Fast Kernel Based Object Detection and Tracking for Stereo Vision System

Rupam Kr. Dewan
Department of Electrical Engineering
Jadavpur University
Kolkata-700032, India
rup21.4u@gmail.com

Ranjit K. Barai, *Member IEEE*
Department of Electrical Engineering
Jadavpur University
Kolkata-700032, India
ranjit.k.barai@gmail.com

***Abstract*— Target detection and tracking is a popular area of research in computer vision. In most of the previous works nonparametric approaches are adopted due to simplicity and robustness. However, the main drawback of nonparametric approaches is that extensive computational power is required for pixel per pixel analysis. In this paper we propose a fast density estimation technique combining the concept of *kernel density estimation* and *integral image*. Target RGB color information is taken as object feature. In each frame target object feature points are collected and fed to the density estimator to calculate the density then the target position is determined calculating the global maxima. For fast calculation image feature points are arranged in a sum table using the concept of *Integral Image*. Finally *kalman Filter* has been employed to produce smooth target trajectory. The system is suitable for real time application.**

Keywords-Target tracking; Kernel Density Estimation; Integral Image; Kalman filter.

I. INTRODUCTION

Object tracking is one of the most important fields in computer vision. Most of the fundamental technologies for real-world computer vision applications are derived from this field. Now days the application of object detection and tracking can be found in almost every field from intelligent surveillance system [1, 2], autonomous navigation[13], human robot interaction to automatic sports analysis[12], multimedia etc. Although the topic has been extensively explored using a monocular camera [3, 14], very few work has been done using a stereo vision camera which takes advantage of the depth information. Also most of the impressive algorithms are still computationally complex for real-time applications. Generally in this type of applications there is always a tradeoff between speed and accuracy. That's why in literature some algorithms are developed for complex object detection while others focuses on developing less computationally complex system for greater frame rate which is very essential for real-time tracking. The method proposed in this paper belongs to the second class. Some of the related works on this topic are discussed in this section.

In stereo vision framework Ling Cai at el in [1] has presented a multi object detection algorithm for autonomous surveillance system using kernel based Mean-Shift algorithm which was originally introduced by Fukunaga and Hostetler [11]. To reduce the computational complexity they generated sparse feature points like edges, corners of the scene instead of dense disparity map. Also they mounted the camera on an overhead position to overcome object occlusions finally their novel density estimation algorithm is quiet capable to coup with complex outdoor scenes. In another approaches Li Peihua in [3] has presented a fast object tracking algorithm. Target object has been represented using an adaptive clustering based color histogram and then exhaustive search is done for similarity measurement. To compensate the computational complexity the concept of Integral Histogram Image [5] has been adopted. The system seems so perform better than traditional Mean-Shift algorithm. A similar work can be found in [4] where image difference information is stored into a tree data structure (Temporal Histogram) for fast histogram calculation and update. Result shows that their proposed Temporal Histogram performs better than traditional Integral Histogram. In some cases this problem has been attacked in a more straight forward manner [2] where using the depth information a robust background model has been created. The detection is done using background subtraction method which is faster than other nonparametric approaches.

Our approach is totally based on nonparametric density estimation which is accomplished by *Kernel Density Estimation* (KDE) algorithm. The reason behind this is, KDE is more powerful tool than other nonparametric approaches (Histogram) and with the help of Integral Image the computational complexity is reduced for fast computation. To the best of our knowledge this approach has never been adopted before. We have implemented the proposed algorithm on a stereo vision camera for taking account the original world position of target object. The total tracking procedure takes place in a number of steps like feature extraction, localization, optimization and tracking.

The remainder of this paper is structured as follows. Section II explains the coordinate system and working principal of stereo vision system. Section III deals with our

target tracking approach. Section IV describes the experimental results and finally in section V we have drawn the conclusions.

II. STEREO COORDINATE SYSTEM

In case of single view camera when the three-dimensional structure projects into a two-dimensional plane the depth information is lost. The motivation of stereo vision system came from this problem of computer vision. It imitates the human vision system. In a stereo vision system a pair of fixed cameras is used where, one camera is slightly displaced from the other. The offset in the viewpoint positions of the cameras causes a relative transformation of the objects in the stereo image pair. Then the difference of positions (disparity) in two images of corresponding points can be established by using the epipolar line constraint and the correlation similarity measure. Different types of algorithms are used to solve the *stereo correspondence problem* like SSD, SAD, NCC [10] etc. where some of them produces dense disparity map others extracts sparse feature points. With this derived disparity map the 3D coordinates of the viewing objects can be calculated by simple triangulation. We used a commercial stereo vision camera [9] for calculating the object coordinates. Let us denote the 3D position of a pixel by (X,Y,Z) in terms of homogeneous coordinates and I_d as the disparity image. Then the coordinates can be calculated using Eqn. 1 [2]. Where f is the focal lengths of the cameras, b is the baseline distance between two camera centers and d is the disparity value of the pixel (u,v) in the image I_d.

$$Z = \frac{fb}{d}, X = \frac{uZ}{f}, Y = \frac{vZ}{f} \quad (1)$$

The calculated positions are referred to the stereo camera reference system which is in our case centered at the right camera. However if the application demands some other kind of camera position and orientation then the reference system can be changed. In that case it is possible to calculate a transformational matrix which will translate the captured points into world reference system. We tried to keep our most of the calculations in terms 2D homogeneous image coordinate to keep the complexity as low as possible, then the final calculated pixel coordinate has been transformed into the 3D coordinates using equation 1.

III. TARGET LOCALIZATION AND TRACKING

The first step for target tracking is to localize the target into the image frame. This can be done by first extracting the target feature points and then grouped those points into different clusters to represent the target object. There is no standard definition for an object. It could be anything which is a subject of further analysis according to the applications need. There are different types of object features which we can use to extract the target object pixels like color, shape, texture, edge etc. We choose the color of the target object as object feature because color representation is less complex and effective. An RGB filter has been employed to enhance the relative color level of the target object and then the target pixels are extracted by thresholding the image. Mean filter is used to smooth the image to reduce the noise. Radius of the mean filter is kept 1 pixel. The next step for target localization is to group the extracted feature points into some clusters. No matter how a target looks like we can represent it by some conventional spatial shape like a rectangle or an ellipse. In our case the clustering has been done using kernel density estimation (KDE) of the feature points. We have represented the target by a square uniform shaped kernel profile.

A. Target Localization

Kernel Density Estimation (KDE) is a classical non-parametric method for density estimation of a random variable without assuming any generative model. Let the *d* dimensional data points are $\{x_{1d}, x_{2d}, x_{3d}, ..., x_{nd}\}$ be some samples taken from an arbitrary distribution where x is an independent variable whose probability density has to be estimated with respect to the sample points, the *kernel density estimator* is defined as [8]:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\{\prod_{j=1}^{d}\frac{1}{h_j}K(\frac{x-x_{ij}}{h_j})\} \quad (2)$$

$\hat{f}(x)$ is the estimated density, $K(.)$ is the *kernel* and $h > 0$ is known as bandwidth or smoothing parameter.

A *kernel* denoted as $k(.)$ is a non-negative real-valued integrable function satisfying the following requirements:

$$k(u) = k(-u)$$

$$\int_{-\infty}^{\infty} k(u)du = 1$$

$$\int_{-\infty}^{\infty} u^2 k(u)du = k_2 \neq 0$$

There are different kernel profiles in literature. In our case the uniform box kernel fits our assumptions to the best. We have assumed the X and Y axis positions of the feature points to be independent. So here we have two different sets of variables for density estimation. To estimate the densities of this two dimensional data we employed bivariate estimator. For uniform kernel profile this density estimator looks like Eqn. 4.

$$\hat{f}(x,y) = \frac{1}{nh^2}\sum_{i=1}^{n}K_x(\frac{x-x_i}{h})K_y(\frac{y-y_i}{h}) \quad (4)$$

Where

$$K_x = \begin{cases} \frac{1}{2} \cdot 1 & where\ |x| \le 1 \\ 0 & otherwise \end{cases}$$

$$K_y = \begin{cases} \frac{1}{2} \cdot 1 & where\ |y| \le 1 \\ 0 & otherwise \end{cases}$$

The estimator calculates the local density around each feature points and then the point having the maximum density value is considered as the centroid of the target object. For reduction of complexity the estimator is only applied on the feature points.

B. Integral Image for Fast Calculations

To localize the target object via local density estimation requires an exhaustive search through all feature points. Unfortunately the numbers of the feature points may vary drastically over time due to several factor like illumination changes, speed of the target etc. Also exhaustive search through all the feature points in each frame is rather time consuming because the computational complexity of the estimator is $O(h.n^2)$. To deal with such uncertainty and complexity we have adopted the idea of *integral image*. The concept was first introduced in 1984 but was not properly introduced in the domain of computer vision till 2001 by Viola and Jones [6] object detection framework. Later it was adopted by Fait Porikli [5] to calculate histograms in Cartesian space.

The *integral image*, denoted by $I(x, y)$, at location (x, y) contains the sum of all the pixel values above and to the left of it (Figure 1), which can be written as: $I(x, y) = \sum_{x' \le x \bullet y' \le y} (x', y')$, where (x', y') is image co-ordinate and (x, y) is the co-ordinate of *integral image* structure. From Figure 2 according to the above definition the value of cell (x, y) can be computed as sum of values of cell $(x-1, y)$ and $(x, y-1)$ subtracted by the value of cell $(x-1, y-1)$. Mathematically the *integral image* can be computed in one-pass over the image using the following recurrence relation:

$$I(x, y) = f(x, y) + I(x-1, y) + I(x, y-1) - I(x-1, y-1) \quad (5)$$

$I(x, y)$ contains sum of all the pixel data transformed by the arithmetic operator $f(.)$.

When this value is computed for each cell then we can calculate the sum of the function $f(x, y)$ at a constant time for any rectangle in the image using the formula:

$$\sum_{rectangle} f(x, y) == I(x_2, y_2) + I(x_1, y_1) - I(x_1, y_2) - I(x_2, y_1) \quad (6)$$

Again it is a matter of only four arithmetic operations which will be calculated in a constant time. For *uniform kernel* we need to find out the number of points which are inside the bandwidth of the *kernel*. For each pixel point the *integral image* calculate where this pixel is extracted as a target pixel or not. This way for target pixels the cell value of the *integral image* is increased by '1' otherwise '0'. The value of each cell is computed by the Eqn. 5 where $f(x, y)$ adds value '1' or '0' for the current pixel. Once the *integral image* is constructed then calculating the sum of $f(x, y)$ within any size of window (Figure 3) is reduces to only a constant arithmetic operation using the Eqn. 6. Thus the total number of pixels inside the current window is calculated in constant time and feed to the *kernel density estimator*. The complexity of the process reduces to $O(h.c.n)$ where c is a constant representing the constant amount of calculation for per rectangular sum in integral image.

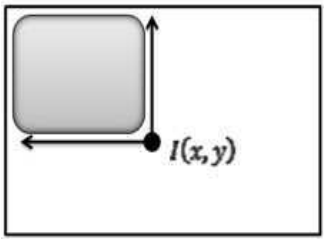

Figure 1. Pixel property in Integral Image.

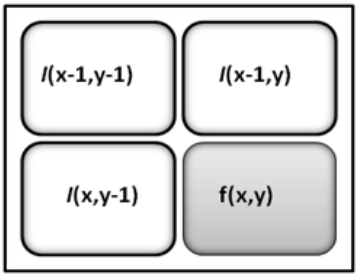

Figure 2. Integral Image formation.

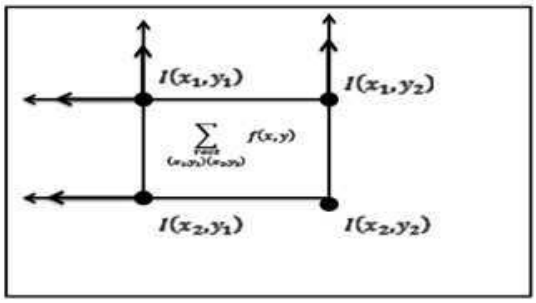

Figure 3. Rectanguler area sum in Integral Image.

C. *Tracking the Target*

There are many statistical tracking techniques available in literature. We have chosen the *Discrete Kalman filtering* [7] approach to generate a smooth target trajectory. As it is mainly a linear filter we formulated an approximate linear model for our system. We assumed that the target will always move in a constant velocity. This assumption is reasonable when the time between the two observations is small and also the amount of acceleration is small. In our case the frame rate is high due to the application of integral image concept and the acceleration generated from random movement is assumed to be within acceptable range. The additional process noise and measurement noise will be taken care of by the *kalman filter*. The constant velocity model is formulated as Eqn. 7.

$$x(t+1) = x(t) + v_x t$$
$$y(t+1) = y(t) + v_y t \qquad (7)$$

Here v_x and v_y are the velocities of the target object in the horizontal and vertical directions which are assumed to be independent of each other.

IV. EXPERIMENT AND RESULTS

For implementation we used Bumblebee2 stereo camera which is developed at the Laboratory for Computational Intelligence at the University of British Columbia and is being marketed by Point Grey Research, Inc. [9]. The camera has two Sony CCD sensors (color). Focal length of the camera is 6mm with 48% horizontal field of view. Camera frame rate is 15fps.Dimention of the camera is 157x36x47.4mm and weight 342 grams. A Software Development Kit (SDK) is also provided by Point Gray with this camera which offers many methods for stereo processing, depth calculation and validation. We only used the stereo processing facility for depth calculation. For our application we choose 240x320 camera resolution at 15fps. The Process noise covarience matrix and measurement noise covarience matrix used for the design of the filter are choosen as digonal matrices because velocities in the virticle axis and horizontal axis are assumed to be independent of each other. These matrices are

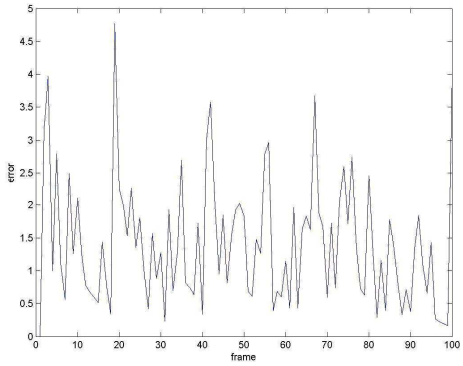

Figure 4. Prediction error in pixels for constant velocity.

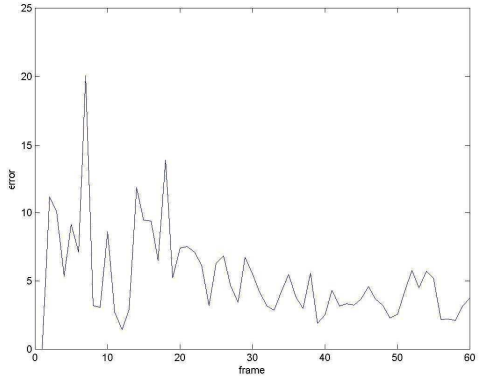

Figure 5. Prediction error in pixels for constant acceleration.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 7.69 & 0 \\ 0 & 0 & 0 & 7.69 \end{bmatrix} \text{ and } \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \text{ respectively.}$$

All parameters and constants are measured in pixel values.

A. *Experiment I (Tracking with Different Velocity)*

In our tracking algorithm we have used discrete kalman filter with a constant velocity model. In our first experiment we tested the performance of the tracker when the target object was moving in almost constant speed. Window size is same as the target size. All distance parameters are measured in pixels. The parameters for the corresponding environment are:
Threshold = 60, Kernel band width = 20x20.
Figure 5 shows the prediction error in this situation.

It is observed that for slow speed and suitable parameters the target is tracked very precisely and there are no false matching. From the Figure 4 it can be seen that the prediction error is not more than 3 pixel length. We have taken the results varying *window size* for this condition but there are no noticeable changes found in tracking performance. To test performance in the present of acceleration we follow the same experimental procedure only this time we move the target in a circular path. The parameters are kept same as before:
Threshold = 60, Kernel bandwidth = 20x20.
In this case a reasonable amount of increase in prediction error can be seen from Figure 5.

B. *Experiment II (Tracking with Different Threshold)*

To find out the performance of the tracker in different threshold we carried out another experiment. For each threshold value tracking performances are collected. For comparison we keep the window (kernel bandwidth) size constant (20x20 pixels). Figure 6 shows the change in prediction error with the variation of threshold value. For a large amount of threshold range the performance is acceptable (change in error is very low). But before or after that range the performance is not reliable (min 40 & max 180). This is because if the threshold is very high then it can extract very little amount of pixel and if the value is too low then it extract a large amount of pixels. The situations are shown in Figure 7.

C. *Experiment III (Tracking with Different Kernel Bandwidth)*

To find the tracking performance for different kernel bandwidth we vary the window size keeping the threshold constant. We observed from previous experiments (Experiment I) that increasing the size of window does not make any difference if the threshold is chosen very precisely according to the environment lighting conditions; so we carried out this experiment in the lower limit of the threshold value (40) to include more background pixels. The result is given in Figure 8.

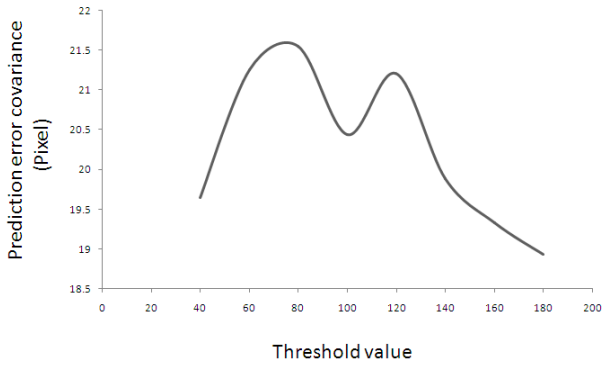

Figure 6. Prediction error in pixels for different thresholds.

Result shows that with the increase of window size in low threshold the performance is degraded. The reason is that in low threshold many background pixels are also extracted. As a result when the window size is increased then those unwanted pixels may shift the maximum density point away from the target. Figure 9 shows this situation.

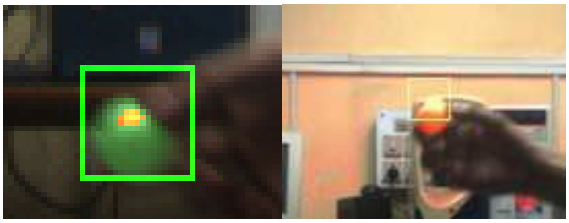

Figure 7. Target pixels for high threshold value (left) and low threshold value (right).

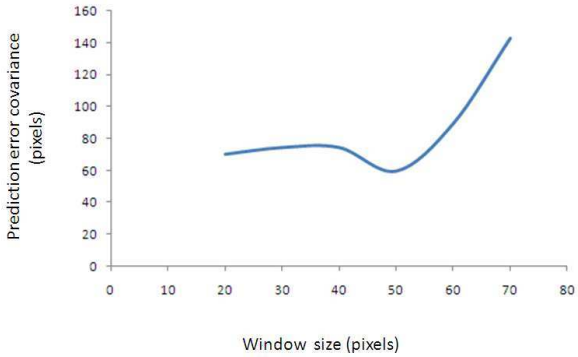

Figure 8. Prediction error in pixels for different kernel bandwidths.

Figure 9. Low threshold and high kernel bandwidth condition.

Figure 10. Some captured tracking sequences.

V. CONCLUSIONS

From the performance statistics of the experimental data it can be concluded that the performance of this tracker is mainly depended on two parameters. They are the threshold value and kernel bandwidth. However, the threshold value has a large amount of bandwidth within which it produces satisfactory result. Also, it is observed that under a suitable threshold value the change in kernel bandwidth does not affect the performance of the tracker too much except the extreme conditions. The overall performance of the tracker is very reliable for performing in indoor environment specifically in robotic applications without using any high level domain knowledge or heavy computational power. The average time for the calculation of each frame is approximately 130 ms out of which the time taken by the camera to grab each image and calculate disparity is 70ms. The frame rate can be made faster with more efficient programming. This means it is suitable for real-time applications. Future works for this application would be introducing an adaptive thresholding to cope with the changing environment automatically. Also using of weighted pixel value may solve the problem with window size variation.

REFERENCES

[1] L. Cai, L. He, Y. Xu, Y. Zhao, X. Yang, "Multi-object detection and tracking by stereo vision." *Pattern Recognition,* vol.43, no. 12, pp. 4028-4041, 2010.

[2] R. Mun˜oz-Salinas, E. Aguirre, M. Garcı´a-Silvente, "People detection and tracking using stereo vision and color." *Image and Vision Computing*, vol.25, pp. 995–1007, 2007.

[3] L. Peihua, "A clustering-based color model and integral images for fast object tracking." *Signal Processing: Image Communication,* vol.21, pp.676–687, 2006.

[4] S. Dubuisson, "Tree-structured image difference for fast histogram and distance between histograms computation." *Pattern Recognition Letters*, vol.32, pp.411-122, 2011.

[5] F. M. Porikli, "*Integral* Histogram: A Fast Way To Extract Histograms in Cartesian Spaces." in *Proc. CVPR*, pp.829-836, 2005.

[6] P. Viola, M. J. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features." in *Proc. CVPR*, pp. 511-518, 2001.

[7] G. Welch, G. Bishop. *An Introduction to the Kalman Filter*. UNC-Chapel Hill, TR 95-041, July 24, 2006

[8] D. W. Scott, *Multivariate Density Estimation: Theory, Practice and Visualization.* John Wiley & Sons, Inc. 1992, pp. 150-155.

[9] Point Grey Research: Bumblebee2
<http://www.ptgray.com/products/stereo.asp>

[10] K. Aizawa, K. Sakaue, Y. Suenaga, *Image Processing Technologies: Algorithms, Sensors, and Applications*. Marcel Dekker, Inc, 2004, pp.7-8.

[11] K. Fukunaga, L. Hostetler, "The estimation of the gradient of a density function, with applications." *pattern recognition. IEEE Transactions on Information Theory*, vol.21, no.1, pp.32–40, 1975.

[12] A. Ekin, A. M. Tekalp, Rajiv Mehrotra, "Autometic Soccer Video Analysis and Summarization." *IEEE Transactions on Image Processing,* vol.12, no.7, pp.796–807, July 15, 2003.

[13] K. H. Chen, W. H. Tsai, "Vision--based obstacle detection and avoidance for Autonomous land vehicle Navigation in outdoor roads." *Automation in Construction*, vol.12, pp.1–25, 2000.

[14] H. Zhou, Y. Yuan , C. Shi, "Object tracking using SIFT features and mean shift." *Computer Vision and Image Understanding,* vol.113, pp.345–352, 2009.