Image Mosaicing 1

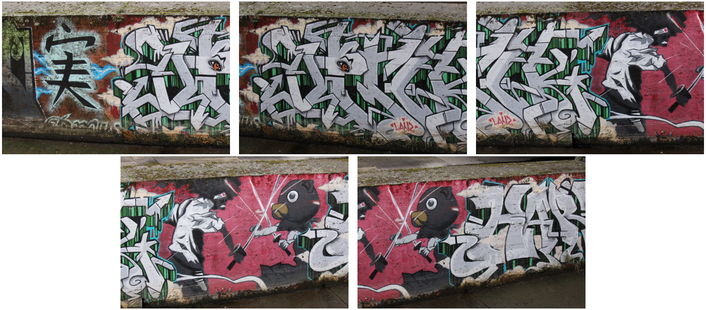
COSC342

Lecture 6
16 March 2016

So What's This All About?

- Image mosaicing
- Mosaicing Geometry
- Homographies
- Features and feature detection

Image Mosaicing



Mosiacing Geometry

- Not all images can be mosiaced correctly
- It turns out that there are two cases which work:
  1. When the camera rotates but does not translate
  2. When the scene is a single planar surface
- In these cases there are no *parallax* effects
- This gives a one-to-one mapping between the images
- This mapping is represented as a *homography*

Homographies

- Suppose we have a point, (x, y), in one image
- We can represent (at least some) transforms of the image with homogenous transformation matrices.
- For example, we might rotate and then shift the image:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- The most general case of this is where we transform by an arbitrary 3×3 matrix, which is called a homography

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathrm{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homographies and Mosaicing

- Now the mosaicing problem becomes:
  - Find a homography that relates points between two images
  - Warp one image with the homography so that it lines up with the other
  - Deal with any residual errors in some way
- How do we find the homography?
  - We can estimate it from a few corresponding points
- Finding corresponding points is hard, and introduces errors
  - The locations of the points are uncertain
  - There is no 100% reliable matching algorithm

Finding a Homography

- Suppose we have a pair of corresponding points

$$\mathbf{p} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T \leftrightarrow \begin{bmatrix} x' & y' & 1 \end{bmatrix}^T = \mathbf{p}'$$

- We know these are related by

$$\begin{aligned} \mathbf{p}' &\equiv \mathrm{H}\mathbf{p} \\ \mathbf{0} &= \mathbf{p}' \times \mathrm{H}\mathbf{p} \end{aligned}$$

  which gives 3 equations:

$$\begin{aligned} 0 &= -xh_{21} - yh_{22} - h_{23} + y'xh_{31} + y'yh_{32} + y'h_{33} \\ 0 &= xh_{11} + yh_{12} + h_{13} - x'xh_{31} - x'yh_{32} - x'h_{33} \\ 0 &= -y'xh_{11} - y'yh_{12} - y'h_{13} + x'xh_{21} + x'yh_{22} + x'h_{23} \end{aligned}$$

Finding a Homography

- We now have 3 equations in the 9 components of H
- Only two are independent, so we drop one
  - Usually drop the third one since it keeps things symmetric
- If we have four point matches we have 8 equations in total
- This lets us solve for H up to a scale (it is homogeneous)
- If we have more than four points we can make a least squares fit

Finding a Homography

Solving for H

- Don't worry too much about this mathematics, but...
- We make an equation $\mathrm{A}\mathbf{h} = 0$ which looks like:

$$\begin{bmatrix} 0 & 0 & 0 & -x_1 & -y_1 & -1 & y_1'x_1 & y_1'y_1 & y_1' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -y_1'x_1 & -y_1'y_1 & -y_1 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & y_2'x_2 & y_2'y_2 & y_2' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -y_2'x_2 & -y_2'y_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_1 & -x_n'y_1 & -x_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ \vdots \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

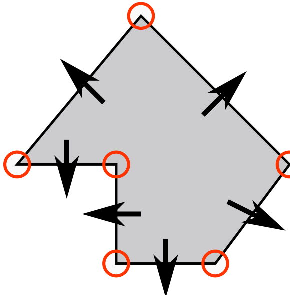
- The Singular Value Decomposition (SVD) gives the solution

Finding the Matches

- The hard part is finding matching points between the two images
- This is often easy for us to do, but is hard for the computer
- There are three main issues to solve:
  - How to *detect* points which can be matched
  - How to *describe* these points so you can match them
  - How to *efficiently match* the points
- This *correspondence problem* is key in many vision tasks

Corner Features

- We want points, or *features*, that can be accurately located
- One idea which is useful is that of a *corner*
- This can be formalised with image gradients
  - In flat areas there is low gradient
  - At edges there is high gradient in one direction
  - At corners there is high gradient in all directions



Corner Features

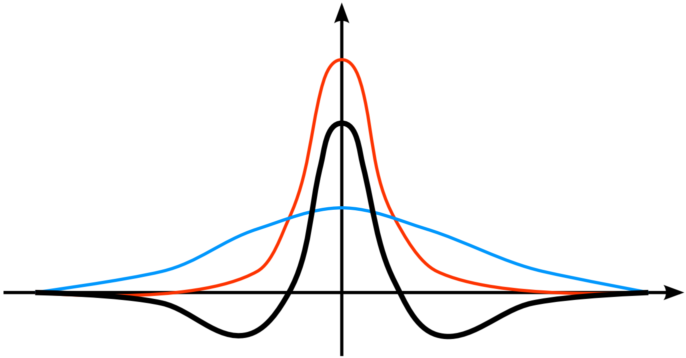
- We can calculate the gradients with filters
- Recall that the Sobel filters find horizontal or vertical edges

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- These can be combined to make a gradient vector, $\mathbf{g} = \begin{bmatrix} S_x \\ S_y \end{bmatrix}$
- Corners can be found by analysing the gradients in a small region
- We look for places where there are gradients in all directions

Blob Features

- More recently, blob features have seen a lot of use
- Blobs are dark regions surrounded by bright regions or vice-versa
- We can find blobs with a *difference of Gaussian* filter



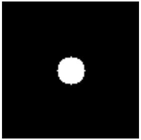
- Blobs have a scale, determined by the variances of the two Gaussians

Blob Detection

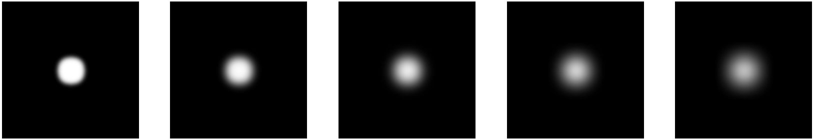
- Blur the image with larger and larger Gaussian kernels
  - You can do this by repeatedly blurring with a small Gaussian kernel
  - For efficiency the image can be halved after every k blurs
- Subtract the adjacent images in the stack from one another
- Blobs are minima and maxima in the stack of difference images
  - Must be locally minimal/maximal in the current difference image
  - Must also be minimal/maximal compared to the two adjacent images

Blob Detection

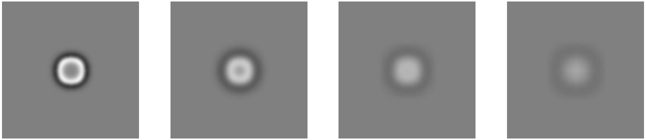
Original Image



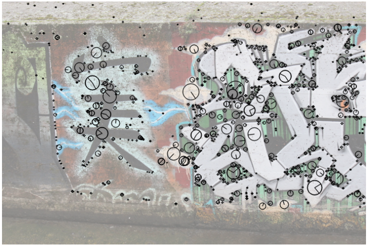
Gaussian blur with $\sigma = 1.5, 3, 4.5, 6, 7.5$



Difference of Gaussians



Corners and Blobs



Coming up...

- Monday's Lab
  - 2D transforms in code
- Lectures next week
  - Describing and matching features
  - Homography estimation
- Tutorials next week
  - 2D Transformations
  - Homogeneous representations