CS4733 Class Notes, Stereo Imaging

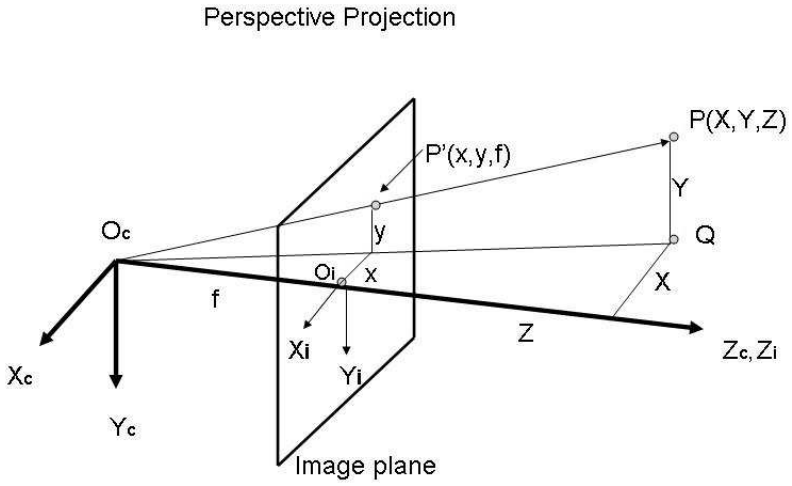


Figure 1: Perspective imaging geometry showing relationship between 3D points and image plane points.

1 Stereo Imaging: Camera Model and Perspective Transform

We typically use a pinhole camera model that maps points in a 3-D camera frame to a 2-D projected image frame. In figure 1, we have a 3D camera coordinate frame X_c, Y_c, Z_c with origin O_c, and an image coordinate frame X_i, Y_i, Z_i with origin O_i. The focal length is f. Using similar triangles, we can relate image plane and world space coordinates. We have a 3D point $P = (X, Y, Z)$ which projects onto the image plane at $P' = (x, y, f)$. O_c is the origin of the camera coordinate system, known as the *center of projection* (COP) of the camera.

Using similar triangles, we can write down the folowing relationships:

$$\frac{X}{x} = \frac{Z}{f} \quad ; \quad \frac{Y}{y} = \frac{Z}{f} \quad ; \quad x = f \cdot \frac{X}{Z} \quad ; \quad y = f \cdot \frac{Y}{Z}$$

If $f = 1$, note that perspective projection is just scaling a world coordinate by its Z value. Also note that all 3D points along a line from the COP through a designated position (x, y) on the image plane will have the same image plane coordinates.

We can also describe perspective projection by the matrix equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} s \cdot x \\ s \cdot y \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where s is a scaling factor and $[x, y, 1]^T$ are the projected coordinates in the image plane.

We can generate *image space* coordinates from the projected camera space coordinates. These are the actual pixels values that you use in image processing. Pixels values (u, v) are derived by scaling the camera image plane coordinates in the x and y directions (for example, converting mm to $pixels$), and adding a translation to the origin of the image space plane. We can call these scale factors D_x and D_y, and the translation to the origin of the image plane as (u_0, v_0).

If the pixel coordinates of a projected point (x,y) are (u,v) then we can write:

$$\frac{x}{D_x} = u - u_0; \quad \frac{y}{D_y} = v - v_0;$$

$$u = u_0 + \frac{x}{D_x}; \quad v = v_0 + \frac{y}{D_y}$$

where D_x, D_y are the physical dimensions of a pixel and (u_0, v_0) is the origin of the pixel coordinate system. $\frac{x}{D_x}$ and $\frac{y}{D_y}$ are simply the number of pixels, and we center them at the pixel coordinate origin. We can also put this into matrix form as:

$$\begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s \cdot x \\ s \cdot y \\ s \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$P^{image} = T^{image}_{persp} T^{persp}_{camera} P^{camera}$$

In the above, we assumed that the point to be imaged was in the camera coordinate system. If the point is in a previously defined world coordinate system, then we also have to add in a standard $4x4$ transform to express the world coordinate point in camera coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^wX \\ {}^wY \\ {}^wZ \\ 1 \end{bmatrix}$$

$$P^{image} = T^{image}_{persp} T^{persp}_{camera} T^{camera}_{world} P^{world}$$

Summing all this up, we can see that we need to find the following information to transform an arbitrary 3D world point to a designated pixel in a computer image:

- 6 parameters that relate the 3D world point to the 3D camera coordinate system (standard 3 translation and 3 rotation): (R, T)
- Focal Length of the camera: f
- Scaling factors in the x and y direcitons on the image plane: (D_x, D_y)
- Translation to the origin of the image plane: (u_0, v_0).

  This is 11 parameters in all. We can break these parameters down into *Extrinsic* parameters which are the 6-DOF transform between the camera coordinate system and the world coordinate system, and the *Intrinsic* parameters which are unique to the actual camera being used, and include the focal length, scaling factors, and location of the origin of the pixel coordinate system.

2 Camera Calibration

Camera calibration is used to find the mapping from 3D to 2D image space coordinates. There are 2 approaches:

- Method I: Find both extrinsic and intrinsic parameters of the camera system. However, this can be difficult to do. The instinsic parameters of the camera may be unknown (i.e. focal length, pixel dimension) and the 6-DOF transform also may be difficult to calculate directly.
- Method 2: An easier method is the "Lumped" transform. Rather than finding individual parameters, we find a composite matrix that relates 3D to 2D. Given the equation below:

  $$P^{image} = T^{image}_{persp} T^{persp}_{camera} T^{camera}_{world} P^{world}$$

  we can lump the 3 T matrices into a 3x4 calibration matrix C:

  $$P^{image} = C\, P^{world}$$

  $$C = T^{image}_{persp} T^{persp}_{camera} T^{camera}_{world}$$

- C is a single 3×4 transform that we can calculate empirically.

$$\overbrace{\begin{bmatrix} C \end{bmatrix}}^{3\times4} \quad \underbrace{\overbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}^{4\times1}}_{\text{3-D homo. vec}} = \underbrace{\overbrace{\begin{bmatrix} u \\ v \\ w \end{bmatrix}}^{3\times1}}_{\text{2-D homo. vec}} \triangleq \underbrace{\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}}_{\text{Pixels}} \quad \begin{matrix} where \\ u' = \frac{u}{w} \\ v' = \frac{v}{w} \end{matrix}$$

- Multiplying out the equations, we get:

$$c_{11}x + c_{12}y + c_{13}z + c_{14} = u$$

$$c_{21}x + c_{22}y + c_{23}z + c_{24} = v$$

$$c_{31}x + c_{32}y + c_{33}z + c_{34} = w$$

- Substituting $u = u'w$ and $v = v'w$, we get:

1. $c_{11}x + c_{12}y + c_{13}z + c_{14} = u'(c_{31}x + c_{32}y + c_{33}z + c_{34})$
2. $c_{21}x + c_{22}y + c_{23}z + c_{24} = v'(c_{31}x + c_{32}y + c_{33}z + c_{34})$

- How to interpret 1 and 2:

1. If we know all the c_{ij} and x, y, z, we can find u', v'. This means that if we know calibration matrix C and a 3-D point, we can predict its image space coordinates.
2. If we know $x,\ y,\ z,\ u',\ v'$, we can find c_{ij}. Each 5-tuple gives 2 equations in c_{ij}. This is the basis for empirically finding the calibration matrix C (more on this later).
3. If we know $c_{ij},\ u',\ v'$, we have 2 equations in x, y, z. They are the equations of 2 planes in 3-D. 2 planes form an intersecton which is a line. These are the equations of the line emanating from the center of projection of the camera, through the image pixel location u', v' and which contains point x, y, z.

- We can set up a linear system to solve for c_{ij}: $AC = B$

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u'_1x & -u'_1y & -u'_1z \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v'_1x & -v'_1y & -v'_1z \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u'_2x & -u'_2y & -u'_2z \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v'_2x & -v'_2y & -v'_2z \\ . & & & & & & & & & & \\ . & & & & & & & & & & \\ . & & & & & & & & & & \\ . & & & & & & & & & & \\ . & & & & & & & & & & \\ . & & & & & & & & & & \\ . & & & & & & & & & & \end{bmatrix} \underbrace{\begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix}}_{\text{We can assume } c_{34}=1} = \begin{bmatrix} u'_1 \\ v'_1 \\ u'_2 \\ v'_2 \\ u'_3 \\ v'_3 \\ . \\ . \\ . \\ u'_N \\ v'_N \end{bmatrix}$$

- Each set of points x, y, z, u', v' yields 2 equations in 11 unknowns (the c_{ij}'s).
- To solve for C, A needs to be invertible (square). We can overdetermine A and find a Least-Squares fit for C by using a pseudo-inverse solution.

  If A is $N \times 11$, where $N > 11$,

$$AC = B$$
$$A^TAC = A^TB$$
$$C = \underbrace{(A^TA)^{-1}}_{\text{pseudo inverse}} A^TB$$

3 COMPUTATIONAL STEREO

Stereopsis is an identified human vision process. It is a passive, simple procedure that is robust to changes in lighting, scale, etc. Humans can fuse random dot stereograms that contain no high-level information about the objects in the fused images, yet they can infer depth from these stereograms. The procedure is:

- Camera-Modeling/Image-acquisition
- Feature extraction - identify edges, corners, regions etc.
- Matching/Correspondence - find same feature in both images
- Compute depth from matches - use calibration information to back project rays from each camera and intersect them (triangulation)

- Interpolate surfaces - Matches are sparse, and constraints such as smoothness of surfaces are needed to "fill in" the depth between match points.

Camera Modeling: An important consideration in computational stereo is the setup of the cameras. The **baseline** between the camera centers determines the accuracy of the triangulation. Large baseline means more accuracy; however as the baseline gets larger, the same physical event in each image may not be found.

The cameras also have to be calibrated and registered. Calibration is relatively straightforward, and a variety of methods exist. Some methods extend the simple least squares model we discussed to include non-linear effects of lens distortion (particularly true with short a focal length lens).

Registration is needed to make use of the epipolar constraint. This constraint consists of a plane that includes both camera's optical centers and a point in 3-D space. This **epilolar plane** intersects both image planes in a straight line.

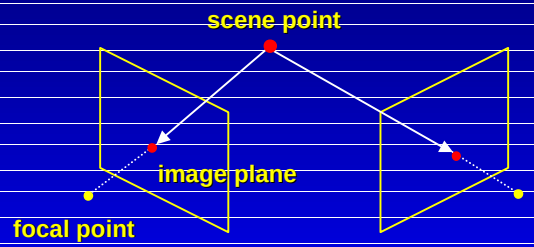
Feature Extraction: Identifying features in each image that can be matched is an important part of the stereo process. It serves 2 purposes: 1) data reduction so we are not forced to deal with every single pixel as a potential match, and 2) stability - features are seen to be more stable than a single gray level pixel.

There are 2 approaches: feature-based methods which find primitives such as edges, corners, lines, arcs in each image and match them; and area-based methods that identify regions or areas of pixels that can be matched using correlation based methods. Sometimes both methods are used, with feature-based methods proposing a match and area-based methods centered on the feature used to verify it.

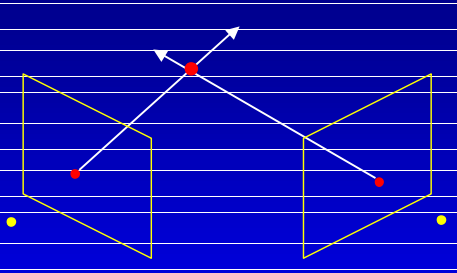
Correspondence: The heart of the stereo problem is a search procedure. Given a pixel in image 1, it can potentially match each of N^2 pixels in the other image. To cut down this search space, cameras are often registered along scan lines. This means that he epipolar plane intersects each image plane along the same scan line. A pixel in image 1 can now potentially match only a pixel along the corresponding scan line in image 2, reducing the search from $O\ (\ N^2\ )$ to $O\ (\ N\ )$. The match criteria can include not only the location of a feature like an edge, but also the edge direction and polarity.

Problems in Matching: A number of problems occur during matching to create false matches: These are occlusions, periodic features such as texture, homogeneous regions without features, baseline separation errors, and misregistered images. Stereo can usually only provide sparse 3-D data at easily identified feature points.

Stereo



Stereo



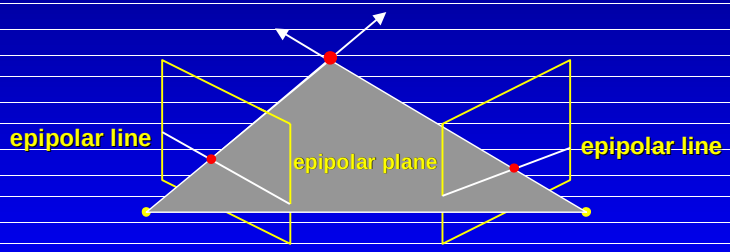
Basic Principle: Triangulation

- Gives reconstruction as intersection of two rays
- Requires ***point correspondence***

Stereo Correspondence

Determine Pixel Correspondence

- **Pairs of points that correspond to same scene point**



Epipolar Constraint

- **Reduces correspondence problem to 1D search along *conjugate epipolar lines***

Stereo Matching Algorithms

Match Pixels in Conjugate Epipolar Lines

- **Assume color of point does not change**
- **Pitfalls**
  - > specularities
  - > low-contrast regions
  - > occlusions
  - > image error
  - > camera calibration error
- **Numerous approaches**
  - > dynamic programming [Baker 81,Ohta 85]
  - > smoothness functionals
  - > more images (trinocular, N-ocular) [Okutomi 93]
  - > graph cuts [Boykov 00]

Zero Crossing of Laplacian of Gaussian

- Identification of features that are stable and match well
- Laplacian of intensity image

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

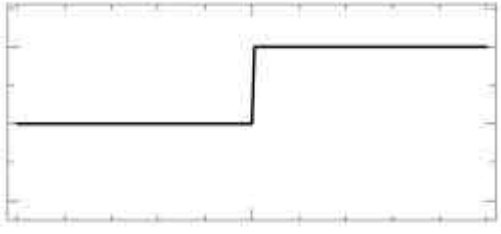
- Convolution with P:

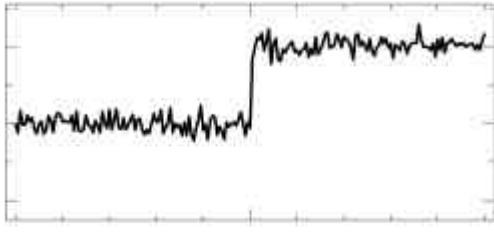
$$L = P \otimes I$$

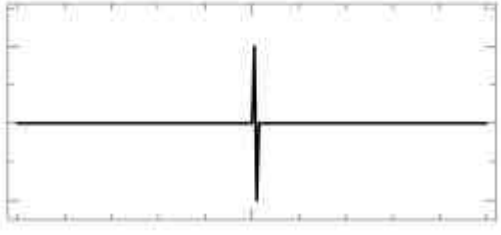
$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

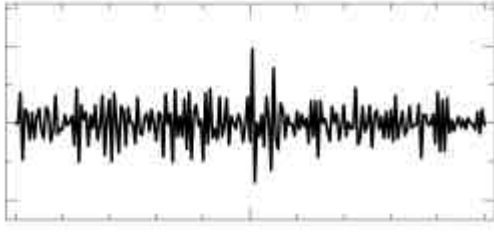
- Step / Edge Detection in Noisy Image
  - *filtering through Gaussian smoothing*

$$\begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$









Stereo Vision Example

- Extracting depth information from a stereo image

  - *a1 and a2: left and right image*

  - *b1 and b2: vertical edge filtered left and right image;*
    filter = [1 2 4 -2 -10 -2 4 2 1]

  - *c: confidence image:*
    bright = high confidence (good texture)

  - *d: depth image:*
    bright = close; dark = far