Intelligent Multimedia Systems

Master AI, 2012, Lecture 4

Lecturers: Theo Gevers

Lab: Intelligent Systems Lab Amsterdam (ISLA)

Email: th.gevers@uva.nl

http://staff.science.uva.nl/~gevers

isis

Intelligent Sensory Information Systems

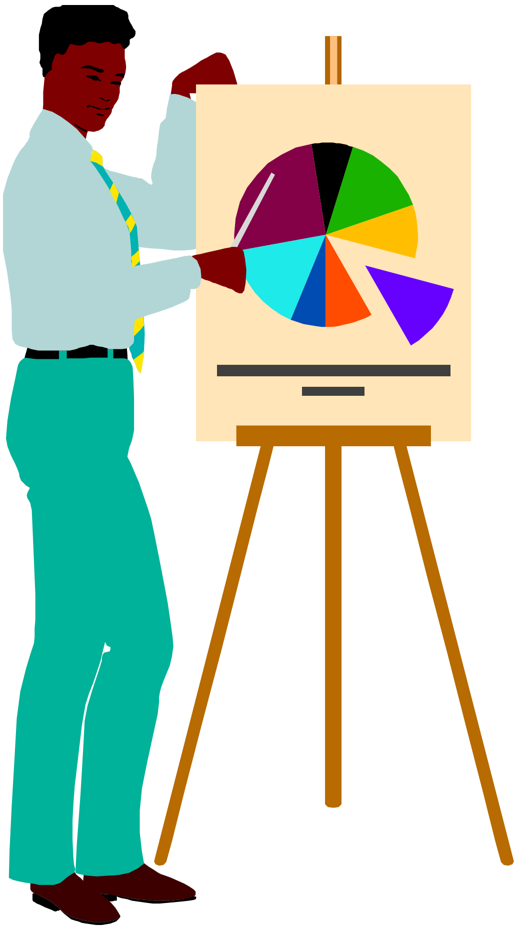
Lectures

- 29-10-2012, Monday, 15:00-17:00, Science Park A1.04 - Introduction
- 05-11-2011, Monday, 15:00-17:00, Science Park A1.04 - Image and Video Formation
- 12-11-2011, Monday, 15:00-17:00, Science Park A1.04 - Color Invariance and Image Processing
- 19-11-2011, Monday, 15:00-17:00, Science Park A1.04 - Feature Extraction and Tracking
- 26-11-2011, Monday, 15:00-17:00, Science Park A1.04 - Learning and Object Recognition
- 03-12-2011, Monday, 15:00-17:00, Science Park A1.04 - Visual Attention and Affective Computing
- 10-12-2011, Monday, 15:00-17:00, Science Park A1.04 - Human Behavior Analysis
- 18-12-2011, Tuesday, 15:00-18:00, Science Park, C1.10 - Examination

Today's class

Feature Extraction

Tracking

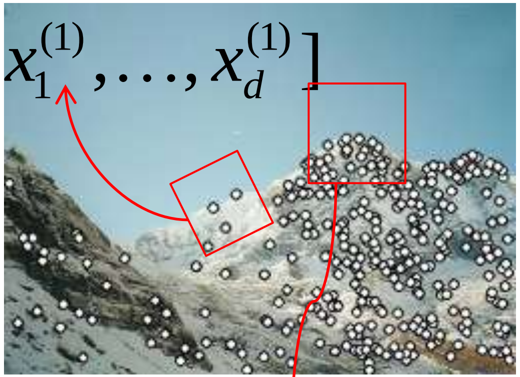


Local Features: Main Components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

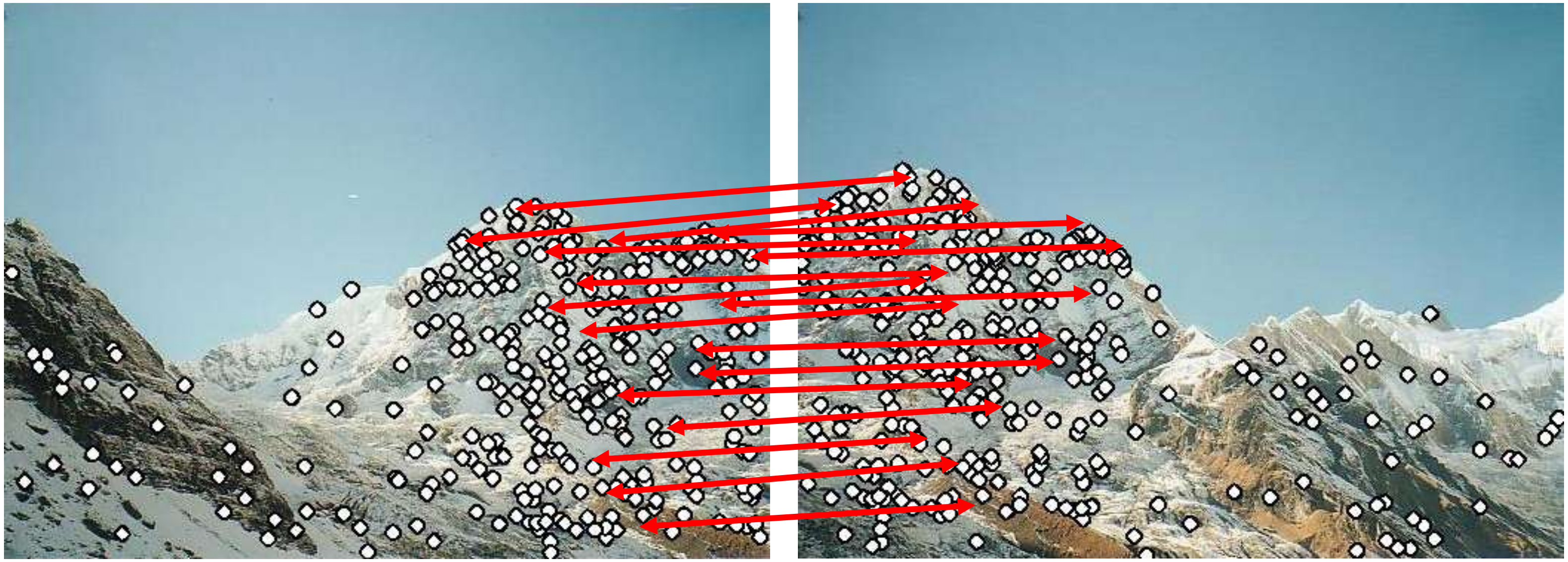
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two images

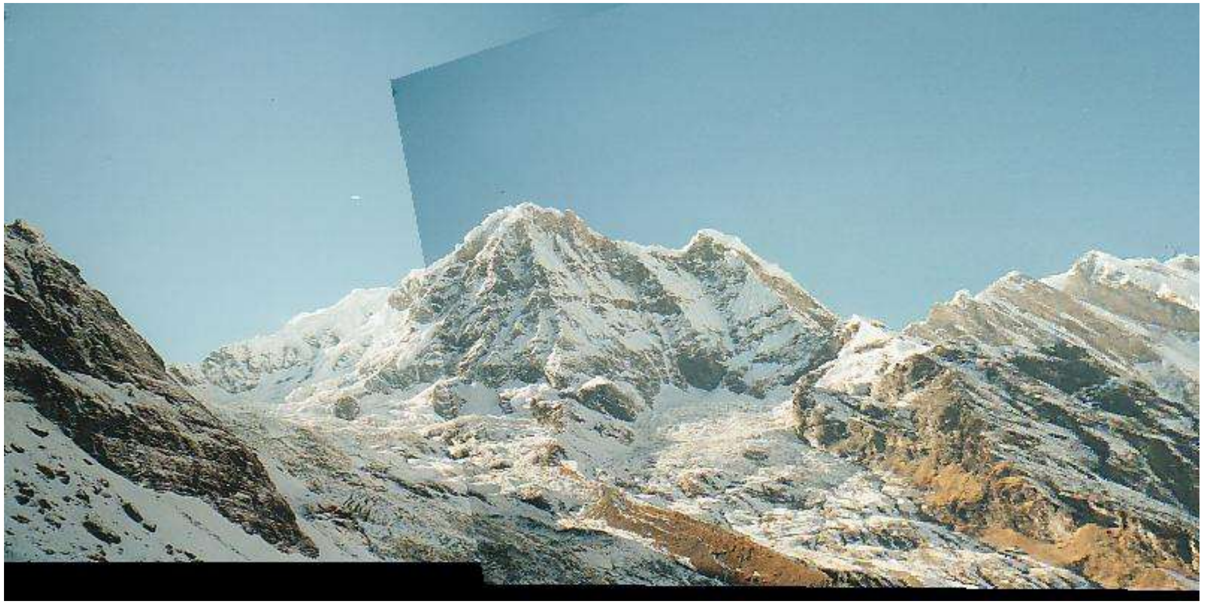
Feature-based Alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

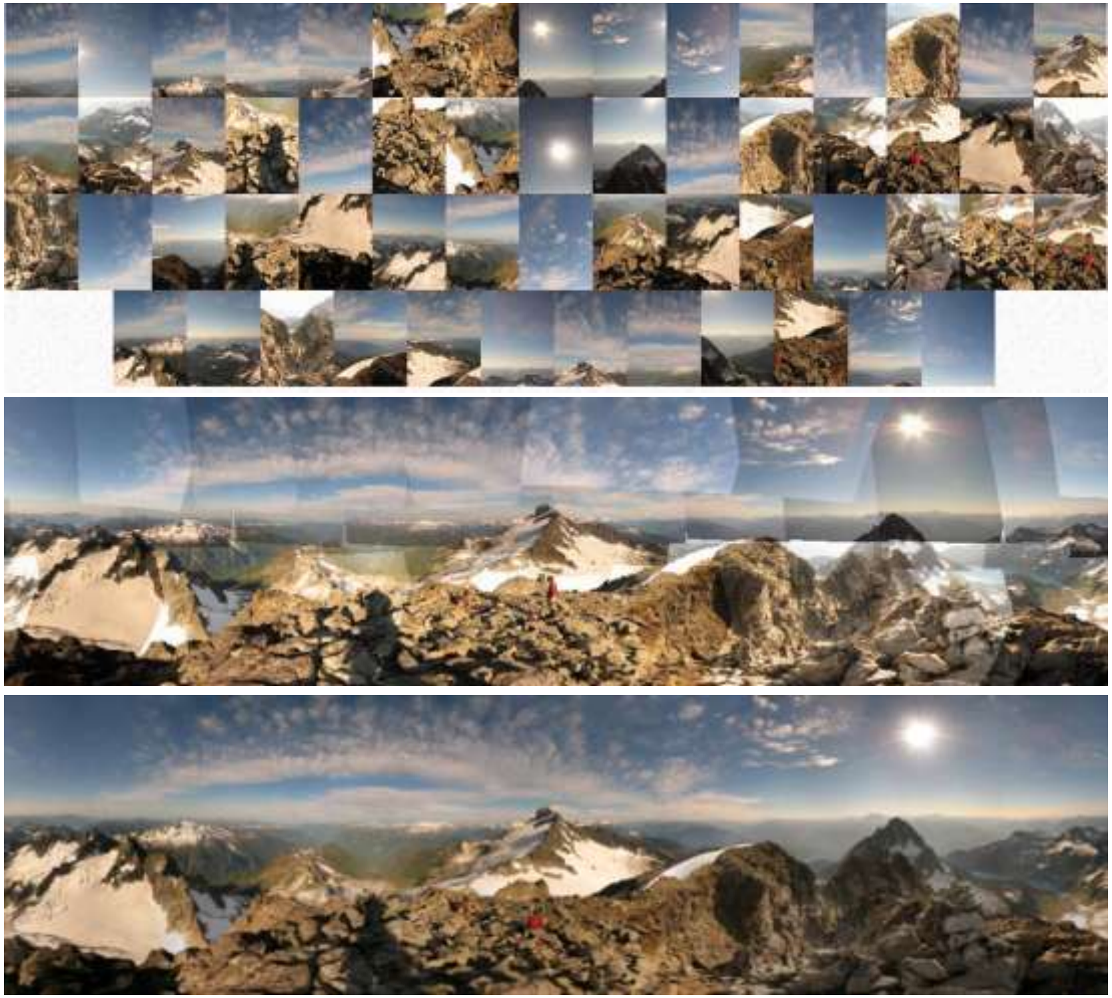
Feature-based Alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

Automatic Mosaicing



http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Recognition of Specific Objects, Scenes

Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

Kristen Grauman

Preview

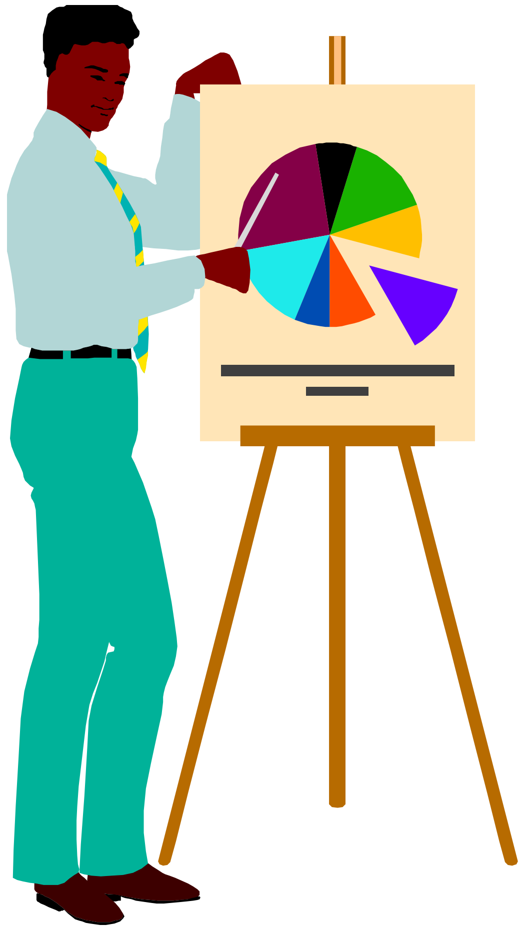
- Interest point detection
  - Edges: derivatives of Gaussians (first and second order)
  - Blobs: Laplacian of Gaussian, automatic scale selection
  - Harris corner detector
  - Template matching

- Invariant descriptors
  - Rotation according to dominant gradient direction
  - Histograms for robustness to small shifts and translations (SIFT descriptor)

Feature Extraction

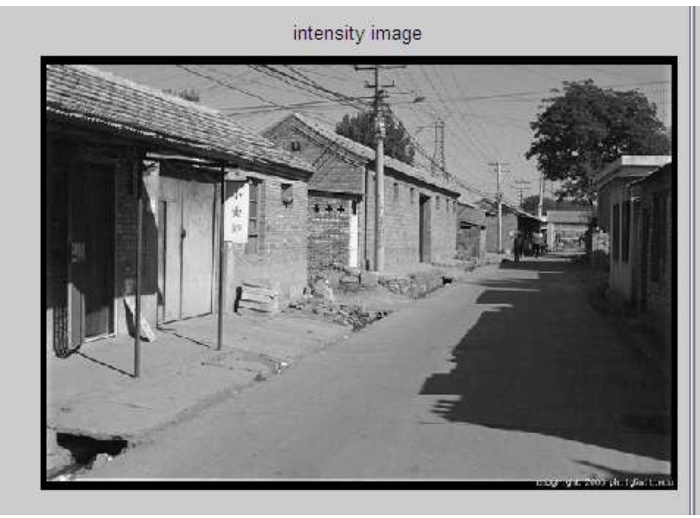
by

Image Filtering

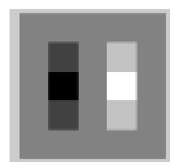


Recap: Edge Detection

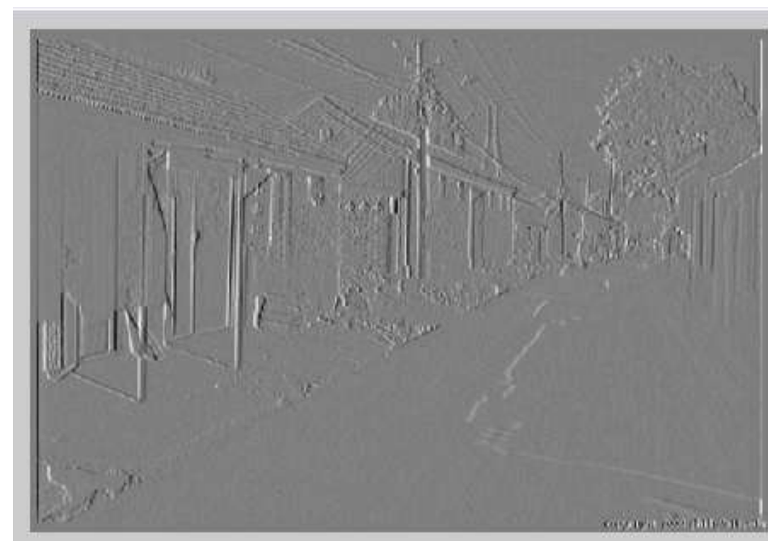
1	0	-1
2	0	-2
1	0	-1



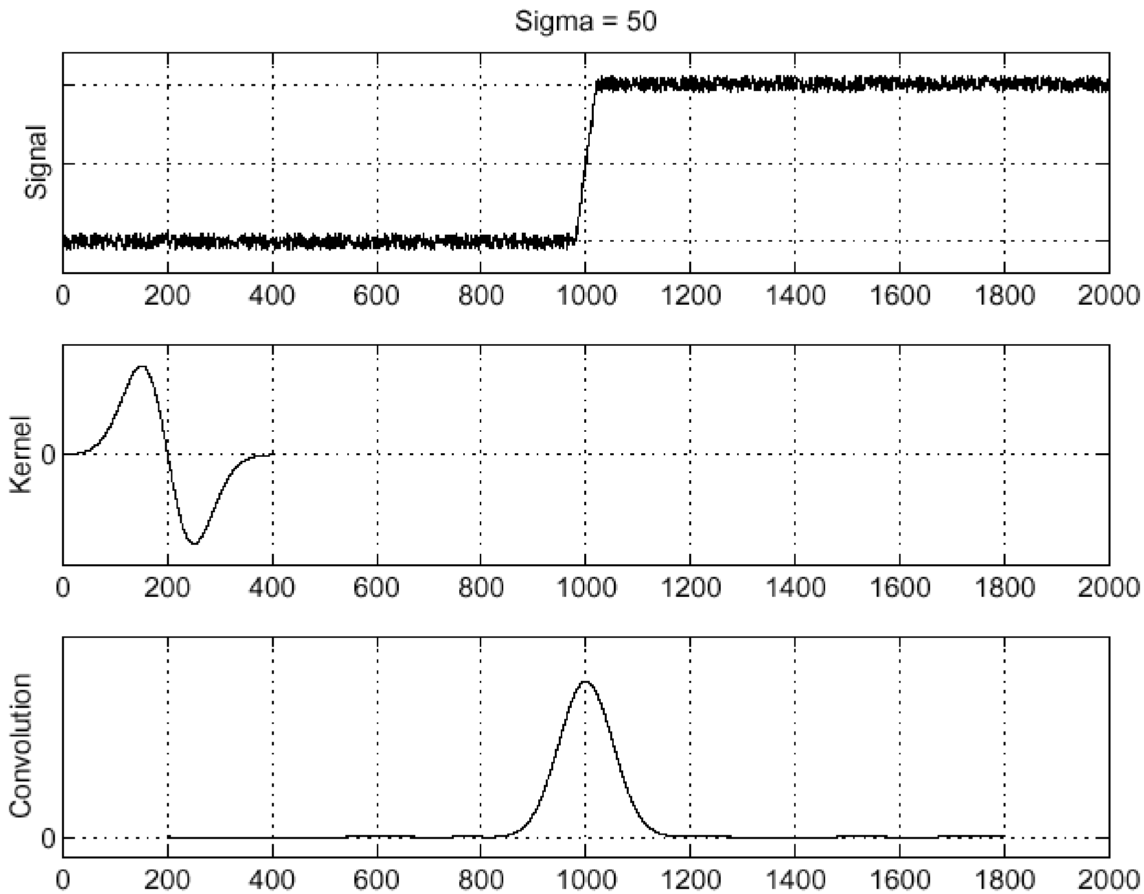
*



=



Recap: Edge Detection



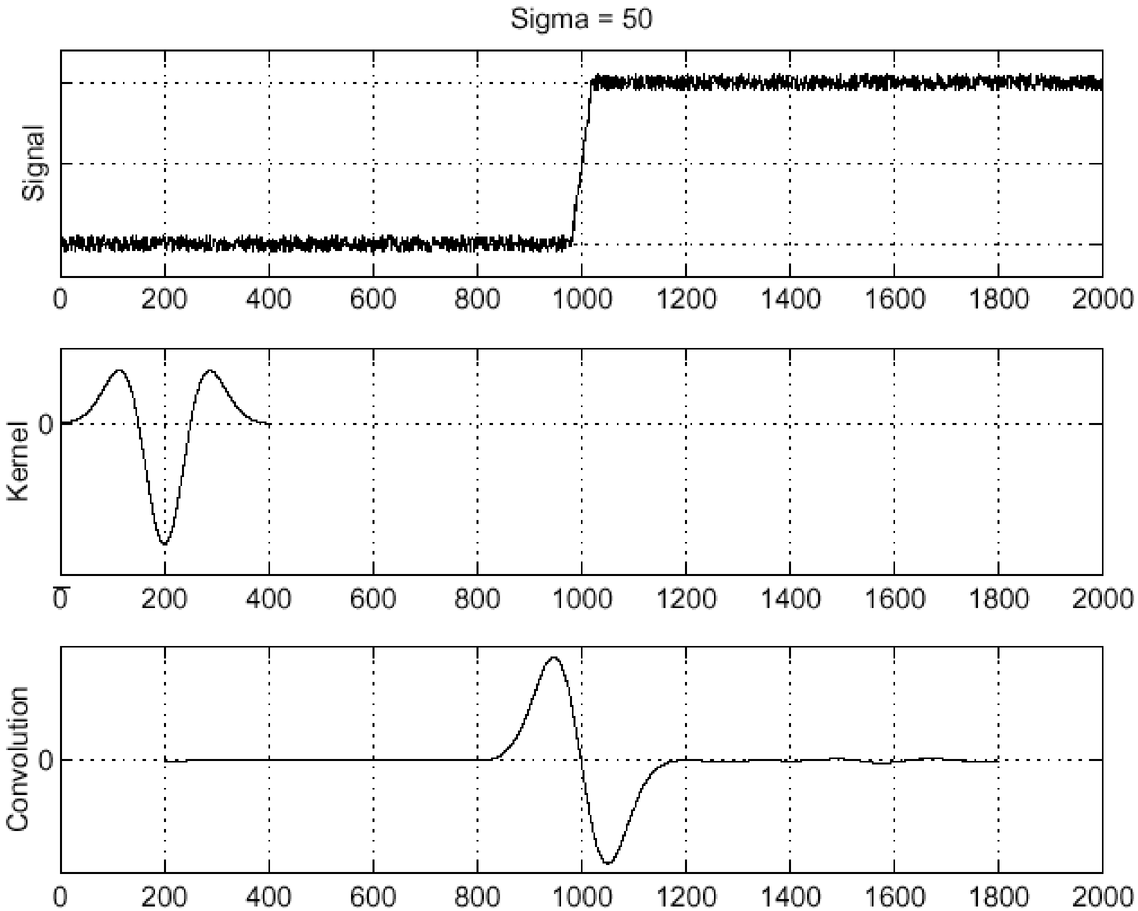
f Edge

$\frac{d}{dx} g$ Derivative of Gaussian

$f * \frac{d}{dx} g$ Edge = maximum of derivative

Source: S. Seitz

Recap: Edge Detection



f Edge

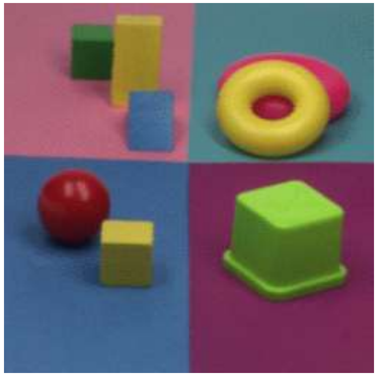
$\frac{d^2}{dx^2} g$ Second derivative of Gaussian (Laplacian)

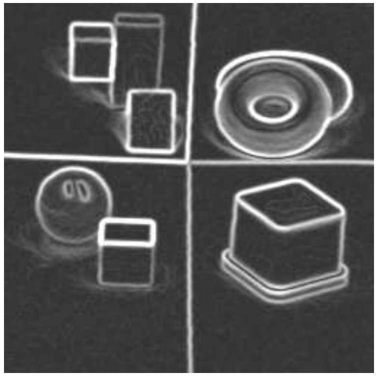
$f * \frac{d^2}{dx^2} g$ Edge = zero crossing of second derivative

Source: S. Seitz

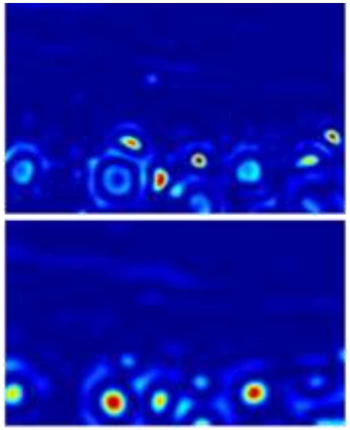
Summary

- Interest point detection
  - Edges: derivatives of Gaussians (first and second order)





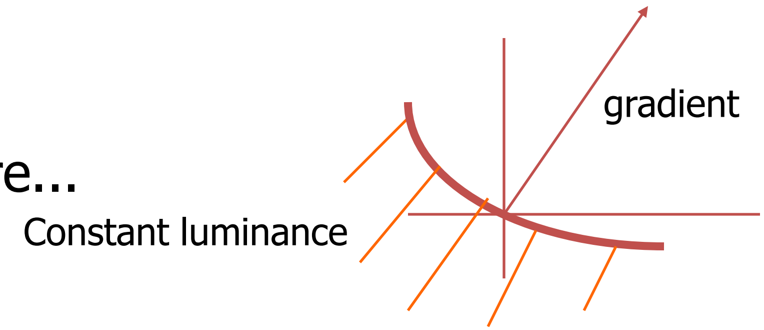
  - Blobs: Laplacian of Gaussian, automatic scale selection



Classifying of Color Edges

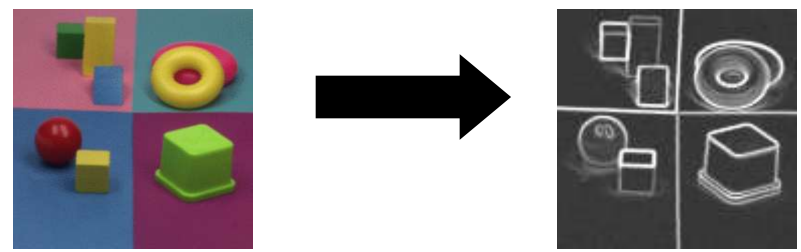
Edge Detection

An intensity edge is defined as a point where...



the gradient is large:

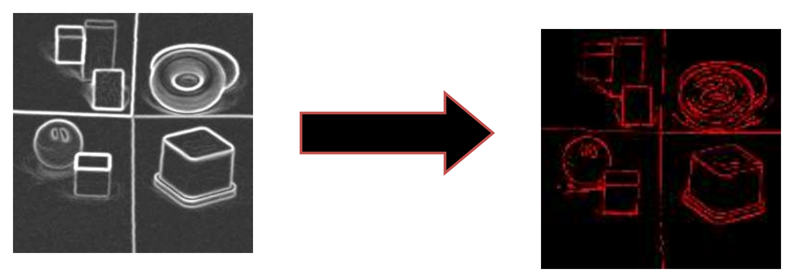
$$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2} >> 0$$



gradient computation

Localization - Laplacian - zero-crossing:

$$\nabla^2 f(x,y) = f_{xx} + f_{yy}$$



zero-crossings at large gradients

Edge Detection

Summing the gradient magnitudes separately:

$$|\nabla C(x,y)| = \sqrt{(R_x^2 + R_y^2)} + \sqrt{(G_x^2 + G_y^2)} + \sqrt{(B_x^2 + B_y^2)}$$

or using the Euclidean metric:

$$|\nabla C(x,y)| = \sqrt{R_x^2 + R_y^2 + G_x^2 + G_y^2 + B_x^2 + B_y^2}$$

or using eigen-values: [diZenzo86], [Sapiro96]

Edge Localization

Edge localization by:

1. Non-maxima suppression based on the direction of the minimal and maximal change given by the eigen-vectors:

$$\theta_{-} \text{ and } \theta_{+}$$

2. Zero-crossing (change in sign) by scanning:

$$\nabla^2 C_{RGB} = (R_{xx} + R_{yy} + G_{xx} + G_{yy} + B_{xx} + B_{yy})$$

$$\nabla^2 C_{c_1 c_2 c_2} = (c_{1_{xx}} + c_{1_{yy}} + c_{2_{xx}} + c_{2_{yy}} + c_{3_{xx}} + c_{3_{yy}})$$

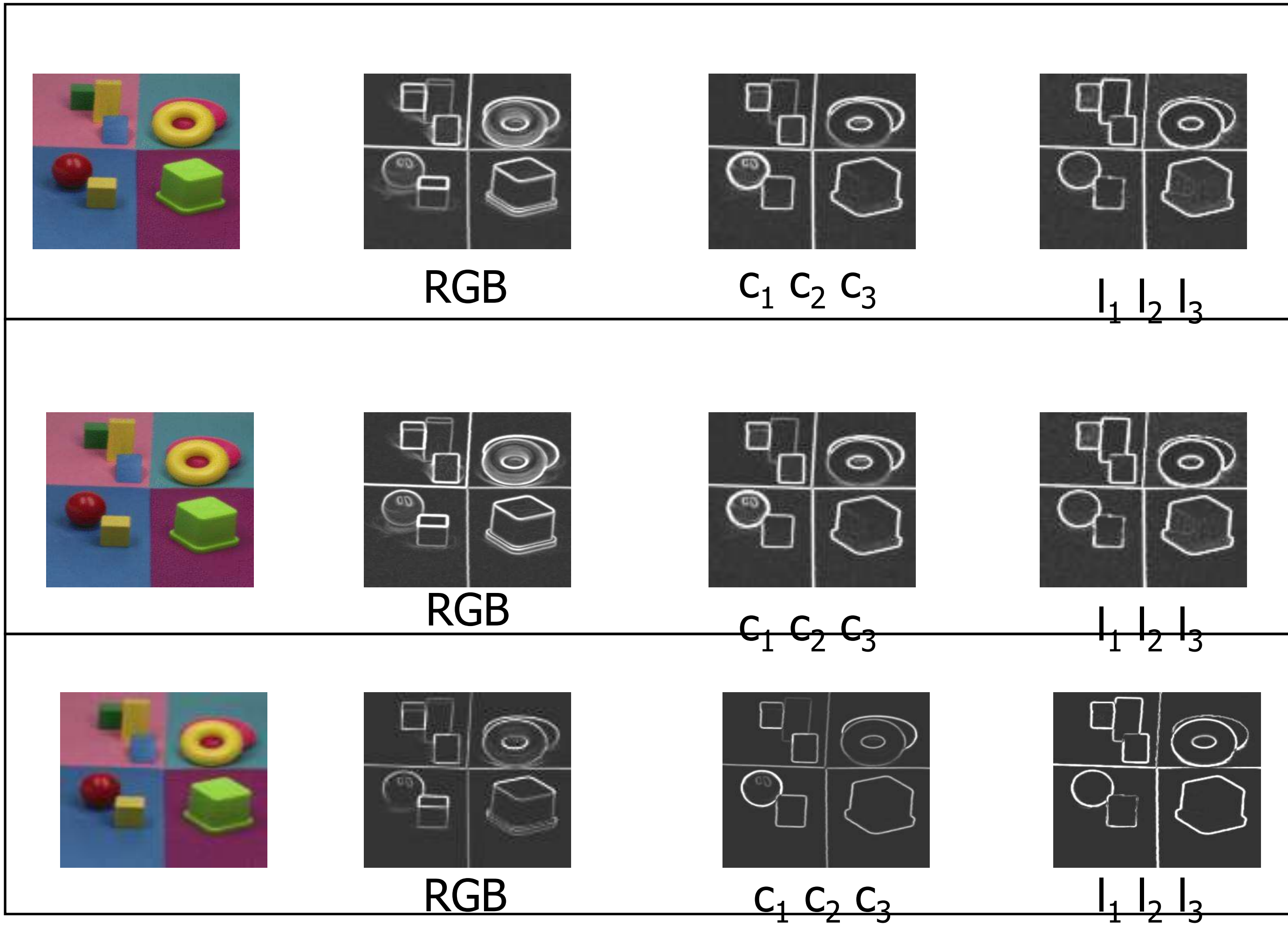
$$\nabla^2 C_{l_1 l_2 l_2} = (l_{1_{xx}} + l_{1_{yy}} + l_{2_{xx}} + l_{2_{yy}} + l_{3_{xx}} + l_{3_{yy}})$$

Edge Classification

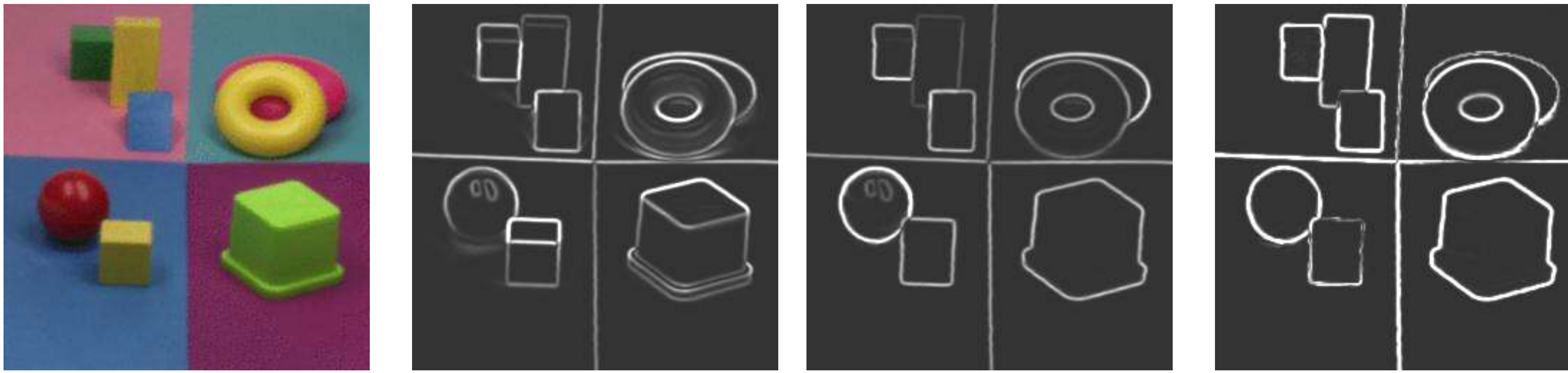
summing gradient magnitudes separately:

Euclidean:

eigen-values:



Edge Classification



RGB $\quad c_1, c_2, c_3 \quad l_1, l_2, l_3$

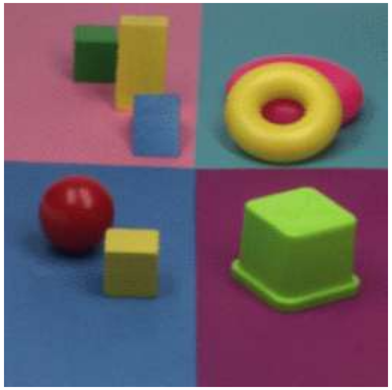
if $(|\nabla C_{c_1c_2c_3}| \geq \mathrm{t}_{c_1c_2c_3} \,\&\, |\nabla C_{l_1l_2l_3}| < \mathrm{t}_{l_1l_2l_3})$ then classify as highlight edge

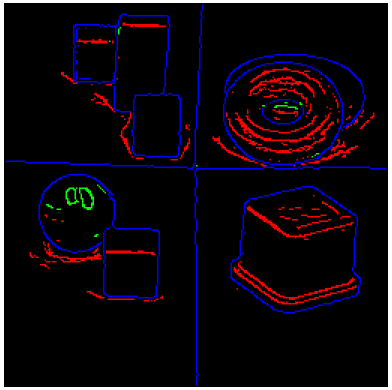
else

if $(|\nabla C_{l_1l_2l_3}| \geq \mathrm{t}_{l_1l_2l_3})$ then classify as color edge

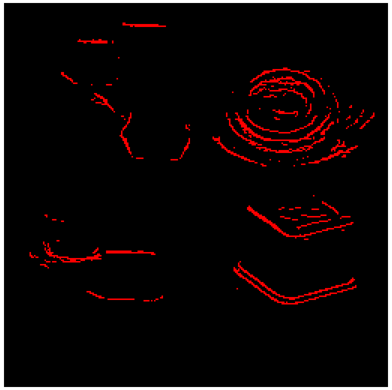
else classify as shadow/geometry edge

Edge Classification

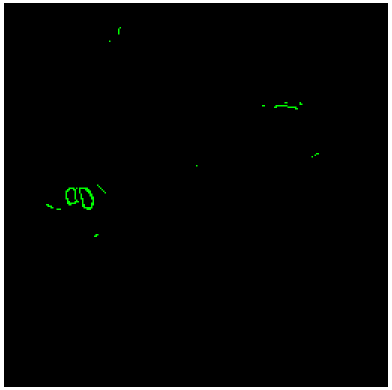




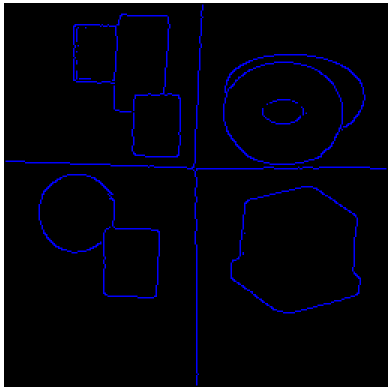
colour edge maxima by type



shadows and geometry



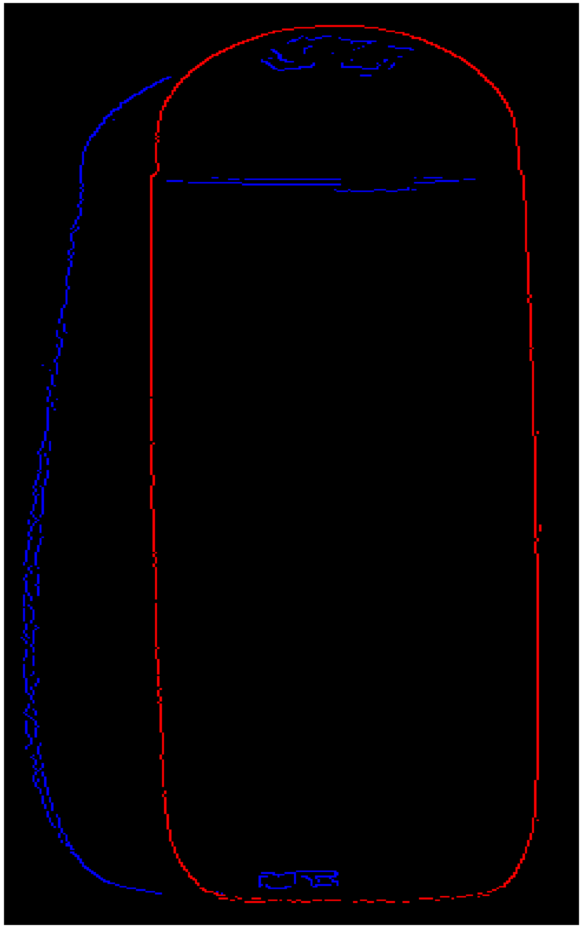
highlights



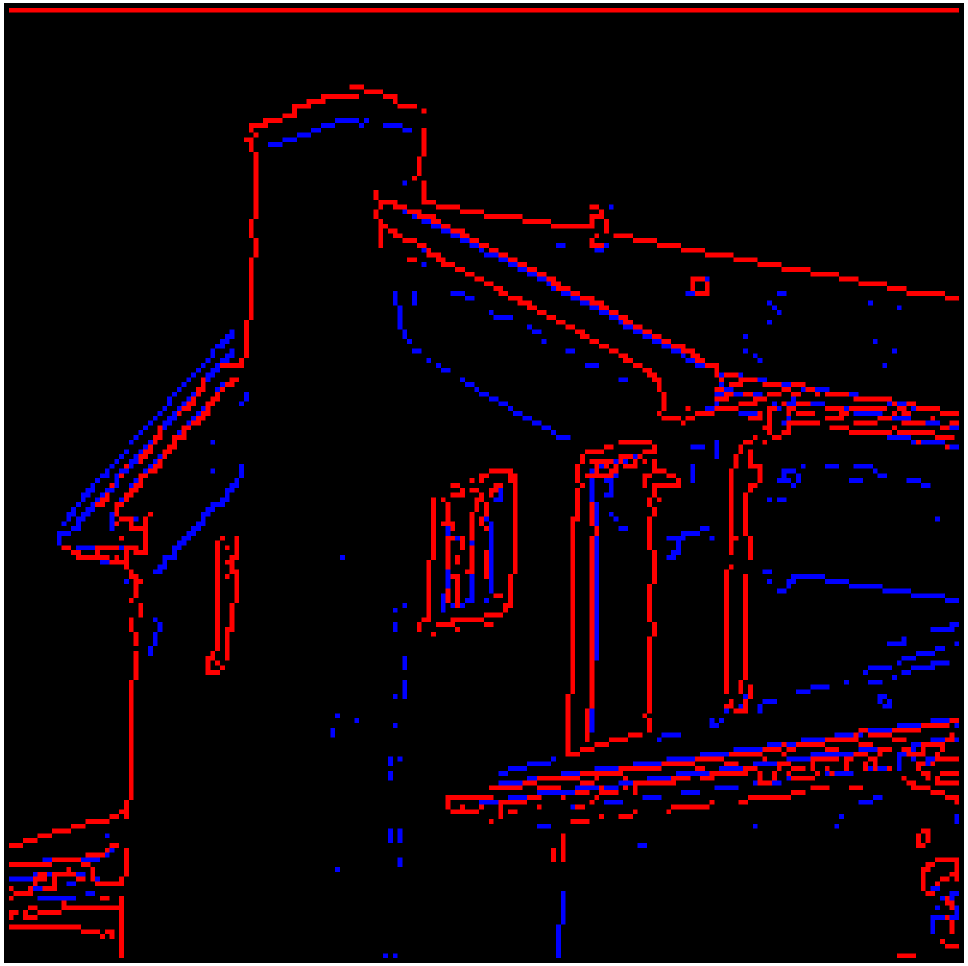
colour edges

Edge Classification

material
shadow or geometry

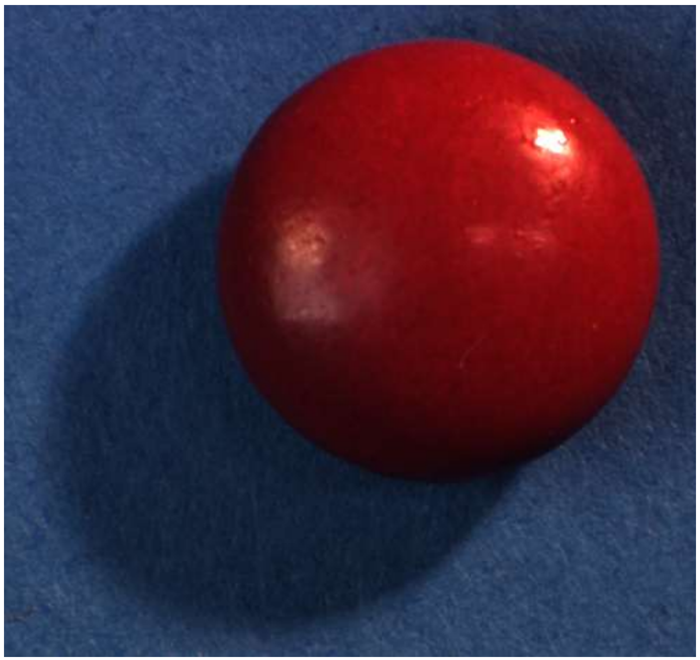


Edge Classification

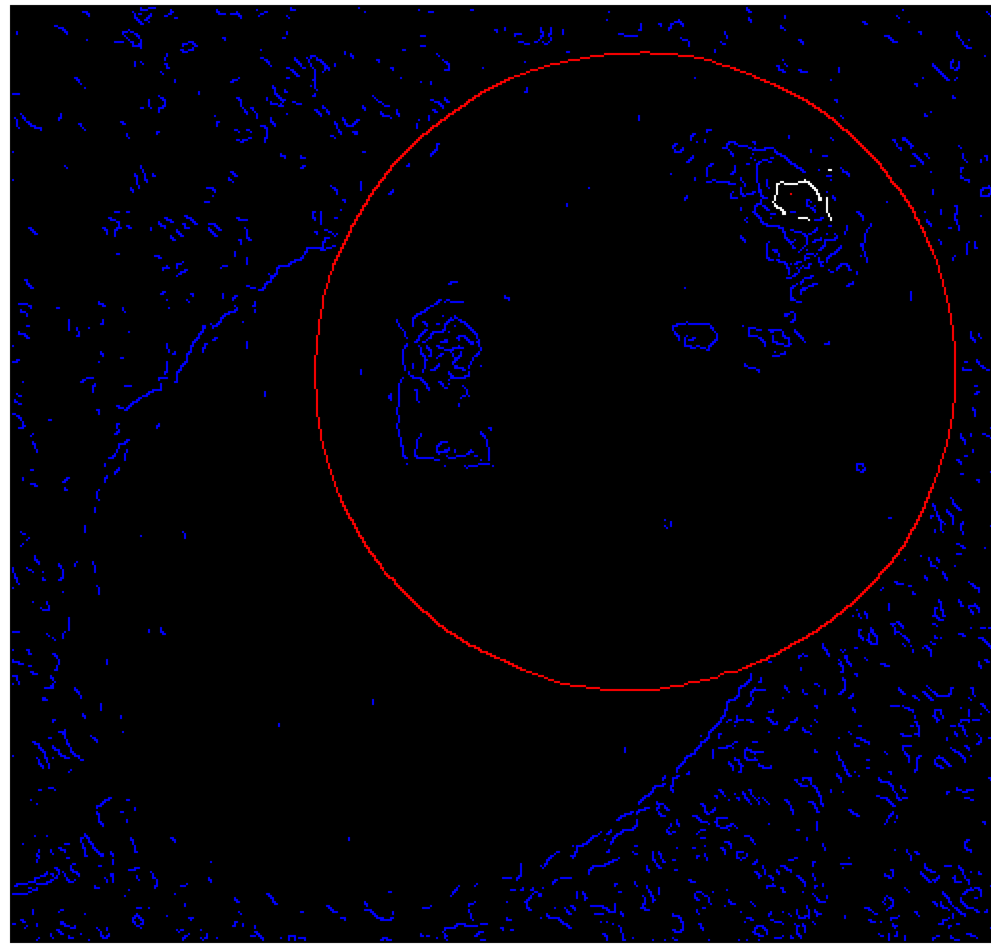


material
highlight
shadow or geometry

Edge Classification



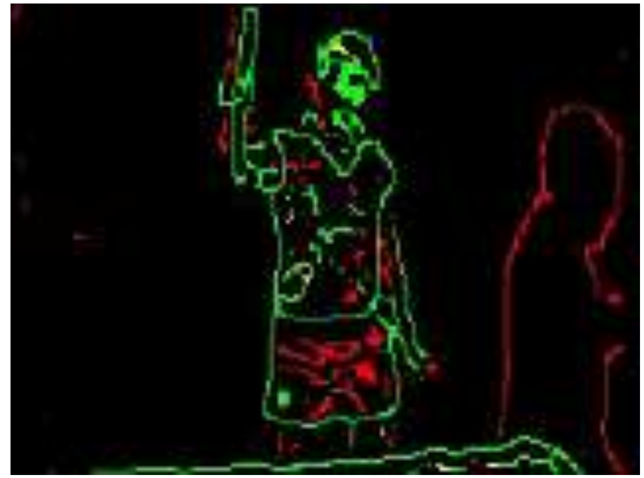
material
highlight
shadow or geometry



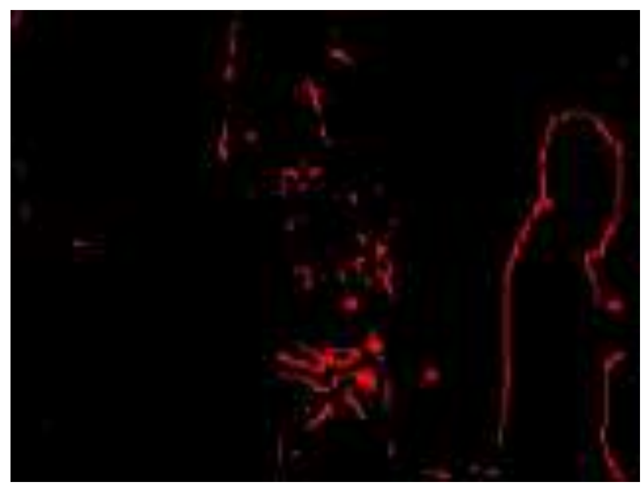
Demo: Edge Classification

video

classification



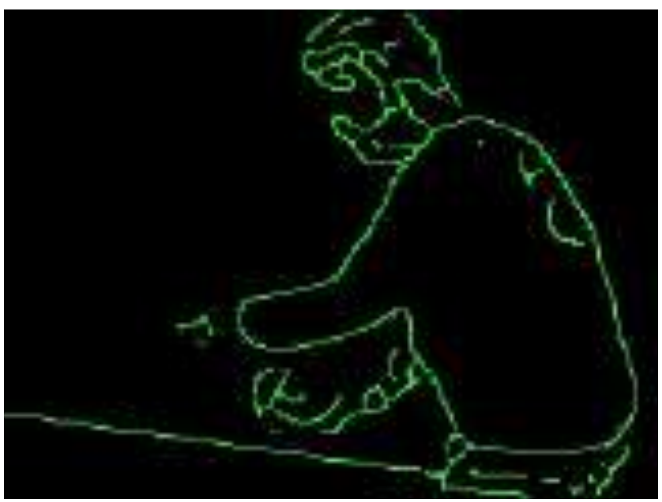
shadow-shading



Demo: Edge Classification

video

classification



material

shadow-shading

Deformable Contours

Contour which minimises elastic energy:

$b|v_k(s)|^2 + f(v(s))$

Advanced segmentation suited for query formulation

Deformable Contours

Deformable curve : $\mathrm{v}(t) = [x(t), y(t)], t \in [0,1]$

The energy : $E = \alpha E_{\mathrm{int}} + \beta E_{\mathrm{ext}}$

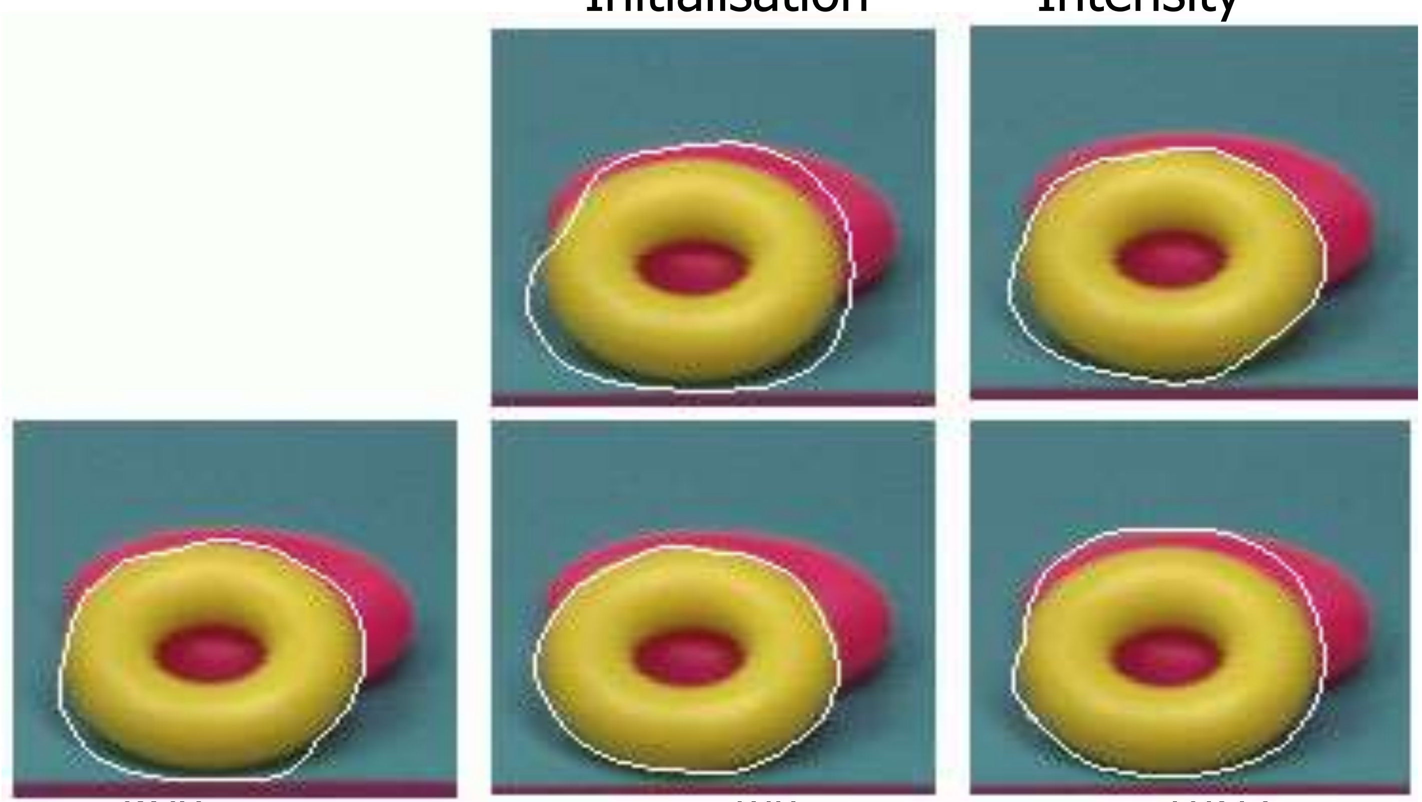
Internal energy : $E_{\mathrm{int}} = (\oint_t (\| \mathrm{v}(t)' \|^2 + (\| \mathrm{v}(t)'' \|^2 \, dt)(\oint_t \| \mathrm{v}(t)' \|^2)$

External energy (intensity gradient) : $E_{\mathrm{ext}} = \oint_t - \| \nabla I(x,y) \| \, dt$

External energy (color invariant gradient) : $E_{\mathrm{ext}} = \oint_t - \| \nabla C(x,y) \| \, dt$

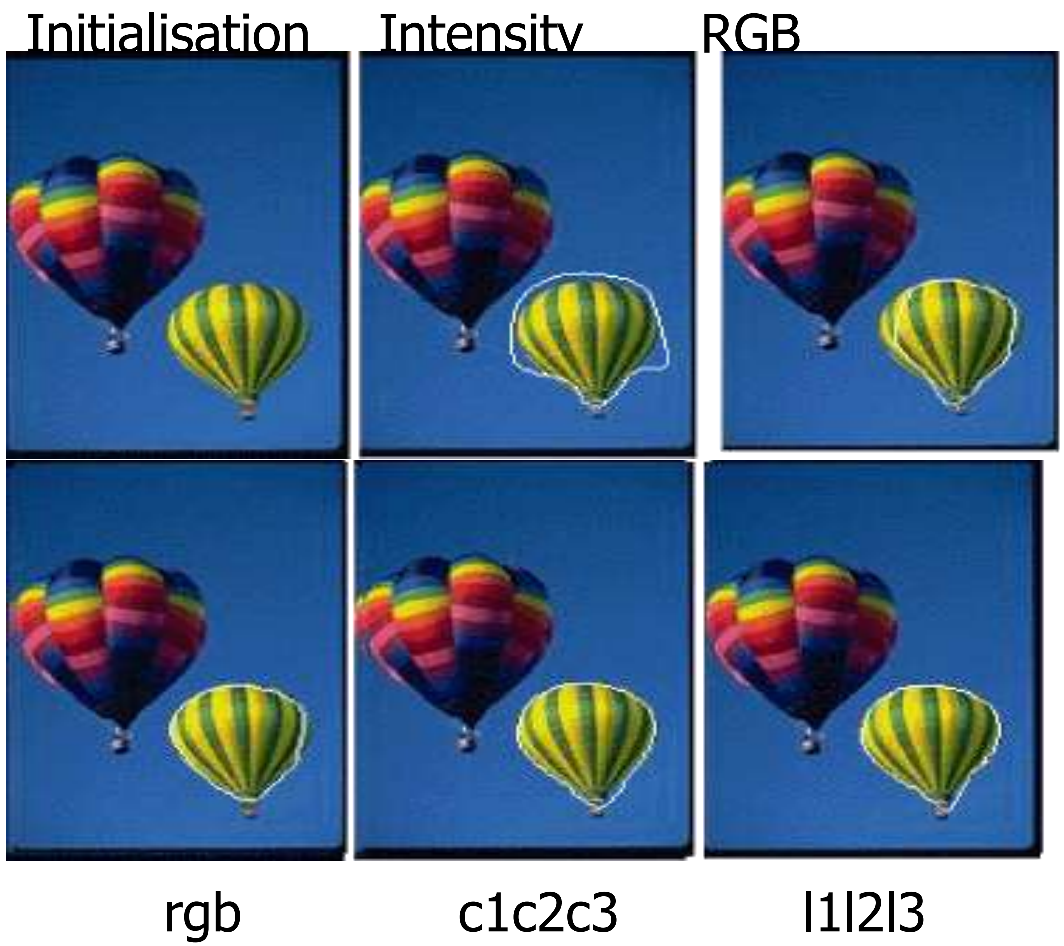
Deformable Contours

Initialisation Intensity

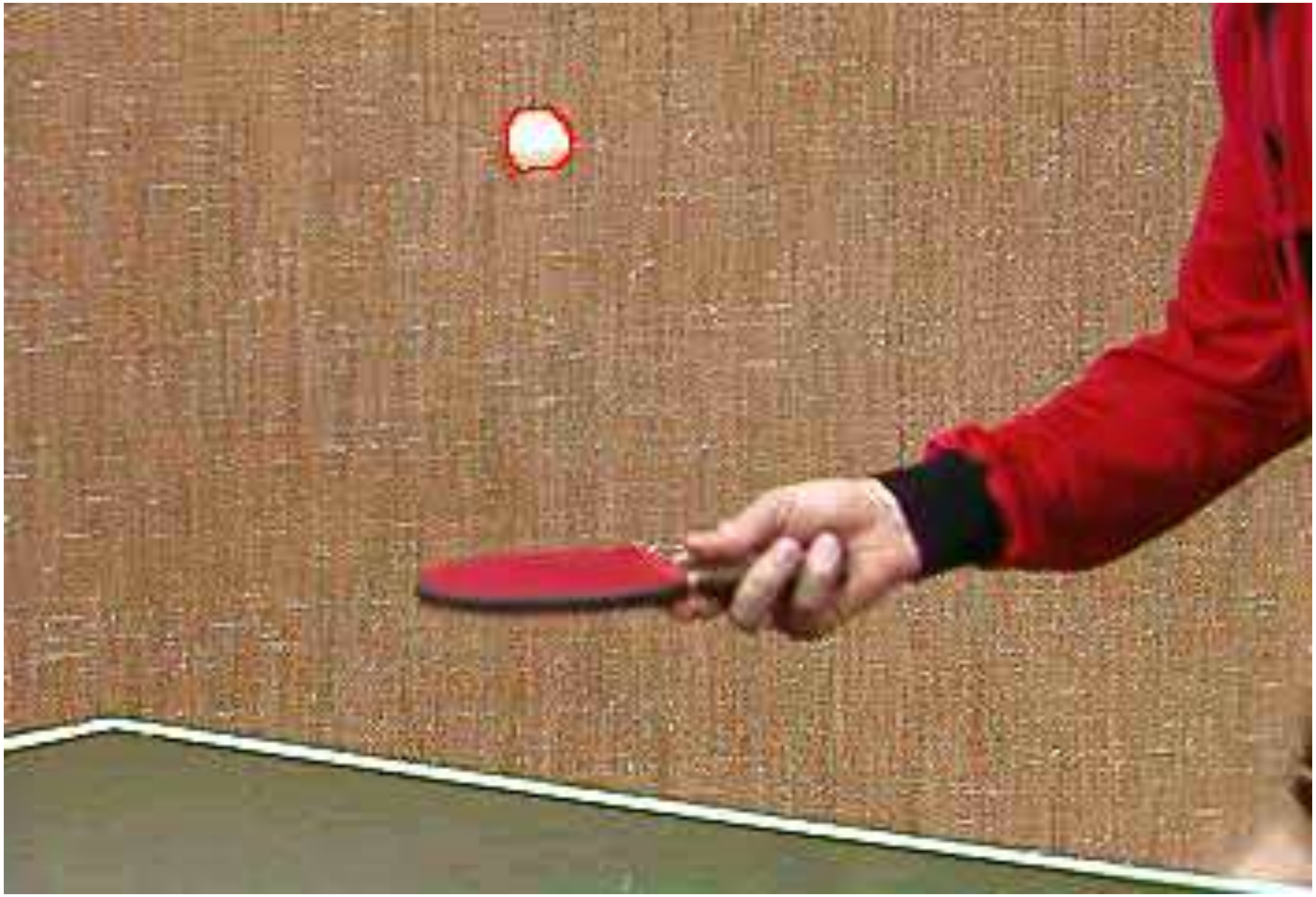


RGB rgb l1l2l3

Deformable Contours



Demo: Tracking by Deformable Contours



Demo: Tracking by Deformable Contours

Shadow Removal

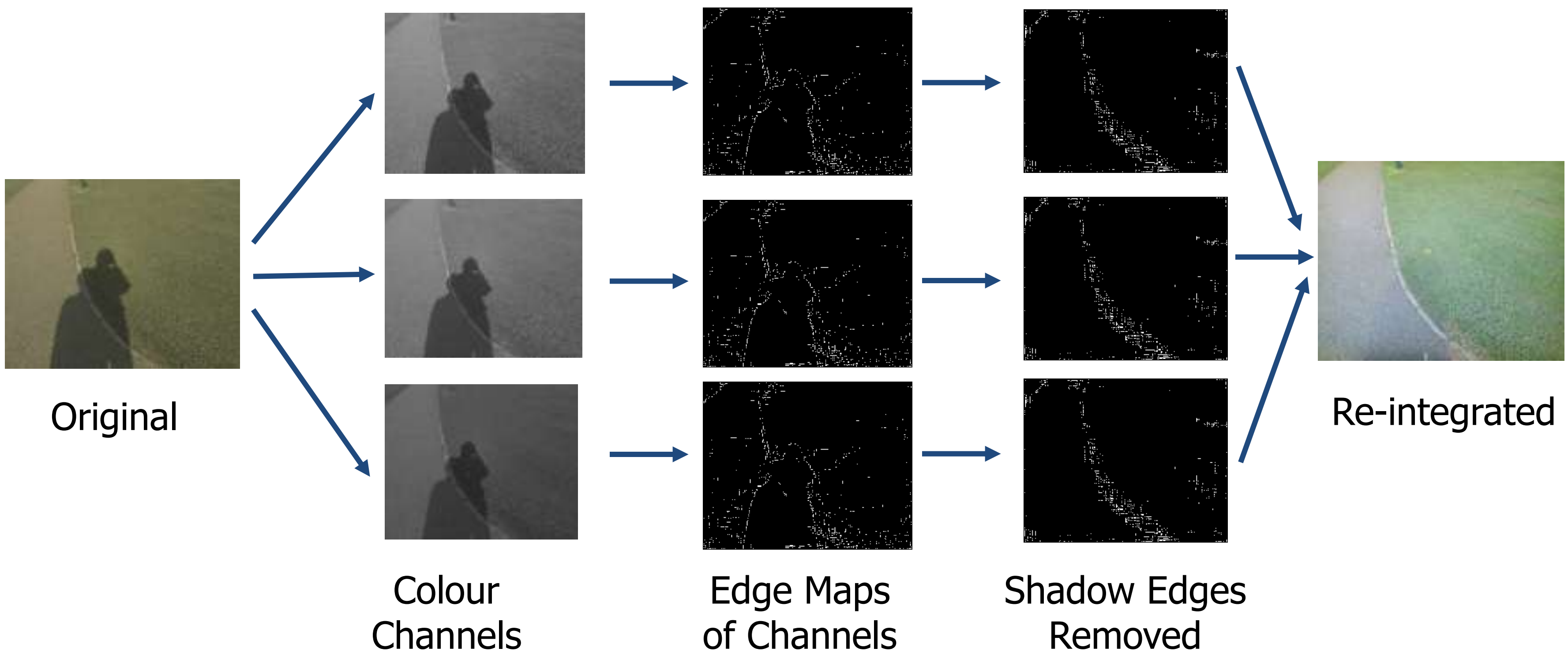
by

Graham Finlayson

Shadow Removal

We would like to go from a colour image with shadows, to the same colour image, but without the shadows.

Shadow Removal



An Example

Original Image

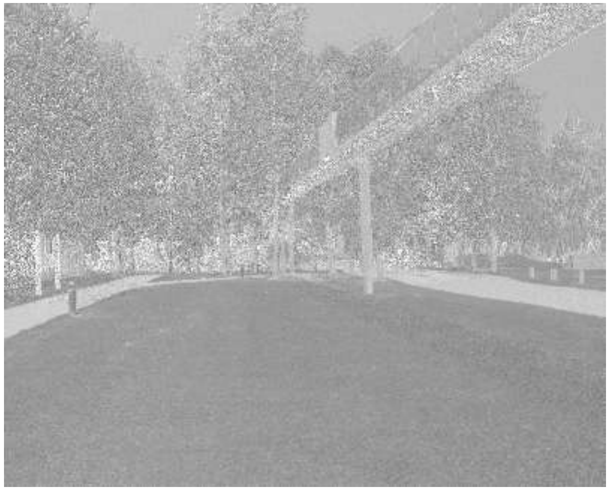
Invariant Image

Detected Shadow Edges

Shadow Removed

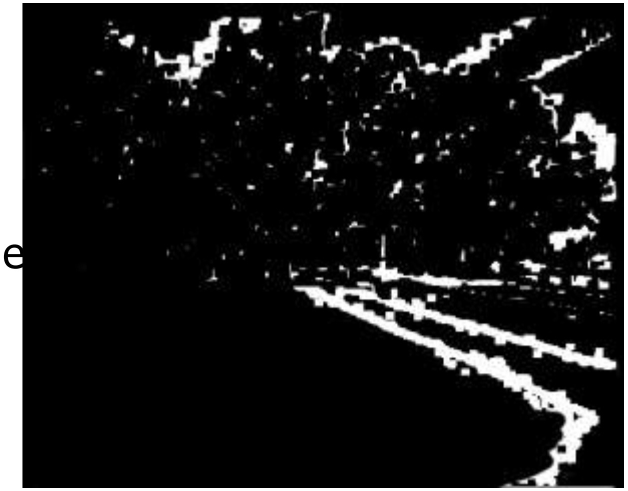
A Second Example

Original Image



Invariant Image

Detected Shadow Edge



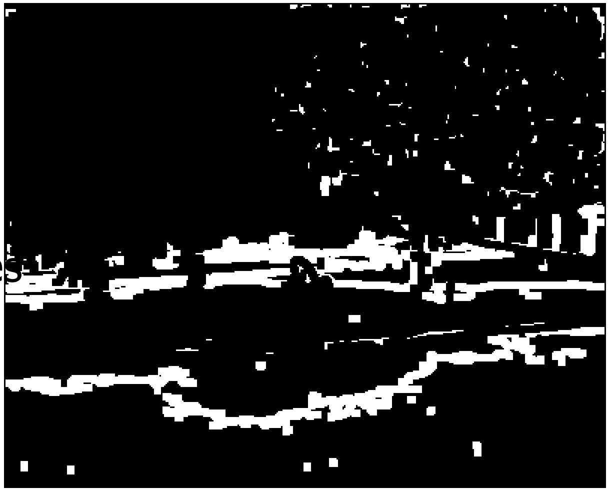
Shadow Removed

More Examples

Original Image

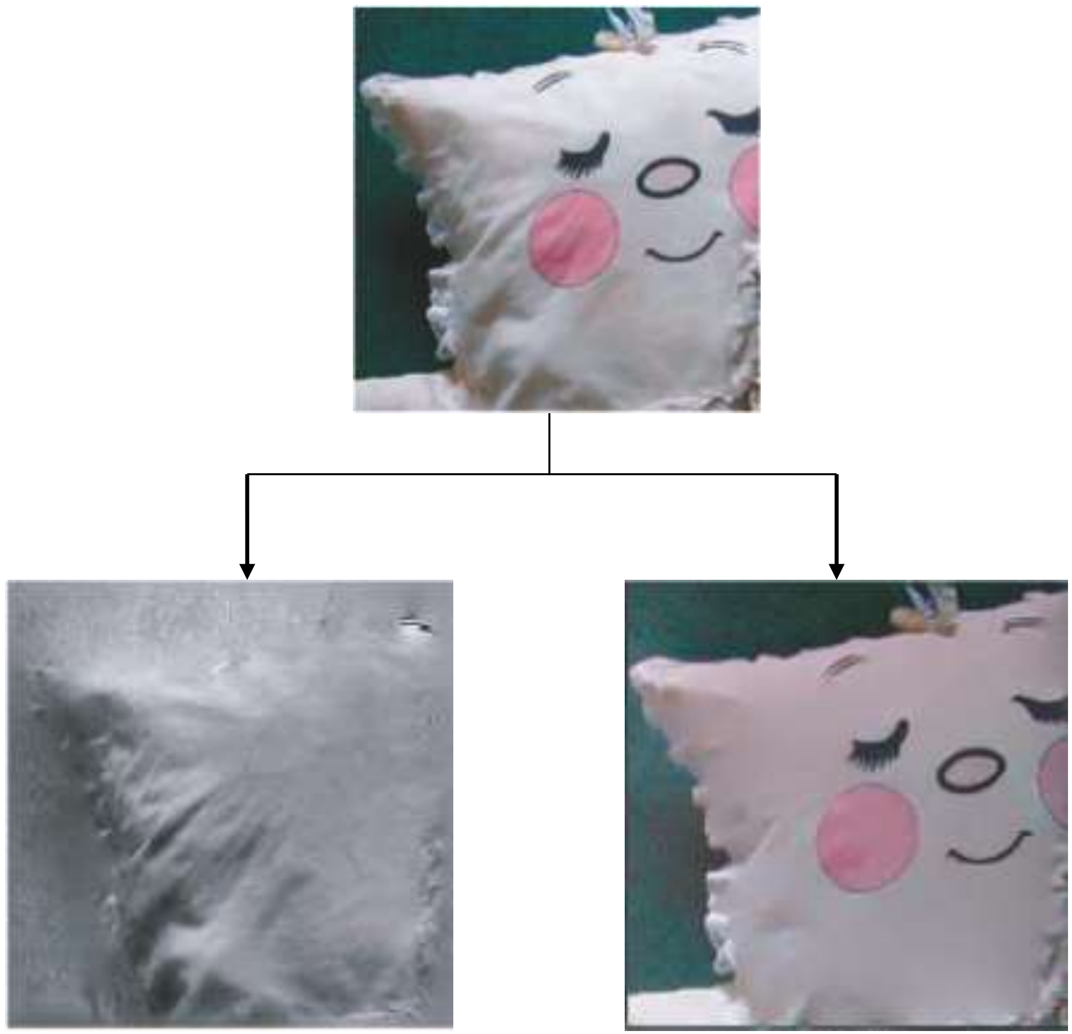
Invariant Image

Detected Shadow Edges

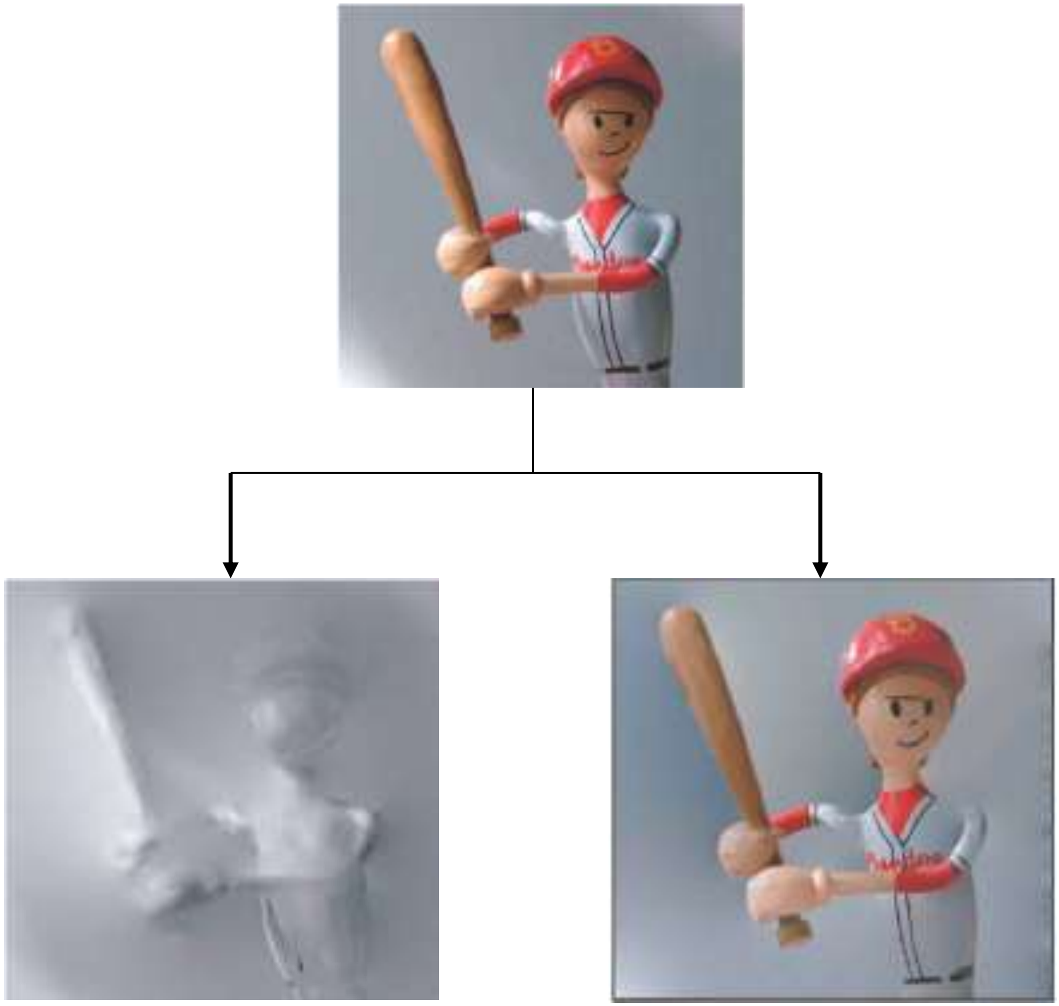


Shadow Removed

Intrinsic Images



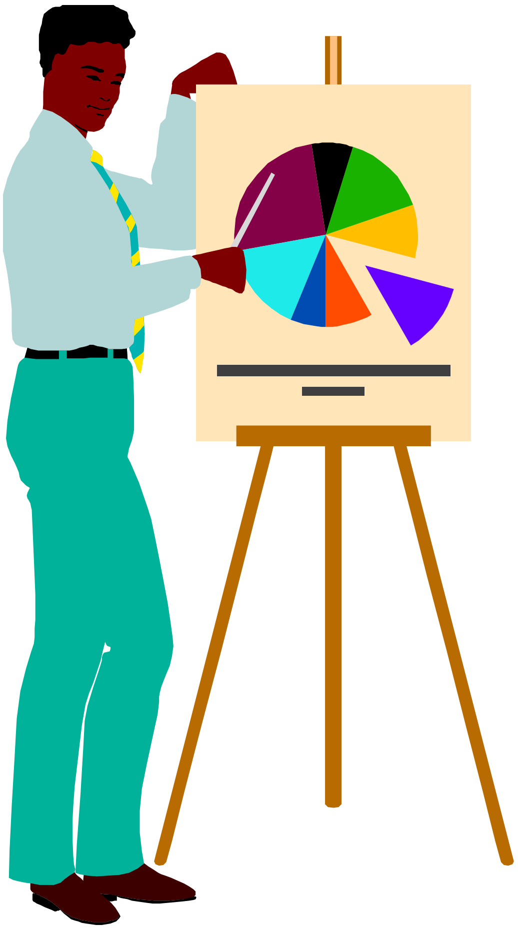
Intrinsic Images



Feature Extraction

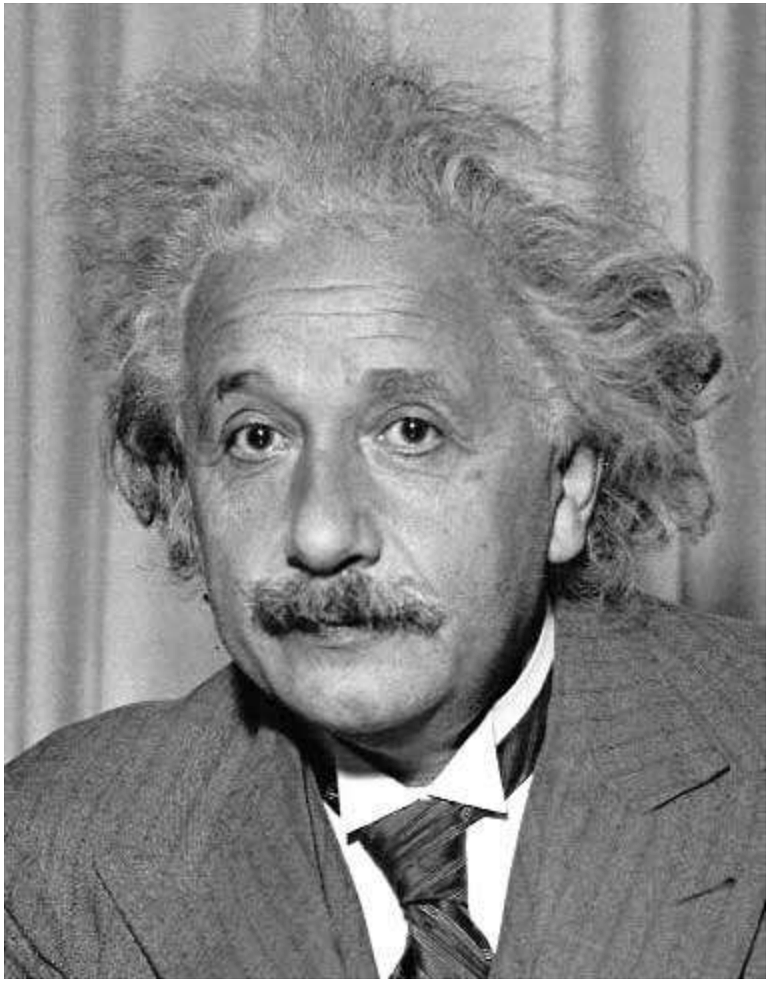
by

Template Matching



Intermezzo: Template Matching

- Goal: find  in image



- Main challenge: What is a good similarity or distance measure between two patches?
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation

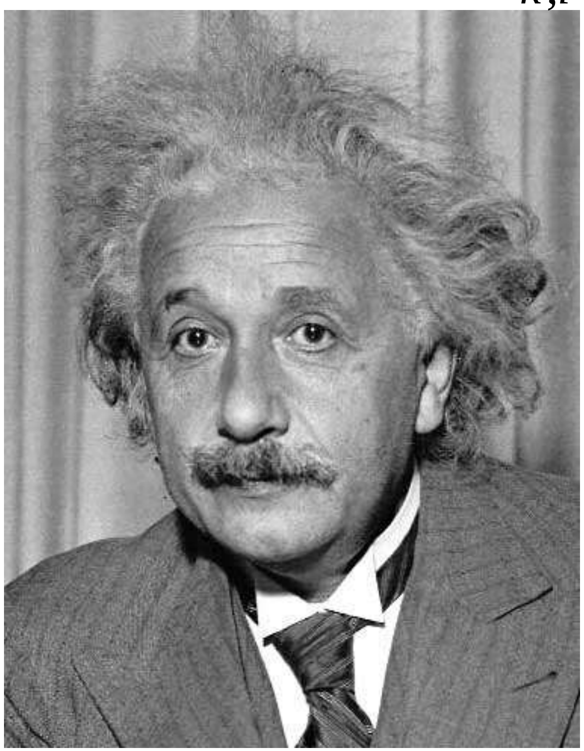
Slide: Hoiem

Intermezzo: Template Matching

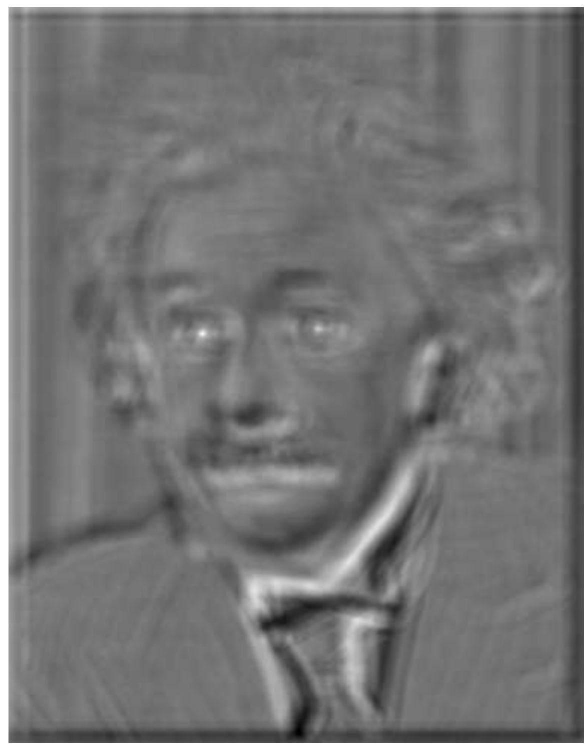
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (f[k,l] - \bar{f})(g[m+k, n+l])$$

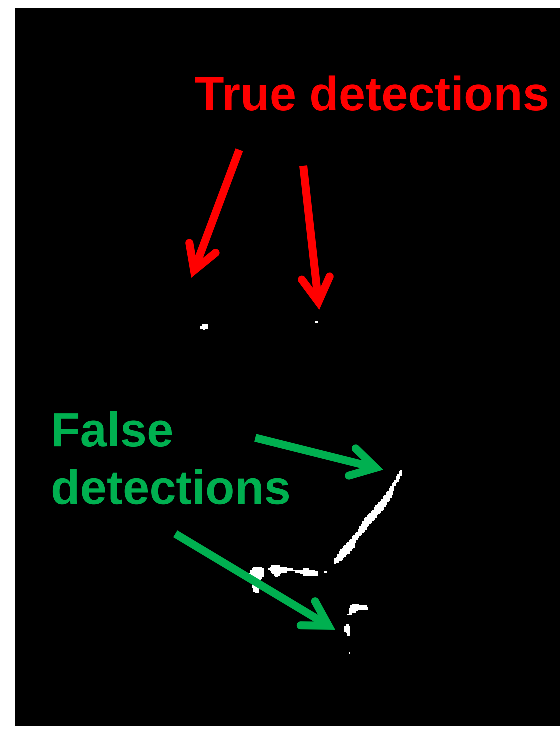
mean of f



Input



Filtered Image (scaled)



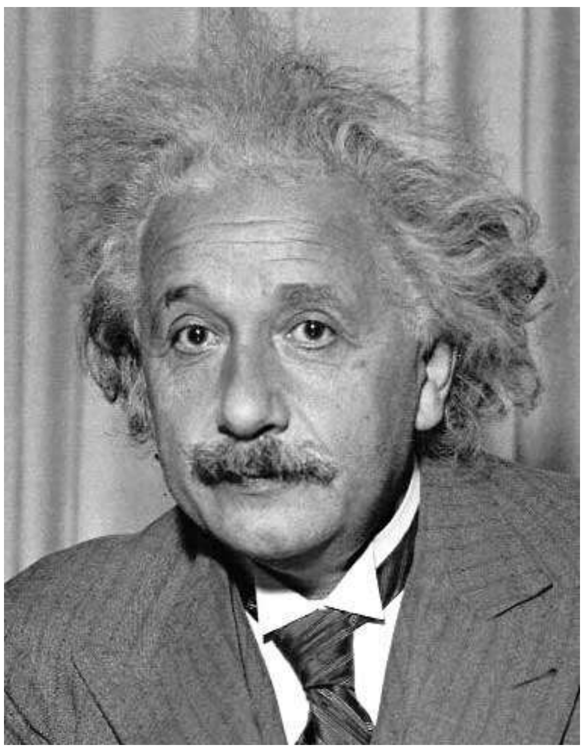
Thresholded Image

Slide: Hoiem

Intermezzo: Template Matching

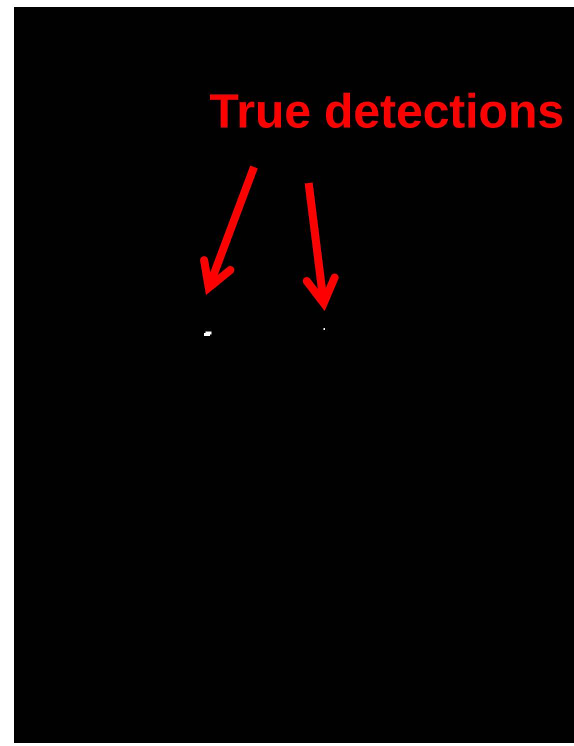
- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)



Thresholded Image

Slide: Hoiem

Intermezzo: Template Matching

- Goal: find  in image
- Method 3: Normalized cross-correlation

mean template

mean image patch

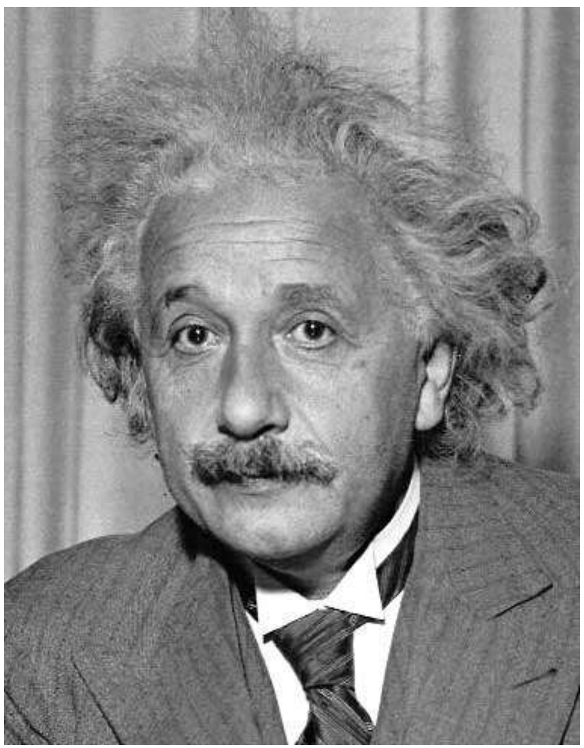
$$h[m,n] = \frac{\sum_{k,l} (g[k,l] - \bar{g})(f[m-k,n-l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k,l] - \bar{g})^2 \sum_{k,l} (f[m-k,n-l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

Slide: Hoiem

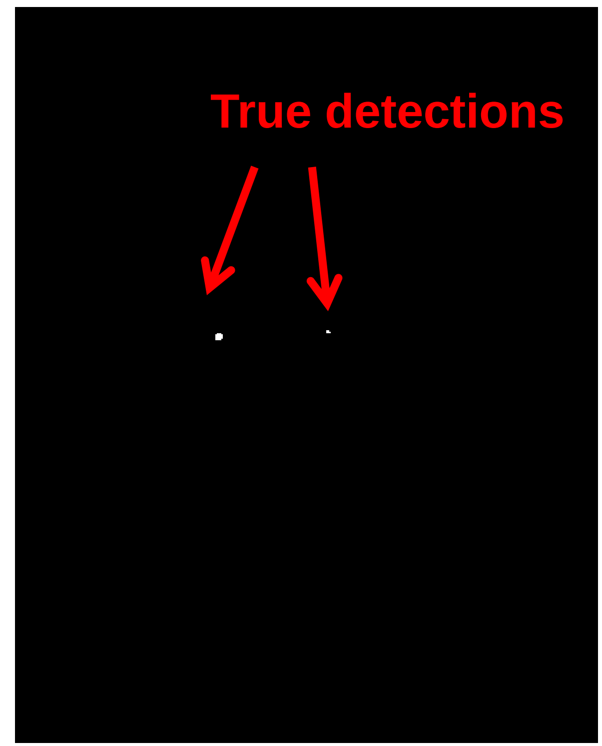
Intermezzo: Template Matching

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Slide: Hoiem

What is the best method to use?

A: Depends

- SSD: faster, sensitive to overall intensity
- Normalized cross-correlation: slower, invariant to local average intensity and contrast

Summary

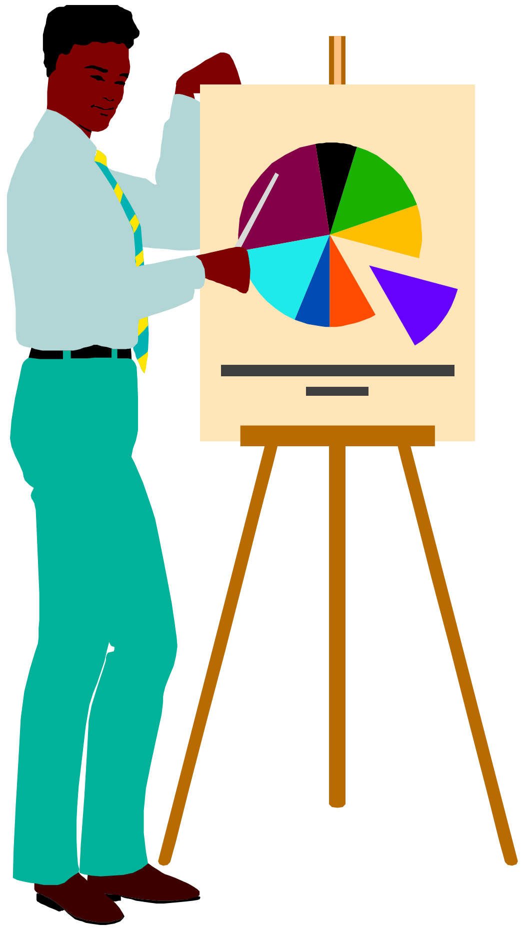
- Applications of filters
  - Template matching (SSD or Normxcorr2)
    - SSD can be done with linear filters, is sensitive to overall intensity
  - Gaussian pyramid
    - Coarse-to-fine search, multi-scale detection
  - Downsampling
    - Need to sufficiently low-pass before downsampling

Feature Extraction

by

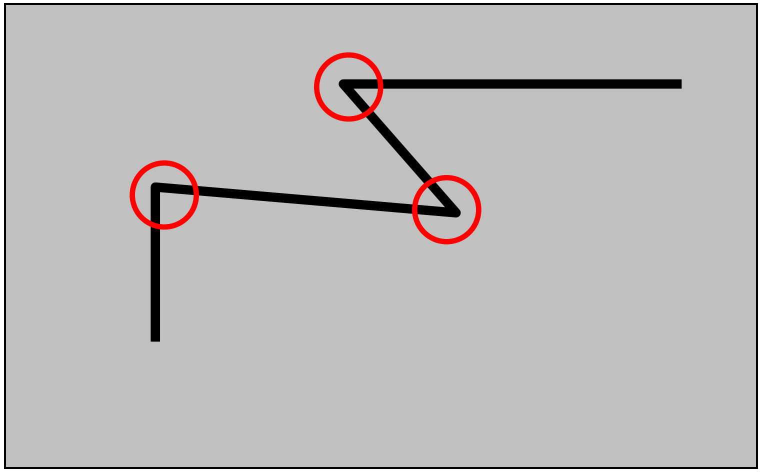
Image Filtering:

Corners



An introductory example:

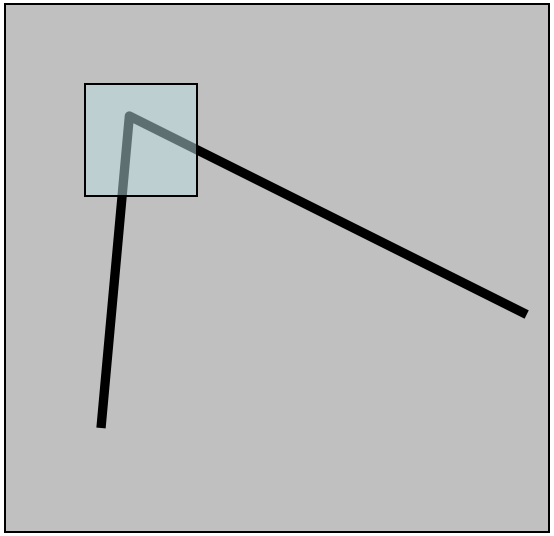
Harris corner detector



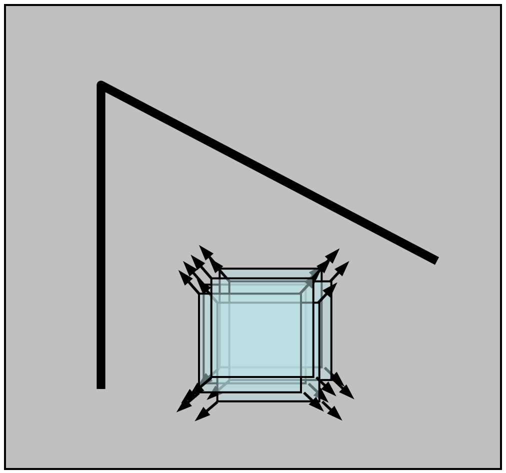
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

The Basic Idea

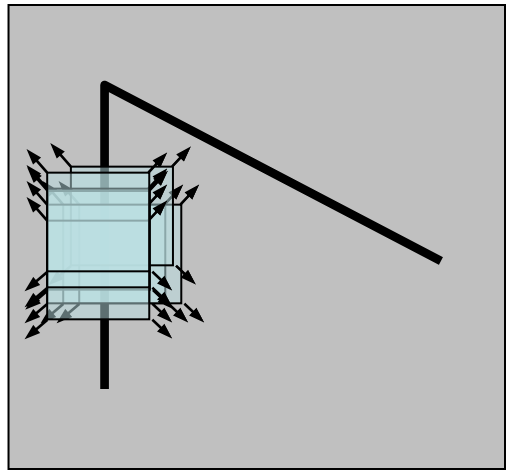
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



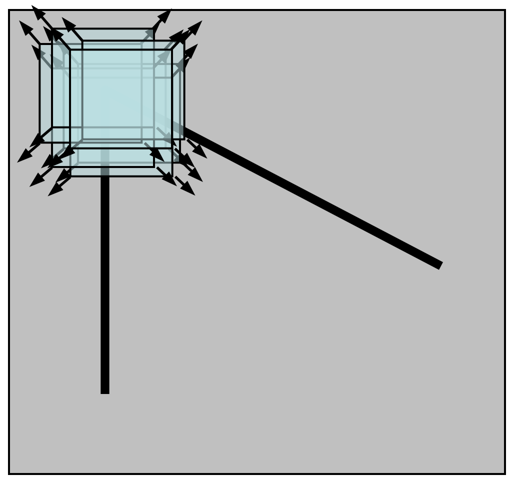
Harris Detector: Basic Idea



“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 the shift [u,v]:

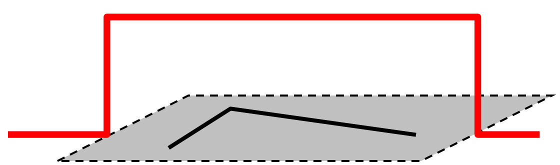
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

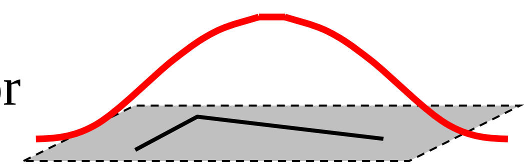
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Harris Detector: Mathematics

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

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

where M is a 2x2 matrix computed from image derivatives:

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

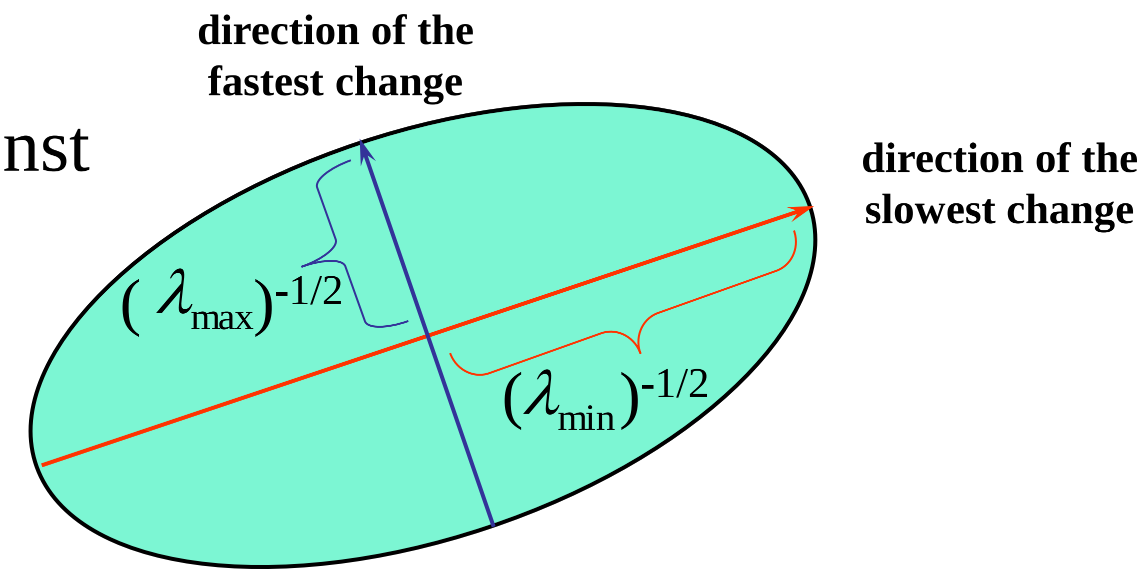
Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

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

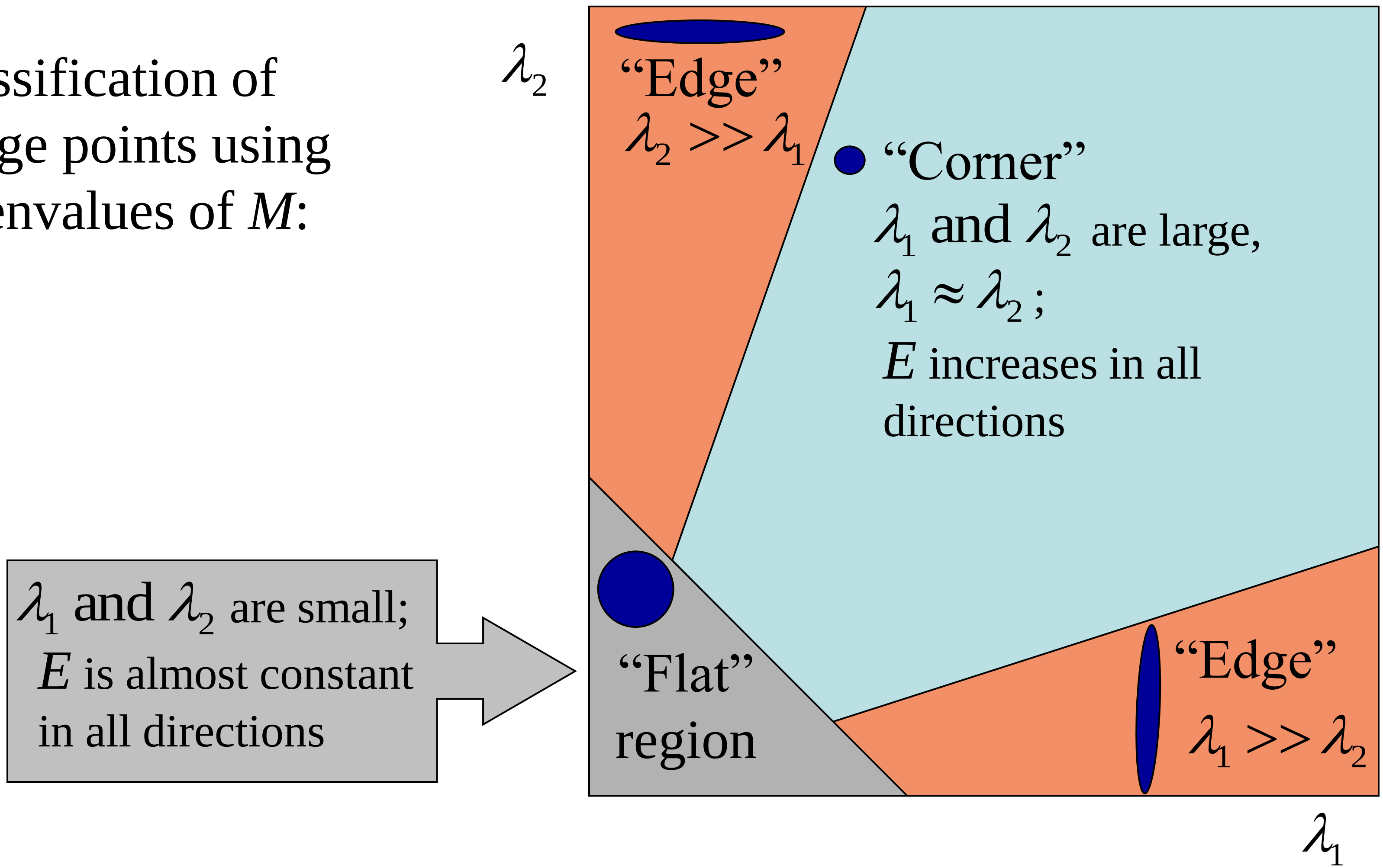
λ_1 and λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



Harris Detector: Mathematics

Classification of image points using eigenvalues of M:



Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

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

(k – empirical constant, k = 0.04-0.06)

Summary of the Harris detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x.I_x \quad I_{y2} = I_y.I_y \quad I_{xy} = I_x.I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma\prime} * I_{x2} \quad S_{y2} = G_{\sigma\prime} * I_{y2} \quad S_{xy} = G_{\sigma\prime} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

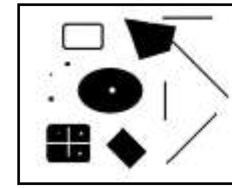
$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

$$R = Det(H) - k(Trace(H))^2$$

6. Threshold on value of R. Compute nonmax suppression.

Harris Detector [Harris88]

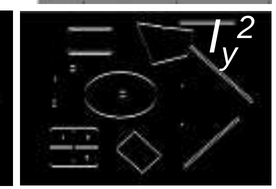


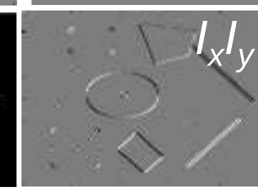
- Second moment matrix (autocorrelation matrix)

$$\mu(\sigma_I,\sigma_D)=g(\sigma_I)*\begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives

2. Square of derivatives





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

3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

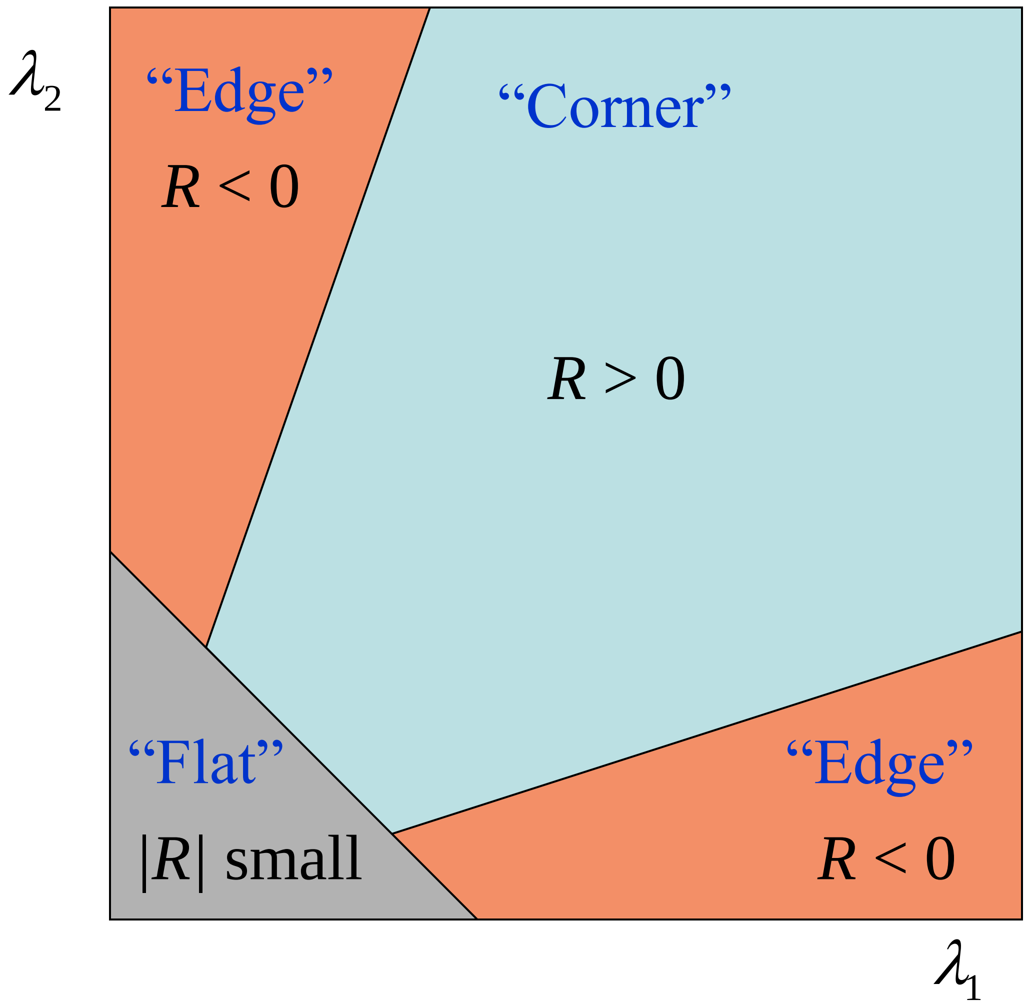
$$har=\det[\mu(\sigma_I,\sigma_D)]-\alpha[\operatorname{trace}(\mu(\sigma_I,\sigma_D))^2]=$$
$$g(I_x^2)g(I_y^2)-[g(I_xI_y)]^2-\alpha[g(I_x^2)+g(I_y^2)]^2$$

5. Non-maxima suppression

Slide: Derek Hoiem

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



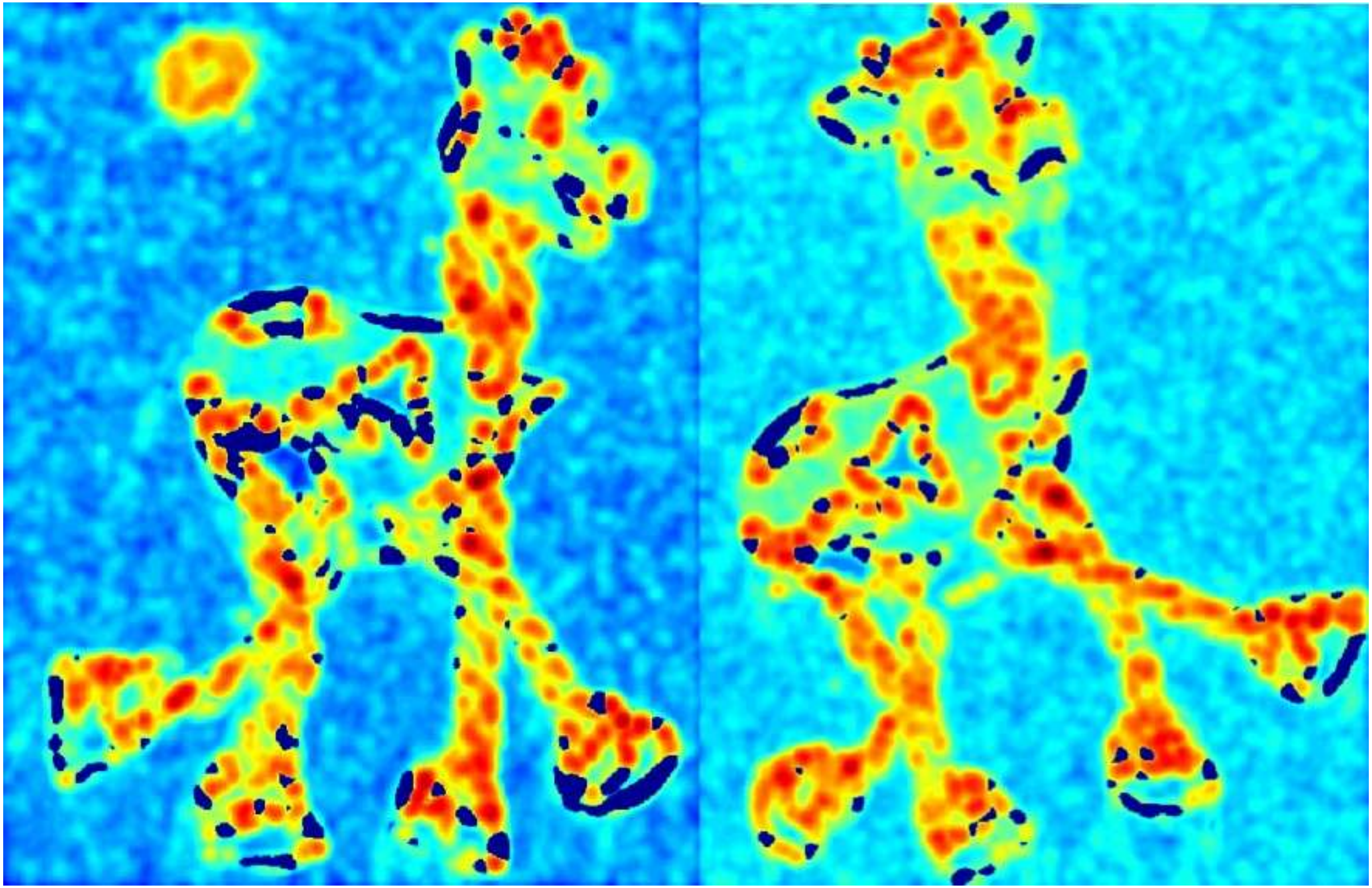
Harris Detector

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

Harris Detector: Workflow

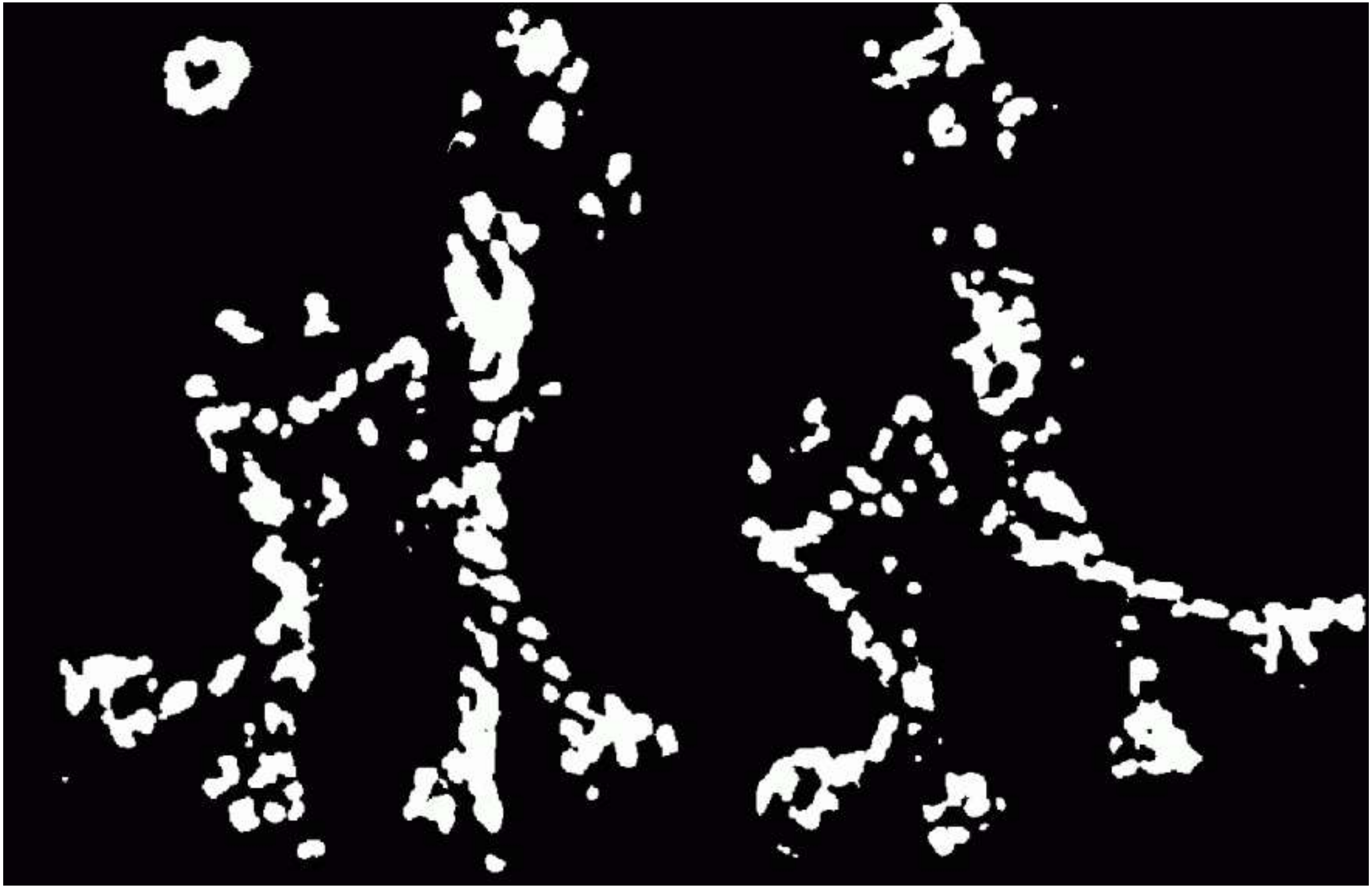
Harris Detector: Workflow

Compute corner response R



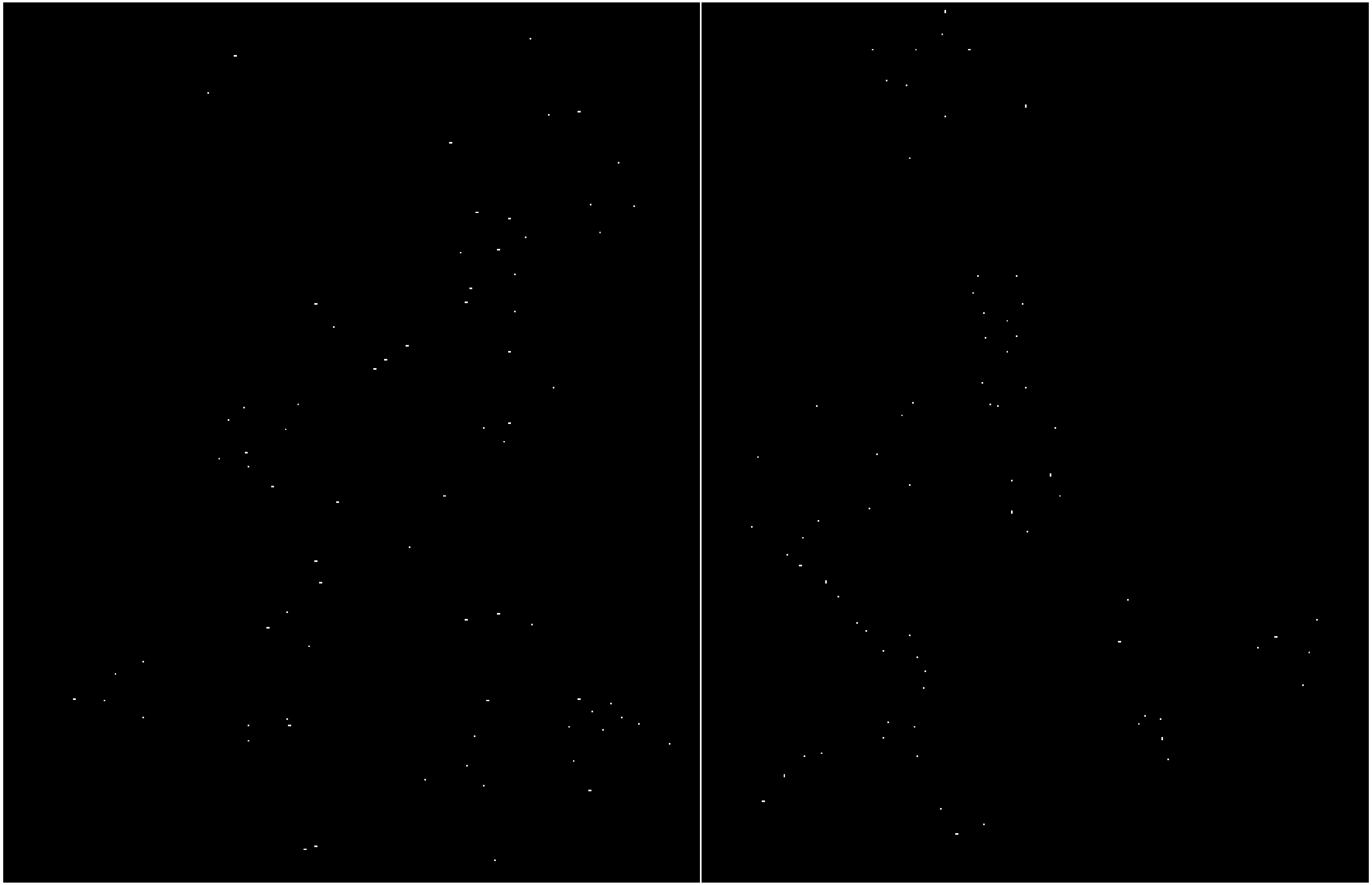
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

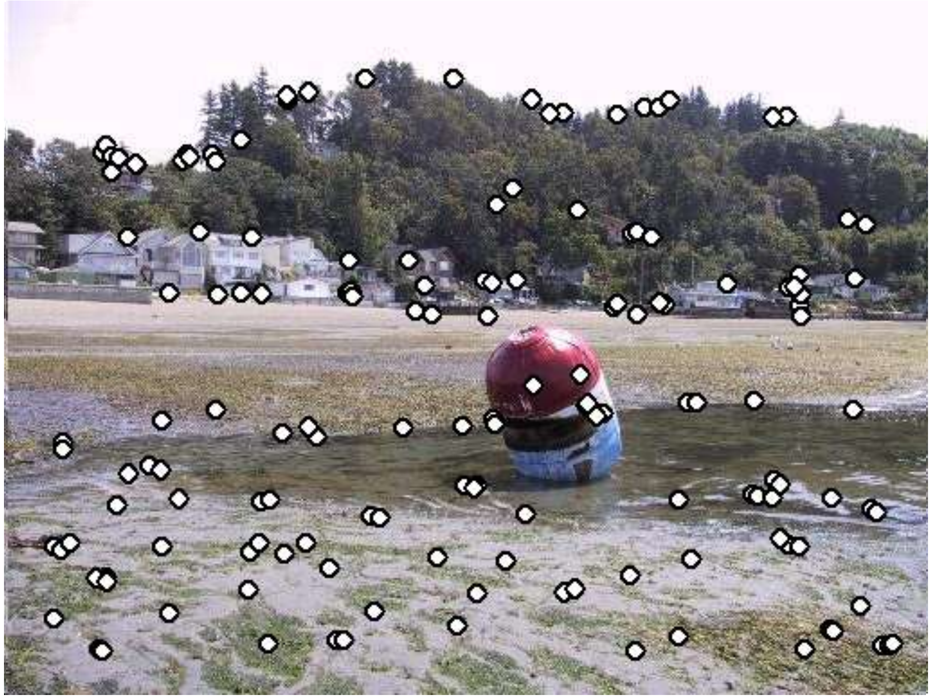
Take only the points of local maxima of R



Harris Detector: Workflow

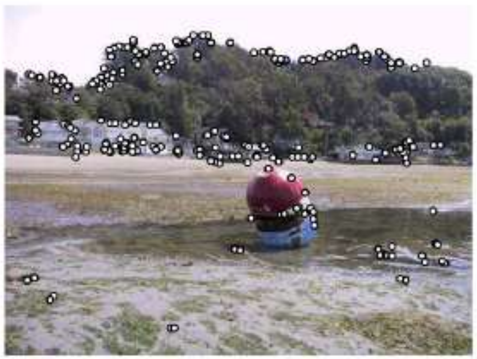
Feature selection

- Distribute points evenly over the image

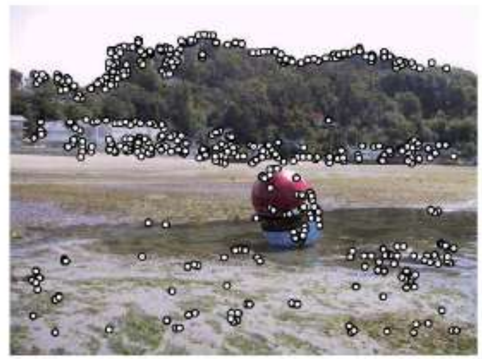


Adaptive Non-maximal Suppression

- Desired: Fixed # of features per image
  - Want evenly distributed spatially…
  - Sort ponts by non-maximal suppression radius [Brown, Szeliski, Winder, CVPR’05]



(a) Strongest 250



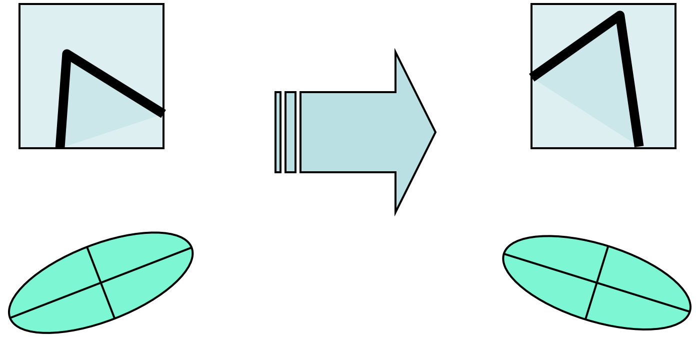
(b) Strongest 500

(c) ANMS 250, $r = 24$

(d) ANMS 500, $r = 16$

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 $I = I + b$

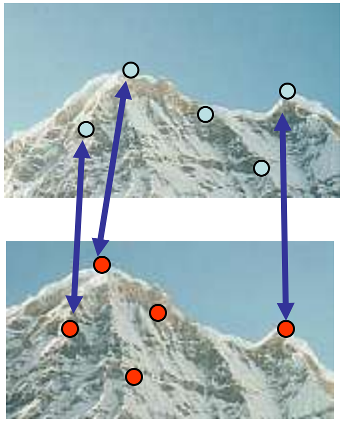
Intensity scale: $I = a\ I$

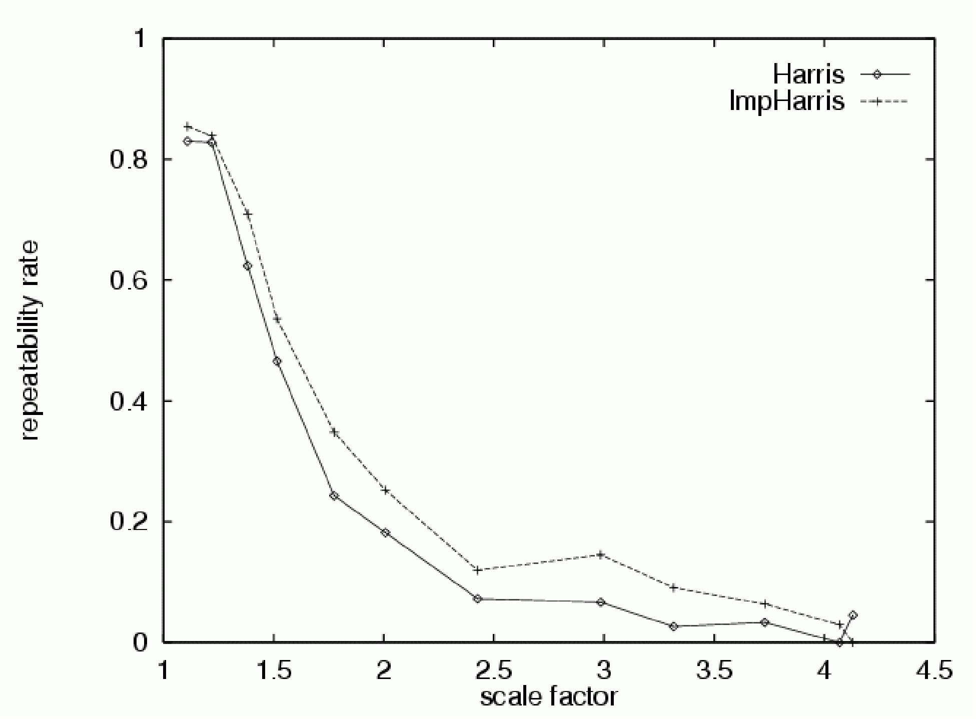
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability rate:

$$\frac{\text{\# correspondences}}{\text{\# possible correspondences}}$$



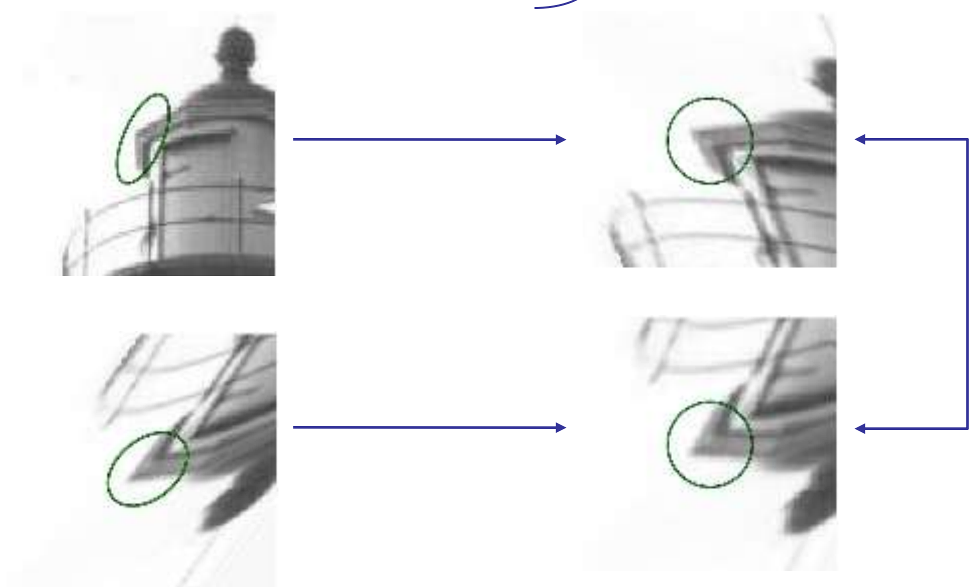


C.Schmid et.al. "Evaluation of Interest Point Detectors". IJCV 2000

Matching with Features

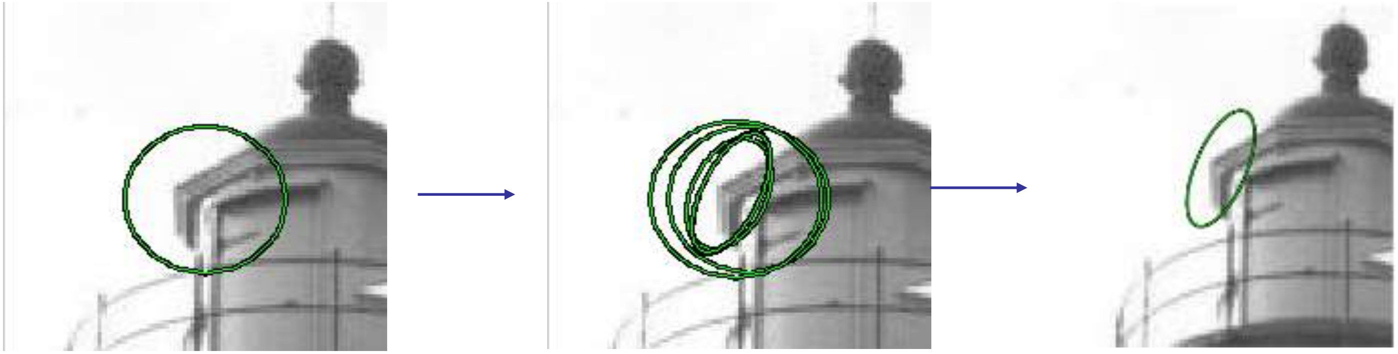
- Rotation
- Scaling
- Viewpoint

Affine invariance
!



Matching with Features

- Existing method by Mikolajczyk
  - Iterative affine invariant point detector
    - Multi-scale Harris corner detector
    - Laplacian characteristic scale selection
    - Second moment matrix shape determination



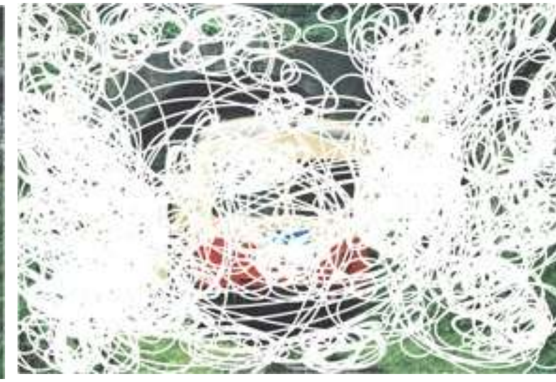
Initial region based on initial scale and location

Iteratively adjust scale, position and shape of region

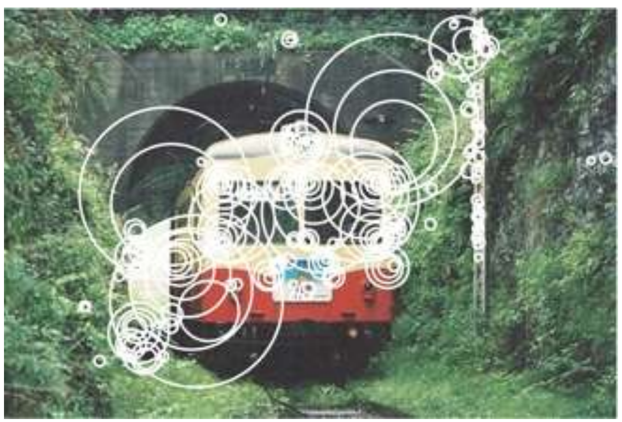
final region

Original Image

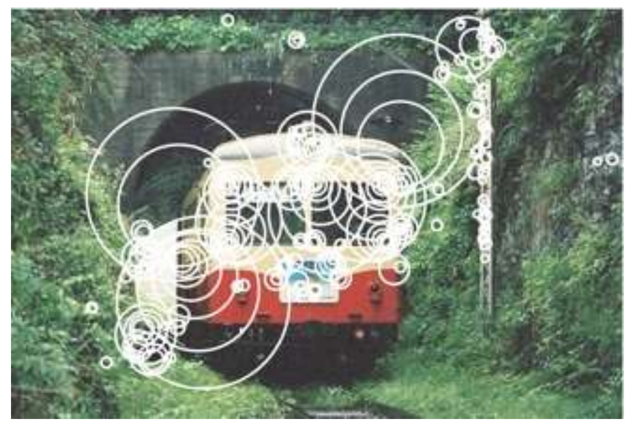
Harris Laplacian
impl. by Mikolajczyk (e.g. CVPR06)



Shape adapted Harris Laplacian
impl. by Mikolajczyk (ICCV07)



Color salient points
Quasi invariant HSI



Color salient points
Color boosted OCS

Most of the time, both color approaches agree on the most salient parts of an image.

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

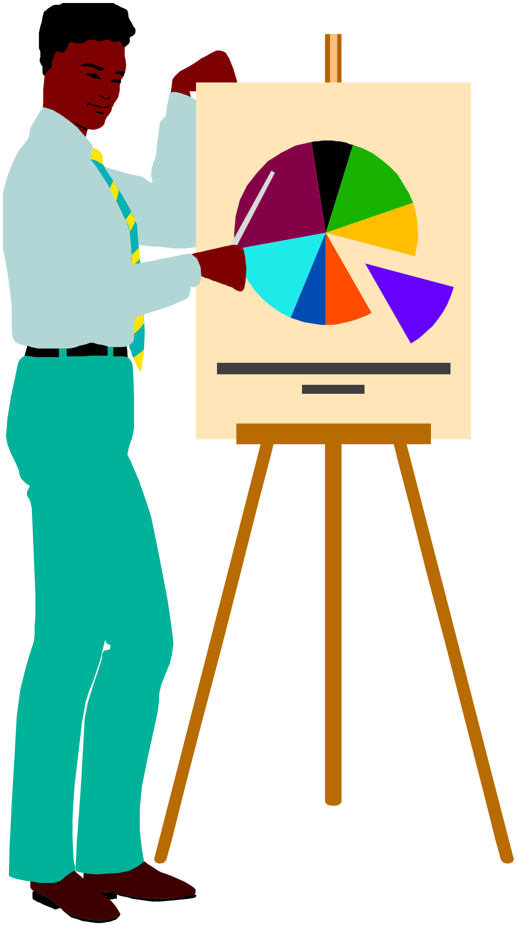
Tuytelaars Mikolajczyk 2008

Summary

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection

- Invariant descriptors
  - Rotation according to dominant gradient direction
  - Histograms for robustness to small shifts and translations (SIFT descriptor)

Local Features

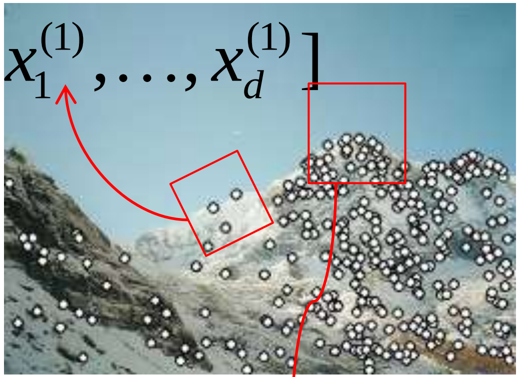


Local features: main components

1) **Detection:** Identify the interest points

2) **Description:** Extract vector feature descriptor surrounding each interest point.

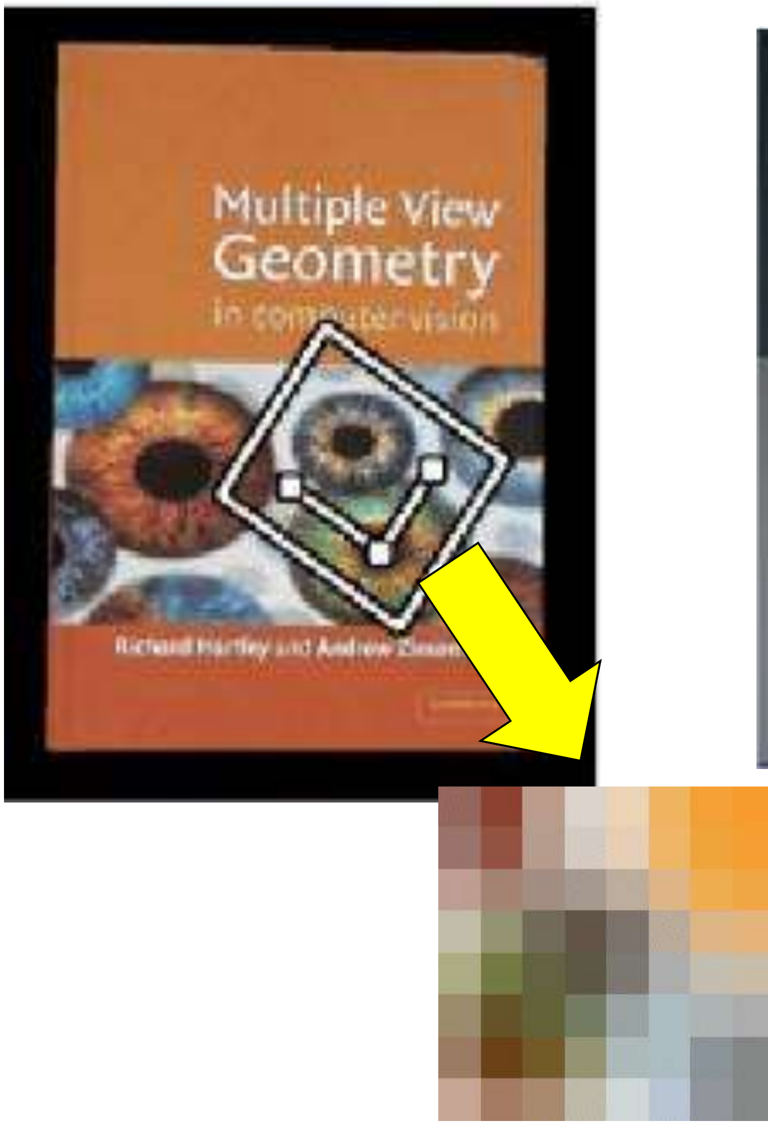
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

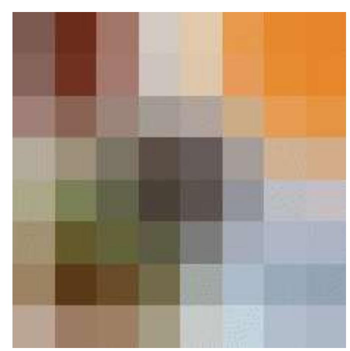


$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching:** Determine correspondence between descriptors in two views

Geometric transformations



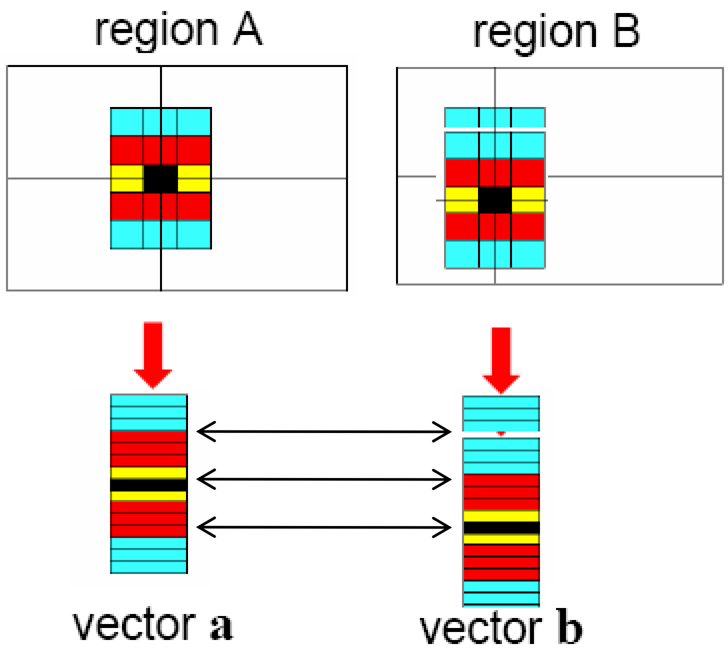


e.g. scale,
translation,
rotation

Photometric transformations

Figure from T. Tuytelaars ECCV 2006 tutorial

Raw patches as local descriptors

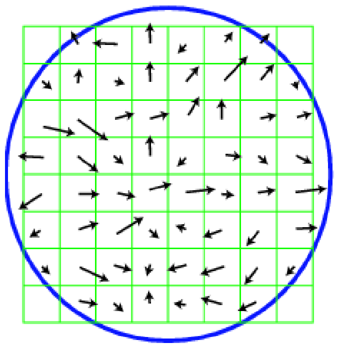


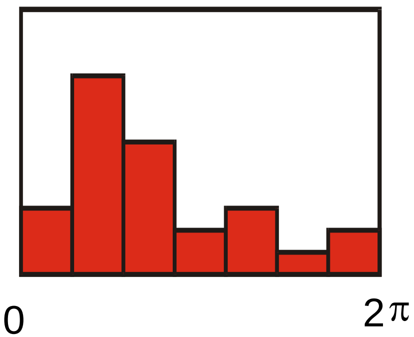
The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

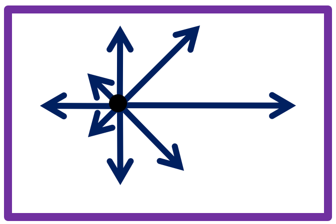
But this is very sensitive to even small shifts, rotations.

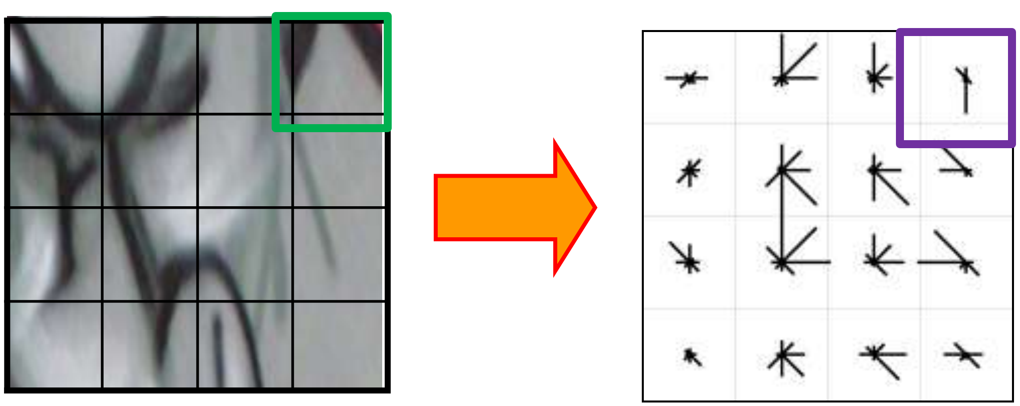
SIFT descriptor [Lowe 2004]

- Use histograms to bin pixels within sub-patches according to their orientation.





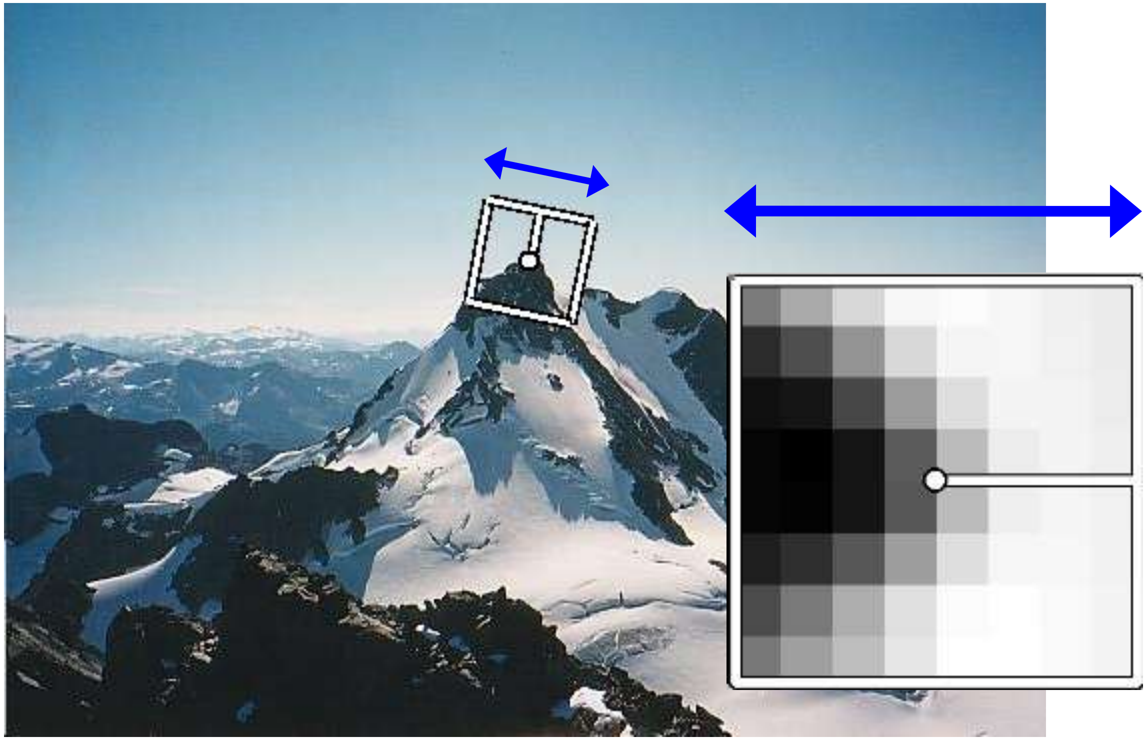




Why subpatches?

Why does SIFT have some illumination invariance?

Making descriptor rotation invariant



- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

Image from Matthew Brown

SIFT descriptor [Lowe 2004]

- 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

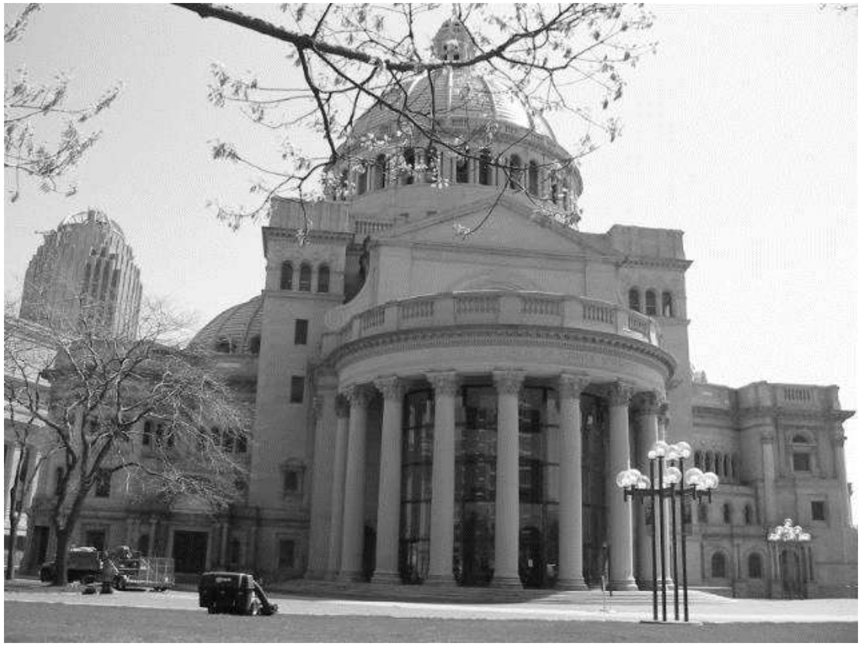
Steve Seitz

SIFT properties

- Invariant to
  - Scale
  - Rotation

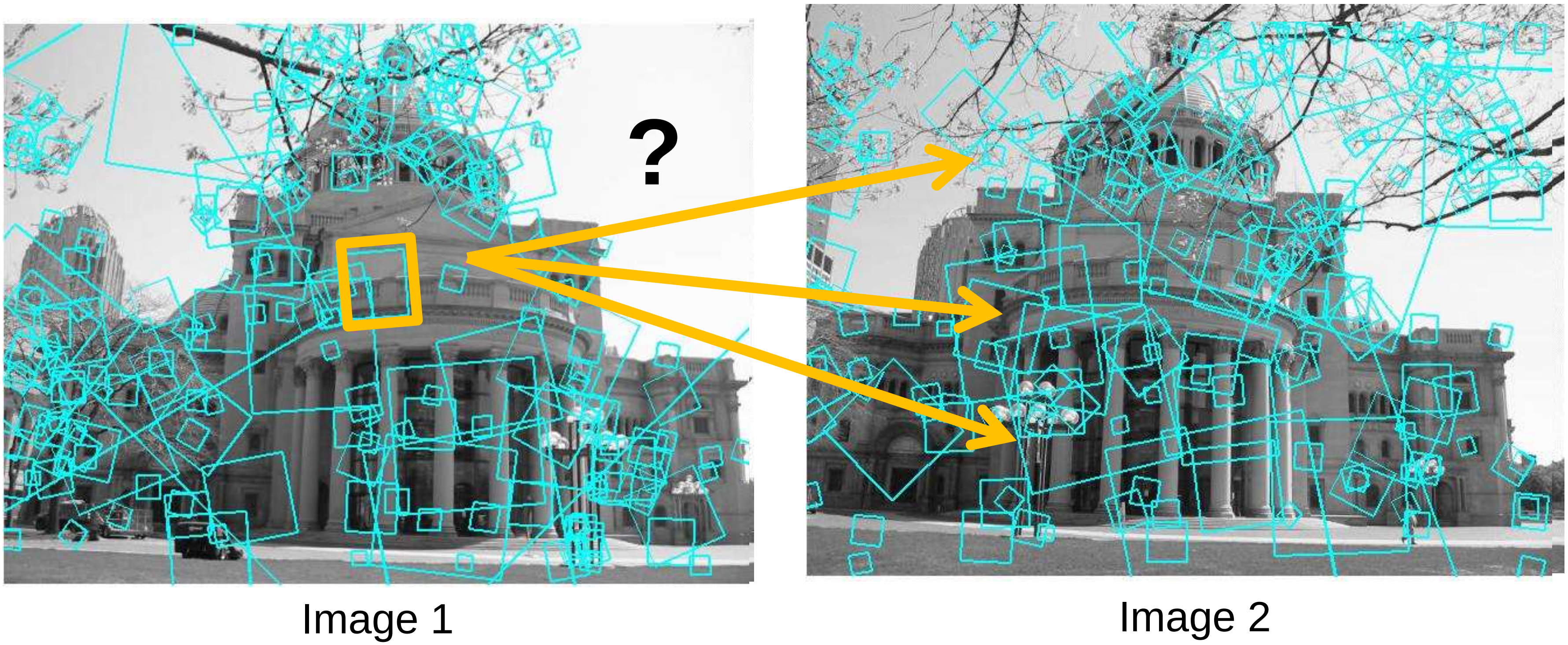
- Partially invariant to
  - Illumination changes
  - Camera viewpoint
  - Occlusion, clutter

Matching local features

Kristen Grauman

Matching local features



To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)

Simplest approach: compare them all, take the closest (or closest k, or within a thresholded distance)

Kristen Grauman

Ambiguous matches

Image 1

Image 2

At what SSD value do we have a good match?

To add robustness to matching, can consider **ratio** : distance to best match / distance to second best match

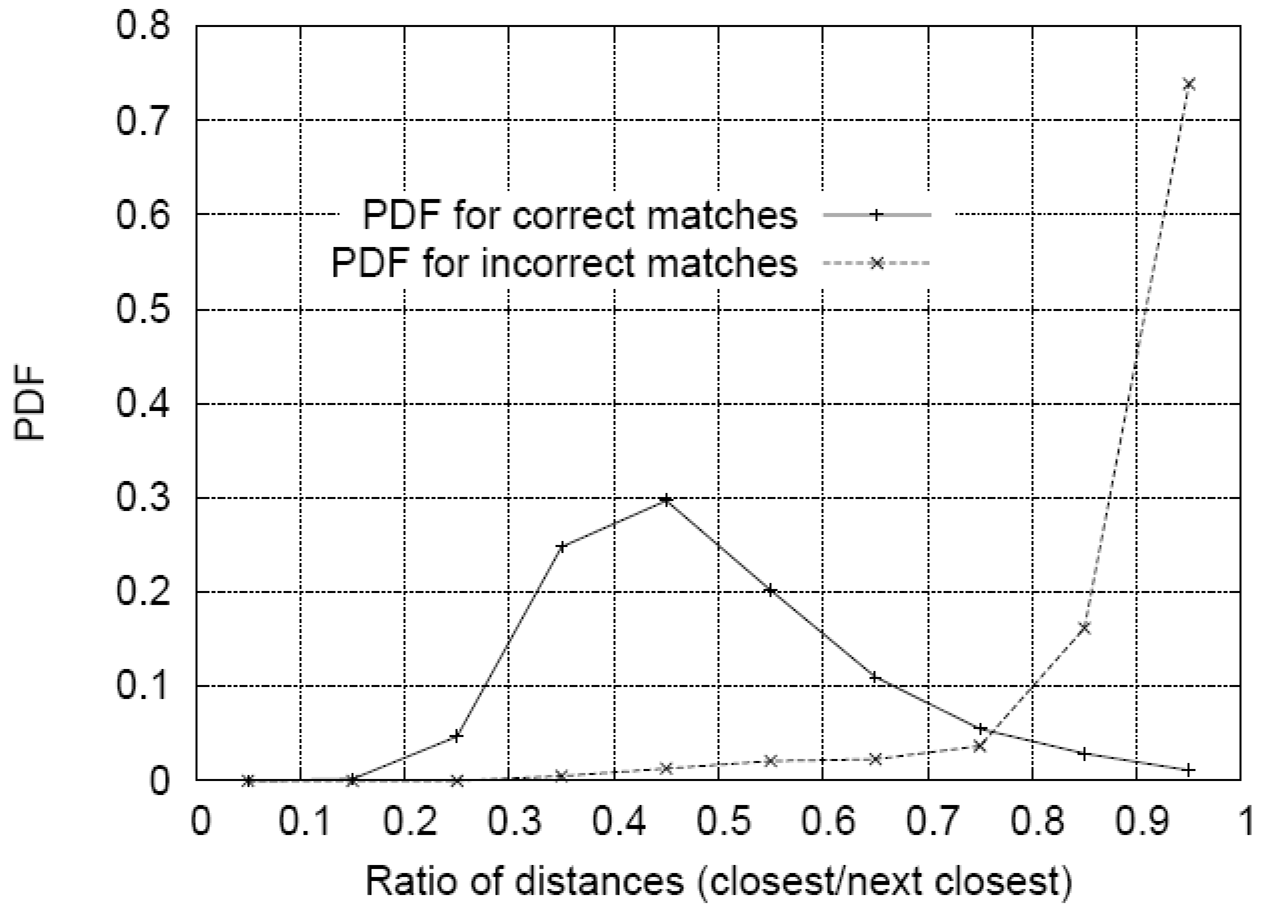
If low, first match looks good.

If high, could be ambiguous match.

Kristen Grauman

Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)



Lowe IJCV 2004

Recognition of specific objects, scenes

Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

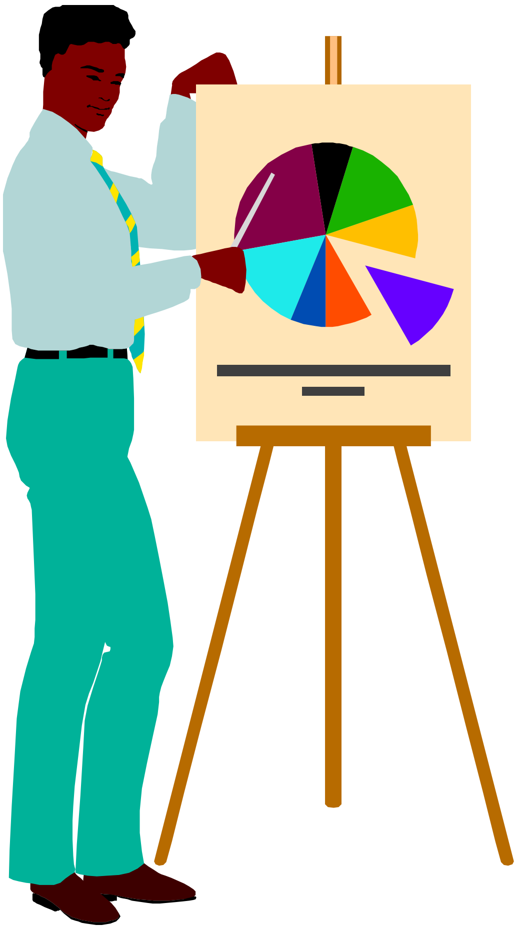
Lowe 2002

Kristen Grauman

Today's class

Feature Extraction

Tracking

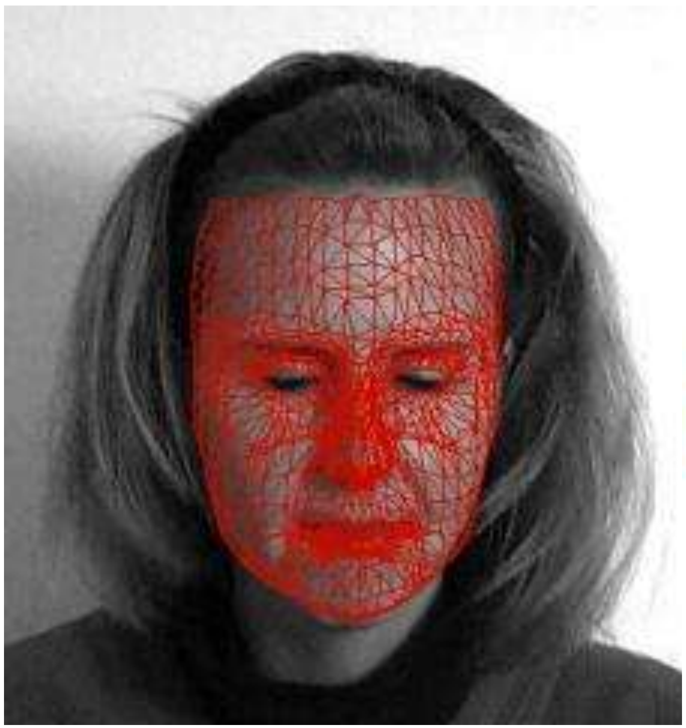


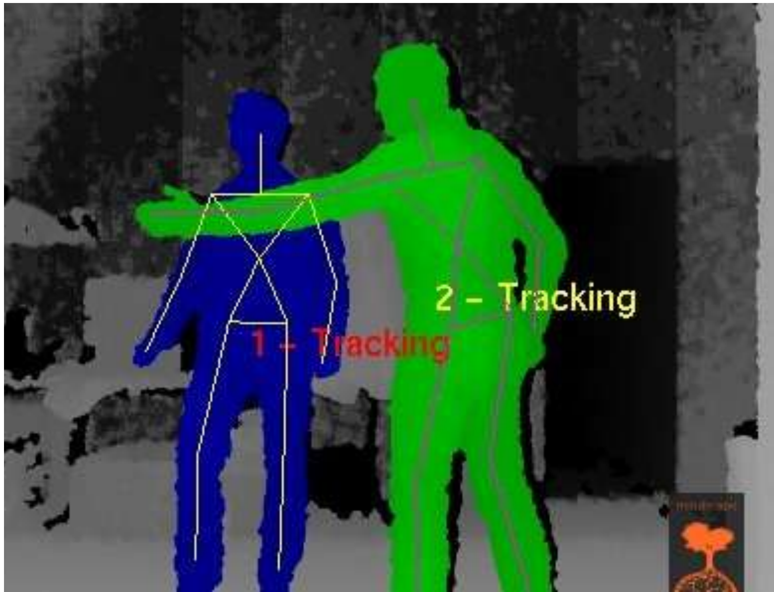
Visual tracking

What is visual tracking?

Visual tracking

What are the applications of tracking ?

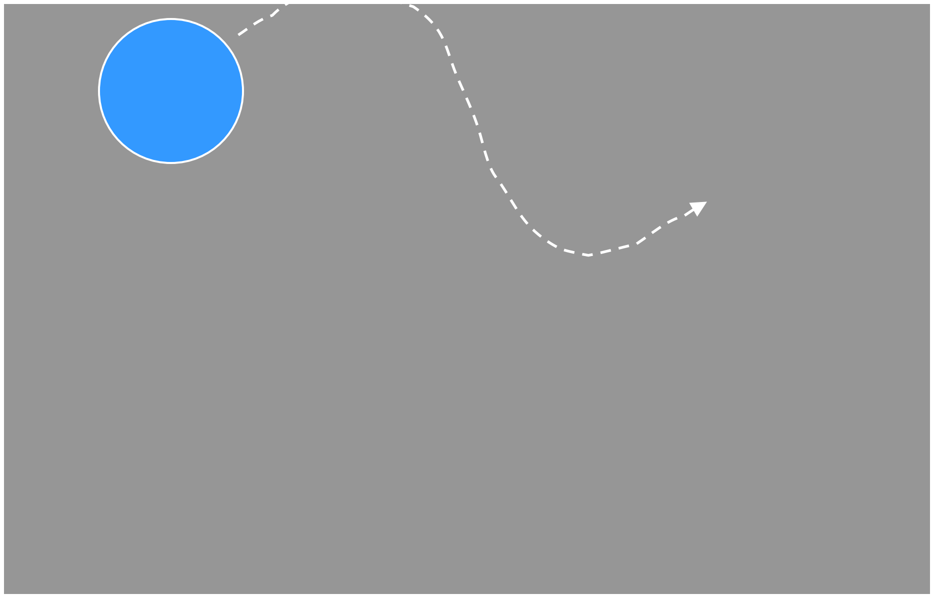




Visual tracking

Tracking can be very easy!

- Tracking can be very easy if both the target and the background are uniform in color.



Visual tracking

What makes tracking so hard?

- **Background clutter**: the presence of other objects or non-informative patterns in the image complicates the detection of the right object.
- A **dynamic background**: moving camera, viewpoint change.
- **Illumination change**: change in direction or intensity of light source, shadow…
- **Non-rigid, unhomogeneous, fast objects:** articulated objects, appearance change, object speed / frame rate
- **Multiple objects tracking**
- **Occlusion**: the target disappears partially or completely behind another object for a while.

- **Setting and application**

Visual tracking

Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Visual tracking

Motion segmentation

Motion segmentation methods aims at separating foreground from background:

- Detections without background model are fast but noisy.

    Image difference, Optical flow

Visual tracking

Motion segmentation

- Pixel-based background model

  Image model, statistical model (Gaussian mixture model), predictive model

Frame

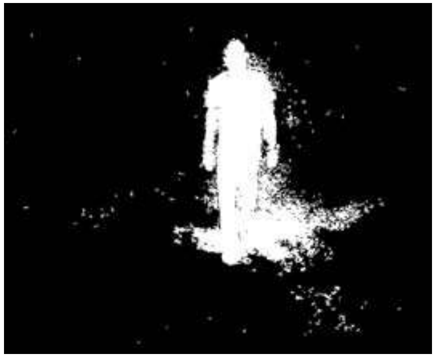
RGB

YUV

Visual tracking

Motion segmentation






- Local background model

  Region detection, texture on block, improve pixel-based model

- Global background model

  Model mixture, eigen background

Visual tracking

Tracking methods

Region-based tracking

Connected regions are detected and matched across frames

–Blobs composing human body.

–Various levels tracking: regions, person, group.

Method that handles various object tracking. However occlusions, objects interactions are not handled.

Contour-based tracking

Active contour uses the objects outline.

The representation is simple, but the precision is limited to the contour and initialization is challenging.

Object Tracking: A Survey*, ACM Computing Surveys, Yilmaz A., Javed, Shah, 2006.*

Visual tracking

Tracking methods

Feature-based tracking

Extract feature to describe and match regions

–Global features: centroid, perimeter, surface area, color moments

–Local features: interest points, corner, curve/lines segments

Multiple object can be tracked and partial occlusion can be handled.

These features can be easily combined.

Model-based tracking

Match a projected object model to image data.

A priori knowledge about the object is required.

Human models include stick figure, volumetric models, skeleton, etc.

Need to build the model. High computational costs.

Visual tracking

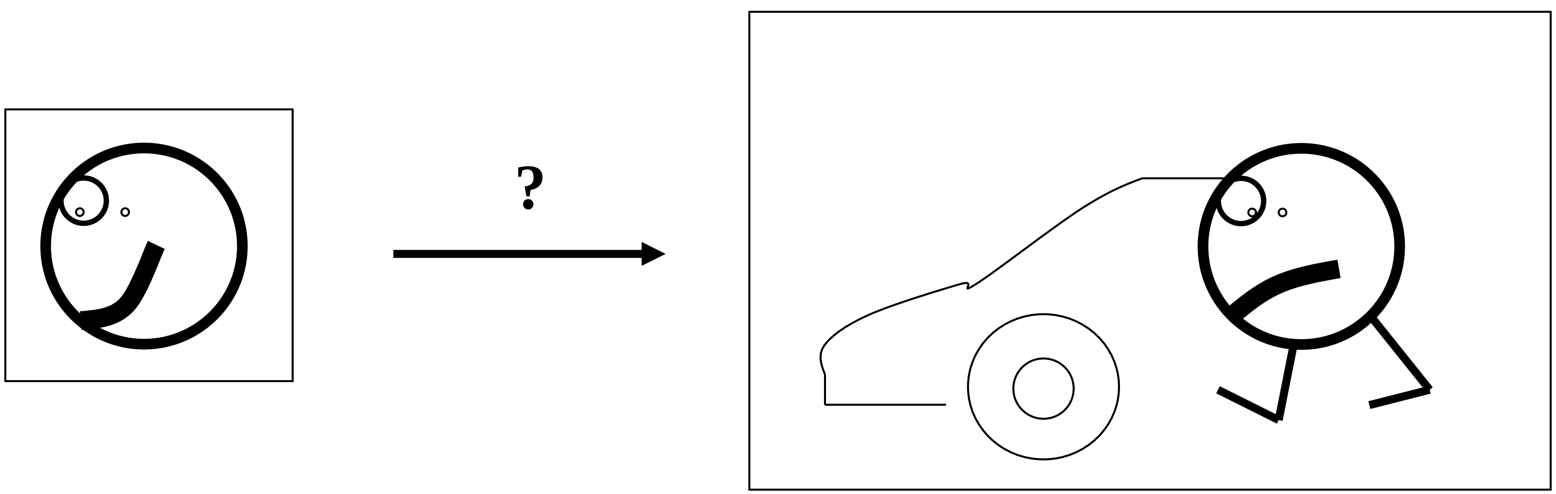
Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Visual tracking

Template based tracking

- Tracking consists in searching for the target object in a frame by comparing with a **template** image.
- We assume that the template is fixed and given in advance.



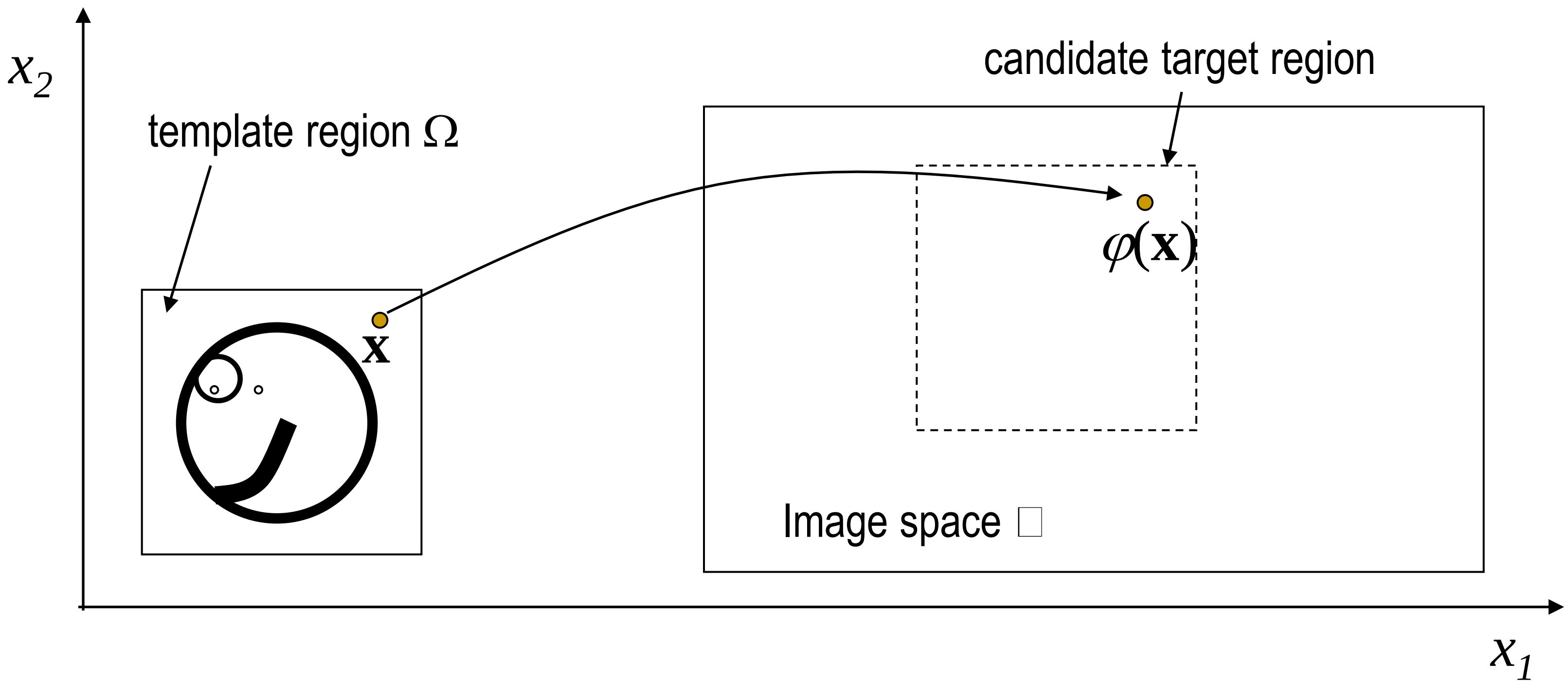
Template image
$T(\mathbf{x})$

$I(\mathbf{x},\mathbf{t})$

Visual tracking

Template to target transformation

- The template is mapped into a candidate target region the image using a transformation of coordinates: $\varphi(\mathbf{x})$: $\Omega \rightarrow$ □. This transformation depends on a parameter vector $\mathbf{y}$. Different candidate regions correspond to different values of $\mathbf{y}$. So we write $\varphi(\mathbf{x};\, \mathbf{y})$.



Visual tracking

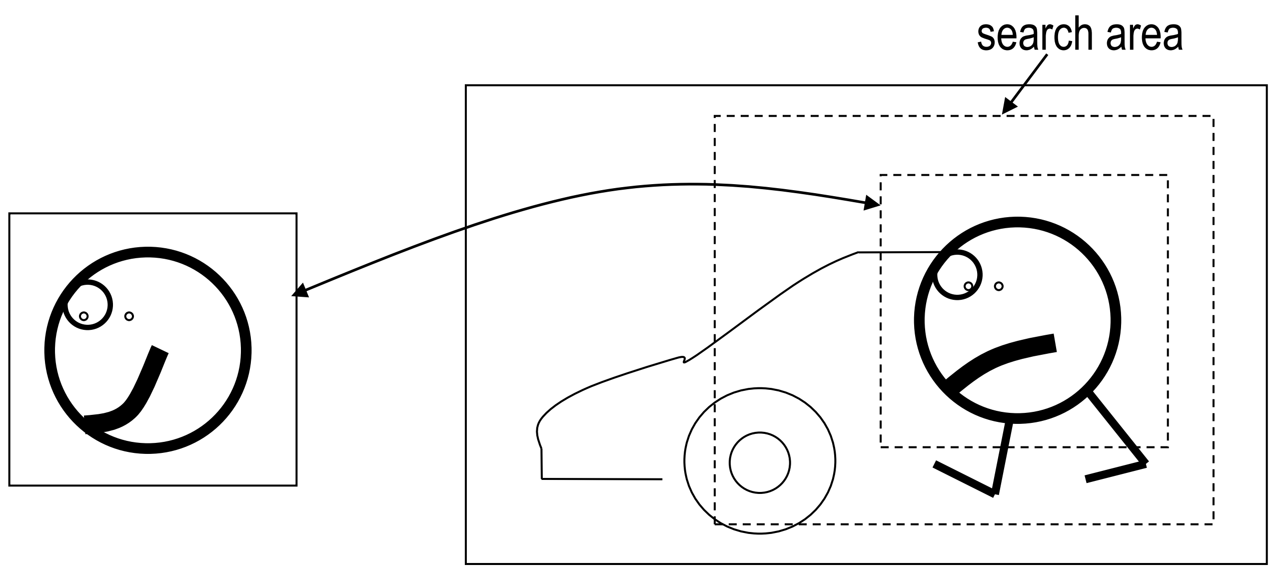
Motion models

- The type of transformation φ specifies the type of object motion that the tracker is able to deal with.
  - Translation: $\varphi(\mathbf{x}; \mathbf{y}) = \mathbf{x} + \mathbf{y}$
  - Rotation: $$\varphi_1 = x_1 \cos y - x_2 \sin y$$ $$\varphi_2 = x_1 \sin y + x_2 \cos y$$
  - Scaling: $$\varphi_1 = yx_1$$ $$\varphi_2 = yx_2$$
  - Affine: $$\varphi_1 = y_1 + y_2 x_1 + y_3 x_2$$ $$\varphi_2 = y_4 + y_5 x_1 + y_6 x_2$$

Visual tracking

Search

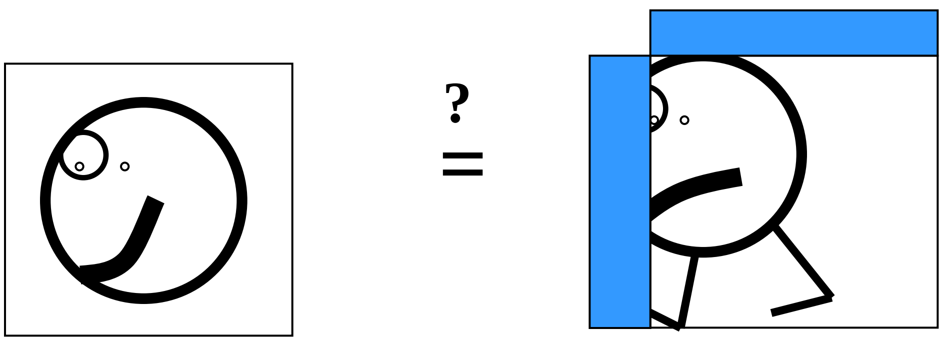
- Align the template with every possible candidate region in the image, and find the most similar candidate according to a **similarity measure**.
- We search the target only in an area around the previous position exploiting general knowledge that the object won't have moved far.



Visual tracking

Similarity measure

- We need a measure of how similar (or far apart) the template and the candidate are.



- The similarity measure can be based on:
  - pixelwise intensity (color) difference: **SSD** and **correlation** trackers,
  - histogram difference: **mean-shift** tracker.

Visual tracking

SSD and correlation

- SSD is short for sum-of-squared-difference:

$$D(\mathbf{y}) = \sum_{\mathbf{x}\in\Omega} [I(\mathbf{x}+\mathbf{y}) - T(\mathbf{x})]^2 \rightarrow \min_{\mathbf{y}}$$

- A simpler similarity measure is the (unnormalized) cross-correlation:

$$C(\mathbf{y}) = \sum_{\mathbf{x}\in\Omega} I(\mathbf{x}+\mathbf{y})T(\mathbf{x}) \rightarrow \max_{\mathbf{y}}$$

Visual tracking

Exhaustive search

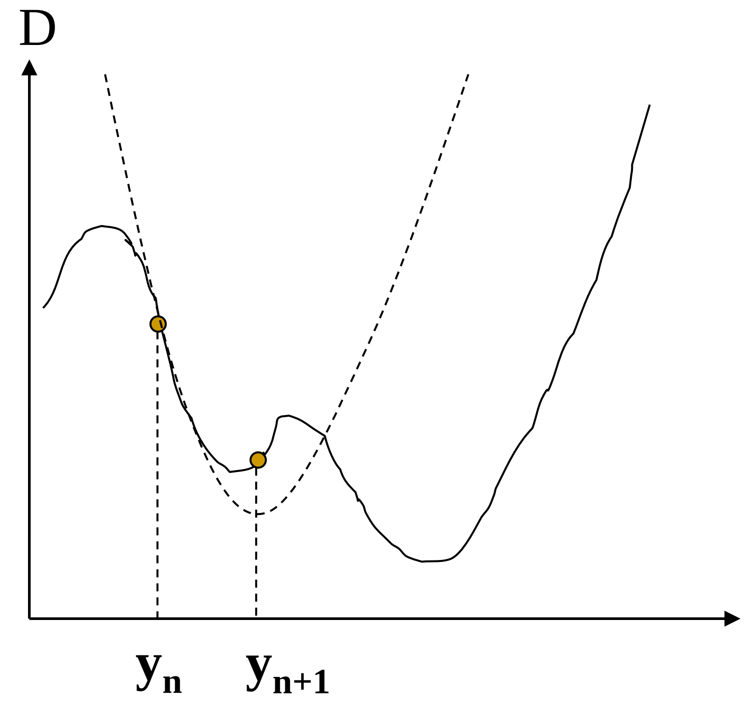
- Calculate SSD for every $\mathbf{y}$ in a search window and choose the position with the least SSD.
- Strengths: robustness and simplicity in implementation.
- Weaknesses:
  - Computations could be time-consuming in case of a large search window.
  - Only suitable for translation.

Visual tracking

Gradient descent

- Iteratively update the solution estimate using Newton's method:

$$\mathbf{y}_{n+1} = \mathbf{y}_n - \alpha \nabla D$$



- Efficient computation. Able to cope with rotation.
- Finds a local minimum only.

Visual tracking

Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Visual tracking

Mean-shift tracking

- Target detection is performed by matching **weighted histograms**.
- Very fast in comparison with SSD or correlation trackers.

***Real time tracking of Non-Rigid Objects using Mean Shift**, CVPR, Comaniciu et al. 2000.*

Visual tracking

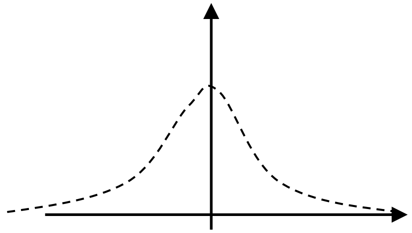
The mean-shift algorithm

- The mean-shift algorithm finds a local maximum of a density function of the form:

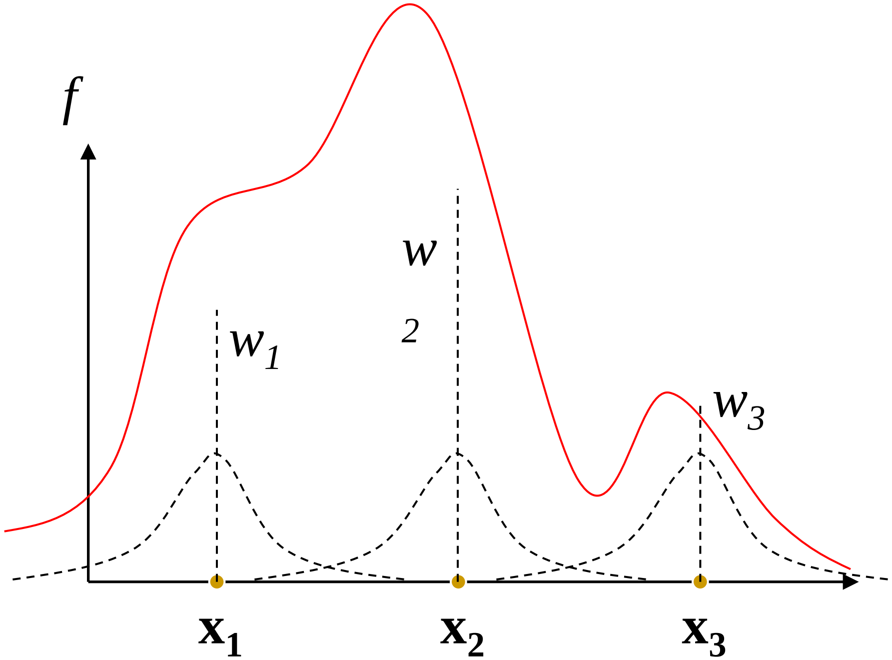
$$f(\mathbf{y}) = \sum_i w_i K\left(\frac{|\mathbf{y}-\mathbf{x}_i|^2}{\sigma}\right)$$

- where K is the local kernel.

Gaussian kernel:



$$K(|\mathbf{x}|^2) = (2\pi)^{-d/2}\exp\left(-|\mathbf{x}|^2/2\right)$$

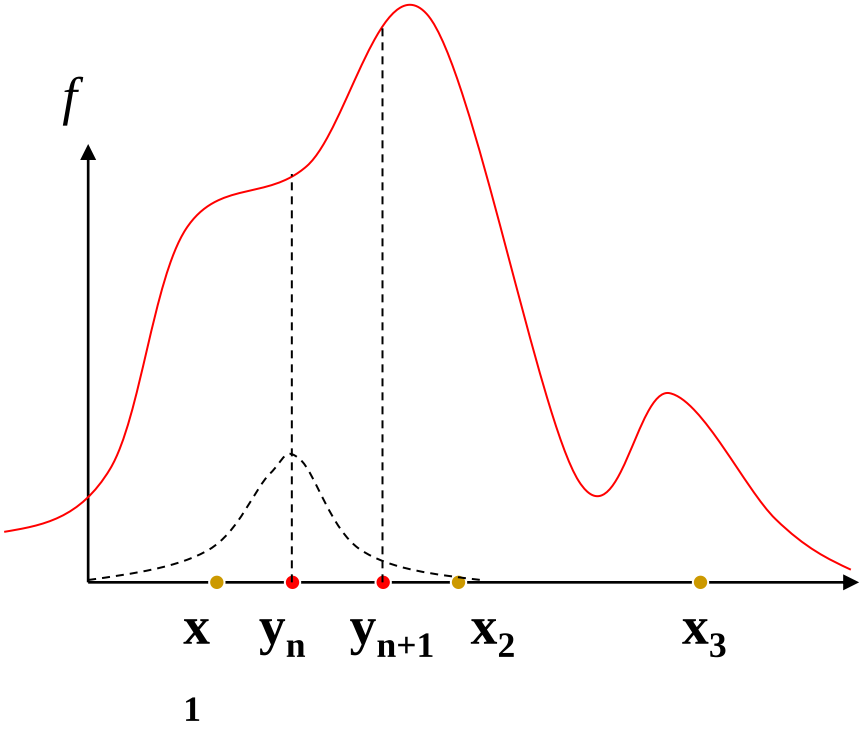


Visual tracking

The mean-shift algorithm

- A local maximum will be found by successively shifting $\mathbf{y}$ to a weighted mean of $\mathbf{x}_\mathrm{i}$ computed with the *derivative kernel* K':

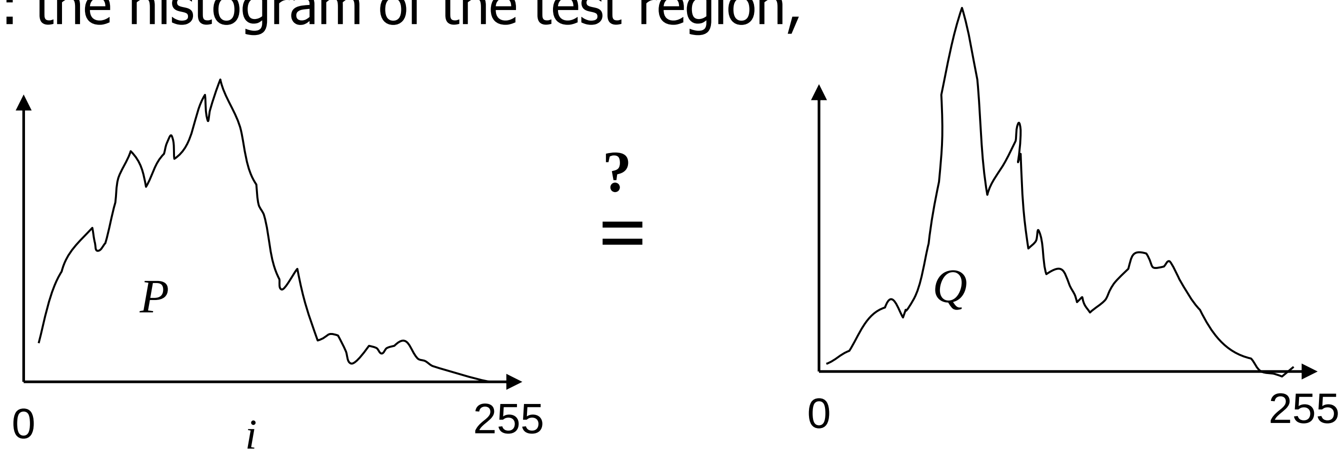
$$\mathbf{y}_{n+1} = \frac{\sum_{i=1}^{N} \mathbf{x}_i w_i K_i^{'}\left(\frac{|\mathbf{y}_n - \mathbf{x}|^2}{\sigma}\right)}{\sum_{i=1}^{N} w_i K_i^{'}\left(\frac{|\mathbf{y}_n - \mathbf{x}|^2}{\sigma}\right)}$$



Visual tracking

Similarity measure

- $P(i)$: the template histogram,
- $Q(i;\mathbf{y})$: the histogram of the test region,



- *The Bhattacharyya coefficient* can measure the similarity between two distributions:

$$r(\mathbf{y}) = r(P, Q(\mathbf{y})) = \sum_{i=0}^{255} \sqrt{P(i)Q(i;\mathbf{y})} \rightarrow \max_{\mathbf{y}}$$

Visual tracking

Mean-shift tracking

- Tracking under occlusions

- Updating process

Visual tracking

Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Visual tracking

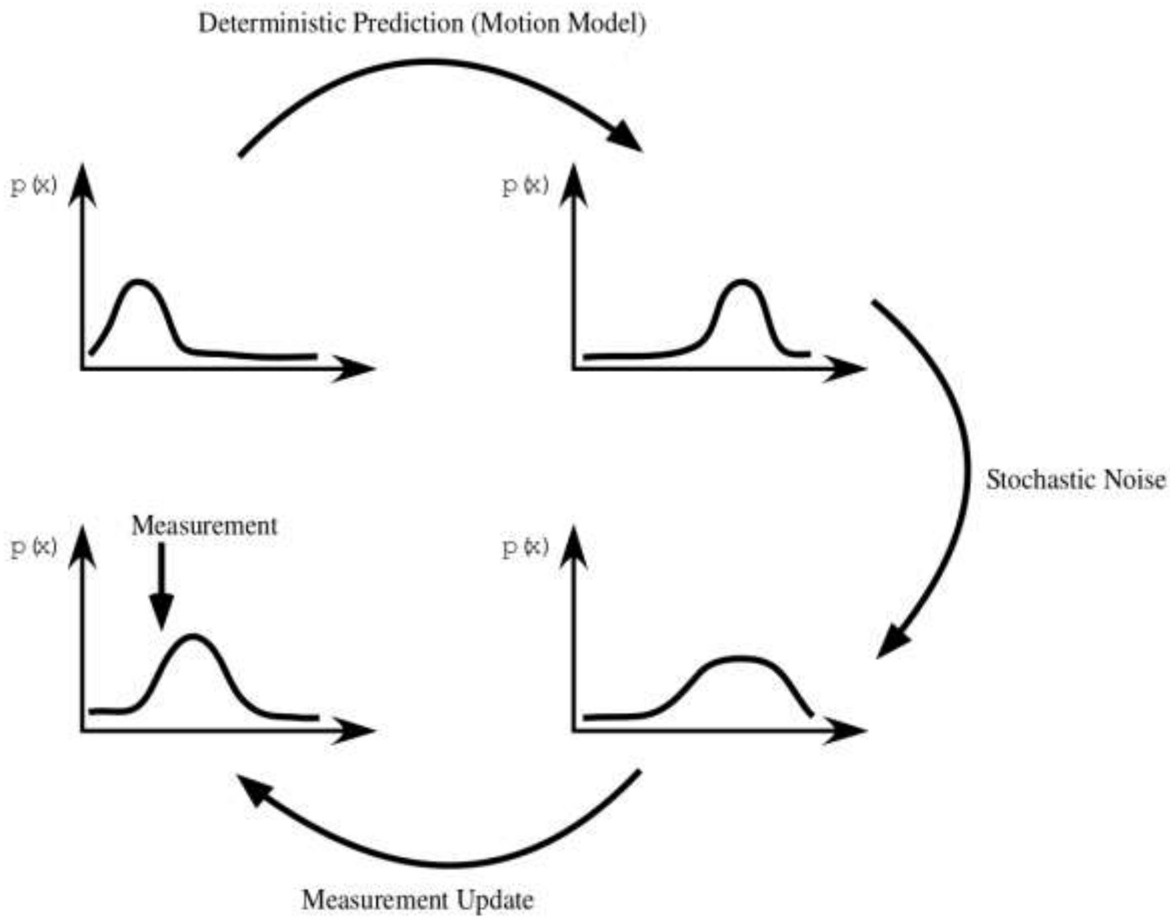
Kalman Filtering

- Provides an optimal estimation of the state of system based on measurements
- Assumptions: linear relationship between variables, distributions are Gaussian (mean, variance).
- 3 steps:
  - Deterministic prediction
  - Stochastic diffusion
  - Estimate refinement via a new measurement

A Review of Visual Tracking*, tech. report, Cannons, 2008*

Visual tracking

Kalman Filtering



Visual tracking

Kalman Filtering

- x_t state of the system at time t and z_t measurements

$$\hat{x}_t^- = A\hat{x}_{t-1} + B\hat{u}_{t-1}$$

- A : system equation (link state at time t-1 to state at time t)
- B : control input: u is omitted
- Simple example: constant velocity

$$x_t = \{p_x, p_y, v_x, v_y\}^T$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Visual tracking

Kalman Filtering

- $\hat{P}_t^-$ covariance estimate $\hat{P}_t^- = AP_{t-1}A^T + Q$
- Q : covariance of the noise associated with the prediction process

- Posterior estimate $\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-)$
- With $K_t = P_t^- H^T (HP_t^- H^T + R)^{-1}$
- *R* : noise associated with the measurement process
- *H* : maps state vectors into measurements vector (Identity in simple cases)
- Posterior estimate $P_t = (I - K_t H)P_t^-$

Visual tracking

Kalman Filtering

- K_t Kalman Gain: determines how much the prediction is trusted against the measurements

- Kalman advantages: combines multiple sources of information in an optimal manner, recursiveness (efficient computation).

- Kalman disadvantages: can not model all systems (linear), requires Gaussian models (problem with cluttered: find one optimal solution)

Visual tracking

Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Visual tracking

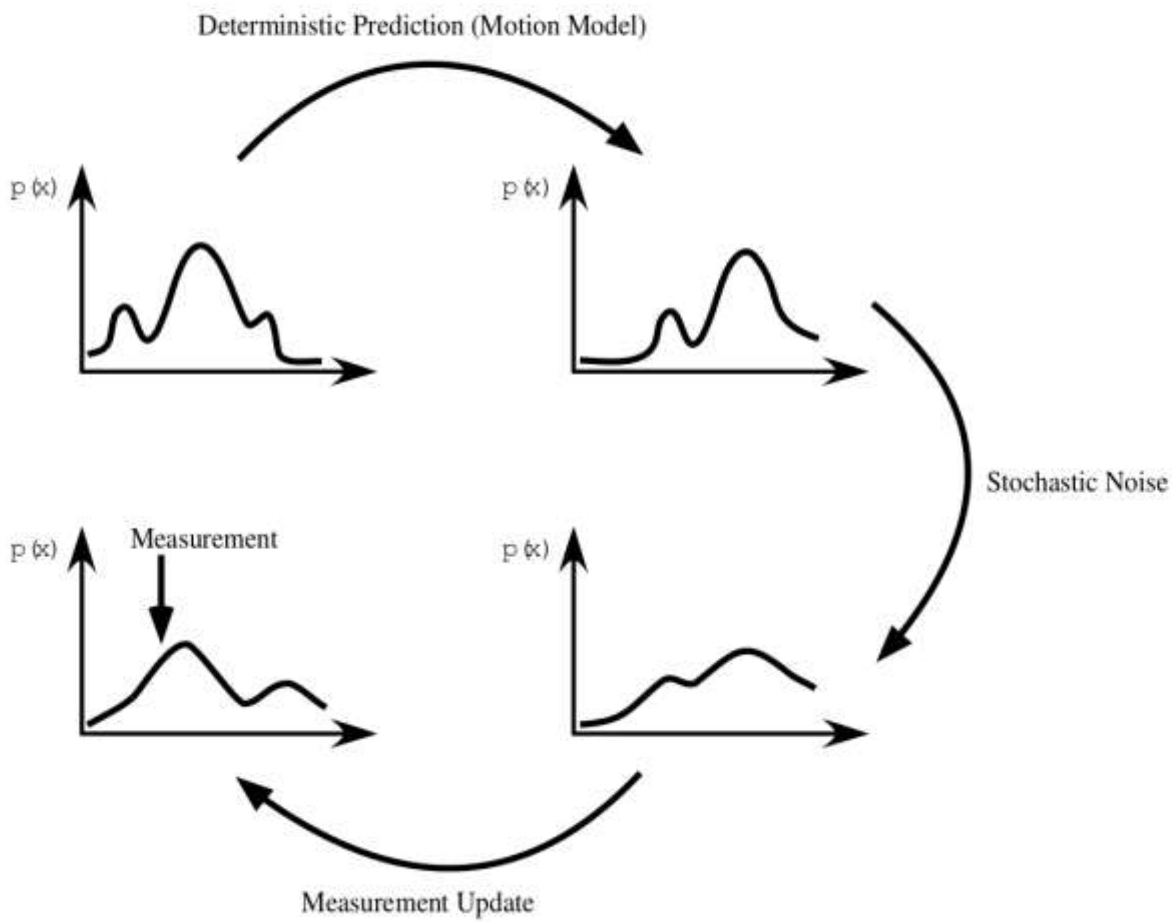
Particle Filtering

- Origin comes from monte carlo: use random sampling to find an approximated solution
- Condensation algorithm: no need to be linear, handle multimodal distribution
- Unlike Kalman samples have unspecified shape

- 3 steps:
  - Deterministic prediction
  - Stochastic diffusion
  - Correction via measurements updates

Visual tracking

Particle Filtering



Visual tracking

Particle Filtering

- The goal is to compute $p(x_t \mid z_t)$ probability distribution that describes the system state, given observed measurements

$$p(x_t \mid z_t) = k \;\; p(z_t \mid x_t)\, p(x_t)$$

- k constant. A set of sample $\{s_t^{(1)} \ldots s_t^{(N)}\}$ is drawn from the prior distribution $p(x_t)$

- Then samples are weighted

$$\pi_t^n = \frac{p(z_t \mid x_t = s_t^{(n)})}{\sum_{j=1}^{N} p(z_t \mid x_t = s_t^{(j)})}$$

- Larger N = better precision = more computation

Visual tracking

Particle Filtering

- Step1: sample from approximated posterior $\{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}), n = 1, \ldots, N\}$

  Deterministic drift

- Step 2: Add noise to the modified samples to obtain an estimate of

$$p(x_t \mid z_{t-1})$$

- Step 3: Determines the samples weights (Bhattacharrya: distance between target and candidates)

Visual tracking

Particle Filtering

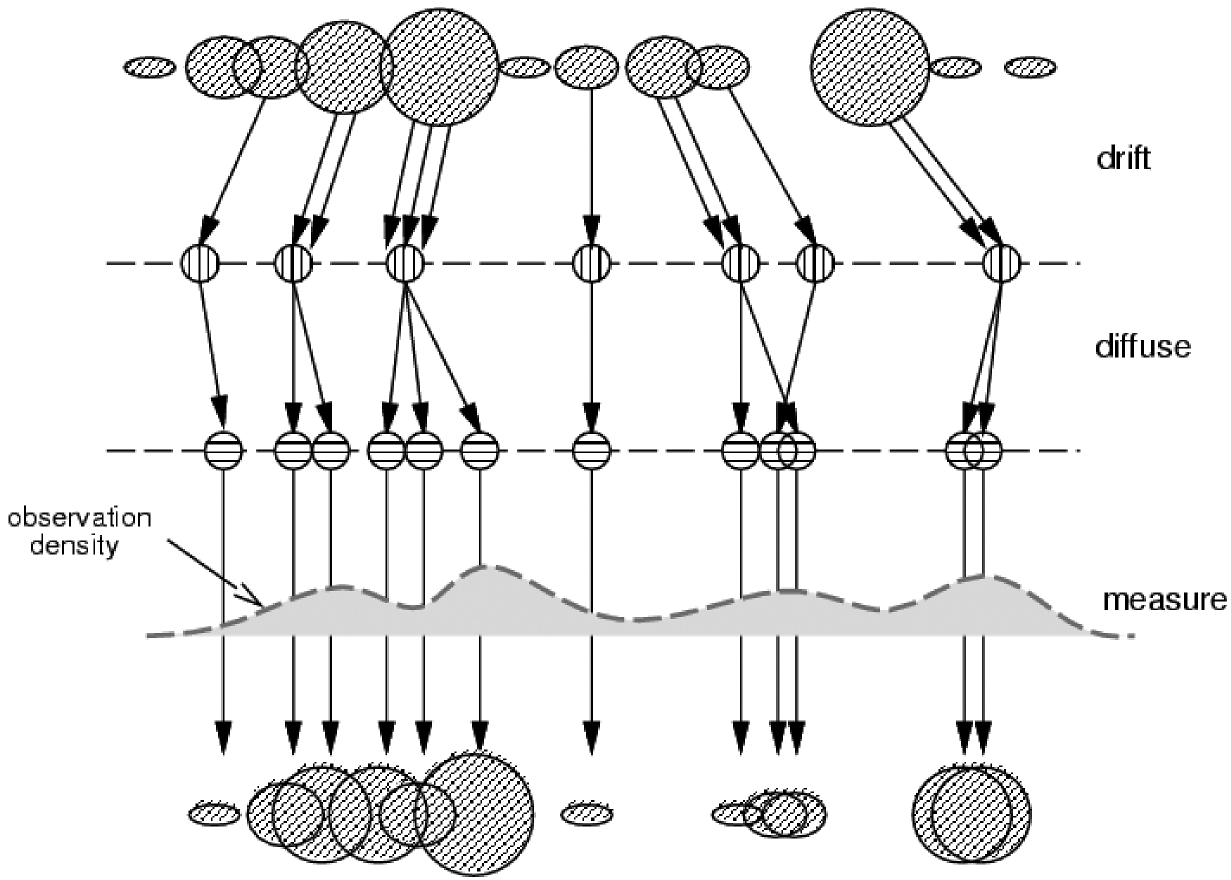
- Advantages over Kalman:
  - Can track through clutter
  - No need to calculate covariance estimates
  - Real-time depending on N

Visual tracking

Standard tracking algorithms

- **Motion segmentation based tracking**
- **Template tracking**
- **Mean-shift tracking**
- **Kalman filter**
- **Particle filtering**

Particle filtering



- Start with weighted samples from previous time step
- Sample and shift according to dynamics model
- Spread due to randomness; this is predicted density $p(x_t|y_{t-1})$
- Weight the samples according to observation density
- Arrive at corrected density estimate $p(x_t|y_t)$