Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 18: Motion - Optic Flow

Initialize

```
Off[General::spell1];
SetOptions[ArrayPlot, ColorFunction → "GrayTones",
  DataReversed → True, Frame → False, AspectRatio → Automatic,
  Mesh → False, PixelConstrained → True, ImageSize → Small];
SetOptions[ListPlot, ImageSize → Small];
SetOptions[Plot, ImageSize → Small];
SetOptions[Plot3D, ImageSize → Small, ColorFunction → "GrayTones"];
SetOptions[DensityPlot, ImageSize → Small, ColorFunction → GrayLevel];
nbinfo = NotebookInformation[EvaluationNotebook[]];
dir = ("FileName" /. nbinfo /. FrontEnd`FileName[d_List, nam_, ___] :→ ToFileName[d]);
```

Outline

Last time

Linearizing shape from shading. Learning shape from shading

Formal ambiguities in the generative model, the bas-relief transform
Python/IPython overview and resources

Today

- Early motion measurement--types of models
  - correlation
  - gradient
  - feature tracking

- Functional goals of motion measurements

- Optic flow
  - Definition: motion field vs. optic flow
  - Cost function (or energy) descent model
  - A posteriori and a priori constraints
  - Gradient descent algorithms
  - Computer vs. human vision and optic flow
    - -- area vs. contour

Introduction to motion

It is convenient to divide motion processing up into several stages:

- Early local motion measurement: from *optic flow* (image intensity change) to *motion flow* (local surface movements)
- Selection & integration of local motion flow information
- End uses of motion measurements, e.g. shape/structure from motion, direction of heading, ...

This lecture looks at mechanisms of early motion measurements, and in particular the approximation of the 2D motion field using optic flow.

The mechanisms of early motion measurement are often interpreted as preliminary steps to obtain local estimates of some aspect of *motion flow*. Each point of a surface moving relative to the eye describes a 3D trajectory that maps onto a 2D image trajectory. The problem of motion flow is to estimate at each point in space and time a vector that points in the direction of the image point's trajectory, and whose magnitude is the speed. The pattern of vectors at a given time is the “instantaneous” projected velocity field.

Estimates of the projected velocity field can then be used to estimate useful scene related information such as direction of heading and spatial layout. The sort of image measurements needed depends on the later uses of these motion measurements. For example, the full vector field may not be needed, because one might be satisfied with only direction information. And even this might be superfluous if we only need to know whether something has moved or not. Let's consider the jobs of vision that use motion.

Functional goals of motion measurement

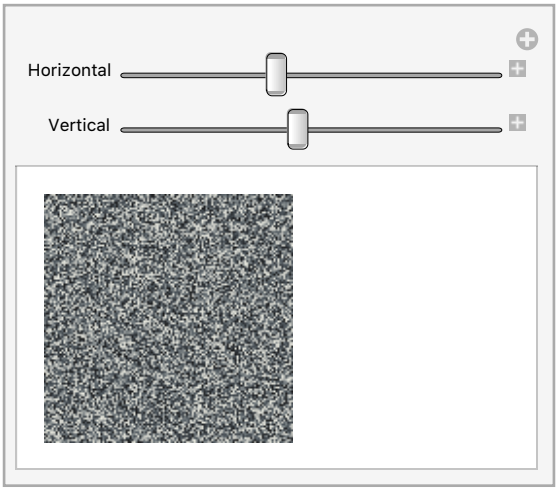
Before embarking on a detailed study of theories of motion measurement, let's list some of the uses of this information. What is motion information good for?

- change detection - "first alert", but not necessarily directional
- image or gaze stabilization - motion measurement in order to reduce its effects on the sensed image
- segmentation--breaking camouflage
- object tracking
- three-dimensional structure-from-motion, object recognition
- relationship of the viewer to the environment
    - -- time to contact
    - -- direction of heading
    - -- 3-D layout

An illustration of change detection and segmentation

In[318]:=
```
nsize = 128; subnsize = 32;
noise = Table[RandomReal[], {nsize}, {nsize}];
subnoise = Table[RandomReal[], {subnsize}, {subnsize}];
Manipulate[
 dnoise = noise;
 dnoise[[x1 ;; x1 + subnsize - 1, y1 ;; y1 + subnsize - 1]] = subnoise;
 ArrayPlot[dnoise, PlotRange → {0, 1}, PixelConstrained → {1, 1}],
 {{y1, 20, "Horizontal"}, 1, nsize - subnsize, 1},
 {{x1, 20, "Vertical"}, 1, nsize - subnsize, 1}]
```

Out[321]=



- A degenerate case of motion detection is pure change. Make a demo in which the small square subpatch above is simply replaced by a new random subpatch.

Keeping in mind these goals provides important constraints on our consideration of the models of early visual motion measurements. One distinction that we mentioned in the context of shape from shading is whether we need to estimate a dense field or a sparse one. A sparse set of velocity estimates would be important for efficiently describing the trajectories of objects--i.e. at a given instant, one “summary” velocity vector for each object could be useful for deciding the chance of collision. On the other hand, a dense velocity field could be useful for segmenting an object from a background to determine its shape.

Let’s first look at how a change in image luminance data is locally related to the velocity of a point in the scene.

Models of early motion measurement

The problem of estimating motion flow is complicated by a number of factors, one of which is that we don't have the motion flow directly available at the image. What we do have, are estimates of intensity change in time at each location, which we will call optic flow. Let's set this lecture in context and outline several historical approaches to motion measurement. As will become clearer later, there are theoretical connections between the various approaches.

Correlation models

Correlation-like models use image correlations to find spatial shifts that can account for image intensity shifts over time. The work dates to the pioneering results of Reichardt and colleagues in Germany on the fly and beetle visual systems (Hassenstein, B., & Reichardt, W., 1956). Later, mechanisms of directional selectivity were studied in vertebrate retina of the rabbit (Barlow, H. B., & Levick, R. W. , 1965). You've already learned enough about how to correlate two images to anticipate how such models might be constructed to find how far to shift say the first image to maximize the correlation. The amount of shift divided by the time between image frames determines the velocity.

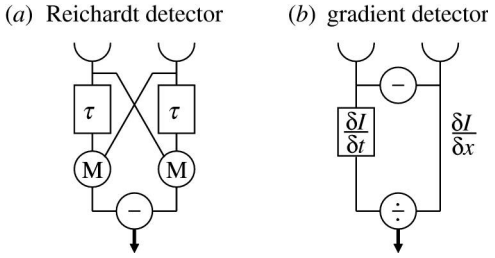
Some phenomena of human motion perception can be explained by modified versions of these early correlation or correlation-like models (Van Santen, J. P. H., & Sperling, G. , 1985; Adelson, E. H., & Bergen, J. R. , 1985). The possible synaptic basis of early motion selectivity has a long history (e.g. Koch, C., Torre, V., & Poggio, T., 1986; He, S., & Masland, R. H. (1997), Barlow, H. (1996)).

See the Appendix below for a demonstration of correlation.

Gradient methods

Gradient models were developed in computer vision. Rather than being derived from an analysis of biological mechanisms, gradient models were developed from a computational analysis of the problem, in conjunction with the use of known mathematical tools for solving the resulting differential equations.

Below, we will introduce the computational problems of motion detection by looking first at gradient models. We will look at one Bayesian model that can account for a large number of human perceptual results on motion integration. We will also see how gradient models are related to biologically plausible receptive field properties and mechanisms.



The above figure from Borst (2007) illustrates the difference between correlation (also "Reichardt" or Hassenstein-Reichardt" detector) and gradient detectors. ("M" stands for multiplication.) For a recent discussion in the context of the fly visual system, see: Borst, A. (2007). Correlation versus gradient type motion detectors: the pros and cons. Philos Trans R Soc Lond B Biol Sci, 362(1479), 369-374. (pdf)

Feature tracking

Feature tracking. Both correlation and gradient models are low-level in that they are based on local image intensity changes. One could imagine a different kind of system that locates features (more complicated than "pixel intensity", and conceivably quite complex, like an "apple") at time t, find that feature again a small time interval later, and from there computes the motion vector.

There is psychological evidence for both low-level (e.g. correlation) and feature tracking models. But the "features" may be defined rather generally, e.g. anywhere from luminance of a point, to

"corner", to "apple". A fair amount of psychophysics went into arguing for two separate motion systems in human vision (Braddick, O. J., 1974), but later work has shown that the distinction is not so clear.

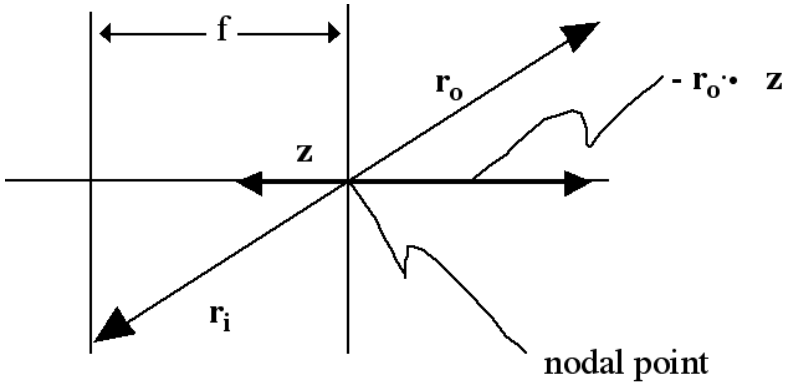
Object tracking is an important problem in computer vision, where discounting view, illumination and occlusion has been taken into account (e.g. Isard & Blake, 1998; Hager & Belhumeur, 1996).

Orientation in space-time

In the next lecture we will see how models of biological motion detectors can be interpreted as edge or "orientation detectors" but where the "orientation" is in *space-time*, rather than space-space. We'll also see how they relate to gradient models.

Geometry & kinematics

We derive a generative model describing how the velocity of a point in 3D gets mapped to the velocity of its image point. The nodal point of a lens is where a light ray passes undeviated. Let $\mathbf{r_o}$ be a vector representing a point (x,y,z) on an object in the 3D scene. Let $\mathbf{r_i}$ be the vector representing its corresponding point in the projected image (e.g. on to the retina). **z** is a vector pointing along the axis between the nodal point and image origin (e.g. fovea). f is the distance from the nodal point to the image plane.



Using similar triangles we have:

$$\mathbf{r_i} = \frac{f}{\mathbf{r_o} \cdot \hat{\mathbf{z}}} \mathbf{r_o}$$

Object velocity by definition is:

$$\frac{d\,\mathbf{r_o}}{dt} = \mathbf{v_o}$$

And the projected image velocity, also by definition, is:

$$\frac{d\,\mathbf{r_i}}{dt} = \mathbf{v_i}$$

With some vector calculus manipulations using identities, one can arrive at:

$$\mathbf{v_i} = \frac{(\mathbf{r_o} \times \mathbf{v_o}) \times \mathbf{z}}{(\mathbf{r_o} \cdot \mathbf{z})^2}$$

(e.g. See Horn's book Robot Vision)

It is important to underscore that this is just a statement about the kinematics of projection geometry. Light intensity does not enter in. The equation tells us how to calculate the *motion field* from a description of the 3D coordinates and velocities. If we had a collection of vectors $\{\boldsymbol{v}_i^j\}$, where j indexes projected vectors across the image, we would have the basis for inferring $\{\boldsymbol{v}_o^j\}$.

But we don't even have access to these projected image velocities of 3D world points. The available image data are intensity changes that are not directly related to geometry, i.e. to $\boldsymbol{v}_i$. How can we estimate the motion flow, $\{\boldsymbol{v}_i^j\}$, from retinal or image intensity changes? This is yet another example of a scene-from-image problem, of inverse inference.

Geometry & photometry: Motion field vs. Optic flow

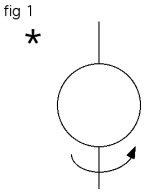
Image data involves measurements of light intensity over space and time, so it is not immediately obvious how to relate intensity change to geometry changes either in the world or on the retina. We've considered just the geometry to see the relationship between velocity in the world and velocity on the retina--ie. how we can calculate the projected velocity in the image from a knowledge of the 3D velocity of a point.

Let's consider the inverse problem in which we would like to go from the retinal flow of brightness (optic flow) to an estimate of the motion field. To emphasize the difference between optic and motion flow, consider a rotating sphere--an illustration adopted from Horn.

Assuming a surface of uniform material, the rotating sphere would not appear to rotate (optic flow would not change), but motion flow would change. However, if the light source is rotated around the stationary sphere, the sphere appears to rotate (optic flow), while the motion field is actually constant. Because the visual system relies on optic flow information, we must understand how to infer motion flow from optic flow.

Non-zero motion field: rotating object, fixed light source

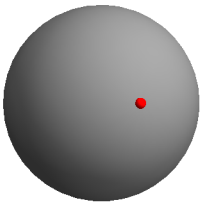
fig 1



Interactively rotate the gray lambertian shaded sphere in the *Mathematica* demo:

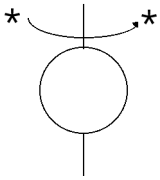
In[346]:=
```
Graphics3D[{{GrayLevel[.5], Sphere[{0, 0, 0}]}, {Red, Sphere[{0, 0, 1}, .05]}},
 Boxed → False, Lighting → "Neutral", ImageSize → Tiny, RotationAction → "Clip"]
```

Out[346]=



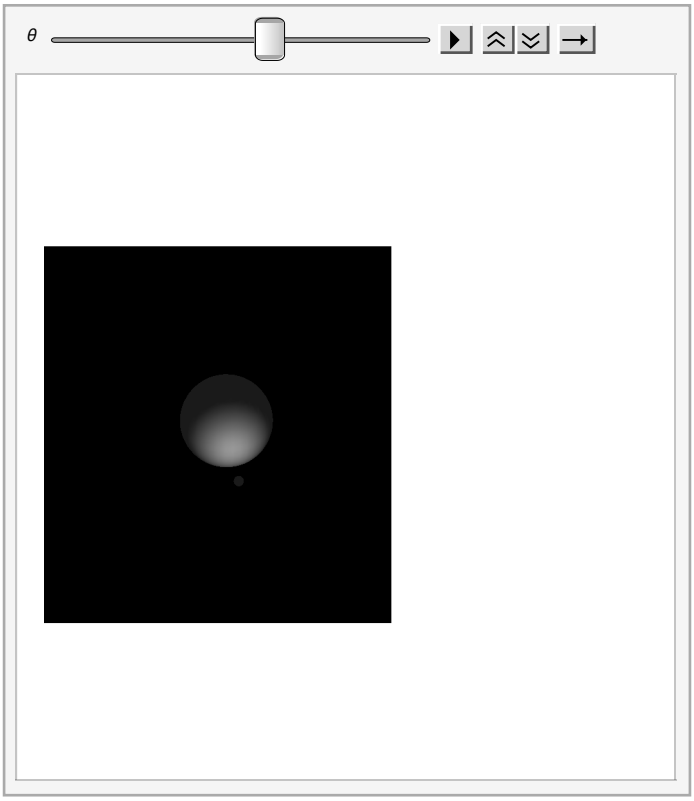
Non-zero optic flow: fixed object, moving light source

fig 2



Now rotate the light source about the lambertian shaded sphere:

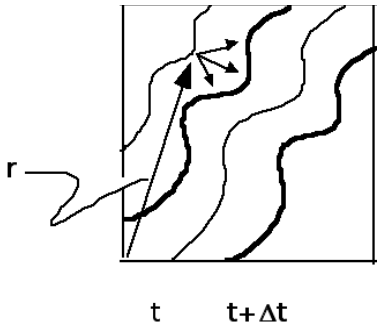
```
p1[θ_] := RotationTransform[θ, {0, 0, 1}][{0, 2, 0}];
Animate[Graphics3D[{Sphere[], Sphere[p1[θ], .1]},
  Lighting → {{"Ambient", GrayLevel[.1]}, {"Point", GrayLevel[.5], p1[θ]}},
  PlotRange → 2.5, Background → Black, Boxed → False, ImageSize → Small],
 {θ, -Pi - Pi / 4, -Pi / 2}, SaveDefinitions → True, AnimationRunning → False]
```



- Imagine moving your head while viewing the hood of a shiny car. Which image features are “stuck” to the hood, and which ones slide?

The gradient constraint & the aperture problem

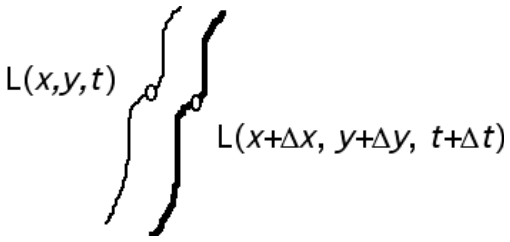
The “optic flow equation”: photometric generative model → gradient constraint



If one could pick a point, r in the image, on a curve at time t, and track it to its new location at time t + Δt, the motion flow would be given by dr/dt. In order to track this point, we need to specify some

"feature". The simplest is the luminance at point r. Although this could be considered a primitive form of feature tracking, tracking the luminance in the image has fundamental ambiguities. The luminance corresponding to a point out in the scene is not guaranteed to remain constant with time in the image. And even if the points do maintain a constant intensity in time, there may be many points at time t+dt that match the intensity of the feature point at time t. However, it seems reasonable to assume that luminance doesn't change much over small displacements occurring over short times. This is a kind of “luminance constancy” assumption.

So lets' suppose that intensity doesn't change over small distances Δx, Δy and short time intervals, Δt, then the total derivative of luminance would be zero:



$$\Delta L = \frac{\partial L}{\partial x}\Delta x + \frac{\partial L}{\partial y}\Delta y + \frac{\partial L}{\partial t}dt = 0$$

$$\frac{-\partial L}{\partial t} = \frac{\partial L}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial L}{\partial y}\frac{\Delta y}{\Delta t}$$

$$\frac{-\partial L}{\partial t} = \nabla \mathbf{L} \cdot \boldsymbol{\mu} \qquad \boldsymbol{\mu} = (u, v)$$

$\boldsymbol{\mu}$ is a quantitative expression for the optic flow at a point, but it is only an estimate of the motion field. This is sometimes called the “optic flow equation”.

Note that even if luminance did remain constant, (not a bad assumption for small Δt), we still have the problem of matching which of possibly many points of the *same* luminance in the second frame belong to our chosen feature in the first frame. This an example of the *correspondence problem*, which also crops up in stereo vision (and in other areas of science).

- How would you approximate $\frac{\partial L}{\partial t}$, $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial y}$ given two image frames: $L_{t+\Delta t}(x,y)$ and $L_t(x,y)$?
- An empirical question is whether human vision effectively behaves as if it assumes lightness constancy or luminance constancy when tracking points. How could you test that?

Aperture problem

Let us look at what the above (photometric) generative equation buys us. What is measurable? Our measurables are: $\frac{\partial L}{\partial t}$, $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial y}$, but u and v are unknown. So we have two unknowns for each equation at a point (as with the shape from shading problem, where at each point we wanted to estimate surface orientation, e.g. slant and tilt, but had only one equation for each point). The gradient flow equation provides a constraint on the solution to the problem of calculating optic flow. But it does not specify a unique solution. Again we see an example of a vision problem that is mathematically underconstrained

or “ill-posed”--even apart from any motion field ambiguity -- there are too many optic flow solutions that satisfy the data.

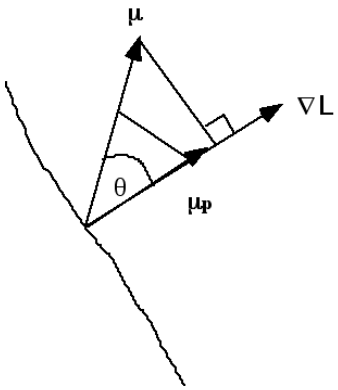
Well, what do our measurements tell us? Note that **∇L** points in the direction of maximum intensity change. (Recall this property of the vector gradient before in the context of edge detection with gradients in x and y.) We can see this by projecting **∇L** onto a little vector d**r** of fixed length that we rotate around so that it makes various angles θ with **∇L**. The dot product

$$\nabla \mathbf{L} \bullet d\mathbf{r} = |\nabla \mathbf{L}||d\mathbf{r}|\cos\theta$$

is biggest when cosθ = 1, or θ = 0 (e.g. at an edge/contour).

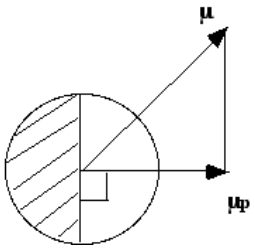
So, we can measure the component of optic flow perpendicular to the contour:

$$\frac{-\partial L}{\partial t} = |\nabla \mathbf{L}| \; |\boldsymbol{\mu}| \cos\theta \qquad |\boldsymbol{\mu}_p| = |\boldsymbol{\mu}|\cos\theta \qquad |\boldsymbol{\mu}_p| = \frac{\frac{-\partial L}{\partial t}}{|\nabla \mathbf{L}|}$$



This formula makes explicit what we get, and what we lack in our effort to find a unique solution to the optic flow. We can get the velocity components in the direction of the gradient at each point, but not perpendicular to it. This ambiguity is known as the aperture problem. This problem is faced by a measurement of motion that is restricted to an edge. For example, the motion and orientation selective cells in V1, are limited by the aperture problem because of their small, limited receptive field sizes.

The true motion of contour is not known from a local measurement. Only the component (μ_p) of the motion perpendicular to the contour is available



Aperture ambiguity: Demo of the aperture problem

In[347]:=
```
gdiamond[x_, θ_, s_] :=
 {EdgeForm[{Thickness[.01], Blue}], FaceForm[], Translate[Scale[Rotate[
     Rectangle[{-diasize, -diasize}, {diasize, diasize}], θ], {s, 1}], {x, 0}]}

gbar[x_, θ_] := {EdgeForm[{Thick, Red}], FaceForm[Red],
  Translate[Rotate[Rectangle[{0, 0}, {bwidth, 1}], θ], {x, 0}]}
```

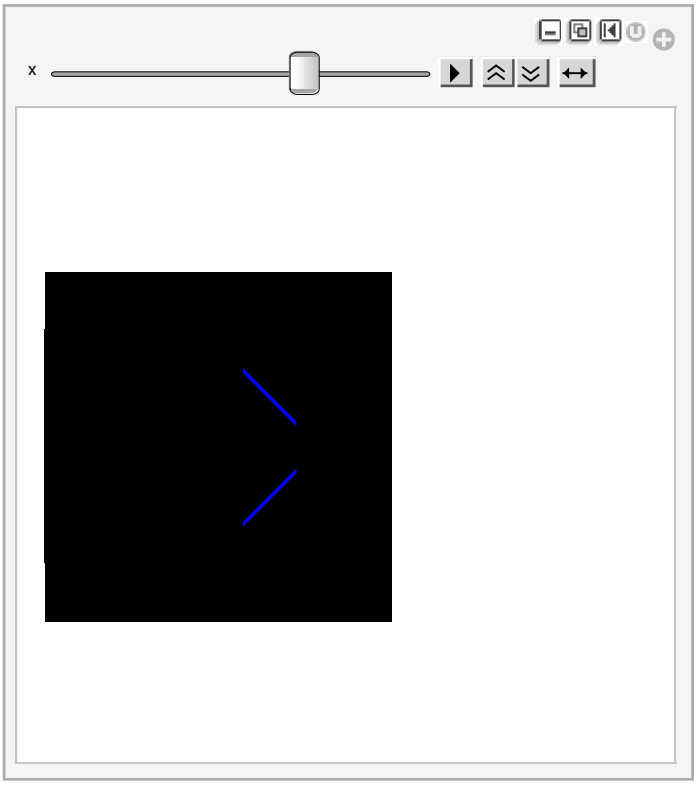
In[349]:=
```
bwidth = 0.5; bspace = .25;
bdisp = .5; diasize = .6; bcolor = Black; bcolor2 = Black;
backcolor = bcolor;

gbar3[x_, θ_, s_] := {EdgeForm[{Thick, bcolor2}], FaceForm[bcolor],
  Translate[Scale[Rotate[Rectangle[{bdisp, -1}, {bdisp + bwidth, 1}], θ], {s, 1}],
   {.2, 0}], EdgeForm[{Thick, bcolor}], FaceForm[bcolor],
  Translate[Scale[Rotate[Rectangle[{-bwidth - bspace + bdisp, -1},
      {bdisp - bwidth - bspace + bwidth, 1}], θ], {s, 1}], {x, 0}],
  EdgeForm[{Thick, bcolor2}], FaceForm[bcolor],
  Translate[Scale[Rotate[Rectangle[{bdisp + 2 * (-bspace - bwidth), -1},
      {2 * (-bspace - bwidth) + bwidth + bdisp, 1}], θ], {4, 1}], {x, 0}]}
```

Translation

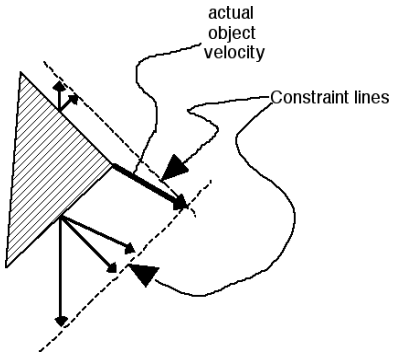
In[353]:=
```
Animate[Graphics[{gdiamond[x * .07, Pi / 4, 1.], gbar3[-.05, 0, 1.0]},
  PlotRange → {{-1.5, 1.5}, {-1.5, 1.5}}, Background → backcolor, ImageSize → Small],
 {x, -1, 1, .1}, AnimationRunning → True, DisplayAllSteps → True,
 AppearanceElements → All, AnimationDirection → ForwardBackward, AnimationRate → 2]
```

Out[353]=



Constraint lines: A sparse velocity solution to the aperture problem

What is a solution to this problem? We need *a priori* constraints, i.e. constraints that are not explicit in the data, but are based on assumptions of what the class of estimates are typically like. If for example, the local measurements come from rigid motion of an object in a plane, the following strategy could be used to estimate the actual velocity



This solution would give us the right motion vector for rigid motion in the plane. See Movshon et al (1986) possible physiological solution to this problem. One could ask whether there are more general constraints that are not restricted to assuming rigid planar motion.

Smoothness constraint to solve for a dense optic flow field

Combining prior smoothness constraint with generative constraint

An early method due to Horn & Schunck. Dense motion field is "area-based" rather than "contour-based". This means that optic flow is computed over each pixel in the field, rather than by tracking contours.

One way to solve the problem of being underconstrained, is to add some fairly general constraints on the class of allowable optic flow fields. In the spirit of a prior for "smooth shape" that we saw earlier, Horn and Schunck's idea was to assume that flow fields are smooth almost everywhere. This constraint is *generic*. That is, the hope is that it will more or less work regardless of the class of scenes causing the flow. In contrast, if one knew ahead of time that the flow would always be caused by a rotating cube illuminated from above, one could use this knowledge to constrain the optic flow solution. This would be a domain specific constraint.

What is a suitable measure of smoothness? Both Laplacians and gradients of the velocity components at each pixel location provide measures:

$$(\nabla^2 u)^2, (\nabla^2 \mathrm{v})^2 \qquad (|\nabla u|, |\nabla \mathrm{v}|, \text{ etc. also possible})$$

The intuition is that that optic flow vector estimates that are nearby in space should be encouraged to point in similar directions and have similar magnitudes. Imagine just two vectors whose bases are almost touching. The difference between the two vectors is approximately represented by the vector connecting their arrows. This vector should be encouraged to be short.

The constraints are that the flow estimates should be consistent with that expected from required the total derivative of the image data to be close to zero:

$$\mathrm{E_i}[u, v : x, y] = (\frac{\partial \mathrm{L}}{\partial t} + \nabla \mathbf{L} \cdot \boldsymbol{\mu})^2$$

In the language of Bayesian inference, E_i is a likelihood constraint, because it is based on the data.

The second constraint is an *a priori* constraint that the flow estimates be locally smooth:

$$\mathrm{E_s}[u, v : x, y] = (\frac{\partial^2 u}{\partial x^2})^2 + (\frac{\partial^2 u}{\partial y^2})^2 + (\frac{\partial^2 \mathrm{v}}{\partial x^2})^2 + (\frac{\partial^2 \mathrm{v}}{\partial y^2})^2$$

E_s is the smoothness cost function. "a priori" means that we were assuming this about our solution independent of the data, as in a Bayesian prior. We saw this distinction between a priori and a posteriori constraints in shape from shading.

The above two local constraints can be used to construct a global cost function, or more precisely a “functional” (if the optic flow is assumed to be a continuous function):

$$e[u(x, y), v(x, y)] = \int\int (\mathrm{E_i}[u, v : x, y] + \lambda \mathrm{E_s}[u, v : x, y]) dx dy$$

The goal is to find the optic flow, u(x,y) and v(x,y), that minimizes this functional. The analagous cost function in the Ikeuchi and Horn shape from shading cost function was:

$$e[p(x,y),\ q(x,y)] = \iint (L(x,y) - L_R(p,q))^2 + \lambda((\nabla^2 p)^2 + (\nabla^2 q)^2)\, dx\, dy$$

How can we find a solution? One way to do this is to use the method of gradient descent. Here we are going to be taking an entirely different gradient--not the gradient of intensity we used to construct the generative constraint above, but the gradient of the cost function e.

Visualization of gradient descent in 2D

(Note: "gradient" here is on a cost function--a totally different function!)

Gradient descent technique

Gradient descent is an important numerical technique that crops up in lots of applications including vision.

Cost function or "Energy potential" surface

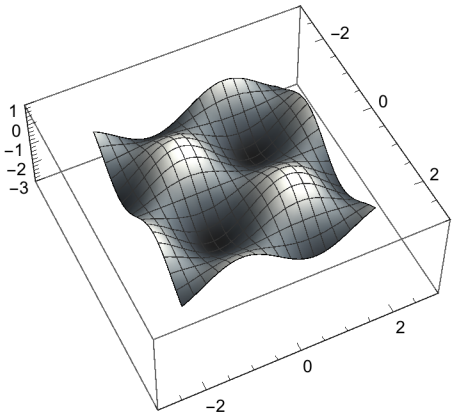
Also called error or objective function. If continuous it is called a "functional", in that the energy depends on functions rather than scalar or vector values. If optic flow is a continuous vector field, it is a function, and you'd like to find exactly which optic flow function gives the smallest cost function. Whatever it is called, the idea is that finding a solution requires finding the argument values that minimize the energy (under certain conditions, this is equivalent to finding the arguments that maximize the posterior probability).

Let's look at a small dimensional example. Here is the energy surface e = z(u,v):

In[354]:=
```
z[u_,v_]:= Sin[2 u] Sin[2 v];
```

In[358]:=
```
Plot3D[z[u,v], {u,-2,2}, {v,-2,2}, ViewPoint->{2*1.3, -2*2.4, 2},
PlotRange->{{-3,3},{-3,3},{-3,1}}, Axes->True,Boxed->True,ImageSize->Small]
```

Out[358]=



By eye, we can see roughly where the minima sit--there are two minima where z(u,v) = -1, i.e. at {Pi/4, -Pi/4} and {-Pi/4, Pi/4}. But what if z is a surface z = f(x_1, x_2, x_3,...,x_n), that is embedded in a much higher

dimensional space of n dimensions. How then can we find the minimum when we can't just look at a plot and guess? Gradient descent is a method that will work in the 2D case, but can also be applied to much higher dimension like finding the optic flow (u's and v's) that minimze the above error function e.

Gradient field of the cost function

We've seen the gradient of a function used in two earlier contexts. First, we briefly noted that the gradient of an image produces a vector field where the vectors point in the direction of maximum intensity change (i.e. tend to be perpendicular to edges). Second we noted that the gradient of a depth function provided a surface normal representation that collectively provide a description of shape. Now we are going to use the gradient from vector calculus in a third way. Because the gradient points in the direction of maximum change, if we take the gradient of our cost function e= z(x,y)

In[359]:=
```
gradZ[u_,v_] := Evaluate[{D[z[u,v],u],D[z[u,v],v]}];
```

we get,

In[360]:=
```
gradZ[u,v]
```

Out[360]=
```
{2 Cos[2 u] Sin[2 v], 2 Cos[2 v] Sin[2 u]}
```

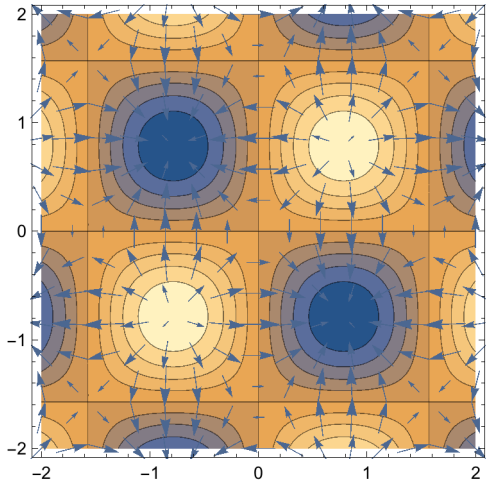
If we put a minus sign in front, the gradient vector points in the direction of greatest decrease, i.e. points in the direction we want to go to minimize a cost function.
Let's visualize the cost landscape by combining **VectorPlot[ ]** and **ContourPlot[]**:

In[361]:=
```
gvf = VectorPlot[-gradZ[u,v],{u,-2,2}, {v,-2,2},ColorFunction->Hue[0.99]];
```

This shows a set of vectors each of which always point towards the direction of local steepest descent on the surface.
Note that the arrows represent the vector field for the gradient of the cost function, *not* the optic flow field. The optic flow field in this case has only two dimensions (rather than the hundreds or thousands) would correspond to the two axes of the plot.

In[362]:=
```
gc = ContourPlot[z[u, v], {u, -2, 2}, {v, -2, 2}];
Show[gc, gvf, ImageSize → Small]
```

Out[363]=



So each vector says which direction to take a step if you are descending. And its length tells you how high a step could be. If you want to go *up* the landscape, just go in the direction given by gradZ[u,v], and you do gradient ascent. The negative just turns the arrows around to put in the opposite direction specifying g*radient descent*.

Gradient descent

The idea behind gradient descent is this. Suppose you are plopped down randomly somewhere on a smooth cost function landscape. If you begin to take steps in the direction given by the negative gradient, you will go down...and this is better than wondering around aimlessly. If there is a low spot, then you can check that you are at a minimum, because the gradient there is zero. Gradient descent isn't perfect. If your steps are too big, you could step right over the minimum. You could end up in different minima depending on your starting position. Further, the minimum you end up in could be a local rather than global minimum. Sometimes this is OK, but it isn't the best for optic flow estimation where we normally want to find the position on the cost landscape (i.e. the optic flow) that minimizes the overall error or cost the best--i.e. the global minimum of the cost function.

Let **r = {u,v}** be a location on the cost function landscape. We are going to start off at **r0 = {u0,v0}**, then calculate the u and v directions of the step to take, **{Δu,Δv} = gradZ[r[[1]],r[[2]]]**, scale this step size by **dt (= α)**, and then set the update equation:

newr = r - dt gradZ[r]

Note that this is discretized form of the equation:

$$\frac{\partial \vec{r}}{\partial t} = -\nabla e$$

where -∇e = -gradZ. Note the time here refers to time steps, ∂t, in the algorithm.

Mathematica demonstration of gradient descent in 2 dimensions

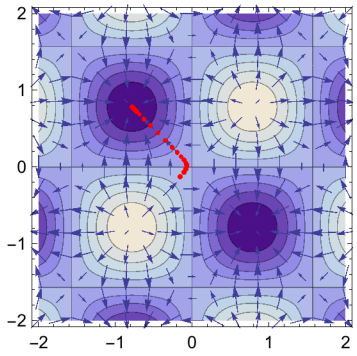
Let's define the update equation in *Mathematica*:

```
u0 = Random[Real,2]-1; v0 = Random[Real,2]-1;
dt = 0.1;
newr[r_] := r - dt*gradZ[r[[1]],r[[2]]];
```

To do the iteration, we use the built-in function **NestList[] .** `NestList[`f`,` $expr$`,` n`]` gives a list of the results of applying f to $expr$ 0 through n times.

```
result = NestList[newr,{u0,v0},200];

gresults = ListPlot[result,
    Joined->False,AxesOrigin->{0,0},
    PlotRange->{{-2,2},{-2,2}},FrameLabel->{"u","v"},
    Frame->True, AspectRatio->1,RotateLabel->False,PlotLabel->"State Space",
    Ticks->None, PlotStyle->{RGBColor[1,0,0], PointSize[0.015]}];
    Show[gc, gvf,gresults,ImageSize->Small]
```



What happens if you start at **r0 = {0,0}**?

For any kind of iterative solution method, we should ask whether the method converges, and if so, whether it converges to the smallest error. A bumpy cost function surface could easily lead our descent into a local minimum--and usually we are interested in the global minimum.

It can be shown that the above energy function for optic flow is convex with a single minimum (unlike our toy 2-dimensional example which has more than one local minimum), and thus gradient descent should provide a unique estimate of the optic flow. This algorithm works fairly well, especially if motion flow at a contour boundary is available to additionally constrain the interior values. One problem, is that without this external constraint, the estimates smooth over discontinuities. A second problem is that a good solution may require many iterations--and is therefore too slow. A lot of research has gone into devising efficient optimization methods, and in actual practice, you would probably want to find some expertise in numerical methods.

Gradient descent solution for optic flow

Let's see how to apply gradient descent to the optic flow problem. One reason that derivatives are chosen for the smoothness constraint is that there exist standard mathematical tools to solve these minimization problems. (See p. 284 in Horn's Robot Vision book for a discussion of the calculus of variations--some mathematics that has been used in many areas of mathematical physics ranging from classical mechanics (least action), to optics (least time), to quantum mechanics.)

Suppose we discretize the flow field into two n x n pixel maps (uik, vik), and estimate the error or cost function at each pair of pixel sites. Now compute the gradient of the error function:

$$\nabla \mathbf{e} = \left(\frac{\partial e}{\partial u_{11}}, \frac{\partial e}{\partial u_{12}}, \ldots \frac{\partial e}{\partial u_{ke}}, \ldots \frac{\partial e}{\partial v_{11}}, \frac{\partial e}{\partial v_{12}}, \ldots \frac{\partial e}{\partial v_{ke}}\right)$$

We make an initial guess (perhaps at random) for the initial flow field:

$$u_{ik}^{0}, v_{ik}^{0}$$

Then iterate according to the following rule:

$$u_{ik}^{n+1} = u_{ik}^{n} - \alpha \left.\frac{\partial e}{\partial u_{ik}}\right|_{u_{ik}=u_{ik}^{n}}$$

$$v_{ik}^{n+1} = v_{ik}^{n} - \alpha \left.\frac{\partial e}{\partial v_{ik}}\right|_{v_{ik}=v_{ik}^{n}}$$

over all points, (i,k). That is, at each iteration, we move the values of the flow field, (uik, vik), in the opposite direction of the gradient of the error function. That is, we always take steps down the error function surface. That way, we eventually, we come to a minimum. a has to be chosen small enough that we don't hop over minima, but big enough to not take forever to find the minimum.

Applicability to human motion perception?

Are the representation, constraints, and algorithm that we've assumed above a good model of human motion perception?

The gradient descent aspect of the algorithm is probably wrong. Iterations may east up valuable time. Although not strictly necessary from a mathematical point of view, both the area based method of Horn and Schunck, and Hildreth's contour method specified some kind of iterative method for solution. How appropriate is the gradient descent algorithm from a biological point of view?

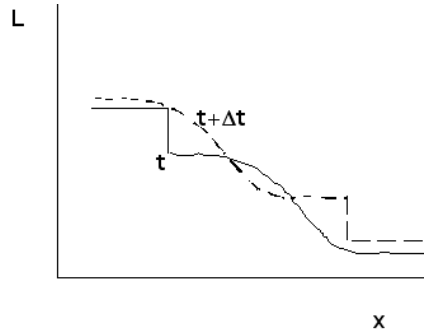
Gradient descent as well as other methods can be implemented in biologically plausible neural network algorithms--both linear, and non-linear (e.g. Cohen, M. A., & Grossberg, S. (1983). Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks. IEEE Transactions SMC-13, , 815-826.; Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings National Academy of Science, USA, 81, 3088-3092.). However, it seems unlikely that biological systems would use a method that could require many iterations--neural elements just aren't fast enough. By the time the flow is computed, we would have been either devoured by the jaguar, or more likely in our society, hit by one on the freeway.

For certain minimization problems, one can show that a feedforward one-shot solution exists. For example, there is straightforward for error or cost functions that are quadratic. where a linear matrix operation can solve the problem in a single shot.

The Lucas-Kanade method, illustrated above, solves the ill-posed problem in a fashion that avoids explicit priors and can be quite fast.

But there are other reasons why the answer seems to be "no". The representation of the input may not be right. Human observers sometimes give more weight to contour movement than to intensity flow. Human perception of the sequence illustrated below differs from "area-based" models of optic flow such as the above Horn and Schunck algorithm. The two curves below would give a maximum correlation at zero--hence zero predicted velocity. Human observers see the contour move from left to right--because the contours are stronger features than the gray-levels. However we will see in Adelson's missing fundamental illusion (next lecture) that the story is not as simple as a mere "tracking of edges" -- and we will return to spatial frequency channels to account for the human visual system's motion

measurements.



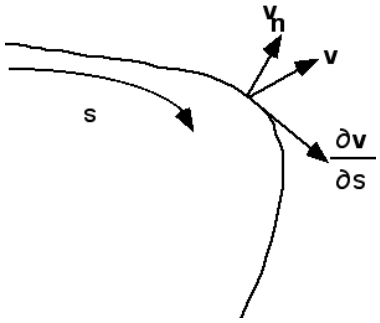
Hildreth's contour model

Dense on a contour, sparse elsewhere.

Some sort of smoothness constraint seems to be important, but it could operate on the contours, rather than intensities. Further, the smoothness constraint has to be generalized (see Yuille, A., & Grzywacz, N., 1988 for a discussion of smoothness constraints). In a beautifully written Ph.D. thesis, Ellen Hildreth suggested a contour-based scheme that is consistent with a number of observations from human perception. Her error term was:

$$e = \sum_{\text{over contour}} (\mathbf{v} \cdot \hat{\mathbf{n}} - |\mathbf{v_n}|)^2 + \lambda \left| \frac{\partial \mathbf{v}}{\partial s} \right|^2$$

where **v** (s) is the velocity at position s of the contour. We want to estimate **v**(s), but the only data available is the magnitude of the component of **v**(s) perpendicular to the contour, call it $\boldsymbol{v}_n$(s). So if possible, we want to choose **v** so that v•n = $\boldsymbol{v}_n$(s), as shown in the figure:



To make the problem well-posed, smoothness is imposed by requiring that a measure of the magnitude of the rate of change of the contour velocity along the contour, e.g. $|\partial v/\partial s|^2$ be small.

Lucas-Kanade method

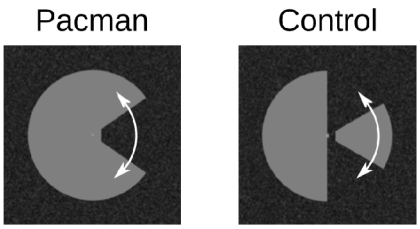
A classic method in computer vision for computing optic flow is the Lucas-Kanade method. It addresses the problem of too few equations per unknown by assuming that one can track an nxn pixel region, rather than just 1 pixel. So for example, if one assumes a 3x3 pixel region in which all pixels have the same velocity vector, optic flow can be estimated by linear regression.

One strategy in applying the algorithm is to first pick out a sparse set of features to track, and then solve the regression equations for just these few points.

The class web page has an IPython notebook illustrating the Lucas-Kanade method.

Some further problems: uniform regions, global motion, spatial scale.

The problem of motion in uniform regions, away from features.



Akin, B., Ozdem, C., Eroglu, S., Keskin, D. T., Fang, F., Doerschner, K., et al. (2014). Attention modulates neuronal correlates of interhemispheric integration and global motion perception. Journal of Vision, 14(12), 30–30. http://doi.org/10.1167/14.12.30

The problem of global motion -- how to summarize the motion path of a large object as a single point, e.g. the trajectory of the “center of mass”?

The problem of spatial scale -- how to deal with features that are not spatially close between nearby frames? One answer: apply an optic flow algorithm across spatial scale represention in an image pyramid. See the Lucas-Kanade demo.

Next time

We have studied some of the computational problems of motion measurement. In the next lecture, we will look at how a Bayesian formulation with the right priors can be used to model a variety of human motion direction perception results. And we will look at neural systems solutions to the problem of motion measurement.

Neither the area-based nor the contour-based algorithms we've seen can account for the range of human motion phenomena or psychophysical data that we now have. We will look at a recent Bayesian model that uses the area-based gradient constraint, combines it with a particular set of prior assumptions to compute a somewhat dense optic flow field that accounts for a wide range of human motion data.

Appendices

Searching for spatial correlations in two frames

In[364]:=
```
argmax2[x_] := Flatten[Position[x, Max[x]]]; (*coords of max in matrix*)
```

In[365]:=
```
size = 32; x0 = 4; y0 = 4; pw = 24; xoffset = 3;
A1 = Table[Random[], {size}, {size}]; A2 = A1;
(*A2=Table[Random[],{size},{size}];*)
A2[[Range[y0, y0 + pw], Range[x0, x0 + pw]]] =
  A1[[Range[y0, y0 + pw], Range[x0 - xoffset, x0 + pw - xoffset]]];

ga1 = ArrayPlot[A1, Mesh → False];
ga2 = ArrayPlot[A2, Mesh → False];
ListAnimate[{ga1, ga2}]
```

Out[370]=

In[371]:=
```
argmax2[ListCorrelate[A1, A1, {size / 2, size / 2}]]
```

Out[371]= {16, 16}

In[372]:=
```
argmax2[ListCorrelate[A1, A2, {size / 2, size / 2}]]
```

Out[372]= {16, 19}

How does correlation work if the patch is small?
Try having dynamic noise in the background, by uncommenting (*A2=Table[Random[],{size},{size}];*).

References

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal Energy Models for the Perception of Motion. Journal of the Optical Society of America, 2(2), 284-299.

Akin, B., Ozdem, C., Eroglu, S., Keskin, D. T., Fang, F., Doerschner, K., et al. (2014). Attention modulates neuronal correlates of interhemispheric integration and global motion perception. Journal of Vision, 14(12), 30–30. http://doi.org/10.1167/14.12.30

Barlow, H. B., & Levick, R. W. (1965). The Mechanism of Directional Selectivity in the Rabbit's Retina. Journal of Physiology, 173, 477-504.

Barlow, H. (1996). Intraneuronal information processing, directional selectivity and memory for spatio-temporal sequences. Network: Computation in NeuralSystems, 7, 251-259.

Braddick, O. J. (1974). A short-range process in apparent motion. Vision Research, 14, 519-527.

Borst, A. (2007). Correlation versus gradient type motion detectors: the pros and cons. Philos Trans R Soc Lond B Biol Sci, 362(1479), 369-374.

Hager GD, Belhumeur PN (1996) Real-Time Tracking of Image Regions with Changes in Geometry and Illumination. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p

403—410.
Hassenstein, B., & Reichardt, W. (1956). Functional Structure of a Mechanism of Perception of Optical Movement. Proceedings First International Congress on Cybernetics, 797-801.
He, S., & Masland, R. H. (1997). Retinal direction selectivity after targeted laser ablation of starburst amacrine cells. Nature, 389(6649), 378-82.
Hildreth, E. C. (1984). The Measurement of Visual Motion. Cambridge, MA: MIT Press.
Isard M, Blake A (1998) Condensation -- conditional density propagation for visual tracking. International Journal of Computer Vision 29:5--28.
Koch, C., Torre, V., & Poggio, T. (1986). Computations in the Vertebrate Retina: Motion Discrimination Gain Enhancement and Differentiation. Trends in Neuroscience, 9(5), 204-211.
Movshon JA, Adelson EH, Gizzi MS, Newsome WT (1986) The Analysis of Moving Visual Patterns. In: Pattern Recognition Mechanisms (Chagas C, Gattas R, Gross CG, eds), pp 117-151. Rome, Italy: Vatican Press.
Van Santen, J. P. H., & Sperling, G. (1985). Elaborated Reichardt Detectors. Journal of the Optical Society of America, 2(7), 300-321.
Yuille, A., & Grzywacz, N. (1988). A computational theory for the perception of coherent visual motion. Nature, 333, 71-74.