Proceedings of the International Conference on Inventive Computation Technologies (ICICT-2018)
IEEE Xplore Part Number: CFP18F70-ART; ISBN:978-1-5386-4985-5

Evaluation and Review of Object Tracking Solutions for Video Sequences

Azra Nasreen*, Chethan K P†, Rahul M Patil‡, Shobha G
Department of Computer Science and Engineering
Rashtreeya Vidyalaya College of Engineering, Bangalore 560059, Karnataka, India
{chethan749†, patilmrahul06‡}@gmail.com, azranasreen@rvce.edu.in*

***Abstract*—A great interest has been fueled in object tracking solutions, mainly because of factors such as an increase in the number of powerful computers, the availability of high-quality video cameras, and the need for automation in video analysis. Tracking plays a vital role in various application domains, such as traffic monitoring, object recognition, autonomous surveillance systems, defence etc. This paper conducts a concise review on some of the few important techniques that are being used for robust tracking of moving objects in videos, completely along with their merits and demerits. Also, a thorough performance analysis of popular techniques – Mean Shift, CamShift, Optical Flow, Kalman Filter and SURF – has been performed, and the results have been interpreted and illustrated.**

***Index Terms*—Object Tracking, Video Analysis, Mean Shift, CamShift, Kalman Filter, Optical Flow, SURF**

I. Introduction

Object tracking is an important application in the field of Computer Vision, as it makes analysing videos much more productive and useful in several application domains. Before the actual tracking is initiated, the object of interest must be detected and segmented, so that it can be tracked in the video sequence. To carry out further analysis, these detected objects must be tracked in consecutive video frames. The main goal of object tracking is to find an object's location in each and every one of the consecutive video frames. Few of the many important applications of this is control for autonomous surveillance systems, identifying threats in missile defence, optimizing traffic control systems, and improving human-computer interaction.

Figure 1 shows the general process of object tracking in video sequences. The step by step procedure used to track a moving object can be summarized as follows:

- *Segmentation:* It refers to identifying the components of image in the region of interest.
- *Foreground Extraction:* It is the process of extracting the foreground of an image.
- *Feature Extraction:* Extracting useful features from sequence of frames required for tracking.
- *Tracking:* It is the actual tracking of objects based on the features extracted.

Object segmentation is an integral part of any video analysis system. Many approaches are proposed to detect moving objects in videos, the most popular being the Background Subtraction method. Foreground modelling using K-means clustering [1] is another method that extracts the foreground objects and works well, even for variable frame rate videos. The blobs of the foreground objects need to be extracted accurately, regardless of the presence of noise, any illumination changes, variable frame rates, lag etc. otherwise tracking efficiency will decrease. Many different object tracking algorithms have been proposed, including Mean Shift tracking, Optical Flow, feature matching, etc. Every algorithm has its own strengths, which depends upon the type of video recorded, it's resolution, adaptability to illumination changes, compatibility with variable frame rates, sudden changes in the speed of the moving object, etc. A robust tracking algorithm should be able to continuously track multiple moving objects, irrespective of noise and any other types of disturbances encountered.

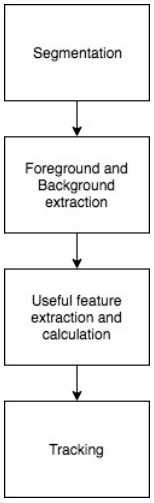


Fig. 1: Object Tracking

II. Tracking of Moving objects in Videos

To track an object, it has to be segmented first and the ROI (Region of Interest) should be obtained using either region based or boundary based approaches. The most popular one being the Background Subtraction algorithm, wherein the background is initially modeled, and the foreground is extracted using Background Subtraction. Once the foreground masks are obtained, the tracking process will follow. There are several techniques available for tracking of moving objects, some of which are described in the following sections.

A. Mean Shift

Mean Shift is a versatile and powerful non parametric algorithm, which is iterative [2] in computation. This algorithm can be used for many purposes, such as finding modes, tracking, clustering, etc. Mean Shift considers the set of points given, as sampled by the underlying probability density function. If dense regions are present in the feature space, then they correspond to the local maxima of the probability density function. Essentially, the Mean Shift algorithm approximates the maxima which are nearest to the initial state. Mode calculation by Mean Shift is explained as follows. Initially, a window is defined around the object to be tracked, and then the centroid data-point within the window is computed. Finally, the centre of the window is shifted to the computed centroid, and the algorithm repeats iteratively until it converges. After each iteration, the window is shifted to the denser regions of the dataset.

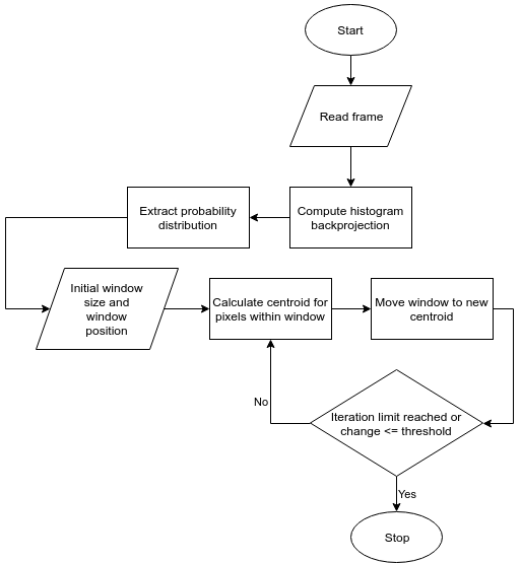


Fig. 2: Mean Shift Pipeline

The main principle that is used for mode detection, is also applied in videos for tracking objects. The Mean Shift tracking pipeline is shown in Figure 2. Initially, a frame is read and the hue component of the frame is extracted, and a histogram is computed using this data. The ROI containing the object is selected, and histogram is computed for the hue component of this ROI as well. The ROI histogram is then back-projected onto the histogram for the entire frame, to get a probability distribution of the object. In this distribution, pixels similar to the object have a higher probability density than the ones that are not, effectively making the region within the frame that has the object to be tracked, to be the maxima (mode). When Mean Shift is applied, it performs mode calculation over the probability distribution obtained. Since the region containing the object is the mode in the probability distribution, the Mean Shift tracking window is able to converge to the object and hence, is able to track the object effectively. This probability distribution is calculated for every frame in the video sequence, and the window is updated accordingly.

The advantage of Mean Shift is that it is good for tracking a single object, which does not have similar color as the background. It is computationally efficient, therefore it is used for real-time tracking of objects. That said, it fails when the orientation of the object changes in consecutive frames. It also fails to adapt to the changing scale of the object that is caused by the object moving either closer or farther relative to it's initial position.

B. CamShift

The Continuously Adaptive Mean Shift Algorithm (CamShift) is an extension of the Mean Shift algorithm. It uses the one-dimensional hue channel histogram to track objects [4]. Fundamentally, the histogram is the tracked object's color probability map. Mainly this algorithm has been developed for tracking faces in videos. It computes the probability of every pixel element that makes up the tracked object. If a color probability distribution is given, CamShift uses the Mean Shift algorithm iteratively to find the centroid of the probability image. This centroid of the image will be used as the centre for calculating another color probability distribution, to be used in the next frame. This process of calculating color probability distribution repeats in each and every frame. CamShift adapts its color histogram continuously in each frame, and hence, the name Continuously Adaptive Mean Shift Algorithm.

The property that makes CamShift adaptive, is it's ability to scale according to the object or the distribution. CamShift can be used for mode calculation, just like Mean Shift. In CamShift, initially Mean Shift is applied on an initial window size, until convergence is obtained. On convergence, the window size is changed in accordance to equation 1:

$$w = 2\sqrt{\frac{M_{00}}{256}} \tag{1}$$

Where, w is the width of the new window and M_{00} is the zero order moment of the image. Height h of the image is then given by equation 2:

$$h = 1.2 \times w \tag{2}$$

The workflow of the algorithm is very similar to Mean Shift, with the additional ability of window scaling that is shown in Figure 3. The initial window is drawn around the target object. When the next frame is read, Mean Shift is used till convergence is reached. Then the window size is changed in the following frame, defined by the ellipse derived from the second order image moments. This new window is then used for Mean Shift, until convergence for the next frame. This is repeated for successive frames of the video sequence, and hence the window scales with respect to the target object.

CamShift is very good for tracking a single moving object, even among multiple objects. Once the ROI for the target object is defined, it is adaptive to the scale of the object as

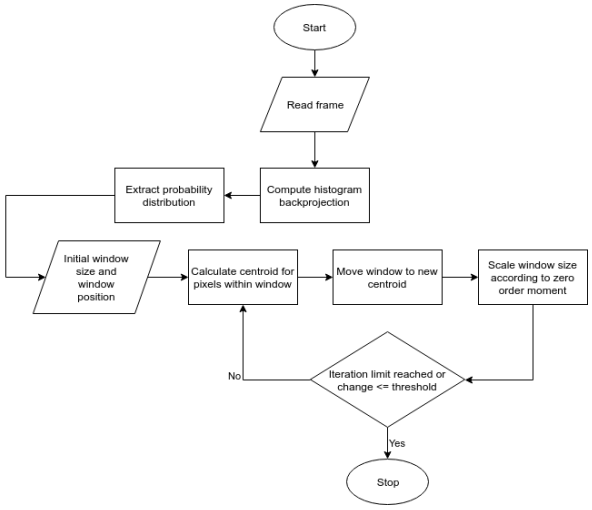


Fig. 3: CamShift Pipeline

well. The algorithm is fairly quick, but not as quick as Mean Shift. It fails when the objects are similarly colored.

C. SURF (Speeded Up Robust Features)

SURF is a feature detector and feature descriptor. It makes the best use of integral images [5] [10]. SURF is based on responses of approximated $2D$ Haar wavelets. Interest points are the points in the image at distinct locations such as corners, T-junctions, blobs etc. Feature vectors are used to represent neighbourhood of each interest point. SURF is scale and rotation invariant, because of the use of orientation descriptors and feature vectors. The feature vectors between target versus scene images are matched by SIFT (Scale Invariant Feature Transform). The distance between the vectors can be used for the matching process.

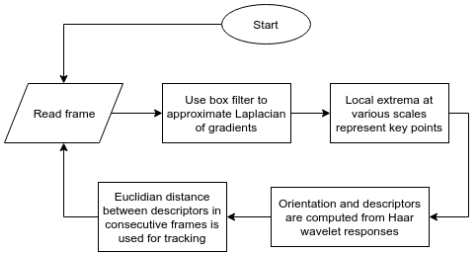


Fig. 4: SURF Pipeline

SURF algorithm detects key points within the image and also describes those key points using a 64 or 128 dimensional vector. The steps involved in this algorithm are summarized in Figure 4. The first step is to find the approximation of the Laplacian of Gaussians in order to represent the image at various scales, and to extract the feature points. This can be computed very efficiently using the integral image along with a box filter. The feature points are where the determinant of Hessian matrix H is maximal:

$$H(p,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix} \tag{3}$$

Where, p is a point (x, y) in the image, and σ is the standard deviation parameter of the Gaussian kernel that is convoluted on the image. It represents the scale of the image. L_{xx} refers to the second order derivative of the convolution of the Gaussian, with the image at point p. Once these feature points are detected, orientation is assigned and descriptor is generated using Haar responses in the x and y directions. For orientation, the responses over a neighborhood of $6s$ is calculated, and weighted with the Gaussian function centered at the interest point and plotted, where s is the scale at which the interest point was generated. The orientation is then determined using a sliding orientation window of $60°$. The descriptor is computed using a $20s$ neighborhood around the interest point, which is done by taking the summation of Haar wavelet responses along the x and y directions in $4s \times 4s$ sub-regions separately as shown in equation:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{4}$$

Where, d_x and d_y refer to Haar wavelet responses in the x and y direction, to finally get a 64-dimensional descriptor v.

The advantage of SURF is that it detects the key points as a part of the algorithm itself and therefore, an initial ROI around the target object need not be given. It is much faster than the very similar SIFT feature detector, thanks to the speed up from integral images in each step, and has comparable performance. A drawback of SURF is the computation time it takes to create and match interest points and feature vectors are high, hence it will be difficult to get real time results.

D. Optical Flow

Optical Flow is the pattern of apparent motion of objects, edges and surfaces in a visual scene, caused by the relative motion between an observer and the scene. Optical Flow tracking techniques allow for distinction between multiple objects and the background in a scene. This is computed based on the relative motion between an observer and the scene. Optical Flow can be implemented in many ways, and a typical solution and workflow for Optical Flow object tracking is shown in Figure 5.

The underlying principle behind all these solutions is the Optical Flow equation. Under the assumption that pixel intensities for an object do not change, if $I(x, y, t)$ is the pixel intensity of a point (x, y) at time t and $I(x+dx, y+dy, t+dt)$ is the pixel intensity of the same object point at time $t + dt$, then Optical Flow equation using Taylor Series expansion can be given as:

$$f_x u + f_y v + f_t = 0 \tag{5}$$

$$f_x = \frac{\partial I}{\partial x} \tag{6}$$

$$f_y = \frac{\partial I}{\partial y} \tag{7}$$

$$f_t = \frac{\partial I}{\partial t} \tag{8}$$

Where, f_x is the image gradient with respect to x, f_y is the image gradient with respect to y, f_t is the image gradient with time, u is the velocity component along x direction and v is the velocity component along y direction.

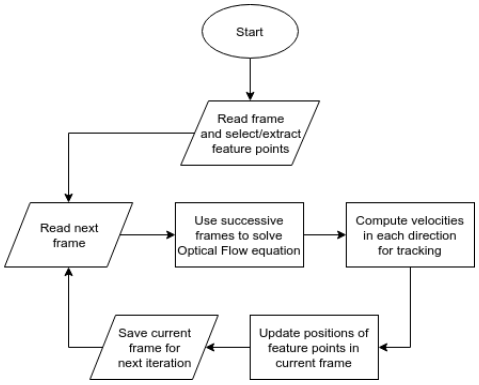


Fig. 5: Optical Flow Pipeline

The Optical Flow pipeline is explained in Figure 5. Here, successive frames are used to determine the velocity vectors of the points being tracked. Based on these vectors, the next location of target points is predicted and in this way, tracking is achieved. This prediction is done using equation 5, under the assumption that the pixel intensities of an object remain fairly similar over consecutive frames and that the neighboring pixels have similar motion. A solution to this equation is explained below. This solution determines the velocity vectors of target points, therefore, predicting their positions and tracking them over consecutive frames.

One of the solutions for Optical Flow is Lucas-Kanade method [6] [7]. This method is a widely used differential method for Optical Flow estimation. An important thing about this solution is, it assumes that the pixel intensities of the object do not change between consecutive frames. Also, the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic Optical Flow equations for all the pixels in that neighbourhood, by the least squares criterion. This solution is given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{y_i} f_{x_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \quad (9)$$

On supplying the interest point in a video sequence, this solution can give the Optical Flow vectors that helps in tracking the interest points. Different scales of motion (small as well as large) are handled by solving the Optical Flow equations over image pyramids.

By combining information from several nearby pixels, the Lucas-Kanade method can often resolve the inherent ambiguity of the Optical Flow equation. Traditional Lucas-Kanade is typically run on small, corner-like features to compute Optical Flow, but can be extended to other features.

E. Kalman Filter

The Kalman Filter, is an optimal recursive prediction filter that utilises the concept of state spaces, and is essentially recursive in nature because it does not need the entire set of data for processing. The time required for computation is less, and it's working is predominantly categorised into two main steps, namely the Prediction and Correction steps.

First, the state is predicted, and then it is corrected using an observation model. This is done such that the error of covariance is minimized. These steps are performed recursively, with the recent information obtained in the previous iteration being used as the initial value in the following iteration. Mathematically, as explained in [12], the Kalman Filter equations fall into two main sub-groups – time update equations and measurement update equations – that are responsible for the prediction and correction steps as mentioned above.

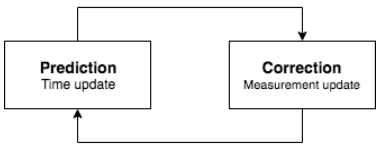


Fig. 6: Steps in Kalman Filter

The time update equations for prediction are as below:

$$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_k \quad (10)$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \quad (11)$$

Here, A is the state matrix and Q is the covariance of the process noise, inside which each point x_k is said to be moved into. In simpler words, the untracked elements are considered to be process noise, with a covariance of Q. Using the signal measured in the previous iteration $\hat{x}_{k-1}$, and control signal u_k, the rough prior estimate $\hat{x}_k^-$ is obtained. Similarly, the prior error covariance P_k^- is also calculated. These values are fed to the measurement update equations, where the posterior state estimates are finally computed for the current k^{th} iteration.

The measurement update equations for the updation or correction steps are:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \quad (12)$$

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}_k(y_k - \mathbf{H}\hat{x}_k^-) \quad (13)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \quad (14)$$

Here, first the Kalman gain K_k must be calculated, followed by measuring of the process to obtain the value of y_k. By using these values, a new posterior state estimate $\hat{x}_k$ is estimated using the computed values of Kalman gain K_k, the measurement value y_k, and the rough prior estimate $\hat{x}_k^-$. Finally, the error covariance P_k is calculated, and this process is repeated recursively to predict new estimates. The final projected state $\hat{x}_k$, and the measured error covariance P_k are used to roughly estimate the object's position in the next iteration, essentially becoming inputs for the $k{+}1^{th}$ priori estimates.This combined knowledge obtained from the prediction and correction steps

can give the best estimate of the position of the object being tracked.

The process followed for object tracking using the Kalman Filter, starts off by modeling the background using the running average of the sequence of frames. This has to be done at every time t, and the background can be obtained by averaging all the frames until the time t. While averaging, the weight chosen to perform the averaging is dependent on the frame rate as well as the degree of movement observed in the video. Then, the current frame is simply differenced with the modeled background to extract the foreground. This helps in identifying the object of interest, from which the position co-ordinates and velocity of object are extracted. This is fed to the Kalman Filter which performs the correction and prediction steps using the time and measurement updation equations. This process is then recursively repeated for the entire video sequence.

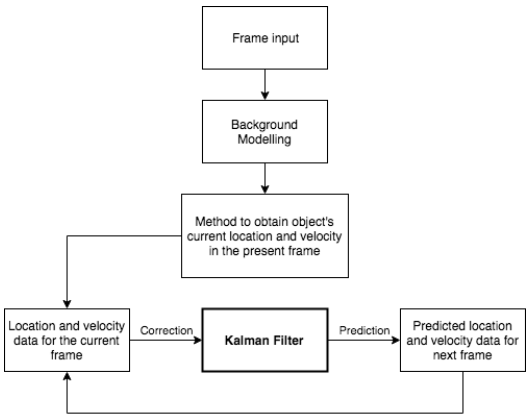


Fig. 7: Kalman Filter Pipeline

To sum it up, the Kalman Filter is an example of an optimal estimator. It identifies the parameters of interest from indirect, uncertain and inaccurate observations. It is recursive, hence new measurements are processed as and when they arrive. The Kalman Filter reduces the mean square error of the estimated parameters, if the noises are Gaussian. The Kalman Filter is the best linear estimator, if the mean and standard deviation of noise is given. It can be extended to work as a non-linear estimator as well. To sum up, the three important features of Kalman Filter for tracking are:

- Object's future location prediction.
- Noise reduction introduced by inaccurate detections.
- Association of Multiple objects to their tracks.

III. Performance Evaluation

To evaluate performance, the first dataset obtained is from YouTube, and it contains frames having a smooth green ball with a white background. The green ball is the object to be tracked by the algorithms. The dataset is a video with frames of dimensions 576×320 pixels, and has a total of 403 frames. A second dataset is used for evaluation as well. This dataset is a video wherein a Raspberry Pi box (target object) is recorded by a moving phone camera from various distances and angles.

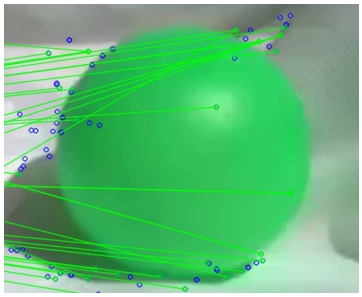


Fig. 8: Feature-poor object's Feature points

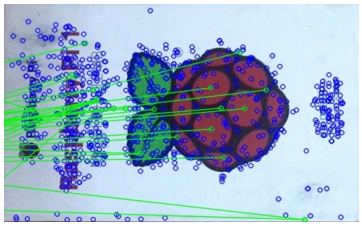


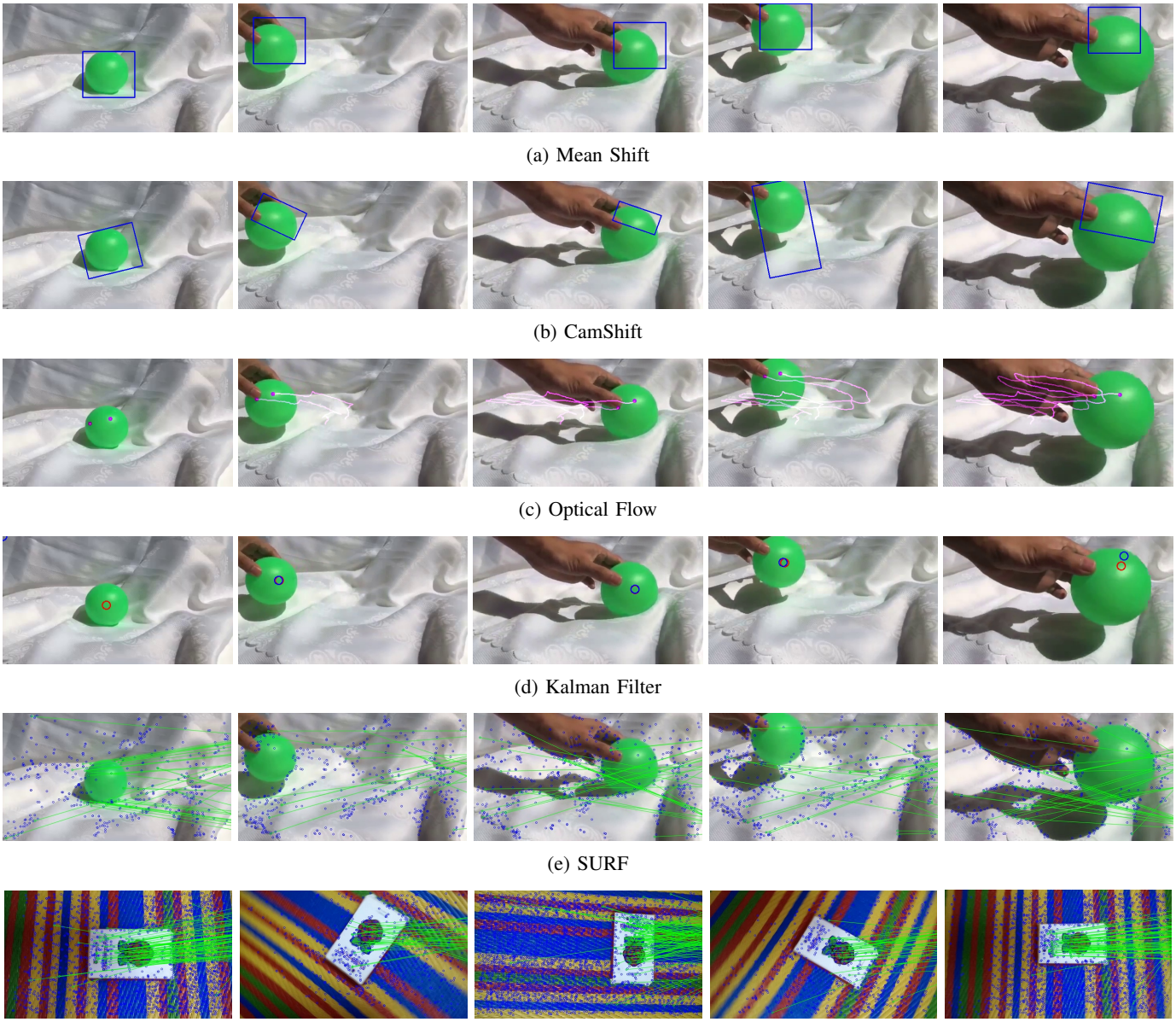
Fig. 9: Feature-rich object's Feature points

This dataset has around 1200 frames of size 800×450 pixels. The performance of the methods on the first dataset is explained as follows. For Mean Shift and CamShift, the ROI containing the ball is given as input by initializing a tracking window around the ball. For Optical Flow and Kalman Filter, performance is evaluated by initializing a point on the ball to be tracked. For SURF, the algorithm itself determines the set of feature points and tracks them in successive frames.

On implementing the algorithms on the dataset, the following observations were made. Mean Shift was able to track the ball through all the frames, but it did not scale with the ball as it moved nearer to, or farther away from the screen. This showed that the tracking window is not dynamic as illustrated in Figure 10(a). CamShift also tracked the object through all frames, and adapted the tracking window with the object as well, showing the dynamic nature of the window, as illustrated in Figure 10(b). The tracking performance for Mean Shift and CamShift are similar, with the only difference being how the tracking window adapts with the target object. Both of these algorithms have been observed to achieve good efficiency and can be used for real-time tracking.

Optical Flow showed very good performance on the dataset. The algorithm tracked user defined feature points on the object with high precision and efficiency, as shown in Figure 10(c). It was found to be faster than Mean Shift and CamShift on the datasets. This can be explained by the fact that Mean Shift and CamShift are tracking an entire window, whereas Optical Flow is tracking a few points, resulting in the performance leap.

The performance of SURF on the other hand was found to be poor on the green ball dataset. The algorithm could not associate much of the feature points it detected with the target object, and therefore failed to track the ball, with the exception of a few frames, shown in Figure 10(e). SURF



(a) Mean Shift

(b) CamShift

(c) Optical Flow

(d) Kalman Filter

(e) SURF

(f) SURF with feature-rich object

Fig. 10: Comparison between the 5 methods: (a) Mean Shift tracks the initialized object without adapting the tracking window, (b) CamShift tracks the object and adapts the tracking window with respect to the object, (c) Optical Flow tracks a defined point and the path of tracking is shown, (d) Kalman Filter tracks a defined point on the object, (e) SURF attempts to track ball using features extracted from Figure 8, and (f) SURF tracks object with features extracted in Figure 9.

uses the presence of corner points on the object in order to generate feature points on it, but in the case of the ball, it has a smooth surface (no corners on the surface). Therefore SURF failed to associate the ball with sufficient number of feature points as shown in Figure 8, which eventually led to its poor performance. The efficiency of this algorithm was also observed to be poor, as it involves calculation of feature points and their descriptors on every single frame, giving a low frame rate. Although, for objects which possess patterns and prominent texture, the features that can be observed are a lot more. For example, the box as shown in Figure 9, where one can observe that large number of features have been learnt. The tracking results for this object can be observed in Figure 10(f), and it can be seen that the results are far better for such feature-rich objects, compared to the smooth green ball in Figure 10(e).

As highlighted before, the Kalman Filter calculates both, the actual and predicted measurements. These measurements have been clearly illustrated in Figure 10(d), with two differently colored circles. Here the red circle represents the actual measurement and the blue circle represents the prediction computed by the Kalman Filter. The tracking is mostly accurate,

TABLE I: Evaluation of stated algorithms against various datasets

Dataset	Mean Shift	CamShift	Optical Flow	SURF	Kalman Filter
Green Ball	248.023	236.811	273.932	9.182	174.585
Raspberry Pi Box	147.422	146.521	196.131	2.206	133.624

and there is a general trend of the actual marker following the predicted marker in the subsequent frames. This makes the Kalman Filter method extremely robust, and is also great for tracking multiple objects in a single scene. It is observed that Kalman Filter tracks objects efficiently, with good frame rates.

The Mean Shift algorithm is an efficient method for tracking objects [3] defined by histograms. CamShift is good enough to track a single object among multiple moving objects, as long as it is distinct from the rest of the scene. Although, methods do exist to perform multiple object tracking using CamShift as in [11]. CamShift consumes less CPU time, hence it is one of the real time tracking algorithms. CamShift is an improved version of Mean Shift, as it overcomes the limitations of Mean Shift, such as changes in orientation of object and it's distance from the camera. But, improvements to CamShift [8] [9] has provided better results. However, when the background color is similar to the foreground object, performance of CamShift deteriorates. Also, it fails if lighting conditions changes or color of the foreground objects changes rapidly.

Table I shows the results of a frame rate analysis performed on these tracking methods. These methods have been tested on two datasets – the smooth green ball and the Raspberry Pi box videos – with 50 iterations each, and the frame rates of the corresponding methods have been averaged. It can be observed that the Optical Flow method has the highest Frames Per Second (FPS) values, and is therefore is the quickest method. The frame rates of Mean Shift, CamShift and Kalman Filter are not far behind, making these methods ideal for real-time applications as well. Whereas for SURF, the FPS has been observed to be far too low, due to the excessive amount of computation being performed for feature matching.

SURF is very good, even under suboptimal lighting or if the objects are colored similarly in the scene. The only drawback of SURF is the long computation time to create and match interest points and feature vectors, making it unsuitable for obtaining real time results. Optical Flow is good for tracking moving objects under normal lighting conditions. It is possible to extend Lucas Kanade method to other parametric models too. It is not highly affected by image noise, compared to other point-wise methods. Major drawbacks of this method are: It assumes a constant flow for all pixels, which is not the case in large window and in most real life scenarios. Also, errors are generated mostly on the boundaries of moving objects. Kalman Filter produces good results compared to other methods due to its structure and optimality. It is used as a real time tracking method, convenient for online real time processing. But major drawback is its assumption that both the system and the observation models are linear equations, which is not true in many real life situations.

IV. Conclusion

Comprehensive analyses of various tracking algorithms, which are used to track moving objects in video sequences have been discussed and implemented. For each algorithm, the techniques being applied to track; it's strengths; and few instances where they fail have been summarized. It has been observed that Mean Shift and CamShift methods work fine to a certain extent, but are limited for cases where there are illumination changes and need for tracking of multiple objects. As delineated in previous sections, SURF works well to track general objects in a scene, the downside being that the features of the object to be tracked must be computed and available before hand. Also, it does not work well for symmetric and uniformly shaped objects. Finally, of all the methods, Kalman Filter and Optical Flow have been found to be impressive, as they give real time results and are capable of tracking multiple moving objects in an efficient and accurate manner.

References

[1] Azra Nasreen, Kaushik Roy, Kunal Roy and Shobha G, "Key frame Extraction and foreground modelling using K-means clustering", in 7th International Conference on Computational Intelligence, Communication systems and Networks, indexed in IEEE Digital library, June 3-4 2015, Riga Technical University, Latvia, pp. 141-145.

[2] Dorin Comaniciu and Peter Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, May 2002, Vol. 24, No. 5.

[3] Zhi-qiang Wen, Zi-xing Cai, "Mean Shift Algorithm and its Application in Tracking of Objects", Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, 2006.

[4] J. G. Allen, Richard Y D and Jesse Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces", Proceedings of the Pan-Sydney area workshop on Visual information processing, VIP 2005, pp. 3-7.

[5] Yu Bai, Li Zhuo, Bo Cheng and Yuan Fan Peng, "SURF feature extraction in encrypted domain", Proceedings of IEEE International Conference on Multimedia and Expo (ICME), 2016, Beijing, pp. 1-6.

[6] Dhara Patel and Saurabh Upadhyay, "Optical Flow Measurement using Lucas Kanade Method", International Journal of Computer Applications, 2013, Vol. 61, (10), pp. 10-28.

[7] Simon Baker and Iain Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework", International Journal of Computer Vision, 2004, Vol. 56, Issue 3, pp. 221-255.

[8] Alper Yilmaz, Omar Javed, Mubarak shah, "Object Tracking: A Survey", ACM Computing Surveys, 13 December 2006, Vol. 38, No. 4, Article, New York.

[9] David Exner, Erich Bruns, Daniel Kurz, Anslem G, "Fast and robust CAMShift tracking", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 13-18 June 2010, San Francisco, pp. 9-16.

[10] Bay H., Tuytelaars T., Van Gool L., "SURF: Speeded Up Robust Features". In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision ECCV 2006, ECCV 2006, Lecture Notes in Computer Science, Vol 3951. Springer, Berlin, Heidelberg.

[11] Aditi Jog and Shirish Halbe, "Multiple Objects Tracking Using Camshift Algorithm In OpenCV", International Journal Of VLSI And Signal Processing, 2012, Vol.1, (2), pp. 41-46.

[12] Sumit Kumar Pal and Sohan Ghorai, "Moving Object Tracking System In Video With Kalman Filter", International Journal of Engineering Research and Technology (IJERT), June 2013, Vol.2, Issue 6.