Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 26: Spatial Relations: scene structure and trajectories

Initialize

In[1]:=
```
Off[General::"spell1"];
SetOptions[ArrayPlot, ColorFunction → "GrayTones",
  DataReversed → True, Frame → False, AspectRatio → Automatic,
  Mesh → False, PixelConstrained → True, ImageSize → Small];
```

Outline

Last time

Recognition: Bidirectional architectures

Today

Earlier, we developed some photometric and geometric models applied to intrinsic shape (e.g. lambertian model, and local shape in terms of gradient space, and slant and tilt). Here we develop tools applicable to the analysis and modeling of large scale spatial structure and trajectories. These tools include modeling: rotation, size, translation and perspective. And in the second part, the Kalman filter for object tracking.

Computational theory for estimating relative depth, eye (camera) motion

---

Space: Where are objects? Where is the viewer?

Recall distinctions: Between vs. within object geometry.

Where are objects? A review of cues

Absolute depth

Spatial, timing relationships, distance of objects or scene feature points from the observer.

"Physiological cues": Binocular convergence--information about the distance between the eyes and the angle converged by the eyes. Crude, but constraining. Errors might be expected to be proportional to reciprocal distance. Closely related to accommodative (focusing by lens) requirements.

"Pictorial cue"--familiar size. E.g. human adults have an average size and range.

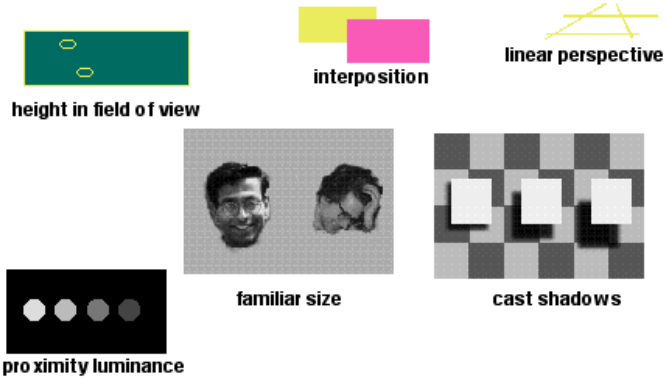
Relative depth

Distance between objects or object feature points. Important for scene layout, planning.

Processes include: Stereopsis (binocular parallax) and motion parallax. Motion parallax is discussed below.

Also information having to do with the "pictorial" cues: occlusion, transparency, perspective, proximity luminance, focus blur, also familiar size & "assumed common physical size", "height in picture plane", cast shadows, texture & texture gradients for large-scale depth & depth gradients

Examples of pictorial information for depth



Cooperative computation & cue integration

...over a dozen cues to depth. Theories of integration (e.g. stereo + cast shadows). Theories of cooperativity (e.g. motion parallax <=> transparency).
Vision for spatial layout of objects, navigation, heading and for reach

Tracking

Methods such as Kalman filtering (and its generalizations) can be used to combine prior assumptions regarding trajectory models of a single point/object with online updating of noisy measurements over long periods of time. These are useful for both dynamic depth relative to viewer and to other objects.

Where am I? Where am I headed?

We'll see that optic flow can't specify where the viewer is in absolute terms, but can be used to compute the relative depth relationships between objects, heading direction, and time to contact.

Calculating structure from motion and heading from the motion field

Estimation of relative depth and viewer/camera motion: Introduction

Earlier we saw:

1) how local motion measurements constrain estimates of optic flow, and thus the motion field.

2) how a priori slowness and smoothness contraints constrain dense and sparse estimates of the flow field (e.g. Weiss et al.).

How can we use an estimate of the motion field to estimate useful information for navigation--such as relative depth, observer motion, and time to collision? And all in an instant!

Goals

Estimate relative depth of points in a scene, the viewer's motion from the motion field, and the viewer's time-to-contact with a surface.
Visual computations provide an "understanding" of the environment from the moving images on our retinas.

There are approaches to structure from motion that are not based directly on a dense motion field, but rather based on a sequence of images in which a discrete set of corresponding points have been identified (Ullman, S., 1979; Dickmanns). A major challenge is to track the corresponding points.

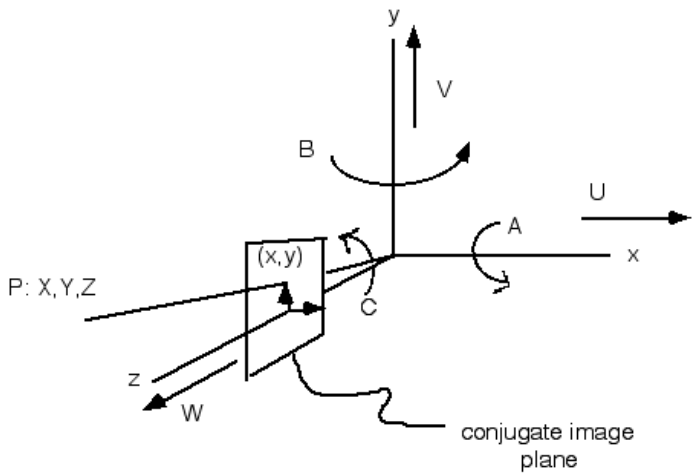
Alternatively, suppose we have estimated the optic flow at time t, and assume it is a good estimate of the motion field--what can we do with it? Imagine the observer is flying through the environment. The flow field should be rich with information regarding direction of heading, time-to-contact, and relative depth (Gibson, 1957).

In this section we study the computational theory for the estimation of relative depth, and viewer/camera heading from the optic flow pattern induced by general eye motion in a rigid environment. We follow a development described by Longuet-Higgins, H. C., & Prazdny, K. (1980). (See also Koenderink and van Doorn, 1976, Horn, Chapter 17, Perrone, 1992 for a biologically motivated model, and Heeger and Jepson, 1990).

Rather than following the particular derivation of Longuet-Higgins et al., we derive the relationship between the motion field and relative depth, and camera motion parameters using homogeneous coordinates. See: https://en.wikipedia.org/wiki/Transformation_matrix and https://en.wikipedia.org/wiki/Homogeneous_coordinates#Use _in _computer _graphics.

Setting up the frame of reference and basic variables

Imagine a rigid coordinate system attached to the eye, with the origin at the nodal point. General motion of the eye can be described by the instantaneous translational (U,V,W) and rotational (A,B,C) components of the frame. Let P be a fixed point in the world at (X,Y,Z) that projects to point (x,y) in the conjugate image plane which is unit distance in the z direction from the origin:



Goal 1: Derive generative model of the motion field, where we express the motion field (u,v) in terms of Z, U,V,W,A,B,C.

This generative model is strictly geometric.

Homogeneous coordinates

First we'll review homogeneous coordinates -- a cool mathematical trick for computing rigid 3D motions and perspective projection. And a tool fundamental to modern computer graphics software and hardware.

Rotation and scaling can be done by linear matrix operations in three-space. But translation and perspective transformations (3D to 2D) do not have a three dimensional matrix representation. By going from three dimensions to four dimensional homogeneous coordinates, all four of the basic operations, rotation, scaling, translation and perspective projection, can be represented using the formalism of matrix multiplication.

Homogeneous coordinates are defined by products of the coordinates with a scalar w: {xw, yw, zw, w }, (w not equal to 0). To get from homogeneous coordinates to three-space coordinates, {x,y,z}, divide the first three homogeneous coordinates by the fourth, {w}. And to get started, one takes the 3D vector and simply appends a 1 to the fourth slot.

Rotation and translation matrices can be used to describe object or eye-point changes of orientation and position. The scaling matrix allows you to squash or expand objects in any of the three directions. Any combination of the matrices can be concatenated through multiplication.

Note: Matrices do not in general commute when multiplying, so the order is important. The translation, rotation, and perspective transformation matrices can be concatenated to describe general 3-D to 2-D perspective mappings.

The 4D matrices for: rotation, scaling, translation and projection.

In[34]:=
```
Clear[x, y, z];
```

In[35]:=
```
XRotationMatrix[theta_] := {{1, 0, 0, 0},
   {0, Cos[theta], Sin[theta], 0}, {0, -Sin[theta], Cos[theta], 0}, {0, 0, 0, 1}};
YRotationMatrix[theta_] := {{Cos[theta], 0, -Sin[theta], 0},
   {0, 1, 0, 0}, {Sin[theta], 0, Cos[theta], 0}, {0, 0, 0, 1}};
ZRotationMatrix[theta_] := {{Cos[theta], Sin[theta], 0, 0},
   {-Sin[theta], Cos[theta], 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
ScaleMatrix[sx_, sy_, sz_] := {{sx, 0, 0, 0}, {0, sy, 0, 0}, {0, 0, sz, 0}, {0, 0, 0, 1}};
TranslateMatrix[x_, y_, z_] := {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {x, y, z, 1}};
ZProjectMatrix[focal_] :=
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, -N[1/focal]}, {0, 0, 0, 1}};
ZOrthographic[vec_] := Take[vec, 2];
ThreeDToHomogeneous[vec_] := Append[vec, 1];
HomogeneousToThreeD[vec_] := Drop[vec/vec[[4]], -1];
```

There are three matrices for general rotation:

z-axis (moving the positive x-axis towards the positive y-axis)

In[44]:=
```
ZRotationMatrix[θ] // MatrixForm
```

Out[44]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\theta] & \text{Sin}[\theta] & 0 & 0 \\ -\text{Sin}[\theta] & \text{Cos}[\theta] & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

x-axis (moving the positive y towards the positive z)

In[45]:=
```
XRotationMatrix[θ] // MatrixForm
```

Out[45]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}[\theta] & \text{Sin}[\theta] & 0 \\ 0 & -\text{Sin}[\theta] & \text{Cos}[\theta] & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

y-axis (moving positive z towards positive x):

In[46]:=
```
YRotationMatrix[θ] // MatrixForm
```

Out[46]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\theta] & 0 & -\text{Sin}[\theta] & 0 \\ 0 & 1 & 0 & 0 \\ \text{Sin}[\theta] & 0 & \text{Cos}[\theta] & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The scaling matrix is:

In[47]:=
```
ScaleMatrix[sx, sy, sz] // MatrixForm
```

Out[47]//MatrixForm=

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation by {d_x, d_y, d_z} is done by applying the matrix:

In[48]:=
```
Clear[x, y, z, d];
TranslateMatrix[dx, dy, dz] // MatrixForm
```

Out[49]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d_z & 1 \end{pmatrix}$$

In[50]:=
```
{x, y, z, 1}.TranslateMatrix[dx, dy, dz]
```

Out[50]=
```
{x + dx, y + dy, z + dz, 1}
```

to {x,y,z,1}

$$(x + d_x, \quad y + d_y, \quad z + d_z, \quad 1) = (x, \quad y, \quad z, \quad 1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d_z & 1 \end{pmatrix}$$

Perspective projection

Perspective transformation is the only homogeneous coordinate matrix operation that requires extracting the three-space coordinates by dividing the homogeneous coordinates by the fourth component w. The projection plane is the x-y plane, and the focal point is at z = d.

Then {x, y, z, 1} maps onto {x, y, 0, -z/d + 1 } by the following transformation:

In[51]:=
```
Clear[d]
ZProjectMatrix[d] // MatrixForm
```

Out[52]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{d} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

After normalization, the image coordinates {x',y',z'} are read from:

$$(x', \quad y', \quad z', \quad 1) = \left(\frac{xd}{d-z}, \quad \frac{yd}{d-z}, \quad 0, \quad 1\right)$$

The steps can be seen here:

In[53]:=
```
Clear[x, y, z, d]
{x, y, z, 1}.ZProjectMatrix[d]
{x, y, z, 1}.ZProjectMatrix[d] / %[[4]]
HomogeneousToThreeD[{x, y, z, 1}.ZProjectMatrix[d]]
Simplify[ZOrthographic[HomogeneousToThreeD[{x, y, z, 1}.ZProjectMatrix[d]]]]
```

Out[54]= $\left\{x, y, 0, 1 - \frac{z}{d}\right\}$

Out[55]= $\left\{\frac{x}{1 - \frac{z}{d}}, \frac{y}{1 - \frac{z}{d}}, 0, 1\right\}$

Out[56]= $\left\{\frac{x}{1 - \frac{z}{d}}, \frac{y}{1 - \frac{z}{d}}, 0\right\}$

Out[57]= $\left\{\frac{d\,x}{d - z}, \frac{d\,y}{d - z}\right\}$

The matrix for orthographic projection has d-> infinity.

In[58]:=
```
Limit[ZOrthographic[HomogeneousToThreeD[{x, y, z, 1}.ZProjectMatrix[d]]], d → ∞]
```

Out[58]= {x, y}

The perspective transformation is the only singular matrix in the above group.This means that, unlike the others its operation is not invertible. Makes sense--given the 2D image coordinates, the original scene points cannot be determined uniquely.

Note that there are different expressions for the perspective transformation depending on where one puts the x-y plane and the focal point. E.g. whether the x-y image plane is in front --the conjugate plane-- or behind the focal point, whether it is at the origin, or whether the focal point is at the origin.

Express velocity V of world point P, (X,Y,Z) in terms of motion parameters of the eye (or camera) frame of reference

Let **r**(t) represent the position of P in homogeneous coordinates:

$$\mathbf{r}(t) = (X, Y, Z, 1)$$

An instant later, the new coordinates are given by:

$$\mathbf{r}(t + \Delta t) = \mathbf{r} + \Delta\mathbf{r} = (X + \Delta X, Y + \Delta Y, Z + \Delta Z, 1) = \mathbf{r}(\Delta R_{\Theta_x}, \Delta R_{\Theta_y}, \Delta R_{\Theta_z}, \Delta T)$$

where infinitesimal rotations and translations are represented by their respective 4x4 matrices. (Matrix multiplication operations do not in general commute; however, for infinitesmal rotations, the order doesn't matter.) Then using homogeneous coordinate representations,

$$\Delta T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\Delta x_0 & -\Delta y_0 & -\Delta z_0 & 1 \end{bmatrix}$$

and

$$\Delta R_{\theta_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) & 0 \\ 0 & sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\Delta\theta_x & 0 \\ 0 & \Delta\theta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \textit{higher order terms}$$

where we've used the small angle approximations for sine and cosine.
We use similar approximations for the other rotation matrices, and the relation

$$\Delta \mathbf{r} = \mathbf{r}\Delta R_{\theta_x}\Delta R_{\theta_y}\Delta R_{\theta_z}\Delta T - \mathbf{r}I$$

where I is the identity matrix. With some manipulation, one can arrive at:

$$\{\Delta X, \Delta Y, \Delta Z, 0\} = \{X, Y, Z, 1\}\cdot\begin{pmatrix} 0 & -\Delta\theta_z & \Delta\theta_y & 0 \\ \Delta\theta_z & 0 & -\Delta\theta_x & 0 \\ -\Delta\theta_y & \Delta\theta_x & 0 & 0 \\ -\Delta x_0 & -\Delta y_0 & -\Delta z_0 & 0 \end{pmatrix} + \text{higher order terms}$$

From there we can derive expressions that describe the 3D velocity of P in the image-plane coordinates of the viewer/camera.

Let's use *Mathematica* to derive this formula and expressions for the velocity of P, i.e. V= (dX/dt, dY/dt, and dZ/dt)

Recall the **Series[]** function:

In[59]:=
```
??Series
```

Series[f, {x, x_0, n}] generates a power series expansion for f about the point $x = x_0$ to order $(x - x_0)^n$.
Series[f, {x, x_0, n_x}, {y, y_0, n_y}, ...] successively finds series expansions with respect to x, then y, etc. >>

```
Attributes[Series] = {Protected}

Options[Series] := {Analytic → True, Assumptions :→ $Assumptions}
```

Expand the rotation matrix into a Taylor series:

In[60]:=
```
Series[XRotationMatrix[Subscript[Δθ, x]], {Subscript[Δθ, x], 0, 1}] // MatrixForm
```

Out[60]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 + O[\Delta\theta_x]^2 & \Delta\theta_x + O[\Delta\theta_x]^2 & 0 \\ 0 & -\Delta\theta_x + O[\Delta\theta_x]^2 & 1 + O[\Delta\theta_x]^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Use Normal[] to chop off higher order terms:

In[61]:=
```
Normal[Series[XRotationMatrix[Subscript[Δθ, x]], {Subscript[Δθ, x], 0, 1}]] //
 MatrixForm
```

Out[61]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \Delta\theta_x & 0 \\ 0 & -\Delta\theta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translational matrix is:

In[62]:=
```
TranslateMatrix[-Subscript[Δx, 0],
  -Subscript[Δy, 0], -Subscript[Δz, 0]] // MatrixForm
```

Out[62]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\Delta x_0 & -\Delta y_0 & -\Delta z_0 & 1 \end{pmatrix}$$

Now let's put all the rotation and translational components together.

In[63]:=
```
Normal[
   Series[TranslateMatrix[-Subscript[Δx, 0], -Subscript[Δy, 0], -Subscript[Δz, 0]].
      XRotationMatrix[Subscript[Δθ, x]], {Subscript[Δθ, x], 0, 1}].
    Series[YRotationMatrix[Subscript[Δθ, y]], {Subscript[Δθ, y], 0, 1}].
    Series[ZRotationMatrix[Subscript[Δθ, z]], {Subscript[Δθ, z], 0, 1}]] -
  IdentityMatrix[4] // MatrixForm
```

Out[63]//MatrixForm=

$$\begin{pmatrix} 0 & \Delta\theta_z \\ \Delta\theta_x\,\Delta\theta_y - \Delta\theta_z & \Delta\theta_x\,\Delta\theta_y\,\Delta\theta_z \\ \Delta\theta_y + \Delta\theta_x\,\Delta\theta_z & -\Delta\theta_x + \Delta\theta_y\,\Delta\theta_z \\ -\Delta x_0 - \Delta z_0\,\Delta\theta_y + \Delta y_0\,\Delta\theta_z + \Delta\theta_x\,(-\Delta y_0\,\Delta\theta_y - \Delta z_0\,\Delta\theta_z) & -\Delta y_0 - \Delta x_0\,\Delta\theta_z - \Delta z_0\,\Delta\theta_y\,\Delta\theta_z + \Delta\theta_x\,(\Delta z_0 - \Delta y_0\,\Delta\theta_y \end{pmatrix}$$

Ignore the contributions of all second-order terms--set them to zero:

$$\begin{pmatrix} 0 & -\theta z & \theta y & 0 \\ \theta z & 0 & -\theta x & 0 \\ -\theta y & \theta x & 0 & 0 \\ -\Delta x_0 & -\Delta y_0 & -\Delta z_0 & 0 \end{pmatrix}$$

In[64]:=
```
{X, Y, Z, 1}.⎛  0     -Δθz    Δθy   0⎞
             ⎜ Δθz     0     -Δθx   0⎟
             ⎜-Δθy    Δθx     0    0⎟ // MatrixForm
             ⎝-Δx0   -Δy0   -Δz0   0⎠
```

Out[64]//MatrixForm=

$$\begin{pmatrix} -\Delta x_0 - Z\,\Delta\theta_y + Y\,\Delta\theta_z \\ -\Delta y_0 + Z\,\Delta\theta_x - X\,\Delta\theta_z \\ -\Delta z_0 - Y\,\Delta\theta_x + X\,\Delta\theta_y \\ 0 \end{pmatrix}$$

$$\{\Delta X,\ \Delta Y,\ \Delta Z,\ 0\} \sim \{-\Delta x - Z\,\Delta\theta_y + Y\,\Delta\theta_z,\ -\Delta y + Z\,\Delta\theta_x - X\,\Delta\theta_z,\ -\Delta z - Y\,\Delta\theta_x + X\,\Delta\theta_y,\ 0\}$$

Dividing by Δt, we obtain the following relations:

$$\frac{\Delta X}{\Delta t} = \frac{\Delta\theta_z}{\Delta t}Y - \frac{\Delta\theta_y}{\Delta t} - \frac{\Delta x}{\Delta t} = CY - BZ - U$$

$$\frac{\Delta Y}{\Delta t} = -\frac{\Delta y}{\Delta t} - \frac{\Delta\theta_z}{\Delta t}X + \frac{\Delta\theta_x}{\Delta t}Z = -V - CX + AZ$$

$$\frac{\Delta Z}{\Delta t} = -\frac{\Delta z}{\Delta t} - \frac{\Delta\theta_x}{\Delta t}Y + \frac{\Delta\theta_z}{\Delta t}X = -W - AY + BX$$

(NOTATION NOTE: Above the lower case Δx, Δy, Δz represent changes in the 3D world coordinates {X,Y,Z} due to the small translation, and should have 0 subscripts (as earlier) to distinguish them from the image-plane values, x and y used below. There we use {x,y} to represent the *projection of* {X,Y,Z}.)

The vector, {dX/dt, dY/dt, dZ/dt}, describing the velocity of world point P can be neatly summarized using the vector cross product:

In[65]:=
```
Clear[U, V, W, A, B, CC, X, Y, Z]
t = {U, V, W};
ω = {A, B, CC};
r = {X, Y, Z};

-Cross[ω, r] - t // MatrixForm
```

Out[69]//MatrixForm=

$$\begin{pmatrix} -U + CC\,Y - B\,Z \\ -V - CC\,X + A\,Z \\ -W + B\,X - A\,Y \end{pmatrix}$$

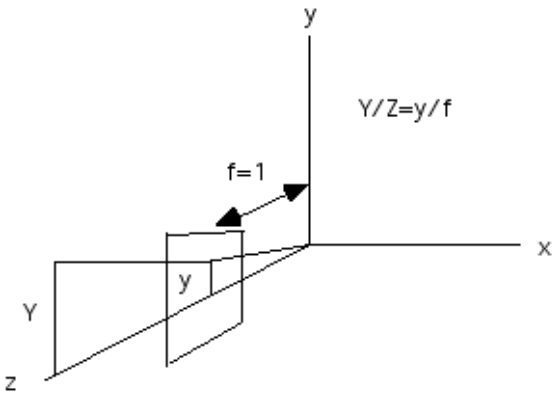
In summary, we have:

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \end{pmatrix} = \begin{pmatrix} -B\,Z + CC\,Y - U \\ A\,Z - CC\,X - V \\ -A\,Y + B\,X - W \end{pmatrix} \tag{1}$$

To simplify notation, we will use the “dot” convention to indicate the temporal derivatives of X,Y, and Z.

So far so good. We have described the velocity of P in world coordinates in terms of the rotational and translational velocity components of the moving coordinate system of the eye (or camera). What is happening in the image--i.e. to the motion field or optic flow?

Next step: Relate P velocity and depth Z to the motion field, {u,v}



(Another note: light travels in straight lines, and so should the projected line in the figure above!)

For convenience, let’s assume the image plane is at f=1, and the nodal point of the eye at the origin. The perspective projection is:

$$(x, y) = \left(\frac{X}{Z}, \frac{Y}{Z}\right)$$

and the motion field in terms of Z, and using the product rule for differentiation, the rates of change of X,Y, and Z are:

$$u = \frac{dx}{dt} = \dot{x} = \frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2} \tag{2}$$

$$v = \frac{dy}{dt} = \dot{y} = \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} \tag{3}$$

- Side note: this is the same result we showed in Lecture 18, relating the velocity $\boldsymbol{v_0}$ of point P ($\mathbf{r} = \boldsymbol{r_0}$), to image velocity $\boldsymbol{v_i}$:

$$\mathbf{v_i} = \frac{(\mathbf{r_0} \times \mathbf{v_0}) \times \mathbf{Z}}{(\mathbf{r_0} \cdot \mathbf{Z})^2}$$

$\mathbf{r_0} = \{X, Y, Z\}$

$\mathbf{v_0} = \{\dot{X}, \dot{Y}, \dot{Z}\}$

```
z = {0, 0, 1}
```

In[89]:=
```
Cross[Cross[{X, Y, Z}, {Ẋ, Ẏ, Ż}], {0, 0, 1}] / ({X, Y, Z}.{0, 0, 1})^2;
Drop[%, -1] // Expand
```

Out[90]= $\left\{\frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2}, \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2}\right\}$

Main result for goal 1, the generative model:

Substituting the expressions for the rate of change of X,Y, and Z (equation 1) in equations 2 and 3, we have:

$$u = \left(\frac{-U + Wx}{Z}\right) + (-B + Cy + Axy - Bx^2) \tag{4}$$

$$v = \left(\frac{-V + Wy}{Z}\right) + (-Cx + A + Ay^2 - Bxy) \tag{5}$$

Note that we have organized the terms on the right of each equation so that the first parts do not depend on A,B, or C--that is

$$u_T = \left(\frac{-U + Wx}{Z}\right)$$

is a purely translational component.

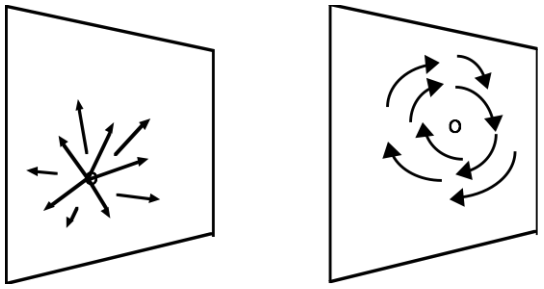
The second term in brackets is purely rotational, and does not depend on Z:

$$u_R = (-B + Cy + Axy - Bx^2)$$

So in general, we can write:

$$u = u_T + u_R \qquad v = v_T + v_R$$

The figure on the left below shows the flow field one would expect from a purely translational motion--there is a center of expansion (which could be off a finite image plane). The right panel shows the flow pattern of a rotational field.



- Note:

The above equations can be summarized in the following matrix equation:

uu(x,y) = p(x,y)**A**(x,y).**t** + **B**(x,y).***ω***

where **uu** = (u,v), p(x,y) = 1/Z(x,y), **t**= (**U**,**V**,**W**), ***ω*** = (**A**,**B**,**CC**).

(We replace **C** by **CC**, because **C** is protected in *Mathematica*.)

We can verify that the matrix equation gives the solution we derived:

In[91]:=
```
Clear[A, B, CC, U, V, W, X, Y, Z];
AA[x_, y_] := {{-f, 0, x}, {0, -f, y}}
BB[x_, y_] := {{(x * y) / f, -(f + x^2) / f, y}, {f + y^2 / f, -(x * y) / f, -x}}
uu[x_, y_] := (1 / Z) * AA[x, y].{U, V, W} + BB[x, y].{A, B, CC}
```

In[95]:=
```
f = 1;
uu[x, y] // MatrixForm
```

Out[96]//MatrixForm=

$$\begin{pmatrix} B\left(-1-x^2\right)+CC\,y+A\,x\,y+\frac{-U+W\,x}{Z} \\ -CC\,x-B\,x\,y+A\left(1+y^2\right)+\frac{-V+W\,y}{Z} \end{pmatrix}$$

- What is the motion field when there is no rotation?

Demo of motion field for planar surface

Let's plot the motion field for a planar surface, such as for motion over the ground plane. The equation for a plane in camera coordinates can be written:

$Z = aX + bY + Z_0$
Substituting from x = fX/Z, y = fY/Z,
we get
$Z(x,y) = \frac{f Z_0}{(f - ax - by)}$
We can use this to define a function p(x,y) =1/Z:
p[x_, y_] := (1/Z0) - (a/(f*Z0))*x - (b/(f*Z0)) y

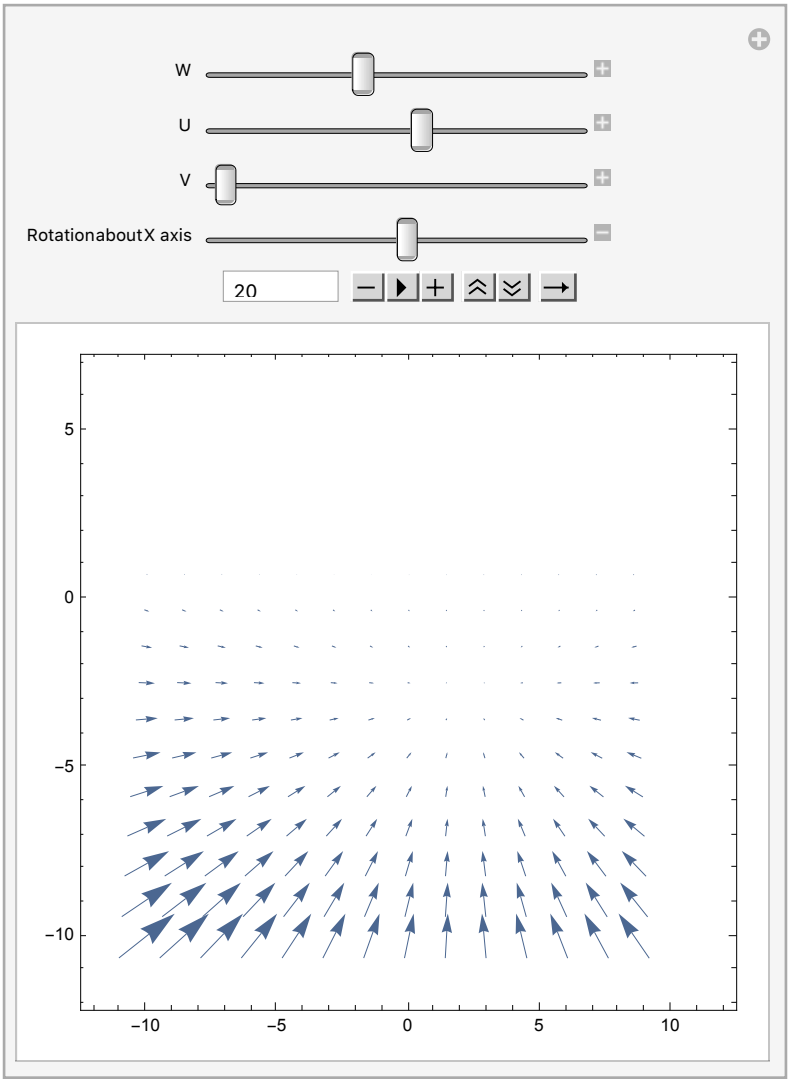
In[97]:=
```
Clear[A, B, CC, U, V, W, X, Y, Z, uu, p];

p[x_, y_] := (1 / Z0) - (a / (f * Z0)) * x - (b / (f * Z0)) y
a = 0; b = 1.0; f = 1.0; Z0 = -1;

AA[x_, y_] := {{-f, 0, x}, {0, -f, y}}
BB[x_, y_] := {{(x * y) / f, -(f + x^2) / f, y}, {f + y^2 / f, -(x * y) / f, -x}}
uu2[x_, y_] := p[x, y] * AA[x, y].{U, V, W} + BB[x, y].{A, B, CC}
```

In[103]:=
```
A = 0; B = 0; CC = -2;
U = 0; V = 0; W = 0;
Manipulate[
 VectorPlot[If[y < 1, p[x, y] * AA[x, y].{U, V, W} + BB[x, y].{A, B, CC}, {0, 0}],
  {x, -10, 10}, {y, -10, 5}, VectorScale → .1], {{W, 20}, 0, 50}, {{U, 0}, -500, 500},
 {V, -100, 50}, {{A, 0.0, "Rotation about X axis"}, -300, 300, 10}]
```

Out[105]=



With the rotation about the X axis set to zero (A=0), observe how changing the camera translational vector elements (U,V,W) changes the field singularity corresponding to the focus of expansion. The focus of expansion indicates the direction of heading.

Now while leaving the (U,V,W) fixed, introduce a non-zero rotation (e.g. A>0). Notice how the focus of expansion moves. The focus no longer corresponds to the direction of heading.

Goal 2: Inference model: given (u,v), how can we obtain estimates of A,B,C,U,V,W,Z?

In general we can't obtain all seven unknowns (see Horn's book, chap. 17). One problem is that scaling Z by a constant, can be exactly compensated for by a reciprocal scaling of (U,V,W) yielding an

unchanged motion field. See equations 4 and 5.

In other words, a given motion flow on your retina could be due to fast movement through a bigger world, or slower movement through a smaller world.
Horn discusses least squares solutions for the direction of camera motion, and for its rotational component. See also Heeger and Jepson (1990).
Although one can use Bayesian methods for the insufficiently constrained problem of estimating any or all of the seven unknowns, let's see how far one can get with simple algebra to get movement direction (not speed) and relative depth (not absolute depth). We follow the original work of Longuet-Higgins et al. (1980) for estimating the camera direction, and relative depth.

Pure translation: Obtaining direction of heading and relative depth

Suppose the rotational component is zero. Then measurements of the motion field will give us the translational components. These components constrain U,V,W, and Z at each point (x,y) in the conjugate image plane.

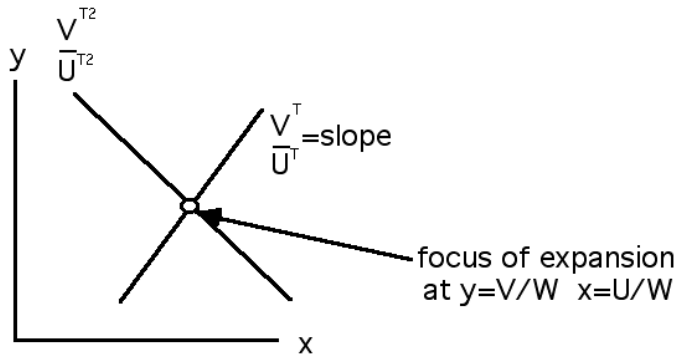
$$u^\tau = \frac{-U + xW}{Z}$$

$$v^\tau = \frac{-V + yW}{Z}$$

Combining these two equations to eliminate Z:

$$(y - \frac{V}{W}) = (x - \frac{U}{W})\frac{v^\tau}{u^\tau}$$

This equation is a straight line whose slope is determined by the ratio of the vertical and horizontal components of the flow field (v^τ / u^τ) , and which passes through the point (U/W,V/W). This point depends only on the camera's translational velocity, so other motion flow field lines with different ratios of vertical and horizontal components ($v^{\tau 2} / u^{\tau 2}$) of the flow also pass through this point. The point (U/W,V/W) is the focus of expansion.



(Notation note: the ratios V^T/U^T in the figure above correspond to v^T/u^T in the equations.)

Two motion field lines determine the focus of expansion, and thus the camera's translational direction:

$$\frac{(U, V, W)}{\sqrt{U^2 + V^2 + W^2}} = \frac{(U / W, V / W, 1)}{\sqrt{U^2 / W^2 + V^2 / W^2 + 1}}$$

We can also obtain an estimate of the relative depth of points:

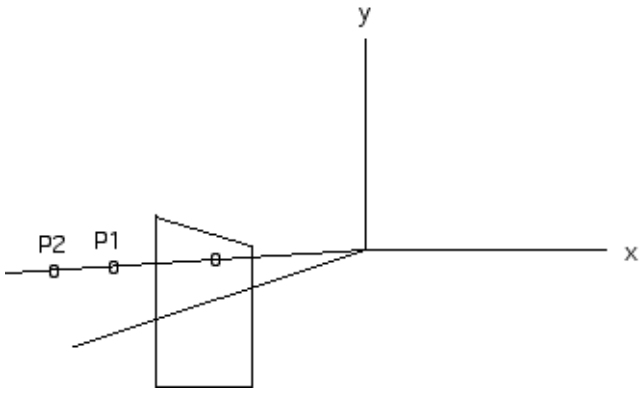
$$\frac{Z}{W} = \frac{x - U/W}{u^T} = \frac{y - V/W}{v^T}$$

Pure rotation

As we noted above, when the camera has zero translational velocity, the motion field does not depend on the depth structure of the scene. The field is a quadratic function of image position.

General motion field: Estimate both rotation and translational components using points of occlusion to get Z, the depth structure

What if we have a mix of translation and rotation? One solution suggested by Longuet-Higgins and Prazdny is to make use of *motion parallax,* where we have two 3D points that project to the same conjugate image point.



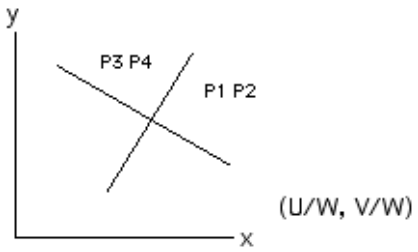
In general, these two points will have different motion field vectors at this image point. If we take the difference, we have:

$$u_1 - u_2 = (-U + xW)\left(\frac{1}{Z_1} - \frac{1}{Z_2}\right)$$

$$v_1 - v_2 = (-V + yW)\left(\frac{1}{Z_1} - \frac{1}{Z_2}\right)$$

$$\left(y - \frac{V}{W}\right) = \left(\frac{v_1 - v_2}{u_1 - u_2}\right)\left(x - \frac{U}{W}\right)$$

Again, finding the focus of expansion (xo,yo), which involves finding at least two motion parallax pairs,



will give us the camera (or eye) direction

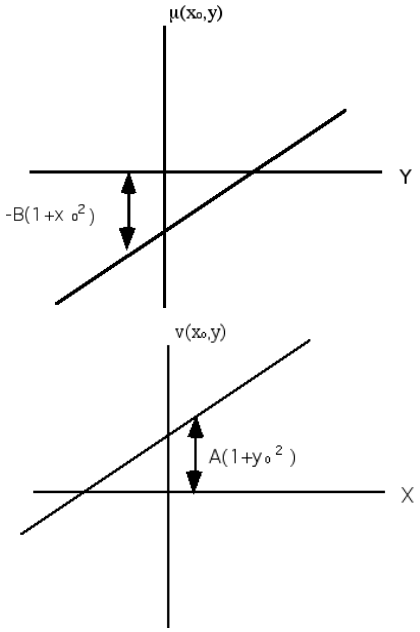
$$x_0 = \frac{U}{W}$$

$$y_0 = \frac{V}{W}$$

To find relative depth, we need to know A,B,C:

$$u(x_0, y) = (C + Ax_0)y - B(1 + x_0^2)$$

$$v(x, y_0) = -(C + By_0)x + A(1 + y_0^2)$$



These relations provide sufficient information to calculate A,B,C (from two or more points). A,B,C in turn determine u^R and v^R.

$$u = (x - x_0)\frac{W}{Z} + u^R;\ v = (y - y_0)\frac{W}{Z} + v^R$$

With some rearrangement, we can obtain a formula for relative depth:

$$u - u^R = (x - x_0)\frac{W}{Z}$$

$$v - v^R = (y - y_0)\frac{W}{Z}$$

$$\frac{Z}{W} = \frac{x - x_0}{u - u^R} = \frac{y - y_0}{v - v^R}$$

Although we won't take the time to go over the results, a potentially important form of information for relative depth, camera motion, and time-to-contact comes from an analysis of the flow patterns generated by textured surfaces (Koenderink, J. J., & van Doorn, A. J., 1976) and the above cited article by Longuet-Higgins and Prazdny. The idea is to compute estimates of the rotation, dilation, and shear of the motion field.

- Exercise: Time-to-contact

Problem: Show that the reciprocal of the temporal rate of expansion of an object heading directly towards you is equal to the time to contact. (Lee and Reddish, 1981).

Heading experiments

Structure from motion: Psychophysics

Warren and Hannon (1988) provided the first compelling evidence that the human visual system could compensate for eye rotation purely from optical information. Royden, Banks & Crowell (1992) later pointed out the role of proprioceptive information in heading computation, especially for faster motions.

Structure from motion: Physiology

A possible neurophysiological basis for derivative measurements of flow (e.g. rotation, dilation, shear), see: (Saito, H.-A., Yukie, M., Tanaka, K., Hikosaka, K., Fukada, Y., & Iwai, E., 1986). For work relating to eye movement compensation in optic flow and heading, See Bradley et al. (1996) and Duffy (2000).

Tracking

Kalman Filter notes

Kalman filter demo

Initializations

```
sw0 = 10.0;
Q = DiagonalMatrix[{sw0, sw0, .2 * sw0, .2 * sw0}];
ndistw = MultinormalDistribution[{0, 0, 0, 0}, Q];
w := RandomReal[ndistw];

sv0 = 250.0;
R0 = {{sv0, 0.0}, {0.0, sv0}};
ndistv = MultinormalDistribution[{0, 0}, R0];
v := RandomReal[ndistv];
```

Generate synthetic data: Model true trajectory plus noise

Initialize and generate

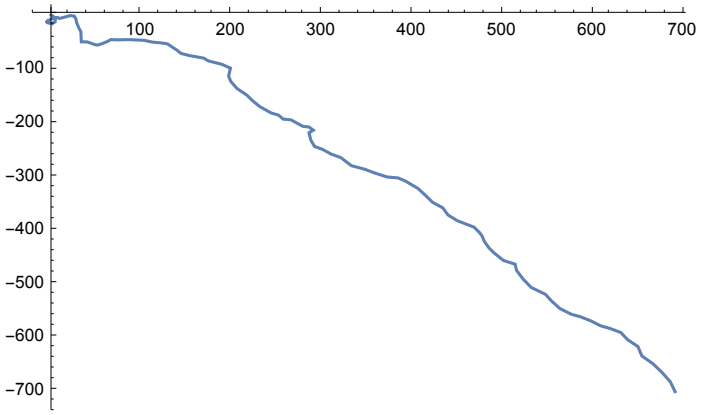
```
iter = 100;
y0 = {.0, 0.0, .1, .1};

A = {{1, 0, 1, 0}, {0, 1, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 1}};
H = {{1, 0, 0, 0}, {0, 1, 0, 0}};

y[k_] := A.k + w
ys = NestList[y, y0, iter - 1];
x = H.# & /@ ys + RandomReal[ndistv, iter];
```

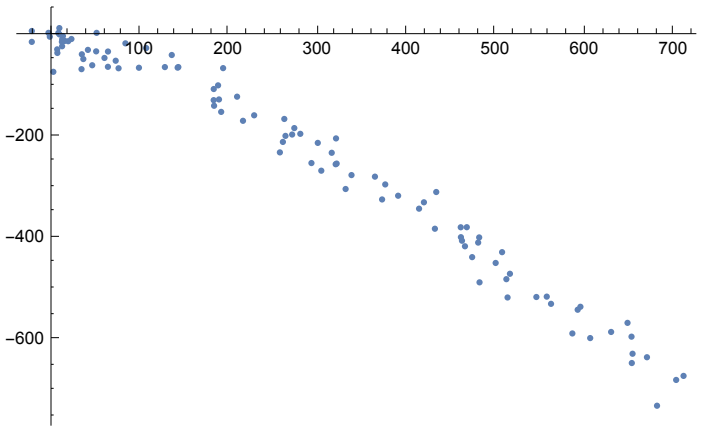
True path

```
g2 = ListPlot[H.# & /@ ys, Joined → True]
```



Noisy measurements

```
g1 = ListPlot[x]
```



Estimate path from data

Initialize arrays

```
yh = Table[{0, 0, 0, 0}, {iter}];
K = Table[{{0, 0}, {0, 0}, {0, 0}, {0, 0}}, {iter}];
P = Table[DiagonalMatrix[{0, 0, 0, 0}], {iter}];
Pm = P;
yhp = yh;
```

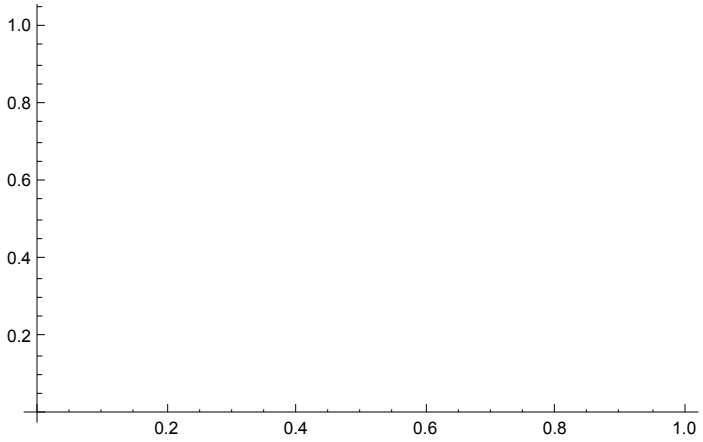
Initial estimates

```
R = 100 R0; (* Change measurement noise assumption. If R is small,
then the data is trusted--the dots get joined. If the
 data is believed to be noisy, the state estimate is smoothed *)
```

Graph here gets dynamically updated when the Kalman fiter section below is evaluated

The path moves from right to left.

```
Dynamic[gxys = ListPlot[{x, H.# & /@ ys, H.# & /@ yhp}, Joined → {False, True, True}]]
```



Kalman filter

```
For[t = 1, t < iter, t++,
  (*Time update--predict the state and error covariance *)
  yh[[t + 1]] = A.yhp[[t]];
  Pm[[t + 1]] = A.P[[t]].Transpose[A] + Q;

  (*Measurement update--correction*)
  K[[t + 1]] = Pm[[t + 1]].H^T.Inverse[H.Pm[[t + 1]].H^T + R];

  P[[t + 1]] = (IdentityMatrix[4] - K[[t + 1]].H).Pm[[t + 1]];

  yhp[[t + 1]] = yh[[t + 1]] + K[[t + 1]].(x[[t + 1]] - H.yh[[t + 1]]);
  (*gxys=ListPlot[{x,H.#&/@ys,H.#&/@yhp},Joined→{False,True,True}];*)
  Pause[.05];
 ];
```

Challenges to computing structure from motion

Multiple motions, transparency

In general, the environment has multiple moving objects in addition to ourselves--imagine driving a car.

If one wants to compute direction of heading, an optimal solution requires separating out the components of flow that do not belong to the rigid background--a segmentation problem. Another segmentation problem arises in depth from cast shadows, which we've seen before.

Computing motion can involve complex surface relationships, such as near bushes in front of a distant set of houses. One approach to dealing with multiple surfaces is to asume the flow field arises from a discrete set of layers, and estimate flow parameters within each. See Jepson and Black.

There has also been considerable work in computer vision to solve the problem of tracking, e.g. multiple pedestrians. Below, we describe the Kalman filter approach to tracking. We don't yet have a complete picture of how the human visual system copes with multiple motions when determining scene structure, direction of heading, or time to contact.

Empirical work

For more information, see work by Royden, Banks, Warren, Crowell and others. There is a substantial literature on the neural basis of optic flow computations and their functions. For a review see: Wurtz, R. H. (1998). Optic flow: A brain region devoted to optic flow analysis? Curr Biol, 8(16), R554-556.
For ideas on neural coding, see: Zemel, R. S., & Sejnowski, T. J. (1998). A model for encoding multiple object motions and self-motion in area MST of primate visual cortex. J Neurosci, 18(1), 531-547.

Depth from viewer

For some combined behavioral and computational modeling work on computing depth from the viewer see: Interception and time-to-contact when the object is not directly approaching the viewer. See Battaglia et al. (2005; 2011). Cue integration: Shadow displacement & size change for depth. Frame of reference issues in cue integration. (Schrater and Kersten, 2000.)

Appendix

Graphical model analysis: Shadow displacement & size change for depth

Frame of reference issues in cue integration (Schrater, & Kersten, 2000).

Earlier we looked at a simple graph for cue integration and showed how a optimal estimate (for the Gaussian case), say for depth, was a weighted combination of the estimates for the individual cues. The weights were determined from the relative reliabilities of the cues.
But a close examination of the generative models that result in multiple cues can show a more complex set of dependencies.

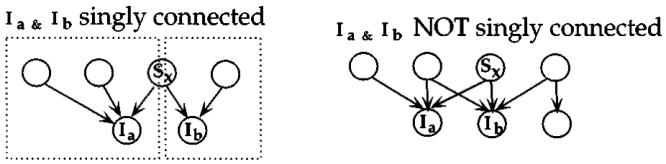
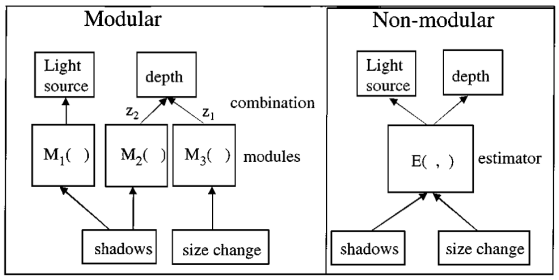


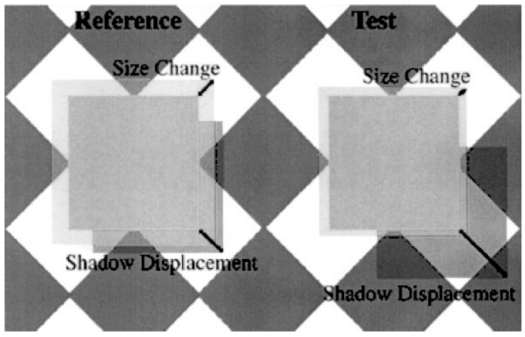
Figure 3. Whether independent data measures are singly connected to the estimated variable S_x determines whether or not estimation modules can be created for S_x. *Left* example of Bayesian modularity. Boxes show how the variables can be split to form two modules. *Right* example of a non-modular estimation.

This has an impact on the architecture for optimal inverse inference--whether the algorithm can be broken into distinct modules or not. The non-modular case below is an example of what Clark and Yuille called "strong fusion". This is related to the notion of "cooperative computation" discussed earlier in

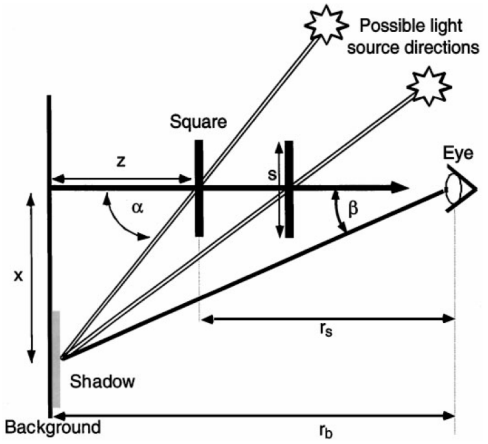
Lecture 23 on perceptual integration.



Let's take a look at a specific case involving size and shadow position as cues for an object's 3D position in space.



The figure below shows some of the relationships between the data (shadow position $\boldsymbol{\beta}$, size of the target square is **a**--not shown), and unknown parameters to be estimated (**z**, $\boldsymbol{r_s}$) of interest, (the unit-less parameter, $\mathbf{z}/\boldsymbol{r_b}$ is not shown), and unknowns to be integrated out ($\boldsymbol{\alpha}$, **s**, $\boldsymbol{r_b}$--depending on the task).



When we perceive a change in depth, *what variable does perceived depth correspond to?* Here are three possibilities: relative (unit-less) distance zr/rb, depth from the observer, rs, and distance from the background z.

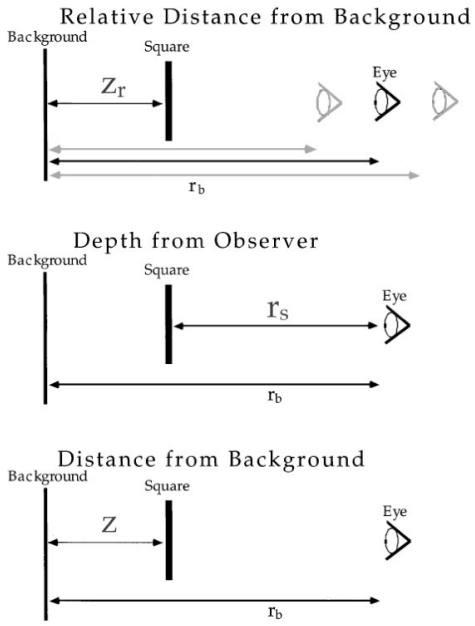


Figure 6. Diagram illustrating the depth variables to be estimated. The variable $z_r = z/r_b$ can't be shown directly, because it is an equivalence class of z and r_b distances.

Paul Schrater worked through the math and showed that these different assumptions about depth representation produced different generative models for producing the image size **a**, and shadow position, ***β***.

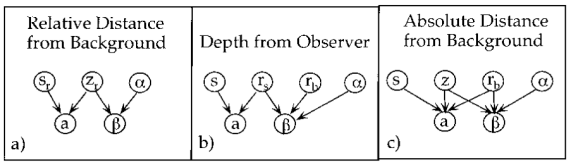


Figure 7. Bayes nets for the three depth representations. a) Bayes net for relative distance to the background. This task involves estimating object relations (world centered), and requires the least prior knowledge. b) Bayes net for distance to observer. Notice that the use of the shadow information requires integrating across two variables, hence the shadow cue should have more uncertainty for this task. c) Bayes net for metric depth from the background. Estimating the distance from the background, z, is complicated by the image size and shadow position measurements also being jointly dependent on the observer's distance to the background.

Fisher information is the asymptotic variance of the estimator, so can be used to calculate a weighted linear combination (an optimal estimator for the modular case).

Table 1. Table of MAP estimates and Fisher information values for the three depth estimate representations. For the representations which admit modular estimates, the estimates are shown separately for the shadow and image size cues.

Task	Est from shadow	Est from size	Shadow Fisher info	Size Fisher info
Relative z	$z_r = \tan(\hat{\beta})$	$z_r = 1 - \frac{\mu_{s_r}}{\hat{a}}$	$\frac{1}{\sqrt{2}\,\tan(\hat{\beta})^2}$	$\frac{2\hat{a}^2}{\mu_{s_r}^2(\sigma_{s_r}^2 + \sigma_{\hat{a}}^2)}$
Dist. from obs.	$r_s = \mu_{r_b}(1 - \tan(\beta))$	$r_s = \frac{\mu_s}{\hat{a}}$	$\frac{1}{\mu_{r_b}^2\,\tan(\hat{\beta})^2}$	$\frac{2\hat{a}^2}{\mu_s^2(\sigma_s^2 + \sigma_{\hat{a}}^2)}$
Absolute z	$z = \frac{\mu_s\,\tan(\hat{\beta})}{\hat{a}(1 - \tan(\hat{\beta}))}$		$\frac{2\hat{a}^2(1 - \tan(\hat{\beta}))^4}{\mu_s^2\,\tan(\hat{\beta})^2}$	

The shadow cue is most reliable when the target object is close to the background. But the size cue is most reliable when the target is close to the viewer. To date, there have been no systematic experimental studies of this general theoretical prediction.

Bottom line

Optimal estimators for depth depend critically on the representation of depth. Different representations result in different generative models, and thus different modular structures for optimal inference. Human judgments of closeness may be better predicted by a model that represents depth from the observer, rather than relative depth from the background, in either absolute (e.g. metric) units, or relative units. More experimental work is needed.

References

Battaglia, P. W., Schrater, P. R., & Kersten, D. J. (2005). Auxiliary object knowledge influences visually-guided interception behavior. *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, 145–152.

Battaglia, P. W., Kersten, D., & Schrater, P. R. (2011). How haptic size sensations improve distance perception. *PLoS Computational Biology*, *7*(6), e1002080. doi:10.1371/journal.pcbi.1002080

Bradley, D. C., Maxwell, M., Andersen, R. A., Banks, M. S., & Shenoy, K. V. (1996). Mechanisms of heading perception in primate visual cortex [see comments]. Science, 273(5281), 1544-7.

Crowell, J. A., & Banks, M. S. (1996). Ideal observer for heading judgments. Vision Res, 36(3), 471-90.

d'Avossa, G., & Kersten, D. (1996). Evidence in human subjects for independent coding of azimuth and elevation for direction of heading from optic flow. Vision Research, 36(18), 2915-2924.

Duffy, C. J. (2000). Optic flow analysis for self-movement perception. Int Rev Neurobiol, 44, 199-218.

Gibson. (1957). Optical motions and transformations as stimuli for visual perception. Psychological Review, 64, 288-295.

Fulvio, J. M., Green, C. S., & Schrater, P. R. (2014). Task-Specific Response Strategy Selection on the Basis of Recent Training Experience. PLoS Computational Biology, 10(1), e1003425–16. http://-doi.org/10.1371/journal.pcbi.1003425

Heeger, D. J., & Jepson, A. (1990). Visual perception of three-dimensional motion. Neural Computation, 2(2), 129-137.

Jepson, A., & Black, M. J. (1993). Mixture models for optical flow computation. Paper presented at the Proc. IEEE Conf. Comput. Vsion Pattern Recog., New York.
(link)

Koenderink, J. J., & van Doorn, A. J. (1976). Local Structure of Movement Parallax of the Plane. J. Opt. Soc. Am., 66, 717-723.

Lappe, M., & Rauschecker, J. P. (1993). A neural network for the processing of optic flow from ego-motion in man and higher mammals. Neural Computation, 5, 374-391.

Lee, D. N., & Reddish, P. E. (1981). Plummeting gannets: a paradigm of ecological optics. Nature, 293(5830), 293-294.

Longuet-Higgins, H. C., & Prazdny, K. (1980). The Interpretation of a Moving Retinal Image. Proceedings of the Royal Society of London B, 208, 385-397.

Martin, W. N., & Aggarwal, J. K. (1988). Motion understanding: robot and human vision. Boston: Kluwer Academic Publishers,

Perrone, J. A. (1992). Model for the computation of self-motion in biological systems. Journal of the Optical Society of America, A, 9(2), 177-194.

Royden, C. S., Banks, M. S., & Crowell, J. A. (1992). The perception of heading during eye movements [see comments]. Nature, 360(6404), 583-5.
Saito, H.-A., Yukie, M., Tanaka, K., Hikosaka, K., Fukada, Y., & Iwai, E. (1986). Integration of Direction Signals of Image Motion in the Superior Temporal Sulcus of the Macaque Monkey. The Journal of Neuroscience, 6(1), 145-157.
Schrater PR, Kersten D (2000) How optimal depth cue integration depends on the task. International Journal of Computer Vision 40:73-91.
Ullman, S. (1979). The Interpretation of Structure from Motion. Proceedings of the Royal Society London B, 203, 405-426.)
Vishwanath, D., Girshick, A. R., & Banks, M. S. (2005). Why pictures look right when viewed from the wrong place. Nature Neuroscience, 8(10), 1401–1410. http://doi.org/10.1038/nn1553
Wang, J. Y. A., & Adelson, E. H. (1994). Representing moving images with layers. IEEE Transactions on Image Processing Special Issue: Image Sequence Compression, 3(5), 625-638.
Warren, W. H., & Hannon, D. J. (1988). Direction of Self-Motion is Perceived from Optical Flow. Nature, 336((10)), 162-163.
Zemel, R. S., & Sejnowski, T. J. (1998). A model for encoding multiple object motions and self-motion in area MST of primate visual cortex. J Neurosci, 18(1), 531-547.