Image registration

Suppose we have two images $I(x,y)$ and $J(x,y)$ that are almost the same. We may have two neighboring image frames in a video, or we may have two images taken by two cameras (stereo) that have the same internal parameters and nearly the same external parameters, i.e. they are close to each other in space and have similar orientations. We would like to find correspondences between points in the two images. This is sometimes known as "image registration". Today we will look at a classical method for solving this problem, known as the Lucas-Kanade method.[1]

1D image registration

Let's first go to the 1D problem to make sure we understand the basic idea, and work with images $I(x)$ and $J(x)$. For any fixed x_0 in image J, we would like to find a corresponding point $x_0 + h$ in image I such that $I(x+h) \approx J(x)$ in a neighborhood of x_0. To find h, we could try to minimize the following:

$$\sum_{x \in Ngd(x_0)} \{I(x+h) - J(x)\}^2$$

We use a summation here because we don't just want to find a matching point. We want a small neighborhood of matching points.

We could minimize the above by just doing a brute force search over h. Note that this would force us to choose particular h values e.g. integer pixel increments. Instead, one typically assumes $I(x)$ is smooth (because we have blurred it with a Gaussian beforehand), takes a first order Taylor series approximation:

$$I(x+h) \approx I(x) + h\frac{dI}{dx}(x).$$

and then minimizes the following:

$$\sum_{x \in Ngd(x_0)} (I(x) - J(x) + h\frac{dI(x)}{dx})^2.$$

The only variable in the above expression is h. The rest are data values. In particular, the above expression is a quadratic $ah^2 + bh + c$. We would like to find the value of h that minimizes this expression. That is easy. We just take the derivative with respect to h and set it to 0,

$$\sum_{x \in Ngd(x_0)} (I(x) - J(x) + h\frac{dI(x)}{dx})\frac{dI(x)}{dx} = 0$$

and so

$$h = \frac{\sum_{x \in Ngd(x_0)} (J(x) - I(x))\frac{dI(x)}{dx}}{\sum_{x \in Ngd(x_0)} (\frac{dI(x)}{dx})^2}.$$

Note that we are not restricted to integer values of h here.

What assumptions have we made?

[1]B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", IJCAI 1981.

- We use a first order approximation of $I(x+h)$ around x, namely $I(x)$ should be linear over a distance h;
- The value h should be valid over the whole neighborhood $x \in Ngd(x_0)$.

To be confident that the first assumption holds, one typically smooths the image with a Gaussian to reduce noise and to smooth out the edges. Without smoothing, obviousy $I(x)$ will not be linear in the neighborhood of an edge!

To be confident that the second assumption holds is another story. At the end of this lecture, we will look at the case in which this assumption doesn't necessarily hold, and we allow for a more general transformation between the domains of $I()$ and $J()$.

Iterative method

The above method assumes that the function $I(x)$ is linear in the neighborhood of x_0. While this approximation may be a good enough to get an estimate of h, we do not expect it to give us h exactly since $I(x)$ will generally have second and higher order terms as well.

Suppose we have estimated h and we wish to refine our estimate. Or after k iterations of estimating h, we have an h_k and we want an estimate h_{k+1}. How do we do it? We want to find an h such that $I(x+h_k+h) = J(x)$ near $x = x_0$. We proceed exactly as before, but now $x + h_k$ replaces x as the parameter for $I()$. So,

$$I(x+h_k+h) \approx I(x+h_k) + h\frac{dI(x+h_k)}{dx}$$

and now we want to minimize

$$\sum_{x \in Ngd(x_0)} (I(x+h_k) - J(x) + h\frac{dI(x+h_k)}{dx}\ )^2$$

and so

$$h = \frac{\sum_{x\in Ngd(x_0)} (J(x) - I(x+h_k))\frac{dI(x+h_k)}{dx}}{\sum_{x\in Ngd(x_0)} (\frac{dI(x+h_k)}{dx}\ )^2}.$$

Notice here that if h_k is not an integer i.e. inter-pixel distance, then we need to interpolate to say what $I(x+h_k)$ and $\frac{dI(x+h_k)}{dx}$) are.

After solving for h at the kth step, we update:

$$h_{k+1} \leftarrow h_k + h\ .$$

We can repeat this until h is small, in which case the estimate has converged. (Or if it doesn't converge, then we give up.)

2D Image registration

Let's next consider 2D images $I(x,y)$ and $J(x,y)$. We would like to know what shift (h_x, h_y) minimizes

$$\sum_{(x,y)\in Ngd(x_0,y_0)} \{I(x+h_x, y+h_y) - J(x,y)\}^2$$

As before, we could use brute force but instead we use a Taylor series expansion. Assuming that the images have been blurred with a 2D Gaussian so that they are smooth,

$$I(x + h_x, y + h_y) \ \approx \ I(x,\ y) + \frac{\partial I}{\partial x} h_x + \frac{\partial I}{\partial y} h_y$$

where the partial derivatives are evaluated at (x, y). We then minimize

$$\sum_{(x,y) \in Ngd(x_0,y_0)} (I(x,y) - J(x,y) + \frac{\partial I}{\partial x} h_x + \frac{\partial I}{\partial y} h_y)^2$$

This is a quadratic expression in two variables and it has a single minimum. To see it is a minimum, not that if we fix either h_x or h_y and let that other go to infinity, then the expression goes to positive infinity.

Notice for each equation

$$I(x,y) - J(x,y) + \frac{\partial I}{\partial x} h_x + \frac{\partial I}{\partial y} h_y = 0.$$

we now have two variables h_x and h_y. Unlike in the 1D case, we now cannot solve this at a single point. Moreover, in neighborhoods so small that the gradient is constant (so each equation is the same), we do not have enough information to solve for (h_x, h_y). This problem is known classically as the *aperture problem.*

There is a subtle point here, so let's go slow. We are assuming the image is smooth in order to be able to take derivatives. But if the image is smooth then the image gradient is smooth as well, which means that the gradient hardly varies in a small neighborhood of (x_0, y_0). To avoid the aperture problem, we thus need to use neighborhoods that are larger than the neighborhood over which the gradient is roughly constant. (This is reminiscent of the corner problem we discussed last lecture. Now, if the neighborhood of a point is so small that the intensity gradient is constant, then a point and its neighbor in the direction perpendicular to the intensity gradient will each have the same local intensity structure.)

Returning to our problem, we take the derivative of the above summation with respect to h_x and h_y and set each to zero. This gives

$$\begin{bmatrix} \sum(\frac{\partial I}{\partial x})^2 & \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & \sum(\frac{\partial I}{\partial y})^2 \end{bmatrix} \begin{bmatrix} h_x \\ h_y \end{bmatrix} = - \begin{bmatrix} \sum(I(x,y) - J(x,y))\frac{\partial I}{\partial x} \\ \sum(I(x,y) - J(x,y))\frac{\partial I}{\partial y} \end{bmatrix}$$

You recognize the matrix on the left. It is the second moment matrix $\mathbf{M}$ we saw in the corner detection method from last lecture.

We can solve for (h_x, h_y) as long as the second moment matrix is invertible, or equivalently, as long as the two eigenvalues are non-zero. Let's look more closely at this condition.

Consider any unit vector $(h_x, h_y) = (\cos\theta, \sin\theta)$. Notice that

$$(\cos\theta, \sin\theta)\ \mathbf{M}\ (\cos\theta, \sin\theta)^T = \sum (\cos\theta, \sin\theta) \ \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

which is the sum of the square of the directional derivatives in direction θ, over the neighborhood of (x_0, y_0).

Since we are talking about a sum of squares, we are obviously talking about a sum of non-negative numbers. But now note from the left side that when $(\cos\theta, \sin\theta)^T$ is an eigenvector of $\mathbf{M}$, the above sum of squares must be the eigenvalue. Thus, we see easily that the eigenvalues of $\mathbf{M}$ are both non-negative.

When do we get a zero eigenvalue? The only way we can have a zero eigenvalue is if the directional derivatives at *all points* in the neighborhood vanish. That can happen in two ways. One way is that the direction of $\nabla I(x,y)$ is constant in the neighborhood (and so the directional derivative in the perpendicular direction would be zero). The other is that $\nabla I(x,y)$ is zero in the entire neighborhood i.e. the image intensity is constant. In the latter case, both of the eigenvalues are zero!

It is no accident that the second moment matrix arises in today's problem and in the corner detection problem last lecture. In corner detection, we are looking for points (x,y) which are locally distinctive. This meant that if we were to compare intensities in the neighborhood of (x_0, y_0) with intensities in the neighborhood of a nearby point $(x_0 + h_x, y_0 + h_y)$ then we would like these two local intensity patterns to be different. Our condition for this being true was that the eigenvalues of $\mathbf{M}$ were non-zero and we now see what this means in terms of the intensity gradients (namely that they cannot all be zero, and they cannot all be in the same direction).

The same idea holds when we are trying to match patches *between images* I and J. If the gradients of I are all in the same direction, then we could slide I locally in the perpendicular direction and the local intensities wouldn't change. But this just means that we cannot decide uniquely where to slide I so it matches J.

A few more points to mention: First, another way of writing the second moment matrix is in terms of an $n \times 2$ matrix $\mathbf{A}$ whose rows are $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ for the different $(x,y) \in Ngd(x_0, y_0)$.

$$\mathbf{M} = \mathbf{A}^T\mathbf{A} = \left[\begin{array}{cc} \sum(\frac{\partial I}{\partial x})^2 & \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & \sum(\frac{\partial I}{\partial y})^2 \end{array} \right]$$

This can clean up the notation a bit. For example, if we write the $I(x,y)$ and $J(x,y)$ values as vectors $\vec{I}$ and $\vec{J}$ then see that we want to solve:

$$\mathbf{A}^T\mathbf{A}(h_x, h_y)^T = \mathbf{A}^T(\vec{J} - \vec{I}).$$

Iterative method

Just like in the 1D case, we have taken a Taylor series expansion of $I(x,y)$ near (x_0, y_0) and this means we cannot expect a perfect estimate of (h_x, h_y). To improve accuracy, we can try to iterate. Suppose we have the k^{th} estimate (h_x^k, h_y^k) and we wish to estimate (h_x^{k+1}, h_y^{k+1}). We define a new function $I^k(x,y) = I(x + h_x^k, y + h_x^k)$ and we solve for (h_x, h_y). Then, we update our estimate

$$h_x^{k+1} \leftarrow h_x^k + h_x$$

$$h_y^{k+1} \leftarrow h_y^k + h_y.$$

We repeat until we converge (or if we don't converge then we give up).

Beyond 2D translation

The above method allows for local translations only, but images can differ from each other in more general ways. A more general model is to allow local scaling, shearing, and rotation. Let's suppose we have two image patches such that, in a neighborhood of $\mathbf{x_0} = (x_0, y_0)$, we have (to first order)

$$I(\mathbf{x_0} + (\mathbf{I} + \mathbf{D})(\mathbf{x} - \mathbf{x_0})) = I(\mathbf{x} + \mathbf{D}(\mathbf{x} - \mathbf{x_0}) + \mathbf{h}) = J(\mathbf{x})$$

where $\mathbf{h} = (h_x h_y)$ is a local translation as above, and $\mathbf{D}$ is a general 2×2 matrix that allows for "deformations" which include shear, rotation, and scaling,

$$\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}.$$

We would like to estimate a constant matrix $\mathbf{D}$ and vector $\mathbf{h}$ such that

$$\sum_{\mathbf{x} \in Ngd(\mathbf{x_0})} (I(\mathbf{x} + \mathbf{D}(\mathbf{x} - \mathbf{x_0}) + \mathbf{h}) - J(\mathbf{x}))^2$$

is as small as possible.

As above, we take a local Taylor series expansion of $I()$ around each $\mathbf{x} = (x, y)$. Letting

$$\Delta\mathbf{x} = \mathbf{x} - \mathbf{x_0}$$

we get

$$I(\mathbf{x} + \mathbf{D}\Delta\mathbf{x} + \mathbf{h}) \approx I(\mathbf{x}) + \frac{\partial I}{\partial x}(D_{11}\Delta x + D_{12}\Delta y + h_x) + \frac{\partial I}{\partial y}(D_{21}\Delta x + D_{22}\Delta y + h_y)$$

Notice that the variables here that we wish to solve for are the four D_* and the two h_*.

Substituting into the sum of squares above, we get an order 2 polynomial in a 6D space, which goes to ∞ as any of the six variables go to infinity. Let

$$\mathbf{d} = (D_{11}, D_{12}, D_{21}, D_{22}, h_x, h_y)$$

and let

$$\mathcal{I} = (\frac{\partial I}{\partial x}\Delta x, \frac{\partial I}{\partial x}\Delta y, \frac{\partial I}{\partial y}\Delta x, \frac{\partial I}{\partial y}\Delta y, \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$$

so we want to minimize

$$\sum (I(x,y) - J(x,y) + \mathcal{I} \cdot \mathbf{d})^2$$

Taking the partial derivatives with respect to the D_* and h_* variables gives six equations:

$$\sum (I(x,y) - J(x,y) + \mathbf{d} \cdot \mathcal{I})\mathcal{I} = \mathbf{0}$$

or

$$(\sum_{(x,y) \in Ngd(x_0,y_0)} \mathcal{I}\mathcal{I}^T)\ \mathbf{d} = \sum_{(x,y) \in Ngd(x_0,y_0)} (J(x,y) - I(x,y))\ \mathcal{I}$$

As we saw earlier in the lecture, one could solve iteratively for the six parameters $\mathbf{d}$.

When can we solve for the $\mathbf{d}$ vector? The 6×6 matrix would need to be invertible, and there is no reason why this is always going to be the case. For example, if $\frac{\partial I}{\partial y} = 0$ for all points in the neighborhood, then we are not going to be able to solve for the h_y variable. Here I am appealing to your intuition of the problem we saw back on page 4.

A more interesting example would be if the image was rotationally symmetric locally around (x_0, y_0), for example, a "bull's eye" pattern. In this case, the translation component of the transformation could be recovered, but the rotational component of $\mathbf{D}$ could not. i.e. You would be able to rotate the intensity pattern without changing the sum of squared errors. In this case, we would find that there is a zero eigenvalue of the 6×6 matrix.

I mentioning this 6D problem for a few reasons. First, it is important to appreciate that a simple translation model is not going to be sufficient for all situations. Second, I want to get you to start thinking about higher dimensional linear problems, and what the issues are when we try to solve them. We will see many examples later in the course.