Safely caching HOG pyramid feature levels, to speed up facial landmark detection

by

Gareth Adam Higgins

Bachelor of Engineering, Ryerson, 2016

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Safely caching HOG pyramid feature levels, to speed up facial landmark detection

Master of Applied Science 2019

Gareth Adam Higgins

Electrical and Computer Engineering

Ryerson University

Abstract

This thesis presents an algorithm for improving the execution time of existing Histogram of Oriented Gradients (HOG) pyramid analysis based facial landmark detection. It extends the work of [1] to video data. A Bayesian Network (Bayes Net) is used as a policy network to determine when previously calculated features can be safely reused. This avoids the problem of recalculating expensive features every frame. The algorithm leverages a set of lightweight features to minimize additional overhead. Additionally, it takes advantage of the wide spread adoption of H.264 encoding in consumer grade recording devices, to acquire cheap motions vectors. Experimental results on a difficult real world data set show that policy network is effective, and that the error introduced to the system remains relatively low. A large performance benefit is realized due to the use of the cached features.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to the numerous people who supported me, and provided guidance throughout the completion of this thesis. Foremost I would like to thank my supervisor Dr. Ling Guan, for his valuable insight, helpful direction, and incredible patience, including the opportunity he made possible to undertake a research internship in Japan with Nippon Electric Company (NEC).

In addition, a large thank you to the members of Ryerson Multimedia Labratory (RML) all of whom were always willing to discuss any topic, academic or otherwise. Specifically I would like to thank my good friend Liang Chengwu who worked alongside me late into the night. Noureldin Elmadany who always had thoughtful suggestions, and a positive attitude. Randy, and Ryan Tan for the motivation, and direction they both constantly provided. Thanh Hong Phuoc for his guidance regarding H.264 encoding, and related material.

I would like to thank Alcohol Countermeasure Systems Inc. (ACS) for their support, including access to their large database of videos, and biometric data. From ACS I would like to thank Dr. Azhar Quddus, and Dr. Ali Shahidi Zandi for their frequent meetings, and advice.

Finally I would like to thank my Mother, and Grandparents for their continued support, understanding, and unwavering support over the many years it took to reach this point.

Contents

List of Tables

List of Figures

List of Algorithms

Glossary

Codec An algorithm, or device which encodes or decodes data. Often decreasing the size of data to enable transmission. 22

FC layer Fully Connected layer in a Neural Network. 12, 13

Flickr An online image, and video hosting service. 10

H.264 A block based video compression algorithm which utilizes motion compensation. iii, iv, vi, ix, 5, 22, 23, 25, 26, 48, 49

Microsoft Kinect Multi-view camera input device, with body tracking capabilities released by Microsoft. 43

Oblique Neither parallel nor at a right angle to a specified or implied line; slanting. x, 18, 41

Acronyms

AAM Active Appearance Model. 11

ACS Alcohol Countermeasure Systems Inc.. iv, ix, 3, 17

AFW Annotated Face in-the-Wild. 10

AUC Area Under receiver operating characteristic Curve. x, 44

Bayes Net Bayesian Network. iii, viii, ix, 5, 6, 27–31, 43, 45, 48, 49

BPA Bayes Point Algorithm. 27, 28

Cascaded CNN Cascaded Convolutional Neural Network. 12, 13

CGI Computer Generated Image. 3

CNN Convolutional Neural Network. vii, 12, 13, 50

HOG Histogram of Oriented Gradients. iii, vi, ix, 3, 4, 6, 14–22, 27, 30–33, 39, 42, 48

ICIAR International Conference on Image Analysis and Recognition. 7

NEC Nippon Electric Company. iv

NN Neural Network. 6, 12, 16, 28, 50

ResNet Residual Network. 16, 19, 20

RL Reinforcement Learning. 49

RML Ryerson Multimedia Labratory. iv, 4

RoC Receiver Operating Characteristic. 43, 44

Chapter 1

Introduction

1.1 Background

Face detection has always been an area of interest in computer vision. With deeper analysis focusing on problems such as facial landmark detection, pose estimation, and face identification. Recently deep learning techniques [2], [3] have been showing promising results for detection, classification, and identification. However traditional machine learning algorithms still provide state of the art performance for pose estimation, especially over multiple scales. This is partially because traditional techniques maintain higher level semantic information.

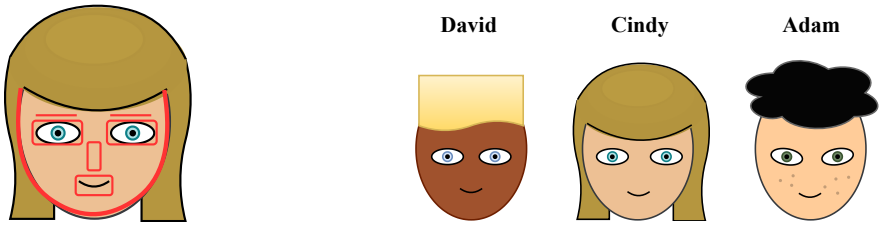
These tasks are separated due to the difference in objective, and difficulty. The easiest task being face detection, which is only concerned with finding the existence of a face, and generally a rough bounding box; an example output is shown in fig. 1.1a. For

this task deep learning currently provides the best results. In pose estimation the goal is to determine the orientation, and rotation of a face, which requires a more refined bounding box, or face model.

(a) Face detection (b) Pose estimation

Figure 1.1: Examples of face detection, and pose estimation

For facial landmark detection, not only must the face be identified, but the location of sub-features such as the eyes, nose, mouth, etc; shown in fig. 1.2a. Generally these sub-features are more difficult to detect, and keep track of in a computationally efficient manner. Finally shown in fig. 1.2b is face identification, where the goal is to detect the unique features of a face, similar to fingerprinting. The two major applications of this are monitoring, and identity verification for authentication.



(a) Facial landmark (b) Face identification

Figure 1.2: Examples of landmark detection, and face identification

With regards to pose estimation, and facial landmark detection they often serve as the first stage in more complex tasks. In the entertainment industry this includes real time facial mapping between live actors, and computer generated models, for use in animated, or Computer Generated Image (CGI) movies. Traditionally this has required physical motion capture markers to be attached to subjects, or actors. Independently, in health monitoring and security, an emerging subject of interest is emotion and attention recognition. This requires the tracking of individual facial features, with minimally invasive equipment or techniques. Histogram of Oriented Gradients (HOG) continues to be a state of the art approach for facial landmark extraction and pose estimation [4].

1.1.1 Purpose and Scope of this Thesis

This work is part of a larger collaborative project with Alcohol Countermeasure Systems Inc. (ACS) aimed at detecting distracted and inebriated drivers in order to prevent access to vehicles. According to [5], within Canada alone in 2015, there were 122 deaths and 596 incidents of bodily harm caused by alcohol impaired drivers. Additionally the number of drug impaired driving incidents has been on the rise. If we are able to detect drivers who are not fit to drive, we hope to decrease the number of injuries and fatalities by preventing vehicle operation. This can be achieved by detecting attention and emotional state using machine learning, based on minimally invasive and economically viable input features. This thesis covers the first stage of the pipeline, which is to design a system that can acquire stable facial features and pose information

of drivers. Other members from Ryerson Multimedia Labratory (RML) have already made progress on subsequent stages of the pipeline, which makes use of these features, such as gaze detection [6].

Although we optimize our setup for use in a car, it generalizes to any model detection applications.

1.1.2 Contributions

The contributions of this work can be summarized as follows:

- **Adaptation of facial landmark and pose estimator [1] from image data to video data.** The initial work was not created to process image sequences and naively processes each frame. We extend the algorithm to video data and utilize the redundant information in previous frames. The motivation is to utilize the facial landmark tracker as the first stage in an emotion and attentiveness recognition system.

- **Introduction of partial HOG pyramid feature level updates.** We introduce the concept of partially updating feature levels within the HOG pyramid. We define a cost function based on the feature level of interest, to control the frequency of updates. This reduces the total number of calculations that need to be performed in each frame.

- **Addition of a Bayesian Network (Bayes Net) based policy network.** We utilize a Bayes Net to determine when it is safe to utilize cached data. The Bayes Net is a lightweight network used to predict the probability that utilizing cached data will decrease accuracy. As an additionally safety measure, low confidence predictions fall back to the original algorithm. This minimizes the likelihood that the modified algorithm will produce different results from the the original implementation. This fallback behaviour allows us to train our Bayes Net with a very small sample input, since we are only looking for easy hints.

- **Utilization of hardware pre-computed H.264 motion vectors.** Inexpensive modern recording devices utilizes H.264 to compress the stream; for storage and transmission. As part of this process, motion vectors are calculated in order to enable inter-frame prediction. We utilize these vectors as a hint to the Bayes Net. We avoid an expensive software implementation by reading them out of the H.264 stream.

1.1.3 Organization of Thesis

The remainder of this thesis is organized as follows:

Chapter 2 — Literature Review: This chapter covers existing works related to facial landmark tracking. First, the distinction between marker based and marker-less tracking systems is made. Followed by a discussion of the motivation to move to marker-

less; primarily cost, and accessibility. Following that, existing techniques are divided into two groups; model based and regression based. Traditional statistical techniques can fall under either categories. However, Neural Network (NN) based techniques fall under regression. Real-time execution is the goal that most facial landmark trackers attempt to achieve while maintaining accuracy. Compared to face detection, this is much more difficult to achieve due to the complexity of localising the landmarks.

Chapter 3 — Methodology: This chapter first briefly describes the original implementation from [1] and our modifications. It is broken up into three sections. The first covers HOG and touches on multi-resolution analysis. It describes how a HOG feature is constructed and subsequent histogram is formed. The second section covers in detail how the original algorithm operates. The final section details our modifications to the system to make it better suited to video data. We first introduce the idea of feature caching in order to reduce calculations. We then introduce a cost function for choosing which features will be updated. Furthermore we introduce a restricted model search to further decrease the number of calculations required every frame. Finally, we describe our Bayes Net based caching policy and overall algorithm. The algorithm attempts to balance accuracy with execution speed.

Chapter 4 — Experiments & Results: This chapter will present the experimentation used to show that the modified algorithm behaves similar to the original with regards to accuracy. We execute the original algorithm and our modified version on the

same database; YouTube Faces DB [7]. We show in general that our algorithm is able execute faster that the original implementation when applied to video. Additionally we show that the error introduced is generally low. For the sake of completeness, we also show that our policy network is performing better than selecting feature levels to skip randomly.

Chapter 5 — Conclusion: This chapter explains why the decrease in execution time occurs. Additionally, the restriction on when this behaviour can be expected. We then identify the scenarios which cause the largest accuracy drop in the experiment. Finally, future work such as a partial online training or the use of more powerful reinforcement based policy networks is discussed.

1.1.4 Publications

Our algorithm will be published in International Conference on Image Analysis and Recognition (ICIAR), held August 2019, as "Safely caching HOG pyramid feature levels, to speed up facial landmark detection"[8].

Chapter 2

Literature Review

2.1 Facial Landmark trackers

Full body tracking in professional environments traditionally uses a visual marker based system to improve results. For more casual users, this can be incorporated into clothing or wearable accessories. If the purpose is for personal entertainment, often the inaccuracy of marker-less is accepted; such as in gaming devices or arcades. When considering facial landmark tracking, both the aforementioned techniques are also utilized. There is a narrow scope in which facial landmark trackers are used with markers for high precision tasks [9], [10]. However they, are not user friendly nor economically viable since the markers must be reapplied each time. Shown in fig. 2.1 are the markers that would be typically utilized for such a system. This leads to a strong desire to operate in marker-less environments.

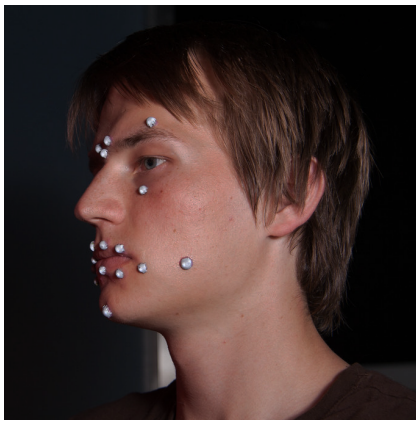


Figure 2.1: Facial motion capture markers, "Motion Capture" by Jirka Matousek is licensed under CC BY 2.0 / Cropped and cleaned up

Some recent works [11] have eliminated the need for special reflective markers and instead are able to use non-toxic paint. This reduces the issue regarding cost, but does not improve the ease of use significantly. While the focus of this paper is facial landmark tracking, the final goal is to be included in a project regarding driver attention and sobriety detection for use in vehicles. This rules out marker based tracking, as the user cannot be expected to apply markers before utilizing their vehicle.

2.1.1 Optimistic and Realistic Databases

The accuracy of facial landmark detectors is shown to drop significantly when applied to difficult databases containing real world photos, not just frontal portraits. This is not unique to facial landmark detection, all visual recognition system suffer from, illumination, reflection, occlusion, and multiple scales. Zhu and Ramanan [1] show

that while their method achieves slightly less than state of the art accuracy on the MultiPie data set [12], it does produce state of the art results on a more difficult data set which they have crafted; Annotated Face in-the-Wild (AFW). AFW is created by hand annotating images from Flickr. The main characteristic that differentiates AFW from other databases is that non-frontal portraits with multiple subjects are specifically included; similar to fig. 2.2. Additionally, they calculate both pose estimation and facial land mark tracking jointly. Traditional algorithms first need a bounding box of the face, which they then apply pose or landmark detection.

Figure 2.2: Busy scene including many subjects with different poses

Traditionally, facial landmark detectors for video have been adapted from versions which operated on pictures or single frames. A brief overview will be discussed below, however an extensive comparison can be found in the survey paper [13]. Several

techniques can be used to improve the naive implementation, which is to process the video as a stream of individual photos. The first technique discussed is to apply a filter to the position of landmarks reported between frames [14], which helps to smooth out both visual noise and loss of tracking. Another approach is to continually refine the estimated face shape based on the predictions from previous frames [15]. Our approach similarly adapts an existing method that originally operated on single frames, however our goal is to improve processing speed by utilizing the nature of video and redundant information without sacrificing significant accuracy.

2.2 Model based

2.2.1 Active Appearance Model

In [16] Active Appearance Model (AAM) is adapted to faces. They use supervised learning with labelled facial feature key points to learn face shape. They then learn the relationship between face texture and shape directly on the gray scale pixel values. To handle faces of different sizes they utilize Gaussian image pyramid multi-resolution analysis. An interesting approach of their technique is that they start from the lowest level of the pyramid, with the smallest resolution. This allows them to fit their model more quickly since the there are less pixels to compare. When checking the subsequent higher level layers, they utilize the previously discovered key point locations.

2.3 Regression Based

Recently, NN based regression techniques dominate this category. Traditional regression techniques are still utilized, such as [17], but they are falling out of favour. Convolutional Neural Networks (CNNs) have been shown to perform extremely well at object detection and classification. Naturally, attempts are being made to extend them to facial landmark detection. There are many NN based methods that outperform statistical methods in the task of face detection [18]. However, it is more difficult to find methods which produce competitive pose estimation and facial feature locations.

2.3.1 Cascaded Convolutional Neural Network

Fan and Zhou [19] introduce a two part Cascaded Convolutional Neural Network (Cascaded CNN) to split the facial landmark detection into two tasks. Their first CNN produces a rough prediction of the key points. They posit that while the individual landmarks may not be accurate, they capture the rotation and scale of the subject. The second network is utilized to refine the key point prediction and learn the relationship between them. For both CNNs they use a VGG [20] like network structure, with half the number of channels in each layer, and only two FC layers. Additionally, the second network utilized only the first three convolutional layers.

2.3.2 Task-Constrained Deep Convolutional Network

Zhang, Luo, Loy, *et al.* [21] compare their work to an early version of Cascaded CNN [22]. Their two main contributions are decreasing the number of CNNs required and creating a network which is able to perform multiple tasks simultaneously. While the more recent version of Cascaded CNN [19] reduces the number of layers, it does not address task sharing. Zhang, Luo, Loy, *et al.* [21] define a network which performs pose estimation, facial landmark detection, sex identification, emotion recognition, and other related tasks utilizing the same convolutional layers. They use task specific FC layers, which are jointly trained, with the CNN. They achieve competitive results in [23].

Chapter 3

Methodology

In this chapter we describe the original implementation [1] and our modifications to it. We start by introducing HOG and HOG pyramid; including their computational limitations. A brief overview of the original implementation proposed by Zhu and Ramanan [1] is presented. We then present our algorithm and motivation. We introduce the idea of feature caching, which utilizes previously calculated HOG features. We then explain how feature level updates are scheduled into a partial update system. Finally, we discuss how the decision on when it is safe to utilize cached features is made.

3.1 Histogram of Oriented Gradients

For each pixel in the original image the filters F_y, F_x from eq. (3.1) are applied in order to calculate a gradient in each direction. These gradients, now represented by a vector,

are then grouped into cells of a fixed size.

$$F_x = \{1, 0, -1\}, F_y = \{1, 0, -1\}^T \tag{3.1}$$

Different applications can see variance in performance by increasing or decreasing this cell size. When the vectors within a cell align they hint that structure behind represents an outline or boundary. Figure 3.1 shows signed vectors (0° to 360°). The left grid represents the vectors before they are grouped into their cell size, and the right grid after. However, [24] shows that in human detection unsigned vectors (0° to 180°) achieve better results. In order to construct the HOG feature a histogram is taken using the direction as bins and aggregating the magnitudes. As shown in [24], the bin interval can also affect the quality of the results. We extend [1], which uses 18 bins for the histogram. They deviate slightly from traditional HOG in that adjacent bins to a particular vector also receive some of the magnitude through linear interpolation.

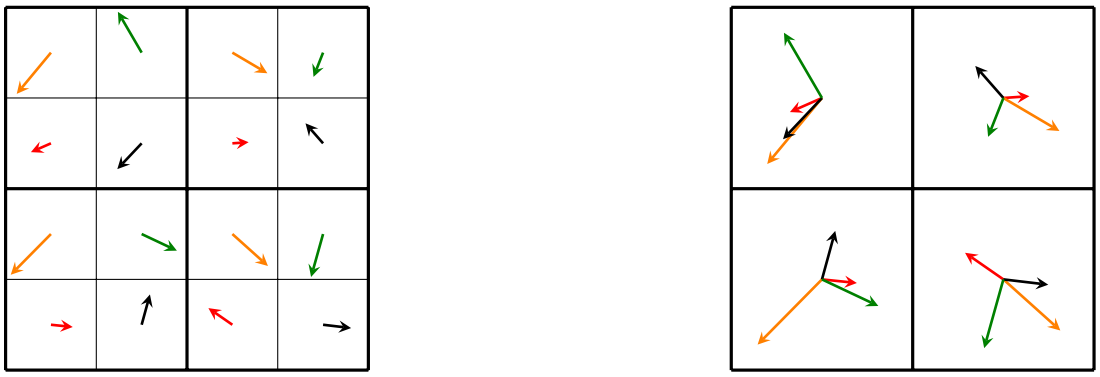


Figure 3.1: Signed HOG vectors with 2x2 cell size

3.1.1 HOG Pyramid

As shown in the leftmost image of fig. 3.2, for large images the HOG features are quite small, which may prevent the detection of objects closer to the camera. Increasing the cell size can alleviate this, but also decreases the quality of the vectors as they are likely to be noisy. Another solution is to process the image at multiple scales, or feature levels. Feature pyramids have been utilized to handle this problem for many years and continue to be used [25]. Popular NN based methods such as VGG [20] and Residual Network (ResNet) [26] emulate this concept; with convolutional layers followed by down sampling.

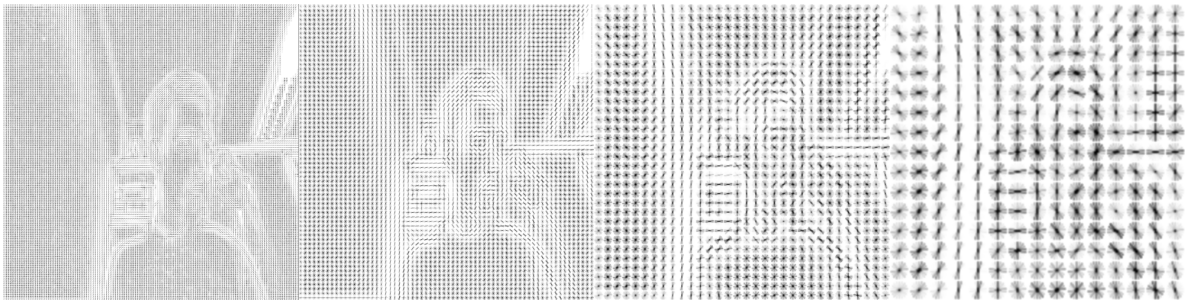


Figure 3.2: Increasingly deep levels of HOG pyramid from left to right.

In the case of HOG pyramid, at feature level the image resolution is down sampled, but the cell sizes are kept constant. This allows both local and global features to be investigated. In fig. 3.2 the noise from textures and small objects is disappears moving from left to right; towards the deeper levels of the pyramid. When approaching the correct scale there is a clear boundary formed between objects. The rightmost image in fig. 3.2 has passed the correct scale, thus object boundaries become less distinguishable.

3.2 Original Implementation

Our work is an extension of [1] which originally targets image data and is un-optimized for video data. A typical output for a single frame is shown in fig. 3.3. The yellow points represent parts of the model; with the blue boxes representing the HOG cell associated with the part. The pose angle is shown inside the yellow box located on the forehead of the subject.

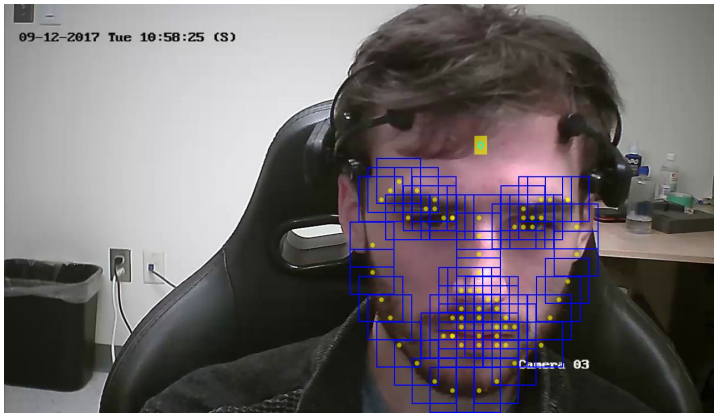
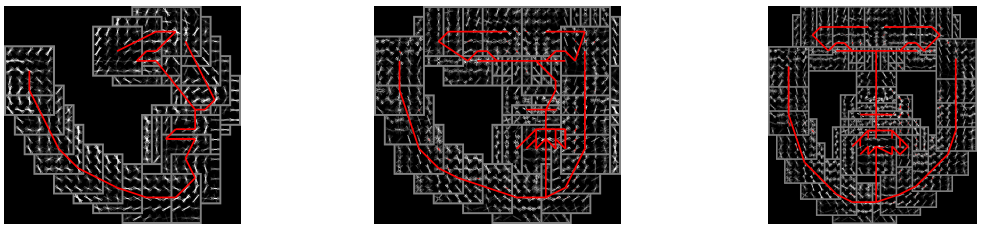


Figure 3.3: Detected face; including eyes, eyebrows, nose, mouth, and jaw line from ACS database

The approach they take is to build a mixture of tree-structured part models, where each part is described by a HOG feature. The models are learned via a supervised training method and annotated from $-90°$ to $90°$. Both the part HOG features and model tree structures are learned. We do not train our own models, instead we use one of the models generated by [1]. For the model we utilized, there are 13 unique views with six left profiles, six right profiles, and one frontal profile shown in fig. 3.4.



(a) Side view (b) Oblique view (c) Front view

Figure 3.4: Tree structure and HOG features for different models

When running the tracker, the part scores for each feature level of the HOG pyramid are calculated for every model. The orientation of the face is determined by the highest scoring model detected in a particular region. We recognize that multi-scale HOG Pyramid adds a considerable number of extra calculations which are not always relevant, nor required. This is because all scales at which a subject of interest could occur must have HOG performed each frame. Our goal is to find a balance between accuracy and processing speed. We achieve this by utilizing slightly stale information, which is still highly correlated to new frames. A similar approach is common in video processing, known as frame skip. We extend that idea to feature levels within a HOG pyramid. However unlike frame skip we utilize cached data from previous frames, instead of just discarding it entirely. The pseudo-code of the original algorithm is provided in alg. 3.1.

Algorithm 1: Pseudo code for original Algorithm

```
input learned models
set detection threshold
foreach image
    foreach feature levels
        Calculate HOG features save as level
        Downsample by scale factor
    foreach model
        foreach level in feature level
                foreach part in model
                    Convolve part with level save score
                Find X,Y of part with highest score and save part
```

3.3 Our Modification

3.3.1 Feature Caching

Conjecture 1. *Upper levels of the HOG pyramid, relative to the scale of interest, will change less between frames and do not need to be updated as frequently.*

Conjecture 2. *Lower levels of the HOG pyramid, relative to the scale of interest, are unlikely to contribute information distinguishable from noise, and can be ignored.*

Under these assumptions, the accuracy penalty from not updating distant feature levels is low. This introduces two issues deciding the frequency of update based on distance, and determining when data has become too outdated to be useful. Recently, this concept has also been applied to ResNet[26] where a policy network controls which

layers will be utilized in a forward pass, dubbed BlockDrop [27]. ResNet successively decreases output feature size by combining neighbouring cells, which is a similar operation to multi-scale analysis.

For any frame in which the scale of interest is not known, we run the original full algorithm. If latency is not acceptable, we can drop queued frames after this initialization under the assumption that the subject has not moved too far. After candidate scales are discovered, we prioritize them over other feature scale levels when performing HOG updates. Non-candidate feature level updates are then scheduled based on their difference to the closest scale of interest. This reduces the workload that must occur in every frame. This works naturally for subjects on the same scale level, and can be extended to subjects at different scales. The degenerate case occurs when a subject exists at each scale. This case is unlikely, as the scale space is often over selected. Overall this method does not completely ignore feature levels, it simply updates them less frequently, which makes it differ from a traditional frame skip approach.

3.3.2 Partial Update

During a partial update, each scale will not be updated every frame. The number of frames between updates U_p is shown in eq. (3.2), where S_i are scales which contain a detected subject of interest and S_p are all the possible scales to be updated.

$$U_p = \lfloor (\min_{S_i} S_i - S_p)^{\gamma} \rceil, \gamma = 1.25 \tag{3.2}$$

The exponent γ is chosen to maintain high update rate on near scales, while lowering update rate of distant scales. Larger values of γ are more likely to cause issue with tracking, but provide better performance. The non-updated HOG features are still considered when scoring the model parts in the subsequent step. This old data can be utilized because at that particular scale significant changes are not likely to have occurred.

3.3.3 Restricted Model Search

Since [1] is designed for standalone images there is no previously predicted model data which can be leveraged. However, when moving to video data, after a candidate model is found we can restrict the model search to adjacent models, under the condition that the motion in the scene does not exceed a threshold. Shown is fig. 3.5 is how the models are laid out relative to the face. There are 13 models, separated by 15° intervals. Relative to the frontal view, they cover a $-90°$ to $90°$ view of the face.

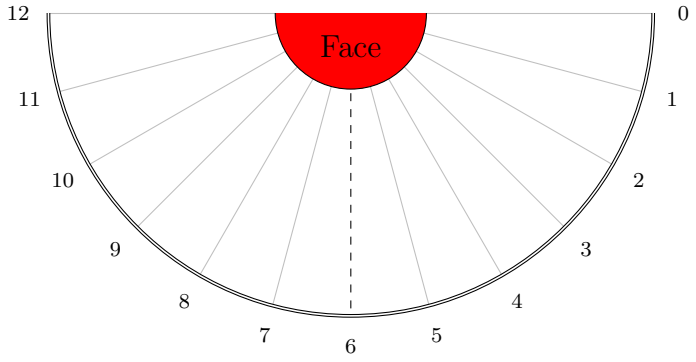


Figure 3.5: Positioning of each model, relative to face

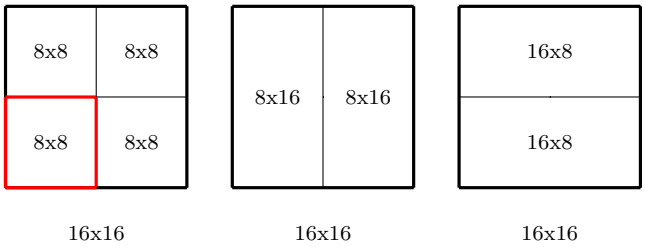
While the most computationally expensive operation in [1] is the HOG pyramid calculations, the second most expensive operation is the part scoring algorithm. This is because it must fit all the parts of every model to all HOG feature in each feature level. Although our implementation does not recalculate every feature level in the pyramid at every frame, we still perform the aforementioned search. By reducing the number of models that must be searched, we are able to further decrease the computation time.

3.3.4 H.264 Encoder & Motion Vectors

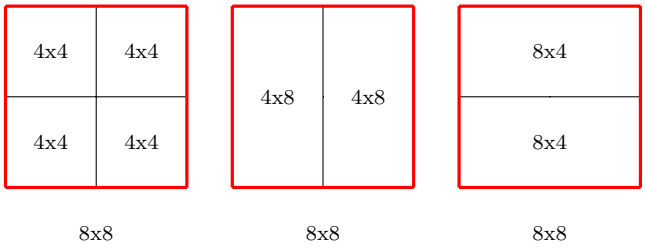
Recently many consumer grade cameras are equipped with hardware video encoders implementing advanced video Codecs such as H.264 [28]. Instead of recalculating the motion vectors at a software level, we extract them directly from the H.264 encoded video. The quality of the vectors is low, however they have practically zero performance cost since they are generated by the device automatically. H.264 is a block based video compression algorithm which utilizes motion compensation. The H.264 encoder has three separate frames types:

- I-Frame: Entire image is present
- P-Frame: Motion prediction is done using previous frames
- B-Frame: Motion prediction is done using both previous and future frames

In realtime or streamed video, B-Frames are not present as they require future data, so only forward prediction can be utilized. With this restriction, we utilize videos that only contain I-Frames and P-Frames.



(a) 16x16 macroblocks



(b) 8x8 macroblocks

Figure 3.6: Macroblocks used in H.264 encoding

During encoding, each frame of the video is first decomposed into macro blocks via a quad-tree like method [29]. As shown in fig. 3.6b, macroblocks are created to describe regions of pixel within the frame, with the minimum block size set at 4 pixels $\times$ 4 pixels. These sizes are selected to maintain a balance between accuracy and compression. If the macroblocks are too small then no compression can be realized. Conversely if the macroblocks are too large then detail will be lost from the image in the P-Frames as large regions are approximated.

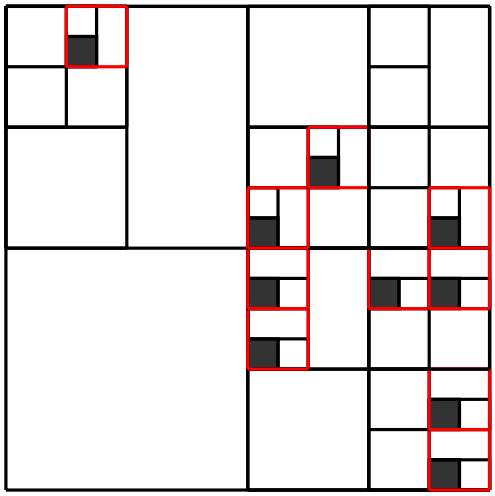


Figure 3.7: Macroblock decomposition of image frame; 8 pixel × 8 pixel block regions highlighted in red, and 4 pixel × 4 pixel block regions shown in black

Where fig. 3.6b shows what an individual macroblock looks like, section 3.3.4 show how a frame looks like after decomposition. Large regions represent areas with little variation, while smaller regions are required to show high variation detail. In the encoding algorithm, once the frame has been decomposed a search is performed against the previous frame to find block which are present in both. If matching blocks are found, then a motion vector is created to represent this.

By transmitting the motion vectors of the macroblocks, instead of the entire block, bandwidth is saved. The final result is a set of sparse motion vectors, as they apply to blocks not pixels. Since we rely on the trend of motion vectors which fall into an area of interest, this sparsity is not a problem.

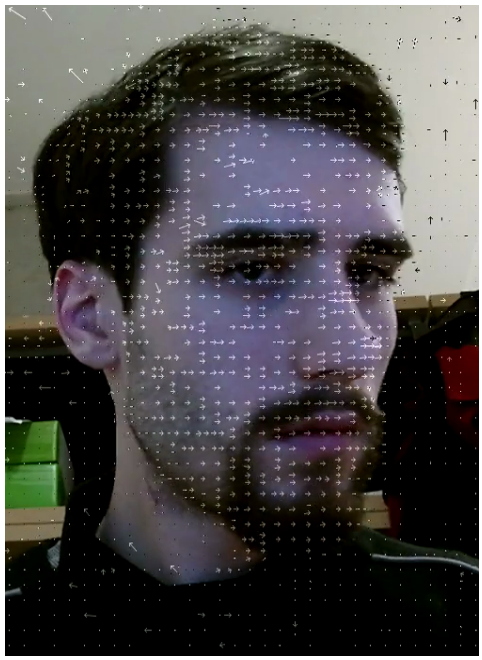


Figure 3.8: H.264 extracted motion vectors

It is possible to observe effects of the macroblocks, by looking at the motion vectors in fig. 3.8. Large macroblocks can be seen in the upper left corner where there are sparse motion vectors since the background provides little variation. In the face regions and around edges, the number of macroblocks increases and their size decreases in order to represent the detail. We find that these sparse motion vectors contain sufficient information to suggest when it is safe to avoid recalculating feature levels. Initially, the motion vectors of the whole image are considered. Once the algorithm has successfully completed an iteration, candidate bounding boxes are available based on the fit models. The bounding box, BB, is created by applying convex hull, denoted by $Conv$, to the

key points contained in the model, represented by M_{key}.

$$BB = Conv\left(\{(x, y) \in M_{key}\}\right) \tag{3.3}$$

In order to detect motion which may occur at the edge of the bounding box, as a subject moves, we use an enlarged copy of the bounding box. The existing bounding box is translated to the origin, by calculating the average (x, y) coordinates of the key points, denoted as $\mu_{(x,y)}$. It is then scaled by a factor α to capture motion that occurs at the edge of the face. Finally, the points are translated back to their coordinate space by utilizing $\mu_{(x,y)}$. Subsequent frames consider only the motion vectors which fall within these bounding boxes, as long as a valid model is found in that frame.

$$BB' = \alpha\left(BB - \mu_{(x,y)}\right) + \mu_{(x,y)} \tag{3.4}$$

Other Encoders

There is no strict requirement to use H.264, other motion estimation based encoders can be substituted. Older implementations such as H.263 may be utilized at the cost of some processing time, and accuracy. More interestingly newer implementations such as H.265 can be dropped in to improve both, accuracy, and speed. Currently H.264 has a larger adoption rate than H.265 in consumer cameras, until that changes utilizing H.264 is less restricting for the algorithm.

3.3.5 Caching Policy

In order to maintain accuracy and reduce result latency in fast moving scenes, a Bayes Net provided by the Infer.Net framework [30] is used to discover a policy for deciding when the HOG features need to be recalculated. More specifically, we utilize a Bayes Point Machine [31] which utilizes Bayes Point Algorithm (BPA).

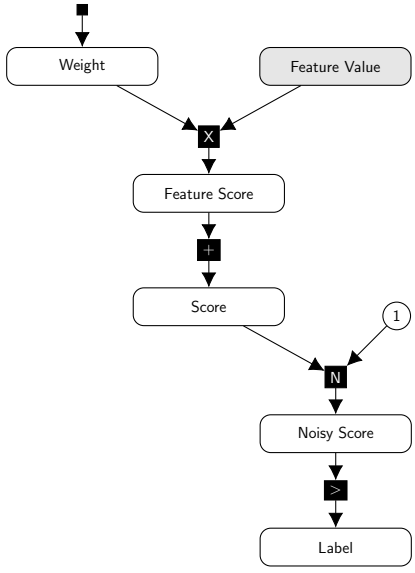


Figure 3.9: Basic Bayesian Network, adapted from example found in [30]

In a Bayes Net, an initial belief weight is supplied for a given feature and is used to generate predictions. A random belief weight can be used like in other machine learning algorithms. However, since the feature value has semantic information understandable by a human, the initial weight can be chosen to represent an expert's belief. Assuming the initial belief is not entirely incorrect the convergence speed of the training will be

improved. Bayes Nets are also known to have less overfitting problems when compared to deep NNs. Another advantage of Bayes Net is that they are able to handle missing data in the input features; both in training and inference. The largest benefit of a Bayes Net is that it can be used to predict the outcome of an intervention or action before that action is taken. This makes it perfect for use in our algorithm, as we are trying to predict the effect of using cached features, instead of recalculating them.

Bayes Net

As noted by Herbrich, Graepel, and Campbell [31] while the optimal approach for small or incomplete training data sets would be to use a full Bayesian classifier, there is a large computational cost. This is because all hypotheses must be examined for new test points. From [31] eq. (3.5) shows how the class selection for new test points is performed.

$$\text{Bayes}_z(x) := \underset{y \in Y}{\text{argmin}} \, E_{H|Z^m=z} \left[l \left(H \left(x \right), y \right) \right] \tag{3.5}$$

This cost is only paid during training time, however we would still like to avoid it; as to allow online training in future implementations. The solution to this problem is to utilise the BPA. The key feature of the BPA is that the entire hypothesis space is not searched, only hypotheses which are computationally efficient for the given test point.

$$\mathcal{A}_{bp}(z) := \underset{h \in H}{\text{argmin}} \, E_X \left[E_{H|Z^m=z} \left[l \left(h \left(X \right), H \left(X \right) \right) \right] \right] \tag{3.6}$$

Evidence

The action taken can be: no update, partial update, or full update; with full update being the most costly with regards to time. Our Bayes Net uses an input feature vector with five items:

1. Summation of frame motion vectors within area of interest
2. Information quality (Number of frames since last full update)
3. Previous frame pose change Occurrence
4. Previous frame scale change Occurrence
5. Previous frame fit model successfully

The output is a prediction of the optimal action required to prevent divergence from the original implementation and the confidence of the prediction. While not explored, it may be possible to avoid decoding the frame, if the Bayes Net determines that an update is not necessary based on the motion vectors.

Training and Prediction

The advantage of the Bayes Net is that it is very lightweight, both in inference, and in training. It also accepts unknowns as inputs, which allows us to deal with cases where we have just lost tracking, or have not yet acquired tracking. We train the Bayes Net on a single segment of video, with a good range of motion that sweeps through all the model poses. The length of the training video was not specifically chosen, however we record it and other statistics in table 3.1. Since we are looking for a hint and not the

exact policy this does not require a large database to train. We show that the Bayes Net provides results better than randomly dropping frames.

Table 3.1: Training input for Bayes Net

Resolution	Frames	Time	Train/Test
1920x1080	746	23s	60/40

The information quality is measure of how old the information contained in the HOG feature pyramid are. This feature was chosen to help predict how long processing can continue based on stale data. The expectation is that as the data quality drops, the uncertainty in the Bayes Net will rise until eventually it predict that loss of tracking would occur if a more substantial HOG pyramid update is not performed.

3.4 Algorithm Flow

The Bayes Net is used to predict which action would be required to prevent loss of accuracy in object tracking. In addition to the recommended action, we also obtain the probability that the Bayes Net believes in this outcome. For low confidence results, we opt to be cautious, and perform a more comprehensive update than recommended by the Bayes Net. With more evidence and testing, a more aggressive stance could be taken to balance accuracy. The pseudo-code for our implementation can be found in alg. 3.2.

Algorithm 2: Psuedo-code for modified Algorithm

```
1  input learned models
2  set detection threshold
3  initialize levels of interest
4  foreach image
5      if detected face last frame
6          Sum motion vectors bounded by detected objects
7      else
8          Sum all motion vectors
9      Query Bayes Net save as prediction
10     foreach fLevel in feature levels
11         if Partial Update
12             foreach iLevel in levels of interest
13                 Calculate distance from iLevel to fLevel as rate
14                 if last updated mod rate
15                     Calculate HOG features save as level
16                 else
17                     Use cached HOG features features
18         Downsample by scale factor
19     if Partial Update
20         models = select adjacent models to previously detected
21     foreach model in models
22         foreach level in feature level
23                 foreach part in model
24                     Convolve part with level save as score
25                 Find X,Y of part with highest score save part
```

3.5 Summary

Based on the discussed methodology, the expected advantages of the implementation is a decrease in average frame processing time. This is achieved by taking advantages of partial HOG level updates. The major concern with utilizing stale data is a drop in accuracy as compared to the original algorithm. To address this a policy network is used to intelligently select when it is safe to reuse stale data.

Chapter 4

Experiments & Results

4.1 Execution Performance

In order to test our method, we utilize the YouTube Faces DB [7]. We use approximately 400 videos chosen at random from the database to demonstrate our method. Due to the diversity of the data set various resolutions are present, with the smallest being 720x480. We also show that we can realize a large performance gain. Our modified algorithm at best takes only 36% of the time to process the same video, since we are able to avoid calculating unused HOG features.

$$\text{Relative Difference} = \frac{x - x_{ref}}{x_{ref}} \tag{4.1}$$

Additionally we show that our algorithm achieves an accuracy negligibly different from the original algorithm. We use relative difference in execution time as our performance metric, shown in eq. (4.1). Applying this metric to the randomly selected videos produces the histogram shown in fig. 4.1. The bar chart highlights the percent of frame samples along the primary vertical axis and the relative difference in execution time achieved along the horizontal. Since the percent of samples are split into many bins, a cumulative line graph is shown on the secondary axis for clarity.

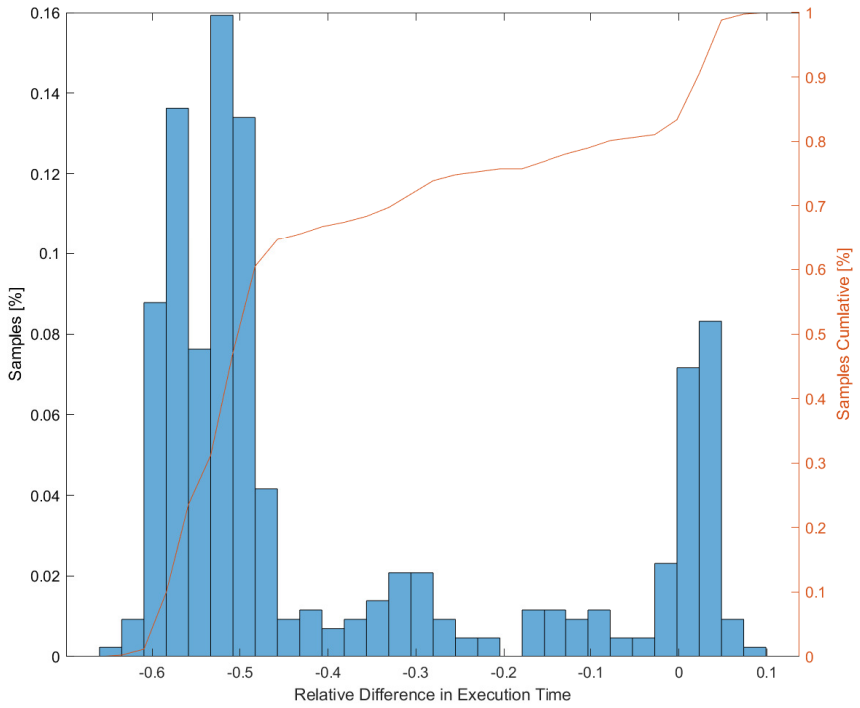


Figure 4.1: Histogram of execution time difference, compared to original in YouTube Faces DB using 433 randomly selected videos

Over 60% of the videos processed by the modified algorithm, take less than 50% of the original processing time. A summary of the results from fig. 4.1 is shown in section 4.1. The best and mean are both negative since they take less time than the original. Our worst case however is positive, indicating our algorithm can take more time. In scenes that contain a lot of motion, in which cached information cannot be utilized, we pay a performance penalty for no gain. The ten percent slower worst case is caused by the construction of the feature vector and inference, when such analysis provides no benefit, this is an expected result as not all frames entirely redundant.

Relative Difference		
Best	Mean	Worst
-0.6347	-0.3876	0.0969

Table 4.1: Performance difference between original algorithm being naively applied to each frame and our method.

In scenes with low motion, we realize the largest gain, using only 36% of the time compared to the original algorithm. For scenes with extremely high motion it may be possible to bypass the feature vector creation and inference entirely to decrease the penalty; this is left to future work.

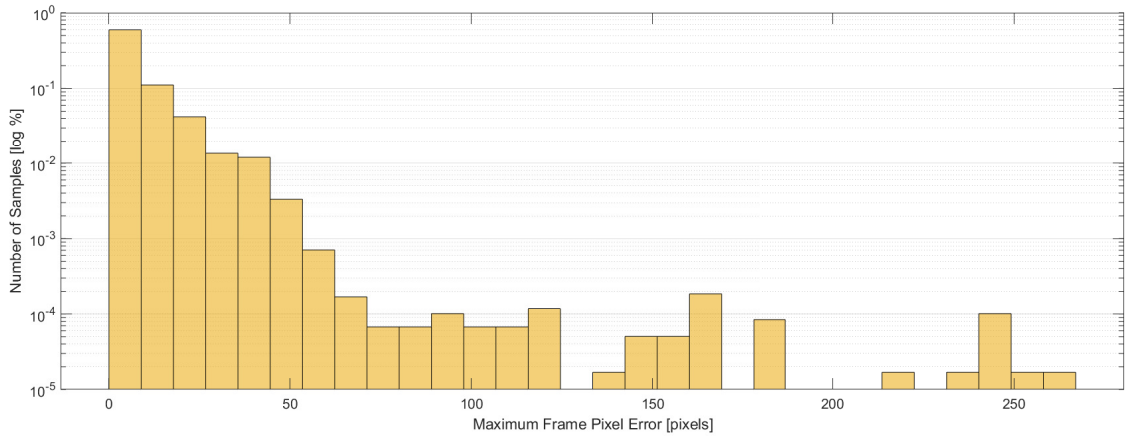
4.2 Accuracy

After observing the performance increase, we verify that our algorithm does not negatively impact the accuracy of the original algorithm. First we confirm that the discovered model is detected at the same scale as the original algorithm. Next we perform nearest neighbour between the key points in our results and those of the base algorithm. We then record the maximum pixel error and mean pixel error based on the result of the nearest neighbour search. This metric is only meaningful when both algorithms successfully find a model. In order to address this we record some additional statistics during the evaluation as shown in table 4.2. We pass the same frame to each algorithm, which each have a separate internal state. If a model is found by both algorithms they are compared as previously discussed. If both miss the model or only the modified algorithm finds a model, no penalty is recorded. Otherwise a symbolic penalty weight is recorded.

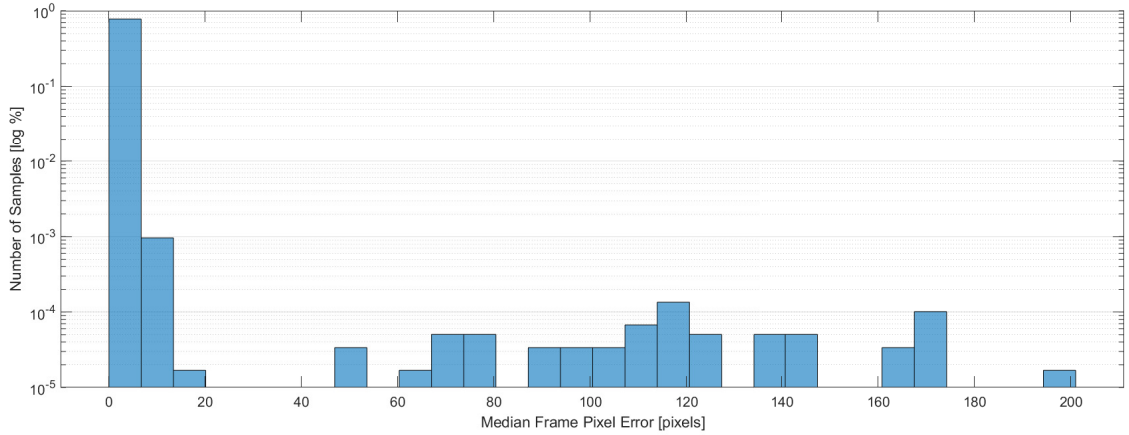
Table 4.2: Table of metrics kept during evaluation detailing if model tracking was lost, over 59349 frames

Found Model in Frame		Number of Frames
Original	Modified	
Yes	Yes	46233
No	No	12970
Yes	No	146
No	Yes	0

The two scenarios which introduce the most concern are listed at the bottom of the table. The second last entry having a non-zero value indicates that our changes have introduced enough error that tracking is occasionally lost. However the number of cases represents less than 0.25 % of the total frames. The final row being zero, indicates that our algorithm did not detect models where the original missed them. This is to be expected since our goal is to increase speed, with a low to marginal impact on accuracy. Since the number of frames where the modified algorithm was not able to find the model is so low, below details only the cases in which a model was found in both. It is important to note that both figs. 4.2a and 4.2b utilize a logarithmic vertical axis; the error would appear to be zero otherwise.



(a) Histogram of maximum pixel error



(b) Histogram of median pixel error

Figure 4.2: Histogram of error in YouTube Faces DB over 400 randomly selected videos, totalling 59350 frames

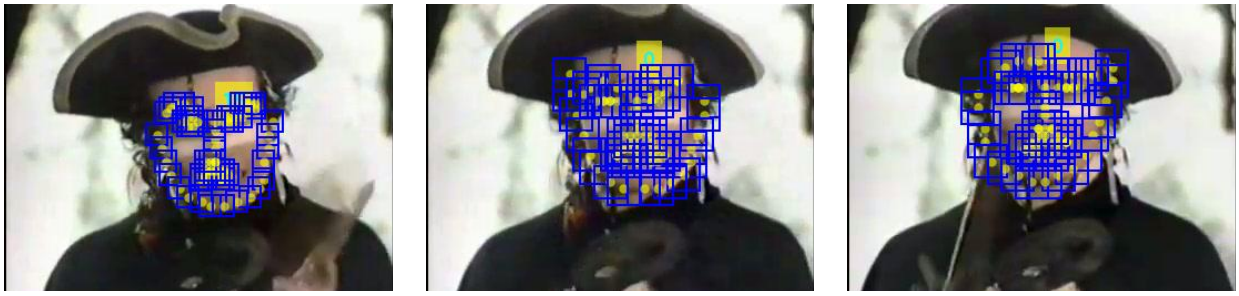
When comparing all the key points in a frame of the original algorithm and our modified version 84 % of the frames have less than 10 pixels of error. When considering the median error of all key points in a frame 99.9 % of the frames have less than 7.8 pixels. Our results suggest that is is safe to utilize the previously calculated HOG levels in very diverse environments as generally the pixel error is quite low.

4.3 Visual comparison

A brief visual comparison is shown by taking three frames, from a number of samples videos, with a five frame interval between each. The following examples have been selected to highlight the diverse environments: lighting, pose, and scale. For forward facing frontal portraits both the original algorithm and modified perform as expected, shown in fig. 4.3.

(i) Frame 5 (ii) Frame 10 (iii) Frame 15

(a) Original Implementation



(i) Frame 5 (ii) Frame 10 (iii) Frame 15

(b) Our Algorithm

Figure 4.3: Frontal portrait with uniform background

Shown in fig. 4.4 are frames from a video which contains a comparatively noisy background. There is a semi-round logo which could be confused for a face, that has been correctly ignored. The subject is both farther from the camera and rotated away, as compared to fig. 4.3. The original and modified algorithm both correctly identify the face and pose.

(i) Frame 5 (ii) Frame 10 (iii) Frame 15

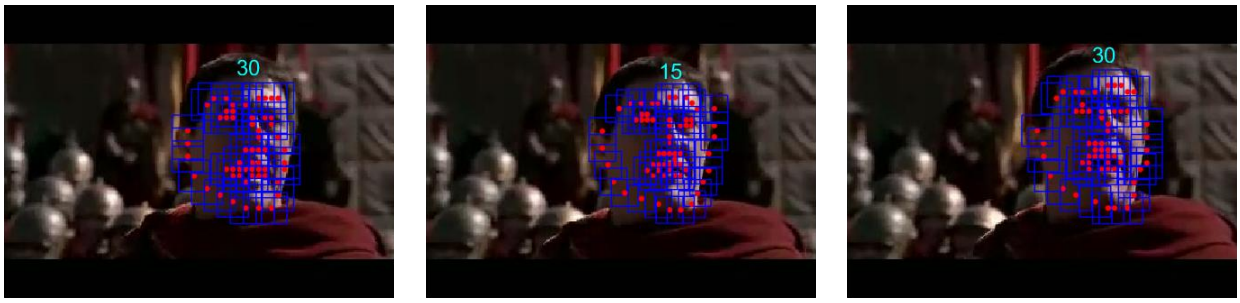
(a) Original Implementation

(i) Frame 5 (ii) Frame 10 (iii) Frame 15

(b) Our Algorithm

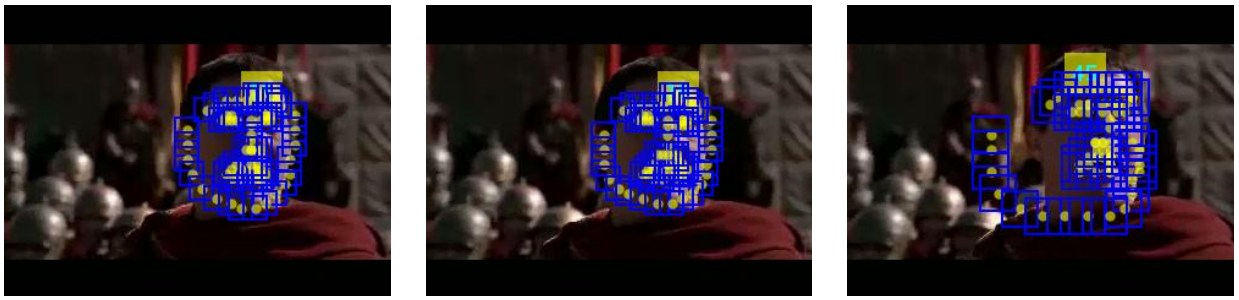
Figure 4.4: Oblique view with cluttered background

However as we can see in the see in third example of fig. 4.5b our modified algorithm occasionally de-synchronizes from the original algorithm. This scenario occurs when, both the policy network has failed to predict that de-synchronization will occur based on the current state and trigger a full update, as well as the actual cached features leading to a different outcome. This is not unexpected, and tends to only last for a single frame.



(i) Frame 5 (ii) Frame 10 (iii) Frame 15

(a) Original Implementation



(i) Frame 5 (ii) Frame 10 (iii) Frame 15

(b) Our Algorithm

Figure 4.5: Noisy background with many face like objects and difficult illumination

This kind of error can occur in the original algorithm as well, when a background object produces the same HOG feature that would normally correspond to a face feature; example shown in fig. 4.6.

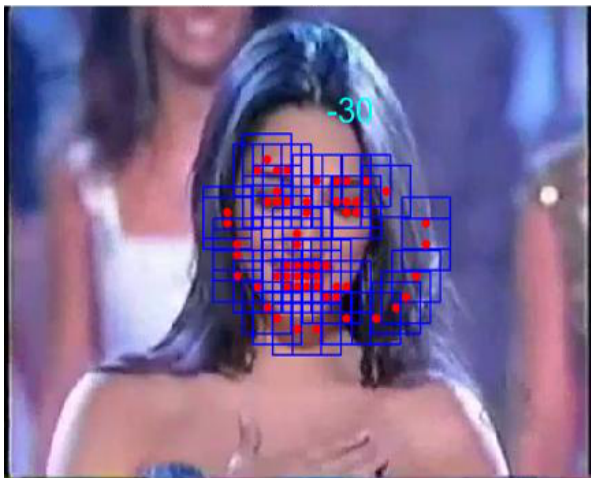


Figure 4.6: Original algorithm incorrectly identifying object (hair) as facial feature (jaw)

4.4 Policy Network

In order to show that the Bayes Net is providing good hints about when skipping updates will not introduce tracking errors, we record the Receiver Operating Characteristic (RoC) curve for the training input; a thirty second webcam video shot from slightly above the subject with a 60/40 train test split. Additionally to show that it generalizes we also test an uncorrelated video, from a Microsoft Kinect, shot looking up at the subject. Shown in fig. 4.7 we find that in general the prediction performs better than randomly guessing, shown as *No Knowledge*. The orange line represents flipping a coin to decide if a partial or full update is required to prevent our modified algorithm from diverging from the original algorithm. Perfect classification would be represented by a unit step function; that is 100 % True Positive, with 0 % False Positive. To better

understand RoC curve the function can be integrated to produce Area Under receiver operating characteristic Curve (AUC). The perfect score would then be an area of 1 A. Shown in fig. 4.7 the achieved AUC is 0.670 A.

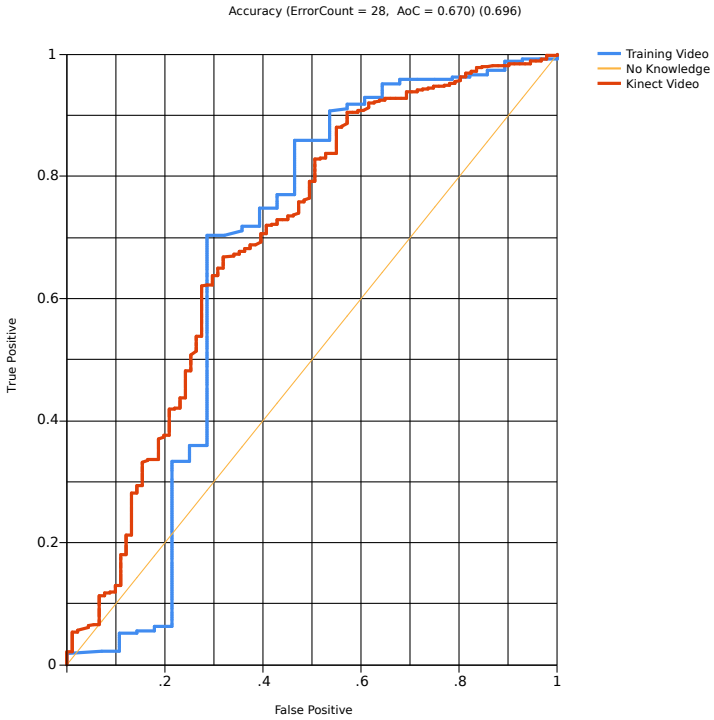


Figure 4.7: RoC curve for training, and testing video, including no knowledge (random selection)

4.5 Restricted Model Search

As previously discussed in section 3.3.3 we can reduce the number of models, or views that be checked against the computed features. The result is that the model candidate

space will be reduced. For each frame, the input image is fed through our algorithm and a set number of models are compared against the hog features to model candidates. In the original algorithm, all models are compared. In our modified algorithm, we are able to restrict it to adjacent models based on the results of the Bayes Net. In fig. 4.8 the affect of this, as well as the feature caching can be seen. The overall proportion between the models remains relatively the same between the algorithms. However our algorithm, has only checked 7 of the 13 models; 3 models adjacent to the previously detected.

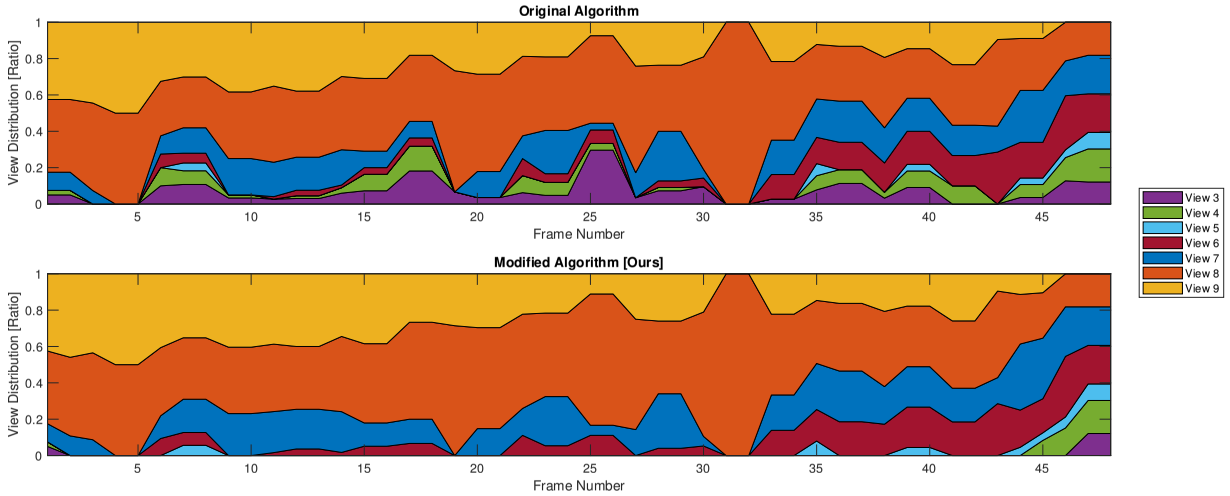


Figure 4.8: Comparison of view distribution between original and modified algorithm for each frame in a randomly selected video from YouTube Faces DB. Views with zero candidates have been excluded

4.6 Results Overview

As discussed in this chapter we confirm that suspicion about not requiring all feature levels to be updated is correct. More importantly we are able realize performance gain, with minimal impact on accuracy. The policy network we utilize is found to produce quality predictions, which helps to minimize error. Additionally the restricted model search is shown to remove low quality candidate parts without affecting the overall result.

Chapter 5

Conclusion

5.1 Summary

In this thesis a technique for improving the execution time of HOG pyramid based facial landmark tracking is proposed. This technique uses a Bayes Net as a policy network for determining when cached data from slightly stale frames can still be utilized. By avoiding calculations which do not contribute significant accuracy, the system is able to realized a large improvement in execution time. In order to reduce the overhead of the algorithm and policy network, the system takes advantage of lightweight feature inputs. The system is able to utilize motion vectors which are a normally an expensive feature, by reading them out of the H.264 encoding.

As shown in fig. 4.1 there are a low number of cases in which the algorithm takes the same or slightly greater time to complete. This is expected since the method for

determining if it is safe to use cached feature does have a cost, even though it is low. In videos where no cached features can be safely used, the penalty is still paid to determine this with no benefit gained. This is the trade-off for using such a simple policy network, when the prediction has too much uncertainty, the system must fall back to running the original algorithm. With regards to overall accuracy the modified algorithm does not differ significantly from the original method as shown in fig. 4.2a with respect to accuracy.

5.2 Future work

Moving forward to more complex input features and policy networks, careful attention needs to be kept on the performance penalty they introduce. As shown with with the H.264 motion vectors, it is often possible to leverage existing systems in use, often from slightly different disciplines.

5.2.1 Policy Network

Since the training of the Bayes Net is incredibly light weight, it may be possible to incorporate an online adaptive algorithm, which can further optimize away unnecessary feature level updates. This does not need to operate every frame and could be run at a slower rate in order not to increase the processing time. Alternatively it may be possible to use a Reinforcement Learning (RL) technique to discover a more complex policy at the cost of extra additionally training time and data.

5.2.2 Inter-Frame Position Smoothing

The focus of this research was to improve the execution performance of the existing method [1]. During the literature review many additional techniques were discovered. In a similar work [14] Kalman filters are used to successfully interpolate frames in which feature detection fails. Implementing such a technique may help to further decrease the number instances where the original and modified algorithm disagree.

5.2.3 Deep CNN

As NN based methods catch up in accuracy this technique may be similarly applicable to them. As previously mentioned it has a similar concept to [27]. It may be possible to utilize our lightweight policy network to gate calculations in a deep CNNs.

Bibliography

[1] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2879–2886. DOI: 10.1109/CVPR.2012.6248014.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[4] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "Openface 2.0: Facial behavior analysis toolkit," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, IEEE, 2018, pp. 59–66.

[5] S. Perreault, *Impaired driving in Canada*, ser. 2016001. 2016. [Online]. Available: https://www150.statcan.gc.ca/n1/en/daily-quotidien/161214/dq161214b-eng.pdf?st=tuX_xzbq.

[6] S. Rabba, M. Kyan, L. Gao, A. Quddus, A. S. Zandi, and L. Guan, "Discriminative robust gaze estimation using kernel-dmcca fusion," in *2018 IEEE International Symposium on Multimedia (ISM)*, IEEE, 2018, pp. 291–298.

[7] L. Wolf, T. Hassner, and I. Maoz, *Face recognition in unconstrained videos with matched background similarity*. IEEE, 2011.

[8] G. Higgins, L. Guan, A. Quddus, and A. S. Zandi, "Safely caching hog pyramid feature levels, to speed up facial landmark detection," in *Lecture Notes in Computer Science*, Springer, 2019.

[9] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, 2006, pp. 43–48.

[10] S. Ravikumar, C. Davidson, D. Kit, N. Campbell, L. Benedetti, and D. Cosker, "Reading between the dots: Combining 3d markers and facs classification for high-quality blendshape facial animation.," in *Graphics Interface*, 2016, pp. 143–151.

[11] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, "Multi-scale capture of facial geometry and motion," in *ACM transactions on graphics (TOG)*, ACM, vol. 26, 2007, p. 33.

[12] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image and Vision Computing*, vol. 28, no. 5, pp. 807–813, 2010.

[13] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 50–58.

[14] M. Uricár, V. Franc, and V. Hlavác, "Facial landmark tracking by tree-based deformable part model based detector," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–17.

[15] S. Xiao, S. Yan, and A. A. Kassim, "Facial landmark detection via progressive initialization," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 33–40.

[16] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[17] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[18] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5325–5334.

[19] H. Fan and E. Zhou, "Approaching human level facial landmark localization by deep learning," *Image and Vision Computing*, vol. 47, pp. 27–35, 2016.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *European conference on computer vision*, Springer, 2014, pp. 94–108.

[22] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3476–3483.

[23] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, IEEE, 2011, pp. 2144–2151.

[24] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *international Conference on computer vision & Pattern Recognition (CVPR'05)*, IEEE Computer Society, vol. 1, 2005, pp. 886–893.

[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[28] "Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," International Organization for Standardization, Geneva, CH, Standard, Sep. 2014.

[29] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions on image processing*, vol. 3, no. 3, pp. 327–331, 1994.

[30] T. Minka, J. Winn, J. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill, *Infer.NET 2.7*, Microsoft Research Cambridge. http://research.microsoft.com/infernet, 2018.

[31] R. Herbrich, T. Graepel, and C. Campbell, "Bayes point machines," *Journal of Machine Learning Research*, vol. 1, no. Aug, pp. 245–279, 2001.