



---

# TSBB15

# Computer Vision

## Lecture 8

## Local invariant features



# Image Matching

---

- KLT tracking and block matching are useful when matching between consecutive frames in a **video** sequence.



# Image Matching

---

- KLT tracking and block matching are useful when matching between consecutive frames in a **video** sequence.
- Images are from **the same camera**
- small changes in **scale**, **rotation** and **illumination**



# Image Matching

---

- KLT tracking and block matching are useful when matching between consecutive frames in a video sequence.
- Images are from **the same camera**
- small changes in **scale**, **rotation** and **illumination**
- Local invariant features work when these conditions are violated.



# Wide-baseline stereo

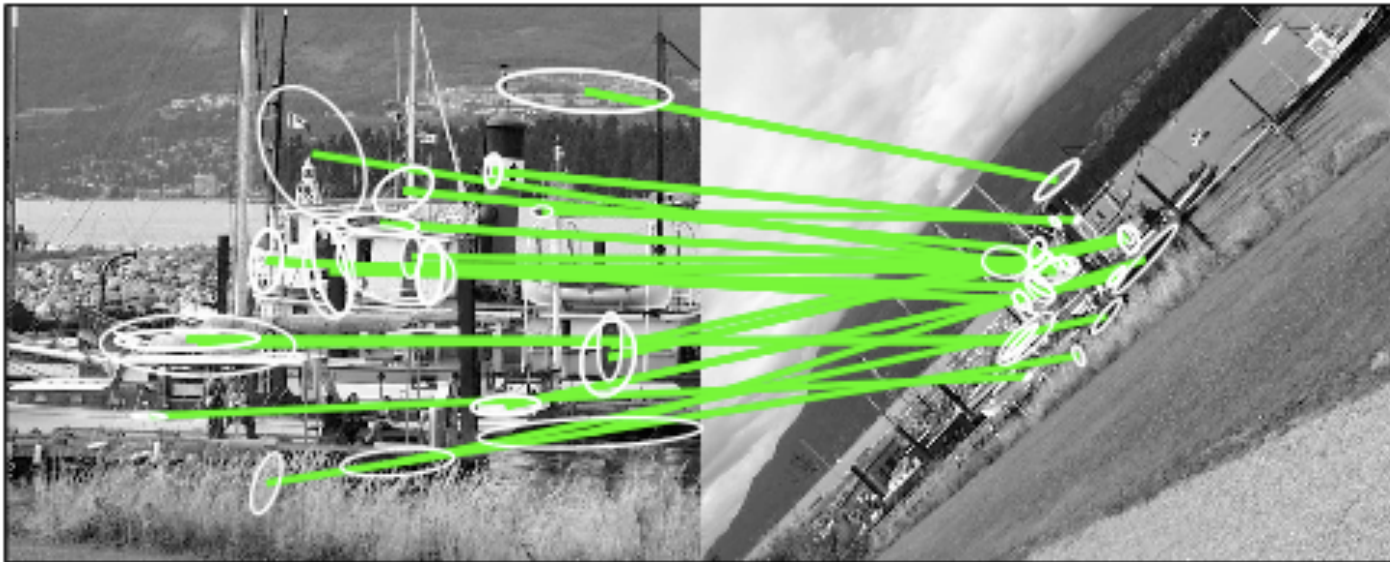
- Problem 1: wide-baseline stereo
  - Matching images of the same scene, captured at different positions.





# Wide-baseline stereo

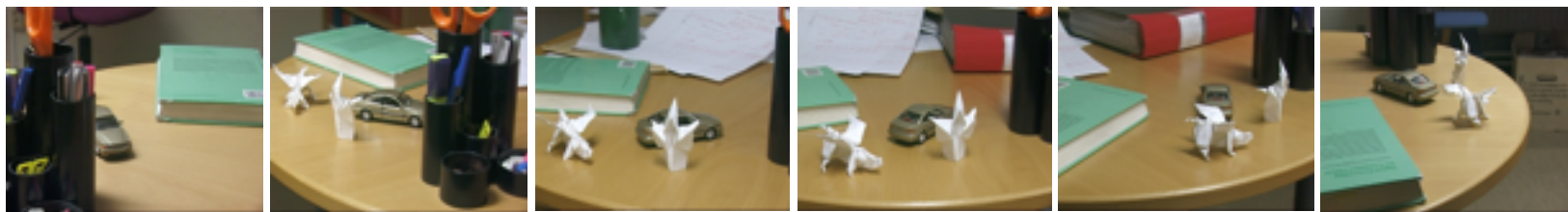
- Problem 1: wide-baseline stereo
  - Matching images of the same scene, captured at different positions.





# Object instance recognition and pose estimation

- Problem 2: object instance recognition
  - Recognition of 3D objects under partial occlusion.
  - training set
  - test set
  - 6dof pose





# Object recognition

---

- Example: Eddie the embodied



- See webpage for details  
<http://users.isy.liu.se/cvl/perfo/>





# Local invariant features

---

- We have already seen how to match across scale and translation. **How?**



# Local invariant features

---

- We have already seen how to match across scale and translation. **How?**
- Another option is to use **interest points** e.g. Harris points [Z. Zhang et al. 95].
  - A. Detect interest points
  - B. Cut out image patches around each point
  - C. Matches can now be found by comparing patches+epipolar geometry constraints.



# Local invariant features

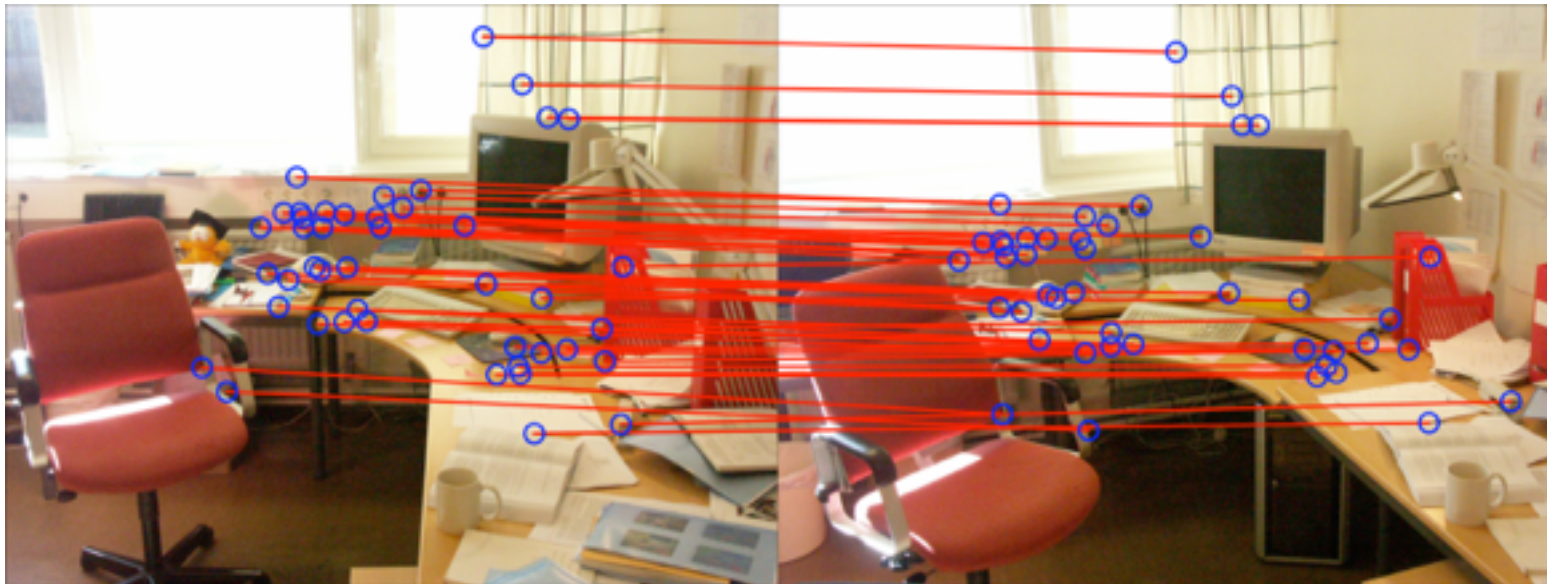
- Correspondences from block matching at Harris points.





# Local invariant features

- After applying the Epipolar constraint  
(You will test this in lab 3).





# Epipolar constraint (recap)

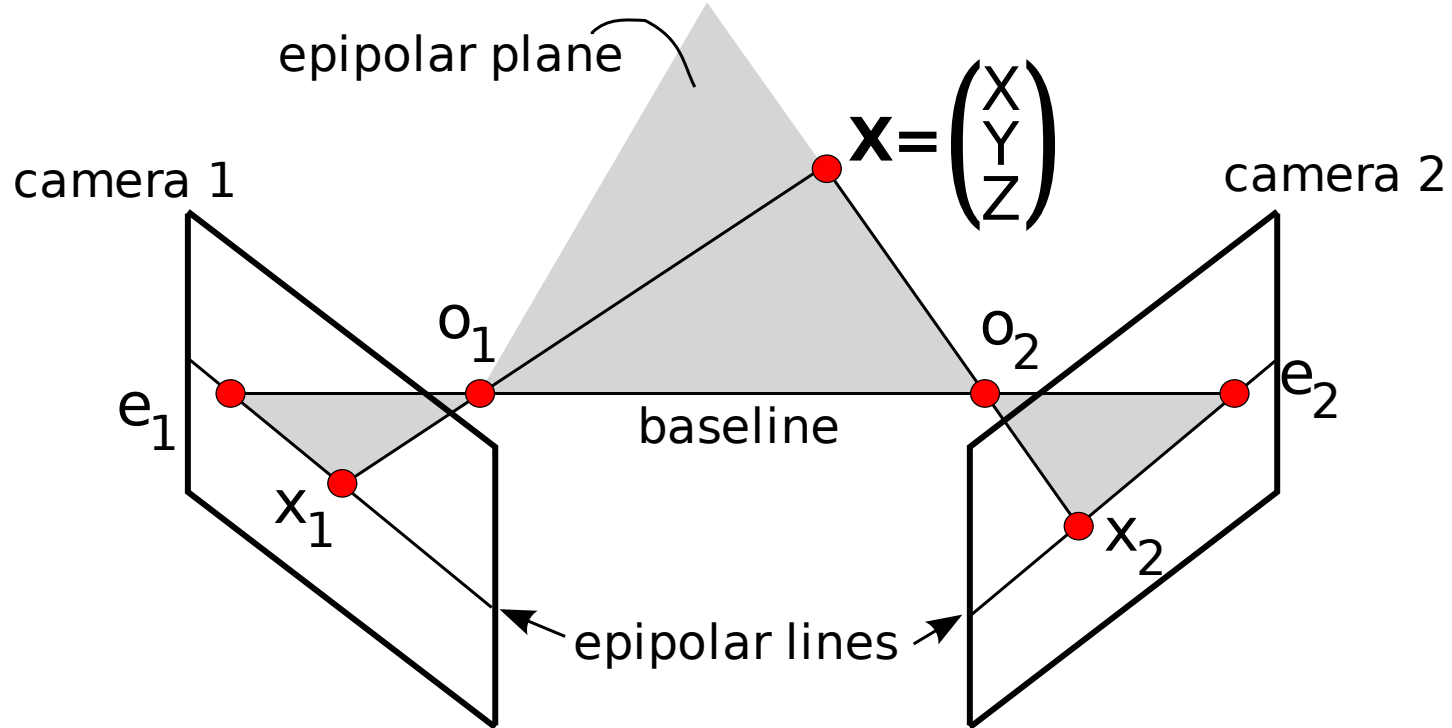
---

- The epipolar constraint:  $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$



# Epipolar constraint (recap)

- The epipolar constraint:  $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$





# Epipolar constraint (recap)

---

- The epipolar constraint:  $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$
- $\mathbf{x}_1$  and  $\mathbf{x}_2$  are projections of the same 3D point in two views.
- Scene is static, i.e. no motion has taken place (except the change of camera position).
- $\mathbf{F}$  can be estimated from 7 or more correspondences. E.g. 8-pt algorithm.



# Epipolar constraint (recap)

---

- The epipolar constraint:  $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$
- See the compendium, *Introduction to Estimation, Representation and Geometry*, Klas Nordberg





# Local invariant features

---

- Zhang's **interest point** method. (repeat)
  - A. Detect interest points
  - B. Cut out image patches around each point
  - C. Matches can now be found by comparing patches+epipolar geometry constraints.



# Local invariant features

---

- Zhang's method is invariant to translation (and partially to scale).

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = s \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \mathbf{t}$$

- 2 degrees-of-freedom (DOF) of invariance (transl. only) (3 if scale is also counted)



# Local invariant features

---

- Zhang's method is invariant to translation (and partially to scale).

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = s \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \mathbf{t}$$

- 2 degrees-of-freedom (DOF) of invariance (transl. only) (3 if scale is also counted)
- What about image rotations?
- What about view changes?



# Local invariant features

---

- In general, the *local invariant feature approach* can be described as three steps:
  - **Detection**: Use a *detector* to find a local, canonical frame (coordinate system)



# Local invariant features

---

- In general, the *local invariant feature approach* can be described as three steps:
  - **Detection**: Use a *detector* to find a local, canonical frame (coordinate system)
  - **Description**: Compute a *descriptor*, by sampling the image in the canonical frame



# Local invariant features

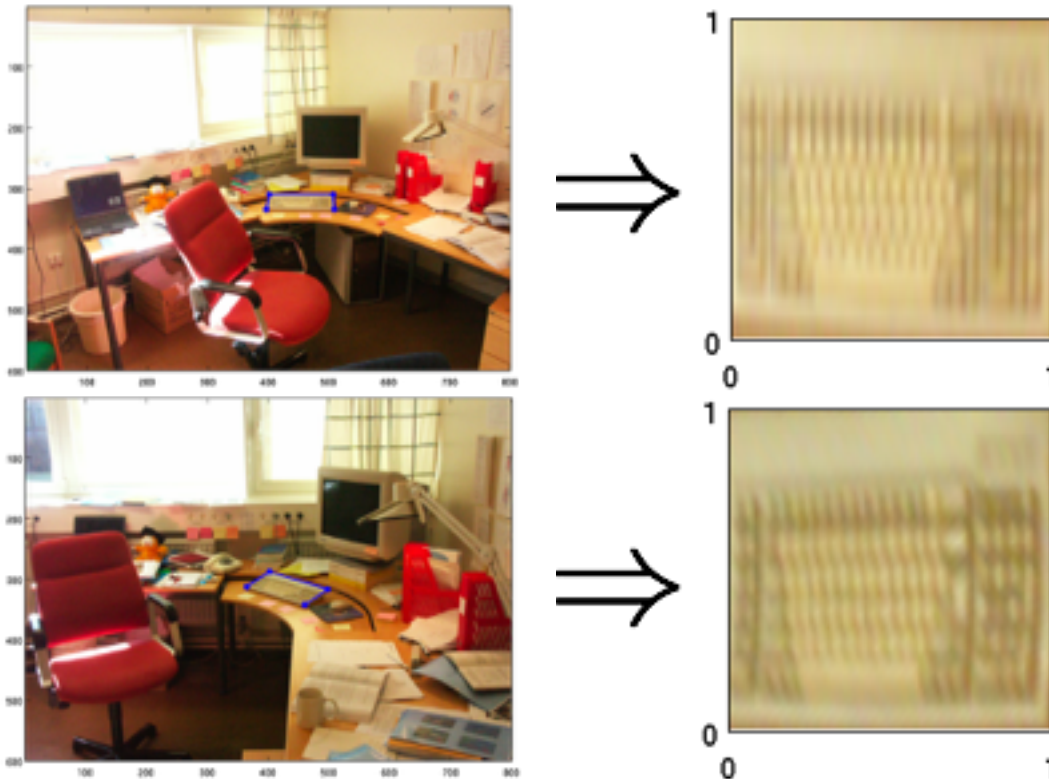
---

- In general, the *local invariant feature approach* can be described as three steps:
  - **Detection**: Use a *detector* to find a local, canonical frame (coordinate system)
  - **Description**: Compute a *descriptor*, by sampling the image in the canonical frame
  - **Matching**: Find correspondences, by comparing descriptors from two images



# Canonical frame example

- Resampling to canonical frame





# Local invariant features

---

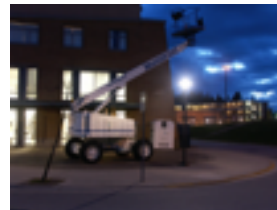
- ***Geometric invariances***

Robustness to view changes



- ***Photometric invariances***

Robustness to illumination changes







# Local invariant features

---

- ***Geometric invariances*** can be obtained by choosing a frame that is *equivariant* to rotations, scalings, and image skews
- ***Photometric invariances*** can be obtained by computing the descriptor in a more advanced way than direct sampling.



# Geometric Invariance

---

- The geometric invariances we use make *a locally planar assumption*.
- They can thus be described using *homographies*.



# Geometric Invariance

- Recap: A **Homography** is a transformation between points  $\mathbf{x}$  on one plane, and points  $\mathbf{y}$  on another.

$$\lambda \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$



# Geometric Invariance

- Recap: A **Homography** is a transformation between points  $\mathbf{x}$  on one plane, and points  $\mathbf{y}$  on another.

$$\lambda \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

- **Degrees of freedom**: number of unique elements in  $\mathbf{H}$ .
  - at most 8dof (for plane projective case), as  $\mathbf{H}$  and  $k\mathbf{H}$ ,  $k \in \mathbb{R} \setminus 0$  gives the same output



# Geometric Invariance

- A hierarchy of transformations:
  - scale+translation (3dof)
  - similarity (4dof)  
(scale+translation+rotation)
  - affine (6dof)  
(similarity+skew)
  - plane projective (8dof)  
(affine+forshortening)

$$\begin{bmatrix} s & 0 & t_1 \\ 0 & s & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} s_1 & s_2 & t_1 \\ -s_2 & s_1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

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



# Geometric Invariance

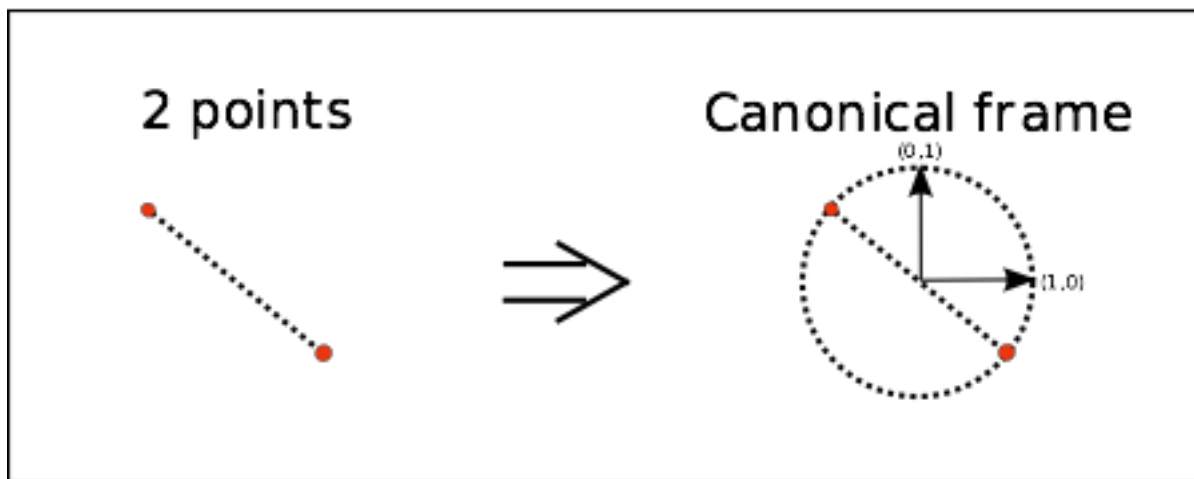
---

- We can find the canonical frame by using more than one point  
[Brown&Lowe 02] aka. *interest-point groups*
- We will now give some examples...



# Geometric Invariance

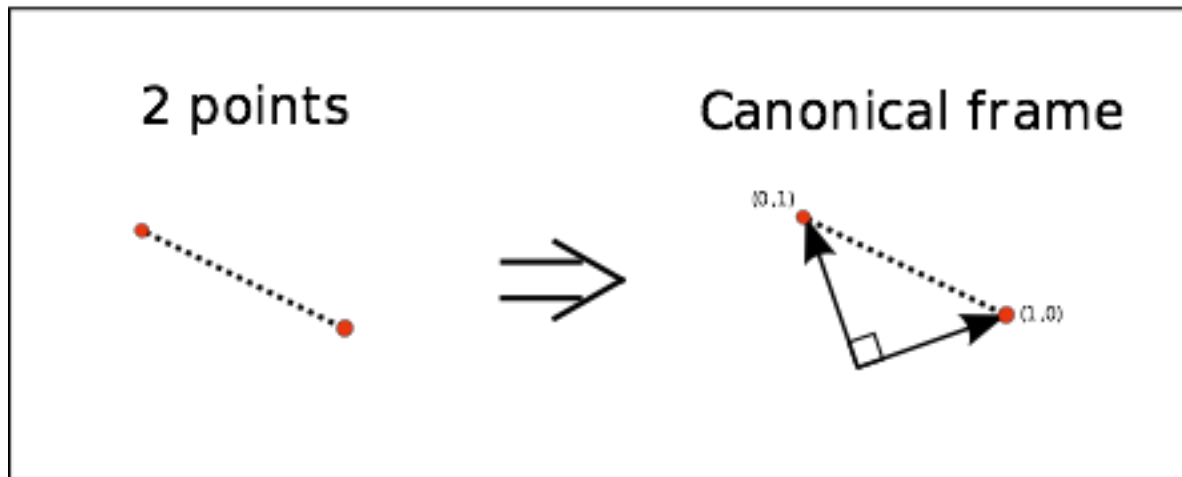
- Scale+translation: Useful if we know that there is no rotation. E.g. for a camera mounted in a car, looking at upright pedestrians.





# Geometric Invariance

- Similarity: Full invariance in image plane, none outside image plane.  
Useful e.g. for pose estimation.

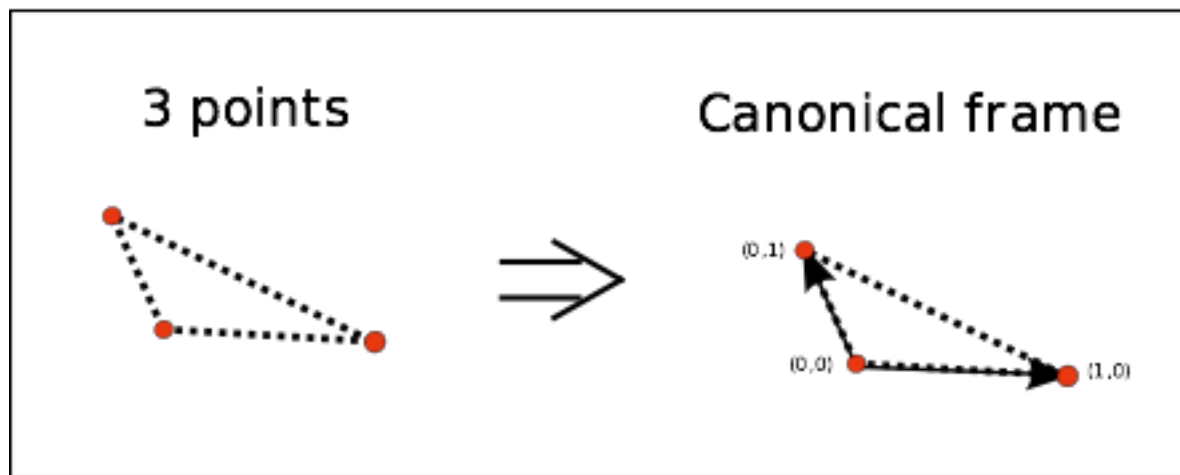






# Geometric Invariance

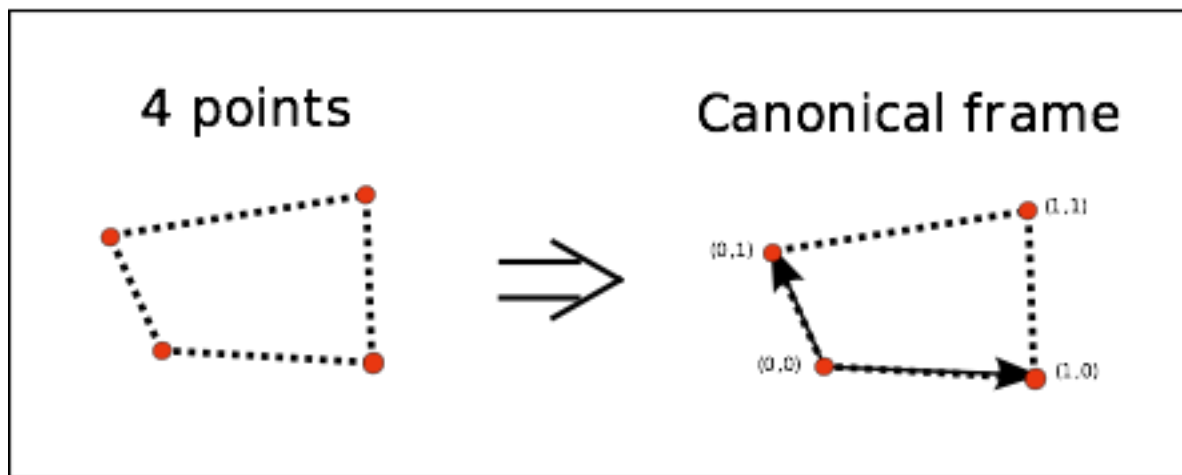
- Affine: Deals with most common projective distortions. Good if patch size is small relative to distance to patch.





# Geometric Invariance

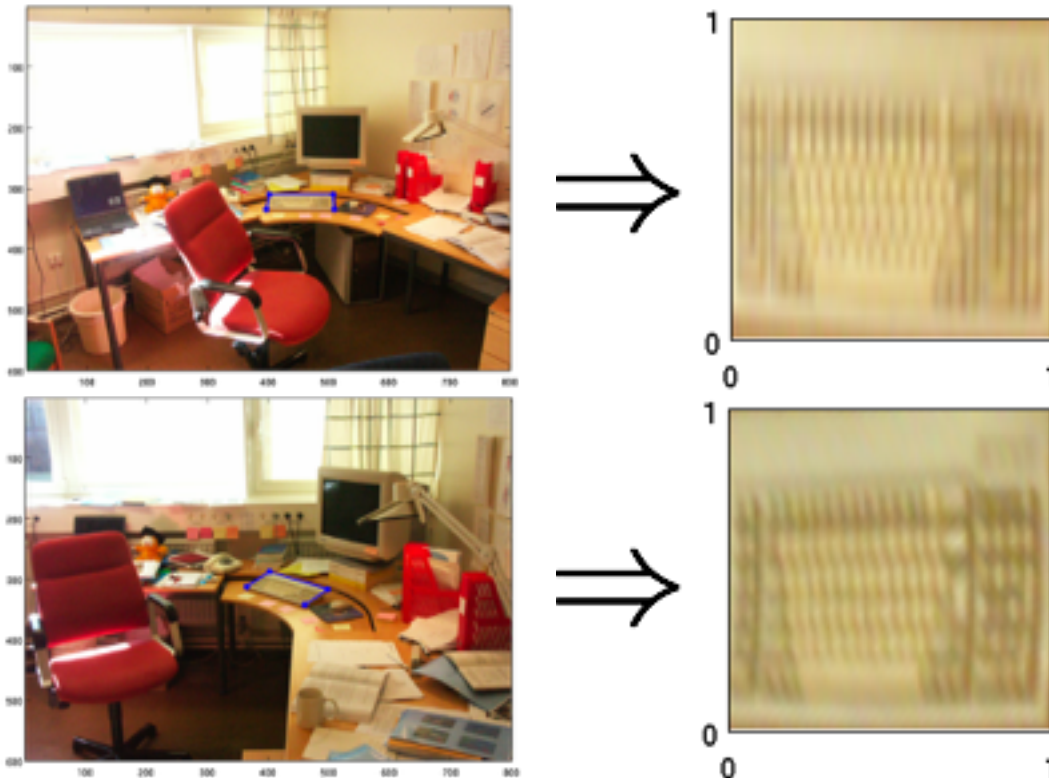
- Plane projective: Full modelling of a plane in 3D. Requires more image measurements, but is better for extreme view angles.





# Geometric Invariance

- Resampling to canonical frame





# Geometric Invariance

- Problems with interest-point groups
  - Sensitive to missing points:  
If  $e = P(\text{point-detected} | \text{present})$  then  
 $P(\text{frame-is-detected} | \text{present}) = e^N$   
where  $N$  is number of points in frame.
  - Combinatorics: if  $K$  points in image, we have  
 $\binom{N}{K}$  possible canonical frames.
  - We will introduce other ways to find the frame soon.



# Photometric Invariance

---

- Image intensity is approx. linear in radiance (at least before gamma correction)
- E.g. adding a second, identical light source will double the sensor activation,  $a(\mathbf{x})$ .

$$a(\mathbf{x}) = \int s(\lambda)r(\lambda, \mathbf{x})e(\lambda)d\lambda$$

- s-sensor absorption spectrum, r-reflectance spectrum of object, e-emission spectrum of light source (attenuated by the atmosphere)



# Photometric Invariance

- If illumination changes, image matching fails:

$$\begin{aligned} I(\mathbf{x}) &= I_0(\mathbf{x})k_1 \\ J(\mathbf{x}) &= I_0(\mathbf{x})k_2 \end{aligned} \Rightarrow \sum_{\mathbf{x}} (I(\mathbf{x}) - J(\mathbf{x}))^2 = \text{non-zero}$$

- We want a function that is invariant to scalings:

$$\sum_{\mathbf{x}} (f(I(\mathbf{x})) - f(J(\mathbf{x})))^2 = \text{small-number}$$

- How should we choose  $f()$ ?



# Photometric Invariance

---

- For cameras with non non-linear radiometric response (and e.g. gamma correction), or if two different cameras are used we may use the **affine model**:

$$I(\mathbf{x}) = I_0(\mathbf{x})k_1 + k_2$$

- How should we choose  $f$ ? we want:

$$\sum_{\mathbf{x}} (f(I(\mathbf{x})) - f(J(\mathbf{x})))^2 = \text{small-number}$$



# Photometric Invariance

---

- Mean subtraction, derivatives, and other DC free linear filters remove a constant offset in intensity
- Normalising a patch by e.g. the L<sub>2</sub>-norm or the standard deviation, removes scalings of the intensity.
- Affine invariance by combining both:

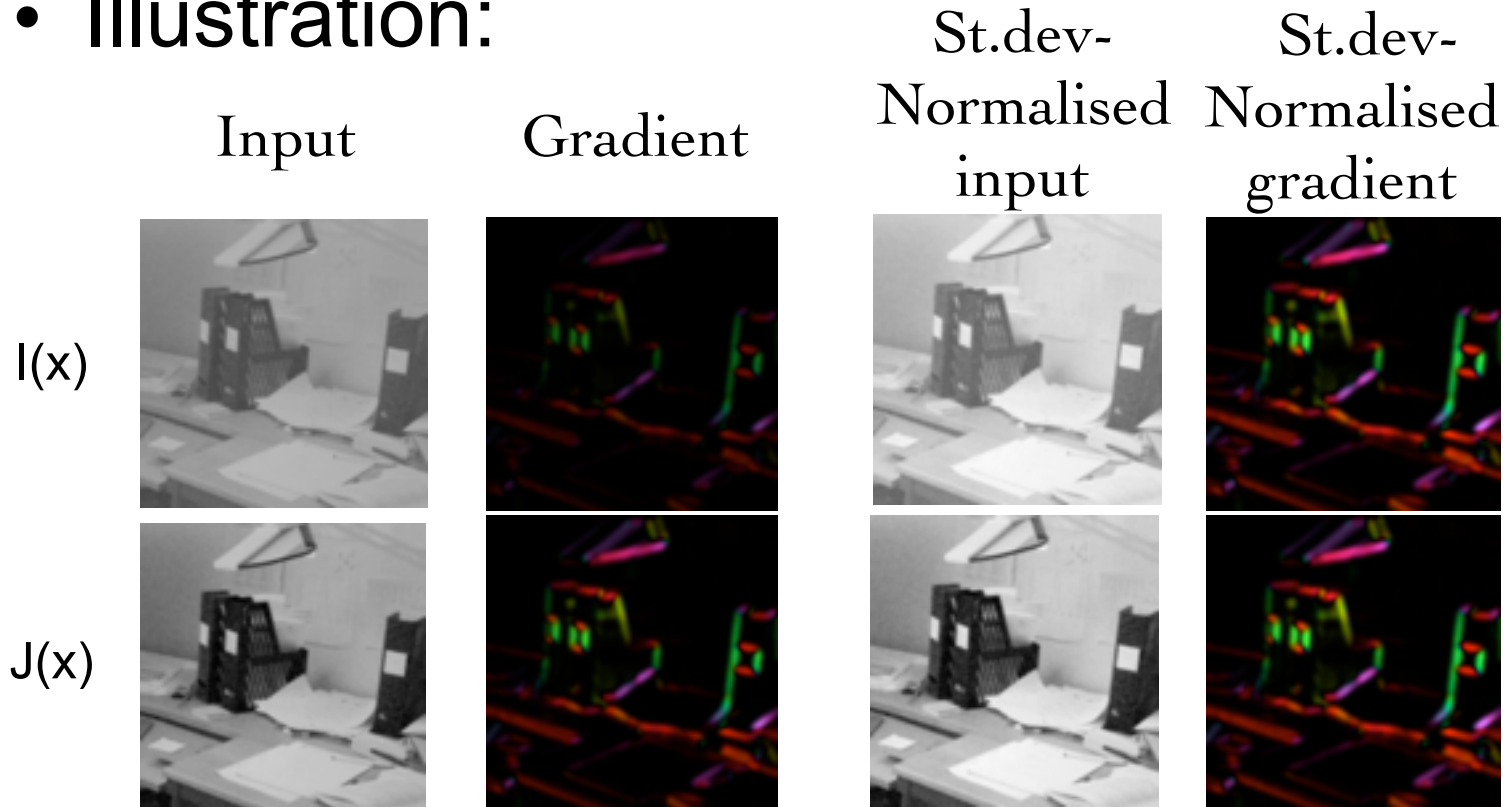
$$\hat{I}(\mathbf{x}) = (I(\mathbf{x}) - \mu_I) / \sigma_I$$





# Photometric Invariance

- Illustration:





# Local Invariant Features

---

- There are many examples of features that fit the descriptor+detector paradigm.
- The two most widely used are:
  - **SIFT** Scale Invariant Feature Transform (Lowe 99)
  - **MSER** Maximally Stable Extremal Regions (Matas et al. 02)
- We will look at these two in more detail.



# SIFT

- Scale Invariant Feature Transform [Lowe'99]. In brief:
  - The *SIFT detector* finds points using Difference-of-Gaussians in a pyramid  
Gives: position  $x,y$  and scale  $s$
  - Rotation is found from a gradient histogram
  - This gives a frame for the *SIFT descriptor*, which is computed from gradient orientation histograms.



# SIFT detector

---

- Scale Space (recap.)
  - The image is extended with an extra dimension for scale/blur:
$$f(x, y, s) = (f_0 * g(s))(x, y)$$
  - The blurring kernel  $g(s)$  is typically a Gaussian:

$$g(\mathbf{x}, s) = \frac{1}{2\pi s} e^{-\mathbf{x}^T \mathbf{x} / 2s^2}$$



# SIFT detector

---

- Scale Selection [Lindeberg'93]
  - Find a characteristic point (e.g. max) on a function of position and scale:

$$(\hat{\mathbf{x}}, \hat{s}) = \arg \max h(f(\mathbf{x}, s))$$

- Example: Maximum of normalised Laplacian:

$$h(f(\mathbf{x}, s)) = s^2 (f * \nabla^2 g(s))(\mathbf{x})$$



# SIFT detector

$$(\hat{\mathbf{x}}, \hat{s}) = \arg \max h(f(\mathbf{x}, s))$$

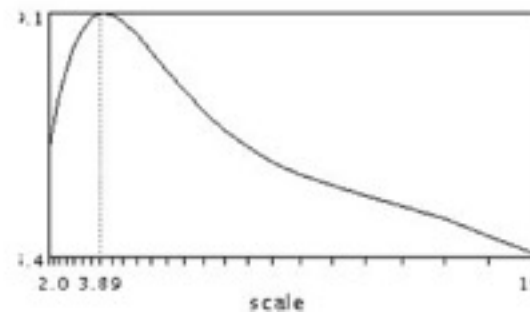
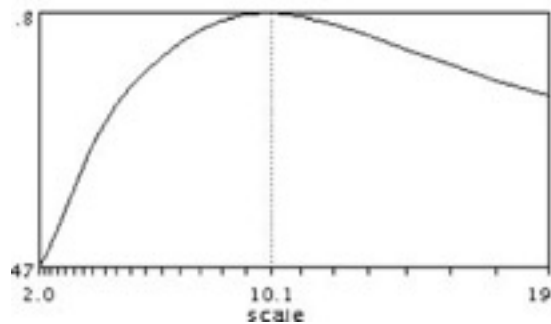


Illustration by (Mikolajczyk et al. 2005)



# SIFT detector

---

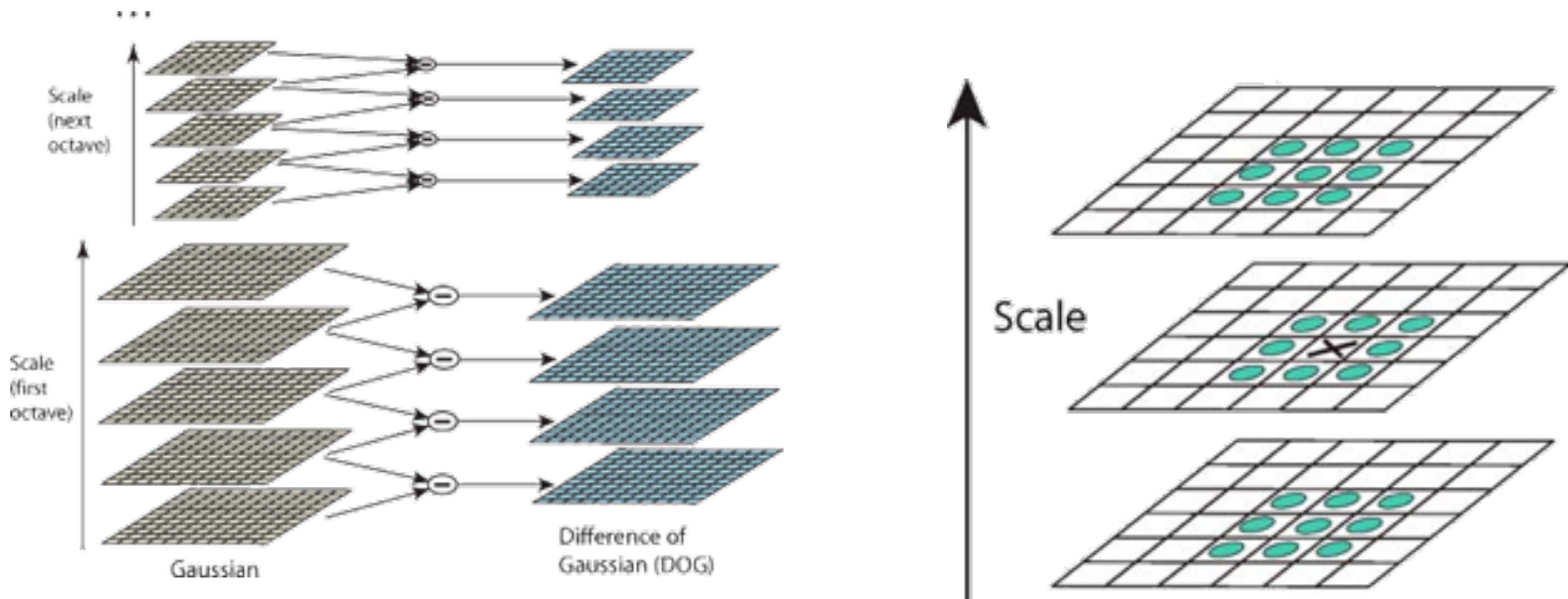
- In SIFT, scale selection is done using difference-of-Gaussians:

$$h(f(\mathbf{x}, \sigma)) = (f * (g(\sigma) - g(k\sigma)))(\mathbf{x})$$

- Efficient implementation using pyramids [Lowe'99]
- Sampling in scale space with  $\Delta\sigma = 1/\sqrt{2}$



# SIFT detector



$$g(\sigma_1) * g(\sigma_2) = g\left(\sqrt{\sigma_1^2 + \sigma_2^2}\right)$$

Non-max suppression in (x,y,s)





# SIFT detector

- Finally we find one or more reference directions using a gradient orientation histogram  $h$  at the found location in scale space.



$$h_k = \sum_{\text{patch}} |\nabla \mathbf{f}(\mathbf{x})| B_k(\tan^{-1} \nabla \mathbf{f}(\mathbf{x}))$$



# SIFT descriptor

---

- The SIFT detector gives us a similarity frame. **What is this?**
  - We now want to convert the image patch at the frame to a 128-byte *descriptor vector*.
  - The purpose of this is to add photometric invariance, and some extra translation and scale robustness.



# SIFT descriptor

- Compute x- and y-gradients through convolution:

$$\nabla \mathbf{f}(\mathbf{x}) = \begin{bmatrix} (d_x * f)(\mathbf{x}) \\ (d_y * f)(\mathbf{x}) \end{bmatrix}$$

- Rotate gradient map to direction from orient-hist:

$$\nabla \hat{\mathbf{f}}(\mathbf{x}) = \mathbf{R} \nabla \mathbf{f}(\mathbf{R}^T \mathbf{x})$$

- Compute gradient orientation histograms in 4x4 spatial regions:

$$h_{kl} = \sum_{\mathbf{x} \in \text{patch}_l} |\nabla \hat{\mathbf{f}}(\mathbf{x})| w(\mathbf{x} + \mathbf{d}_l) B_k(\tan^{-1} \nabla \hat{\mathbf{f}}(\mathbf{x}))$$



# SIFT descriptor

- Compute gradient orientation histograms in 4x4 spatial regions :

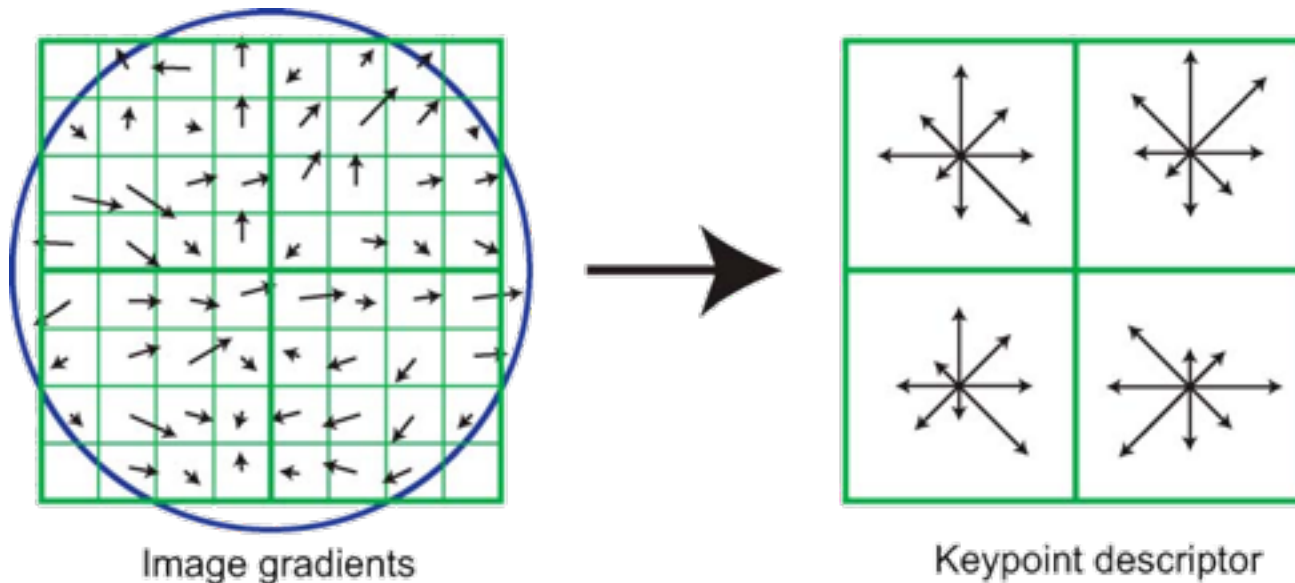
$$h_{kl} = \sum_{\mathbf{x} \in \text{patch}_l} |\nabla \hat{\mathbf{f}}(\mathbf{x})| w(\mathbf{x} + \mathbf{d}_l) B_k(\tan^{-1} \nabla \hat{\mathbf{f}}(\mathbf{x}))$$

- $B_k(x)$  linear interpolation kernel  
Quadratic is better (Jonsson&Felsberg)
- Subwindows  $l \in [1 \dots 16]$  directions  $k \in [1 \dots 8]$
- Spatial weight  $w(\mathbf{x} + \mathbf{d}_l)$  (Gaussian decay)



# SIFT descriptor

- Implementation with source code in both VLFeat and OpenCV.



Note that  $4 \times 4$  regions are actually used, with 8 orientations  $\rightarrow$  128 elements



# SIFT descriptor

- Affine illumination invariance by using gradients and normalising descriptor  $\hat{\mathbf{h}} = \mathbf{h}/\|\mathbf{h}\|$
- Some robustness by truncating and normalising again  $\hat{\mathbf{h}} = \min(t, \hat{\mathbf{h}})/\|\hat{\mathbf{h}}\|$
- The spatial histogramming gives robustness to scale/rotation/translation errors.
- SIFT is used commercially in many places. (The Sony AIBO was an early example.)



# MSER

---

- Maximally Stable Extremal Regions  
[Matas et al.'02]
- Consider the set of all possible thresholdings of an image...

[Movie clip]



# MSER







# MSER

- Connected regions form segments.
  - Cf. Watershed algorithm (similar idea but different output)
  - Look at stability of a function of segment across image evolution. e.g.
$$\text{area}(\text{component}(t))$$
  - MSERs are components that are maximally stable, i.e., have a local minimum of the rate of change: 
$$\frac{\partial \text{area}(\text{component}(t))}{\partial t}$$



# MSER

---

- compare: Maximal Stability, Scale Selection
- Stability measure: Range of stable thresholds  $t_2-t_1$  around min is called the *margin* of the region.

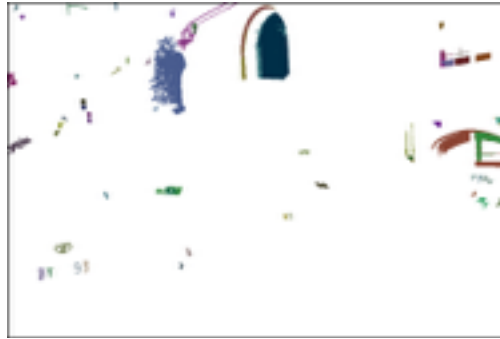


# MSER

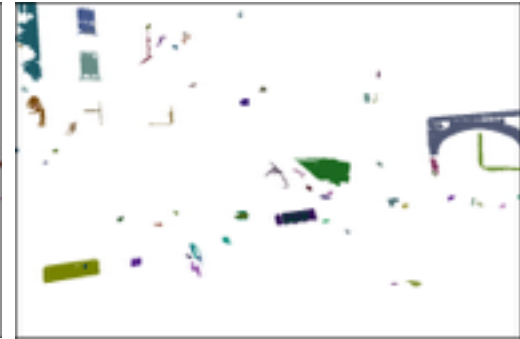
- Two possible thresholdings:  $I(\mathbf{x}) < t$  ,  $I(\mathbf{x}) > t$



Input image



64 MSER- (total 272)



64 MSER+ (total 294)

- Very fast (using union/find+path compression).
- MSER type (+/-) is useful for matching **How?**



# MSER

- MSER is invariant to monotonic changes of intensity.  
i.e.  $I(x)$  and  $f(I(x))$  have the same output if
$$f(x + k) > f(x) \quad \forall k > 0$$
- Wide range of sizes obtained without a scale pyramid.  
Better still with a pyramid (Forssén&Lowe ICCV'07)
- Colour objects can be tracked by computing MSERs  
on the Mahalanobis distance to a colour distribution.  
(Donoser&Bischof CVPR'06)
- Colour regions by looking at gradients.  
Called MSCR (Forssén CVPR'07)



# MSCR





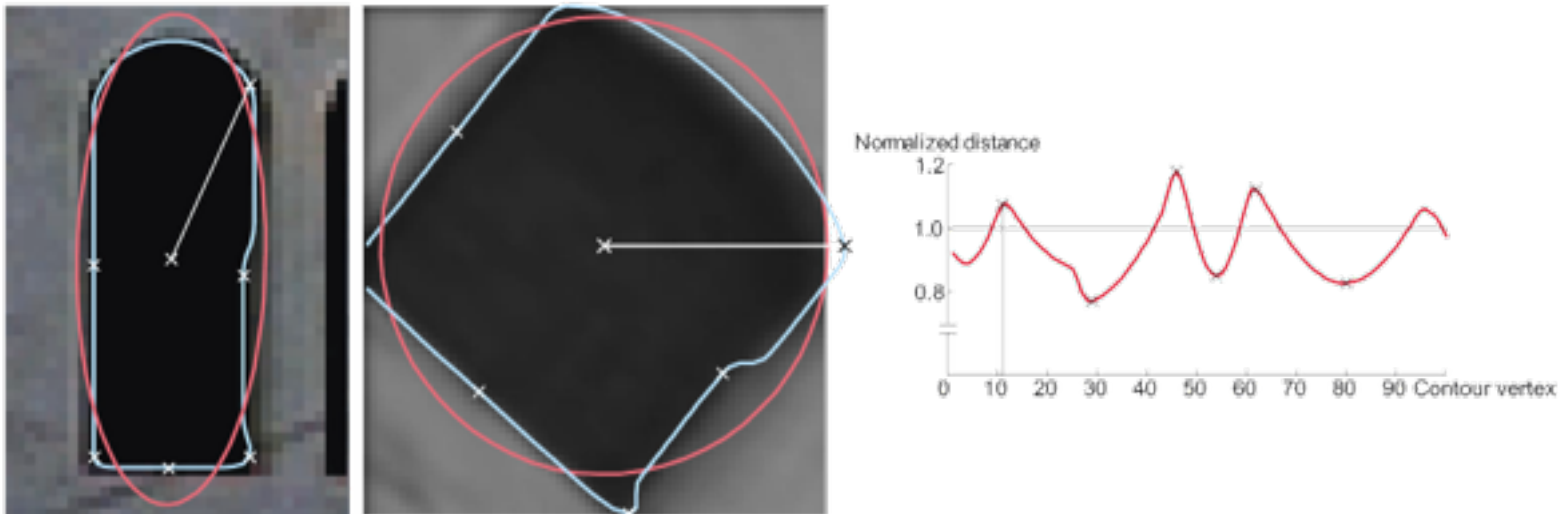
# MSCR





# MSER

- Reference directions from extremal points along ellipse-normalized contour.



Matas et al. ICPR'02



# MSER

- Approximating ellipse

- from moments of binary mask  $v : \Omega \mapsto \{0, 1\}$

$$\mu_{k,l}(v) = \sum_x \sum_y x^k y^l v(x, y)$$

$$\mathbf{m} = \frac{1}{\mu_{0,0}} \begin{bmatrix} \mu_{1,0} \\ \mu_{0,1} \end{bmatrix} \quad \mathbf{C} = \frac{1}{\mu_{0,0}} \begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix} - \mathbf{m}\mathbf{m}^T$$

$$\mathcal{R}(\mathbf{m}, \mathbf{C}) = \{\mathbf{x} : (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}) \leq 4\}$$





# MSER

- Normalisation to a circle (axis aligned)  
Compute the eigenfactorisation:

$$\mathbf{C} = \mathbf{R}\mathbf{D}\mathbf{R}^T, \quad \det \mathbf{R} > 0$$

The circle normalisation can now be performed as:

$$\mathbf{x} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{m}, \quad \text{for } \mathbf{A} = 2\mathbf{R}\mathbf{D}^{1/2}$$

$\hat{\mathbf{x}}$  - canonical coordinates

$\mathbf{x}$  - image coordinates



# MSER

---

- Ellipse+extrema of distance to centre is just one frame construction option.
- Other (affine covariant) choices:
  - Points of maximum curvature.
  - Bi-tangens.
  - See Obdrzalek&Matas BMVC'02
- Implementation w. source:  
in both VLfeat and OpenCV



# MSER descriptor

- The MSER detector originally used normalized colour patches as descriptor vectors:

$$\begin{aligned}\hat{I}_r(\mathbf{x}) &= (I_r(\mathbf{x}) - \mu_r) / \sigma_r \\ \hat{I}_g(\mathbf{x}) &= (I_g(\mathbf{x}) - \mu_g) / \sigma_g \\ \hat{I}_b(\mathbf{x}) &= (I_b(\mathbf{x}) - \mu_b) / \sigma_b\end{aligned}$$

- Nowadays other descriptors, e.g. the SIFT descriptor are used.



# Other local invariant features

---

- **SFOP**

<http://www.ipb.uni-bonn.de/sfop/>

- **BRISK**

Source Code+description

<http://www.asl.ethz.ch/people/lestefan/personal/BRISK>

- **FREAK and ORB**

In OpenCV

- **SURF and SIFT**

in OpenCV nonfree



# Binary descriptors

- To save memory and time, new descriptors use **local binary patterns**:



- Has affine illumination invariance (even monotonic invariance)



# Binary descriptors

- To save memory and time, new descriptors use **local binary patterns**:

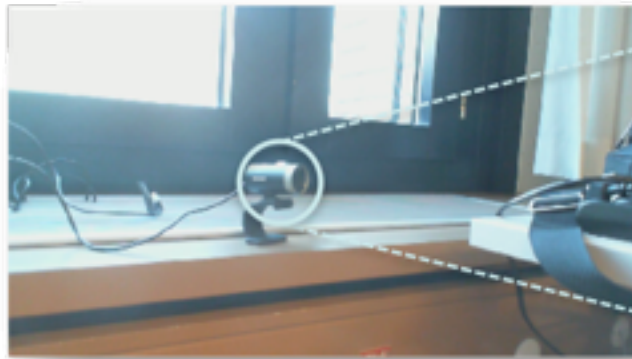
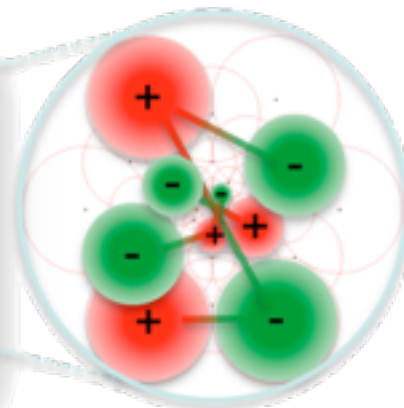


Image from Alexandre et al. CVPR 2012



10110

- E.g. **BRIEF** (ECCV'10), **BRISK** (ICCV'11), **ORB** (ICCV'11), **FREAK** (CVPR'12)



# A note on invariance

---

- Always strive to limit amount of invariance
- Use knowledge on imaging situation,
  - e.g. A car mounted camera may not need rotation invariance for pedestrians.
  - e.g. in a video with smooth illumination changes, affine illumination invariance is not necessary



# Descriptor Matching

---

- The ***Local Invariant Feature*** method:
- Detection
- Description
- **Matching**





# Descriptor Matching

---

- For a descriptor  $q$  in a query image. Which prototype in memory  $(p_1, p_2, \dots, p_N)$  is *most likely* to correspond to the same world object?



# Descriptor Matching

- For a descriptor  $\mathbf{q}$  in a query image. Which prototype in memory  $(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N)$  is *most likely* to correspond to the same world object?
- Assuming additive i.i.d. Gaussian noise on all elements:

$$p(\mathbf{q}|\mathbf{p}_k) \propto \prod_{l=1}^D e^{-.5(p_{kl} - q_l)^2 / \sigma^2}$$

$$\max(J) \Leftrightarrow \min(-\log(J))$$

$$-\log(p(\mathbf{q}|\mathbf{p}_k)) \propto \sum_{l=1}^D (p_{kl} - q_l)^2$$



# Descriptor Matching

---

- So, the match with smallest distance is most likely correct, assuming i.i.d. Gaussian noise.
- What about the scalar product for normalised vectors/NCC?



# Descriptor Matching

---

- So, the match with smallest distance is most likely correct, assuming i.i.d. Gaussian noise.
- What about the scalar product for normalised vectors/NCC?

$$\|\mathbf{p} - \mathbf{q}\|^2 = \mathbf{p}^T \mathbf{p} + \mathbf{q}^T \mathbf{q} - 2\mathbf{p}^T \mathbf{q} = 2(1 - \mathbf{p}^T \mathbf{q})$$

- But are all values identically distributed?
- ...are they all independent?



# Descriptor Matching

---

- For binary descriptors (e.g. **BRIEF**) the Hamming distance is used:

$$s = \text{bitcnt}(\text{XOR}(P, Q))$$

- Also makes i.i.d. assumption.



# Dense invariant features

---

- (semi-)dense flow for wide baseline problems can be obtained by matching invariant features
- located **at every pixel** and
- also at several scales
- e.g. **SIFTflow**, **DSIFT**, **PHOW**, **DAISY**
- Much more expensive to compute. GPGPU etc. is helpful here.



# Summary

---

- Use local invariant features:  
**when KLT fails**
- But use no more invariance than needed
- **Photometric** and **Geometric** invariance
- Recognition in three steps: **Detection**,  
**Description** and **Matching**



# Upcoming course events

---

- Lab 2 tomorrow (11/2 13-17).  
Opportunity to demonstrate working lab1. at the beginning, 13.00
- Next Lecture (16/2 10-12)  
Biological vision. Voluntary.