Administrivia

Slides now being posted:
http://www.cs.arizona.edu/classes/cs477/spring08/lectures
Need to connect via a UA machine or use id (“me”; pw=“vision4fun”).

Lectures and assignments will require either connecting from a UA machine, OR
login id (“me”) and password (“vision4fun”).

Office hours (**tentative**) on T/R 5-5:30 and Friday 11:30-12 (not every week) by electronic signup.
login id (“public”) and password (“meetkobus”).
Also, if you coming from off campus, you will need
login id (“me”) and password (“pw4cal”) to start.

Eight machines in 9th floor lab (gr01-gr08) will be available for this course (only).

Matlab Tricks

Matlab is good for quick experiments (use it!).

Good way to check C code.

Good way to check that the math does what you expect. Exploit “rand”.

```
x = rand(5,3)
y= x`*x              % pos-def
[a,b]=eig(y)         % positive e-vals
a`*a                 % check orthogonal
a*b*a' - y           % check decomp
```

```
x = rand(4)
[a,b]=eig(x)         % likely complex
y=x`+x               % symmetric
[a,b]=eig(y)         % real e-vals
```

Very Brief Course Intro

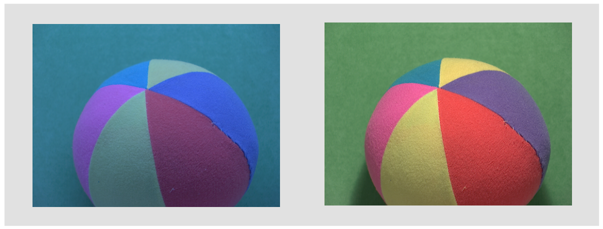
(Subject to change)

Part* I: How images are formed

- Image abstraction level is a single pixel
- Cameras
  - What a camera does
- Light
  - How to measure light
  - What light does at surfaces
  - How the brightness values we see in cameras are determined
- Color

* Here “part” refers to the book section. We cannot do them all in detail.

The Computational Colour Constancy Problem



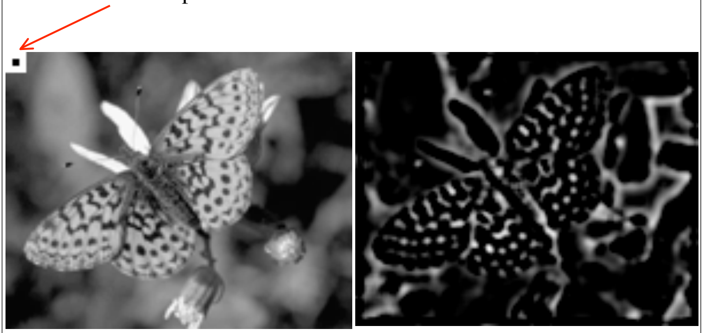
(Same scene, but different illuminant)

Part II: Early Vision in One Image

- Representing small patches of image
  - Often want to match points in different images, so we need to describe the neighborhood of the points (e.g for stereo)
  - Sharp changes are important in practice --- known as “edges”
  - Representing texture by statistics of local structure.
    - Zebras have lots of bars, few spots, leopards are the other way

Filters as templates

Filters respond to structures that “look” like the filter

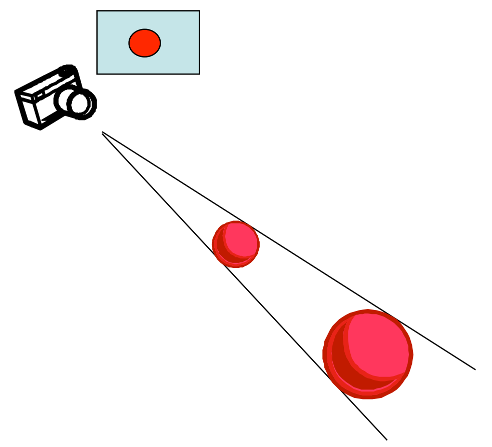


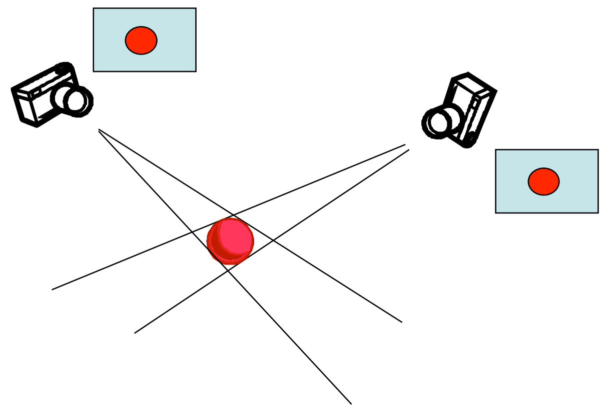
Texture

- Many objects are distinguished by their texture
  - Tigers, cheetahs, grass, trees
- We represent texture with statistics of filter outputs
  - For tigers, bar filters at a coarse scale respond strongly
  - For cheetahs, spots at the same scale
  - For grass, long narrow bars
  - For the leaves of trees, extended spots
- Objects with different textures can be segmented
- The variation in textures is a cue to shape

Part III: Early Vision in Multiple Images

- (May not have time in 2008)
- The geometry of multiple views
  - Two views of the same patch reveal geometry (if you can **match** them---“correspondence”)
- Stereo vision, structure from motion



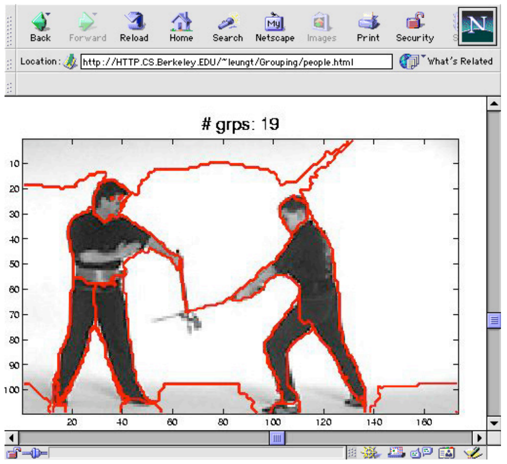


Part IV: Mid-Level Vision

- Finding coherent structure so as to break the image or movie into big units
  - Segmentation:
    - Breaking images and videos into useful pieces
      - finding video sequences that correspond to one shot
      - finding image components that are coherent in internal appearance
  - Tracking:
    - Keeping track of a moving object through a sequence of views
    - Use a model to predict next position and refine using next image

Grouping

- Which image components "belong together"?
- Belong together == lie on the same object
- Cues
  - similar colour
  - similar texture
  - not separated by contour
  - form a suggestive shape when assembled
  - move together



High Level Vision

- Visual information --> semantics
- Object recognition
  - Specific object (my car)
  - Object category (a car)
- Object / scene understanding
  - What are the pose (orientation) and location of scene objects
  - What about the object structure (very leafy plant?)
  - What might an object be good for (a step?)
- Activity recognition

High Level Vision

- Many applications
  - Digital libraries
    - Find me the picture of JFK and Marilyn Monroe embracing
  - Surveillance
    - Warn me if there is a mugging in the grove
  - HCI
    - Do what I show you
  - Military
    - Shoot this, not that

What are the problems in recognition?

- Variable appearance
  - Objects appear different due to pose, illumination, occlusion, etc.
- Which bits of image should be recognized together?
  - Segmentation (issue--segment first or recognize first)
- How can objects be recognized without focusing on detail?
  - Levels of abstraction (object categories vs individuals)

Part V: High Level Vision (Geometry)

- We will not do much of this part in 2008
- The relations between object geometry and image geometry
- One difficulty in recognition is pose change
  - One approach is model based vision
    - find the position and orientation of known objects
  - Another approach is to use descriptors which are invariant to (small) pose changes

Part VI: High Level Vision (Probabilistic)

- Using probabilistic models and classifiers to recognize objects
  - Probabilistic methodology
    - Bayes rule: p(object | data) ~ p(data | object) p(object)
  - Templates and classifiers
    - Find objects from a canonical view (e.g. frontal face) by matching a template
      - best when combined with a probabilistic classifier
      - transformed views require looking with more templates (e.g. rotated, scaled, faces)
  - Relations
    - find the parts with a classifier, and then reason about the relationships between the parts to find the object.

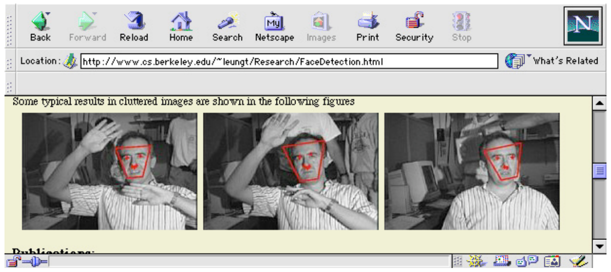
Matching templates

- Some objects are 2D patterns
  - e.g. faces
- Build an explicit pattern matcher
  - need to discount changes in illumination
  - changes in background are hard
  - changes in pose are hard

http://www.ri.cmu.edu/projects/project_271.html

Relations between templates

- e.g. find faces by
  - finding eyes, nose, mouth
  - finding an assembly of the three that has the "right" relations



Location: http://www.cs.berkeley.edu/~leungt/Research/FaceDetection.html

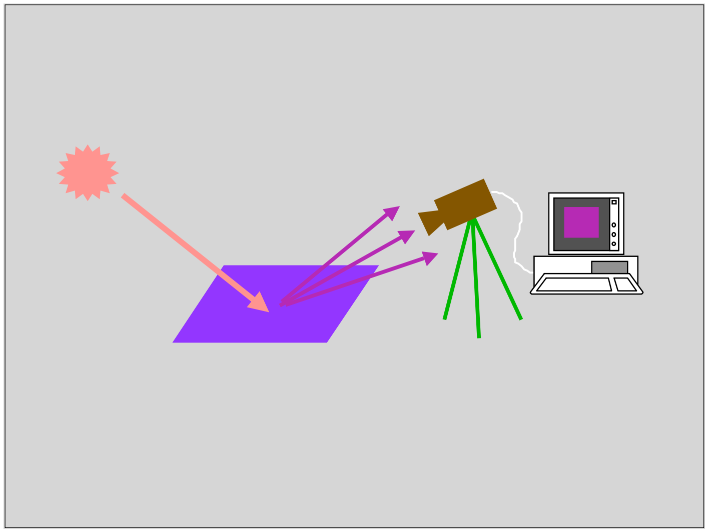
Some typical results in cluttered images are shown in the following figures

Official Start of Course

Image Formation

§1 (focus on §1.1.1, §1.4.2), highlights of §4

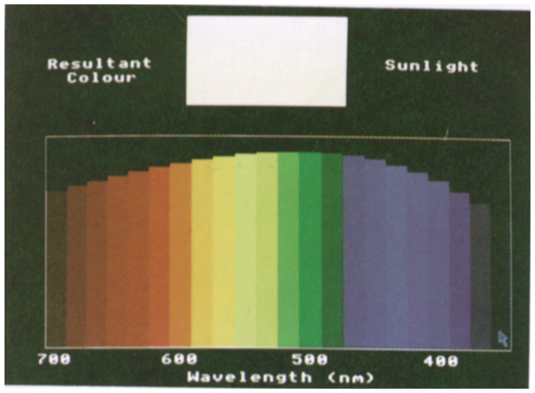
- Images
  - spatial sampling
    - typically of light on a plane
    - typically encoded as an array of values
  - carry information about the world when light has interacted with it
- Image formation is essentially at the level of a "pixel"
- We will briefly study in turn
  - light and its interaction with the world
  - camera geometry
  - camera spectral response



Light

- Geometrically approximate light by rays (vectors)
- Typical scene has light going in a multitude of directions
- A bit of light has additional characteristics (energy/wavelength, polarization)
- The light in a certain direction is a mix, so we get a "spectrum" over wavelengths
- Spectrum records how much power is at each wavelength
- Visible portion is about 400 to 700 nm (Certain applications may require modeling some IR and/or UV also).

Example---sunlight spectra



Two disparate source spectra

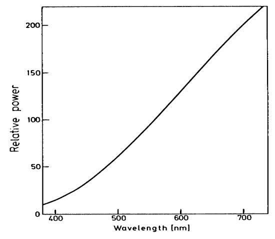


Fig. 4.1. Wavelength composition of light from a tungsten-filament lamp [typified by CIE ILL A (Sect. 4.6)]. Relative spectral power distribution curve. Color temperature: 2856 K

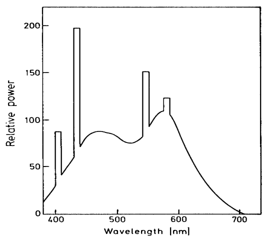
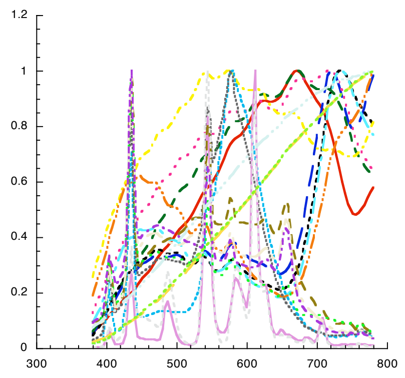


Fig. 4.2. Wavelength composition of light from a daylight fluorescent lamp. Typical relative spectral power distribution curve. Correlated color temperature: 6000 K. (Based on data of Jerome reported in [Ref. 3.14, p. 37])

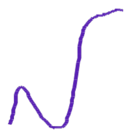
Spectra of many sources



Light (summary)

Light energy reaching a camera sensor has a distribution over wavelength, λ.

(*Recall from physics that wavelength is inversely related to photon energy)



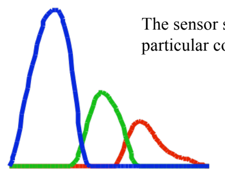
<--- wavelength --->

* Things marked with red stars are optional comments in the current context. If cover them in more detail later on, then they might not be optional anymore.

Sensors

Sensors (including those in your eyes) have a varied sensitivity over wavelength

Different variations lead to different kinds of sensor responses (“colors” in a naïve sense)



The sensor sensitivity spectra for a particular color camera (they vary a lot).

<--- wavelength --->

Math warm-up --- linear operators

h(x) is a linear operator means that ?

Math warm-up --- linear operators

h(x) is a linear operator means that

$$h(a*x+b*y) = a*h(x) + b*h(Y)$$

Examples?

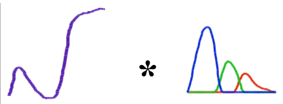
Math warm-up --- linear operators

Examples of linear operators

Discrete: $h(\mathbf{x})=\mathbf{k}\cdot\mathbf{x}$
(**x** and **k** are vectors)

Continuous: $h(f) = \int k(x) f(x) dx$

Image Formation (Spectral)

$$(\color{red}{R},\color{green}{G},\color{blue}{B}) = \int_{380}^{780}$$  $$d\lambda$$

On the next page, ($\color{red}{R}$,$\color{green}{G}$,$\color{blue}{B}$) is the row vector, □, with elements $□^{(k)}$. So, $\color{red}{R}=□^{(1)}$, $\color{green}{G}=□^{(2)}$, $\color{blue}{B}=□^{(3)}$.

More formally,

The response of an image capture system to a light signal $L(\lambda)$ associated with a given pixels is modeled by

$$\rho^{(k)} = F^{(k)}(\rho^{(k)}) = F^{(k)}\left(\int L(\lambda)R^{(k)}(\lambda)d\lambda\right)$$

where $R^{(k)}(\lambda)$ is the sensor response function for the kth channel and $\rho^{(k)}$ is the kth channel response.

In this formulation, $R^{(k)}(\lambda)$ includes the contributions due to the aperture, focal length, sensor position in the focal plane.

$F^{(k)}$ absorbs typical non-linearities such as gamma.

Important

Discrete Version

Often we represent functions by vectors. For example, a spectra might be represented by 101 samples in the range of 380 to 780 nm in steps of 4nm.

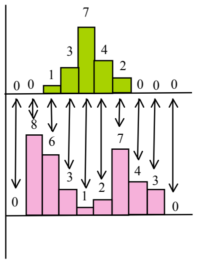
Then $L(\lambda)$ becomes the vector **L**, $R^{(k)}(\lambda)$ becomes the vector $\mathbf{R}^{\mathbf{k}}$, and the response (ignoring non-linearity issues) is given by a dot product:

$$\rho^{(k)} = \mathbf{L} \bullet \mathbf{R}^{(\mathbf{k})}$$

Important

Sensor/light interaction example

R=(0,0,1,3,7,4,2,0,0,0)



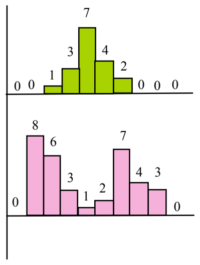
L=(0,8,6,3,1,2,7,4,3,0)

Multiply lined up pairs of numbers and then sum up

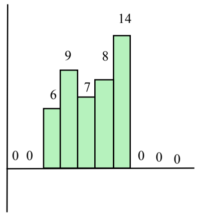
Important

Sensor/light interaction example

R=(0,0,1,3,7,4,2,0,0,0)



L=(0,8,6,3,1,2,7,4,3,0)



L .* R=
(0*0, 0*8, 1*6, 3*3, 7*1, 4*2, 2*7, 0*4, 0*3, 0*0)
=(0,0,6,9,7,8,14,0,0,0)

L • P = 0 + 0 + 6 + 9 + 7 + 8 + 14
= 44

Image Formation (Spectral)

- Note that (ignoring $F^{(k)}$) image formation is linear.
- Formally this means **if**:

  $L_1(\lambda) \rightarrow \rho_1^{(k)}$ and $L_2(\lambda) \rightarrow \rho_2^{(k)}$
- **Then**:

  ?

Image Formation (Spectral)

- Note that (ignoring $F^{(k)}$) image formation is linear.
- Formally this means **if**:

  $L_1(\lambda) \rightarrow \rho_1^{(k)}$ and $L_2(\lambda) \rightarrow \rho_2^{(k)}$
- **Then**:

  $aL_1(\lambda) + bL_2(\lambda) \rightarrow a\rho_1^{(k)} + b\rho_2^{(k)}$

Image Formation (Spectral)

- Note that image formation loses spectral information
- Technically, the process is a projection
- This means that two **very** different spectra can map into the same color
- This is the key to color reproduction

Supplemental material

Image Formation (Spectral)

$F^{(k)}$ is often ignored (assumed to be the identity), but this is not a safe assumption, especially when color or radiometric measurements matter. To compensate for the non-linearity of CRT display monitors, camera manufactures will "gamma" correct the signal, typically by raising the signal (normalized to [0,1]) to the power 1/(2.2).

Note that in such an image, a number twice as large does not mean that the light had twice the power!

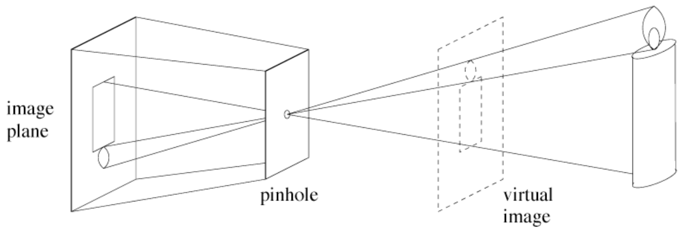
To linearize RGB's from such a signal we compute:

$p=F^{-1}(v)=255*(v/255)$^2.2

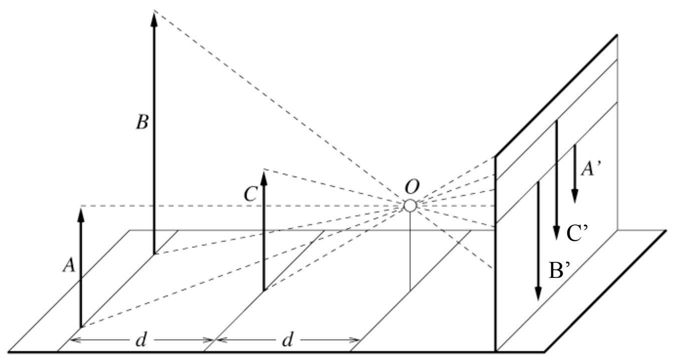
Image Formation (Geometric)

Pinhole cameras

- Abstract camera model--box with a small hole in it
- Pinhole cameras work for deriving algorithms--a real camera needs a lens

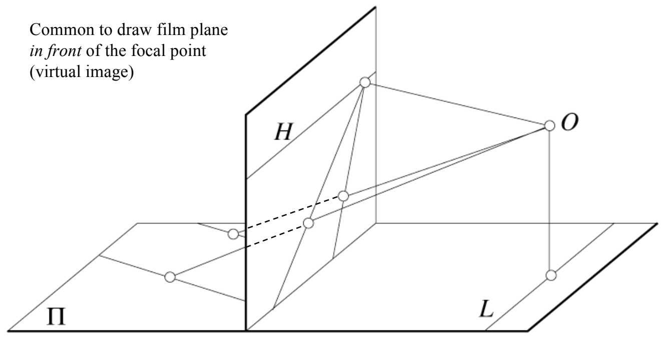


Distant objects are smaller



Parallel lines meet*

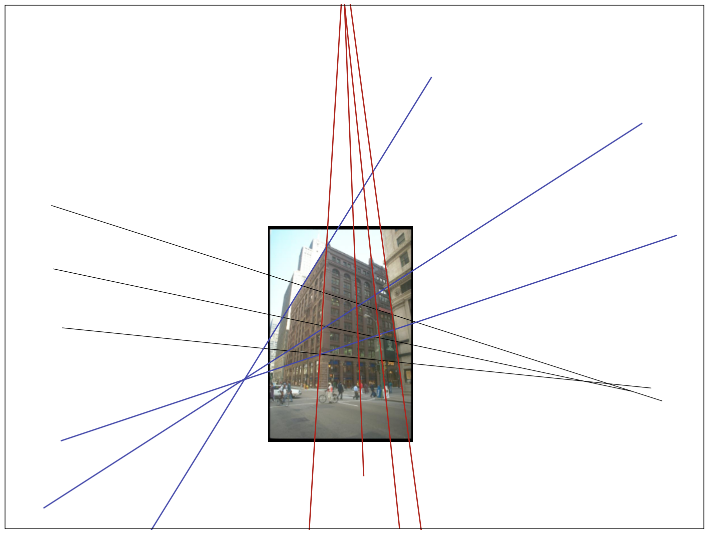
Common to draw film plane *in front* of the focal point (virtual image)

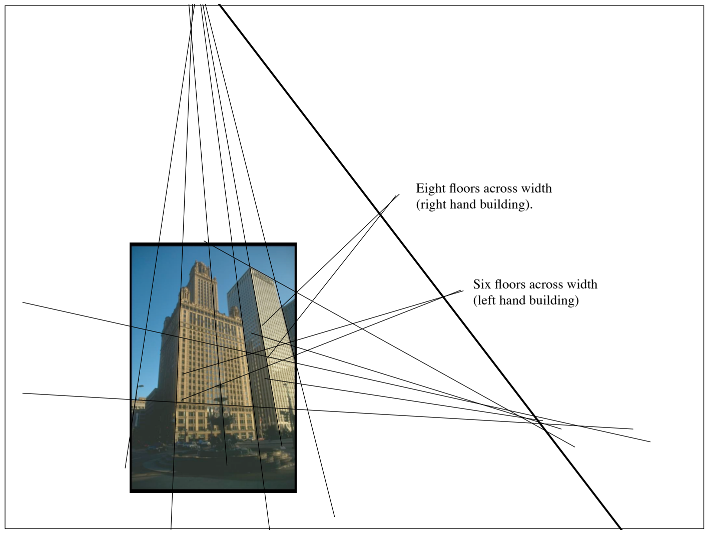


*Exceptions?

Vanishing points

- Each set of parallel lines (=direction) meets at a different point
  - The *vanishing point* for this direction
- Sets of parallel lines on the same plane lead to *collinear* vanishing points.
  - The line is called the *horizon* for that plane
  - Standard horizon is the horizon of the ground plane.
- One way to spot fake images
  - scale and perspective don't work
  - vanishing points behave badly





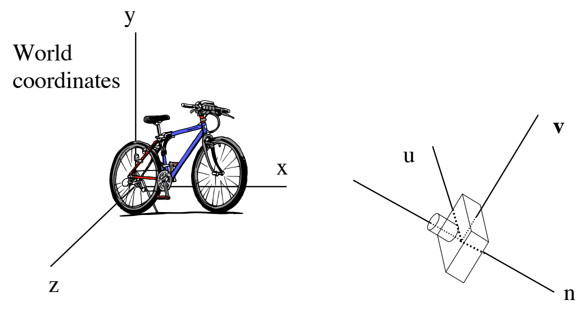
Geometric properties of projection

- Points go to points
- Lines go to lines
- Polygons go to polygons
- Degenerate cases
  - line through focal point projects to a point
  - plane through focal point projects to a line

Geometric Camera Model

- Transform world coordinates to standard camera coordinates
  - (Extrinsic parameters)
- Project onto standard camera plane
  - (3D becomes 2D)
- Transform into pixel locations
  - (Intrinsic camera parameters)

World and camera coordinates



Math aside, #2

Representing Transformations

- Need mathematical representation for mapping points from the world to an image (and later, from an image taken by one camera to another).
- Represent linear transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- To transform line segments, transform endpoints
- To transform polygons, transform vertices

2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- Recall the definition of matrix times vector:

$$\begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrix multiplication is linear

- In particular, if we define f(**x**)=M • **x**, where M is a matrix and **x** is a vector, then

?

Matrix multiplication is linear

- In particular, if we define f(**x**)=M • **x**, where M is a matrix and **x** is a vector, then

$$\begin{aligned} f(a\mathbf{x}+b\mathbf{y}) &= M(a\mathbf{x}+b\mathbf{y}) \\ &= aM\mathbf{x}+bM\mathbf{y} \\ &= af(\mathbf{x})+bf(\mathbf{y}) \end{aligned}$$

- Where the middle step can be verified using algebra (next slide)

Supplemental material

Proof that matrix multiplication is linear

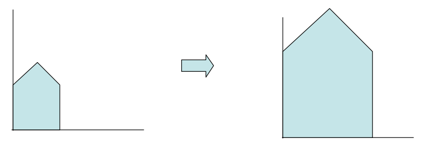
$$\begin{aligned} M(a\mathbf{x}+b\mathbf{y}) &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} ax_1+by_1 \\ ax_2+by_2 \end{bmatrix} \\ &= \begin{bmatrix} a_{11}ax_1+a_{11}by_1+a_{12}ax_2+a_{12}by_2 \\ a_{21}ax_1+a_{21}by_1+a_{22}ax_2+a_{22}by_2 \end{bmatrix} \\ &= \begin{bmatrix} a_{11}ax_1+a_{12}ax_2+a_{11}by_1+a_{12}by_2 \\ a_{21}ax_1+a_{22}ax_2+a_{21}by_1+a_{22}by_2 \end{bmatrix} \\ &= a\begin{bmatrix} a_{11}x_1+a_{12}x_2 \\ a_{21}x_1+a_{22}x_2 \end{bmatrix} + b\begin{bmatrix} a_{11}y_1+a_{12}y_2 \\ a_{21}y_1+a_{22}y_2 \end{bmatrix} \\ &= aM\mathbf{x}+bM\mathbf{y} \end{aligned}$$

Composition of Transformations

- If we use one matrix, M_1 for one transform and another matrix, M_2 for a second transform, then the matrix for the first transform followed by the second transform is simply $M_2 M_1$
- This follows from the associativity of matrix multiplication
  - $M_2 (M_1\mathbf{x}) = (M_2 M_1)\mathbf{x}$
- This generalizes to any number of transforms

Transformation examples in 2D

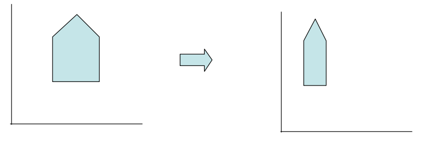
- Scale (stretch) by a factor of k



$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$ (k = 2 in the example)

Transformation examples in 2D

- Scale by a factor of (S_x, S_y)



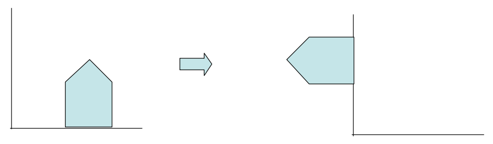
$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$ (Above, $S_x = 1/2$, $S_y = 1$)

Orthogonal Transformations

- Orthogonal transformations are defined by $O^TO=I$
- Also have $|\det(O)|=1$
- Rigid body rotations and flips

Transformation examples in 2D

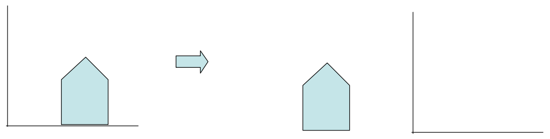
- Rotate around origin by θ (Orthogonal)



$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$ (Above, $\theta=90^o$)

Transformation examples in 2D

- Flip over y axis (Orthogonal)

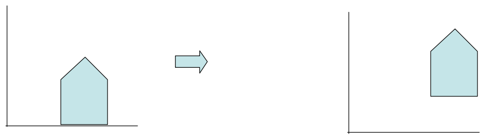


$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

Flip over x axis is ?

2D Transformations

- Translation ($\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$)



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now represent translations by matrix multiplications

2D Scale in H.C.

$$M = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Rotation in H.C.

$$M = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Translation in H.C.

- $\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$
- $(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

Transformations in 3D

- Homogeneous coordinates now have four components - traditionally, (x, y, z, w)
  - ordinary to homogeneous: (x, y, z) -> (x, y, z, 1)
  - homogeneous to ordinary: (x, y, z, w) -> (x/w, y/w, z/w)
- Again, translation can be expressed as a multiplication.

Transformation examples in 3D

- Translation:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Anisotropic scaling:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformation examples in 3D

- Rotation about x-axis:

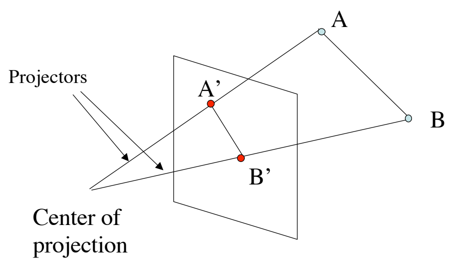
$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

- See supplementary material for rotation about an arbitrary axis.
- A rotation matrix can be thought of as either a rotation about an axis, or a rigid transformation represented by an orthogonal matrix

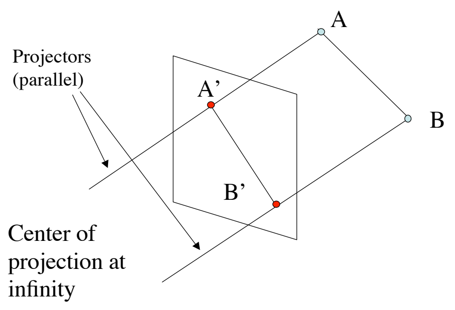
Projections

- Want to think about geometric image formation as a mathematical transformation taking points in the 3D world and mapping them into an image plane.
- Mathematical definition of a projection: PP=P
- (Doing it a second time has no effect).
- Generally rank deficient (non-invertable)--exception is P=I
- Transformation looses information (e.g., depth)
- Given a 2D image, there are many 3D worlds that could have lead to it.

Projections



Parallel Projection

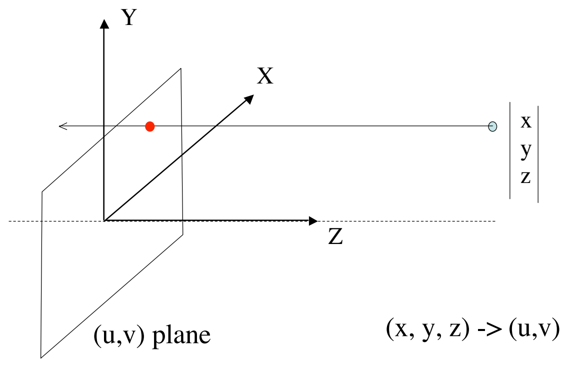


Parallel Projection

Parallel lines remain parallel, some 3D measurements can be made using 2D picture

If projection plane is perpendicular to projectors the projection is orthographic

Orthographic example (onto z=0)



The equation of projection (orthographic, onto z=0)

- In homogeneous coordinates

  $$(x,\ y,\ z,\ 1) \rightarrow (x,\ y,\ 1)$$

- Graphics course survivors: You will notice slight changes in style to be consistent with the book. Perhaps most notably we will explicitly, rather than implicitly, ignore the third projected coordinate, so projection matrices will be 3 by 4 not 4 by 4.

The projection matrix

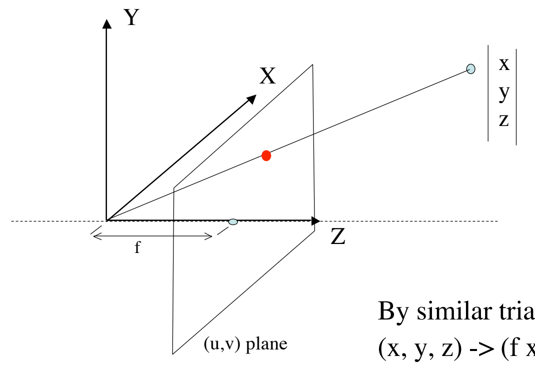
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix} ? \begin{bmatrix} \\ \\ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$(x,\ y,\ z,\ 1) \rightarrow (x,\ y,\ 1)$$

The projection matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective example (onto z=f)



By similar triangles,
(x, y, z) -> (f x/z, f y/z, f)

The equation of projection

- In homogeneous coordinates

$$(x,\ y,\ z,\ 1) \rightarrow (f\frac{x}{z},\ f\frac{y}{z},\ 1)$$

- Equivalently

$$(x,\ y,\ z,\ 1) \rightarrow (x,\ y,\ \frac{z}{f})$$

- Homogeneous coordinates are being used to store foreshortening

- Note that using regular coordinates does **not** yield a linear transformation (inconvenient!).

The projection matrix

$$\begin{bmatrix} x \\ y \\ \frac{z}{f} \end{bmatrix} = \begin{bmatrix} \; \end{bmatrix} \quad ? \quad \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$(x,\ y,\ z,\ 1) \rightarrow (x,\ y,\ \frac{z}{f})$$

The projection matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix}$$

Assembling the pieces

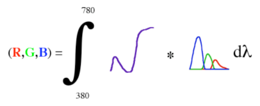
$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \underbrace{\begin{bmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{bmatrix}}_{\text{Camera matrix, } M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Note that we assume that the intrinsic parameters absorb the focal length (so it is one in the projection matrix)

First part makes it so that we are in standard camera coords where we know how to project.

Cameras so far

- Model for the brightness due to light reaching a region (say, CCD element)



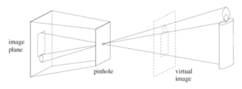
$$\rho^{(k)} = F^{(k)}(\rho^{(k)}) = F^{(k)}\left(\int L(\lambda)R^{(k)}(\lambda)d\lambda\right)$$

or

$$\rho^{(k)} = \mathbf{L} \bullet \mathbf{R}^{(\mathbf{k})}$$

- Spectral camera calibration task is to determine R (and F) from data.

- Model the image location corresponding to a point in the world by projection (developed using pinhole camera model).



- Represent using matrix multiplication using homogenous coordinates

- (u,v,w) = **M** * (X, Y, Z, 1)

- Geometric camera calibration task is to determine **M** from data.

Real cameras

Supplementary material

(Supplementary material on these topics posted on-line)

- Real cameras need lenses
  - Focus now depends on distance (unlike pinhole cameras)
  - Various aberrations and distortions
    - E.g. Chromatic aberration

- Brightness falls off towards edges
  - Fall off due to projection onto flat surface
  - Vignetting

- Scattering at optical surfaces (flare)

- Capture process has many sources of noise

Linear Least Squares (§3.1)

- Very common problem in vision: solve an over-constrained system of linear equations
  - e.g., Ux=y, where U has more rows than needed
- More equations allows multiple measurements to be used
- Least squares means that you minimize squared error (the difference between your model and your data)
- Least squares minimization is (relatively) easy
- Not very robust to outliers (assumes error is Gaussian)
- Used, overused, and abused.

Linear Least Squares (§3.1)

We will look at two problems

First, $U\mathbf{x} = \mathbf{y}$ where U has more rows than needed

Second, $U\mathbf{x} = \mathbf{0}$ subject to $|\mathbf{x}| = 1$ were U has more rows than needed

We will use the **second** problem for geometric camera calibration.

We will start with the **first** (a bit easier)

Math aside, #3

Linear Least Squares (§3.1)

Problem one $U\mathbf{x} = \mathbf{y}$ where U has more rows than needed

U is not square, so inverting it does not work

(Following solution sketch is optional, but try to get main idea)

Define $\mathbf{e} = U\mathbf{x} - \mathbf{y}$ and $E = |\mathbf{e}|^2 = \mathbf{e}^T\mathbf{e}$

Least squares solution is defined by finding the **x** that gives the minimum value for E (by differentiating by each x_i, and setting all resulting equations to zero).

Details optional

Linear Least Squares (§3.1)

$$\frac{\partial E}{\partial x_i} = 2\frac{\partial \mathbf{e}^T}{\partial x_i}\mathbf{e} = 0 \quad \text{(for minimum)}$$

This is true for all components, x_i

Details optional

Linear Least Squares (§3.1)

$$\frac{\partial E}{\partial x_i} = 2\frac{\partial \mathbf{e}^T}{\partial x_i}\mathbf{e} = 0 \quad \text{(for minimum)}$$

This is true for all components, x_i, so we get:

$$\begin{bmatrix} \cdots \\ \frac{\partial \mathbf{e}^T}{\partial x_i} \\ \cdots \end{bmatrix}\mathbf{e} = 0$$

Details optional

Linear Least Squares (§3.1)

Evaluating $\frac{\partial \mathbf{e}^T}{\partial x_i}$

By definition $\frac{\partial \mathbf{e}^T}{\partial x_i} = \left[\frac{\partial}{\partial x_i}(U\mathbf{x} - \mathbf{y})\right]^T = \left[\frac{\partial}{\partial x_i}U\mathbf{x}\right]^T$

$U\mathbf{x}$ is a column vector with elements that look like

$$(U\mathbf{x})_k = \sum_j U_{kj}x_j$$

Details optional

Linear Least Squares (§3.1)

$$(U\mathbf{x})_k = \sum_j U_{kj}x_j$$

So $\frac{\partial}{\partial x_i}(U\mathbf{x})_k = U_{ki}$

And $\frac{\partial}{\partial x_i}U\mathbf{x}$ is the i’th column of U

And $\frac{\partial \mathbf{e}^T}{\partial x_i} = \left[\frac{\partial}{\partial x_i}U\mathbf{x}\right]^T$ is the i’th row of U^T

Details optional

Linear Least Squares (§3.1)

$\frac{\partial \mathbf{e}^T}{\partial x_i} = \left[\frac{\partial}{\partial x_i}U\mathbf{x}\right]^T$ is the i’th row of U^T

So $\begin{bmatrix} \cdots \\ \frac{\partial \mathbf{e}^T}{\partial x_i} \\ \cdots \end{bmatrix}\mathbf{e} = 0$ becomes $U^T(U\mathbf{x} - \mathbf{y}) = 0$

Details optional

Linear Least Squares (§3.1)

From the previous slide our condition is $U^T(U\mathbf{x} - \mathbf{y}) = 0$

Or $U^T U\mathbf{x} = U^T\mathbf{y}$

So $\mathbf{x} = (U^T U)^{-1} U^T\mathbf{y}$ Note that $(U^T U)^{-1}$ can easily be shown to exist

Thus

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger} = (U^T U)^{-1} U^T$ is the pseudoinverse of U

Important

Non-homogeneous linear least squares summary

(the part you need to know)

You should be able to set up

$U\mathbf{x} = \mathbf{y}$

You should know that it is solved by

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger}$ is the pseudoinverse of U

You can assume that you can look up

$U^{\dagger} = (U^T U)^{-1} U^T$

*You should also keep in mind that for numerical stability, one may have to use a different approach to solve (without matrix inversion) the following

$U^T U\mathbf{x} = U^T\mathbf{y}$

Important

Non-homogeneous linear least squares (example one ---naïve line fitting)

Fit best line to a bunch of points (naive way, but note that if you ask a software package to do linear regression, this is likely what you get).

Assume that a line is specified by:

y=mx + b

You have a bunch of (x,y)

What is U and **x** and **y**?

Important

Non-homogeneous linear least squares (example one ---naïve line fitting)

Can write y=mx + b as:

(x 1)*(m b) = y

Important

Non-homogeneous linear least squares
(example one---naïve line fitting)

Can write y=mx + b as:
(x 1)*(m b) = y

So form
a matrix U with rows (x_i 1)
a vector **y** with elements y_i
a vector of unknowns **x**=(a,b)

and use the formula to solve U**x**=**y**

Important

Non-homogeneous linear least squares
(example two---naïve spectral camera calibration)

Remember the fact that the camera has a spectral sensitivity R(λ). So how do we find it out?

Recall that $\rho = \int L(\lambda)R(\lambda)d\lambda$

has the discrete version

$$\rho = \mathbf{L} \bullet \mathbf{R}$$

(previously we accounted for multiple channels with the superscript (k), but here we just consider each channel separately)

Important

Non-homogeneous linear least squares
(example two---naïve spectral camera calibration)

Strategy: measure some spectra entering the camera, $\mathbf{L}_i$, and note the response, ρ_i.

So we have, for a bunch of measurements, i:
$\rho_i = \mathbf{L}_i \bullet R$

If we don’t have enough measurements, then the problem is under constrained. To account for noise, we want to use multiple measurements.

Important

Non-homogeneous linear least squares
(example two---naïve spectral camera calibration)

From:
$\rho_i = \mathbf{L}_i \bullet \mathbf{R}$

The path is clear. Just form a matrix L with rows $\mathbf{L}_i$, a vector **P** with elements ρ_i, and solve the least squares equation
U**R** = **P**