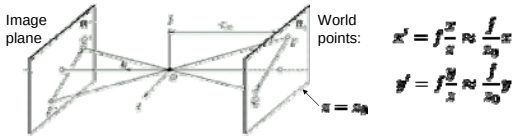
Thursday, Oct 16

Today

- Pset1 examples
- Midterm solutions
- Homography recap, computing mosaics

Weak perspective

- Approximation: treat magnification as constant
- Assumes scene depth << average distance to camera



$$x' = f\frac{x}{z} \approx \frac{f}{z_0}x$$
$$y' = f\frac{y}{z} \approx \frac{f}{z_0}y$$
$$z = z_0$$

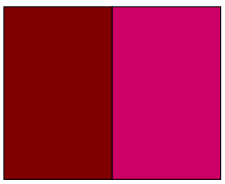
- Write matrix equation that relates world point (X,Y,Z) to its image point according to weak perspective.

Which is more suited for weak perspective projection model?

Color matching experiment 2

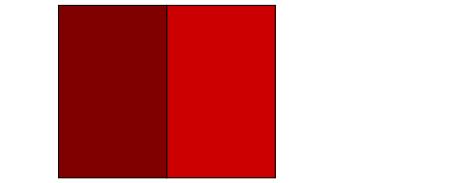
Slide credit: W. Freeman

Color matching experiment 2



Slide credit: W. Freeman

Color matching experiment 2



Slide credit: W. Freeman

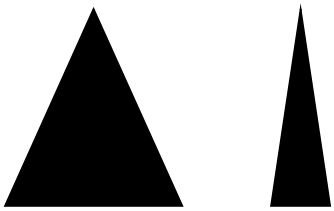
Color matching experiment 2

We say a "negative" amount of p_2 was needed to make the match, because we added it to the test color's side.

The primary color amounts needed for a match:

$$a = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad c = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

[5, 12, 5, 9, 8, 9, 5, 5, 12, 12, 5, **?**]



K-means

- Suppose we are using *k*-means clustering to group pixels in a (tiny) image based on their intensity. The image's intensities are: 5, 10, 3, 20, 9, 0. We pick the initial centers randomly to be 0 and 9, and set the number of clusters *k*=2.
- Cluster membership?
- New cluster centers?

- Affinity score that will discourage intervening contours between pixels.

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, **known** gradient direction

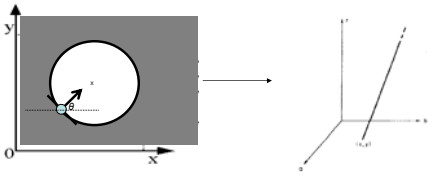
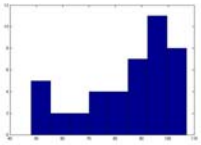
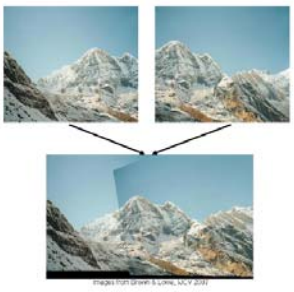


Image space Hough space

Midterm

- Average overall: 83.8 (+/- 17.5)
- Undergrad average: 79 (+/- 18)
- Grad average: 97 (+/- 6)
- Question 5 treated as extra credit (8 pts possible)
- 100+ = A+, 95:99 = A, 90:94 = A-
- 85:89 = B+, 80:84 = B, 75:79 = B-
- 70:74 = C+, 65:69 = C, 60:64 = C-
- 50:60 = D





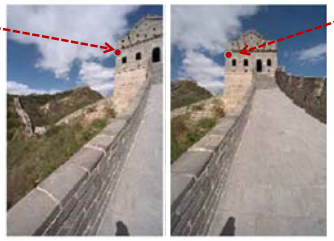
Mosaics: main steps

- **Collect correspondences (manually)**
- **Solve for homography matrix H**
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
- **Overlay im2 content onto the warped im1 content.**

- **`ginput`** to collect clicked points
- What kinds of images to choose as input?

Homography

(x, y)



$\left(wx'/w, wy'/w\right) = (x', y')$

To **apply** a given homography **H**

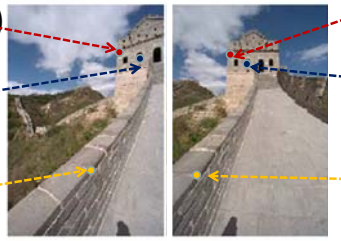
- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

p' H p

Homography

(x_1, y_1) (x_2, y_2) ⋮ (x_n, y_n)



(x'_1, y'_1) (x'_2, y'_2) ⋮ (x'_n, y'_n)

To **compute** the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of **H** are the unknowns…

Solving for homographies

p' = Hp

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor *i*=1. So, there are 8 unknowns.

Set up a system of linear equations:

Ah = b

where vector of unknowns h = [a,b,c,d,e,f,g,h]^T

Need at least 8 eqs, but the more the better…

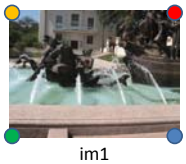
Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

>> **help lmdivide**

Mosaics: main steps

- **Collect correspondences (manually)**
- **Solve for homography matrix H**
  - Least squares solution
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp : `interp2` (watch for nans : `isnan`)
- **Overlay im2 content onto the warped im1 content.**
  - Careful about new bounds of the output image: minx, miny

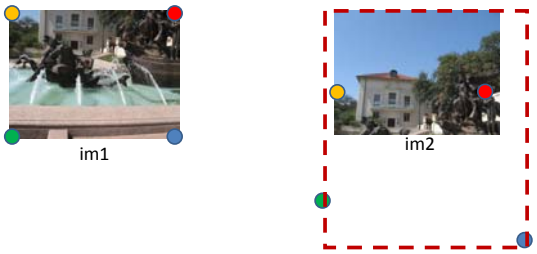


im1

im2

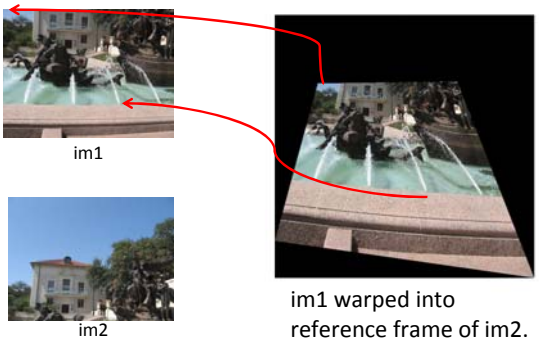
Mosaics: main steps

- **Collect correspondences (manually)**
- **Solve for homography matrix H**
  - Least squares solution
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp : `interp2` (watch for nans : `isnan`)
- **Overlay im2 content onto the warped im1 content.**
  - Careful about new bounds of the output image: minx, miny



Mosaics: main steps

- **Collect correspondences (manually)**
- **Solve for homography matrix H**
  - Least squares solution
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp : `interp2` (watch for nans : `isnan`)
- **Overlay im2 content onto the warped im1 content.**
  - Careful about new bounds of the output image: minx, miny



Use interp2 to ask for the colors (possibly interpolated) from im1 at all the positions needed in im2's reference frame.

Mosaics: main steps

- **Collect correspondences (manually)**
- **Solve for homography matrix H**
  - Least squares solution
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp : `interp2` (watch for nans : `isnan`)
- **Overlay im2 content onto the warped im1 content.**
  - Careful about new bounds of the output image: minx, miny

RANSAC for estimating homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H} p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

Sanity checks

- Click on corresponding points, solve for H, then check that when you plot the transformed points from one image in the other, they land on the right features
- Do the same, but with the corners of one image.

Misc matlab (from pset)

- Watch for index conventions: ginput gives back (x,y), while matrices are indexed in y,x order
- uint8's vs. doubles; give interp2 a matrix of doubles

Possible interface

```
Main script

H = computeH(pts1, pts2)

[im1warped, minx, miny] =
          warpImage(im1, H, im2h, im2w)
```

- For Tuesday:
  - Read F&P 10.1.1-10.1.2, F&P 11.1-11.3
  - [T&V Chapter 7]