Egomotion

We now return to a problem discussed in lecture 2, and show how to solve it. When an observer moves through a rigid 3D scene, the changing image projection defines a vector field of image velocities. If there are N pixels in the image, then the velocity field has $N + 6$ degrees of freedom, namely the Z values of the points, the three translation parameters T_X, T_Y, T_Z and the three rotation parameters $\Omega_X, \Omega_Y, \Omega_Z$.

The problem we examine today is this: Given two images – namely adjacent frames in a video – estimate the velocity field (v_x, v_y) and use the velocities to estimate the camera rotation and translation and the scene Z values. This is a fundamental problem that our own vision systems are solving all the time as we move, and it is a problem that robotic vision systems now can solve too.

We assume today that the camera is calibrated, so that the matrix $\mathbf{K}$ is known. This allows us to express the problem in terms of image projection plane coordinates, as was done in lecture 2. We assume that we can apply pixel positions to image projection plane positions using $\mathbf{K}^{-1}$ and we won't discuss the calibration matrix $\mathbf{K}$ any further today.

Motion field (revisited – see lecture 2)

From lecture 2, the image velocity vector $\vec{v}$ at (x, y) is the sum of two component vectors,

$$\vec{v} = \vec{v}_T + \vec{v}_\Omega \tag{1}$$

which are due to the observer's translation $\mathbf{T}$ and rotation $\mathbf{\Omega}$, respectively. The translation component is the sum of three vector fields, which are due to components in the X, Y, Z directions, and can be written

$$\vec{v}_T = \frac{1}{Z(x,y)} \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}. \tag{2}$$

The special image position

$$(x_T, y_T) \;=\; \frac{f}{T_z}(T_x, T_y) \tag{3}$$

is called the *direction of heading*. In the special case of lateral motion, namely where $T_z = 0$, the direction of heading is at infinity in the image plane. A few observations:

- There is a depth–speed ambiguity here between the depth map $Z(x, y)$ and the translation vector $\mathbf{T}$. Multiplying the function $Z(x, y)$ by a constant and the speed $\mathbf{T}$ by that same constant does not change the image velocity field. For this reason, the vision system can (at best) estimate these variables up to this unknown scale factor. This reduces the number of degrees of freedom from $N + 6$ to $N + 5$. For example, if the translation vector is non-zero, then you could just assume it is of unit length.

- The direction of heading is unaffected by scaling $\mathbf{T}$, since the direction of heading is defined by ratios of components of $\mathbf{T}$, so the unknown scale factor cancels.

- $Z(x, y)$ is a continuous function if and only if the translation component of the velocity field is continuous. In particular, if there is a depth discontinuity – namely at a boundary between two objects that are at different depths – then there will be a discontinuity in this component of the velocity field as well.

Verify that the rotation component is the sum of the three canonical rotations discussed in lecture 2:

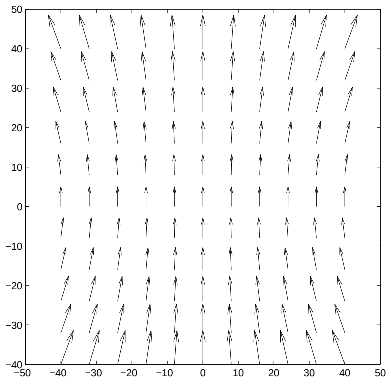
$$\vec{v}_\Omega = \begin{bmatrix} \frac{xy}{f} & f(1+\frac{x^2}{f^2}) & -y \\ \\ f(1+\frac{y^2}{f^2}) & \frac{xy}{f} & x \end{bmatrix} \begin{bmatrix} \Omega_X \\ \Omega_Y \\ \Omega_Z \end{bmatrix} \tag{4}$$

It is not obvious that you can just add rotation fields together (especially if you are concerned that rotation matrices typically do not commute). The reason it works here is that we are assuming very small rotations between frames. Multiplication of two infinitesimal rotations *does* commute. (Proving this requires some work, which I skip here.)

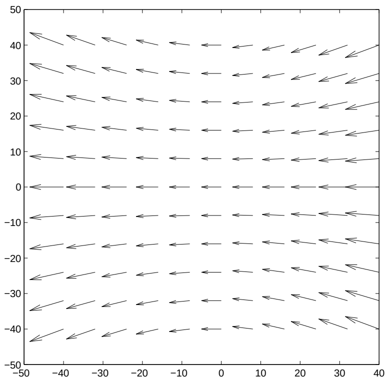
Two other observations:

- If the field of view is small, then the Ω_y component of the rotation field is very similar to a T_x component of a translation field with constant depth, and similarly the Ω_x component of the rotation field is very similar to a T_y component of a translation field with constant depth. This suggestions that if the field of view is small then it will be very difficult to solve for camera motion (since are unlying ambiguities).[1]

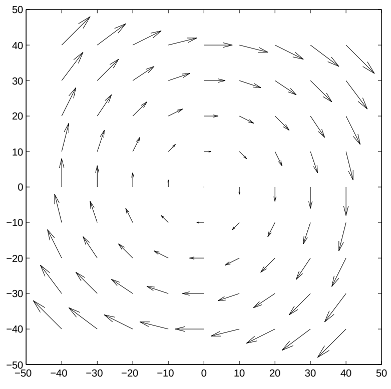
- The rotation component is smooth. Always. It does not depend on depth. This is closely related to what we saw with case 3 of the homography lecture (19).



$\mathbf{\Omega} = (1,0,0)$
pitch



$\mathbf{\Omega} = (0,1,0)$
pan



$\mathbf{\Omega} = (0,0,1)$
roll

Review of image motion estimation

To estimate the velocity field, one can use the image registration method discussed in lecture 10. There we examined how to register an image $J(x,y)$ with $I(x,y)$ near (x_0,y_0), namely find the (h_x,h_y) vector that minimizes

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

[1]G. Adiv, "Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field", PAMI 1989.

Here the images I and J are two neighboring frames in a video. Recall that we could solve this problem in two ways. One was brute force – just examine various (h_x, h_y) vectors, compute the sum of squared differences for each, and then pick the (h_x, h_y) that minimizes the difference. Another was to estimate it by linearizing the above expression and using least squares. I also discussed a scale space version in lecture 12.

The (h_x, h_y) vectors are our velocity vectors, and we can say something about how reliable are our estimates of them. There are basically two situations in which the estimates are *not* reliable: first, if the minimum of the sum of squared differences is a large value, then we might not be so interested in this solution since the model fails to explain the transformation from I to J; second, if the sum of squared differences is a minimum but there are many solutions (h_x, h_y) that give roughly the same minimum (the minimum might be flat in both directions, or it might be flat in one direction e.g. if one of eigenvalues of the second moment matrix is small).

The review is now over, and we are ready to apply these ideas to solve our problem.

Estimating camera motion

If the moving observer knew $\mathbf{\Omega}$ and $\mathbf{T}$ and if the observer could estimate the image velocity (v_x, v_y) at a set of points, then the observer could compute the depths Z at these points. This is because $\mathbf{\Omega}$ and $\mathbf{T}$ determine the image velocity at each image point up to an unknown variable $\frac{1}{Z}$, and so if you knew the image image velocity vector then you could trivially compute this variable.

Estimating motion parallax at co-located points

So how can an observer estimate the translation $\mathbf{T}$ (or direction of heading) and the rotation vector $\mathbf{\Omega}$ from image motion? Let's first address translation. One idea[2] is to compare the image velocities at two points that lie on opposite sides of a depth discontinuity. Assume first that the two points that straddle the edge are both very close to each other (and to the edge). Because the rotation component of the velocity field is smooth – and in particular, it does not depend on depth Z – the difference of the velocity vectors at the two points is the difference of their translation components. This difference is called *motion parallax.* (The phenomenon of motion parallax has been known for centuries.)

Note that since the two points lie at different depths, their translation components will be vectors with different magnitudes. Since both of these translation components point away from the direction of heading, the difference of the two velocity vectors must also point away from (or toward) the direction of heading, in particular, from Eq. (2) the difference is approximately

$$\Delta(v_x, v_y) = T_Z(\frac{1}{Z_0} - \frac{1}{Z_1})(x - x_T, y - y_T),$$

where we are assuming the two points are close to each other, $(x_0, y_0) \approx (x_1, y_1)$.

It follows that the direction of heading lies on a line that passes through the depth discontinuity (where the two points are) and whose direction is defined by the velocity difference vector above. By computing a set of such lines and finding their intersection, a vision system can estimate the direction of heading.

[2] C. Longuet-Higgins and K. Prazdny, "The Interpretation of a Moving Retinal Image", 1980

Notice that the last component of this problem is computationally identical to finding a vanishing point. There, we detect an edge and use the fact that the line through the edge passes through the vanishing point. Here we detect a location in the image (say halfway between two nearby points), and we compute a direction that points to the direction of heading. The direction of heading is, in a sense, formally equivalent to a vanishing point. This is no accident – when a observer translates through the scene, all points in the scene trace out lines in camera coordinates and these lines are parallel to the **T** vector.

Estimating motion parallax in local neighborhood

The above sketch assumes that we can find two image points that straddle a depth discontinuity, and that we can reliably measure the velocity of these two points, and hence the velocity difference as well (motion parallax). This is a big assumption. Recall that the model for estimating image velocity at a point (x_i, y_i) requires that the true image velocity is approximately constant over the neighborhood. But if the pixel is near a depth discontinuity, then this requirement will not be met. So, if we want reliable velocities of points that lie on opposite sides of a depth discontinuity (so that we can take their difference), then the points need to be some finite distance from the discontinuity. The trouble is that this new requirement means that the the rotation components of the two velocity vectors might not be exactly the same – and hence the difference of the two velocity vectors typically does not fully cancel the rotation, and so the difference vector might not exactly point to direction of heading.[3].

Here is a more concrete description on how the motion parallax direction is computed. We estimate the velocity field (v_x, v_y) and keep only estimates that are reliable. Then for each (x_i, y_i), consider the sum of 2×2 matrices

$$\sum_{(x,y)\in Ngd(x_i,y_i)} = \begin{bmatrix} v_x - v_{x,i} \\ v_y - v_{y,i} \end{bmatrix} \begin{bmatrix} v_x - v_{x,i}, v_y - v_{y,i} \end{bmatrix}$$

where the sum is in fact taken over velocity vectors that are deemed reliable. If there are two significantly different (inverse) depths present in the neighborhood, then there will be a set of difference vectors that will point in the direction of heading. This will contribution to a eigenvector of the summed matrices which is in the direction of heading – i.e. the motion parallax direction – and has a large eigenvalue (relative to the other eigenvalue, whose eigenvector will be in the perpendicular direction since the matrix is symmetric).

Estimating heading direction

As I argued at the top of this page, our problem now is to find the vanishing point defined by these motion parallax vectors. In lecture 14, we set up several Hough transform techniques for finding vanishing points. We did not bother with least squares methods, since we were assuming there were many outliers.

[3]For further analysis, see Rieger and Lawton's original paper, "Processing Differential Image Motion" JOSA, 1985. Also, for those with some background in Fourier analysis, have a look at a paper I wrote: Mann and Langer, "Spectral estimation of motion parallax and application to egomotion", JOSA 2005.

How *would* you set up a least squares method though? Let's say you have a set of image points (x_i, y_i) and a set of motion parallax unit direction vectors $(\cos\tau_i, \sin\tau_i)$. These define a set of lines

$$(x - x_i, y - y_i) \cdot (-\sin\tau_i, \cos\tau_i) = 0$$

which can be rewritten

$$\begin{bmatrix} -\sin\tau_1 & \cos\tau_1 \\ -\sin\tau_2 & \cos\tau_2 \\ \vdots & \\ -\sin\tau_N & \cos\tau_N \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x_1\sin\tau_1 + y_1\cos\tau_1 \\ -x_2\sin\tau_2 + y_2\cos\tau_1 \\ \vdots \\ -x_N\sin\tau_n + y_N\cos\tau_N \end{bmatrix}$$

and since the direction of heading (vanishing point) lies near each of these lines, we solve for the direction of heading $(x, y) = (\frac{T_X}{T_Z}, \frac{T_Y}{T_Z})$ by least squares.

Estimating observer rotation $\mathbf{\Omega}$

We can also use the (x_i, y_i, τ_i) values to solve for Ω. Consider Eq. (4) which shows the rotation component $\vec{v}_{i,\Omega}$ of the velocity vector at this point. Let's examine the projection of $\vec{v}_{i,\Omega}$ onto the unit vector that is perpendicular to the translation component v_T. The translation component is perpendicular to $(-\sin\tau_i, \cos\tau_i)$ and so the projection of $\vec{v}_{i,\Omega}$ onto $(-\sin\tau_i, \cos\tau_i)$ must be the same as the projection of $(v_x, v_y)_i$ onto $(-\sin\tau_i, \cos\tau_i)$, and so

$$(-\sin\tau_i, \cos\tau_i) \cdot (v_x, v_y)_i \; = \; (-\sin\tau_i, \cos\tau_i) \begin{bmatrix} \frac{x_i y_i}{f} & -f - \frac{x_i^2}{f} & y \\ f + \frac{y_i^2}{f} & -\frac{x_i y_i}{f} & -x_i \end{bmatrix} \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix}$$

which is a linear equation with three unknowns $(\Omega_x, \Omega_y, \Omega_z)$. If we have estimated the image velocity $(v_x, v_y)_i$ at $N \geq 3$ points, then have N of these equations, and so we can solve for three unknowns, i.e. the rotation vector, again using least squares. (Keep in mind that the estimated velocities $(v_x, v_y)_i$ and not exactly the same as the true velocities – i.e. there are errors – and similarly the estimated τ_i are not the same as the true parallax directions – again, errors. That's why we need to use least squares.

Estimating Depth

Once we know the observer's rotation and translation, we can estimate the depth Z at any point where we have a reliable estimate of the image velocity. We just subsitute our estimated values of $\mathbf{\Omega}, \mathbf{T}, (v_x, v_y)_i$, and (x_i, y_i) into Eqs. (2), (4), 1 and solve for the only remaining variable $Z(x_i, y_i)$. Keep in mind that we only know $\mathbf{T}$ up to some unknown scale factor, so we can only estimate $Z(x_i, y_i)$ up to this scale factor as well.