Computational Vision
U. Minn. Psy 5036
Daniel Kersten
Lecture 24: Space, Structure from Motion

Outline

Last time
Object recognition overview

Today
Object recognition: compensating for viewpoint changes
Computational theory for estimating relative depth, camera motion
Challenges to computational theories of depth and spatial layout

Geometric variation: 3D scene-based modeling

Homogeneous coordinates

Example: transforming, projecting a 3D object

Define 3D target object - Wire with randomly positioned vertices
ScatterPlot3D[threeDtemplate, PlotJoined -> True,
    PlotStyle -> {Thickness[0.02], RGBColor[1, 0, 0]}];

First view

View from along Z-direction

ScatterPlot3D[threeDtemplate, ViewPoint -> {0, 0, 100},
    PlotJoined -> True, PlotStyle -> {Thickness[0.02], RGBColor[1, 0, 0]},
    PlotRange -> {(-.5, 1.5), (-.5, 1.5), (-.5, 1.5)}];

ListPlot view

ovg = ListPlot[orthoproject[threeDtemplate],
    PlotJoined -> True,
    PlotStyle -> {Thickness[0.02], RGBColor[1, 0, 0]},
    PlotRange -> {(-.5, 1.5), (-.5, 1.5)}];

New View

Use Homogeneous coordinates

swidth = 1.0; sheight = 1.0; slength = 1.0; d = 0;

homovertex = Transpose[Map[ThreeDToHomogeneous, threeDtemplate]]; newtransformMatrix = TranslateMatrix[1, 0, 0].XRotationMatrix[\[Pi]/16].
           YRotationMatrix[\[Pi]/16].ScaleMatrix[swidth, sheight, slength];

temp = N[newtransformMatrix.homovertices];

Take a look at the new view

newvertices = Map[HomogeneousToThreeD, Transpose[temp]];
Geometric variation: 2D image-based modeling

Suppose one wants to check to see if the blue image above is indeed a 3D rotated version of the original familiar object. One could look for rotation parameters that minimize the squared error of the image projection.

Is there a 2D image approximation that provides an alternative, even if not perfect?

If one projects a small rotation in 3D onto a 2D view, the rotation can be approximated by a 2D affine transformation. Because a 2D affine transformation is a simple 2D operation, perhaps it is sufficient to account for the generalization of familiar to unfamiliar views in human observers (Liu and Kersten).

Affine transformation preserve parallel lines.

We know that rotations, scale and shear transformations will do this. So will translations. It is not immediately apparent that any matrix operation is an affine transformation, although one has to remember that translations are not represented by matrix operations unless one goes to homogeneous coordinates. Here is a simple demo of the parallel line preservation for transformations of a cube.

Define a square

```
square = {{1, 1}, {-1, 1}, {-1, -1}, {1, -1}};
```

```
Show[Graphics[Polygon[square]],
AspectRatio -> 1, PlotRange -> {{-2, 2}, {-2, 2}}];
```

A rotation

```
Show[Graphics[Polygon[M[theta_].#1 & /@ square]],
AspectRatio -> 1, PlotRange -> {{-2, 2}, {-2, 2}}];
```

```
M[theta_] := {{Cos[theta], Sin[theta]}, {-Sin[theta], Cos[theta]};
```
Let's try a random matrix to see if it preserves parallel lines

```math
\text{MR} = \text{Table}[\text{Random[]}, \{2\}, \{2\}]
```

Show[Graphics[Polygon[MR.\& square]],
AspectRatio -> 1, PlotRange -> ((-2, 2), (-2, 2))];

`{(0.737607, 0.561893), (0.741276, 0.948084)}`

Compute closest least squares affine match with translation

```math
\text{aff} = \{(a, b), (c, d)\};
\text{tra} = \text{Transpose}[[f, g], \{f, g\}, \{f, g\}];
\text{errorsum} :=
\text{Apply}[\text{Plus}, \text{Flatten}[[\text{aff}.\text{Transpose}[\text{orthoproject}[\text{newvertices}]] + \text{tra} -
\text{Transpose}[\text{orthoproject}[\text{threeDtemplate}]]^2]]];
\text{temp} = \text{FindMinimum}[\text{errorsum}, \{(a, .8), \{b, .2\}, \{c, .16\}, \{d, .8\},
\{f, 0.0\}, \{g, 0.0\}, \text{MaxIterations} \to 200]\];
\text{minvals} = \text{Take}[\text{temp}, -1][[1]]; \text{minerr} = \text{Take}[\text{temp}, 1][[1]];\n\text{naff} = \text{aff} /. \text{minvals}; \text{ntra} = \text{tra} /. \text{minvals};\n\text{minerr}
```

0.171834

Check match with estimated view

```math
\text{estim} = \text{naff}.\text{Transpose}[\text{orthoproject}[\text{newvertices}]] + \text{ntra};
```

Plot familiar view, new view and the affine estimate of the old from the new

```math
\text{avg} = \text{MultipleListPlot}[[\text{orthoproject}[\text{threeDtemplate}]],
\text{Transpose}[\text{estim}], \text{orthoproject}[\text{newvertices}], \text{PlotJoined} \to \text{True},
\text{PlotStyle} \to \{\{\text{Thickness}[0.002], \text{RGBColor}[1, 0, 0]\},
\{\text{Thickness}[0.005], \text{RGBColor}[0, .5, 0]\},
\{\text{Thickness}[0.005], \text{RGBColor}[0, 0, 1]\},
\text{PlotRange} \to \{(-5, 5), (-5, 5)\}];
```

Note that we haven't used 3D rotations to try to get a match between the original familiar view (red) and the new view (blue)–we've assumed that an affine match might get us close. Comparing the red and the green shows how close.

Compute closest least squares affine match without translation

If we have some other means to compensate for translation, we can look for the subset of affine parameters (i.e. matrix parameters) that minimize the total squared error in the reconstruction. Then we can use the built-in \text{PseudoInverse[]} function to find the solution, essentially in one line:

```math
\text{naff2} = \text{Transpose}[\text{orthoproject}[\text{threeDtemplate}]].\text{PseudoInverse}[\text{Transpose}[\text{orthoproject}[\text{newvertices}]]];
```

```math
\{(1.43981, 0.798144), (0.11845, 2.33565)\}
```
**Check match with estimated view**

\[
\text{estim2} = \text{naff2}.\text{Transpose} [\text{orthoproject [newvertices]}];
\]

- Plot familiar view, new view and the affine estimate of the old from the new

\[
\text{evg} = \text{MultipleListPlot} [\text{orthoproject [threeDtemplate]}, \\
\text{Transpose} [\text{estim2}], \text{orthoproject [newvertices]}, \text{PlotJoined} \to \text{True}, \\
\text{PlotStyle} \to \{(\text{Thickness}[0.02], \text{RGBColor}[1, 0, 0]), \\
(\text{Thickness}[0.01], \text{RGBColor}[0, 1, 0]), \\
(\text{Thickness}[0.01], \text{RGBColor}[0, 0, 1])\}, \\
\text{PlotRange} \to \{(-0.5, 1.5), (-0.5, 1.5)\};
\]

Note that we again haven't used 3D rotations to try to get a match between the original familiar view (red) and the new view (blue)—we've assumed that an 2D matrix operation might get us close to a match. Comparing the red and the green shows how close.

**Spatial layout: Where are objects? Where is the viewer?**

Recall distinctions: Between vs. within object geometry.

**Where are objects?**

- **Absolute**
  
  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 requirements.

  "Pictorial cue"—familiar size

- **Relative**
  
  Distance between objects or object feature points. Important for scene layout.

  Processes include: Stereopsis (binocular parallax) and motion parallax.

  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
Where is the viewer? And where is the viewer headed?

Structure from motion computations provide an important source of information for vision. Can’t say where the viewer is, but can say something about the relative depth relationships, and can say something about heading direction, and time to contact.

Calculating structure from motion and heading from the motion field

Estimation of relative depth and eye (or camera) motion

Introduction

Earlier we saw:
1) how local motion measurements constrain estimates of optic flow;
2) how a priori slowness and smoothness constraints constrain dense and sparse estimates of the flow field.

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?

Goals

Estimate relative depth, and eye’s motion from motion field estimates time-to-contact

Ultimately we would like to gain some understanding of the environment from the moving images on our retinas. There are approaches to structure from motion that are not based directly on the 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).

Alternatively, suppose we have estimated the optic flow, 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 camera or eye-point 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 derivation of Longuet-Higgins et al., we derive the relationship between the motion field and relative depth, and camera motion parameters using homogeneous coordinates.

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 \(Z, U,V,W,A,B,C\).

Express velocity \((X,Y,Z)\) of world point \(P\) in terms of motion of the frame of reference

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

\[
r(t) = (X, Y, Z, 1)
\]

An instant later, the new coordinates are given by:

\[
r(t + \Delta t) = r + \Delta r = (X + \Delta X, Y + \Delta Y, Z + \Delta Z, 1) = r\Delta R_0, \Delta R_0, \Delta R_0, \Delta T
\]

where infinitesimal rotations and translations are represented by their respective 4x4 matrices. (Note that matrix operations do not in general commute). Then,
Next step: Relate P velocity and depth Z to the motion field, \( (u, z) \)

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

and

\[
\Delta R_\theta = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\theta_z) & -\sin(\theta_z) & 0 \\
0 & \sin(\theta_z) & \cos(\theta_z) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Using similar approximations for the other rotation matrices, and the relation

\[
\Delta r = r\Delta R_{\theta_x} \Delta R_{\theta_y} \Delta R_{\theta_z} \Delta T - rI
\]

we have

\[
(\Delta x, \Delta y, \Delta z, 0) = (X, Y, Z, 1) \begin{bmatrix}
\Delta \theta_z & \Delta \theta_y & \Delta \theta_x & 0 \\
\Delta \theta_y & 0 & -\Delta \theta_x & 0 \\
0 & -\Delta \theta_x & 0 & 0 \\
-\Delta x_0 & -\Delta y_0 & -\Delta z_0 & 0
\end{bmatrix} + \text{higher order terms}
\]

By dividing by \( \Delta t \), we can derive the following relations:

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

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

\[
\frac{\Delta Z}{\Delta t} = \frac{\Delta z}{\Delta t} - \frac{\Delta \theta_x}{\Delta t} Y + \frac{\Delta \theta_y}{\Delta t} X = -W - AY + BX
\]

(APOLOGISTIC NOTE: The lower case \( x, y, z \) should have 0 subscripts to distinguish them from the \( x \) and \( y \) used below. Below we use \( (x, y) \) to represent the projection of \( (X, Y, Z) \). Will fix later!)

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. What is happening in the image—i.e. to the motion field or optic flow?

\[
\text{Next step: Relate } P \text{ velocity and depth } Z \text{ to the motion field, } (u, z)
\]

\[
\text{Main result for goal 1, the generative model:}
\]

Substituting the expressions for the rate of change of \( X, Y, \) and \( Z \), we have:

\[
u = \left( \frac{-U + WX}{Z} \right) + \left( -B + Cy + Ax y - Bx^2 \right)
\]

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

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, and the second term in brackets is purely rotational, and further does not depend on \( Z \):

\[ u_R = \left( -B + Cy + AxY - Bx^2 \right) \]

So in general, we can write:

\[ u = u_T + u_R \]

\[ v = v_T + v_R \]

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.

**Diversion: Using Mathematica to derive structure from motion and heading equations**

Here is a start. I'll leave it to the reader to prove the rest of Longuet-Higgins and Padeny's results using Mathematica to manipulate homogeneous coordinates.

![Flow fields](image)

Recall the `Series[]` function:

\[ \text{Series}[XRotationMatrix[\theta], \{\theta, 0, 1\}] \]

\[ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1+O(\theta)^2 & -\theta + O(\theta)^2 & 0 \\ 0 & -\theta + O(\theta)^2 & 1+O(\theta)^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

Use `Normal[]` to chop off higher order terms:

\[ \text{Normal[Series[XRotationMatrix[\theta], \{\theta, 0, 1\}]]} \]

\[ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

Translational matrix is:

\[ \text{TranslateMatrix[-x0, -y0, -z0]} \]

\[ \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

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

**Known rotation, estimate translation.**

First, suppose we know the rotational component. 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.

\[
\begin{align*}
    u^T &= -\frac{U + xW}{Z} \\
    v^T &= -\frac{V + yW}{Z}
\end{align*}
\]

Combining these two equations to eliminate \(Z\):

\[
(y - \frac{V}{W}) = (x - \frac{U}{W}) \frac{v^T}{u^T}
\]

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

**Unknown rotation: Estimate both rotation and translational components.**

What if we don't know the rotational component? 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:

\[
u_1 - u_2 = (-U + xW) \left( \frac{1}{Z_1} - \frac{1}{Z_2} \right)
\]
\[ v_1 - v_2 = (-V + yW)(\frac{1}{Z_1} - \frac{1}{Z_2}) \]

\[ (y - \frac{V}{W}) = (\frac{v_1 - v_2}{u_1 - u_2})(x - \frac{U}{W}) \]

Again, finding the focus of expansion \((x_0, y_0)\), which involves finding at least two motion parallax pairs,

\[
\begin{align*}
\begin{array}{ccc}
P_1 & P_2 & P_3 \ P_4 \\
\end{array}
\end{align*}
\]

will give us the camera direction

\[ x_0 = \frac{U}{W} \]
\[ y_0 = \frac{V}{W} \]

To find relative depth, we need to know \(A, B, C\).

\[
\begin{align*}
u(x_0, y) &= (C + Ax_0)y - B(1 + x_0^2) \\
v(x_0, y_0) &= -(C + By_0)x + A(1 + y_0^2) \\
u &= (x - x_0)\frac{W}{Z} + u^R; \ v &= (y - y_0)\frac{W}{Z} + v^R \\
u - u^R &= (x - x_0)\frac{W}{Z} \\
v - v^R &= (y - y_0)\frac{W}{Z} \\
\frac{Z}{W} &= \frac{x - x_0}{u - u^R} = \frac{y - y_0}{v - v^R}
\end{align*}
\]

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. See also Royden, Banks & Crowell (1992) for the possible role of proprioceptive information in heading computation.

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). See Duffy (2000) for recent work.

Challenges to computational theories of structure from motion

Depth between objects

- Depth from shadows

http://vision.psych.umn.edu/www/kersten-lab/demos/shadows.html

Depth from viewer

- Shadows

Frame of reference issues in cue integration
References


© 2000 Daniel Kersten, Computational Vision Lab, Department of Psychology, University of Minnesota. (http://vision.psych.umn.edu/www/kersten-lab/kersten-lab.html)

Structure from motion, heading


© 2000 Daniel Kersten, Computational Vision Lab, Department of Psychology, University of Minnesota. (http://vision.psych.umn.edu/www/kersten-lab/kersten-lab.html)