DIGITAL IMAGE PROCESSING

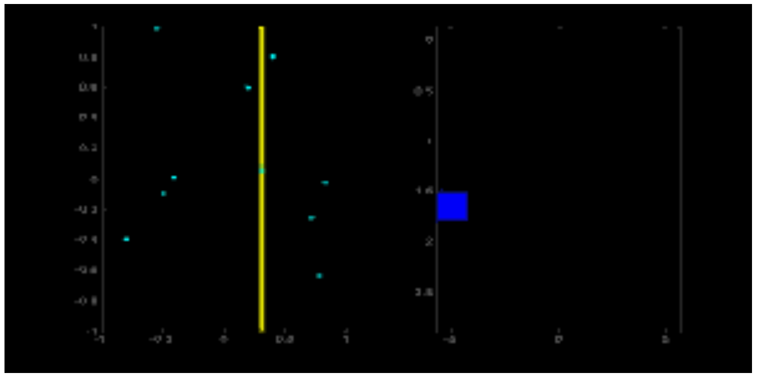
Lecture 7

Hough transform, descriptors

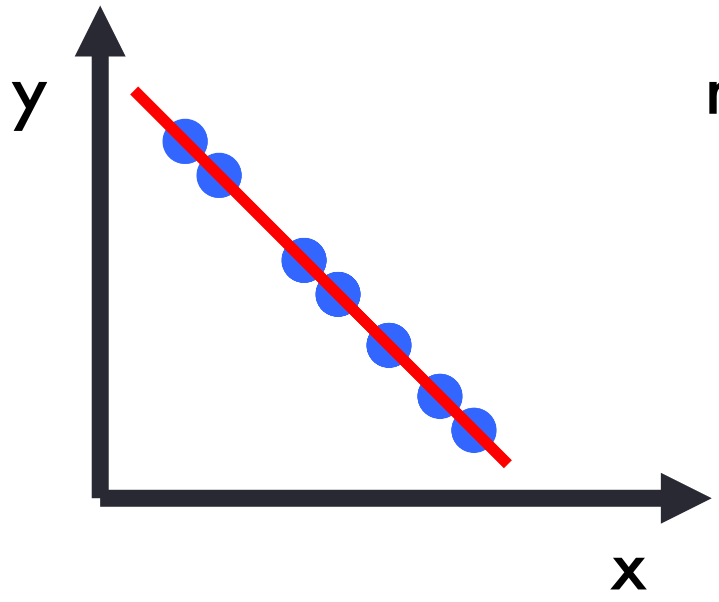
Tammy Riklin Raviv

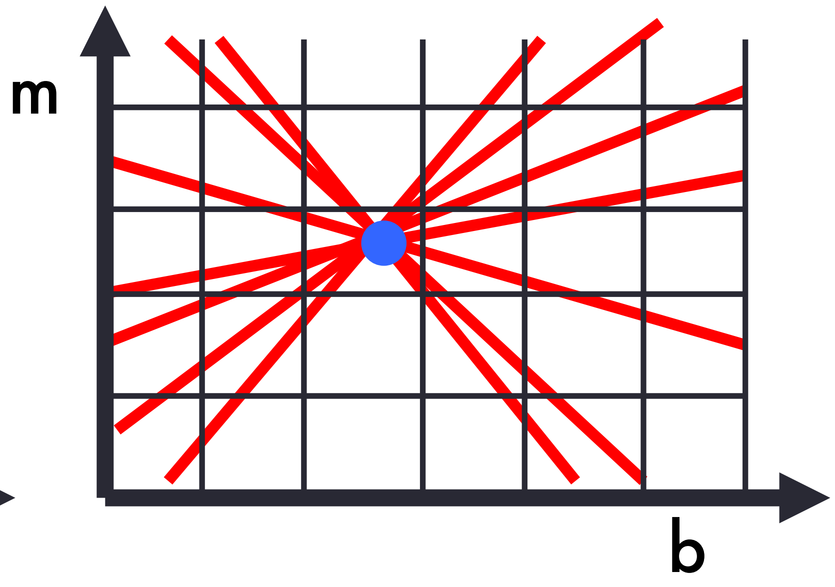
Electrical and Computer Engineering

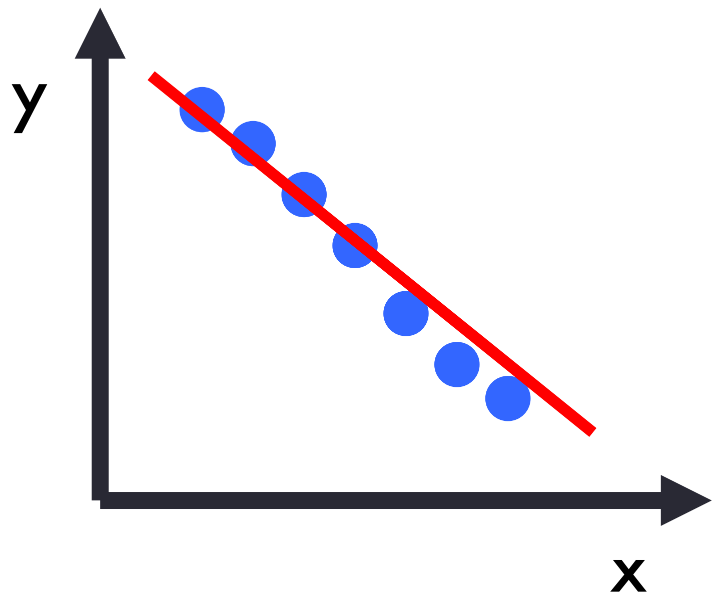
Ben-Gurion University of the Negev

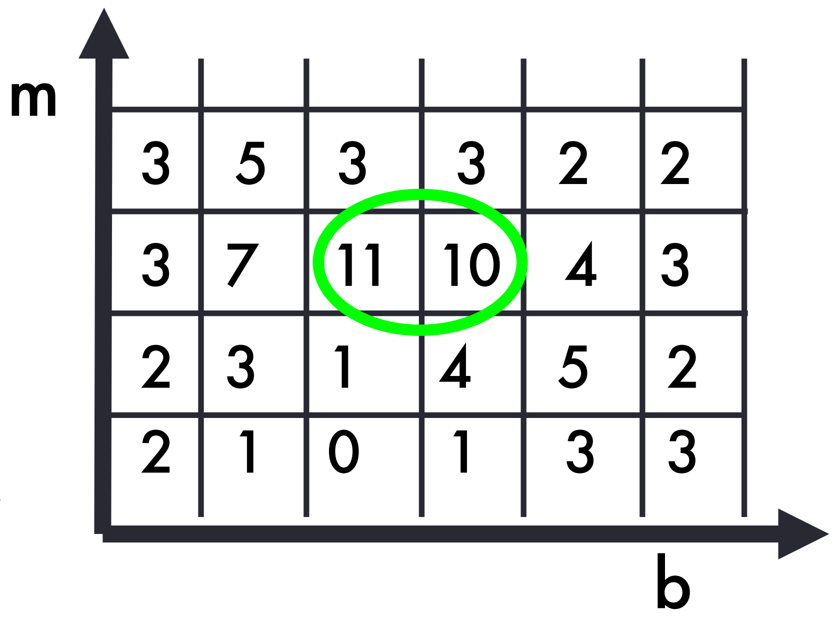


Hough transform









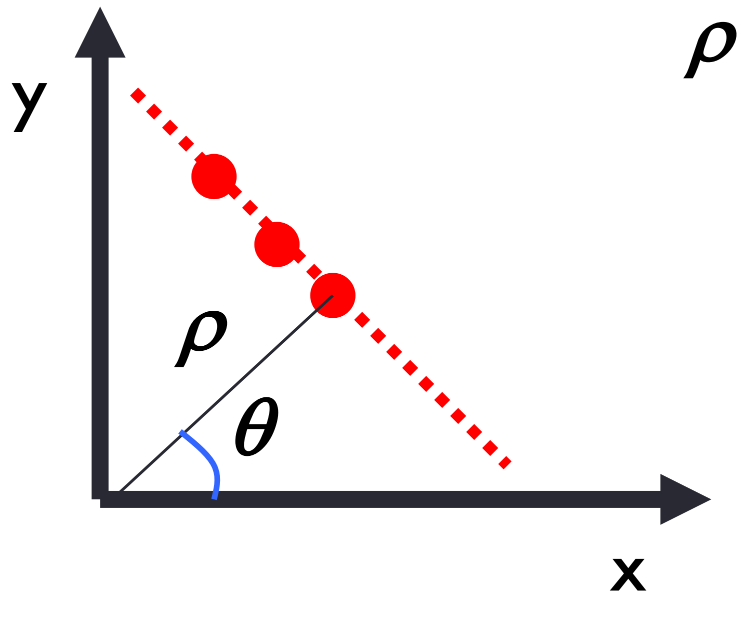
Slide from S. Savarese

Hough transform

Issues:

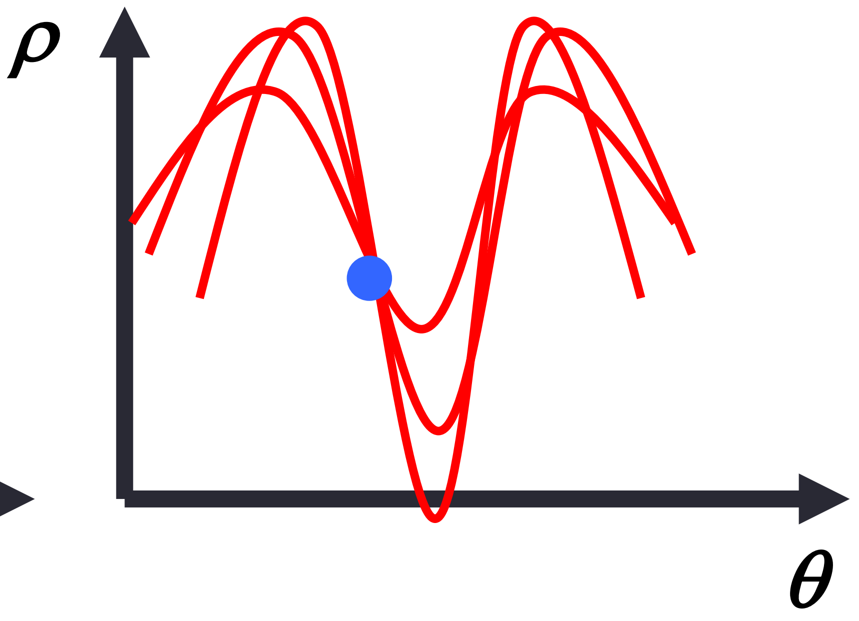
- Parameter space [m,b] is unbounded.
- Vertical lines have infinite gradient.

Use a polar representation for the parameter space



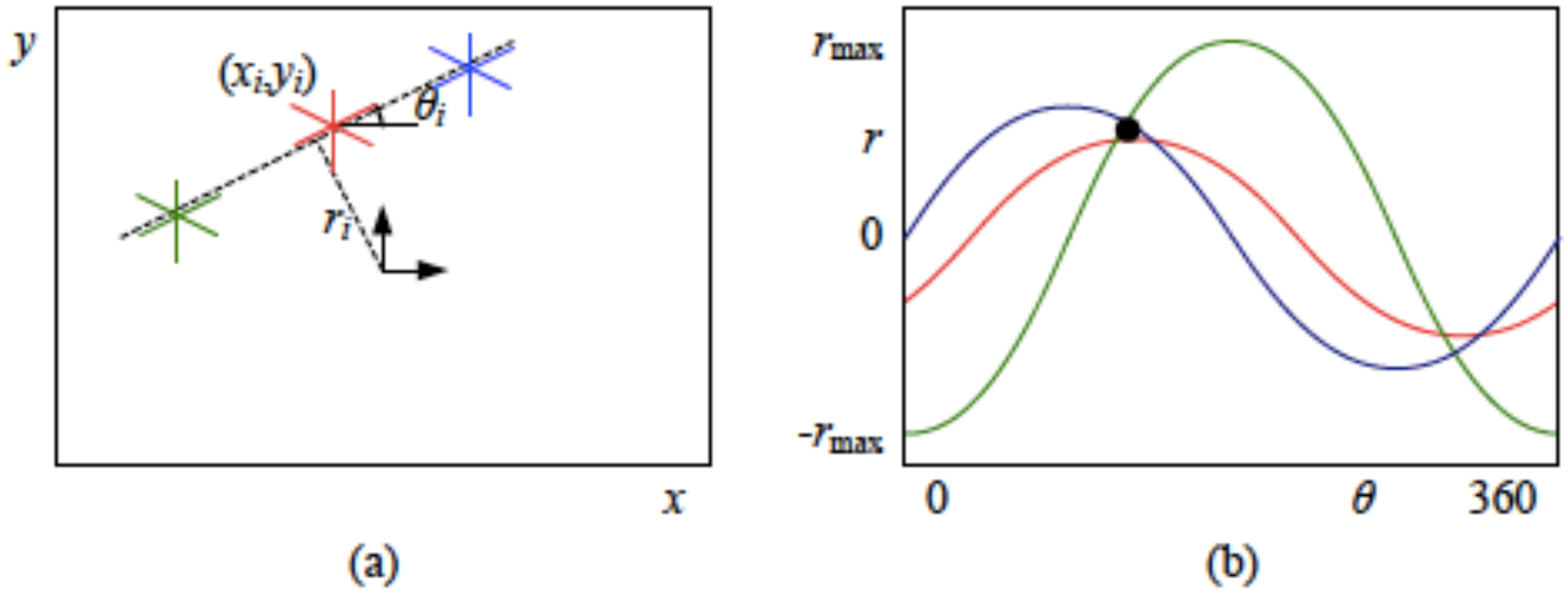
$$x \cos\theta + y \sin\theta = \rho$$

Hough space



Slide from S. Savarese

Hough Transform



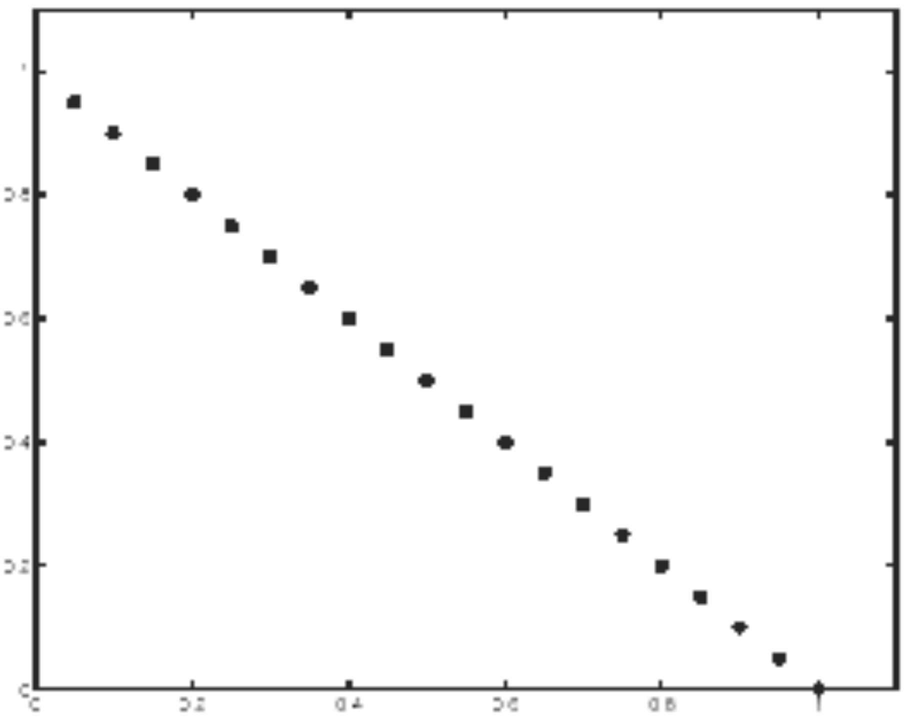
Each point votes for a complete family of potential lines:

$$r_i(\theta) = x_i \cos\theta + y_i \sin\theta$$

Each pencil of lines sweeps out a sinusoid in (r, θ)

Their intersection provides the desired line equation.

Hough transform - experiments



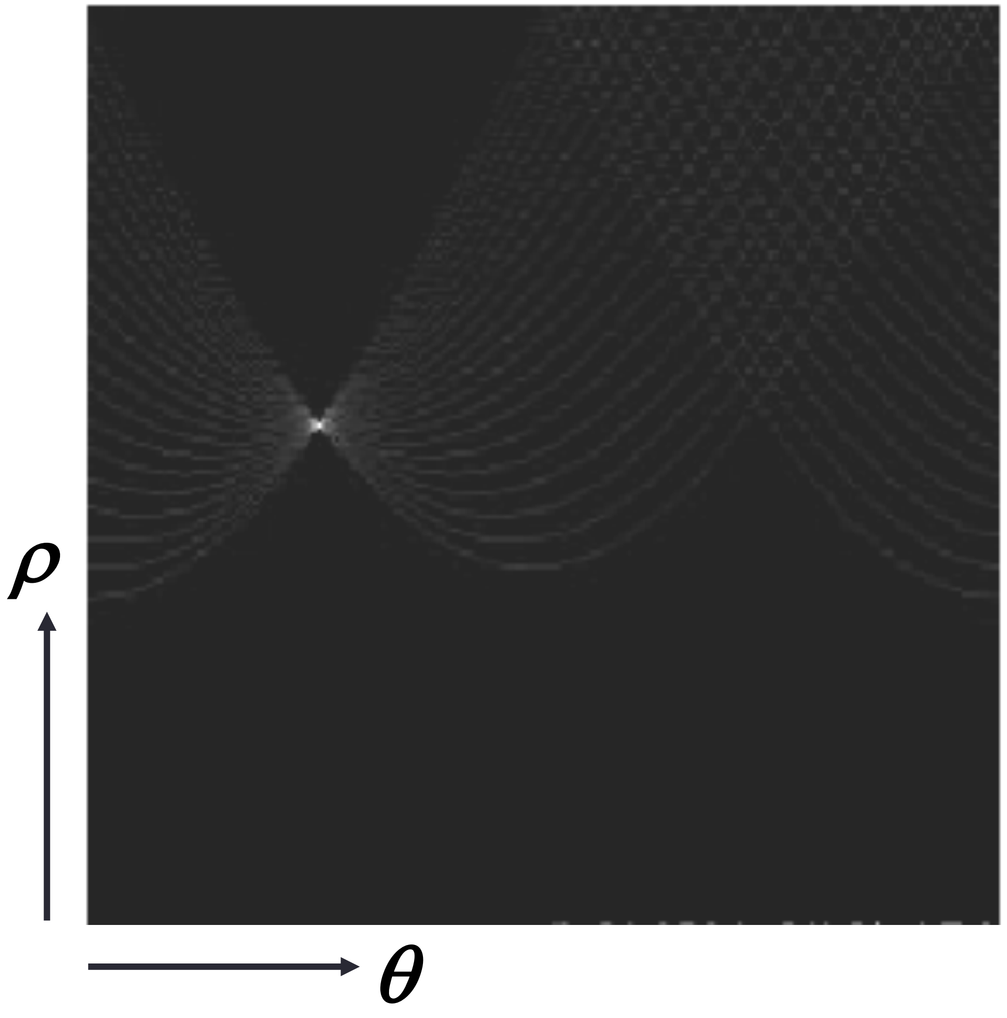


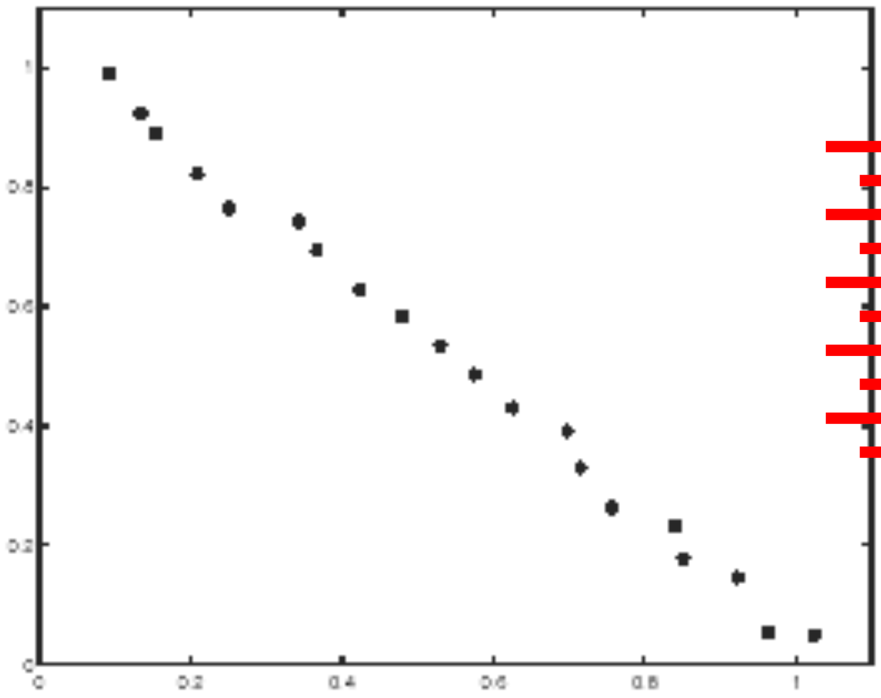
Image features

ρ,θ model parameter histogram

Slide from S. Savarese

Hough transform - experiments

Noisy data



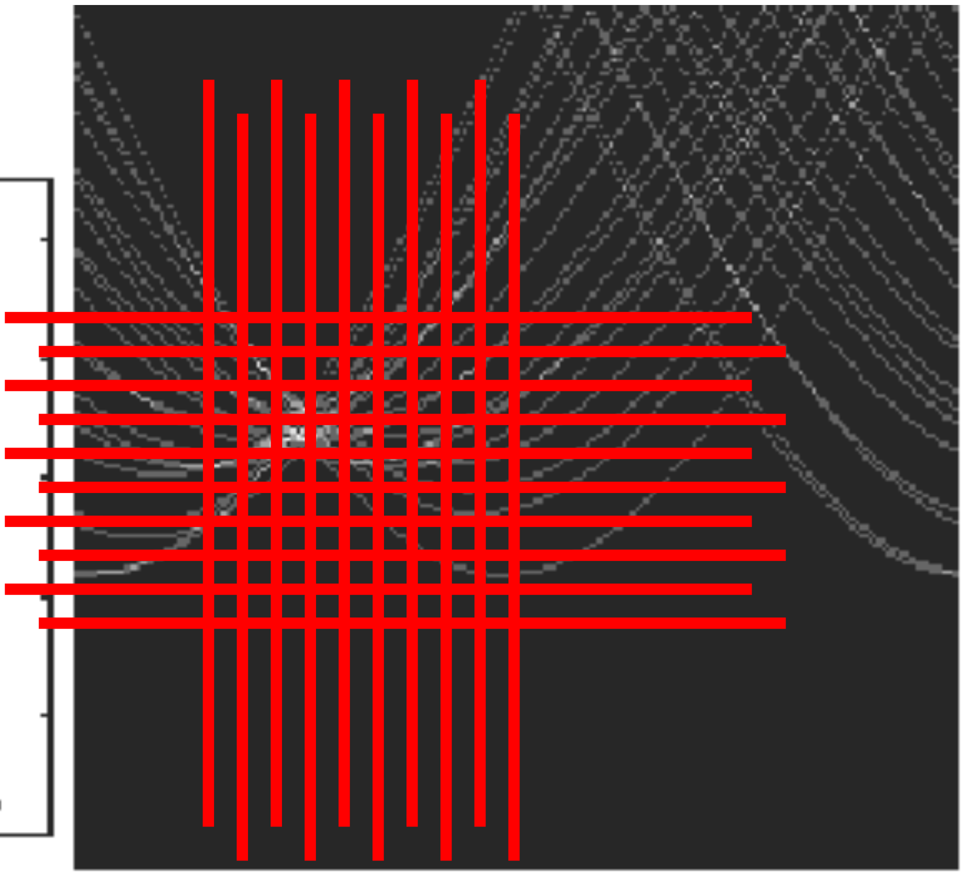


Image features

ρ,θ model parameter histogram

Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments

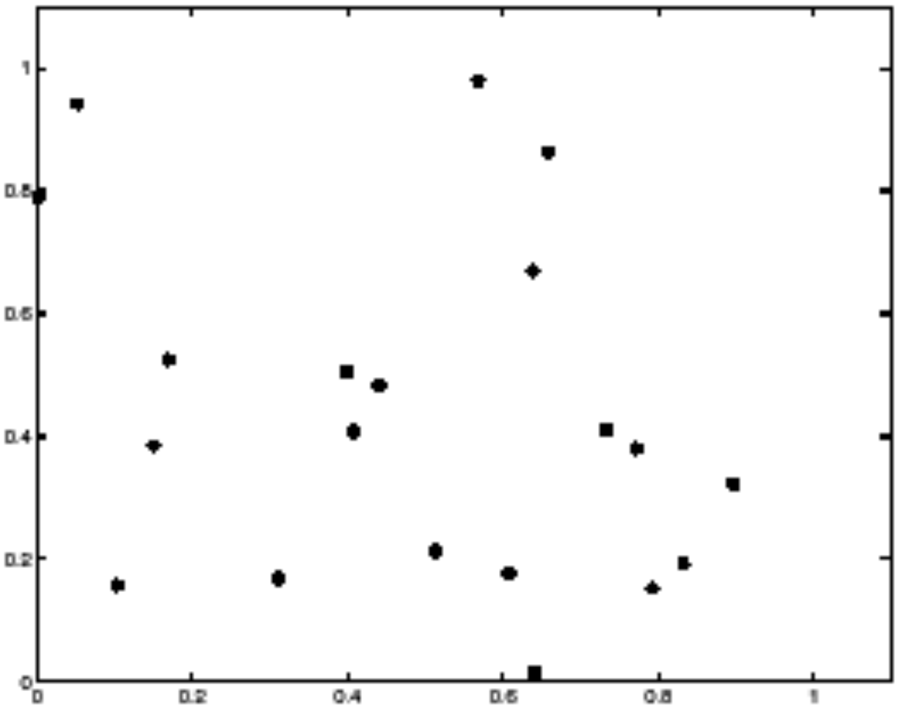
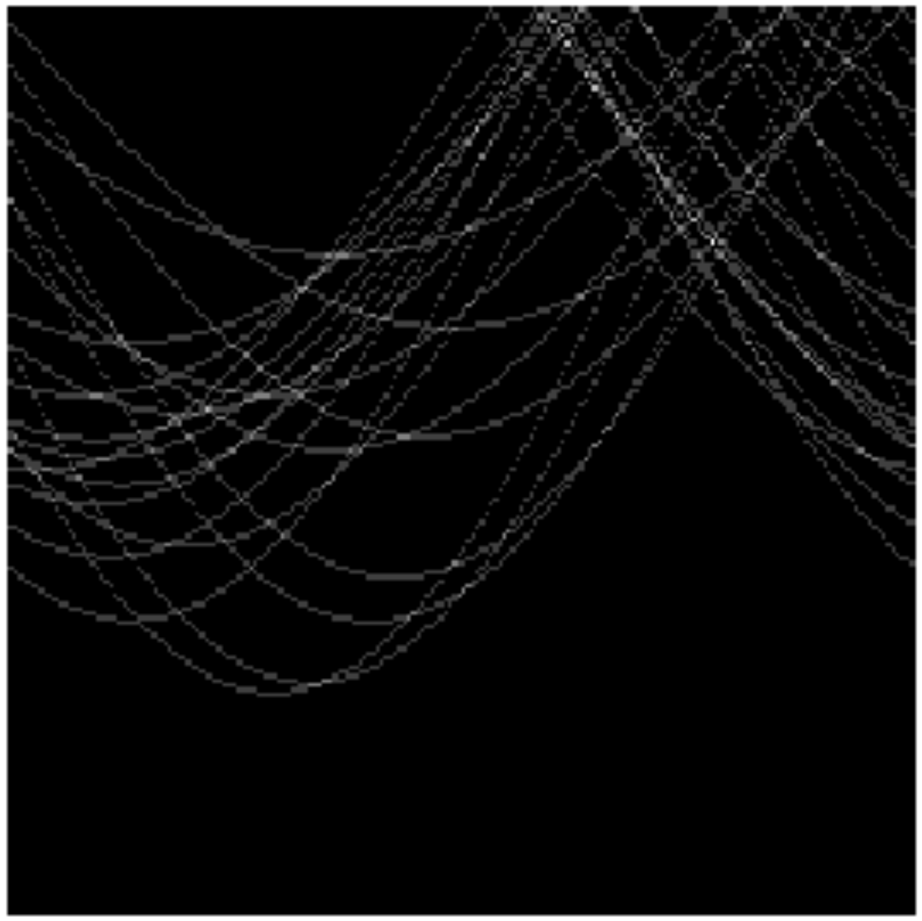


Image features



ρ,θ model parameter histogram

Issue: spurious peaks due to uniform noise

Slide from S. Savarese

Hough Transform Algorithm

procedure *Hough*$(\{(x, y, \theta)\})$:

1. Clear the accumulator array.

2. For each detected edgel at location (x, y) and orientation $\theta = \tan^{-1} n_y/n_x$, compute the value of

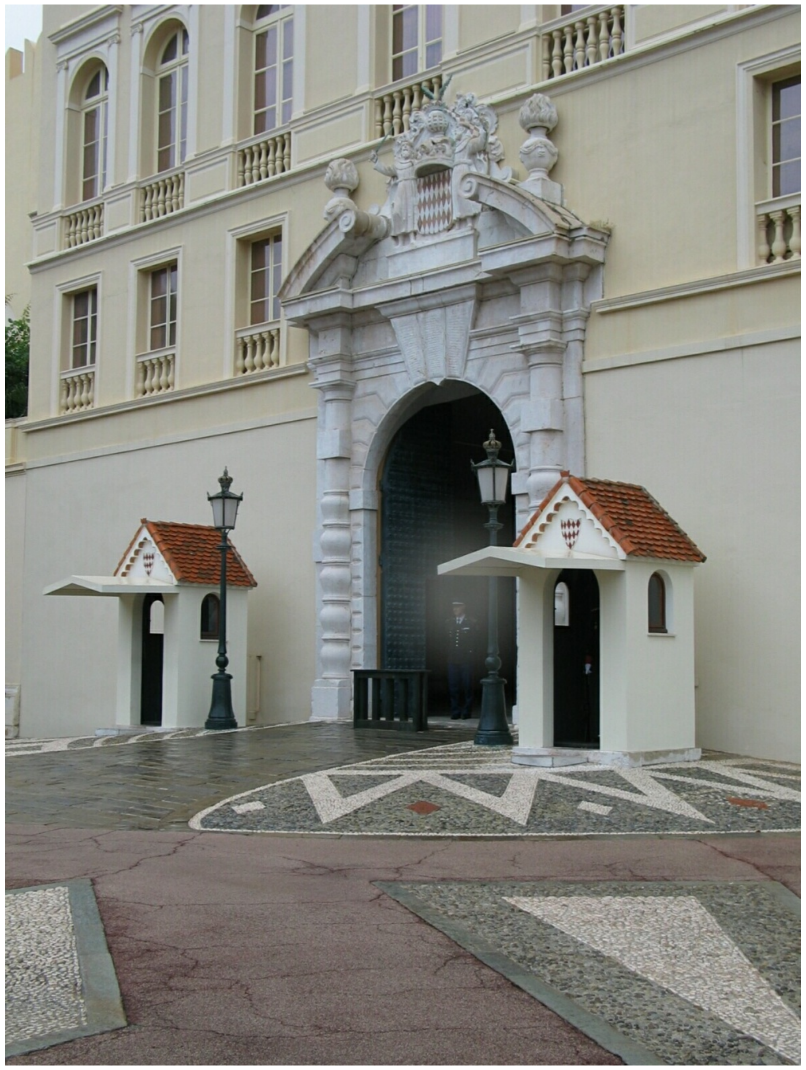
$$d = x\,n_x + y\,n_y$$

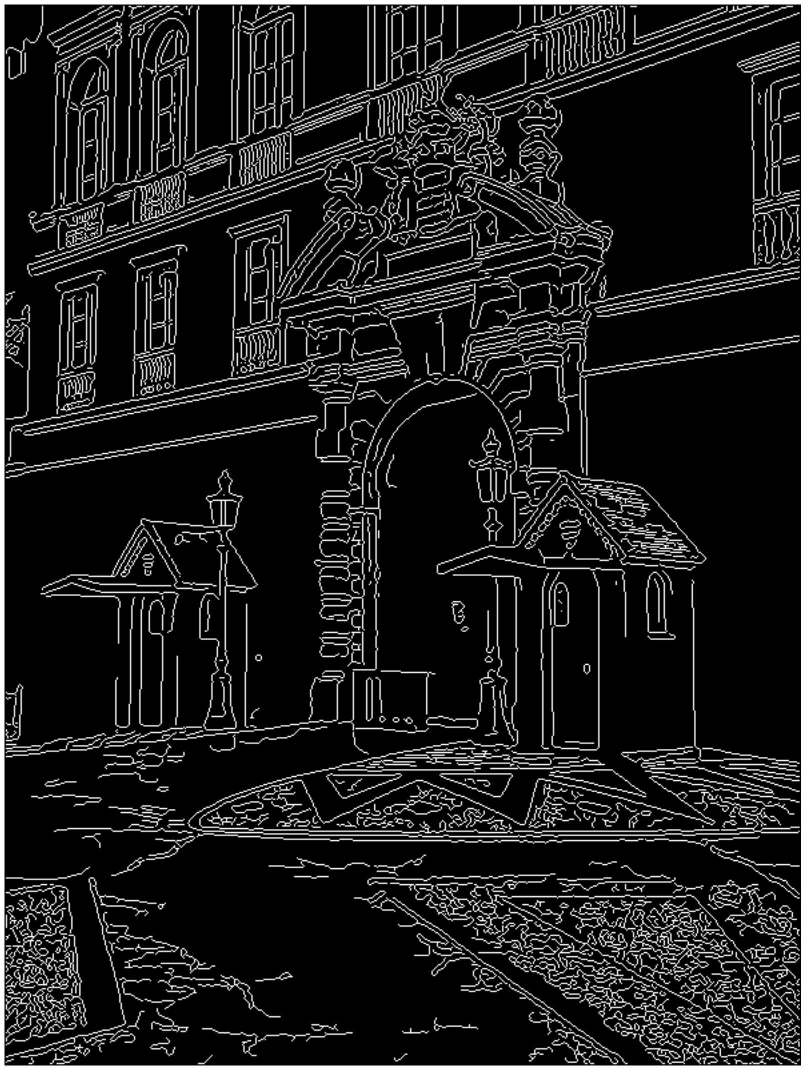
   and increment the accumulator corresponding to (θ, d).

3. Find the peaks in the accumulator corresponding to lines.

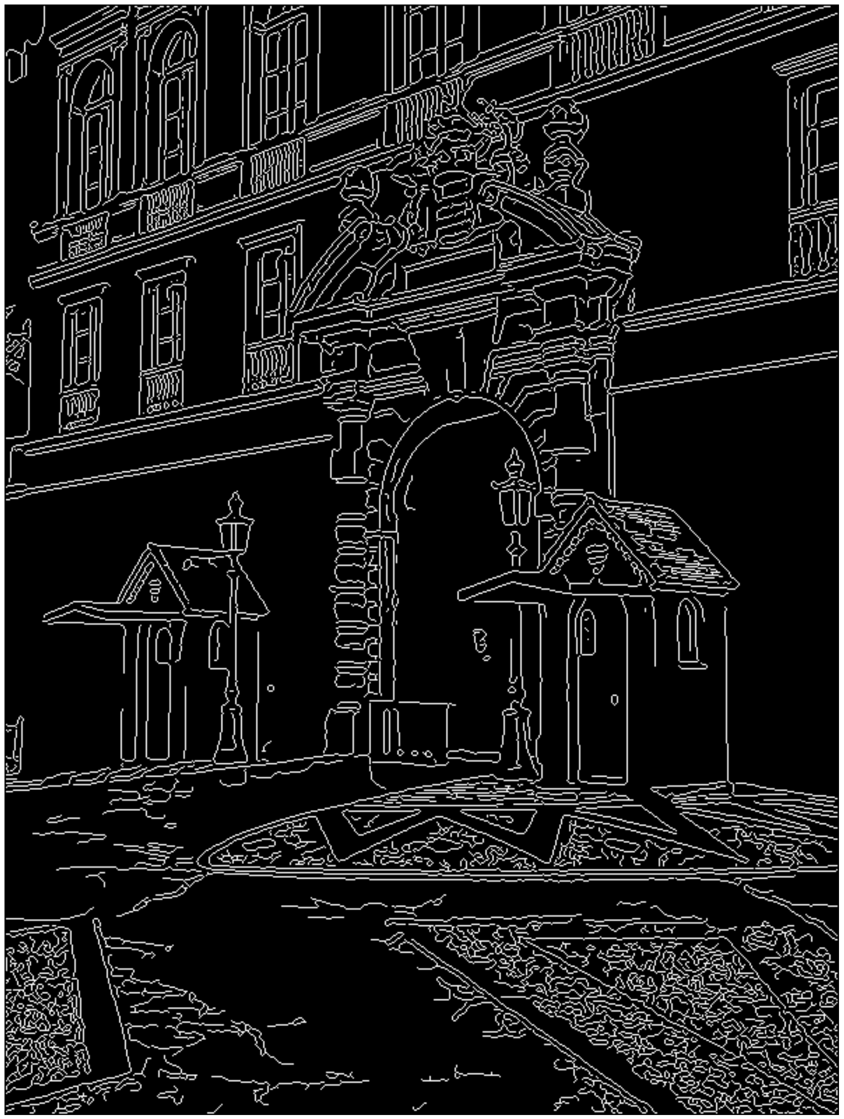
4. Optionally re-fit the lines to the constituent edgels.

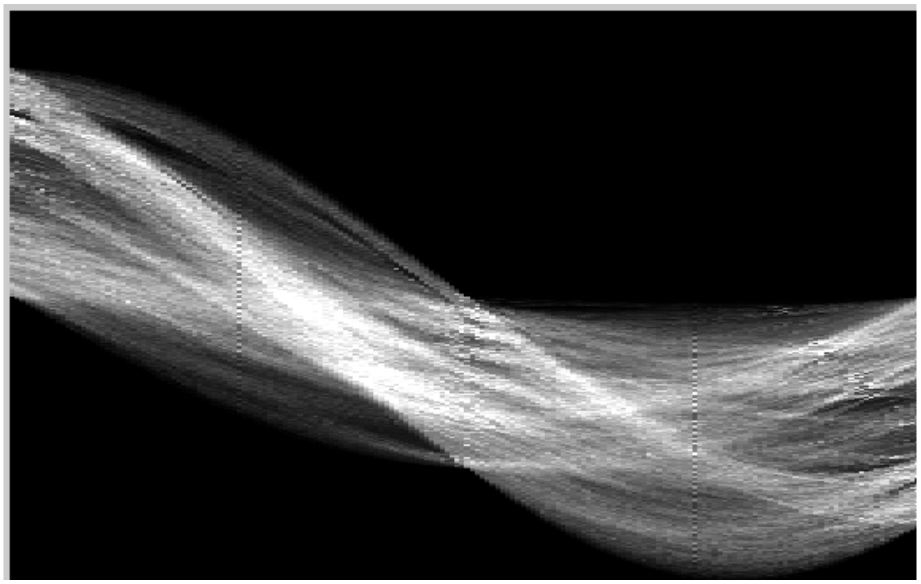
1. Image → Canny





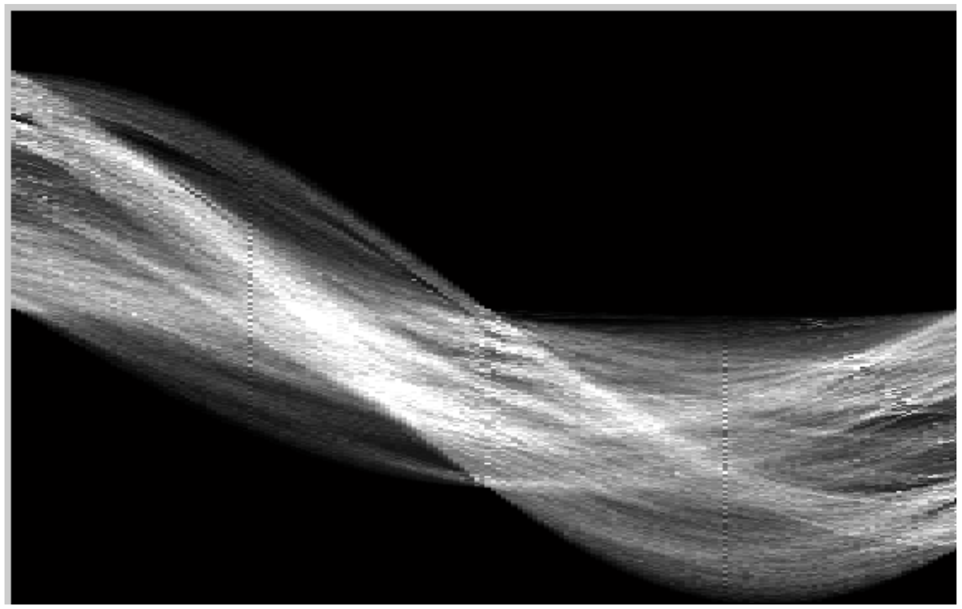
2. Canny → Hough votes

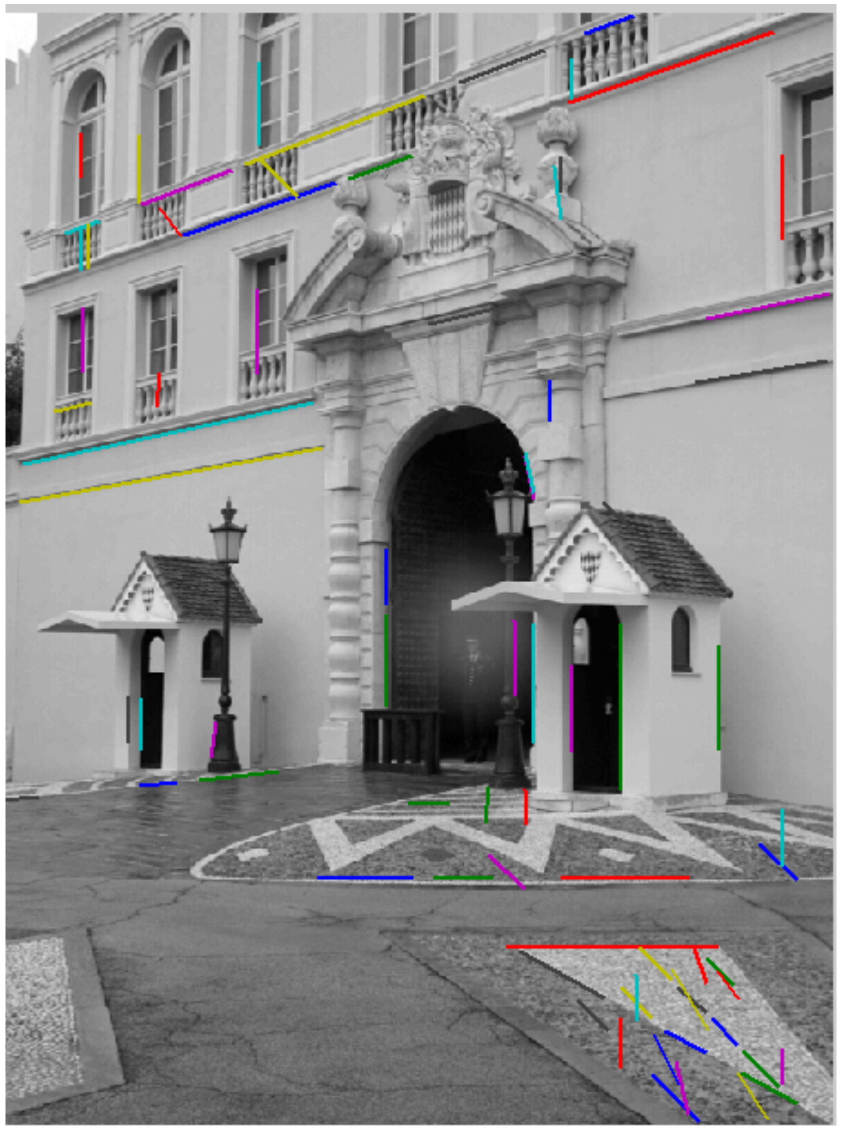




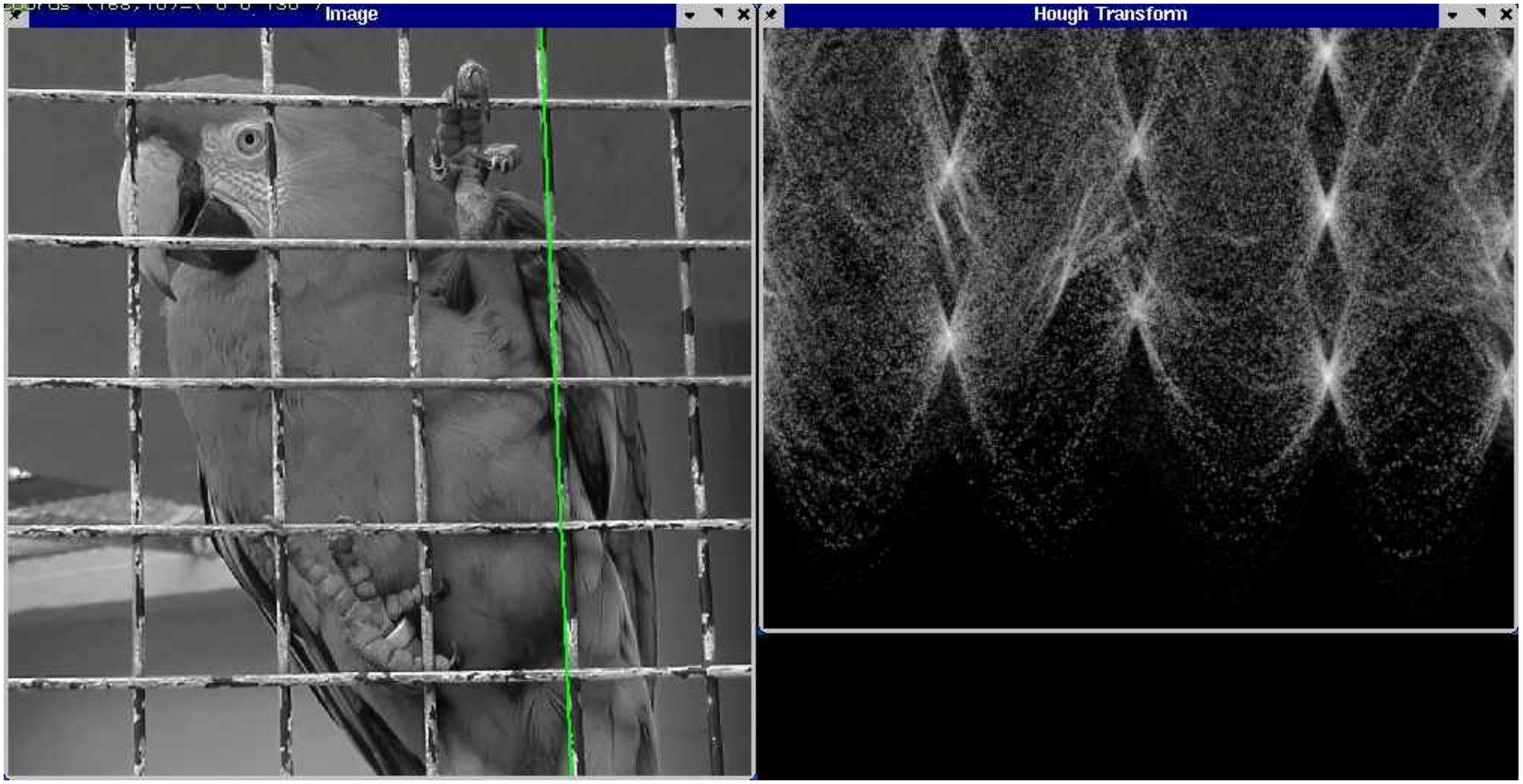
3. Hough votes → Edges

Find peaks and post-process



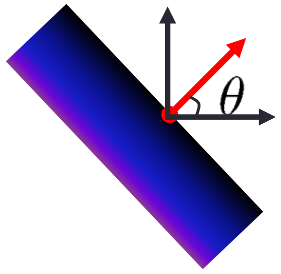


Hough transform example



http://ostatic.com/files/images/ss_hough.jpg

Incorporating image gradients



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!

- Modified Hough transform:

```
for each edge point (x,y)
        θ = gradient orientation at (x,y)
        ρ = x cos θ + y sin θ
        H(θ, ρ) = H(θ, ρ) + 1
end
```

Finding lines using Hough transform

- Using m,b parameterization
- Using r, theta parameterization
  - Using oriented gradients
- Practical considerations
  - Bin size
  - Smoothing
  - Finding multiple lines
  - Finding line segments

Hough Transform for Detection of Circles

- The parametric equation of the circle can be written as

$$(x-a)^2+(y-b)^2=r^2$$

- The equation has three parameters – a, b, r
- The curve obtained in the Hough Transform space for each edge point will be a right circular cone
- Point of intersection of the cones gives the parameters a, b, r

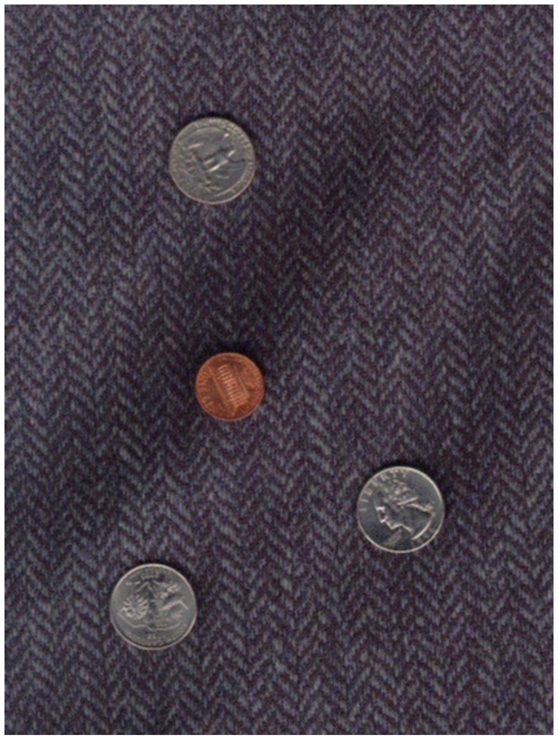
Hough Transform for Circles

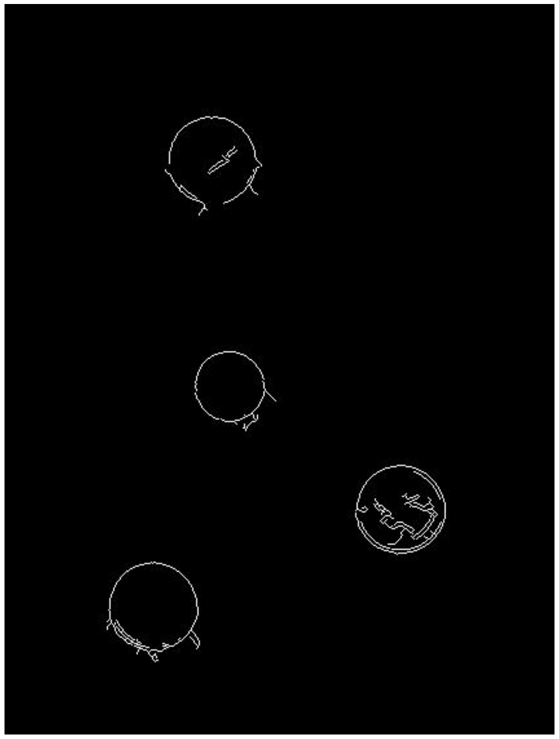
- Gradient at each edge point is known
- We know the line on which the center will lie

$$x_0 = x_i - R\cos\theta$$
$$y_0 = y_i - R\sin\theta$$

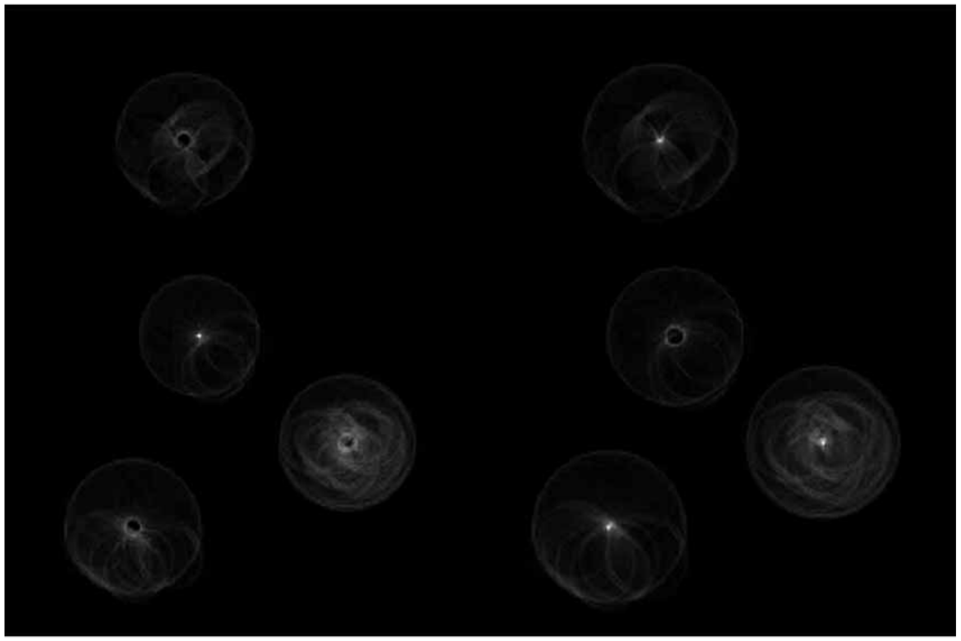
- If the radius is also known then center of the circle can be located

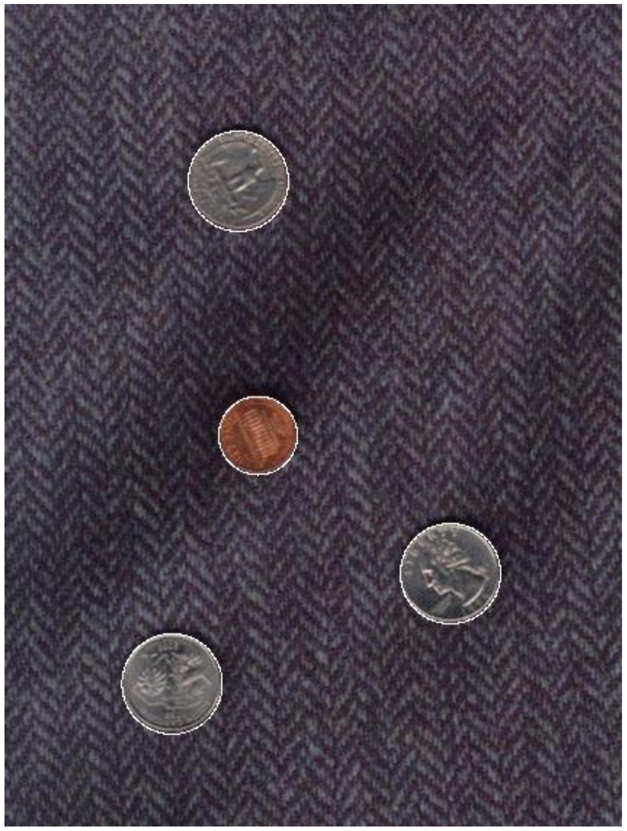
4.3 Detection of circle by Hough Transform - example


Original Image


Circles detected by Canny Edge Detector

4.4 Detection of circle by Hough Transform - contd



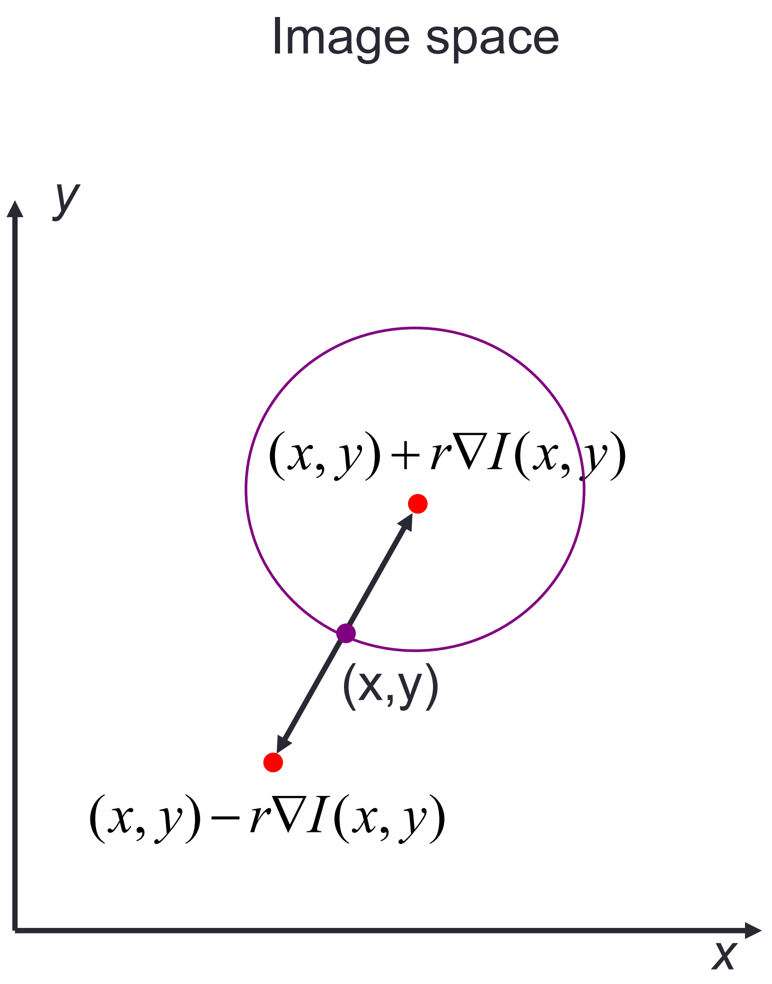


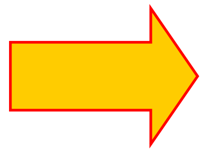
Hough Transform of the edge detected image Detected Circles

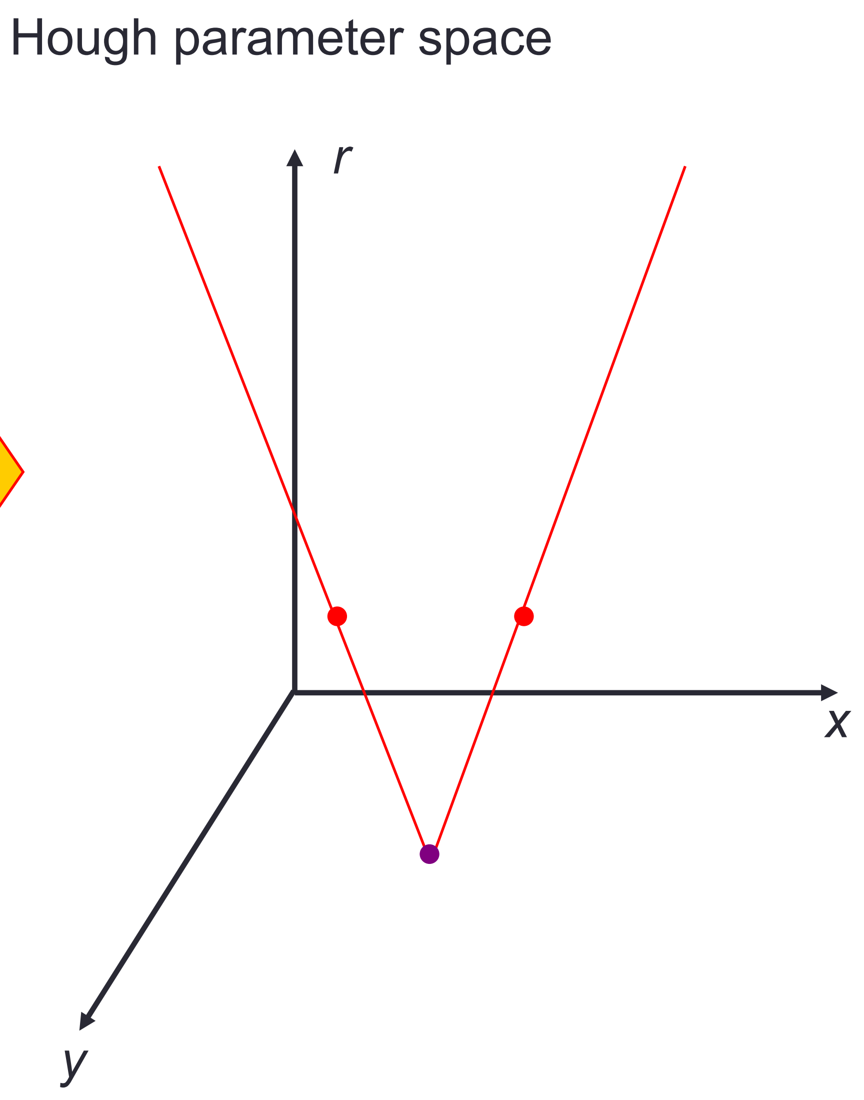
Hough Transform

- How would we find circles of unknown radius?

Hough transform for circles

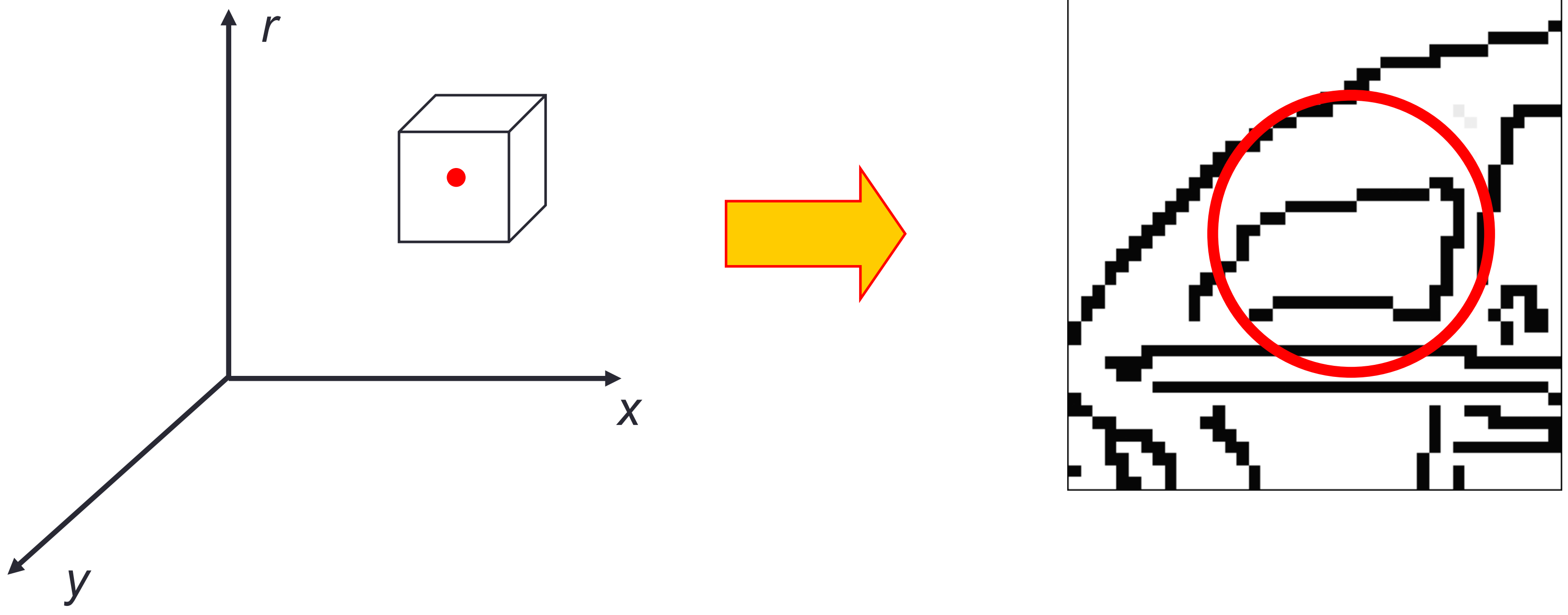






Hough transform for circles

- Conceptually equivalent procedure: for each (x,y,r), draw the corresponding circle in the image and compute its "support"



Is this more or less efficient than voting with features?

Recap

- In detecting lines
  - The parameters ρ and θ were found out relative to the origin (0,0)
- In detecting circles
  - The radius and center were found out
- In both the cases we have knowledge of the shape
- We aim to find out its location and orientation in the image
- The idea can be extended to shapes like ellipses, parabolas, etc.

Parameters for analytic curves

Analytic Form	Parameters	Equation
Line	ρ, θ	$x\cos\theta + y\sin\theta = \rho$
Circle	x_0, y_0, ρ	$(x-x_o)^2+(y-y_0)^2=r^2$
Parabola	x_0, y_0, ρ, θ	$(y-y_0)^2=4\rho(x-x_o)$
Ellipse	x_0, y_0, a, b, θ	$(x-x_o)^2/a^2+(y-y_0)^2/b^2=1$

Generalized Hough Transform

- The Generalized Hough transform can be used to detect arbitrary shapes
- Complete specification of the exact shape of the target object is required in the form of the R-Table
- Information that can be extracted are
  - Location
  - Size
  - Orientation
  - Number of occurrences of that particular shape

Creating the R-table

- Algorithm
  - Choose a reference point
  - Draw a vector from the reference point to an edge point on the boundary
  - Store the information of the vector against the gradient angle in the R-Table
  - There may be more than one entry in the R-Table corresponding to a gradient value

Generalized Hough Transform - Algorithm

- Form an Accumulator array to hold the candidate locations of the reference point
- For each point on the edge
  - Compute the gradient direction and determine the row of the R-Table it corresponds to
  - For each entry on the row calculate the candidate location of the reference point

$$x_c = x_i + r\cos\theta$$

$$y_c = y_i + r\sin\theta$$

  - Increase the Accumulator value for that point
- The reference point location is given by the highest value in the accumulator array

Generalized Hough Transform – Size and Orientation

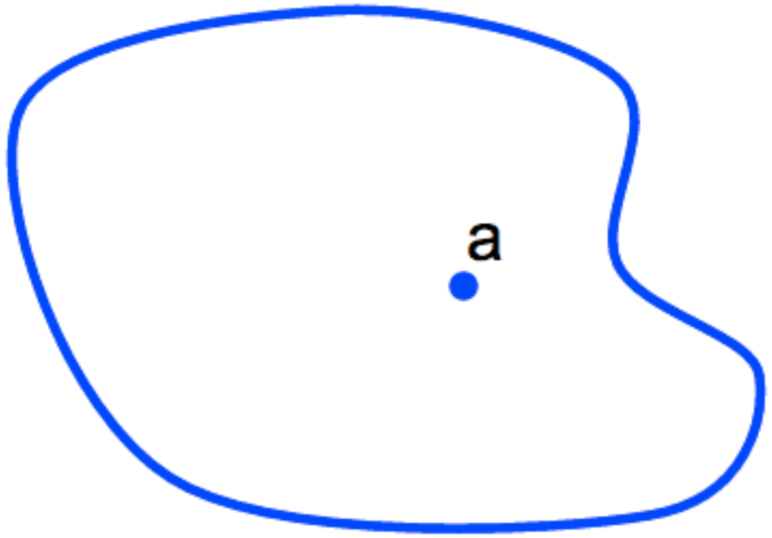
- The size and orientation of the shape can be found out by simply manipulating the R-Table
- For scaling by factor *S* multiply the R-Table vectors by *S*
- For rotation by angle θ, rotate the vectors in the R-Table by angle θ

Generalized Hough Transform – Advantages and disadvantages

- Advantages
  - A method for object recognition
  - Robust to partial deformation in shape
  - Tolerant to noise
  - Can detect multiple occurrences of a shape in the same pass
- Disadvantages
  - Lot of memory and computation is required

Generalized Hough Transform

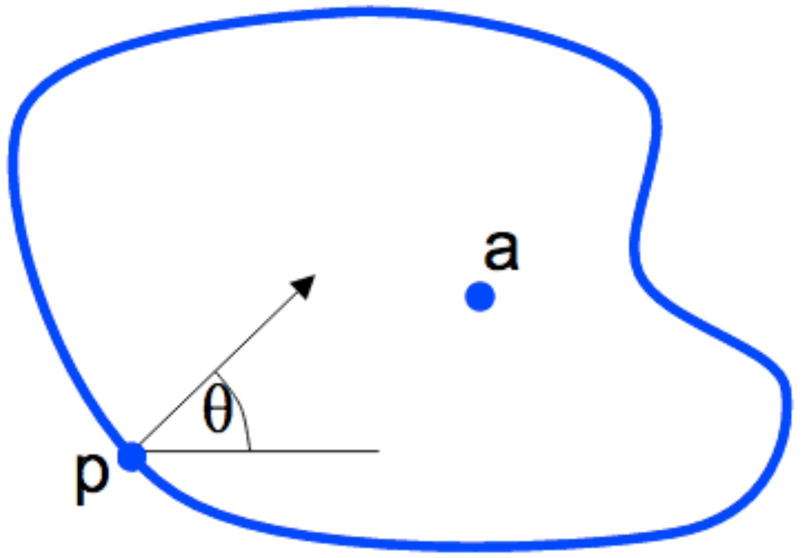
- We want to find a shape defined by its boundary points and a reference point



D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981, pp. 111-122.

Generalized Hough Transform

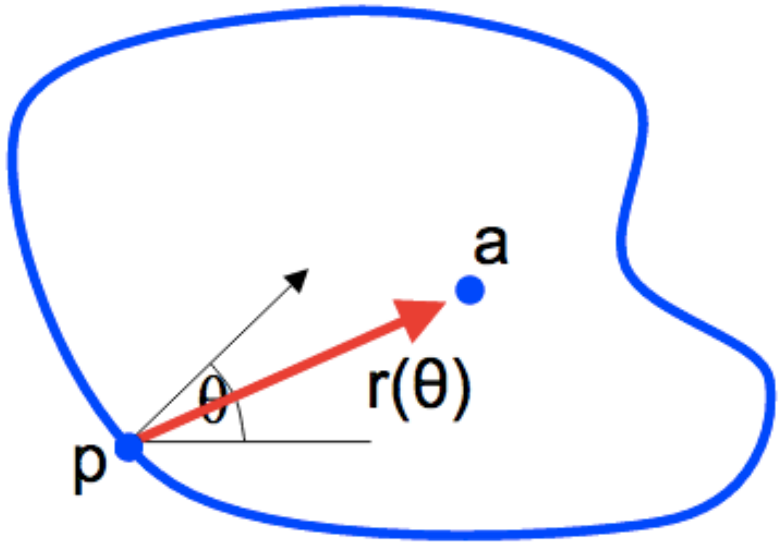
- We want to find a shape defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector r = a – p as a function of gradient orientation θ



D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981, pp. 111-122.

Generalized Hough Transform

- We want to find a shape defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector r = a – p as a function of gradient orientation θ



D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981, pp. 111-122.

Generalized Hough Transform

- For model shape: construct a table indexed by θ storing displacement vectors r as function of gradient direction
- Detection: For each edge point p with gradient orientation θ:
  - Retrieve all r indexed with θ
  - For each $r(\theta)$, put a vote in the Hough space at $p + r(\theta)$
- Peak in this Hough space is reference point with most supporting edges
- Assumption: translation is the only transformation here, i.e., orientation and scale are fixed

Source: K. Graumar

Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (much faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

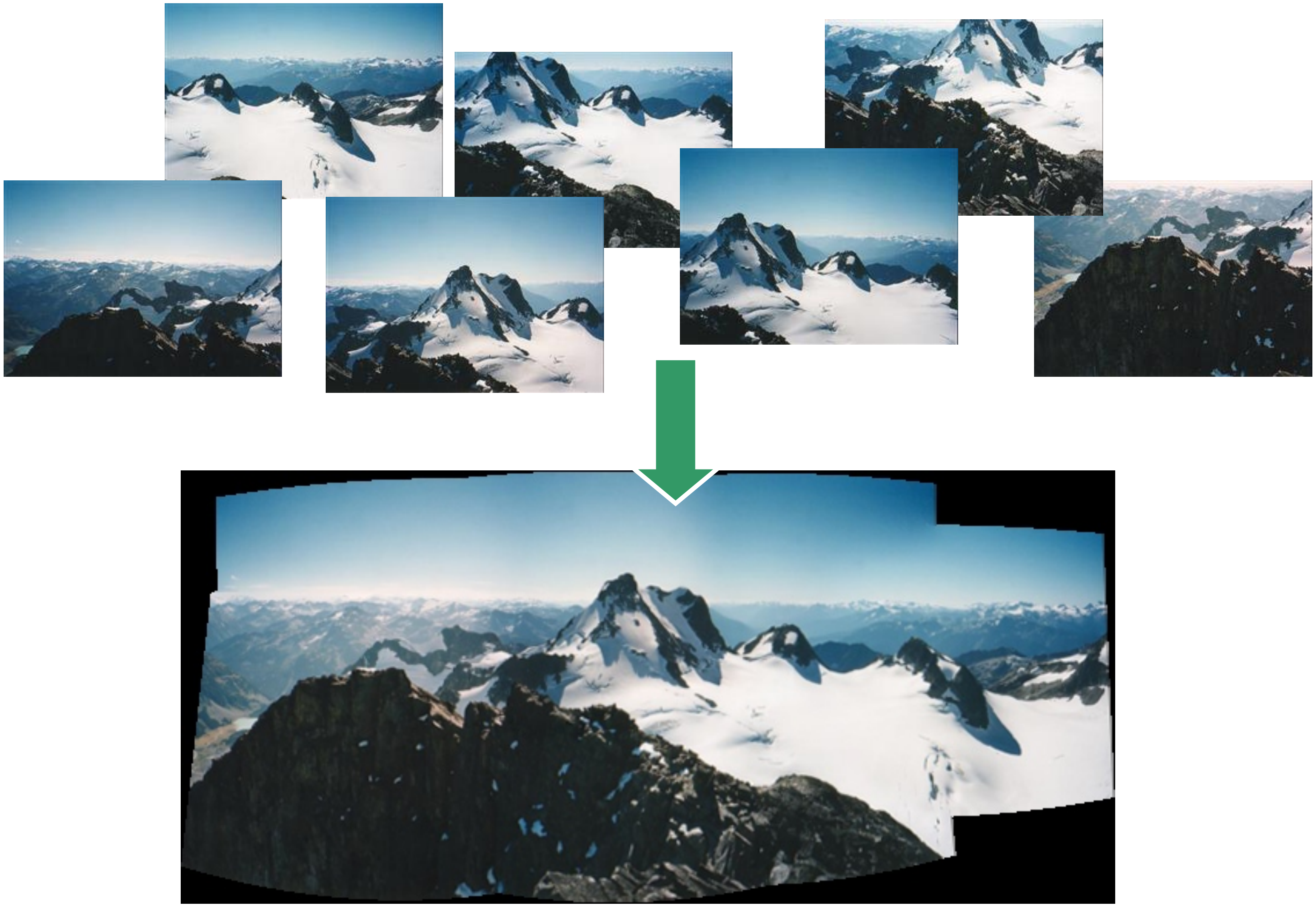
Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are affine transform)
- Object category recognition (parameters are position/scale)

Feature detection and matching – Why?

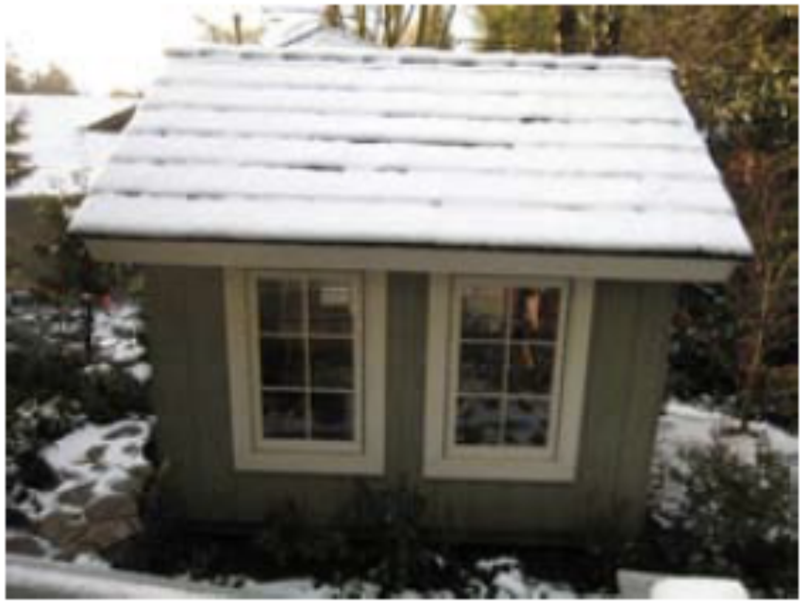
Image stitching

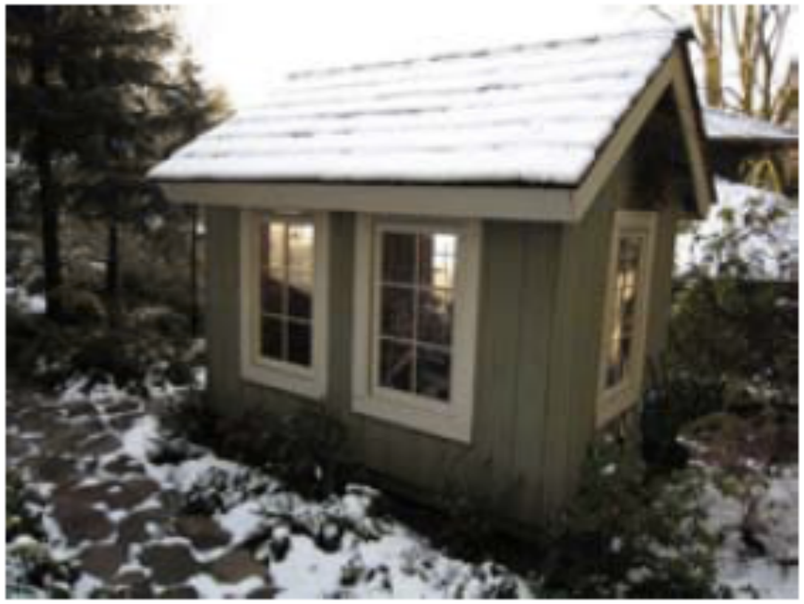
Image Stitching



[Brown, Szeliski, Winder CVPR 2005]

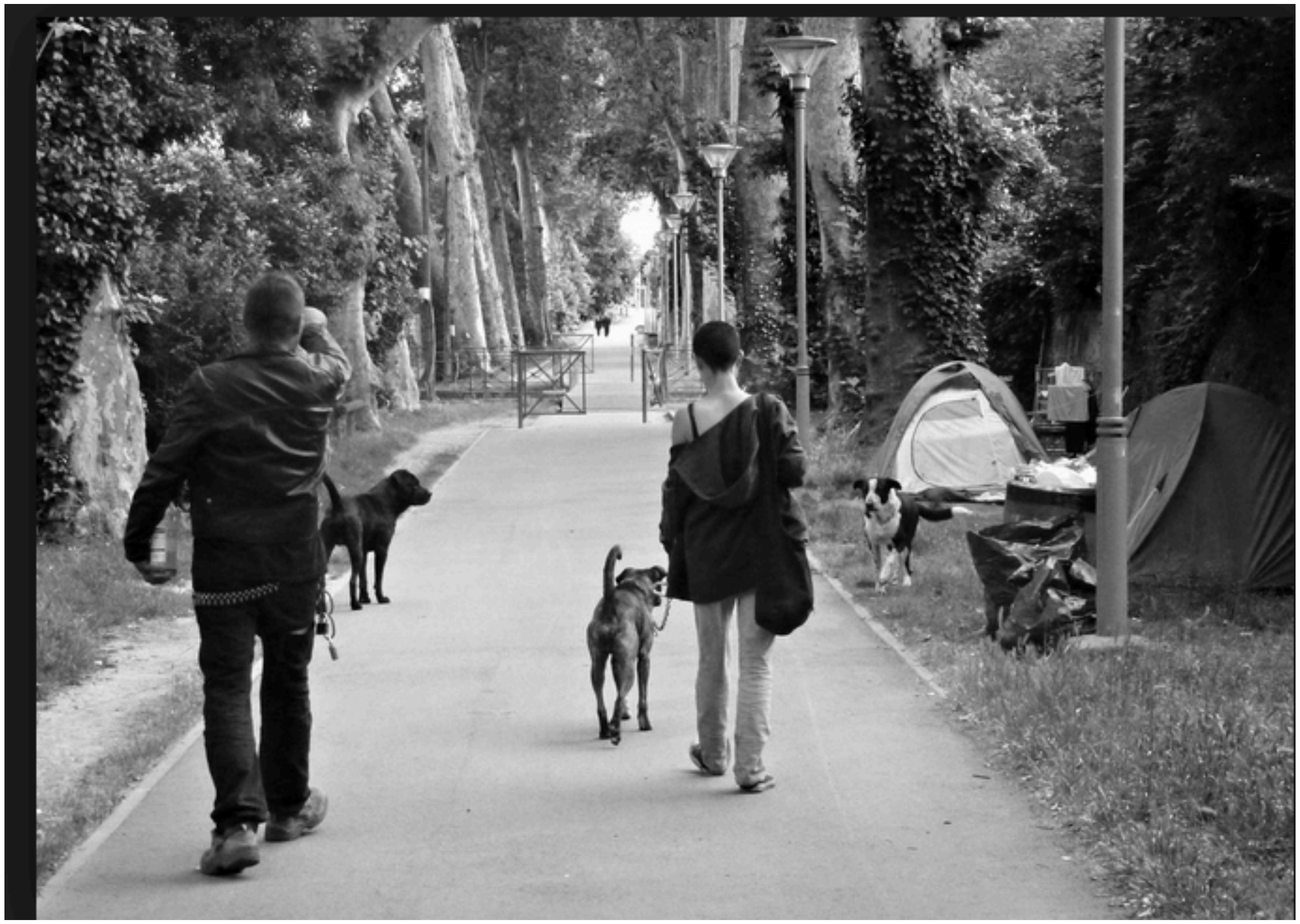
Feature detection and matching – Why?





3D Reconstruction and Alignment

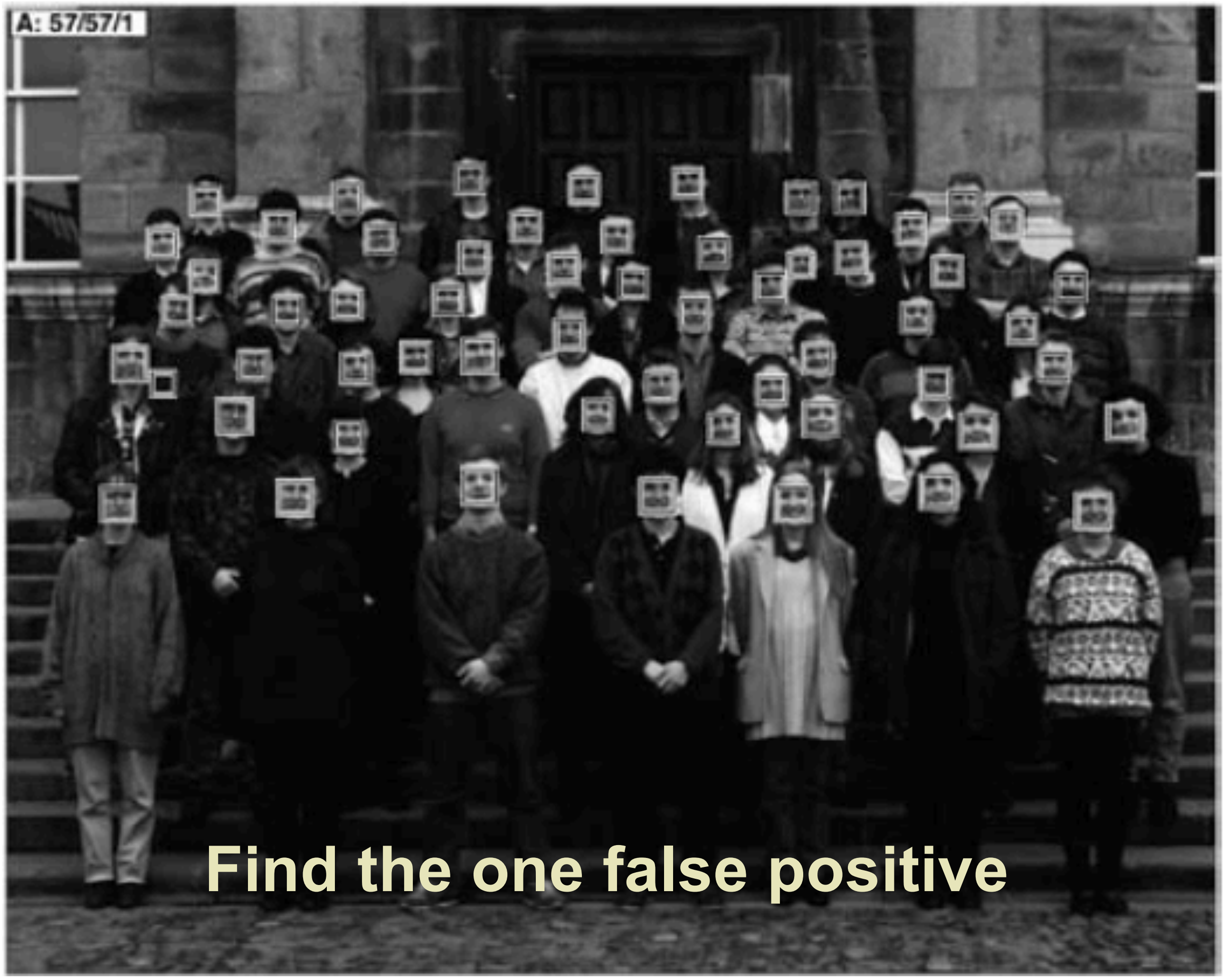
Feature detection and matching – Why?



Object detection and classification

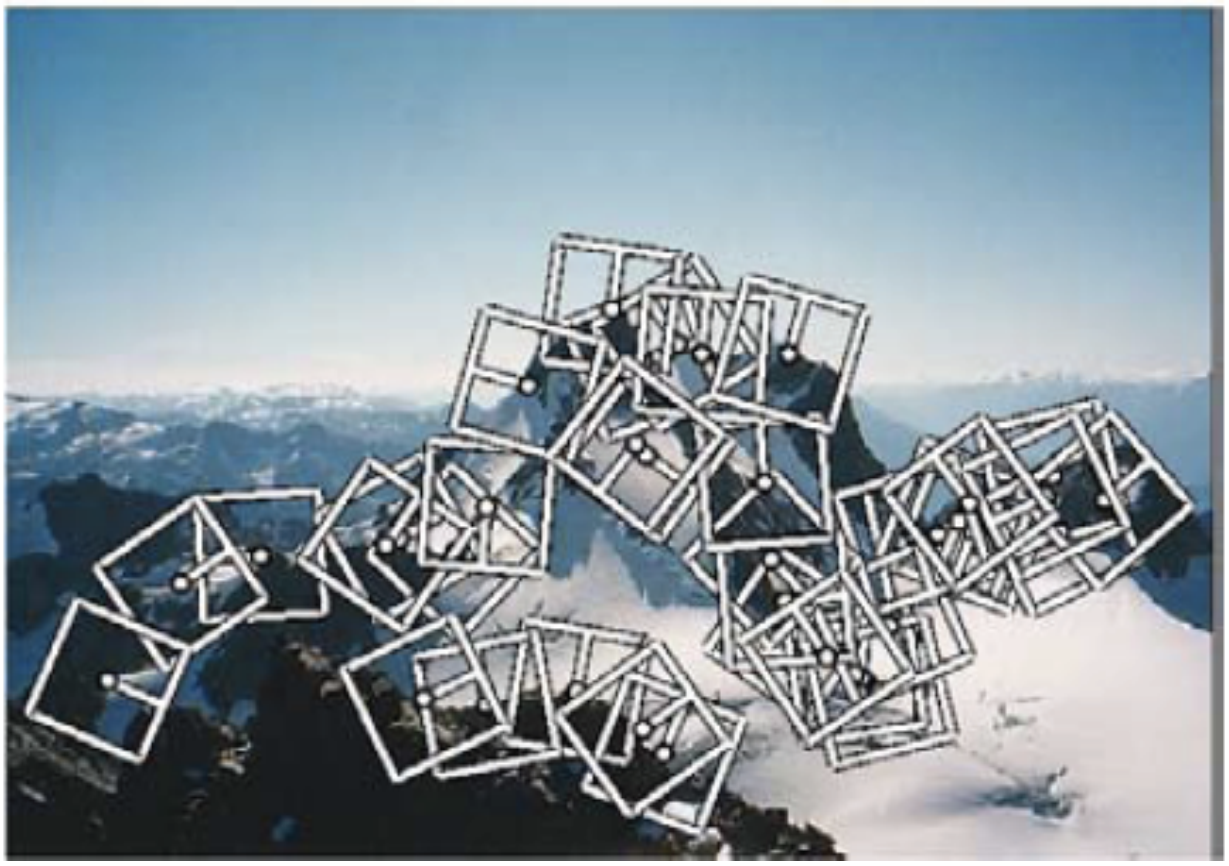
http://inthefray.org/2015/07/strays-street-people-and-their-dogs/

Feature detection and matching – Why?



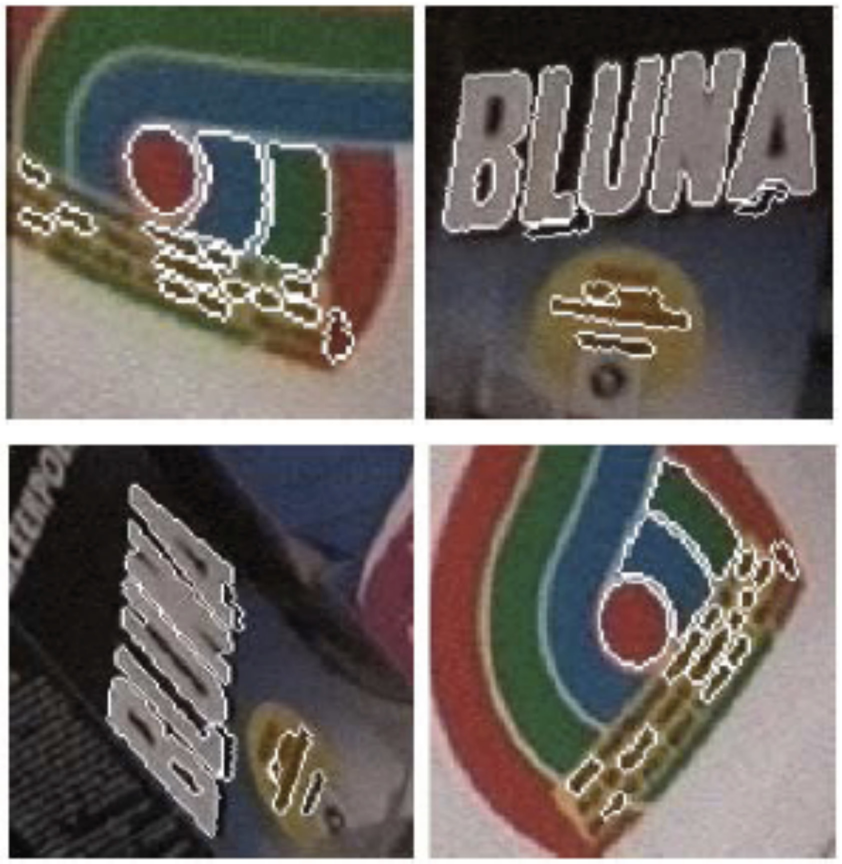
Object detection and classification

Feature detectors and descriptors



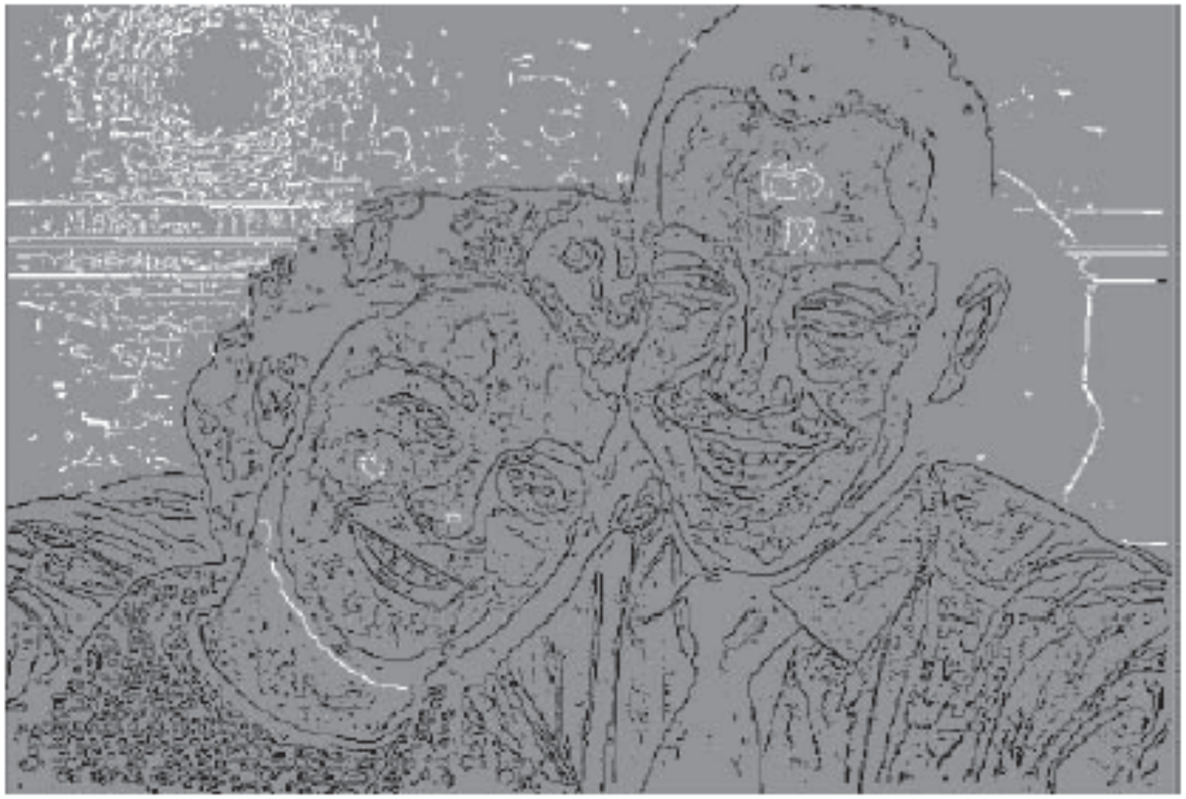
Point-like interest operators (Brown, Szeliski, and Winder 2005)

Feature detectors and descriptors



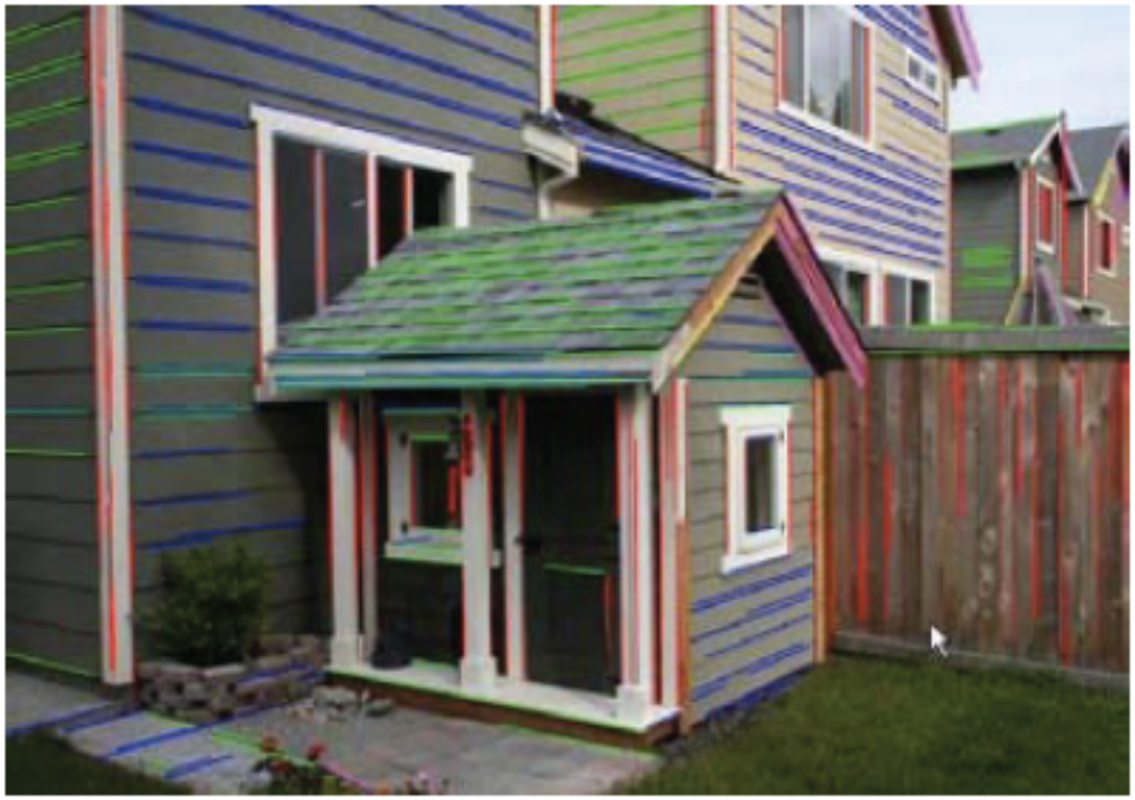
region-like interest operators (Matas, Chum, Urban et al. 2004)

Feature detectors and descriptors



Edges (Elder and Goldberg 2001)

Feature detectors and descriptors



Straight lines (Sinha, Steedly, Szeliski et al. 2008)

Finding feature points and their correspondences

- Two main approaches:
  - Find features in one image that can be accurately tracked using a local search technique, such as correlation or least squares

  Nearby viewpoints

- Independently detect features in all the images under consideration and then match features based on their local appearance

  Large distance, appearance change

Feature detection and matching

- Feature detection (extraction)
- Feature description
- Feature matching
- Feature tracking

Feature detection and matching

- Feature detection (extraction)
  - each image is searched for locations that are likely to match well in other images.
- Feature description
- Feature matching
- Feature tracking

Feature detection and matching

- Feature detection (extraction)
- Feature description
  - each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
- Feature matching
- Feature tracking

Feature detection and matching

- Feature detection (extraction)
- Feature description
- Feature matching
  - efficiently searches for likely matching candidates in other images.
- Feature tracking

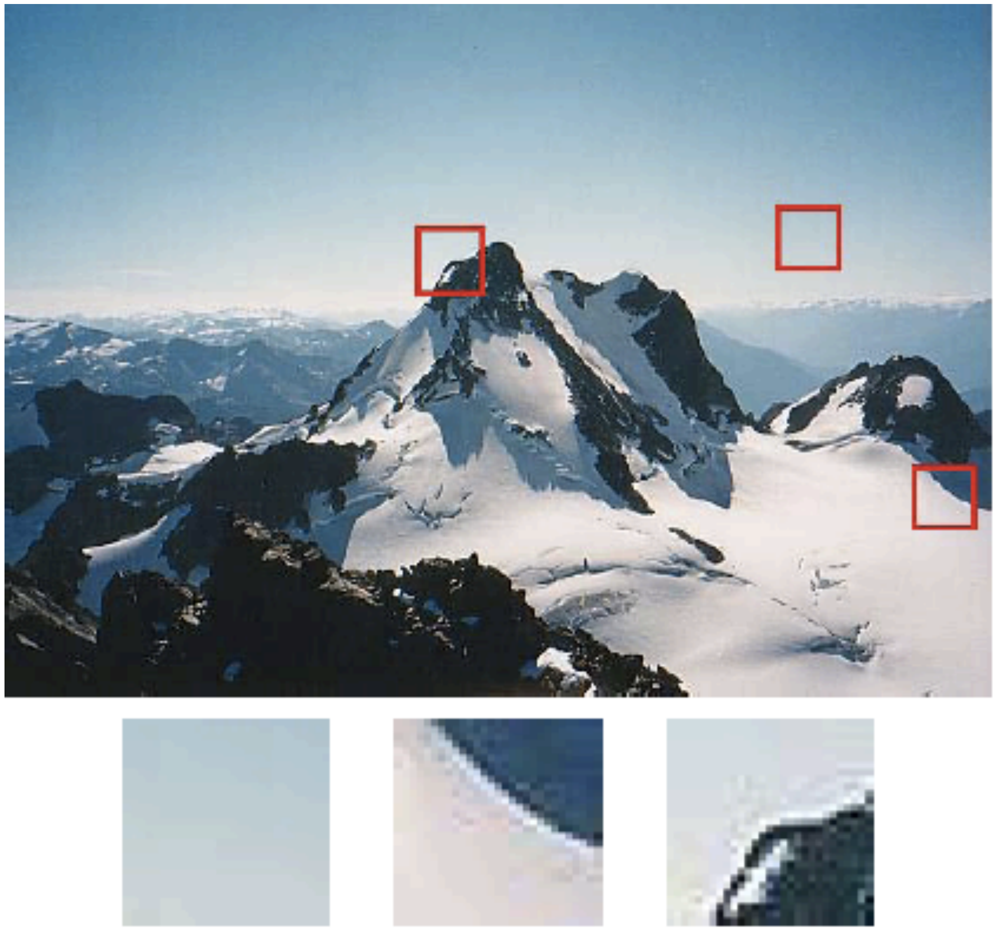
Feature detection and matching

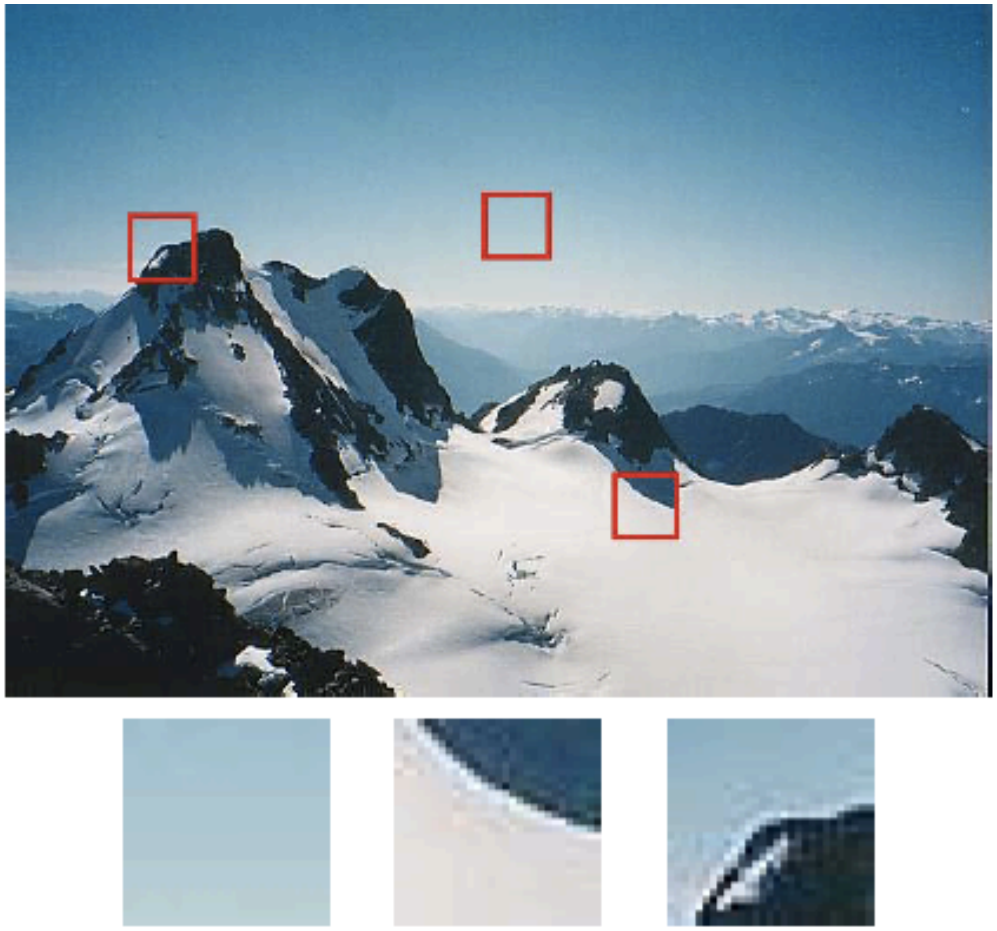
- Feature detection (extraction)
- Feature description
- Feature matching
- Feature tracking
    - alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

Feature detection and matching

- Feature detection (extraction)
  - each image is searched for locations that are likely to match well in other images.
- Feature description
  - each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
- Feature matching
  - efficiently searches for likely matching candidates in other images.
- Feature tracking
  - alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

What are good key-points (patches)?





Comparing two image patches

$$E_{\mathrm{WSSD}}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$

Weighted Sum Square Differences (WSSD)

I_0, I_1	two images being compared
$\mathbf{u} = (u, v)$	displacement vector
$w(x_i)$	Spatially varying weighting function

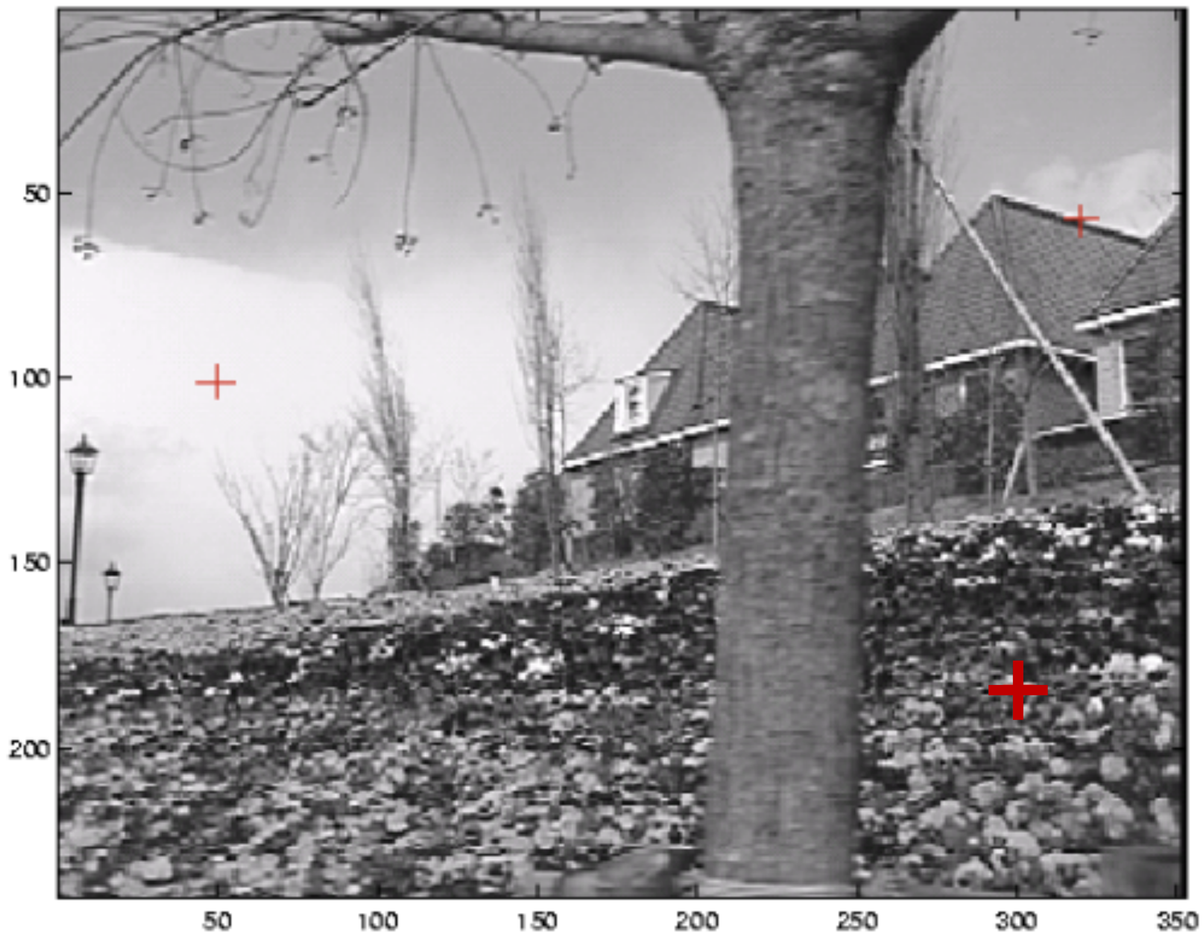
Comparing an image patch against itself

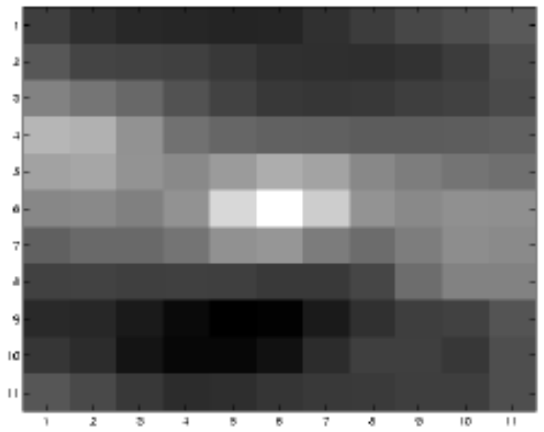
$$E_{\mathrm{AC}}(\Delta \boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i + \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$

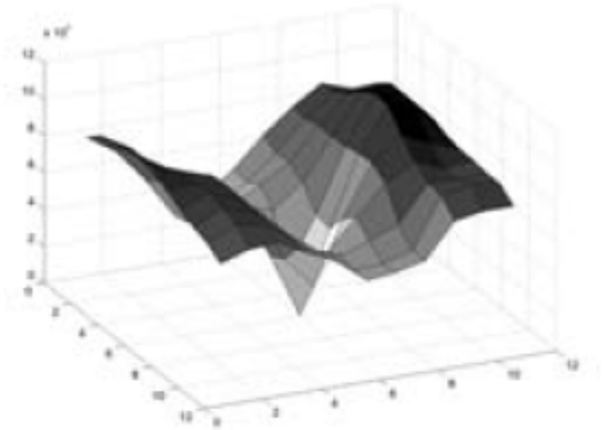
an auto-correlation function or surface

Measure how stable this metric with respect to small variations in positions $\Delta \mathbf{u}$

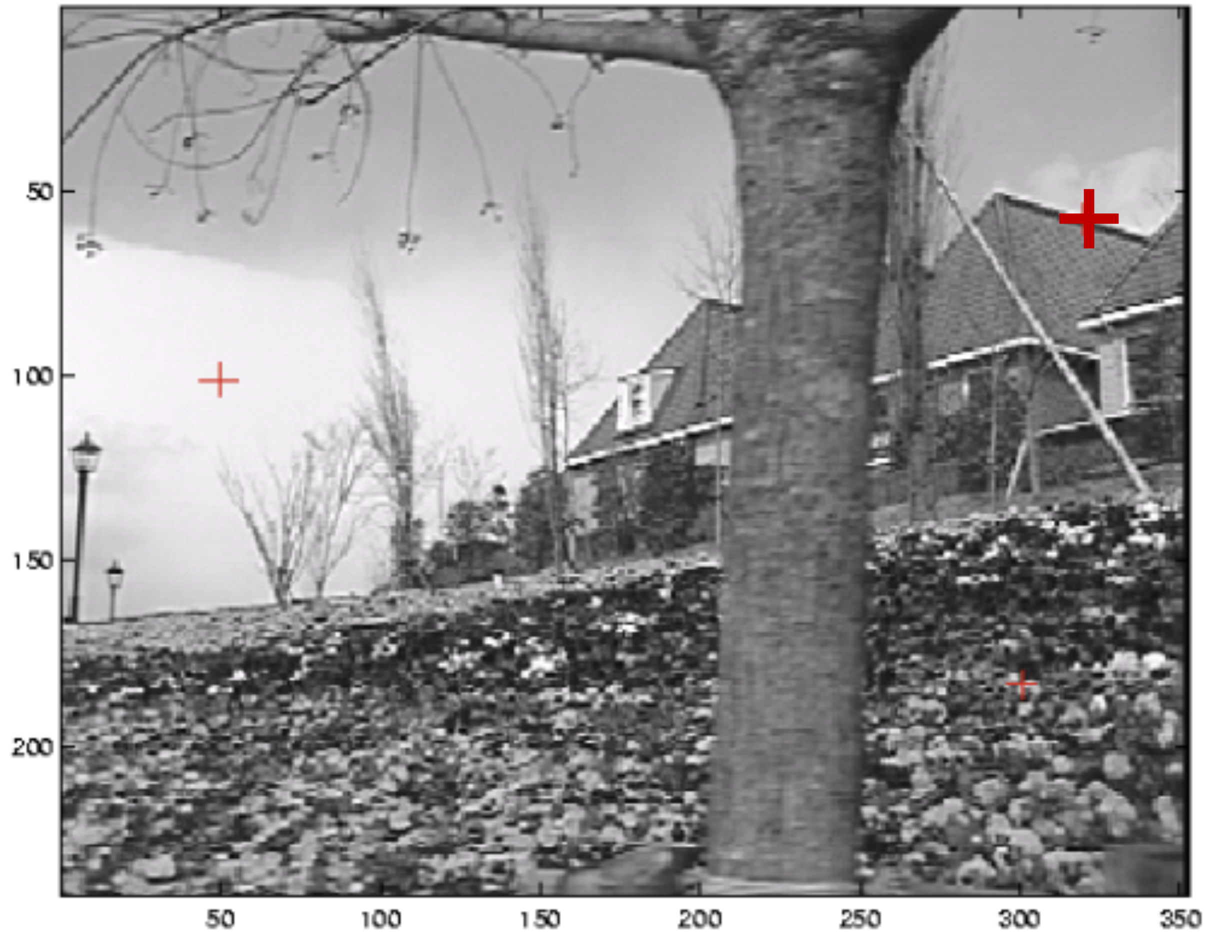
Auto-correlation surfaces

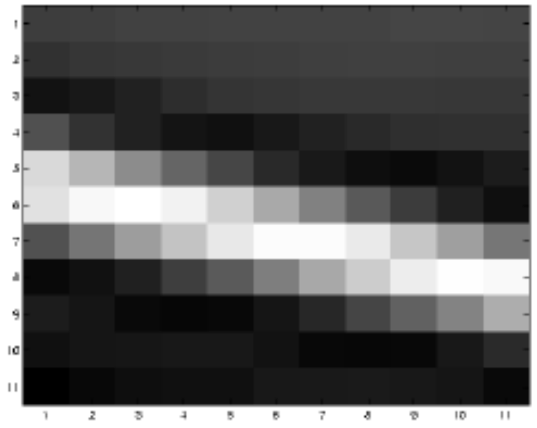


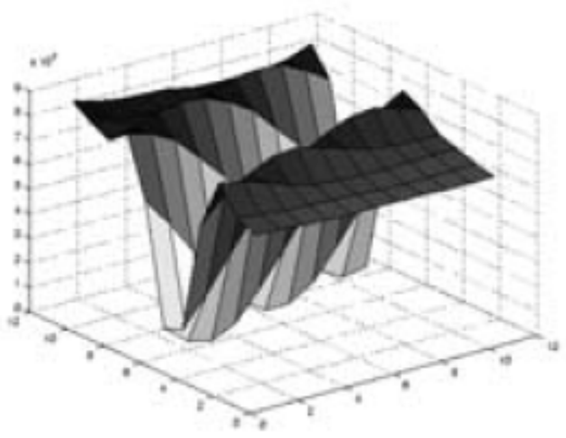




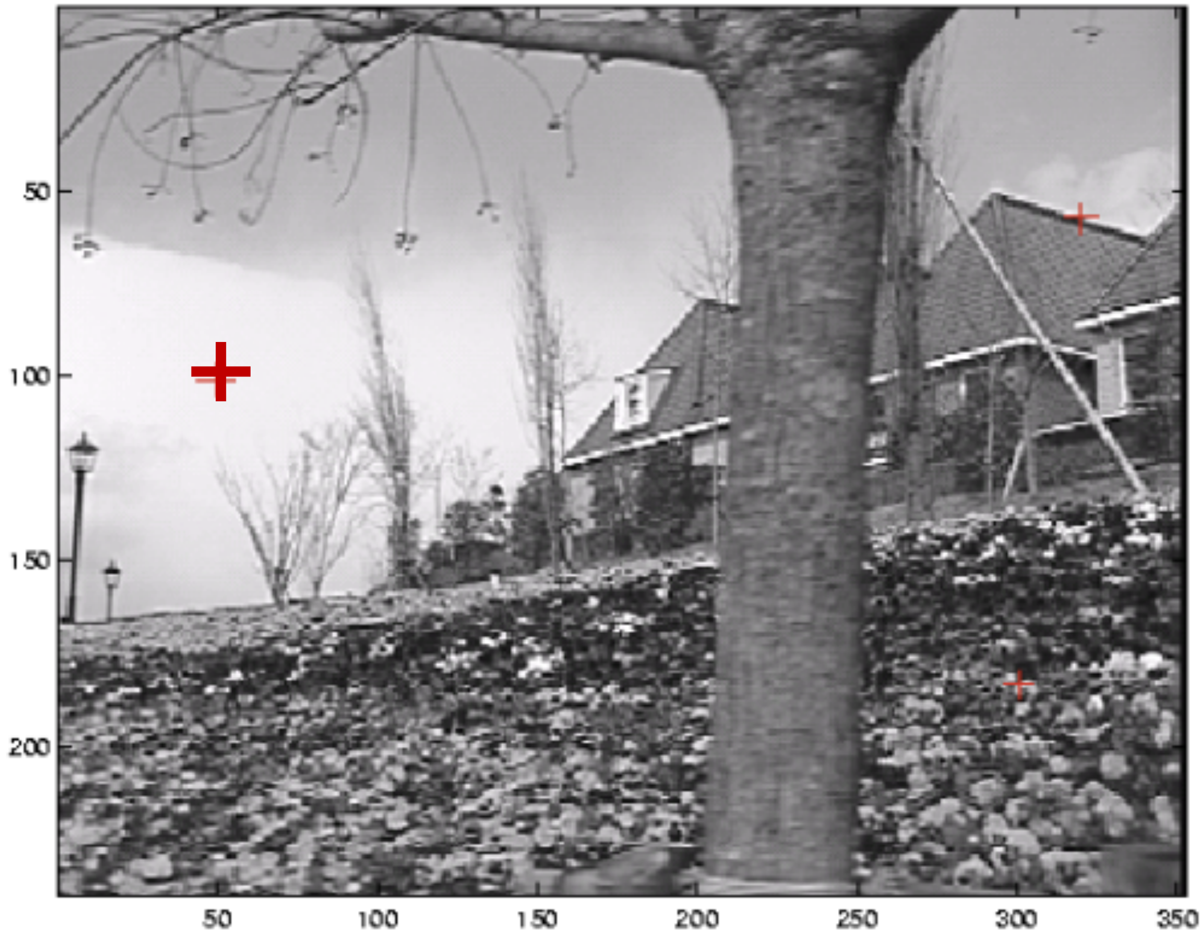
Auto-correlation surfaces

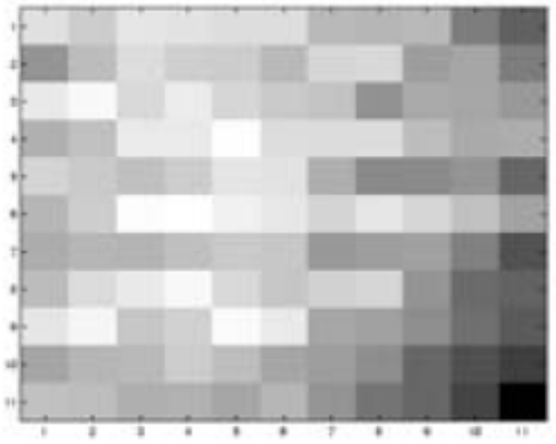


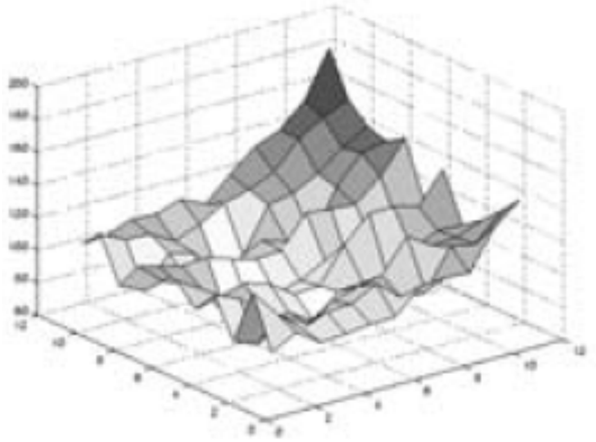




Auto-correlation surfaces







Auto-correlation surfaces

Using a Taylor Series expansion of the image function

$$I_0(\boldsymbol{x}_i+\Delta \boldsymbol{u}) \approx I_0(\boldsymbol{x}_i)+\nabla I_0(\boldsymbol{x}_i)\cdot \ \Delta \boldsymbol{u}$$

we can approximate the auto-correlation surface as

$$\begin{aligned} E_{\mathrm{AC}}(\Delta \boldsymbol{u}) &= \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i+\Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2 \\ &\approx \sum_i w(\boldsymbol{x}_i)[I_0(\boldsymbol{x}_i)+\nabla I_0(\boldsymbol{x}_i)\cdot \Delta \boldsymbol{u} - I_0(\boldsymbol{x}_i)]^2 \\ &= \sum_i w(\boldsymbol{x}_i)[\nabla I_0(\boldsymbol{x}_i)\cdot \Delta \boldsymbol{u}]^2 \\ &= \Delta \boldsymbol{u}^T \boldsymbol{A} \Delta \boldsymbol{u}, \end{aligned}$$

where, $\nabla I_0(\boldsymbol{x}_i) = (\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y})(\boldsymbol{x}_i)$ is the image gradient at x_i .

Auto-correlation surfaces

Calculating the gradient:

Classic Harris detector: [-2 -1 0 1 2] filter.

Modern variants: convolve the image with horizontal and vertical derivatives of a Gaussian (typically with $\sigma = 1$).

Auto-correlation surfaces

The auto-correlation matrix A can be written as

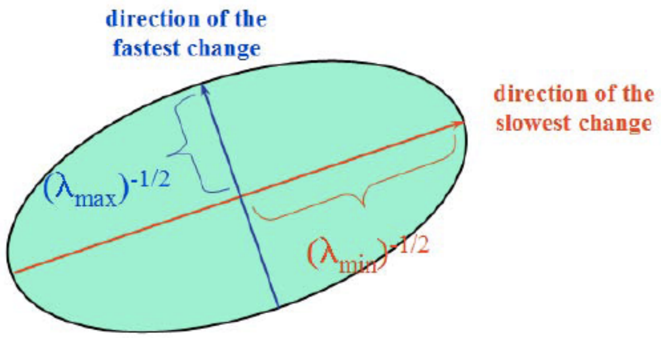
$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

As first shown by Anandan (1984; 1989) that the inverse of the matrix A provides a lower bound on the uncertainty in the location of a matching patch.

It is therefore a useful indicator of which patches can be reliably matched. See examples

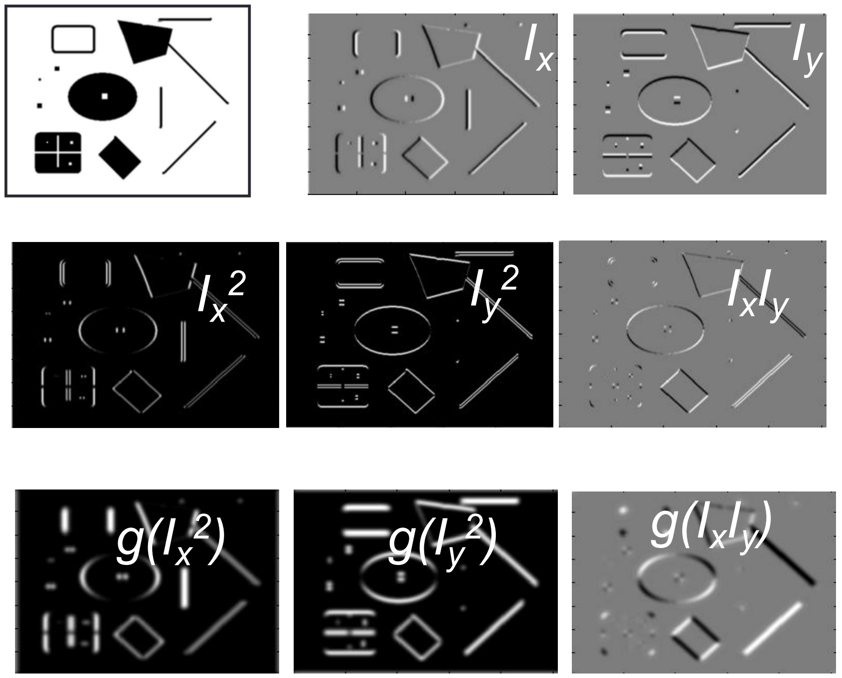
Auto-correlation surfaces

Performing an eigenvalue analysis of the auto-correlation matrix A produces two eigenvalues (λ_0, λ_1) and two eigenvector directions:

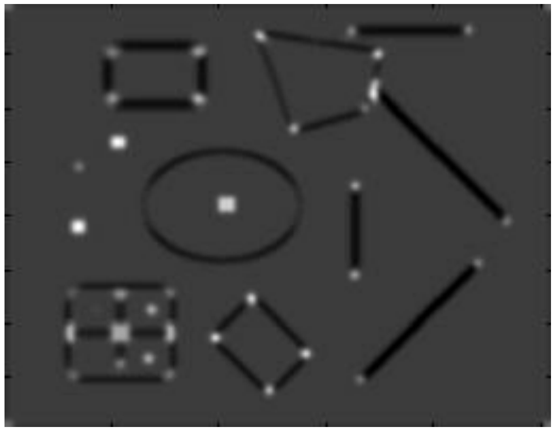

direction of the fastest change
direction of the slowest change
$(\lambda_{max})^{-1/2}$
$(\lambda_{min})^{-1/2}$

Since the larger uncertainty depends on the smaller eigenvalue, e.g. $\lambda_0^{-1/2}$ it makes sense to find maxima in the smaller eigenvalue to locate good features to track (Shi and Tomasi 1994).

Harris Feature detector (Harris 88)

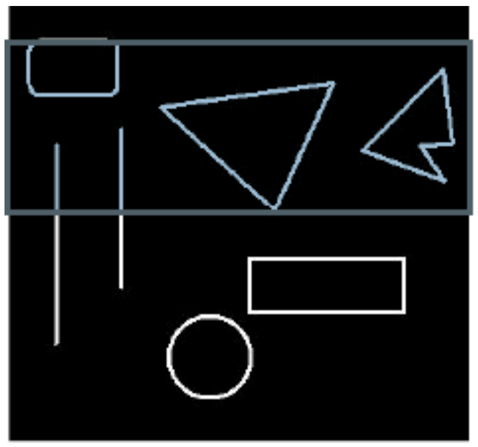


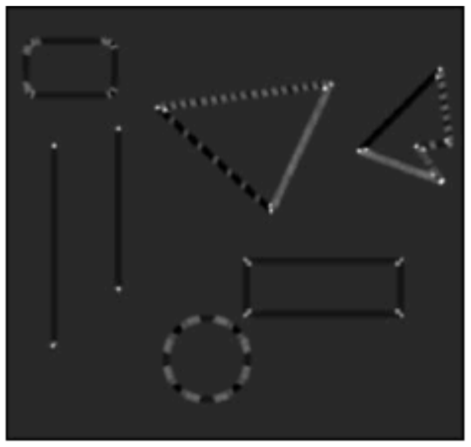
$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$



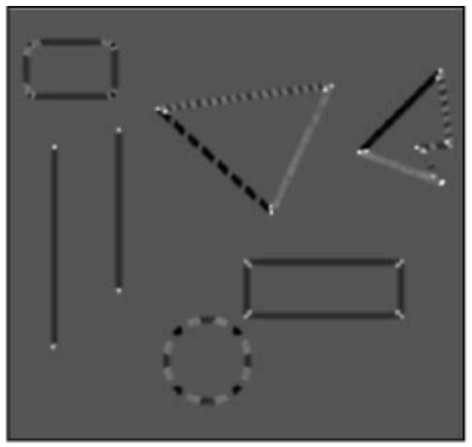
Harr= $\det(\mathbf{A}) - \alpha\, \mathrm{trace}(\mathbf{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$

Cornerness – Harris Corner

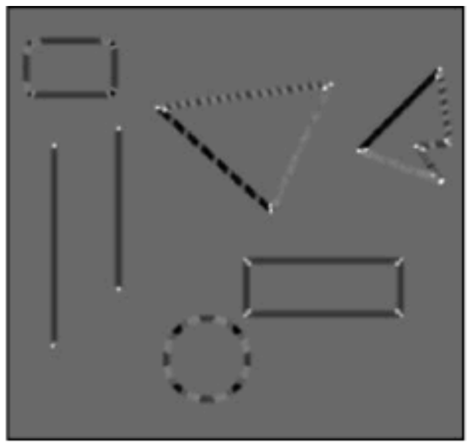




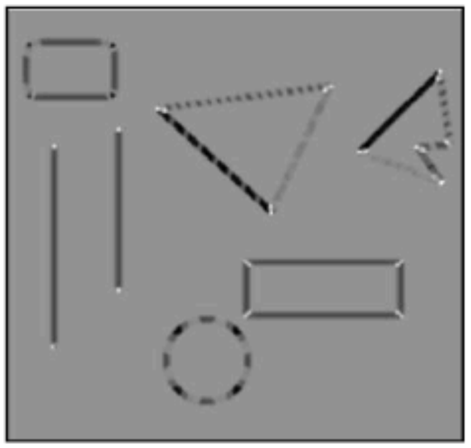
α = .04



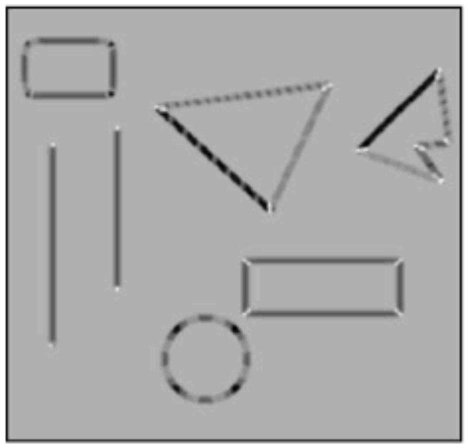
α = .08



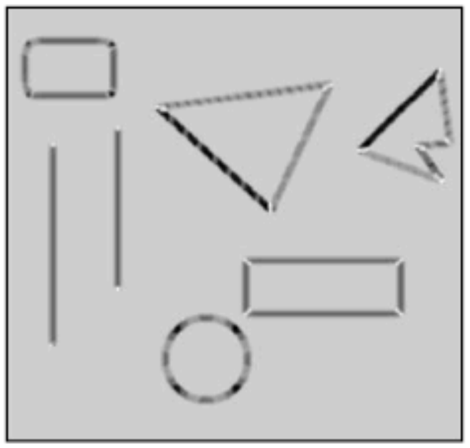
α = .1



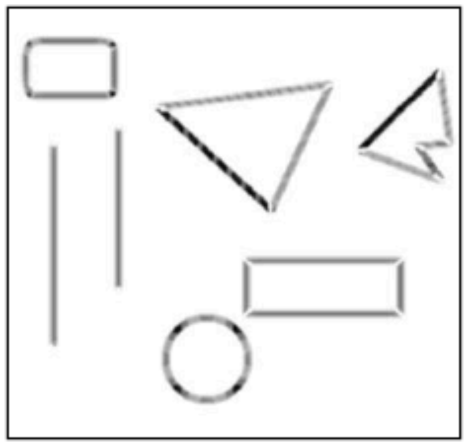
α = .14



α = .17



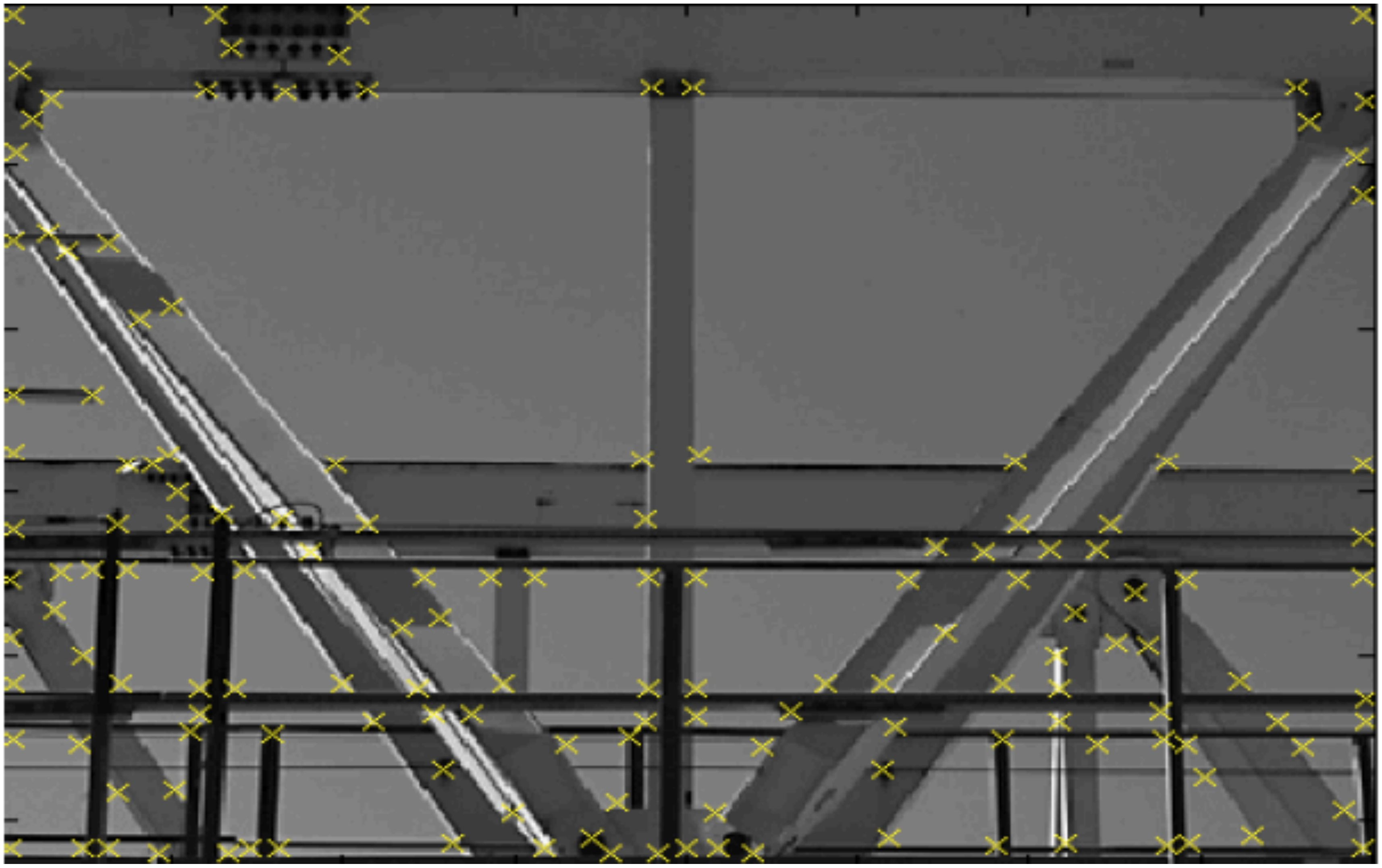
α = .2



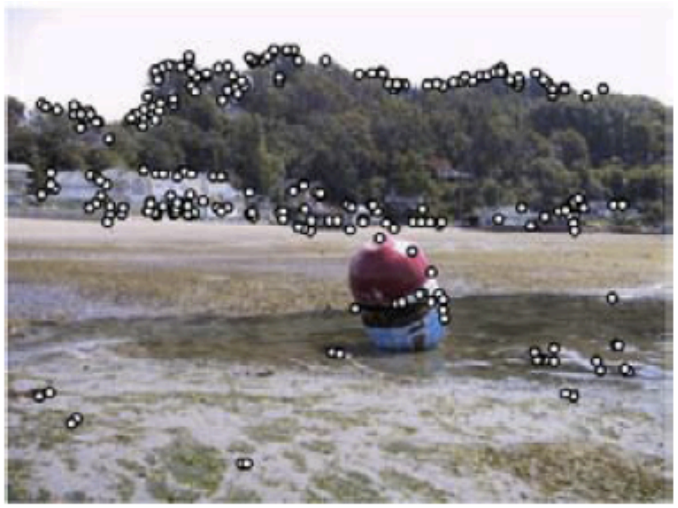
α = .25

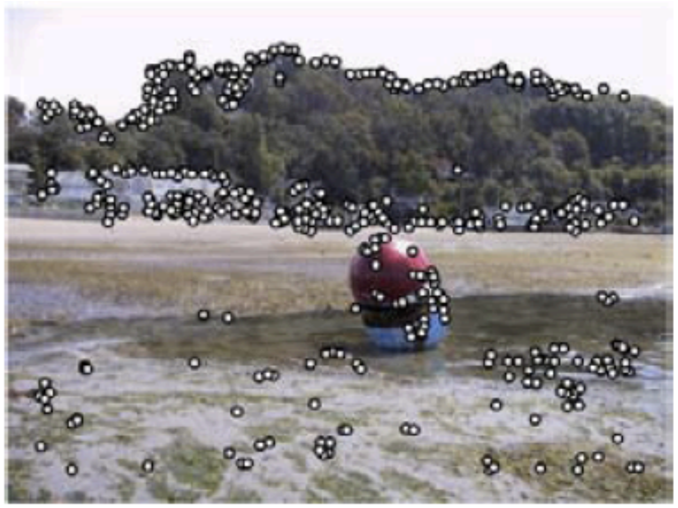
Fei-Fei Li

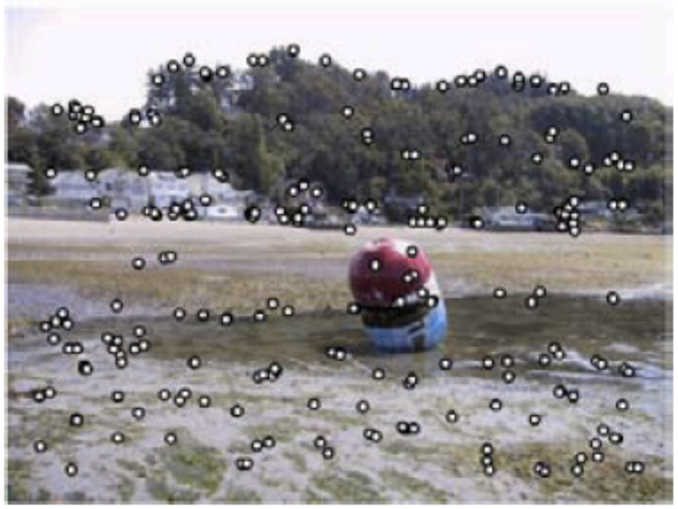
Example: Harris Corner

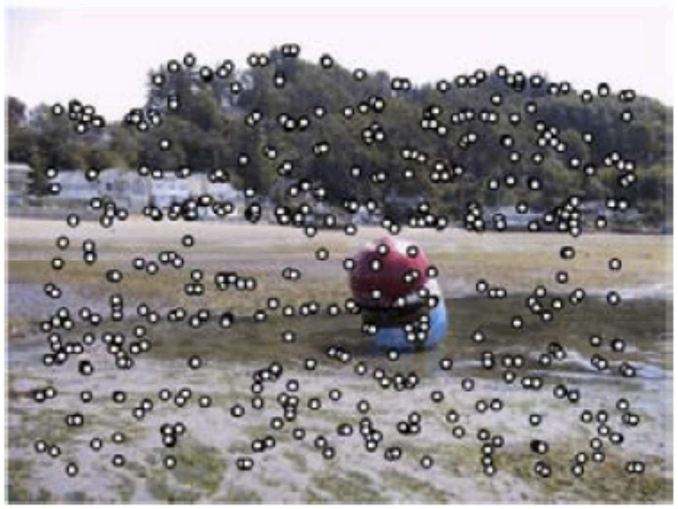


Adaptive non-maximal suppression (ANMS, Brown, Szeliski, and Winder 2005)

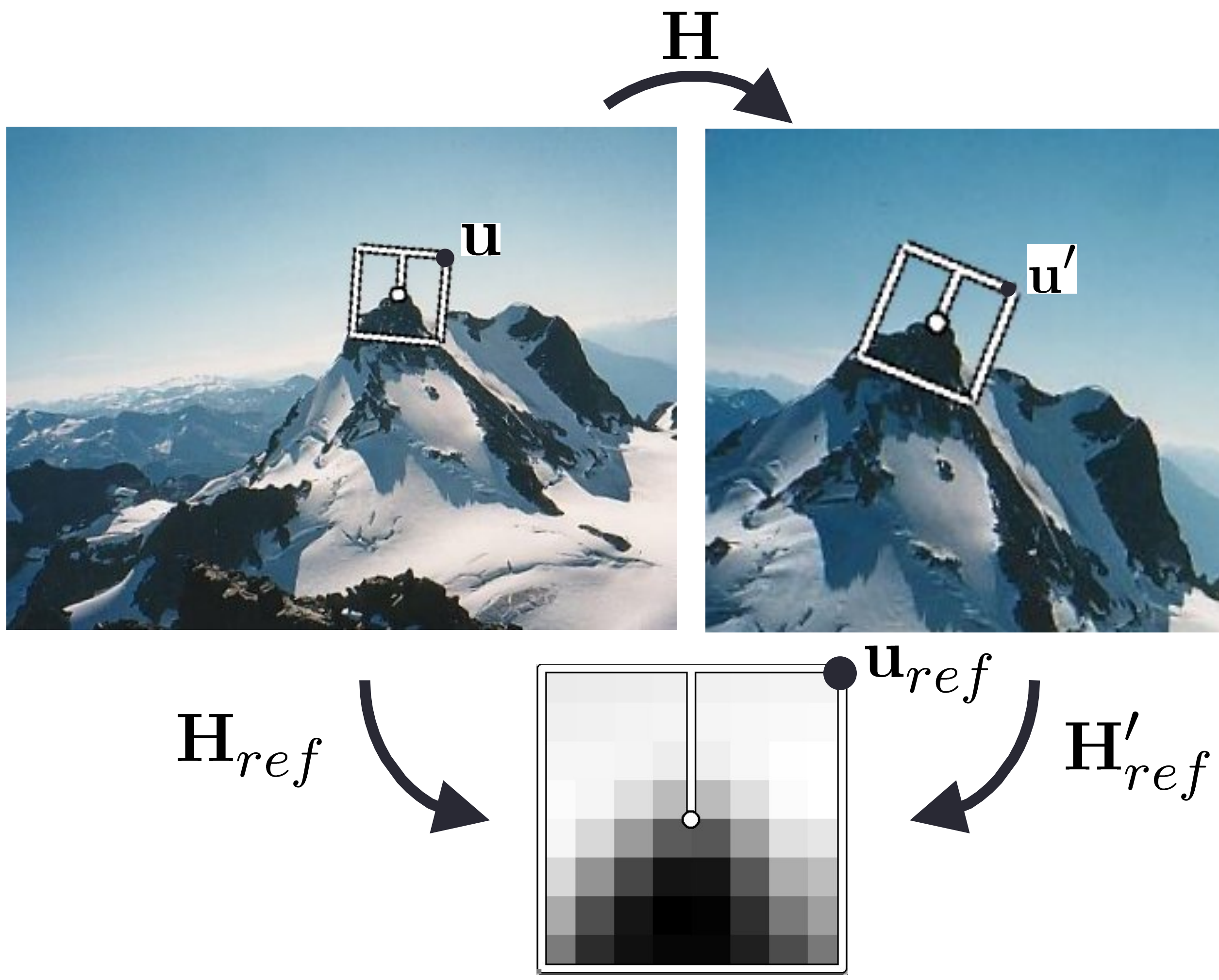

(a) Strongest 250


(b) Strongest 500

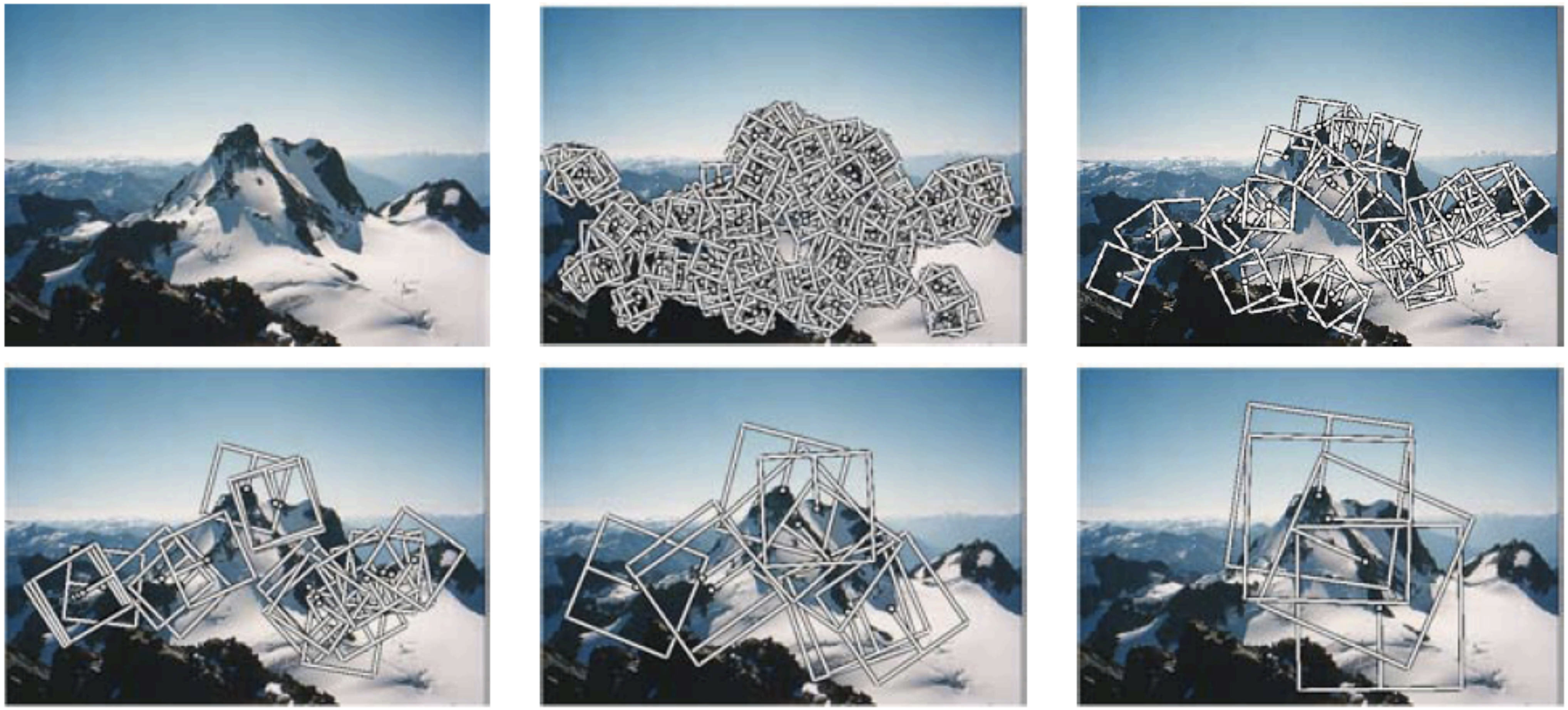

(c) ANMS 250, $r = 24$


(d) ANMS 500, $r = 16$

Rotation Invariance (Brown et al)



Scale Invariance



Multi-scale oriented patches (MOPS) extracted at five pyramid levels (Brown, Szeliski, and Winder 2005). The boxes show the feature orientation and the region from which the descriptor vectors are sampled.

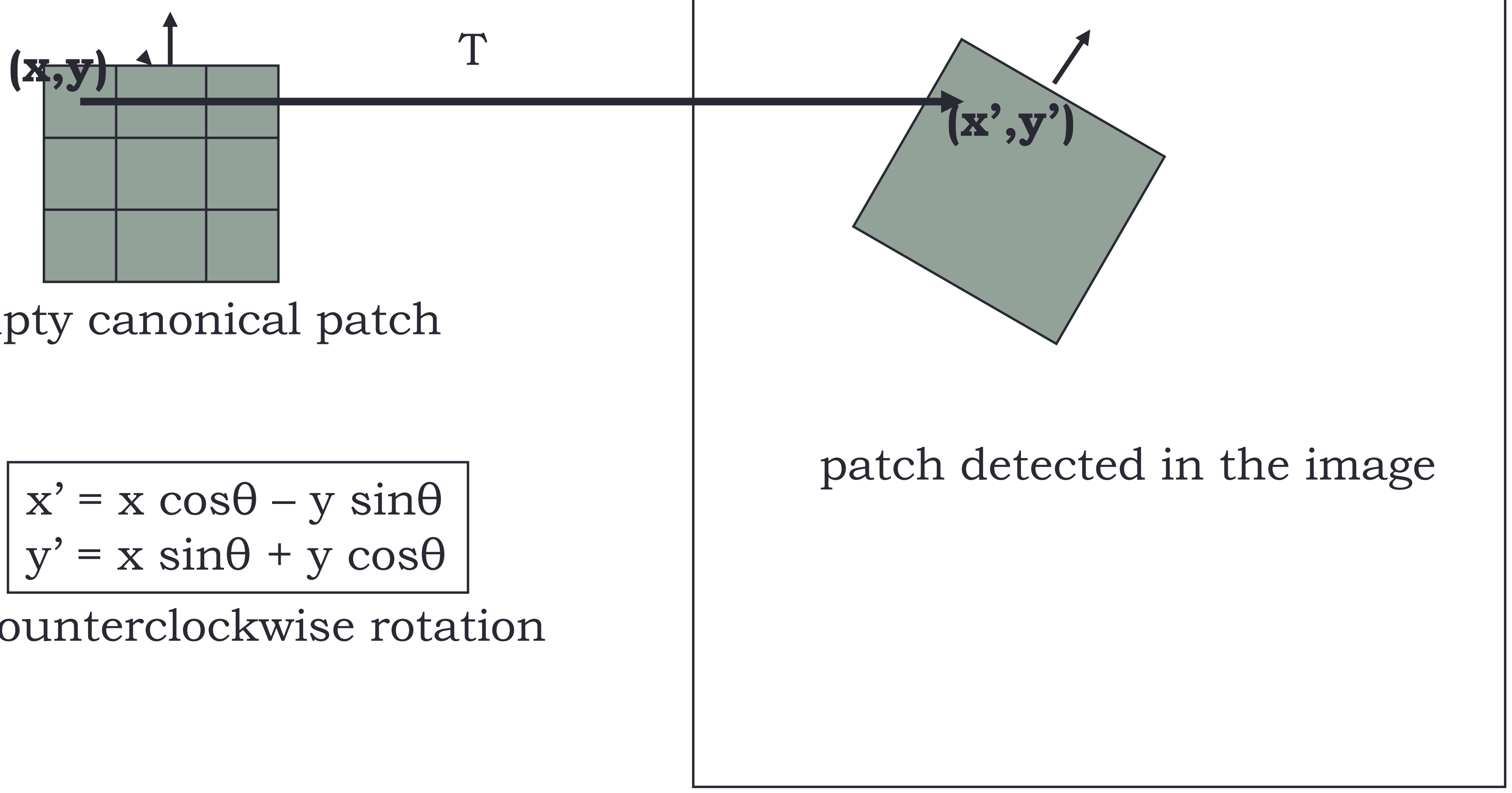
Ideas from Brown’s Multi-Scale Oriented Patches

- 1. Detect an interesting patch with an interest operator. Patches are translation invariant.
- 2. Determine its dominant orientation.
- 3. Rotate the patch so that the dominant orientation points upward. This makes the patches rotation invariant.
- 4. Do this at multiple scales, converting them all to one scale through sampling.
- 5. Convert to illumination “invariant” form

Implementation Concern: How do you rotate a patch?

- Start with an “empty” patch whose dominant direction is “up”.
- For each pixel in your patch, compute the position in the detected image patch. It will be in floating point and will fall between the image pixels.
- Interpolate the values of the 4 closest pixels in the image, to get a value for the pixel in your patch.

Rotating a Patch

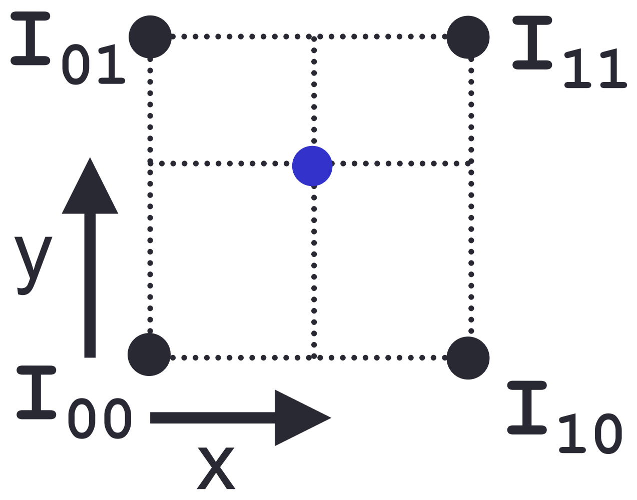


T $\boxed{\begin{array}{l} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{array}}$

counterclockwise rotation

Using Bilinear Interpolation

- Use all 4 adjacent samples



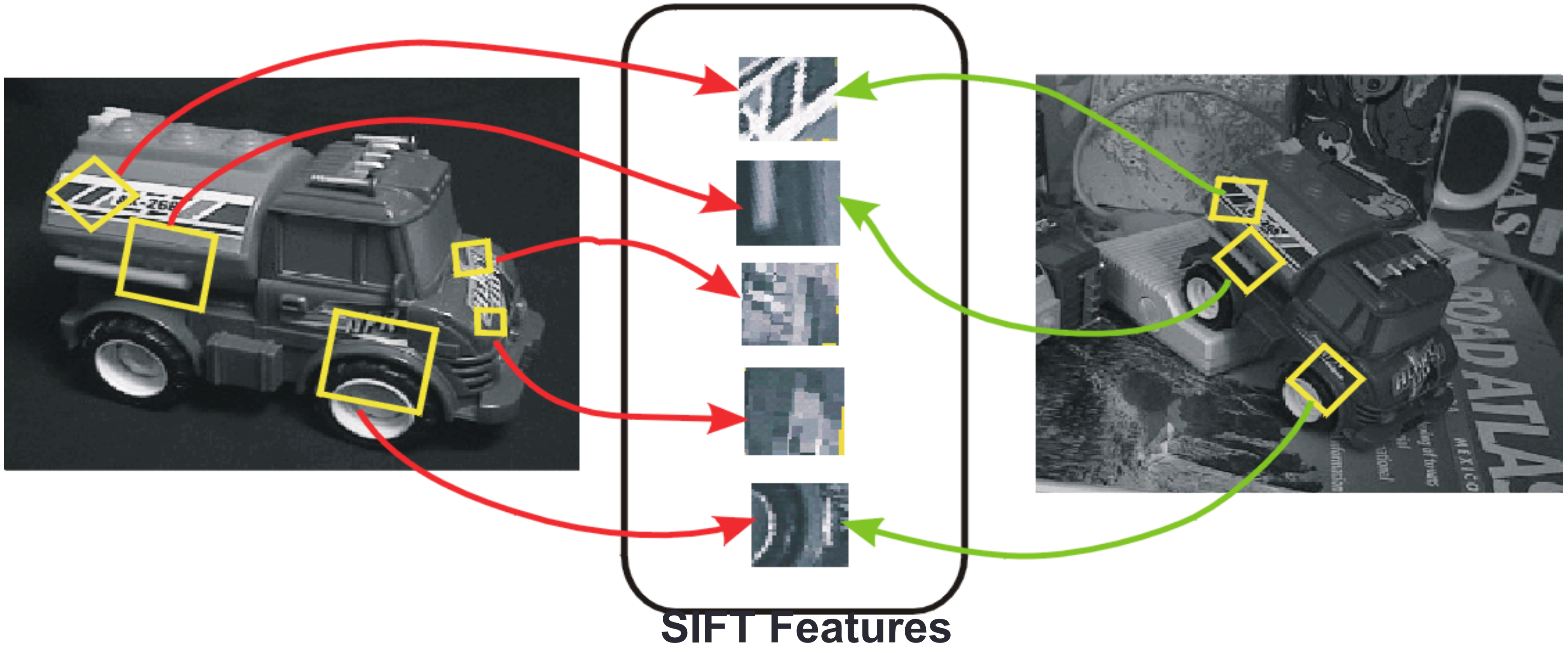
SIFT: Motivation

- The Harris operator is not invariant to scale and correlation is not invariant to rotation[1].
- For better image matching, Lowe's goal was to develop an interest operator that is invariant to scale and rotation.
- Also, Lowe aimed to create a descriptor that was robust to the variations corresponding to typical viewing conditions. The descriptor is the most-used part of SIFT.

[1]But Schmid and Mohr developed a rotation invariant descriptor for it in 1997.

Idea of SIFT

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



Claimed Advantages of SIFT

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

Overall Procedure at a High Level

1. Scale-space extrema detection

   Search over multiple scales and image locations.

2. Keypoint localization

   Fit a model to detrmine location and scale.
   Select keypoints based on a measure of stability.

3. Orientation assignment

   Compute best orientation(s) for each keypoint region.

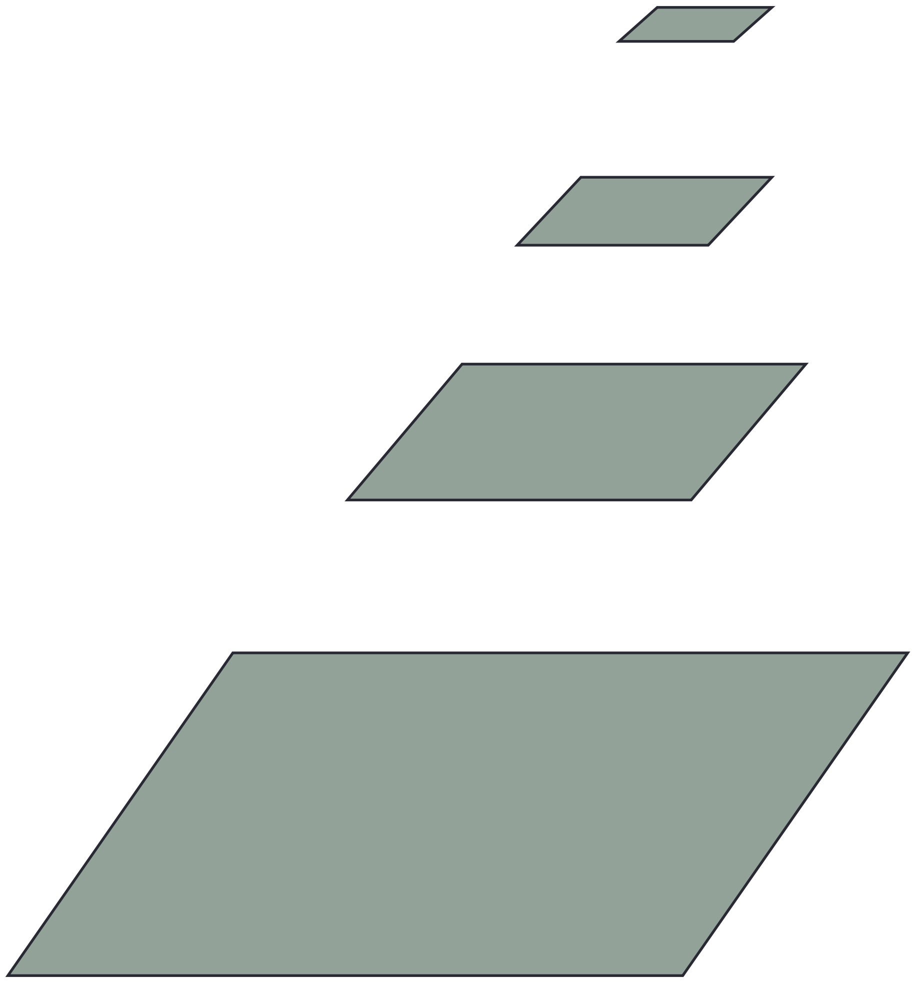
4. Keypoint description

   Use local image gradients at selected scale and rotation to describe each keypoint region.

1. Scale-space extrema detection

- Goal: Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- Method: search for stable features across multiple scales using a continuous function of scale.
- Prior work has shown that under a variety of assumptions, the best function is a Gaussian function.
- The scale space of an image is a function $L(x,y,\sigma)$ that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

Aside: Image Pyramids



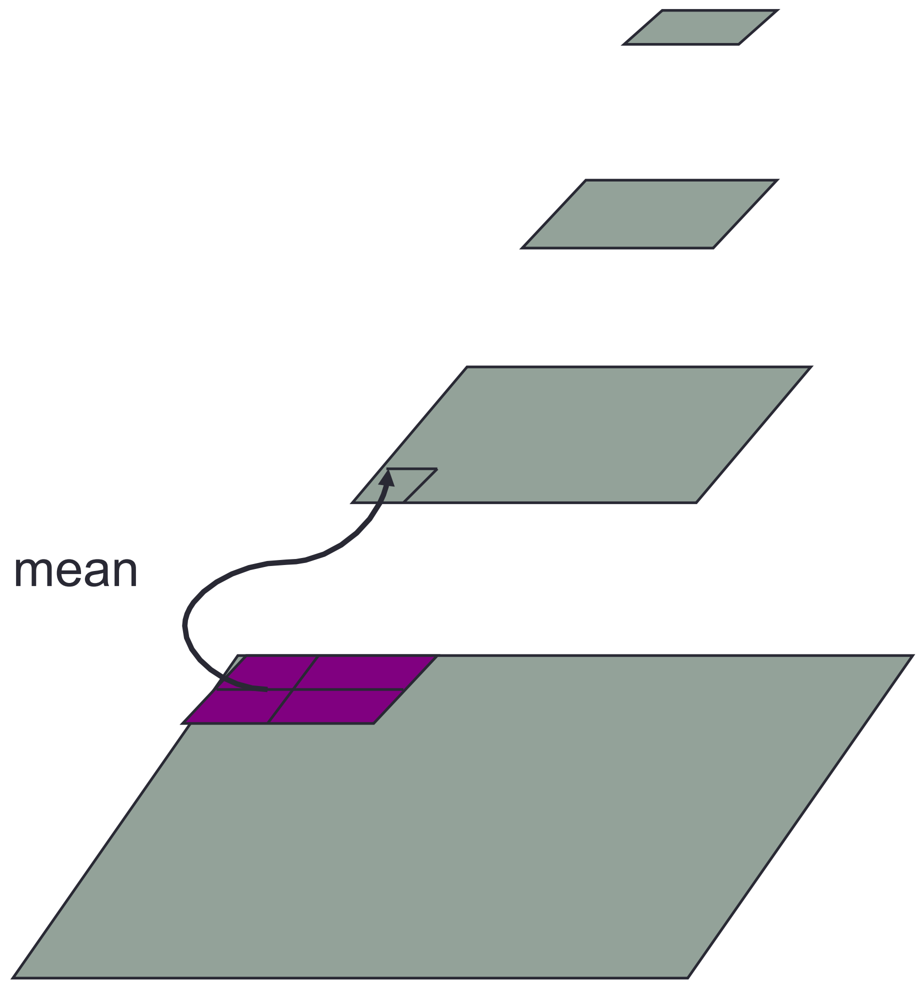
And so on.

3rd level is derived from the 2nd level according to the same funtion

2nd level is derived from the original image according to some function

Bottom level is the original image.

Aside: Mean Pyramid



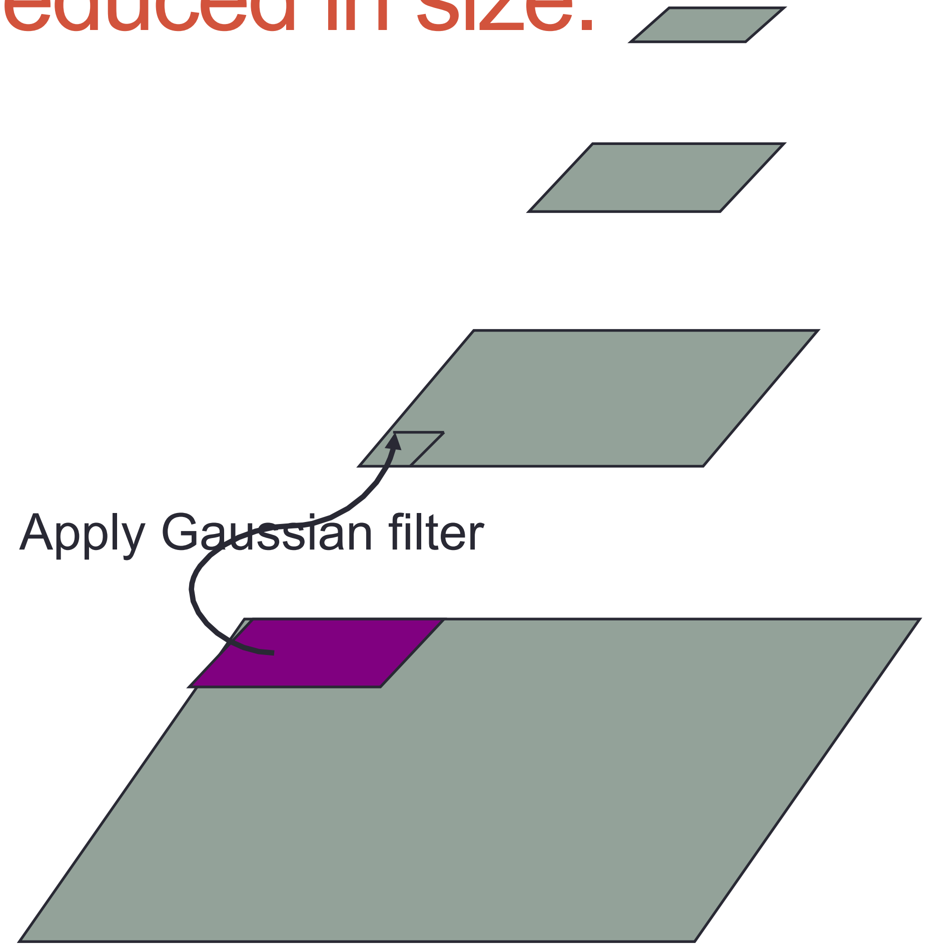
And so on.

At 3rd level, each pixel is the mean of 4 pixels in the 2nd level.

At 2nd level, each pixel is the mean of 4 pixels in the original image.

Bottom level is the original image.

Aside: Gaussian Pyramid
At each level, image is smoothed and reduced in size.

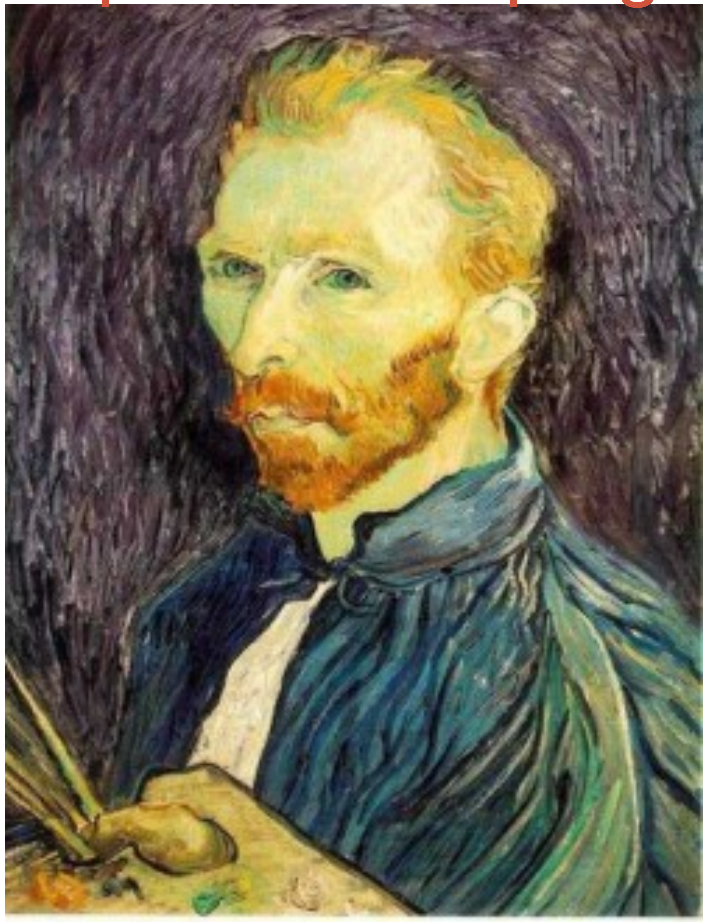


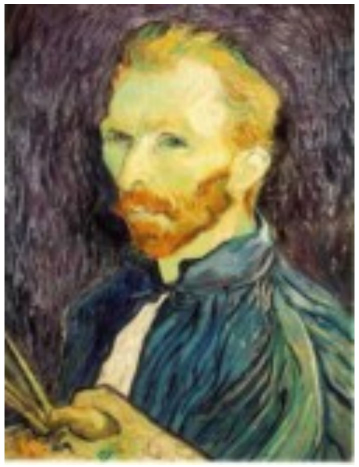
And so on.

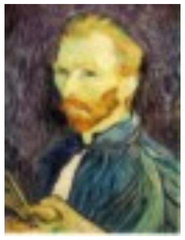
At 2nd level, each pixel is the result of applying a Gaussian mask to the first level and then subsampling to reduce the size.

Bottom level is the original image.

Example: Subsampling with Gaussian pre-filtering

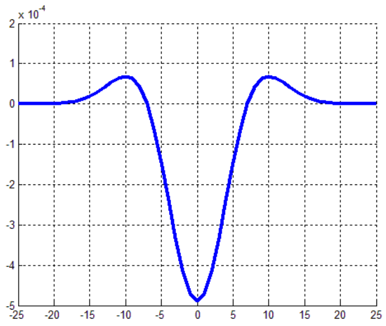

Gaussian 1/2

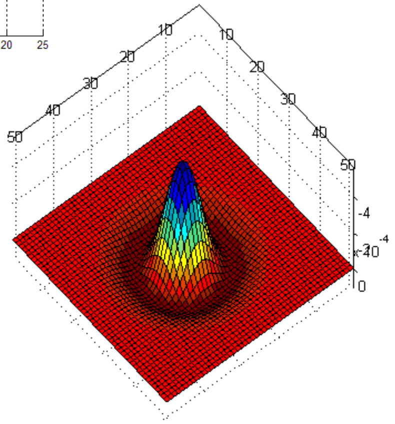

G 1/4

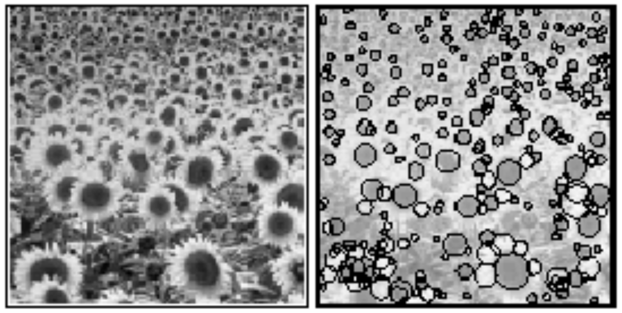

G 1/8

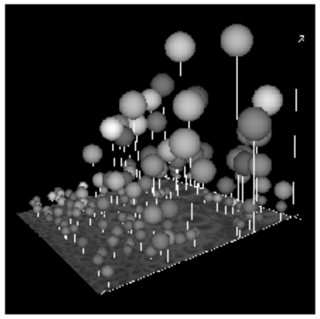
Lowe’s Scale-space Interest Points

- **Laplacian of Gaussian** kernel
  - Scale normalised (x by scale^2)
  - Proposed by Lindeberg
- Scale-space detection
  - Find local maxima across scale/space
  - A good “blob” detector







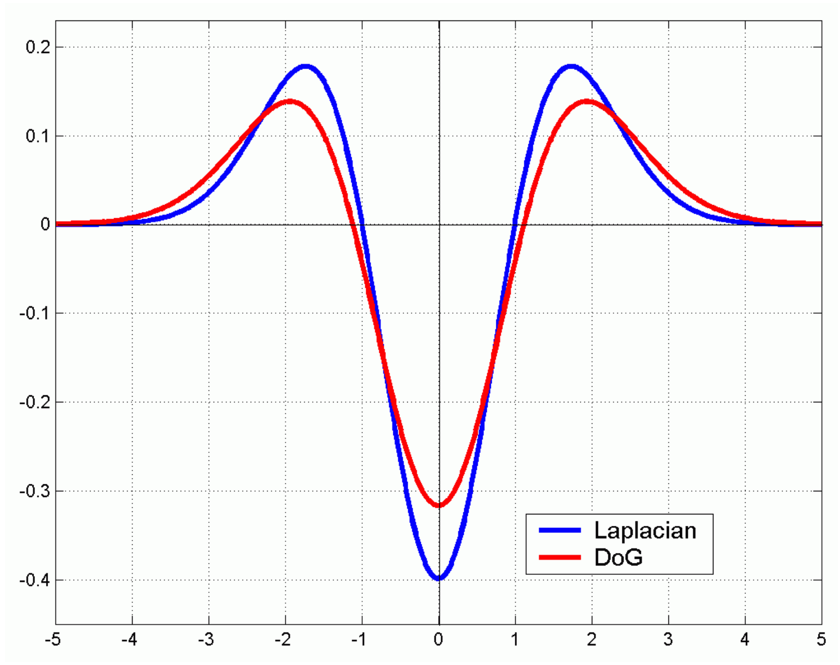


$$G(x,y,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

$$\nabla^2 G(x,y,\sigma) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

[T. Lindeberg IJCV 1998]

Lowe's Scale-space Interest Points: Difference of Gaussians



- Gaussian is an ad hoc solution of heat diffusion equation

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G.$$

- Hence

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G.$$

- k is not necessarily very small in practice

Lowe's Pyramid Scheme

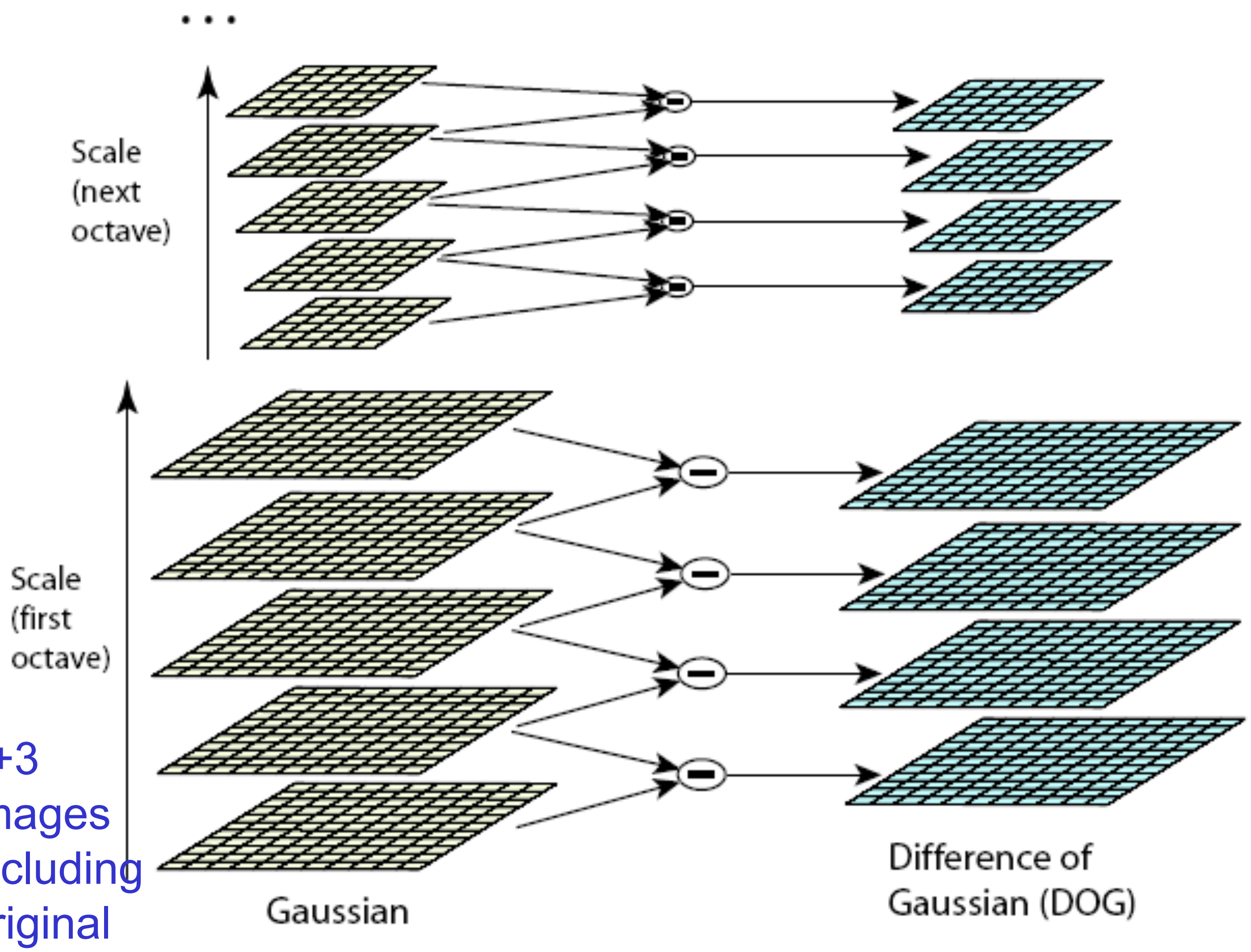
- Scale space is separated into octaves:
  - Octave 1 uses scale σ
  - Octave 2 uses scale 2σ
  - etc.

- In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images.

- Adjacent Gaussians are subtracted to produce the DOG

- After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image ¼ the size to start the next level.

Lowe's Pyramid Scheme



s+2 filters

$\sigma_{s+1}=2^{(s+1)/s}\sigma_0$

.

.

$\sigma_i=2^{i/s}\sigma_0$

.

.

$\sigma_2=2^{2/s}\sigma_0$

$\sigma_1=2^{1/s}\sigma_0$

σ_0

s+3 images including original

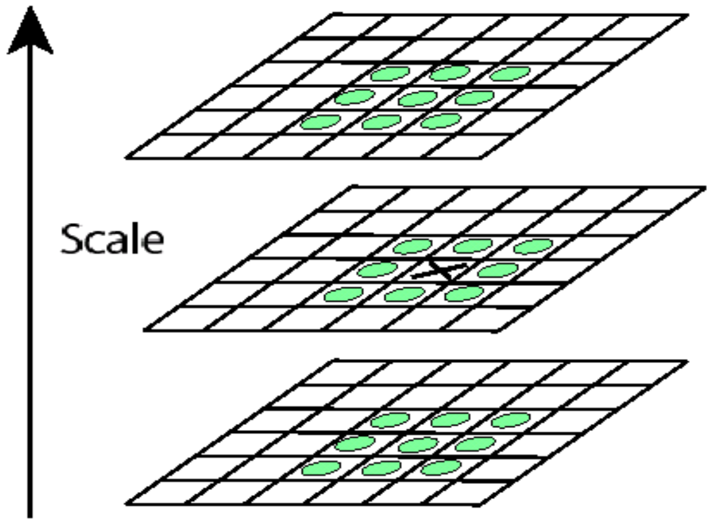
s+2 differ-ence images

The parameter **s** determines the number of images per octave.

Key point localization

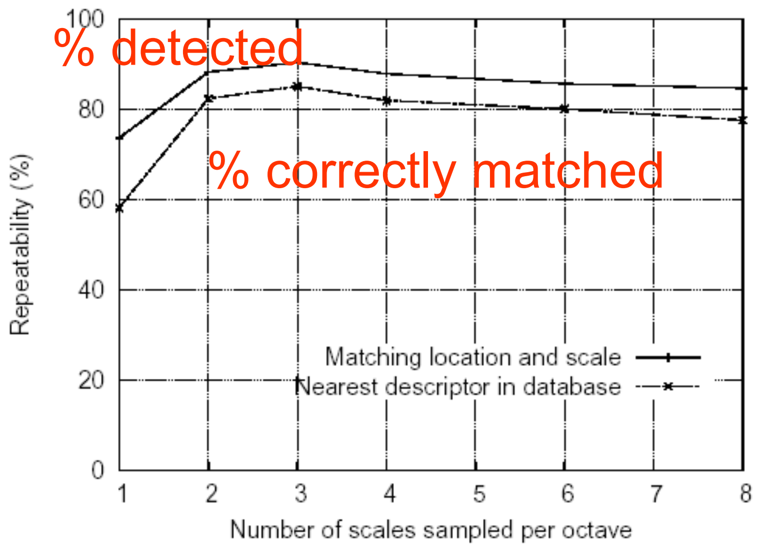
s+2 difference images.
top and bottom ignored.
s planes searched.

- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

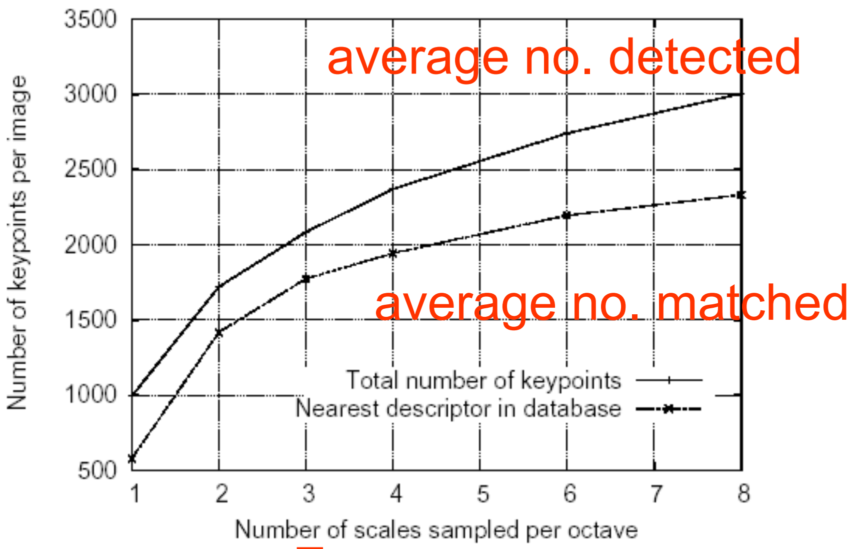


For each max or min found, output is the **location** and the **scale**.

Scale-space extrema detection: experimental results over 32 images that were synthetically transformed and noise added.



Stability



Expense

- Sampling in scale for efficiency
  - How many scales should be used per octave? S=?
    - More scales evaluated, more keypoints found
    - S < 3, stable keypoints increased too
    - S > 3, stable keypoints decreased
    - S = 3, maximum stable keypoints found

2. Keypoint localization

- Once a keypoint candidate is found, perform a detailed fit to nearby data to determine
  - location, scale, and ratio of principal curvatures
- In initial work keypoints were found at location and scale of a central sample point.
- In newer work, they fit a 3D quadratic function to improve interpolation accuracy.
- The Hessian matrix was used to eliminate edge responses.

Eliminating the Edge Response

- Reject flats:
  - $|D(\hat{\mathbf{x}})| < 0.03$
- Reject edges:

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

Let α be the eigenvalue with larger magnitude and β the smaller.

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

Let $r = \alpha/\beta$.
So $\alpha = r\beta$

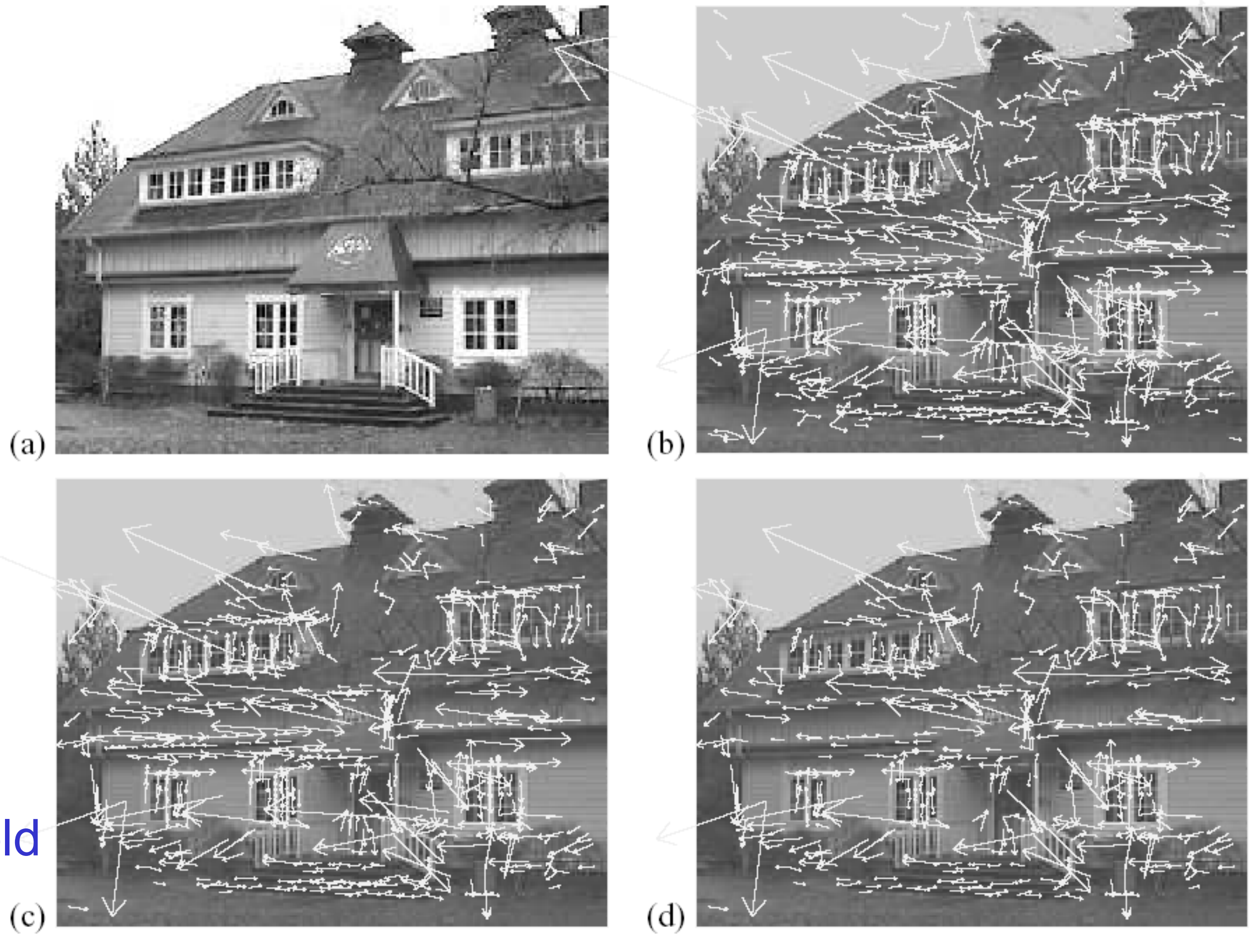
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

  - $r < 10$
- What does this look like?

Keypoint localization with orientation

233x189



832

initial keypoints

729

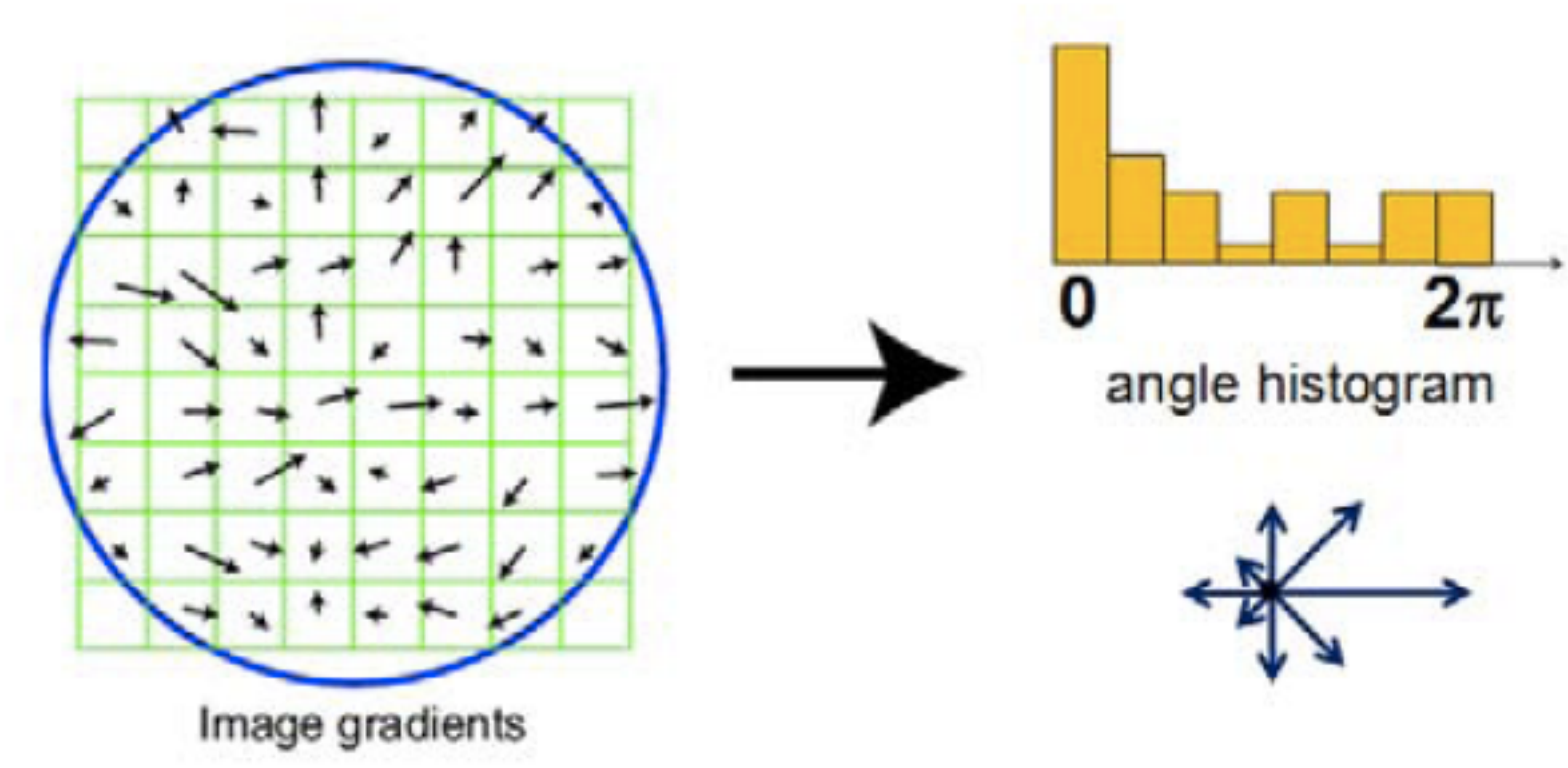
keypoints after gradient threshold

536

keypoints after ratio threshold

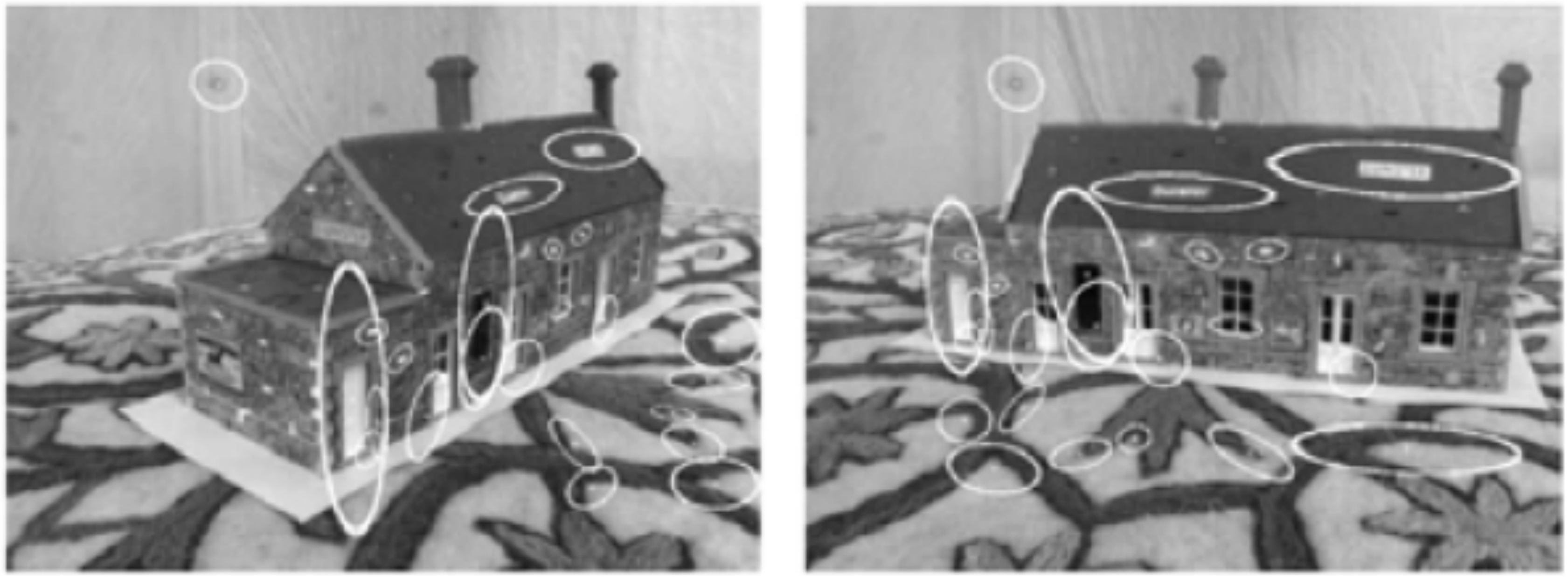
3.Orientation estimation

If 2 major orientations, use both.



Create histogram of local gradient directions at selected scale
Assign canonical orientation at peak of smoothed histogram
Each key specifies stable 2D coordinates (x, y, scale,orientation)

Affine Invariance (not SIFT)

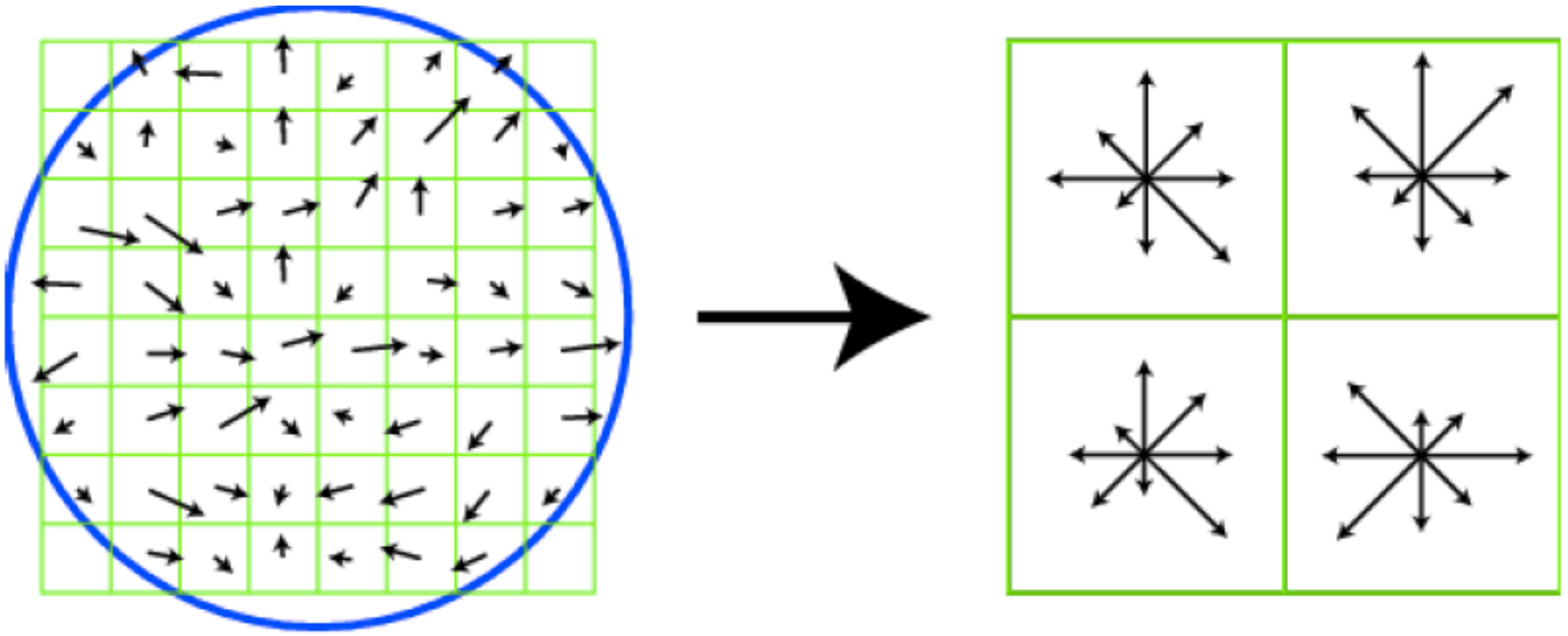


Affine region detectors used to match two images taken from dramatically different viewpoints (Mikolajczyk and Schmid 2004)

4. Keypoint Descriptors

- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

SIFT



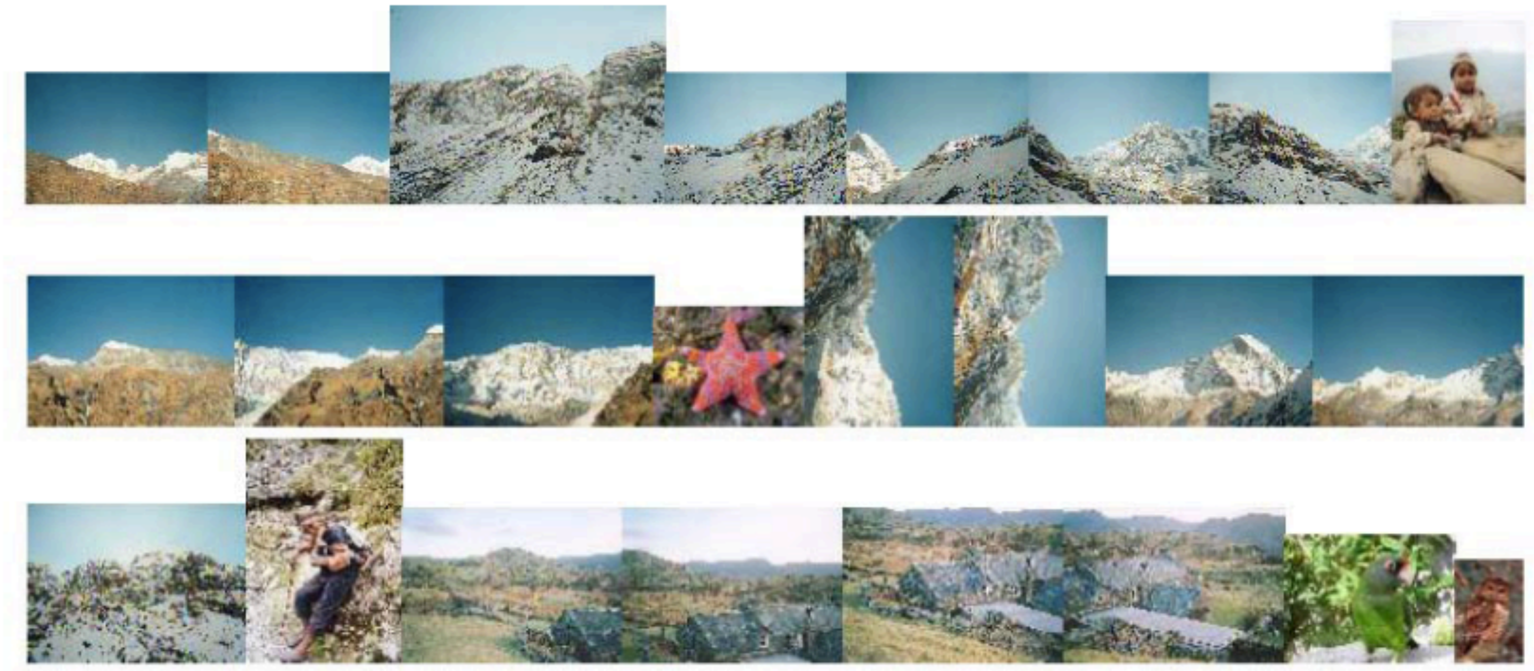
A schematic representation of Lowe's (2004) scale invariant feature transform (SIFT): (a) Gradient orientations and magnitudes are computed at each pixel and weighted by a Gaussian fall-off function (blue circle). (b) A weighted gradient orientation histogram is then computed in each sub-region, using trilinear interpolation. While this figure shows an 8 X8 pixel patch and a 2X2 descriptor array, Lowe's actual implementation uses 16X16 patches and a 4 4 array of eight-bin histograms.

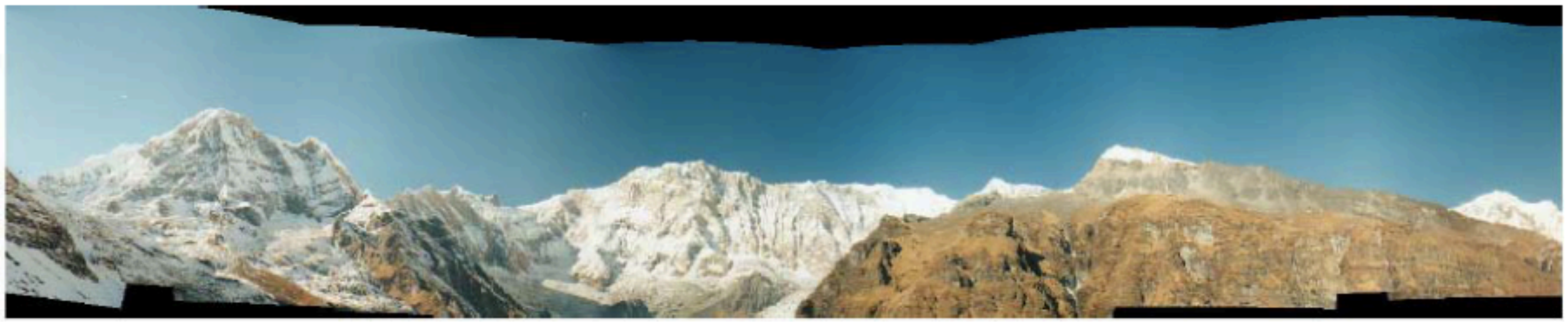
SIFT Keypoint Descriptor

- use the normalized region about the keypoint
- compute gradient magnitude and orientation at each point in the region
- weight them by a Gaussian window overlaid on the circle
- create an orientation histogram over the 4 X 4 subregions of the window
- 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives a vector of 128 values.

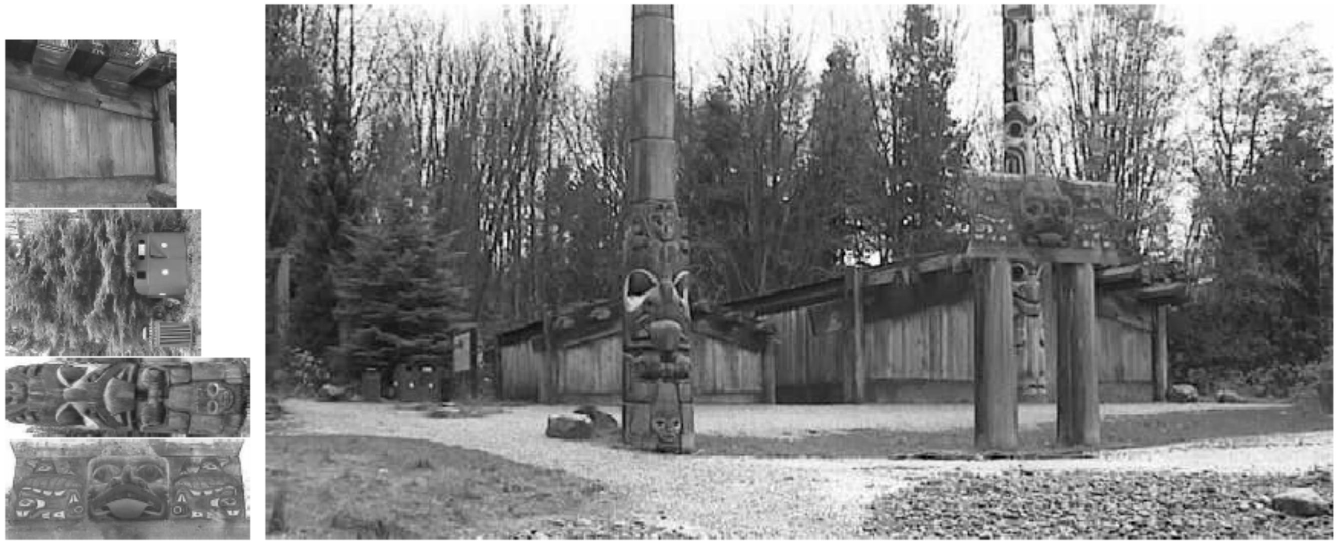
SIFT Results

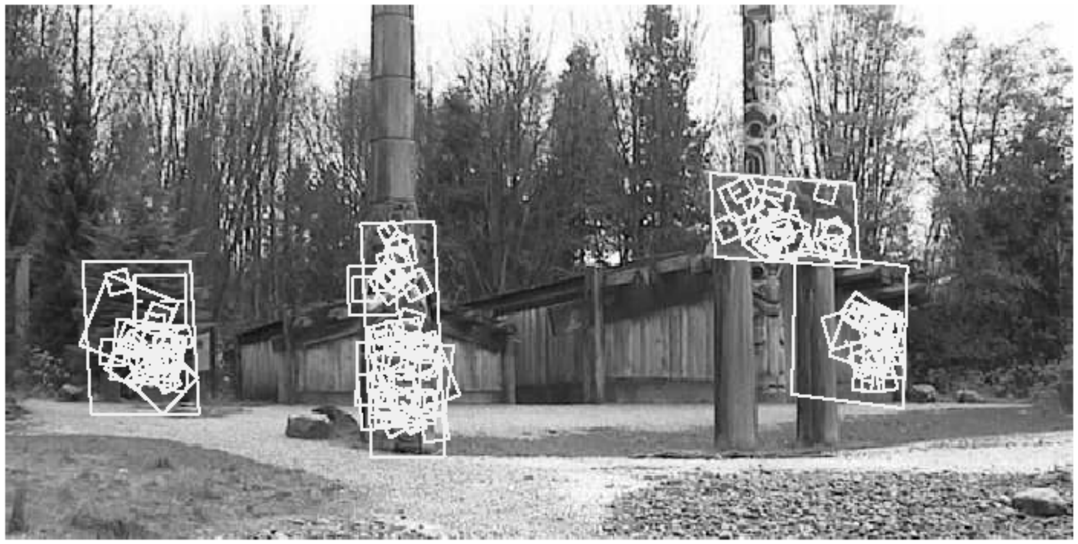
Panorama



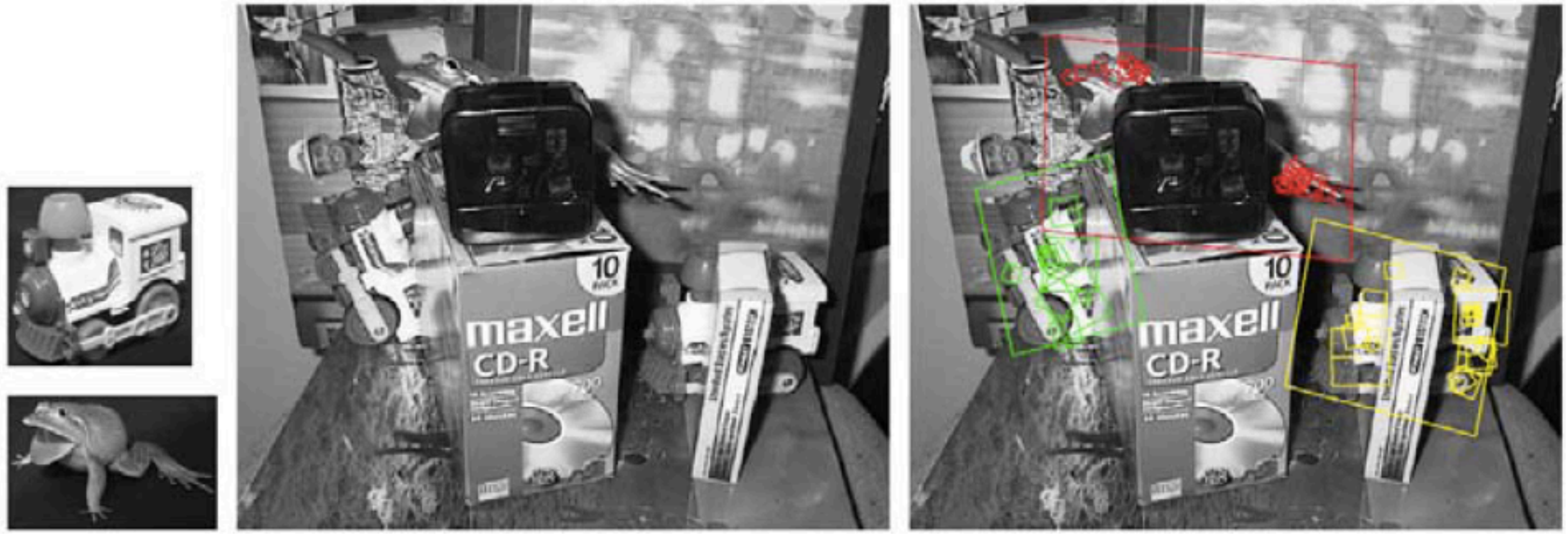


SIFT Results: Matching “Objects”





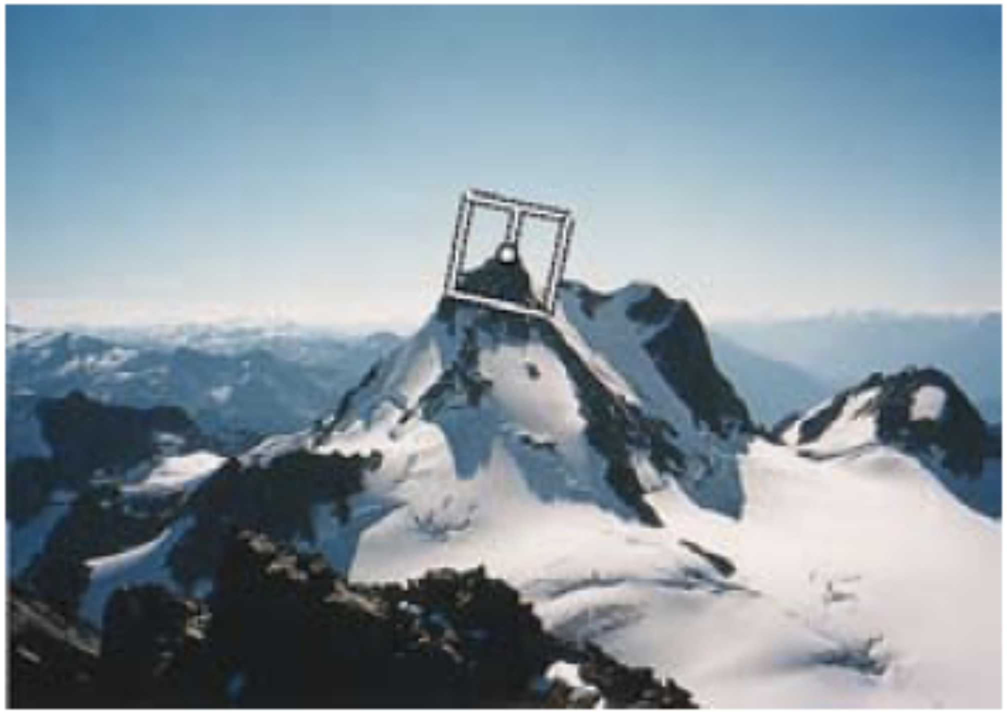
SIFT Results: Recognizing objects in clutter scenes

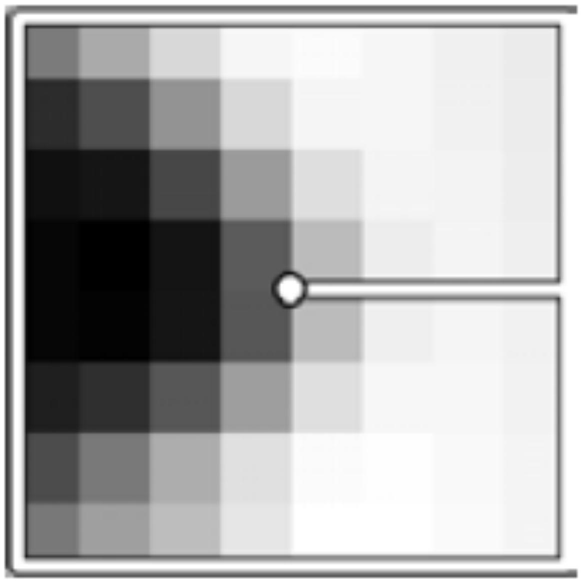


Feature Descriptors (other than SIFT)

- Multiscale Oriented Patches (MOPs).
- Scale invariant feature transform (MSERs)
- PCA-SIFT
- Gradient location-orientation histogram (GLOH).
- Histograms of Oriented Gradients (HOGs)
- Speeded Up Robust Features (SURF)
- and many others …
- (e.g. BRISK)

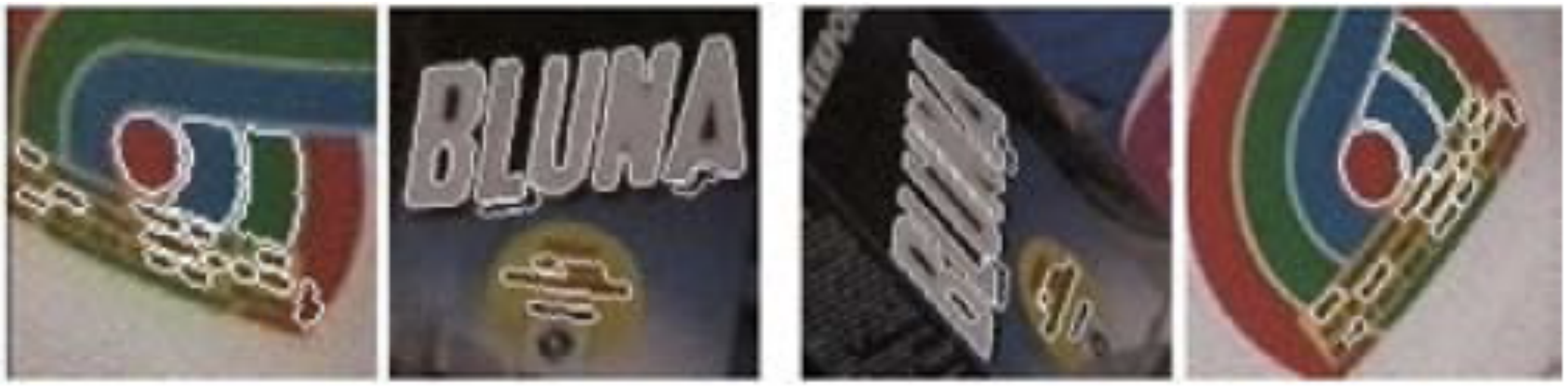
MOPs Descriptors





MOPS descriptors are formed using an 8 x8 sampling of bias and gain normalized intensity values, with a sample spacing of five pixels relative to the detection scale. This low frequency sampling gives the features some robustness to interest point location error and is achieved by sampling at a higher pyramid level than the detection scale.

Maximally stable extremal regions (MSERs)



MSER

Binary regions are computed by thresholding the image at all possible gray levels

- This operation can be performed efficiently by
- first sorting all pixels by gray value and then incrementally adding pixels to each connected component

As the threshold is changed, the area of each component (region) is monitored; regions whose rate of change of

area with respect to the threshold is minimal are defined as maximally stable.

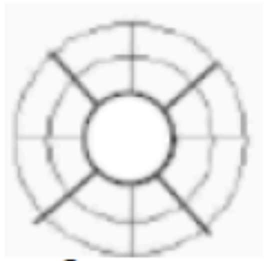
Matal et al, 2004

Gradient location-orientation histogram (GLOH) descriptor

First 3 steps – same as SIFT

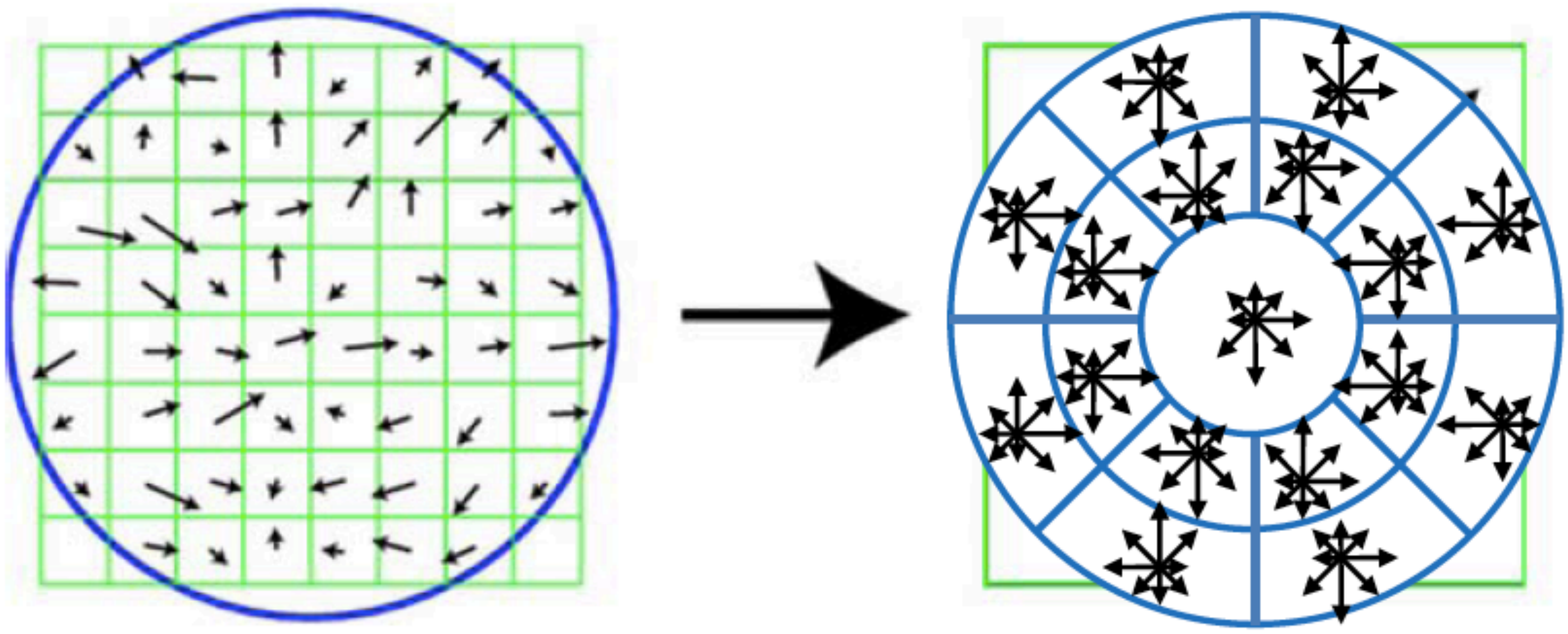
Step 4 – Local image descriptor

- **Consider log-polar location grid with 3 different radii and 8 angular direction for two of them, in total 17 location bin**
- **Form histogram of gradients having 16 bins**
- **Form a feature vector of 272 dimension (17*16)**
- **Perform dimensionality reduction and project the features to a 128 dimensional space.**



Mikolajczyk
and Schmid (2005),

Gradient location-orientation histogram (GLOH) descriptor


(a) image gradients (b) keypoint descriptor

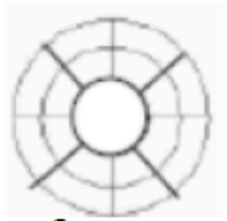
The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).

GLOH

First 3 steps – same as SIFT

Step 4 – Local image descriptor

- Consider log-polar location grid with 3 different radii and 8 angular direction for two of them, in total 17 location bin
- Form histogram of gradients having 16 bins
- Form a feature vector of 272 dimension (17*16)
- Perform dimensionality reduction and project the features to a 128 dimensional space.



192 correct matches (yellow) and 208 false matches (blue).

Histogram of Oriented Gradients Descriptors (Hogs)

- Local object appearance and shape within an image are described by the distribution of intensity gradients or edge directions.
- The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled.
- The descriptor is the concatenation of these histograms.
- For improved accuracy, the local histograms are
- contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block.
- This normalization results in better invariance to changes in illumination and shadowing.

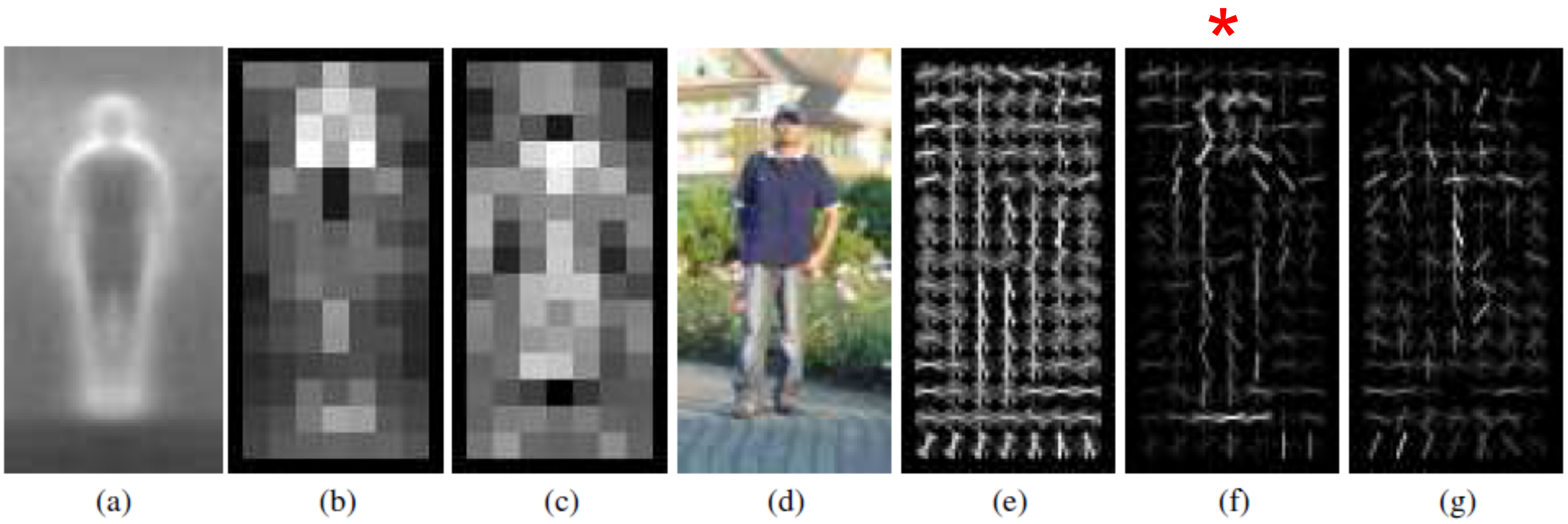
HOGs – Block Normalization

L2-norm: $f = \dfrac{v}{\sqrt{\|v\|_2^2 + e^2}}$

L1-norm: $f = \dfrac{v}{(\|v\|_1 + e)}$

L1-sqrt: $f = \sqrt{\dfrac{v}{(\|v\|_1 + e)}}$

Hogs

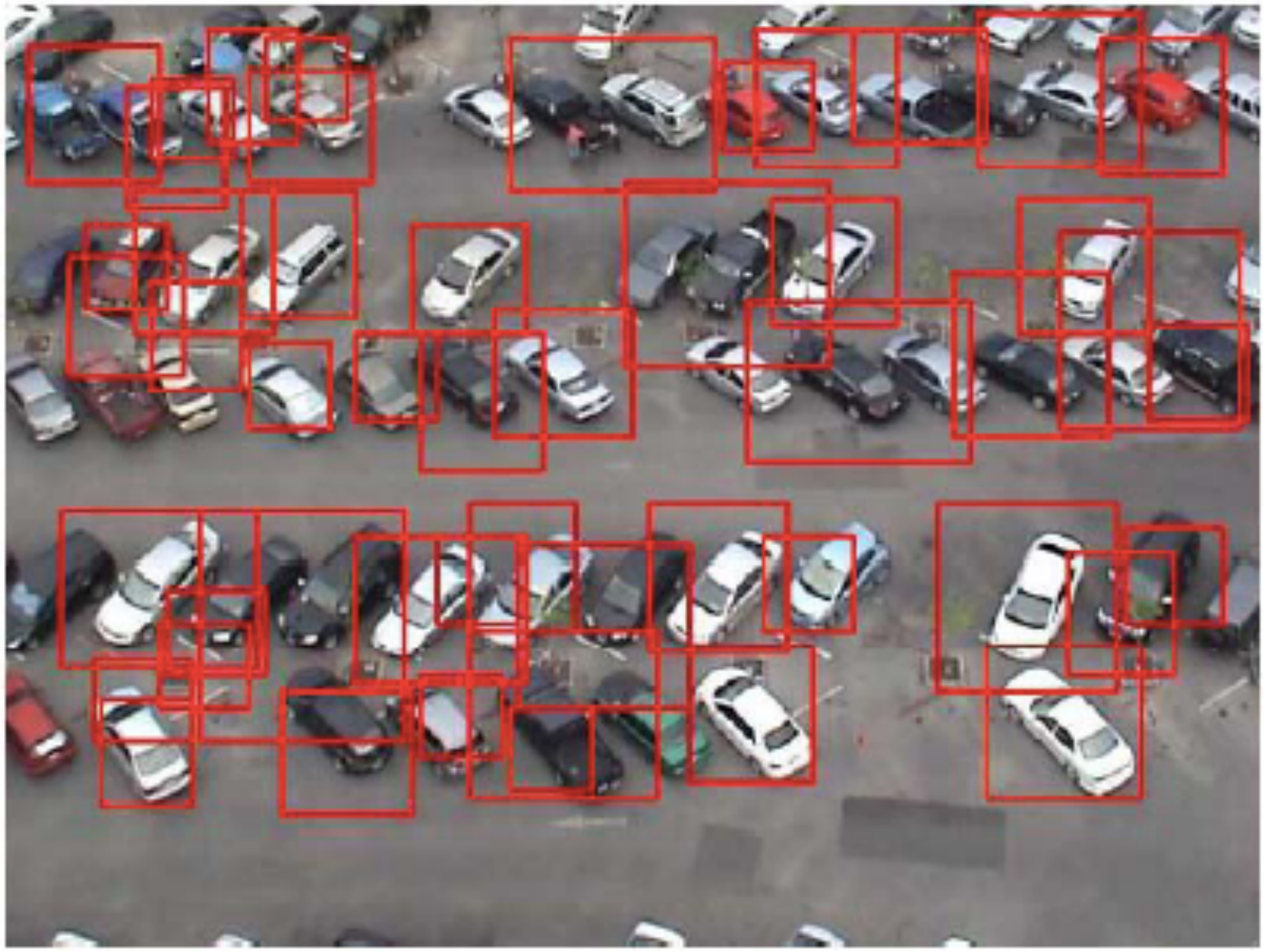


(a) average gradient image over training examples
(b) each "pixel" shows max positive SVM weight in the block centered on that pixel
(c) same as (b) for negative SVM weights
(d) test image
(e) its R-HOG descriptor
(f) R-HOG descriptor weighted by positive SVM weights
(g) R-HOG descriptor weighted by negative SVM weights

HOGs Examples

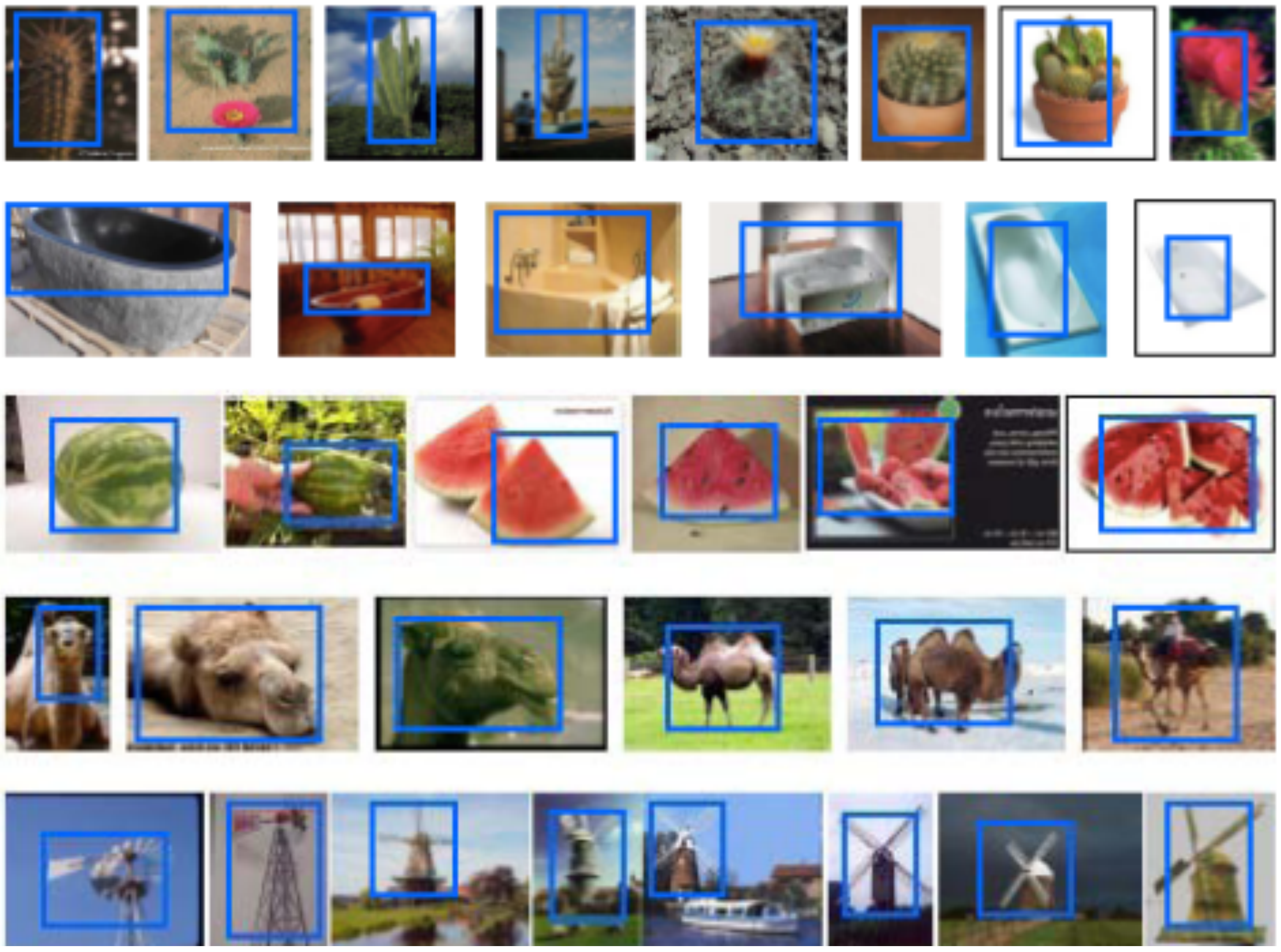
adapted from Fei-Fei Li

SURF example



adapted from Fei-Fei Li

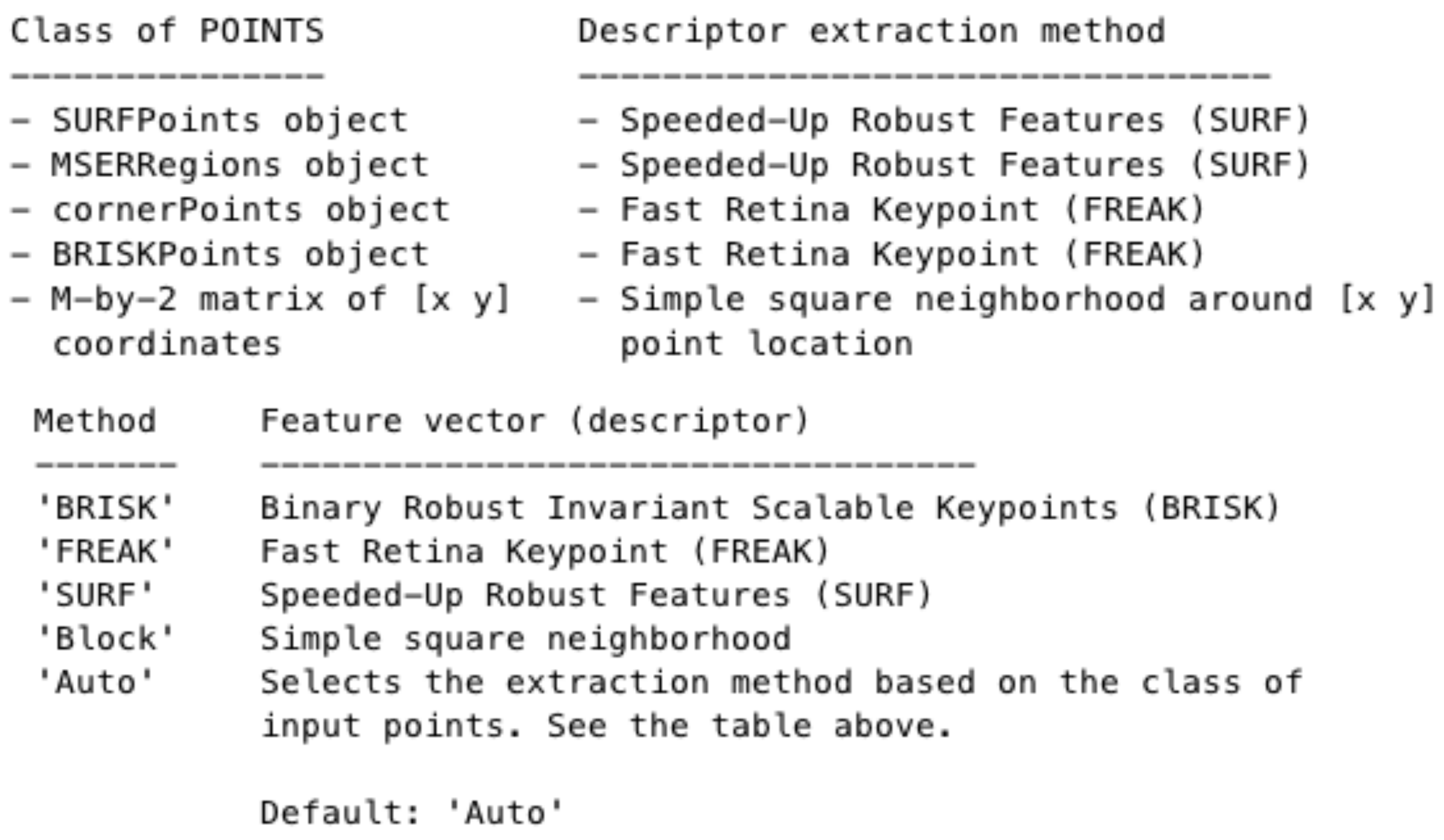
Example: Pyramid Histogram Of Words (PHOW)



Bosch et al, ICCV 2007 (variant of dense SIFT descriptor)

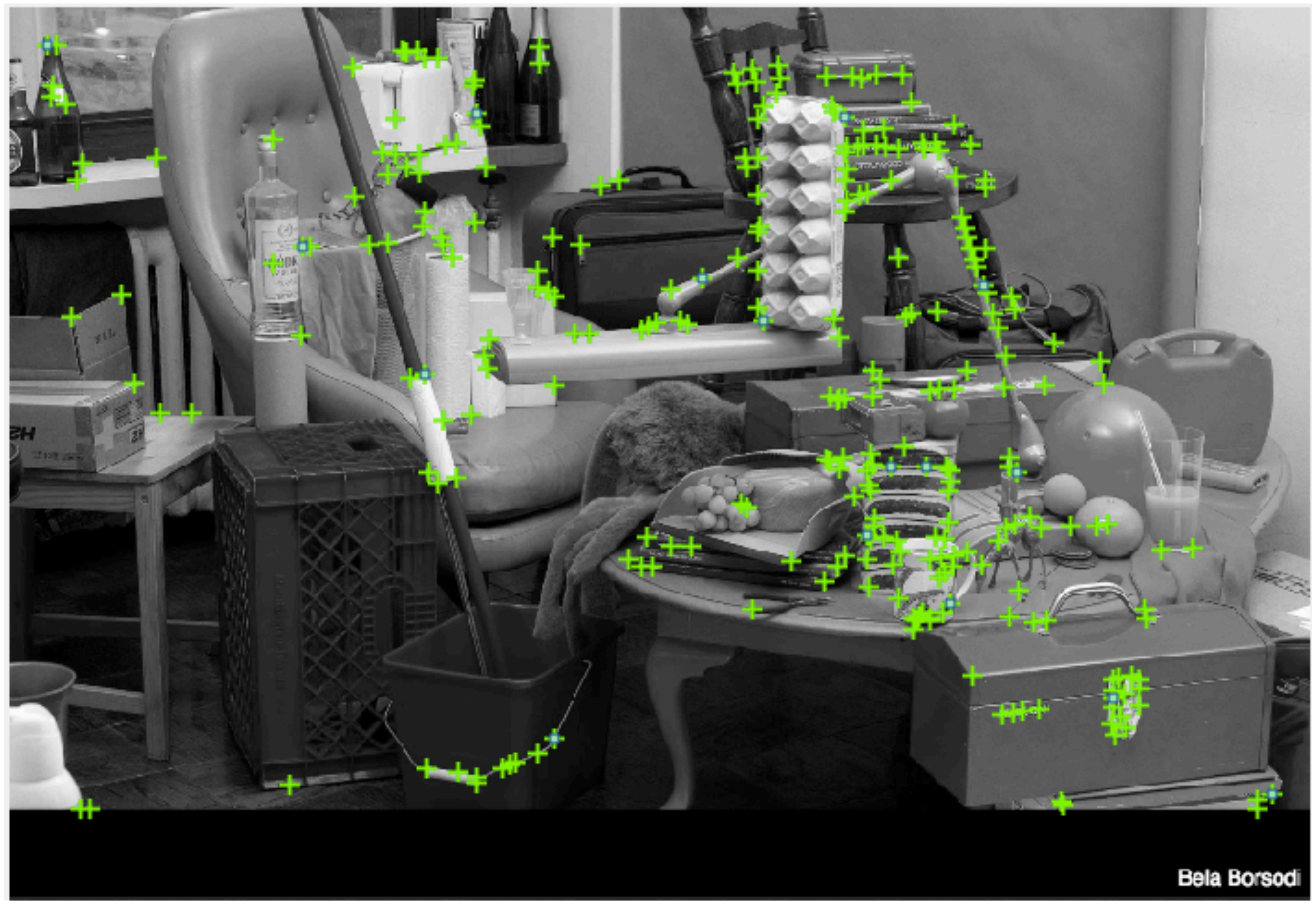
Features in Matlab

[FEATURES, VALID_POINTS] = extractFeatures(I, POINTS, Name, Value)



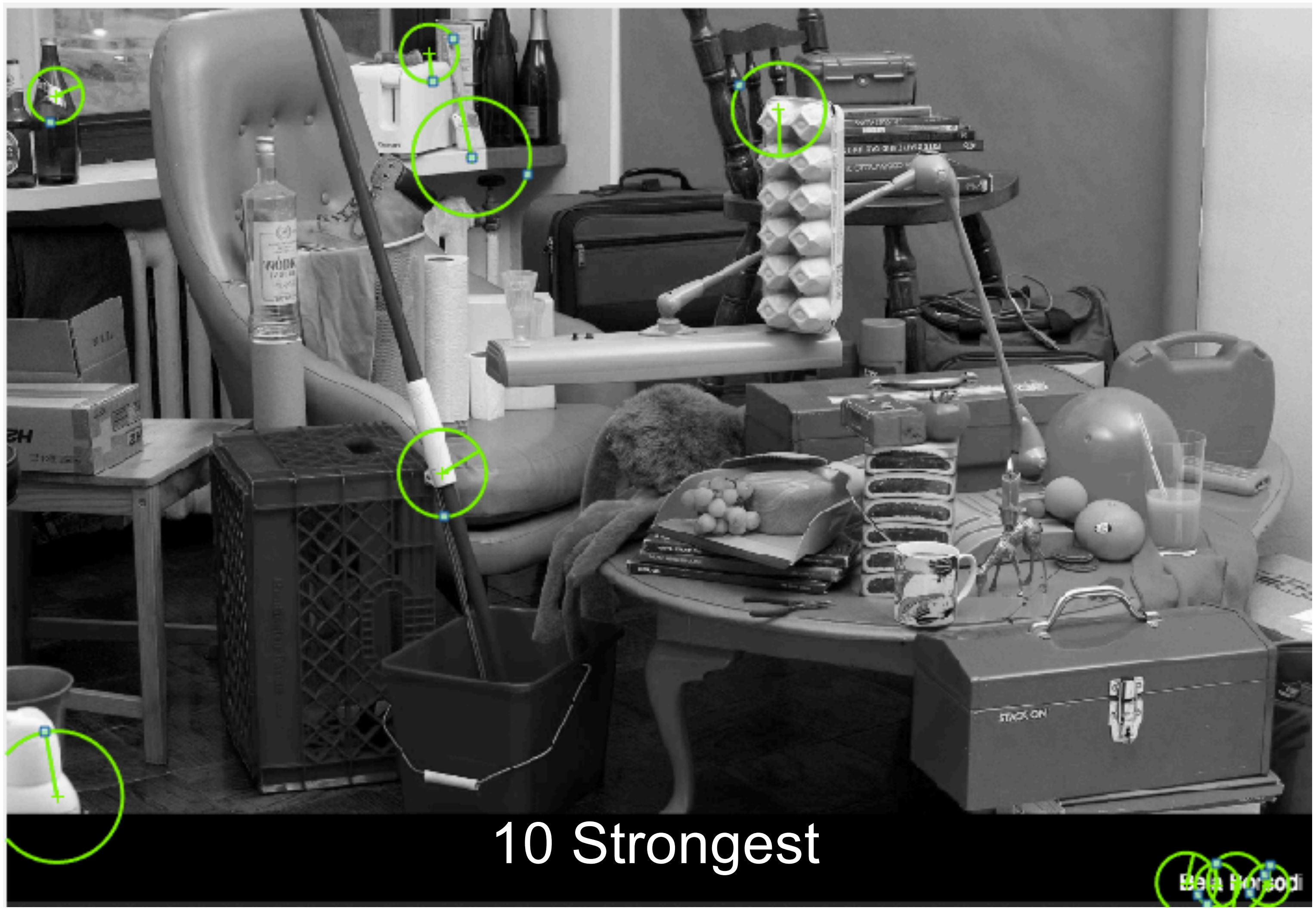
```
Class of POINTS              Descriptor extraction method
---------------              ---------------------------------
- SURFPoints object          - Speeded-Up Robust Features (SURF)
- MSERRegions object         - Speeded-Up Robust Features (SURF)
- cornerPoints object        - Fast Retina Keypoint (FREAK)
- BRISKPoints object         - Fast Retina Keypoint (FREAK)
- M-by-2 matrix of [x y]     - Simple square neighborhood around [x y]
  coordinates                  point location

 Method     Feature vector (descriptor)
 -------    ---------------------------------
 'BRISK'    Binary Robust Invariant Scalable Keypoints (BRISK)
 'FREAK'    Fast Retina Keypoint (FREAK)
 'SURF'     Speeded-Up Robust Features (SURF)
 'Block'    Simple square neighborhood
 'Auto'     Selects the extraction method based on the class of
            input points. See the table above.

            Default: 'Auto'
```

Example: Harris Corner Detector



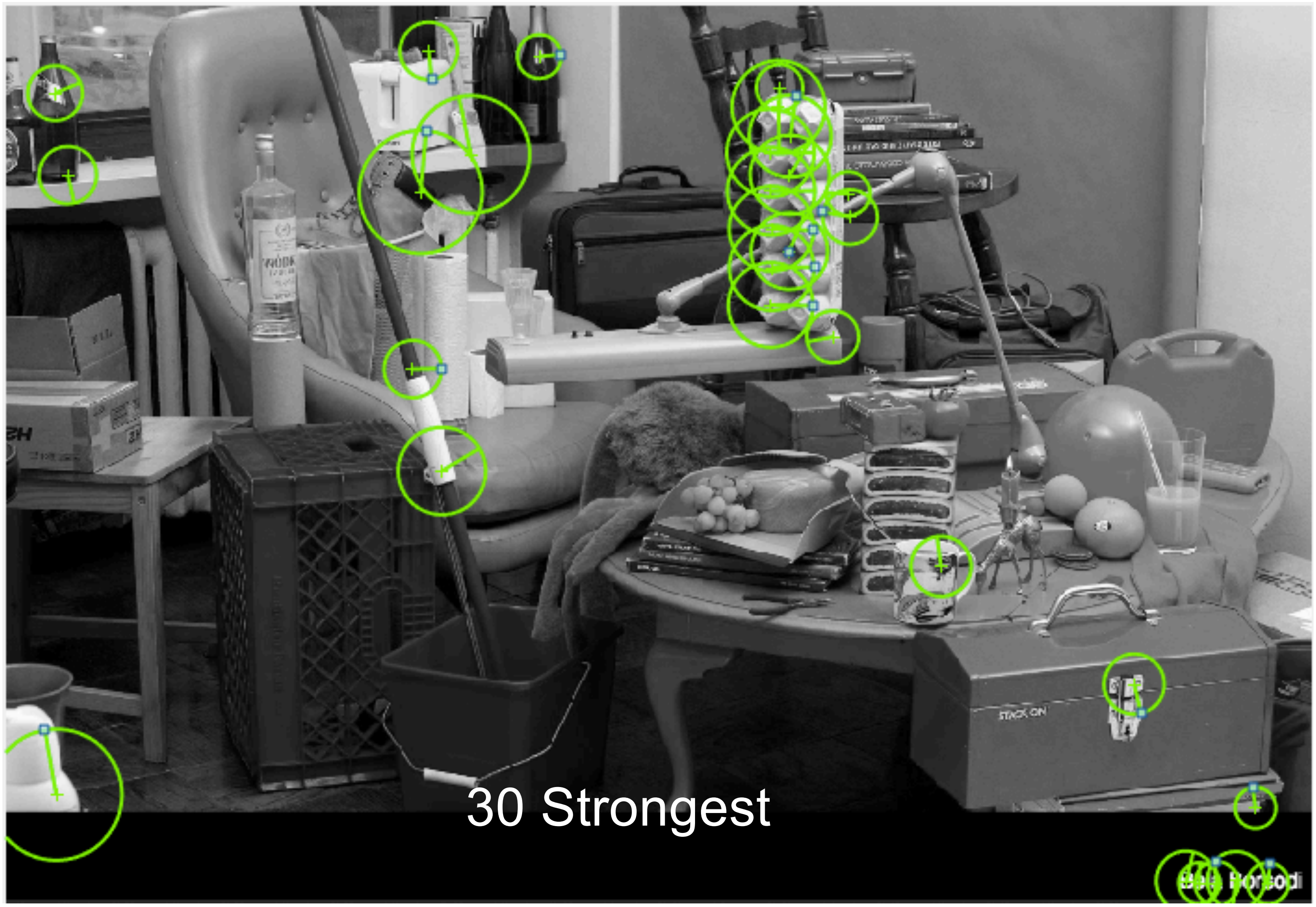
```
corners = detectHarrisFeatures(I);
```

Example: SURF features

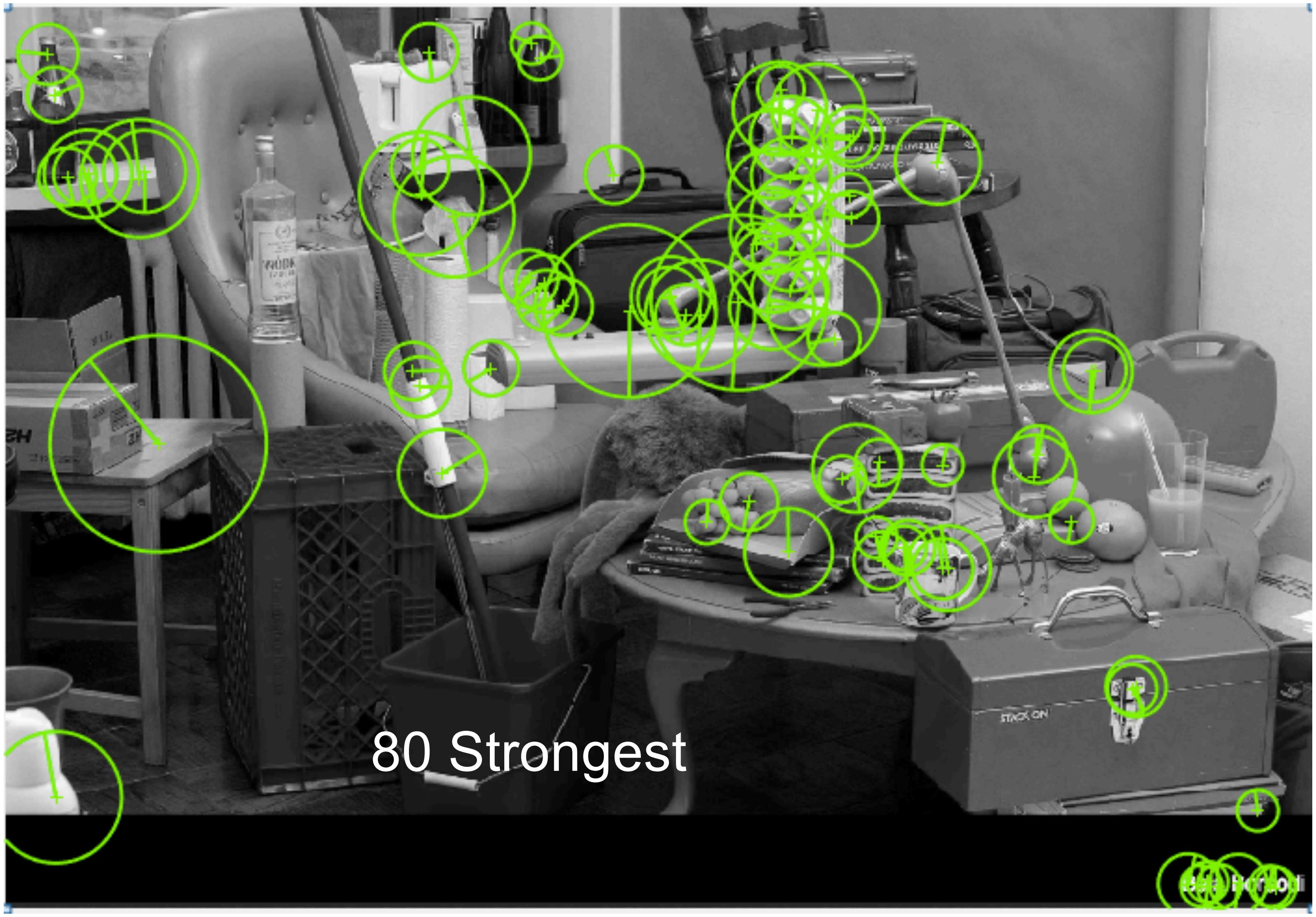


```
points = detectSURFFeatures(I);
```

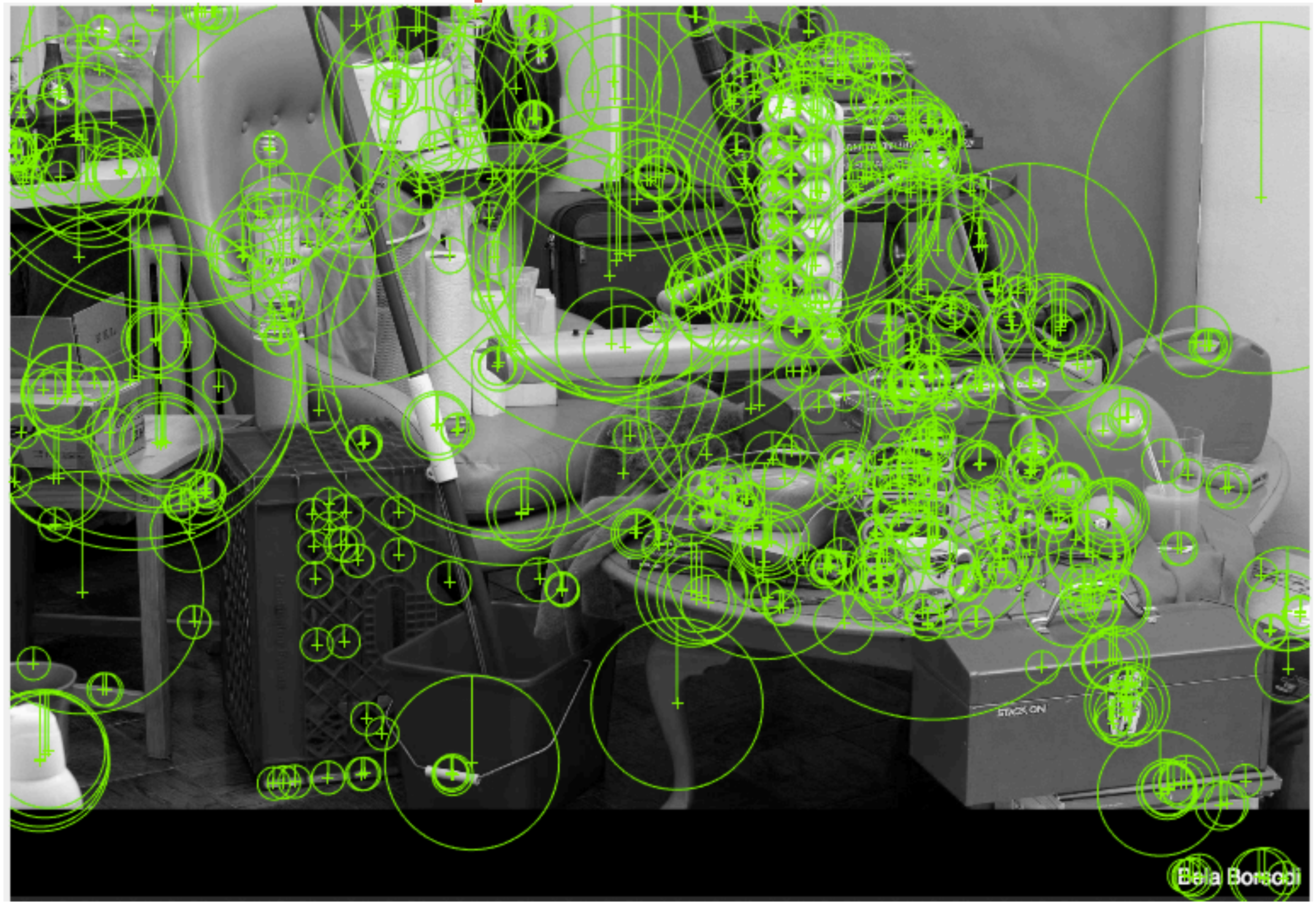
Example: SURF features



Example: SURF features

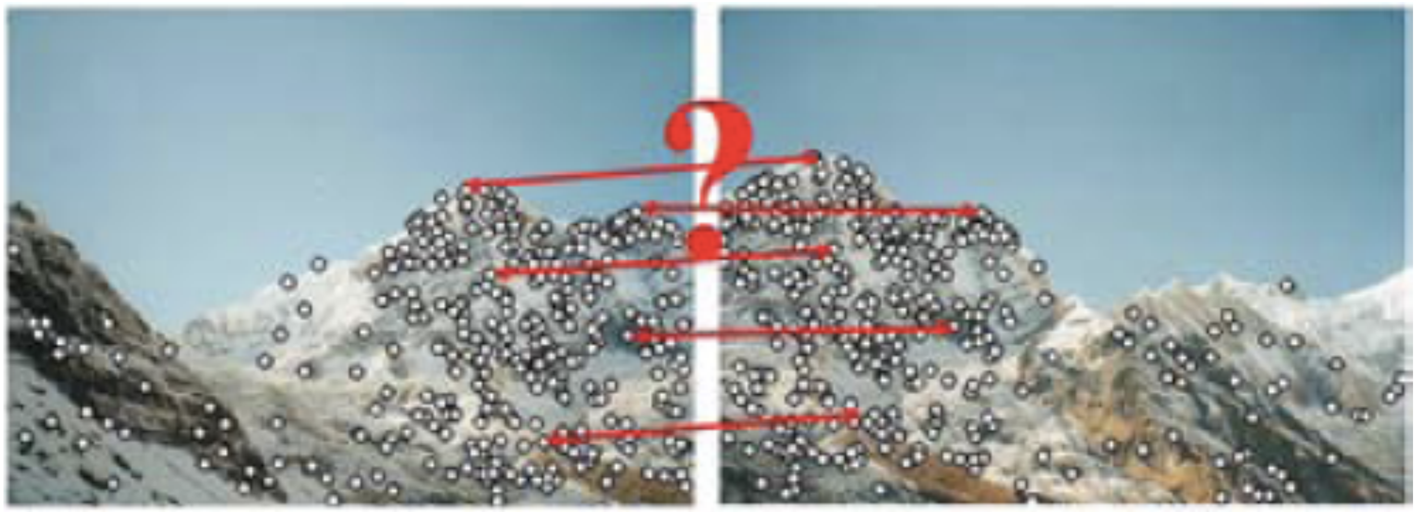


Example: MSER with upright SURF feature descriptor



```
regions = detectMSERFeatures(I);
```

Feature Matching



how can we extract local descriptors that are invariant
to inter-image variations and yet still discriminative enough to establish
correct correspondences?

Matching strategy and error rates

- Context and application dependent
  - matching a pair of images with large overlap
  - object detection
- Euclidean distances in feature space can be directly used for ranking potential matches.
- Thresholding

Performance quantification of matching algorithms

TP: true positives, i.e., number of correct matches;

FN: false negatives, matches that were not correctly detected;

FP: false positives, proposed matches that are incorrect;

TN: true negatives, non-matches that were correctly rejected.

Performance quantification of matching algorithms

- true positive rate (TPR),

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}} = \frac{\text{TP}}{\text{P}};$$

- false positive rate (FPR),

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} = \frac{\text{FP}}{\text{N}};$$

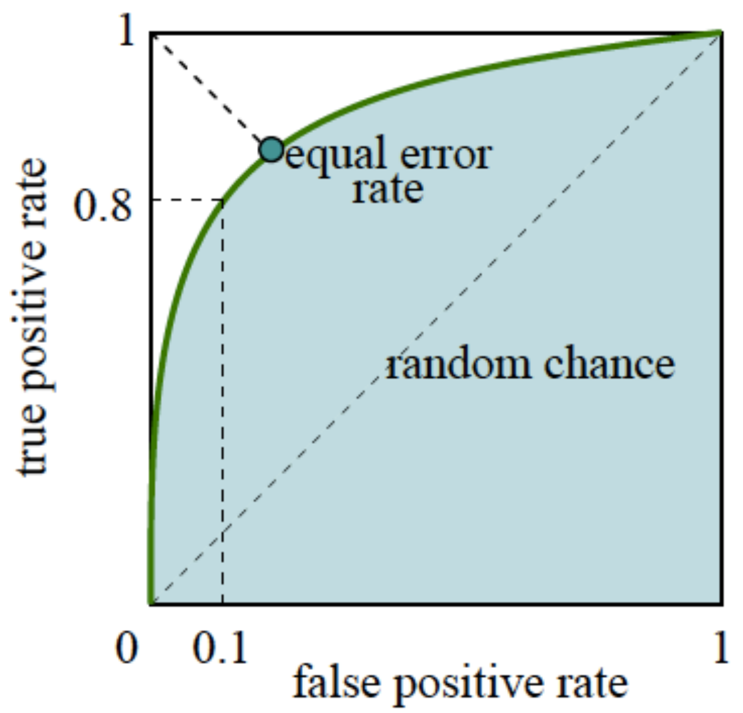
- positive predictive value (PPV),

$$\text{PPV} = \frac{\text{TP}}{\text{TP+FP}} = \frac{\text{TP}}{\text{P'}};$$

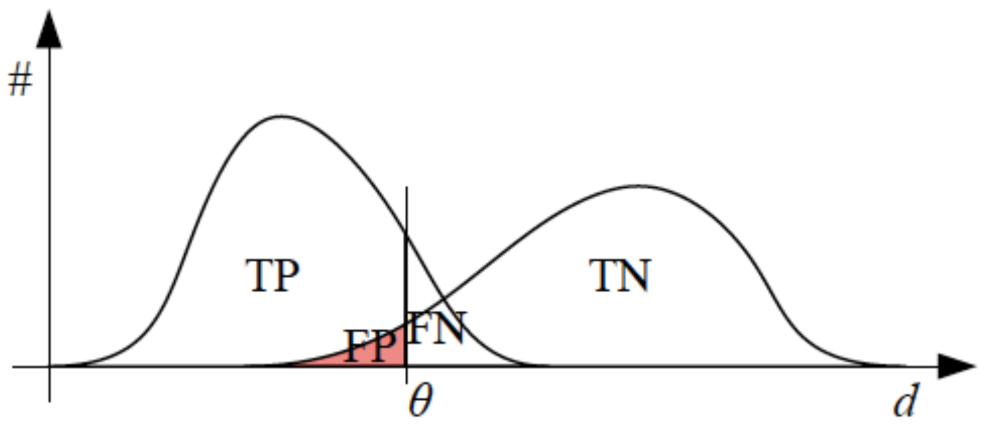
- accuracy (ACC),

$$\text{ACC} = \frac{\text{TP+TN}}{\text{P+N}}.$$

ROC curve and its related rates



(a)



(b)

Efficient Matching

- Multi-dimensional search tree
- Hash table