Optical Flow: Techniques and Applications

Peter O'Donovan
502425
The University of Saskatchewan*

April 6, 2005

*peo499@mail.usask.ca

1 Introduction

The analysis of sequence of images, used to approximate motion, is a core area of computer vision. The 3-D velocity vector of objects, projected onto the image plane, is known as the image flow field. This could be considered the ideal, the actual movement of objects that we expect to see. Unfortunately, we are often working in the reverse direction: given a sequence of images, we want to determine the movement of the objects. This leads to an approximation called the optical flow field(see Figure 1) which corresponds to the movement of pixels in an image, or what we actually see. It is easy to imagine situations where a sequence of images do not correspond with the actual object movement. A perfectly black matte ball rotating for instance, will show no sign of movement in an optical flow field, but the image flow field would show the rotation. As long as the optical flow field provides a reasonable approximation though, it can be used in a variety of situations, including time-to-collision calculations, segmentation, structure of objects, movement parameters, among many others. This paper will initially discuss sone issues surrounding optical flow calculation followed by several common methods used including local and global differentiation techniques, correlation, feature based methods and hierarchical approaches. In the following section, several uses of optical flow previously mentioned will be discussed.

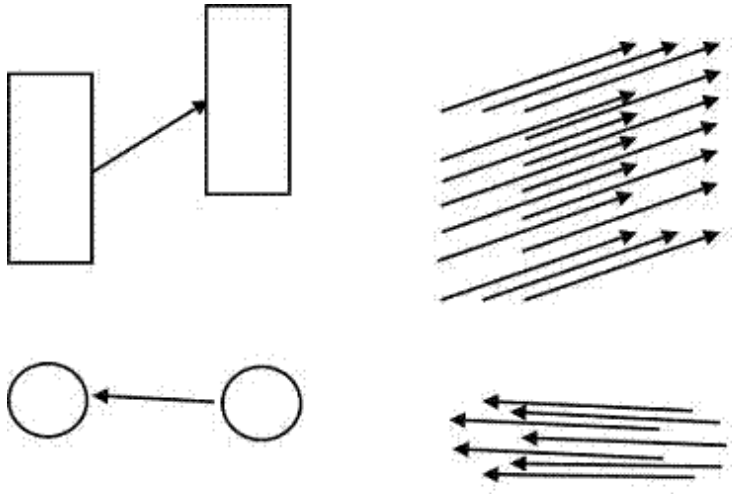

Figure 1: On the left we see the movement of objects over two frames. On the right, the optical flow vectors for each of the objects.

2 Optical Flow Constraint Equation

The most basic assumption made in optical flow calculations is image brightness constancy. This is simply the assumption that from a short interval $t1$ to $t2$, while an object may change position, the reflectivity and illumination will remain constant. Mathematically, this is

$$f(x + \Delta x, y + \Delta y, t + \Delta t) \approx f(x, y, t) \tag{1}$$

where $f(x, y, t)$ is the intensity of the image at position (x,y) and at time t, and Δx, Δy is the change in position and Δt is the change in time.

If we apply a Taylor series expansion to the left hand side, we get

$$f(x + \Delta x, y + \Delta y, t + \Delta t) = f(x, y, t) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y + \frac{\partial f}{\partial t}\Delta t + h.o.t. \tag{2}$$

where $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial t}$ are the partial derivatives of the image function in the x, y, and t dimensions. We ignore the higher order terms since we are dealing with a small neighborhood. Once we substitute equation 1 into equation 2, we are left with the optical flow constraint equation:

$$\nabla I \bullet \mathbf{v} + I_t = 0 \tag{3}$$

Where $\nabla I = (I_x, I_y)$ is the spatial gradient, $\mathbf{v} = (u, v) = (\Delta x, \Delta y)$ is the optical flow vector and I_t is the temporal gradient. Since we are dealing with a single time displacement between two frames, $\Delta t = 1$ and thus disappears. The three gradients are easy to calculate using derivative operators. The optical flow vector (u, v) is what we are searching for.

This equation tells us that when we apply a flow vector to the spatial gradient of the image it will be exactly canceled by the temporal gradient. This makes sense since we have assumed that there will be no change in the brightness of the image.

2.1 Aperture Problem

While the previous equation is certainly useful, there remains a large problem. The equation provides only a single constraint for the two components of the optical flow vector. Only the component of the vector in the direction of the spatial gradient is provided. This can be seen in the following diagram which illustrates how the component perpendicular to the gradient is missing. Since we are calculating the flow from individual pixels, we are limited by the size of the gradient operators. They might be too small to see the other component of the optical flow vector. Obviously, at positions where there is a gradient in two directions, like a corner, this problem will not arise. This realization inspires one good method of finding optical flow vectors: calculate the optical flow vector only at points of high curvature. However, this will result in a sparser optical flow field.

2.2 Assumptions

There are a number of assumptions made during the calculation of optical flow fields. As mentioned before, we must assume that objects are illuminated uniformly and that surface reflectance does not contain specular highlights. Imagery such as shadows, highlights, variable illumination, and surface transparency will cause this assumption to be violated. Occlusion or transparencies are other events which cause inconsistencies in the optical flow field. Beauchemin [4] mentions these along with current research done to mitigate these problems. In the context of this project however, we assume the most basic image sequences possible and will therefore ignore these more complex issues.

3 The Computation of Optical Flow

There are numerous methods to calculate optical flow. In his survey, Beauchemin [4] mentions six classes of methods without even scratching the surface of feature detection based methods. In this paper, we initially consider two separate and widely known techniques from the differential-based class. Horn and Schunk use derivatives to calculate the first constraint on the flow vector, and then solve for the orthogonal component using a global method of minimizing a smoothness constraint. Lucas and Kanade use a local method which calculates the flow vector using the constraints of a neighborhood around the pixel. We will then look at using correlation techniques to determine the optical flow field. Related to these are feature-based methods which find points of high curvature and calculate the flow vectors only there. We also discuss another class of feature based methods

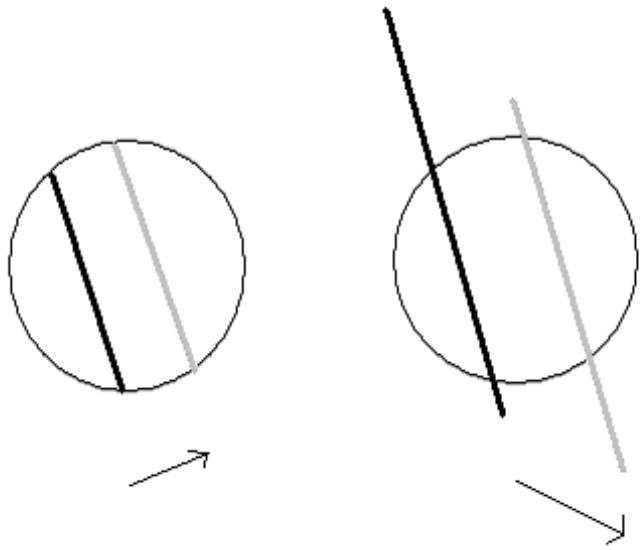

Figure 2: On the left, only the spatial gradient component of the optical flow vector is found. On the right, we see the missing orthogonal component.

which use rigid edges to resolve ambiguities. Lastly, we examine hierarchical approaches to optical flow.

3.1 Differential Methods

3.1.1 Horn and Schunk

One of the classics of optical flow calculation, Horn and Schunk's [15] method is global. Calculation of any flow vector can be based on the entire image. Their method uses the optical flow constraint equation but also imposes a smoothness constraint. The rational is that the optical flow field should vary smoothly and should have few discontinuities. They therefore add a constraint to minimize the square of the magnitude of the optical flow vector $\mathbf{v} = (u, v)$.

$$E_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{4}$$

where $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}$ is the change of the u component along the two dimensions and $\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}$ is the change of the v component. This error is the difference of the optical vector compared to its neighbors. If the vector is significantly different the gradient of the flow vectors along either the x or y dimension will be large. The sum of this error along with the error of the optical flow constraint equation gives us a total error to be minimized. Remember, the optical flow constraint equation will be 0 when the optical flow vector matches perfectly with the spatial and temporal gradients.

Therefore, the total error to be minimized is

$$E^2 = \int\int (\nabla I \bullet \mathbf{v} + I_t) + \alpha^2 \left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right) \text{dxdy} \tag{5}$$

where the first term is the optical flow constraint equation and the second is the smoothness constraint which is multiplied by a constant weighting factor α^2.

To recapitulate, what we are doing is taking the sum over all the pixels of the error for each pixel's optical flow vector. The error can either be the amount that the flow vector deviates from the spatial and temporal gradient, or it could be lack of smoothness of the flow vector over a range of pixels. If the flow vector is significantly different from its neighbors then this gradient will be high, and the corresponding error will be higher.

Horn and Schunk work out the previous equation using a digital estimation of the Laplacian for the optical flow gradients, to get a large system with two equations for each pixel.

$$(\alpha^2 + I_x^2 + I_y^2)(u - \overline{u}) = -I_x[I_x\overline{u} + I_y\overline{v} + I_t] \tag{6}$$

$$(\alpha^2 + I_x^2 + I_y^2)(v - \overline{v}) = -I_y[I_x\overline{u} + I_y\overline{v} + I_t] \tag{7}$$

where $\overline{u}, \overline{v}$ are the averages of that component of the flow vector (u, v) in a small neighborhood around the current pixel(x, y), and which arise from using the Laplacian operator to find the optical flow gradients. I_x, I_y are the spatial gradients and I_t is the temporal gradient. α is the weighting term from equation 5 for the error of the optical flow's dissimilarity to its neighbors.

The result of all this is a huge number of equations for solve. Horn and Schunk propose solving this large system by an iterative method. Each flow vector component is calculated using the previous iteration's neighborhood average and the spatial and temporal gradients of that pixel.

$$u^{n+1} = \overline{u}^n - \frac{I_x[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{8}$$

$$v^{n+1} = \overline{v}^n - \frac{I_y[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{9}$$

where (u^{n+1}, v^{n+1}) represents the new optical flow vector at iteration $n+1$ and $(\overline{u}^n, \overline{v}^n)$ represents the neighborhood average of the flow vector at iteration n. I_y, I_x are the spatial gradients and I_t is the temporal gradient. α is the weighting term from equation 5 for the error of the optical flow's dissimilarity to its neighbors. The optical flow vector components for each pixel are calculated using two elements: the previous iteration's average flow vector and the current pixel's spatial and temporal gradients. The latter serves to keep the current vector close to the proper value. It constantly reinforces the optical flow constraint equation at each iteration.

One of the effects of this approach, often run for 100 iterations, is that vectors will be propagated to fill in empty regions of the optical flow field which correspond to homogenous regions in the image. A simple way to conceptualize this idea is to consider the optical flow field generated by a uniform square object. At the corners, we have both components of the flow vector. Along the edges, we will only have the component of the flow vector along the gradient. As the iterations run on, the component that is missing from the edges will be filled in, propagated from the corners, down along the edges. Inside the square, there will be no optical flow since there is no gradient to latch onto. As the iterations run on, the corners and the edges will propagate their values into the square and fill it in until everything in the square has the same uniform flow field.

3.1.2 Lucas and Kanade

One of the more popular methods for optical flow computation is Lucas and Kanade's [19] local differential technique. This method involves solving for the optical flow vector by assuming that the vector will be similar to a small neighborhood surrounding the pixel. It uses a weighted least squares method to approximate the optical flow at pixel (x, y).

$$E_{\mathbf{v}} = \sum_{\mathbf{p}\in\Omega} W^2(\mathbf{p})[\nabla I(\mathbf{p}) \bullet \mathbf{v} + I_t(\mathbf{p})] \tag{10}$$

where $\nabla I(\mathbf{p})$ and $I_t(\mathbf{p})$ represents the spatial gradient and temporal gradient at neighboring pixel $\mathbf{p}$ respectively. $\mathbf{v}$ is the optical flow vector for pixel (x, y) and $W(\mathbf{p})$ is the weight we associate with that neighboring pixel.

For each pixel we find an optical flow vector consistent with the neighboring spatial and temporal gradients. We consider a surrounding neighborhood Ω of size n where each neighbor is represented as $\mathbf{p}_i$. Ignoring the weights for an instant, this equation simply sums the error of applying the flow vector $\mathbf{v}$ to the spatial and temporal gradients of all the surrounding neighbors using the optical flow equation of Equation 3. If this considered flow vector is inconsistent with the spatial and temporal gradients of some neighbors, the error will be higher. The weights are used to diminish the importance of distant neighbors. The farther the neighbor is from the pixel the smaller the associated weight. In this way, the influence of more distant pixels is reduced.

This least squares problem can be solved by

$$A^T W^2 A\mathbf{v} = A^T W^2 \mathbf{b} \tag{11}$$

$$A = \begin{bmatrix} \nabla I(\mathbf{p}_1) \\ \nabla I(\mathbf{p}_2) \\ \cdots \\ \nabla I(\mathbf{p}_n) \end{bmatrix} \quad W = diag[W(\mathbf{p}_1), W(\mathbf{p}_2), \cdots, W(\mathbf{p}_n)] \quad \mathbf{b} = \begin{bmatrix} -I_t(\mathbf{p}_1) \\ -I_t(\mathbf{p}_2) \\ \cdots \\ -I_t(\mathbf{p}_n) \end{bmatrix}$$

where A is a vector of all the spatial gradients of all the n neighbors of the neighborhood Ω, W are the weights for each neighbor and $\mathbf{b}$ is a vector of the temporal gradients.

Solving for $\mathbf{v}$ gives us

$$\mathbf{v} = (A^T W^2 A)^{-1} A^T W^2 \mathbf{b} \tag{12}$$

Where

$$A^T W^2 A = \begin{bmatrix} \sum W^2(\mathbf{p}) I_x^2(\mathbf{p})) & \sum W^2(\mathbf{p})) I_x(\mathbf{p}) I_y(\mathbf{p}) \\ \sum W^2(\mathbf{p}) I_x(\mathbf{p}) I_y(\mathbf{p}) & \sum W^2(\mathbf{p}) I_y^2(\mathbf{p}) \end{bmatrix}$$

and can be easily found by summing the spatial derivatives I_x, I_y of the neighborhood surrounding the current pixel along with the associated weights. If this matrix is invertible, $\mathbf{v}$ can be found.

This technique has numerous advantages. Firstly, the support for the flow vector is local rather than global like the iterative technique of Horn and Schunk. This means that we have a good estimate without having to rely on the entire image. For some images with large homogenous regions, a global method may produce satisfactory results but for most cases, the flow vectors of different regions should not impact separate regions. Iterative techniques such as Horn and Schunk allow vector information to spread out over the image to possibly different regions. Reinforcement of the constraint equation can serve to mitigate this, but the problem remains. Imagine two occluding objects passing, both with similar spatial gradients but with different orthogonal components. The iterative scheme will merge these two flow field regions at the boundary and will not preserve the sharp discontinuity. The vectors produced by local techniques will not suffer this problem. The downside is that the homogeneous regions will not be filled in. I would argue however that this is exactly what is desired from an initial optical flow calculation. The task of deciding and filling in homogenous regions is obviously important but should be accomplished at a later stage in the process, using the initial local estimates as input. In this way, the user has more control over the final optical flow field and will probably produce better results.

In Barron's [17] empirical evaluation of nine different optical flow computation techniques, Lucas and Kanade's was one of two that they recommended as performing consistently and robustly. Another advantage is the ease of implementation.

3.2 Correlation Based Methods

Correlation is an intuitively obvious method for the computation of flow vectors. It uses the optical flow constraint's assumption that displaced pixel values generally do not change between frames to apply a similarity measure between the neighborhood of the pixel in the first image and the neighborhood of a candidate pixel in the second image. If this similarity measure is maximized,

then we assume that the distance between these two pixels is the optical flow vector. It should be pointed out that this method will result only in integer components u and v to the optical flow vector as opposed to the previous methods which can produce floating point values. The magnitude for both approaches can be real numbers though. Mathematically, the similarity measure is simply

$$\int_{\mathbf{x}\in\mathbf{\Omega}} \phi(I_1(x,y), I_2(x+\delta x, y+\delta y)) \tag{13}$$

and is the sum of some function ϕ applied to all the neighbors $\mathbf{x}$ of the pixel (x, y) for some displacement $(\delta x, \delta y)$ between the two images I_1 and I_2. Common similarity measure are the product of the pixel values or the sum of squared difference(SSD). The displacement which maximizes the similarity measure gives the flow vector sought. A simple example of this can be found in the work of Bulthoff et al [14]. They define a maximum displacement of n in each direction for each pixel. Each pixel in the $n \times n$ search area is compared using a neighborhood of size v using the SSD similarity measure. The lowest value found in the search area defines the proper displacement.

Correlation-based methods have a long history in the analysis of stereo images where correlation is a simple and useful method in determining the displacement between two images, taken at the same time but from different locations. They do have a number of attractive qualities. For one, they tend to be less sensitive to noise than differential methods. Even if the neighborhood surrounding a pixel changes between frames, as long as the proper neighborhood maintains a greater similarity relative to others, it will continue to be chosen. They also tend to be less demanding of significant image structure for producing reasonable results and are also useful for determination at occluding boundaries.

Correlation methods are far from perfect however. The great difficulty is the expense incurred by having to perform the similarity measure between neighborhoods for a number of different displacements for each pixel. In reality, there can be large displacements in image sequences, much greater than in stereo images. If the area surrounding the pixel is too small and the displacement exceeds it, then the optical flow vectors are essentially useless. There have been methods, such as Ogato and Sato's [23], which calculate a rough estimate of the optical flow and then use a correlation method to match properly. There are also issues of ambiguity which arise when performing correlation methods. In homogenous areas and along edges, the aperture problem still exists and will result in ambiguous results. Often, a smoothness constraint such as Horn and Schunk's [15] is used to make decisions in these cases. As we shall see later, hierarchical methods can also be quite useful for both large displacements and for ambiguous areas.

There is also an issue in the size of the neighborhood or window used for the similarity measure. The window must be large enough to contain enough detail that a proper match can be found. It must be proportional to the size of the search area for the displacement otherwise ambiguous matches arise. On the other hand, the window must not be so large that it will be fooled by change within the neighborhood itself. While the image brightness constancy assumption is wonderful, in reality, noise and change do occur, especially over larger areas and this must be taken into account when choosing the window size.

In practice, though there are some methods to produce dense optical flow fields, correlation tends to be used mostly in conjunction with feature based methods. A series of features are chosen from the image and then correlation is used to determine the optical flow vectors. As we shall see later, these methods are cheaper since only a certain number of pixels are being checked. They also tend to bypass the aperture problem by simply not choosing features where this is an issue. The downside is the sparser flow field produced.

3.2.1 Barnard and Thompson

The work of Barnard and Thompson [3] provides a good example of a more sophisticated correlation-based method. Firstly, although this method is feature-based, it does use correlation to determine similarity between points. It then uses a probabilistic relaxation labeling algorithm to help make the final determination for the flow vector.

Features are first chosen by applying a simple interest operator described by Moravec [22] to the entire image. Each feature pixel in the first image is defined as a node a_i and then assigned a series of labels l_j which correspond to potential matches in the second image. All potential pixels within a distance r of a_i will be given such a label. The algorithm then determines a probability associated with each of the labels. Initially, the sum of the square difference (SSD) between a neighborhood surrounding the feature pixel and the candidate pixel are found. We represent these SSD values as $s_i(l)$ for node a_i and label l. A SSD values closer to 0 represents a more similar region, so we first invert these values to get an associated weight for each label that increases with the similarity.

$$w_i(l) = \frac{1}{1 + cs_i(l)} \tag{14}$$

where c is a positive constant and $s_i(l)$ was the SSD value for node a_i and label l. Therefore, an area which has a high similarity with the SSD measure, and therefore a low SSD value, will have a high weight value. These weights are then normalized to obtain the probabilities.

$$p_i(l) = \frac{w_i(l)}{\sum_{l'} w_i(l')} \tag{15}$$

where for each label l for a_i, we divide the weight by the sum of the weights for all the labels. This gives us the probability this label is correct. Once these initial probabilities are determined, an iterative relaxation algorithm is applied to update the probability of l based on the similarities of this displacement to neighboring displacements which are below some threshold Θ.

$$q_i^k(l) = \sum_{a_j near a_i} \left[\sum_{l' s.t. \|l - l'\| \leq \Theta} p_j^k(l') \right] \tag{16}$$

where the sum is over all the nodes a_j which are within some distance of a_i. For each of these close nodes, we take a further sum of all the labels l' which are within a threshold Θ of the considered label l. In this case, we mean the difference between the associated displacements for each label. The sum of all probabilities for nearby similar displacement labels are then used to update the current label's probability. Intuitively, we are simply performing a process similar to the smoothness constraint of Horn and Schunk. We are looking to maximize the similarity of the label's displacement within a neighborhood, providing a smooth flow field.

$$\hat{p}_i^{k+1}(l) = p_i^k(l)(\alpha + \beta q_i^k(l)) \tag{17}$$

where α, β are parameters controlling how much influence the neighboring values should have. $p_i^k(l)$ is the probability associated with label l at iteration k. $q_i^k(l)$ is the previous measurement of the sum of similar label's probability in a surrounding neighborhood for iteration k. We use these values to compute a new probability for label l at iteration $k+1$.

Finally, the probabilities are normalized.

$$p_i^{k+1}(l) = \frac{\hat{p}_i^{k+1}(l)}{\sum_{l'} \hat{p}_i^{k+1}(l')} \tag{18}$$

This process is then repeated for a number of iterations or until stabilization occurs. The labels for each node are then checked and the label with the highest probability will be chosen, if it exceeds some threshold, as the optical flow vector.

4 Feature Based Methods

Many of the methods previously discussed produce dense optical flow fields, that is, a flow vector for a significant number of the pixels. The disadvantage, which was previously discussed in Section

2, is the accuracy of many of these vectors will be questionable. In homogenous regions the flow vectors are undefined and may produce unreliable results. In areas with edges, only the component orthogonal to the edge can be found for the optical flow vector. The component along the edge is missing due to the aperture problem. Only those points in the image function which contain enough gradient, in at least two directions, for a unique correspondence to be made will produce dependable flow vectors.

This leads to the obvious question: why not simply ignore all the unreliable vectors and deal only with certain ones? This approach motivates feature based methods and gives rise to two subproblems: detection of features and determining correspondences between images. The methods for matching of feature points between images are most often correlation methods. Conventional correlation, squared correlation, absolute value difference, sum of squares difference(SSD), and various normalization factors have all been used, often with good results. Further processing and smoothing is often done after, but correlation is the standard method for determining similarity.

Next, it is important to distinguish which features we are interested in pursuing. Using only high curvature two-dimensional features(ie.corners or junctions) to determine unique correspondences is a common and popular method. However, this approach ignores the great deal of information which is present in the edges, that of the orthogonal component of each flow vector. Therefore, another class of feature based methods have arisen which attempt to "fill" in the missing component by propagating information from points of high curvature. We therefore continue this discussion with a description of a variety of "corner" based methods, followed by edge based methods.

4.1 Two Dimensional Features

One of the main issues for two dimensional feature methods is the detection of features which will produce valid flow vectors. Though this field is not strictly part of the field of optical flow, there is enough overlap to justify some brief words on popular methods.

One of the most common detection methods is the Moravec operator [22]. Directional variance is determined by the minimum of the autocorrelation of each pixel in 4 directions (vertical, horizontal, and the two diagonals). This can easily be seen by considering 3 cases. First, in a homogenous region all four values will be low so then will the minimum be also. Secondly, for an edge, the values will be high along the directions of the gradient but low along the edge direction. The minimum will thus also be low. Lastly, for a corner, the variance will be high along all directions so too will the minimum be high.

The Plessey operator is an improvement to the Moravec operator which was proposed by Harris and Stephens [12]. One of the issues with the Moravec operator is anisotropism. That is, the values can change depending on the directions in which the measurements were made. The response is also often noisy due to the square correlation window used. Also, since only the minimum of the autocorrelation function is used, the operator responds too quickly to edges. The Plessey operator is similar to the Moravec operator in that it also uses autocorrelation but critically, it uses first derivatives so that small directional shifts can be found rather than the 45° shifts of the Moravec operator. For a small shift (x, y), the error is defined as

$$E(x, y) = (x, y)M(x, y)^T \tag{19}$$

$$M = \begin{bmatrix} X^2 \otimes m & XY \otimes m \\ XY \otimes m & Y^2 \otimes m \end{bmatrix} \quad X = \frac{\partial I}{\partial x} \; Y = \frac{\partial I}{\partial y}$$

Where X, Y are the first order derivatives of the image function, m is a small gaussian window surrounding the current pixel, and $\otimes$ is the correlation function. The shift values are used to influence the error function; the farther the shift, the greater the error.

E is quite similar to the autocorrelation function. The eigenvalues of M are proportional to the principle curvature of the autocorrelation function and are also rotationally invariant.

Lastly, we consider a non-linear method of feature detection which has been recently proposed by Smith and Brady [26]. In their SUSAN detector, a small circular mask is moved along the image function. At each pixel, called the nucleus, the number of pixels within the mask which have an intensity value within some threshold of the nucleus are counted. This area of similar intensity values is called the "USAN" or more verbosely, the "Uniform Segment Assimilating Nucleus". To go back to a previous 3 cases, in areas with homogeneous intensity, the USAN value will be high relative to the mask size. Along straight edges, the USAN value will decrease but will still be significantly high. Finally, in corners or junctions, the USAN value will be relatively small. Their SUSAN approach involves looking for minima along this function, called the "Smallest Uniform Segment Assimilating Nucleus". In practice, points along this function which have less than half the mask size are considered two-dimensional features since SUSAN values along straight edges will always be greater than half the mask size because the nuclei will be on one side of the edge. The SUSAN detector is also robust in the presence of noise; image derivatives are not considered, so noisy values tend to be influence a smaller area.

Another important class of feature detectors includes work by Beaudet [5], Dreschler and Nagel [9], among many others, which search for corners at regions in the image function where there is high gaussian curvature. There are also many other methods which have been explored by the community, far too many to discuss in this short paper.

4.2 Two Dimensional Features Methods

We have already seen an example of a method which uses two-dimensional features in the previous section on correlation. Work by Barnard and Thompson [3] involved using the Moravec operator to determine feature points and using a sum of squares difference correlation measure to similarity of nearby pixels. Further label relaxation is done after this, using the similarity measure to smooth the optical flow field. For greater detail, see Section 3.2.1. An almost identical approach was taken by Burger and Bhanu [7] in their work on determining 3D egomotion. They also used both the Moravec operator and SSD correlation to determine optical flow. However, they did not use the more advanced label relaxation of Barnard and Thompson.

Lawton's work [18] is another good example of this class of methods. Initially, a Laplacian of Gaussian filter is applied to input image and the zero crossings are used to determine edges. The curvature at each edge point is then found by taking the dot product of the normalized vectors which describe the direction to the adjacent edges along the contour. Edges with high curvature will have values near 1.0 where edges with low curvature will have values nearer -1.0. These are thresholded to isolate the points of high curvature. He then uses normalized correlation as a similarity metric to match features.

4.3 Edge Feature Methods

In this section, we discuss methods of optical flow calculations around edge features. We will omit a discussion on common edge detection methods since they are a widely understood area of computer vision that far outreach the field of optical flow.

One of the earliest examples of calculating flow vectors along edges is the work of Ellen Hildreth [10]. In her work, she calculates edges along surfaces using the common rigid body assumption: the contour edge of an object is rigid and any change is based solely on movement of the object. Initially, she uses the Laplacian of Gaussian (LoG) operator to find the zero crossings which correspond to edges. These edges are then combined to form chains or contours and an optical flow is calculated for each of these edge points, using a derivative based method for instance.

Then a smoothness constraint is used to restrict the variation of flow vectors along the contour S.

$$E_1(\mathbf{V}) = \int_S \left(\left(\frac{\partial u}{\partial s} \right)^2 + \left(\frac{\partial v}{\partial s} \right)^2 \right) \mathrm{ds} \tag{20}$$

where $\mathbf{V} = (u_1, u_2, ..., u_n, v_1, v_2, ..., v_n)$ is the set of flow vectors and $\frac{\partial u}{\partial s}\frac{\partial v}{\partial s}$ are the changes of the flow vector along the contour S. This error is simply the squared sum of the change is the flow vectors along a contour. If you imagine a straight edge contour, there would be little change between points on the contour so this error would be low. If the contour fluctuates a great deal, this error will be higher.

An additional image constraint is needed however to maintain the correspondence between the newly calculated vectors and those components that were determined from the spatial derivatives of the image.

$$E(\mathbf{V}) = E_1(\mathbf{V}) + \beta \int_S \left(\mathbf{v} \cdot \mathbf{u}^\perp - v^\perp\right) \mathtt{d}s \tag{21}$$

where the dot product of the calculated flow vector $\mathbf{v}$ and the edge direction $\mathbf{u}^\perp$ for each point is taken and subtracted by the original flow vector $v^\perp$. β is a weighting factor of the confidence of that first flow vector. This subtraction of the originally calculated flow vector, which corresponds to the spatial gradient, keeps the vectors of $\mathbf{V}$ from deviating too much from the spatial component of the flow vector. This equation is quite similar to Horn and Schunk's approach. The first term is a smoothness constraint which serves to keep the vectors from deviating from their neighbors. The second term reinforces the measured spatial component of the flow vector at that point.

It is also important that the points along the contour are evenly spaced for the above function to work correctly. If the points are not equidistant, then a further weighting is necessary to normalize the influences. To solve for the flow vectors, Hildreth proposes using gradient descent methods to find the minimum of the error function $E(\mathbf{V}) = (u_1, u_2, ..., u_n, v_1, v_2, ..., v_n)$.

One of the criticisms of Hildreth's approach, and one she acknowledges is her paper, is the computational expense of performing gradient descent for each countour in the image. This sequentiality of the algorithm is one reason for the expense incurred. This problem was addressed by Gong and Brady [11] in their work on parallel computation of edge-based optical flow.

Their approach used a wave-diffusion algorithm to spread the parallel components along the edge. The vectors from the points of high curvature travel along the countour influencing the vectors with lower curvature. Since the information will only spread so far, the algorithm can be run in parallel with sections of the countour being influenced by local high curvature points being combined in a later stage. The basic form of the recursive wave diffusion equation is

$$v(s, t+1) = v(s,t) + C^2 \frac{\partial^2 d(s,t)}{\partial s^2} \tag{22}$$

$$d(s, t+1) = v(s,t) + d(s,t) \tag{23}$$

Where $v(s,t)$ is the velocity of the wave equation and $d(s,t)$ is the wave displacement at time t and at contour point s. C is the speed the wave propagates along the countour. The initial values for the functions are $d(s,0) = v^\perp$ i.e. the initial measured displacement, and $v(s,0) = 0$. The equations are repeated until the wave from high curvature points have reached waves from other high curvature points and we have found the final displacement value of the contour.

5 Hierachical Methods

One of the significant practical differences between the analysis of stereo images and that of temporal image sequences is that, in the latter, displacements can often occur over large areas of the screen. If an object or camera is traveling quickly, displacements could conceivably occur to anywhere on the image. This problem is further exasperated if the frame rate of the image sequence is too low to capture the motion over many frames. For correlation based methods, useless optical flow values can result if the displacement is greater than the search area. This is also a problem for differential methods of optical flow calculation where the accuracy of using spatial and temporal gradients decreases with large displacements.

A common solution is to use a hierarchical approach whereby a series of images are created from the input image with decreasing spatial frequency, i.e. a decreasing resolution. Often this takes the form a pyramid data structure such as a Gaussian or Laplacian pyramid. Optical flow analysis is done on images higher in the pyramid which will have a lower spatial frequency and will thus correspond to greater displacements. Results from these are then projected downwards onto the next level of the pyramid where they are used as an initial guess which is then further refined. Once it has been refined, it will then be passed onto the next level. The process continues until the bottom of the pyramid or the input image is reached.

5.0.1 Anandan

One of the well known hierarchical methods is that of Anandan [2]. Although it is a correlation based method, in his paper Anandan describes a general framework for hierarchical optical flow calculation which he shows is equivalent to well-known hierarchical methods that use differential techniques. His algorithm initially creates a Laplacian pyramid for each image, usually of 3 levels. To determine the displacement at each level in the pyramid, the sum of squares difference is used. Interestingly, the surface of the SSD values are also analyzed to determine a confidence value for each flow vector. When the minimum of the SSD surface in the search area is found (corresponding to the best displacement), the curvature of the surface at that point is found along the two principle axes: the direction of maximum curvature and the direction of minimum curvature. The axes are denoted by the unit vectors $\hat{e}_{max}$ and $\hat{e}_{min}$ and the corresponding curvatures are C_{max} and C_{min}. The confidence measure are two values determined from both the curvature and the SSD at the minimum point S_{min}.

$$\begin{aligned} c_{max} &= \frac{C_{max}}{k_1 + k_2 S_{min} + k_3 C_{max}} \\ c_{min} &= \frac{C_{min}}{k_1 + k_2 S_{min} + k_3 C_{min}} \end{aligned}$$

where k_1, k_2, k_3 are used to normalize the confidence measures.

The minimum of a quadratic approximation of the SSD surface is also used to determine sub-pixel displacements for the following level in the pyramid.

Finally, the optical flow field is also smoothed at each stage to allow for a more uniform field. This smoothing is done by minimizing the following error term over all the displacement vectors.

$$E^2 = \int\int \left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right) + c_{max}(\mathbf{v}\hat{e}_{max} - \mathbf{d}\hat{e}_{max})^2 + c_{min}(\mathbf{v}\hat{e}_{min} - \mathbf{d}\hat{e}_{min})^2 \mathrm{dxdy} \tag{24}$$

where $\mathbf{v} = (u, v)$ is the displacement vector produced by the iterative minimization of this error equation and $\mathbf{d}$ is the displacement vector produced by finding the minimum in the SSD search area. Notice that the first term of the error equation is the same smoothness constraint used by Horn and Schunk described earlier in the paper.

For each level, Anandan's algorithm determines the displacements using the SSD and then smoothes the field using equation 24. Once the field has been smoothed it is then projected downwards in the pyramid until the input image has been reached and we have the proper optical flow field.

6 Applications of Optical Flow

6.1 Differences of Flow Types

The optical flow field is a projection of real world motion onto an image plane. As such, there are a number of real-world motions which can cause these velocity vectors, either by the camera or objects

within the scene moving. The two fundamental motions which must be considered are translation and rotation. In Figure 3 we can see the three types of translational motion. On the left, we see a purely translational motion where the camera(or object) is moving backwards. This produces a field of converging vectors around a point called the focus of contraction(FOC). In the middle field, we can see that the camera(or object) is moving forwards so the flow vectors diverge around a point called the focus of expansion. On the right field, there are 4 objects moving perpendicular to the image plane which produces a field of parallel flow vectors which converge at infinity.

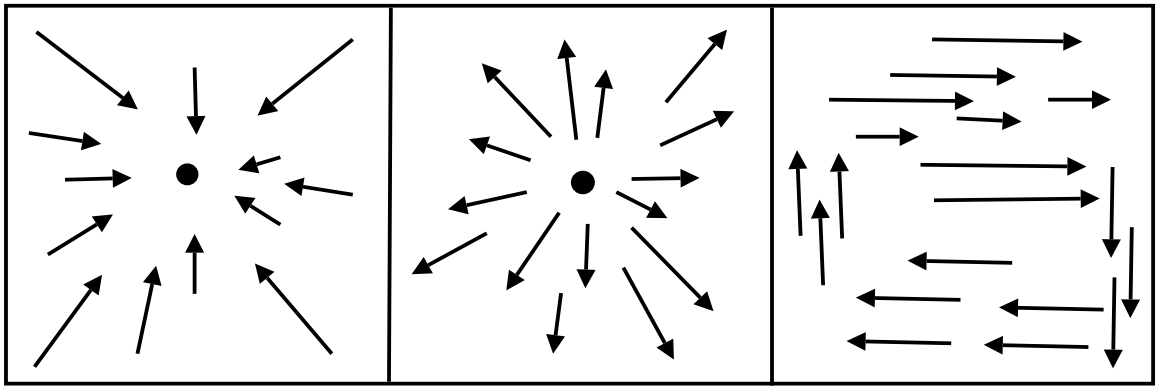

Figure 3: The three types of optical flow vectors produces by translational motion.

Rotational motion also produces flow vectors which can be easily imagined. An interesting problem that arises in the interpretation of optical flow fields is the type of motion the camera or an object within the image is undergoing, especially as the motions become more complex. For example, a flow field of a zooming and panning camera observing rotating balls bouncing. There has been work done to isolate the translational and rotational components, for example Riger and Lawton's algorithm [24] is able to determine the components when there is a great deal of depth variation in the scene.

6.2 Time to Contact and Focus of Expansion

One the most practical uses for an optical flow field is the determination of time-to-contact or time-to-collision. If you can imagine an autonomous robot moving through an environment, being able to determine the time until the robot reaches a particular area, assuming constant velocity, is a valuable piece of information. Interestingly, this can be found without any knowledge of the distance to be traveled or the velocity the robot is moving. Note that we need to have access to the translational component of the flow field; the rotational component will have no affect on these calculations.

To find this, first we need to be able to calculate the focus of expansion of a flow field. Theoretically, all we need are two vectors since we can simply take the lines defined by these two vectors and determine where they meet. This will be the focus of expansion. Of course in reality, noise and other errors from the many steps to reach this point will result in imperfect optical flow vectors. A natural solution, proposed by Tistarelli et al [21], is to take the least squares solution of all flow vectors and use this to find the the focus of expansion.

$$FOE = (A^T A)^{-1} A^T \mathbf{b} \tag{25}$$

$$= \begin{bmatrix} \sum a_{i0} b_i \sum a_{j1}^2 - \sum a_{i1} b_i \sum a_{j0} a_{j1} \\ -\sum a_{i0} b_i \sum a_{j0} a_{j1} + \sum a_{i1} b_i \sum a_{j0}^2 \end{bmatrix} - \frac{1}{\sum a_{j0}^2 a_{j1}^2 - (\sum a_{i0} a_{i1})^2} \tag{26}$$

$$A = \begin{bmatrix} a_{00} & a_{01} \\ \cdots & \cdots \\ a_{n0} & a_{n1} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_0 \\ \cdots \\ b_n \end{bmatrix}$$

where, for each pixel $p_i = (x, y)$, the associated flow vector $\mathbf{v} = (u, v)$ gives $a_{i0} = v$, $a_{i1} = u$, $b_i = xv - yu$

Other methods for determining the focus of expansion also exist, including an interesting method used by Hatsopoulos and Warren [13] using a two layer neural network.

Now that we have found the focus of expansion, we are equipped to calculate the time-to-contact. To keep the following discussion simplified, we will assume that the camera is facing the same direction as the direction of motion. We are also concerned only with objects impacting which are close to the direction of travel. To deal with objects and camera motion which do not follow these assumptions, we need a more general framework, including using the optical flow field to calculate depth values and find the structure of the scene.

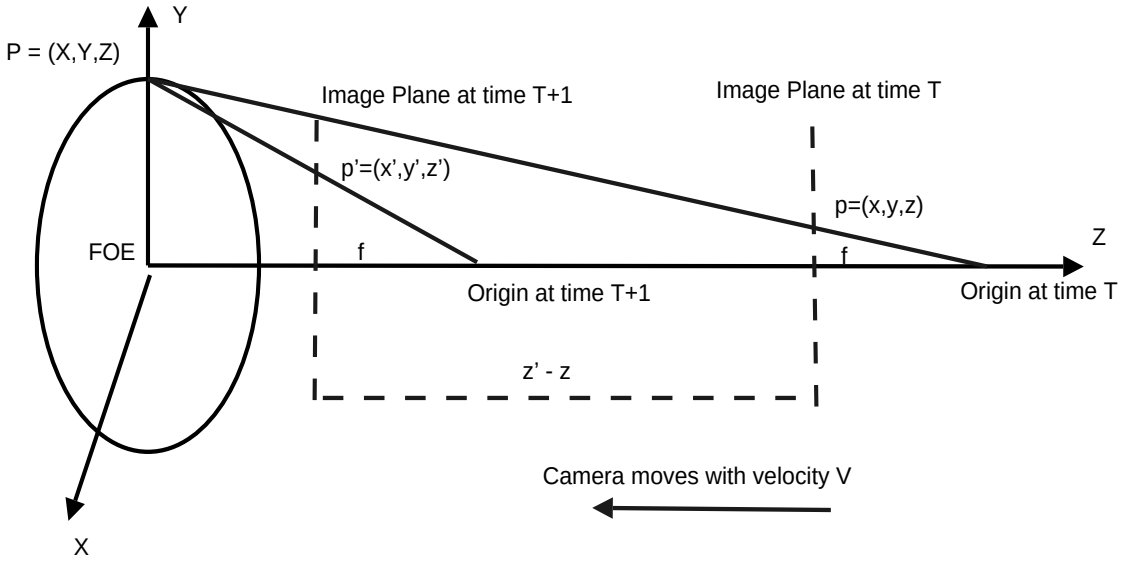

Figure 4: The projections of a point P onto the image plane of a moving camera.

Now consider Figure 4 which illustrates this graphically. This diagram and much of the derivation were interpreted from Ted Camus' PhD thesis on Real-Time Optical Flow [8]. We observe that there is a point $P = (X, Y, Z)$ which projects through the image plane at time T to point $p = (x, y, z)$. The image plane is positioned a distance f from the origin. The camera moves along the z-axis, at a velocity $V = \frac{\partial Z}{\partial t}$ over a distance $\Delta z = z' - z$, approaching the focus of expansion. At time T+1, the point p now projects to a new point on the image plane $p' = (x', y', z')$.

From similar triangles we know that

$$\frac{y}{f} = \frac{Y}{Z}$$

or equivalently,

$$y = f\frac{Y}{Z}$$

We now differentiate y with respect to time so that

$$\frac{\partial y}{\partial t} = f\left(\frac{\frac{\partial Y}{\partial t}}{Z}\right) - fY\left(\frac{\frac{\partial Z}{\partial t}}{Z^2}\right)$$

We can simplify this by noting that $\frac{\partial Y}{\partial t} = 0$ since Y never changes. We can substitute $Y = \frac{yZ}{f}$ and also, $\frac{\partial Z}{\partial t} = V$ by our previous definition.

$$\frac{\partial y}{\partial t} = -y\left(\frac{V}{Z}\right)$$

If we now divide $\frac{\partial y}{\partial t}$ by y and take the reciprocal, we get

$$\frac{y}{\frac{\partial y}{\partial t}} = -\frac{Z}{V} = \tau \tag{27}$$

where this quantity τ is the time-to-contact. It is possible to get some intuition behind this equation by perceiving that the distance Z, divided by the velocity V gives the time until impact, which is what one would hope for. The reason that the negative of the fraction is needed is due to the fact that the distance is negative for this example.

Now we can observe the interesting result; to find the time-to-contact of the camera, we only need the optical quantities y and $\frac{\partial y}{\partial t}$, both of which can be easily calculated from an image sequence observing a motion with constant velocity.

Armed with this result, we now observe that we the length of the optical flow vector gives us $\frac{\partial y}{\partial t}$, and the distance from the pixel to the FOE gives us the y value. With these two values, we can easily calculate τ.

6.3 Motion Parameters and Depth

The optical flow field is a vast mine of information of the observed scene. The greatest of these low-level applications of optical flow is the determination of the motion parameters and depth of objects in the scene. Unfortunately, we are only able to calculate relative information rather than absolute. If you consider a camera watching a moving homogenous region, it is impossible without knowledge to determine whether the object representing the region is moving or is stationary and the camera moving. As will be discussed later, this is also true for depth. It should also be noted that the following discussion assumes that all the flow vectors arise from a single motion. That is, segmentation(see section 6.4) for different moving regions has already been performed so we have only a single set of motion parameters to find.

To begin, we formalize the problem and derive the proper equations for motion. Most of this discussion was taken, and simplified, from Adiv's paper [1] on motion parameters. In figure 5 we can see the a motion captured in a perspective projection. A point (X, Y, Z) is projected onto the image plane at (x, y) at time t and the new displaced point (X', Y', Z') projects to (x', y') at time t'.

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} 1 & -R_z & R_y \\ R_z & 1 & -R_x \\ -R_y & R_x & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{28}$$

As we can see, the new point (X', Y', Z') is a function of the previous point (X, Y, Z) multiplied by a matrix of rotation parameters (R_x, R_y, R_z) and added with the translation parameters (T_x, T_y, T_z). This small rotation matrix will approximate the real rotation if small values are used [16].

We can now use these motion parameters and substitute them into the projection equation for x' and y'.

$$x' = \frac{X'}{Z'} = \frac{x - R_z y + R_y + \frac{T_x}{Z}}{-R_y x + R_x y + 1 + \frac{T_z}{Z}} \tag{29}$$

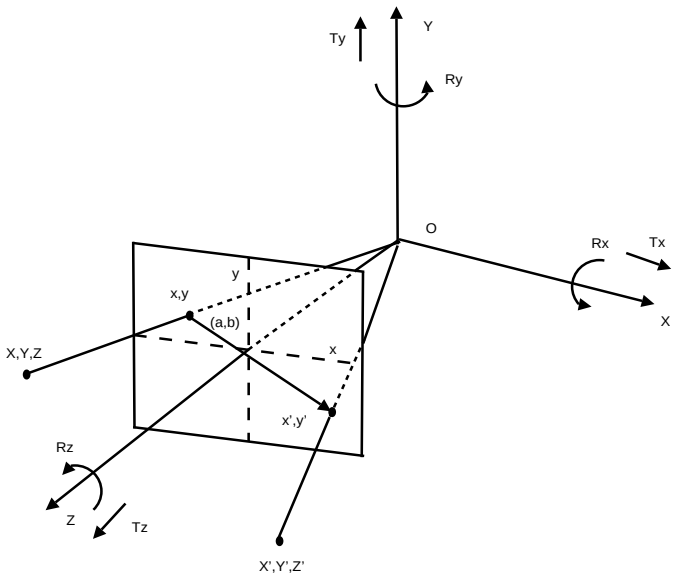

Figure 5: Perspective projection of a scene. The world view coordinate system is $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and the image plane $(\mathbf{x}, \mathbf{y})$. A point (X, Y, Z) moves to point (X', Y', Z') according to translation (T_x, T_y, T_z) and rotation (R_x, R_y, R_z), the optical flow vector (a, b) shows the displacement of the projection onto the image plane.

$$y' = \frac{Y'}{Z'} = \frac{R_z x + y - R_x + \frac{T_y}{Z}}{-R_y x + R_x y + 1 + \frac{T_z}{Z}} \tag{30}$$

This gives us the new pixel position (x', y') based on the previous pixel position (x, y), rotational (R_x, R_y, R_z) and translational motion parameters(T_x, T_y, T_z), and the depth Z.

Since the displacement vector $(\alpha, \beta) = (x' - x, y' - y)$, by subtracting x and y from the previous equations we get

$$\alpha = \frac{-R_x xy + R_y(1 + x^2) - R_z y + \frac{(T_z - T_z x)}{Z}}{-R_y x + R_x y + 1 + \frac{T_z}{Z}} \tag{31}$$

$$\beta = \frac{-R_x(1 + y^2) + R_y xy + R_z x + \frac{(T_y - T_z y)}{Z}}{-R_y x + R_x y + 1 + \frac{T_z}{Z}} \tag{32}$$

which gives an equation relating the motion parameters of the point to the measurable optical flow vector. The end result will perform the reverse and use the measured optical flow field to calculate those motion parameters. We can also simplify the equations slightly if we make some small assumptions: a small depth translation T_z compared to the depth Z, a small field of view and small rotation parameters. We can also divide the equation into the translational and rotational components.

$$(\alpha, \beta) = (\alpha_R, \beta_R) + (\alpha_T, \beta_T) \tag{33}$$

$$\alpha_R = -R_x xy + R_y(1 + x^2) - R_z y \quad \beta_R = -R_x(1 + y^2) + R_y xy + R_z x$$
$$\alpha_T = \frac{(T_z - T_z x)}{Z} \qquad \beta_T = \frac{(T_y - T_z y)}{Z}$$

These reduced equations for determining the optical flow vector from position and motion parameters allow us to make some interesting observations. Firstly, as we can see from the previous equations, the effect of the rotational parameters and the translational parameters on the optical flow vector can be isolated into two separate components $(\alpha_R, \beta_R) + (\alpha_T, \beta_T)$ Secondly, the rotational components do not depend on the depth of the point.

We now require a way to take our optical flow field and, using this equation, determine the rotational and translational components including depth. A simple approach using least squares was proposed by Bruss and Horn [6]. Each set of motion parameters defines an optical flow vector for each position. The deviation that all the measured optical flow vectors make from these predictions gives the error for those motion parameters. If the motion parameters describe the motion perfectly and there is no error in computing the optical flow vectors, this error will be zero. Of course, these assumptions will rarely hold so the task it to choose motion parameters to minimize this error function.

$$E(\mathbf{T}, \mathbf{R}, Z) = \sum_{i=1}^{n} (\alpha_i - \alpha_{Ri} - \alpha_{Ti})^2 + (\beta_i - \beta_{Ri} - \beta_{Ti})^2 \tag{34}$$

where $\mathbf{T} = (T_x, T_y, T_z)$ are the translational parameters, $\mathbf{R} = (R_x, R_y, R_z)$ are the rotational parameters, Z is the depth, (α_i, β_i) is the optical flow vector for pixel i, $(\alpha_{Ri}, \beta_{Ri})$ and $(\alpha_{Ti}, \beta_{Ti})$ are the rotational components and translational components of the optical flow vector respectively. It was hinted earlier that depth further complicates our equations. The reason is that we have no way of finding the absolute values of (T_x, T_y, T_z) and $Z_i, i = 1...n$. This can be seen by considering a small region moving slowly, such as the sun. It is impossible to determine if that object is truly a small object with a slow speed, or a huge object in the far distance moving quite rapidly. What we can find is the direction of motion and the relative depth of the object. Let $\mathbf{U} = (U_x, U_y, U_z)$ be the normalized direction of motion where r is the magnitude of the translation component $\mathbf{T} = (T_x, T_y, T_z)$.

$$(U_x, U_y, U_z) = \frac{(T_x, T_y, T_z)}{r} \tag{35}$$

and the relative depth is

$$\tilde{Z}_i = \frac{r}{Z_i} \;\; i = 1...n \tag{36}$$

and we can therefore rewrite the translational components to normalized components

$$\alpha_U = \frac{\alpha_T}{\tilde{Z}} = U_x - U_z x \tag{37}$$

$$\beta_U = \frac{\beta_T}{\tilde{Z}} = U_y - U_z y \tag{38}$$

Rewriting the error function with respect to $\mathbf{U}$, we get

$$E(\mathbf{U}, \mathbf{R}, \tilde{Z}) = \sum_{i=1}^{n} (\alpha_i - \alpha_{Ri} - \alpha_{Ui}\tilde{Z}_i)^2 + (\beta_i - \beta_{Ri} - \beta_{Ui}\tilde{Z}_i)^2 \tag{39}$$

where $\mathbf{U} = (U_x, U_y, U_z)$ are the normalized translational parameters, $\mathbf{R} = (R_x, R_y, R_z)$ are the rotational parameters, $\tilde{Z}$ is the relative depth, (α_i, β_i) is the optical flow vector for pixel i, $(\alpha_{Ri}, \beta_{Ri})$ and $(\alpha_{Ui}, \beta_{Ui})$ are the rotational components and normalized translational components of the optical flow vector respectively.

Now comes a slightly tricker part. For all possible values of $\tilde{Z}_i$ we wish to minimize this error function $E(\mathbf{U}, \mathbf{R}, \tilde{Z})$. Therefore, we take the derivative of the error with respect to Z and set it equal to 0. This gives us an optimal value for $\tilde{Z}_i$ at each point in the image according to our motion parameters.

$$\tilde{Z}_i = ((\alpha_i - \alpha_{Ri})\alpha_{Ui} + \frac{(\beta_i - \beta_{Ri})\beta_{Ui}}{\alpha_{Ui}^2 + \beta_{Ui}^2} \tag{40}$$

Finally, we can substitute this value back into the error function to get its last formulation.

$$E(\mathbf{U},\mathbf{R}) = \sum_{i=1}^{n} \frac{[(\alpha_i - \alpha_{Ri})\beta_{Ui} - (\beta_i - \beta_{Ri})\alpha_{Ui}]^2}{\alpha_{Ui}^2 + \beta_{Ui}^2} \tag{41}$$

The importance of getting rid of the Z is that we have formulated the error in terms of just **U** and **R**. We can ignore the depth values and find only the optimal motion parameters. At the end, we can then substitute these back into equation 40 and find the optimal depth value.

6.3.1 Rotational Motion

It was mentioned earlier that rotation is the easier of the two components to find since the motion parameters are depth independent. It is even easier when we realize that the 3 rotational motion parameters interact linearly. Thus, all we need to do is find three linear equations for the parameters and solve with a simple least squares approach developed by Bruss and Horn [6].

Formally, for solely rotational motion, the error function is just

$$E_R(\mathbf{R}) = \sum_{i=1}^{n} (\alpha_i - \alpha_{Ri})^2 + (\beta_i - \beta_{Ri})^2 \tag{42}$$

where $\mathbf{R} = (R_x, R_y, R_z)$ are the rotational parameters, (α_i, β_i) is the optical flow vector for pixel i, $(\alpha_{Ri}, \beta_{Ri})$ are the rotational components of the vector. Again, similarly to what we did with $\tilde{Z}$ in equation 40, we note we really want a value of (R_x, R_y, R_z) which will minimize the error function. Therefore, all we do is differentiate the error function with respect to each of these parameters and set it equal to 0. This gives us the three linear equations we seek.

$$\sum_{i=1}^{n} [(\alpha_i - \alpha_{Ri})xy + (\beta_i - \beta_{Ri})(y^2 + 1)] = 0 \tag{43}$$

$$\sum_{i=1}^{n} [(\alpha_i - \alpha_{Ri})(x^2 + 1) + (\beta_i - \beta_{Ri})xy] = 0 \tag{44}$$

$$\sum_{i=1}^{n} [(\alpha_i - \alpha_{Ri})y + (\beta_i - \beta_{Ri})x] = 0 \tag{45}$$

where (α_i, β_i) is the optical flow vector for pixel $i = (x, y)$, $(\alpha_{Ri}, \beta_{Ri})$ are the rotational components of the vector. We can rewrite these equations and expand them to get them into matrix form.

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = \begin{bmatrix} a & d & f \\ d & b & e \\ f & e & c \end{bmatrix}^{-1} \begin{bmatrix} k \\ l \\ m \end{bmatrix} \tag{46}$$

$$a = \sum_{i=1}^{n} [x^2y^2 + (y^2 + 1)] \quad b = \textstyle\sum_{i=1}^{n} [(x^2 + 1) + x^2y^2] \quad c = \sum_{i=1}^{n} [x^2 + y^2]$$

$$d = \sum_{i=1}^{n} [xy(x^2 + y^2 + 2)] \qquad e = \textstyle\sum_{i=1}^{n} y \qquad f = \sum_{i=1}^{n} x$$

$$k = \sum_{i=1}^{n} [uxy + v(y^2 + 1)] \quad l = \textstyle\sum_{i=1}^{n} [u(x^2 + 1) + vxy] \quad m = \sum_{i=1}^{n} [uy - vx]$$

where (u, v) is the measured flow vector for pixel (x, y).

So finding the rotational parameters is almost trivial. We simply need to do a few summations, a couple of multiplications, and a matrix invertion and we have our rotational motion parameters.

6.3.2 General Motion

Now we tackle the more difficult issue of general motion, that is, finding all six motion parameters. The ease of finding the rotational parameters leads one to ask the question, why not use the same technique for all six? Simply differentiate over all the parameters to get six linear equations and solve them. Unfortunately, it is not that easy. Optical flow cannot capture unambiguously the six motion parameters. The six parameters interact non-linearly so finding six linear equations is impossible. Bruss and Horn [6] use three linear equations for the rotational motion and include four quadratic equations to add in the translational motion. These equations are then solved using a numerical method.

A more intuitive method though is to explicitly search the space of translational motions and determine the optimal rotational parameters. An error function is used to guide the search. This method described here is a simplification of Adiv's algorithm [1] for determining motion parameters.

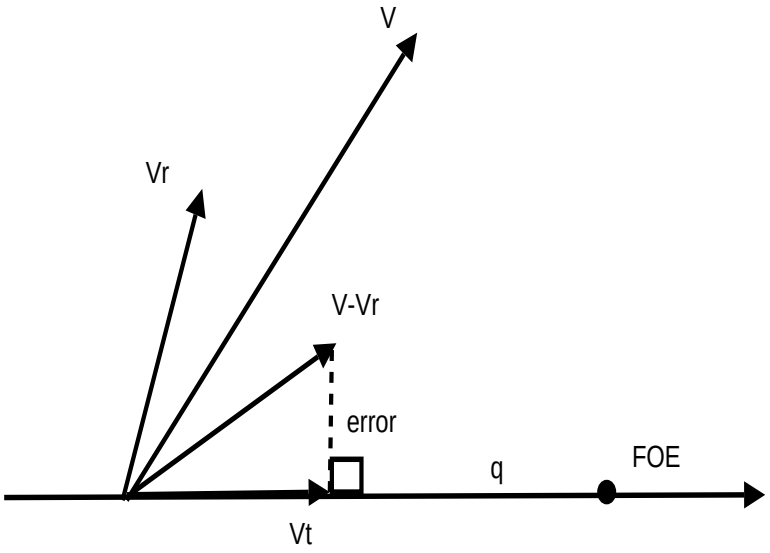

Figure 6: An optical flow vector v can be decomposed into a rotational component and a translational component v_r and v_t. v_t will always point towards the focus of expansion, so the difference between $v - v_r$ and $v - v_r$ projected onto q (the line to the FOE) defines the error for these motion parameters.

Figure 6 provides general idea of the algorithm. For any optical flow vector v there is a rotational component and a translational component, v_r and v_t. v_t will always point towards or away from the focus of expansion(FOE). If the optical flow vector were perfectly decomposed into the two components, the rotational component subtracted from the flow vector would lie on the line q which joins the point to the FOE. The distance between $v - v_r$ and the line q is the error for those motion parameters.

Adiv's algorithm works by first choosing a FOE somewhere on the image plane. The optimal rotational parameters v_r for that point are determined using E_R. The error for these parameters are then used to judge the correctness of this FOE.

First we note that since we dealing with only the direction of the translation, the search space S of translational motion is the unit sphere.

$$S = \{\mathbf{U} : |\mathbf{U}| = 1\} \tag{47}$$

where $\mathbf{U} = (U_x, U_y, U_z)$ are the normalized translational parameters from the discretized unit sphere. Each member of S defines a translational motion which can then be substituted into E_R and the

error for the optimal rotational parameters of that motion found.

$$E_U(\mathbf{U}) = E_R(\mathbf{R}^*) \tag{48}$$

where $\mathbf{R}^*$ is the optimal rotational parameters found using the least squares technique of the previous section.

What is happening is that for each translational motion in $\mathbf{U}$, we take the value of translation motion as a constant in the error equation. This allows us to assume no translational motion and use E_R for finding error of the optimal rotational parameters. The task therefore is to search S to find the value of $\mathbf{U}$ which will produce the smallest error of E_R.

Going back to Figure 6, the search space we are dealing with is the image plane. When we have chosen a translational motion, this defines the FOE. We then use E_R which will find the optimal value v_r and return the error for that choice.

This search through S is non-trivial due to its large size. Therefore, a multi-resolution scheme is used. The search is done initially at a low resolution where S is sampled coarsely. The minimum value from a high resolution gives the most promising area for further exploration. This area is sampled at a finer rate and again, the smallest region found and the process repeated. This is continued until the resolution is small enough to give a fine enough value for $\mathbf{U}$.

6.4 Segmentation

The task of segmentation is to break up an image into a series of regions which correspond uniquely to objects. Obviously, the motion of regions can be of great help in performing this task. Closely placed objects which have identical texture are a difficult problem for single image segmentation techniques. If the objects are moving separately however, the problem becomes a great deal easier. Much like single image segmentation, there are a number of different general techniques. Segmentation based on local areas is somewhat analogous to edge based segmentation. These arise from the observation that areas of discontinuity in the flow field often correspond to the boundaries between objects. There are also global techniques similar to region based segmentation which isolates whole regions in the image which have similar motion. A technique by Irani, Rousso, and Peleg [20] which falls in the latter category will be explored in more detail. A more sophisticated method proposed by Adiv [1] that allows segmentation of regions undergoing arbitrary motion will also be discussed.

6.4.1 Irani, Rousso, and Peleg

First, consider taking the simple difference between two images. The areas which correspond to stationary regions will have a low difference while regions which are moving will have a high difference. The pixels higher than some threshold are classified as moving and those under as stationary. However, moving homogeneous regions will have low differences inside the region and thus be misclassified as stationary.

To overcome the problems associated with the differencing of images, Irani, Rousso, and Peleg use several techniques. First, hierarchical methods such as a Gaussian pyramid are applied to create a pyramid of increasing resolutions. The motion detected from the lower resolutions will be passed down the pyramid, allowing homogeneous regions to inherit the motion of the object while also allowing non-homogeneous regions further refinement.

Secondly, the motions for each pixel are not simply the difference, but rather the average of the normal flow magnitudes for a small neighborhood Ω around the current pixel.

$$M = \frac{\parallel I_t \parallel \parallel \nabla I \parallel}{\sum_{\mathbf{x}\in\Omega}(\parallel \nabla I(\mathbf{x}) \parallel)^2 + C} \tag{49}$$

where I_t and ∇I are the temporal and spatial gradients respectively, and C is used to deal with numerical instabilities.

Lastly, the reliability of each motion measurement is used to determine if a motion for a particular pixel should be accepted or not. Motions for regions are passed up from lower resolutions to higher resolutions. There, the pixels in the region are checked again. If they have a high motion measure or have a low motion measure with high reliability, the new values are accepted. Otherwise, the inherited motion from the lower resolution is used.

Finally, once motion measurements for each pixel have been made, a determination (such as thresholding) can be made as to whether a pixel is either stationary(hence in the region) or moving(not in the region).

This simple technique can be used to segment an entire image with multiple moving regions. To accomplish this, one common motion is taken at a time. A simple registration of the images serves to cancel out the motion detected. The previous algorithm is used to isolate stationary pixels (which will correspond to the moving region) and moving pixels(which corresponds to the background and other regions). Segmentation is performed and the resultant region is no longer considered in following iterations for detecting the other regions.

6.4.2 Adiv

One problem with the previous algorithm is the strict constraints it makes on object motion. Since the regions are canceled by simple registration, it limits the motions to objects to just translation. A region merging technique by Adiv [1] allows segmentation to be done with general motion. An assumption that this algorithm makes is that an object can be represented by piecewise planar patches. If the patches used are small enough, this assumption is not too constraining.

A planar surface in the image can be represented by a slightly modified plane equation

$$K_x X + K_y Y + K_z Z = 1 \tag{50}$$

where K_x, K_y, K_z are the plane coefficients. We can rearrange the terms to get

$$\frac{1}{Z} = K_x x + K_y y + K_z \tag{51}$$

where (x, y) are the pixel coordinates in the image plane. Remembering our previous equations 31 and 32 for the optical flow vectors, we can substitute the equations and simplify to get

$$\alpha = a_1 + a_2 x + a_3 y \tag{52}$$

$$\beta = a_4 + a_5 x + a_6 y \tag{53}$$

$$\begin{array}{lll} a_1 = R_y + K_z T_x & a_2 = K_x T_x - K_z T_z & a_3 = -R_z + K_y T_x \\ a_4 = -R_x + K_z T_y & a_5 = R_z + K_x T_y & a_6 = K_y T_y - K_z T_z \end{array}$$

where K_x, K_y, K_z are the plane coefficients, $\mathbf{T} = (T_x, T_y, T_z)$ are the translational parameters, and $\mathbf{R} = (R_x, R_y, R_z)$ are the rotational parameters. Note that our simplification involved reducing the motion transforms from a general 8 term motion equation to this reduced 6 terms. These six terms define an affine transformation. Adiv uses the reduced affine transforms in initial stages but later moves back to the general 8 term motion. This increased complexity is not necessary however to grasp the key concepts of his approach and we will limit ourselves to affine motions for this discussion.

Region merging algorithms in general have three essential components

1. An initial segmentation is performed which creates small homogeneous regions.
2. A region merging criteria is selected to permit allowable mergings.
3. Regions are merged according to the criteria until no more mergings can be done. When two regions are merged, re-evaluate the criteria for the newly created region.

To perform the initial segmentation, Adiv uses an interesting generalized Hough technique where the dimensions of parameter space correspond the coefficients $a_1...a_6$. Remember that a single optical flow vector can be part of many possible larger motions of an object. Therefore, using the Hough technique, each flow vector votes for all those motions(represented by our 6 parameters) which it is consistent with.

So, for a flow vector (α, β) we check point $a_1...a_6$ in parameter space by

$$\sqrt{\delta_x^2 + \delta_y^2} \leq \epsilon_1 \tag{54}$$

$$\delta_x = \alpha - (a_1 + a_2 x + a_3 y)$$
$$\delta_y = \beta - (a_4 + a_5 x + a_6 y)$$

If the deviation of the measured optical flow vector (α, β) from the correct optical flow vector for that motion is less than an error threshold ϵ_1, this vector will vote for that point.

Our initial segmentation thus works as follows. The vectors are first processed so that all the compatible motions for each vector are put into the parameter space. Then, we search for maxima in the parameter space. These will define probable motions since they have significant support from many flow vectors. When a maxima is found, and it above a certain threshold, we must then decide which flow vectors are consistent with this motion. Therefore, we use the previous error equation to determine the deviation of each flow vector from a particular motion. If the error is below a threshold ϵ_2, then this vector is classified as being part of that moving region. In this way, all the maxima of the parameter space are analyzed and the vectors assigned to their corresponding motions. Also, if a vector has been classified by a particular motion and another maxima results in a lower error, then the vector will be assigned this new error. The end result is that our flow field is divided into small regions of compatible motion. This is a simplification of Adiv's algorithm since regions can be non-contiguous whereas in Adiv's regions must be connected.

Further simplifications exist. Firstly, a 6D parameters space is too vast to be computationally feasible so it is divided into two 3D spaces which correspond to the two parts of equations 52 and 53. In practice this works reasonably well since maxima in the 6D space often correspond to maxima in the 3D spaces. A multi-resolution technique is also used so that the parameter space is initially coarsely discretized and the maximal areas found. These areas are then further explored at a finer resolution.

The next stage of the region merging process is to define a merging criteria. Note that for each initial region, we have an optimal motion described by the six parameters and a set of optical flow vectors consistent with that motion. When combining two sets of vectors, the task is thus to determine the optimal motion for the newly created set as well as the error which results. Therefore, as could easily be guessed from previous sections, a least squares approach will be used to minimize the error function over n, the vectors from both regions combined.

$$E(a_1...a_6) = \sum_{i=1}^{n} (\alpha_i - (a_1 + a_2 x_i + a_3 y_i))^2 + (\beta_i - (a_4 + a_5 x_i + a_6 y_i))^2 \tag{55}$$

This is done with similar techniques to those described earlier using partial derivatives and systems of linear equations. If this error from the two regions is less than a threshold, the two regions will be merged into one permanently.

The third stage of the region merging process is fairly easy. The largest regions is considered first and the region merging criteria is applied to all the remaining regions to determine possible merges. If the largest region cannot merge with any other region, the next largest is considered with respect to the remaining regions. This continues until either a merging takes place and the process starts over, or there was no merging so the regions are final.

Finally, all the unassigned flow vectors which were missed during the initial segmentation are considered and added to compatible regions.

6.5 Traffic Analysis and Vehicle Tracking

Optical flow is among the lowest-level operations for motion analysis. This has influenced the choices of applications for this paper. Many have been correspondingly low level: segmentation, motion parameters and time-to-contact. These are building blocks to the more complex systems that we often associate with applications for video sequences such as autonomous vehicle navigation or facial analysis. High level applications for optical flow are limited only by the ingenuity of designers. While the narrow focus of this paper on optical flow does not permit a survey of these high level applications, it would be unfortunate to not mention them, even in a slight way. This section tries to provide a bridge from the low level applications we've seen to a higher level subsystem for real-time traffic analysis and vehicle tracking. ASSET-2, developed by Smith and Brady [25], is a system for detecting and tracking moving objects, an obviously necessary subsystem for autonomous navigation. Their system is used for traffic analysis: tracking vehicles from a camera that is either a stationary or attached to a moving vehicle itself. The system works in three general stages: feature-based optical flow estimation, flow segmentation into clusters, and cluster tracking and filtering. It also permits occlusion of vehicles to occur.

The first stage is the calculation of the optical flow field for an individual frame. A feature-based method is used to detect the flow vectors using the SUSAN feature detector. Detected features are matched using feature tracking, a significant field in it's own right not discussed in this paper because of time constraints. Briefly defined, feature tracking estimates motion models for each feature and uses these to track features over many frames. Models are also updated continuously. After this stage, the feature tracker returns the flow field of consistent flow vectors, either the instantaneous displacement for the feature or the displacement over the past N frames. Providing the displacement over several frames means the features are less likely to be confused with noise or spurious objects. This field is then fed as input to the flow segmenter.

The next stage is to segment the flow field into separate clusters. This is done using a simple least squares fit of an affine motion model for each cluster. A new cluster is started for a region in the image using some flow vector to instantiate a new affine motion. The error from adding neighbors of that flow vector to the current affine model is determined using

$$E = \frac{|\mathbf{v} - \mathbf{v_a}|}{\frac{|\mathbf{v}|+|\mathbf{v_a}|}{2} + \omega} \tag{56}$$

where $\mathbf{v}$ is the current flow vector,$\mathbf{v_a}$ is the vector the affine model estimates for that position and ω is the error of the current estimate.

If the error is less than some threshold, this vector will be added to this cluster and the affine model updated. These clusters are grown from the neighbors and the flow vectors removed from the available list when they have been assigned to a cluster. When no more clusters can be created, we then determine the bounding box, centroid and model motion for each large enough cluster. These clusters are then passed to the tracker.

The cluster tracker and filter takes the clusters from the current frame and matches them with a list of previous clusters. Matching is done using the motion model, the shape of the bounding box and the centroid of the clusters. If the error in matching is less than some threshold, we assign the cluster to a previous one and update the information. Otherwise, the cluster is added to the list. Finally, a filtered list of clusters is passed on to a higher level for interpretation.

ASSET-2 had been implemented and tested in real-time in real situations at full frame rate. It consistently tracks the movement of other vehicles and tracks them, even from a moving vehicle.

7 Conclusion

The determination of optical flow forms a vital low level stage for motion analysis in computer vision. However, as we have seen, optical flow fields are not trivial to determine. The aperture problem fundamentally limits reliability in many areas by restricting the calculated optical flow to

the component along the image gradient. Differential, correlation or feature based methods all try to solve this problem and find the entire flow vector. Optical flow can also be used for a variety of low level tasks like time-to-contact, motion parameter, depth or segmentation. These tasks are often used as input to many higher level tasks like autonomous navigation, object tracking, and image stabilization.

References

[1] G. Adiv, *Determining three-dimensional motion and structure from optical flow generated by several moving objects*, IEEE Transactions on Pattern Analysis and Machine Intelligence **7** (1985), no. 4, 384–401.

[2] P. Anandan, *A computational framework and an algorithm for the measurement of visual motion*, International Journal of Computer Vision **2** (1989), no. 3, 283–310.

[3] S.T. Barnard and W.B. Thompson, *Disparity analysis of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **2** (1980), 333–340.

[4] S.S. Beauchemin and J.L. Barron, *The computation of optical flow*, ACM Computing Surveys **27** (1996), no. 3, 433–467.

[5] P.R. Beaudet, *Rotational invariant image operators*, Proc. of the Int. Conf. on Pattern Recognition (1978), 579–583.

[6] A.R. Bruss and B.K.P. Horn, *Passive navigation*, Computer Vision, Graphics, and Image Processing **21** (1983), no. 1, 3–20.

[7] W. Burger and B. Bhanu, *Estimating 3D egomotion from perspective image sequence*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990), no. 11, 1040–1058.

[8] T. Camus, *Real-time optical flow*, Ph.D. thesis, Brown University, Department of Computer Science, 1994.

[9] L. Dreschler and H.H. Nagel, *Volumetric model and 3D trajectory of a moving car derived from monocular tv frame sequences of a street scene*, Computer Vision, Graphics and Image Processing **20** (1981), no. 3, 199–228.

[10] Hildreth E.C., *Computations underlying the measurement of visual motion*, Artificial Intelligence **23** (1984), no. 3, 309–354.

[11] S. Gong and M. Brady, *Parallel computation of optical flow*, Proc. First European Conf. on Computer Vision (1990), 124–133.

[12] Chris Harris and Mike Stephens, *A combined corner and edge detector*, Proc. 4th Alvey Vision Conference (1988), 147–151.

[13] N.G. Hatsopoulos and W.H. Warren Jr, *Visual navigation with a neural network*, Neural Networks **4** (1991), 303–317.

[14] J. Little H.H. Blthoff and T. Poggio, *A parallel algorithm for real time computation of optical flow*, Nature **337** (1989), 549–553.

[15] B.K.P. Horn and B.G. Schunk, *Determining optical flow*, Artificial Intelligence **17** (1981), 185–203.

[16] J.-Q.Fang and T.S. Huang, *Solving three-dimensional small-rotation motion equations*, Proc. IEEE Conf. Computer Vision and Pattern Recognition (1983), 253–258.

[17] D.J. Fleet J.L. Barron and S. Beauchemin, *Performance of optical flow techniques*, International Journal of Computer Vision **12** (1994), 43–77.

[18] D.T. Lawton, *Processing translational motion sequences*, Computer Vision, Graphics and Image Processing **22** (1983), 116–144.

[19] B.D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*, Proc. 7th International Joint Conference on Artificial Intelligence (1981), 674–679, Vancouver (CA).

[20] B. Rousso M. Irani and S. Peleg, *Computing occluding and transparent motions*, International Journal of Computer Vision **12** (1994), 5–16.

[21] E. Grosso M. Tistarelli and G. Sandini, *Dynamic stereo in visual navigation*, Proceedings of the IEEE CVPR (1991), 186–193.

[22] H.P. Moravec, *Towards automatic visual obstacle avoidance*, Proc. 5th Int. Joint Conf. Artificial Intell. (1977), 584.

[23] M. Ogata and T. Sato, *Motion-detection model with two stages: Spatiotemporal filtering and feature matching*, Journal of the Optical Society of America **9** (1992), no. 3, 377–387.

[24] J.H. Rieger and D.T.Lawton, *Processing differential image motion*, Journal of the Optical Society of America A **2** (1985), 354–360.

[25] S.M. Smith and J.M. Brady, *Asset-2: Real-time motion segmentation and shape tracking*, IEEE Trans. on Pattern Analysis and Machine Intelligence **17** (1995), no. 8, 814–820.

[26] _____, *Susan-a new approach to low level image processing*, International Journal of Computer Vision **23** (1997), no. 1, 45–78.