Lucas-Kanade in a Nutshell

Prof. Dr. Raúl Rojas*

1 Motivation

The Lucas-Kanade optical flow algorithm is a simple technique which can provide an estimate of the movement of interesting features in successive images of a scene. We would like to associate a movement vector (u, v) to every such "interesting" pixel in the scene, obtained by comparing the two consecutive images.

The Lucas-Kanade algorithm makes some implicit assumptions:

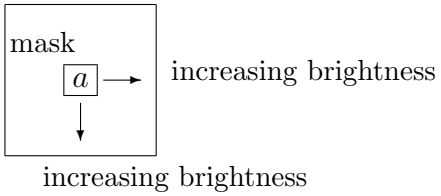
- The two images are separated by a small time increment Δt, in such a way that objects have not displaced significantly (that is, the algorithm works best with slow moving objects).
- The images depict a natural scene containing textured objects exhibiting shades of gray (different intensity levels) which change smoothly.

The algorithm does not use color information in an explicit way. It does not scan the second image looking for a match for a given pixel. It works by trying to guess in which direction an object has moved so that local changes in intensity can be explained.

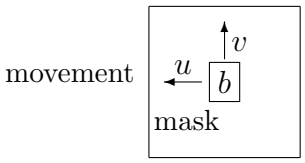
*Freie Universität Berlin, Dept. of Computer Science, Arnimallee 7, 14195 Berlin, Germany

2 Technique

Assume that we watch a scene through a square hole. The intensity a visible through the hole is variable.



In the next frame the intensity of the pixel has increased to b. It would be sensible to assume that a displacement of the underlying object to the left and up has occurred so that the new intensity b is now visible under the square hole.



If we know that the increase in brightness per pixel at pixel (x, y) is $I_x(x, y)$ is the x-direction, and the increase in brightness per pixel in the y direction is $I_y(x, y)$, we have a total increase in brightness, after a movement by u pixels in the x direction and v pixels in the y direction of:

$$I_x(x, y) \cdot u + I_y(x, y) \cdot v$$

This matches the local difference in intensity $(b - a)$ which we call $I_t(x, y)$, so that

$$I_x(x, y) \cdot u + I_y(x, y) \cdot v = -I_t(x, y)$$

The negative sign is necessary because for positive I_x, I_y, and I_t we have a movement to the left and down (think about this for a minute).

3 Neighborhoods

Of course, a simple pixel does not usually contain enough "structure" useful for matching with another pixel. It is better to use a neighborhood of pixels, for example the 3×3 neighborhood around the pixel (x, y). In that case we set 9 linear equations:

$$I_x(x+\Delta x,\, y+\Delta y)\cdot u + I_y(x+\Delta x,\, y+\Delta y)\cdot v = -I_t(x+\Delta x,\, y+\Delta y)$$
for $\Delta x = -1, 0, 1$ and $\Delta y = -1, 0, 1$

The linear equations can be summarized as the matrix equality

$$S\begin{pmatrix} u \\ v \end{pmatrix} = \vec{t}$$

where S is a 9×2 matrix containing the rows $\Big(I_x(x+\Delta x,\, y+\Delta y),\ I_y(x+\Delta x,\, y+\Delta y)\Big)$ and $\vec{t}$ is a vector containing the 9 terms $-I_t(x+\Delta x,\, y+\Delta y)$.

The above equation cannot be solved exactly (in the general case). The Least Squares solution is found by multiplying the equation by S^T

$$S^T S\begin{pmatrix} u \\ v \end{pmatrix} = S^T\ \vec{t}$$

and inverting $S^T S$, so that

$$\begin{pmatrix} u \\ v \end{pmatrix} = (S^T S)^{-1} S^T\ \vec{t}$$

4 Invertibility

The solution given above is the best possible, whenever $S^T S$ is invertible. This might not be the case, if the pixel (x, y) is located in a region with no structure (for example, if I_x, I_y and I_t are all zero for all pixels in the

neighborhood). Even if the matrix is invertible it can be ill conditioned, if its elements are very small and close to zero.

One way of testing how good the inverse of $S^T S$ for our purposes is, is to look at the eigenvalues of this matrix. $S^T S$ is a symmetrical matrix, and as such can be diagonalized and written in the form

$$S^T S = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^T$$

where U is a unitary 2×2 matrix. If $S^T S$ is not invertible, λ_1 or λ_2, or both, are zero. If one of them, or both, are very small, then the inverse matrix is ill-conditioned.

Testing the size of the eigenvectors can be done by solving the characteristic equation

$$\det(S^T S - \lambda I) = 0$$

which reduces to

$$\det \begin{bmatrix} \sum_N I_x^2(x+\Delta x, y+\Delta y) - \lambda & \sum_N I_x(x+\Delta x, y+\Delta y) I_y(x+\Delta x, y+\Delta y) \\ \sum_N I_x(x+\Delta x, y+\Delta y) \cdot I_y(x+\Delta x, y+\Delta y) & \sum_N I_y^2(x+\Delta x, \ell+\Delta y) - \lambda \end{bmatrix} = 0$$

This is a quadratic equation for λ, which can be readily solved. The sums are computed over the neighborhood of pixels N.

5 Conclusions

The Lucas-Kanade algorithm makes a "best guess" of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel we have two unknowns (u and v) and one equation (that is, the system is underdetermined). We need a neighborhood in order to get more equations. Doing so makes the system overdetermined and we have to find a least squares solution. The LSQ solution averages the optical flow guesses over a neighborhood.

We assume that all intensity changes can be explained by intensity gradients. The method breaks down when the gradients are random (think of an image of random points) or when the gradients are negligible (no structure, as in flat surfaces). The Lucas-Kanade algorithm eliminates regions without structure by looking at the invertibilily of the matrix $S^T S$ in an indirect way, that is, through the eigenvalues of this matrix.

The result of the algorithm is a set of optical flow vectors distributed over the image which give an estimation idea of the movement of objects in the scene. Of course, some optical flow vectors will be erroneous.

The main advantage of the algorithm is, that for a neighborhood of fixed size, the number of operations needed to compute $(S^T S)^{-1} S^T \ \vec{t}$ are constant, and therefore the complexity of the algorithm is linear in the number of pixels examined in the image. Alternative algorithms that match similar regions using a neighborhood, and scanning the second image, have quadratic complexity.

Summarizing: The Lucas-Kanade algorithm is an efficient method for obtaining optical flow information at interesting points in an image (i.e. those exhibiting enough intensity gradient information). It works for moderate object speeds.

References:

B.D. Lucas, T. Kanade, "An Image Registration Technique with an Application to Stereo Vision", in Proceedings of Image Understanding Workshop, 1981, pp. 121-130.