MULTI-FACE TRACKING USING KANADE-LUCAS-TOMASI (KLT) ALGORITHM

Francois Legrand, Olivier Agbohoui, Vamshi Kodipaka

University Of Burgundy - Centre Universitaire Condorcet
Master in Computer Vision
Supervisor: Prof. Marc Blanchon

Multi-Sensor Fusion and Tracking

ABSTRACT

Our project emphasis on the implementation of a tracking feature algorithm through face detection process. The method focuses on a chained implementation of the Viola-Jones method used for real time face detection with the Lucas-Kanade-Tomasi tracker to follow them through a video. This project presents a system able to operate close to real time over high quality video, while staying robust and reliable through common situation use. Multi-face tracking in the videos is possible.

Index Terms— *Face Detection, Feature Tracking, KLT, Viola-Jones, BRISK, Kanade, Similarity*

1. INTRODUCTION

1.1. Aim and Context

Visual Tracking belongs to those newly expanding domains, and is still subject of exclusive explorations and refinements. In context to facial recognition, it gives higher scope than biometric recognition methods like iris and fingerprint analysis. Highly secure and difficult to hack, makes it increasingly crucial for use in security systems. Usually tracking using the facial features of moving human in recorded videos or a live surveillance cameras are defined as Face Tracking. In 1964, Bledsoe, Helen Chan and Charles Bisson, first worked on computer based vision techniques on human faces. Etienne Corvee and Francois Bremond led strong ideas of combining face detection and people tracking in video sequences in 2009.

In 2018, Ranganatha S and Y P Gowramma **[1]** published a paper on Development of Robust Multiple Face Tracking Algorithm and Novel Performance Evaluation Metrics for Different Background Video Sequences. Some of the major issues these tools were encountering at the beginning were directly linked to the accuracy.

Face Tracking is specifically one of the most important and evolving branch in the domain of artificial vision. With this context in mind, this project came into implementation and development. In author prespective the performance of the latest face recognition algorithms. These were tested in the Face Recognition Competitions using High-resolution face images, 3-D face scanning, and iris images.

The results denotes that the newer algorithms are much better in terms of accuracy than once developed around the year 2002. This topic of research and conception encounters many challenges depending of the extended range of possible applications it can have. These include occlusion, movement speed,image quality, face expressions, all of them seen during the development and testing.

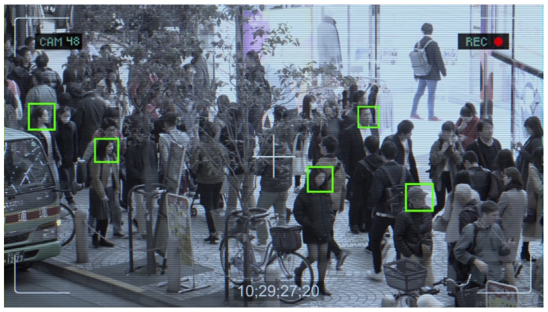


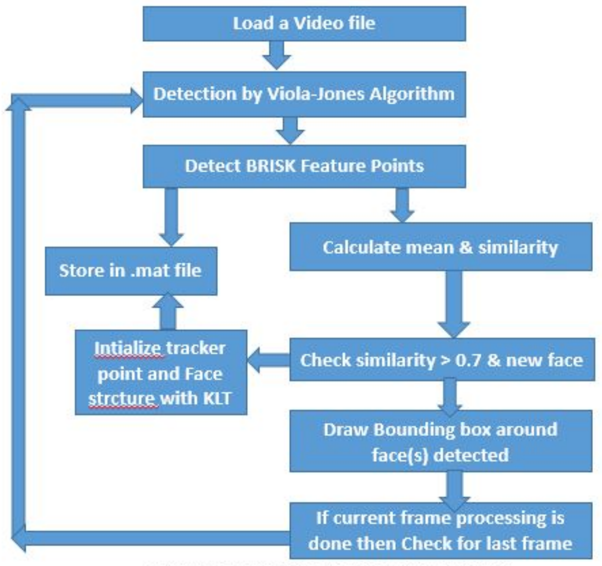
Fig1: Example for Moving Face Tracking in Videos

The aim here is to perform an efficient face detection as well as tracking, while taking into account these different parameters. Moreover, it is also feasible towards multiple face detection and tracking on single frame, relying on the already existing algorithms. Ranganatha's work will more explains our project idea. The motto of video tracking is to associate target moving objects in consecutive video frames. However we observe the evolution of Dynamic Vision System where camera is static and inspect the moving faces to a stage of tracking the moving faces in the moving camera. The association can be especially difficult when the real objects are moving quicker compared to the frame rate. Other example is case of increase in complexity of the problem when the tracked object changes orientation over time.

For such cases video tracking methods generally a motion model is deployed which describes the image of the target changes for various expected motions of the object. Other important consideration one has to take into account is the choice of features based on the time complexity and the space or the dimensionality. We are already introduced to various visual tracking algorithms based the intensity, features, area, edges and the prototyping. The difficulty of tacking face(s) is observed when the captured video has a low frame rate, the developed system might lose track of the face or skip from one frame to another. As a result, when the person moves within the frame, the system may identify the same face as two separate person face(s). To avoid this problem, the normal or recommended in particular with minimum frame rate is of 6 frames per second (fps) for accurate Face Tracking.

1.2. Framework

The first part of the project is focused toward face detection. This is a mandatory step of the program as it serves as a basis for more in-depth processing actions. In our case, feature tracking will be possible only if a face is properly detected. In the provided implementation, the Viola Jones face detection algorithm associated to the BRISK feature points which are then used in detection of faces through video frames. This part of the process offers the opportunity to perform tracking of the previously detected features. Below **fig.2** explains the implementation.


Load a Video file
Detection by Viola-Jones Algorithm
Detect BRISK Feature Points
Store in .mat file
Calculate mean & similarity
Intialize tracker point and Face strcture with KLT
Check similarity > 0.7 & new face
Draw Bounding box around face(s) detected
If current frame processing is done then Check for last frame

BLOCK REPRESENTATION OF THE PROJECT

Then the next step is tracking through the use of the Kanade-Lucas-Tomasi tracking algorithm. In our case it allows us to track through self-similarity features in a constrained bounding box around detected faces. The combination of both delivers an efficient face detection and tracking tool which can be performed in real time for standard video resolution.

2. METHODOLOGY

This project will first discuss about the methods employed to build the Face Tracking algorithm, then gives the implementation and detailed structure of the program, as well as its tool reshaping through a GUI, before observing to observations and results. We reported exactly as explained. KLT comes under Deterministic methods in which iteratively search is performed for a local maximum of a similarity measure between a template of the target and the current frame.

2.1. Viola-Jones face Detection

The first part of the presented algorithm consists into performing an efficient face detection. To do so, an implementation of the Viola-Jones face detector has been used. Viola–Jones object detection architecture is the first object detection architecture to provide efficient and competitive object detection rates in real-time which was proposed in 2001 by Paul Viola and Michael Jones. According to the authors **[2]**, this method is considered as robust when performing front face detection, and uses basic features which are passed through a boosted perceptron network.

In details, the features used here are called "The Haar" features, and mimic in a simple yet efficient way human faces through simple kernels **fig.3** . These features are then passed into an integral image which will be used by an embedded Ada-boost classifier. The cascade object detector (System Object in Matlab) uses the Viola-Jones algorithm to detect people's faces, noses, eyes and mouth etc. Adaboost classifier performs a classification based on features weights, but keeps only the most important ones, which are in this case linked to the Haar features. Given a set of weak classifiers.

$$h_j(\mathbf{x}) \in \{+1, -1\}$$

Iteratively combine classifiers by:

$$C(x) = \theta\left(\sum_t h_t(x) + b\right)$$

The final classifier is linear combination of weak classifiers.

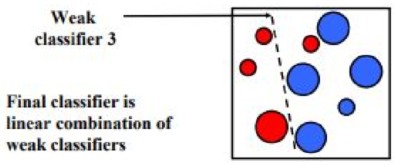


Fig.3: Adaboost with Haar Feature classifier representation.

Boosted Face Detection with Image Features is given by:

$$h_t(x_i) = \begin{cases} \alpha_t & \text{if } f_t(x_i) > \theta_t \\ \beta_t & \text{otherwise} \end{cases}$$

$$C(x) = \theta\left(\sum_t h_t(x) + b\right)$$

Although it is linked to machine learning, it differs in the sense that it only keeps a few weights, and constrains the prediction part to those weights. The minor ones being discarded **[3]**. Haar Features explained through the figure.

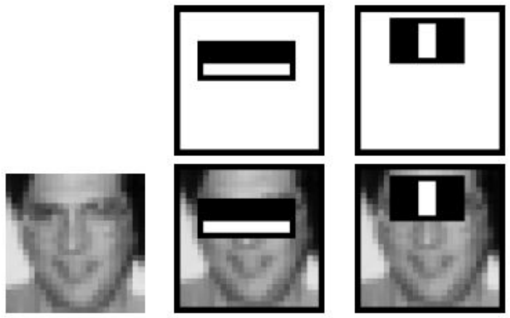


Fig.4: Haar Features representation

For each round of boosting one has to evaluate each rectangle filter on each example, sort examples by filter values, select best threshold for each filter, select best filter/threshold (= Feature) – Reweight examples. Also have to choose M filters, T thresholds, N examples, L learning time as in O(MT L(MTN)) for Naïve Wrapper Method and in order of O(MN) for Adaboost feature selector.

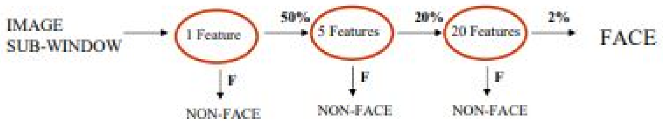


Fig.5: Feature check through the sub-window

Once classified, the results are injected one by one into a cascade architecture. This part of the process is the most important as it allows focusing only on potential positive results across the frame, hence constraining detection to a much smaller window, and in consequence reducing the computation power by a wide margin. Moreover, the step does not have an influence on previously classified results, meaning no losses from the previously acquired classifications. Making prediction, reduces the search area is the advantageous step and template matching can do the job efficiently.

2.2. Binary Robust Invariant Scalable KeyPoints (BRISK)

Once a face has been detected, a BRISK points detector is performed inside the previously found bounding box. The ”Binary Robust Invariant Scalable Keypoint” algorithm (BRISK) point detector allows the program to compute self similarities between frame points, to help preventing face detection recomputation, and mainly allowing the multiple faces detection. **[6]**. A comprehensive evaluation on benchmark datasets unfolds BRISK’s adaptive, better quality performance as in state-of-the-art techniques, though at a dramatically lower computational cost.

2.3. Kanade-Lucas-Tomasi Tracking

In 1981, two seminal works on optical flow estimation were published, namely the works of Lucas and Kanade, and of Horn Schunck. Lucas-Kanade methods gives sparse flow vectors under the assumption of constant motion in local neighborhood.

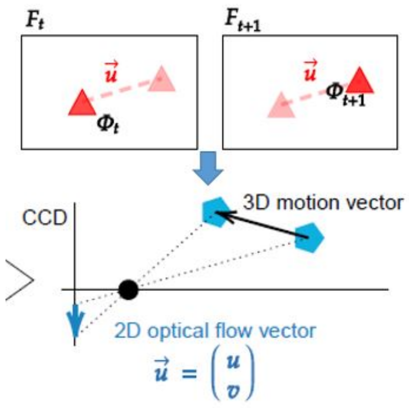


Fig.6: Base idea for Optical Flow used for KLT

Equation represents the variation of intensity w.r.t the spatial coordinates and time framed in under-determined system, one equation, two unknowns. The detected feature are moved between frame F(t) and frame F(t+1) according to a velocity vector v = dx/dt or in pixel coordinate **u** = (u v)’. **[8]**. With brightness constancy Eqn: I(x; y; t) = I(x + dx; y + dy; t + dt) **[8]** Determined points are given to a KLT tracking tool.

$$\boxed{I_x u + I_y v = -I_t}$$

In computer vision, the Kanade–Lucas–Tomasi feature tracker is an approach considered to feature tracking. It is proposed mainly for the sake of solving the issue of formal image registration process and generally are costly. KLT makes use of spatial domain’s intensity data to direct the search for the position that yields the best match including the feature vector.

$$\frac{d}{dt}I(x(t),t) = \nabla I^T\left(\frac{dx}{dt}\right) + \frac{\delta I}{\delta t} = 0.$$
$$\nabla I(x',t)^T v + \frac{\delta I}{\delta t} = 0$$

Equation: Brightness Constancy assumption with optical flow constraint and Constant motion in a neighborhood

It is more fast than formal method for testing far fewer potential matches between the images.The point tracker object (Matlab) tracks a set of points using the Kanade-Lucas-Tomasi (KLT) feature-tracking algorithm. Lucas and Kanade (1981) therefore compute the best velocity vector v for the point x by minimizing the least squares error.

$$E(v) = \int_{W(x)} |\nabla I(x',t)^T v + I_t(x',t)|^2 dx'.$$

Estimating Local Displacements is given by: Translational Motion and Affine Motion. **[10]**

$$E(b) = \int_{W(x)} |\nabla I^T b + I_t|^2 dx'$$

$$E(b) = \int_{W(x)} |\nabla I(x')^T S(x')p + I_t(x')|^2 dx'$$

$$S(x)p = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \begin{pmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{pmatrix}^T.$$

First two equations derivated and are equated to zero. In the formalism of Lucas and Kanade, one cannot always estimate a translational motion **[7]**. The problem is often referred to as the aperture problem. It arises for example, if the region in window W(x) around the point x has entirely constant intensity (white wall for example), because then grad(I(x)) = 0 and Intensityt(x) = 0 for all points in the window. To guarantee a solution of b to be unique, the structure tensor.

$$M(x) = \int_{W(x)} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} dx'$$

needs to be invertible, i.e. det(M) not equal to 0. If the structure tensor is not invertible but not zero, then one can estimate the normal motion, i.e. motion in direction of image gradient. For those points with det(M) not equal to 0, we can compute a motion vector b(x). This leads to the following feature tracker with modification called KLT tracker. the local velocity is given by:

$$b(x,t) = -M(x)^1 \begin{pmatrix} \int I_x I_t dx' \\ \int I_y I_t dx' \end{pmatrix}$$

Local smoothness is given by:

$$I_x u + I_y v = -I_t \quad \Rightarrow \quad \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

Further we want to find u and Linear Least Square solution:

$$\hat{u} = (A^T A)^{-1} A^T b$$

We finally need the matrix transpose(A).A to be invertible.

$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Based on transpose(A).A good Features to Track is explained clearly**[10]**. It is usually adopted for video stabilization, camera motion estimation, and object tracking. It works extremely well for tracking objects that do not change shape and for those that exhibit visual texture and thus often employed for short-term tracking part framing in large tracking architecture. As the point tracker algorithm progresses over time, points can be lost due to lighting variation, out of plane rotation, or articulated motion. To track an object over a long period of time, you may need to reacquire points periodically.

3. IMPLEMENTATION

3.1. Face Detection

The previously given methodology has been implemented into a program linked to this report, following the two main steps described further in **fig.9**.

Step-1: The face detection part is initialized using a simple model and can be considered as having a weak prior knowledge. This model is constrained between a maximum and minimum bounding-box sizes of 250 by 250 and 70 by 70 pixels. It is also settled for full resolution detection, meaning no sub-sampling before process. The last parameter to take into account lays in its sensitivity. A default sensitivity of 10 is given. This value can be decreased to give more detection, but will also rise the issue of being too responsive, leading to false positive detection.

Step-2: Viola-Jones algorithm is then performed through an in-built function. This method uses feature-based instead of pixel-based classifier, which is important in this case as according to the authors**[2]**, the computation time of features based algorithm is faster than pixel-based ones, which helps improving computation time for real time detection. Those features, inspired by Haar basic patterns **[8]**, are then compared to extract differences, before being combined to make an integral image.

Step-3: The function uses 'FrontalFaceCART' classification model which is based on Classification And Reduction Trees method, described in detail by Breiman. et all **[4]**. As we are performing detection on the entire frame, no ROI are used here. The pCascadeClassifier taken from Opencv is called inside the function, and will be used to the cascading detection, the step following the Adaboost part. A hidden constrained training function is also used to set the lower boundary for object size detection. This classifier works in a similar manner to random forests one **[5]**. The frames can be converted to grayscale to avoid loss of time on color channel computation, as well as associated errors if any.

Step-4: The model is given through the function as a set of pre-established parameters. The Haar 's like features are encoded through .xml framing language file. These features are each associated to one specific facial characteristic, up to a total of 12 different pattern models. The integral image serves as a basis for the classifier. Adaboost method is applied on any basic classification algorithm to accelerate them through fusion of weak weighted functions. This part allows the reduction of parameters to a lower quantity with enhanced classification result, and by extension the improvement of time performance and the time computation issue is however vastly increased during the last step of the Viola-Jones method and used to accelerate the classification, its application has been done prior the function implementation and is not directly used here.

Step-5: Cascade of the detection allows for a proper rejection of negative results through all the generated sub-windows with the obtained integral image, while keeping mainly the positive ones. The process, being in itself similar to the Adaboost is performed on a wider scale, as this time the overall result of sub-windows are taken into account instead of the sub-window level computational ones. It is then redone and improved through the selected sub-windows iterating on all the video frames through the association of a variable to the detection method. The function then initialize the tracker points at the first iteration, or retrieve the previous ones from an already existing .mat file (The function runs the tracking avoiding processing delay)

Step-6: Once performed, the obtained bounding box is again constrained by the previously given model at the beginning. The bounding box is drawn and fixed through a run system object algorithm which compares the detection region to the video frame. To introduce the first results and cover the several faces issue, the program will run trough a multiple face detection function. When several detection are present, a similarity comparison is done between them. If the similarity is below a threshold of 0.6, the program considers a new face has been detected and will be added to the tracking.

$$\text{Similarity} = \left(\frac{\varphi}{\Psi 1} + \frac{\varphi}{\Psi 2}\right) / 2$$

3.2. Face Tracking

Prior to the Kanade-Lucas-Tomasi algorithm in the case of a new detection, the BRISK point extraction is performed inside the previously obtained bounding box.This step is necessary as it will allow the tracking necessary to keep the bounding box active around the face and avoid detection computation for each frame.

Step-1: The Binary Robust Scalable Invariant Keypoints has been chosen here due to its fast and reliable feature point detection **[6]**. The detection of stable feature points on a face is important since the program will focus mainly on tracking once done. AGAST algorithm **[9]**, which is an accelerated performance wise version of the FAST feature extractor. The algorithm extracts keypoints using Scale Space detection after performing the time optimized FAST algorithm. To do, so, the function uses recursive downsampling, while keeping the original 9-16 rule, in which 9 consecutive pixels must have a brightness difference sufficiently different from the central pixel in a circular mask of 16 pixels to put the later a potential candidate for feature detection.

Flow Chart:.

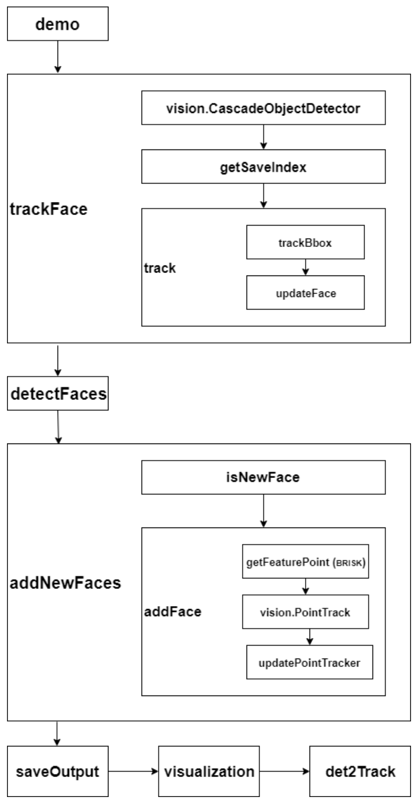


Fig.7: Flow chart explanation program sequential flow.

This approach is also reminiscent of the work of Smith .et all **[11]**. The down-sampled layers in the scaled space are then compared to each other to determine where lies the maximum, resulting in a precise coordinates position of the BRISK feature to extract after further refinement steps. The obtained coordinates are then averaged to the nearest neighbor pixel.

Step-2: Inside the current implementation, the BRISK feature detector is performed Within the previously defined bounding boxes. The number of obtained points vary in function of the detection. The contrast limit is set at 0.001 and is lower than the smallest gray level increment in an 8 bit encoded video frame.

Step-3: The trackers are attached to the extracted features, and given the KLT function. Here, a window around the determined feature is given since the tracking of a single pixel is nearly impossible and heavily constrained by changing environment parameters **[7]**. The Window tracking brings two other problems through changing similarity between frames and non-uniform displacements.

Step-4: The tracking is performed with the default value of 30 iterations before reaching convergence for each point to track. The prediction windows are settled to 31 by 31 pixels. In a similar fashion to the BRISK feature extraction part with Scaled Space, the function performs pyramidal down sampling by a factor of two each time, for three iterations. The more the iterations are performed, the more the possible amplitude for maximum displacement will increase, though with a decrease of time-wise performance. In the function, the frames are processed in gray scale.

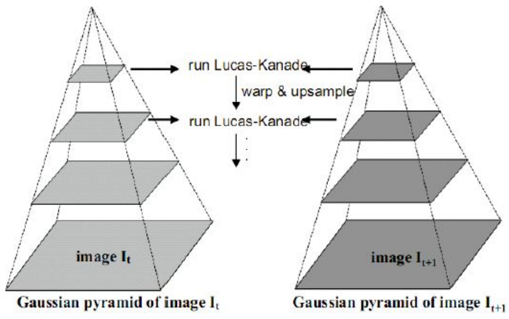


Fig.8: Pyramidal KLT analysis

Step-5: These can be however solved through self-similarity checking like in the previous detection part for the first one, and through transformations, allowing pixel by pixel vector characterization of the window. The self similarity computation uses a higher threshold than the previously given one. The tracking and by extension the detection are interrupted if the similarity ratio between two consecutive frame bounding boxes falls below 0.7.

Step6: The new detection data set is added to the existing ones and drawn onto the current analyzed frame, setting the way for tracking in the next loop. in the case the trackers were detected previously, the points are updated and randomly resampled prior to tracking in the next frame.

Step-7: Every 1000 frames, the output is saved for process monitoring and to constitute the .mat data file. The visualization is then performed when the process is run the first time on the video. The display shows the locations of the detection through bounding boxes and the detected BRISK features points. If the process is redone, then the program displays only the boxes and reads the acquired data. In a last step, a function is used to convert any newly detected feature into the track readable data, as they might serve in case of future tracking within the following frames.

4. RESULTS

The GUI created in this project helps us to visualize the results from the program and manipulate them. The Algorithm performs proper detection most of the time, however with respect to the condition that the face to track must be front-facing the camera.

Some issues are observed in case the face is misaligned with the camera plane. In a video example, a face which should be put in evidence by the algorithm was not detected because of being seen from profile. One can also note that the detection is not robust against motion blur, an issue which is still a challenge to solve nowadays.

4.1. Limitations

This problem comes also on part with too important displacement between two frames, in which case the detection can work for each of them but not the tracking in between. The issue could be solved by increasing pyramidal down-sampling, yet at the cost of power and time consumption rendering the program not usable in real time. Moreover some false detection will also occur in those cases.

The resolution is also the last major concern which comes in mind when performing face detection. If too low, no detection will occur due to the constraints of the assumed model size for bounding boxes, as well as lack of sharp features in some cases (up sampled low-resolution videos). On the other side, a too important resolution will lead to a higher computation time rendering the real time process more difficult to achieve.

4.2. Advantages

Within a regular and common usage, the program performs proper detection, with varying number of tracked features per detection. The code runs in real time with compressed video data. The implementation can be also done on mid-range systems for real time detection and tracking. A simple yet efficient user interface has also been implemented to help the user visualize the ongoing process when the algorithm is performed. Moreover, the system is able to reliably track several faces at the same time (**fig.9**).

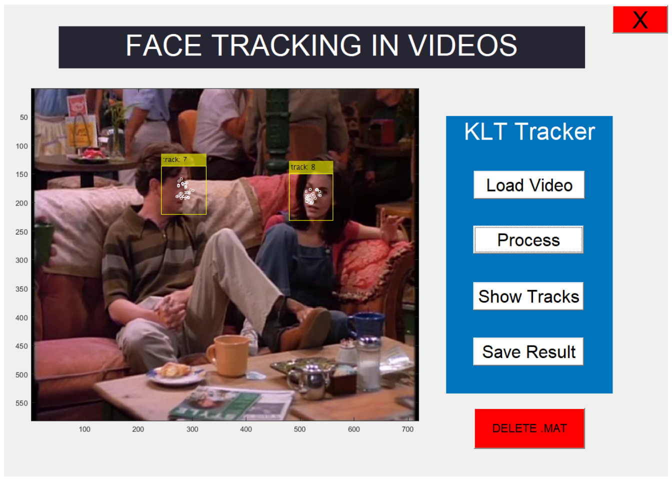


Fig.9: Output of Multiple Face detection

5. CONCLUSION

We presented during this project an algorithm to perform face tracking in real time over various video format. This program is joined with a Graphical user interface which can be adapted to do real time people faces detection and tracking through simple camera device. The algorithm performs quite well on mid-range system, while delivering interesting results.

6. REFERENCES

[1] Ranganatha S, Y P Gowramma, Development of Robust Multiple Face Tracking Algorithm and Novel Performance Evaluation Metrics for Different Background Video Sequences,I.J. Intelligent Systems and Applications, 2018, 8, 19-35.

[2] Viola, Paul & Jones, Michael. (2004). Robust Real-Time Face Detection. International Journal of Computer Vision. 57. 137-154. 10.1023/B:VISI.0000013087.49260.fb.

[3] Schapire, Robert. (2013). Explaining AdaBoost. 10.1007/978-3-642-41136-6_5.

[4] Breiman, L., J. Friedman, R. Olshen, and C. Stone, 1984: Classification and regression trees. Wadsworth Books, 358

[5] Breiman, Leo (2001). ”Random Forests”. Machine Learning 45 (1): 5–32. doi:10.1023/A: 1010933404324

[6] Leutenegger, Stefan & Chli, Margarita Siegwart, Roland. (2011). BRISK: Binary Robust invariant scalable keypoints. Proceedings of the IEEE International Conference on Computer Vision. 2548-2555. 10.1109/ICCV.2011.6126542.

[7] Tomasi, Carlo & Kanade, Takeo. (1999). Shape and Motion from Image Streams: a Factorization Method—Part 3 Detection and Tracking of Point Features Technical Report CMU-CS-91-132.

[8] Papageorgiou, C.P. Oren, Michael Poggio, Tomaso. (1998). General framework for object detection. Proceedings of the IEEE International Conference on Computer Vision. 6:. 555 - 562. 10.1109/ICCV.1998.710772.

[9] Adaptive and generic corner detection based on the accelerated segment test. / Mair, Elmar; Hager, Gregory; Burschka, Darius; Suppa, Michael; Hirzinger, Gerhard. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 6312 LNCS PART 2. ed. 2010. p. 183-196 (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 6312 LNCS, No. PART 2).

[10] Equations are referred from lecture slides of Mr. Marc Blanchon, (Centre Universitaire Condorcet MSCV2 2019-2020), VISUAL TRACKING: Multi-Sensor Fusion and Tracking.

[11] Dr. Smith, S. & Brady, Michael. (1997). A New Approach to Low Level Image Processing Tech. International Journal of Computer Vision. 23.