pages 547–601. Springer International Publishing, Cham, 2020a.

Kristan, M. *et al.* The eighth visual object tracking vot2020 challenge results. 2020b.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Ma, C., Yang, X., Zhang, C., and Yang, M.-H. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

Matthews, L., Ishikawa, T., and Baker, S. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004. doi:10.1109/TPAMI.2004.16.

Murphy, K. Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning series. MIT Press, 2012. ISBN 9780262018029.

Nair, B., Foytik, J., Tompkins, R., Diskin, Y., Aspiras, T., and Asari, V. Multi-pose face recognition and tracking system. *Procedia CS*, 6:381–386, 12 2011. doi:10.1016/j.procs.2011.08.070.

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

Ozuysal, M., Fua, P., and Lepetit, V. Fast keypoint recognition in ten lines of code. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 2007. doi:10.1109/CVPR.2007.383123.

Pentland, Moghaddam, and Starner. View-based and modular eigenspaces for face recognition. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 84–91. 1994. doi:10.1109/CVPR.1994.323814.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.

Schrittwieser, J., Hubert, T., Mandhane, A., Barekatain, M., Antonoglou, I., and Silver, D. Online and offline reinforcement learning by planning with a learned model. In **Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W.**, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27580–27591. Curran Associates, Inc., 2021.
URL https://proceedings.neurips.cc/paper/2021/file/e8258e5140317ff36c7f8225a3bf9590-Paper.pdf

Turk, M. and Pentland, A. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 01 1991. ISSN 0898-929X. doi:10.1162/jocn.1991.3.1.71.