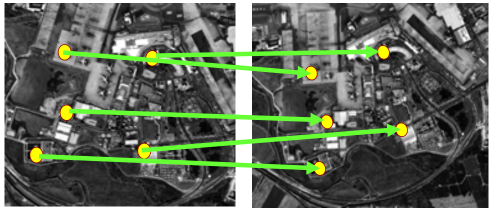
Feature Based Registration - Image Alignment



Many slides from Alexei Efros http://graphics.cs.cmu.edu/courses/15-463/2007_fall/463.html
D.Frolova, D. Simakov - http://www.wisdom.weizmann.ac.il/~deniss/2004
03_invariant_features/InvariantFeatures.ppt

Image Registration

Image registration is the process of estimating an optimal transformation between two or more images.

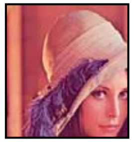
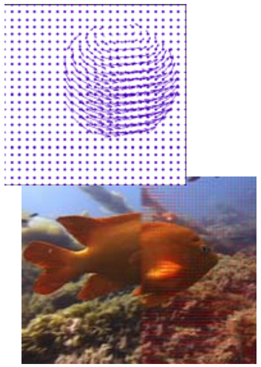


Image Registration

Motion Estimation and Optical Flow



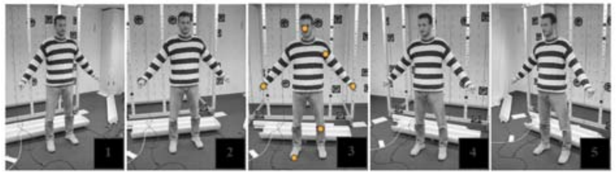
Assumes:
- Constant Brightness assumption
- Small Motion
- Spatial Coherence.

Affected by:
- Window Size, Image Noise, numerical convergence

Example: Mosiacing (Panorama)

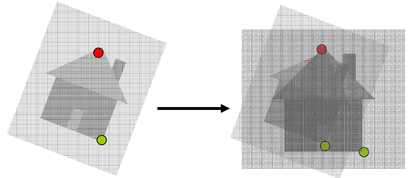
M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

Example – 3D Reconstruction



Source: http://www.photogrammetry.ethz.ch/general/persons/fabio/fabio_spie0102.pdf

Image Registration



How do we align two images automatically?

Two broad approaches:

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

Direct Method (brute force)

The simplest approach is a brute force search

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

  e.g. for translation:

```
for Δx=x0:step:x1,
 for Δy=y0:step:y1,
   calculate Dist(image1(x,y),image2(x+Δx,y+Δy))
 end;
end;
```

Direct Method (brute force)

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

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

```
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,
     calculate Dist(image1 , H(image2))
end; end; end; end; end; end; end; end;
```

Problems with brute force

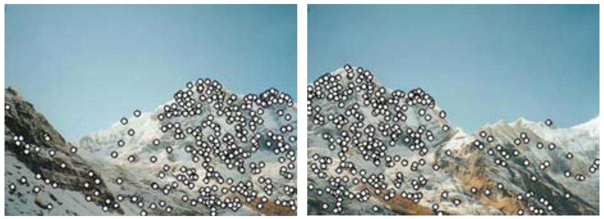
- Not realistic for large number of parameters.
- Alternatives:
  - Reduce parameter range using pyramids.
  - Gradient decent on the error function.

Direct v.s. Feature Registration

- Direct methods:
  - Can be applied locally (OF, video coding)
  - Can handle complicated motions
  - Gradient methods need good initial guess
- Feature based
  - Fast
  - No need for initial guess
  - Can handle only global motion models

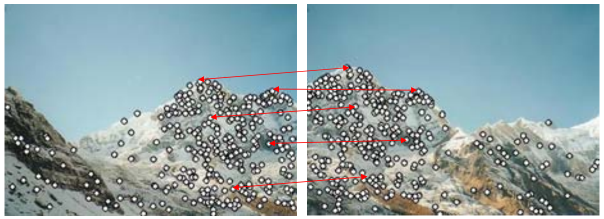
Feature-based registration

- Detect feature points in both images



Feature-based registration

- Detect feature points in both images
- Find corresponding pairs



Feature-based registration

- Detect feature points in both images
- Find corresponding pairs
- Compute image transformation

Features: Issues to be addressed

- What are "good" features to extract?
  - –Distinctive
  - –Invariant to different acquisition conditions
    - –Different view-points, different illuminations, different cameras, etc.
- How can we find corresponding features in both images?

no chance to match!

Feature Correspondence -Example

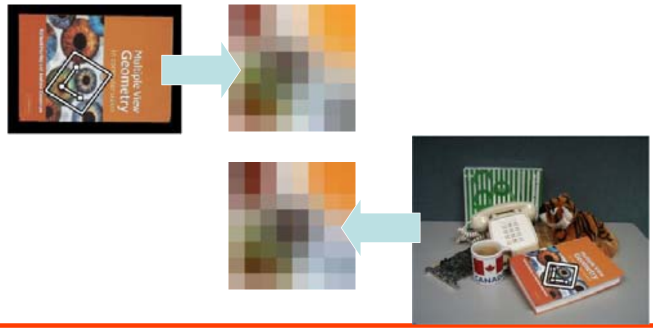
Given an image point in left image, what is the corresponding point in the right image, which is the projection of the same 3-D point.

Image Features

- Feature Detectors - where
- Feature Descriptors - what
- Methods:
  - – Harris Corner Detector (multi-scale Harris)
  - – SIFT (Scale Invariant Features Transform)

Invariant Feature Descriptors

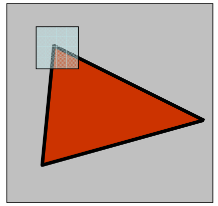
- Schmid & Mohr 1997, Lowe 1999, Baumberg 2000, Tuytelaars & Van Gool 2000, Mikolajczyk & Schmid 2001, Brown & Lowe 2002, Matas et. al. 2002, Schaffalitzky & Zisserman 2002



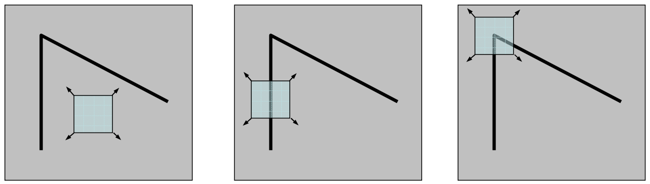
Harris Corner Detector

C.Harris, M.Stephens. "A Combined Corner and Edge Detector". 1988

- We should easily recognize a corner by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



Harris Detector: Basic Idea



"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Harris Detector: Mathematics

Change of intensity for shift in [u,v] direction:

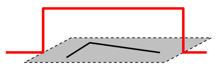
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

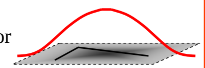
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Harris Detector: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

For small [u,v]: $I(x+u,y+v) = I(x,y) + uI_x + vI_y$

We have:

$$E(u,v) = \sum_{x,y} w(x,y)\left\| \begin{bmatrix} I_x(x,y) & I_y(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right\|^2 =$$

$$\begin{bmatrix} u & v \end{bmatrix} \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris Detector: Mathematics

For small shifts $[u,v]$ we have a *bilinear* approximation:

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Harris Detector: Mathematics

Denote by $\mathbf{e}_i$ the i[th] eigen-vector of M whose eigen-value is λ_i:

$$\mathbf{e}_i^T M \mathbf{e}_i = \lambda_i > 0$$

Conclusions:

$$\arg\max_{\|(u,v)\|=1} E(u,v) = \mathbf{e}_{\max}$$

$$E(\mathbf{e}_{\max}) = \lambda_{\max}$$

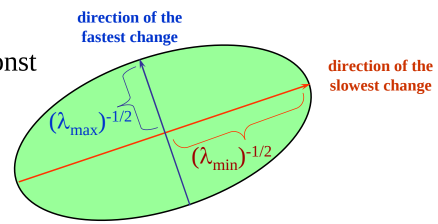
Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

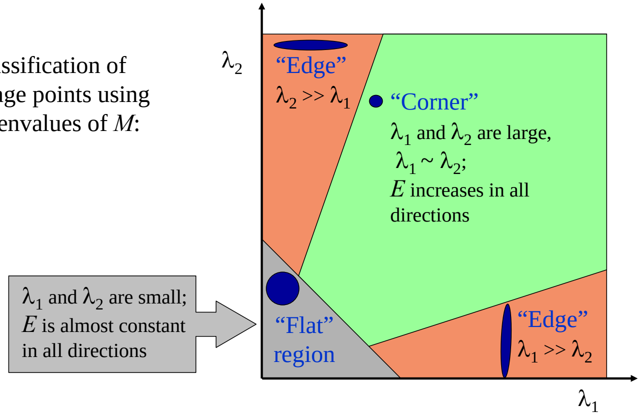
λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



Harris Detector: Mathematics

Classification of image points using eigenvalues of M:



Harris Detector: Mathematics

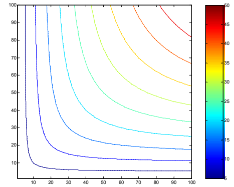
Measure of corner response (without calculating the e.v.):

$$R = \frac{\det M}{\text{Trace } M}$$

$$\det M = \lambda_1 \lambda_2$$
$$\text{trace } M = \lambda_1 + \lambda_2$$

R is associated with the smallest eigen-vector (why?)

R v.s. λ_1, λ_2



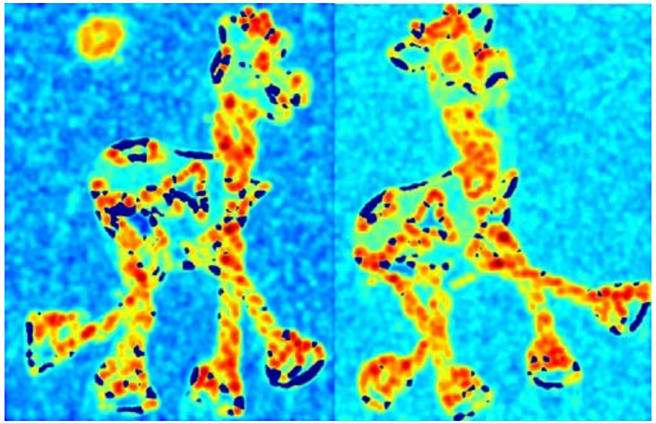
Harris Corner Detector

- The Algorithm:
  - Find points with large corner response function R (R > threshold)
  - Take the points of local maxima of R

Harris Detector: Workflow

Harris Detector: Workflow

Compute corner response R

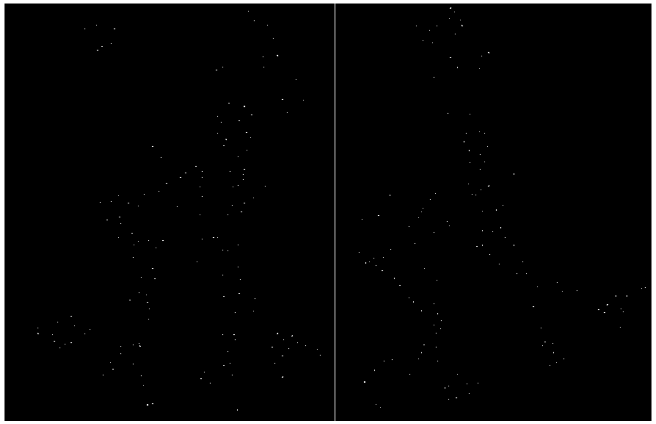


Harris Detector: Workflow

Find points with large corner response: R>threshold

Harris Detector: Workflow

Take only the points of local maxima of R

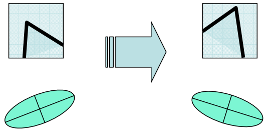


Harris Detector: Workflow

Harris Detector: Example

Harris Detector: Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

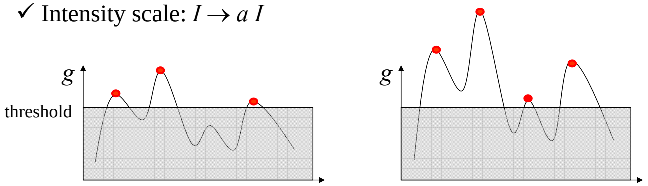
Corner response R is invariant to image rotation

Harris Detector: Some Properties

Partial invariance to affine intensity change

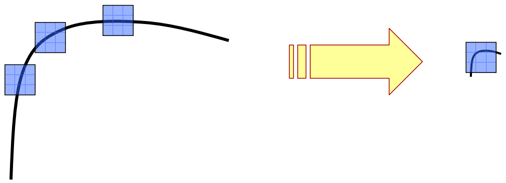
✓ Only derivatives are used => invariance to intensity shift $I \to I + b$

✓ Intensity scale: $I \to a\, I$



Harris Detector: Some Properties

- But: non-invariant to spatial scale!

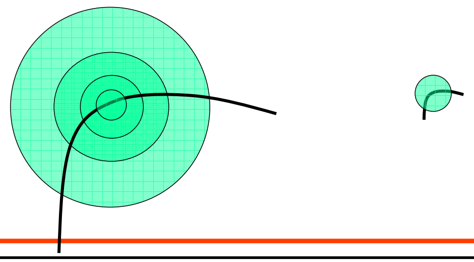


All points will be classified as edges

Corner !

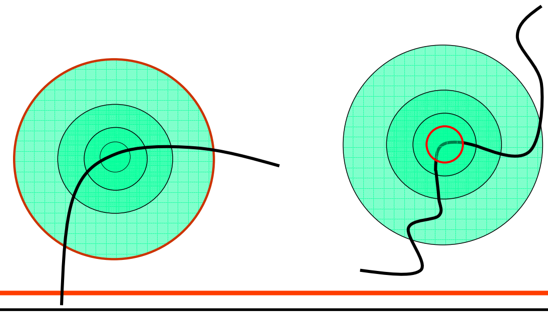
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

- **The problem**: how do we choose corresponding circles *independently* in each image?
- **Solution**: choose the scale of the "best" corner.

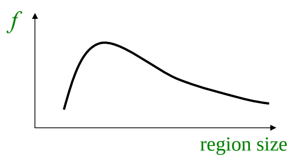


Scale Invariant Detection

– Design a function on regions (circle).

Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

– For a point in an image, we can consider it as a function of region size (circle radius)



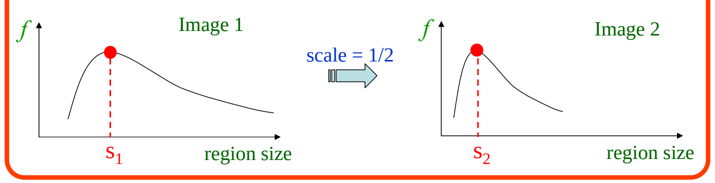
Scale Invariant Detection

- Common approach:

  Take a local maximum of this function

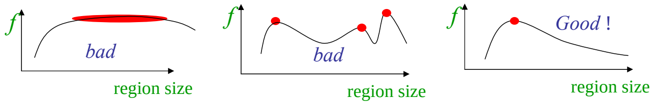
  Observation: region size, for which the maximum is achieved, should be *equivariant* to image scale.

  Important: this scale invariant region size is found in each image independently!



Scale Invariant Detection

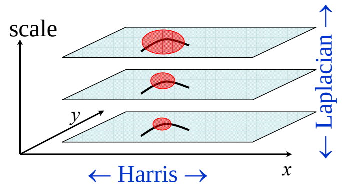
- A “good” function for scale detection: has one stable sharp peak



- For usual images: a good function would be one that responds to contrast (sharp local intensity change)

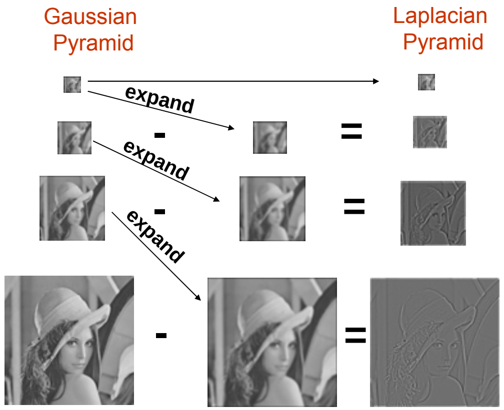
Harris-Laplacian Point Detector

- Harris-Laplacian
  Find local maximum of: Harris corner detector for a set of Laplacian images.



[1] K.Mikolajczyk, C.Schmid. “Indexing Based on Scale Invariant Interest Points”. ICCV 2001

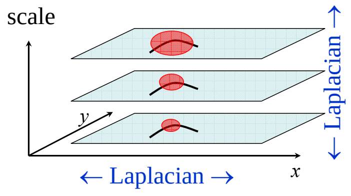
Memo: Gaussian / Laplacian Pyramids



Harris - Laplacian Detector

SIFT – Scale Invariant Feature Transform

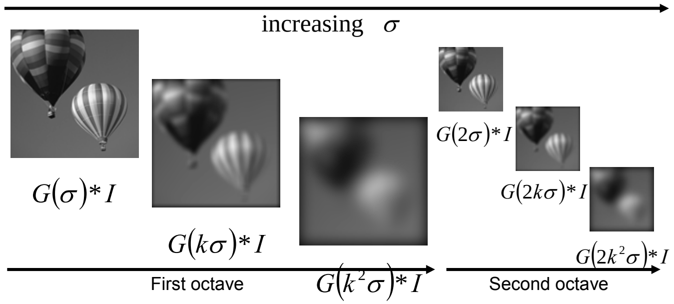
- *Find local maximum of* Laplacian in space and scale



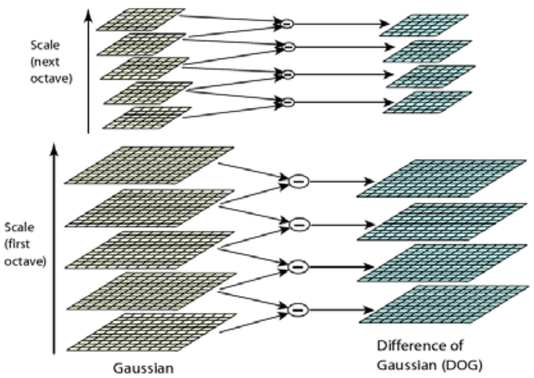
David G. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

SIFT - Point Detection

- Construct scale-space:

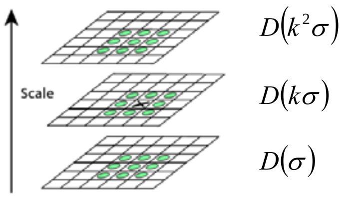


SIFT – Scale Space

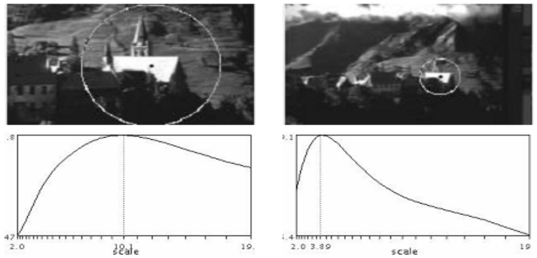


SIFT – point detection

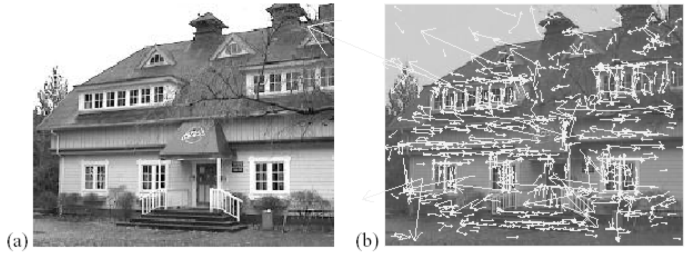
- Scale Space extrema detection.
- Choose all extrema within 3x3x3 neighborhood.



SIFT – point detection



SIFT – point detection



233x189 image

832 SIFT extrema

Scale Invariant Detection: Summary

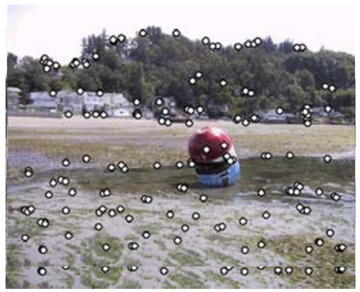
- Given: two images of the same scene with a large *scale difference* between them
- Goal: find *the same* interest points *independently* in each image
- Solution: search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Methods:

1. Harris-Laplacian [Mikolajczyk, Schmid]: maximize Laplacian over scale, Harris' measure of corner response over the image
2. SIFT [Lowe]: maximize Laplacian over scale and space

One More thing…. Feature selection

- Want feature points to be distributed approx evenly over the image.



Adaptive Non-maximal Suppression

Desired: Fixed # of features per image

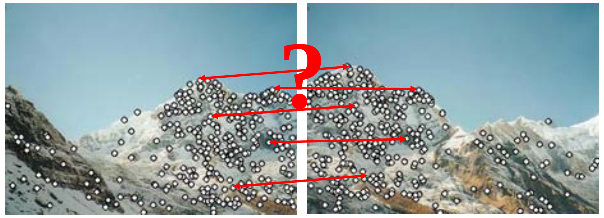
- Want evenly distributed spatially…
- Sort points by non-maximal suppression radius [Brown, Szeliski, Winder, CVPR'05]

(a) Strongest 250 (b) Strongest 500

(c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

Feature descriptors

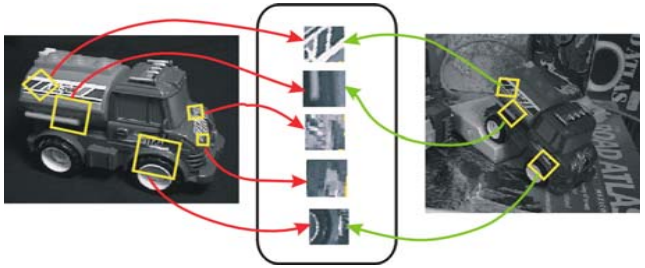
- We know how to detect points
- Next question: **How to match them?**
- **Solution**: Generate point descriptors



Point descriptor should be:
1. Invariant
2. Distinctive

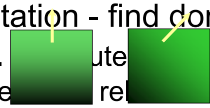
Invariant Point Descriptors

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Features Descriptors

Descriptors Invariant to Scale + Rotation + Contrast

- Local scale is given by the feature detection procedure.
- Local orientation - find dominant direction of gradient.  ute descriptors/fe re o this orientation (e.g. derivatives)
  - Normalize image descriptors/features relative to scale and orientation and relative to patch contrast.

[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". Accepted to IJCV 2004

Descriptors Invariant to Rotation, Scale, Intensity

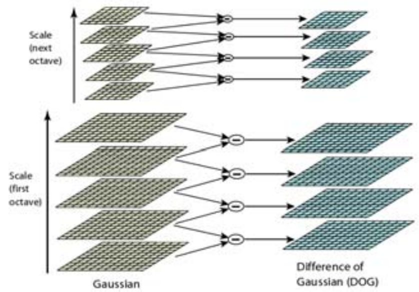
SIFT = Scale Invariant Feature Transform

David G. Lowe,
"Distinctive image features from scale-invariant keypoints",
International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

- SIFT Feature point is associated with: location, Orientation, Scale.
- SIFT Descriptor is a vector of 128 values each between [0 -1]

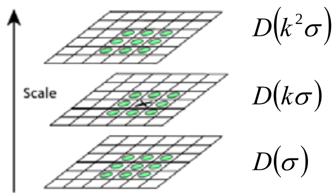
SIFT - Step 1: Interest Point Detection

- DOG (laplacian Pyramid): take differences.



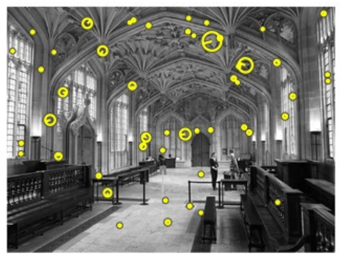
SIFT - Step 1: Interest Point Detection

- Scale Space extrema detection.
- Choose all extrema within 3x3x3 neighborhood.



SIFT - Step 1: Interest Point Detection

Detections at multiple scales



Some of the detected SIFT frames.

http://www.vlfeat.org/overview/sift.html

Interest Points

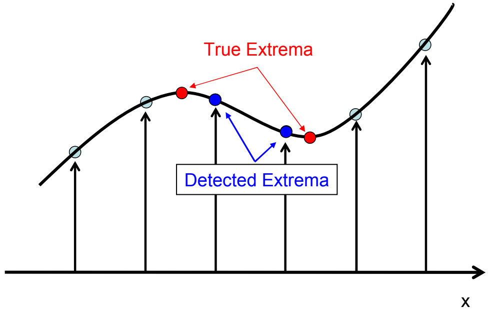
(b)

233x189 image

832 DOG extrema

SIFT - Step 2: Interest Localization & Filtering

1) Improve localization of Interest points.



For each maximum, fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero

SIFT - Step 2: Interest Localization & Filtering

2) Remove bad Interest points:
 a) Remove points with low contrast
 b) Remove Edge points (Eigenvalues of Hessian Matrix must BOTH be large).

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

Interest Points

(c)

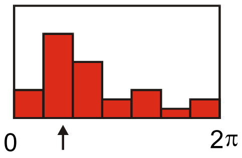
(d)

(c) 729 left after peak value threshold (from 832)
(d) 536 left after testing ratio of principle curvatures

SIFT – Descriptor Vector

STEP 3: Select canonical orientation

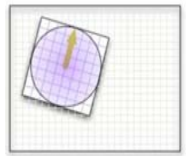
- Each SIFT interest point is associated with location (x,y) and scale (σ)
- Compute gradient magnitude and orientation for each SIFT point:
  - Create histogram of local gradient directions computed at selected scale
  - Assign canonical orientation at peak of smoothed histogram



SIFT – Descriptor Vector

STEP 3: Select canonical orientation

- Each SIFT interest point is associated with location (x,y), scale (σ), gradient magnitude and orientation (m, θ).



- Compute SIFT feature - a vector of 128 entries.

SIFT – Descriptor Vector

- Gradients determined in 16x16 window at SIFT point in scale space.
- Histogram is computed for gradients of each 4x4 sub window in 8 **relative** directions.
- A 4x4x8 = 128 dimensional feature vector is produced.

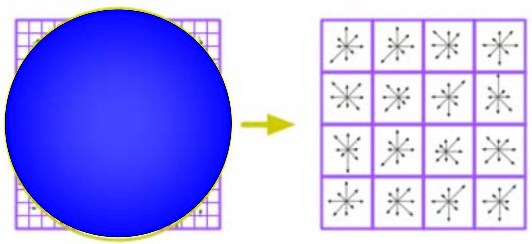
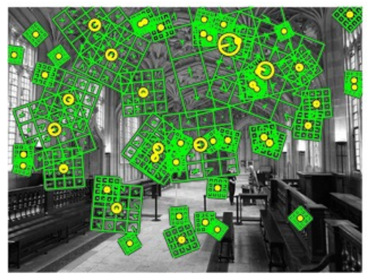


Image gradients   Keypoint descriptor

Image from: Jonas Hurrelmann

SIFT – Descriptor Vector

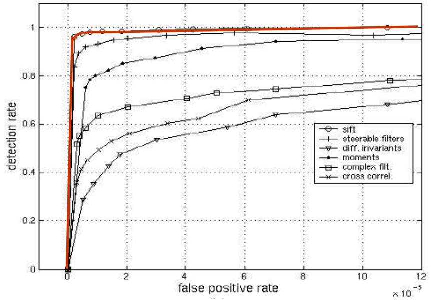
STEP 4: Compute feature vector



SIFT – Scale Invariant Feature Transform

- Empirically found[2] to show very good performance, invariant to *image rotation*, *scale*, *intensity change*, and to moderate *affine* transformations

Scale = 2.5
Rotation = 45^0

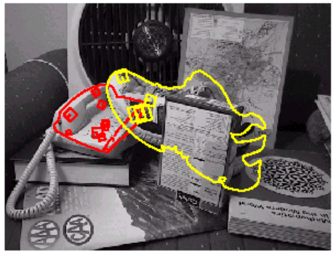


[1] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". Accepted to IJCV 2004
[2] K.Mikolajczyk, C.Schmid. "A Performance Evaluation of Local Descriptors". CVPR 2003

3D Object Recognition

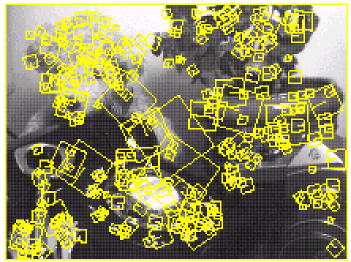
- Only 3 keys are needed for recognition, so extra keys provide robustness

Recognition under occlusion



Test of illumination Robustness

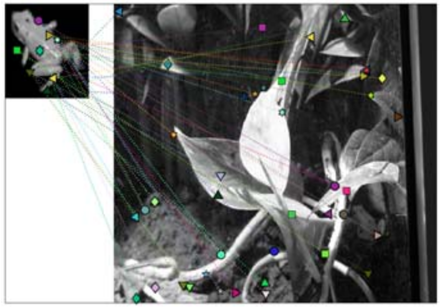
- Same **image** under differing illumination



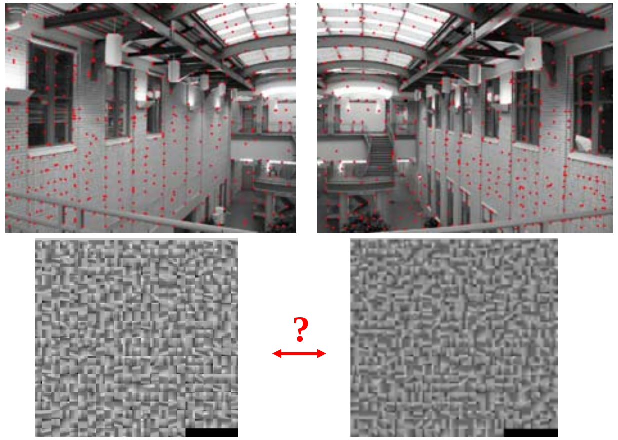
273 keys verified in final match

Cases where SIFT didn’t work

- Same **object** under differing illumination
- 43 keypoints in left image and the corresponding closest keypoints on the right (1 for each)



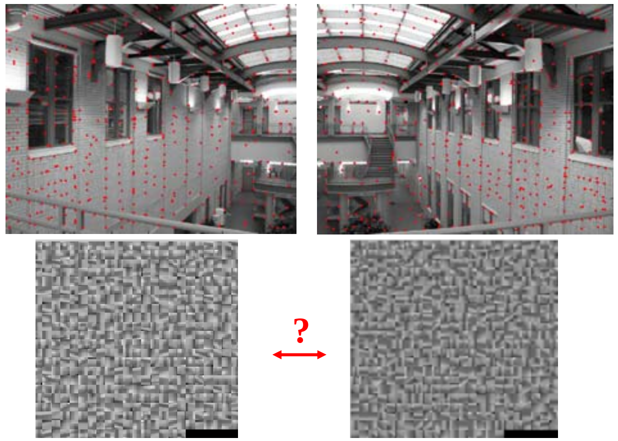
Feature matching

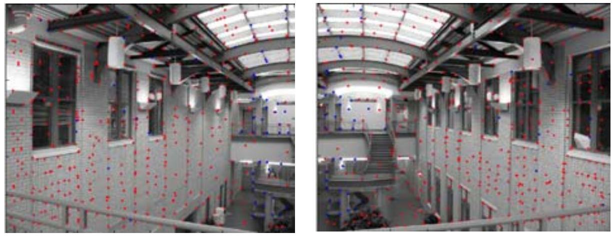


Feature matching

- Exhaustive search
  - for each feature in one image, look at *all* the other features in the other image(s)
- Hashing
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
  - *kd*-trees and their variants

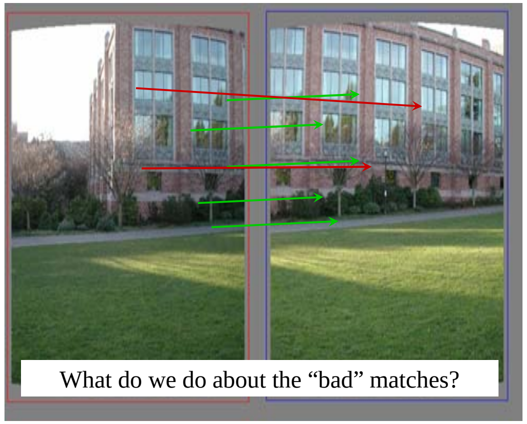
What about outliers?





All red points are outliers

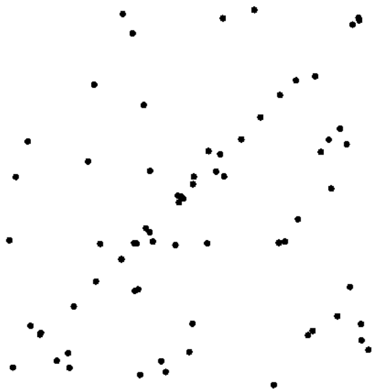
Matching features



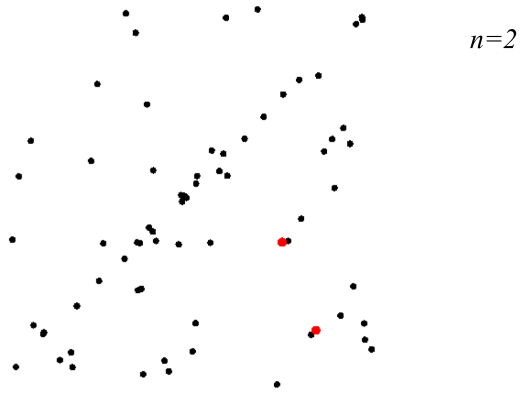
RANSAC

- RANSAC = Random Sample Consensus
- An algorithm for robust fitting of models in the presence of many data outliers
- Given N data points $\{x_i\}$, assume that mjority of them are generated from a model with parameters Θ, try to recover Θ

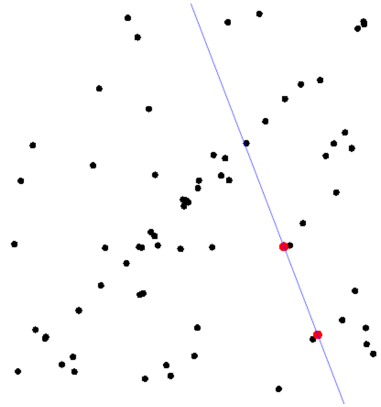
Example: line fitting



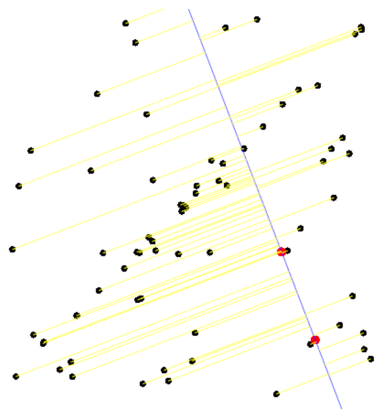
Example: line fitting



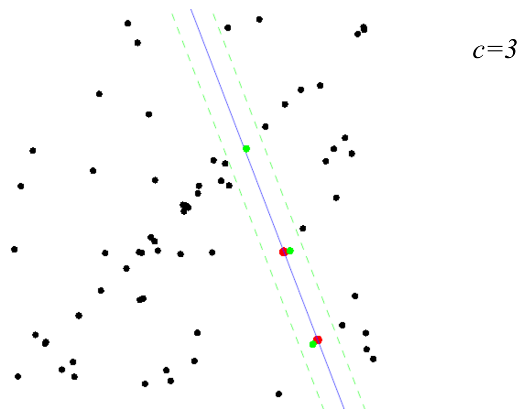
Model fitting



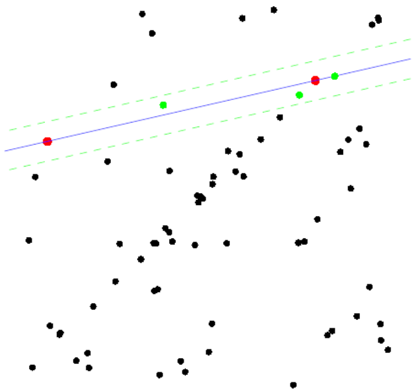
Measure distances



Count inliers

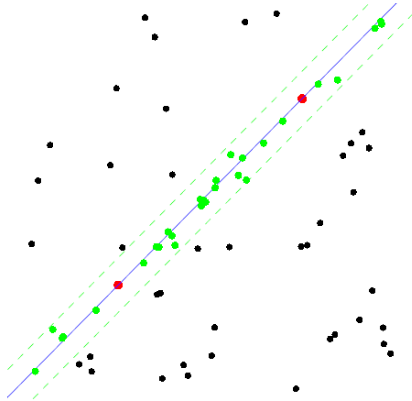


Another trial



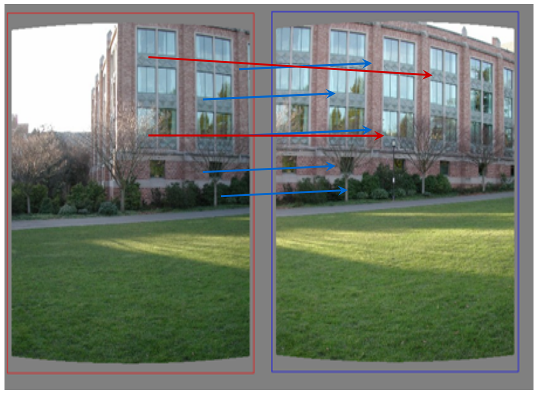
c=3

The best model



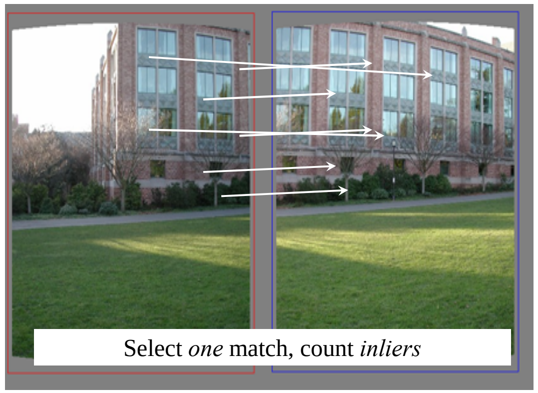
c=15

Matching features

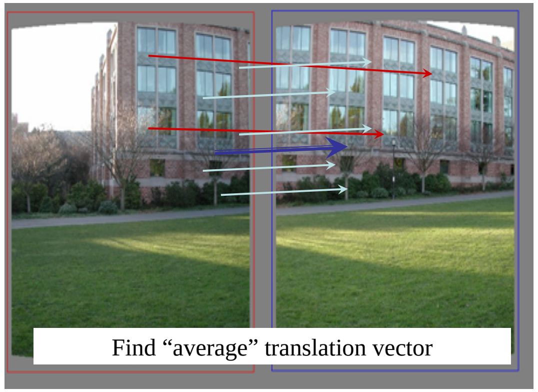


Ground Truth

RANSAC



Least squares fit



Find “average” translation vector

RANSAC algorithm

Run *k* times: — How many times?

(1) draw *n* samples randomly — How big? Smaller is better

(2) fit parameters Θ with these *n* samples

(3) for each of other *N-n* points, calculate their distance to the fitted model, count the number of inlier points, *c*

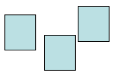
Output Θ with the largest *c*

How to define? Depends on the problem.

How to determine n ?

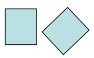
- Minimum n value depends on Model.

Translation 2dof $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$



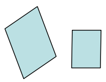
n=1

Rigid 3dof $\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$



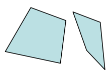
n=2

Affine 6dof $\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$



n=3

Projective 8dof $\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$



n=4

How to determine k

- *p*: probability of real inliers
- *P*: probability of success after k trials

$$P = 1 - \underbrace{\underbrace{(1 - \underbrace{p^n}_{\text{n samples are all inliers}})}_{\text{a failure}}{}^k}_{\text{failure after k trials}}$$

$$k = \frac{\log(1-P)}{\log(1-p^n)}$$

for $P=0.99$

n	*p*	*k*
3	0.5	35
6	0.6	97
6	0.5	293

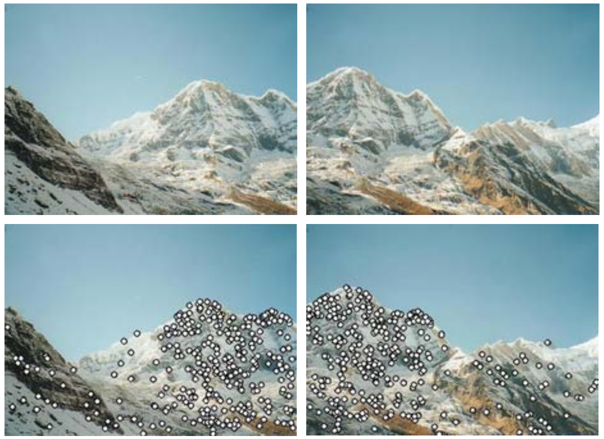
RANSAC for estimating homography

RANSAC loop:

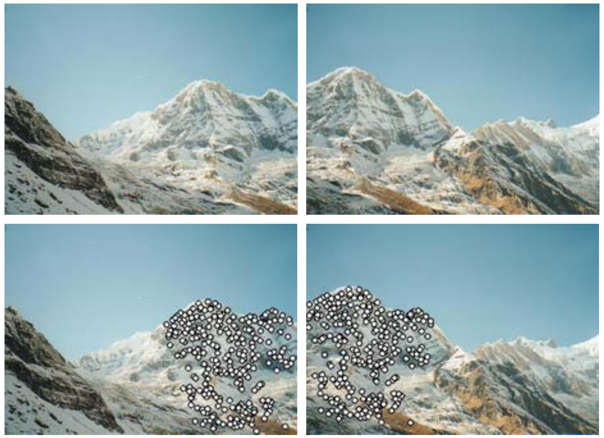
1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \mathbf{H}\, p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

RANSAC

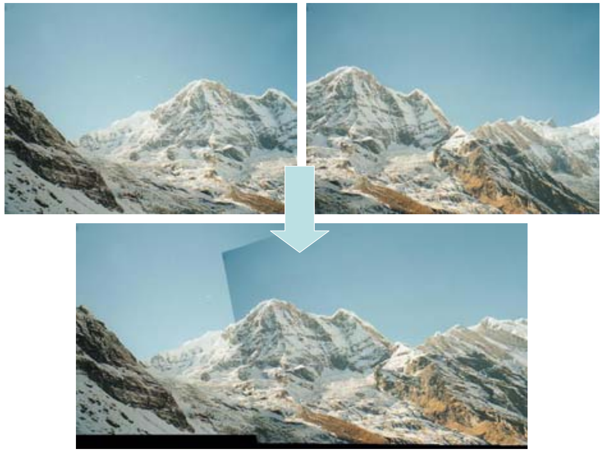
RANSAC for Homography



RANSAC for Homography

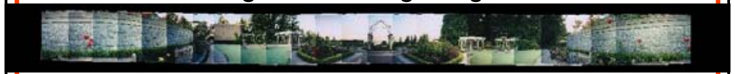


RANSAC for Homography



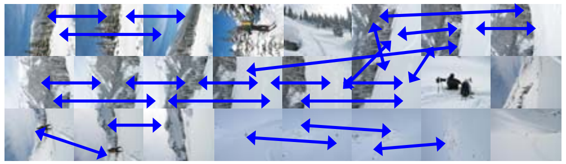
Example Application - Mosaicing

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

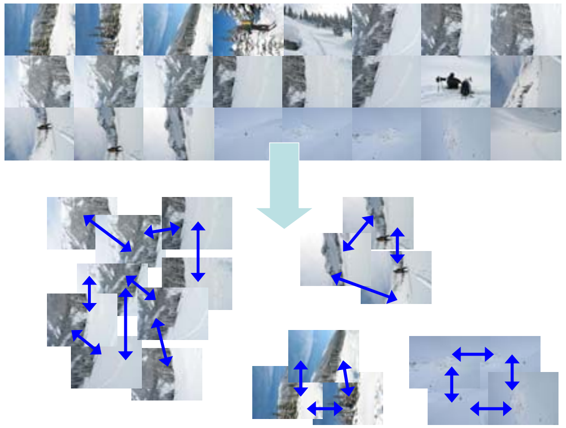


2D Rotations (θ, ϕ)
- Ordering $\nRightarrow$ matching images

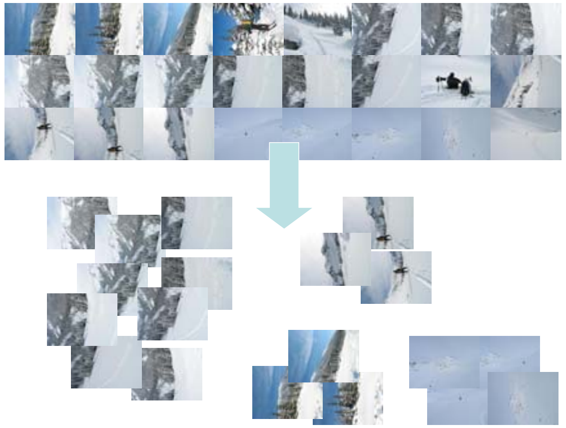
Finding the panoramas



Finding the panoramas

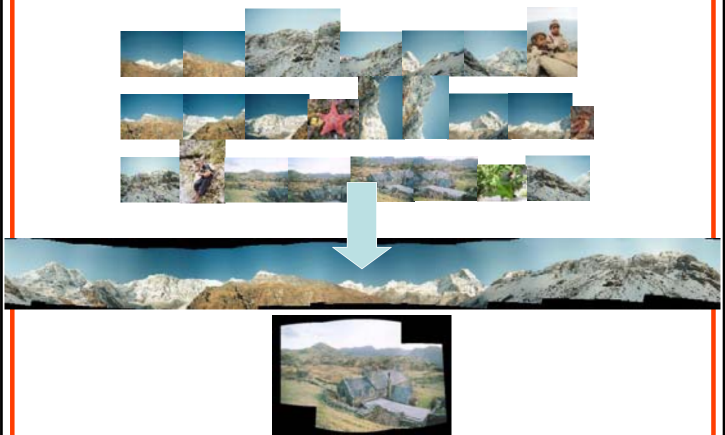


Finding the panoramas



Finding the panoramas

Results



Next Lectures

- How to build Panoramas
- How to compose seamlessly image parts

Feature Based Image Registration

- Detect feature points in both images
- Find corresponding pairs
- Compute image transformation

- Harris Corner Detection
- Sift Feature Detector
- RANSAC - Random Sample Consensus