M.Tech. (Computer Science) Dissertation Series.

Region Based Object Tracking Algorithm Using Intensity Surface Information.

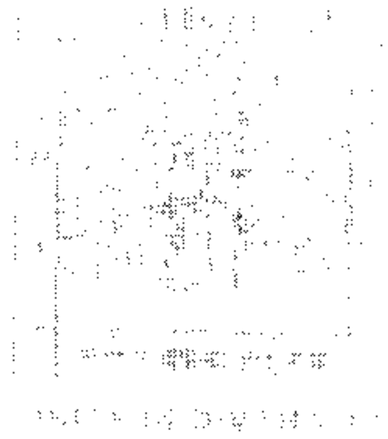
A dissertation submitted in partial fulfillment of the requirements for the Master of Technology (Computer Science) degree of the Indian Statistical Institute.

By

Tushar Koley

Under the supervision of

Dr. Dipti Prasad Mukherjee



INDIAN STATISTICAL INSTITUTE

203, Barrackpore Trunk Road

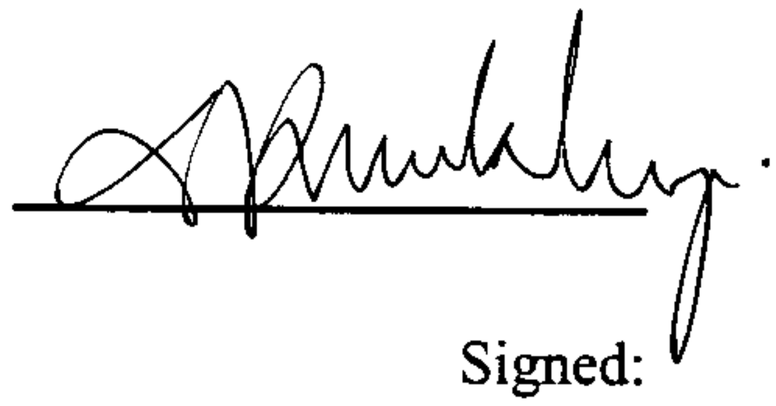
Kolkata-700108

2003

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled '**Region Based Object Tracking Algorithm Using Intensity Surface Information**' is an authentic record of dissertation carried by **Tushar Koley** under my supervision and guidance. The work fulfils the requirements for the award of the M.Tech. degree in Computer Science.

Dated: The 1st of July, 2003.



Signed:

(Dr. Dipti Prasad Mukherjee)

Supervisor

ACKNOWLEDGEMENT

I take this opportunity to express my deep sense of gratitude and indebtedness to my guide **Dr. Dipti Prasad Mukherjee** of Electronics and Communication Science Unit of Indian Statistical Institute, Kolkata for his generous and whole hearted support to me in completing this project.

I wish to express my sincere respect and thanks to **Mr. Partha Mohanty** of electronics and Communication Science Unit for his kind help and cooperation that he extended to me throughout the tenure of my project.

I also wish to place on record my sincere thanks to **Mr. Arindam Ghosh**, Research Scholar of electronics and Communication Science Unit for his huge help and cooperation in making my project thesis colorful and attractive.

Lastly, I would like to express my profound gratitude to all other teachers of Indian Statistical Institute, Kolkata and sincere thanks to all of my classmates who were the source of inspiration during the M.Tech programme.

Tushar Koley

Contents

Region Based Object Tracking Algorithm Using Intensity Surface Information

Tushar koley

Abstract

We propose an algorithm for object detection and tracking in video image sequence. Our algorithm starts from a single point, considered as the center of the object for tracking. The user in the first frame initializes this point. The basic principle of the algorithm depends on surface approximation and its normal. The surface is approximated on the image data representing a sub-section of an image and this sub-section encloses an object that is to be identified for tracking. Least Square Surface fitting technique is used for the surface approximation. We introduce three external forces from the surface normal that can stretch an active contour from the initialized center of an object towards its boundary. In our algorithm, calculation of spatial movements of the object is governed by the principle of error minimization. The error term is computed from the change of the surface and its normal in advance of time. The algorithm is tested on both synthetic and real image sequences.

1. Introduction

"Object Tracking" is one of the most popular topics in recent years. The objective of this application is to detect and track a moving object in a video sequence (of frames). User initializes the center of the object in the first frame. In order to track an object, the first task is to detect the boundary of the object in the first frame. Next task is to track the object through the frame sequences. In order to track, detected boundary in the first frame is shifted according to the object movement. Then the boundary is morphed as per the deformation from the previously detected shape.

In a video sequence, detection of a moving object is usually done by aligning pairs of images and then creating the difference image. Moving objects can be found in the difference image using algorithms such as clustering [1]. A problem in the clustering technique is that usually alignment is not perfect and the difference image is noisy. Another problem in image alignment and subtraction is the execution time, since every image points is examined for registration, alignment and subtraction followed by clustering. In our algorithm we do not need to create difference image; moreover, we do not need to scan the entire image, only scanning the region where the object is present, is sufficient. We also reduce noise present into the scanning region of the image by approximating a surface to image data points. The detail of surface approximation is explained in the section 2.1.

There are also algorithms such as 2-D contour based object tracking [2]. Active contours are curves defined within an image domain that can move under the influence of internal forces coming from within the curve itself and external forces computed from the image data [3]. One limitation in contour based algorithm is that information that is very close the object boundary is used for tracking and region information present within the object is neglected. The gray level information of pixels within the object boundary and their brightness change pattern may sometimes provide important information to help developing a tracking algorithm. In our tracking algorithm, all such informations are utilized as the algorithm uses a surface equation that closely approximates pixel values within the object boundary.

There are several approaches where active contour methods are used for object boundary detection in a still image. One such approach is Gradient Vector Flow [3]. As mentioned earlier that active contour changes its shape under the influence of internal and external forces. Internal forces are dependent on the tension and the rigidity coefficients of the contour, whereas external forces are implementation dependent. Image gradient can be used as an external force. Image gradient vector has the property that it is larger near the boundary and it is normal to the boundary at the boundary points. But one problem of gradient vector is that it is high only near the boundary (also near noise) but it falls off drastically as we move away from the boundary. So in order to detect actual boundary user must initialize it very close to actual boundary. Several types of dynamic external forces are invented to improve this problem. For example pressure force used in balloon [2] is a dynamic external force, which can pull the contour from any arbitrary position to

the actual boundary. We use three new external forces for boundary detection of our target object. These three forces are similar to the pressure force used in balloon [2]. These three forces are computed from some interesting properties of surface normals. The details of this computation are explained in the section 2.2.

In our application we first fit a surface to the image data, into the region of the object, by the use of least square surface fitting technique. The main motivation of surface fitting is to gather all the information contained within the object boundary by a single surface equation. For example if we want to find the pixel value at any position within the object boundary we can get its approximate value directly from the surface equation, so we do not need to store entire data once we have approximated a surface. Another advantage of surface approximation is that it automatically smoothen the image and, therefore, reduces noise. Of course the procedure may blur edges to some extent but information lost due blurring can easily be recovered from the change of surface normals near the object boundary.

For object boundary detection we have adopted a method as suggested by [4]. This is a generalized method and can be used for several applications. Applying this method boundary can be defined as a continuous function of a parametric variable. Boundary is first subdivided into finite number of boundary elements. Each element has two end points, considered as two nodes, which has four degrees of freedom corresponding to positions and tangents of those end points. The boundary changes its shape under the influence of dynamic force governed by Lagrangian motion equation. As already mentioned three new dynamic external forces are computed from the surface normal.

Assuming object movements between two consecutive frames are marginal, so in order to search a point from these two consecutive frames, we have assumed that the hight and normal to that point should be equal. This assumption is stronger than the optical flow [5] because, the algorithm assumes intensity value (equivalent to gray value) as well as normal at a point is invariant between two consecutive frames in order to search that point in advance of time whereas optical flow assumes only intensity (gray) value to be equal. We also assume rigid body motion, so, the deformation of the object between two consecutive frames is neglected. With he help of this second assumption, we can express movement of any point of the object by the movement of its center and the rotation of the object with respect to its center. So, it is sufficient to find the velocity of the center and the angular velocity of the object with respect to its center, in order to find the velocity of each individual points of the object. The computation of velocity of the center of the object and its angular velocity is governed by the principle of error minimization. The detail of the expressions required for this technique is given later.

In the next section we discuss methodology of the surface approximation, object boundary detection and object tracking in detail. Section 3 discusses experimental results followed by conclusion in section 4.

2. Methodology

In this section we discuss the detail of the methodology used in our tracking algorithm. First we discuss surface approximation technique (2.1), and then we discuss boundary detection of the target object (2.2) and finally actual object tracking method (2.3).

2.1 Surface approximation: We discuss here surface approximation technique within a rectangular region of the image data points for simplicity. But it is also possible to approximate a surface of any arbitrary shaped region when the boundary is defined properly. As already told that user initializes a point in the first frame that is considered as the object center. The data value required for the surface approximation contains pixel values within the rectangle containing the object. Now let I be the image matrix containing data. Suppose $z = I(x, y)$ be a data value on the matrix at column and row position x and y respectively. We may consider this as a point $p(x, y, z)$ where $z = I(x, y)$. We want to generate a surface, which approximates all such points $p(x, y, z)$. For surface approximation we use least square surface fitting technique.

The least square method is used to fit a curve or surface to the data point minimizing mean square error. To fit a θ th order polynomial surface, the approximated surface is given by

$$f(x,y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3 + \cdots + a_{0\theta}y^{\theta} = \sum_{k=0}^{\theta}\sum_{l=0}^{k} a_{k-l,l}x^{k-l}y^{l} \quad , \tag{1}$$

where k, l are indices of coefficients and $0 \le k \le \theta, 0 \le l \le k$. Error term of the approximated surface to that data points $p(x, y, z)$ is given by

$$\mathrm{E} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\Big(\sum_{k=0}^{\theta}\sum_{l=0}^{k} a_{k-l,l}x_i^{k-l}y_j^{l} - z_{ij}\Big)^2 , \tag{2}$$

where $m \times n$ is the total number of data points within the enclosed region. To estimate coefficients $a_{k-l,l}$ setting $\partial E / \partial a_{k-l,l} = 0$, we get

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\Big(\sum_{k=0}^{\theta}\sum_{l=0}^{k} a_{k-l,l}x_i^{k-l}y_j^{l}\Big)x_i^{k-l}y_j^{l} = z_{ij}x_j^{k-l}y_j^{l} , \tag{3}$$

Differentiating with respect to all the unknown coefficients and setting them equal to zero we have total $1+2+3+\ldots+(\theta+1) = (\theta+1)(\theta+2)/2$ equations and same no of

unknowns. We opt for Crammer's rule for matrix solutions [6] of the above system of linear equations. The matrix equation can be given by $\mathbf{B}.\mathbf{A} = \mathbf{C} \Rightarrow \mathbf{A} = \mathbf{B}^{-1}\mathbf{C}$ where $\mathbf{A}$ is a column vector of length $(\theta+1)(\theta+2)/2$ comprising coefficients, $\mathbf{B}$ is a square nonsingular matrix of size $(\theta+1)(\theta+2)/2 \times (\theta+1)(\theta+2)/2$ and $\mathbf{C}$ is a column vector of length $(\theta+1)(\theta+2)/2$ comprising data values. $\mathbf{A}.\mathbf{B}$ and $\mathbf{C}$ are given by

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a_{00} & a_{10} & \cdots & a_{k-l\,l} & \cdots & a_{0\theta} \end{bmatrix} \tag{4}$$

$$\mathbf{B} = \begin{bmatrix} \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} 1 & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_i & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x^{k-l} y^{l} & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} y^{\theta} \\ \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_i & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_i x_i & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (x^{k-l} y^{l}) x_i & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} y^{\theta} x \\ \vdots & \vdots & & \vdots & & \vdots \\ \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x^{k-l} y^{l} & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (x^{k-l} y^{l}) x & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (x_i^{k-l} y_j^{l}) x_i^{k-l} y_j^{l} & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_i^{k-l} y^{l} y^{\theta} \\ \vdots & \vdots & & \vdots & & \vdots \\ \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} y^{\theta} & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (x^{k-l} y^{l}) x & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_i^{k-l} y^{l} y^{\theta} & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} y^{\theta} y^{\theta} \end{bmatrix} \tag{5}$$

$$\mathbf{C}^{\mathrm{T}} = \begin{bmatrix} \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} z_{ij} & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} z_{ij} x_i & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} z_{ij} x_i^{k-l} y_j^{l} & \cdots & \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} z_{ij} y^{\theta} \end{bmatrix} \tag{6}$$

Since the matrix $\mathbf{B}$ does not contain image data, it is fixed in advance of time. Only column vector $\mathbf{C}$ need to be recomputed in order to find new coefficients ($\mathbf{A}$) for surface approximation. It can be shown that matrix equation $\mathbf{B}.\mathbf{A} = \mathbf{C} \Rightarrow \mathbf{A} = \mathbf{B}^{-1}\mathbf{C}$ provides unique solution provided each x_i's and y_j's are unique. However, when x_i's and y_j's are equally spaced, then matrix $\mathbf{B}$ is a Hilbert matrix which is ill conditioned. The degree of approximating surface is limited in most cases by the round of error [7]. Since x_i's and y_j's are equally spaced for our application, matrix $\mathbf{B}$ is a Hilbert matrix. Gauss Elimination method is used here to solve this matrix equation [7].

Some examples of surface approximations are shown in the *Figure 1*.

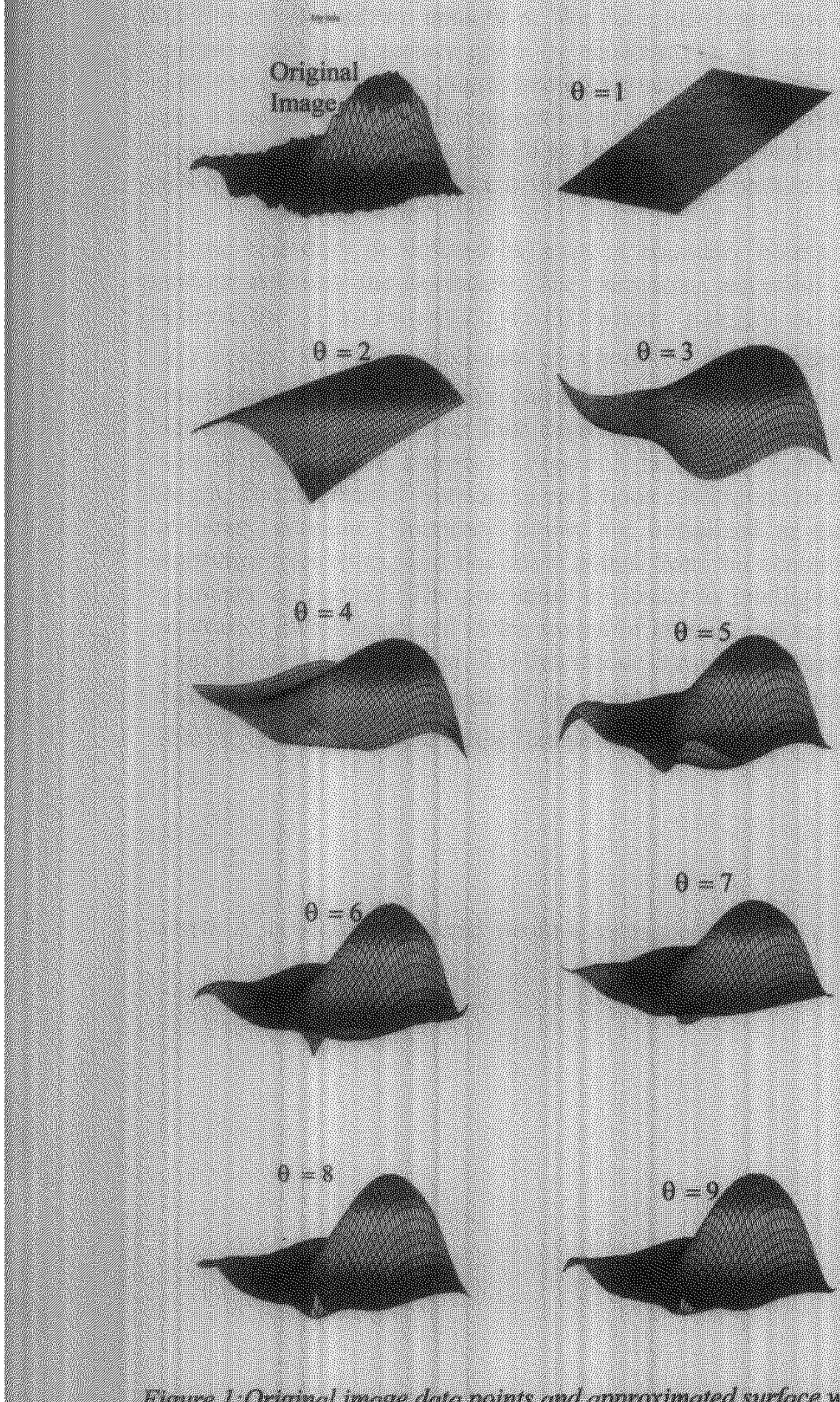


Figure 1:Original image data points and approximated surface with various orders (θ).

After approximating a surface within a rectangle enclosing the object we can go for its boundary detection. The next section (2.2) describes the boundary detection in detail.

2.2 Object Boundary detection: We have just seen how an image data points can be approximated by a surface given by equation (1). So given a data matrix we can easily have the corresponding approximated surface. Now, suppose this data matrix contains gray values of a rectangular region of an image and this rectangle contains an object that we want to identify. So we should go for boundary detection of the said object. Let us see how our approximated surface helps us to detect boundary of the object.

Let the boundary be defined by a set of boundary points. We want to create a closed contour defining the boundary. The parametric equation of boundary contour can be defined by $X(s,t)=(x(s,t),y(s,t))$ where $s\in[0,1]$ is the parametric variable and $(x(s,t),y(s,t))$ is the coordinate of a point on the boundary. We consider each pair of consecutive boundary points constitutes a finite element of the contour. The finite element of the contour has four degrees of freedom between two nodes located at the ends of the segment shown in *Figure 2*. In *Figure 2*, h represents parametric element length. The degrees of freedom at each node correspond to the position and tangent of the boundary element. Boundary points are initialized at the center of the object. Our algorithm tries to expand boundary points from the center of the object until the actual boundary is reached. The procedure is similar to intelligent balloon [8] where user can arbitrarily initialize a seed in the region of interest and the region grows until actual 3-d object boundary is reached. We first divide continuous domain of s into m element sub-domain and then approximate X as weighted sum of continuous basis functions N_i [4]. This N_i's are also called shape functions. Thus

$$X \cong \sum_{i=0}^{n} X_i N_i \,. \tag{7}$$

where X_i is vector of a nodal variable associated with node i.

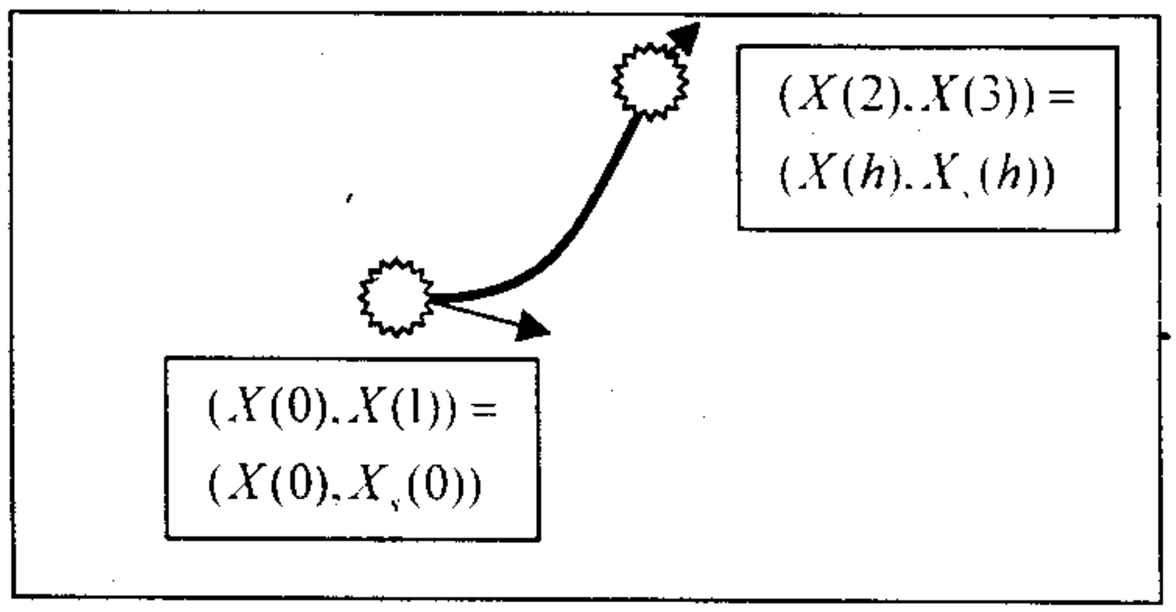


Figure 2: 2 nodes 4 degree of freedom boundary element where h is the parametric element length.

The shape functions N_i's are fixed in advance whereas nodal variables X_i's are unknown and need to be solved. At any particular time instant X_i's posses some discrete values and it is updated in advance of time. Since N_i are continuous functions of s. weighted sum given by equation (7) becomes continuous function of s. For the present application basis functions are derived from Hermite polynomial.

Each segment of boundary can be approximated as weighted sum of set of Hermite polynomial [6] given by

$$X \cong \sum_{i=0}^{3} X_i N_i , \tag{8}$$

$X_i, i = 0,1,2,3$ for an element is shown in the *Figure 2* and N_i s are given by

$$N_0 = 1 - 3(s/h)^2 + 2(s/h)^3 ,$$

$$N_1 = h(s/h - 2(s/h)^2 + (s/h)^3) ,$$

$$N_2 = 3(s/h)^2 - 2(s/h)^3 ,$$

$$N_3 = h(-(s/h)^2 + (s/h)^3) . \tag{9}$$

Now let $\tilde{X} = [X_0, X_1, X_2, X_3]$. Then value of $\tilde{X}$ at any particular time satisfies Lagrangian motion equation given by [4]

$$M\frac{d^2\tilde{X}}{dt^2} + C\frac{dX}{dt} + KX = F_{int}^{N} , \tag{10}$$

where M is the mass matrix, C is the damping matrix. K is the stiffness matrix and F_{tot}^{N} is the forcing matrix. The matrices are computed as

$$M = \int_0^1 \mu V^T N ds . \tag{11}$$

$$C = \int_0^1 \gamma N^T N ds . \tag{12}$$

$$K = \int_0^1 (N_s^T w_1 N_s + N_{ss}^T w_2 N_{ss}) ds \tag{13}$$ and

$$F_{tot}^{N} = \int_0^1 N^T f_{tot}^{N} ds . \tag{14}$$

where $N = [N_0 \quad N_1 \quad N_2 \quad N_3]$, $N_s = \frac{dN}{ds}$, $N_{ss} = \frac{d^2 N}{ds^2}$, N^T represents transpose of N, μ is the mass density, γ is the viscosity, w_1 is the tension coefficient. w_2 is the rigidity coefficient and f_{tot}^{N} is the total external force acting on an element (of boundary contour).

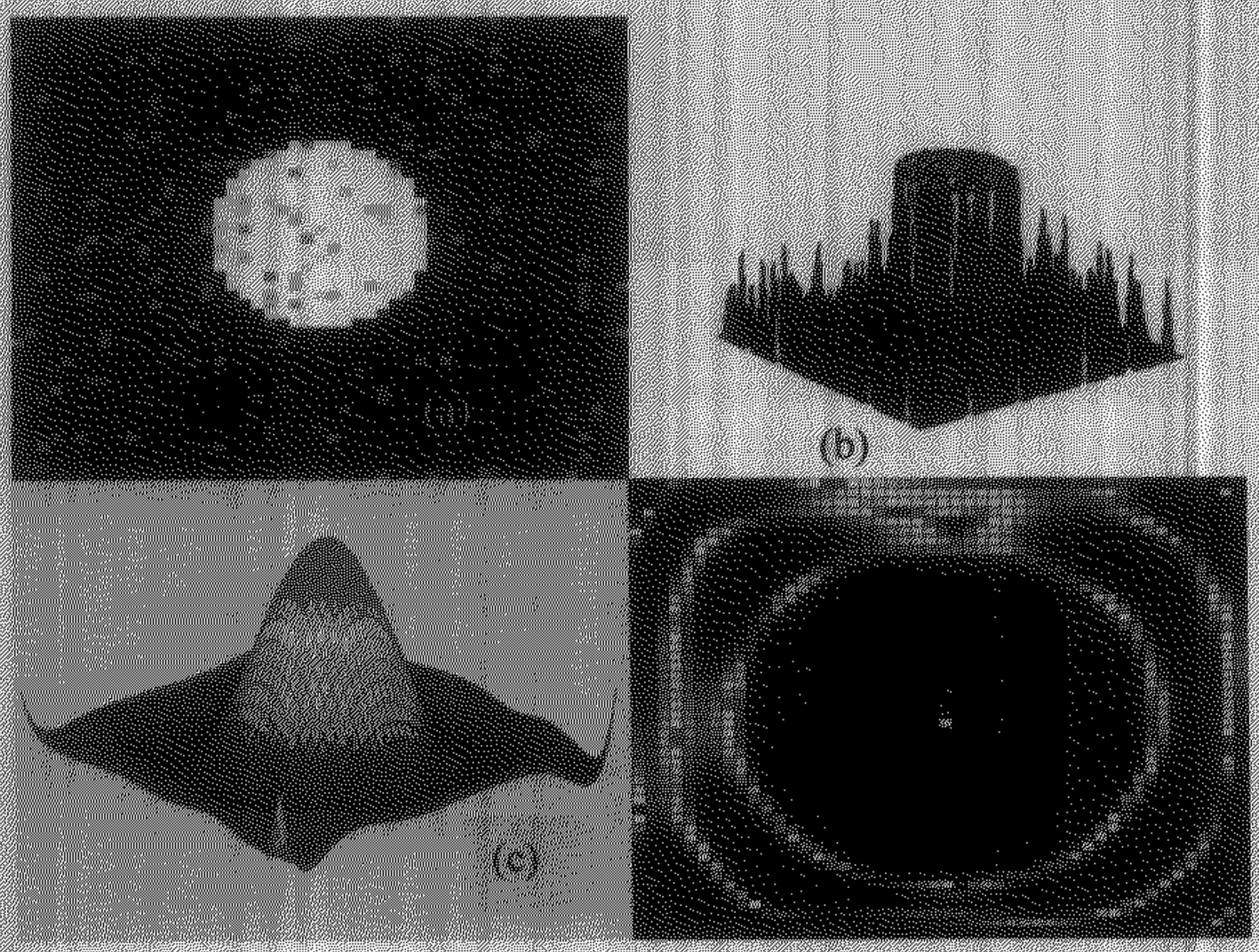
Now our task remains to find total external force (f_{tot}^{N}) acting on an element. An interesting behavior of the normal (drawn on the approximated surface) can be used to find a force. If we draw the normal on the approximated surface we can see the angel (ψ) between normal and image plane varies smoothly except at the boundary points. That means that the spatial differentiation of the angle (ψ) will give high values at the boundary points. Let M_{th} is the matrix, which contains the angle (ψ) for all surface normals described by rectangular image region. Also let M_{thsp} denotes matrix obtained by spatial differentiation of M_{th} and m_{thsp} is a value of M_{thsp} which is greater then all the values of this matrix corresponding to the position inside the object. So we may assume a force given by

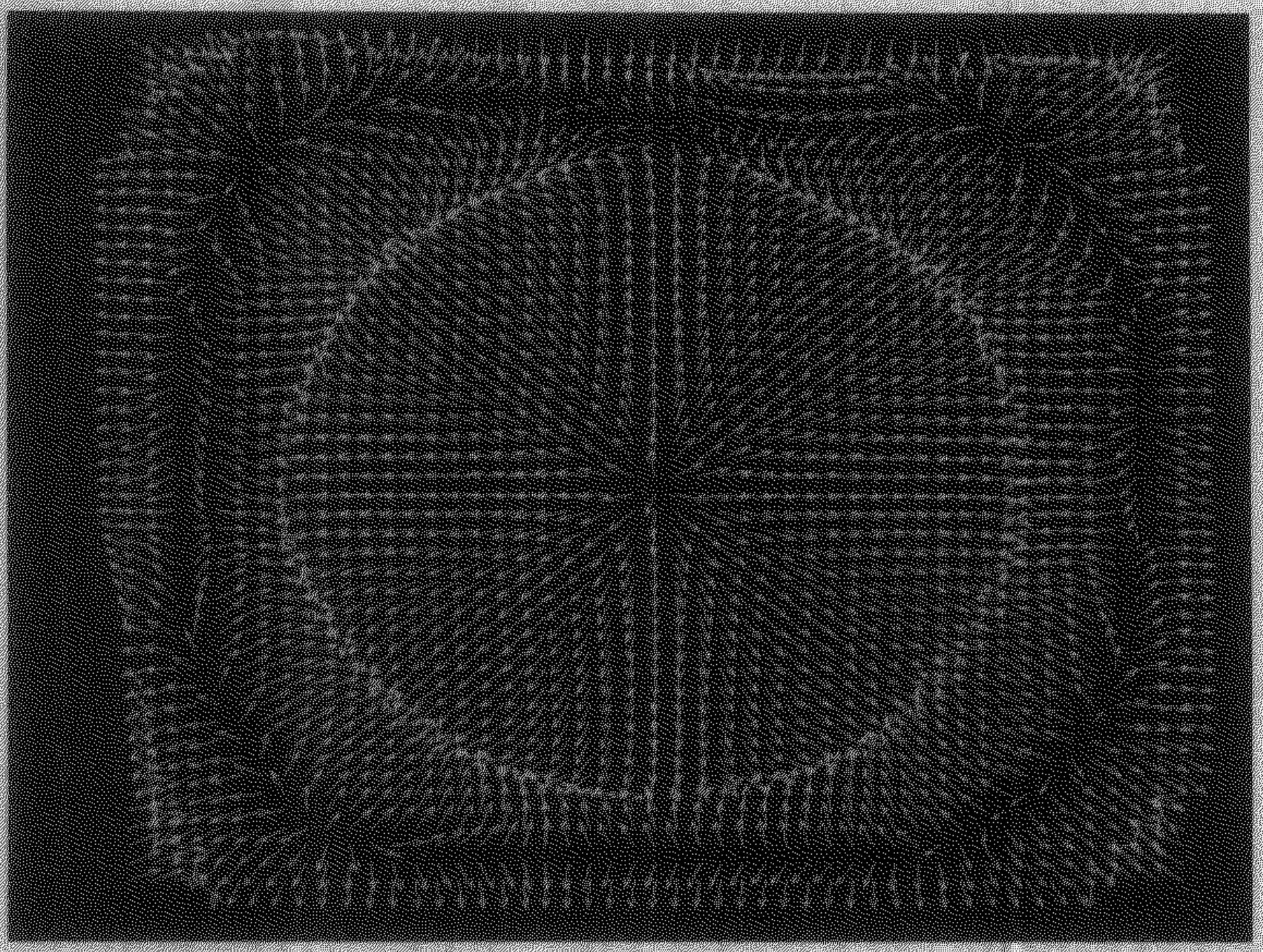
$$f_1^{N} = (m_{thsp} - m)\mathbf{r} , \tag{15}$$

where m is a value of M_{thsp} corresponding to the position of the boundary element and $\mathbf{r}$ is the position vector of the boundary element relative to the object center and it is given by

$$r = \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix}, \tag{16}$$

where (x, y) is the position of the boundary element, (x_c, y_c) is the position of the center of the object. But if the object is not very large, it is seen that this value of angle (ψ) is very less inside the object and high at the boundary of the object. So we can directly use M_m instead of M_{thsp} to compute this force. To visualize why this should work let us consider *Figure 3*. *Figure 3(a)* is a white synthetic circle image inside a black background. The image is sprayed with 30% random salt and pepper type noise. *Figure 3(b)* is the visualization of intensity as 3d surface map while *Figure 3(c)* is the approximated surface where $\theta = 9$. The matrix M_m for this synthetic image is shown in *Figure 3(d)*. *Figure 3(e)* shows projection of normals to the image plane. The projection of the normals can be obtained by neglecting z component of the normals. *Figure 3(d)* shows that value of ψ is less (black) inside the object and high at the boundary (white).





(e)

Figure 3: (a) Actual image (b) Actual image data points, (c) Approximated surface, (d) image representation of the angle between the normals and the image plane (M_{ih}) and (d) projection of normals on the image plane

Now M_{ih} can be calculated by the expression given by

$$M_{ih} = \left[\tan^{-1}\left(\frac{N_z(x,y)}{\sqrt{N_x(x,y)^2 + N_y(x,y)^2}} \right) \right], \qquad (17)$$
$$= [M_{ih}(x,y)]$$

and M_{imp} can be calculated as

$$M_{imp} = \left[\sqrt{\left(\frac{\partial M_{ih}(x,y)}{\partial x}\right)^2 + \left(\frac{\partial M_{ih}(x,y)}{\partial y}\right)^2} \right], \qquad (18)$$

Now consider the *Figure 3(e)*, which shows the projection of normal in the image plane. It can be easily seen from the *Figure 3(e)* that the projection of normals in the image

plane is outwards from inside the object. It is zero or in opposite direction outside the object. From this observation we can assume a force (f_2^N) whose magnitude and direction is given directly by the projection of the normal in the image plane. Thus, a force f_2^N can be given by

$$f_2^N = \begin{bmatrix} N_x \\ N_y \end{bmatrix}, \tag{19}$$

where N_x and N_y are the components of the normal in the x and y directions respectively.

Also image gray value is important information to find object boundary. Let I_{av} is the average gray value of the background. Then a force f_3^N can be given as

$$f_3^N = |I - I_{av}|\mathbf{r} \ . \tag{20}$$

where I is gray value at the current position of boundary element and $|I - I_{av}|$ indicates magnitude of difference of I and I_{av}. The parameter $\mathbf{r}$ is explained earlier.
So to compute total force, f^N_{tot}. we add this three above-mentioned force. Thus

$$f_{tot}^N = f_1^N + f_2^N + f_3^N \ . \tag{21}$$

From the Lagrangian motion equation (10) it is clear that the value of X satisfying this equation is a continuous function of t and hence in order to solve this equation (10) we need to express dynamic force (F_{tot}^N) also as a continuous function of t. But we can compute this dynamic force (F_{tot}^N) only at discrete time. Thus we require numerical implementation of this equation (10) that can provide us the solution of X at the discrete time step. Next sub-section describes the actual numerical implementation in order to solve X.

2.2.1 Numerical implementation: For numerical implementation we assume critically damped condition. That means in equation (10) we put $M = 0$ so the equation becomes

$$C\frac{dX(t)}{dt} + KX(t) = F_{tot}^N \tag{22}$$

$$\Rightarrow C\frac{X(t+\Delta t) - X(t)}{\Delta t} + KX(t) = F_{tot}^N$$

$$\Rightarrow C(X(t+\Delta t) - X(t)) = \Delta t(F_{tot}^N - KX(t))$$

$$\Rightarrow C(X(t+\Delta t)) = CX(t) + \Delta t(F_{ext}^{N} - KX(t))$$

$$\Rightarrow C(X(t+\Delta t)) = (C - K\Delta t)X(t) + \Delta t F_{ext}^{N}$$

$$\Rightarrow X(t+\Delta t) = (I - C^{-1}K\Delta t)X(t) + C^{-1}\Delta t F_{ext}^{N}, \qquad (23)$$

From this equation we can iteratively solve $X(t+\Delta t)$ from $X(t)$. We stop our iteration when boundary is reached that means when no significant change of $X(t)$ is found. However this damping matrix C is often sparse and ill conditioned. In that case finding C^{-1} is very difficult. There are several approaches to solve this problem. The simplest way is keeping only the diagonal coefficients of C and discarding all of the diagonal coefficients [4].

As described earlier after detection of the boundary from the initial frame, next task is to track the moving object from the frames of sequence. Our next sub-section (2.3) provides the algorithm for the object tracking in detail.

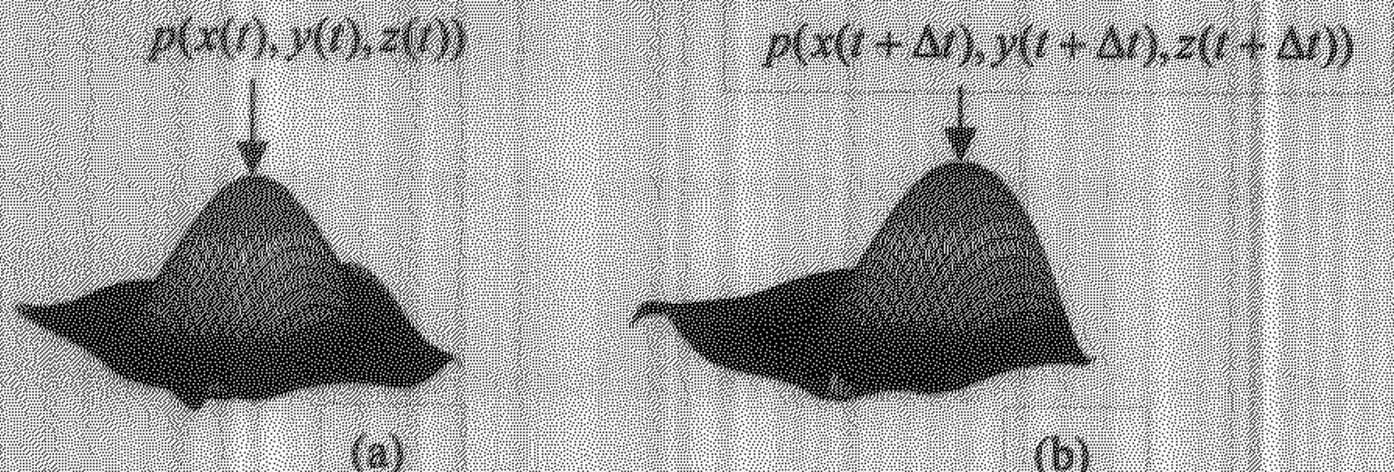


Figure 4: (a) Surface drawn at time t showing a point $p(x(t), y(t), z(t))$ and (b) Surface drawn at time $t+\Delta t$ identifying the same point as $p(x(t+\Delta t), y(t+\Delta t), z(t+\Delta t))$

2.3 Tracking with approximated surface: So far we have been able to approximate a surface given an imaged object and we are also able to detect boundary of the said object. We now want to track the target object by scanning the sequence of frames. In the procedure described here user identify the object in the first frame. As described earlier our algorithm first approximates a surface inside the region containing the object and detects its boundary. The detected boundary in the tth frame is projected in the same spatial location in $t+1$th frame. A new surface is approximated in the region containing the object in $t+1$th frame. The algorithm then uses both the surfaces corresponding to tth and $t+1$th frame to calculate spatial movements in x and y directions. The procedure details are described below.

A point p is defined earlier as $p(x, y, z)$ where $z = f(x, y)$. If the point concerned corresponds to tth frame we may rewrite this point as $p(x(t), y(t), z(t))$ where $z(t) = f(x, y, t)$. Suppose the object is now moved to a new position in the $t+1$th frame

by the amount Δx and Δy in the x and y directions respectively. Then the point p can be specified in the $t+1$th frame as $p(x(t+\Delta t), y(t+\Delta t), z(t+\Delta t))$ where. Δt represents the time elapse between tth and $t+1$th frames. $x(t+\Delta t) = x(t) + \Delta x$, $y(t+\Delta t) = y(t) + \Delta y$ and $z(t+\Delta t) = f(x+\Delta x, y+\Delta y, t+\Delta t)$. This is shown in the *Figure 4*. Since between frame movements are marginal we can have the following assumptions

1. The z values of the two points are approximately equal. that means $z(t-\Delta t) \cong z(t)$ so that

$$f(x+\Delta x, y+\Delta y, t+\Delta t) \cong f(x, y, t). \tag{24}$$

2. All the components of normal at the two points are also approximately equal (because object deformation is assumed negligible).
So we have following assumptions

$$N_x(x+\Delta x, y+\Delta y, t+\Delta t) \cong N_x(x, y, t). \tag{25}$$

$$N_y(x+\Delta x, y+\Delta y, t+\Delta t) \cong N_y(x, y, t). \tag{26}$$

$$N_z(x+\Delta x, y+\Delta y, t+\Delta t) \cong N_z(x, y, t). \tag{27}$$

where N_x, N_y and N_z are the components of the normal in the x, y and z directions respectively.

We know that outward normal of any surface of the form $\sigma(x, y, z) = k$ is given by $N = \nabla\sigma$ where ∇ is defined by $\nabla = i\frac{\partial}{\partial x} + j\frac{\partial}{\partial y} + k\frac{\partial}{\partial z}$. Rewriting $z = f(x, y)$ in the form of $\sigma(x, y, z) = k_0$ we get $\sigma(x, y, z) = f(x, y) - z = 0$. From this we get $\frac{\partial\sigma}{\partial x} = \frac{\partial f}{\partial x}$, $\frac{\partial\sigma}{\partial y} = \frac{\partial f}{\partial y}$ and $\frac{\partial\sigma}{\partial z} = -1$. So we have normal $N = \nabla\sigma = i\frac{\partial f}{\partial x} + j\frac{\partial f}{\partial y} + k(-1)$. Thus

$$N = \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ -1 \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ -1 \end{bmatrix}, \tag{28}$$

Using equation (28) we can rewrite equation (25), (26) and (27) as

$$f_x(x+\Delta x, y+\Delta y, t+\Delta t) \cong f_x(x, y, t), \tag{29}$$

$$f_y(x+\Delta x, y+\Delta y, t+\Delta t) \cong f_y(x, y, t)\,. \tag{30}$$

$-1 \cong -1$.

where $f_x = \frac{\partial f}{\partial x}$ and $f_y = \frac{\partial f}{\partial y}$. . From equation (24) expanding $f(x+\Delta x, y+\Delta y, t+\Delta t)$ by Taylor's series we get

$$f(x, y, t) = f(x, y, t) + [\frac{\partial}{\partial x} f(x, y, t)\Delta x + \frac{\partial}{\partial y} f(x, y, t)\Delta y + \frac{\partial}{\partial t} f(x, y, t)\Delta t] + e$$

where e contains second and higher order terms in $\Delta x, \Delta y$ and Δt. Neglecting e, canceling $f(x, y, t)$ and dividing by Δt we get and taking the limit as $\Delta t \to 0$ we get

$$f_x(x, y, t)\frac{dx}{dt} + f_y(x, y, t)\frac{dy}{dt} + f_t(x, y, t) = 0$$

$$\Rightarrow f_x(x, y, t)u + f_y(x, y, t)v + f_t(x, y, t) = 0\,. \tag{31}$$

where $f_t = \frac{\partial f}{\partial t}$ and f_x, f_y are explained earlier. This equation (31) is a constraint equation and is similar to optical flow constraint equation. Similarly from equation (29) and (30) we get

$$f_{xx}(x, y, t)u + f_{yx}(x, y, t)v + f_{tx}(x, y, t) = 0\,, \tag{32}$$

$$f_{xy}(x, y, t)u + f_{yy}(x, y, t)v + f_{ty}(x, y, t) = 0\,, \tag{33}$$

where $f_{xx} = \frac{\partial f_x}{\partial x}$, $f_{xy} = \frac{\partial f_y}{\partial x}$, $f_{yx} = \frac{\partial f_x}{\partial y}$, $f_{ty} = \frac{\partial f_y}{\partial t}$ and $f_{tx} = \frac{\partial f_x}{\partial t}$. So now we have two unknowns u, v and three equations (31), (32) and (33) for every special coordinate (x, y). Thus it is possible to find values of u and v at any particular time for every (x, y). The pair (u, v) is actually the velocity component in the directions x and y respectively. We may represent this as a velocity vector $V = \begin{bmatrix} u \\ v \end{bmatrix}$. The velocity vector V for every image point (x, y) is called motion field [5]. But we would not go for computing all such pair (u, v) because of two main reasons. First computation cost is very high as for every spatial coordinate (x, y) we need compute these three equations (31), (32) and (33). Second these three equations are the approximate representation obtained by the expansion using Taylor's series, where second and higher order terms are neglected. To eliminate these two problems we have made another assumption of rigid body motion that means deformation of the object is assumed negligible. Thus, the net displacement of

any point (x, y) of the object in the image plane can be calculated from the displacement of its center the object and its rotation about the center. As a result the velocity vector of any point (x, y) can be calculated from the velocity of the center V_c and the rate of change of rotational angle (ϕ) of the object about the center. Let velocity of the center is defined as $V_c = \begin{bmatrix} u_c \\ v_c \end{bmatrix}$ and the rate of change of rotational angle (ϕ) or the angular velocity is defined as $\omega = \frac{d\phi}{dt}$. It is now sufficient to calculate only these three parameters u_c, v_c and ω to calculate velocity vectors for every image point (x, y) (motion field). Let us now see how to find velocity vector V of any arbitrary point (x, y) from V_c and ω.

Consider *Figure 5*(a) and (b) shows a rigid body drawn at time t and $t + \Delta t$ respectively.

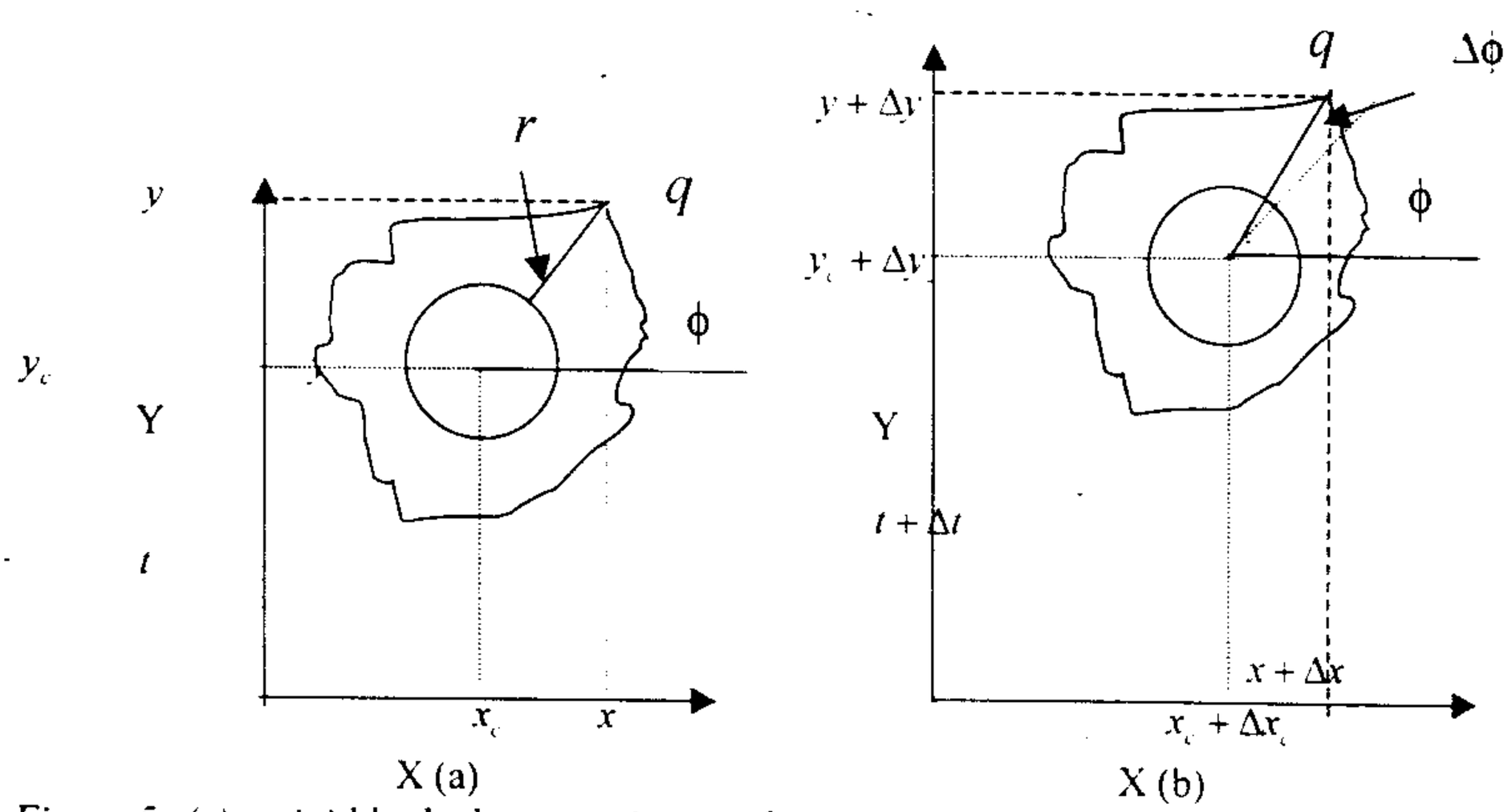


Figure 5: (a) a rigid body drawn at time (t) showing its center and a point (q). (b) Same body at time $t + \Delta t$ showing movements of its center and the point (q).

From the *Figure 5* it is clear that

$$x = x_c + r\cos\phi \quad , \tag{34}$$

and

$$y = y_c + r\sin\phi \quad , \tag{35}$$

where r is given by

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \quad (36)$$

and ϕ is given by

$$\phi = \tan^{-1}\left(\frac{y - y_c}{x - x_c}\right). \quad (37)$$

Since r is the relative distance of a point q of the object from its center and the object deformation is negligible, it will be invariant with respect to time for that particular point concerned.

From (36), and (37) and writing $\frac{dx_c}{dt} = u_c$, $\frac{dy_c}{dt} = v_c$ we get

$$u = u_c - r\omega \sin\phi . \quad (38)$$

$$v = v_c + r\omega \cos\phi . \quad (39)$$

From equation (38) and (39) we can easily calculate velocity vector V of any image point (x, y) when values of u_c, v_c and ω are known. To calculate u_c, v_c and ω we just replace the terms u, v from equation (31), (32) and (33). Now any deviation from equation (31), (32) and (33) is consider error. Since error can be both positive and negative we take square of error from those three equations. Again since error introduced by equation (31) is nonlinear with error given by equation (32) and (33) a factor λ is multiplied to left hand side of equation (31) in order to compute error. So the total error becomes

$$\begin{aligned} E = {} & \iint [f_{xx}(x,y,t)u + f_{xy}(x,y,t)v + f_{xt}(x,y,t)]^2 dxdy \\ & + \iint [f_{xy}(x,y,t)u + f_{yy}(x,y,t)v + f_{yt}(x,y,t)]^2 dxdy \\ & + \lambda \iint [f_x(x,y,t)u + f_y(x,y,t)v + f_t(x,y,t)]^2 dxdy \end{aligned} \quad (40)$$

Replacing u and v from equation (40) by the expression given by equation (38) and (39) we get error term given by.

$$\begin{aligned} E = {} & \iint [f_{xx}(x,y,t)(u_c - r\omega \sin\phi) + f_{xy}(x,y,t)(v_c + r\omega \cos\phi) + f_{xt}(x,y,t)]^2 dxdy \\ & + \iint [f_{xy}(x,y,t)(u_c - r\omega \sin\phi) + f_{yy}(x,y,t)(v_c + r\omega \cos\phi) + f_{yt}(x,y,t)]^2 dxdy \\ & + \lambda \iint [f_x(x,y,t)(u_c - r\omega \sin\phi) + f_y(x,y,t)(v_c + r\omega \cos\phi) + f_t(x,y,t)]^2 dxdy \end{aligned} \quad (41)$$

We now want to find the values of u_c, v_c and ω so that total error E becomes minimum. So differentiating equation (43) with respect to u_c we get

$$\frac{\partial E}{\partial u_c} = 2\iint[f_{xx}(u_c - r\omega\sin\phi) + f_{xy}(v_c + r\omega\cos\phi) + f_{tx}]f_{xx}dxdy$$
$$+ 2\iint[f_{xy}(u_c - r\omega\sin\phi) + f_{yy}(v_c + r\omega\cos\phi) + f_{ty})]f_{xy}dxdy$$
$$+ 2\lambda\iint[f_x(u_c - r\omega\sin\phi) + f_y(v_c + r\omega\cos\phi) + f_t]f_x dxdy$$

where $f_x = f_x(x,y,t)$, $f_y = f_y(x,y,t)$, $f_{xy} = f_{xy}(x,y,t)$, $f_{yy} = f_{yy}(x,y,t)$, $f_{xx} = f_{xx}(x,y,t)$, $f_t = f_t(x,y,t)$, $f_{tx} = f_{tx}(x,y,t)$, and $f_{ty} = f_{ty}(x,y,t)$. Simplifying and setting $\frac{\partial E}{\partial u_c} = 0$, we get

$$[\iint(f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)dxdy]u_c + [\iint(f_{xx}f_{yx} + f_{xy}f_{yy} + \lambda f_x f_y)dxdy]v_c$$
$$+ [\iint\{f_{xx}f_{yx} + f_{xy}f_{yy} + \lambda f_x f_y)r\cos\phi - (f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)r\sin\phi\}dxdy]\omega$$
$$+ [\iint(f_{xx}f_{tx} + f_{xy}f_{ty} + \lambda f_x f_t)dxdy] = 0 \qquad (42)$$

Similarly setting $\frac{\partial E}{\partial v_c} = 0$ we get

$$[\iint(f_{xx}f_{yx} + f_{xy}f_{yy} + \lambda f_x f_y)dxdy]u_c + [\iint(f_{yx}^2 + f_{yy}^2 + \lambda f_y^2)dxdy]v_c$$
$$[\iint(f_{yx}^2 + f_{yy}^2 + \lambda f_y^2)r\cos\phi dxdy - (f_{xx}f_{yx} + f_{xy}f_{yy} + \lambda f_x f_y)r\sin\phi]\omega$$
$$+ [\iint(f_{xy}f_{tx} + f_{yy}f_{ty} + \lambda f_y f_t)dxdy] = 0 \qquad (43)$$

Similarly setting $\frac{\partial E}{\partial \omega} = 0$ we get

$$[\iint\{(\lambda f_x f_y + f_{xx} f_{xy} - f_{xy} f_{yy})\cos\phi - (f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)\sin\phi\}dxdy]u_c$$

$$+[\iint\{(f_{xy}^2 + f_{yy}^2 + \lambda f_y^2)\cos\phi - (f_{xx} f_{xy} + f_{xy} f_{yy} + \lambda f_x f_y)\sin\phi\}dxdy]v_c$$

$$[\iint\{\sin^2\phi(f_{xx}^2 - f_{xy}^2 + \lambda f_x^2) + \cos^2\phi(f_{xy}^2 + f_{yy}^2 + \lambda f_y^2) -$$

$$2\sin\phi\cos\phi(f_{xx} f_{yy} + f_{xy} f_{yy} + \lambda f_x f_y)\}rdxdy]w$$

$$+[\iint\{(f_{tx} f_{yx} + f_{ty} f_{yy} + \lambda f_t f_y)\cos\phi - (f_{tx} f_{xx} + f_{ty} f_{xy} + \lambda f_t f_x)\sin\phi\}dxdy] = 0 \quad (44)$$

Equation (43), (44) and (45) can be written as

$$b_{11}u_c + b_{12}v_c + b_{13}\omega = d_1 \quad (45)$$

$$b_{21}u_c + b_{22}v_c + b_{23}\omega = d_2 \quad (46)$$

$$b_{31}u_c + b_{32}v_c + b_{33}\omega = d_3 \quad (47)$$

where $b_{11} = \iint(f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)dxdy]$, $b_{12} = [\iint(f_{xx} f_{xx} + f_{xy} f_{xy} + \lambda f_x f_y)dxdy]$

$b_{13} = [\iint\{f_{xx} f_{xx} + f_{xy} f_{xy} + \lambda f_x f_y)r\cos\phi - (f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)r\sin\phi\}dxdy]$.

$d_1 = [\iint(f_{xx} f_{tx} + f_{xy} f_{ty} + \lambda f_x f_t)dxdy]$. $b_{21} = [\iint(\lambda f_x f_y + f_{xx} f_{xy} + f_{xy} f_{yy})dxdy]$

$b_{22} = [\iint(\lambda f_y^2 + f_{xx}^2 + f_{yy}^2)dxdy]$. $b_{23} = [\iint(\lambda f_y f_t + f_{xy} f_{tx} + f_{yy} f_{ty})dxdy]$

$d_2 = -[\iint(f_{xy} f_{tx} + f_{yy} f_{ty} + \lambda f_y f_t)dxdy]$.

$b_{31} = [\iint\{(\lambda f_x f_y + f_{xx} f_{xy} + f_{xy} f_{yy})\cos\phi - (f_{xx}^2 + f_{xy}^2 + \lambda f_x^2)\sin\phi\}dxdy]$.

$b_{32} = [\iint\{(f_{xy}^2 + f_{yy}^2 + \lambda f_y^2)\cos\phi - (f_{xx} f_{xy} + f_{xy} f_{yy} + \lambda f_x f_y)\sin\phi\}dxdy]$.

$b_{33} - [\iint\{\sin^2\phi(f_{xx}^2 + f_{xy}^2 + \lambda f_x^2) + \cos^2\phi(f_{xy}^2 + f_{yy}^2 + \lambda f_y^2) -$

$2\sin\phi\cos\phi(f_{xx} f_{xy} + f_{xy} f_{yy} + \lambda f_x f_y)\}rdxdy]$.

$$d_3 = [\iint\{(f_{tx} f_{yx} + f_{ty} f_{yy} + \lambda f_t f_y)\cos\phi - (f_{tx} f_{xx} + f_{ty} f_{xy} + \lambda f_t f_x)\sin\phi\}dxdy]$$

Then knowing the values of coefficients of the equations (45), (46) and (47), we can solve them by the expression given by

$$u_c = \frac{\begin{bmatrix} d_1 & b_{12} & b_{13} \\ d_2 & b_{22} & b_{23} \\ d_3 & b_{32} & b_{33} \end{bmatrix}}{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}, v_c = \frac{\begin{bmatrix} b_{11} & d_1 & b_{13} \\ b_{12} & d_2 & b_{23} \\ b_{13} & d_3 & b_{33} \end{bmatrix}}{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}, \omega = \frac{\begin{bmatrix} b_{11} & b_{12} & d_1 \\ b_{12} & b_{22} & d_2 \\ b_{13} & b_{32} & d_3 \end{bmatrix}}{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}} \tag{48}$$

Let us now find the value of $f_x, f_t, f_{xx}, f_{xt}, f_{xy}, f_t, f_{tt}$, and f_{yy} from the surface equation given by (1). Since we know that the coefficients of the equation of the surface given by equation (1) changes with time so equation (1) can be rewritten, to include a new time variable, as

$$\begin{aligned} f(x,y,t) &= a_{00}(t) + a_{10}(t)x + a_{01}(t)y + a_{20}(t)x^2 + a_{11}(t)xy + a_{02}(t)y^2 + a_{30}(t)x^3 \\ &\quad + a_{21}(t)x^2y + a_{12}(t)xy^2 + a_{03}(t)y^3 \ldots\ldots a_{0\theta}(t)y^\theta \\ &= \sum_{k=0}^{\theta}\sum_{l=0}^{k} a_{k-l,l}(t)x^{k-l}y^l \end{aligned} \tag{49}$$

From this equation we have

$$f_x(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k}(k-l)a_{k-l,l}(t)x^{k-l-1}y^{l-1} \tag{50}$$

$$f_y(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k} l a_{k-l,l}(t)x^{k-l}y^{l-1} \tag{51}$$

$$f_{xx}(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k}(k-l)(k-l-1)a_{k-l,l}(t)x^{k-l-2}y^l \tag{52}$$

$$f_{xy}(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k}(k-l)l a_{k-l,l}(t)x^{k-l-1}y^{l-1} \tag{53}$$

$$f_{yy}(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k} l(l-1)a_{k-l,l}(t)x^{k-l}y^{l-2} \tag{54}$$

$$f_t(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k}\frac{da_{k-l,l}(t)}{dt}x^{k-l}y^l \tag{55}$$

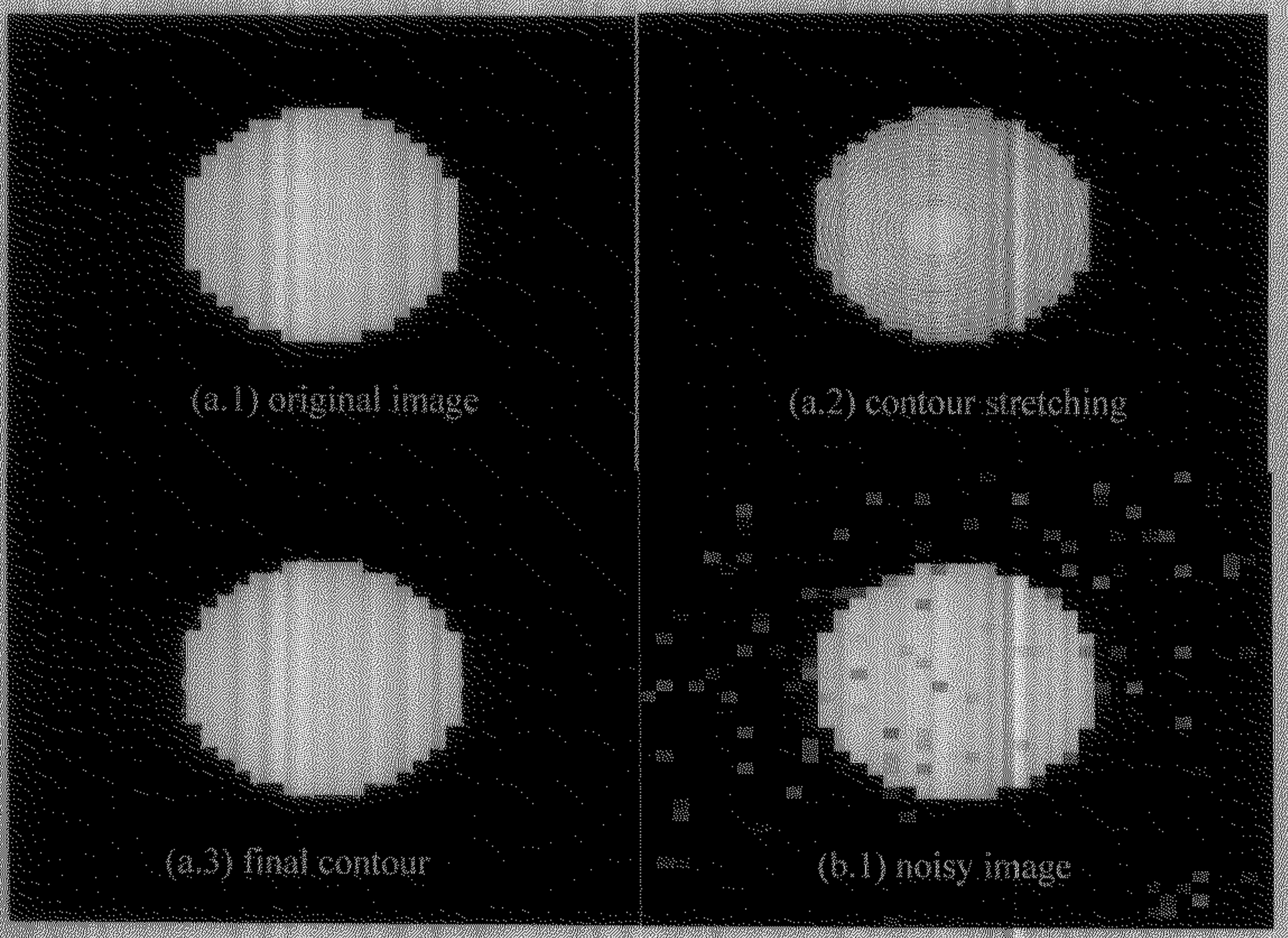
$$f_{xt}(x,y,t) = \sum_{k=0}^{\theta}\sum_{l=0}^{k}(k-l)\frac{da_{k-l,l}(t)}{dt}x^{k-l-1}y^{l-1} \tag{56}$$

$$f_w(x,y,t) = \sum_{k=0}^{n}\sum_{l=0}^{k} l \frac{da_{k-l,l}(t)}{dt} x^{k-l} y^{l-1} \qquad (57)$$

In the next section we show the experimental results of object boundary detection and tracking examples using some synthetic as well as real life image sequence with the help of the algorithm described in this section (2).

3. Results

In this section we discuss some implementations of our algorithm in details. All the results shown in this section are tentative results of our program, we have not attempted for any code optimization. We have shown an example of surface approximation in *Figure 1* showing the surfaces drawn for various order (θ). Closeness of the surface to the actual data increases with increasing θ and decreases with increasing size of image data. We have used $\theta = 9$ for surface approximation as, the approximated surface for this θ value, is seen to be close enough with the actual data for our applications.



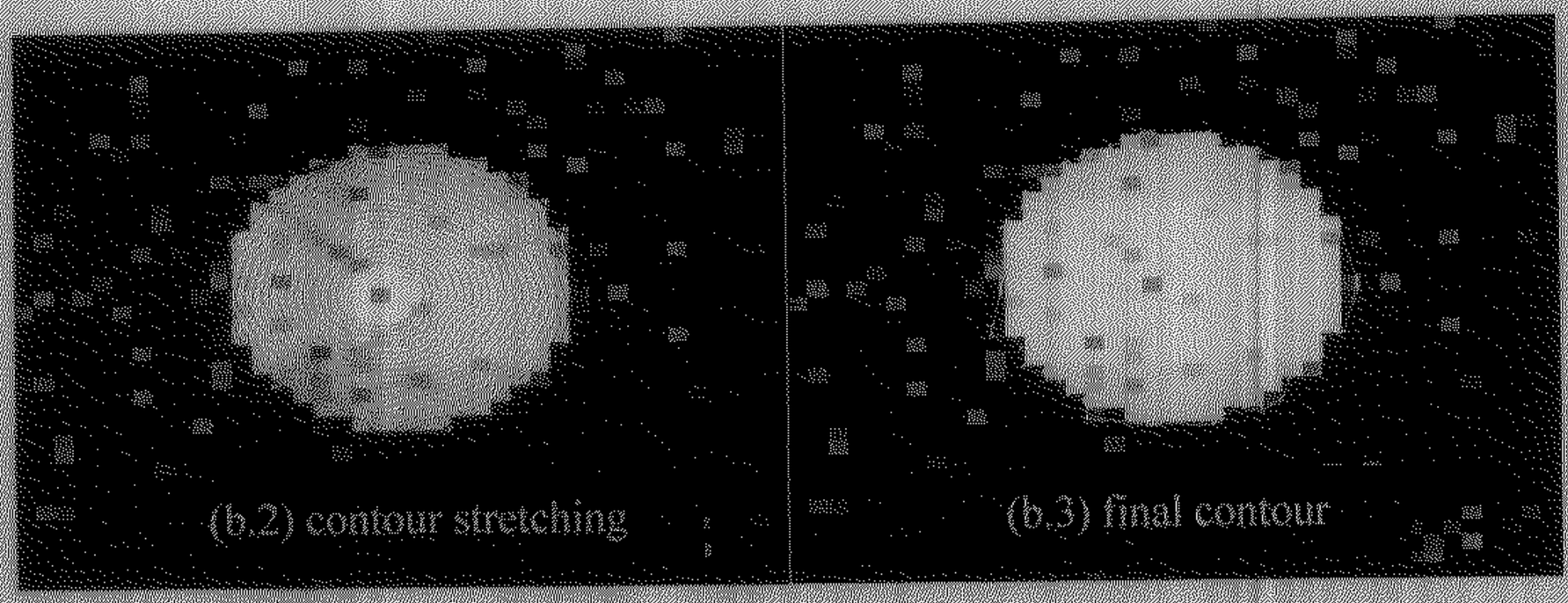


Figure 6: Example of Boundary detection of a synthetic circular image (a) before and (b) after addition of random noise.

Figure 6(a) shows the results of boundary detection of a synthetic white circular image inside a black background. *Figure 6(b)* shows the results of the same image after addition of 30% random salt and paper type noise. The series of blue contours in *Figure 6(a.2)* or *Figure 6(b.2)* shows how initial contour from the center of the object has been stretched under the influence of three forces described under section 2.2. Each blue contour shown in Figure 6(a.2) or Figure 6(b.2) is the result of a particular iteration. From the figure it is also clear that the initial boundary contour has expanded rapidly inside the object but it has stopped it's stretching when boundary is reached. The green contour in Figure 6(a3) or in Figure 6(b.3) is the result of final iteration. If we compare Figure 6(a.2) and Figure 6(a.3) we can see that contour in the Figure 6(a.2) has smooth stretching whereas in Figure 6(b.2) contour stretching is not so smooth. This is because Figure 6(b) is noisy and noise acts as obstacle to the movement of the contour. Similarly Figure 7(a) shows the results of boundary detection of a synthetic rectangular image. Figure 7(b) shows the results of the same image after addition of 30% random noise. Finally three real life application of boundary detection is shown in Figure 8(a), (b) and (c). The complete result is shown in Table1.

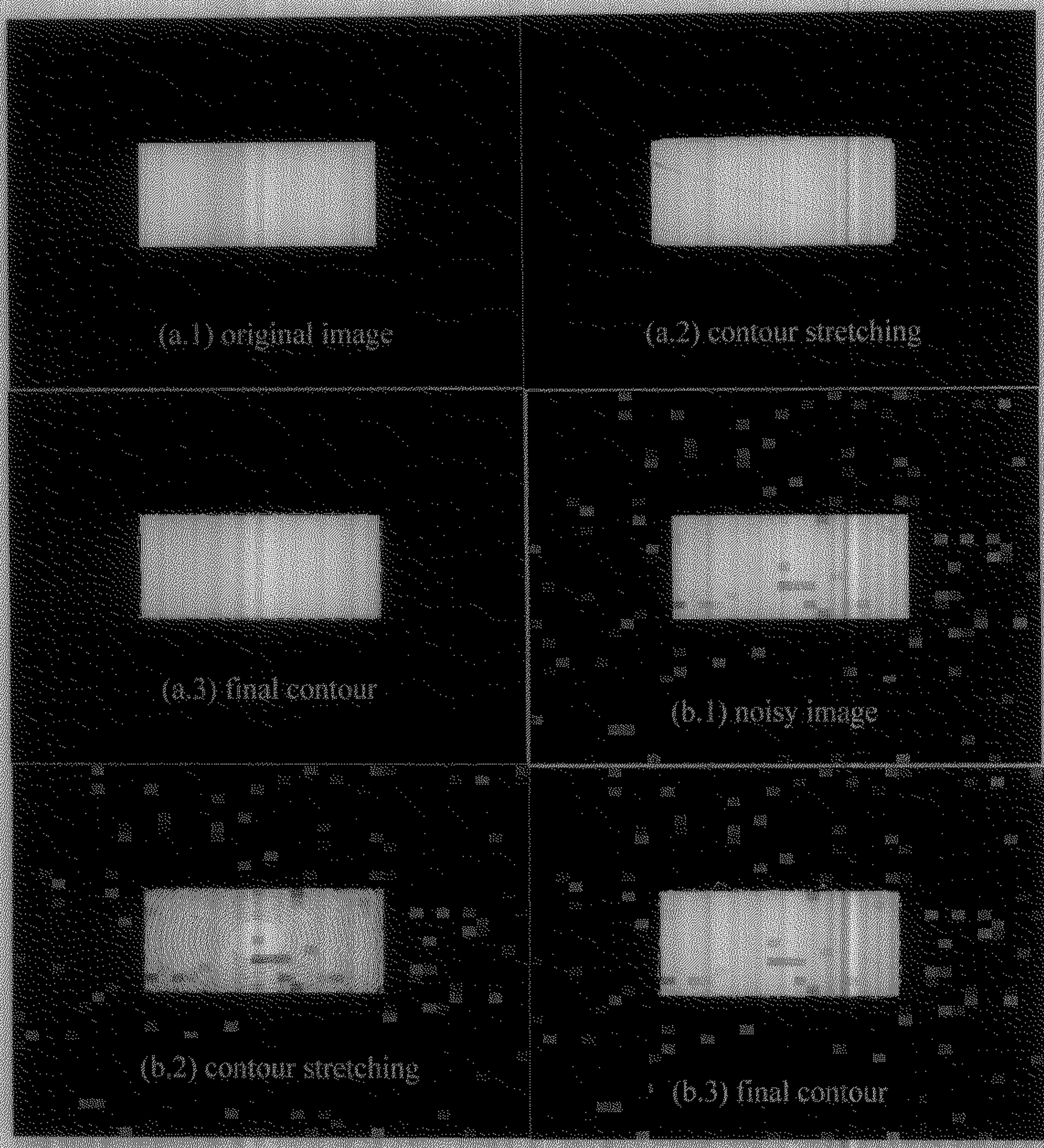
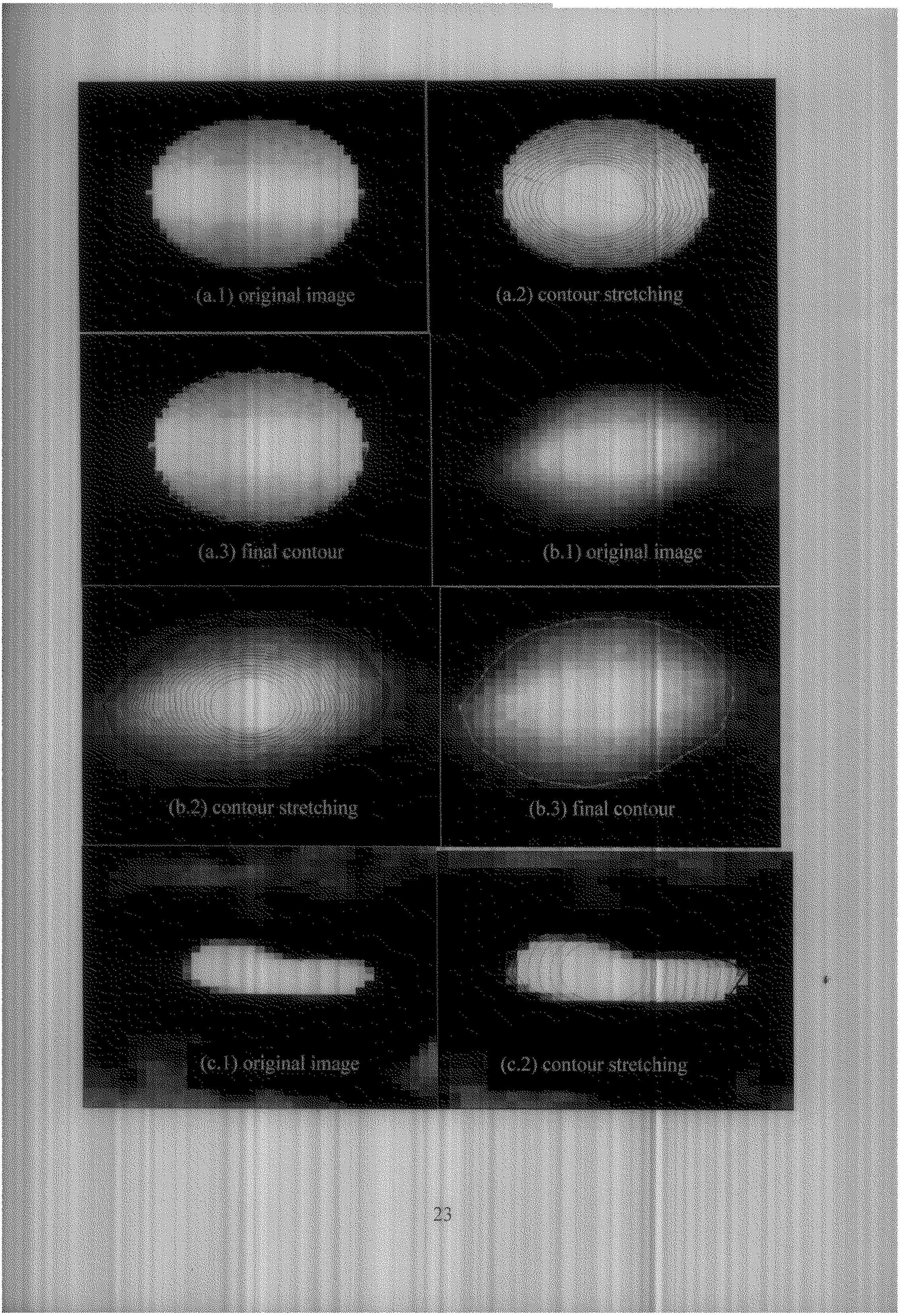


Figure 7: Boundary detection of a synthetic rectangular image (a) before addition of noise and (b) after addition of noise.


(a.1) original image
(a.2) contour stretching
(a.3) final contour
(b.1) original image
(b.2) contour stretching
(b.3) final contour
(c.1) original image
(c.2) contour stretching

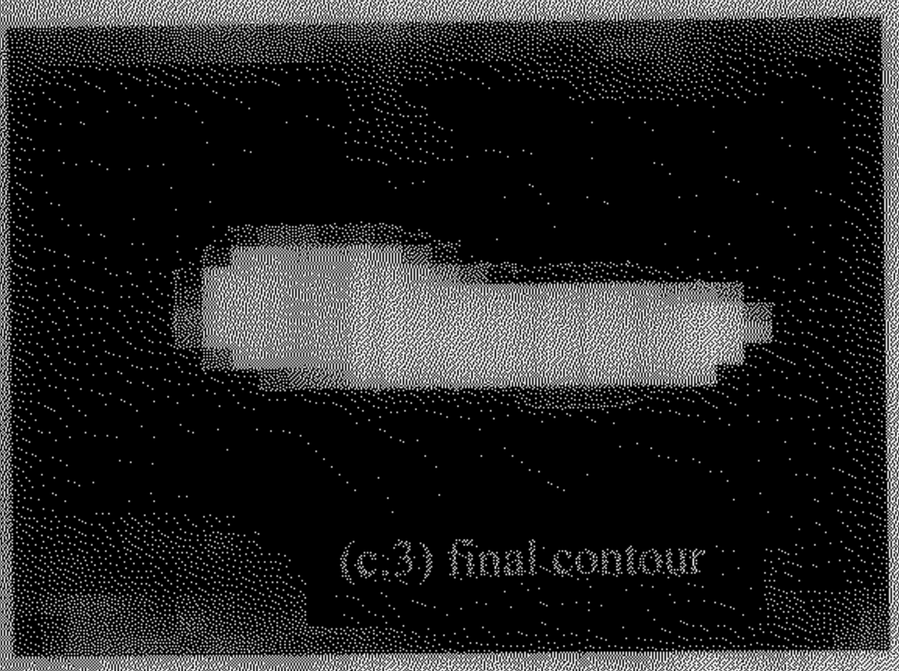


Figure 8: Boundary detection of three typical images shown in (a), (b) and (c) respectively

Table1: Results of the boundary detection in Figure 7,8 and 9

Figure no.	*γ*	w_1	w_2	*Δt*	*No. Of iteration used*	*Total time(s)*
7(a)	1	.05	.05	.06	50	.9300
7(b)	1	.05	.05	.06	50	.9300
8(a)	1	.05	.05	.06	50	.9300
8(b)	1	.05	.05	.06	50	.9300
9(a)	1	.05	.05	.06	150	2.8000
9(b)	1	.05	.05	.06	150	2.8000
9(c)	1	.05	.05	.06	150	2.8600

Let us now see some examples of object tracking from sequence of frames. For object tracking we have set up value of γ experimentally. In all the examples we show the results in the gap of five sequences each. First in *Figure 9* we show moving ball tracking. The complete result is given in *Table 2*. At every step, we have detected the object boundary and the boundary is drawn by blue contour. Similarly *Figure 10* shows the tracking of bat from the same sequence. In the figure we have shown the result up to 50 frames, but the *Table 3* shows the results up to 100 frames. *Figure 11* shows another example of tracking of a ball from a sequence of frames. In the sequence shown in *Figure11* all the five balls are moving, but we select a particular ball for our tracking. The complete result of this tracking is given in *Table 4*. In *Figure 13* we have shown tracking of a typical moving object. *Table 5* shows the complete result of this tracking.

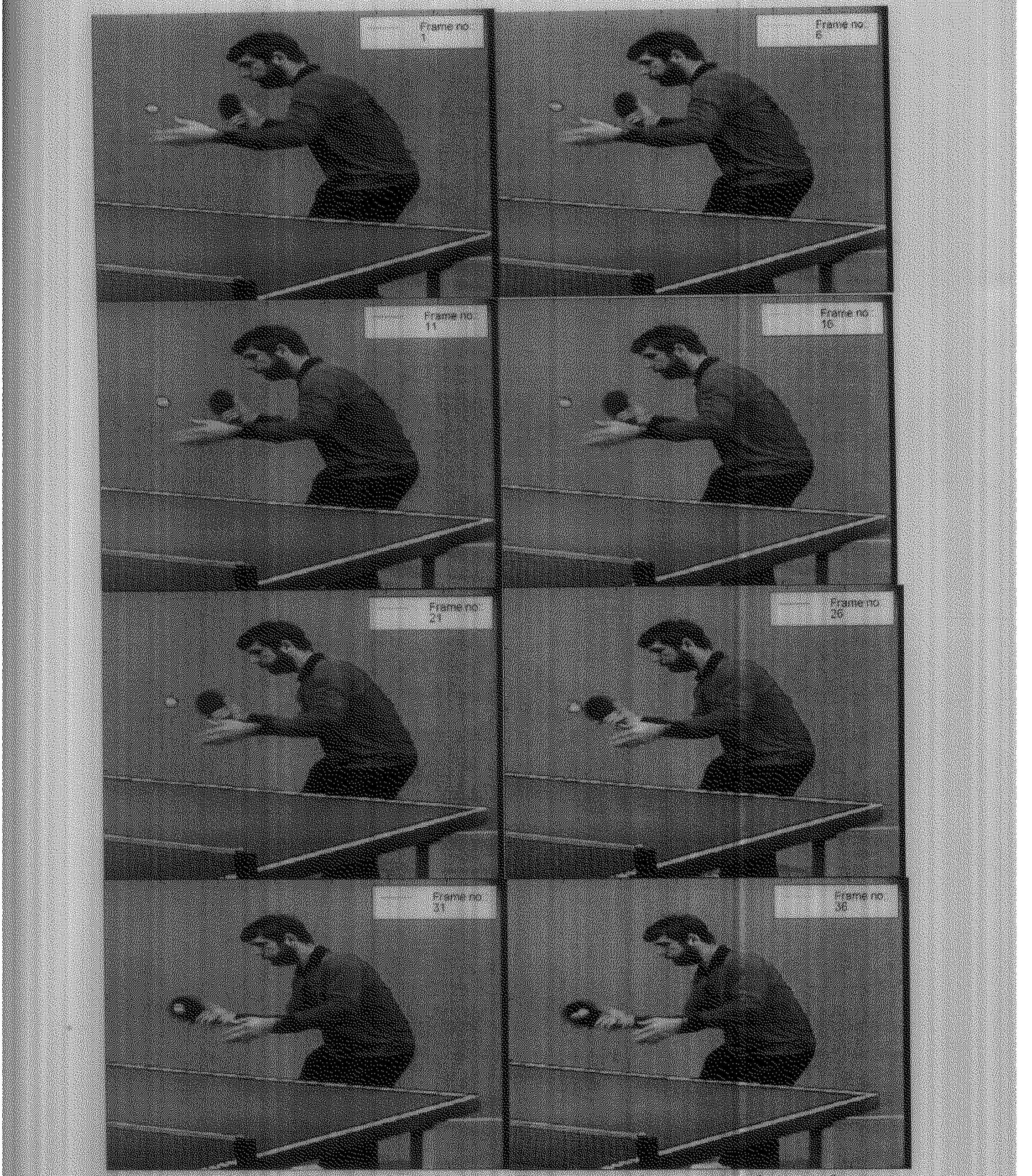


Figure 9: Example of moving ball detection from a typical sequence of image frames. Results of the object detection is shown on frame numbers (1), (6), (11), (16), (21), (26), (31) and (36).


Frame no.: 1
Frame no.: 5
Frame no.: 10
Frame no.: 15
Frame no.: 20
Frame no.: 25
Frame no.: 30
Frame no.: 35

Figure10: Example of moving bat detection from a typical sequence of image frames. Results of the object detection is shown on frame numbers (1), (5), (10), (15), (20), (25), (30), (35), (40), (45) and (50).

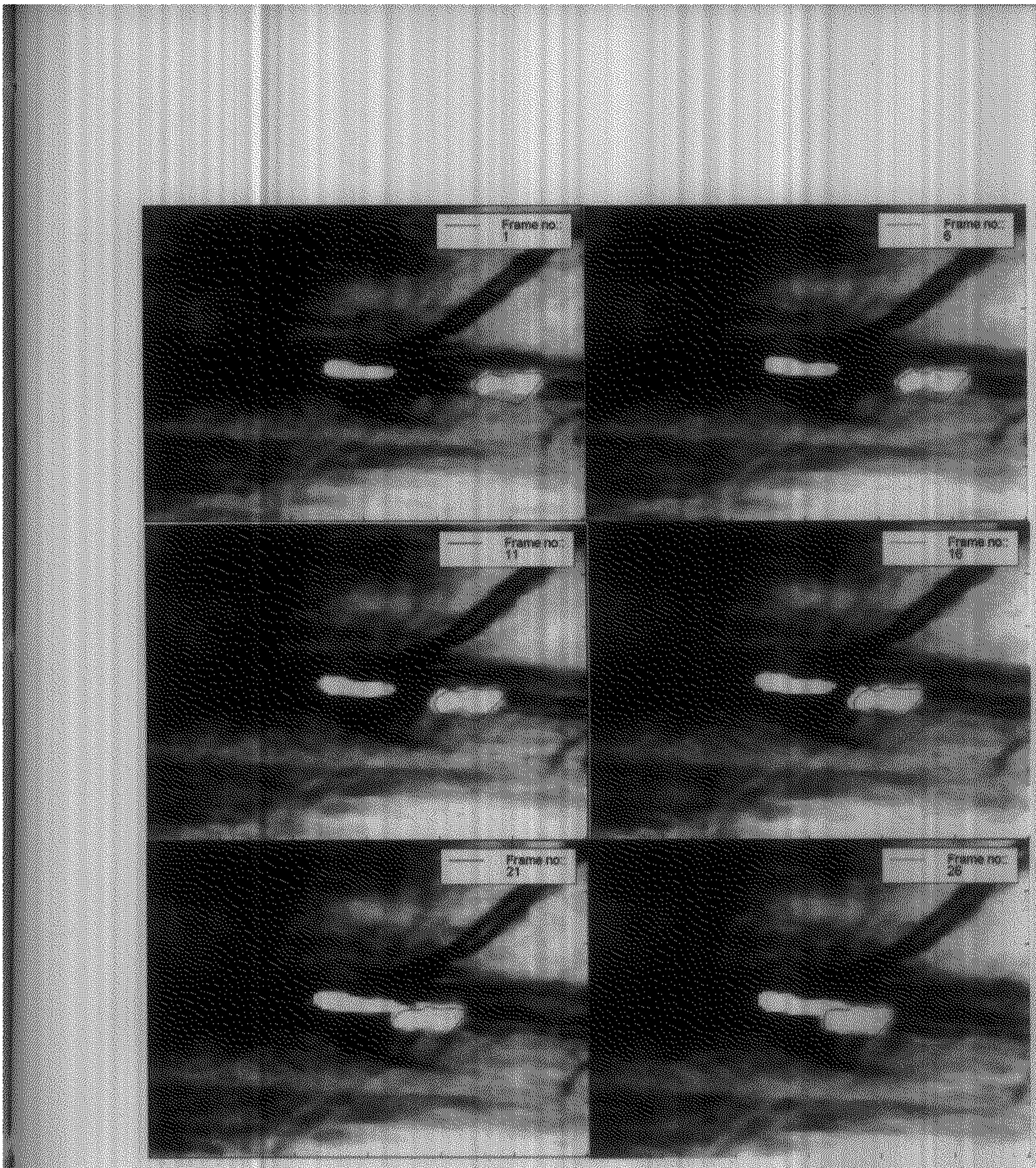


Figure12: Example of moving object detection from a typical sequence frames. Results of the object detection is shown on frame numbers (1), (6), (11), (16), (21) and (26).

Table 2: Results of the ball tracking shown in Figure 10.

From frame no.	To frame no.	λ	No of iteration	Total time(s)
1	2	1	1	1.16
2	3	1	1	1.73
3	4	1	1	1.1
4	5	1	1	1.15
5	6	1	1	1.32
6	7	1	1	1.1
7	8	1	4	4.39

8	9	1	1	1.15
9	10	1	1	1.1
10	11	1	3	3.52
11	12	1	1	1.1
12	13	1	1	1.15
13	14	1	1	1.1
14	15	1	1	1.1
15	16	1	1	1.32
16	17	1	1	1.09
17	18	1	1	1.16
18	19	1	1	1.21
19	20	1	1	1.2
20	21	1	1	1.38
21	22	1	1	1.15
22	23	1	1	1.15
23	24	1	1	1.21
24	25	1	1	1.21
25	26	1	2	2.47
26	27	1	1	1.21
27	28	1	1	1.21
28	29	1	1	1.21
29	30	1	1	1.21
30	31	1	1	1.37
31	32	1	1	1.21
32	33	1	1	1.21
33	34	1	1	1.2
34	35	1	1	1.21
35	36	1	1	1.38
36	37	1	1	1.2
37	38	1	4	4.45
38	39	1	1	1.21
39	40	1	1	1.21
40	41	1	1	1.37

Table 3: Results of the bat tracking shown in Figure11.

From frame no.	To frame no.	λ	No of iteration	Total time(s)
1	2	1	3	12.85
2	3	1	3	10.71
3	4	1	1	3.57
4	5	1	3	10.71
5	6	1	2	7.86
6	7	1	1	3.57
7	8	1	4	14.5
8	9	1	1	3.57
9	10	1	1	3.62
10	11	1	3	11.32
11	12	1	1	3.57
12	13	1	1	4.17
13	14	1	1	3.57

14	15	1	3	12.2
15	16	1	1	4.61
16	17	1	1	5.11
17	18	1	4	16.64
18	19	1	1	4.07
19	20	1	3	12.2
20	21	1	1	4.78
21	22	1	1	3.96
22	23	1	1	4.01
23	24	1	1	4.01
24	25	1	1	4.06
25	26	1	4	16.42
26	27	1	1	4.01
27	28	1	1	4.07
28	29	1	6	23.45
29	30	1	1	3.95
30	31	1	1	4.78
31	32	1	3	12.2
32	33	1	1	4.01
33	34	1	1	4.01
34	35	1	4	15.71
35	36	1	1	4.78
36	37	1	1	4.01
37	38	1	3	12.2
38	39	1	1	4.01
39	40	1	1	4.01
40	41	1	1	4.72
41	42	1	2	7.85
42	43	1	1	3.68
43	44	1	1	3.79
44	45	1	5	17.9
45	46	1	1	4.34
46	47	1	1	3.68
47	48	1	3	10.77
48	49	1	1	3.79
49	50	1	1	3.68
50	51	1	2	7.85
51	52	1	1	3.68
52	53	1	1	3.63
53	54	1	1	3.68
54	55	1	1	3.62
55	56	1	1	4.28
56	57	1	1	3.68
57	58	1	1	3.68
58	59	1	1	3.68
59	60	1	1	3.68
60	61	1	1	4.29
61	62	1	1	3.73
62	63	1	2	7.2
63	64	1	1	3.68

64	65	1	1	3.68
65	66	1	2	7 85
66	67	1	1	3.68
67	68	1	1	3.63
68	69	1	2	7.25
69	70	1	1	3.62
70	71	1	1	4.34
71	72	1	4	14.28
72	73	1	1	3.68
73	74	1	1	3.68
74	75	1	5	17.8
75	76	1	1	4.34
76	77	1	2	7.25
77	78	1	1	3.68
78	79	1	1	3.68
79	80	1	4	14.55
80	81	1	1	4.4
81	82	1	1	3.68
82	83	1	2	7.19
83	84	1	1	3.68
84	85	1	1	3.68
85	86	1	3	11.48
86	87	1	1	3.68
87	88	1	1	3.68
88	89	1	3	10.82
89	90	1	1	3.68
90	91	1	1	4.34
91	92	1	2	7.36
92	93	1	1	3.68
93	94	1	1	3.68
94	95	1	5	17.91
95	96	1	1	4.39
96	97	1	1	3.68
97	98	1	2	7.90
98	99	1	1	3.68
99	100	1	1	3.79

Table4: results of the ball tracking shown in Figure12

From frame no.	To frame no.	λ	No of iteration	Total time(s)
1	2	1	1	3.02
2	3	1	1	3.37
3	4	1	1	3.13
4	5	1	4	12.2
5	6	1	1	3.57
6	7	1	1	3.07
7	8	1	4	12.2
8	9	1	1	3.13
9	10	1	1	3.13

10	11	1	4	12.79
11	12	1	1	3.14
12	13	1	1	3.13
13	14	1	1	3.13
14	15	1	4	12.25
15	16	1	1	3.73
16	17	1	1	3.08
17	18	1	4	12.3
18	19	1	1	3.13
19	20	1	4	12.36
20	21	1	1	3.68
21	22	1	1	3.13
22	23	1	4	12.25
23	24	1	1	3.07
24	25	1	1	3.13
25	26	1	2	6.35
26	27	1	1	3.13
27	28	1	1	3.14
28	29	1	4	12.36
29	30	1	1	3.19

Table 5: results of the object tracking shown in Figure13

From frame no.	To frame no.	λ	No of iteration	Total time(s)
1	2	1	1	1.16
2	3	1	2	2.25
3	4	1	3	3.4
4	5	1	2	2.26
5	6	1	2	2.25
6	7	1	1	1.15
7	8	1	2	2.25
8	9	1	2	2.25
9	10	1	2	2.26
10	11	1	3	3.4
11	12	1	1	1.15
12	13	1	2	2.25
13	14	1	2	2.25
14	15	1	2	2.26
15	16	1	3	3.4
16	17	1	2	2.25
17	18	1	2	2.25
18	19	1	1	1.16
19	20	1	2	2.25
20	21	1	2	2.25
21	22	1	3	3.41
22	23	1	1	1.21
23	24	1	2	2.25
24	25	1	2	2.26
25	26	1	1	1.15
26	27	1	2	2.31

27	28	1	3	3.4
28	29	1	1	1.15
29	30	1	1	1.15
30	31	1	1	1.15

4. Conclusion

We have introduced a new method for object tracking. Three new external forces are derived that can be used in a wide range of active contour method for boundary detection. As described earlier computation time required for this application is much lesser than existing algorithm such as clustering [1]. A major computation burden is in approximating intensity surface. Total computation time increases with increasing object size. We can reduce computation time by calculating normals only at some selected points instead of all the points of the approximating surface. Farther reduction of computation time is possible by sampling the data for the surface approximation. Our experimental results show that our algorithm can work efficiently even in very noisy environment. The algorithm can detect and track a slow as well as relatively fast moving object.

Experimental results show that the algorithm can accommodate minor deformation of the object. But it needs to be investigated how the proposed algorithm can be adopted for moving objects with significant shape deformation and intensity variation. It can be simply said that the closeness of the approximated surface increases with increasing degree of the surface and decreases with increasing size of the object: but their exact relationship is yet to be determined. This algorithm can model rotation of an imaged object with respect to an axis perpendicular to the image plane. but additional information is needed to model intensity variation due to 3d rotation of the actual object. Extension of the algorithm to the color image is straightforward and in that case three surfaces corresponding to three basic colors (R, G and B) need to be used instead of using just single surface for gray level image. But computation burden for this implementation will be higher. Alternatively. the algorithm can be used directly by extracting a gray level image from the color image. The gray level image is extracted by tacking a particular color component from the color image that is the color of the object. For example if the object is colored red with arbitrary background, then out of three component. red component of the color can be used as a gray level image as an input to this algorithm. Thus the image would contain a white object inside a black background. The algorithm is expected to work much better in this implementation.

References

[1] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object tracking with a moving camera". In *IEEE Workshop on Visual Motion*, pages 2–12. 1989.

[2] L. D. Cohen, "On active contour models and balloons." *CVGIP: Image Understand.*, vol. 53, pp. 211–218. Mar. 1991.

[3] C. Xu, and J.L. Prince "Snakes, Shapes, and Gradient Vector Flow", 1997 Conference on Computer Vision and Pattern Recognition (CVPR'97).

[4] Ling Guan, Sun-Yang Kung and Jan Lersen "Multimedia Video and Image Processing", CRC press London, New York, p.223-227.

[5] Berthold Klaus and Paul Horn, "Robot Vision", McGraw-Hill Book Company.1986. p.278-294

[6] B.S.Grewal, "Higher Engineering Mathematics", Kahnna Publishers Delhi, p-588.

[7] B. Kharab and R. B. Guenther, "An Introduction to Numerical Methods, A MATLAB Approach", Chapman & Hall/CRC publication.2001, p.182, 92.

[8] Ye Duan Hong Qin. "Intelligent Balloon: A Subdivision-Based Deformable Model For Surface Reconstruction Of Arbitrary Topology", Proceedings of Sixth ACM Symposium on Solid Modeling and Applications (Solid Modeling'01), pages 47-58. Ann Arbor, Michigan, June 2001.

[9]L.D. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2-D and 3-D images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, p. 1131–1147, 1993.

[10] Dipti Prasad Mukherjee, Nilanjan Ray and Scott T. Acton, "Level Set Analysis for Cell Detection and Tracking", IEEE Transaction paper.