Computing Optic Flow

Frank Glazer

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

ABSTRACT

Optic flow is the apparent motion of image features in the imaging plane. The computation of optic flow is a first step in determining the particular motions of lighting, objects, and viewers in an image sequence. New approaches are presented for the computation of optic flow from dynamic image data.

0. Introduction

The essential ingredients of the physics of visual perception are light sources , objects to be viewed, and the viewer (eye and camera). Elaborating on this system we can relate brightness in an image to various imaging factors including incident illumination, object range, surface reflectance and orientation, camera focal length, etc.

In a dynamic imaging situation imaging factors vary over time. This is usually the result of the motion of objects, viewers, or light sources. All else being equal, these motions induce the corresponding motions of their projections on the imaging plane. Optic flow is the apparent motion of image features in the imaging plane and can be represented as a vector field in that plane. The computation of this flow field is a first step in determining the particular motions of lighting, objects, and viewers in an image sequence.

Current methods for computing optic flow fall into two categories. The first, feature matching, involves detecting features in image frames and matching them in successive frames [1,6,8]. The second method attempts to "sense" optic flow directly from, the dynamic image data [2,1,7]. This method has also been seen to have two stages [2,1,7,8]. In the first stage the component of optic flow perpendicular to an edge (or parallel to the image gradient) is measured locally. These vectors will be called optic flow cues. In the second stage the mutual constraints of optic flow cues are used to compute the true optic flow. This paper presents new approaches to both of these stages.

1. Computing Optic Flow Cues

1.1 Image Gradient Methods

Fennema and Thompson, and Horn and Schunck derive a computation for optic flow cues based on the assumption of constant brightness (in time) of an object's image [2,1]. If the dynamic image is represented as a scalar field $I(x,y,t)$, then they give the magnitude of the component of optic flow in the direction of the (2-D) brightness gradient (I_x, I_y) as

$$\frac{-I_t}{\sqrt{(I_x^2 + I_y^2)}} \qquad (1.1)$$

where $(I_x, I_y\ I_t)$ is the gradient of $I(x,y,t)$. This computation also assumes that the brightness is smoothly varying in X and Y. This requirement is violated when the local detection window contains any one of the following:

1) An occlusion boundary between objects with different velocities
2) A shadow boundary with a velocity different from that of the shadowed object
3) A specular reflection (highlight) on an object moving relative to that object

All of these cases involve moving boundaries. It is proposed here that the detection of the motion cues at these boundaries must be computed with respect to some edge operator. 3D~edge operators are interesting in that they provide both the edge and the motion cue information.

1.2 Analysis of Dynamic Images in 3D XYT space

Discontinuities (edges) in the image result from discontinuities in the imaging factors, eg. occlusion of objects, shadowing (occlusion of lighting), surface markings, and internal edges of objects (see Table 1).

Table 1

Imaging factor	*Discountinuity*
surface range	occlusion boundary
surface orientation	internal object edge
incident lighting	shadow boundary
surface reflectance	surface markings

In a dynamic imaging situation we have an image I(x,y,t) with objects that can move, spin, shrink, expand, and deform (translate, rotate, scale, etc.) relative to the camera. Discontinuities in I(x,y,t) can be viewed as two-dimensional surfaces swept out by the edges in successive time-slices of I(x,y,t). These 3D-edges contain information about both the location of 2D-edges in a single frame and the velocity of these edges.

1.3 Moving 2D-edges as 3D-edges

A small local piece of a 2D-edge is essentially linear and a small local piece of a 3D-edge is essentially planar. Hereafter "2D-edge" and "3D-edge" will refer specifically to these small local pieces. A 2D-edge is the intersection of a 3D-edge with an XY plane (a plane of constant T). Conversely, successive occurrences of a 2D-edge sweep out a 3D-edge. The direction (orientation) of the 2D-edge in the XY plane and its velocity determine the orientation of the 3D-edge.

Let a 2D-edge E be represented in polar coordinates (p,G) as in Fig. 1. In the XYT coordinate system this normal form defines the "static" unit vectors U_1 - (cos(0) ,sin(6) ,0) and U_2 = (sin(0) ,-cos(6) ,0) in the XY plane which *are* perpendicular and parallel to E respectively.

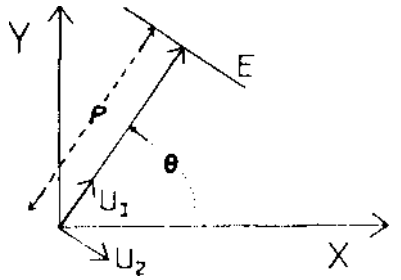


Figure 1 : Normal form representaion of edges
he edge E has the normal form of the line on which it lies, viz. (p,Θ) where p is the length of the perpendicular dropped to the origin and Θ is the angle in radians between that perpendicular and the X axis.

Let the velocity of E be the vector V = (V_x.V_y.I) = d/dt(x,y,t). The 3D-edge P swept out by E moving with velocity V is parallel to the vectors U_2 and V (see Fig. 2). Thus it is perpendicular to their cross product W = V x U_2 r (cos(G) ,sin(Θ) ,-Vx*cos(Θ)-Vy*sin(6)). This vector defines the orientation of JP. Note that the t component of W is -<V,U_1>. The inner product <V,U-|> is the length of the projection of V onto U_1,.

If we are given P = (Px,Py,Pt) a vector perpendicular to the 3D-edge P, it must be a multiple of W, viz. p * P = W. By noting that the projection of W into the XY plane is equal to U_1, a unit vector, we see that $p = 1/\sqrt{(Px^2 + Py^2)}$. We then have $-\langle V,U_1\rangle = p*Pt$, from which we get

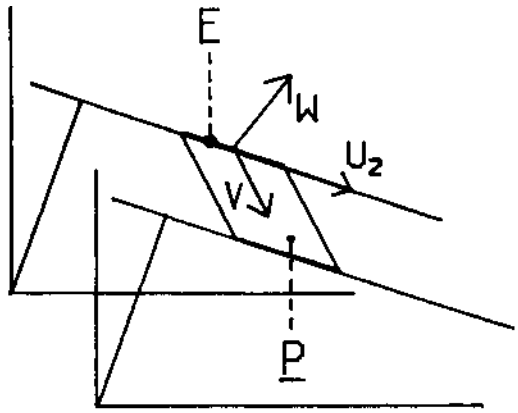


Figure 2.
A 2D-edge E moving with velocity V sweeps out a 3D-edge P. P is normal to the vector W = V x U_2.

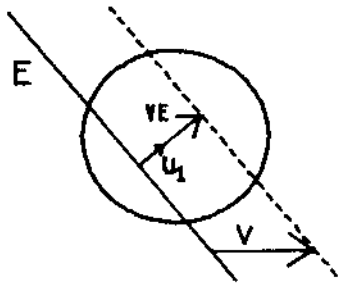


Figure 3 : The local ambiguity of a moving edge
The 2D edge E is shown moving with velocity V. Let U_1 be the unit vector perpendicular to E. Only the component of V parallel to U_1, (call it VE) can be detected. We have that VE = <V,U_1>»U_1

$$\langle V,U_1\rangle = \frac{-Pt}{\sqrt{(Px^2 + Py^2)}} \qquad (1.2)$$

This is the magnitude of the component of the velocity of the 2D-edge perpendicular to that edge.

The component of velocity parallel to the 2D edge cannot be computed from the 3D edge. This local ambiguity in the detection of optic flow can be demonstrated geometrically in the XY plane (see Fig. 3 from [8]).

The geometric interpretation of local ambiguity in three dimensional XYT space is seen in the fact that the locus *of* velocity vectors V that would generate JP (along with a given U_2) is the Intersection of P with the plane t = 1 . This is a line of solutions so one degree of freedom remains to be specified .

2. Computing Optic Flow

2.1 The velocity vector as a pseudo-Intersection

An optic flow cue provides only one component VE of the velocity V of a 2D-edge. The actual velocity can lie anywhere on the line In velocity space which has VE as its base vector (see Fig. *Ha)*. This line will be called the velocity line of VE. If we have two non-parallel optic flow cues VE_1 and VE_2 for the same actual optic flow, then the intersection of their velocity lines yields a unique solution for the actual velocity (Fig. 4*b)*.

When more than two cues are given for a single velocity, the velocity lines form a bundle through the actual velocity point. In practical applications these lines cannot be expected to intersect at a unique point. Imaging errors, discrete sampling, various noise sources, etc.

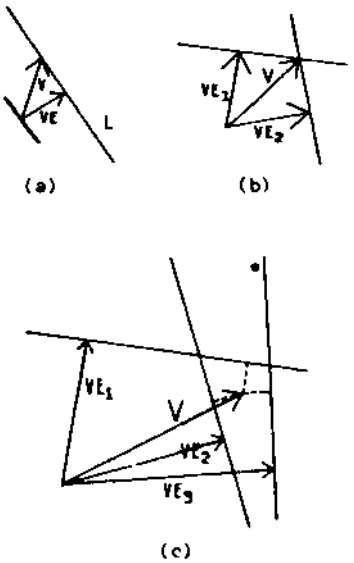
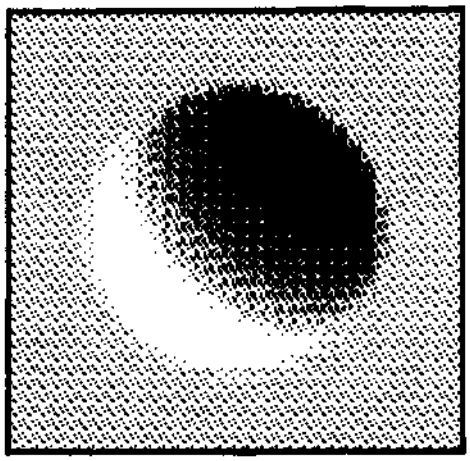
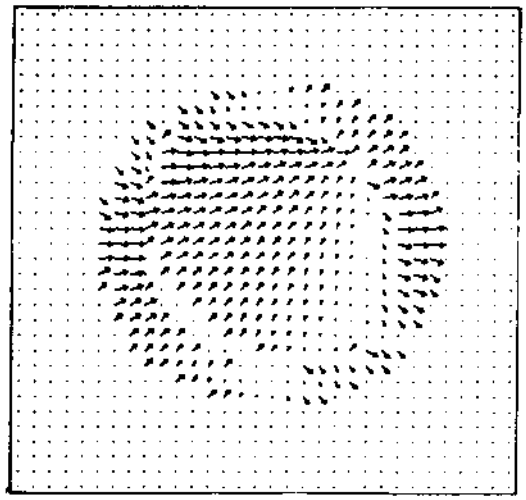


Figure 4_ Computing optic flow in velocity space
All vectors shown have their tails at the" origin in velocity space.
a)The optic flow V associated with an optic flow cue VE can be any vector lying along the velocity line L.
b)Two non-parallel optic flow cues determine a unique flow velocity.
c)The pseudo-intersection of velocity lines provides a "best fit" flow velocity V for a set of non-parallel optic flow cues VE_1.


(5a)


(5b)

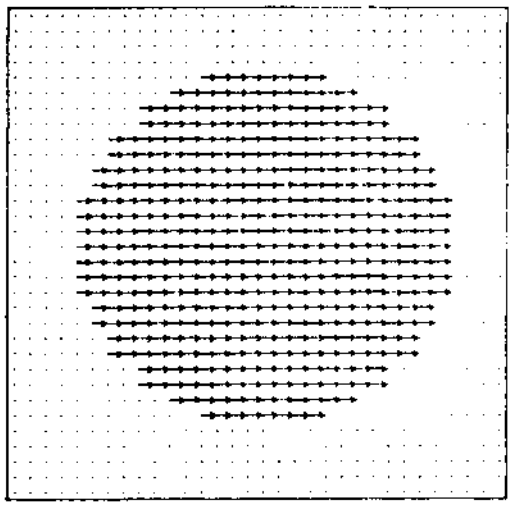

(5c)

Figure 5.
a) First frame of the translating sphere.
b)Optic flow cues computed from equation 1.1
c)Optic flow derived from (b) by computing the pseudo-Intersection of velocity lines in local 3 by 3 neighborhoods.

introduce alterations in the data and produce errorful measurements of the various VE_i. Given the measured velocity lines we can look for the point in velocity space that best "fits" these lines. This is the dual problem to that of best fitting a set of points by a line. As in that problem, there are different ways of specifying what is "best". Such measures are typically some function of the distances between the point(s) and the line(s). We have used the least square error (LSE) as the goodness-of-fit measure in our experiments. It reflects a standard intuitive definition of "best fit" and is easy to compute. The solution for the LSE error fit of a point to a set of lines in a plane is given in [3]. This point is called the pseudo-intersection of the given set of lines. Fig. 4c presents an example of the pseudo-intersection of a set of velocity lines.

An application of the pseudo-intersection method (PIM) to optic flow cues in an image plane is shown in Fig. 5. The first of two artificially generated images of a sphere is shown in 5(a). In the second frame (not shown) the sphere is translated one pixel to the right. The initial optic flow cues computed from (1.1) are shown in 5(b). In 5(c) we see the result of taking the pseudo-intersection of the velocity lines in al1 3 by 3 neighborhoods.

Successful computation of optic flow as the pseudo-intersection of velocity lines depends on those lines being derived from a single velocity. If optic flow cues from other velocities are introduced to the computation, the pseudo-intersection is moved towards those velocities and the LSE error increases. This condition may be detected by its high error, but the offending cues cannot be isolated by the methods discussed so far.

The crux of the problem is the need to partition optic flow cues into sets whose members are all derived from the same velocity. In the plane of velocities an equivalent problem can be formulated. There we want to partition the velocity lines into bundles. In the ideal case of errorless optic flow cues and exact velocity lines this problem is easy to solve. In the practical case we must look for good pseudo-bundles, i.e. sets of lines and their pseudo-intersections having low fitting errors. A solution to this problem has been worked out by Kender [5] who was looking for the vanishing points of texture patterns in 2D Images.

3- Discussion, Conclusions, and Future Directions

New approaches have been presented here for the detection of optic flow cues and the computation of optic flow from these cues. These two stages operate independently and so can be integrated with the alternative methods of Prager, Fennema and Thompson, Marr, or Horn [3].

The need for detection of moving edges is pointed out and the computation of optic flow at these edges from 3D-edges in XYT space is derived. A single step (non-iterative) local procedure is given for computing the optic flow for a set of optic flow cues.

Related research [3] not reported here includes:

1) study of the relevance of particular methods of 3D-edge detection for flow detection;
2) measures of certainty (of results) in the computation of flow b> the pseudo-intersection method (PIM);
3) iterative flow computations derived from the PIM which are more noise resistant.

Future directions include:

1) study of 3D-edge detection methods that are separable in XY vs. T coordinates (These methods avoid excessive storage requirements);
2) decoupling of local methods of optic flow computation across edges of discontinuity in flow;
3) integration of slow, iterative, noise resistant flow computation methods [4,8] with the fast PIM.

ACKNOWLEDGEMENT

This research was supported in part by the National Science Foundation under Grant MCS79-18209. The author acknowledges support from a University of Massachusetts Fellowship. The author would like to thank Al Hanson and Ed Riseman for providing useful comments and criticism.

REFERENCES

[1] Barnard,S.T. and Thompson,W.B., "Disparity Analysis of Images", IEEE-PAMI, 2(4): 333-340, July,1980.

[2] Fennema,C.L. and Thompson,W.B., "Velocity Determination in Scenes Containing Several Moving Objects", CGIP 9(4) :301-315, 1979.

[3] Glazer,F., "Computing Optic Flow'', Tech. Rep. in preparation, Computer and Information Science Department, Univ. of Massachusetts, Amherst,MA. 1981.

[4] Horn.B.K.P. and Schunk.B.G., "Determining Optical Flow", Proc.IUW(Darpa) , April,1981. Also A.I. Memo-572. MIT Al Lab., Cambridge,MA, 1980.

[5] Kender, J. R. , "Shape From Texture: An Aggregation Transform that Maps a Class of Textures Into Surface Orientation", Proc. IJCAI-6, Tokyo,Japan, 1979.

[6] Lawton.D.T., Processing Dynamic Images and the Control of Robot Behaviour,Ph.D. Thesis in preparation, Univ. of Massachusetts, Amherst,MA.

[7] Marr,D. and Ullman.S., "Directional Selectivity and its Use in Early Visual Processing", A.I. Memo-524, MIT Al Lab.. Cambridge,MA, 1979.

[8] Prager,J.M., Segmentation of Static and Dynamic Scenes, Ph.D. Thesis and COINS Tech. Rep. 79-7, Univ. of Massachusetts, Amherst,MA 1979.