LONG-RANGE SPATIOTEMPORAL MOTION UNDERSTANDING USING SPATIOTEMPORAL FLOW CURVES

by

Mark Allmen
and
Charles R. Dyer

Computer Sciences Technical Report #985

December 1990

Long-Range Spatiotemporal Motion Understanding Using Spatiotemporal Flow Curves

Mark Allmen
allmen@cs.wisc.edu

Charles R. Dyer
dyer@cs.wisc.edu

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Abstract

A spatiotemporal (ST) cube, created by "stacking" a temporally-dense sequence of images together, is a temporally-coherent data representation. Using ST surface flow, the extension of optical flow to ST surfaces, ST flow curves are recovered and used to group coherent regions of the ST cube such that each group represents an object or surface in the scene undergoing motion. ST flow curves are defined such that the tangent at a point on the curve equals the ST surface flow at that point. Our algorithm forms clusters of similar flow curves and is based the following two intuitive, yet powerful constraints called the temporal uniqueness constraints. First, a point in an image can only move to at most one point in the next image. Second, a point in an image can come from at most one point in the previous image. When these constraints are violated, or it appears that they are violated, occlusion or disocclusion has occurred. Successful grouping of coherent regions of the ST cube for five gray level image sequences is shown.

The support of the University of Wisconsin Graduate School under Project No. 910288 and the National Science Foundation under Grant Nos. IRI-8802436 and DCR-8521228 is gratefully acknowledged.

1 Introduction

Temporal coherence is a powerful constraint for understanding visual motion. Just as the gray levels of pixels around a point in an image are not independent, i.e., spatial coherence exists, the temporal neighbors of a point are not independent. If a sequence of images is taken with a short enough time interval between the images, the between-frame motion will be relatively small. This results in little change in the gray level at a point between frames, i.e., there is temporal coherence. Spatial coherence has been used innumerable times for image understanding. For example, edge points in a gray level image are defined as points where there exists a sudden change in gray level. Points where the gray level varies smoothly, i.e., there exists spatial coherence, are not edge points. Edge operators detect edge points based on this definition, recovering edge points only where spatial coherence does not exist [8]. Temporal coherence, while used in short image sequences for the computation of optical flow [1], spatiotemporal (ST) surface flow [3] and simplifying the correspondence problem [14], has not been used as a constraint for motion understanding over long image sequences.

In addition to temporal and spatial coherence of intensity values, there is temporal and spatial coherence of motion. Except at depth discontinuities, optical flow of an image does not vary randomly between pixels, i.e., there is spatial motion coherence. Also, the ST surface flow at a point does not change significantly between frames, i.e., there is temporal motion coherence. This paper shows how temporal motion coherence over long image sequences can be used for motion understanding.

An ST cube, constructed by stacking a sequence of temporally-close images together, has temporal motion coherence (as well as temporal gray level coherence) and is therefore an ideal structure for studying image sequences using temporal coherence [3, 18, 5]. A first step in understanding the motion in an ST cube is to determine the instantaneous motion of each point in the cube, called the *ST surface flow* [3]. The next step is to compute the motion of each point through the entire ST cube, or until the point vanishes. An ST curve through an ST cube such that the tangent at a point on the curve equals the ST surface flow

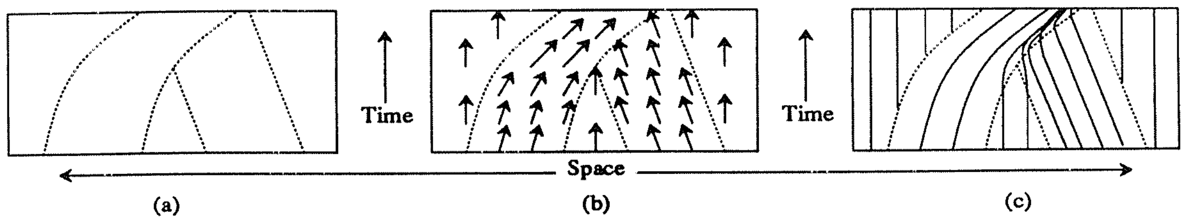


Figure 1: (a) A spatiotemporal image with two objects, one accelerating right and one translating left. (b) The ST surface flow. (c) ST flow curves.

at that point is called an *ST flow curve.* ST flow curves are recovered from the ST surface flow and then curves with similar shape can be clustered together. Each cluster represents a temporally-coherent structure in the ST cube, i.e., structures that result from an object or surface in the scene undergoing motion.

By using ST flow curves, the ST motion over an arbitrarily long image sequence is represented in a concise, well-defined form. Therefore, straightforward methods can be used to study them. For example, using clusters of ST flow curves, separate moving objects in the scene can be hypothesized, and occlusion and disocclusion between them can be identified by examining how clusters merge and split.

Figure 1 shows an example of how temporal motion coherence can be used for the interpretation of a long image sequence. Figure 1(a) shows a slice from an ST cube with one spatial and one temporal dimension. There are two objects, one accelerating to the right and one translating to the left. The ST surface flow is shown in Figure 1(b). As expected, the flow for the left object points toward the right, the flow for the right object points toward the left, and the flow for background points straight into time. Figure 1(c) shows the resulting ST flow curves.

The flow curves for the left object all have similar shape, as do the flow curves for the right object. These curves can be clustered using properties of curves that measure the shape of the curve, e.g., curvature and torsion. However, the flow curves for the right object and the background have identical shape, i.e., they are straight. So in addition to examining the shape, for straight flow curves, the slope of the curve is also required. A space curve is completely defined, up to a rigid translation and rotation, by its curvature and torsion [10]. By applying standard clustering techniques using curvature, torsion and slope, three distinct clusters are formed. Each cluster represents the motion of a single object or the background over a long period of time.

As shown in Figure 1(c), by definition the flow curves associated with the right object merge into the flow curves associated with the left object as the right object becomes occluded. Over time the clusters associated with the two objects merge to form one cluster. This results because the temporal coherence assumption is violated. Where occlusion events such as this occur, temporal motion coherence does not exist. Temporal motion coherence can be interpreted by the following two intuitive yet powerful constraints that we call the *temporal uniqueness constraints.* First, a point in an image can only move to at most one point in the next image. Second, a point in an image can come from at most one point in the previous image. In areas where these constraints are not violated, i.e., temporal motion coherence exists, the flow curves will correspond to the motion of a single point, and therefore they can be clustered together to identify surfaces in motion in the scene. Where these constraints are violated, or it appears that they are violated, occlusion or disocclusion between surfaces has occurred resulting in clusters of flow curves splitting and merging (see Figure 1(c)).

The temporal uniqueness constraints can be interpreted formally as a mapping from one image to a subsequent image. Let $\mathbf{f}$ map a pixel from one frame to a subsequent frame. That is, $\mathbf{f} : \Re^2 \rightarrow \Re^2$, $\mathbf{f}(p_1) = p_2$, where p_1 and p_2 are pixels in a frame and a subsequent frame respectively, and projections of the same point in the scene. The temporal uniqueness constraints say that $\mathbf{f}$ is a bijection, i.e., 1-1 and onto.

1.1 Recovering a Qualitative Description

Interpretation of motion in an image, e.g., locating separate objects, can be done *prior* to recovery of 3D scene motion or 3D scene structure. A common paradigm when examining image sequences is to recover 3D structure or 3D motion immediately after optical flow is computed [1]. Recently, some work has even completely skipped the flow recovery step and computed structure and motion directly [16, 30, 11]. However, a great deal of information exists in the flow field and therefore our approach is to recover a qualitative description of image motion from ST surface flow. Qualitative analysis of image motion is not new. In the past, this usually meant a qualitative study of optical flow or short image sequences [12, 9, 32, 20]. Our work can be viewed as an extension of that problem into the temporal dimension, taking advantage of temporal motion coherence over long image sequences. This allows the recognition of higher-level motions, e.g., cyclic motion [4], which occur over long image sequences.

By clustering flow curves, a qualitative *grouping* of regions in the ST cube is recovered as opposed to a *segmentation* of the ST cube. Segmentation requires that every pixel in the cube be classified into some set. Our results are more qualitative, classifying pixels only in coherent areas. This approach is desirable for two reasons: first, all gradient-based methods for computing optical flow and ST surface flow are undefined at occlusion boundaries; second, only a qualitative result is needed in many situations.

In practice, optical flow and ST surface flow are not only undefined at occlusion boundaries, but also within a neighborhood around these boundaries. Gradient-based methods for computing flow assume motion and intensity values vary smoothly, an assumption that is violated at occlusion boundaries. To compute the flow, a window around each pixel is used. If any part of the window overlaps an occlusion boundary, the intensity values do not vary smoothly within the window and the computed flow is likely to be incorrect. To segment an ST cube, even these unstable pixels must be classified when, in fact, their values are incorrect.

Since the ST surface flow cannot be computed reliably near occlusion boundaries, our approach recovers a qualitative description of motion. However, in many visual motion tasks, exact localization of motion boundaries is not required. For example, if one is interested in tracking an object, the exact location of the object boundary is not necessarily as important as, say, where the centroid of the object moves [2]. An automated surveillance system may need to detect only that motion occurred, and track the centroid of the motion [7]. In obstacle avoidance, unless the margin for error is small, the exact boundaries of objects in relative motion are not needed [24]. So rather than attempt to classify pixels based on inaccurate data, exact localization of motion boundaries is sacrificed.

Related work in psychology supports the view that localization of occlusion boundaries is not required in order to recover a useful description of motion. Johansson [19] attached lights to the joints of a person and then filmed the person undergoing motion, such as walking, in the dark. Only the motion of the lights was visible in these *moving light displays* (MLD). When people observed these MLD's they almost immediately recognize the motion. Clearly, localization of boundaries is not required for recognition of motion since no boundaries exist in an MLD, only the path of the lights through time and space exists. Note that these paths are precisely ST flow curves. This suggests that ST flow curves represent salient properties of the image sequence and are useful for motion description and recognition.

Related work in computer vision falls into two categories, that dealing with segmenting an ST cube and that dealing with segmenting an optical flow field. The latter can be viewed as a 2D version of our problem. However, while the problem appears similar, approaches must differ since temporal coherence is the dominate constraint in our method and optical flow segmentation deals with flow at a single instant in time. In addition to working with an instant in time, most optical flow segmentation approaches attempt to localize discontinuities in the flow [22, 31, 27] even though the image flow is in error in these areas [29]. As stated earlier, our approach does not deal with these areas, but rather uses areas where the flow is well defined.

Jain segmented an ST cube by examining the signs of three principle curvatures of an ST 3-surface [18]. These signs were determined by the type of the motion that generated the surface, translational for example. As such, only the type of motion was used and not the direction of motion, as we do when using ST surface flow. Later, Liou and Jain presented a volume-growing algorithm based on gray level [21]. While both of these approaches address the problem of ST cube segmentation, they used temporal and spatial gray-level coherence as opposed to temporal and spatial motion coherence.

Peng and Medioni presented an algorithm that analyzed oriented slices of an ST cube [25]. An edge operator was applied to each slice and the resulting edge slices were then examined to recover the normal component of motion. λ and Y junctions, indicators of occlusion and disocclusion respectively, were located in each slice. The problem with their approach is that edge detectors model an edge as a sudden intensity change, a model not satisfied near λ and Y junctions. Since our method need not localize boundaries of occlusion and disocclusion, this is not a problem. Our method will identify occlusion and disocclusion but there is no reliance on detecting boundaries such as λ or Y junctions.

In the next section, it is shown how flow curves are recovered from ST surface flow. Section 3 describes the type of flow curves that are generated and how they are clustered such that each cluster represents the motion of one coherent object in the scene. Finally, Section 4 shows how to detect occlusion and disocclusion from flow curve clusters. Results are presented in Section 5. We assume that the ST cube is composed of gray level images. However, our methods are easily modified to also work with edge images. But, as reported by Allmen and Dyer [3], the additional problem of recovering coherent ST surfaces must be solved when using edge images rather than gray level images.

2 Recovering Flow Curves

In this section we show, given a sequence of images forming an ST cube, how the motion of points through the cube are recovered. The first step is the computation of the *spatiotemporal surface flow*, i.e., the instantaneous motion of each point in the ST cube. The spatiotemporal surface flow is computed as described by Allmen and Dyer [3] and gives a vector for each pixel in an ST cube indicating the motion of that pixel. Given the ST surface flow over many frames, flow curves through the ST cube are then recovered. Loosely, flow curves are started in the first frame and flow through the cube to the most recent frame. In theory, a flow curve could be started at every pixel in the first frame. For practical reasons, flow curves are started at uniformly-spaced intervals in the first frame, e.g., one curve for each non-overlapping 16×16 block of pixels. Nothing other than computation time prevents a more dense placement of flow curves.

An ST flow curve can be represented as a parameterized space curve. A parameterized ST 3D curve, $\alpha(\mathrm{t})$, is a map $\alpha : \mathbf{I} \rightarrow \Re^3$ of an open interval $\mathbf{I} = (\mathrm{a}, \mathrm{b})$ of the real line $\Re$ into $\Re^3$. α defines a correspondence which maps each $t \in \mathbf{I}$ into a point $\alpha(t) = (x(t), y(t), t) \in \Re^3$.

Using prime to denote the partial derivative with respect to t, the vector $(x'(t), y'(t), 1) = \alpha'(t) \in \Re^3$ is called the tangent or velocity vector of the curve α at t.

Assume for the moment that the ST surface flow is smooth. Given a smooth ST surface flow, an ST flow curve α is defined such that the velocity vector of α at each point equals the ST surface flow at that point. (See Figure 1.) Let the ST surface flow be defined by

$$\mathbf{F} : \Re^3 \rightarrow \Re^3 \qquad \mathbf{F}(\mathbf{x}) = (\Delta x, \Delta y, 1)$$

Requiring the velocity of α to equal $\mathbf{F}$ is equivalent to

$$\alpha'(t) = \mathbf{F}(\alpha(t))$$

A given flow curve has the initial condition $\alpha_i(0) = (x_0, y_0, 0)$, where (x_0, y_0) are the coordinates of the pixel in the first frame. Using coordinates $(x,\ y,\ t)$ for the ST cube, the preceding equation can be rewritten as the simultaneous equations

$$\begin{aligned} x'(t) &= F_1(x(t), y(t), t) \\ y'(t) &= F_2(x(t), y(t), t) \\ t'(t) &= F_3(x(t), y(t), t) \quad = \quad 1 \end{aligned}$$

with initial conditions

$$(x(0), y(0), t) = (x_0, y_0, 0)$$

where $\mathbf{F} = (F_1, F_2, F_3)$.

Many methods can be used to solve this system of equations given the starting points in the first frame and the ST surface flow, $\mathbf{F}$, defined at every pixel in the ST cube. However, since $\mathbf{F}$ is only defined at coordinate points and must be interpolated at intermediate pixels, the relatively simple Runge-Kutta method [26, 13] is appropriate. More sophisticated methods where the increment in t varies depending upon the complexity of $\mathbf{F}$ are not used since there is no reason not to use the smallest increment in t available, namely 1.

Two issues remain regarding the recovery of flow curves: when do they terminate and when are they created? That is, how is the interval associated with a flow curve determined? Clearly, flow curves must be created in the first frame and terminated in the most recent frame. But this is not sufficient. For example, Figure 1(c) shows flow curves starting where the background becomes disoccluded. These issues will be addressed in Section 4.

The results of the Runge-Kutta method give a sequence of points that define each flow curve. In order to compute shape-description properties of flow curves such as curvature, a quadratic curve is fit to each set of points. A separate curve segment, centered at each point, is fit for every point making up a flow curve. This is done using a 1D version of the quadratic surface fitting procedure described by Besl and Jain [6]. Once the quadratic curve segment is fit, the partial derivatives at a point can be recovered and the curvature computed using the following equation:

$$\kappa = \frac{\sqrt{A^2 + B^2 + C^2}}{((x')^2 + (y')^2 + (t')^2)^{3/2}}$$

where

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix} \qquad B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix} \qquad C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}$$

The velocity vector at a point of a flow curve, $(x'(t), y'(t), 1)$, will also be a useful curve descriptor since it gives the instantaneous direction and speed of motion through the ST cube. Also, the difference between two velocity vectors of two flow curves at time t measures the instantaneous difference in speed and direction of the two flow curves.

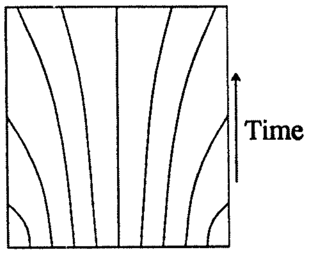
In summary, flow curves are recovered from the ST surface flow using the Runge-Kutta method. For each curve the Runge-Kutta method gives a sequence of points that define the curve. After fitting a quadratic curve segment to each point of each curve, the curvature and velocity vector at each point can be computed as shape descriptors of the curve.

3 Clustering Flow Curves

Once flow curves are recovered from an ST cube they must be grouped, or clustered, so that all the flow curves that make up a cluster are "similar." Ideally, similarity should be defined so that two flow curves are similar if they both are generated by the motion of a single rigid object. Torsion, curvature and velocity are three flow curve descriptors that can be used to measure similarity. How and why these measures will correctly group curves as similar will be discussed below. How to cluster flow curves using these measures will then be described.

3.1 Describing Flow Curves

Any space curve, and hence any flow curve, is completely defined, up to rigid rotation and translation, by its curvature and torsion. Intuitively, the curvature at a point on a curve measures how fast the curve pulls away from the velocity or tangent vector at the point. The normal vector at a point is defined as the instantaneous change of the velocity vector at that point. The plane formed by the velocity vector and the normal vector is called the osculating plane. The torsion at a point measures how fast the curve pulls away from the osculating plane. So the curvature is a measure of curving in the plane and the torsion is a measure of twisting out of the plane. Given a starting point, an orientation, and the curvature and torsion values at every point on a curve, that curve is uniquely defined [10]. Therefore, the torsion and curvature are good measures of the shape of curves [3, 23, 17]. By classifying the different types of motion that can occur in the scene, and the resulting projected motion, one can gain an understanding of the types of flow curves that can be produced. Then it



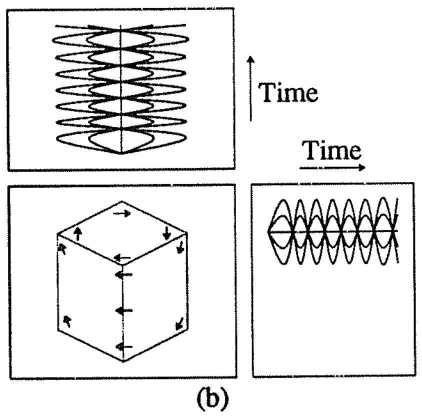


Figure 2: (a) Flow curves from a slice of the ST cube where an object is moving toward the camera. The focus of expansion is in the center of the slice. (b) The lower-left view shows the ST surface flow of a rotating cube. Time is into the page. The top and right views show the flow curves for the top surface of the box.

can be shown how torsion, curvature, and velocity can be used so that two curves generated by the same object will be considered similar.

Objects in a scene can move in only two ways — they can translate and rotate. These motions can be further classified as parallel to the image plane or in depth, resulting in five separate categories of motion: translation parallel to the image plane, translation in depth, rotation parallel to the image plane, rotation in depth, and no motion. The resulting flow curves for each type of motion can be studied by examining how the curvature, torsion and velocity change in each case.

When an object is not moving or is translating at constant speed parallel to the image plane, the flow curves generated are straight lines as shown by the right object in Figure 1. Curvature in this case is zero and torsion is undefined. The velocity vector indicates the direction and speed of motion.

If orthographic projection is used, there is no apparent motion when an object translates in depth. In the case of perspective projection, flow curves will curve away from the focus of expansion (foe). See Figure 2(a). Because the flow curves are planar, torsion is always equal to zero and curvature varies along the curve. Curvature is larger the farther the flow curves are from the foe. Since the flow curves form a symmetric pattern around the foe, the velocity vector varies between flow curves.

An object rotating parallel to the image plane results in cork-screw-like flow curves. The curvature increases from the center of rotation where it equals zero to the curvature of the flow curve farthest from the center of rotation. Torsion is undefined at the center of rotation and decreases with distance from the center. The velocity vectors are cyclic along the curve and increase with distance from the center of rotation.

To illustrate the types of motion that can be produced when an object rotates in depth, Figure 2(b) shows flow curves resulting from a cube rotating in depth. The flow curves resulting from the top face have an oval, cork-screw shape. Curvature, torsion and velocity behave the same as for rotation parallel to the image plane except the values along the curve are cyclic. The flow curves generated by the side faces of the cube are also oval shaped but since these faces turn away from the viewer, the flow curves merge with the background after each face becomes occluded.

The torsion values produced by these four types of motion are of limited usefulness for distinguishing between these cases. This does not mean that torsion is a poor measure for distinguishing between space curves in general, only that it is poor for the types of space curves produced in an ST cube. Here, since curvature varies as torsion does, or inversely to it, torsion will not help in distinguishing flow curves. Hence, using only curvature and velocity, all flow curves of one type can be distinguished from the other types.

Table 1 summarizes the above analysis by showing how curvature, torsion and velocity change along a flow curve (temporally) and between flow curves (spatially) of the same type. For example, consider the flow curves associated with an object rotating parallel to the image plane. The curvature and torsion along a flow curve is constant and the velocity is cyclic.

Using Table 1 it can be shown that curvature is sufficient to distinguish between the four motion types. Given two different types of curves from Table 1, it must be shown that the curvature values along the curves differ. This will show that curvature is sufficient to

Motion	Spatial			Temporal		
	Curvature	Torsion	Velocity	Curvature	Torsion	Velocity
No Motion	Zero	Undefined	Zero	Zero	Undefined	Zero
Translate Parallel To Image Plane	Zero	Undefined	Constant	Zero	Undefined	Constant
Translate In Depth	Increases	Zero	Increases	Increases	Zero	Increases
Rotate Parallel To Image Plane	Increases	Decreases	Increases	Constant	Constant	Cyclic
Rotate In Depth	Increases	Decreases	Increases	Cyclic	Cyclic	Cyclic

Table 1: Curvature, torsion and velocity changes between curves (spatial) and along curves (temporal). "Increases" and "Decreases" is with respect to distance from the center of rotation or foe.

distinguish the different types of flow curves.

Consider the Temporal, Curvature column of Table 1. Ignoring for the moment the No Motion row, curvature along a curve varies differently for the four types of flow curves. So given two different types of flow curves, the difference between these curves must be large since the curvature values along the curves are different most everywhere. One exception is a flow curve with zero curvature, generated by an object translating parallel to the image plane, and a flow curve at the center of rotation or foe. Since these two curves are similarly shaped, no measure can distinguish them based on shape alone. See Section 5 for how position can be used in this case.

The only types of flow curves that curvature cannot distinguish are the ones resulting from translation parallel to the image plane and no motion because in both cases the curvature is zero. Therefore velocity is also required in general to classify flow curves uniquely according to the type of motion that generated them.

Torsion could be used to differentiate between most of the motion types, but nothing is gained by using torsion as well as curvature. Therefore, because torsion adds nothing and is a third derivative operation making it very susceptible to noise, only curvature and velocity are used. Other work has also used only curvature for similar reasons [4].

From these five primitive cases of 3D motion, we have shown how curvature and velocity are sufficient for describing and classifying flow curves. When objects undergo a combination of these types of motion, accelerate, or there is camera motion, the flow curves become more complicated of course. In these cases, velocity becomes less useful for clustering and therefore curvature must be used as the primary description of a curve.

3.2 Flow Curve Difference Measures

Given a set of flow curves, we would like to be able to cluster those curves so that all the curves in each cluster were generated by a single rigid object. Typically, a clustering algorithm is given a set of items and a way of measuring the "difference" between objects. For a given number of clusters there is an optimal clustering of the items that minimizes the total difference. However, finding this optimal clustering requires exponential time. Therefore, most clustering algorithms attempt to find a clustering that is close to the optimal but requires considerably less time to compute.

All clustering algorithms suffer from the fact that the algorithm does not know how many clusters exist *a priori*. In many applications it is reasonable to expect that this information is supplied. However, in our case this would amount to specifying how many objects are moving in the scene. Clearly this is an undesirable assumption. Therefore we will use heuristic techniques for automatically determining how many clusters exist [15, 28].

A clustering algorithm needs a function that measures the difference between two items. It has already been shown why curvature and velocity are sufficient to distinguish between ST flow curves, but it has not been shown how, using these measures, to compute the difference between a pair of curves. We will first show how this is done in the continuous case. After converting this result to the discrete case, a very intuitive difference measure is obtained.

The curvature and velocity of a curve can be thought of as functions, κ and α':

$$\kappa(\alpha(t)) \rightarrow \Re \qquad\qquad \alpha'(t) \rightarrow \Re^3$$

κ and α' can be considered as points in Hilbert space. Hilbert space is an infinite dimensional vector space such that the vectors have finite length. For example, the vector $(1, \frac{1}{2}, \frac{1}{3}, ...)$ has finite length $\sqrt{(1)^2 + (\frac{1}{2})^2 + (\frac{1}{3})^2 + \cdots}$ so it is a vector in Hilbert space.

Consider the function $\kappa(\alpha(t))$ on the interval $t_i \leq t \leq t_j$. $\kappa(\alpha(t))$ can be considered as a vector with a continuum of components along the interval. Intuitively, the squared length of such a vector is the summation of all the squared components, which becomes the integral of $\kappa(\alpha(t))$. I.e.,

$$\|\kappa(\alpha(t))\|^2 = \int_{t_i}^{t_j} [\kappa(\alpha(t))]^2 dt$$

Since the length of κ and α' (or the distance to the point in Hilbert space) can be computed, the distance between two functions in Hilbert space can also be found. The direction from one function to another is the component by component difference of the two functions. The squared distance then is just the squared length of this difference vector. Since the length of each function is finite, the length of the difference vector, or the distance between the functions, will also be finite.

The discussion above assumes that the functions κ and α' have finite length. For every point in some bounded interval, κ and α' each have a finite value. The values can be arbitrarily large, but they are still finite. Therefore over the bounded interval the length of the function is bounded. Again, the length of each function can be arbitrarily large, but it will be finite.

In the discrete case the integral is replaced with a summation and we arrive at the following distance or difference measure between functions:

$$\begin{aligned}
\|\kappa(\alpha_1) - \kappa(\alpha_2)\|^2 &= \sum_{t=t_i}^{t=t_j} [\kappa(\alpha_1(t)) - \kappa(\alpha_2(t))]^2 \\
\|\alpha_1' - \alpha_2'\|^2 &= \sum_{t=t_i}^{t=t_j} [\alpha_1'(t)) - \alpha_2'(t)]^2 \\
&= \sum_{t=t_i}^{t=t_j} \left[\sqrt{(x_1'(t) - x_2'(t))^2 + (y_1'(t) - y_2'(t))^2 + (t - t)^2)}\right]^2 \\
&= \sum_{t=t_i}^{t=t_j} \left[(x_1'(t) - x_2'(t))^2 + (y_1'(t) - y_2'(t))^2\right]
\end{aligned}$$

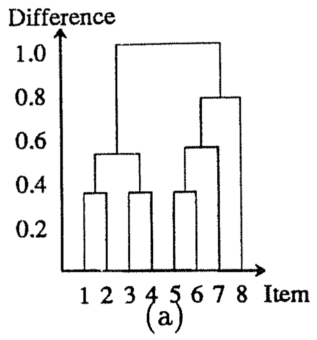
Note this summation is over a fixed segment of the two curves. This segment can be the entire curve from time 0 to the current time or say, the most current T steps. In any case, for each time, the difference is computed, squared and summed. A weighted average of these two difference measures is used to compute the difference between flow curves. See Section 5 for how these weights are set.

3.3 Clustering Algorithms

Given a difference measure, many possible clustering algorithms could be used. As stated earlier, most clustering algorithms require the number of clusters be given *a priori*. In order to determine the number of clusters, we chose to use a hierarchical clustering method. This algorithm uses the first n frames to initially determine the number of clusters at time n. Next, K-Means clustering clustering is used to update the clusters at each subsequent time step. K-Means works in an iterative manner, updating the clustering as each flow curve is extended into the current frame.

The hierarchical clustering method used for initially determining the number of clusters begins by forming singleton clusters with each flow curve calculated over the interval $[0,\ldots,n]$. The two closest clusters are then combined. At each step, the curvature and velocity values of the merged clusters are averaged and become the curvature and velocity values of the new cluster. The iterative combination of pairs of clusters with the smallest inter-cluster difference continues until only one cluster remains. This clustering process forms a binary tree, with the singleton clusters at the leaves and the final cluster at the root. The difference between two adjacent nodes in the tree is defined by the difference between the two associated clusters. See Figure 3.

Clearly, in most cases, clustering should stop before only one cluster remains. One heuristic, suggested by Romesburg [28], determines the number of clusters such that the decision is least sensitive to error in the difference measure. Figure 3(b) shows the number of clusters obtained for different widths of the ranges of difference between the clusters in



# Clusters	Range of difference	Width of range
8	$0.0 < difference < 0.35$	0.35
2	$0.78 < difference < 1.04$	0.26
3	$0.56 < difference < 0.78$	0.22
5	$0.35 < difference < 0.54$	0.19
4	$0.54 < difference < 0.56$	0.02
1	$1.04 < difference < \infty$	-

(b)

Figure 3: (a) The leaves show the initial singleton clusters. The vertical axis indicates the difference between the two clusters being merged. For example, the difference between items 1 and 2 is 0.35. The difference between items 3 and 4 is also 0.35. The difference between these two merged clusters is 0.54. (b) Number of clusters obtained for different widths of the ranges of difference for the tree shown in (a).

Figure 3(a). Romesburg suggests that the decision to terminate merging pairs of clusters is least sensitive to error when the width of the range is the largest. For Figure 3 this results in deciding that two clusters exist in the data.

We use a heuristic similar to Romesburg to identify the number of clusters and hence number of moving objects. Ideally, there should be a "clear" point, indicated by a large difference, where, by combining clusters, flow curves from different objects are merged. When the minimum difference between clusters is greater than a threshold times the minimum difference in the previous iteration, hierarchical clustering terminates. The number of clusters at this point is the number used for further processing.

Romesburg's heuristic is desirable since it does not require a threshold. But consider the situation where there are three moving objects and a static background. The number of clusters in this case should be four. Our method will terminate when there are four clusters remaining since the difference between all pairs of remaining clusters is high. But there is no reason to believe that the minimum difference with four clusters remaining will be greater than with three clusters remaining, or two clusters, etc., as required using Romesburg's heuristic.

Once the initial number of clusters has been determined and the flow curves have been clustered, a different, more incremental-type clustering is performed. For each new frame, the curvature and velocity of each flow curve's next point is computed. Then, for each flow curve, using a fixed width interval ending at the current time, the difference of the flow curve and its cluster's mean is computed. If this difference is greater than the difference of the flow curve and another cluster's mean, the flow curve is moved into the other cluster. This is called the K-Means clustering method [15, 28]. Note that the number of clusters does not change, but it allows flow curves to move between existing clusters. This is important in areas of occlusion and disocclusion.

4 Occlusion and Disocclusion Detection

Detection of areas of occlusion and disocclusion is straightforward once the initial clusters have been computed. Occlusion is characterized by a cluster associated with an occluded surface merging into a cluster associated with an occluding surface. Occlusion areas are areas where this merging occurs. Disocclusion areas exist where flow curves split from one cluster to another. In practice, clusters do not actually split since once a flow curve is following the ST surface flow associated with an object, it cannot flow out of this area into the ST surface flow area associated with the object being disoccluded. Consequently, disocclusion results in areas where no flow curves exists. So rather than using splitting clusters, disocclusion is detected by noting areas where flow curves do not exist.

The temporal uniqueness constraints were defined earlier as: a point in an image can only move to at most one point in the next image, and a point in an image can come from at most one point in the previous image. Figure 1(b) shows the ST surface flow for a scene with two objects. In regions away from occlusion boundaries, these constraints are not violated. However, at occlusion boundaries two points flow to one point in the next frame.

Figure 1 shows the ideal case where an occlusion discontinuity occurs. In practice this discontinuity appears as a strip of random flow with the width of the strip dependent upon the neighborhood size used to compute the ST surface flow. This occurs because gradient-based methods to compute ST surface flow are undefined at occlusion and disocclusion boundaries. A method that tries to find these occlusion and disocclusion boundaries by looking locally for this discontinuity will have to actually look for a strip of random-like flow.

If we ignore these unstable boundary areas by simply extending the flow curves, they will eventually leave the unstable strip and enter a more stable area, an area away from an occlusion or disocclusion boundary. Figure 10 shows the flow curves for an image sequence where one object is moving in front of another object. The flow curves generated by the occluded surface quickly move though the area where the flow is undefined and merge with the flow curves generated by the occluding surface.

The temporal uniqueness constraints can be viewed on two different levels. Viewed locally, the temporal uniqueness constraints are violated when two points flow to one point in the next frame or a single point flows to more than one point in the next frame. Viewed globally, the temporal uniqueness constraints are violated when flow curves from one cluster merge with another cluster or flow curves split from a cluster. By using the temporal uniqueness constraints at the global level, we avoid having to rely on areas where the ST surface flow is undefined.

Given the neighborhood size used to compute the ST surface flow, the width of the random-like strip can be estimated. The temporal neighborhood used to compute the difference between flow curves can be chosen large enough so that any noise in the curve due to the occlusion region will be minimal. Therefore it is expected that a flow curve will stay in the cluster representing the occluded object and then switch into the cluster representing the occluding object.

Once a flow curve has merged into another cluster and has remained there for some period of time, it could be deleted. Other than computation time, there is no reason for this step, however.

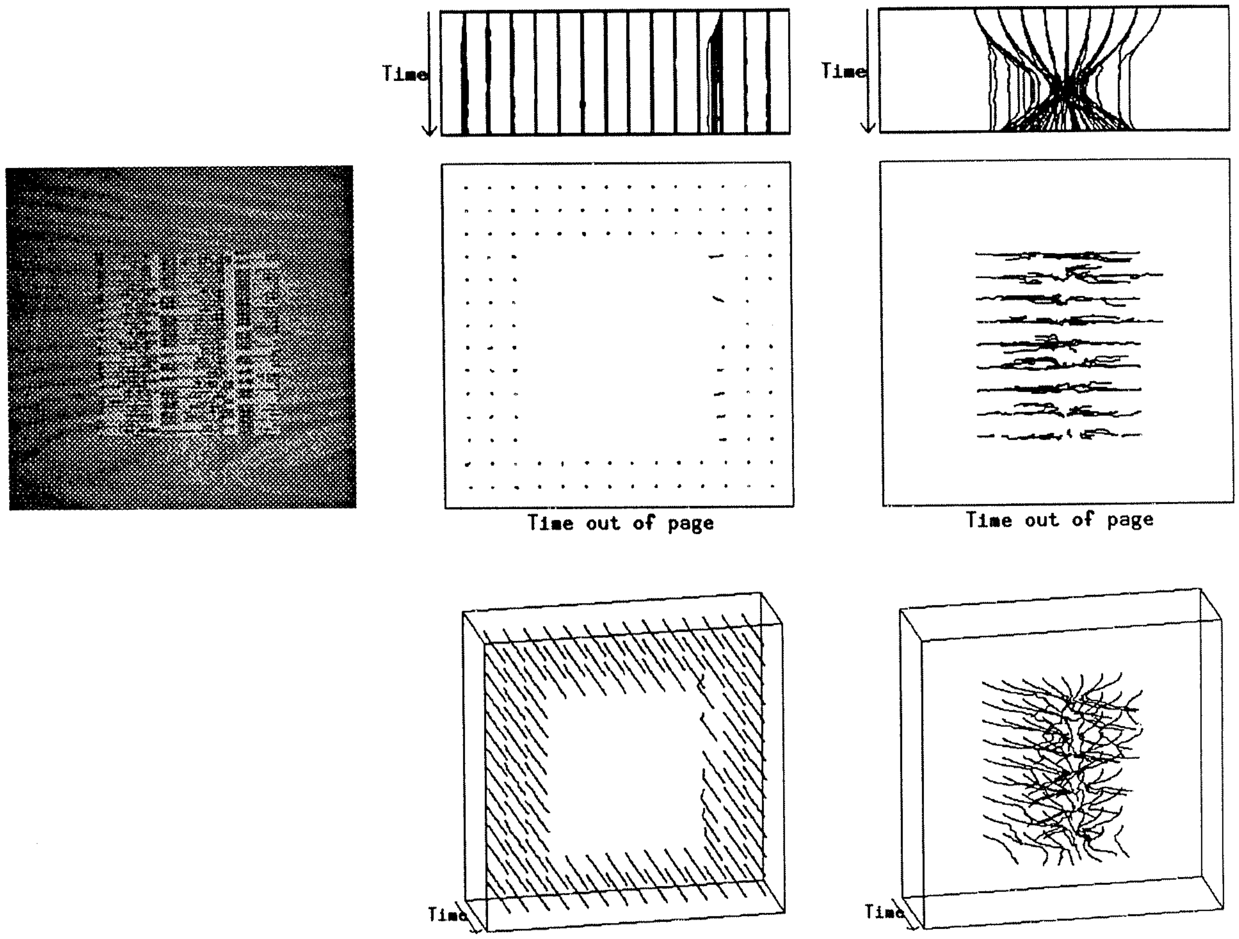


Figure 4: ST flow curves from a 115 frame sequence of a phone book page rotating against a static background about a vertical line through the middle of the page at 0.02 radians/frame. One frame of the sequence is shown on the left. The center and right columns show the top, front and oblique views of the two clusters. Note that some flow curves far from the center of rotation failed to follow the rotating page. Most of these curves then followed the background and became part of the background cluster as shown by the top view of the background cluster.

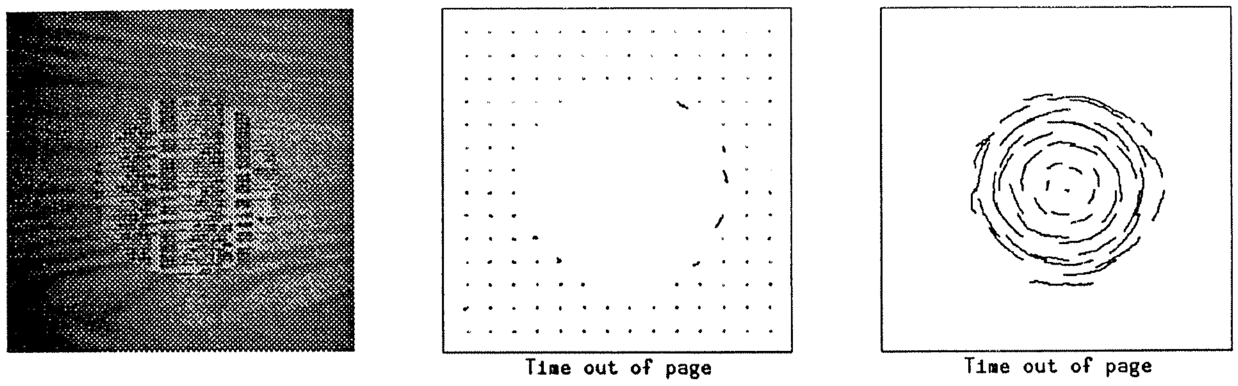


Figure 5: ST flow curves from a 115 frame sequence of a phone book page rotating about its center at 0.008 radians/frame against a static backgound. One frame of the sequence is shown on the left. The shape of the flow curves resulting from the background (center) and the phone book page (right) are straight and cork-screw shaped, respectively.

The situation is similar at disocclusion boundaries. Flow curves must be created in regions where the density of flow curves drops below some threshold. Note there is no problem if too many curves exist. Near boundaries of disocclusion it is expected that the density of flow curves will decrease because all curves will follow the disoccluding object and no flow curves will follow the disoccluded object.

5 Results

In this section results of the algorithm on five image sequences are presented. Also, while not necessary from a theoretical standpoint, an additional flow curve descriptor is presented that helps to correctly cluster flow curves. We also show why it is desirable to use the position of flow curves when computing the difference between them.

Figures 4 to 8 show the flow curves and resulting clusters for five image sequences. The set of clusters are the result of the initial hierarchical clustering algorithm. The results of the subsequent cluster updating procedure using K-Means is shown in Figures 9 and 10. All image sequences were synthetically generated by transforming a planar region of a phone book page over a larger image of a table top. Each image sequence was smoothed using a 3D

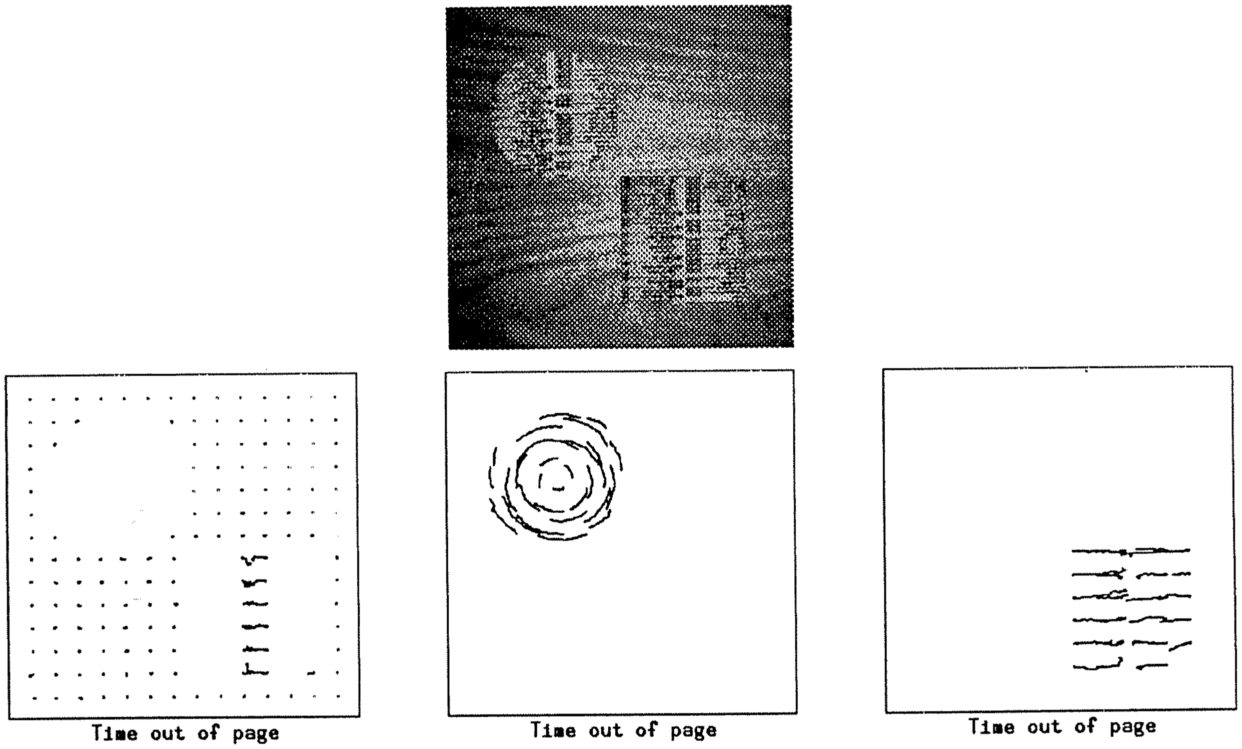


Figure 6: ST flow curves from a 115 frame sequence of a phone book page rotating about a vertical line through the middle of the page at 0.02 radians/frame and a phone book page rotating about its center at 0.01 radians/frames. One frame of the sequence is shown at the top. The front view of the resulting clusters is shown below. The flow curves near the vertical line of rotation are almost straight and therefore clustered with the background. The position of the flow curves did not force their clustering with the page rotating in depth because the nearest neighbor measure was used to compute distance.

Gaussian-weighted kernel with standard deviation of 2 pixels. ST surface flow was computed at every fourth pixel in every frame. See [3] for details on computing ST surface flow. The resulting flow for each sequence was convolved with a 3×3×3 median filter and smoothed using a 3D Gaussian-weighted kernel with standard deviation 0.66. Flow curves started in the first frame are spaced 16 pixels apart resulting in 196 flow curves.

Figure 5 shows cork-screw shaped flow curves resulting from a circular region rotating parallel to the image plane. The curvature of flow curves near the center is close to zero and increases for curves farther away from the center. Since the curvature values are similar, the curves were clustered together. The velocity vectors are in all directions so they do little

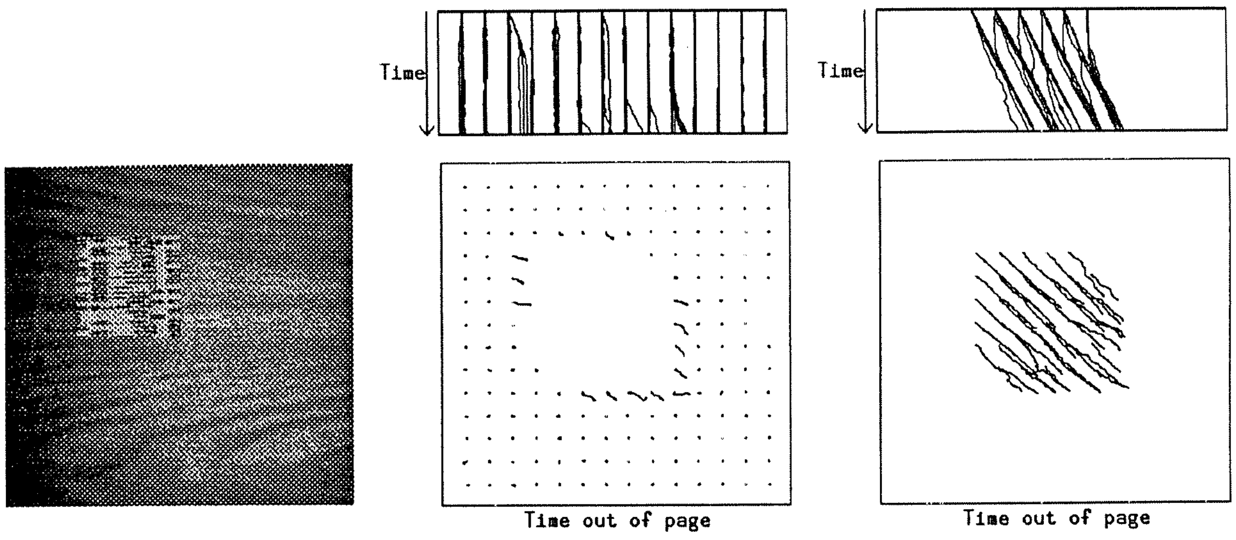


Figure 7: ST flow curves from a 115 frame sequence of a phone book page translating down and right at 0.4 pixels/frame. One frame of the sequence is shown on the left. The center and right columns show the top and front views of the two clusters. The flow curves angled down and right near the occlusion boundary in the background cluster are about to move into the other cluster.

to help the curves form a cluster. However, the *speed* of the vectors are similar. Speed is defined as the length of the spatial component of the velocity vector. Because it does not measure direction, it is unaffected by velocity vectors that are oriented in different directions. Using speed when measuring the difference between curves and clusters amounts to using the heuristic that flow curves associated with a single object move with similar speed in the ST cube. Note that speed could not be used instead of velocity since two objects translating with the same speed but in different directions would then be clustered as one object.

In general, using both speed and curvature may still not be enough to cause the flow curves near the center of rotation to be clustered with flow curves farther from the center. This is because the curves near the center are shaped like the curves associated with a static background; therefore, no matter what difference measure is used there may be problems computing the correct clusters if only motion is used. To alleviate this problem, the position of flow curves can also be used. By including position information, flow curves near the

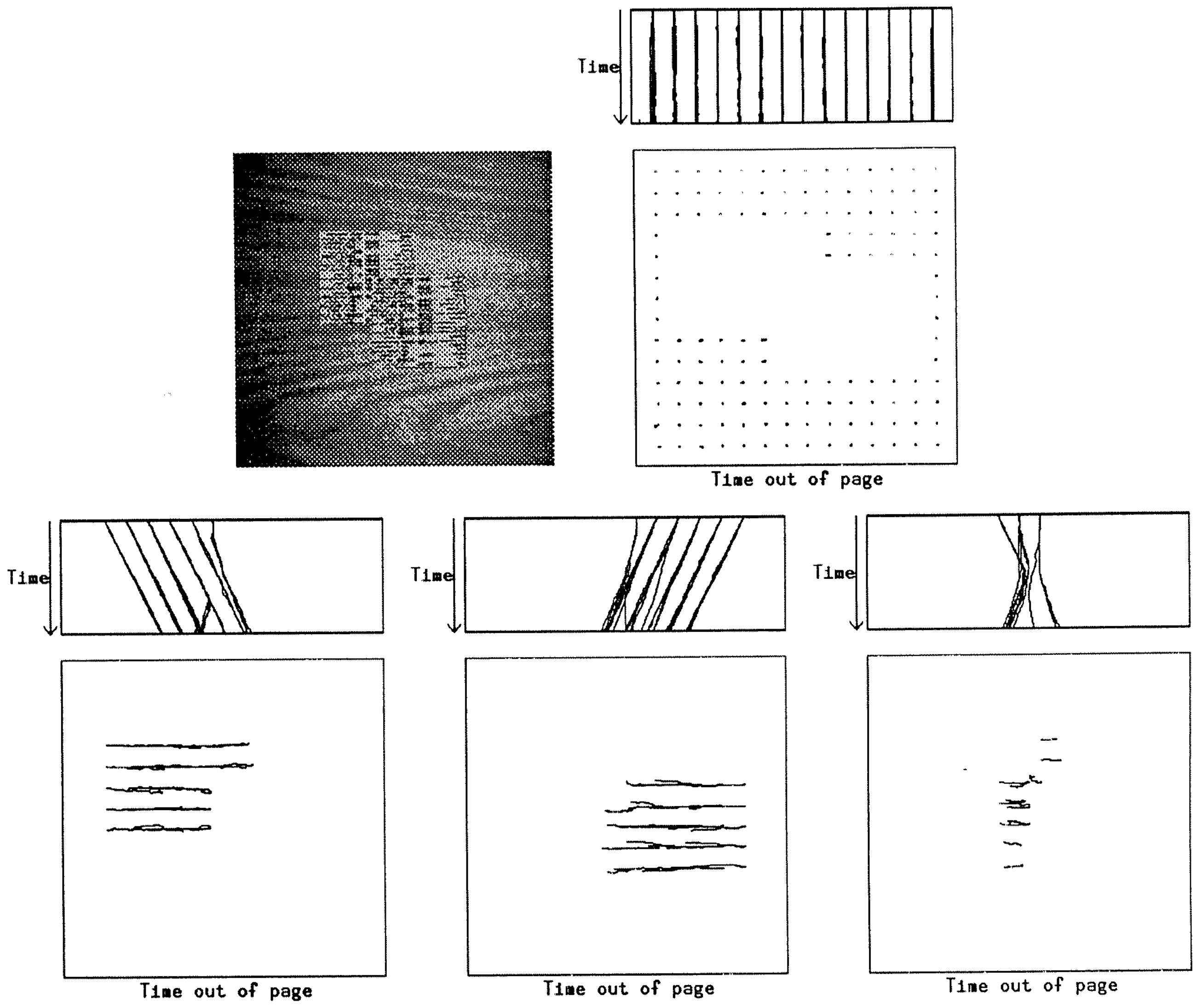


Figure 8: ST flow curves from a 115 frame sequence of two phone book pages, one translating left at 0.4 pixels/frame and partially occluding the other, which is translating right at 0.4 pixels/frame. One frame of the sequence is shown on the left. The front and top views of the resulting four clusters are shown. A fourth cluster results in the area where the two pages overlap because the flow curves in this area are shaped like flow curves generated by the left page for a while then shaped like the flow curves generated by the right object. This results in the flow curves shaped like no other flow curves so a fourth cluster remains. As the flow curves were extended into subsequent frames, the flow curves in the extra cluster became shaped more like the flow curves following the occluding object. K-Means then moved the flow curves into the cluster associated with the occluding object. This is shown in Figure 10.

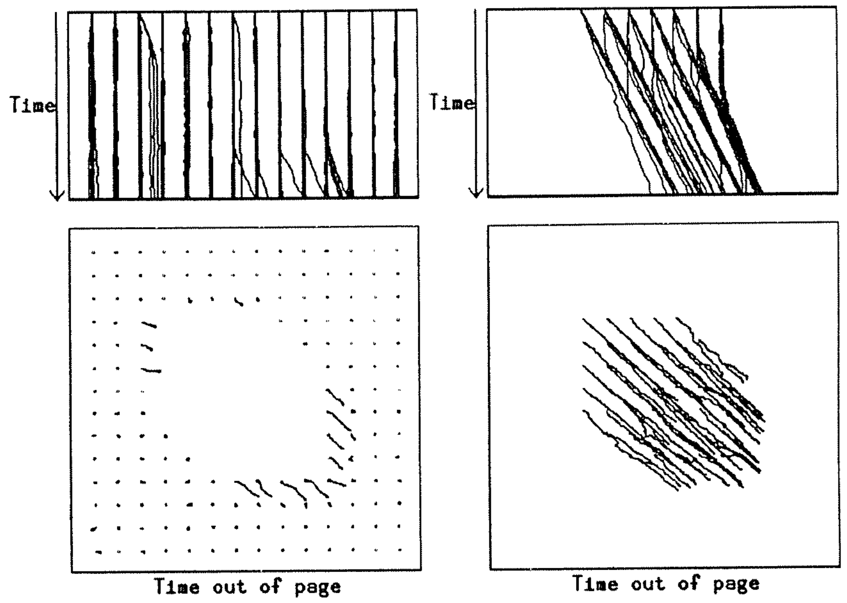


Figure 9: ST flow curves from Figure 7 extended to 170 frames. The top and front views of the two clusters resulting from K-Means are shown. The top view of the cluster for the translating object shows flow curves originally following the background and then following the page as the page occludes the background. Flow curves generated from the background near the leading edge of the page have merged into the cluster associated with the page.

center will cluster with those curves nearby it rather than with areas farther away. To compute the spatial distance between two clusters, the minimum spatial distance between all pairs of curves is used. Using distance when measuring the difference between curves and clusters amounts to using the heuristic that flow curves associated with an object are near each other. Since the nearest neighbor measure is used to compute the distance between clusters, this heuristic does not imply that a cluster cannot extend across large areas.

In the current algorithm, when computing the difference between curves and clusters, all components used to compute the difference are weighted equally. With the exception of the weight for the position of flow curves, similar results were obtained for other combinations of weights. When the weighting for position was low, flow curves near the center of rotation were clustered with the background. The fixed interval used to compute the difference between curves and clusters, was 33 frames. However, similar results were obtained when

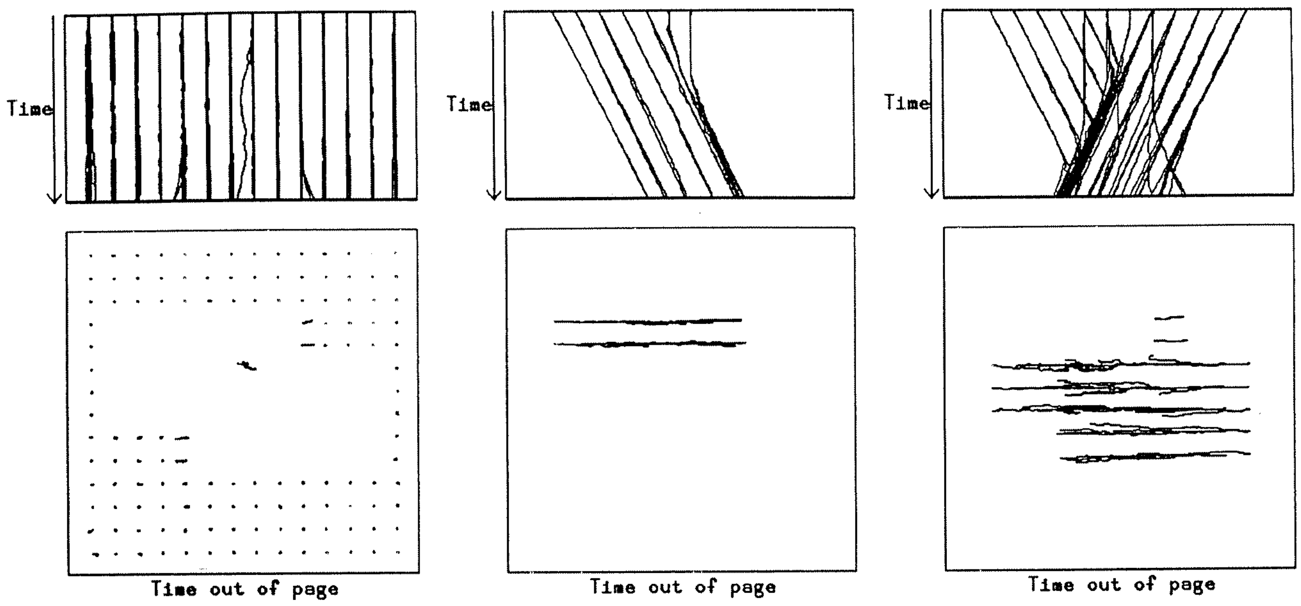


Figure 10: ST flow curves from Figure 8 extended to 170 frames. The top and front views of the three clusters resulting from K-Means are shown. The number of flow curves in the fourth cluster was low enough that it was deleted. All of the flow curves that were generated by the lower part of the object translating right have merged into the cluster associated with the object translating left. Also, flow curves generated from the background have merged into the clusters for the translating objects. There are two flow curves in the right cluster that should be in the center cluster. These errors are corrected by K-Means after the flow curves are extended a few more frames.

23 frames were used. Based on empirical results, when the minimum distance between clusters is greater than 3.5 times the minimum difference in the previous iteration, hierarchical clustering was terminated and the remaining clusters were then used by K-Means.

It is possible for an entire object to become occluded by another object. In this case the cluster associated with the occluded object should be deleted when the number of flow curves in the cluster becomes small. Based on the initial spacing of flow curves, this threshold was chosen to be 9. This threshold was also used for deleting clusters of erratic flow curves resulting from occlusion. For example, Figure 8 shows a cluster which resulted because flow curves followed one object for a while then followed another after the object became occluded. During K-Means the flow curves merged into the cluster for the occluding object and the cluster for the occluded object was deleted because its size fell below the threshold.

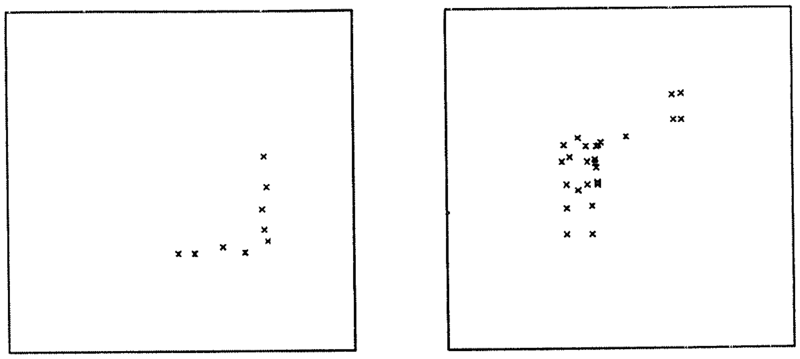


Figure 11: Occlusion boundaries from Figure 9 where the region translates down and right (left), and from Figure 10 where two regions translate (right). The position of each "x" indicates the position where a flow curve changed clusters. The occlusion boundaries for the left image show the background flow curves merging into the phone book page. The occlusion boundaries for the right image also show the background flow curves merging into the phone book pages and the flow curves associated with the left page merging into the right page.

6 Concluding Remarks

We have presented a method that takes advantage of temporal coherence over very long image sequences. Flow curves are recovered from the ST surface flow. Flow curves are then clustered such that each cluster represents the motion of a single surface or object through the ST cube. By analyzing the merging and splitting of clusters, occlusion and disocclusion boundaries are identified.

With coherent areas of the ST cube identified, we have a good starting point for higher-level understanding of the motion in an image sequence. For example, each area can be reduced to a single representative flow curve. This results in a significant reduction in the amount of data used in further processing. This is an important property in spatiotemporal image analysis since the amount of data is very large. With these representative flow curves, motion recognition can be performed without prior knowledge of the objects undergoing motion. Johansson's results with MLD's suggests that this is possible and that using flow curves is a promising approach [19].

References

[1] Aggarwal, J. K. and N. Nandhakumar. On the computation of motion from sequences of image - A review. *Proc. IEEE*, 76:917–935, 1988.

[2] Allen, P. K. Real-time motion tracking using spatio-temporal filters. In *Proc. Image Understanding Workshop*, pages 695–701, 1989.

[3] Allmen, M. and C. R. Dyer. Computing spatiotemporal surface flow. In *Proc. Int. Conf. on Computer Vision*, 1990. (to appear).

[4] Allmen, M. and C. R. Dyer. Cyclic motion detection using spatiotemporal surfaces and curves. In *Proc. 10th Int. Conf. on Pattern Recognition*, pages 365–370, 1990.

[5] Baker, H. H. and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Int. J. of Computer Vision*, 3:33–49, 1989.

[6] Besl, P. J. and R. C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Computer Vision, Graphics, and Image Processing*, 33:33–80, 1986.

[7] Burt, P. J. Smart sensing within a pyramid vision machine. *Proc. IEEE*, 76:1006–1015, 1988.

[8] Canny, J. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[9] Carlsson, S. Information in the geometric structure of retinal flow field. In *Proc. 2nd Int. Conf. on Computer Vision*, pages 629–633, 1988.

[10] DoCarmo, M. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, 1976.

[11] Faugeras, O. On the motion of 3D curves and its relationship to optical flow. In *Proc. 1st European Conf. on Computer Vision*, pages 107–117. Springer-Verlag, 1990.

[12] Francois, E. and P. Bouthemy. The derivation of qualitative information in motion analysis. In *Proc. 1st European Conf. on Computer Vision*, pages 226–230. Springer-Verlag, 1990.

[13] Gear, C. W. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, 1971.

[14] Grzywacz, N. M., J. A. Smith, and A. L. Yuille. A common theoretical framework for visual motion's spatial and temporal coherence. In *Proc. Workshop on Visual Motion*, pages 148–155, 1989.

[15] Hartigan, J. A. *Clustering Algorithms.* Wiley, 1975.

[16] Heel, J. Direct estimation of structure and motion from multiple frames. A.I. Memo 1190, MIT, 1990.

[17] Hoffman, D. D. and W. Richards. Representing smooth plane curves for recognition: Implications for figure-ground reversal. In Richards, W., editor, *Natural Computation*, pages 76–82. MIT Press, 1988.

[18] Jain, R. Dynamic vision. In *Proc. 9th Int. Conf. on Pattern Recognition*, pages 226–235, 1988.

[19] Johansson, G. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[20] Koenderink, J. J. and J. J. Van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22:773–791, 1975.

[21] Liou, S. P. and R. C. Jain. A parallel technique for three-dimensional image segmentation. In *Proc. 10th Int. Conf. on Pattern Recognition*, pages 201–203, 1990.

[22] Little, J. J., G. Blelloch, and T. Cass. Parallel algorithms for computer vision on the connection machine. In *Proc. 1st Int. Conf. on Computer Vision*, pages 587–591, 1987.

[23] Mokhtarian, F. Multi-scale description of space curves and three-dimensional objects. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 298–303, 1988.

[24] Nelson, R. C. and J. Y. Aloimonos. Using flow field divergence for obstacle avoidance: Towards qualitative vision. In *Proc. 2nd Int. Conf. on Computer Vision*, pages 188–196, 1988.

[25] Peng, S. L. and G. Medioni. Interpretation of image sequences by spatio-temporal analysis. In *Proc. Workshop on Visual Motion*, pages 344–351, 1989.

[26] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[27] Reichardt, W., T. Poggio, and K. Hausen. Figure-ground discrimination by relative movements in the visual system of the fly. Part ii: Towards the neural circuitry. *Biological Cybernetics*, 46:1–30, 1983.

[28] Romesburg, H. C. *Cluster Analysis for Researchers.* Lifetime Learning, 1984.

[29] Schunck, B. G. The image flow constraint equation. *Computer Vision, Graphics, and Image Processing*, 35:20–46, 1986.

[30] Taalebinezhaad, M. A. Direct recovery of motion and shape in the general case by fixation. In *Proc. Int. Conf. on Computer Vision*, 1990. (to appear).

[31] Thompson, W. B., K. M. Mutch, and V. A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7:374–383, 1985.

[32] Verri, A. and T. Poggio. Against quantitative optical flow. In *Proc. First Int. Conf. on Computer Vision*, pages 171–180, 1987.