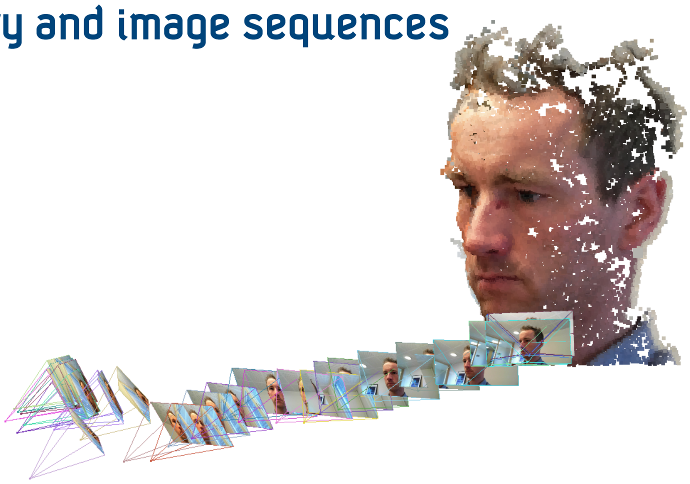
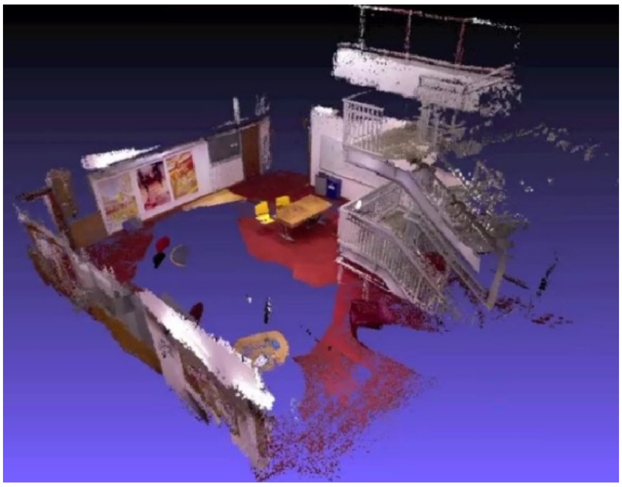
INF 5300 Advanced Topic: Video Content Analysis

Geometry and image sequences

Asbjørn Berge



Demo: Realtime 3D mapping



- Track features in 3D data from a Kinect to simultaneously map the surroundings and locate the camera.
- Fundamentally these ideas behind autonomous robot navigation.

Reading materials and tools

R. Szeliski: **Computer Vision: Algorithms and Applications**

Chapters 4.1, 6.1 and 7.1+7.2, http://szeliski.org/Book/

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

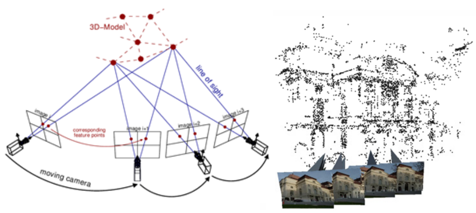
M. Zuliani: **Ransac for dummies**
http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf

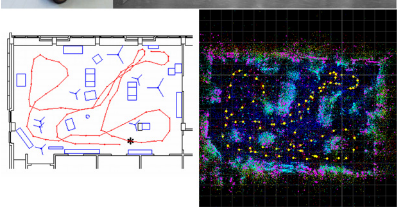
Snavely, Seitz, Szeliski, **Photo Tourism: Exploring Photo Collections in 3D**. SIGGRAPH 2006. http://phototour.cs.washington.edu/Photo_Tourism.pdf

Ransac toolbox : https://github.com/RANSAC/RANSAC-Toolbox
VlFeat toolbox : http://www.vlfeat.org
OpenCV 3D reconstruction: http://docs.opencv.org/modules/calib3d/doc/calib3d.html
VisualSfm :http://homes.cs.washington.edu/~ccwu/vsfm/
PhotoSynth: http://photosynth.net/

Inferring 3D from 2D images

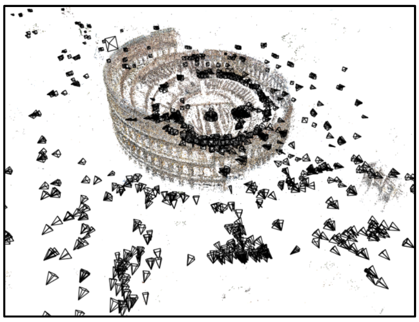
- Structure from motion
  - Obtain 3D scene structure from multiple images from the same camera in different locations, poses
  - Typically, camera location & pose treated as unknowns
  - Track points across frames, infer camera pose & scene structure from correspondences
- Simultaneous Location And Mapping (SLAM)
  - Localize a robot and map its surroundings with a single camera





3D Reconstruction

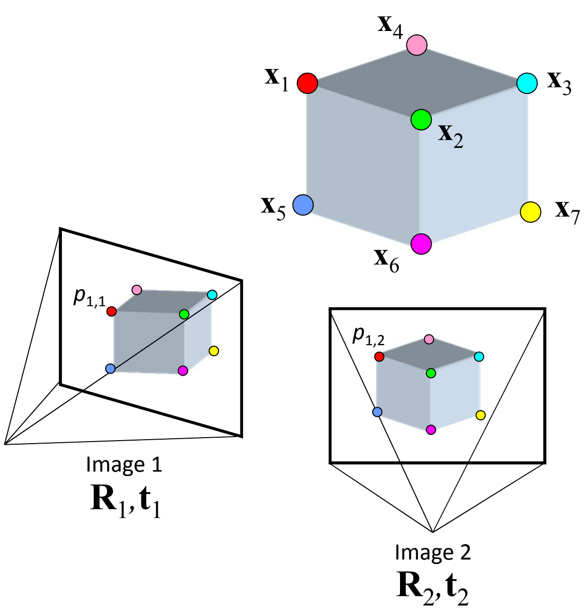
Internet Photos (“Colosseum”)



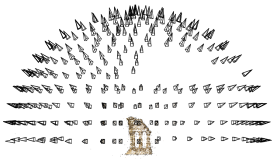
Reconstructed 3D cameras and points

http://photosynth.net/default.aspx

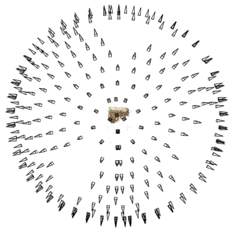
http://phototour.cs.washington.edu/applet/index.html



Structure from motion

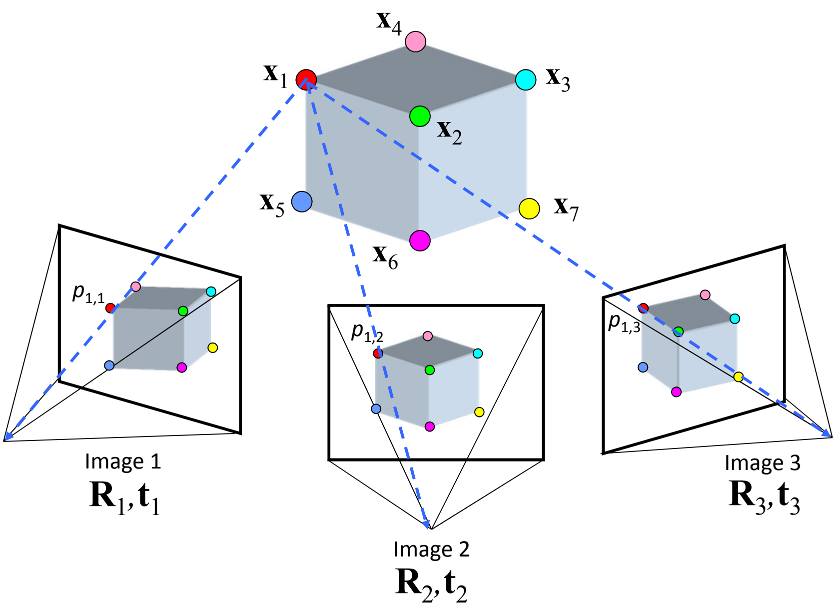


Reconstruction (side)



(top)

- Input: images with points in correspondence $p_{i,j} = (u_{i,j}, v_{i,j})$
- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$
- Objective function: minimize *reprojection error*



SINTEF Technology for a better society

SfM objective function

- Given point **x** and rotation and translation **R**, **t**

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{Rx} + \mathbf{t} \qquad u' = \frac{fx'}{z'} \quad v' = \frac{fy'}{z'} \qquad \begin{bmatrix} u' \\ v' \end{bmatrix} = \mathbf{P}(\mathbf{x}, \mathbf{R}, \mathbf{t})$$

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

SINTEF Technology for a better society

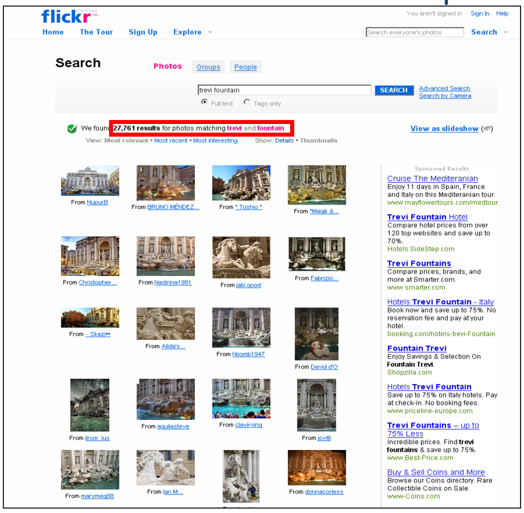
Solving structure from motion

- Minimizing g is difficult:
  - g is non-linear due to rotations, perspective division
  - lots of parameters: 3 for each 3D point, 6 for each camera
  - difficult to initialize
  - gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)
- Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)
  - Levenberg-Marquardt is one common algorithm for NLLS
  - Lourakis, **The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm**, http://www.ics.forth.gr/~lourakis/sba/
  - http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

SINTEF Technology for a better society

Photo Tourism

- Structure from motion on Internet photo collections



SINTEF

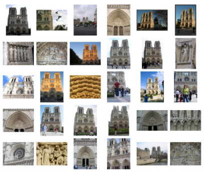
Technology for a better society

Photo Tourism

SINTEF

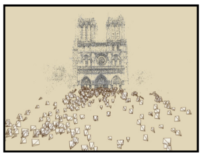
Technology for a better society

Photo Tourism overview



Input photographs

Scene reconstruction



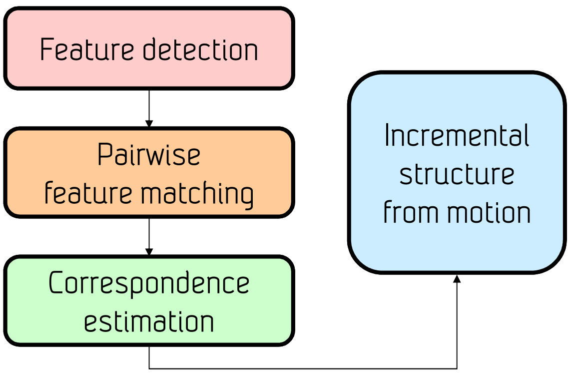
Relative camera positions and orientations

Point cloud

Sparse correspondence

Photo Explorer

Scene reconstruction



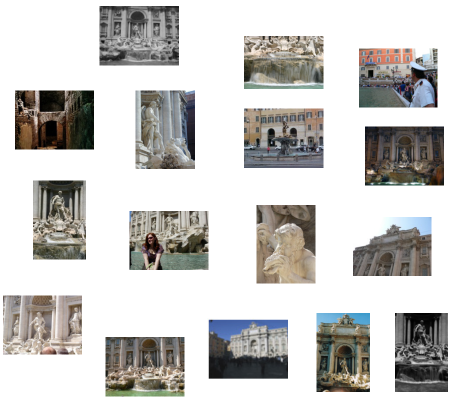
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

SINTEF Technology for a better society

Feature detection

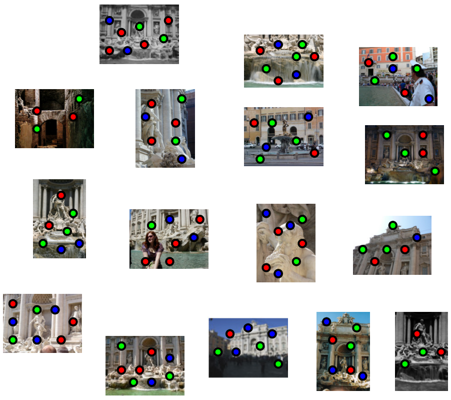
Detect features using SIFT [Lowe, IJCV 2004]



SINTEF Technology for a better society

Feature detection

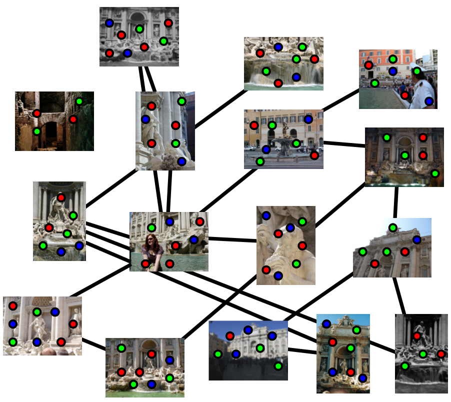
Detect features using SIFT [Lowe, IJCV 2004]



SINTEF Technology for a better society

Feature matching

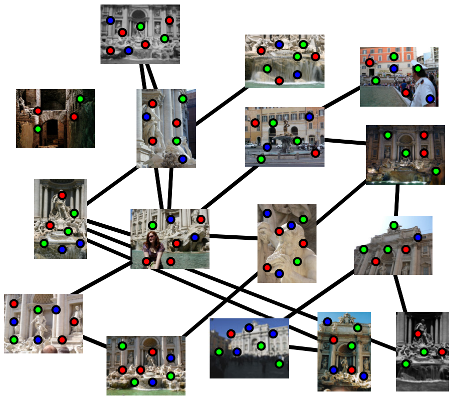
Match features between each pair of images



SINTEF Technology for a better society

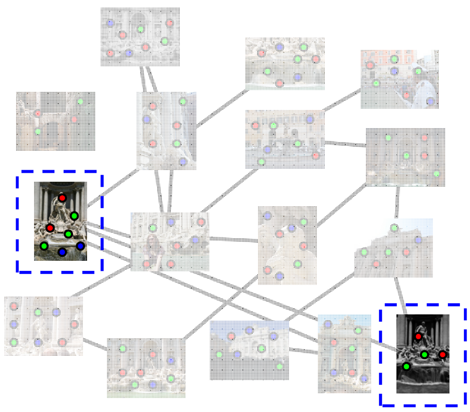
Feature matching

Refine matching using RANSAC [Fischler & Bolles 1987] to be consistent with a 3D rigid motion



SINTEF Technology for a better society

Incremental structure from motion



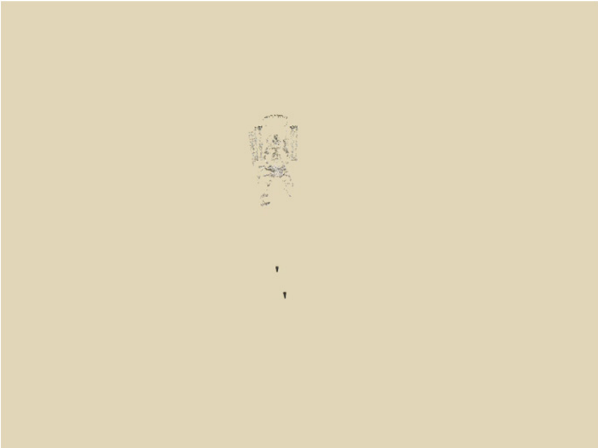
SINTEF Technology for a better society

Incremental structure from motion

SINTEF

Technology for a better society

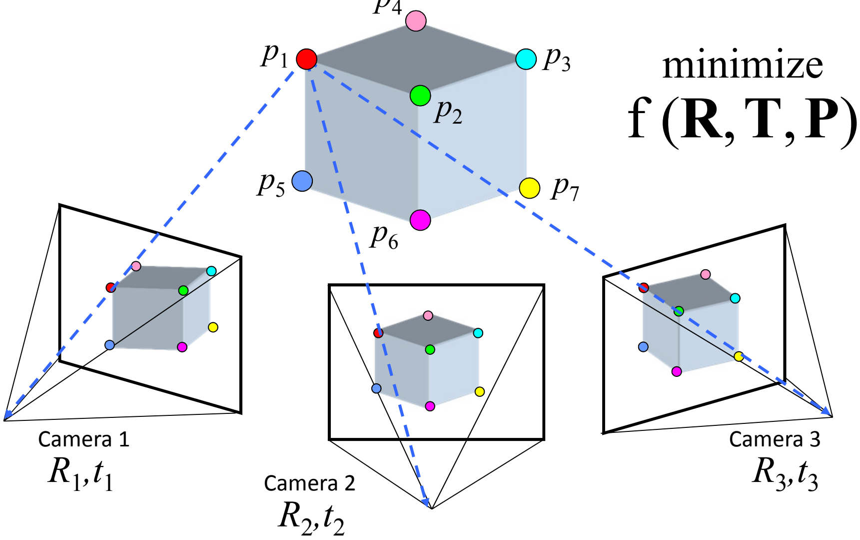
Incremental structure from motion



SINTEF

Technology for a better society

Structure from motion



SfM objective function

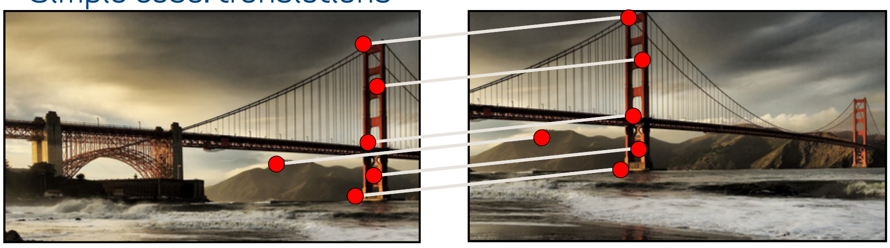
- Given point **x** and rotation and translation **R**, **t**

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{Rx} + \mathbf{t} \qquad u' = \frac{fx'}{z'} \quad v' = \frac{fy'}{z'} \qquad \begin{bmatrix} u' \\ v' \end{bmatrix} = \mathbf{P}(\mathbf{x}, \mathbf{R}, \mathbf{t})$$

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

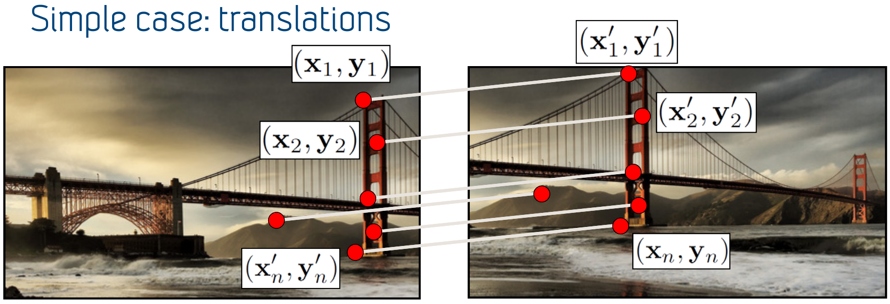
Simple case: translations



$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$?

Simple case: translations

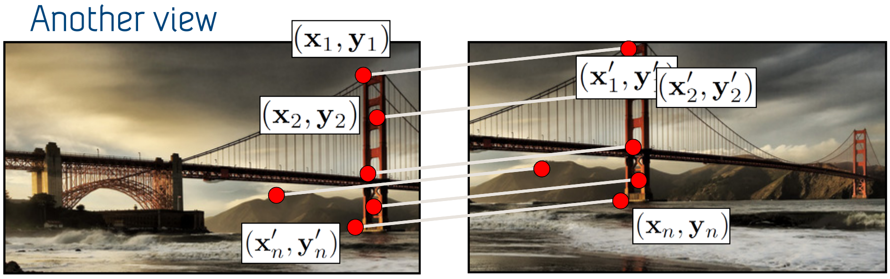


Displacement of match *i* = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

SINTEF   Technology for a better society

Another view

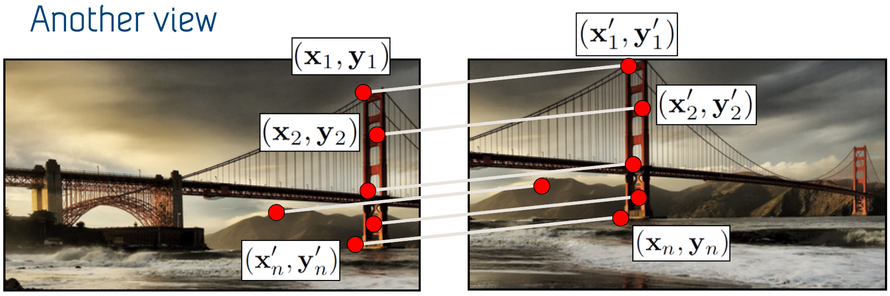


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

SINTEF   Technology for a better society

Another view



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations
  - We will find the *least squares* solution

Least squares formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- we define the *residuals* as

$$r_{\mathbf{x}_i}(\mathbf{x}_t) = (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i$$
$$r_{\mathbf{y}_i}(\mathbf{y}_t) = (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- "Least squares" solution
- For translations, is equal to mean displacement

SINTEF Technology for a better society

Least squares formulation

- Can also write as a matrix equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

SINTEF Technology for a better society

Least squares

- Find **t** that minimizes

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

- To solve, form the *normal equations*

$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{t} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Projection matrix

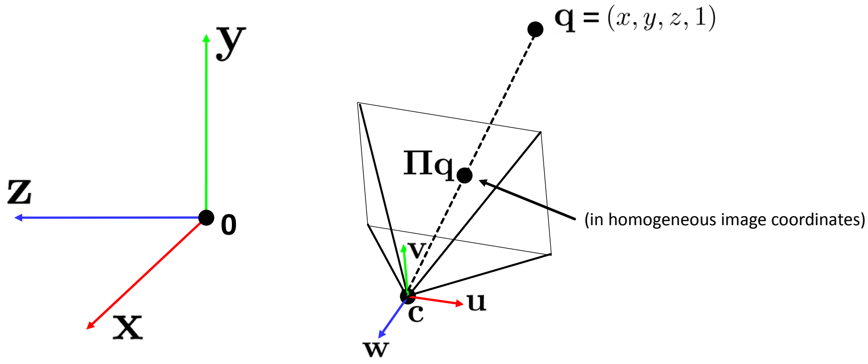
$$\mathbf{\Pi} = \underbrace{\mathbf{K}}_{\text{intrinsics}} \underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}}_{\text{translation}}}$$

$$\left[\ \mathbf{R}\ \middle|\ \underbrace{-\mathbf{R}\mathbf{c}}\ \right]$$

(**t** in book's notation)

$$\mathbf{\Pi} = \mathbf{K}\left[\ \mathbf{R}\ \middle|\ -\mathbf{R}\mathbf{c}\ \right]$$

Projection matrix

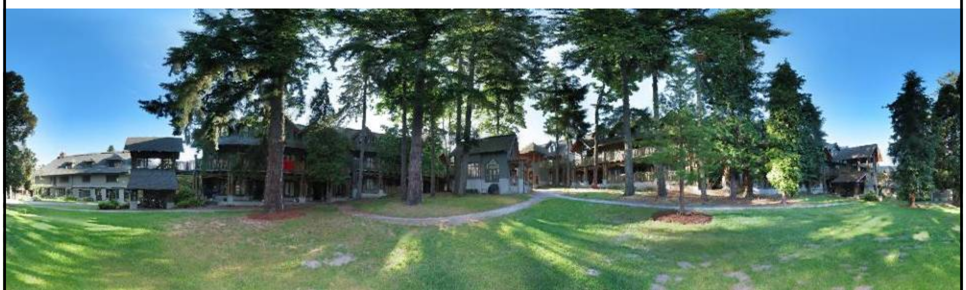


SINTEF

Technology for a better society

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°



SINTEF

Technology for a better society

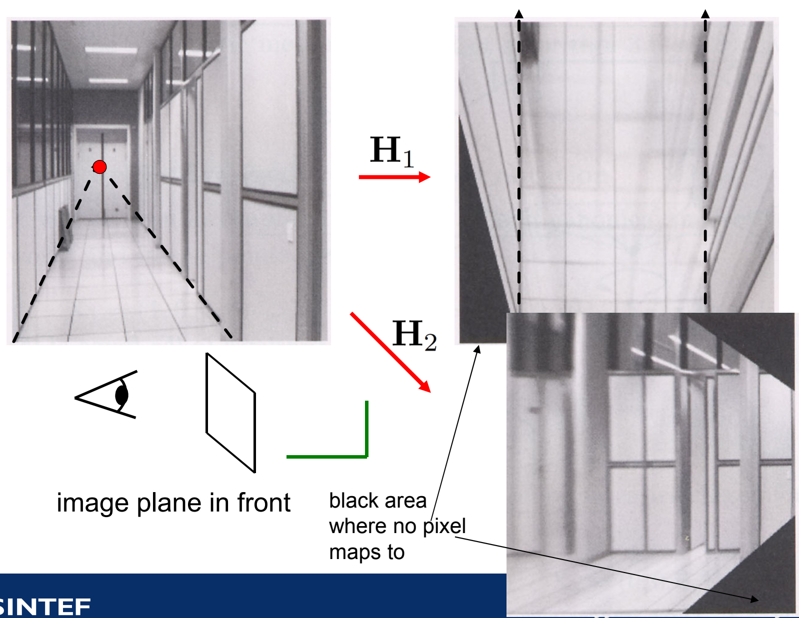
Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)

projection of 3D plane can be explained by a (homogeneous) 2D transform

Image warping with homographies



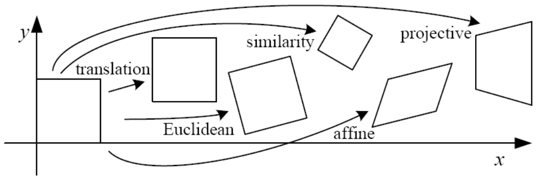
Homographies

- Homographies …
  - Affine transformations, and
  - Projective warps

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

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

2D image transformations



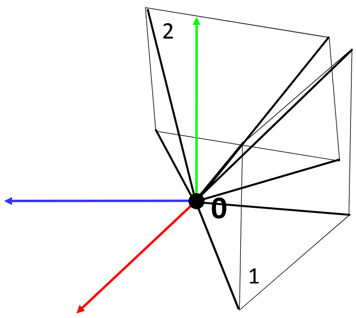
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation + ···	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths + ···	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles + ···	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism + ···	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

These transformations are a nested set of groups

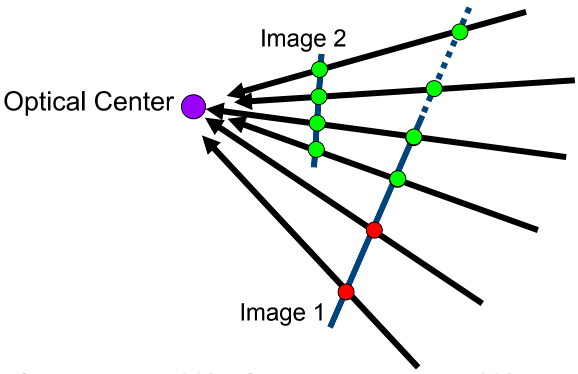
- Closed under composition and inverse is a member

SINTEF Technology for a better society

Geometric interpretation of mosaics

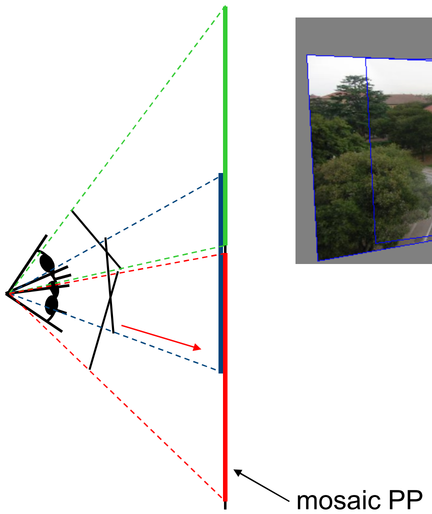


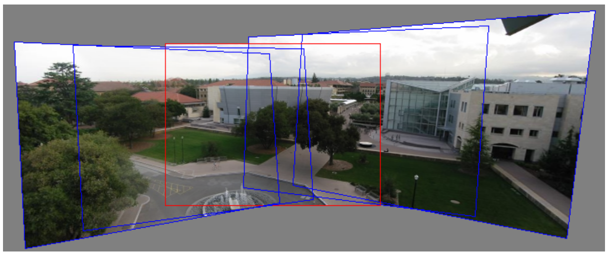
Geometric Interpretation of Mosaics



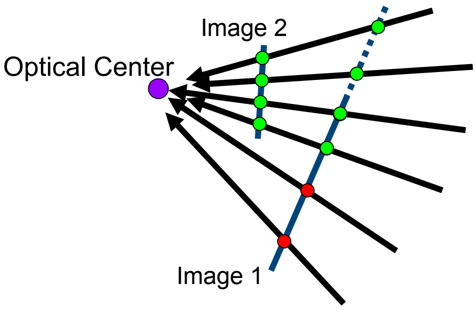
- If we capture all 360º of rays, we can create a 360º panorama
- The basic operation is projecting an image from one plane to another
- The projective transformation is scene-INDEPENDENT
  - This depends on all the images having the same optical center
    - http://archive.bigben.id.au/tutorials/360/photo/nodal.html

Projecting images onto a common plane





What is the transformation?



How do we transform image 2 onto image 1's projection plane?

image 1	image 2
$\mathbf{K}_1$	$\mathbf{K}_2$
$\mathbf{R}_1 = \mathbf{I}_{3\times3}$	$\mathbf{R}_2$

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

3D ray coords (in camera 2) — image coords (in image 2)

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{R}_1 \mathbf{R}_2^T \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

3D ray coords (in camera 1) — image coords (in image 2)

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \mathbf{K}_1 \mathbf{R}_1 \mathbf{R}_2^T \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

image coords (in image 1) — **3x3 homography** — image coords (in image 2)

Image alignment

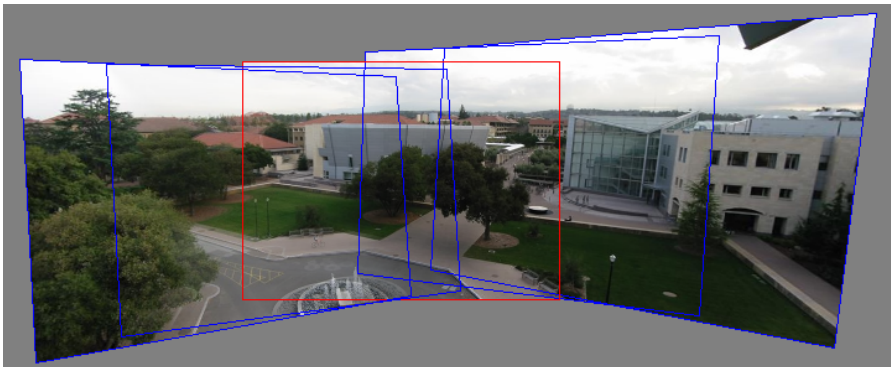
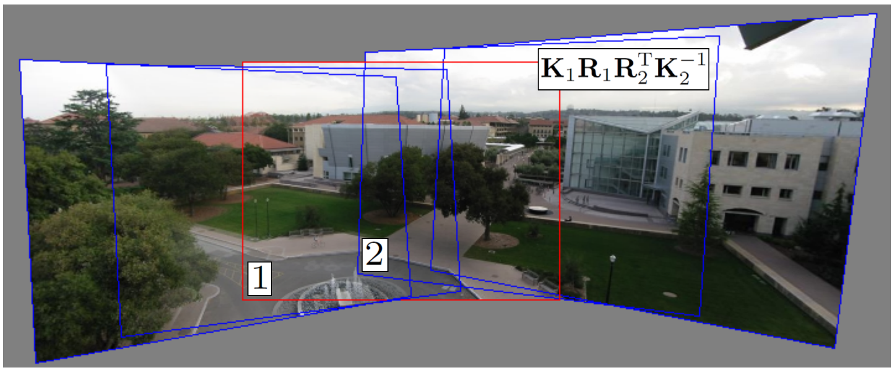


Image alignment



Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine transformations

- Residuals:

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

$$C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

SINTEF Technology for a better society

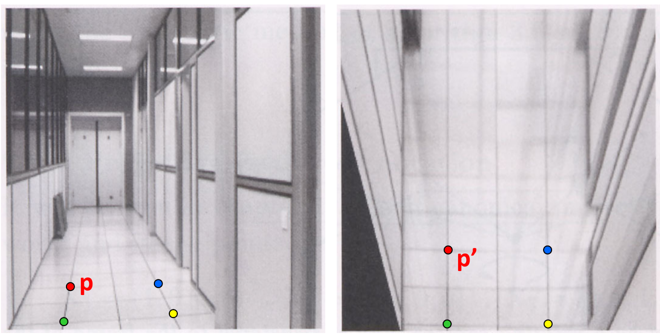
Affine transformations

- Matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\underset{2n \times 6}{\mathbf{A}} \quad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

Homographies



To unwarp (rectify) an image

- solve for homography **H** given **p** and **p’**
- solve equations of the form: w**p’** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Image Alignment Algorithm

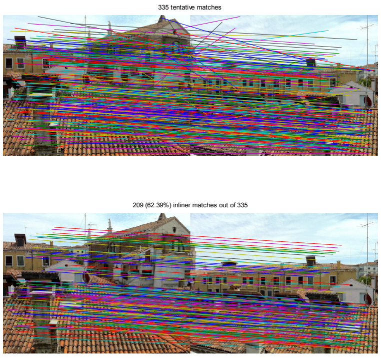
Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

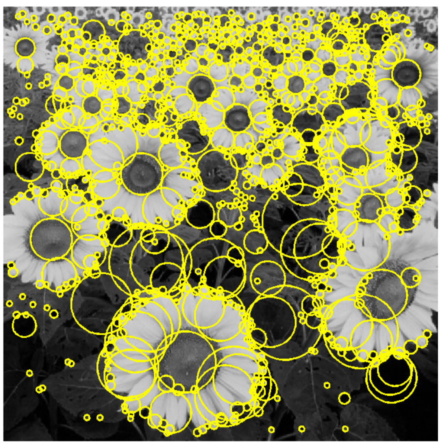
What could go wrong?

Robustness

VLFeat demo of Ransac Homography fit



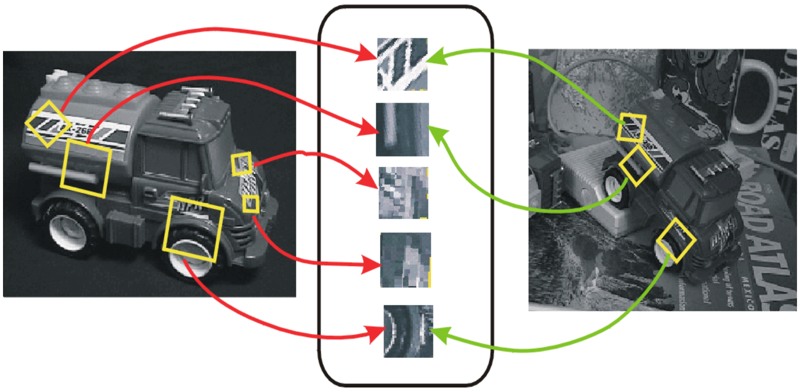
Feature extraction: Corners and blobs



Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Feature Descriptors

Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

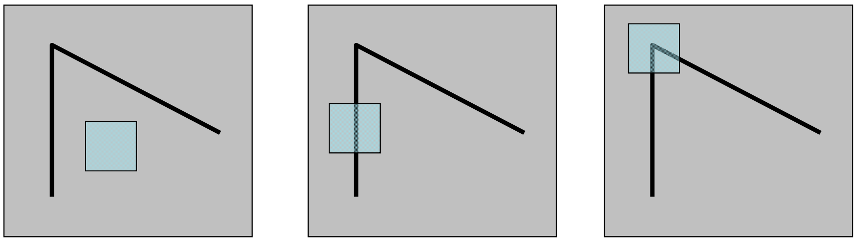
Efficiency

- real-time performance achievable

Local measures of uniqueness

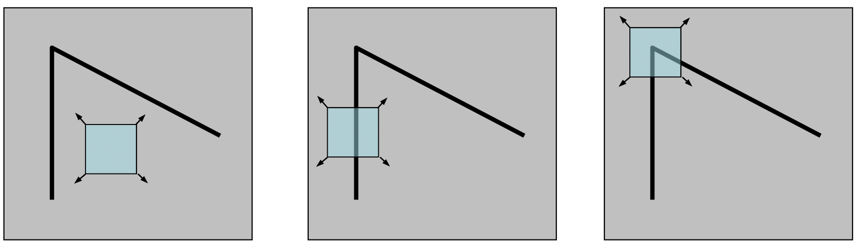
Suppose we only consider a small window of pixels

– What defines whether a feature is a good or bad candidate?



Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions

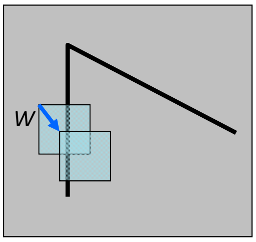
"edge":
no change along the edge direction

"corner":
significant change in all directions

Harris corner detection: the math

Consider shifting the window *W* by (*u*,*v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" *E*(*u*,*v*):



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

Small motion assumption

Taylor Series expansion of *I*:

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

If the motion (u,v) is small, then first order approximation is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

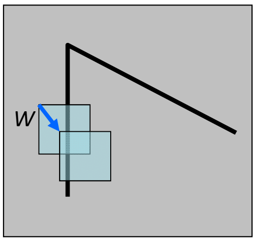
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Corner detection: the math

Consider shifting the window *W* by (*u*,*v*)

- define an SSD "error" *E(u,v)*:



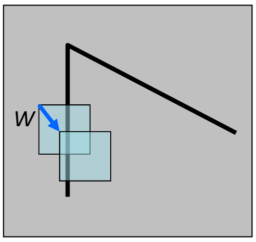
$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

SINTEF Technology for a better society

Corner detection: the math

Consider shifting the window *W* by (*u*,*v*)

- define an SSD "error" *E(u,v)*:



$$
\begin{aligned}
E(u,v) &\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\
&\approx Au^2 + 2Buv + Cv^2
\end{aligned}
$$

$$
A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2
$$

- Thus, *E*(*u*,*v*) is locally approximated as a quadratic error function

SINTEF Technology for a better society

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

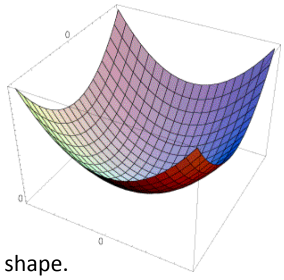
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



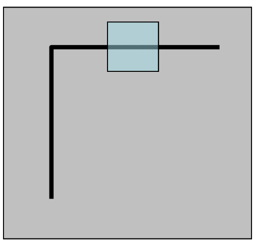
Let's try to understand its shape.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

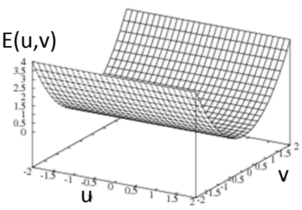
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

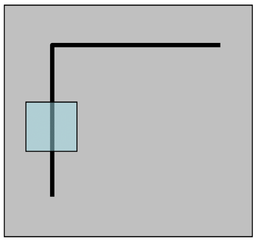


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

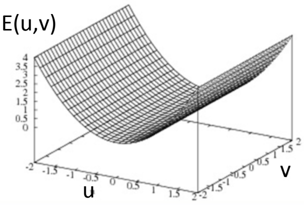
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

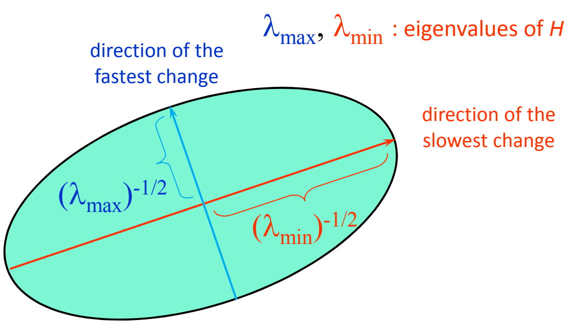


General case

We can visualize *H* as an ellipse with axis lengths determined by the *eigenvalues* of *H* and orientation determined by the *eigenvectors* of *H*

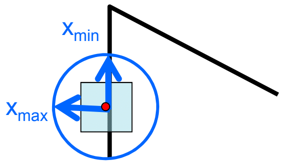
Ellipse equation:

$$[u \;\; v] \;\; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



Corner detection: the math

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



$$Hx_{\max} = \lambda_{\max} x_{\max}$$

$$Hx_{\min} = \lambda_{\min} x_{\min}$$

Eigenvalues and eigenvectors of H

- Define shift directions with the smallest and largest change in error
- x_{max} = direction of largest increase in *E*
- λ_{max} = amount of increase in direction x_{max}
- x_{min} = direction of smallest increase in *E*
- λ_{min} = amount of increase in direction x_{min}

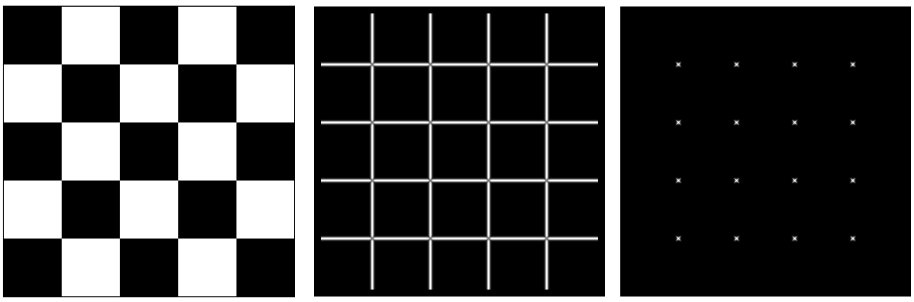
Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want *E*(*u*,*v*) to be large for small shifts in all directions

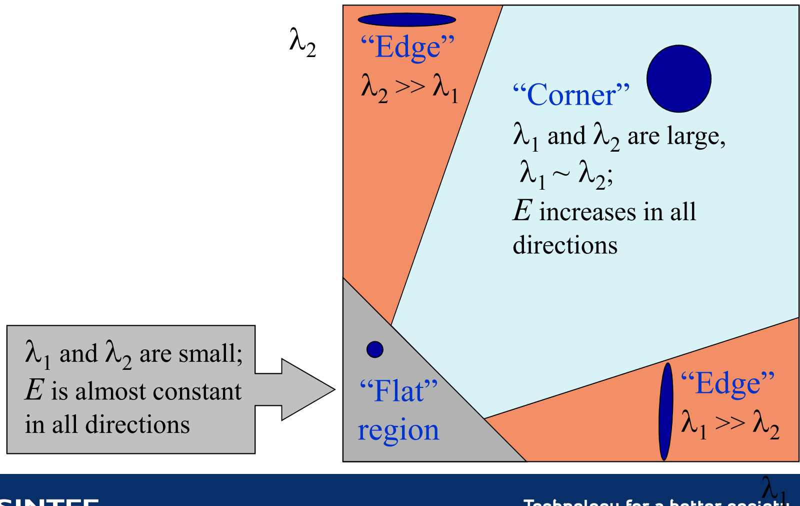
- the minimum of *E*(*u*,*v*) should be large, over all unit vectors [*u v*]
- this minimum is given by the smaller eigenvalue (λ_{min}) of *H*



I $\lambda_{\max}$ $\lambda_{\min}$

Interpreting the eigenvalues

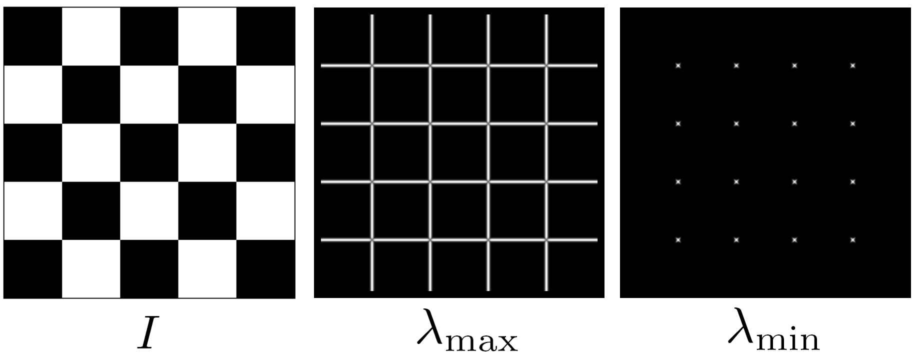
Classification of image points using eigenvalues of *M*:



SINTEF Technology for a better society

Corner detection summary

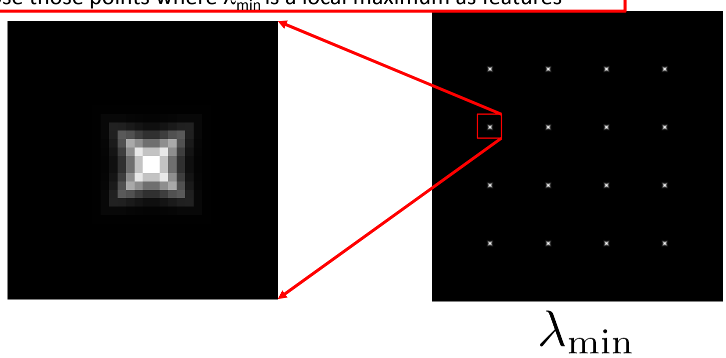
- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



SINTEF Technology for a better society

Corner detection summary

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



The Harris operator

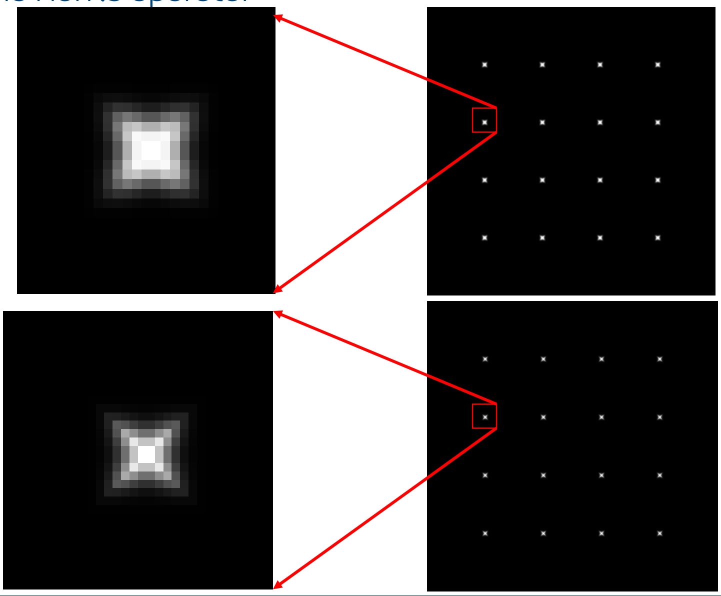
λ_{min} is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

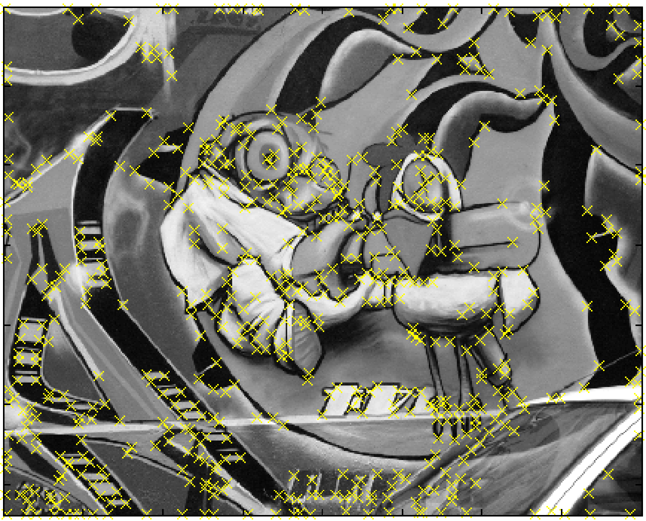
- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to λ_{min} but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator



SINTEF Technology for a better society

Harris Detector – Responses [Harris88]



SINTEF Technology for a better society

Weighting the derivatives

- In practice, using a simple window *W* doesn't work too well

$$H = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

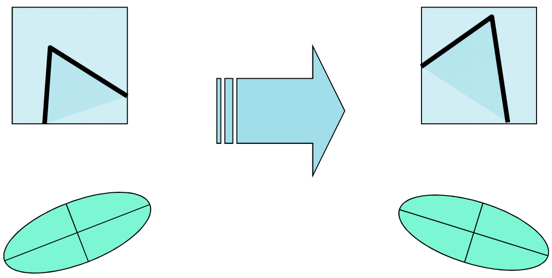
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y) \in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$w_{x,y}$

Harris Detector: Invariance Properties

- Rotation



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

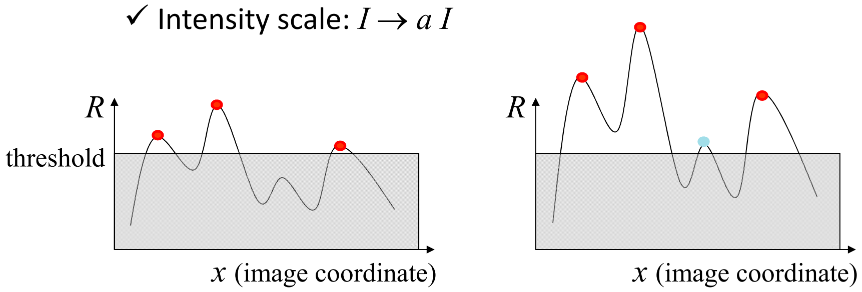
Corner response is invariant to image rotation

Harris Detector: Invariance Properties

- Affine intensity change: $I \rightarrow aI + b$

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

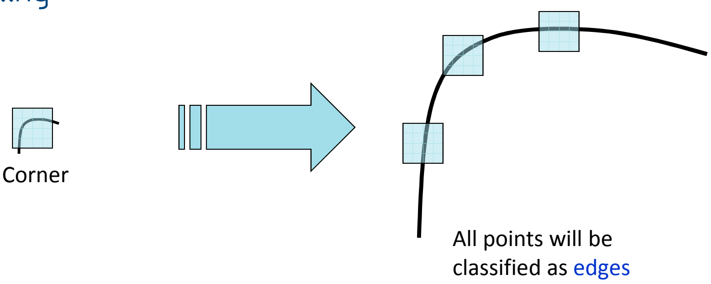
✓ Intensity scale: $I \rightarrow a\,I$



Partially invariant to affine intensity change

Harris Detector: Invariance Properties

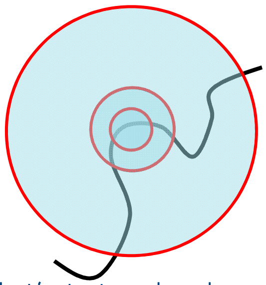
- Scaling



Not invariant to scaling

Scale invariant detection

Suppose you're looking for corners



Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Harris operator

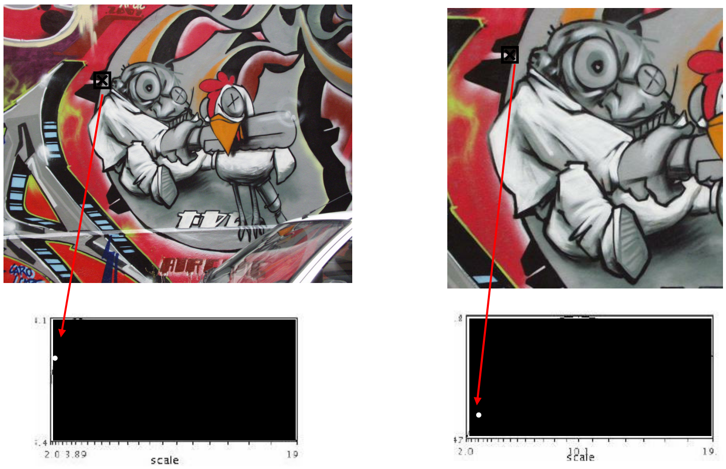
Automatic Scale Selection

$$f(I_{i_1 \ldots i_m}(x, \sigma)) = f(I_{i_1 \ldots i_m}(x', \sigma'))$$

Same operator responses if the patch contains the same image up to scale factor. How to find corresponding patch sizes?

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



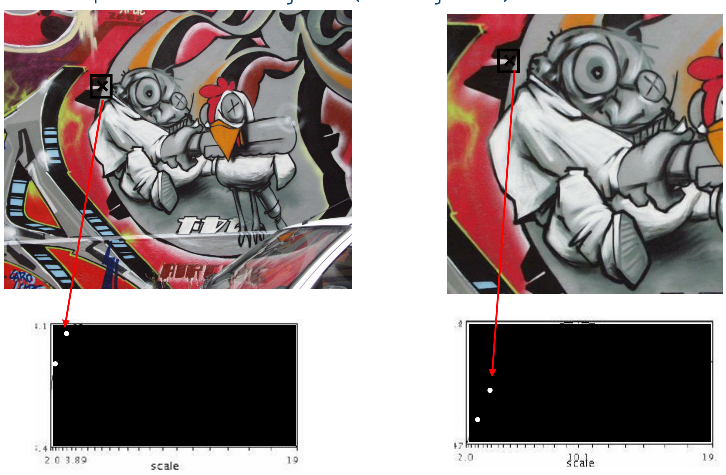
$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

SINTEF

Technology for a better society

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



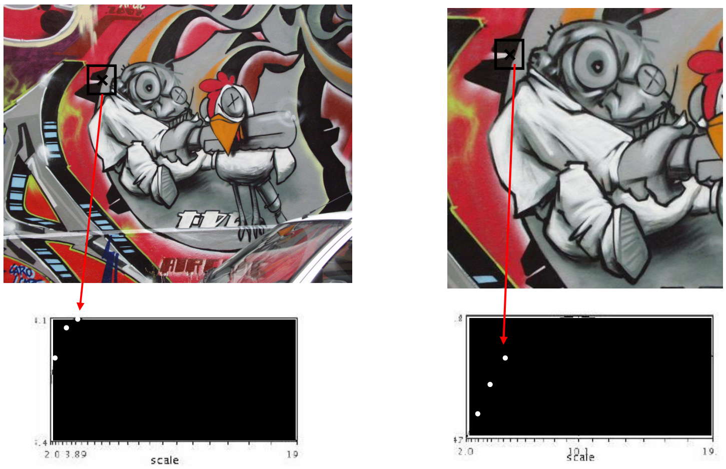
$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

SINTEF

Technology for a better society

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

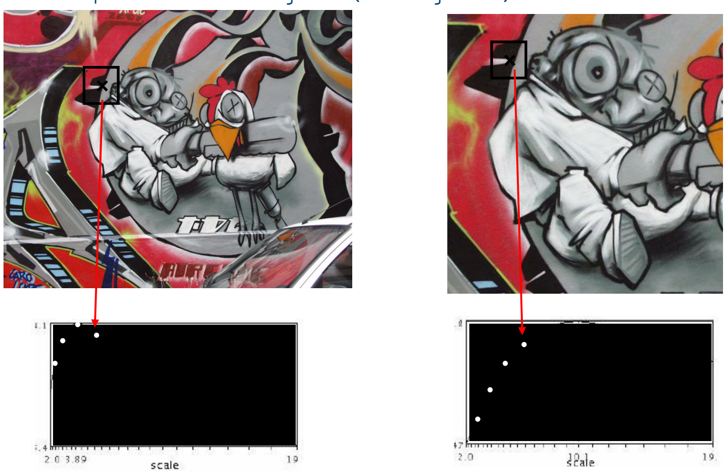


$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma))$

SINTEF Technology for a better society

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

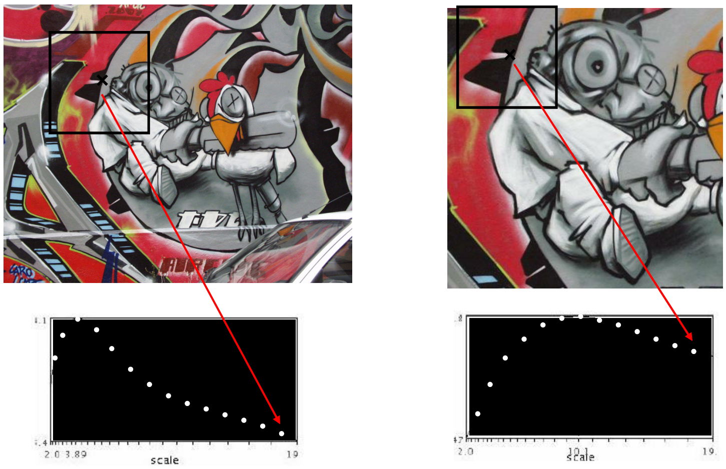


$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma))$

SINTEF Technology for a better society

Automatic Scale Selection

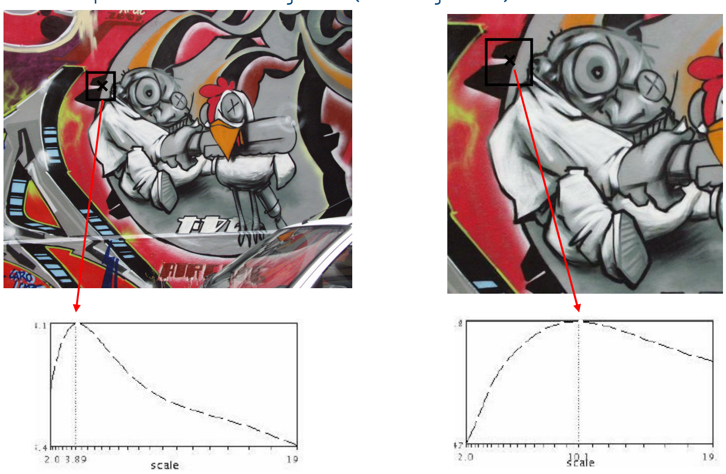
- Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma'))$

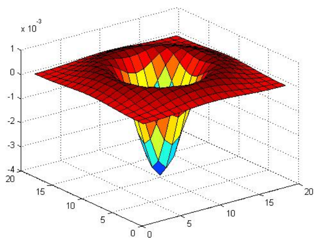
Implementation

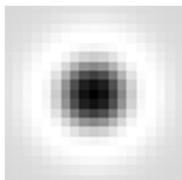
- Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid

(sometimes need to create in-between levels, e.g. a ¾-size image)

Another common definition of *f*

- The *Laplacian of Gaussian* (*LoG*)





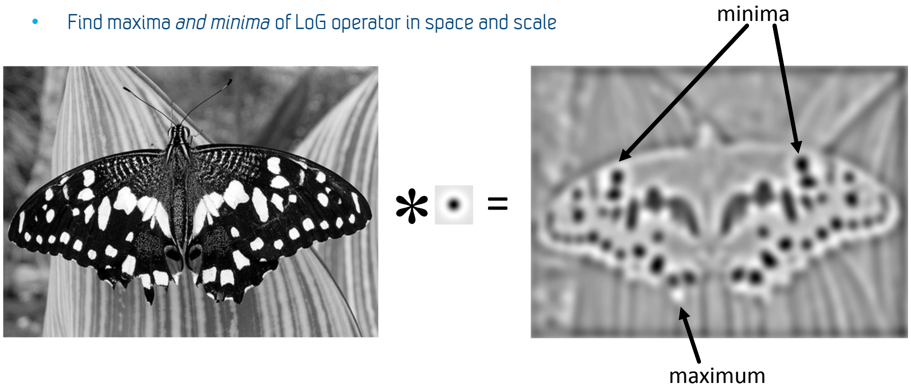
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

(very similar to a Difference of Gaussians (DoG) – i.e. a Gaussian minus a slightly smaller Gaussian)

SINTEF Technology for a better society

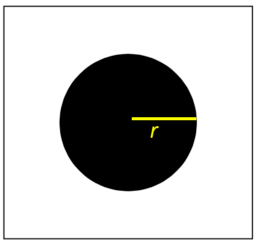
Laplacian of Gaussian

- "Blob" detector
- Find maxima *and minima* of LoG operator in space and scale

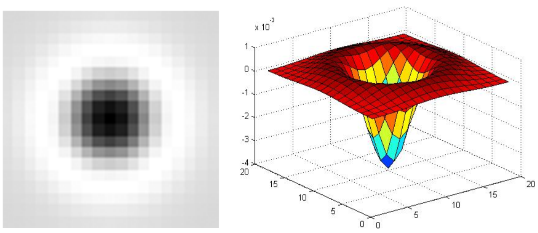


Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?

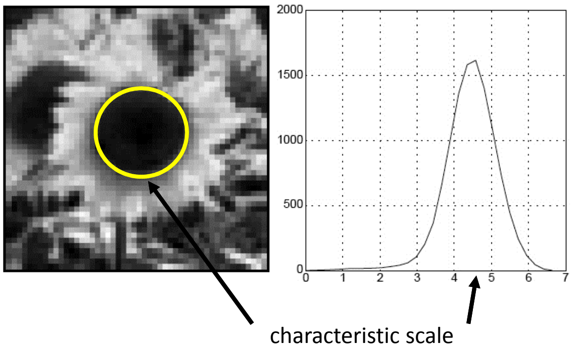


image


Laplacian

Characteristic scale

- We define the characteristic scale as the scale that produces peak of Laplacian response



T. Lindeberg (1998). ["Feature detection with automatic scale selection."](#) *International Journal of Computer Vision* **30** (2): pp 77--116.

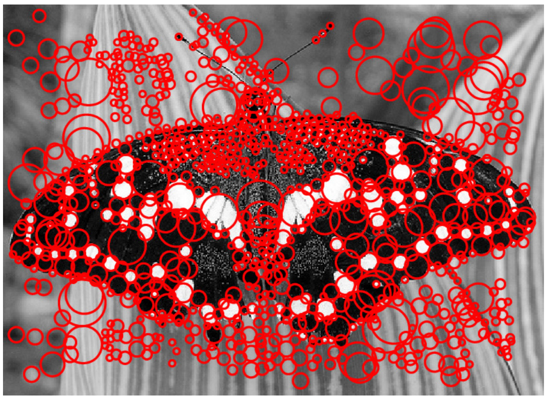
Scale-space blob detector: Example

Scale-space blob detector: Example

sigma = 11.9912

SINTEF Technology for a better society

Scale-space blob detector: Example



SINTEF Technology for a better society

Scale Invariant Feature Transform (SIFT)

1. Take a 16 x16 window around interest point (i.e., at the scale detected).
2. Divide into a 4x4 grid of cells.
3. Compute histogram of image gradients in each cell (8 bins each).

16 histograms x 8 orientations
= 128 features

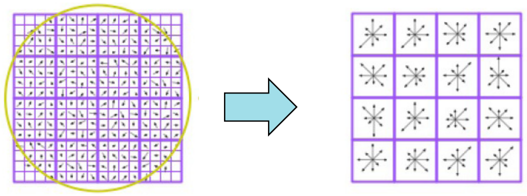


Image gradients Keypoint descriptor

SIFT Computation – Steps

(1) Scale-space extrema detection
- Extract scale and rotation invariant interest points (i.e., keypoints).

(2) Keypoint localization
- Determine location and scale for each interest point.
- Eliminate "weak" keypoints

(3) Orientation assignment
- Assign one or more orientations to each keypoint.

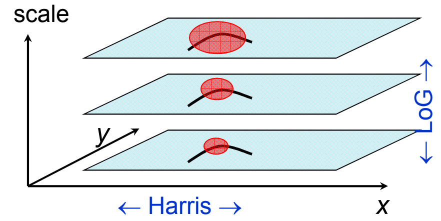
(4) Keypoint descriptor
- Use local image gradients at the selected scale.

D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", **International Journal of Computer Vision**, 60(2):91-110, 2004.

Cited 13629 times (as of 17/4/2012)

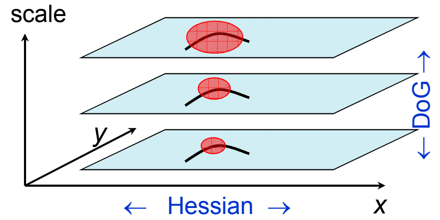
Scale-space Extrema Detection

- Harris-Laplace
- *Find local maxima of:*
  - Harris detector in space
  - LoG in scale



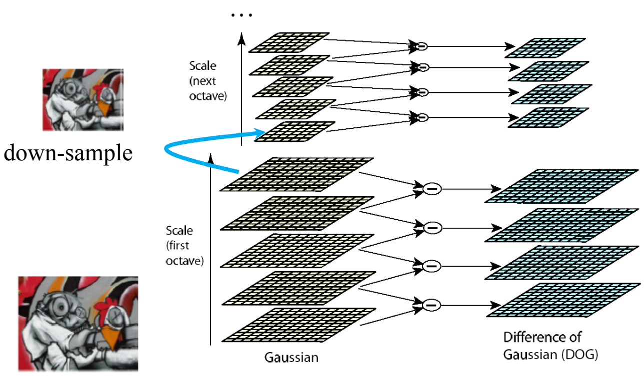
- SIFT

  Find local maxima of:
  - Hessian in space
  - DoG in scale



Scale-space Extrema Detection

- DoG images are grouped by octaves (i.e., doubling of σ_0)
- Fixed number of levels per octave



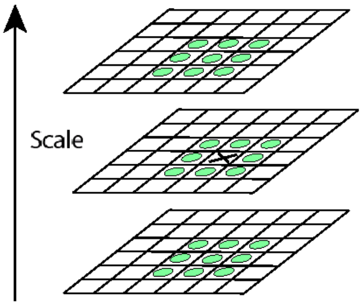
$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

where

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

Scale-space Extrema Detection

- Extract local extrema (i.e., minima or maxima) in DoG pyramid.
  - -Compare each point to its 8 neighbors at the same level, 9 neighbors in the level above, and 9 neighbors in the level below (i.e., 26 total).



SINTEF Technology for a better society

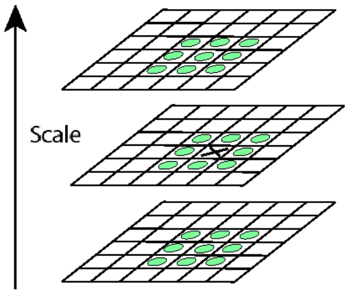
Keypoint Localization

- Determine the location and scale of keypoints to **sub-pixel** and **sub-scale** accuracy by fitting a 3D quadratic polynomial:

$$X_i = (x_i, y_i, \sigma_i)$$ keypoint location

$$\Delta X = (x - x_i, y - y_i, \sigma - \sigma_i)$$ offset

$$X_i \leftarrow X_i + \Delta X$$ sub-pixel, sub-scale Estimated location



Substantial improvement to matching and stability!

SINTEF Technology for a better society

Keypoint Localization

- Use Taylor expansion to locally approximate D(x,y,σ) (i.e., DoG function) and estimate Δx:

$$D(\Delta X) = D(X_i) + \frac{\partial D^T(X_i)}{\partial \mathrm{X}} \Delta \mathrm{X} + \frac{1}{2} \Delta \mathrm{X}^T \frac{\partial^2 D(X_i)}{\partial \mathrm{X}^2} \Delta \mathrm{X}$$

- Find the extrema of D(ΔX):

$$\frac{\partial D(X_i)}{\partial X} + \frac{\partial^2 D(X_i)}{\partial X^2} \Delta X = 0$$

SINTEF Technology for a better society

Keypoint Localization

$$\frac{\partial^2 D(X_i)}{\partial X^2} \Delta X = -\frac{\partial D(X_i)}{\partial X} \rightarrow \Delta X = -\frac{\partial^2 D^{-1}(X_i)}{\partial X^2} \frac{\partial D(X_i)}{\partial X}$$

- ΔX can be computed by solving a 3x3 linear system:

$$\begin{bmatrix} \frac{\partial^2 D}{\partial \sigma^2} & \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial \sigma x} \\ \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial yx} \\ \frac{\partial^2 D}{\partial \sigma x} & \frac{\partial^2 D}{\partial yx} & \frac{\partial^2 D}{\partial x^2} \end{bmatrix} \begin{bmatrix} \Delta \sigma \\ \Delta y \\ \Delta x \end{bmatrix} = - \begin{bmatrix} \frac{\partial D}{\partial \sigma} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial x} \end{bmatrix}$$

$$\frac{\partial D}{\partial \sigma} = \frac{D_{k+1}^{i,j} - D_{k-1}^{i,j}}{2}$$

$$\frac{\partial^2 D}{\partial \sigma^2} = \frac{D_{k-1}^{i,j} - 2D_k^{i,j} + D_{k+1}^{i,j}}{1}$$

$$\frac{\partial^2 D}{\partial \sigma y} = \frac{(D_{k+1}^{i+1,j} - D_{k-1}^{i+1,j}) - (D_{k+1}^{i-1,j} - D_{k-1}^{i-1,j})}{4}$$

use finite differences:

If $\Delta X > 0.5$ in any dimension, repeat.

SINTEF Technology for a better society

Keypoint Localization

- Reject keypoints having low contrast.
  - i.e., sensitive to noise

If $|D(X_i + \Delta X)| < 0.03$ reject keypoint

– i.e., assumes that image values have been normalized in [0,1]

SINTEF Technology for a better society

Keypoint Localization

- Reject points lying on edges (or being close to edges)
- Harris uses the auto-correlation matrix:

$$A_W(x,y) = \sum_{x \in W, y \in W} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix}$$

$R(A_W) = \det(A_W) - \alpha \, \text{trace}^2(A_W)$

or $R(A_W) = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$

SINTEF Technology for a better society

Keypoint Localization

- SIFT uses the Hessian matrix (for efficiency).
  - i.e., Hessian encodes principal curvatures

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

α: largest eigenvalue (λ_{max})
β: smallest eigenvalue (λ_{min})
(proportional to principal curvatures)

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

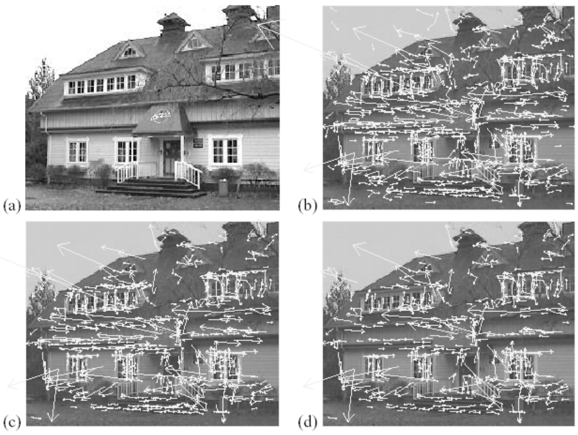
$$\rightarrow \frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

$(r = \alpha/\beta)$

Reject keypoint if: $\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$ (SIFT uses r = 10)

SINTEF Technology for a better society

Keypoint Localization



(**a**) 233x189 image

(**b**) 832 DoG extrema

(**c**) 729 left after low contrast threshold

(**d**) 536 left after testing ratio based on Hessian

SINTEF Technology for a better society

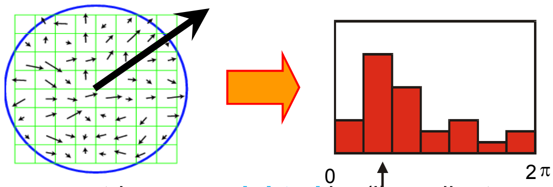
Orientation Assignment

- Create histogram of gradient directions, within a region around the keypoint, at selected scale:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = a\tan 2((L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)))$$

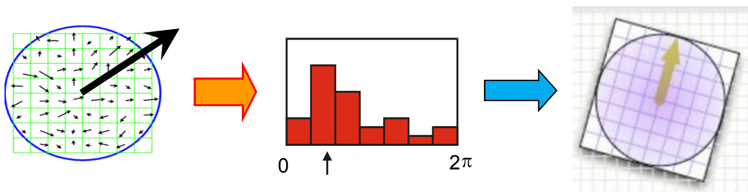


36 bins (i.e., 10º per bin)

- Histogram entries are **weighted** by (i) gradient magnitude and (ii) a Gaussian function with σ equal to 1.5 times the scale of the keypoint.

Orientation Assignment

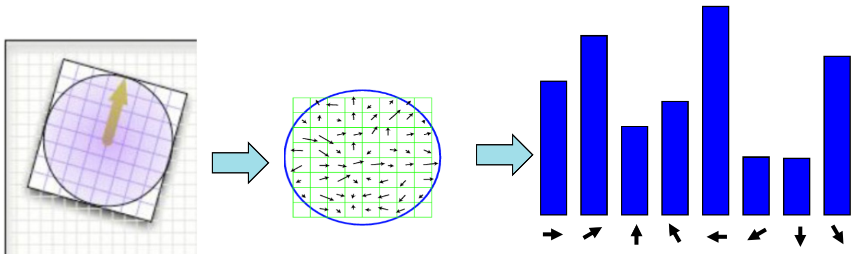
- Assign canonical orientation at peak of smoothed histogram (fit parabola to better localize peak).



- In case of peaks within 80% of highest peak, multiple orientations assigned to keypoints.
  - About 15% of keypoints has multiple orientations assigned.
  - Significantly improves stability of matching.

SINTEF Technology for a better society

Keypoint Descriptor

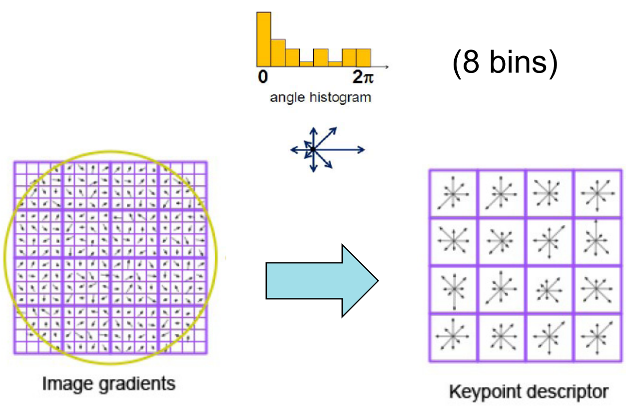


Keypoint Descriptor

1. Take a 16 x16 window around detected interest point.

2. Divide into a 4x4 grid of cells.

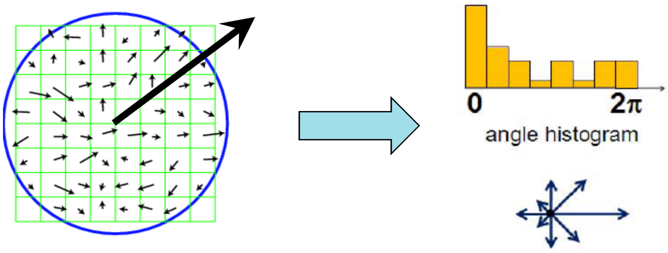
3. Compute histogram in each cell.



16 histograms x 8 orientations
= 128 features

Keypoint Descriptor

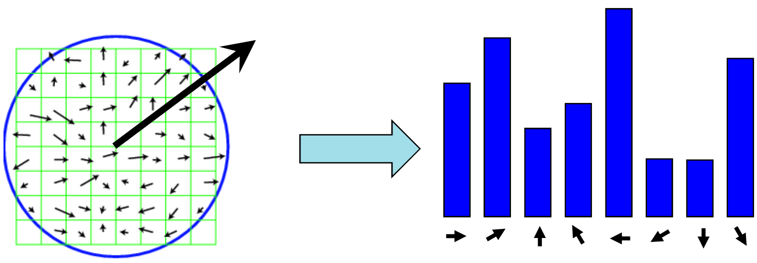
- Each histogram entry **is** **weighted** by (i) gradient magnitude and (ii) a Gaussian function with σ equal to 0.5 times the width of the descriptor window.



SINTEF Technology for a better society

Keypoint Descriptor

- Partial Voting: distribute histogram entries into adjacent bins (i.e., additional robustness to shifts)
  - Each entry is added to all bins, multiplied by a weight of 1-d, where d is the distance from the bin it belongs.

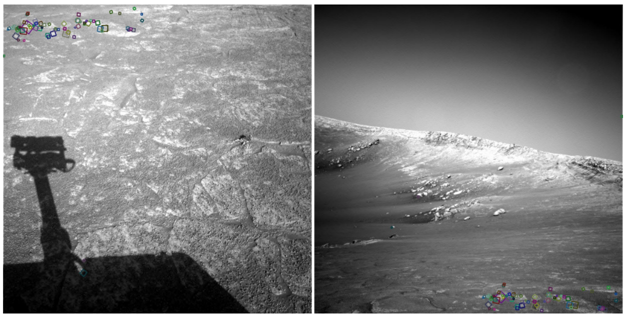


SINTEF Technology for a better society

Properties of SIFT

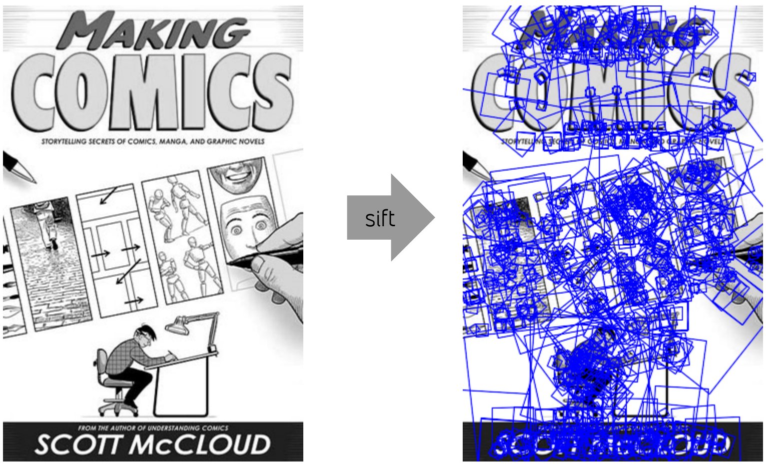
Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



NASA Mars Rover images with SIFT feature matches

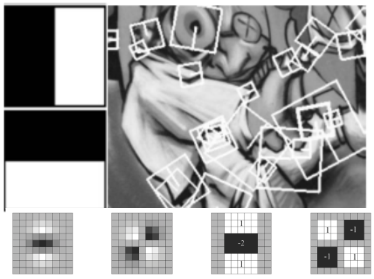
SIFT Example



868 SIFT features

SINTEF Technology for a better society

Local Descriptors: SURF



Fast approximation of SIFT idea

- Efficient computation by 2D box filters & integral images
  $\Rightarrow$ 6 times faster than SIFT
- Equivalent quality for object identification

GPU implementation available

- Feature extraction @ 100Hz
  (detector + descriptor, 640 × 480 img)
  http://www.vision.ee.ethz.ch/~surf

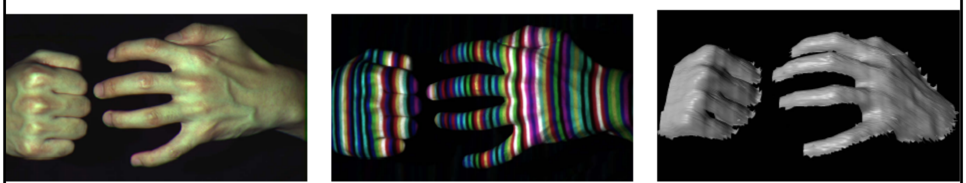
[Bay, ECCV'06], [Cornelis, CVGPU'08]

Main points of this lecture

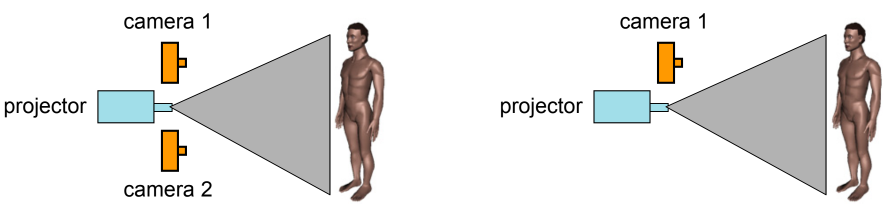
- Moving the same camera restricts the geometry allowing inference about 3D
  Potential uses range from mosaicing to egomotion estimation
  - In principle the same mechanism that human depth perception is based on
- Stereo / multiview stereo. You should be able to describe the concepts.
- Remember the RANSAC algorithm and understand why it works
  - Simple, fast algorithm applicable in very many tasks
  - Important part of your toolbox
- Grasp the concept of scale-invariant features
  - Example: SIFT algorithm (location and description)
- Geometry and image transforms is out of scope for this course
  - But part of INF 2310 – so you know all this!

Bonus slides

Active stereo with structured light

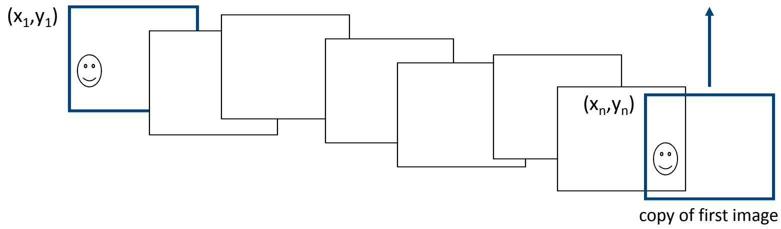


Li Zhang's one-shot stereo



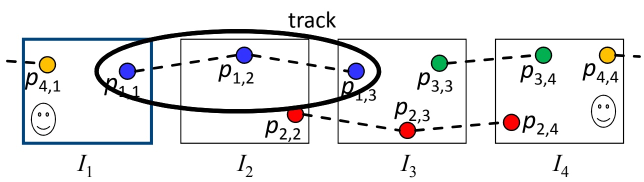
- Project "structured" light patterns onto the object
  - simplifies the correspondence problem

Related topic: Drift



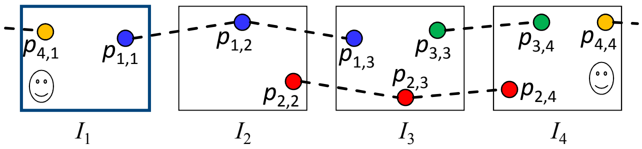
- add another copy of first image at the end
- this gives a constraint: $y_n = y_1$
- there are a bunch of ways to solve this problem
  - add displacement of $(y_1 - y_n)/(n - 1)$ to each image after the first
  - compute a global warp: $y' = y + ax$
  - run a big optimization problem, incorporating this constraint
    - best solution, but more complicated
    - known as "bundle adjustment"

Global optimization



- Minimize a global energy function:
  - What are the variables?
    - The translation $t_j = (x_j, y_j)$ for each image I_j
  - What is the objective function?
    - We have a set of matched features $p_{i,j} = (u_{i,j}, v_{i,j})$
      - We'll call these *tracks*
    - For each point match $(p_{i,j}, p_{i,j+1})$: $p_{i,j+1} - p_{i,j} = t_{j+1} - t_j$

Global optimization



$$p_{1,2} - p_{1,1} = t_2 - t_1$$
$$p_{1,3} - p_{1,2} = t_3 - t_2$$
$$p_{2,3} - p_{2,2} = t_3 - t_2$$
$$\dots$$
$$v_{4,1} - v_{4,4} = y_1 - y_4$$

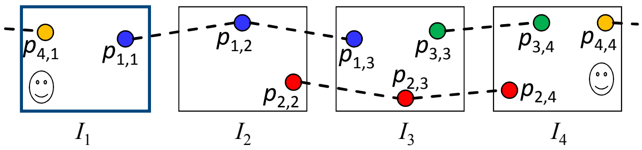
minimize

$$\sum_{i=1}^{m} \sum_{j=1}^{n-1} w_{ij} \cdot \left\| (p_{i,j+1} - p_{i,j}) - (t_{j+1} - t_j) \right\|^2 + \sum_{i=1}^{m} w_{in} \cdot \left\| (v_{i,1} - v_{i,n}) - (y_1 - y_n) \right\|^2$$

w_{ij} = 1 if track i is visible in images j and j+1
0 otherwise

SINTEF Technology for a better society

Global optimization



$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ & & & \dots & & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} = \begin{bmatrix} u_{1,2} - u_{1,1} \\ v_{1,2} - v_{1,1} \\ \vdots \\ v_{4,1} - v_{4,4} \end{bmatrix}$$

A 2m x 2n　　**x** 2n x 1　　**b** 2m x 1

SINTEF Technology for a better society

Global optimization

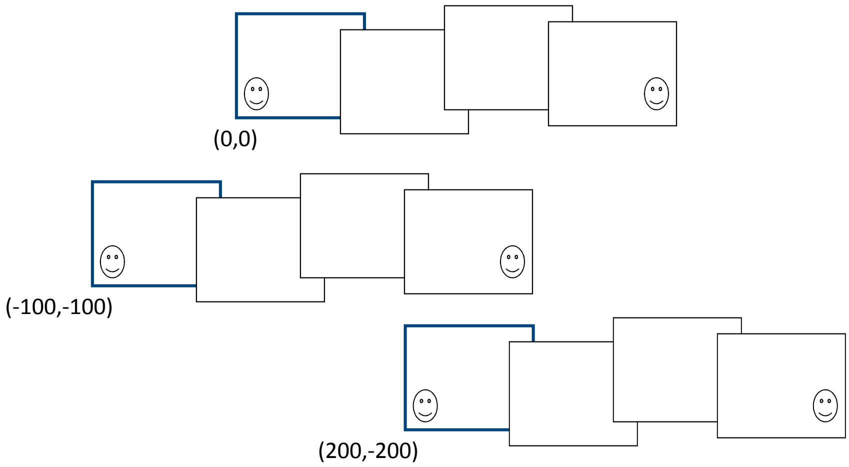
$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ & & & \cdots & & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} = \begin{bmatrix} u_{1,2} - u_{1,1} \\ v_{1,2} - v_{1,1} \\ \vdots \\ v_{4,1} - v_{4,4} \end{bmatrix}$$

A 2m x 2n **x** 2n x 1 **b** 2m x 1

Defines a least squares problem: minimize $\|\mathbf{Ax} - \mathbf{b}\|$

- Solution: $\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$
- Problem: there is no unique solution for $\hat{\mathbf{x}}$! (det($\mathbf{A}^T\mathbf{A}$) = 0)
- We can add a global offset to a solution $\hat{\mathbf{x}}$ and get the same error

Ambiguity in global location



- Each of these solutions has the same error
- Called the *gauge ambiguity*
- Solution: fix the position of one image (e.g., make the origin of the 1st image (0,0))