12-2014

Modeling and tracking relative movement of object parts

Praneeth Talluri
University of Nebraska at Omaha

Modeling and tracking relative movement of object parts

A Thesis
Presented to the
Department of Computer Science
and the
Faculty of the Graduate College
University of Nebraska
In Partial Fulfillment
of the Requirements for the Degree
Master of Science
University of Nebraska at Omaha

by

Praneeth Talluri

December, 2014

Supervisory Committee:

Dr. Qiuming Zhu

Dr. Haifeng Guo

Dr. William Mahoney

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Abstract

Modeling and Tracking relative movement of object parts

Praneeth Talluri, MS

University of Nebraska, 2014

Advisor: Dr. Qiuming Zhu

Video surveillance systems play an important role in many civilian and military applications, for the purposes of security and surveillance. Object detection is an important component in a video surveillance system, used to identify possible objects of interest and to generate data for tracking and analysis purposes. Not much exploration has been done to track the moving parts of the object which is being tracked. Some of the promising techniques like Kalman Filter, Mean-shift algorithm, Matching Eigen Space, Discrete Wavelet Transform, Curvelet Transform, Distance Metric Learning have shown good performance for keeping track of moving object.

Most of this work is focused on studying and analyzing various object tracking techniques which are available. Most of the techniques which are available for object tracking have heavy computation requirements. The intention of this research is to design a technique, which is not computationally intensive and to be able to track relative movements of object parts in real time. The research applies a technique called

foreground detection (also known as background subtraction) for tracking the object as it is not computationally intensive. For tracking the relative movement of object parts, a skeletonization technique is used. During implementation, it is found that using skeletonization technique, it is harder to extract the objects parts.

Dedication

To Mom and Dad,

who encouraged critical thinking

Table of Contents

List of Multimedia Objects

Figures:

I. Introduction

I.1. Problem

Not much work is done to find the relative movement of parts of the object which is being tracked. Most of this research is focused on studying and analyzing various techniques which are available for tracking object. The main intention behind this research is to find a less computationally intensive technique which can be used to track the relative movements of the parts of the object which is being tracked in real time.

I.2. Motivations

If we are able to track the relative movements of object parts in real time then this can be used in lot of applications like gaming, analyzing objects behavior, etc.

I.3. Significance

If we are able to understand the features of the moving object, we can utilize those features according to our need. Some of the things like optimizing athletic performance, analyzing assembly lines for any malfunctions can be done. This can also be used to biometrically and forensically analyze humans depending on step length, step width, walking speed, mean joint angles, etc.

I.4. Challenges

The problem is solvable, since there are lot techniques for extraction and tracking of a moving object. All the available techniques have to be studied to find out which technique can be used to extract object in a better way. Tracking the parts of a moving object which is being tracked is a challenging task.

I.5. Objectives

The main intention behind this research is to find a computationally less intensive method for tracking relative movement of the parts of an object which is being tracked in real time.

II. Overview

II.1. History of the problem

The research problem came into existence when the people were getting more interested in processing the data which is there in their region of interest only, rather than whole data. The problem gradually evolved and many techniques have been developed to process the objects and understand their features.

II.2. State of the art

As of today, many parameters and algorithms have been developed for understanding the features of the object.

Some of the prominent works have been carried out by National Laboratory of Pattern Recognition in Institute of Automation Chinese Academy of Sciences [Hu et al. 2012] [Chen et al. 2013], School of Automation in Beijing Institute of Technology [Zhou et al. 2010] [Zheng et al. 2012], Department of Electrical Engineering in University of Washington [Chu et al. 2011] [Lee et al. 2013], Department of Automation in Tsinghua University [Wang et al. 2011] [Wang et al. 2012] and many more. Mainly their current work focuses on improving the techniques in extraction and tracking of the objects.

III. Technical Section

III.1. Principles, Concepts, and Theoretical Foundations of the research problem

Region based tracking:

The region can be interpreted as the silhouette of the projection of an object in the scene, in relative motion with respect to camera. Region tracking generally reduces to the tracking of the center of gravity of regions [Meyer and Bouthemy 1992]. The problem with this method is inability to capture complex motion of objects in the image plane. Since the center of gravity of a region in the image does not correspond to the same physical point throughout the sequence, its motion does not accurately characterize the motion of the concerned region.

The representation of a region should not intended to capture the exact boundary. It should give a description of the shape and location that supports the task of tracking even in presence of partial occlusion.

Model based tracking:

3D models of the objects to be tracked are built in the form of CAD model, a set of planar parts, or even a rough 3D model such as an ellipsoid. 3D models can also be created either by using automated techniques or commercially available products.

[Lepetit and Fua 2005] There exists a trade-off between the inconvenience of building the 3D model and the increased reliability it affords and how to choose one approach over the other depending on the application at hand. In general, depending on the nature of the image features used, we can distinguish into two families. The first one is formed by edge based methods that match the projections of the target object's 3D areas to the area of high image gradient. The second includes all the techniques that rely on the information provided by the pixels inside the object's projection. It can be derived from optical flow, template matching or interest point correspondences.

Contour based tracking:

Contour based object tracking technique involves the tracking of the boundary contour of a moving and deforming object in a sequence of images [Patel and Patel 2012]. In general, the contour of the object is obtained in the first frame. When a rough contour of the desired structure is available on the first image of the sequence, the system automatically outlines the contours on the subsequent images. In parametric active contours, the contour is approximated by an explicit parametric model, typically by using a set of control points. B-splines are often used. In geometric active contours, the contour is approximated by an implicit function, as in the level set method [Wang et al. 2013]. In general, parametric contour methods are more efficient, and are thus more suitable for contour tracking in real time.

Feature based tracking:

Feature based object tracking tracks sub-features such as distinguishable points or lines on the object [Beymer et al. 1997]. The advantage of this approach is that even in the presence of partial occlusion, some of the sub features of the moving object remain visible. An object could have multiple sub features, grouping has to be done to find out what set of features belong to same object.

Deterministic methods:

These methods localize the tracked object in each frame by iteratively searching for a region and target window. For example, Mean shift algorithms, least-squared tracking, gradient ascent or decent algorithms are used [Wang and Hong 2012]. These methods are computationally efficient. However, these methods may converge to a local maximum. They are sensitive to background distraction, clutter, occlusions, and quick moving objects.

Stochastic (Probabilistic) methods:

These methods are able to maintain multiple hypothesis in the state space which can achieve more robustness to the local maximum. Common problems and methods:

- Linear - Gaussian estimation problem – Kalman filter [Lepetit and Fua 2005]
- Non- linear – Gaussian estimation problem – Extended Kalman filter(EKF) [Lepetit and Fua 2005]

- Non-linear – Non-Gaussian estimation problem – Particle filter(sequential Monte Carlo methods)[Lepetit and Fua 2005]

Kalman Filter:

Kalman filtering is a generic tool for recursively estimating the state of a process [Lepetit and Fua 2005]. The successive states $s_t \in R^n$of a discrete-time controlled process are assumed to evolve according to a dynamics model of the form:

$$s_t = As_{t-1} + w_t$$

Where matrix A is called the state transition matrix, and w_t represents the process noise, taken to be normally distributed with zero mean. For tracking purposes, this relation is used to enforce a motion model and is directly related to the term $p(s_t|s_{t-1})$.

The image measurements z_t, such as image location at time t of some feature points, are assumed to be related to the state s_t by a linear measurement model:

$$z_t = Cs_t + v_t$$

Where $v_t$represents the measurement noise. At each time step, the Kalman filter makes a first estimate of the current state called the a priori state estimate and that we denote $\bar{s}_t$. It is then refined by incorporating the measurements to yield a posteriori estimate s_t. $\bar{s}_t$, as well as its covariance matrix $\bar{S}_t$, are computed during a time update or prediction stage. Given knowledge of the process prior to time t and using the dynamic model, we can write

$$\bar{s}_t = As_{t-1}$$

$$\bar{S}_t = AS_{t-1}A^T + \Lambda_w$$

Where S_{t-1} is a posteriori estimate error covariance for the previous time step, and Λ_w is the process covariance noise that measures how well the motion model is respected in reality. The above expression of $\bar{S}_t$ is derived from the classical propagation formula. Next, the Kalman filter proceeds to a "measurement update" or correction. A posteriori state estimate s_t and its covariance matrix S_t are now generated by incorporating the measurements z_t by writing

$$s_t = \bar{s}_t + G_t(z_t - C\bar{s}_t),$$

$$S_t = \bar{S}_t - G_t C\bar{S}_t,$$

Where the Kalman gain G_t is computed as

$$G_t = \bar{S}_t C^T (C\bar{S}_t C^T + \Lambda_v)^{-1},$$

with Λ_v being the covariance matrix of the measurements.

In the context of tracking, the a priori state estimate $\bar{s}_t$ can be used to predict the location of an image feature. The predicted measurement vector $\bar{z}_t$ is indeed simply

$$\bar{z}_t = C\bar{s}_t$$

The uncertainty on this prediction can be represented by the covariance matrix Λ_z estimated by propagating the uncertainty, which gives us

$$\Lambda_z = C\bar{S}_t C^T + \Lambda_v$$

$\bar{z}_t$ and Λ_z are useful to restrict the search for image features to a region of the image.

The Kalman filter is a powerful and popular tool to combine noisy measurements from different image cues in a statistically well-grounded way. It is also useful to stabilize the

camera trajectory using a motion model. However, this has a price. A simple motion model, such as one that assumes constant velocity, is fully justified in some applications such as visual surveying. But for applications that involve human motion that can be jerky, a low-order dynamical model is not very realistic. For Augmented Reality applications, this may result in some "lag" of the inserted virtual objects [Lepetit and Fua 2005].

Another limitation comes from the fact that the measurements are often assumed to be mutually independent. While this assumption is not inherent to the Kalman filter formulation, it is difficult to avoid in practice without greatly increasing the complexity of the computation.

In reality, the measurements are rarely independent.

Extended Kalman Filter:

Kalman filtering assumed that the relationship between the state and the measurements was linear. It is rarely the case in tracking applications. The relationship should be expressed as

$$z_t = c(s_t, v_t)$$

Where c is a non-linear function. The Extended Kalman Filter (EKF) approximates this function c by its first order Taylor expansion, which allows the use of the formalism introduced below. This yields the following update equations [Lepetit and Fua 2005].:

$$G_t = \bar{S}_t C^T (C\bar{S}_t C^T + V\Lambda_V V^T)^{-1},$$

$$s_t = \bar{s}_t + G_t\big(z_t - c(s_t, 0)\big),$$

where C is now the Jacobian of c with respect to the state s computed at $\bar{s}_t$. V is the Jacobian of c with respect to v and is often taken to be the Identity matrix in practice. The last update equation evaluates $S_t$with C computed at the updated state s_t, which is usually considered to be the best linearization point, giving self-consistent linearization estimates:

$$S_t = \bar{S}_t - \ G_t C \bar{S}_t$$

Particle Filter:

The probability distribution of states in Kalman filtering is restricted to be Gaussian, which is not optimal in ambiguous cases when multiple hypotheses may have to be considered. Particle

Filters, such as Condensation or Monte Carlo filters, have been introduced as a more general representation by a set of weighted hypotheses [Lepetit and Fua 2005], or particles. Another advantage is that they do not require the linearization of the relation between the state and the measurements. Each particle can be seen as a hypothesis for the state estimate. In other words particle filters can maintain several hypotheses over time, which gives them increased robustness.

A large number of particles – perhaps as many as several thousand when the motion is poorly defined – can be required [Lepetit and Fua 2005], which slows down the tracking process. Next, for online applications, the system must provide a state estimate in each

frame, usually taken to be the mean or the median of the particles. While robust, this estimate is not particularly accurate. This results in motion estimates that are not as smooth as they should be, which is a major drawback for many applications.

Particle filtering is a sequential importance sampling algorithm for estimating properties of hidden variables in a hidden Markov model, given observations. Given some set of observations of feature values $O_t = (o_1, o_2, o_3, \dots\dots, o_t)$ for a target up to time t, the aim of a particle filter system is to estimate the posterior $p(s_t|O_t)$, where s_t is the state of the target at time t, based on the observation model (the likelihood) $p(o_t|s_t)$ and the dynamic model $p(s_t|s_{t-1})$ [Wang et al. 2013]:

$$p(s_t|O_t) \propto p(o_t|s_t) \int p(s_t|s_{t-1}) p(s_{t-1}|O_{t-1})\, ds_{t-1}$$

The tracking result is obtained as the maximum a-posteriori (MAP) estimate, which is: $x_t^* = \arg\max p(s_t|O_t)$.

The particle filter approach approximates the integral by using a set of weighted samples (particles) $\{s_t^i, w_t^i\}_{i=1}^N$, where each s_t^i is an estimate of state and w_t^i the corresponding weight. These particles are generated during the initialization stage, and evolve continually.

Medial Axis Transformation:

To get an intuitive feeling for this concept, consider starting a grass fire along a curve in the plane, like the outer closed curve. The fire starts at the same moment, everywhere along the curve, and it grows at constant speed in every direction. The medial axis is the

set of locations where the front of the fire meets itself. In mathematical language: it is the set of points that have at least two closest points on the curve. If we start the fire along the boundary of a geometric shape in $\mathbb{R}^k$, we generically get a medial axis of dimension $k-1$, one less than the dimension of the space. In the plane, the medial axis is a (one-dimensional) graph whose branches correspond to regions of the shape it represents.

The medial axis transform (MAT) [Mann and Singh 2012] of an image is computed by calculating the Euclidean distance transform of the given input image pattern. The MAT is described as being the locus of the local maxima on the distance transform. After the computation of Euclidean distance transform (EDT) of the input image, the EDT is represented in image representing the Euclidean distances as gray levels. The same is shown in above images. The maximum Euclidean distance is represented as maximum gray level intensity in the EDT image. The pixel coordinates of the maximum gray level intensity are extracted from the EDT image by converting the EDT image into row x column matrix. The row and column of the matrix gives the coordinates of the MAT line of the image pattern.

Moment Invariants:

A uniqueness theorem (Ming-Kuei Hu 1962) states that if $f(x,y)$ is piecewise continuous and has nonzero values only in a finite part of the xy plane, moments of all orders exist, and the moment sequence (M_{pq}) is uniquely determined by $f(x,y)$. Conversely, (M_{pq})

uniquely determines *f*(*x*,*y*). In practice, the image is summarized with functions of a few lower order moments. For *p*,*q* = 0,1,2,... Adapting this to scalar (greyscale) image with pixel intensities *I*(*x*,*y*), raw image moments M_{ij} are calculated by

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

If *f*(*x*, *y*) is a digital image, the central moments can be calculated by

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

Where

$$\bar{x} = \frac{M_{10}}{M_{00}} \text{ and } \bar{y} = \frac{M_{01}}{M_{00}}$$

It is possible to calculate moments which are invariant under translation, changes in scale, and also rotation. Most frequently used are the Hu set of invariant moments.

$$\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].
\end{aligned}$$

A general theory on deriving complete and independent sets of rotation invariant moments was proposed by J. Flusser and T. Suk. They showed that the traditional Hu's invariant set is not independent nor complete. I_3 is not very useful as it is dependent on the others. In the original Hu's set there is a missing third order independent moment invariant:

$$I_8 = \eta_{11}[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$$

III.2. Techniques that have been used by other researchers for the research problem

Region based tracking:

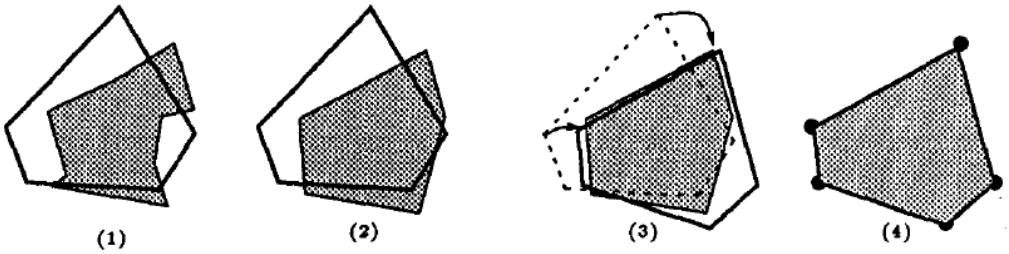
The region is represented using some of its boundary points. The contour is sampled in such a way that it preserves shape information of the silhouette. The points that best capture the global shape of the region are selected. This is achieved through a polygonal approximation of the region. A good approximation should be close to the original shape and have the minimum number of vertices.

The region can be approximated accurately by this set of vertices. This representation offers the property of being flexible enough to follow the deformations of the tracked silhouette. Furthermore this representation results in a compact description which decreases the amount of data required to represent the boundary, and it yields easily

tractable models to describe the dynamic evolution of the region. [Meyer and Bouthemy 1992]

The region tracking algorithm requires the matching of the prediction and an observation. The matching is achieved more easily when dealing with convex hull. Among the boundary points approximating the silhouette of the region, we retain only those which are also the vertices of the convex hull of the considered set of points. These polygonal approximations only play a role as "internal items" in the tracking algorithm to ease the correspondence step between prediction and observation. It does not restrict the type of objects to be handled.

The region descriptor is intended to represent the silhouette of the tracked region, all along the sequence. The tracked region is represented with the same number of points during successive time intervals of variable size. At the beginning of the interval, in the segmented image the number of points, n, necessary to represent the concerned region is determined. This number fixed as long as the distance between the predicted region and the observation extracted from the segmentation is not too important. The moment the distance becomes too large, the region descriptor is reset to an initial value equal to the observation. This announces the beginning of a new interval. The region descriptor is represented with a vector of dimension 2n. This vector is the juxtaposition of the coordinates (x_i, y_i) of the vertices of the polygonal approximation of the region: $[x_1, y_1, x_2, y_2, \dots\dots, x_n, y_n]^T$.



The measurement algorithm : (1) Observation obtained by the segmentation (grey region), and prediction (solid line) ; (2) Convex hull of the observation ; (3) Matching of polygons ; (4) Effective measurement : vertices of the grey region.

Figure 1: The measurement algorithm for region based tracking [Meyer and Bouthemy 1992]

Contour based tracking:

Traditional snake models suffer from a serious limitation when used for tracking in image sequences: the convergence of results is very sensitive to the initial contour location. To deal with this problem, various estimation tools, such as the Kalman filter and particle filters, can be used to update parameter values over the sequence. For example, [Terzopoulos and Szeliski 1992] used a Kalman filter to track a fixed number of marker points, or parametric values, such as a B-spline's control points. But, the Kalman filter assumes linear system and measurement models, which is unsuitable for many applications.

In order to track contours with non-Gaussian and nonlinear state densities in cluttered video sequence, [Isard and Blake 1998] introduced the condensation

algorithm. They used a B-spline representation for object contours, and particle filters to track the curve parameters given noisy observations. Since their approach only allows affine deformations of the contour, it is unsuitable for deforming objects, undergoing local deformations. [Rathi et al. 2007] combined a particle filtering algorithm with the geometric active contour framework to give an approach that can be used for tracking moving and deforming objects. However, they directly track affine deformations, while using an approximately linear observer to estimate any non-affine deformation of the object contour. Thus, their method cannot deal with complex contour deformations. [Vaswani et al. 2010] proposed a further algorithm, Deform PF-MT. They suggest that in most real problems, much of the contour deformation depends on a few parameters, while the deformation in the rest of the state space is small. Hence they use the deformations at a small sub-sampled set of locations along the contour as an effective basis space for particle filtering. However, they still explicitly track the contour deformations. In the presence of complexity and uncertainty of object deformations, their method is error prone. Furthermore, the above approaches only use simple observation models, which do not provide stable tracking target in the presence of large shape changes or significant occlusion.

Kalman filter:

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups:

time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance. [Welch and Bishop 2001]

The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems as shown below

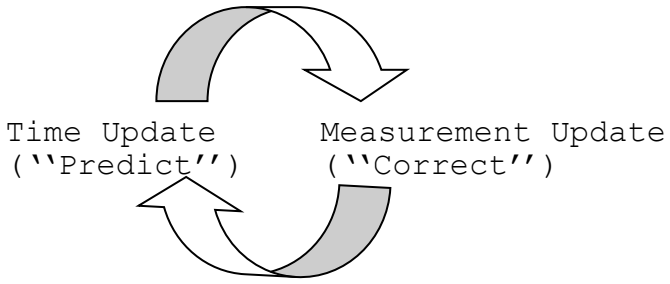


Figure 2: The ongoing discrete Kalman filter cycle

The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement of time. The specific equations for the time and measurement updates are presented below

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$

The time update equations project the state and covariance estimates forward from time step to step k-1.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (1 - K_k H)P_k^-$$

The first task during the measurement update is to compute the Kalman gain, K_k. The next step is to actually measure the process to obtain z_k , and then to generate an a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance estimate.

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible.

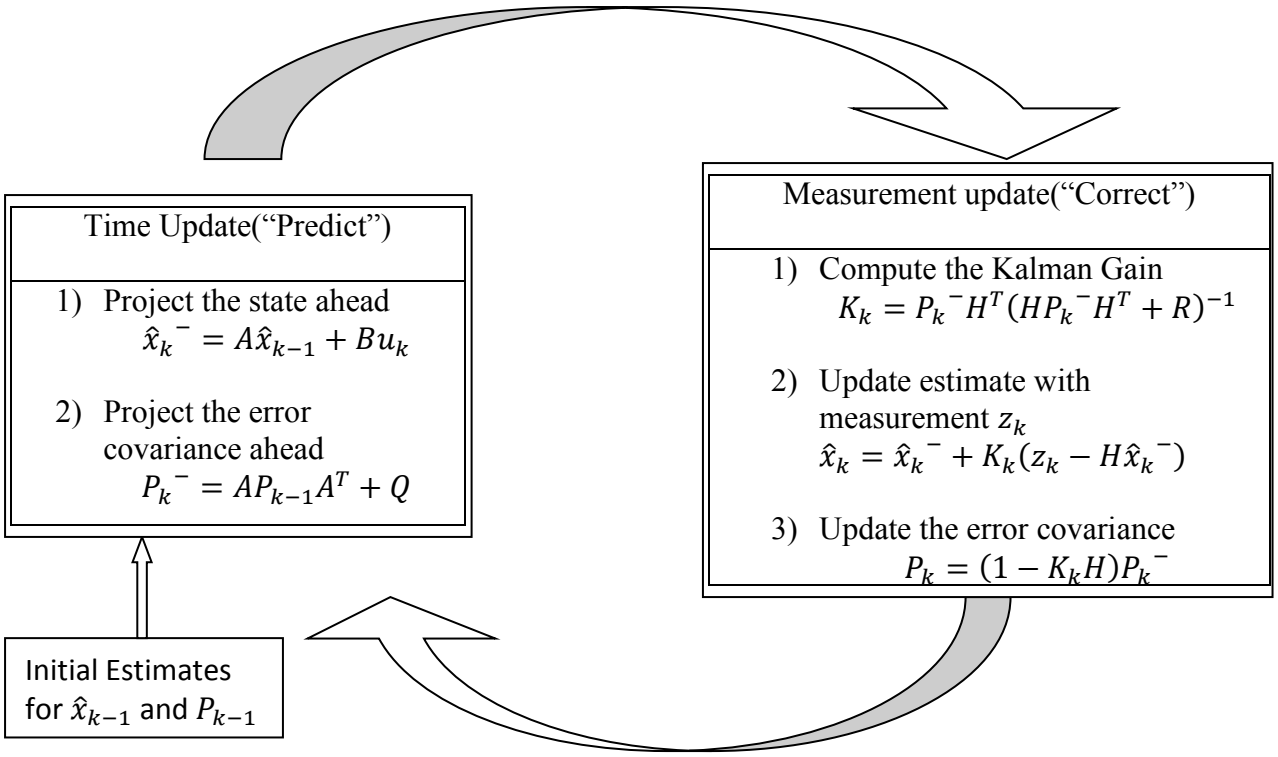


Figure 3: The complete picture of the operation of the Kalman Filter

Extended Kalman filter:

The complete set of EKF equations is shown below. We now attach the subscript to the Jacobians A, W, H, and V, to reinforce the notion that they are different at (and therefore must be recomputed at) each time step. [Welch and Bishop 2001]

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0)$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$$

As with the basic discrete Kalman filter, the time update equations project the state and covariance estimates from the previous time step k-1 to the current time step k.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\big(z_k - h(\hat{x}_k^-, 0)\big)$$

$$P_k = (1 - K_k \mathrm{H_k}) P_k^-$$

As with the basic discrete Kalman filter, the measurement update correct the state and covariance estimates with the measurement z_k. H_k and V are the measurement Jacobians at step k, and R_k is the measurement noise covariance at step k. (We subscript R allowing it to change with each measurement.)

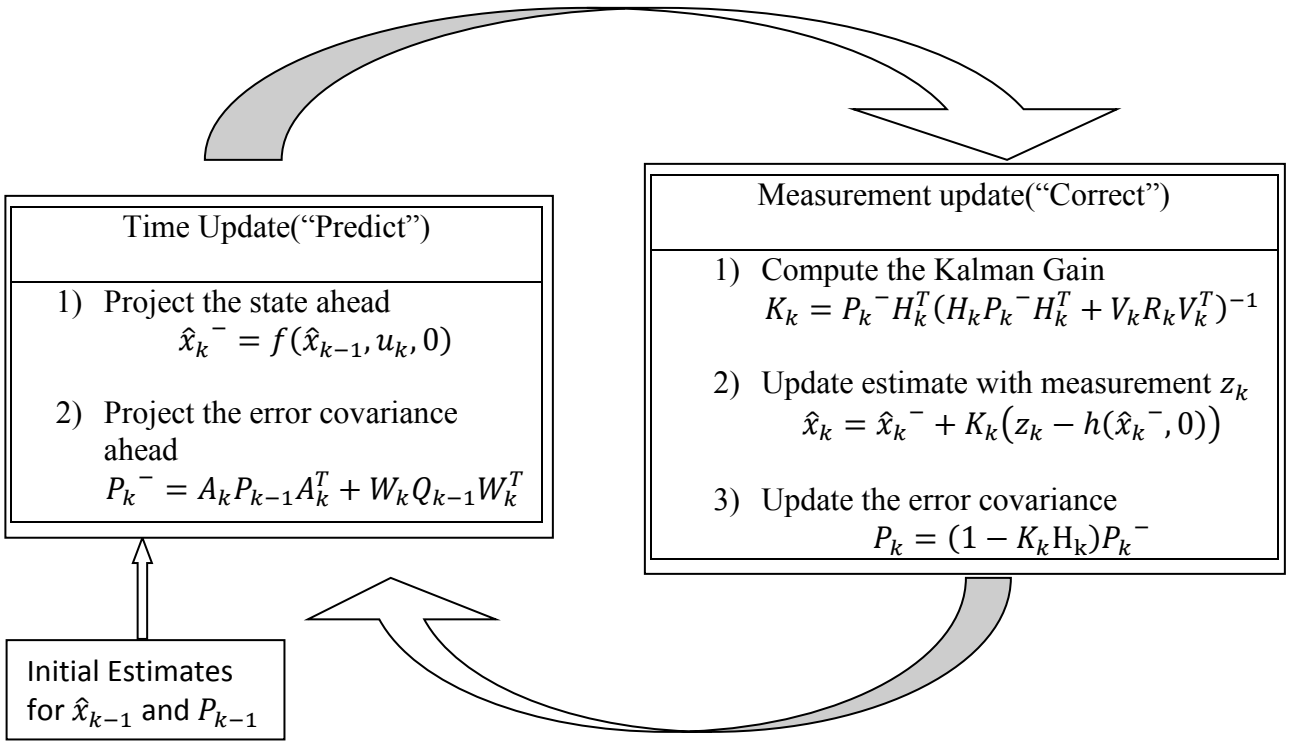


Figure 4: The complete picture of the operation of the Extended Kalman Filter

An important feature of the EKF is that the Jacobian $H_k$in the equation for the Kalman gain $K_k$serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_k and the state via h, the Jacobian H_k affects the Kalman gain so that it only magnifies the portion of the residual $z_k - h(\hat{x}_k, 0)$ that does affect the state. If in overall measurements there is not a one-to-one mapping between the measurement z_k and the state via h, then the filter will quickly diverge. In this case the process is unobservable.

Rao – Blackwellished particle filter:

Given N particles (samples) $\left\{r_{0:t-1}^{(i)}, x_{0:t-1}^{(i)}\right\}$ at time t-1, approximately distributed according to distribution $\left(r_{0:t-1}^{(i)}, x_{0:t-1}^{(i)} | y_{1:t-1}^{(i)}\right)$, RBPFs allow us to compute N particles $\left(r_{0:t}^{(i)}, x_{0:t}^{(i)}\right)$ approximately distributed according to the posterior $p\left(r_{0:t}^{(i)}, x_{0:t}^{(i)} | y_{1:t}^{(i)}\right)$, at time t. This is accomplished with the algorithm shown below: [Doucet et al. 2000]

Generic RBPF:

1. Sequential Importance Sampling

   - For i = 1,….,N, sample:

     $$\left(\hat{r}_t^{(i)}\right) \sim q\left(r_t | r_{0:t-1}^{(i)}, y_{1:t}\right)$$

     and set:

     $$\left(\hat{r}_{0:t}^{(i)}\right) \triangleq \left(\hat{r}_t^{(i)}, \hat{r}_{0:t-1}^{(i)}\right)$$

   - For i = 1,…,N, evaluate the importance weights up to a normalizing constant:

     $$w_t^{(i)} = \frac{p\left(\hat{r}_{0:t}^{(i)} | y_{1:t}\right)}{q\left(\hat{r}_t^{(i)} | \hat{r}_{0:t-1}^{(i)}, y_{1:t}\right) p\left(\hat{r}_{0:t-1}^{(i)} | y_{1:t-1}\right)}$$

   - For i = 1,…,N, normalize the importance weights:

     $$\widetilde{w}_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)}\right]^{-1}$$

2. Selection Step

   - Multiply/suppress samples $\left(\hat{r}_{0:t}^{(i)}\right)$with high/low importance weights $\widetilde{w}_t^{(i)}$, respectively, to obtain N random samples $\left(\tilde{r}_{0:t}^{(i)}\right)$approximately distributed according to $p\left(\tilde{r}_{0:t}^{(i)}|y_{1:t}\right)$.

3. MCMC step

   - Apply a Markov transition kernel with invariant distribution given by $p\left(r_{0:t}^{(i)}|y_{1:t}\right)$ to obtain$\left(r_{0:t}^{(i)}\right)$.

Medial Axis Transformation:

Medial Axis Transformation can be achieved by using Thinning Algorithm [Kardos et al. 2009] or by distance field based methods like computing Voronoi regions [D. T. LEE 2009].

By Thinning:-

The points of an image can be considered as a set of points in 2-dimensional digital space denoted by $\mathbb{Z}^2$. Each point p is represented as a pair $p = (p_x, p_y)$. A 2-dimensional (8, 4) binary digital picture can be described with the quadruple $(\mathbb{Z}^2, 8,4, B)$, where $\mathbb{Z}^2$ is the set of picture points, B $\subseteq \mathbb{Z}^2$ is the set of black points, for which we will assign the value “1”; its complement, $\bar{B} = \mathbb{Z}^2 \backslash B$ is the set of white points

to which the value ”0“ is assigned. A black component is a maximal 8–connected set of black points, while a white component is defined as a maximal 4–connected set of white points.[Kardos et al. 2009]

Proposed KNP uses the subsets of black points denoted by S_{inner}, S_{α}, S_{β}, S_{γ}, S_{corner}, $S_{visited}$, and we need four additional Boolean functions. These functions have value of ”1” (true) if the following conditions hold:

$\Delta_{\alpha}(p)$: $p \in S_{\alpha} and \nexists q \in S_{\alpha}\ black\ point\ such\ that\ \Gamma(p,q)\ and\ p \prec q$

$\Delta_{\beta}(p)$: $p \in S_{\beta} and\ the\ following\ two\ conditions\ hold$:

$i)\ If\ \exists q \in S_{\beta}\ black\ point\ such\ that\ \Gamma(p,q)\ and\ q \prec p, then\ \exists r \in S_{\alpha}\ such\ that\ \Gamma(q,r)$

$ii)\ For\ each\ q \in S_{\alpha}\ black\ point, \rightarrow \Gamma(p,q)\ or\ \exists r \in S_{\alpha}\ such\ that\ \Gamma(q,r) and\ q \prec r$

$\Delta_{\gamma}(p)$: $p \in S_{\gamma}, \nexists q \in S_{\alpha} \cup S_{\beta}\ black\ point\ such\ that\ \Gamma(p,q), and\ if\ p\ is\ not\ a\ safe\ \gamma - point, then\ \nexists q \in S_{\gamma}\ black\ point\ such\ that\ \Gamma(p,q)$

$\Delta_{corner}(p)$: $p \in S_{corner}\ and\ \exists q \in \left(S_{inner} \cap N_8^*(p)\right)\ black\ point\ or\ \exists r \in \left(S_{corner} \cap N_8^*(p)\right)\ black\ point\ such\ that\ r \prec p$

Preference rules for decision pairs are built in these Boolean functions. First of all, a priority order was determined for the introduced classes of points, in which α–points are the most preferred, followed by β–points, which are preferred to γ–points, and finally, safe γ–points have higher priority than non–safe γ–points. If p has higher priority

than q for every decision pair {p, q}, then p can be removed, but if q is preferred in such a pair, then p still has a chance to be removed, namely, if q is not preferred in another pair.

It can also occur that p and q have the same priority. (This can only happen in the case of α– and β–pairs, as we have already proved, that γ–pairs contain exactly one safe γ–point.) In such situations, we use the relation $\prec$ for the decision: for α–points, we prefer p to q if $q \prec p$, however, in the case of β–pairs, $p \prec q$ must be fulfilled for deletion of p. (These opposite conditions for α– and β–pairs will help to ensure maximal thinning.)

By Voronoi Methods:-

The algorithm for constructing the Voronoi diagram of a simple polygon. We assume that G is represented by a list of N = n + m elements $e_1, e_2, \ldots, e_N$ where n is the number of edges and m the number of reflex vertices. The algorithm to be described is based on the divide-and-conquer technique. That is, we shall divide G into two lists G1 = (e_1, e_2, $e_{N/2}$) and G2 = ($e_{N/2+1}$, e_N) and recursively construct the Voronoi diagrams VOD(G_1) and VOD(G_2). Then we merge VOD(G_1) and VOD(G_2) to form the diagram VOD(G). Since the merge process takes O(N) time, the overall running time is O(N log N). [D. T. LEE 2009]

Because we are interested in only the portion of the diagram that is internal to G, we shall restrict our discussion to the interior of G. To distinguish the interior of G from the exterior we shall assume that G is traversed in a counterclockwise direction so that

the interior of G always lies to the left. Furthermore, when we say the Voronoi diagram of a list of elements, we mean the portion that is to the left of the list of elements. For implementation purposes we instead partition the list of elements of G into chains C_1, C_2, …, C_h and apply the divide-and-conquer technique to S = { C_1, C_2, …, C_h.} The reason for it is that the Voronoi diagram of a chain can be computed straightforwardly in time proportional to the number of elements in the chain, and yet the running time of the modified algorithm remains the same, i.e., O(NlogN).

Next these h Voronoi diagrams are merged two at a time to form VOD(C_1 U C_2), VOD(C_3 U C_4), etc., as illustrated by the binary merge tree shown

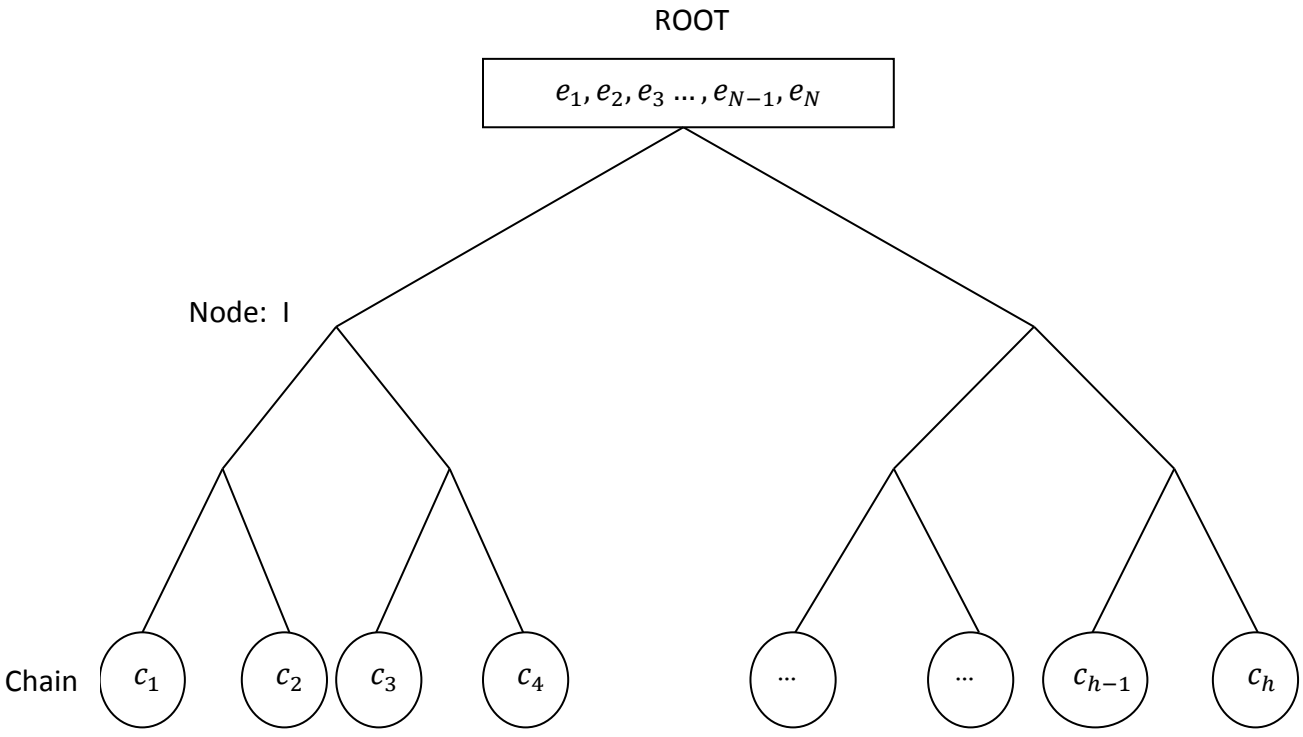


Figure 5: Binary merge tree for computing the Voronoi diagram of a simple polygon

The leaves are the Voronoi diagrams for chains and the internal nodes represent the Voronoi diagrams for the elements in the corresponding subtrees. Thus, the root of the binary merge tree will represent the final Voronoi diagram of the simple polygon. The height of the merge tree is $[\log_2 h]$ and each internal node I represents work for merging of the two Voronoi diagrams associated with its two sons and is $0(t_l + t_r)$, where t_l and t_r denote the numbers of elements in the left and right subtrees of node I. Thus, the time required for the program is at most $[\log_2 h] * O(N)$ which is $O(N \log N)$.

To facilitate the computation of the merge curve $B(S_1, S_2)$ for some sets S_1 and S_2 of elements we make, use of the following.

1) The merge curve $B(S_1, S_2)$ consists of component bisectors of the form $B(e_i, e_j)$ where $e_i \in S_1$ and $e_j \in S_2$. To compute the curve imagine moving a point z along $B(e_i, e_j)$. In order for z to be in $B(e_i, e_j)$ the projections of z onto e_i and $e_j$must belong to e_i and e_j respectively. That is, if e_i is a segment, then z must be within the region defined by two parallel lines through endpoints of e_i and perpendicular to e_i and if e_i is a reflex vertex, the z must be within the wedge defined by the lines perpendicular to e_{i-1} and to e_{i+1} and by the apex e_i.
2) In computing $B(S_1, S_2)$, where $S_1 = \{e_i, e_{i+1}, \dots, e_j\}$ and $S_2 = \{e_{j+1}, e_{j+2}, \dots, e_k\}$, we terminate the computation when the stopping condition is met, i.e., when $B(e_u, e_v)$ is reached where $e_u \in S_1$ and $e_v \in S_2$ and all the elements in $S_1$and S_2 lie on the side of $\overleftrightarrow{e_u, e_v}$. It can be shown that this can only happen

when both e_u and e_v are reflexive vertices. Note that e_u can be q_i which is an endpoint of e_i and e_v can be q_{k+1} which is an endpoint of e_k.

Skeleton Pruning:

The most significant factor in skeleton matching is the skeleton's sensitivity to objects boundary deformation. A little noise or variation in boundary generates redundant skeleton branches that disturb the topology of the of the skeleton's graph. Below are the various methods to prune those redundant branches.

Skeleton Pruning with Contour Matching:[Bai et at. 2007]

Given a partition Γ of the boundary ∂D of a simply connected set D (i.e, ∂D consist of one simple closed curve), the skeleton pruning is defined as the removal of all skeleton points $s \in S(D)$ whose generating points lie in the same open segment of the partition. More precisely, the pruned skeleton is composed of all points $s \in S(D)$ such that $Tan(s)$ is not contained in the same open segment of the partition Γ.

Skeleton Pruning using Discrete Curve Evolution: [Bai et at. 2007]

Discrete curve evolution(DCE) works by simplifying the shape. The basic idea of DCE of polygons is:

- In every evolutional step, a pair of consecutive line segments s_1, s_2 is replaced by a single line segment joining the endpoints of $s_1 \cup s_2$. The

key property of this evolution is the order of the substitution. The substitution is achieved according to a relevance measure K given by:

$$K(s_1, s_2) = \frac{\beta(s_1,s_2)l(s_1)l(s_2)}{l(s_1)+l(s_2)},$$

Where line segments s_1, s_2 are the polygon sides incident to a vertex $v, \beta(s_1, s_2)$ is the turn angle at the common vertex of segments s_1, s_2, l is the length function normalized with respect to the total length of a polygonal curve C.

- The higher value of $K(s_1, s_2)$, the larger is the contribution of the arc $s_1 \cup s_2$ to the shape.

Given the input boundary polygon P with n vertices, DCE produces a sequence of simpler polygons $P = P^n, P^{n-1}, \dots, P^3$ such that $P^{n-(k-1)}$ is obtained by removing a single vertex v from P^{n-k} whose shape contribution measured by K is the smallest.

Given a skeleton $S(D)$ of a planar shape D and given a DCE simplified polygon P^k, we perform skeleton pruning by removing all points $s \in S(D)$ such that the generating points $Tan(s)$ of s are contained in the same open DCE segment. Each pruned point s results from a local contour part with respect to the DCE partition and, therefore, s can be considered as an unimportant skeleton point and can be removed. The simplification of the boundary contour with DCE corresponds to pruning complete branches of the

skeleton. In particular, a removal of a single convex vertex v from P^{n-k} to obtain $P^{n-(k-1)}$ by DCE implies a complete removal of the skeleton branch that ends at v.

Iterative algorithm to prune skeleton:[Bai and Latecki 2007]

In each step one branch with the lowest weight is removed. Removing the end branch will not change skeletons topology. The end branch with low contribution to the reconstruction is removed first.

Weight w_i for each branch $P\big(l_i, f(l_i)\big)$ is defined as

$$w_i = 1 - \frac{A\Big(R\left(S - P\big(l_i, f(l_i)\big)\right)\Big)}{A\big(R(S)\big)}$$

where function $A()$ is the area function, $R(s)$ denotes the radius of the maximal disk $B(s, r(s))$ centered at a skeleton point s, $P\big(l_i, f(l_i)\big)$ is the shortest skeleton path between l_i and $f(l_i)$. The intuition for skeleton pruning is that an end branch with a small weight w_i has a negligible influence on the reconstruction, since the area of the reconstruction without this branch is nearly the same as the area of the reconstruction with it. Therefore, it can be removed. The below skeleton pruning is based on iterative removal of end branches with the smallest weight until the desirable threshold is met:

1. Initialize the weights of all end branches $w_i^{(0)}\big(i = 1,2, \dots, N^{(0)}\big)$ based on the original skeleton $S^{(0)}$:

$$w_i^{(0)} = 1 - \frac{A\left(R\left(S^{(0)} - P\left(l_i^{(0)}, f\left(l_i^{(0)}\right)\right)\right)\right)}{A\big(R(S^{(0)})\big)}$$

2. In the kth iteration step, for $i = 1,2, \dots, N^{(k)}$ compute the weight for each end branch in the skeleton $S^{(k)}$:

$$w_i^{(k)} = 1 - \frac{A\left(R\left(S^{(k)} - P\left(l_i^{(k)}, f\left(l_i^k\right)\right)\right)\right)}{A\left(R\left(S^{(k)}\right)\right)}$$

3. Select the minimal weight $w_{min}^{(k)}$. If $w_{min}^{(k)}$ is smaller than threshold t, go to 4; else, stop the evolution and output the $S^{(k)}$ as the final result.

4. Remove end branch $P_{min}^{(k)}$ with the lowest weight $w_{min}^{(k)}$ and obtain the new skeleton:

$$S^{(k+1)} = S^{(k)} - P_{min}^{(k)}$$

5. Set $k = k + 1$ and go to 2.

It is easy to see that this algorithm preserves the skeleton topology, since only end branches are removed.

Moment Invariants:

Invariant moment group is a special function. If the target image is a binary image, moment group can only describe the target point in the X-Y plane of space on the arrangement of information, which is the target shape information. Different targets species of the same moment group is different, and therefore it can be used to identify the target.

The recognition algorithm on surface targets can be divided into two parts, offline and online. First, BP network is trained off-line by using infrared image samples. Then, the infrared image is identified on line by the trained BP network. The identification can be completed by extracting the target from the infrared image after preprocessing and region segmenting, and then computing invariant moments on the binary image of the target, and finally entering the image invariant moment group as the input of the BP network. The surface target recognition principle based on BP neural network and moment invariants is shown

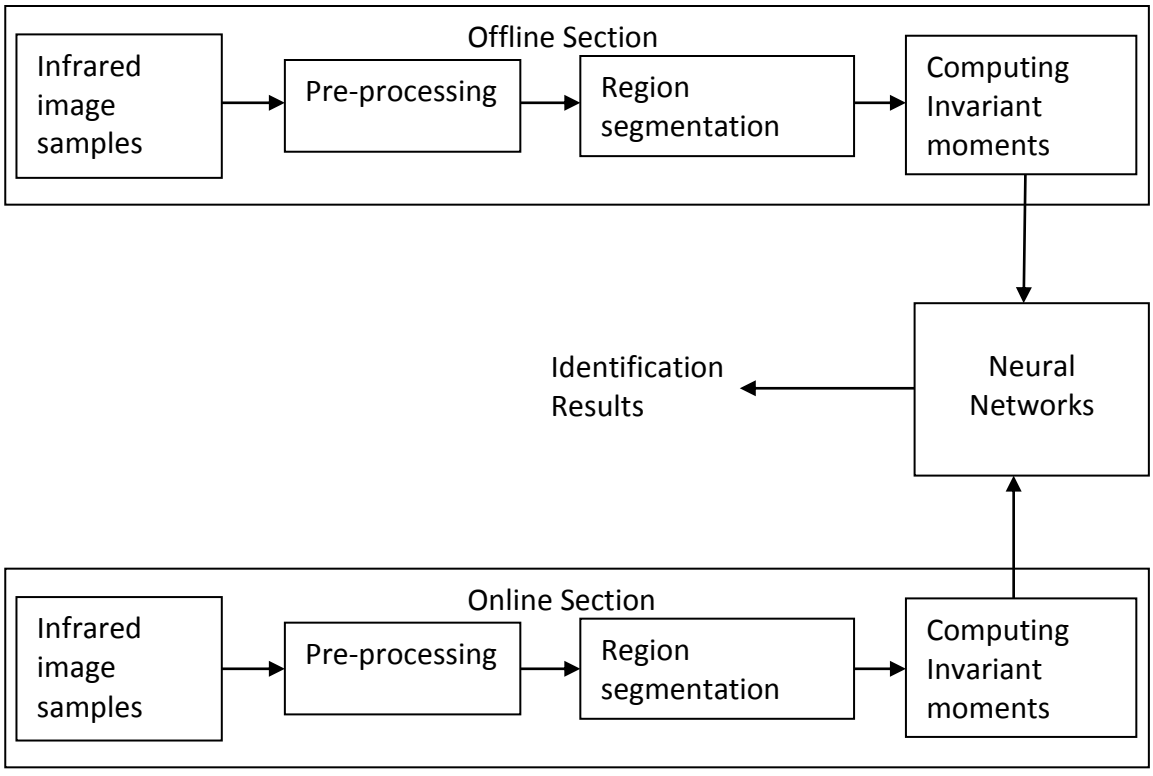


Figure 6: Target Recognition Algorithm Based on BP Neural Network and Moment Invariants [Tian and Qi 2013]

After the extraction of the image feature, a group of data to reflect the target attribute is obtained (feature vectors). The next step is to identify and classify based on this set of feature vectors. Practical classification algorithm is one of the core technologies of infrared image automatic target recognition system.

BP network is a multilayer feedforward neural network and takes the error back propagation learning algorithm, which is widely used in function approximation, pattern recognition, classification, data compression and so on. Processing of the neural network classifier includes training and identification, where training is the process of the classifier design on the basis of the training sample; identify is the process of identifying unknown images by matching unknown images and classifiers which have been trained.

In their paper [Tian and Qi 2013], three-layer BP network is designed as the classifier with each input of the network corresponding to one of the characteristics of the sample, the output nodes equaling to the number of categories, all output nodes composing an output column vector and an output column vector corresponding to a category. In the simulation, training sample includes two classes. In the training phase, if the input training samples are for the first goals, then the desired output is (1, 0), while (0, 1) for the second goals. In the recognition phase, if a new image is used to the input, a one-dimensional column vector output can be got via the trained BP network mapping.

III.3. Relevant technologies that would be useful to this research

Foreground detection is the basic technique which has been used to extract the image's region of interest. Medial Axis Transformation is another prominent technique to which has been used to obtain the skeleton of the objects. But all the techniques discussed above were quite useful in analyzing the problem.

III.4. Methodologies I applied in this research

I have studied and analyzed various conference proceedings and various journals which were related to this topic. These helped me understand the various types of techniques which are currently being used and previously used for this kind of problem.

I intended to design an algorithm which is computationally less intensive. So I have used previously known techniques and designed an algorithm which might work for the problem. In designing the algorithm I have used foreground detection which is a basic technique to detect objects. For tracking relative movement of object parts I have used skeletonization (medial axis transformation) technique.

My end goal was to extract the moving object as it is so that I can perform some computations on the extracted object in a computationally less intensive way. Then I came up with the following technique.

This technique consists of the following procedures:

- Moving object extraction
- Morphological Operations
- Medial Axis Transformation
- Moving Parts of the moving object

Moving object extraction:

Moving objects can be obtained either by using background subtraction or frame difference methods depending on the situation. Background subtraction is used in the situations where an initial background can be obtained. Frame difference can be obtained where an initial background cannot be obtained. But for my experiments I have used background subtraction since we can easily get all the pixels of the moving object.

Morphological Operations:

We will get rid of unwanted pixel groups by setting up a threshold. Then morphological operations are done on the object which is obtained. The morphological operations help us to remove speckle noise and also help us to have a smoother boundary. This will also fill up the gaps if present inside the object. First, dilate so that we can remove speckle noise, any breaks in the boundary and also fills the gaps which might be present inside the object. Since we dilated, now we have to erode to obtain a right boundary.

Medial Axis Transformation:

We have to shrink the object from all the sides at the same time, until we can no longer shrink it. This gives us the skeleton of the object.

Moving Parts of the moving object:

With the help of branch points in a skeleton we can get branches in a skeleton. Each branch corresponds to one part in the object. If we track the movement of branches we can keep track of movements of the object parts.

III.5. Processes I have applied in this research

The programs were developed in Matlab. Experiments were conducted in a room with ample lighting conditions. Subjects were single persons. Tests are conducted by a person making different types of movements. The snap shots of the resulting images have been taken for different types of movements. Test results were analyzed to see whether the moving object has been accurately detected or not. If the moving object has been successfully detected then, it will be further analyzed whether different parts of objects have been accurately identified or not.

III.6. Algorithms, Process modules, and solution methods

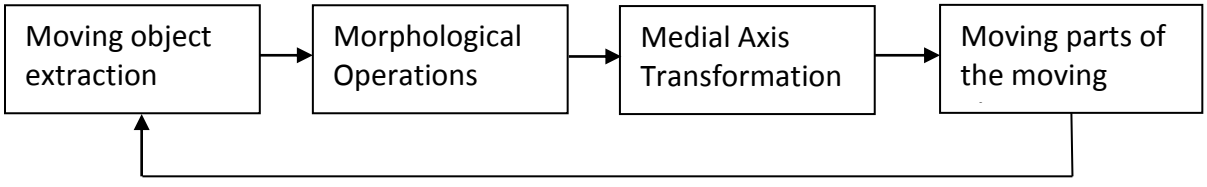


Figure 7: Generalized description of technique

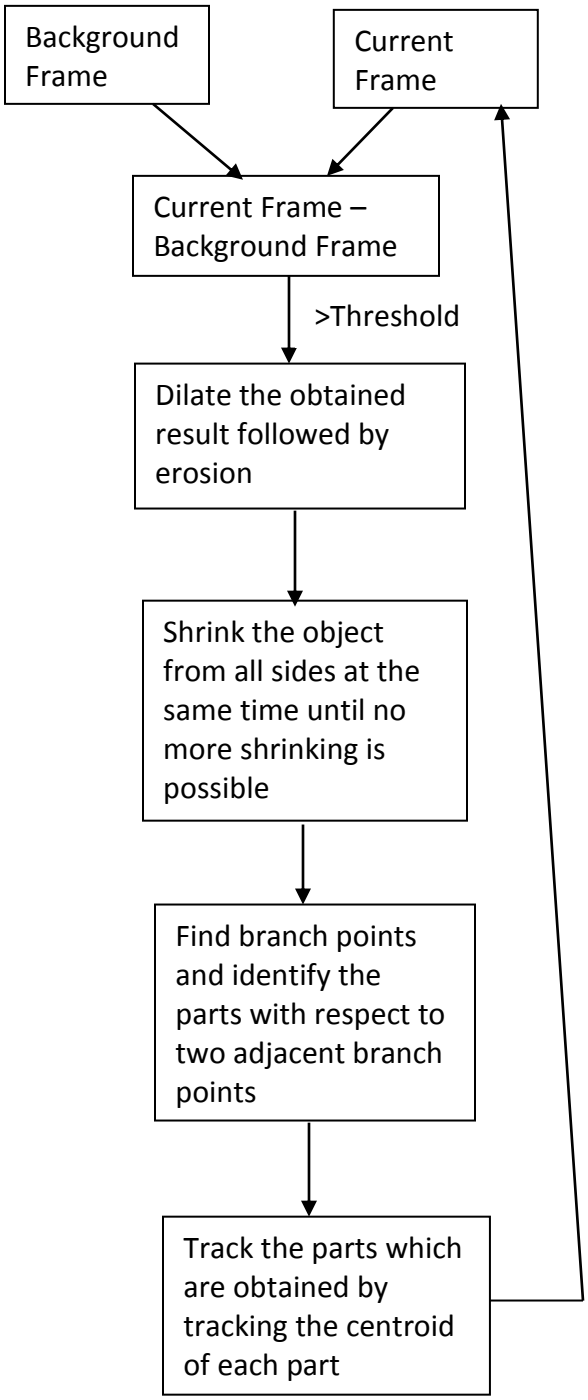


Figure 8: Detailed flow of the technique

Pseudo code:

i) B = background frame (first frame).

ii) C = current frame

iii) I = C – B;

iv) I = I > t (threshold)

v) Dilate and Erode I

vi) Track the objects in I

vii) Skeletonize the objects in I

The following iterations are repeated until the image stops changing. The two subiterations together make up one iteration.

1. In the first subiteration, delete pixel p if and only if the conditions $G_1, G_2,$ and G_3 are all satisfied.
2. In the second subiteration, delete pixel p if and only if the conditions $G_1, G_2,$ and G_3' are all satisfied.

Condition G_1:

$$X_H(p) = 1$$

where

$$X_H(p) = \sum_{1}^{4} b_i$$

$$b_i = \begin{cases} 1, if\ x_{2i-1} = 0\ and\ (\ x_{2i} = 1\ or\ x_{2i+1} = 1) \\ 0, otherwise \end{cases}$$

$x_1, x_2, \dots, x_8$ are the values of the eight neighbors of p, starting east neighbor and numbered in the counter clockwise order.

Condition G_2:

$$2 \leq min\{n_1(p), n_2(p)\} \leq 3$$

where

$$n_1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k}$$

$$n_2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1}$$

Condition G_3:

$$(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0$$

Condition G_3':

$$(x_6 \vee x_7 \vee \bar{x}_4) \wedge x_5 = 0$$

viii) Find the branch points in the skeletons obtained.

ix) Overlap branch points onto the skeleton

x) Display the resultant skeleton with branch points

xi) Repeat steps ii to x

An initial background is obtained. A current frame is obtained in real time. A subtracted frame is obtained by subtracting the background frame which was initially obtained from the current frame. We will then remove the unwanted noise (i.e. unwanted small pixel groups) from the subtracted frame by setting up a threshold. The objects in the resulted subtracted frame are then dilated, so that we can obtain a smoother boundary as well as to fill small gaps within the object. Since we have dilated once, we have to

erode those objects so that we will have an object whose size is similar to the original object. The resulted objects are then tracked.

We will then skeletonize the objects in the resulted frame which is obtained after erosion. The skeletonization is done by eroding all the points on the object boundary at the same time until a thin line is obtained (i.e. we cannot erode the object boundary any more). The two subiterations described in the pseudo code forms a single iteration. The iterations are repeated until the image stops changing [Guo and Hall 1989]. Condition G_1, preserves local connectivity. $X_H(p)$ represents number of times one crosses over from a white point to a black point when points in neighborhood of p are traversed in order. In condition G_2, $n_1(p)$ and $n_2(p)$ each break the ordered set of p's into four pairs of adjoining pixels and count the number of pairs which contain 1 or 2 ones. $min\{n_1(p), n_2(p)\}$ is useful for end point detection and also helps to achieve thinner results. Condition G_3 and G_3', tends to identify pixels at the north and east, south and west boundary of the objects respectively. A branch point is a point at which a pixel has more than two adjacent pixels. When the skeleton of the object is obtained, there will the branch points. Branch are obtained for each object's skeleton. The branch points are then overlapped on the skeleton of each object. The resultant is an object's skeleton with all the branch points, which will be displayed on the screen. Again the new frame is obtained which becomes the current frame and the whole process is repeated. Since this is a real time application the process keeps on repeating until we quit the process. Time Complexity for most of the pseudo code is in $O(n)$ except for the skeletonization

step, which is a fully parallel computational step which has shannon complexity of 80xR [Bernard and Manzanera 1999], where R is the radius of the biggest ball contained in the image.

III.7. Results and Analysis of Results

I have studied and analyzed various conference proceedings and scientific journals on the topics related to this. I have designed a technique based on the available information. I have implemented this technique in Matlab®.

The obtained results are checked for the intended output. If intended output is obtained then it is checked whether it can work in real time or not.

Skeletonized objects are obtained in real time. It was hard to identify the parts of the object corresponding with skeleton. So the tracking of the relative movements of the object parts was not possible. There are techniques like Iterative algorithm to prune skeleton [Bai and Latecki 2007], Skeleton Pruning with Contour Matching [Bai et at. 2007] which can help in identifying object part corresponding to skeleton but those nearly took 15 seconds for computing each object, so those cannot be used in real time.

New technique has to be developed for matching object parts correspondingly to skeleton which can perform real time.

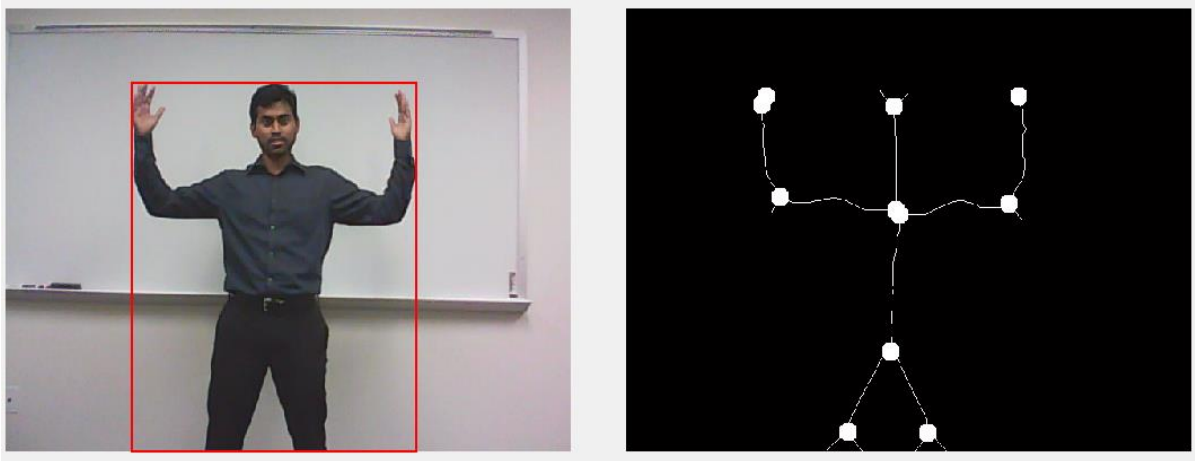


Figure 9: Original object along with its skeleton and branch points

In Figure 9, we can see that skeleton has been created. We can also observe that there are two extra branch points created, one in the middle of the skeleton and one near the wrist of the object.

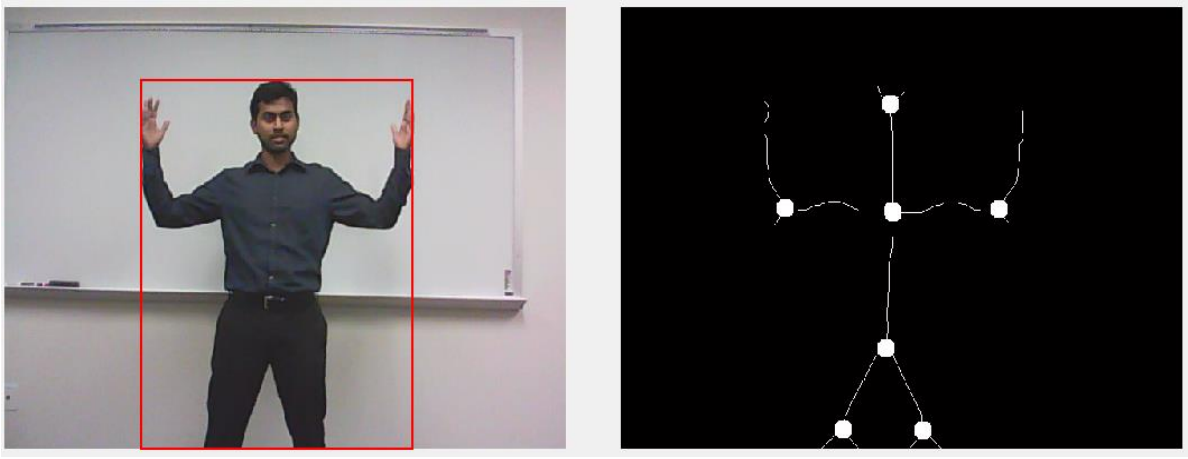


Figure 10: Original object along with its skeleton and branch points

In Figure 10, we can see that skeleton has been created. But we can observe that there are two branch points missing, both at the wrist of the object.

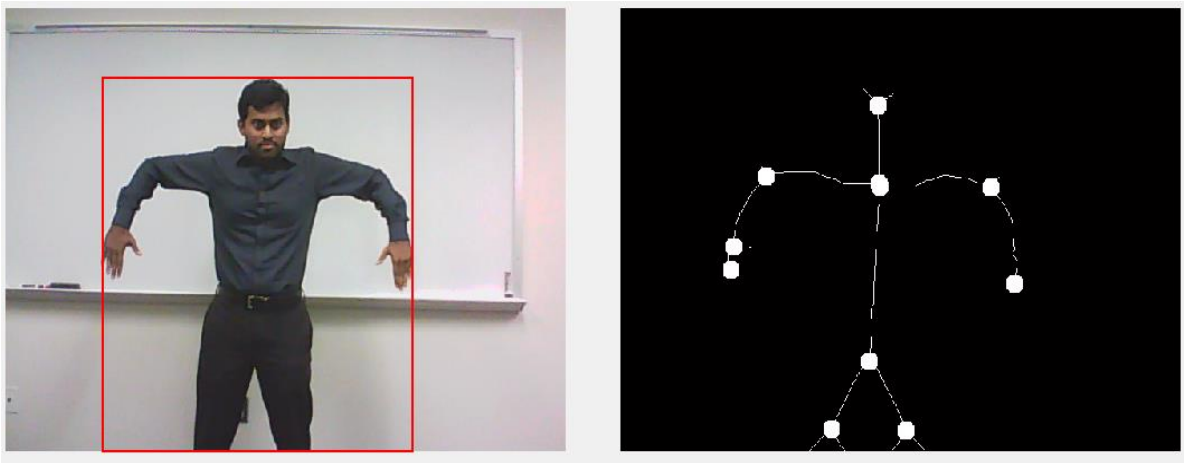


Figure 11: Original object along with its skeleton and branch points

In Figure 11, we can see that skeleton has been created. We can also observe that there are two extra branch points created, one in the middle of the skeleton and one near the wrist of the object.

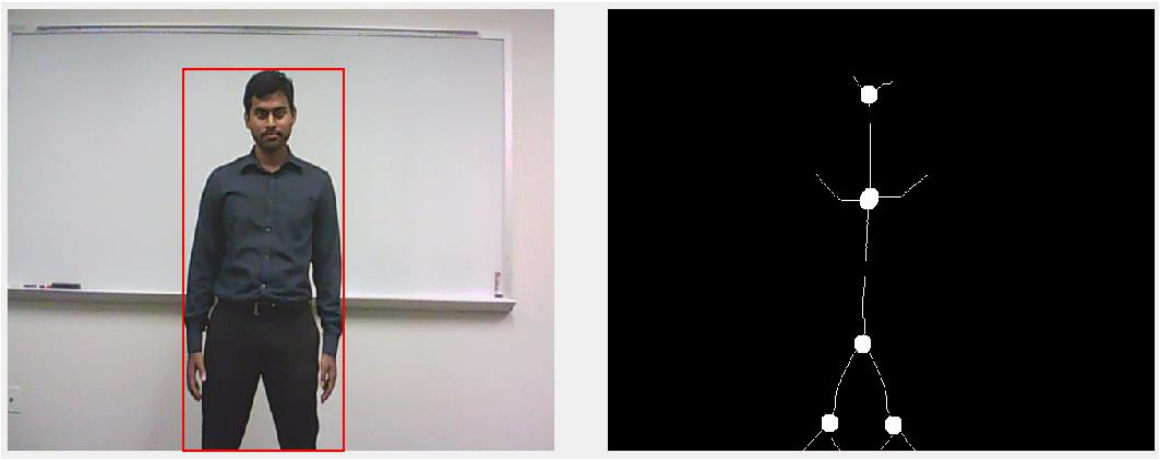


Figure 12: Original object along with its skeleton and branch points

In Figure 12, we can see that skeleton has been created. We can also observe that there is one extra branch points created in the middle of the skeleton. We can also observe that even there is small gap between body and hand, but because of dilating and eroding step in the technique, gap could not be created in the skeleton.

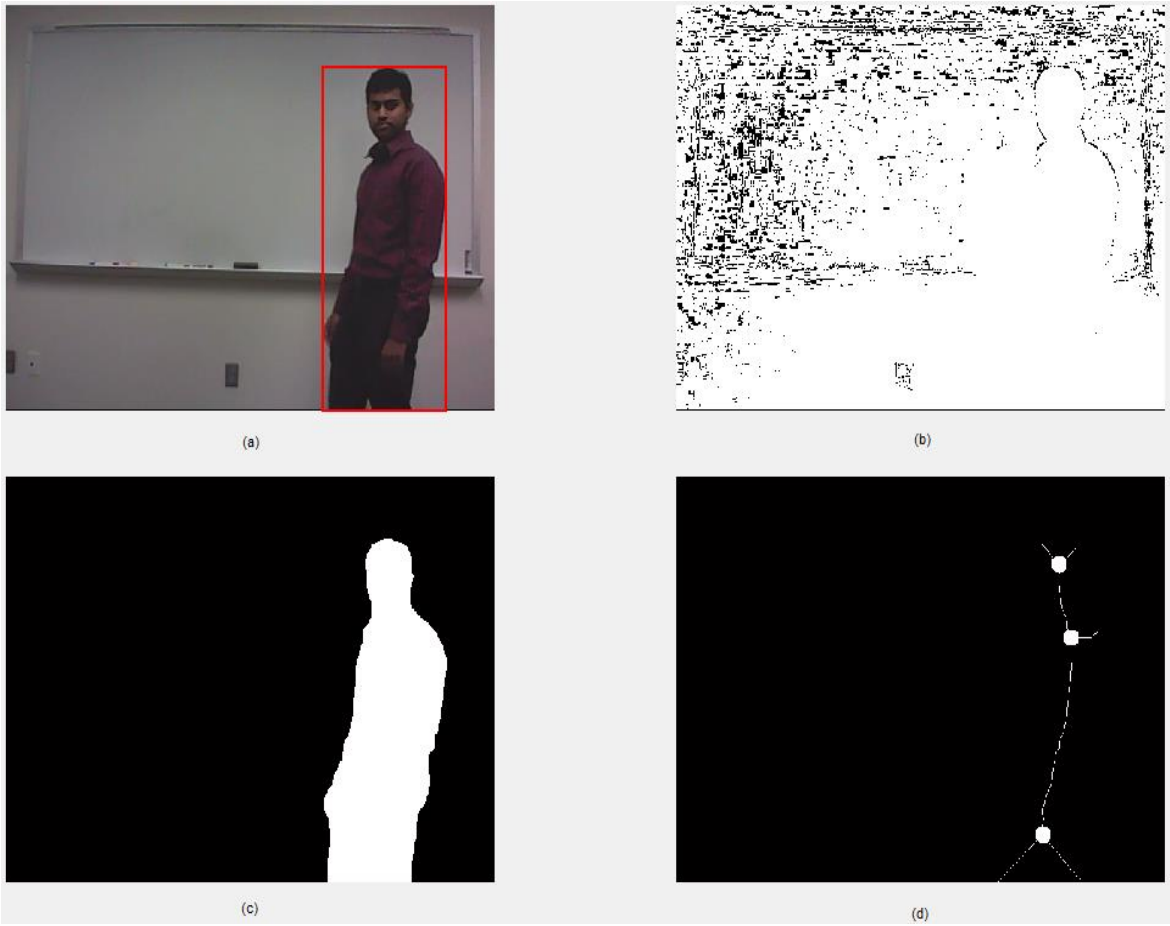


Figure 13: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 13(b) we can see a lot of back ground noise, but all the noise have been deleted by setting up a threshold. In Figure 13(c) we can see a good representation of object.

Figure 14: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 14(b), we can see that there is small amount of noise which might be due to the shadow of the object. In Figure 14(c) we can see that small gap which was there between hand and body in (b) has been lost, due to dilation and erosion step. In figure 14(d) we see that there are three extra branch points, one near the wrist, one in the middle of the object, and other in the lower part of the object

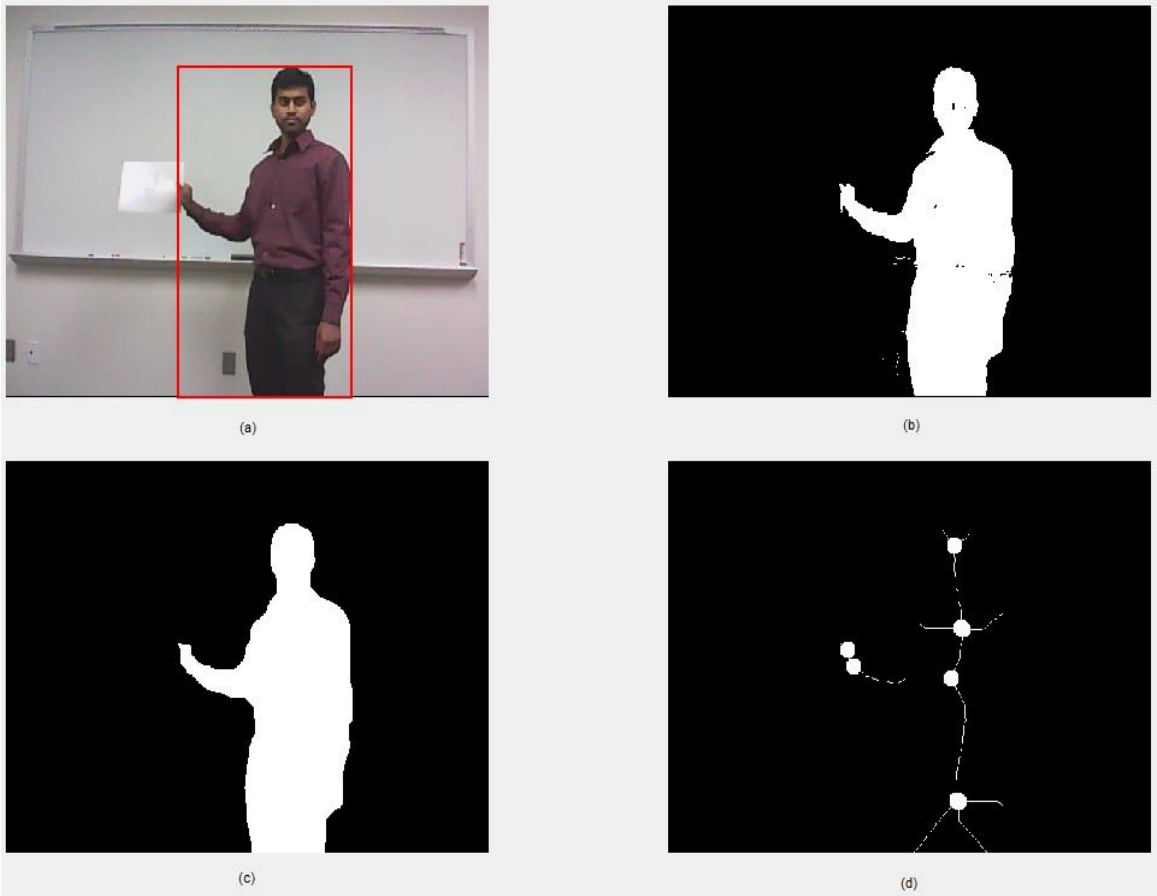


Figure 15: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 15(b), we can observe that the paper in the scene was not recognized, since it almost matches the background. In Figure 15(d) we see two extra branch points, one at the wrist and other in the lower part of the object.

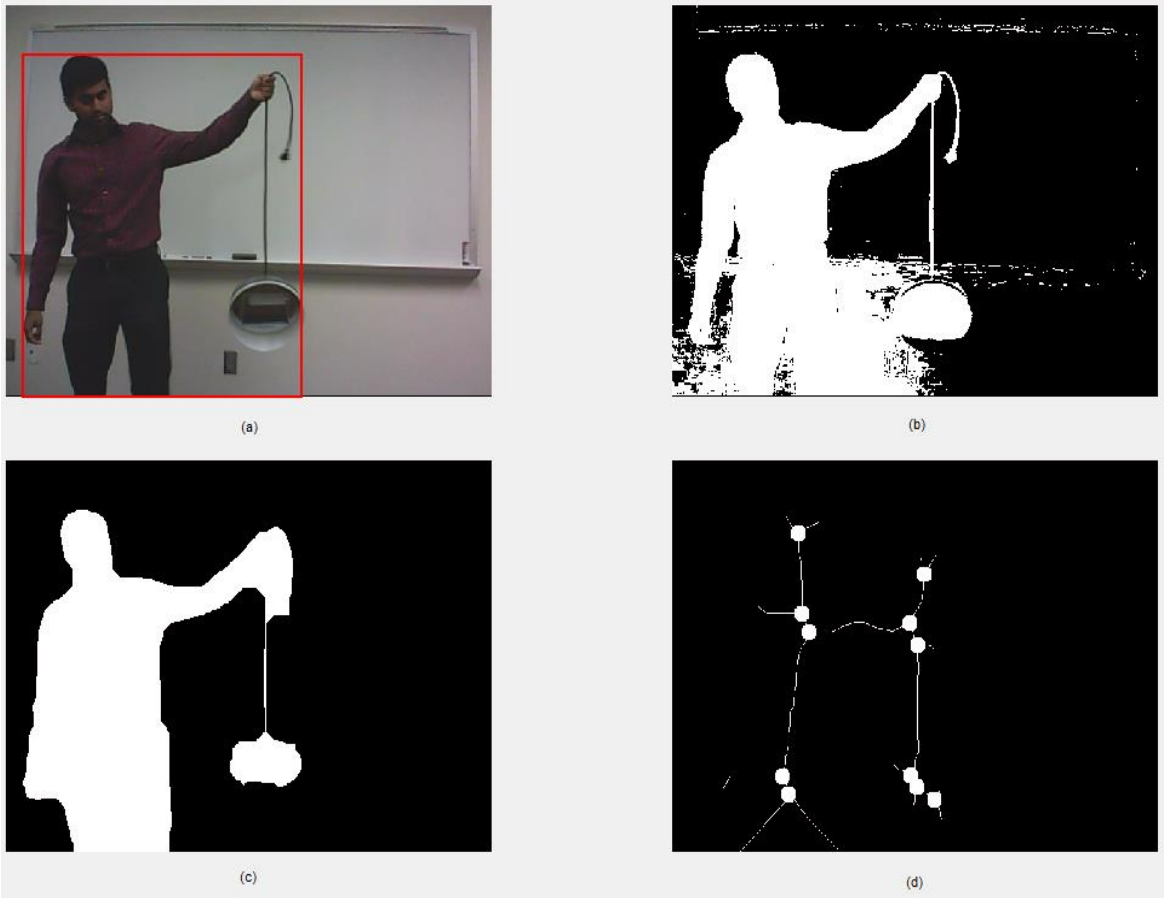


Figure 16: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 16(c) we can see that a closed loop of the wire is formed at the wrist and also the gap between is the hand and body is also filled, this is due to background noise and also due to dilation and erosion step.

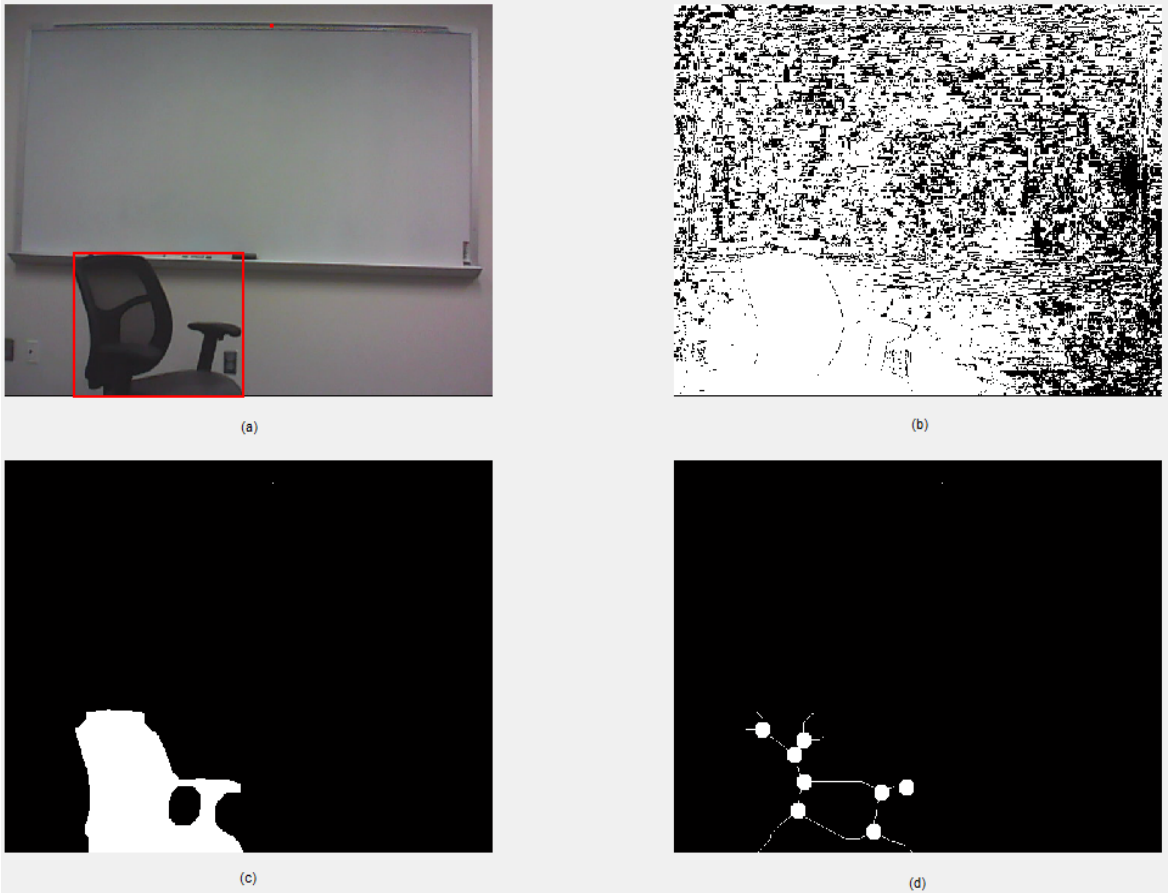


Figure 17: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure17(c), we can see that gap between the arm and back of the chair is connected due to background noise and also dilation and erosion step. In figure 17(d), we can see that some extra branches are created since the object's boundary is not smooth.

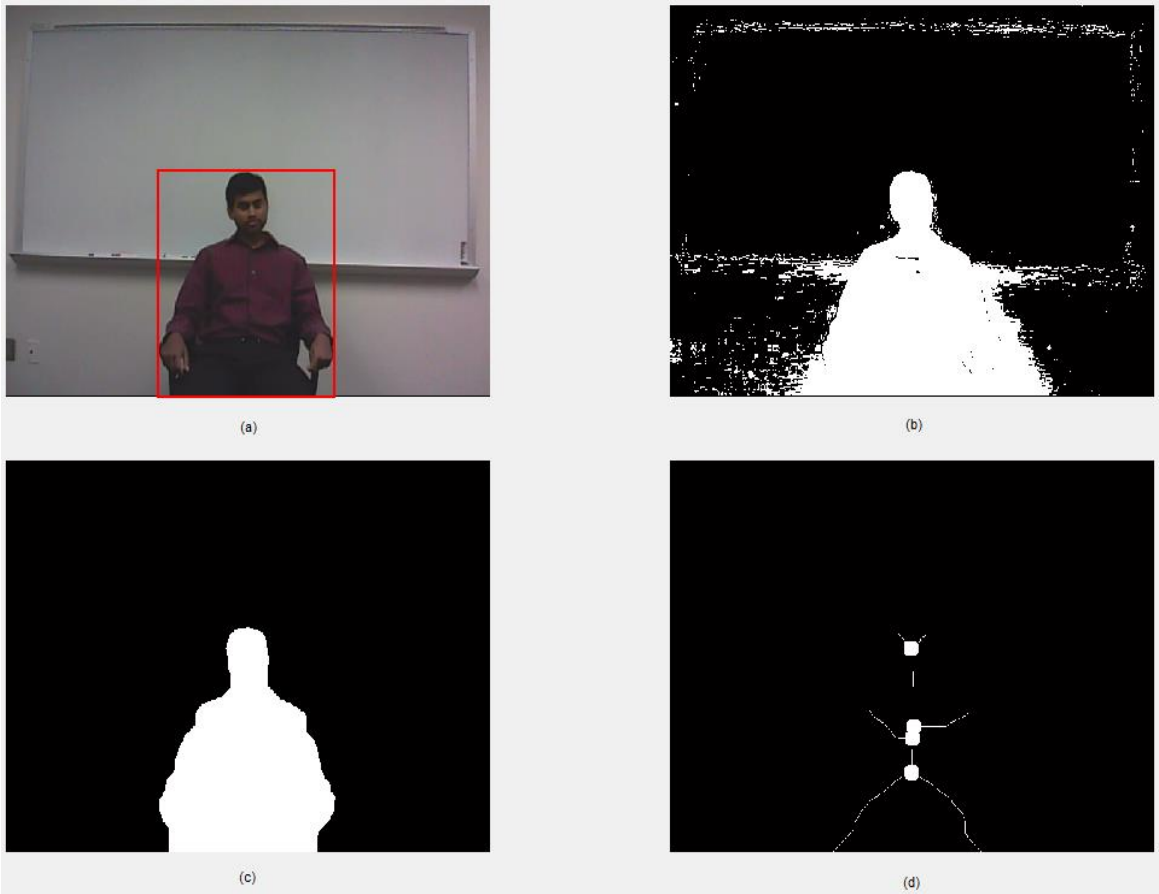


Figure 18: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In figure 18(c), we can see that lot of background noise has been eliminated.

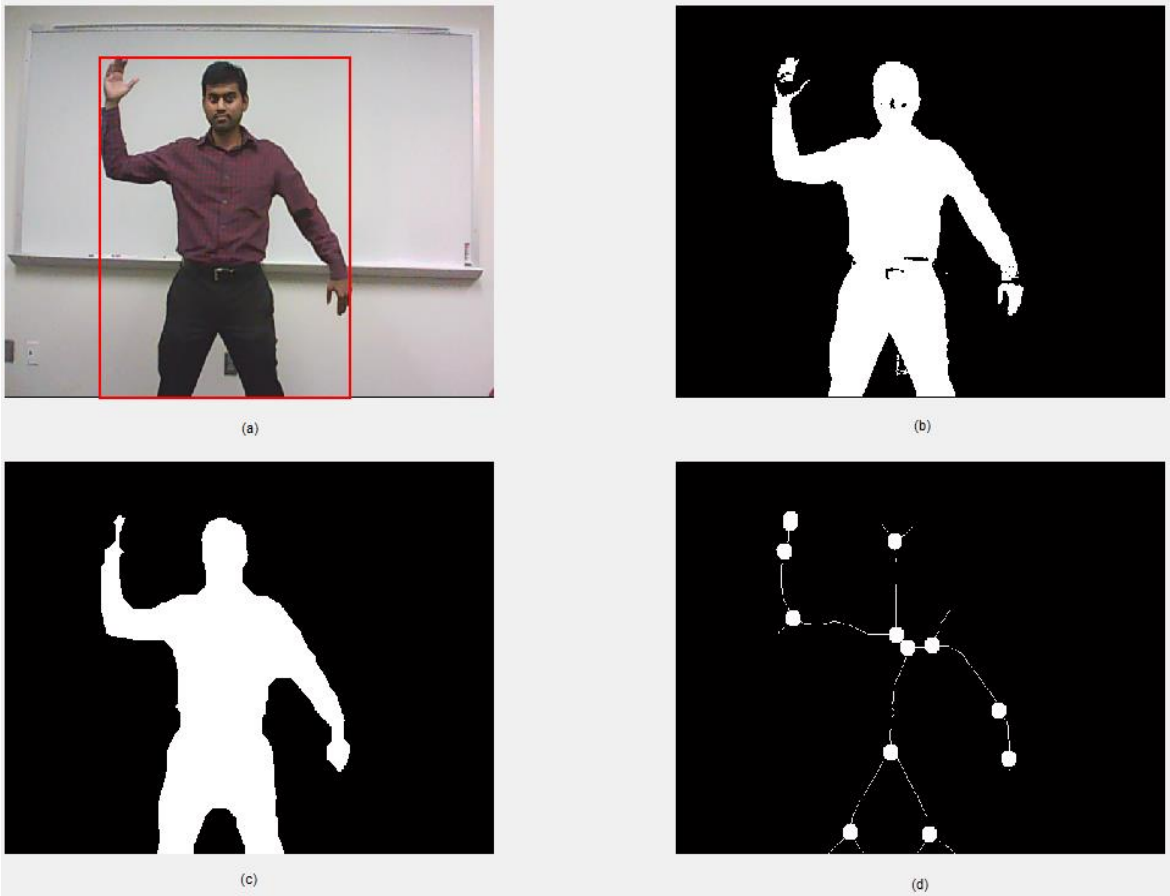


Figure 19: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 19(c), we can see that lot gaps in the object is filled by the dilation and erosion step. In figure 19(d), we can see that couple of extra branch points are created at the wrist of the object.

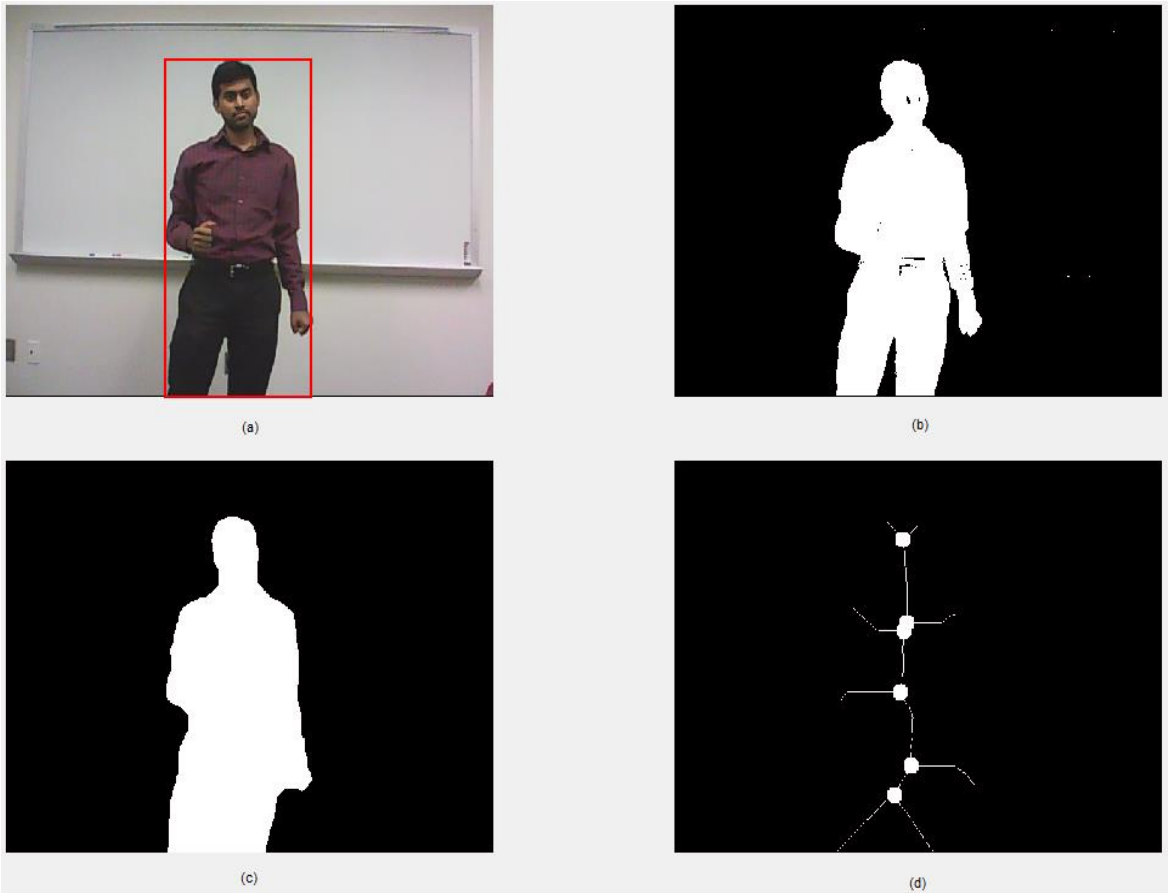


Figure 20: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In Figure 20(b), we can see that we are not able to recognize the hand, since it overlaps with the body of the object. In Figure 20(c), the gap between the other hand and body is filled, due to dilation and erosion step.

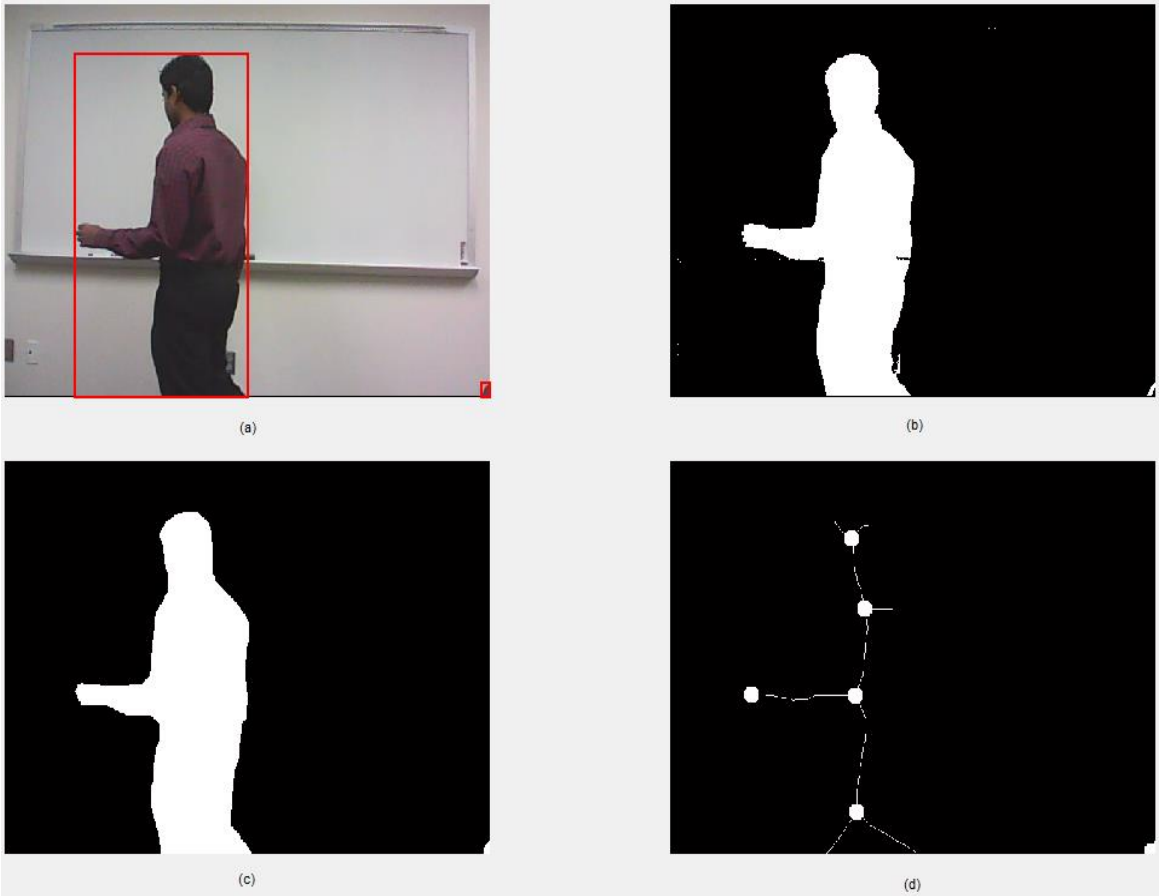


Figure 21: (a) original image with the object being tracked (b) image obtained by foreground detection (c) image obtained after dilation and erosion step (d) skeleton of the object along with branch points

In figure 21(c), we can see that the object created is a good replica of the original object. In Figure 21(d), we can see that there no extra branch points created.

III.8. Facilities and supplies used in this research

A computer equipped with Intel® Core™2 6700 processor clocked @2.66GHz 2.66GHz has been used. The computer was also equipped with 4.00GB RAM. A 64-bit Windows 7 Enterprise operating system was installed. A webcam attached to the computer was used to capture the real time datastream. MATLAB® R2013b was used to develop the technique.

III.9. Program Analysis and Results Evaluations

Most of the available techniques for skeletonizing the object require help of device like accelerometer, or some kind of sensors attached to the object or camera to work in real time. There are some techniques which do not require the aid of sensors but they are not able to perform well in real time [Bai and Latecki 2007]. This technique is able to produce the skeleton of the object regardless of its type in real time. This technique will produce consistent results given the input remains same. But this technique does not produce the same number of branch points for the object which is being tracked in consecutive frames. With this technique, it is very hard to match the object part and with the skeleton when the parts of the object overlap each other or when different objects overlap each other. This technique will not be able to skeletonize the finer details in the object, since there is dilation and erosion step involved which might close the small gaps between different parts of the object.

Skeletonized objects are obtained with this technique. But the branch points obtained from the skeleton are not accurate enough to match them object parts. Resulted skeletons will not be accurate when the object's parts overlap each other or when different objects overlap each other. This technique will produce the same results when the same input is used, i.e. results are consistent as long as input remains the same. Errors in skeletonization of the objects depend on the noise in the image. Since this technique depends on foreground detection, it does not work if the moving object is in camouflage with the background. This technique also does not work well for rotating or spinning objects, as rotation and spinning certainly overlaps the objects parts over each other.

IV. Summary

Very less research was done in tracking relative movements of object parts. This research focused on finding a technique which can track relative movements of objects parts in real time. There are some techniques available to track the movements of object parts but most of the techniques require some kind of learning step. I intended to develop a technique which is computationally less intensive and will work real time. I taught that skeletonizing the object and tracking the skeleton movements could work in real time. But after implementing, I found that it is very hard to obtain a skeleton where we can distinguish the object parts easily with the skeleton parts. With the failed experiment, I realized that either skeletonization technique has to be improved or some kind of learning technique has to be used.

I have studied various conference proceedings and various journals which were related to this topic. These helped me understand and analyze the various types of techniques which are currently being used and previously used for this problem. First the moving object has to be detected. Foreground detection technique has been applied, since there is not a lot of computation involved. Unwanted pixel groups are eliminated by setting up a threshold. Morphological operations are performed on the obtained object to fill the holes within the object and also to reduce the boundary distortions in the object. For this we dilate the object and then erode it. The resultant object is completely filled and has a smoother boundary. The smoother boundary helps in not having

unwanted branches in the skeletonization process. Then we apply the medial axis transformation on the resulted object. For this we start eroding from all sides at the same time until we can't erode any further. The result is the skeleton of the object. It was very hard to obtain same kind of skeleton each time with same number of branch points for the same object, so it was not possible to track the relative movements of the object parts.

V. Contributions and Significances

Even though this technique was not successful in tracking relative movement of the object parts, it was able to skeletonize the object in real time. Real time skeletonization might help some researcher who is trying to create and work on skeletons in real time.

Contributions:

- Track objects and create their skeletons

  This techniques uses foreground detection for tracking the objects, since it computationally less intensive. Then, this technique dilates and erodes the object so the gaps within the obtained object are filled and a smoother boundary can be obtained. But the trade off is that, we won't be able to skeletonize the object finely, i.e. small gaps between the object parts are also filled. We will then skeletonize the object to obtain skeleton of it.

- Real time working technique

  Lot of techniques are available to skeletonize the object in real time, but they require the aid of accelerometers, or some kind of sensors attached to the object or camera. There some techniques which can skeletonize the object without using any kind of sensors, but they are not able to

perform well in real time. This technique is able to create skeletonized objects in real time without the aid of any kind of sensors attached to the object or camera.

XI. Directions for Future Research

Skeletonization is the most important step in this technique. Further research has to be done on how we can obtain same kind of skeleton each time. For this, either available skeleton pruning methods have to be improved or a learning step has to be introduced in the technique. I have tried improving some skeleton pruning methods which are available, but I was not quite successful with them to work them out in real time. I did not try to use a learning step, so I cannot say whether it might be successful in real time or not.

The main difficulty which might be encountered in the further research is the unwanted skeleton branches. This technique will not be able to detect the object which is in camouflage with the background. And this technique cannot efficiently track a spinning or rotating object.

References {*or* Bibliography}

François Meyer, Patrick Bouthemy, "Region-based tracking in an image sequence," *Second European Conference on Computer Vision Santa Margherita Ligure*, Volume 588, May 19–22, 1992, pp. 476-484.

Chris Harris, Carl Stennet, "RAPiD -- A video-rate object tracker," *British Machine Vision Conference*, 1990, pp. 73-77.

D. Terzopoulos and R. Szeliski, "Tracking with kalman snakes," in Active Vision, MIT Press, 1992, pp. 3–20

N. T. Binh, A. Khare,"Object Tracking of Video Sequences in Curvelet Domain,"*International Journal of Image and Graphics,* Volume 11, No. 1, 2011, pp. 1-20.

Ming-Kuei Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol.8, no.2, 1962, pp.179-187

M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *Journal of the Society for Industrial and Applied Mathematics*, Volume 29, Issue 1, 1998 , pp. 5-28.

Weiming Hu; Xi Li; Wenhan Luo; Xiaoqin Zhang; Maybank, S.; Zhongfei Zhang, "Single and Multiple Object Tracking Using Log-Euclidean Riemannian Subspace and Block-Division Appearance Model," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.34, no.12, Dec. 2012, pp. 2420-2440

Weihua Chen; Lijun Cao; Junge Zhang; Kaiqi Huang, "An Adaptive Combination of Multiple Features for Robust Tracking in Real Scene," *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on* , Dec. 2013, pp. 129-136

Tian, Gao, and Wang Qi. "Target Recognition Algorithm Based on BP Networks and Invariant Moments." *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, Issue 2, 2013, pp. 969-974

Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi, "Tracking deforming objects using particle filtering for geometric active contours," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 29, 2007, pp. 1470–1475.

A. Doucet, J. de Freitas, K. Murphy, and S. Russel, “Rao-Blackwellized particle filtering for dynamic Bayesian networks,” in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence,* 2000, pp. 176–183

Peter Kardos, Gabor Nemeth, Kalman Palagyi, “An order Independent Sequential Thinning Algorithm,” in *Proceedings 13th International Workshop, IWCIA 2009*, 2009, pp. 162–175

V. Namrata, R. Yogesh, Y. Anthony, and T. Allen, “Deform pf-mt: particle filter with mode tracker for tracking nonaffine contour deformations,” *IEEE Trans. Image Processing*, vol. 19, no. 4, 2010, pp. 841–857.

J.-L. Starck, E. J. Candes and D. L. Donoho, “The curvelet transform for image denoising,” *IEEE Transactions on Image Processing*, Volume 11, Issue 6, 2002, pp. 670-684.

Tsagkatakis, Savakis, “Online Distance Metric Learning for Object Tracking,” *IEEE Transactions on Circuits and Systems for Video Technology*, Volume 21, No.12, 2011, pp. 1810-1821.

Wang, Aiping, Zhi-Quan Cheng, Ralph R. Martin, and Sikun Li. "Multiple-Cue-Based visual object contour tracking with incremental learning." In *Transactions on Edutainment IX*, 2013, pp. 225-243

Chirag. I. Patel and Ripal Patel, "Contour Based Object Tracking," *International Journal of Computer and Electrical Engineering* vol. 4, no. 4, 2012, pp. 525-528

J. Li, J. Wang, “Adaptive object tracking algorithm based on Eigen basis space and compressive sampling,” *IET Image Process.*, Volume 6, No. 8, 2012, pp. 1170-1180.

Z. Wang, K. Hong, “A New Method for Adaptive Object Tracking Based on Kalman Filter and Bayesian Law,” *Journal of Information and Computational Science*, Volume 9, No. 18, 2012, pp. 5953- 5960.

N. Mann, P. Singh, “Medial Axis Transformation based Skeletonzation of Image Patterns using Image Processing Techniques,” *International Journal of Science and Research*, Volume 1, No. 3, Dec. 2012, pp. 220-223

P. Hall, D. Marshall, and R. Martin, “Incremental eigenanalysis for classification,” *In Proceedings of British Machine Vision Conference,* 1998, pp. 286–295.

J.A. Dowley, K. Dogancay, and R.S.A. Brinkworth, “A novel detection approach using bio-inspired vision for enhanced object tracking in video,” *IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing,* 2013, pp. 497–502.

Lee, D. T, “Medial Axis Transformation of a Planar Shape,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1982, pp. 363 – 369

Xiang Bai, Longin Jan Latecki, and Wen-Yu Liu, “Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2007, pp. 449 – 462

Xiang Bai and Longin Jan Latecki, “Discrete Skeleton Evolution,” *EMMCVPR,* 2007

Hyung-Bok Kim and Kwee-Bo Sim, “A particular object tracking in an environment of multiple moving objects,” *International Conference on Control, Automation and Systems,* 2010, pp. 1053–1056.

D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, “A real-time computer vision system for measure traffic parameters,” *Proc. IEEE Conference on Computer Vision and Pattern Recognition,*1997, pp. 496 -501

V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision,* vol. 1, no. 1, Oct. 2005, pp. 1-89

Kai Zhou; Rui-Xia Fan; Wei-Xing Li, "A MeanShift-Particle Fusion tracking algorithm based on SIFT," *Control Conference (CCC), 2010 29th Chinese*, July 2010, pp. 2717 - 2720

Chun-Te Chu; Jenq-Neng Hwang; Hung-I Pai; Kung-Ming Lan, "Robust video object tracking based on multiple kernels with projected gradients," *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on* ., May 2011, pp. 1421 - 1424

Kuan-Hui Lee; Yong-Jin Lee; Jenq-Neng Hwang, "Multiple-kernel based vehicle tracking using 3-D deformable model and license plate self-similarity," *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* , May 2013, pp. 1793 - 1797

Bin Zheng; Xiangyang Xu; Yaping Dai; Yuanyuan Lu, "Object Tracking Algorithm Based on Combination of Dynamic Template Matching and Kalman Filter," *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on* , vol.2, no., Aug. 2012, pp. 136 - 139

Qing Wang; Feng Chen; Wenli Xu; Ming-Hsuan Yang, "Object Tracking via Partial Least Squares Analysis," *Image Processing, IEEE Transactions on* , vol.21, no.10, Oct. 2012, pp.4454 – 4465

Qing Wang; Feng Chen; Wenli Xu, "Tracking by Third-Order Tensor Representation," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* , vol.41, no.2, April 2011, pp. 385 - 396

Zicheng Guo and Richard W. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol.32, no.3, March 1989, pp. 359 – 373

Thierry M. Bernard and Antoine Manzanera, "Improved low complexity fully parallel thinning algorithm," *Image Analysis and Processing, 1999. Proceedings. International Conference on* , vol., no., Sept. 1999, pp. 215 -220

Sema Candemir, Kannappan Palaniappan, Filiz Bunyak, Raghuveer M. Rao and Guna Seetharaman, "Feature prominence-based weighting scheme for video tracking," *Eighth Indian Conference on Computer Vision, Graphics and Image Processing,* 2012, Article 25.

G. H. Golub and C. F. Van Loan, *Matrix Computations,* The Johns Hopkins University Press, ISBN 0 8018 5414 8, 1996

G. Bishop and G. Welch, "An Introduction to the Kalman Filter," in SIGGRAPH 2001, Course 8, 2001

Anonymous, "Image Moment," http:// http://en.wikipedia.org/wiki/Image_moment (as of October 9, 2014)

Liu Yang, Rong Jin, "DisLearnKit," http://www.cs.cmu.edu/~liuy/distlearn.htm (as of October 9, 2014)

D. Ross, J. Lim, R.S. Lin, M.H. Yang," Incremental Learning for Robust Visual Tracking," http://www.cs.toronto.edu/~dross/ivt/ (as of October 9, 2014).