CHAPTER 34

Optic Flow

The stream of pictures that comes from a camera moving through a world of stationary objects reveals a great deal about how the camera moves and the shape of the world. A point that projects to $\mathbf{x}_1$ in frame 1 moves to $\mathbf{x}_2$ in frame 2. This movement can be visualized by placing the *flow vector* $\mathbf{v}(\mathbf{x}) = \mathbf{x}_2 - \mathbf{x}_1$ at each location in frame 1. Figure 34.1 shows some fields of flow vectors that might be observed by a moving camera. These flow fields convey a great deal of information about how the camera moved and what the shape of the world is. Estimating such a flow field from a pair of images is referred to as an *optic flow* problem. There are two big families of method for estimating optic flow: elementary methods, which use various forms of geometric, physical and statistical reasoning to recover flow fields; and regression methods, which follow the recipe of Chapter 22.6.

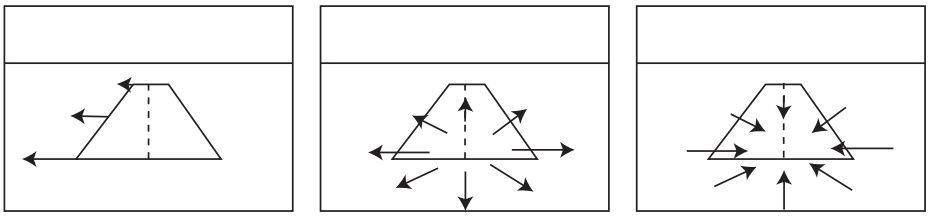

FIGURE 34.1: *Flow offers simple, clear signals about how one is moving with respect to the world. Look out of a bubble canopy on the nose of a small aircraft. If you see the flow on the* **left**, *you are heading from left to right. If you see the flow in the* **center**, *you are leaving the runway behind. If you see the flow on the* **right**, *you are heading towards the near end of the runway. Various reflexes related to flow patterns are found in a number of flying animals. For example, the flow on the right – if large enough compared to the size of the object in the image – will often trigger some collision management reflex.*

34.1 OPTIC FLOW AS A CUE

Here is a useful trick, widely familiar to flying animals. Compute some measure S of the size of an object in an image. You could use diameter, area, or a similar measure. Move the camera, or eye and compute the time derivative of that measure, S_t. The expression S/S_t is an estimate of time to contact *which is unaffected by camera calibration* (Figure 34.2; **exercises**). If this number is less than some threshold which has to do with your ability to accelerate, etc. you're in trouble. This trick is a simple manifestation of the information that optic flow has to offer about the movement of an agent in the world.

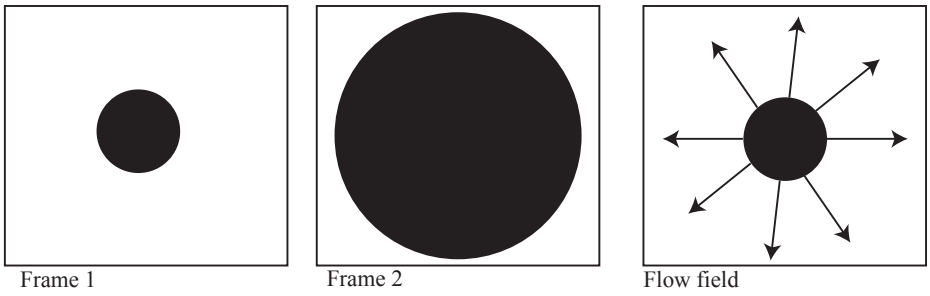

FIGURE 34.2: *Images 1 and 2 here offer an alarming warning of an impending collision, which is captured in the flow field. In my experience, you can startle an audience with carefully constructed slides showing scenes like these two. A darkened room and a bright projector seem to help.*

34.1.1 Simple Applications of Flow Estimates

Flow is informative. A moving agent that sees the two frames of Figure 34.2 is in trouble. It is likely to hit the object in the center of frame 1 quite soon. The flow field signals this trouble rather efficiently.

Now assume that the camera is calibrated, starts in the standard configuration, and translates, so camera 2's focal point is at $\mathbf{p}$ in camera 1's frame. Equivalently, the point at $\mathbf{X}$ in camera 1's frame is at $\mathbf{X} - \mathbf{p}$ in camera 2's frame (check you are sure about the minus sign). Now the flow in image 1 will be

$$\begin{bmatrix} \frac{X_1 - p_1}{X_3 - p_3} \\ \frac{X_2 - p_2}{X_3 - p_3} \end{bmatrix} - \begin{bmatrix} \frac{X_1}{X_3} \\ \frac{X_2}{X_3} \end{bmatrix}$$

so the flow will be zero at the point

$$\begin{bmatrix} \frac{p_1}{p_3} \\ \frac{p_2}{p_3} \end{bmatrix}$$

in image 1 *however far away the corresponding 3D point is.* This point is often referred to as the focus of expansion of the flow. It is also the epipole (**exercises**). If the flow points away from this point, the camera will hit the focus of expansion if it keeps moving as it has been moving. Similarly, if the flow points toward this point, the camera is escaping from that point in the scene. The three flow fields of Figure 34.1 signal quite different things to a flying agent. These examples are built around the epipole of the first frame, but flow conveys more information than an epipole does.

You can recover 3D shape information from flow. Figure 34.3 shows a calibrated aerial camera translating parallel to the image plane while looking down on a scene. For simplicity, the coordinate system is camera 1's standard coordinate

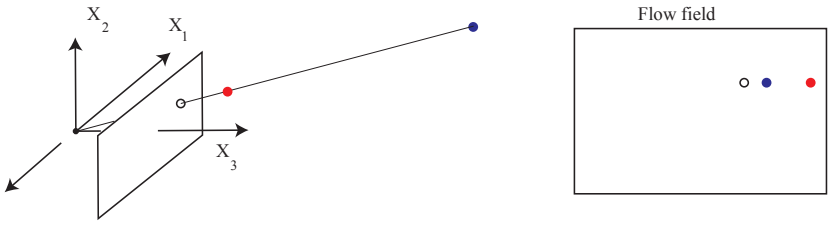

FIGURE 34.3: *Flow depends on depth. The camera (***left***) translates parallel to the image plane along the small arrow. A point at the red location and will move a long way in the image; a point at the blue location will move by a much smaller amount. The text shows that the flow is proportional to the reciprocal of depth.*

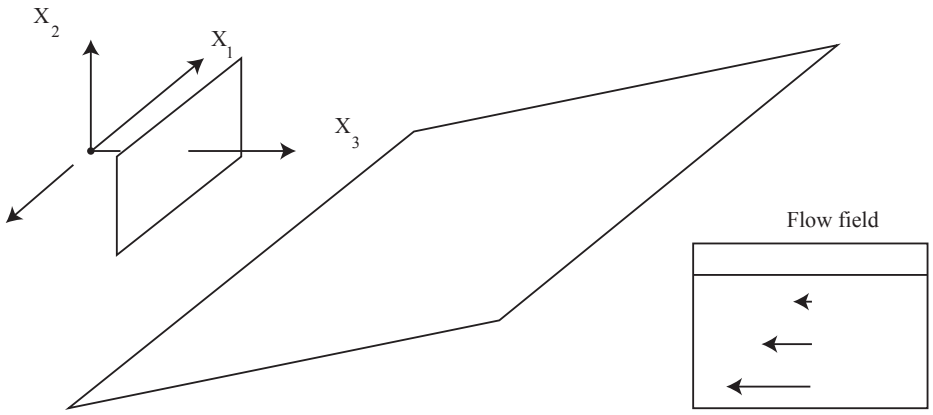

FIGURE 34.4: *A camera looks out of a train window at right angles to the direction of travel (the small arrow) and views a ground plane. As the text shows, the flow field looks like the inset – the flow at the horizon is zero, and flow grows linearly as you move down from the horizon.*

system. The flow at $\mathbf{x} = [x_1, x_2]^T$ in the image plane is

$$\begin{bmatrix} -\frac{p_1}{X_3(x_1,x_2)} \\ -\frac{p_2}{X_3(x_1,x_2)} \end{bmatrix}$$

meaning that if you know the flow field and you know the camera translation, you immediately know the depth to each point in the camera. Alternatively, recovering the flow field is very like recovering the depth.

Another simple example of 3D shape information from flow is a calibrated camera on a moving vehicle above a ground plane (Figure 34.4). Camera 1 is in the standard camera coordinate system. The ground plane does not pass through the focal point, so write $bX_2 + cX_3 - 1 = 0$ in the camera coordinate system for the ground plane. The camera translates in the X_1 direction by t_x. image plane. One

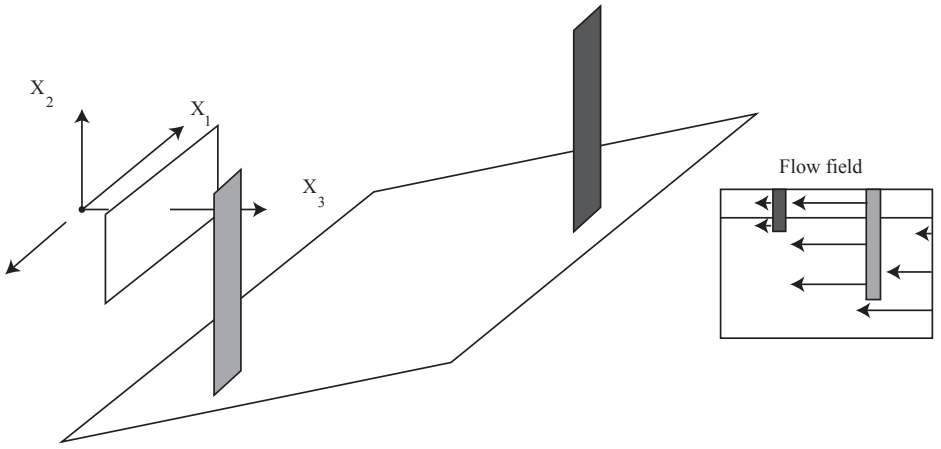

FIGURE 34.5: *A camera looks out of a train window at right angles to the direction of travel (the small arrow) and views a ground plane with two trees on it. The flow field on the ground plane looks like the inset of Figure 34.4, but the flow at the trees is different. The nearby tree has a large flow, and the distant tree has a small flow.*

example of this case is a train travelling on a ground plane, with a camera looking out the window at right angles to the direction of travel. The ray from the focal point through $\mathbf{x} = [x_1, x_2]$ is $t\,[x_1, x_2, 1]$. At the point where the ray pierces the plane, $X_3 = 1/(bx_2 + c)$. This means the flow is

$$\begin{bmatrix} -t_x(bx_2 + c) \\ 0 \end{bmatrix}.$$

This flow is linear in image position. Notice the flow is zero when $(bx_2 + c) = 0$, which is also the horizon of the plane (check this **exercises**). The vector

$$\mathbf{p} = \frac{1}{\sqrt{a^2 + b^2}} \begin{bmatrix} b \\ -a \end{bmatrix}$$

is a unit vector perpendicular to the horizon. Choose a point $\mathbf{h}$ on the horizon, and consider the ray in the image $\mathbf{h} + t\mathbf{p}$. The length of the flow vector increases linearly in t – as you work your way down the line, the corresponding image point is closer and the flow gets bigger (Figure 34.4). This illustrates a general point – when the camera moves, closer points get longer flow vectors and further points get shorter flow vectors.

34.1.2 Difficulties Estimating Flow

The train example can be used to show some difficulties in estimating flow. Figure 34.5 shows the flow field when there are two trees sticking out of the ground plane. The flow inside the boundary of either tree is very different from that outside the boundary. This should – and does – generate problems estimating the flow.

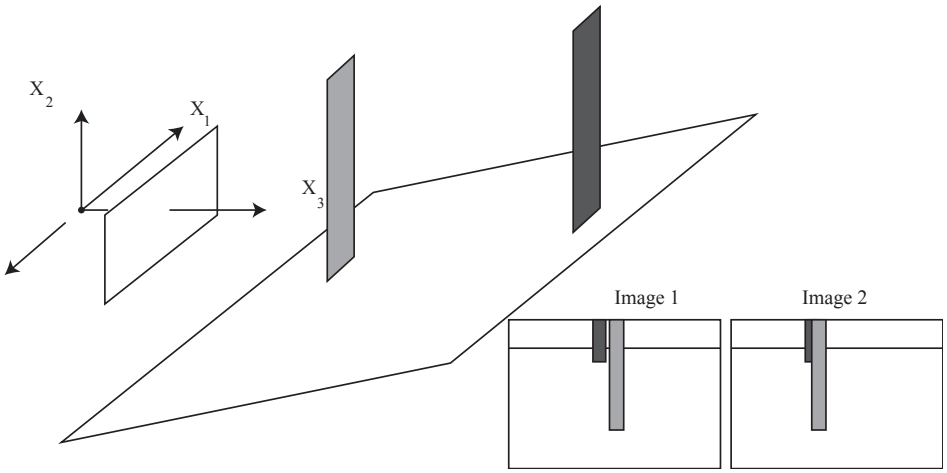

FIGURE 34.6: *A camera looks out of a train window at right angles to the direction of travel (the small arrow) and views a ground plane with two trees on it. Because the nearby tree moves by more than the distant tree (as in Figure 34.5), for this geometry the distant tree will disappear behind the nearby tree. Reversing the direction of camera movement will get an example where the distant tree reappears from behind the nearby tree. In either case, there are pixels in one image that aren't seen in the other, which must create estimation problems.*

You can't estimate flow at a pixel that is there in one image but not in the other image. Parts of the distant tree disappear behind the nearby tree in Figure **??**, which must create estimation problems – you can't estimate flow at a pixel if the piece of surface that produced the pixel isn't in the second image. Similarly, parts of the distant tree might reappear from behind the nearby tree.

Fast moving objects present particular difficulties. A fast moving object may cover a pixel only for part of the time that the camera shutter is open, meaning that the value reported by the pixel is some weighted average of the background color and the object color. Some locations may see mostly background, and others will see mostly object (Figure 34.7). The result is blur around the boundary of the object,

FIGURE 34.7: *Fast moving objects produce heavily blurred images. These are five consecutive frames of a sequence where I dropped a pen on my laptop. Notice in the* **far left** *frame, my hand is in focus as is the pen. Dropping the pen involves a fairly fast movement of my thumb, blurred in the* **left** *frame. As the pen accelerates under gravity, it is increasingly blurred.*

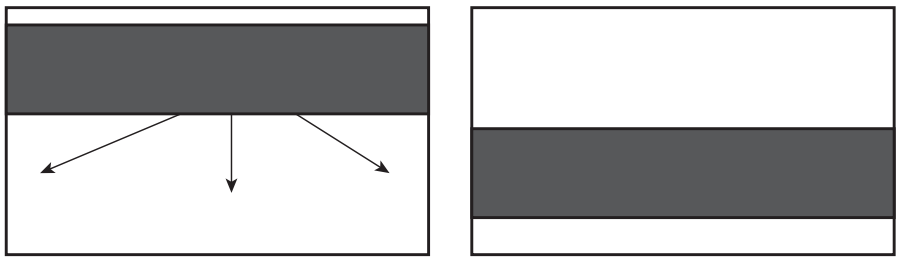

FIGURE 34.8: *Each of the three flow vectors is a plausible explanation of the flow on this moving bar, because you cannot tell how it has translated horizontally. This effect is known as the aperture problem.*

usually referred to as *motion blur*. This will affect the image gradient estimates and the time derivative of intensity, and so tend to make the flow constraint unreliable. Small, fast moving objects are particularly bad, because there may be few pixels on the object and all could be affected by motion blur.

Figure 34.8 illustrates another basic problem estimating flow. Locally, the flow at a straight edge is ambiguous, because you can't tell whether there is any flow *along* the edge. This isn't just a property of edges. If the line segment in the figure was an *isophote* (a curve of equal brightness points in the image) the same problem would occur. You would not be able to tell *locally* if there was any flow *along* the isophote. This problem is known as the *aperture problem*.

34.2 FLOW ESTIMATION BY CONSTRAINTS

You can tell where corners are, so you can tell where they have gone to, which means flow at corners is unambiguous. This is the keystone of flow estimation. Because there are some image locations where flow is quite accurately known, you can estimate a flow field by "filling in" flow between these locations. The "filled in" flow needs to meet constraints. This suggests an important early strategy for estimating flow. Get the best estimate you can at each image location, then smooth the estimates so that reliable estimates (at corners, for example) improve the quality of unreliable estimates (along straight segments)

34.2.1 Constraints on Flow

At time t, a point $\mathbf{V}$ on an object projects to the point $\mathbf{x}$ in the image, producing intensity $I(\mathbf{x}, t)$. The camera then moves by a small amount and at time $t + \delta t$ a new image is obtained. The point now projects to $\mathbf{x} + (\delta t)\mathbf{u}$, where $\mathbf{u}$ is the flow. Then $I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t)$ is the same as $I(\mathbf{x}, t)$, so

$$I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t) - I(\mathbf{x}, t) = 0.$$

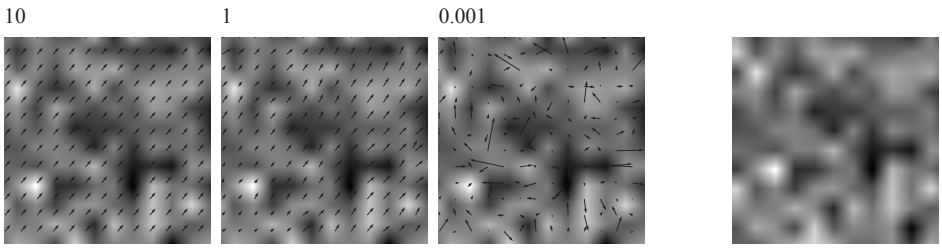

FIGURE 34.9:

If the flow and δt are small enough, a Taylor series is a good approximation, and you find

$$I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t) \approx (I(\mathbf{x}, t) + (\delta t)\left[(\nabla I)^T\mathbf{u} + \frac{\partial I}{\partial t}\right]$$

so that for sufficiently small δt

$$(\nabla I)^T\mathbf{u} + \frac{\partial I}{\partial t} = 0.$$

For convenience, I will call this the *optical flow equation*. At any pixel, it is easy to compute estimates ∇I and $\frac{\partial I}{\partial t}$, so this equation could yield one linear constraint on the flow vector $\mathbf{u}$. This isn't enough to recover the flow vector exactly, but can strongly constrain the flow.

The linearized constraint has some issues. The assumption $I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t) - I(\mathbf{x}, t) = 0$ is not always true. The assumption boils down to assuming that objects do not change appearance when they move. This very often, but not always, true. There are a variety of reasons the amount of light reflected from the surface of the object to the camera might change when the object moves by a small amount. If the material of the object has a strong specular or glossy component, the intensity might change because the angles to sources will change. Alternatively, the object might move into a shadow volume. Finally, the illumination intensity varies across space (for example, as a result of interreflections). Experience teaches that the assumption is quite reliable, but not exact. Even if the assumption holds, the movement might be so large that linearizing the expression is reckless. This issue is a particular nuisance for small, fast moving objects. It can be managed by coarse-to-fine search (Section 22.6).

34.2.2 Estimating Smoothed Flow

Now think about the flow field around a blob in the image. Flow at a point on the boundary of the blob should be quite like flow at points nearby, otherwise the blob is disintegrating. A useful smoothing constraint is to assume that the gradient of each component of flow is small. Furthermore, the constraint is only true if the estimates ∇I and $\frac{\partial I}{\partial t}$ are exact. This suggests recovering flow by minimizing a weighted sum of the gradient magnitudes of the flow and the extent to which the flow violates constraints.

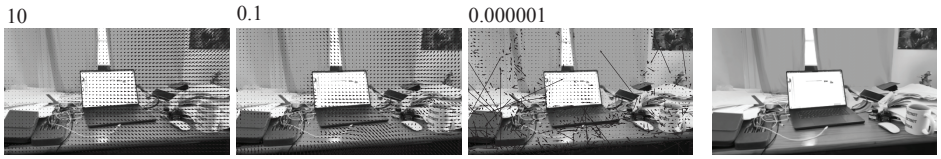

FIGURE 34.10: *Horn-Schunk estimates of flow for a pair of images of my laptop, with a small time interval. The Taylor series approximation is reliable. When the smoothing is large, the flow field is heavily smoothed and not particularly informative. As the smoothing reduces, the flow field is more variable. When there is very low smoothing, the flow field becomes completely unreliable.*

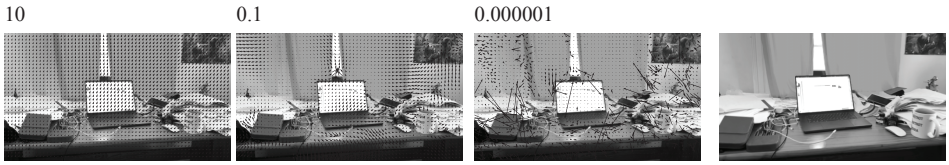

FIGURE 34.11: *Horn-Schunk estimates of flow for a pair of images of my laptop, with a moderate time interval. The Taylor series approximation is not reliable. When the smoothing is large, all pixels have about the same flow; as the smoothing reduces, the flow field is more variable but not particularly good.*

You can see this as an instance of the master recipe for denoising, but now applied to flow. Write $\mathbf{u}$ and $\mathbf{v}$ for vectors giving the value of the flow at each pixel. The recipe seeks an estimated flow field that is (a) close to meeting the constraints and (b) like a real flow field. Write

$$\begin{aligned} C(\mathbf{u},\mathbf{v}) &= \begin{array}{c}[\text{the extent to which } \mathbf{u}, \mathbf{v} \text{ is consistent with image data}] + \\ [\text{the extent to which } \mathbf{u}, \mathbf{v} \text{ is smooth}]\end{array} \\ &= [data] + [penalty] \end{aligned}$$

and choose $\mathbf{u}$, $\mathbf{v}$ that minimize this cost function. Notice you can think about each component of flow as an image (one value per pixel).

Write $\mathcal{D}_x$ and $\mathcal{D}_y$ for matrices that compute a smoothed gradient from a vector representing a flow component or the image (rearrange the finite differences of Section 5.1.2, or the derivative of gaussians of Section 5.2.2, **exercises**). Write $\mathbf{I}_1$ for image 1 expressed as a vector, $\mathbf{I}_x$ for $\mathcal{D}_x\mathbf{I}_1$, $\mathbf{I}_y$ for $\mathcal{D}_y\mathbf{I}_1$, $\mathbf{I}_t$ for $\mathbf{I}_2 - \mathbf{I}_1$, and $\mathsf{diag}\,(\mathbf{w})$ for the operator that constructs a matrix with $\mathbf{w}$ on the diagonal. Then choose the flow field that minimizes

$$\begin{array}{rcl} [\mathsf{diag}\,(\mathbf{I}_x)\mathbf{u} + \mathsf{diag}\,(\mathbf{I}_y)\mathbf{v} + \mathbf{I}_t]^T [\mathsf{diag}\,(\mathbf{I}_x)\mathbf{u} + \mathsf{diag}\,(\mathbf{I}_y)\mathbf{v} + \mathbf{I}_t] & + & [data] \\ \alpha \left[\mathbf{u}^T(\mathcal{D}_x^T\mathcal{D}_x + \mathcal{D}_y^T\mathcal{D}_y)\mathbf{u}\mathbf{v}^T(\mathcal{D}_x^T\mathcal{D}_x + \mathcal{D}_y^T\mathcal{D}_y)\mathbf{v}\right] & & [penalty]\,. \end{array}$$

Here α is a parameter that controls the degree of smoothing. Notice that large α leads to a very smooth flow field which may not meet the constraints, and small α

will produce a less smooth field that is more likely to meet the constraint. Solving this flow equation is straightforward (**exercises**).

The method I have described is known as the *Horn-Schunk* method, and was originally described in []. It will be helpful to write the data term as

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{1} \end{bmatrix}^T \begin{bmatrix} \mathsf{diag}\left(\mathbf{I}_x\right)\mathsf{diag}\left(\mathbf{I}_x\right) & \mathsf{diag}\left(\mathbf{I}_x\right)\mathsf{diag}\left(\mathbf{I}_y\right) & \mathsf{diag}\left(\mathbf{I}_x\right)\mathsf{diag}\left(\left(\mathbf{I}_t\right)\right) \\ \mathsf{diag}\left(\mathbf{I}_y\right)\mathsf{diag}\left(\mathbf{I}_x\right) & \mathsf{diag}\left(\mathbf{I}_y\right)\mathsf{diag}\left(\mathbf{I}_y\right) & \mathsf{diag}\left(\mathbf{I}_y\right)\mathsf{diag}\left(\left(\mathbf{I}_t\right)\right) \\ \mathsf{diag}\left(\mathbf{I}_x\right)\mathsf{diag}\left(\left(\mathbf{I}_t\right)\right) & \mathsf{diag}\left(\mathbf{I}_y\right)\mathsf{diag}\left(\left(\mathbf{I}_t\right)\right) & \mathsf{diag}\left(\left(\mathbf{I}_t\right)\right)\mathsf{diag}\left(\left(\mathbf{I}_t\right)\right) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{1} \end{bmatrix}^T$$

.

34.2.3 Pooled Estimates

Assume that the optical flow equation applies. You have one constraint on flow at every pixel, so you can't recover the flow at a given pixel. But if you assume that all pixels in a $N \times N$ window have the same flow, you might be able to. Write $(\nabla I)_{ij}$ for the image gradient at the i, j'th pixel in the window, $I_{t,ij}$ for the time derivative at that pixel, and find

$$\begin{bmatrix} (\nabla I)_{11}^T \\ \dots (\nabla I)_{NN}^T \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t,00} \\ \dots \\ I_{t,NN} \end{bmatrix}.$$

Now write $\mathbf{d}_{ij} = \left[(\nabla I)_{ij}^T, I_{t,ij}\right]^T$. To obtain a least squares solution, you would choose the (u, v) that minimizes

$$[u, v, 1]^T \sum_{ij} \left[\mathbf{d}_{ij}\mathbf{d}_{ij}^T\right] [u, v, 1].$$

Now imagine doing this at every pixel. This requires a minor suspension of disbelief. If you assume that the flow is constant in an $N \times N$ window, *and* estimate it in overlapping windows, it must be the same in every window. But the solution minimizes a cost function rather than exactly satisfying an equation, so ignore this concern. Doing so would involve some minor surgery on the data term in the Horn-Schunk method. Compute the average of each derivative in an $N \times N$ window around each pixel, making the window smaller as required so you can obtain an average at the boundaries and corners. Write $\bar{\mathbf{I}}_x$ for the average of $\mathbf{I}_x$ obtained in this way, and so on. Now solve

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{1} \end{bmatrix}^T \begin{bmatrix} \mathsf{diag}\left(\bar{\mathbf{I}}_x\right)\mathsf{diag}\left(\bar{\mathbf{I}}_x\right) & \mathsf{diag}\left(\bar{\mathbf{I}}_x\right)\mathsf{diag}\left(\bar{\mathbf{I}}_y\right) & \mathsf{diag}\left(\bar{\mathbf{I}}_x\right)\mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right) \\ \mathsf{diag}\left(\bar{\mathbf{I}}_y\right)\mathsf{diag}\left(\bar{\mathbf{I}}_x\right) & \mathsf{diag}\left(\bar{\mathbf{I}}_y\right)\mathsf{diag}\left(\bar{\mathbf{I}}_y\right) & \mathsf{diag}\left(\bar{\mathbf{I}}_y\right)\mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right) \\ \mathsf{diag}\left(\bar{\mathbf{I}}_x\right)\mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right) & \mathsf{diag}\left(\bar{\mathbf{I}}_y\right)\mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right) & \mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right)\mathsf{diag}\left(\left(\bar{\mathbf{I}}_t\right)\right) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{1} \end{bmatrix} + [\textit{data term}]$$
$$\alpha \left[\mathbf{u}^T(\mathcal{D}_x^T\mathcal{D}_x + \mathcal{D}_y^T\mathcal{D}_y)\mathbf{u}\mathbf{v}^T(\mathcal{D}_x^T\mathcal{D}_x + \mathcal{D}_y^T\mathcal{D}_y)\mathbf{v}\right] \quad [\textit{smoothness}].$$

34.3 FLOW ESTIMATION WITH PARAMETRIC MODELS

In some useful cases, it is straightforward to write out the flow as a function of (say) camera motion. This gives a parametric flow field. You then solve for the parameters using the optical flow equation or other procedures. The advantage of

this procedure is that it manages the ambiguity in the optical flow equation. The procedure is particularly useful for objects moving parallel to the ground, or high above the ground. Model the ground as a plane, and you can get a fair estimate of flow.

34.3.1 Parametric Flow Examples: Looking at a Ground Plane

A calibrated camera looks at a ground plane from a fixed height above it. The camera translates parallel to the ground plane, and the image plane is perpendicular to the ground plane. Set up a coordinate system so that, at time 1, the camera is in the standard coordinate system for a perspective camera. The ground plane will be $Y = c$ for some constant. The camera translates parallel to the ground plane, so by $(t_x, 0, t_z)$. The time interval is small. A point at $(x, y)^T$ in the first image plane is at

$$\begin{bmatrix} \frac{xc}{y} \\ c \\ \frac{c}{y} \end{bmatrix}$$

in 3D. This means the flow induced by the translation will be

$$\begin{bmatrix} \frac{x - y\frac{t_x}{c}}{1 - y\frac{t_z}{c}} - x \\ \frac{y}{1 - y\frac{t_z}{c}} - y \end{bmatrix}$$

Here I have grouped the variables quite deliberately to show an ambiguity. You can estimate two parameters, $\frac{t_x}{c}$ and $\frac{t_y}{c}$. This is an ambiguity in the geometry, but not in the flow. A large movement by a camera high above the ground plane results in the same flow as a small movement by a camera closer to the plane. Insert this flow into the optical flow equation, cross multiply and find

$$\frac{\partial I}{\partial x}\left[x - y\frac{t_x}{c} - x(1 - \frac{t_z}{c})\right] + \frac{\partial I}{\partial y}\left[y^2\frac{t_z}{c}\right] + \frac{\partial I}{\partial t}\left[1 - y\frac{t_z}{c}\right] = 0$$

And you could solve this using least squares for the two parameters.

34.3.2 Parametric Flow Examples: Looking at a Ground Plane with Relief

Now the calibrated camera looks directly down at a ground plane from a fixed height above it. The camera translates parallel to the ground plane, and the image plane is perpendicular to the ground plane. The ground plane will be $Z = c$ for some constant. But the ground isn't really a plane. Houses, trees, people and such stick up out of the ground, and generate variations in the flow field, so write the ground as $Z = c + \delta Z$ where δZ is small compared to c and represents this relief. Choose the ground plane so that the relief is always non-negative.

Now set up a coordinate system so that, at time 1, the camera is in the standard coordinate system for a perspective camera. The camera translates parallel to the ground plane, so by $(t_x, t_y, 0)$. The time interval is small. A Taylor series yields

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{X}{c+\delta Z} \\ \frac{Y}{c+\delta Z} \end{bmatrix} \approx \begin{bmatrix} \frac{X}{c}(1 - \frac{\delta Z}{c}) \\ \frac{Y}{c}(1 - \frac{\delta Z}{c}) \end{bmatrix}$$

so that the flow is

$$\begin{bmatrix} \frac{-t_x}{c}(1 - \frac{\delta Z}{c}) \\ \frac{-t_y}{c}(1 - \frac{\delta Z}{c}) \end{bmatrix}$$

. Again I have grouped the variables quite deliberately to show an ambiguity. You can estimate two parameters, $\frac{t_x}{c}$ and $\frac{t_y}{c}$ and a relative height field $\frac{\delta Z}{c}$. This is an ambiguity in the geometry, but not in the flow.

A straightforward strategy to recover flow from a pair of images uses alternating reestimation. Start by assuming the relief is 0. Now iterate:

- for fixed relief, use least squares on the optical flow equation to estimate $\frac{t_x}{c}$ and $\frac{t_y}{c}$;
- for fixed $\frac{t_x}{c}$ and $\frac{t_y}{c}$, use least squares on the optical flow equation to estimate relief.

If the relief or the camera translation are large compared to c, either the optical flow equation or the Taylor series for the flow will become unreliable. Multiscale estimates help in this case.

34.3.3 Multiple Objects on a Plane

A calibrated camera translates in the x direction at a fixed height above a ground plane. The ground plane is at $Y = c$ in camera coordinates, so the image plane is perpendicular to the ground plane. A collection of N flat objects stick up out of the ground plane at fixed depths (Figure **??**; the i'th object is at $Z = d_i$). For this setup, the flow in the y direction in the image plane is always zero. Check that, if the pixel at $(x, y)^T$ sees the ground plane, the z coordinate is c/y. The x component of flow at a pixel takes one of $N + 1$ values

$$\begin{cases} -\frac{t_x c}{y} & \text{if the pixel sees the ground plane} \\ -\frac{t_x}{c_1} & \text{if the pixel sees object 1} \\ \dots & \dots \\ -\frac{t_x}{c_N} & \text{if the pixel sees object N} \end{cases}$$

Assuming you don't know the ground plane constant, the translation, or the depth of any object, you still have only $N + 1$ constants to estimate. Estimating these constants is tightly intertwined with segmenting the image into objects and ground plane.

If the flow at pixel $(x, y)^T$ is (u, v), then $I_2(x+u, y+v)$ should be very similar to $I_1(x, y)$. The term

$$(I_1(x + u, y + v) - I_2(x, y))^2$$

is sometimes referred to as *photometric error*. This error can be used to estimate simple flow rather reliably.

Here is a simple estimation algorithm, which is quite strongly analogous to k-means. Assume that, for every pixel, you know which of the $N + 1$ cases applies; that is, whether it is background, or one of the objects. Now take all pixels that lie on object i, and search for the u such that

$$\sum_{(x,y) \in \text{pixels on } i} (I_2(x - u, y) - I_1(x, y))^2$$

is minimized. Then $(-u, 0)$ is the flow for that object. For the background, search for the c such that

$$\sum_{(x,y)\in\text{background pixels}} (I_2(x - c/y, y) - I_1(x, y))^2$$

is minimized; the flow at the background is the $(-c/y, 0)$. Now assume that you know the flow for each object and the background. Then assign each pixel to the segment that has the flow that produces the lowest photometric error. The choices are

$$\begin{cases} (I_2(x - c/y, y) - I_1(x, y))^2 & \text{if the pixel is background} \\ (I_2(x - c_1, y) - I_1(x, y))^2 & \text{if the pixel is on object 1} \\ \cdots & \cdots \\ (I_2(x - c_N, y) - I_1(x, y))^2 & \text{if the pixel is on object N} \end{cases}.$$

34.3.4 Mixtures of More Complex Parametric Flow Models

The objects on a plane example is a simple example of a powerful procedure. For the general form, assume the flow field consists of a set of regions. In each region, the flow follows a parametric model with a manageable number of parameters. In the objects on a plane example, there was one parameter per segment, but you could have more. A flow model where flow at the pixel at $(x, y)^T$ is given by

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \mathcal{M} \begin{bmatrix} 1 \\ x \\ y \\ 1/x \\ 1/y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

covers a number of useful cases (Figure **??**). Estimating a flow field of this form can be easier than estimating a general flow field, because there are very few constants to estimate. Further, the flow estimate is very tightly linked to a segmentation of the image that is interesting – pixels are grouped together in a segment if they move in similar ways.

This model has properties like that of the previous section. If you know which flow model a pixel belongs to, you can estimate the parameters for that flow model by minimizing photometric error. Further, if you know the parameters of the flow models, you can tell which segment a pixel belongs to by checking which of the flow models minimizes the photometric error. Using the outline sketched in the previous section should seem reasonable to you.

The algorithm has one weakness. Each pixel is assigned to the segment whose flow yields the best photometric error. If two segments have about the same photometric error, this strategy could lose significant amounts of information. There is little justification for one segment's flow estimate getting all the constraint from that pixel, and the other segment's flow estimate getting none.

FIGURE 34.12:

Improvements can be obtained with *soft assignment*, where each pixel can be assigned to more than one segment using a weight (the original strategy is then called *hard assignment*). To avoid overcounting, the weights must sum to one. The photometric error at a pixel yields a natural set of soft weights. Assume there are N different parametric models (equivalently, segments). Write θ_i for the parameters of the i'th segment, $\mathbf{f}(\mathbf{x}; i, \theta_i)$ for the flow predicted by the i'th model at $\mathbf{x}$ in the first frame and $E(\mathbf{x}; i, \theta_i) = (I_2(\mathbf{x} + \mathbf{f}(\mathbf{x}; i, \theta_i)) - I_1(\mathbf{x}))^2$ for the photometric error resulting from using the i'th segment to predict flow $\mathbf{x}$. Then a natural set of weights is

$$w_i(\mathbf{x}) = \frac{e^{\frac{-E(\mathbf{x};i,\theta_i)}{2\sigma^2}}}{\sum_u e^{\frac{-E(\mathbf{x};u,\theta_u)}{2\sigma^2}}}.$$

Here σ is a scale that you choose. Too small a value of σ and the soft assignment is largely a hard assignment because one weight is much larger than all the others. Too large a value of σ and the soft assignment strategy oversmoothes the estimated flow field.

Assume a moving camera in a stationary world. The flow vector attached to a point that is far away from the cameras will be smaller than that attached to a point that is close to the cameras (Figure 34.12).

34.4 INTEREST POINTS

34.4.1 Scattered Reliable Flow Estimates

Assume that the optical flow equation applies. You have one constraint on flow at every pixel, so you can't recover the flow at a given pixel. But if you assume that all pixels in a $N \times N$ window have the same flow, you might be able to. Write $(\nabla I)_{ij}$ for the image gradient at the i, j'th pixel in the window, $I_{t,ij}$ for the time derivative at that pixel, and find

$$\begin{bmatrix} (\nabla I)_{11}^T \\ \dots (\nabla I)_{NN}^T \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t,00} \\ \dots \\ I_{t,NN} \end{bmatrix}.$$

A least squares solution is obtained by solving

$$\left[\sum_{ij} (\nabla I)_{ij} (\nabla I)_{ij}^T \right] \begin{bmatrix} u \\ v \end{bmatrix} = \mathcal{M} \begin{bmatrix} u \\ v \end{bmatrix} = \left[- \sum_{ij} I_{t,ij} (\nabla I)_{ij} \right].$$

You should notice an old friend from Section 22.6. The matrix $\mathcal{M}$ is the same as the matrix used to build a corner detector. This is not accidental. If this matrix has large eigenvalues, the least squares solution will be reliable, but if one or both eigenvalues are small, it will not. Equivalently, because you can tell where a corner is, you can tell where it has gone to.

Now pass over the image, and at each window compute $\mathcal{M}$ and check its eigenvalues. You might use the Harris criterion, but this isn't compulsory. Locations where they're both big are likely to form blobs, so use non-maximum suppression as in Section 22.6. You now have a set of points where you can rely on the flow estimate, so compute the estimate at these points. You now have flow estimates at scattered points in the image.

34.5 IMPROVING FLOW ESTIMATES

34.5.1 Improving Estimates with Robust Methods

34.5.2 Improving Estimates with Coarse-to-Fine Search

here?

34.6 REGRESSION METHODS FOR OPTIC FLOW

34.7 SMALL FAST OBJECTS AND FLOW