Department of Electronics & Communication Engineering
National Institute of Technology Rourkela
Rourkela - 769008

Robust Object Tracking Algorithms using C++ and MATLAB

B.Tech Project Thesis

Submitted By:

Rahul Yadav
110EC0184

Supervisor:

Prof. K. K Mahapatra
Dept. of Electronics & Communication Engineering
National Institute of Technology Rourkela

National Institute of Technology, Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**Robust Object Tracking Algorithms using C++ and MATLAB**" submitted by **Rahul Yadav** bearing Roll No – **110EC0184** in partial fulfilment of the requirements for the award of Bachelor of Technology degree in **Electronics and Communication Engineering** during the session 2013-14 at National Institute of Technology, Rourkela (Deemed University) is an authentic work done by him under my supervision and guidance.

To the best of my knowledge, the matter contained in this thesis has not been submitted to any other university/institute for the award of any Degree/Diploma.

Place: Rourkela **Prof K. K Mahapatra**

Date: Dept. of Electronics and Communication Engineering

National Institute of Technology Rourkela

Rourkela-769008

Acknowledgement

I am very grateful to my thesis guide Prof. Kamalakanta Mahapatra for providing me with this opportunity to work under him and guiding me throughout the whole time-span of the Project work. Without his guidance, motivation and invaluable inputs the project might have not been possible.

I would like to thank Mr. Vijay Kumar Sharma, Ph. D Scholar under Prof. Kamalakanta Mahapatra, for his indispensable mentoring and doubt clearing as and when I needed.

Finally I would like to thank my alma mater National Institute of Technology Rourkela for giving me the opportunity to study here, enrich my knowledge and hone my skills. It was a great learning experience in the mentally stimulating ambience of the institute.

Rahul Yadav
110EC0184

Abstract

Object tracking, all in all, is a testing issue. Troubles in tracking objects emerge because of unexpected motion of the object, scene appearance change, object appearance change, structures of objects that are not rigid. Besides this full and partial occlusions and motion of the camera also pose challenges. Commonly, we make some assumptions to oblige the tracking issue in the connection of a specific provision. Ordinarily it gets important to track all the moving objects in the real time video. Tracking using colour performs well when the colour of the target is unique compared to its background. Tracking using the contours as a feature is very effective even for non-rigid targets. Tracking using spatial histogram gives satisfactory results even though the target object undergoes size change or has similar coloured background. In this project robust algorithms based on colour, contour and spatiograms to track moving objects have been studied, proposed and implemented.

Keywords: Object tracking, contours, Spatial histogram, Spatiogram.

Contents

1. Introduction

In the domain of computer vision, object tracking plays a very important role. With the advent of powerful computers, the proliferation of high definition and economical video cameras, and the applications that require automated analysis of a video, a great increase in the interest in object tracking algorithms has come in picture. Video analysis consist of three primary steps: detection of objects that are moving called the target objects, tracking of target objects in consecutive frames, and analysis of tracks to study behaviour and motion. Therefore, tracking of objects is indispensable in the following domains [12]:

—recognition based on motion that involves identification of human depending on gait, automatic object detection, etc.

—video surveillance that involves analysing a scene for detection of unwanted or suspicious events;

—automated indexing, which is numbering and searching of videos in large databases;

—interaction of human-computer, which involves gesture recognition, eye gaze tracking for data input to computers, etc.

—monitoring the traffic which involves real-time gathering the statistics of traffic to control traffic.

—automated navigation of the vehicle, which involves path finding depending on video and capability to avoid obstacles.

Basically, tracking may be defined as the task of finding the path of an object in frames when it moves in the background. Conversely, an object tracker puts consistent marks to the objects tracked in consequent video frames. Besides, based on the tracking application, a tracker may incorporate additional object-specific information for instance area, shape or orientation of the object. Tracking of objects poses a challenge because of [12]:

—information-loss due to 3D projection on a 2D image,

—image noise,

—complex motion of the object,

—objects being articulated and non-rigid,

—image occlusions,

—complex shape of the target,

—brightness changes in the scenes, and

—requirement of real-time motion tracking.

Usually an attempt is made to simplify tracking by putting constraints or making some assumptions on the object motion and/or object appearance. Most of the tracking algorithms make an assumption that the object moves smoothly and does not undergo any sudden changes. Besides this constrain can be added to the motion of the object to have constant velocity/acceleration depending on some information made available beforehand. Prior information about the size and number of objects, or the shape and appearance, can also be used to further constrain our tracker. A number of approaches for tracking objects have been developed over the years. These primarily differ from one another based on their approach to the following questions: How to represent the object? How to select the image features? How to model the shape, motion, and appearance of the object? The answers are found to vary with the context/environment in which the object is being tracked and the final application for which the information of the tracks is required. Consequently, numerous tracking methods have been found that answer these questions for various situations [12].

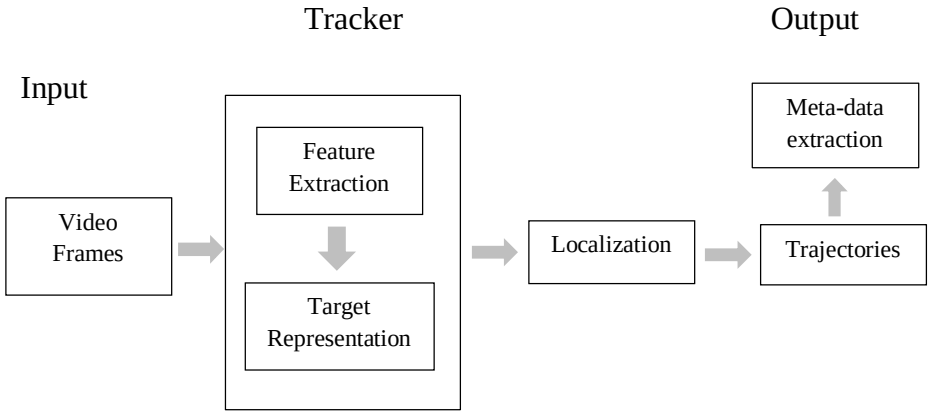

Fig.1.1 Components of a tracking system

2. Object Tracking- A Literature Survey

Finding the trajectory of an object in motion in subsequent frames from a video taken in real time is quite challenging. If the camera with which the video is taken moves too, then we cannot use background subtraction algorithms. Besides this, occlusion, real-time processing requirement, complexity of object motion, non-rigidity of object etc. add increase the challenge of object tracking. However in the past decade a number of high performing algorithms have come to surface.

The tracking of an object comprises of two primary steps namely representation and localization. The former depends on the modelling of the target object whereas the latter deals with method of searching the target in subsequent frames. Colour histogram [3-5], feature point [7-9] and object contour [10, 11] etc. are some of the models that are very popular for target representation. The target localization depends on the model followed by target representation step. Tracking based on object contour gives good result in case of non-rigid objects. It consists in constructing a B-Spline curve to parameterize the contour followed by particle filtering method to perform tracking [11]. One B-Spline curve is used to represent each particle from a set of N particles at each frame and these set of points have to be continuously updated and maintained depending on the local map of the edges. It is evident that for a good tracking performance we would need a large number of particles. Again if we want to track more than one object simultaneously then this algorithm fails due to its high computational requirement.

One of the other approaches that track based on contours make use of a level set array and two linked lists. The linked lists elements are switched on to perform the realization of adaption of contour. Though the algorithm mentioned above has comparatively low complexity in the computation we observe that the representation of the contour which is non-parametric tends to decrease the constraint in the contour. If the background has same colour as our target object then to include the back ground the contour expands that leads to failure in tracking.

If rich textures are present in the object then we can use feature points that gives very good results. One of the methods which are usually used to find a feature point that gives good results is the iterative Newton-Raphson minimisation algorithm [8, 9]. This algorithm finds the corresponding feature points in subsequent frames. When feature points are used in

tracking the implementation is faster and found to be more reliable. But this tracking fails if partial occlusion takes place or if the object motion includes many turns.

It has been found that for target representation, if the colours histogram is used then the tracking performs well even if there is partial occlusion or pose changes of the object. One of the very popular algorithms named 'Camshift' [5] consists in making a colour histogram of the target object to track it in subsequent frames. It employs an iterative procedure that makes use of the mean shift to find the boundary of the object in subsequent frames. In each frame, the boundary position is changed until it finally converges. This algorithm was primarily developed for tracking of human faces; however this can be used for other objects also. Another algorithm, known as the Kernel-based-tracking algorithm (KBT) [4] depicts the distribution of colour of the object using a Kernel weighted colour histogram. Here, the weights of the pixels at the boundary of the object are smaller whereas those pixels that are around the object centre are given higher weights. Here also, the target localisation is done by performing through a mean shift method (generally Camshift) iteratively. These algorithms using mean shift have been found to perform very well with the use of Kernel-weighted histogram. But a serious short-coming of these algorithms is that when the object to be tracked undergoes a size change, the performance deteriorates [12].

3. Tracking an object based on its colour

Tracking an object by its colour is based on the following assumptions that the colour of the object is different from that of its background and remains unique throughout the duration of the tracking [12].

Here we use the HSV (Hue, Saturation, and Value) colour space, instead of the more common RGB(Red, Green, Blue) colour space. In HSV, each “tint” of colour is represented by a number called the Hue. The “quantity” of colour is represented by a number called the Saturation and the illumination of the colour is represented by a number called the Value.

This has a certain advantage over the RGB scale in that we have a single number (hue) for the object in spite of multiple shades of colour (from dark to a bright) that the object may have.

To implement this algorithm the open source computer vision library (openCV) was used. This library is written in C/C++ and is optimized and intended for real time applications. The version used was OpenCV 2.4.6.0 configured with Microsoft visual studio 2010.

3.1 Algorithm

The complete algorithm description can be found in [12]. The outline of the steps followed is mentioned in a simple manner below:

1. The first frame is read from the webcam and the colour of object to be tracked is found out. The colour space should be HSV (Hue-Saturation-Value) since the hue value depends on only the tint and is same ranging from dark to bright.
2. Then the image is thresholded using the HSV value.
3. Then calculate moment10 and moment01- the two first order moments and the zeroth order moment (area).
4. Then divide moment10 by area to get the X coordinate, and similarly, divide moment01 by area to get the Y coordinate.
5. The centre of the object thus found using the moments is saved and then the next frame is captured and the steps are repeated for it too.
6. In the current frame the centre of the object is then joined with that of the previous one thus giving us a scribble that denotes the path of the motion.

3.2 Results

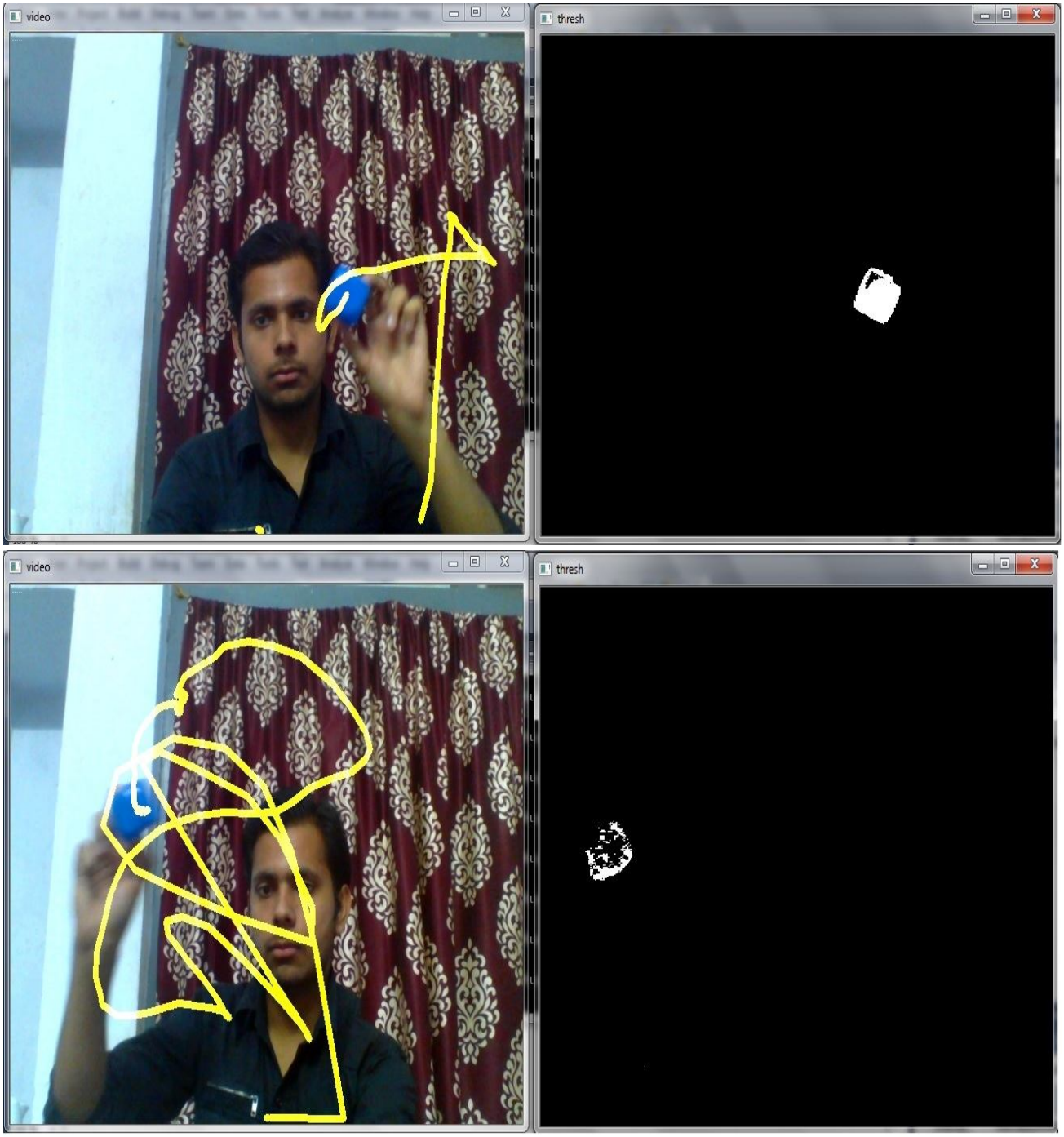

Figure 3.1: The left side shows the tracking result of the blue coloured object using yellow coloured scribbles and the right side shows the frame after thresholding operation using the hue (105 – 130),saturation – (130 -255) and value – (130-255)

3.3 Conclusion

This method was implemented in OpenCV 2.4.6.1 and the results obtained were found to be successful in real time.

This method can be used to track the object even if the background is not constant, the only constraint being its colour should be unique.

Occlusion of the object does not have any effect on performance. Even if the object gets completely occluded in some of the frames, it keeps up the tracking once the occlusion is removed.

The motion of the camera while taking the video has little effect on the tracking result.

However it should be kept in mind that the colour of the object to be tracked must be unique compared to its background.

4. Tracking moving objects using contours

A contour [2, 3] is a set of points that directly or indirectly depicts in an image, a curve. This depiction depends on the situation in particular. There are numerous methods in which a contour/curve can be represented. In OpenCV the representation of the contours is carried out by sequences in which the entry has encoded information of the location of the next point on the curve.

To find the pixel mean in a set of images/frames, add all the pixel values across the frames and divide it by the total number of frames. However an alternative is often used called the *running average*. The running average may be calculated as [2, 3]:

$$\text{accml}(x, y) = (1-\alpha)*\text{accml}(x, y) + \alpha*\text{img}(x, y) \qquad \text{iff } \text{msk}(x, y) \neq 0$$

If the value that is given to α is constant then the result of summing and that of running average will differ. The parameter α determines the extent to which the previous frame influences the accumulator. Equivalently, it gives the length of time it takes for the effect of previous frames to diminish.

4.1 Algorithm

The details of object tracking using contours can be found in [2, 3]. The outline of the algorithm used is mentioned below:

1. The first frame is read from standard PETS2001 dataset video.
2. Using the next frames the running average is calculated and accumulated.
3. For each frame the running average accumulated is subtracted.
4. Then the difference is converted to grey scale using the standard formulae.
5. The grey image then thresholded to make the contours more pronounced.
6. The noise is reduced using morphological operations- eroding and dilating.
7. Contours are then computed on the difference image.
8. For each contour the boundary is determined and a rectangle is drawn surrounding it to show the object being tracked.
9. Steps 2-7 are repeated until the completion of the video.

4.2 Results

After implementing the algorithm in OpenCV, the results on various datasets obtained are shown below:

Figure 4.1: Tracking result using the given algorithm on PETS2001 (1) dataset.

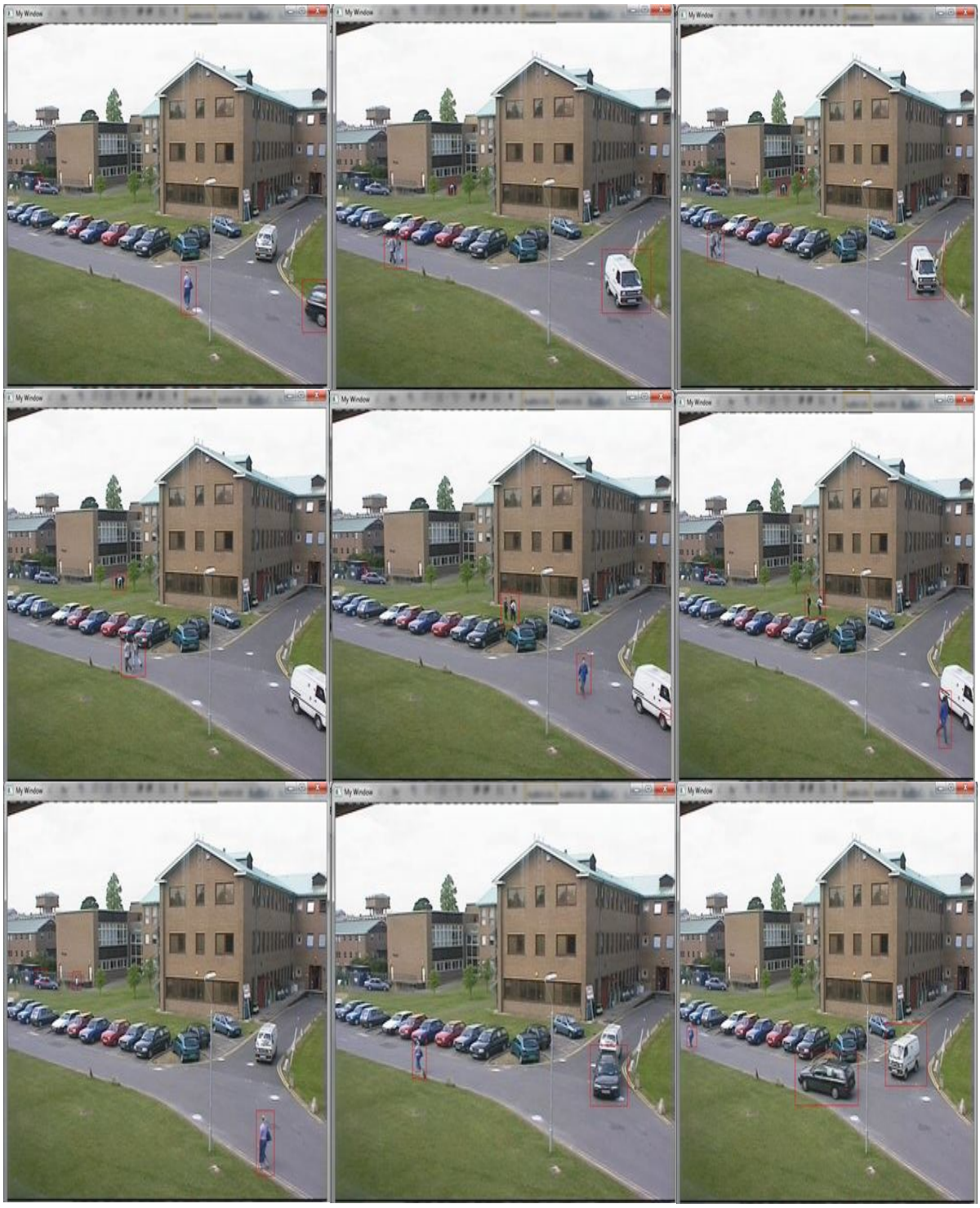

Figure 4.2: Tracking result using the given algorithm on PETS2001 (2) dataset.

Figure 4.3: Tracking result using the given algorithm on PETS2001 (3) dataset.

4.3 Conclusion

This method was implemented in OpenCV 2.4.6.1 and the results obtained were found to be successful in real time. This method is quite robust in the sense that it tracks all the moving objects in the video despite their arbitrary shapes, sizes and locations. In applications such as surveillance for security it is very efficient as it detects any motion in the captured view. This also performs very well when the object in motion and its background share similar colour. The varying brightness, up to a certain level, has no effect on the performance of the algorithm. Besides this it also performs satisfactorily when the object undergoes partial occlusion.

However the background should be fixed for this algorithm to work otherwise the contours may also include the background and the moving objects can’t be tracked effectively.

5. Spatial Colour Histogram based object tracking

In this method the concept of spatiogram [1] is used to represent the object by a new class of spatial colour histogram model. Every bin of spatial histogram has the information of total pixel number that fall in the particular bin and also the pixels positions with respect to the centre of the object. Using the centre voting scheme similar to that of generalized Hough transform, the localization of the target object is performed. After localizing the object, it is segmented out using the back projection algorithm from the background [1]. Once the object is segmented out, we estimate the object size in the current frame and change the search range.

5.1 Algorithm

Let $\vec{c}$ be the centre of the object to be tracked in the current frame. We need to find its centre in the subsequent frames. We refer the object to be tracked as the target object.

We segment the object in the initial frame using background subtraction algorithms, the details of which can be found in [12], or else by a manual selection.

5.1.1 Target Representation

Let $\{\overrightarrow{X_i} = (x_i, y_i)\}_{i=1,\dots,N}$ be the pixel locations of the object we want to track, the target object.

$\vec{c} = (c_x, c_y)$ is the object centre and (l_x, l_y) are half the dimensions of rectangle bounding the target object. For target representation we form a spatial colour histogram given as [1]

$$h = \{ \vec{\mu}_{(b,k)} , n_{(b,k)} \}_{b=1,\dots,B,\ k=1,2\dots}$$

Where $\vec{\mu}_{(b,k)}$ = Mean vector that represents the location of k^{th} pixel cluster of in the b^{th} bin with respect to the object centre.

$n_{(b,k)}$ = the probability value determined using total pixel number in the b^{th} bin and k^{th} cluster.

Mathematically [1],

$$n_{(b,k)} = C\sum_{i=1}^{N} K(\overrightarrow{X_i})\ \boldsymbol{\delta}_{ibk}$$

Where $C = \frac{1}{\sum_{i=1}^{N} K(\overrightarrow{X_i})}$ to ensure that $\sum_{(b,k)} n_{b,k} = 1$

$\boldsymbol{\delta}_{ibk}$ = 1 if the pixel lies in b^{th} bin, k^{th} cluster otherwise it is zero.

The Kernel function is defined as [1]

$$K(\overrightarrow{X_\iota}) = \begin{cases} 1 - d^2(\overrightarrow{X_\iota}) & if\ d(\overrightarrow{X_\iota}) \leq 1, \\ 0 & Otherwise \end{cases}$$

Here $d(\overrightarrow{X_\iota})$ is Euclidean distance between pixel $\overrightarrow{X_\iota}$ and the object centre. We calculate this by normalizing $(x_i - c_x)$ by l_x and $(y_i - c_y)$ by l_y.

$$d(\overrightarrow{X_\iota}) = \sqrt{\left(\frac{x_i - c_x}{l_x}\right)^2 + \left(\frac{y_i - c_y}{l_y}\right)^2}$$

This is same as the ellipse equation centred at $\vec{c}$. Those pixels located outside it are given zero weights and those close to centre are given higher weights [1].

Estimation of $\vec{\mu}_{(b,k)}$ [1]:

Input: Pixel Location

Output: Cluster of pixels

For each $\overrightarrow{X_\iota}$:

1. Determine the bin index b of pixel $\overrightarrow{X_\iota}$.
2. Form a new cluster $\vec{\mu}_{(b,1)} = \overrightarrow{X_\iota}$ if the b^{th} bin is empty.
3. Else, for each k
    a. Find the distance of $\vec{\mu}_{(b,k)}$ from $\overrightarrow{X_\iota}$.
    b. If the distance is less than ε, then include $\overrightarrow{X_\iota}$ into $\vec{\mu}_{(b,k)}$.
    c. Update $\vec{\mu}_{(b,k)}$.
4. Form a new cluster $\vec{\mu}_{(b,k)} = \overrightarrow{X_\iota}$ if $\overrightarrow{X_\iota}$ cannot be assigned to any of the pre-existing clusters.

5.1.2 Target Localization

In this step we do the centre voting followed by back projection. The centre voting involves the following steps [1]:

1. Only those pixels can cast a vote whose colour is present in the histogram target model.
2. The pixel belonging to b^{th} bin puts its vote at $\overrightarrow{X_i} - \vec{\mu}_{(b,k)}$.
3. If the reliability of a pixel is high then it votes with larger weight compared to the pixel whose reliability is low.

The reliability of a pixel depends on how unique is its colour when compared to the background. The voting weight to cast a vote at $\vec{\mu}_{(b,k)}$is defined as

$$w_{(b,k)} = \max\{\frac{n_{(b,k)} - m_b}{n_{(b,k)} + m_b}, 0\}$$

Here m_b is the probability value obtained from the background spatial histogram. Let $\overrightarrow{X_i}$'s are the background pixels located within ellipse with axis $(l_x + \Delta)$ and $(l_y + \Delta)$ and

$$\Delta = \eta \times \min(l_x, l_y);$$

where the parameter η varies with the object speed.

The value m_b is the background histogram value at b^{th} bin and is defined as

$$m_{(b)} = C_{BG} \sum_{i=1}^{N} K_{BG}(\overrightarrow{X_i})\ \delta_{ib}$$

$$\text{Where } C_{BG} = \frac{1}{\sum_{i=1}^{N} K_{BG}(\overrightarrow{X_i})}$$

$\boldsymbol{\delta}_{ibk}$ = 1 if the pixel lies in the bth bin and kth cluster otherwise it is zero.

The kernel function is similar to the previous one and defined as:

$$K_{BG}(\overrightarrow{X_i}) = \begin{cases} 1 - \lambda\left(\sigma - d(\overrightarrow{X_i})\right)^2 & if\ d_\Delta(\overrightarrow{X_i}) \le 1\ and\ d(\overrightarrow{X_i}) \ge 1, \\ 0 & Otherwise \end{cases}$$

Where

$$\sigma = \frac{1}{2}(1 + \frac{d(\overrightarrow{X_i})}{d_\Delta(\overrightarrow{X_i})})$$

$$\lambda = \frac{1}{(\sigma - 1)^2}$$

For calculating $d_\Delta(\overrightarrow{X_i})$, we use $(l_x + \Delta)$ and $(l_y + \Delta)$ as the normaliser.

Since in the new frame the object is located somewhere closer to the last one, only the pixels nearby cast their votes. Thus the search region is defined to have the dimension of $2(l_x + \Delta) \times 2(l_x + \Delta)$.

After we find the new object centre in the following frame, we search for pixels that voted for the centre correctly and set those pixel locations as 1 in a new foreground image. This is the back projection method and the algorithm [1] is as follows:

For all $\overrightarrow{X_i}$ that are in the search region

dist = $\left\|(\text{vote}(\overrightarrow{X_i}) - \vec{c})\right\|$

If dist<ζ then

Set $\overrightarrow{X_i}$ as foreground

Else

Set $\overrightarrow{X_i}$ as background

End if

End for

Once we obtain the foreground image we calculate the object sizes. Let l_x^* and l_y^* be the new dimensions of the object calculated from the foreground image. We calculate the new object size by [1]

$$l'_x = (1-\alpha)l_x * + \alpha * l_x^*$$
$$l'_y = (1-\alpha)l_y * + \alpha * l_y^*$$

Here α is a constant that determines the rate with which we update the object dimensions.

5.1.3 Model Update

When the object moves its background changes continuously along with its own shape and size. Thus we need to update both the object and background model.

a. Background model update

First calculate the histogram of the background $m_b{}^*$ at the current frame. Then update the background histogram [1] as

$$m_b = (1 - \beta) \times m_b + \beta \times m_b{}^*$$

Here the constant β determines the influence of the previously obtained histogram on the updated histogram.

5.1.4 Object histogram update

This step involves three processes namely merging followed by appending and finally pruning. Let $h^* = \{\vec{\mu}_{(b,l)}{}^*, n_{(b,l)}{}^*\}$ be new spatial histogram determined at current frame with new kernel dimensions. Following steps are taken to update [1]:

1. If the distance between $\vec{\mu}_{(b,k)}$ and $\vec{\mu}_{(b,l)}{}^*$ is less than the clustering threshold then take their average and merge them. Update the probability as

$$(1-\alpha) * n_{b,k} + \gamma * n_{(b,l)}{}^*$$

   If $\vec{\mu}_{(b,k)}$ could not find a match in $\vec{\mu}_{(b,l)}{}^*$, set the probability value as

$$(1-\gamma) * n_{b,k}$$

2. If $\vec{\mu}_{(b,l)}{}^*$ cannot find any match in $\vec{\mu}_{(b,k)}$, then append it in *h*.
3. If the probability $n_{b,k}$ of $\vec{\mu}_{(b,k)}$ is less than the pruning threshold xi then take it out of the model.

5.2 Algorithm Summary

Input: An object whose spatial centre c is given in the first frame [1].

1. Find the object spatial histogram h.
2. Find the background histogram m_b.
3. In the next frame, ask to each pixel in the search region to cast votes for the centre with appropriate weight.
4. Find the position in the image that received the maximum votes. Consider it the new centre of the object.
5. Implement the back projection.
6. Change the dimension of the target object and update it.
7. Find the new histogram of the background $m_b{}^*$ at updated location c and subsequently update the original histogram of the background m_b.
8. Determine the new histogram of the object h^* at updated object location and finally update h.
9. Return to step 3.

5.3 Parameters used and their selection

A detailed study of the effect of these parameters can be found in [1].

Spatial Clustering threshold (ε):

If the value of spatial clustering threshold is large then more pixels would be grouped in the same cluster [1]. This will lead to a coarse model. Generally we take small value for ε.

Rigidity threshold (ζ):

This depends on the characteristics of the target object. If the target object is rigid then a small value should be selected otherwise a higher value for objects that are non-rigid. However if the value slightly deviates from the optimal one it does not affect the result.

Object size update (α):

The parameter α determines the rate at which the object size should be changed. It is found that smaller values usually give good results.

Background model update ratio (β):

This parameter determines the pace at which we update the background model. The result is relatively insensitive except when the object and background share a similar colour.

Object model update ratio (γ):

We assume that appearance of the object does not undergo a large change in subsequent frames. Hence we set a small value for this parameter. Setting a large value hampers the performance as more background pixels are allowed in the object model.

Pruning threshold (ξ):

This parameter is used to prune those clusters the probability of whom to be a part object model is quite less. As they cluster cast their votes with extremely less weights we don't need them in our model.

Speed of the object (η):

Usually a value of $\eta = 2$ suffices for most of the applications. If the speed of the object is very high between frames then a higher value may be used.

5.4 Results

Values of parameters: $\beta = 0.5$; $\alpha = 0.1$; $\gamma = 0.3$; $\zeta = 2$; $\xi = 0.00001$; $\varepsilon = 5$; $\eta = 3$

Figure 5.1: Tracking result of the coke can as it moved forward towards the camera and undergoes partial occlusion behind the leaves.

Figure 5.2: Tracking result of the coke under partial occlusion and illumination changes.

Values of Parameters used:

$\beta = 0.5; \alpha = 0.01; \gamma = 0.3; \zeta = 2; \xi = 0.00001; \varepsilon = 5; \eta = 2$

Figure 5.3: Tracking result of the toy under normal slow motion between frames

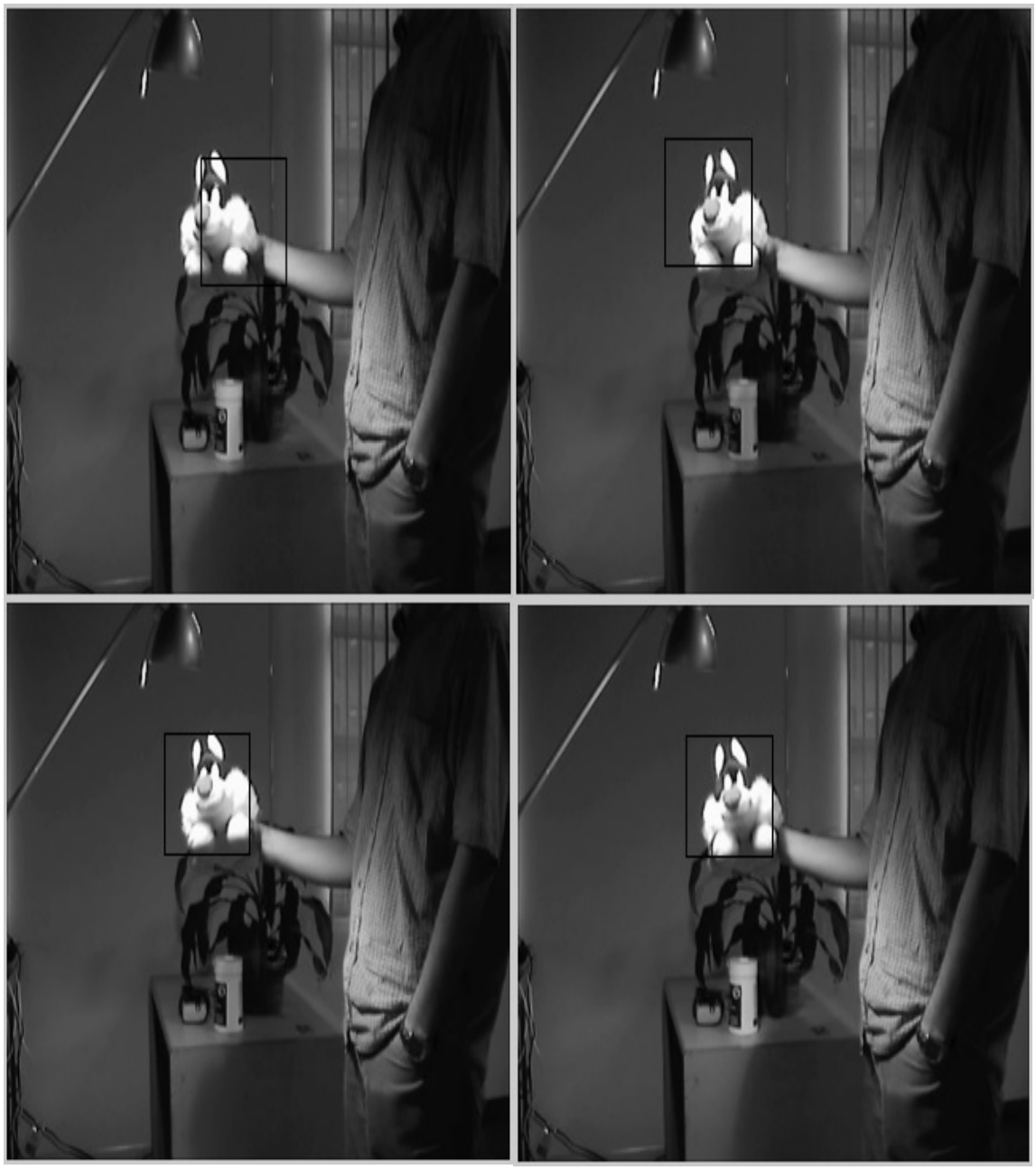

Figure 5.4: Tracking result of the toy when it is moved very fast between frames. The tracker loses track of the toy in the first frame but regains the location of toy in the third frame.

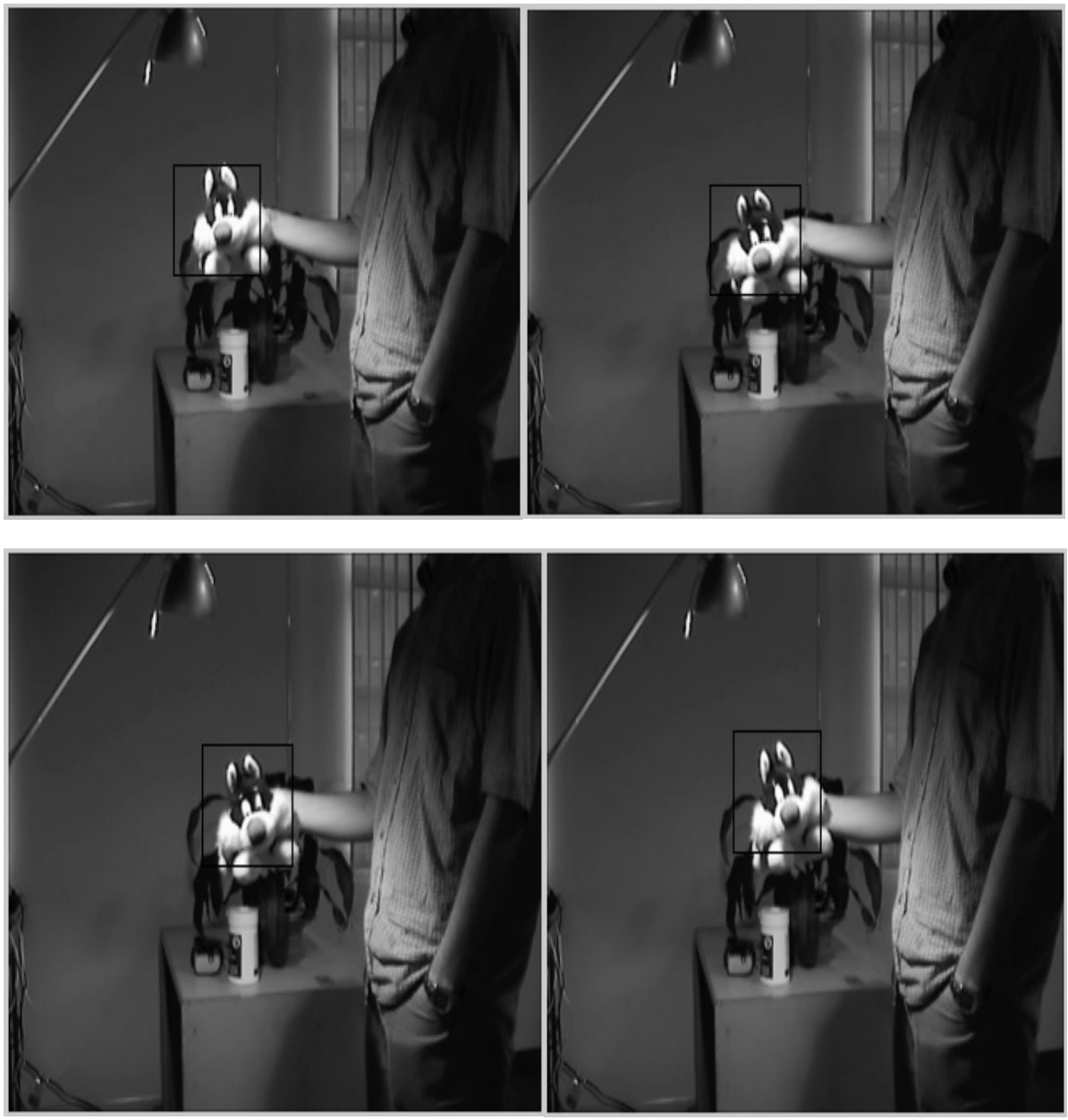

Figure 5.5: Tracking result of the toy under shape change i.e. when it undergoes affine transformation.

5.5 Conclusion

This algorithm was implemented in MATLAB. It tracks a moving object in the video despite its arbitrary shape, size and location and is not affected when the background shares similar colour. The varying brightness, up to a certain level, has no effect on the performance. Performs satisfactorily even if the object suffers partial occlusion and also when background is not fixed. Thus, camera motion does not affect performance.

This algorithm can be utilised in understanding the motion of humans in a video sequence. This algorithm finds the distance using the inpixel locations. Further improvement is possible if adopt a different representation that would be insensitive to shape and size changes and thus will increase its robustness

However the computational requirement is very high and makes real time implementation difficult.

Parallel processing may be used when real time application is required. Boundary information may be integrated into colour histogram to make it more robust. Another shortcoming is that appropriate values for the large number of parameters need to be selected beforehand. Error correction learning may be applied to adaptively change the parameters during run-time.

6. References

[1]Suryanto, Kim, D.-H., Kim, H.-K., Ko, S.-J.,Spatial Color Histogram based centre voting method for subsequent object tracking and segmentation (2011) *Image and Vision Computing*, 29 (12), pp. 850-860.

[2] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, Proceedings of International Joint Conference on Artificial intelligence, 2, 1981, pp. 674–679.

[3] S.T. Birchfield, S. Rangarajan, Spatiograms versus histograms for region-based tracking, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2, 2005, pp. 1158–1163.

[4] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based objects tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 564–575.

[5] G.R. Bradski, Real time face and object tracking as a component of a perceptual user interface, IEEE Workshop on Applications of Computer Vision, 1998, pp. 214–219.

[6] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision 60 (2004) 91–110.

[7] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (2008) 346–359.

[8] C. Tomasi, T. Kanade, Detection and tracking of point features, Technical Report, Carnegie Mellon University, 1991.

[9] J. Shi, C. Tomasi, Good features to track, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600.

[10] Y. Shi, W.C. Karl, Real-time tracking using level sets, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2, 2005, pp. 34–41.

[11] M. Isard, A. Blake, CONDENSATION—conditional density propagation for visual tracking, Int. J. Comput. Vision 29 (1998) 5–28.

[12] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (2006)13.