Chapter 14

Detection and Representation of Events in Motion Trajectories

K. Gould, K. Rangarajan, M. Shah

14.1 Introduction

The world we live in changes with time. Our visual system is capable of discovering these changes; in particular we are able to detect moving objects and recover structure from their motion. In our daily life we move our eyes, head and sometimes whole body in order to perceive the changing environment and to interact with the surroundings. We can recognize objects by looking at their shape only, however, recognition tremendously improves if motion information is incorporated as well. Conventional approaches to dynamic scene analysis attempt to recover structure of objects using a sequence of frames, under the assumption that the structure information will be used by a recognition system. However, motion itself has generally not been used explicitly for recognition. Moreover, most approaches to the *structure from motion* problem involve a number of assumptions regarding the objects and their motion, and can only deal with a restricted set of cases with a certain minimum number of points in some minimum number of frames. It is also necessary in these approaches to solve systems of non-linear equations using approximate methods which are very sensitive to noise.

We propose a *different* approach for the use of motion, in which the motion characteristics of moving objects are used without actually recovering the structure. In this approach, we consider extended trajectories followed by the objects. We believe that in many cases, where an object has a fixed and predefined motion, the trajectories of several points on the object may serve to uniquely identify the object itself. Therefore, it should be possible to recognize certain objects based on motion characteristics obtained from trajectories of representative points.

A group of trajectories carry information about the motion as well as the shape of the object they belong to. Therefore, a method which uses trajectory information for recognizing objects will be superior to the method using *only shape* information. Because it will be able to distinguish between objects having the same structure,

but different motion, and between objects having the same motion, but different structure. One isolated trajectory does not carry any explicit information about the object shape, for instance a point on rotating cylinder and a cube will give rise to very similar trajectories. However, a translating cube and rotating cube can always be distinguished, because the spatial relationship between the points in sequential frames as well as the trajectories of points themselves will differ.

One domain of application for a system that recognizes objects based on their trajectories is in a controlled environment such as a factory, where the trajectories of objects can be used to distinguish between stationary and various moving obstacles. This information can then be used in planning a path for a moving piece of machinery. TPS can also be applied to track the motion of stars in the solar system, the trajectory followed by a military target, and chromosome motion in the human body.

In our approach, trajectories will be represented in a manner which will simplify the identification of changes in motion. The representation of the trajectories are analyzed at multiple scales in order to identify important *events* corresponding to discontinuities in direction, speed, and acceleration using scale-space. These important events are recorded in a representation called *Trajectory Primal Sketch* (TPS).

In Section 2, methods for representing trajectories and the changes in motion which occur in the trajectories will be discussed. In order to identify which of these changes are significant, the scale-space of the trajectory representation is calculated. This is discussed in Section 3. Section 4 deals with identifying the primitives of motion as represented by trajectories. We have found that the primitives of motion are the *straight line, circle, ellipse, cycloid, and projectile*. The exact behavior of each of these primitives in the trajectory representations and in scale space has also been determined so that areas of the trajectory which correspond to one of these primitives can easily be identified. Determining the primitive types of unknown trajectories is discussed in Section 5. Section 6 presents some of the results we have obtained. Finally, the composite TPS is discussed in section seven.

14.1.1 Related Work

A great deal of work has been done in the field of psychology to show that people can recognize objects from their trajectories [5, 11]. It has been theorized that humans can recognize an object based on the motion of several points on that object by inferring the three dimensional structure of the object from the transformations the two dimensional image undergoes.

Cutting [2] discusses two similar concepts to be used for recovering the structure of objects from their trajectories. The first concept, which is due to Johansson [4] is that the motion of an object can be separated into two parts, the *common motion* or the motion of the objects as a whole, and the *relative motion* or the motion of the individual parts of the object with respect to the entire object. Cutting defines the term *absolute motion* which is the path of a particular point on the object. The absolute motion of a point is equal to the sum of the common motion of the object and the relative motion of the point. The absolute motion of a point is also its trajectory.

The second concept Cutting discusses is due to Wallach [12] who says that

there are two types of displacements an object can undergo, those relative to the *object* and those relative to the *observer*. These ideas are similar to those of Marr, whose *object centered coordinate system* corresponds to Wallach's displacements relative to the object and whose *viewer centered coordinate system* corresponds to Wallach's displacements relative to the observer. Cutting points out that the perceived center of the object centered coordinate system is critical to correctly identifying the motion of the object and the object itself.

Cutting gives examples of six different types of motion, *rolling wheels, walking people, swaying trees, aging faces, the rotating night sky* and *expanding flow fields*. From experiments with human observers, Cutting concludes that the type of motion determines which coordinate system, object centered or viewer centered, should be used. In the first four examples Cutting suggests that an object centered coordinate system is used by the observers since only one object in each image is moving. In the last two examples, expanding flow fields and the rotating night sky, a viewer centered coordinate system is more appropriate because the entire environment is changing rather than just one object.

Todd [11] is interested in distinguishing between rigid and several types of non-rigid motion such as bending, stretching, twisting and flowing. By displaying the trajectories of either rigid or non-rigid objects, Todd shows that human observers are able to distinguish between the two. Human observers are also able to visualize the three dimensional form of the rigidly moving objects from their trajectories.

Todd explains this by noting that the two dimensional projections of rotating rigid objects are elliptical trajectories, the minor axes of these ellipses lie along a single straight line, and the points must traverse these ellipses at the same frequencies. The purpose of Todd's experiments was to determine what would affect the observer's ability to distinguish between rigid and non-rigid motion, such as whether the axis of rotation is moving or stationary, the frame rate, and the speed of rotation, none of which affected performance significantly, and the number of frames, which did significantly affect the performance of the observers. He also found that some non-rigid motions are easier to detect than others. Non-rigid motions due to differences in relative frequency and orientation are easier to detect than those due to relative eccentricity.

14.1.2 Background - Primal Sketch

The term primal sketch has previously been used in shape representation. Marr introduced the term when he defined the *raw primal sketch* [6]. The raw primal sketch contains primitives which are *edges, bars, blobs* and *terminations*. Each primitive is further described by its orientation, contrast, length, width and position. These primitives represent the information from the zero-crossings of several channels. The raw primal sketch is used to create the *full primal sketch*. This is done by grouping the primitives in the raw primal sketch into tokens and finding the boundaries among sets of tokens. The main idea is to integrate the information from several channels of zero-crossings and identify primitives which correspond to significant intensity changes, and then recursively grouping these changes into boundaries.

A second example of the use of the term primal sketch is Asada and Brady's *curvature primal sketch* [1]. The curvature primal sketch is a method for representing the significant curvature changes on the boundaries of objects. The primitives

are *corner* and *smooth join*, and the *crank, end*, and *bump* or *dent* which are combinations of corners.

We would like to define a new primal sketch, the trajectory primal sketch, which is used not in shape representation, but in motion representation. We will define a set of primitives which will represent the motion characteristics of trajectories, and show how unknown trajectories can be described using these primitives.

14.2 Trajectory Representations

Significant changes in motion refer to a segment of the trajectory where the direction of motion of the point, the speed of the point or both are changing rapidly. In order to identify the significant changes, a representation which contains all the information of the trajectory itself while making the motion analysis simpler is necessary. In other words changes in motion must be accurately represented and easy to identify. Specifically we would like to work with several 1-dimensional functions rather than the 2-dimensional projection of the trajectory. The approach we have taken to find an adequate trajectory representation is discussed below.

14.2.1 Trajectory $\Psi - S$ Curve Representation

Our first approach was to use a representation based on the $\Psi - S$ curve. The $\Psi - S$ curve is typically used in shape representation to define the boundaries of objects. In the case of trajectories, Ψ refers to the direction of motion of a point between two sequential time frames with respect to a horizontal line. S refers to the distance that a point moves between time t_i and t_{i-1}. This is a parametric one-dimensional function where $\sum_i S_i$ is the parameter. $\sum_n S_n$, where n is the number of frames in the trajectory, equals the total distance a point covers in moving from its initial position to its final position.

S and Ψ are calculated by applying the following equations to the trajectory points where (x_i, y_i) are the coordinates of a point in frame i.

$$S = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{14.1}$$

$$\Psi = tan^{-1}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) \tag{14.2}$$

The $\Psi - S$ curve accurately records the shape of the trajectory however two problems exist. First, $\sum_i S_i$ is a difficult parameter to work with. Second, the $\Psi - S$ curve itself does not consider the possible changes in speed of the point. This is a *serious* problem since the speed of a point, and therefore the time at which a significant change in motion occurs, is essential to the understanding of motion. This problem is illustrated in Figure 14.1.

Figure 14.1 shows the trajectories of two different rolling balls which cover the same amount of distance over the same path. The only difference between these two rolling balls is that the ball in Figure 14.1.(b) is moving faster than the ball in Figure 14.1.(a). It takes the ball in Figure 14.1.(a) 377 frames to cover the same distance that the ball in Figure 14.1.(b) covers in only 94 frames. As shown in Figure 14.1.(c) and (d), the $\Psi - S$ curves of the trajectories in Figure 14.1.(a) and (b), respectively, are virtually the same shape. In other words the significant changes in the shape of the two curves occur at the same position on the S axis.

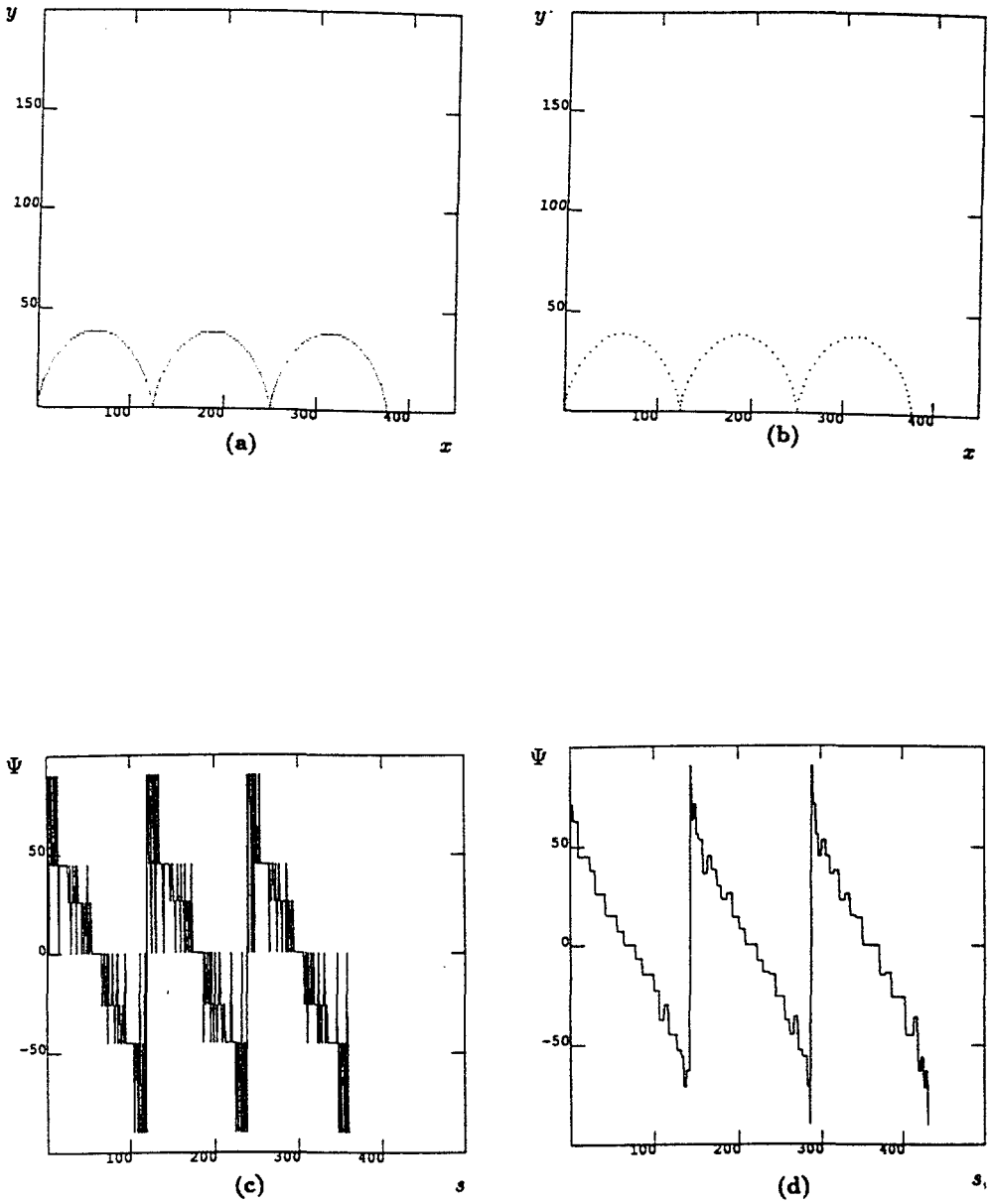


Figure 14.1: Trajectory of one point on (a) A rolling ball through 377 frames, (b) A second rolling ball through 94 frames, (c) $\Psi - S$ curve of ball in (a), (d) $\Psi - S$ curve of ball in (b).

Figure 14.2: A point on a translating object. (a) Case a, (b) Case b. Even though the two motions are significantly different, since there is no change in the direction in both cases, the $\Psi - S$ like representation will not be able to distinguish between them.

Therefore the $\Psi - S$ curve representation is not adequate to distinguish between the two trajectories since a very valuable piece of information is lost, that is time. Another simple example is shown in Figure 14.2. Here the locations of a point on the translating object in five frames is shown for two cases. Even though the two motions are significantly different, since there is no change in the direction in both cases, the $\Psi - S$ like representation will not be able to distinguish between them.

14.2.2 Trajectory Direction and Speed Representations

When dealing with shape, the only relevant information is the direction and length of a line segment on the boundary. However, when dealing with motion, time is an essential piece of information to record. Therefore, instead of considering Ψ as a function of s, as is done in shape representation, two graphs are used. The first one plots Ψ, or the direction of motion, as a function of time. This is known as the *trajectory direction representation.* The second graph plots S, or distance, as a function of time and is known as the *trajectory speed representation.*

$$\Psi = \frac{tan^{-1}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)}{\Delta t} \tag{14.3}$$

$$S = \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{\Delta t} \tag{14.4}$$

To simplify these equations, if t refers to the frame number rather than the actual time, then Δt is always equal to one, and two divisions are eliminated. The trajectory direction and speed representations make the changes in direction and speed very obvious to the human observer. They also make a strong distinction between changes in speed versus changes in direction.

Figure 14.3.(a) shows the direction curve for the trajectory in Figure 1.(a). Its shape is very similar to the $\Psi - S$ curve of that trajectory, except that it is not as long. This is because Ψ, or the direction, is plotted as a function of time rather than distance. If the object moved exactly one pixel each time frame, the $\Psi - S$ curve and the direction curve of that trajectory would be exactly the same since the point would be moving one unit of distance every unit of time. Figure 14.3.(b) shows the speed curve of the trajectory in Figure 14.1.(a). The small peaks in the speed curve are the areas of the trajectory where the ball is moving faster. By comparing the speed curve with the direction curve it is seen that the increase in

speed occurs at the same time as the ball passes over the peaks in its trajectory. The ball is moving faster in these areas because the points in the trajectory are closer together, which means more distance is traveled in that time frame. The direction and speed curves for the rolling ball in Figure 14.1 .(b) are shown in Figure 14.4. These curves have similar shape but the elapsed time is much shorter.

14.2.3 Trajectory Velocity Representations

The trajectory velocity representations are another way to incorporate time into the $\Psi - S$ information. In this case there are again two graphs, the first one called v_x plots the velocity in the positive x direction against the time, the second graph called v_y plots the velocity in the positive y direction against time.

v_x and v_y are calculated from the following equations:

$$v_x = \frac{x_i - x_{i-1}}{\Delta t} \tag{14.5}$$

$$v_y = \frac{y_i - y_{i-1}}{\Delta t} \tag{14.6}$$

As with previous method, if we let t be the time between frames Δt becomes 1, simplifying the equations. Examples of the trajectory velocity representations for the two rolling balls in Figure 14.1 are shown in Figures 14.3 and 14.4. The v_x curves for both balls are cosine curves plus some constant which raises the curve above zero. The v_y curves for both balls are sine curves. Although v_x and v_y for the two different balls have the same general form, the amplitude and the frequency of these two sets of curves are very different.

The trajectory velocity representations record all the information available in the trajectory. Changes in motion are also easily identified from this trajectory representation. Segments of the curve which are fairly horizontal correspond to times when the motion of the point is fairly constant. Segments of the curve where the slope is not zero correspond to times when the motion of the point is changing. The greater the magnitude of the slope the more rapidly the motion is changing.

14.3 Trajectory Primal Sketch

This section describes a method for converting the trajectory velocity representations into the Trajectory Primal Sketch. The TPS must be a compact representation of the significant changes in motion. We create the first level of this compact representation by identifying the changes at various scales. This results in a set of contours known as *TPS contours*, where each contour corresponds to a change in motion. The important information contained in the TPS contours is the position or frame number at which a contour occurs, the height or strength of the TPS contour and the shape of the contour. Calculating the TPS contours, and extracting these important values from the contours are explained in the next two sections.

14.3.1 Calculating TPS Contours

Once the trajectory representations have been calculated, the changes in motion which appear as discontinuities in these representations need to be identified. This

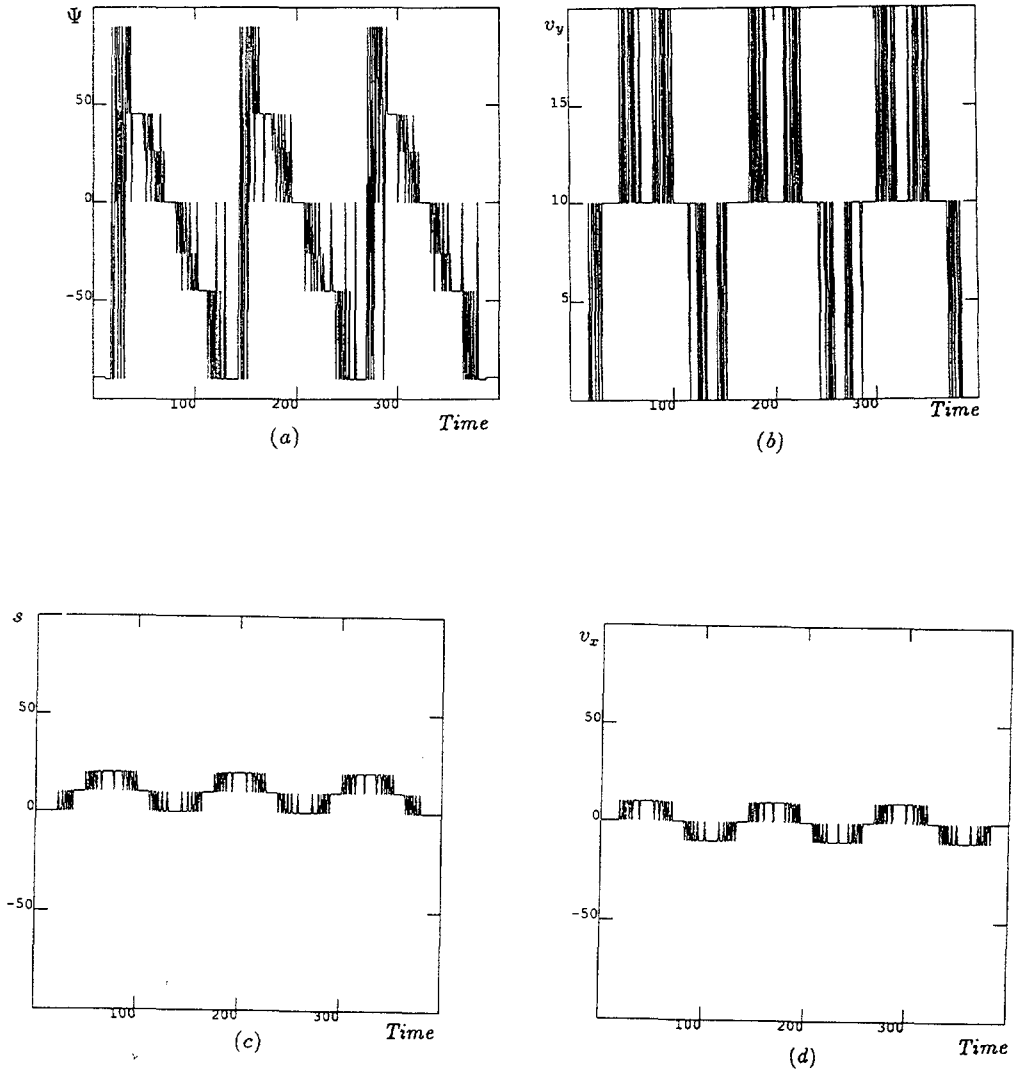


Figure 14.3: For rolling ball of Figure 1.a (a) Direction curve. (b) Speed curve. (c) v_x curve. (d) v_y curve.

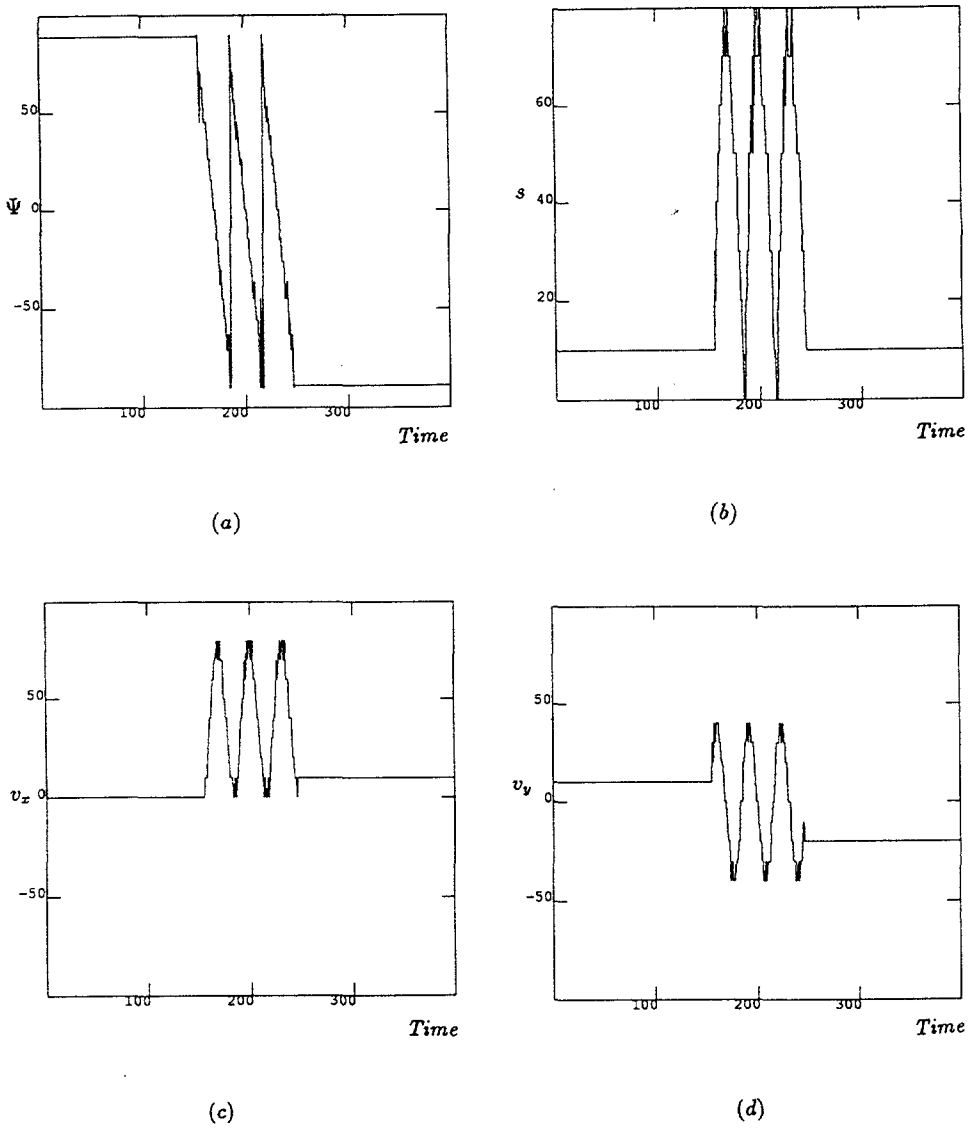


Figure 14.4: For rolling ball of Figure 1.b (a) Direction curve. (b) Speed curve. (c) v_x curve. (d) v_y curve.

is done by studying the scale-space of the trajectory representation [14, 10] which we call *TPS contours*. Scale space is used because the discontinuities may appear at various scales depending on the physical motion of the point.

The *TPS contours* are calculated by applying a mask to the input signal. This mask is created using the Gaussian and its derivatives, where the value of σ can vary between one and fifteen. The size of the mask is dependent on σ. Since the Gaussian approaches zero after 3σ, the size of the mask should be approximately $2 * (3 * \sigma)$. We have used a mask size which is equal to $6 * \sigma + 1$ to insure the size of the mask is odd. The results of applying this mask are then checked for zero-crossings which correspond to discontinuities at a particular scale. TPS contours are the curves which result from plotting the position of a zero-crossing, or the frame at which a zero-crossing occurred, on the horizontal axis and the value of σ on the vertical axis. An example of a typical trajectory and scalespace contours related to its v_x, and v_y are shown in Figure 14.5. It is clear from this Figure that nature of the contours is complex, there are large number of contours at the lower scale, which do not survive at higher scales. Therefore a method is needed to identify contours corresponding to important events.

14.3.2 Parsing TPS Contours

The TPS contours referred to in the previous section were represented as a simple list of coordinates, $(frame, \sigma)$, which correspond to the frame number and the value of σ at which a zero-crossing occurred. This list does not show any relation between the various coordinates, such as which zero-crossings lie on the same contour. In order to use this data to characterize motion we would like to link these unrelated zero-crossings and determine a representative location, strength and shape for each contour. Witkin refers to this process as Coarse-to-Fine tracking [14].

We begin by creating a two dimensional array in which elements that correspond to zero-crossings will have a value of one, all other elements will have a value of zero. This array will be used to link the zero-crossings into contours by tracking relatively adjacent elements of the two dimensional array with a value of one.

Tracking begins at the smallest σ value rather than at the largest σ value as in coarse-to-fine tracking. We have found that in some cases it is impossible to track a contour entirely from one endpoint to the other. Therefore we will start tracking at the smallest scale since the location of the zero-crossing is most accurate at this scale, and the accurate location of the zero-crossing is extremely important. After the two dimensional matrix is created with the frame number on its horizontal axis and σ on its vertical axis, the first zero-crossing is located. Next, we begin to track the contour which may begin with this zero-crossing by checking a neighborhood around this element for another zero-crossing. This neighborhood is shown in Figure 14.6. The current zero-crossing is marked by an $*$ in this Figure. The numbers refer to the order in which the adjacent elements are checked for the next zero-crossing. This tracking process continues until there are no zero-crossing in the neighborhood of the current zero-crossing. Then we return to the smallest value of σ and search for the next zero-crossing which may mark the beginning of another contour.

While performing this tracking process, three values are calculated which describe the contours *location, strength* and *shape*. The first value, the *location* of the contour, is simply the frame number where the contour originated. The *strength*

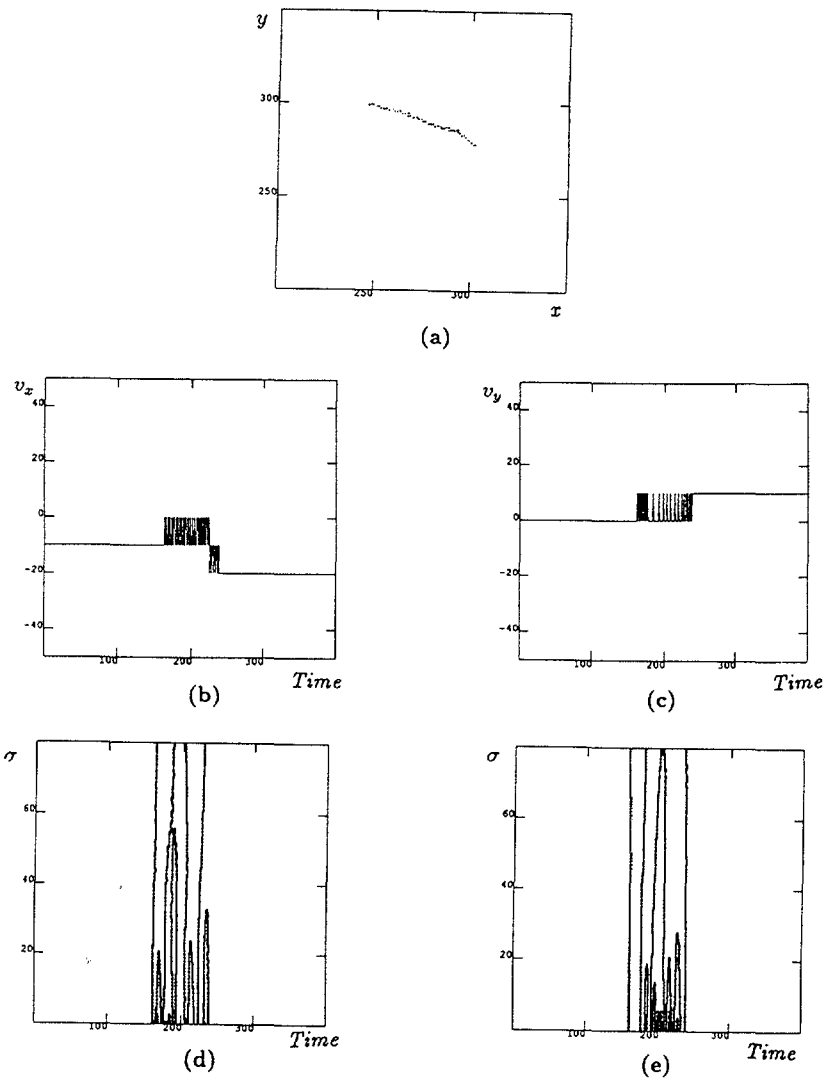


Figure 14.5: A typical trajectory and its representations. (a) Trajectory, (b) v_x, (c) v_y, (d) scalespace of v_x, (e) scalespace of v_y.

7	5	4	6	8
	2	1	3	
		*		

Figure 14.6: Neighborhood for tracking contours.

of the contour is the number of zero-crossings that belong to that contour. The value which represents the *shape*, or the *straightness*, of the contour is calculated by summing the distance between each zero-crossing as it is linked to the next zero crossing. For example if the next zero-crossing is in region one, then the distance equals one, or if the next zero-crossing is in region seven, then the distance equals $\sqrt{8}$. After the entire contour has been linked this sum is divided by the $strength - 1$. A contour whose next zero-crossing was always found in region one would have a *shape* value equal to one. The greater the shape value is the more curved and irregular the contour is.

Only the contours which survive over a large percentage of the range of σ should be considered to correspond to significant changes in motion. For our experiments we required that the contours survive over 60% of the σ values. The shape value could also be used to pick contours that represent significant changes in motion, since drastic changes in motion result in trajectory representation segments which are similar to step edges, and the scale-space model of a step edge is known to be a straight line [10]. The TPS contains these three values for all contours which survive.

14.4 Primitive Trajectories

In this section, we will present a number of *primitive* trajectories which can be written as analytical expressions. We will compute the expressions for their v_x, and v_y, and the *trajectory primal sketch contours*. For each primitive trajectory, we will identify important *events* in its TPS. The aim is to express an arbitrary trajectory as a composition of these primitives. We will consider four main types of motion: translation, rotation, projectile and cycloid.

14.4.1 Translation

The trajectory corresponding to translation can be expressed as a sum of straight lines. We know that the equation of a straight line is given by:

$$y = mx + c \tag{14.7}$$

where m, and c are the slope and y-intercept respectively. Now, incorporating the *time* in the above equation we can rewrite it as:

$$x = at \tag{14.8}$$

$$y = mat + c \tag{14.9}$$

Differentiating x, y with respect to t we get the expressions for v_x and v_y:

$$v_x = a \tag{14.10}$$
$$v_y = ma. \tag{14.11}$$

The sum of n such straight lines for a translation trajectory are given by:

$$x = \sum_{i=1}^{n} a_i t \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.12}$$
$$y = \sum_{i=1}^{n} (m_i a_i t + c_i) \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.13}$$

where $U(t)$ is the unit step function, defined such that $U(t) = 1$ for $t > 0$ and $U(t) = 0$ otherwise. Now, v_x, v_y are given by:

$$v_x = \sum_{i=1}^{n} a_i t \left(\delta(t-t_i) - \delta(t-t_{i+1})\right) + \sum_{i=1}^{n} a_i \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.14}$$
$$v_y = \sum_{i=1}^{n} (m_i a_i t + c) \left(\delta(t-t_i) - \delta(t-t_{i+1})\right) + \sum_{i=1}^{n} (m_i a_i) \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.15}$$

where $\delta(t)$ is the impulse function. Now, the TPS contours are the loci of zero-crossings of the following expression:

$$g^{\sigma}_{tt} * v_x = g^{\sigma}_{tt} * \sum_{i=1}^{n} a_i t \left(\delta(t-t_i) - \delta(t-t_{i+1})\right) + \sum_{i=1}^{n} a_i \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.16}$$
$$= \sum_{i=1}^{n} a_i \left(t_i\, g^{\sigma}_{tt}(t_i) - t_{i+1}\, g^{\sigma}_{tt}(t_{i+1})\right) + \sum_{i=1}^{n} a_i \left(g^{\sigma}_{t}(t_i) - g^{\sigma}_{t}(t_{i+1})\right) \tag{14.17}$$

where $g_t{}^{\sigma}$, g^{σ}_{tt} are the respectively the first and second derivatives of Gaussian with zero mean and standard deviation σ ($g^{\sigma}(t) = e^{-\frac{t^2}{2\sigma^2}}$). And the TPS contours for v_y is given by:

$$g^{\sigma}_{tt} * v_y = g^{\sigma}_{tt} * \sum_{i=1}^{n} (m_i a_i t + c) \left(\delta(t-t_i) - \delta(t-t_{i+1})\right) + \sum_{i=1}^{n} (m_i a_i \left(U(t-t_i) - U(t-t_{i+1})\right) \tag{14.18}$$
$$= \sum_{i=1}^{n} (m_i a_i\, t_i + c)\, g^{\sigma}_{tt}(t_i)) - (m_i t_{i+1} + c)\, g^{\sigma}_{tt}(t_{i+1})$$

$$+ \sum_{i=1}^{n} m_i a_i \left(g_t^\sigma(t_i) - g_t^\sigma(t_{i+1})\right) \tag{14.19}$$

In Figure 14.7.a we have shown an example of translation motion of a point which is moving from location A to E in time $t_E - t_A$. Between locations A, and B the point is moving with the constant speed in the same direction. At location B the speed is changed instantaneously, but the direction remains the same, the point continues to move with the same speed and direction up to location C where the direction changes instantaneously but the speed remains the same. The point continues with the constant speed and in the same direction up to location D, where there is a sudden change in the speed as well as in the direction. The TPS for this trajectory contains the location, strength and shape of the TPS contours which correspond to significant changes in motion. The locations of these significant changes in motion are is shown in Figure 14.7.a . We have indicated the discontinuities in v_x by '+', and the discontinuities in v_y by 'X'. It is clear that the points B, C, and D are identified as events in the trajectory. At B we observe the discontinuity in v_x, while at C and D we have discontinuities in both v_x and v_y.

14.4.2 Rotation

The trajectory corresponding to the rotation around a *fixed* axis can be expressed by an *ellipse*. The ellipse is defined as follow:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{14.20}$$

where a, and b are the major and minor axes respectively. By incorporating time and writing this equation separately for v_x and v_y we get:

$$x = -a * \cos(\omega * t) \tag{14.21}$$

$$y = b * \sin(\omega * t) \tag{14.22}$$

where ω is frequency. Figure 14.8, shows the parametric representation of ellipse. The angle of rotation $\theta = \omega * t$. Notice, that for $a = b = r$ the above equations reduce to equations for a *circle* of radius r around the origin. The TPS contours of this trajectory can be obtained by convolving the above equations with g_{tt}^σ:

$$g_{tt}^\sigma * v_x = -g_{tt}^\sigma * (-a\omega)\sin\omega t \tag{14.23}$$

$$= -a\omega \sin\omega t \tag{14.24}$$

since there is no effect of convolving Gaussian with the sine function. Similarly, we get the following for v_y:

$$g_{tt}^\sigma * v_y = g_{tt}^\sigma * b\omega * \cos\omega t \tag{14.25}$$

$$= b\omega \cos\omega t \tag{14.26}$$

Therefore, in the interval 0— 2π the TPS contours of v_x will be three straight lines one each at $0, \pi, 2\pi$ for ωt. While the TPS contours of v_y will be two straight lines one each at $\frac{\pi}{2}, \frac{3\pi}{2}$. One example trajectory of a point rotating about a fixed axis is shown in Figure 14.7.b. Four events corresponding to this trajectory are also shown. The endpoints of the major axis are identified as significant changes in v_x.

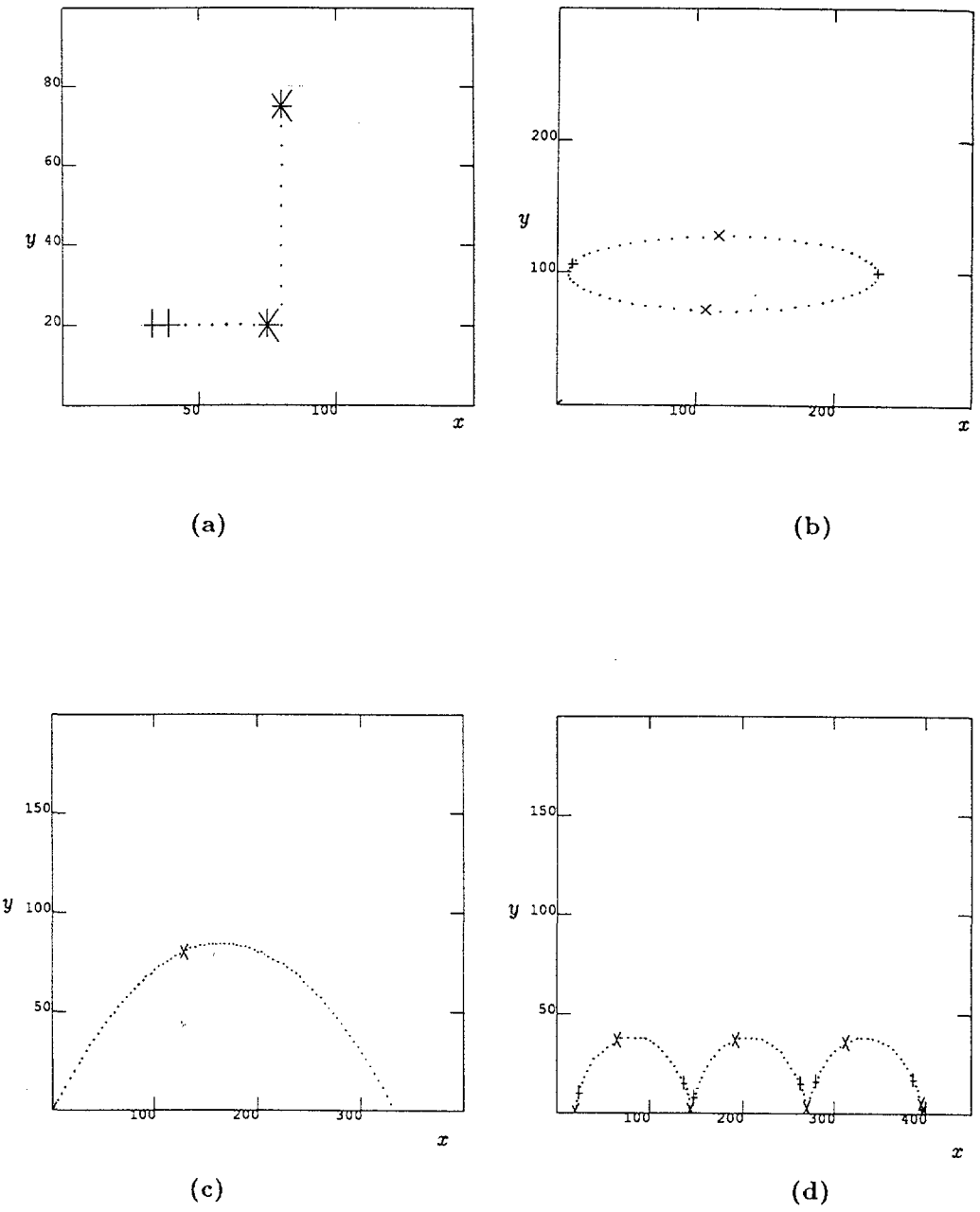


Figure 14.7: Examples of primitive types. (a) Translation, (b) Rotation, (c) Projectile, (d) Cycloid.

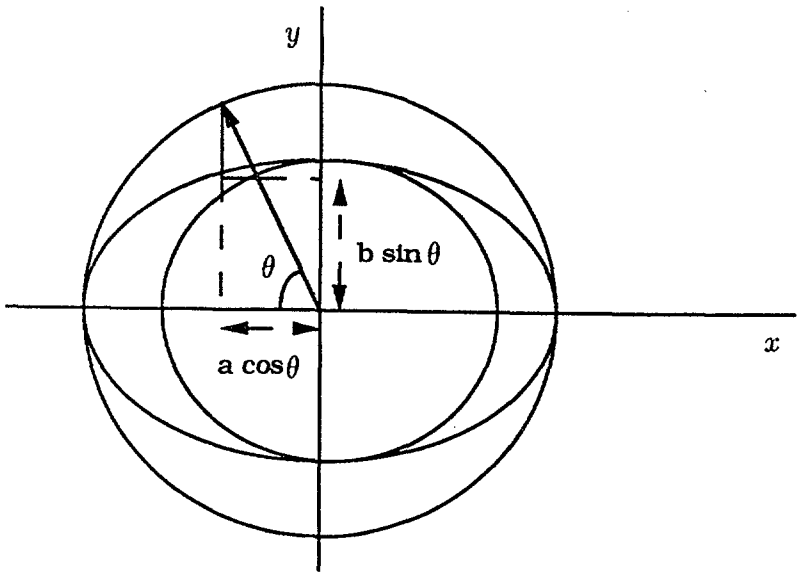


Figure 14.8: Parametric representation of an ellipse.

These are the points where, due to the direction of rotation (counter-clockwise), v_x makes the transition from positive values to negative on the right side of the ellipse and from negative to positive values on the left side of the ellipse. Similarly, at the endpoints of the minor axis, v_y changes from positive to negative values on the top and from negative to positive on the bottom. If the object were to rotate in the clockwise direction, the positions of the events would remain the same, but the positive to negative and negative to positive transitions would be exchanged.

14.4.3 Projectile

The trajectory followed by an object in the space due to *gravity* is called projectile motion, and it is defined as follows:

$$x = v_0(\cos\alpha)\,t \tag{14.27}$$

$$y = v_0 \sin\alpha\, t + \mathcal{G}\frac{t^2}{2} \tag{14.28}$$

where v_0 is initial velocity, $\mathcal{G}$ is the acceleration due to gravity, and α is the angle. Differentiating above equations with respect to t we get:

$$v_x = -v_0 \cos\alpha \tag{14.29}$$

$$v_y = v_0 \sin\alpha + \mathcal{G}\,t \tag{14.30}$$

Figure 14.7.c shows a synthetic trajectory and its TPS for projectile. A sole event in this trajectory is a change in v_y which is indicated by an 'X'. This event occurs at the position where the values of v_y shift from positive changes to negative changes.

14.4.4 Cycloid

The trajectory followed by the object which is rotating around a translating axis is defined by a cycloid. The cycloid is defined as:

$$x = a(\omega t - \sin \omega t) \quad (14.31)$$

$$y = a(1 - \cos \omega t) \quad (14.32)$$

where a is constant, and ω is the frequency. The expressions for v_x, and v_y are:

$$v_x = a(\omega - \omega \cos \omega t) \quad (14.33)$$

$$v_y = a\omega \sin \omega t \quad (14.34)$$

We have shown in Figure 14.7.d a trajectory corresponding to cycloid motion. This trajectories might be obtained from a point on the rim of a rolling wheel. This trajectory contains both types of discontinuities. The changes in v_y, indicated by an 'X', occur at the top of each peak and in between peaks. These are the positions where v_y changes from positive to negative or negative to positive. The changes in v_x occur between the peaks on both sides of the v_y event. This is due to a change in speed as each point approaches the bottom of the peak. The first '+' in each pair marks a slowing down in the positive x direction, while the second '+' marks a speeding up of the point in the positive x direction.

14.5 Determining Primitive Type

The *trajectory primal sketch* contains the location, strength, and shape of the TPS contours for the discontinuities in velocity as well as the discontinuities in the first derivative of velocity i.e. the discontinuities in acceleration. From the analysis and examples in section 4, it is clear that the first derivative discontinuities of both the rotation and cycloid have a sine and cosine relationship to each other. Therefore the first step in identifying primitive types is to separate all the trajectories into two categories. The first category or the *rotation/cycloid group* will contain all trajectories which exhibit a sine/cosine relationship in the first derivative. Everything else will be placed into the *translation group*.

The rotation/cycloid group can be broken down further into either rotation or cycloid motion by examining the discontinuities in velocity as well as the discontinuities in the first derivative of velocity. As predicted in the analysis of cycloid motion, the v_x graph for cycloid motion will be shifted some value above or below the zero axis. So when the velocity discontinuities for v_x are calculated, the scale space contours will not look like a cosine curve as they did with the acceleration discontinuities. Therefore v_x and v_y will not have a sine/cosine relationship. However, the scale space of v_x and v_y of a rotating trajectory will have a sine/cosine relationship. Rotation and cycloid motion can be differentiated by checking for a sine/cosine relationship in the velocity discontinuities. If it exists, then the trajectory is an example of rotation, otherwise it is a cycloid.

The trajectories placed into the translation group can also be further classified. The trajectories themselves will be broken down into segments, and each of these segments will be classified as either *straight line translation* or *curved translation*. The value of the slope of the velocity curve is the same as the value of the acceleration of the point within that segment. When a point is moving in a straight

line, the acceleration of v_x and v_y must be the same. However, when a point is turning steadily, i.e., moving in a curved line, then the acceleration of v_x and v_y will be different. The slope of v_x and v_y within a segment are compared to see if they are similar to each other within a reasonable bound; if they are then the corresponding segment of the trajectory is labeled as straight line translation. If the values of the slopes are significantly different from each other, then the segment of the trajectory is labeled as curved translation.

The segmentation is accomplished by using the locations of the TPS contours of both the v_x and v_y graphs of a trajectory as the end points of each segment. The slope of v_x and v_y are calculated within each segment using the values of the velocity graphs at each of the end points of the segment. The slope of v_x and v_y are then compared to determine if the translation is straight line or curved.

Using the locations of significant changes in motion found in the TPS and this method, the primitive type of all the trajectories in this paper have been correctly determined.

14.6 Examples

In this section we will present results analyzing the TPS for a number of trajectories obtained from real scenes. Since the trajectories from the real scenes were given for a small number of frames, we used interpolation to explode the original trajectories to a large number of frames. We assumed that the motion between any two consecutive frames was in a straight line. In fact, one can compute the extended trajectories with a large number of frames from a moving sequence without using any assumptions. Here, we do not consider that to be any significant factor in our results.

Figure 14.9 shows four trajectories from the Superman sequence which was used by Sethi and Jain [9]. The TPS for one of the trajectories is shown in Figure 14.9.(d). This particular trajectory is made by tracking a point on the head of the leftmost man in the scene. Note that all important events, i.e. the places where the direction or speed of the man changes, in this trajectory are captured by the TPS. The changes occur as the man moves both toward the camera and slightly to the right. The changes in v_x occur as the man makes small changes in direction and changes in speed. The changes in direction in v_y correspond to the man's head bobbing up and down as he runs toward the camera. There are also changes in speed. Figure 14.10 shows the v_x and v_y and corresponding scalespace contours for this trajectory.

Next, the results for the Blocks scene are shown in Figure 14.11. In this scene four different blocks are moving towards the center of the image. Trajectories of points on two of these block are shown in Figure 14.11.(a). These two block are a rectangular shaped block which travels from left to right across the image, and a triangular shaped block which moves from the upper right corner of the image to the center of the image. The TPS of a point on the rectangle is shown in Figure 14.11.(b). The two blocks whose trajectories are not shown are a rectangle which does not move and a triangle which moves from the bottom center of the image to the top center. The Figure 14.12 shows the results for Object scene. This scene consists of three rectangular blocks, one begins at the lower left corner of the image, the second begins in the lower right, and the third begins at the top center of the

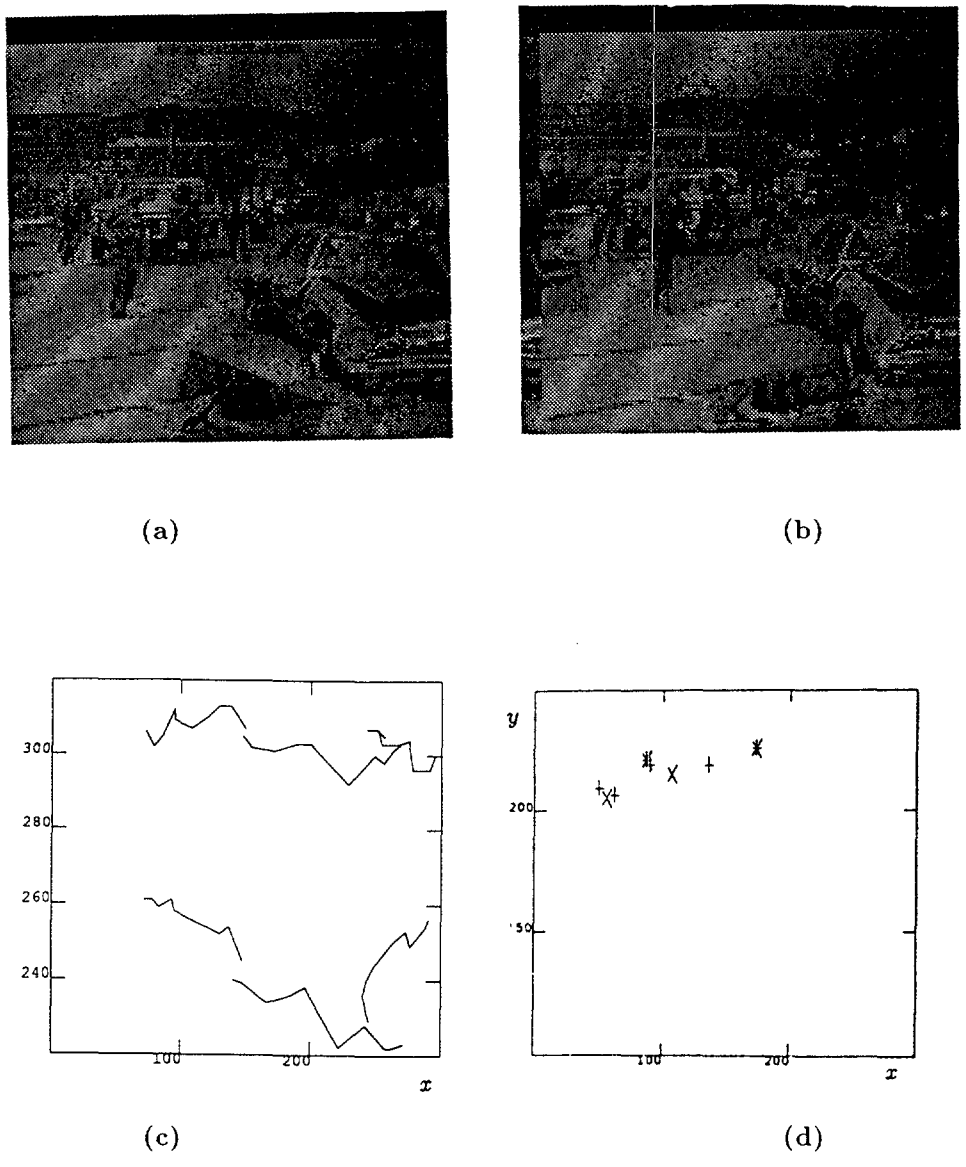


Figure 14.9: Superman Sequence (a)-(b). Two frames of sequence, (c). Trajectories, (d). TPS of S1 trajectory.

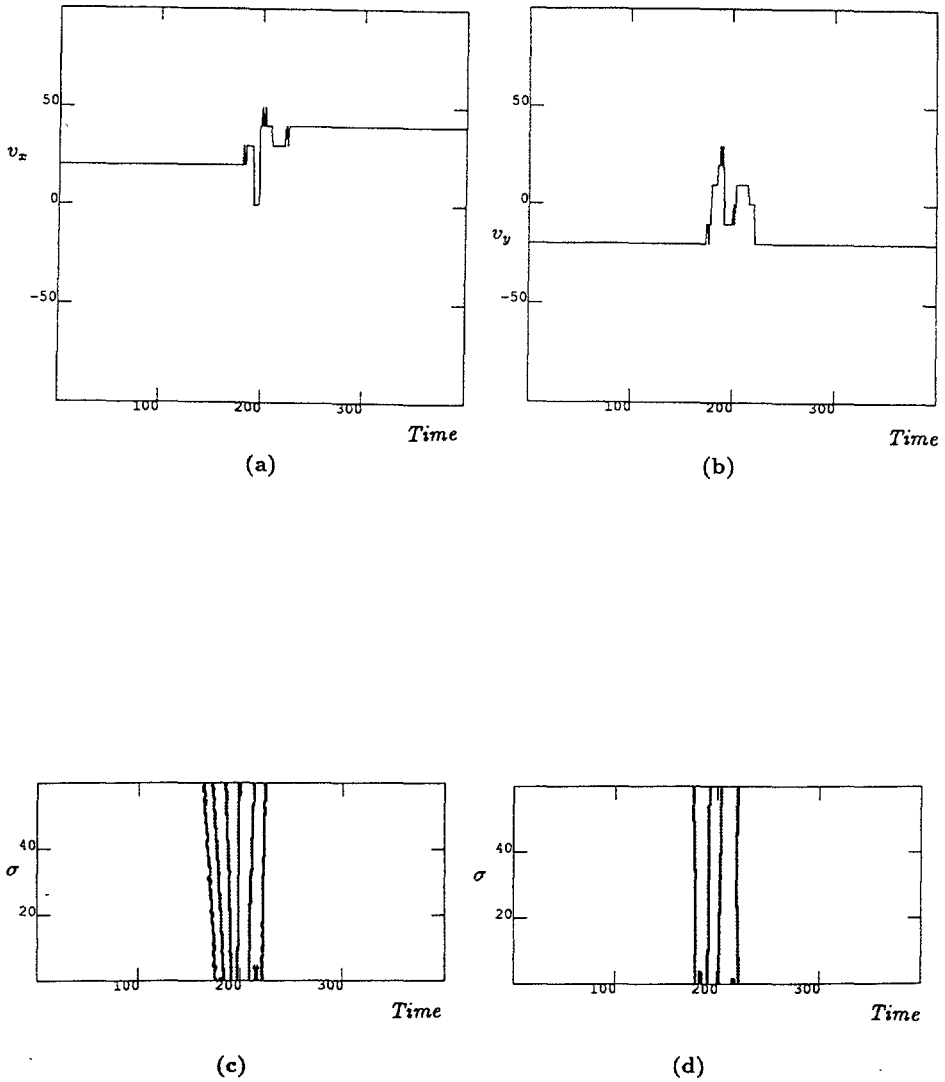


Figure 14.10: Trajectory Velocities of S1 and their scalespaces. (a). v_x, (b). v_y, (c). Scalespace of v_x, (d). Scalespace of v_y.

image. All three of the blocks move toward the center of the image in a straight line, they meet in the middle and then continue away from each other. Trajectories of three points on the block which begins at the lower left corner of the image are shown in Figure 14.12.(a). The TPS of one of these trajectories is shown in Figure 14.12.(b).

Figure 14.13 contains the nine trajectories of the Card sequence and three images from this sequence. This sequence consists of two objects moving from right to left and slightly upwards. Figure 14.13.(e) shows the TPS of one trajectory on the top object, and (f) is the TPS of a trajectory on the lowest object. Both these objects are translating, but each has several velocity changes in both x and y.

Figure 14.14 shows the trajectories obtained from the *Walker* sequence. This sequence was obtained from a program reported by Cutting [3]. In this sequence a person is shown walking. Eleven points representing the head, right shoulder, left and right knees, left and right elbows, right hip, left and right wrists, and left and right ankles of a person are tracked through ten frames. The trajectory is then expanded to obtain 99 frames using the interpolation method described earlier. The entire body moves up and down in a slight bouncing manner. The movements of the shoulder and hip are ellipsoidal and the movement of arms and legs are pendular. Figure 14.14.(c) shows the TPS for one foot of the walker while (d) show the TPS for an elbow. The trajectory of the foot changes velocity in both the x and y direction. The velocity of the elbow changes in the x direction with only one change in the y direction. Figure 14.14.(e) shows the TPS for one wrist of the walker. This trajectory has several velocity changes in the x direction, and two changes in the y direction where the wrist move up slightly and then down again.

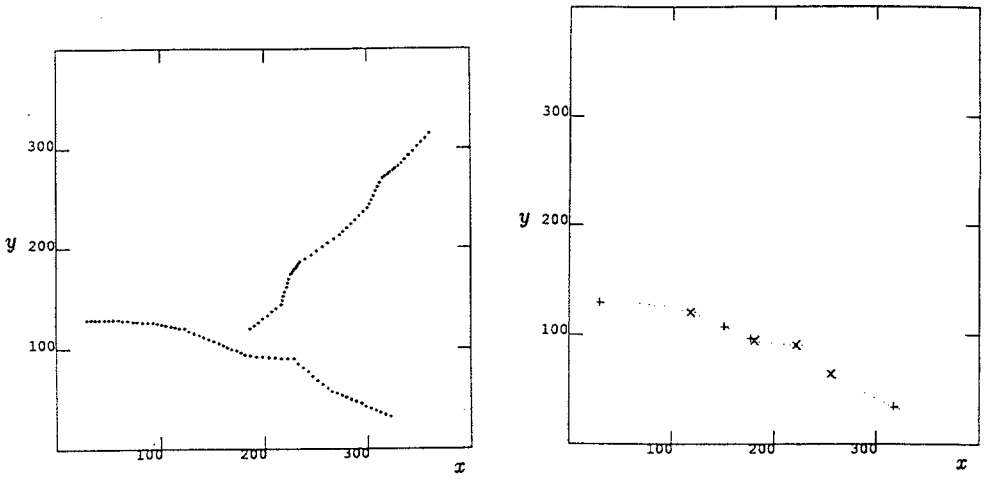
The primitive type of all the trajectories in this section is translation, and based on the information in the TPS, they are classified as translation by our procedure for determining primitive types.

14.7 Composite Trajectory Primal Sketch

So for we have been dealing with individual trajectories. There can be several trajectories belonging to a single object, we would like to create a composite representation which captures the commonalities between trajectories belonging to the same object. This representation, we shall refer to as *CTPS, Composite Trajectory Primal Sketch.* A representation of a set of data is complete, if the data can be regenerated from the representation with a desired accuracy. In the case of CTPS, the input data is the coordinates of the feature points in the frames taken at different instants. Therefore, we will also present a reconstruction scheme for translation and rotation cases.

14.7.1 Translation

When the objects are quite far away from the camera, orthographic projection can be assumed. Under orthographic projection, trajectories of points on an object that undergoes translation only are identical and hence have identical sets of events in their TPS. The events could be either in v_x or v_y or both. The CTPS representation stores the events on one of the trajectories and the spatial coordi-



(a) (b)

Figure 14.11: Block Scene Object-A (a). Trajectories, (b). TPS of Block1

nates of all the tracked points in the first frame. Each event is characterized by its location i.e the frame number of its occurrence, and average v_x and v_y at that instant. The average v_x at the instance of an event is calculated by dividing the displacement in x between this event and the immediate next event by the time interval between them. The average v_y at the instance of an event is calculated by dividing the displacement in y between this event and the immediate next event by the time interval between them. The CTPS representation also stores an initial event located in the first frame, in addition to the observed events. With this information, we will be able to recreate the entire sequence for all the points on the object.

Figure 14.15 shows the CTPS structure for the translation case. Figure 14.16.a shows one of the trajectories with the TPS events marked on it. The trajectory of the Cube has been chosen so as to include all possible combinations of events in v_x and v_y. Initially the Cube is traveling along the x direction with a velocity of $v_x = 3$ and $v_y = 0$, and in frame 21 it changes its direction and moves along the y axis with $v_x = 0$ and $v_y = 2$. This is a speed change in both x and y directions and is accompanied by a direction change. This change is registered as an event in both v_x and v_y in TPS. In frame 51, cube increases its speed alone i.e. $v_x = 0$, $v_y = 4$, without changing its direction. This shows up as an event in v_y alone. In frame 81 the Cube changes its velocity to $v_x = 2$, $v_y = 4$ with a direction change. This is registered as an event in v_x only. Next, in frame 111 the Cube changes its velocity to $v_x = 4$, $v_y = 0$ with a direction change, this shows up an event both in v_x and v_y. In frame 146 it changes its velocity to $v_x = 6$, $v_y = 0$, which is an event

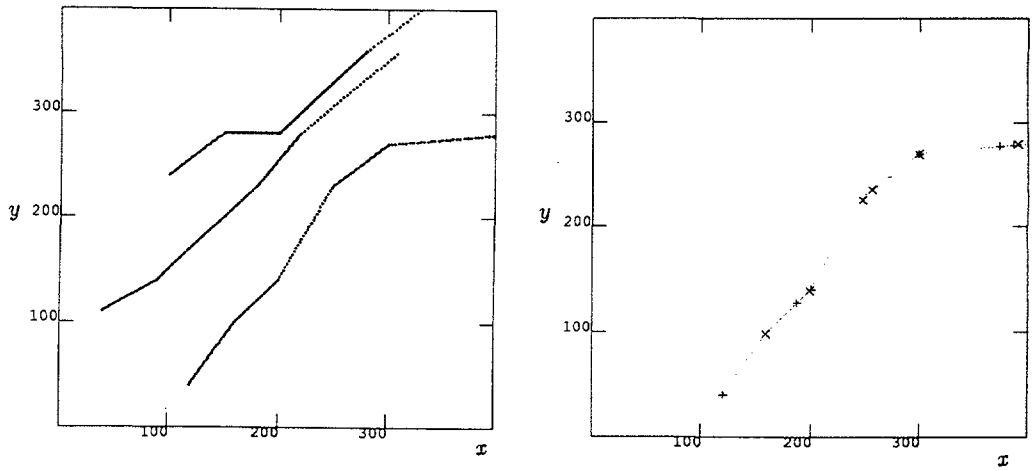


Figure 14.12: Block Scene Object-B (a). Trajectories, (b). TPS of OA1

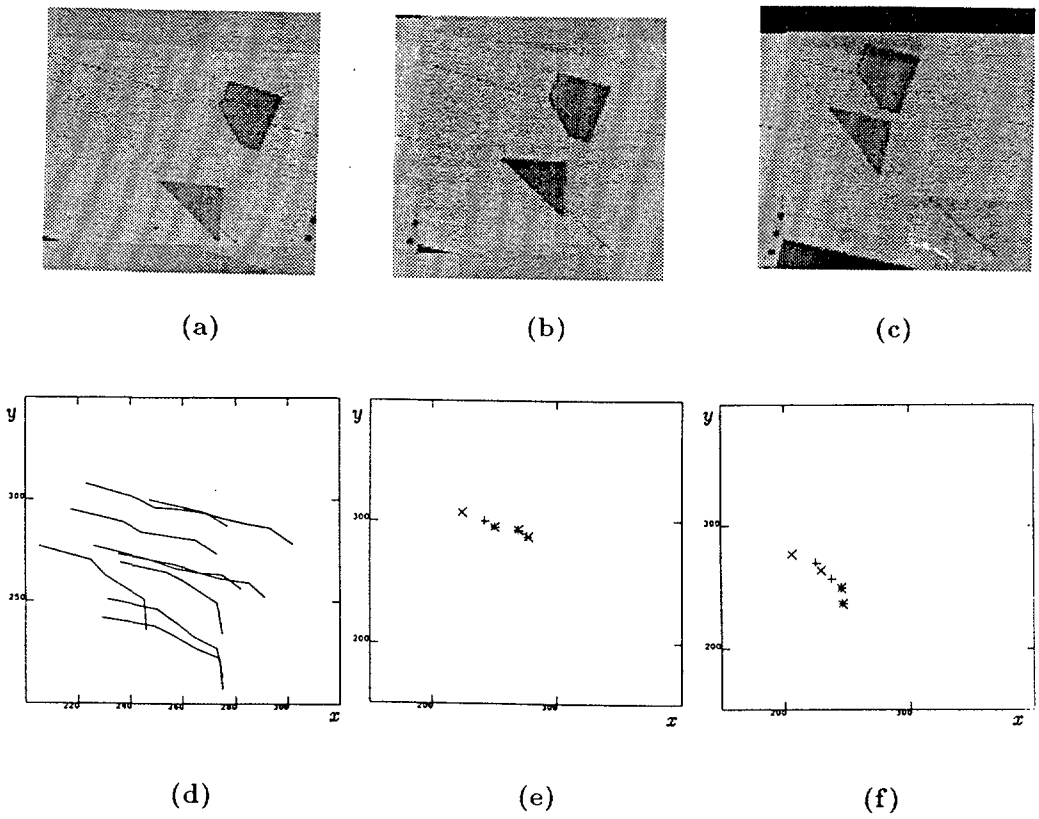


Figure 14.13: Card Sequence. (a) Frame-1, (b) Frame-3, (c) Frame-5, (d) Trajectories, (e). TPS of Block1, (f) TPS of Block2.

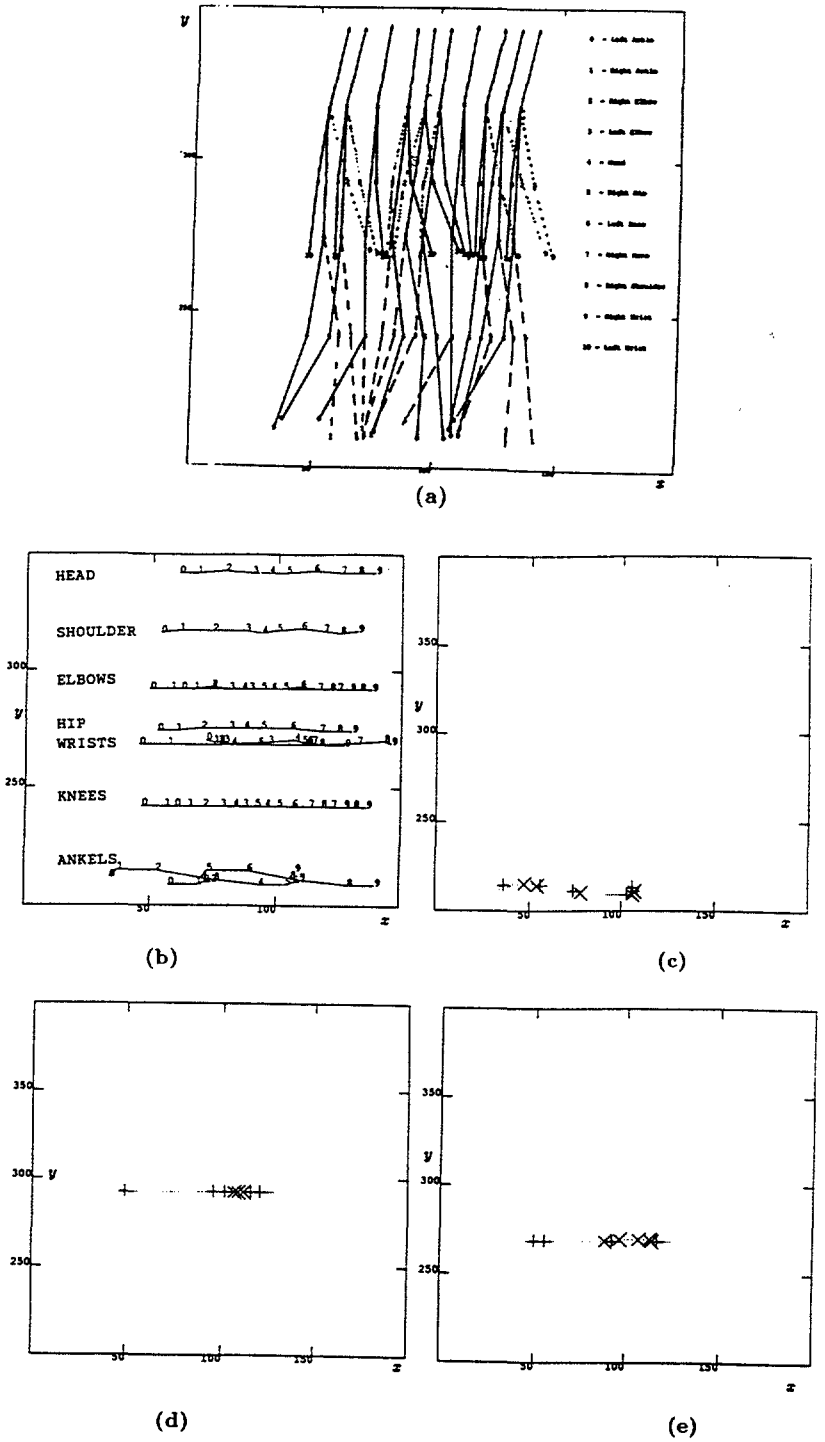


Figure 14.14: Results for Walker Scene. (a) Walker Sequence, (b) Trajectories, (c) TPS of Right Foot, (d) Left Elbow, (e) Wrist.

in x alone without direction change. In frame 176 it changes its velocity to $v_x = 6$, $v_y = -2$ which is an event in v_y alone without a direction change. In frame 206 it changes its velocity to $v_x = 12$, $v_y = -4$ which is an event in both v_x and v_y without a direction change. Finally, in frame 241 it changes its velocity to $v_x = 0$, $v_y = -4$ which is an event in v_x alone and it involves a direction change. Figure 14.16.b shows the CTPS representation for the Cube. Figure 14.16.c shows the reconstructed trajectories from the representation.

Figures 14.17.a-b and 14.17.c-d show shows respectively the first and last two frames in a 15 frame sequence. We applied the Moravec interest operator [7] to this sequence of images and selected interesting points by setting a threshold. The interest points in the first frame were then edited to choose two points on the walking man, one on his head and another on his shoulder. The interest points selected in the first frame are shown in Figure 14.17.e. The correspondence method [8], with minor modifications was used to compute the trajectory of these two points. The generated trajectories of these two points are almost parallel, as expected and are shown in figure 14.17.f. The ctps representation of this trajectory is shown in figure 14.17.g. The reconstructed trajectories, by choosing events which survive over 5% of the σ values are shown in figure 14.17.h. The number of events picked up is nearly equal to the number of frames in the trajectory. One reason being with a low threshold of 5%, even weak events survive. Therefore, the number of events can be varied depending on the application by changing the threshold. The other reason being the number of frames considered is not large. We had problems in generating a long sequence with a stationary camera, as the object was moving out of view in a small interval of time.

14.7.2 Rotation

As noted earlier trajectories corresponding to the rotation around a fixed axis can be represented by an ellipse. An ellipse is characterized by its phase, frequency, eccentricity, orientation, size, x-intercept and the distance between the x-intercept and the center of ellipse [11]. Storing the spatial coordinates of the end points of its major and minor axis along with the frequency and phase information also completely specifies an ellipse. The line joining the end points of the minor axis is extended to obtain the x-intercept, the orientation of this line is the orientation of the ellipse, also the lengths of the major and minor axis are used to compute the size and eccentricity. The point of intersection between the major and the minor axis marks the center of the ellipse and hence the distance between the $x-$intercept and the center of the ellipse is computed. Trajectories of points lying on the same rotating object are ellipses with the same frequency, orientation and x-intercept [11]. The fixed axis assumption [13] makes an equivalent statement that the minor axis of the ellipses generated by the points on the same rotating object will all be collinear. Figure 14.18 shows our CTPS for trajectories of rotating objects. The representation stores the frequency of rotation of the object and the component trajectories of the points on the object which are all ellipses. Each component trajectory is represented by the spatial coordinates of the end points of the major and minor axis of that ellipse along with the phase information.

Consider a simple case in which the major and minor axis of the ellipse are aligned respectively with the x and y axis. Let the origin of the $x - y$ coordinate system be shifted to (x_c, y_c), the center of the ellipse. Then, parametric equation

Translation CTPS			
Number Of Points On the Object		:: *npoints*	
Number Of Events In the Trajectory		:: *nevents*	
Location			
Point		x	y
1		..	..
2		..	..
..		..	..
..		..	..
..		..	..
npoints		..	..
Events			
No	frame	v_x	v_y
1	..	..	..
2	..	..	..
..	..	..	..
..	..	..	..
..	..	..	..
nevents	..	..	..

Figure 14.15: Translation CTPS. The information regarding the locations of points in the first frame, and the events is recorded. Each event is characterized by its location i.e the frame number of its occurrence, and the average velocities at that instant.

of the ellipse with respect to the origin at (x_c, y_c) can be written as:

$$x = -a * \cos\theta \tag{14.35}$$

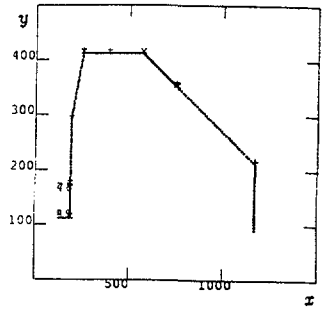
$$y = b * \sin\theta \tag{14.36}$$

Figure 14.8 shows the notation used. The frequency of rotation of the arm OP is the same as that of the corresponding point that is rotating in 3D. Also, $\theta = \alpha + \omega * t$, where α is the phase angle, and ω the frequency of rotation. Hence, if we know ω, the frequency of rotation, the parameters a, b, x_c, y_c of the ellipse, and the phase angle α of the trajectory, we will be able to determine the position of the point at any instant of time t.

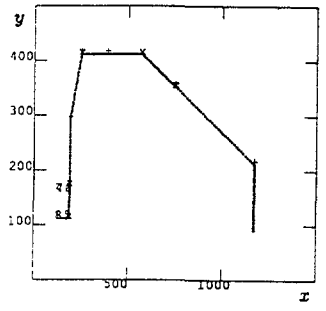
The orientation of the ellipse is decided by the axis about which the object rotates. The decomposition of a trajectory into v_x, and v_y makes it sensitive to orientation. When the ellipse is oriented such that the major and minor axis respectively are along the x-axis and y-axis, the two events in the v_x and v_y respectively appear at the end points on the major and minor axes. As the ellipse gets reoriented by an angle θ, the events are offset by θ and hence don't appear at the end points of the major and minor axes. On the other hand we find that the direction and speed representation is orientation insensitive. However, the observed events do not appear at the end points of the minor axis of the ellipse. The events in speed appear at angles $\pi/4$, $3 * \pi/4$, $5 * \pi/4$ and $7 * \pi/4$, and the events in direction are at angles 0, π. Since the distance between points doesn't change with the change of axis, events in speed are orientation independent. Similarly, change in direction is orientation independent as change in direction doesn't change with the change

Translation CTPS			
Number Of Points On the Object :: 8			
Number Of Events In the Trajectory :: 9			
Location			
Point		x	y
1		110	110
2		160	110
3		160	160
4		110	160
5		160	110
6		160	160
7		110	160
8		110	110
Events			
No	frame	v_x	v_y
1	1	3	0
2	21	0	2
3	51	0	4
4	81	2	4
5	111	4	0
6	146	6	0
7	176	6	-2
8	206	12	-4
9	241	0	-4

(b)



(a)



(c)

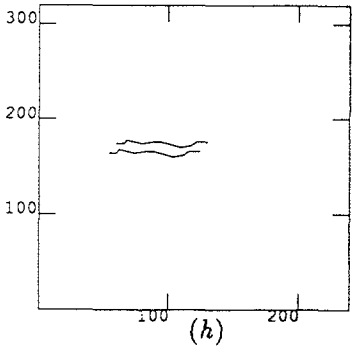
Figure 14.16: (a) One of the trajectories with the TPS events marked on it. (b) The CTPS representation for this translating cube. (c) The reconstructed trajectories from the representation.

(a) (b) (c)

(d) (e) (f)

Translation CTPS			
Number Of Points On the Object	:: 2		
Number Of Events In the Trajectory	:: 10		
Location			
Point		x	y
1		110	110
2		160	110
Events			
No	frame	v_x	v_y
1	1	6	0
2	2	2	-4
3	3	6	2
4	4	6	2
5	5	10	-2
6	6	2	0
7	8	5.33	2
8	11	8	-2
9	12	4	-4
10	13	4	0

(g)



(h)

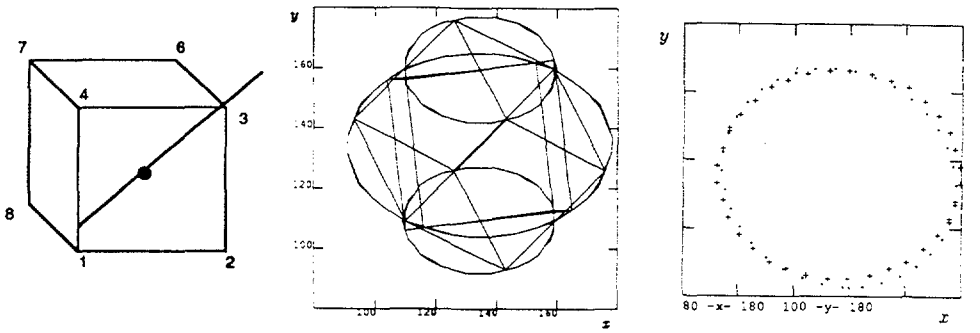
Figure 14.17: (a)-(b). The first two frames in a 15 frame real sequence. (c)-(d). The last two frames in the sequence. (e). The interest points detected in the first frame. (f). The generated trajectories of a point on the head and a point on the shoulder. The trajectories of these two points are almost parallel, as expected. (g). The ctps for these trajectories. (h). The reconstructed trajectories, by choosing events which survive over 5% of the σ.

Rotation CTPS									
Number Of Points :: npoints									
Frequency Of Rotation :: ω									
Point	Major Axis				Minor Axis				Phase
	x1	y1	x2	y2	x3	y3	x4	y4	
1	..	..	..	..	..	..	..	..	..
2	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..
npoints	..	..	..	..	..	..	..	..	..

Figure 14.18: Rotation CTPS. The representation stores the frequency of rotation of the object and the component trajectories of the points on the object which are all ellipses. Each elliptical trajectory is represented by the spatial coordinates of the end points of the major and minor axis of that ellipse along with the phase information.

of axis. The events in direction are at the end points of the major axis. The four events in speed will mark the end points of the major and minor axis if they are shifted by 45^o. In our implementation we use the direction events in identifying the end points of the major axis of the ellipse and the speed events for identifying the endpoints of the minor axis. Points belonging to the same object will have the same frequency of rotation. The frequency of rotation ω can be determined from the TPS of the trajectories, as follows $\omega = \frac{1}{2.\Delta\tau}$, where $\Delta\tau$ is the number of frames elapsed between two consecutive events in direction. The point of intersection between the major and minor axis is the center (x_c, y_c) of the ellipse, a is the half length of the major axis of the ellipse, and b the half length of the minor axis. Ideally, the major and minor axis should be perpendicular to each other and intersect at their mid points. However, due to delocalization of the events this is not the always the case. We have noticed that the delocalization of the end points of the major axis is negligibly small as the movement near these end points is very small. On the other hand the movement near the end points of the minor axis is large, and even a delocalization by only one frame makes a significant change in the orientation of the minor axis. The difference between delocalization of the events at the end point of major and minor axis will be more evident in an ellipse with a high eccentricity value. Hence, we use the following procedure to compute the parameters for reconstructing the ellipse. The spatial coordinates of the end points of the major and minor axis are computed from the direction and speed events. The point of intersection and the average length of the half segments of the minor axis is computed first. The mid point of the major axis gives (x_c, y_c). Thus the parameters x_c, y_c, a and b are computed consistently. Next, the phase angle α of a trajectory is computed as $\alpha = 2\pi - \omega.t^x$, where t^x is the frame number at which the direction event at the left end of the major axis is observed.

Figure 14.19.b shows the ellipse generated by the corner-1 of a Cube shown

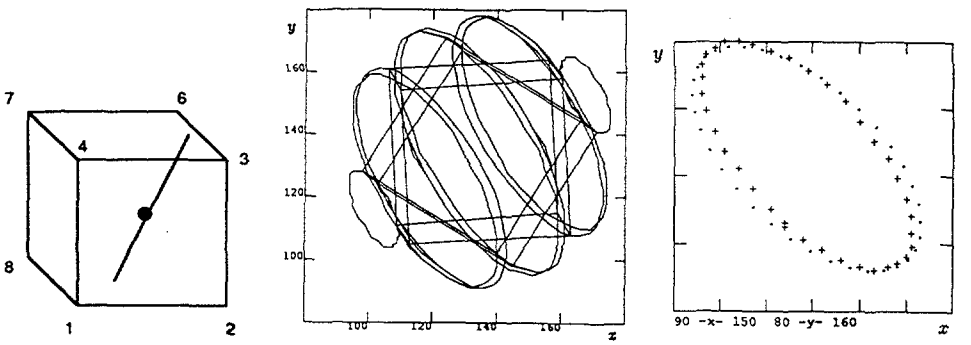


(a) (b) (c)

Rotation CTPS									
Number Of Points					:: 8				
Frequency Of Rotation					:: $\frac{11}{18} rad/sec$				
Point	Major Axis				Minor Axis				Phase
	x1	y1	x2	y2	x3	y3	x4	y4	
1	93	142	176	127	146	164	123	105	190
2	91	137	178	132	138	165	131	104	130
3	110	163	159	156	139	177	130	142	70
4	110	163	159	156	139	177	130	142	250
5	110	113	159	110	139	127	130	92	70
6	93	142	176	127	146	164	123	105	10
7	91	137	178	132	138	165	131	104	310
8	110	113	159	110	139	127	130	92	250

(d)

Figure 14.19: (a) A Cube rotating about an axis with direction *(0,1,1)*, passing through its centroid at $(135, 135, 135)$. (b) The elliptic trajectories traced out by the corners of the Cube. (c) The reconstructed trajectory of corner 1, superimposed on the original trajectory shown in b. Points on the original trajectory are marked by '+' and on the reconstructed trajectory by '.' (d) The rotation CTPS of the Cube.



(a) (b) (c)

Rotation CTPS									
Number Of Points :: 8									
Frequency Of Rotation :: $\frac{\Pi}{18} rad/sec$									
Point	Major Axis				Minor Axis				Phase
	x1	y1	x2	y2	x3	y3	x4	y4	
1	98	157	137	93	135	124	100	126	320
2	130	177	169	109	166	145	133	141	100
3	161	165	174	141	162	151	170	158	30
4	115	173	156	97	155	136	116	134	210
5	113	172	154	96	153	135	114	133	30
6	132	176	171	112	169	143	134	145	320
7	139	92	100	160	136	128	103	124	280
8	94	127	107	104	107	118	99	111	30

(d)

Figure 14.20: (a) A Cube rotating about an axis with direction *(2,1,1)*, passing through its centroid (135, 135, 135). (b) The elliptic trajectories traced out by the corners of the Cube. (c) The reconstructed trajectory of corner 1, superimposed on the original trajectory shown in b. Points on the original trajectory are marked by '+' and on the reconstructed trajectory by '.' (d) The rotation CTPS of the cube.

#	Max	Min	Av
1	6.11	0	3.77
2	2.03	0	1.69
3	2.06	0	1.28
4	2.06	0	1.28
5	4.72	0	3.03
6	6.11	0	3.77
7	2.03	0	1.69
8	4.72	0	3.03

(a)

#	Max	Min	Av
1	10.99	0	8.33
2	6.62	0	4.89
3	3.57	0	1.58
4	8.99	0	6.23
5	8.99	0	6.22
6	10.99	0	8.335
7	6.62	0	4.894
8	3.94	0	2.59

(b)

Figure 14.21: Error Tables for (a) Figure 19, (b) Figure 20.

in figure 14.19.a rotating about an axis with direction *(0,1,1)*, passing through (135, 135, 135). Orthographic projection was used to generate the trajectories. The Cube was rotating at a frequency of $\frac{1}{36}$ rotation per second i.e. 10^o per second. The observed events in direction which mark respectively the end points of the major and minor axis are separated by 18 frames. Figure 14.19.c shows the reconstructed trajectory of corner-1, superimposed on the original trajectory. The small discrepancy between the original and the reconstructed trajectory is due to the error in detecting the end points of major and minor axis from scale space.

The Table in figure 14.21.a shows the maximum, minimum and the average error in reconstructing the trajectories of the rotating Cube shown in figure 14.19. The Euclidean distance between a point on the original trajectory and the position of the point on the reconstructed trajectory at the same time instant is considered as the error measure. The minimum error is always 0, since the original trajectory and the reconstructed trajectory are aligned at the top end of the minor axis.

Figure 14.20.b shows the ellipses traced out by the corners of a Cube shown in Figure 14.20.a rotating about an axis with direction *(2,1,1)*, passing through (135, 135, 135). In this case also, orthographic projection was used to generate the trajectories, and it was assumed that the Cube was rotating at a frequency of $\frac{1}{36}$ rotation per second. The observed events in direction and speed respectively were used to find the end points of the major and minor axes. Figure 14.20.c shows the reconstructed trajectory of corner-1, superimposed on the original trajectory. The small discrepancy between the original and the reconstructed trajectory is due to the error in detecting the end points of the major and minor axis from scale space. The Table in Figure 14.21.b shows the maximum, minimum and the average error in reconstructing the trajectories of the rotating Cube shown in Figure 14.20.

14.8 Conclusion

In this paper, we proposed a new approach for use of motion in a computer vision system. In our approach, we use motion characteristics of objects without actually recovering the structure. We outlined a multi-scale scheme for representing the important events in the motion trajectories. These important events correspond to the discontinuities in speed, direction and acceleration of the objects. Our

method consists of converting the 2-D trajectory into two 1-D parametric functions, v_x and v_y. These functions are then analyzed at multiple scales to identify the significant changes in motion. The results of the multi-scale analysis are then used to determine the primitive type of the trajectory based on a relationship between the significant changes in motion in v_x and v_y. From experimental results with both real and synthetic trajectories, the method is quite promising. Further, we showed how a set of trajectories belonging to a single object can be compactly represented in a composite trajectory primal sketch representation. We also presented a method for reconstructing a trajectory from its CTPS.

References

[1] Asada, H., and Brady, M. The curvature primal sketch. Technical report, MIT AI memo 758, Cambridge: MIT, 1984.

[2] J.E. Cutting. *Motion representation and perception*, chapter Perceiving and recovering structure from events, pages 264–270. North Holland, New York, 1986.

[3] Cutting, J.E. A program to generate synthetic walkers as dynamic point-light displays. *Behaviour Research Methods and Instrumentation*, 10(1):91–94, 1977.

[4] Johansson, G. *Configurations in event perception.* Almqvist and Wiskell, Uppsala, Sweden, 1950.

[5] Johansson, G. Visual perception of biological motion. *Scientific American*, 232:76–89, June, 1975.

[6] Marr, D. *Vision.* Freeman, San Francisco, CA, 1982.

[7] Moravec, H.P. Towards automatic visual obstacle avoidance. In *Proceedings of International Joint Conference on Artificial Intelligence-5*, page 584, MIT, Cambridge, Masachussets, 1977.

[8] Rangarajan, K., and Shah, M. Establishing motion correspondence. Technical Report CS-TR-88-26, University of Central Florida, Computer Science Department, November, 1988.

[9] Sethi, I.K., and Jain, R. Finding trajectories of points in a monocular image sequence. Technical Report RSD-TR-3-85, University of Michigan Center for Research on Integrated Manufacturing, April, 1985.

[10] Shah, M. and Sood, A. and Jain, R. Pulse and staircase edge models. *Computer Vision, Graphics, and Image Processing*, 34:321–341, June, 1986.

[11] Todd, J.T. Visual information about rigid and nonrigid motion: A geometric analysis. *J. Experimental Psychology: Human Perception and Performance*, 8:238–252, 1982.

[12] Wallach, H. *The nature and art of motion*, chapter Visual perception of motion. George Braziller, New York, 1965.

[13] Webb, J. A. , Aggarwal, J.K. Visually interpretting the motion of objects in space. *Computer*, 14(8):40–46, August, 1981.

[14] Witkin, A. Scale-space filtering. In *Proceedings Eighth International Joint Conference on Artificial Intelligence*, pages 1019–1021, Karlsruhe, W.Germany: IEEE Computer Society, 1983.