Automatic Image Alignment via Motion Estimation

15-463: Rendering and Image Processing

Alexei Efros

…with a lot of slides stolen from Steve Seitz

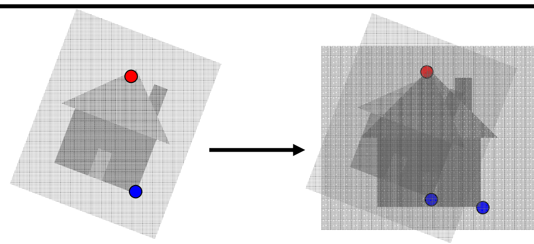
Today

Direct (pixel-based) alignment
- Brute Force Search
- Gradient Search (Motion Estimation)
- Lucas-Kanade

Feature-based alignment
- Interest Points
- SIFT
- Brown & Lowe, “Recognising Panoramas”

Reading:
- Szeliski, Sections 3 and 4

Image Alignment



How do we align two images automatically?

Two broad approaches:

- Feature-based alignment
  - Find a few matching features in both images
  - compute alignment
- Direct (pixel-based) alignment
  - Search for alignment where most pixels agree

Direct Alignment

The simplest approach is a brute force search (hw1)

- Need to define image matching function
  - SSD, Normalized Correlation, edge matching, etc.
- Search over all parameters within a reasonable range:

e.g. for translation:

```
for tx=x0:step:x1,
  for ty=y0:step:y1,
     compare image1(x,y) to image2(x+tx,y+ty)
  end;
end;
```

Need to pick correct `x0,x1` and `step`

- What happens if `step` is too large?

Direct Alignment (brute force)

What if we want to search for more complicated transformation, e.g. homography?

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

```
for a=a0:astep:a1,
  for b=b0:bstep:b1,
    for c=c0:cstep:c1,
      for d=d0:dstep:d1,
        for e=e0:estep:e1,
          for f=f0:fstep:f1,
             for g=g0:gstep:g1,
                for h=h0:hstep:h1,
     compare image1 to H(image2)
end; end; end; end; end; end; end; end;
```

Problems with brute force

Not realistic

- Search in $O(N^8)$ is problematic
- Not clear how to set starting/stopping value and step

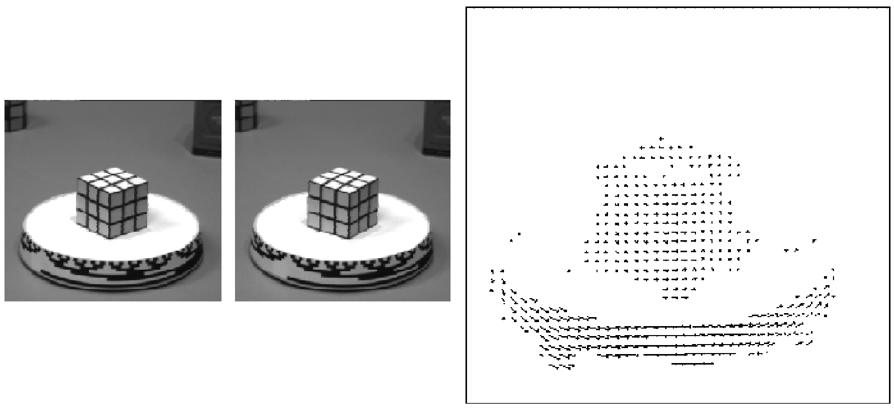
What can we do?

- Use pyramid search to limit starting/stopping/step values
- For special cases (rotational panoramas), can reduce search slightly to $O(N^4)$:
  - $H = K_1 R_1 R_2^{-1} K_2^{-1}$ (4 DOF: f and rotation)

Alternative: gradient decent on the error function

- i.e. how do I tweak my current estimate to make the SSD error go down?
- Can do sub-pixel accuracy
- BIG assumption?
  - Images are already almost aligned (<2 pixels difference!)
  - Can improve with pyramid
- Same tool as in **motion estimation**

Motion estimation: Optical flow

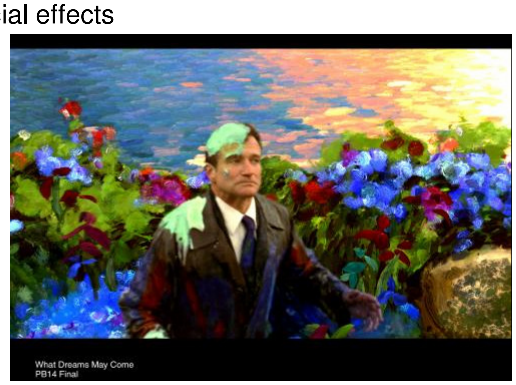


Will start by estimating motion of each pixel separately
Then will consider motion of entire image

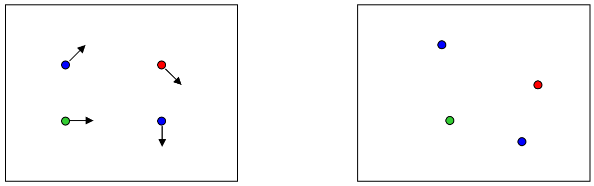
Why estimate motion?

Lots of uses

- Track object behavior
- Correct for camera jitter (stabilization)
- Align images (mosaics)
- 3D shape reconstruction
- Special effects



Problem definition: optical flow



$H(x, y)$ $I(x, y)$

How to estimate pixel motion from image H to image I?

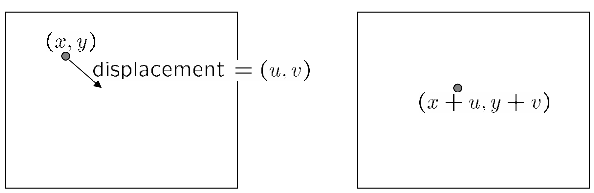
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- **color constancy:** a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

Optical flow constraints (grayscale images)



$H(x, y)$ $I(x, y)$

Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?
- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$
$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Optical flow equation

Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$
\begin{aligned}
0 &= I(x+u, y+v) - H(x,y) \\
&\approx I(x,y) + I_x u + I_y v - H(x,y) \\
&\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot [u \;\; v]
\end{aligned}
$$

In the limit as u and v go to zero, this becomes exact

$$0 = I_t + \nabla I \cdot [\tfrac{\partial x}{\partial t} \;\; \tfrac{\partial y}{\partial t}]$$

Optical flow equation

$$0 = I_t + \nabla I \cdot [u \;\; v]$$

Q: how many unknowns and equations per pixel?

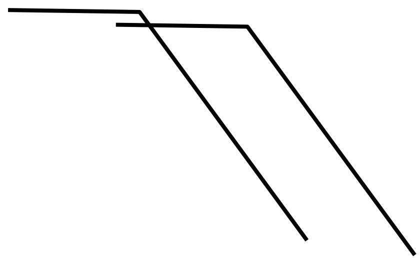
Intuitively, what does this constraint mean?

- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

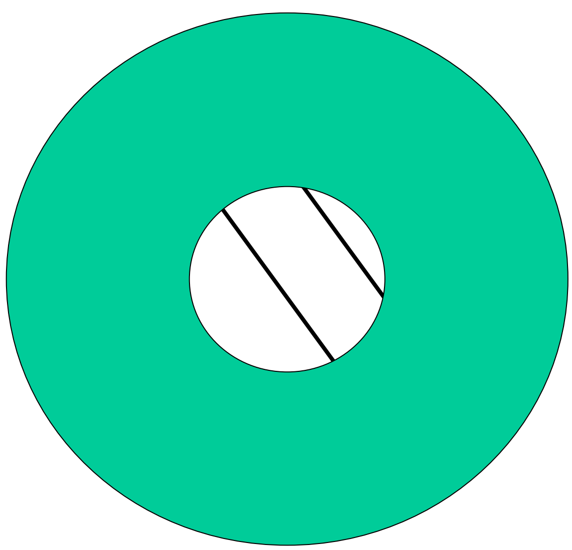
This explains the Barber Pole illusion

http://www.sandlotscience.com/Ambiguous/barberpole.htm

Aperture problem



Aperture problem



Solving the aperture problem

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$$

$$\underset{\substack{A\\25\times2}}{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}} \underset{\substack{d\\2\times1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{\substack{b\\25\times1}}{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}$$

RGB version

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25*3 equations per pixel!

$$0 = I_t(\mathbf{p_i})[0,1,2] + \nabla I(\mathbf{p_i})[0,1,2] \cdot [u \ v]$$

$$\underset{\substack{A\\75\times2}}{\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix}} \underset{\substack{d\\2\times1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{\substack{b\\75\times1}}{\begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}}$$

Lukas-Kanade flow

Prob: we have more equations than unknowns

$$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

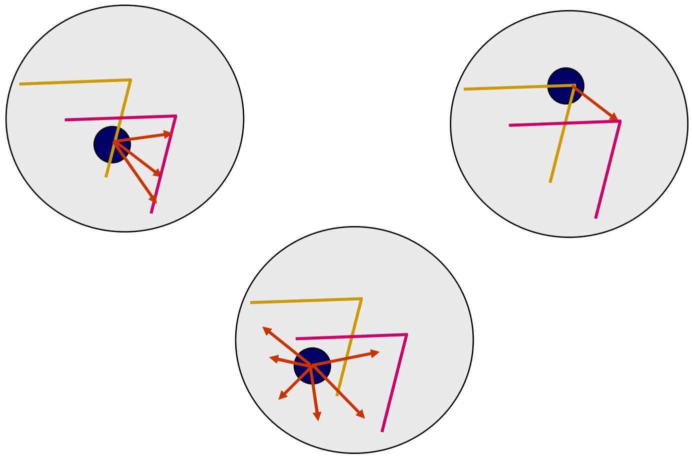
When is This Solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

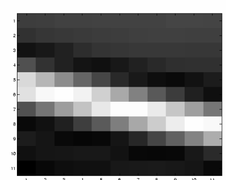
$\mathbf{A^T A}$ is solvable when there is no aperture problem

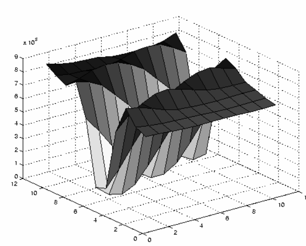
$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x\ I_y] = \sum \nabla I (\nabla I)^T$$

Local Patch Analysis



Edge

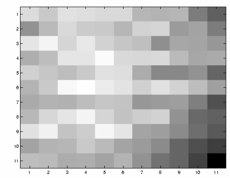


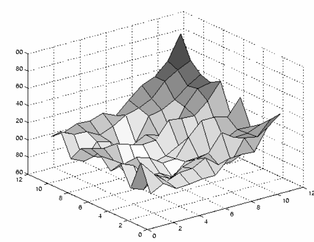


$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same
– large λ_1, small λ_2

Low texture region

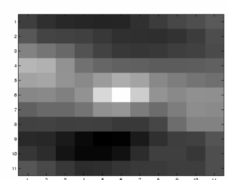


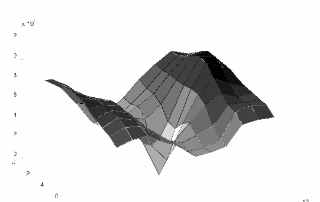


$\sum \nabla I (\nabla I)^T$

- gradients have small magnitude
- small λ_1, small λ_2

High textured region





$\sum \nabla I (\nabla I)^T$

- gradients are different, large magnitudes
- large λ_1, large λ_2

Observation

This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - very useful later on when we do feature tracking...

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose $A^T A$ is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

- Brightness constancy is **not** satisfied
- The motion is **not** small
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

Iterative Refinement

Iterative Lukas-Kanade Algorithm

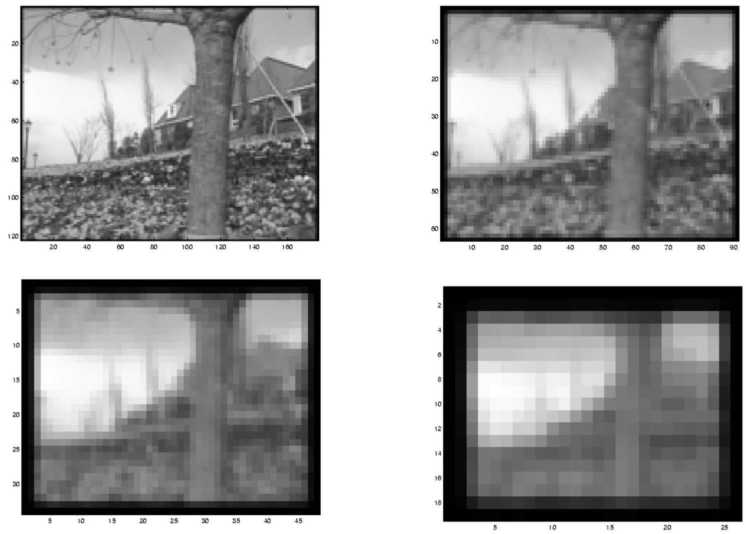
1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
   - *- use image warping techniques*
3. Repeat until convergence

Revisiting the small motion assumption

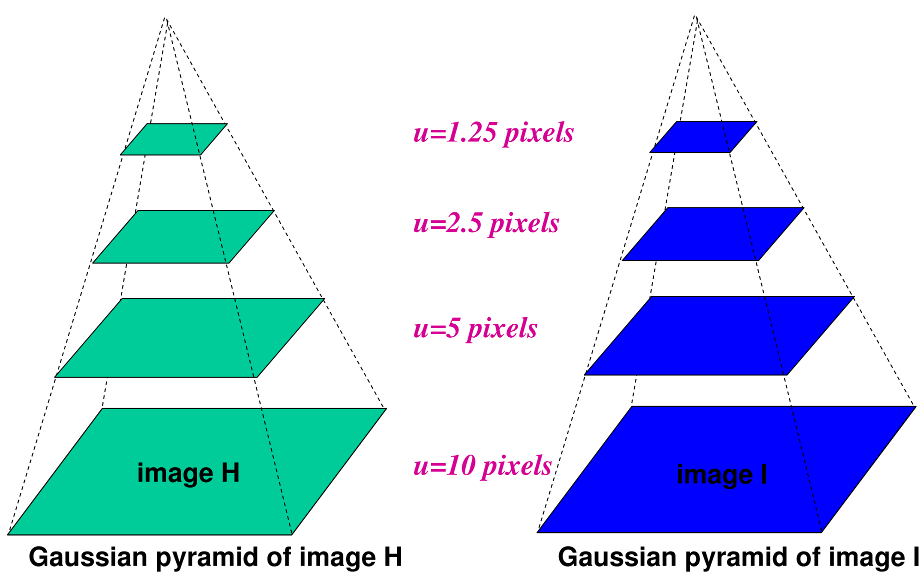
Is this motion small enough?

- Probably not—it's much larger than one pixel (2nd order terms dominate)
- How might we solve this problem?

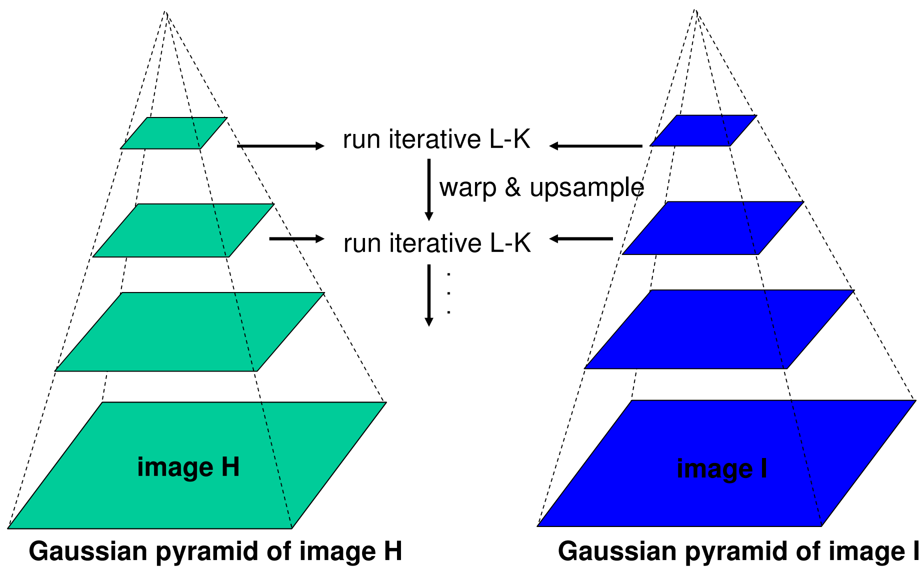
Reduce the resolution!



Coarse-to-fine optical flow estimation



Coarse-to-fine optical flow estimation



Beyond Translation

So far, our patch can only translate in (u,v)

What about other motion models?

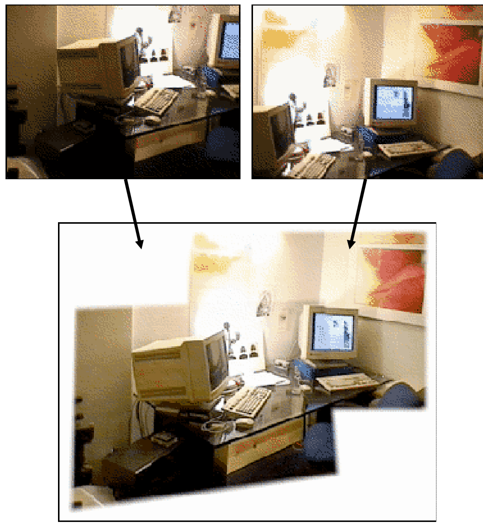
- rotation, affine, perspective

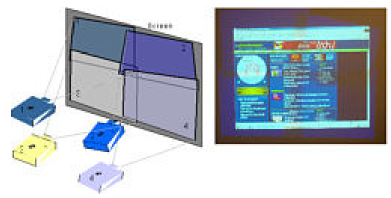
Same thing but need to add an appropriate Jacobian (see Table 2 in Szeliski handout):

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \sum_i \mathbf{J}\nabla\mathbf{I}(\nabla\mathbf{I})^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{b} = -\sum_i \mathbf{J}^{\mathrm{T}} I_t (\nabla\mathbf{I})^{\mathrm{T}}$$

Image alignment





Goal: estimate single (u,v) translation for entire image

- Easier subcase: solvable by pyramid-based Lukas-Kanade

Lucas-Kanade for image alignment

Pros:

- All pixels get used in matching
- Can get sub-pixel accuracy (important for good mosaicing!)
- Relatively fast and simple

Cons:

- Prone to local minima
- Images need to be already well-aligned ☺

What if, instead, we extract important “features” from the image and just align these?

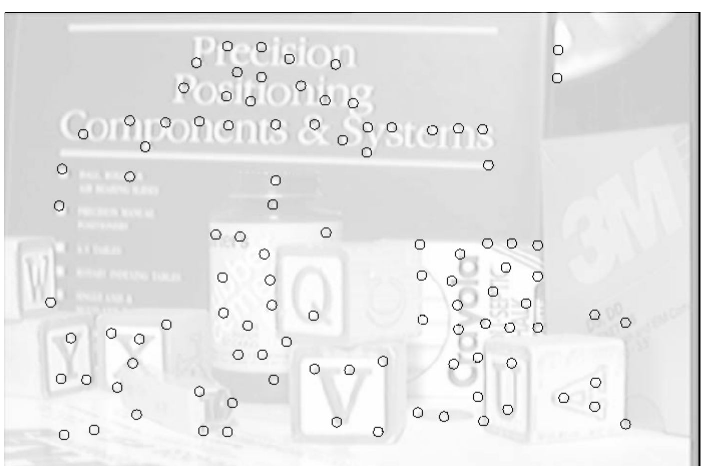
Feature-based alignment

1. Find a few important features (aka Interest Points)
2. Match them across two images
3. Compute image transformation as per HW#2

Choosing Features

- Choose only the points (“features”) that are salient, i.e. likely to be there in the other image
- How to find these features?
  - windows where $\sum \nabla I (\nabla I)^T$ has two large eigenvalues
- Called the Harris Corner Detector

Feature Detection



Feature Matching

One possibility:

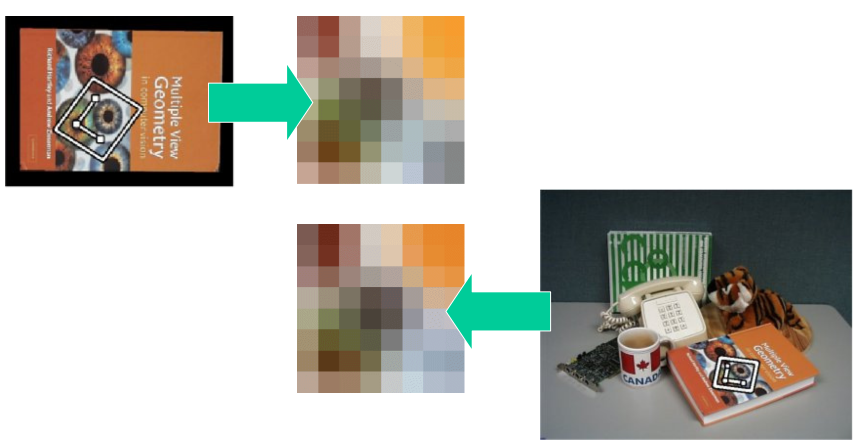
- Match features found in image1 with features found in image2
  - e.g. SSD of image patches around each feature
- Use successful matches to estimate homography
  - Do something to get rid of outliers

Problems:

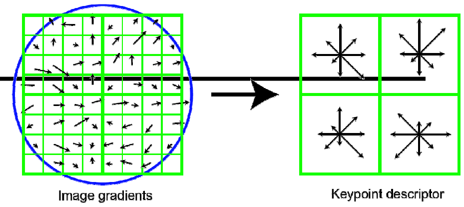
- What if the image patches for several interest points look similar?
  - Make patch size bigger
- What if the image patches for the same feature look different due to scale, rotation, etc.
  - Use Lucas-Kanade with affine motion model
  - Better solution: Scale-Invariant Feature Transform (SIFT)

Invariant Features

Schmid & Mohr 1997, Lowe 1999, Baumberg 2000, Tuytelaars & Van Gool 2000, Mikolajczyk & Schmid 2001, Brown & Lowe 2002, Matas et. al. 2002, Schaffalitzky & Zisserman 2002



SIFT Features



Invariant Features

- Establish invariant frame
  - Find optimal scale
    - » Maxima/minima of scale-space DOG $\Rightarrow$ x, y, s
  - Find optimal orientation
    - » Maximum of distribution of local gradients $\Rightarrow \theta$
- Form descriptor vector
  - Histogram of smoothed local gradients
  - 128 dimensions

SIFT features are…

- Geometrically invariant to similarity transforms,
  - some robustness to affine change
- Photometrically invariant to affine changes in intensity

Example: Recognising Panoramas

M. Brown and D. Lowe,
University of British Columbia

Why “Recognising Panoramas”?

Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

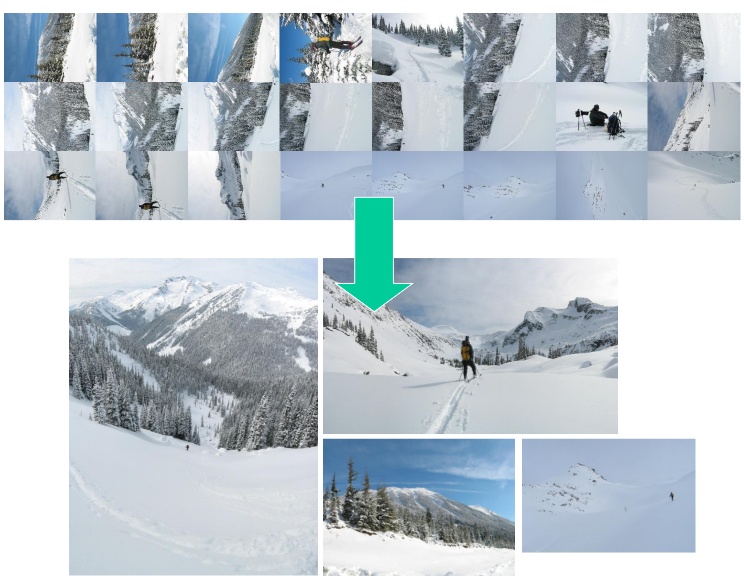
Why “Recognising Panoramas”?

1D Rotations (θ)

- Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\nRightarrow$ matching images

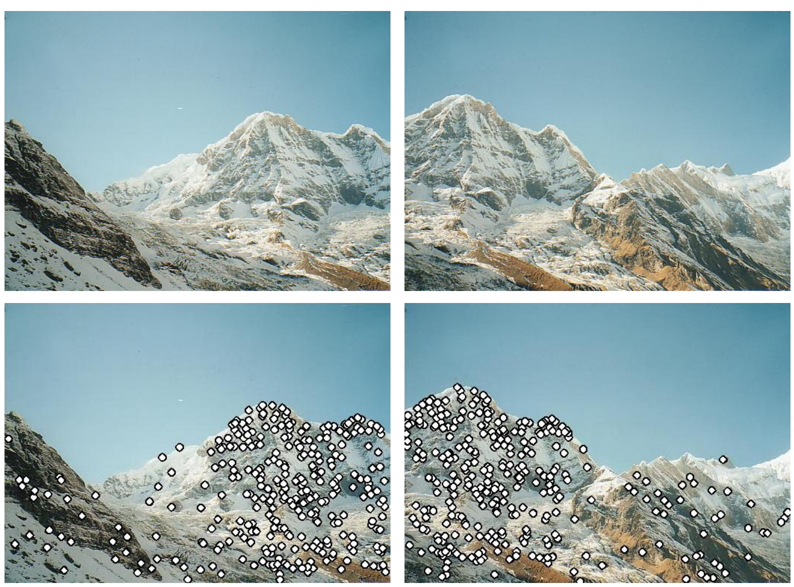
Why “Recognising Panoramas”?



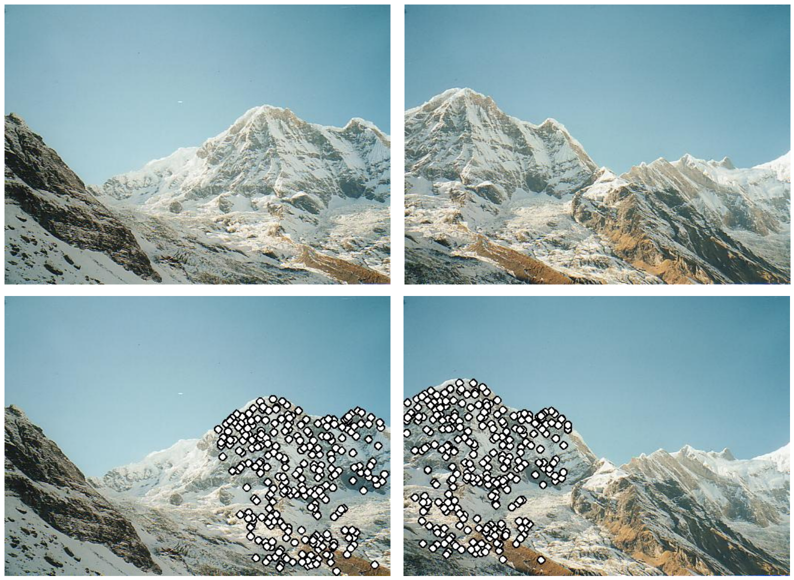
Overview

Feature Matching
Image Matching
Bundle Adjustment
Multi-band Blending
Results
Conclusions

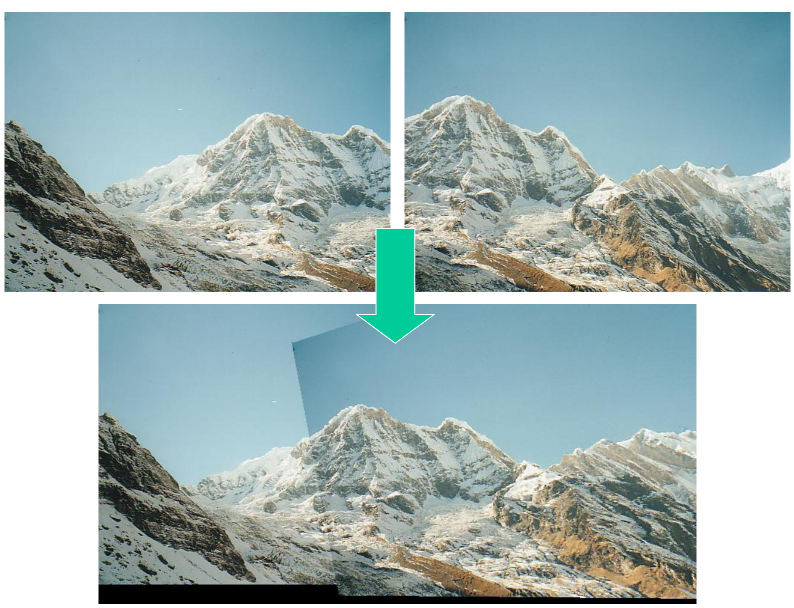
RANSAC for Homography



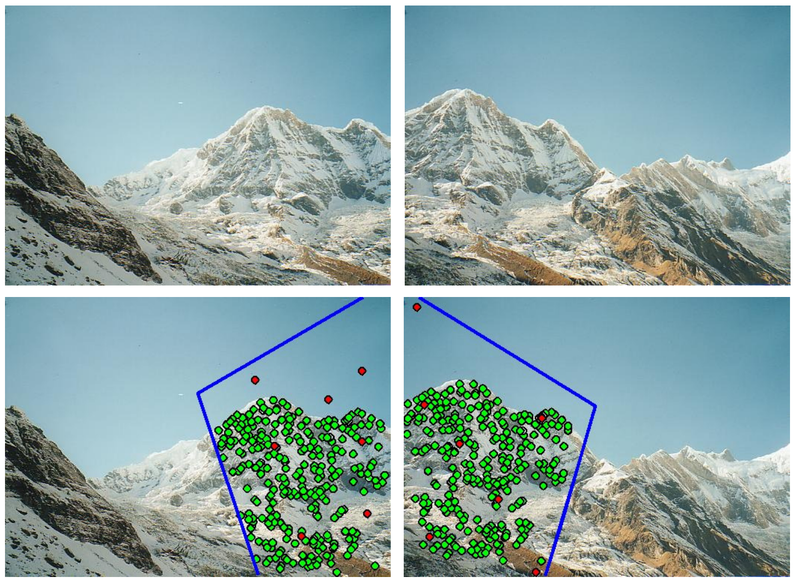
RANSAC for Homography



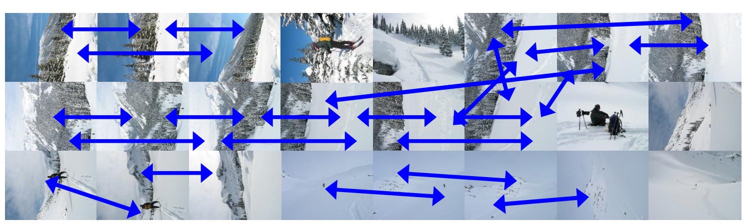
RANSAC for Homography



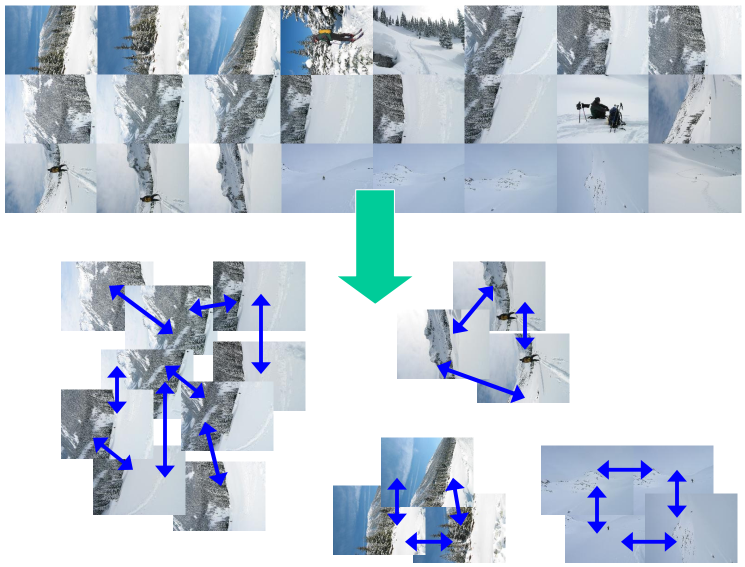
Probabilistic model for verification



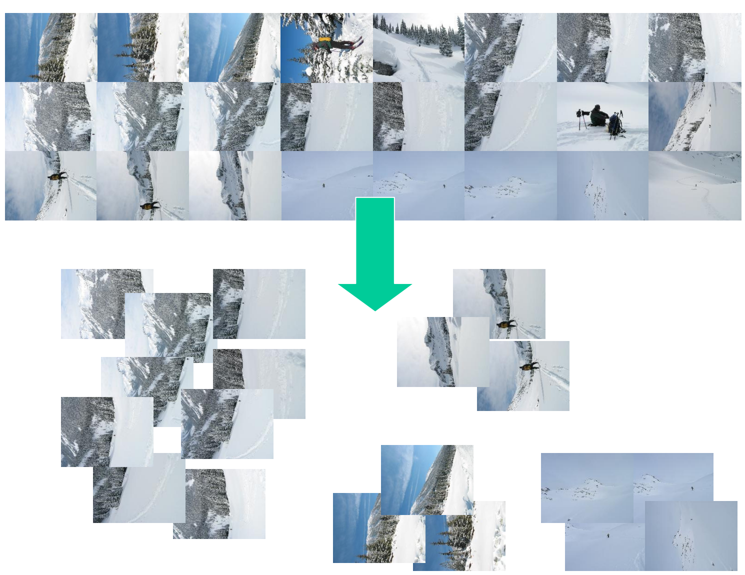
Finding the panoramas



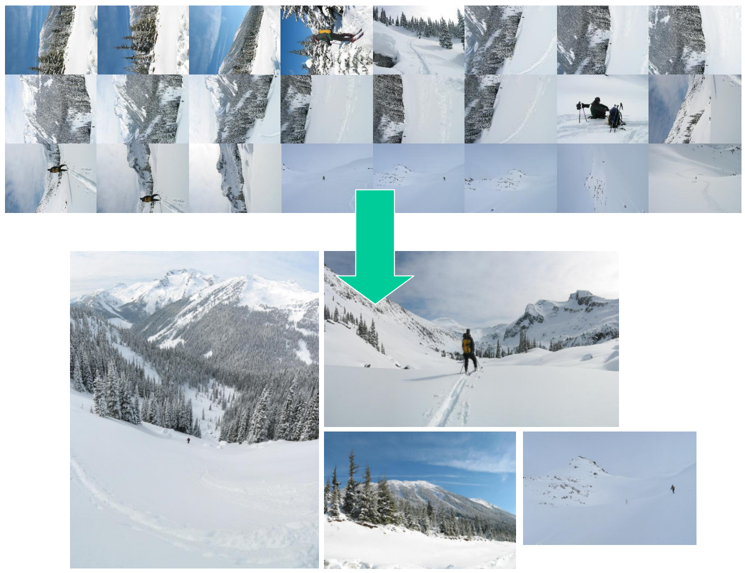
Finding the panoramas



Finding the panoramas



Finding the panoramas



Homography for Rotation

Parameterise each camera by rotation and focal length

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \quad [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This gives pairwise

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j, \quad \mathbf{H}_{ij} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^T\mathbf{K}_j^{-1}$$

Bundle Adjustment

New images initialised with rotation, focal length of best matching image

Bundle Adjustment

New images initialised with rotation, focal length of best matching image

Multi-band Blending

Burt & Adelson 1983

- Blend frequency bands over range $\propto \lambda$

Results