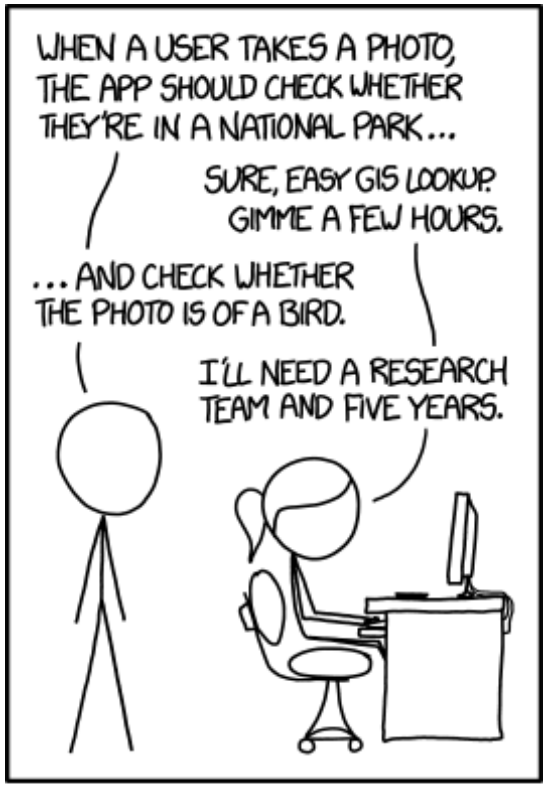
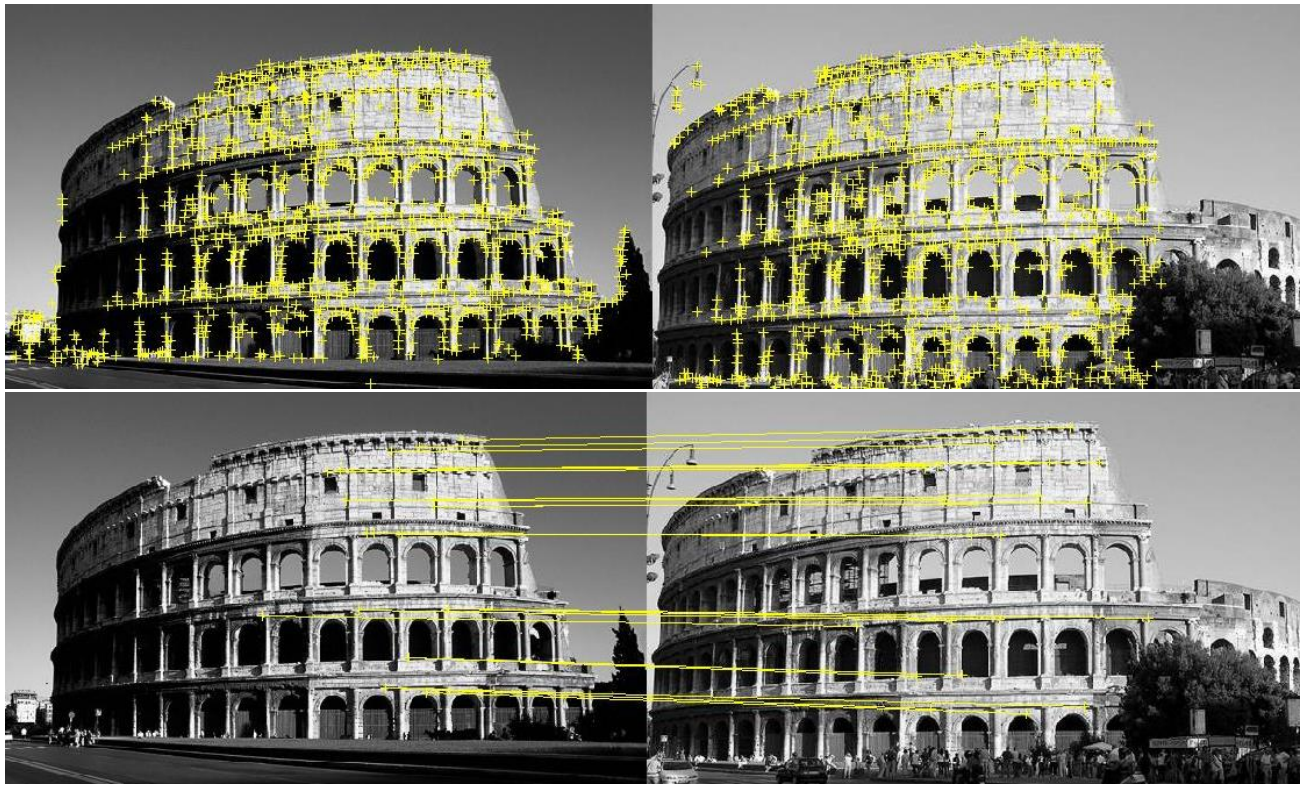
VISUAL NAVIGATION

Elements of computer vision





Elements of computer vision

- Image characteristics
- Features
- Corner detectors
- Descriptors
- Feature matching
- Feature tracking

Image characteristics

- Image (digital): ordered collection of pixels, each representing a "quantum of information" relative to the brightness of a specific color

- The quantized digital information contained in the pixel depends on
  - *scene illumination*
  - *characteristics of medium*
  - *camera optics (aperture, focal length, distortions, etc..)*
  - *number and characteristics of photo-sensors used*
  - *quantization levels*

- Numerical point of view: tensor (multi-dimensional matrix) in which each two-dimensional submatrices provides one channel information. We name such tensor the "intensity" of the pixel at position $\mathbf{x}$: $I(\mathbf{x})$

Image characteristics

- Example: RGB images

$I(\mathbf{x}_r) \quad ; \quad I \in [0, 255]$

$I(\mathbf{x}_g) \quad , \quad I \in [0, 255]$

$I(\mathbf{x}_b) \quad , \quad I \in [0, 255]$

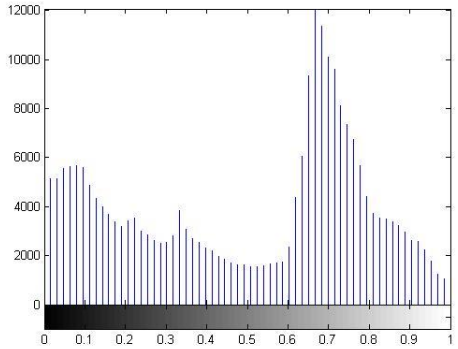
$I(\mathbf{x}) \quad , \quad I \in [0, 255] \times 3$

Image characteristics

- Manipulation of images:

  Image processing (mapping from images to images)

Histograms of intensity values (gray scale)



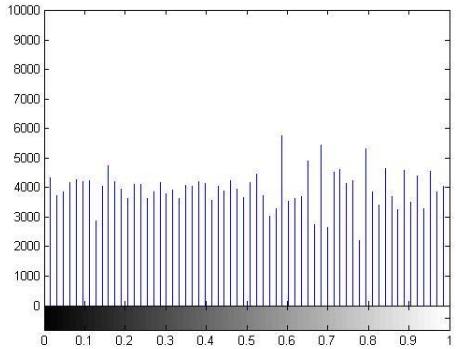


Image characteristics

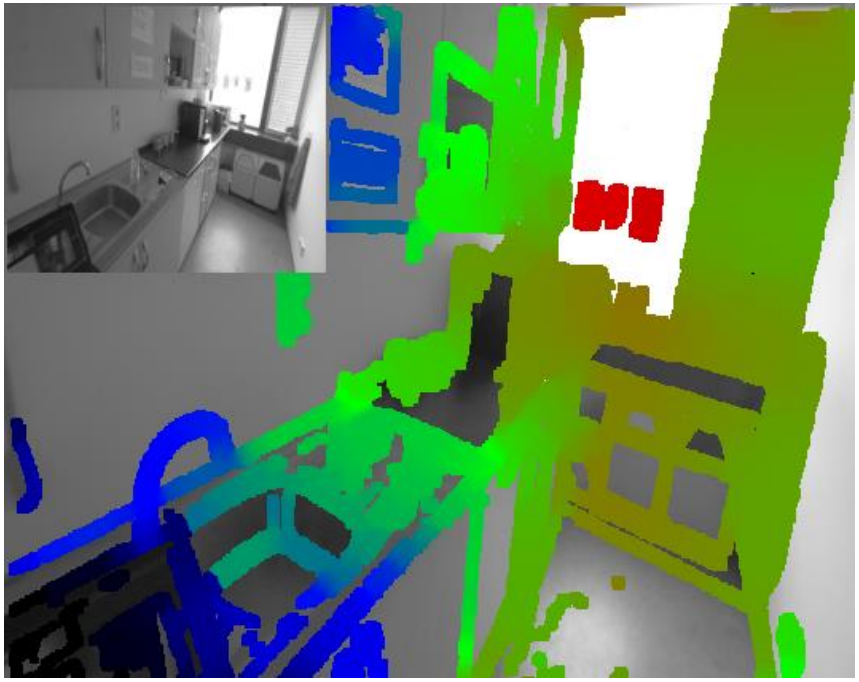
- Manipulation of images:

  Computer graphics (mapping a model into an image)

Image characteristics

- Manipulation of images:

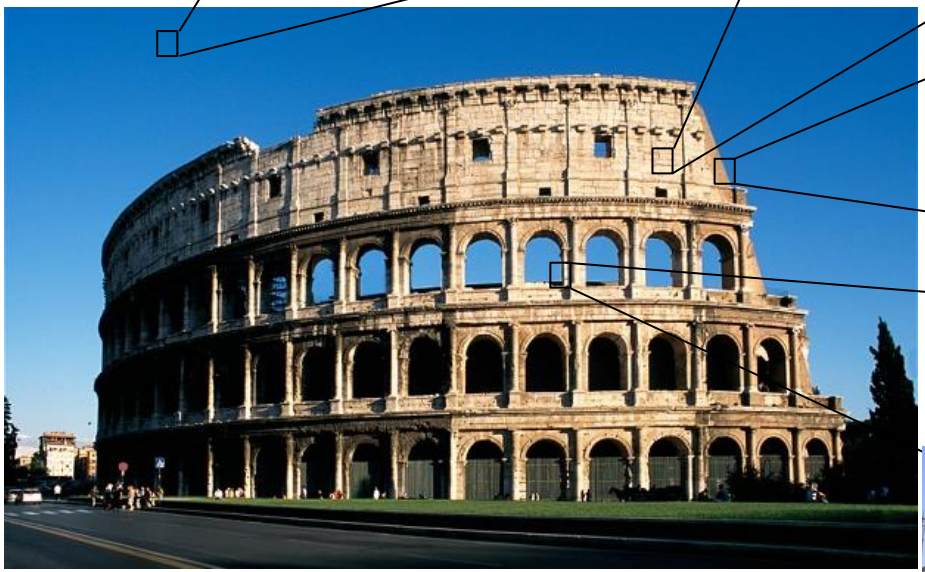
Computer vision (extracting a model from an image)

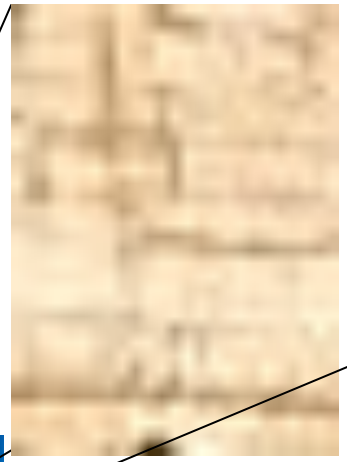


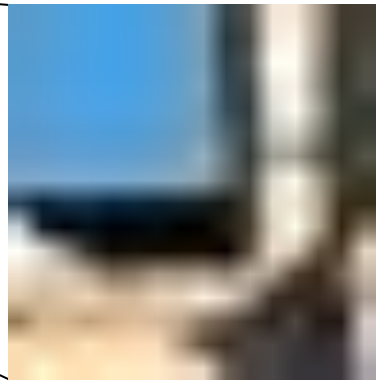
Features

- How can we compare two images of a same scene taken from two different points of view, aiming at determining whether they are images of the same object in space?

Features



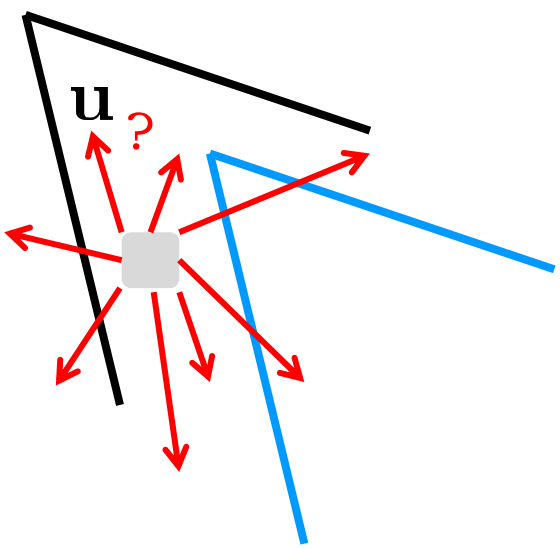


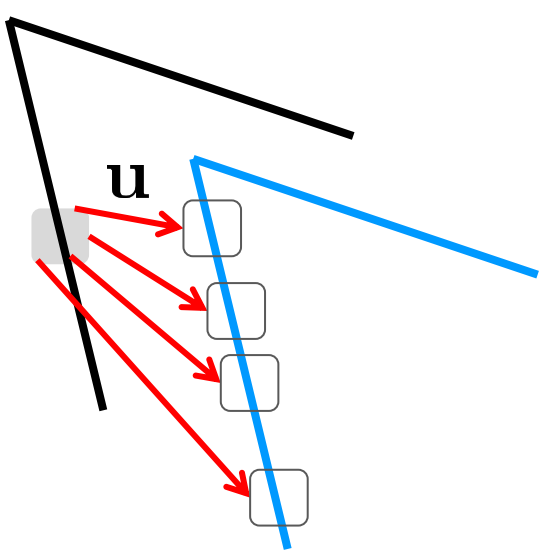


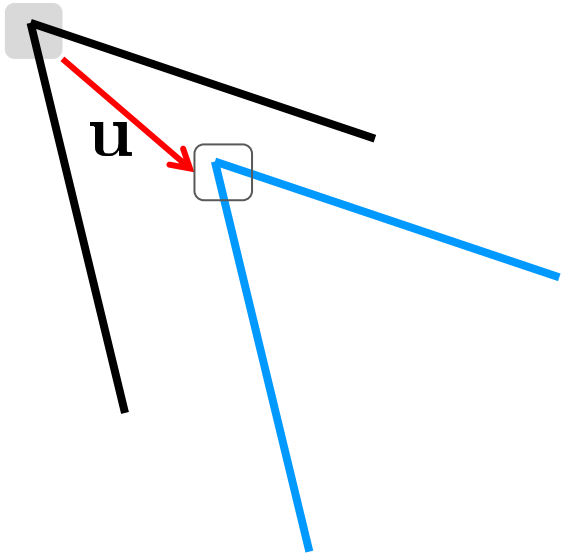
- Typical good features are edges and corners

Features

- Prevalence of edges and corners
- *E.g. : find the vectorial distance between the black and blue objects by looking at a single "patch" in the image*





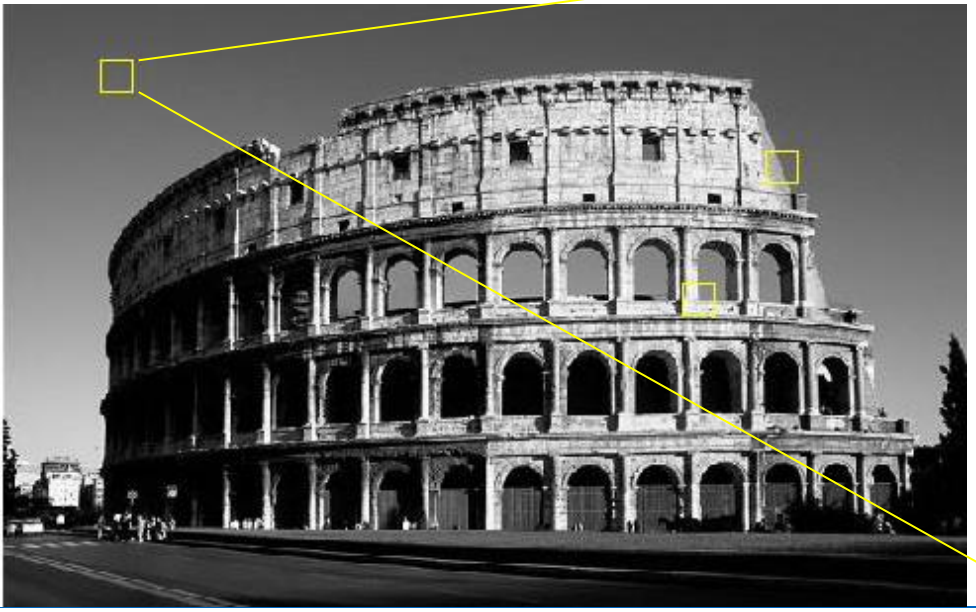


- Simplest matching criterion: minimization of SSE over the pixels in the selected patch $f(\mathbf{u}) = \sum_i \left(I_2(\mathbf{x}_i + \mathbf{u}) - I_1(\mathbf{x}_i)\right)^2$

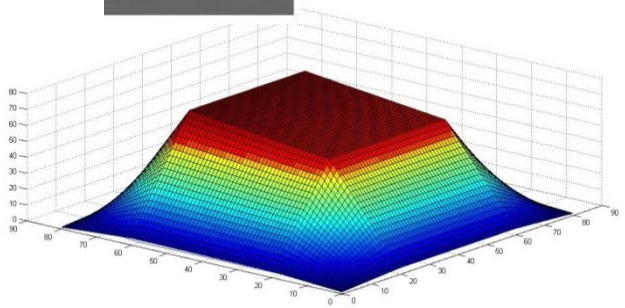
Features

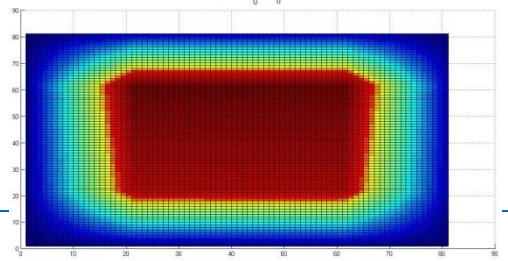
- Scan for features in an image: find those “patches” for which small displacements cause large variations in intensity
- Weighted SSD (WSSD) within the same image:

$$f_{\mathrm{ac}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i)\,(I(\mathbf{x}_i + \Delta\mathbf{u}) - I(\mathbf{x}_i))^2$$



Autoconvolution around the patch:



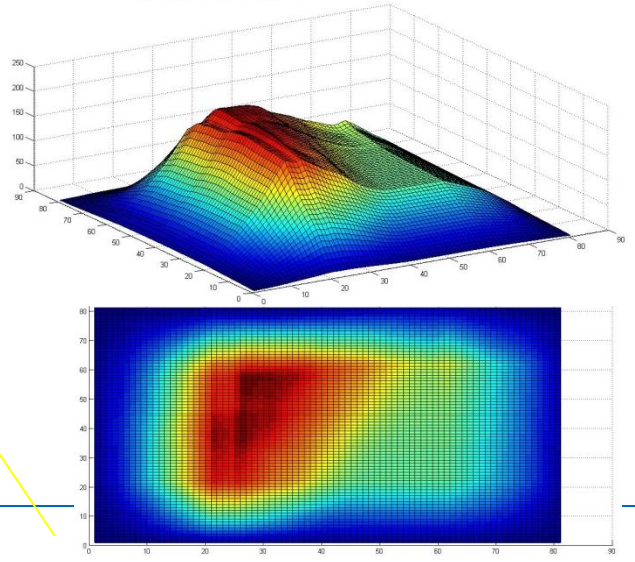


Features

- Scan for features in an image: find those “patches” for which small displacements cause large variations in intensity
- Weighted SSD (WSSD) within the same image:

$$f_{\mathrm{ac}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i)\,(I(\mathbf{x}_i + \Delta\mathbf{u}) - I(\mathbf{x}_i))^2$$

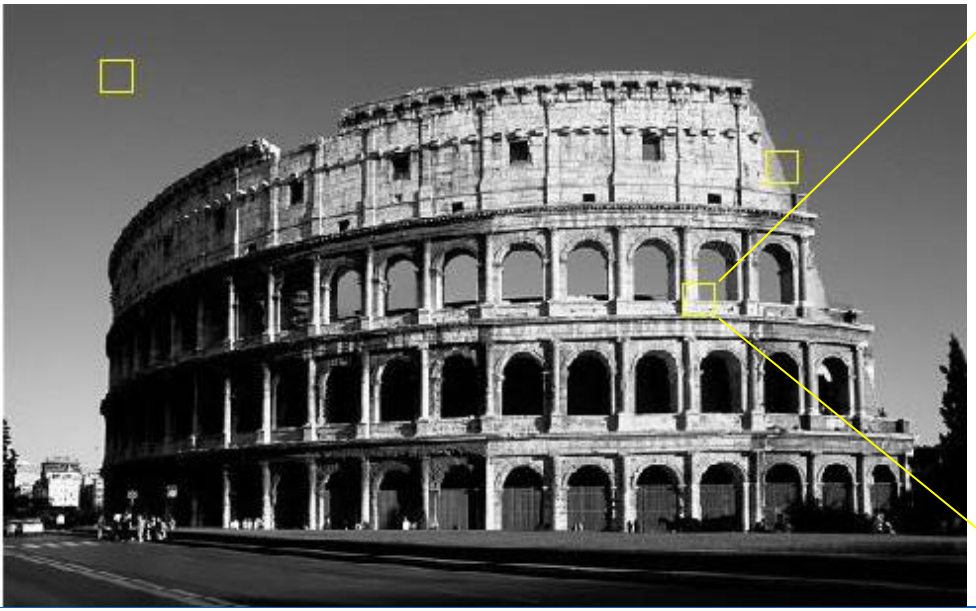
Autoconvolution around the patch:



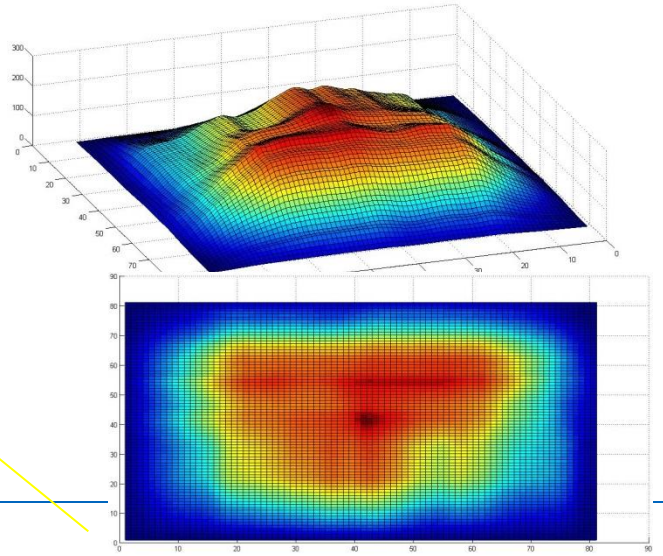
Features

- Scan for features in an image: find those “patches” for which small displacements cause large variations in intensity
- Weighted SSD (WSSD) within the same image:

$$f_{\mathrm{ac}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i)\,(I(\mathbf{x}_i + \Delta\mathbf{u}) - I(\mathbf{x}_i))^2$$



Autoconvolution around the patch:



Features

- This suggests to look for “gradients” in the image

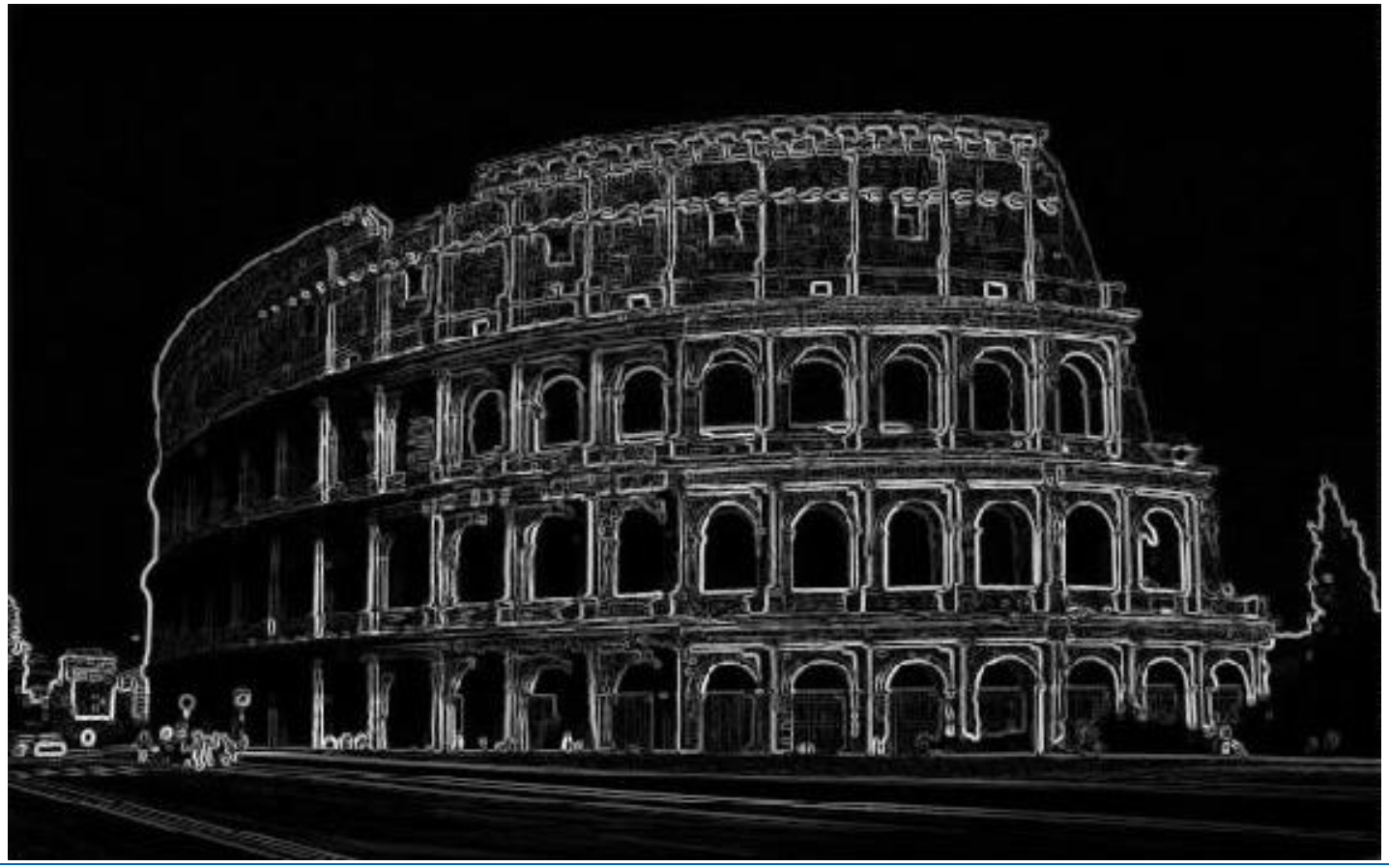
- First-order Taylor expansion:

$$
\begin{aligned}
f_{\mathrm{ac}}(\Delta\mathbf{u}) &= \sum_i w(\mathbf{x}_i)\left(I(\mathbf{x}_i+\Delta\mathbf{u}) - I(\mathbf{x}_i)\right)^2 \\
&\approx \sum_i w(\mathbf{x}_i)\left(I(\mathbf{x}_i) + \nabla I(\mathbf{x}_i)\Delta\mathbf{u} - I(\mathbf{x}_i)\right)^2 \\
&= \Delta\mathbf{u}^{\mathrm{T}}\mathbf{A}\Delta\mathbf{u}
\end{aligned}
$$

$$
\mathbf{A} = \sum_i \mathbf{A}(\mathbf{x}_i) = \underbrace{\begin{bmatrix} \sum_i w(\mathbf{x}_i) I_x^2(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) \\ \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_y^2(\mathbf{x}_i) \end{bmatrix}}_{\textit{IMAGE GRADIENT}}
$$

Features

- Image gradient:



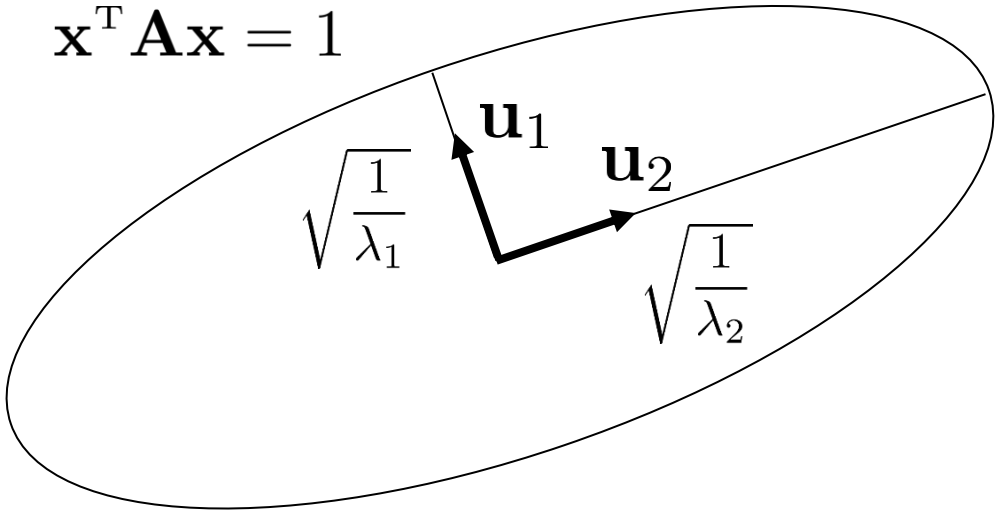
Corner detectors

- Image gradient is also known as Harris matrix

$$\mathbf{A} = \sum_i \mathbf{A}(\mathbf{x}_i) = \begin{bmatrix} \sum_i w(\mathbf{x}_i) I_x^2(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) \\ \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_y^2(\mathbf{x}_i) \end{bmatrix}$$

- Relation with axis of an ellipse: $\mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x} = 1$

$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^{\mathrm{T}}$$

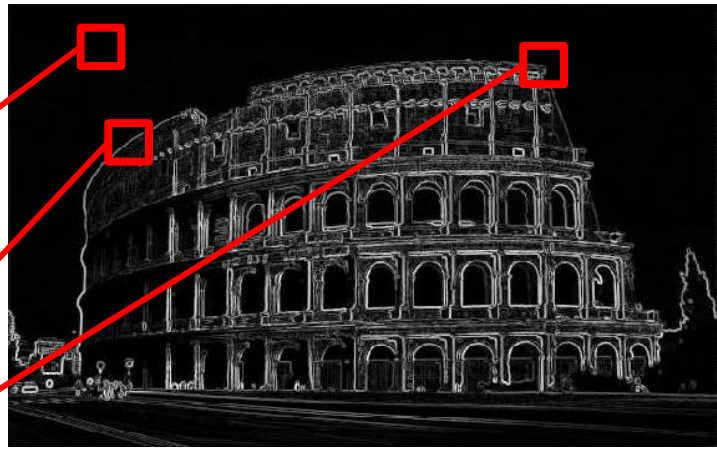


Corner detectors

- Harris matrix

$$\mathbf{A} = \sum_i \mathbf{A}(\mathbf{x}_i) = \begin{bmatrix} \sum_i w(\mathbf{x}_i) I_x^2(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) \\ \sum_i w(\mathbf{x}_i) I_x(\mathbf{x}_i) I_y(\mathbf{x}_i) & \sum_i w(\mathbf{x}_i) I_y^2(\mathbf{x}_i) \end{bmatrix}$$

- How to detect corners by looking into the Harris matrix?



- $\lambda_{\max}$ *and* $\lambda_{\min}$ *small: no change detected*
- $\lambda_{\max}$ *large and* $\lambda_{\min}$ *small: edge detected*
- $\lambda_{\max}$ *and* $\lambda_{\min}$ *large: corner detected*

Corner detectors

- Detection of corners in an image through inspection of Harris matrix:

Harris-Stephenson detector

$$r(\mathbf{A}) = \lambda_{\max}\lambda_{\min} - \alpha(\lambda_{\max} + \lambda_{\min})^2$$

$0.04 \leq \alpha \leq 0.06$ is a tunable parameter (empirical)

$r(\mathbf{A}) < -\text{th}$	$\lvert r(\mathbf{A})\rvert < \text{th}$	$r(\mathbf{A}) > \text{th}$
Edge detected	*No features*	*Corner detected*

- Harris-Stephenson detector uses a [-2 -1 0 1 2] patch window (filter)

Corner detectors

- Computation of eigenvalues not numerically efficient: workaround in the Harris-Stephenson detector as

$$r(\mathbf{A}) = \lambda_{\max}\lambda_{\min} - \alpha(\lambda_{\max} + \lambda_{\min})^2 = \det \mathbf{A} - \alpha(\mathrm{tr}\mathbf{A})^2$$

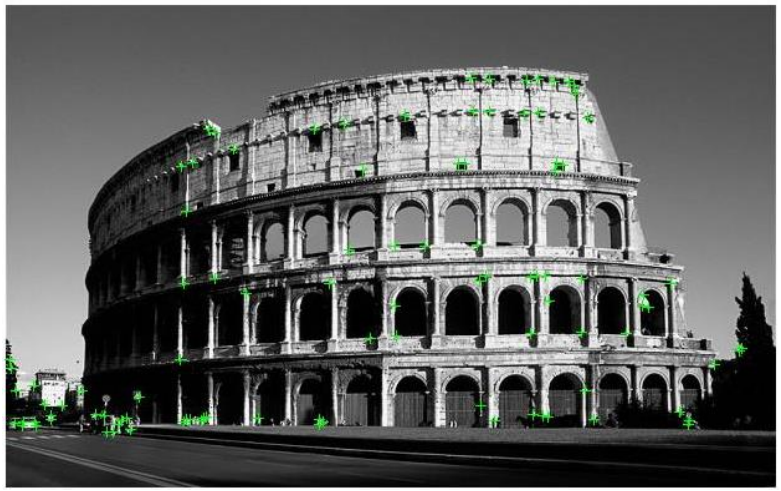
- Other detectors based on same principle:
- Shi-Tomasi / Kanade-Tomasi detector: $\lambda_{\min}$
- Harmonic mean: $2\frac{\det \mathbf{A}}{\mathrm{tr}\mathbf{A} + \epsilon}$
- Triggs: $\lambda_{\max} - \beta\lambda_{\min}$

Corner detectors

- Often an exceeding number of features is detected

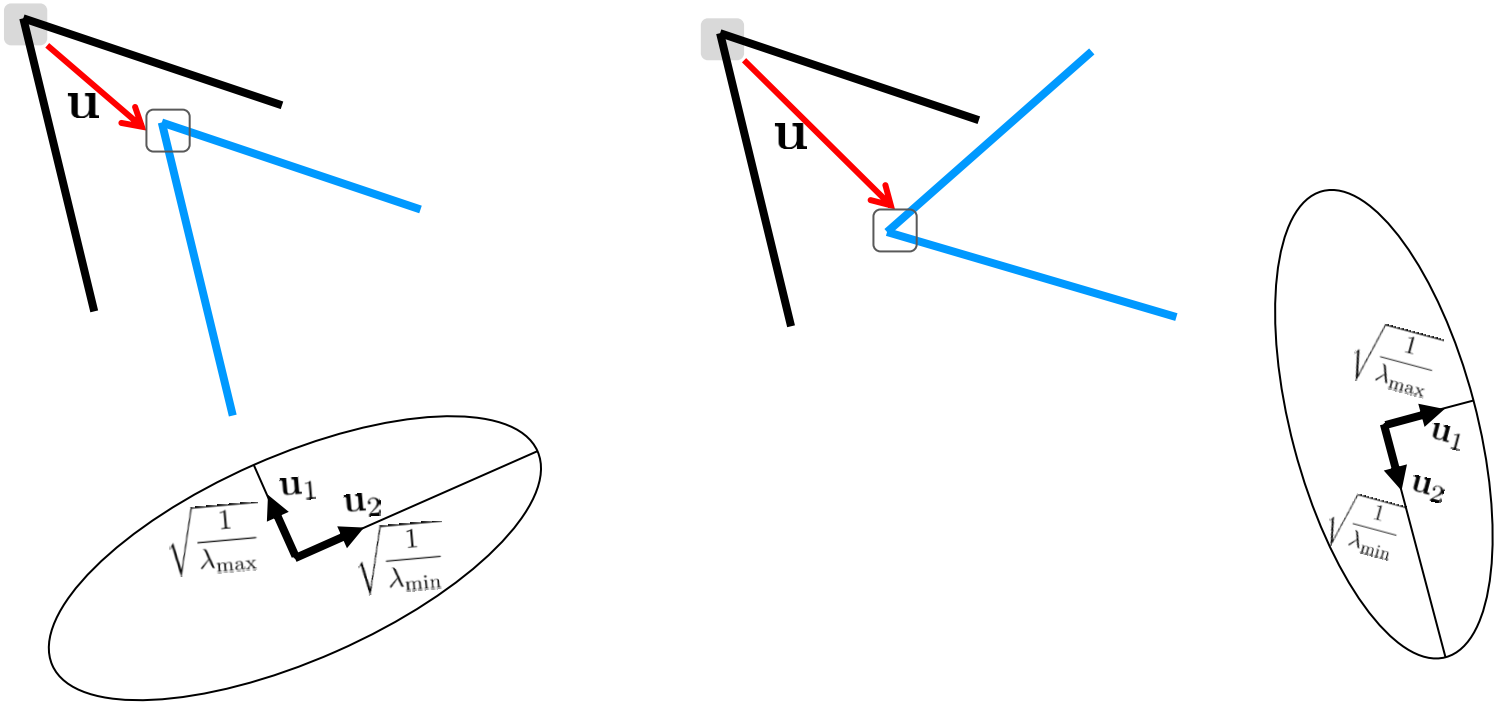
Non-maxima suppression: elimination of closely-detected corners by selectively picking the maxima

Example: Harris-Stephenson detector ($\alpha = 0.04$), when only keeping "strongest" features (highest $r(\mathbf{A})$ values)



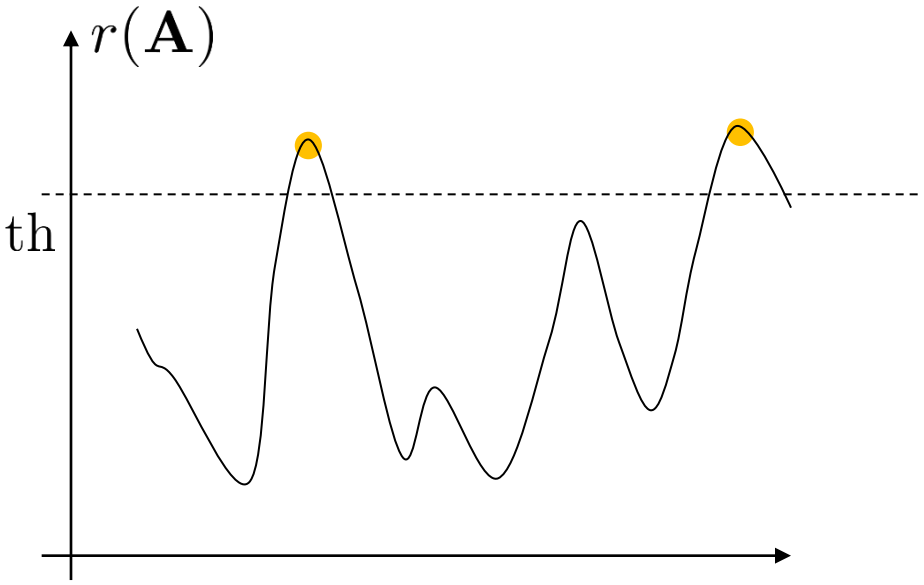
Corner detectors

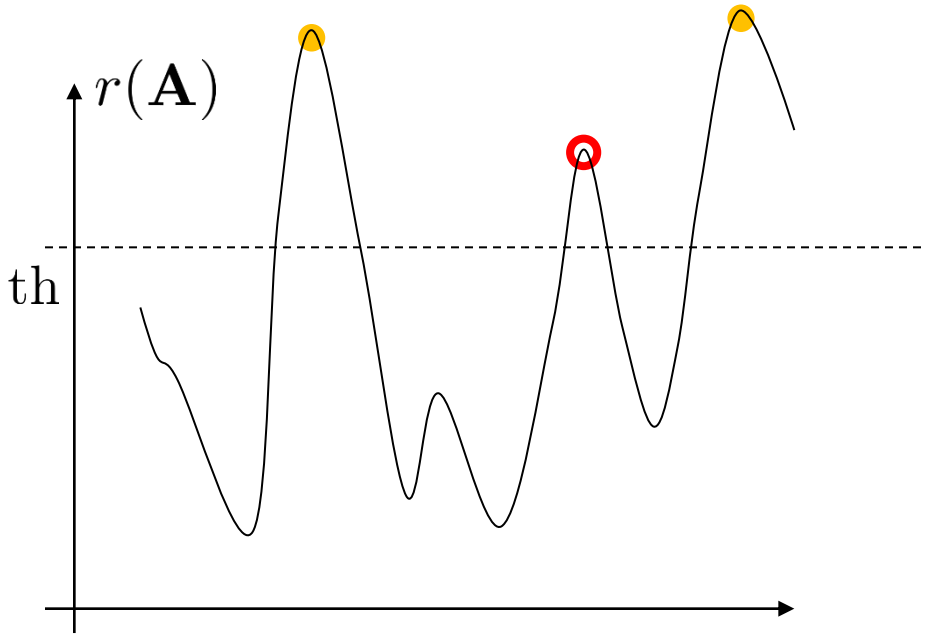
- Harris detector: invariants
- Invariance to rotation: corners are detected as such independently from rotations (magnitude of eigenvalues does not change)



Corner detectors

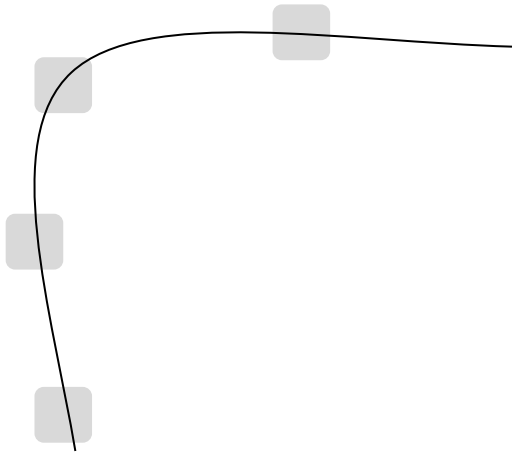
- Harris detector: invariants
- Partial invariance to additive and multiplicative intensity changes

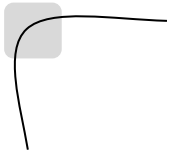




Corner detectors

- Harris detector: invariants
- Not invariant to scaling





- *All edges* *Corner!*

Corner detectors

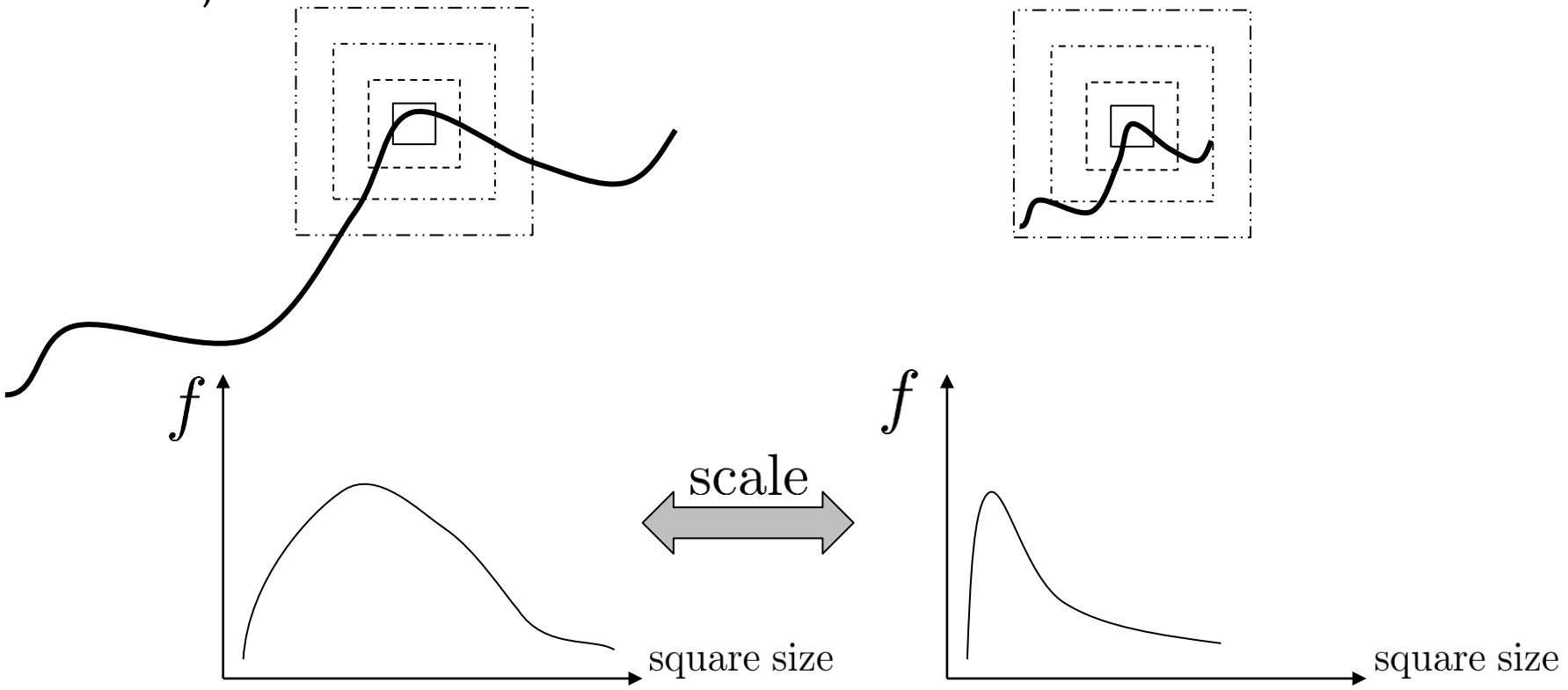
- Problem: how do we define a feature detection strategy that is invariant to
  - Rotation
  - Scaling

- Solution: design a detector that extract features by investigating image AND scale (three-dimensional search), aiming at detecting features at all possible scales

- Scale invariance is useful for database matching, in which images/regions at different scales are compared. If the feature extraction operates on videos or image sequences that are close (in time and space) to each other, scale invariance may not be as important

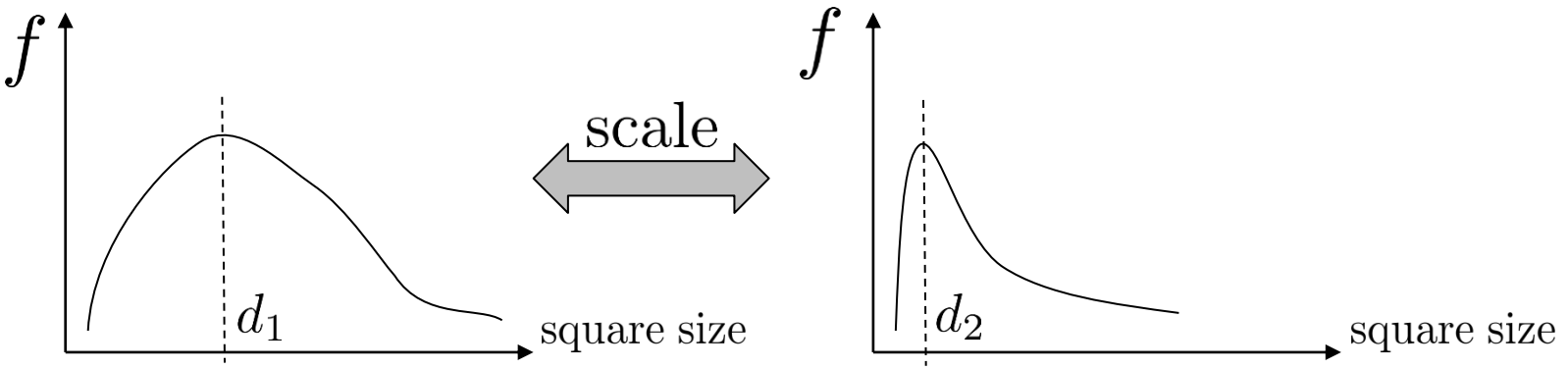
Corner detectors

- Example: design function of the image points (on the squared region of variable size) whose behavior is scale-independent (other than rotation!)



Corner detectors

- Example: design function of the image points (on the squared region of variable size) whose behavior is scale-independent (other than rotation!)
- Detect the value of the region size that maximizes the chosen function
- Same feature detected at region size (scale) d_1 on the left and at region size (scale) d_2 on the right



Corner detectors

- Widely (almost exclusively) used scale-functions: combination of Gaussians
- Scale-space kernel (mask): Gaussian $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$
- Example: Scale Invariant Feature Transform (SIFT)

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

Scale space of image in x,y

Laplacians

Convolution

- Local maximum of the scale space of the image returns the scale-invariant feature

Corner detectors

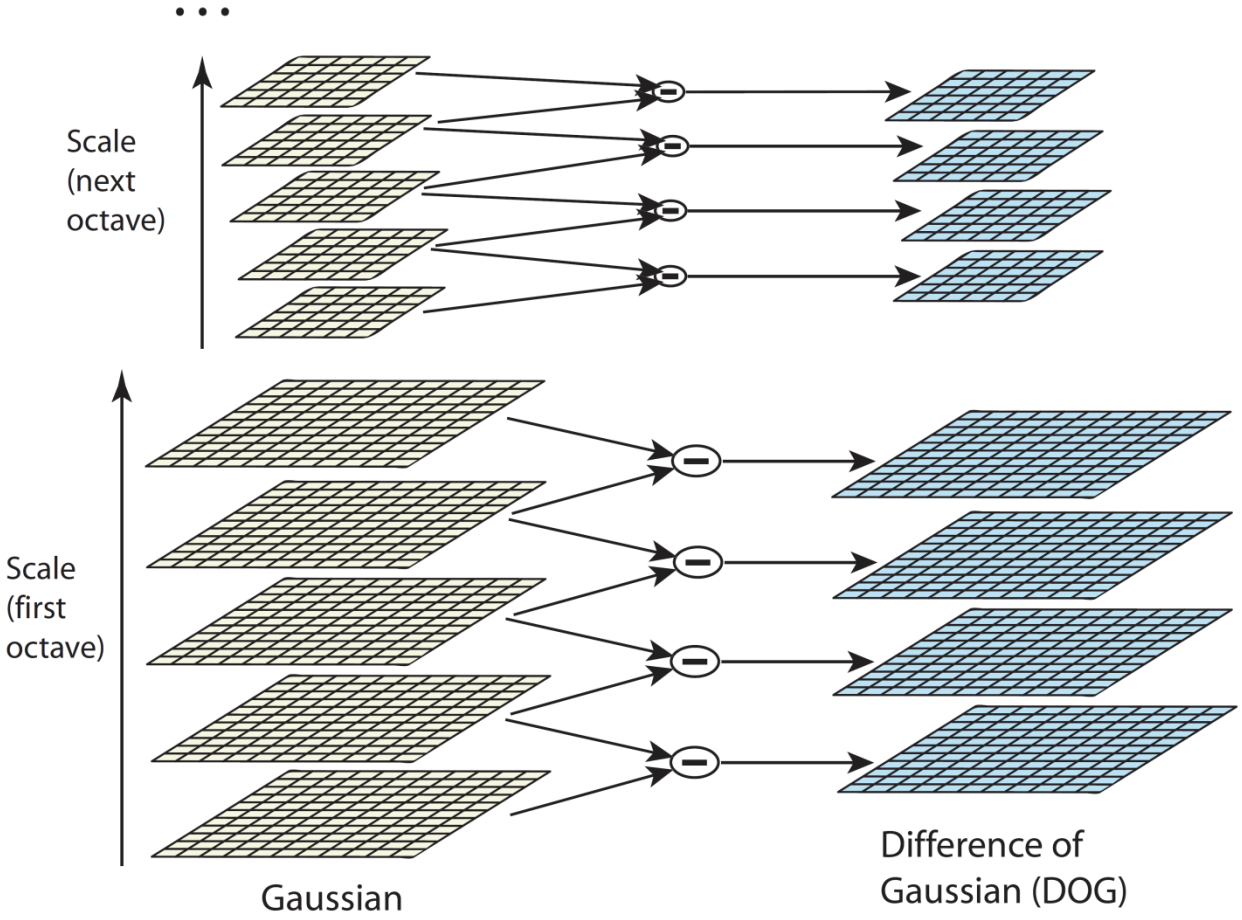
- SIFT procedures

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

- Cycle Gaussian filtering through σ (E.g. $\sigma_i = \sigma_{i-1} \cdot 2^{i/2}$)
- Cycle scale: descale original image by a factor 2 (keep every other row and column)
- Typical iterations: 5 blur levels (sigmas), 4 scaling levels
- Nota: the scale-space of the image has to be used in extracting the descriptors, so it will be re-used

Corner detectors

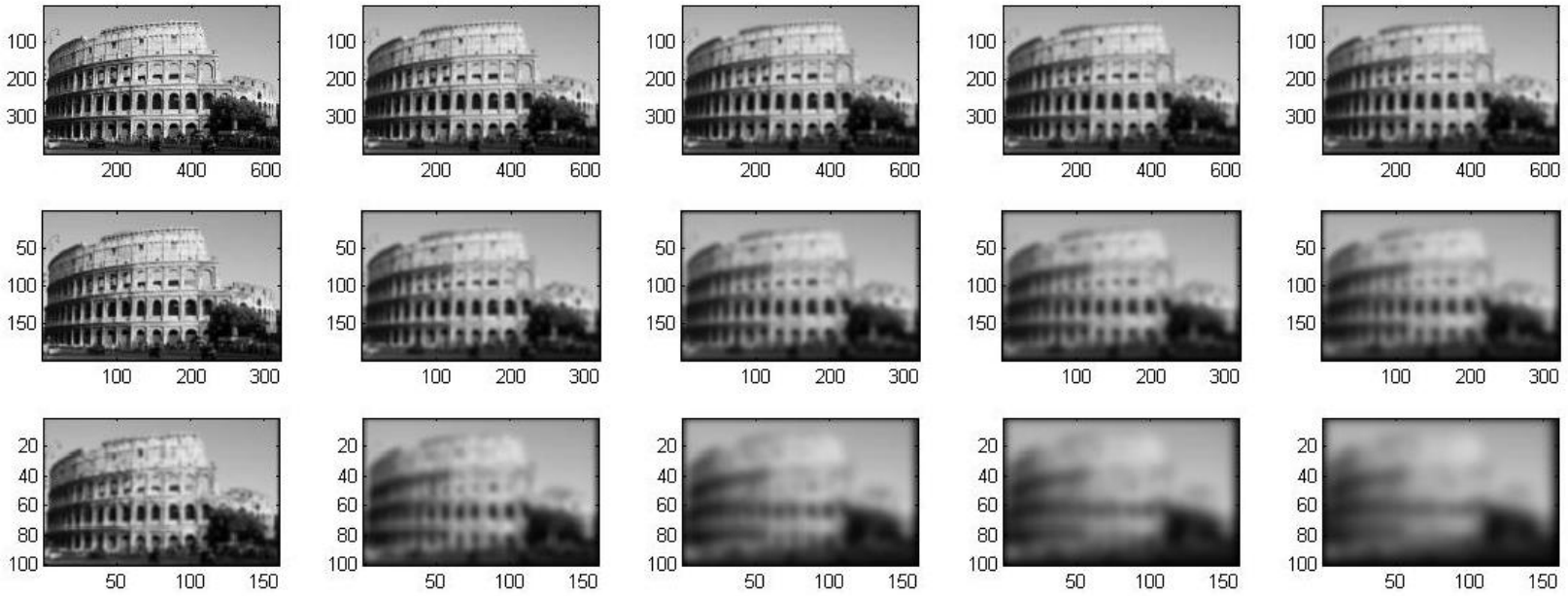
- SIFT procedures



Credit: Lowe, D. G.: "Distinctive Image Features from Scale-Invariant Keypoints"

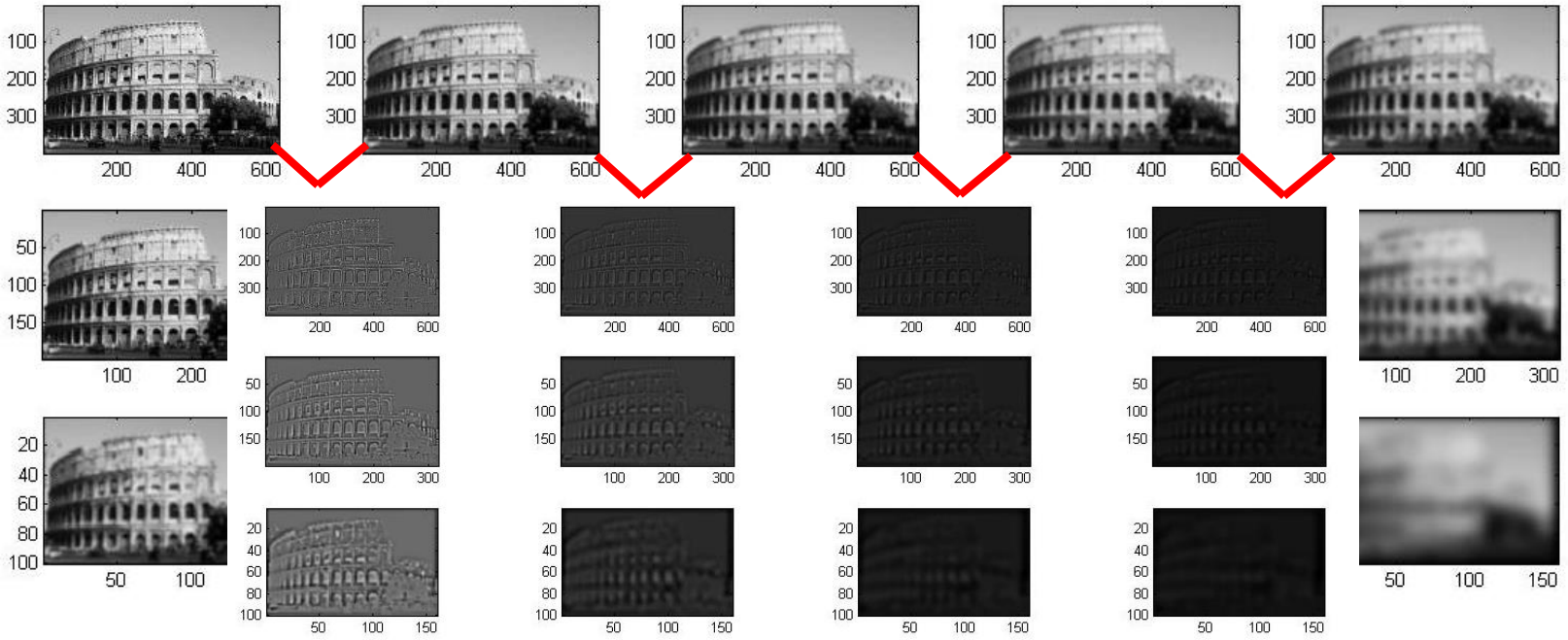
Corner detectors

- SIFT procedures (here 3 scales and 5 blur levels)



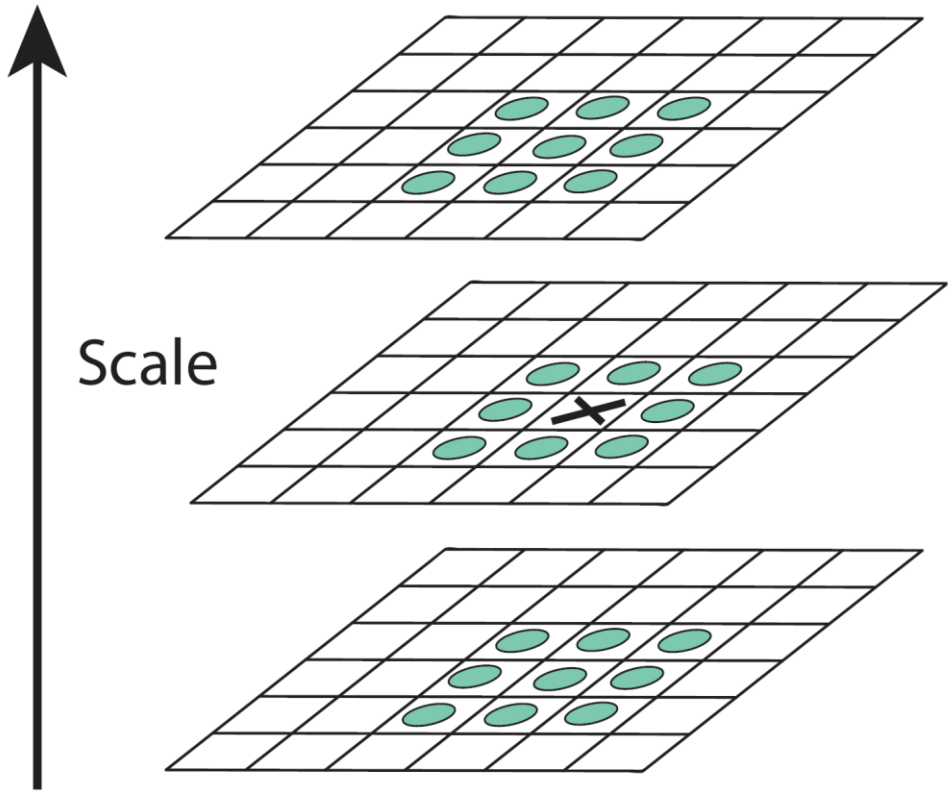
Corner detectors

- SIFT procedures $D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$



Corner detectors

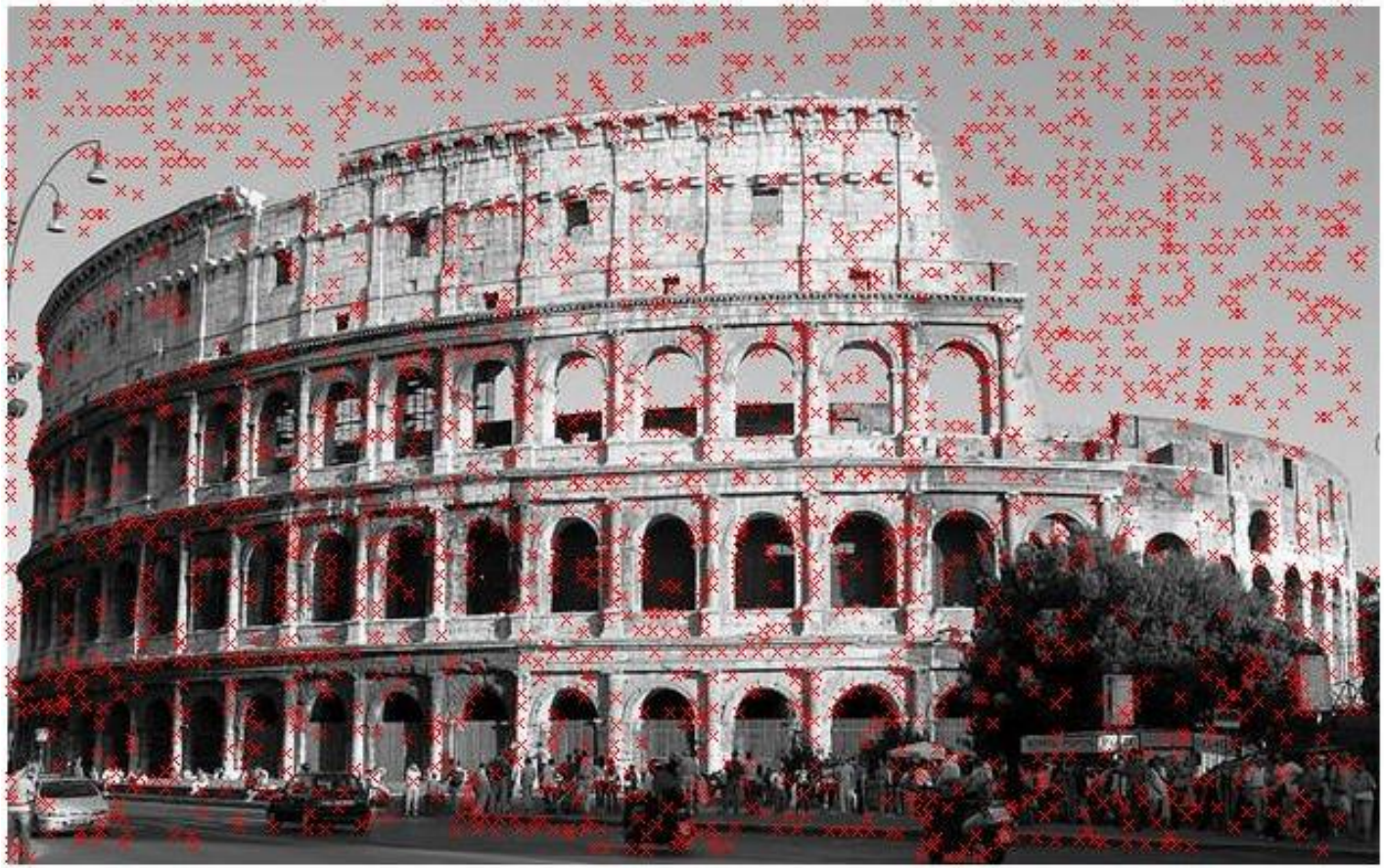
- SIFT procedures
- *Detect maxima in both scale and image space*



Credit: Lowe, D. G.: "Distinctive Image Features from Scale-Invariant Keypoints"

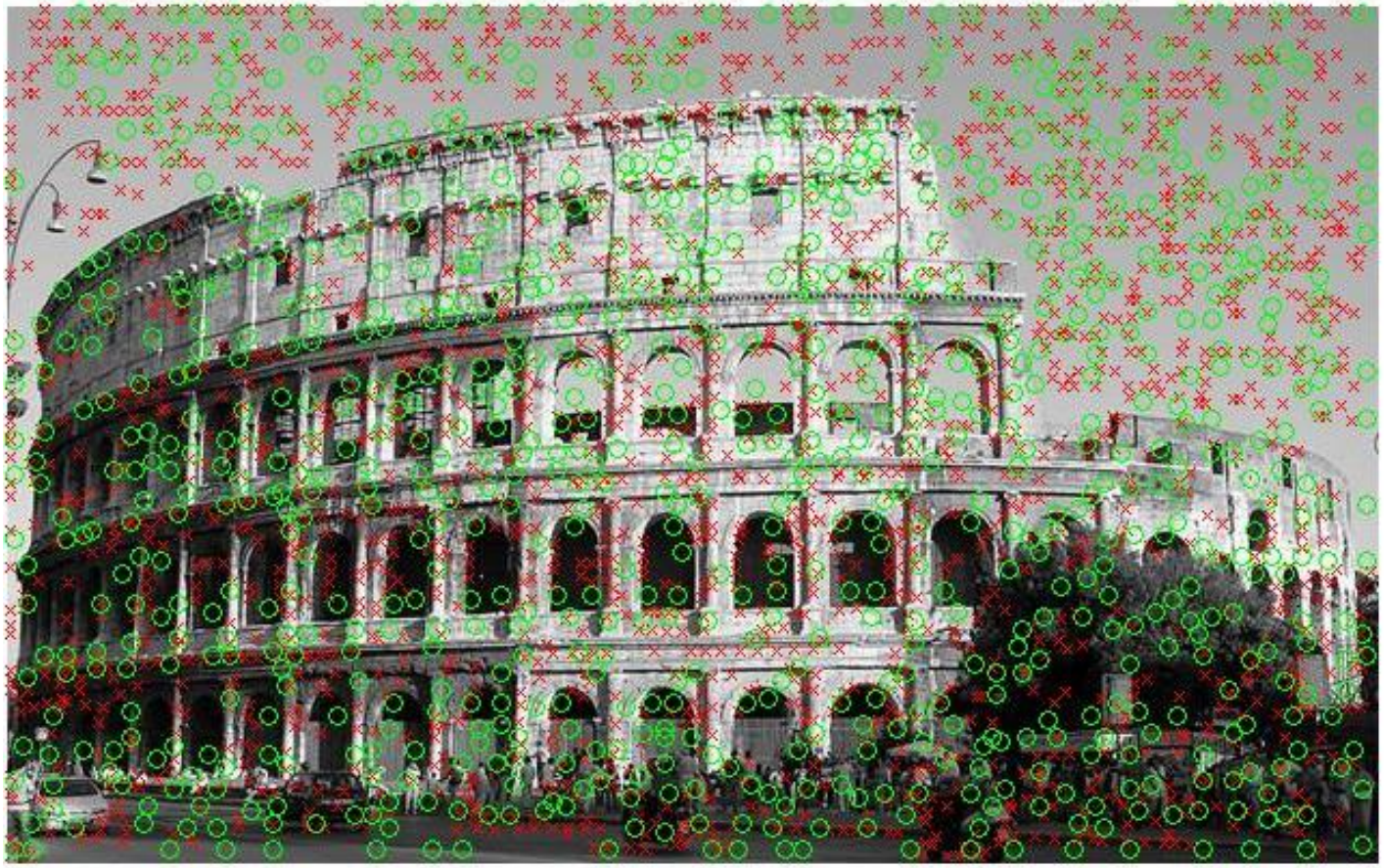
Corner detectors

- SIFT procedures
- 1st scale



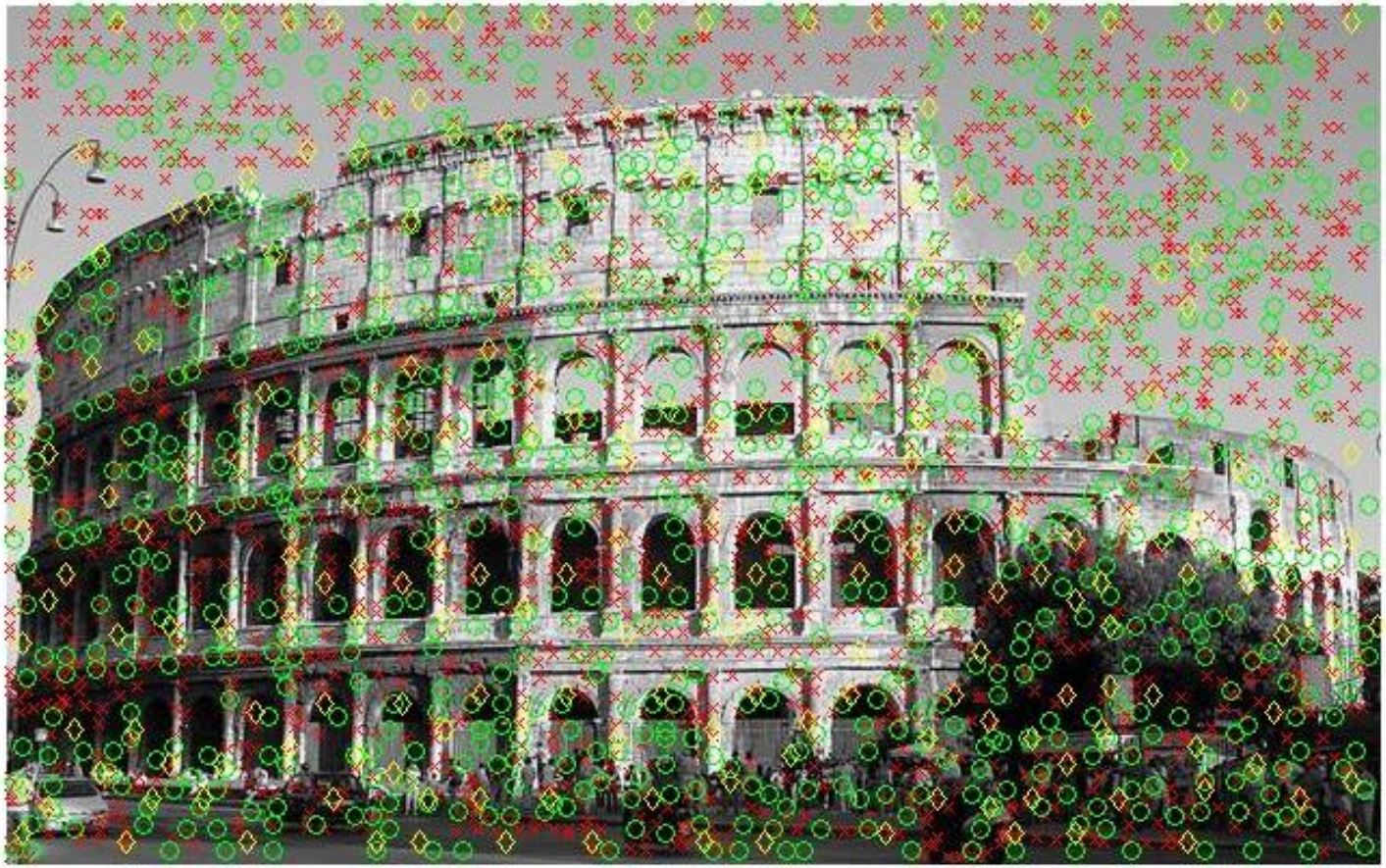
Corner detectors

- SIFT procedures
- 2nd scale



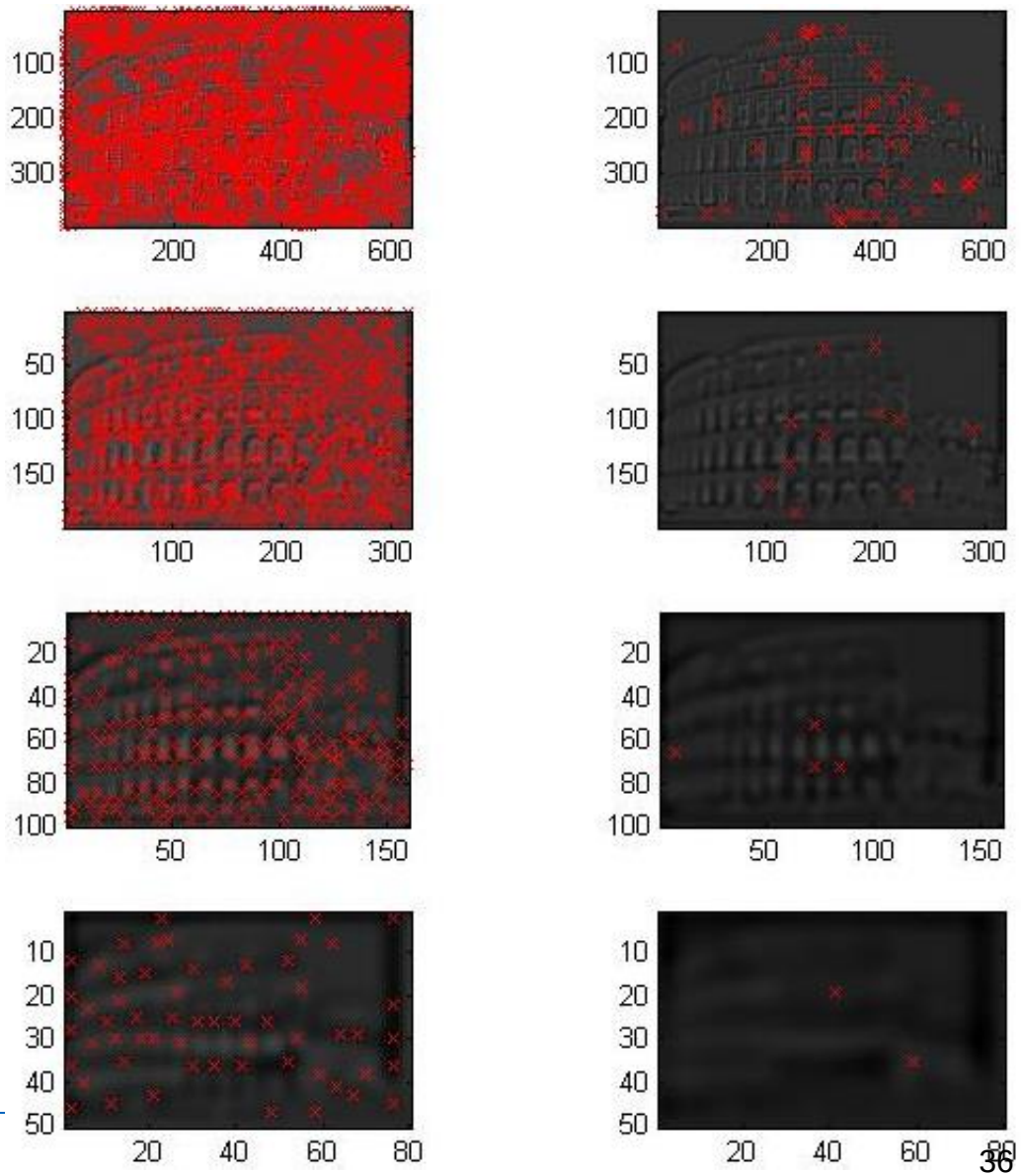
Corner detectors

- SIFT procedures
- 3rd scale



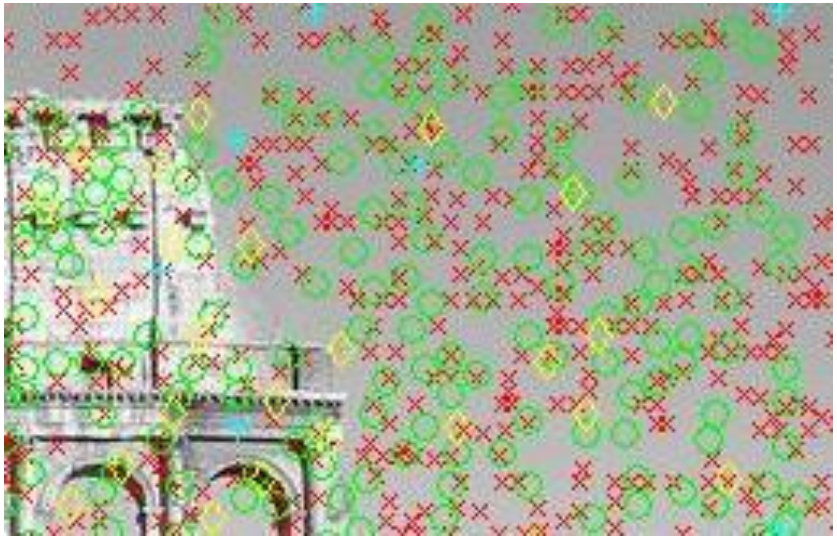
Corner detectors

- SIFT procedures
- Example of detected corners when examining difference of Gaussians

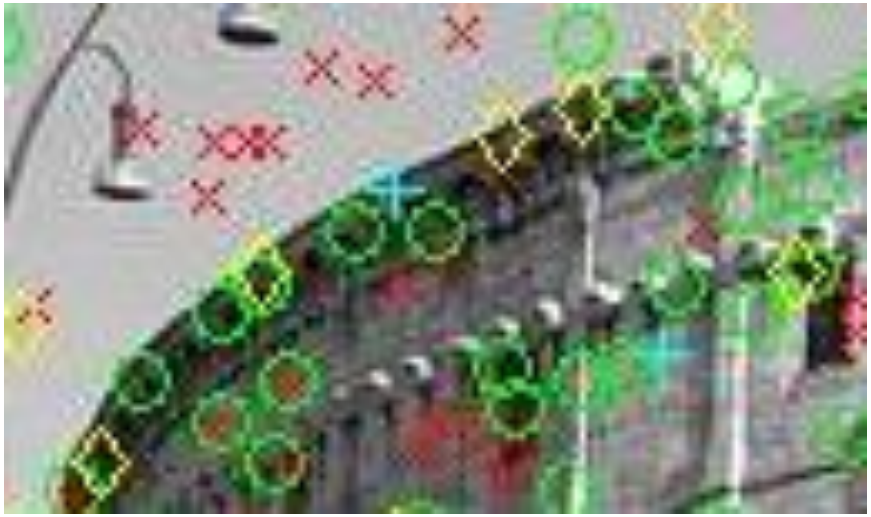


Corner detectors

- Some features are ‘weak’: not enough distinction from neighboring pixels



- Some features lie along edges



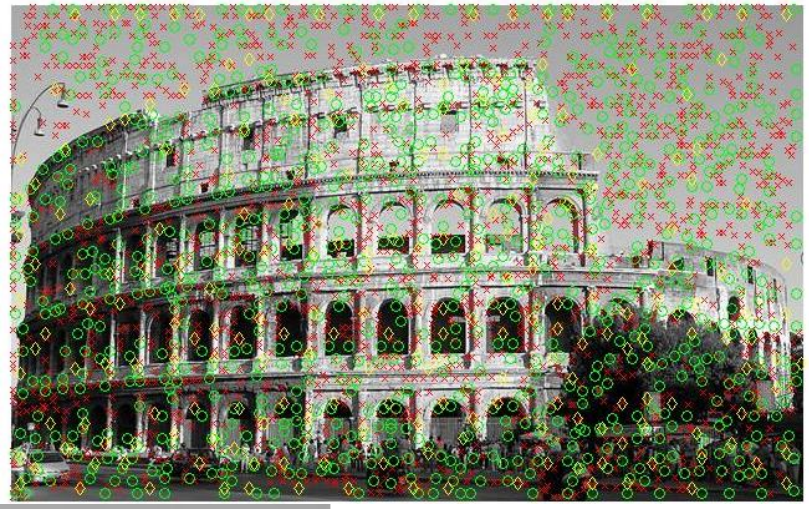
Corner detectors

- Removal of weak features

$$\left|\frac{\partial D(x,y,\sigma)}{\partial \mathbf{x}}\right| < \mathrm{w_{th}} \qquad \mathbf{x} = \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Corner detectors

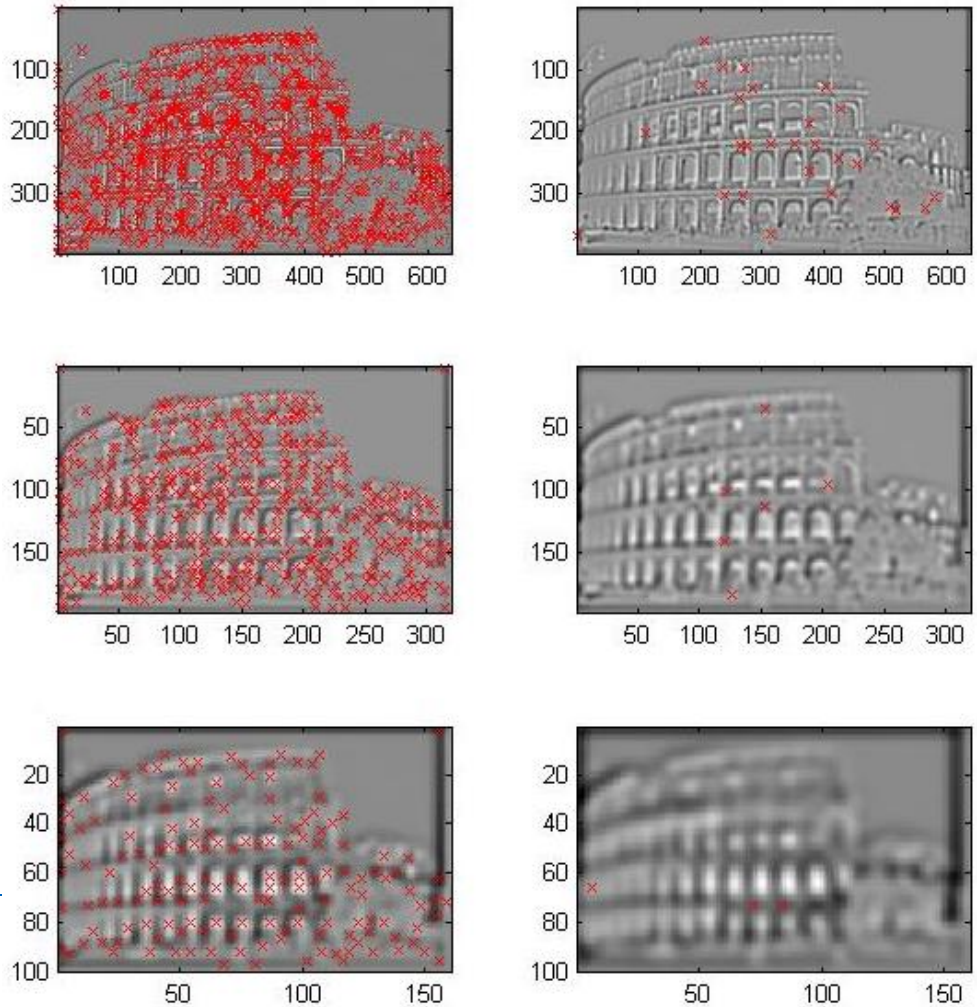
- Removal of weak features

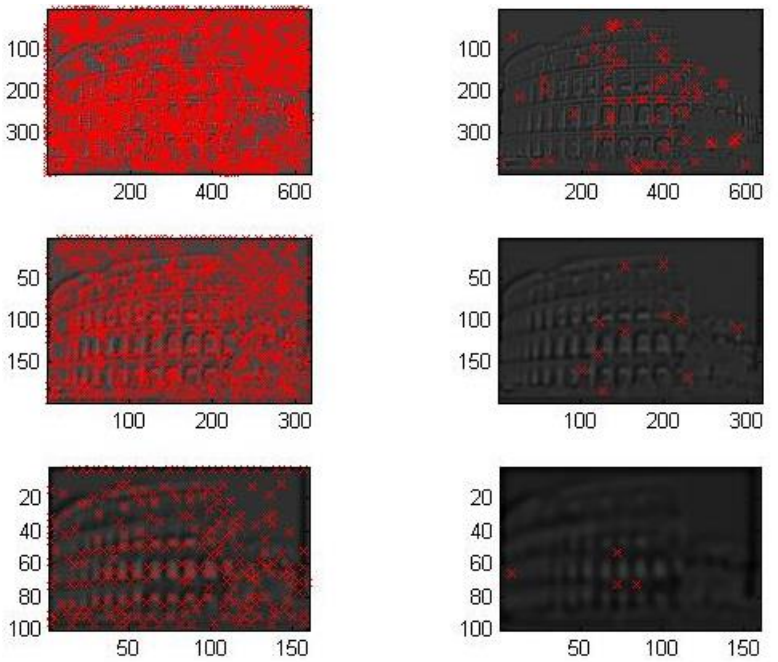


Original

Corner detectors

- Removal of weak features





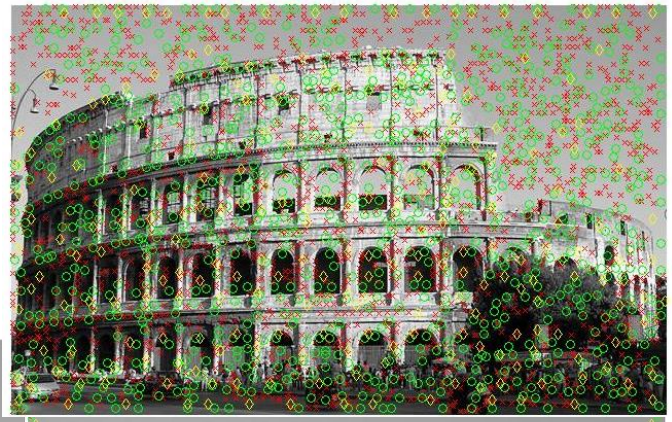
Original

Corner detectors

- Removal of edges
- Compute Hessian $$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} D_{xx}(\mathbf{x}) & D_{xy}(\mathbf{x}) \\ D_{yx}(\mathbf{x}) & D_{yy}(\mathbf{x}) \end{bmatrix}$$
- Compute combination of eigenvalues $$\det \mathbf{H} = \lambda_{\max}\lambda_{\min}$$ $$\mathrm{tr}\mathbf{H} = \lambda_{\max} + \lambda_{\min}$$
- Ratio (curvature): $$r = \frac{\lambda_{\max}}{\lambda_{\min}}$$
- Set threshold r_{th}
- Discard corner if $$\frac{\mathrm{tr}^2\mathbf{H}}{\det \mathbf{H}} < r_{\mathrm{th}}$$

Corner detectors

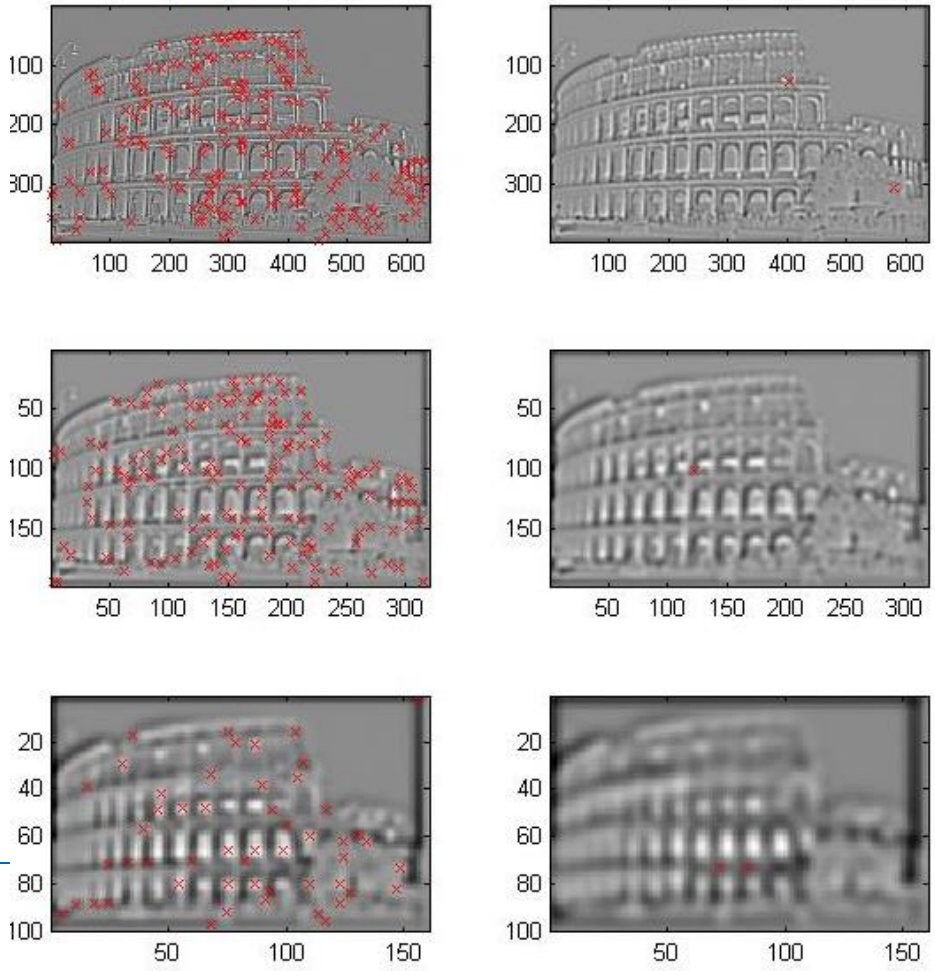
- Removal of edges

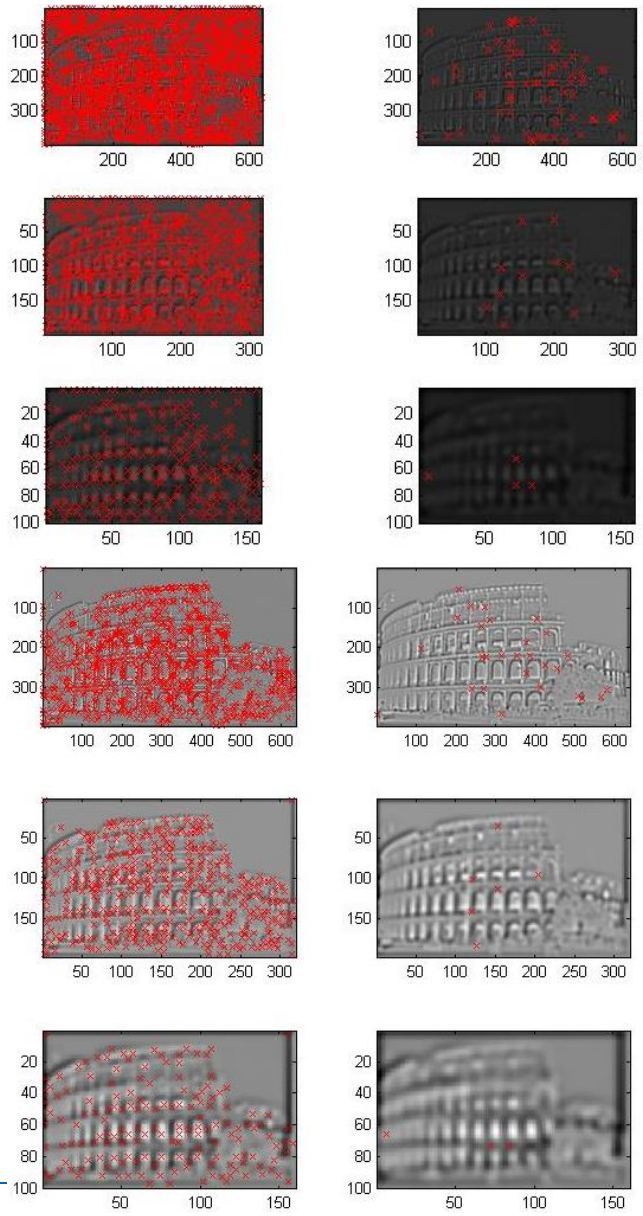


Originals

Corner detectors

- Removal of edges





Originals

Corner detectors

- Panoramic of invariant features detectors:

- *Harris corner detector*
- *Minimum eigenvalues algorithm*
- *FAST (Features from Accelerated Segment Test)*
- *BRISK (Binary Robust Invariant Scalable Keypoints)*
- *SURF (Speeded Up Robust Features)*

Corner detectors

- Harris corner detector

Corner detectors

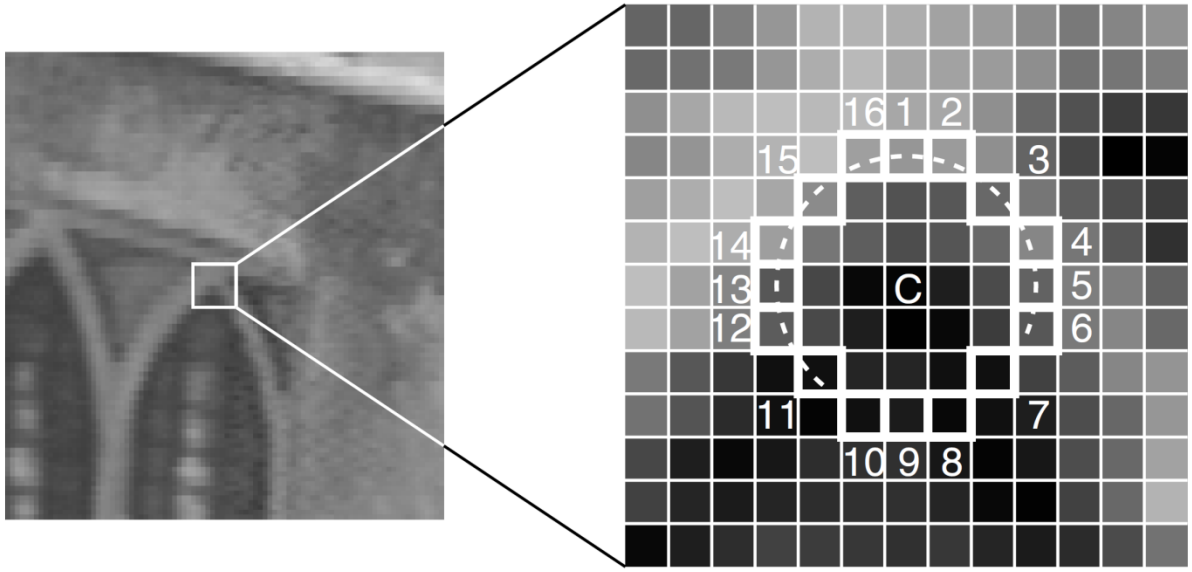
- Minimum eigenvalues algorithm
- Shi-Tomasi / Kanade-Tomasi detector: $\lambda_{\min} > \lambda_{\text{th}}$
- 1-pass Gaussian filter to smooth the gradient of the input image
- Not scale invariant

Corner detectors

- Minimum eigenvalues algorithm

Corner detectors

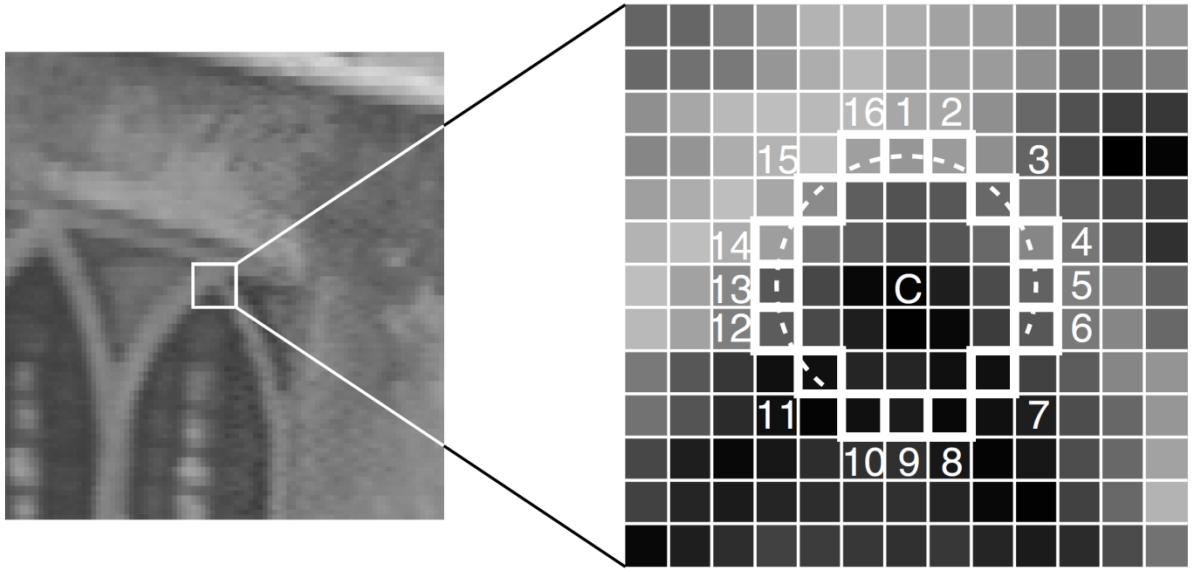
- FAST (Features from Accelerated Segment Test)
- Detection test for a feature at a pixel *p* by examining a circle of 16 pixels (a Bresenham circle of radius 3) surrounding *p*
- A feature is detected at *p* if the intensities of at least 12 contiguous pixels are all above or all below the intensity of *p* by some threshold *t* *(12-16 mask)*.



Credit: Rosten E., Drummond, T., "Fusing Points and Lines for High Performance Tracking"

Corner detectors

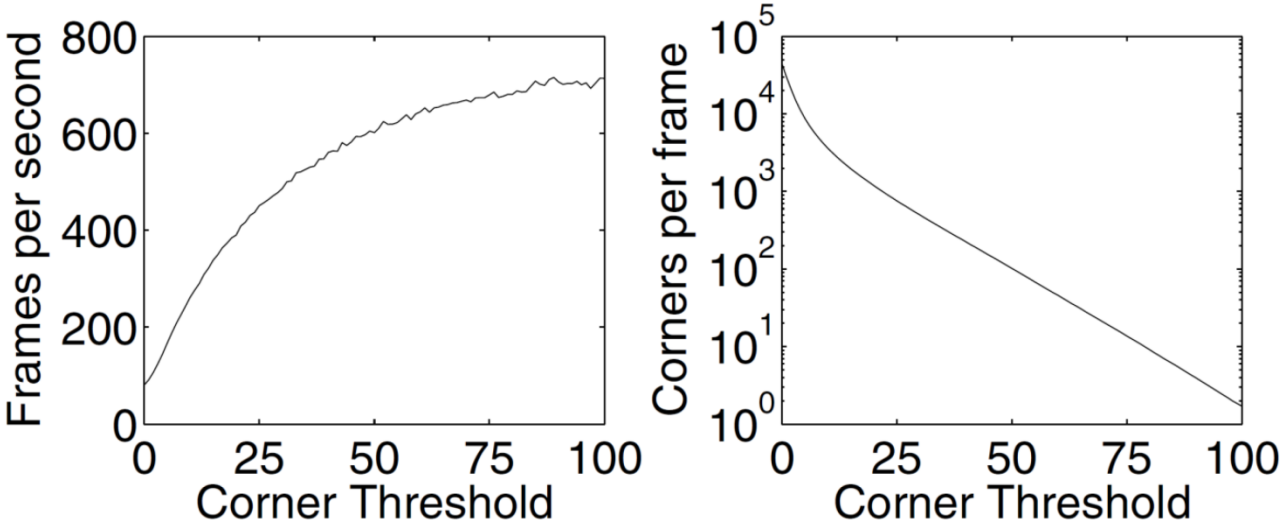
- FAST (Features from Accelerated Segment Test)
- The test is accelerated by examining pixels 1, 9, 5 and 13, since a feature can only exist if three of these test points are all above or below the intensity of *p* by the threshold.
- Features categorized as positive (where the pixels are greater than the center) and negative
- Fast, <u>but not scale invaraint</u>



Credit: Rosten E., Drummond, T., "Fusing Points and Lines for High Performance Tracking"

Corner detectors

- FAST (Features from Accelerated Segment Test)
- Optimization of threshold for speed vs detection



Credit: Rosten E., Drummond, T., "Fusing Points and Lines for High Performance Tracking"

Corner detectors

- FAST (Features from Accelerated Segment Test)

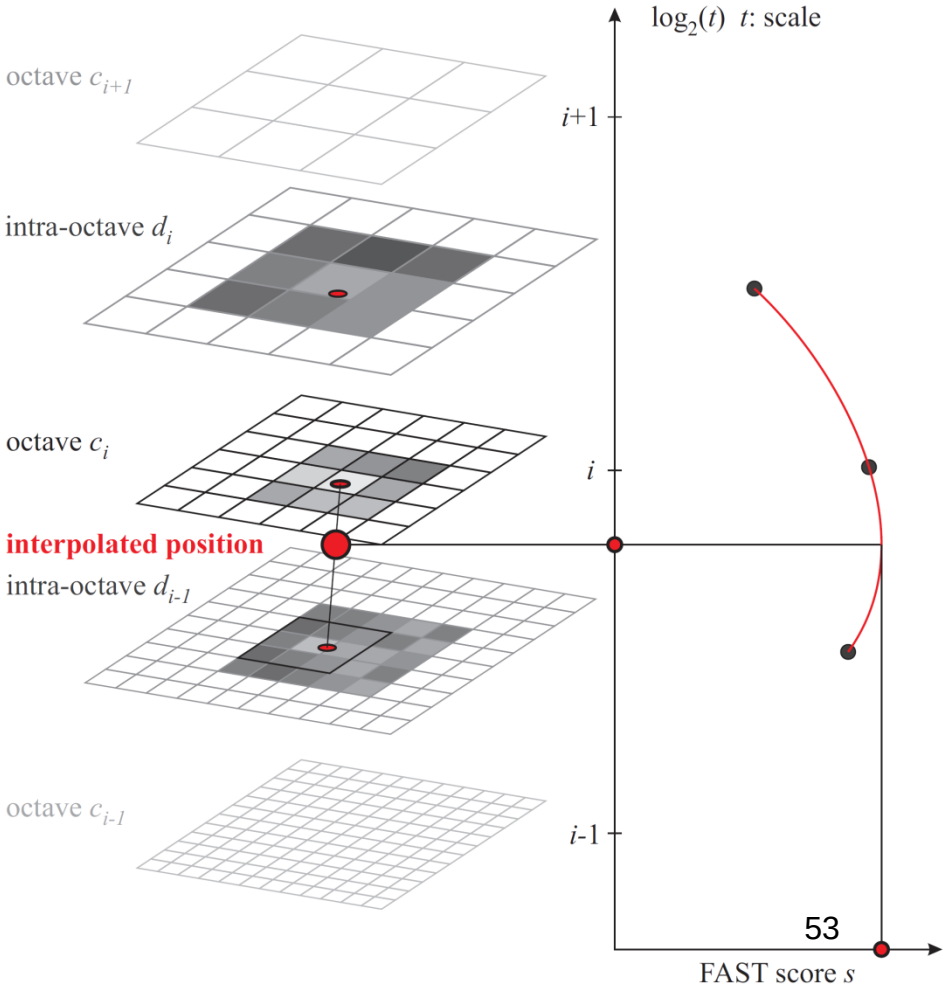
Corner detectors

- BRISK (Binary Robust Invariant Scalable Keypoints)
- Adaptation of FAST to obtain scale-invariance: *n* octaves (each half-sampled from the previous) and *n* intra-octaves (each down-sampled by 1.5 w.r.t previous octave)
- FAST is applied to each layer with a 9-16 mask
- Features with max saliency (FAST score: difference between intensity at keypoint and 16 surrounding pixels) are detected also in scale

Corner detectors

- BRISK (Binary Robust Invariant Scalable Keypoints)

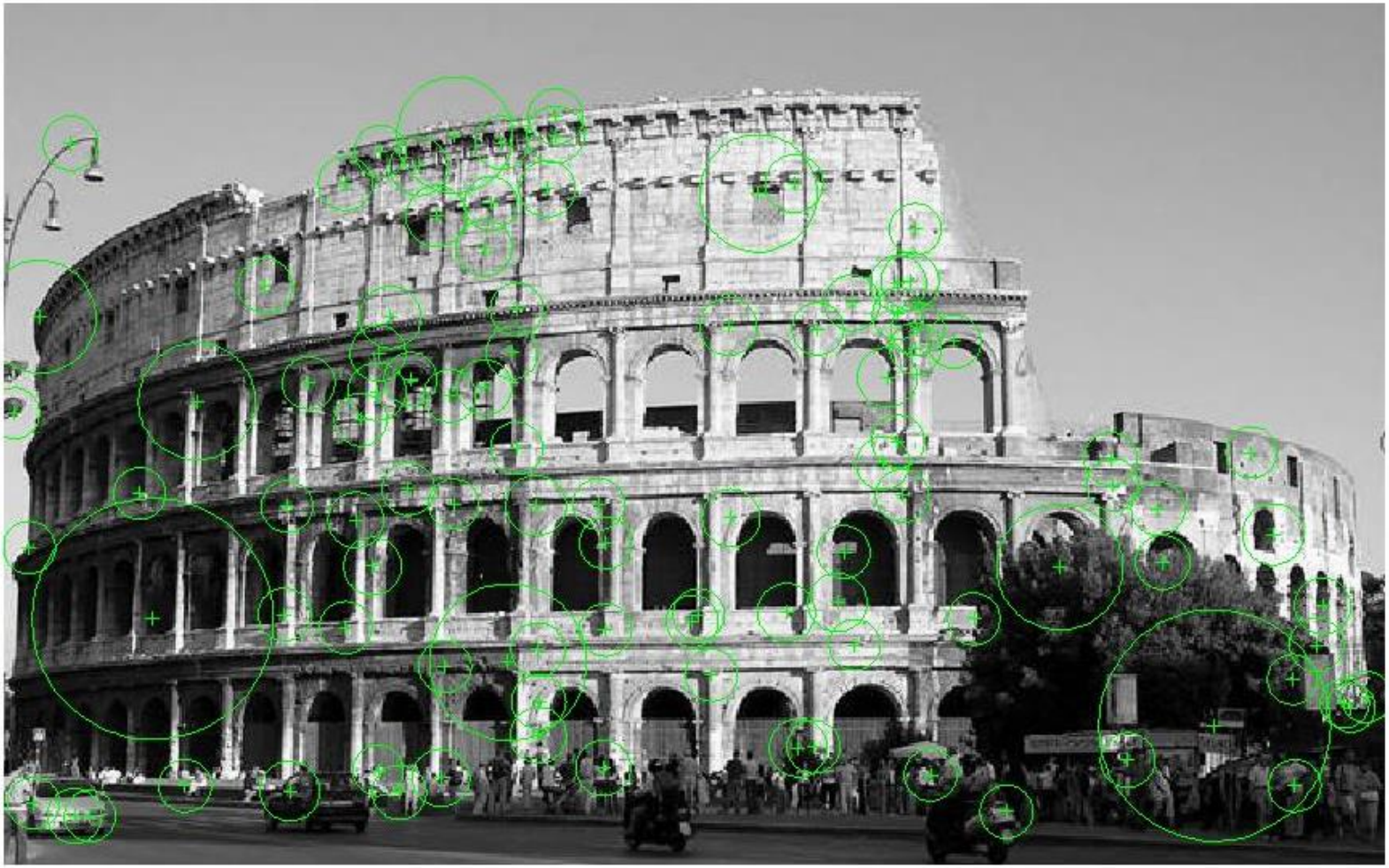
- *Sub-pixel accuracy with second-degree interpolation*



Credit: Leutenegger, S., Chli, M., Siegwart, R.Y.,"BRISK: Binary Robust Invariant Scalable Keypoints"

Corner detectors

- BRISK (Binary Robust Invariant Scalable Keypoints)

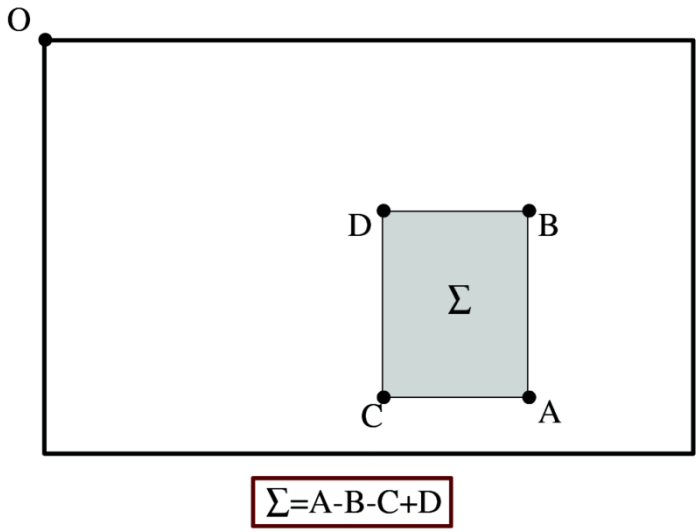


Corner detectors

- SURF (Speeded Up Robust Features)

- Similar in concept to SIFT, operates with the integral image (sum of all pixels in the input image *I* within a rectangular region formed by the origin and x.) for computational speed

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{i \leq y} I(i,j)$$



Credit: Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., "Speeded-Up Robust Features (SURF)"

- Select features at different scales by examining the Hessian of the difference of Gaussians

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} D_{xx}(\mathbf{x}) & D_{xy}(\mathbf{x}) \\ D_{yx}(\mathbf{x}) & D_{yy}(\mathbf{x}) \end{bmatrix}$$

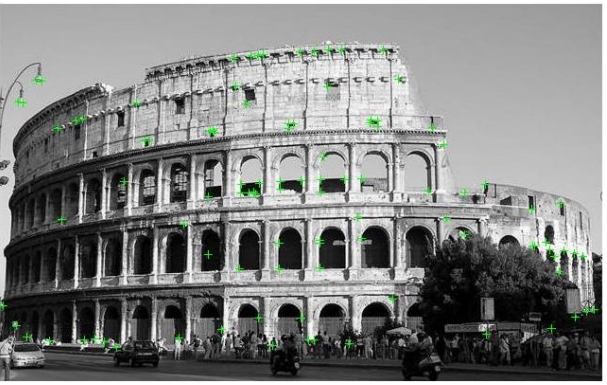
Corner detectors

- SURF (Speeded Up Robust Features)

Corner detectors

- Panoramic

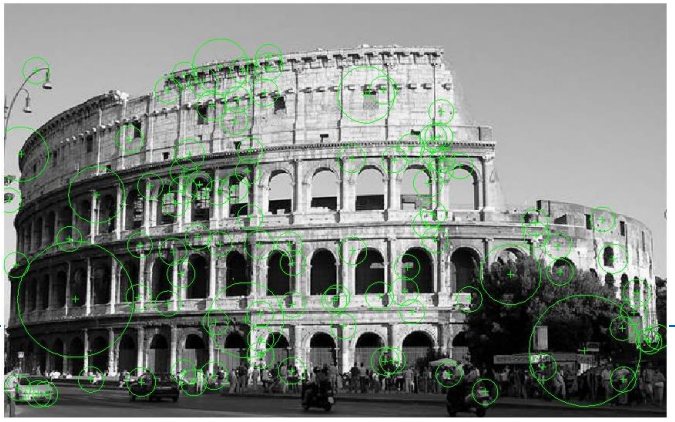
Harris



Min Eigenvalue

FAST

BRISK



SURF

Descriptors

- How to “tag” image features?
- We must be able to describe as uniquely as possible each detected feature
- Information about the region surrounding the feature is extracted
- The information is contained in a ‘feature vector’ containing the necessary distinctive “qualities” of the detected feature

Descriptors

- SIFT: for each keypoint at scale σ the gradient magnitude and directions are computed:

Magnitude:

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

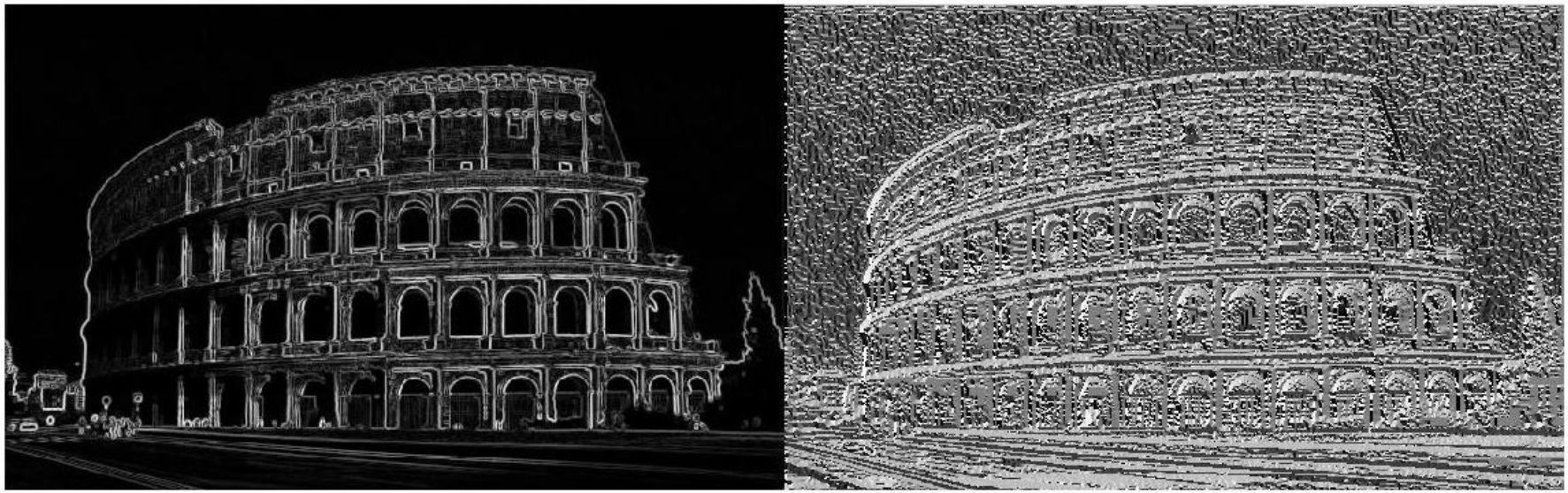
Direction:

$$\theta(x,y) = \arctan \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

Descriptors

- *Example of magnitude and direction at each pixel*

Gradient Magnitude (left), and Gradient Direction (right)



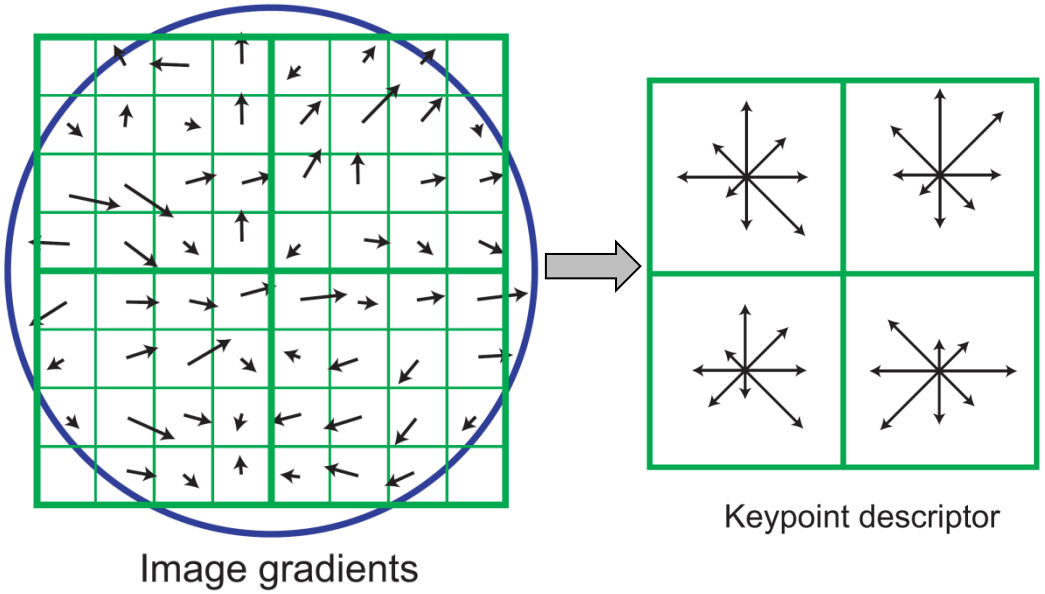
Lighter for higher magnitudes

Angles - π to π measured counterclockwise from the positive x-axis.

Descriptors

- SIFT feature vector : example of construction
  1) compute gradient directions in a region around the keypoint
  2) each contribution from pixels within the region is weighted by its gradient magnitude and by a Gaussian-weighted circular window with $\sigma = 1.5\sigma_{\text{keypoint}}$

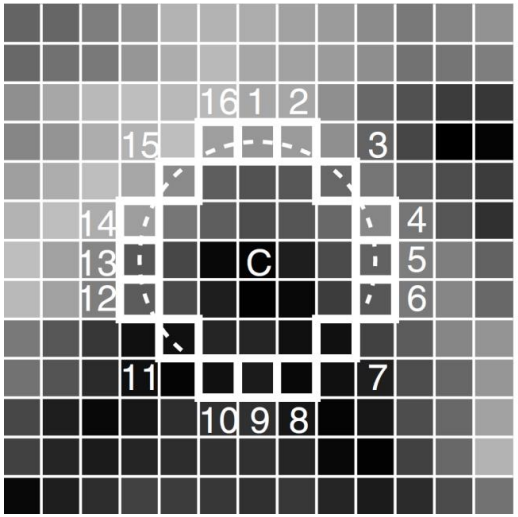
  3) Samples accumulated into orientation histograms summarizing the contents over 4x4 subregions
  4) Build the 2x2x8 feature vector (descriptor)



Credit: Lowe, D. G.: "Distinctive Image Features from Scale-Invariant Keypoints"

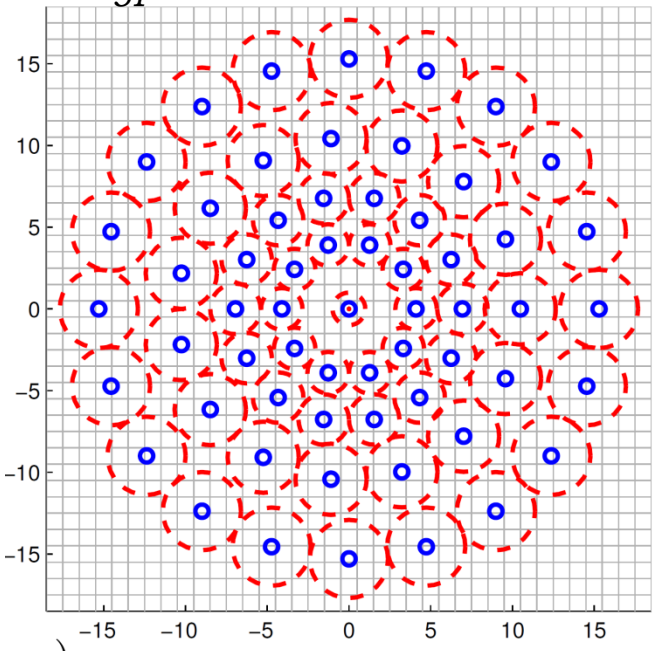
Descriptors

- FAST feature vector: intensity values of surrounding 16 pixels, plus positive (below intensity at keypoint)/negative (above intensity at keypoint) tag



Descriptors

Credit: Leutenegger, S., Chli, M., Siegwart, R.Y.,"BRISK: Binary Robust Invariant Scalable Keypoints"

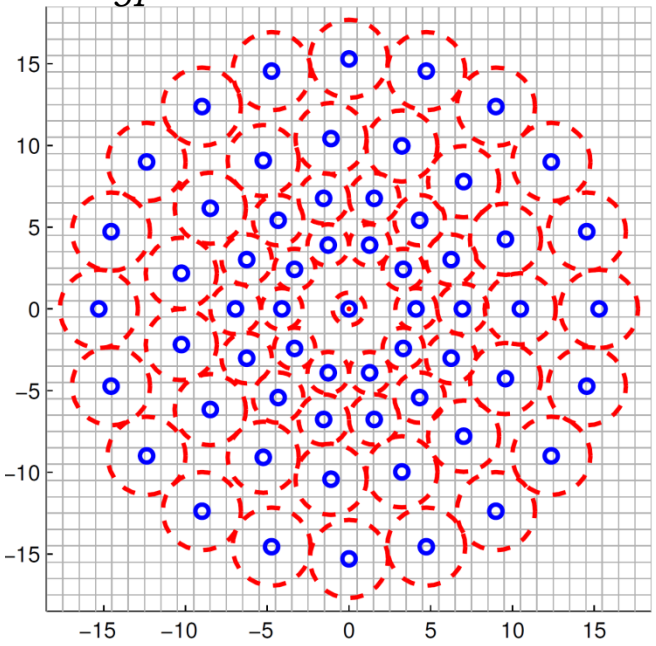


- BRISK feature vector: binary string obtained by concatenating the results of simple brightness comparison tests
- Select N sample points on concentric circles around keypoint (here *N=60*)
- Gaussian smoothing proportional to distance (red circles): σ_i
- Local gradient: $\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j) = \frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}(\mathbf{p}_j - \mathbf{p}_i)$
- Partition of sample points as

$$\mathcal{S} : \{(\mathbf{p}_i, \mathbf{p}_j) \mid \|\mathbf{p}_j - \mathbf{p}_i\| < 9.75t\}$$
$$\mathcal{L} : \{(\mathbf{p}_i, \mathbf{p}_j) \mid \|\mathbf{p}_j - \mathbf{p}_i\| > 13.67t\}$$

Descriptors

Credit: Leutenegger, S., Chli, M., Siegwart, R.Y.,"BRISK: Binary Robust Invariant Scalable Keypoints"



- BRISK feature vector: binary string obtained by concatenating the results of simple brightness comparison tests

- Define the overall characteristic pattern direction of keypoint examined as

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{\mathbf{p}_i, \mathbf{p}_j \in \mathcal{L}} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j)$$

- Rotate samples in $\mathcal{S}$ by $\alpha = \arctan(g_y / g_x)$

- Construct descriptor bit by bit:

$$\forall \mathbf{p}_i, \mathbf{p}_j \in \mathcal{S}$$

$$b = \begin{cases} 1 & I(\mathbf{p}_j^{\alpha}, \sigma_j) > I(\mathbf{p}_i^{\alpha}, \sigma_i) \\ 0 & \text{otherwise} \end{cases}$$

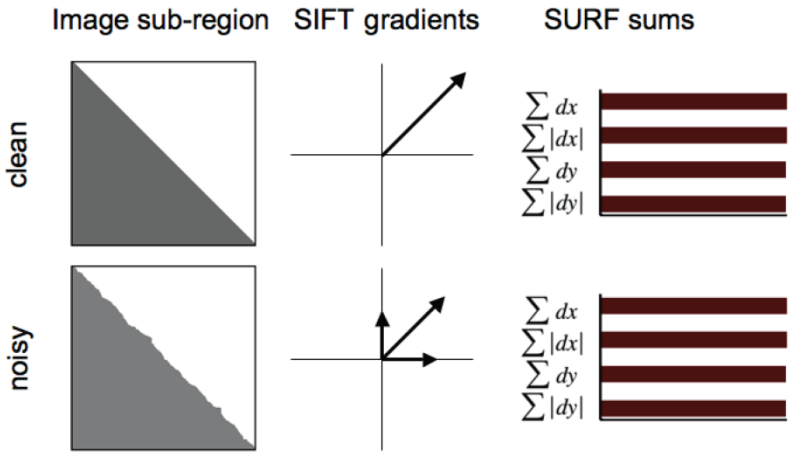
$$\mathcal{S} : \{(\mathbf{p}_i, \mathbf{p}_j) \mid \|\mathbf{p}_j - \mathbf{p}_i\| < 9.75t\}$$

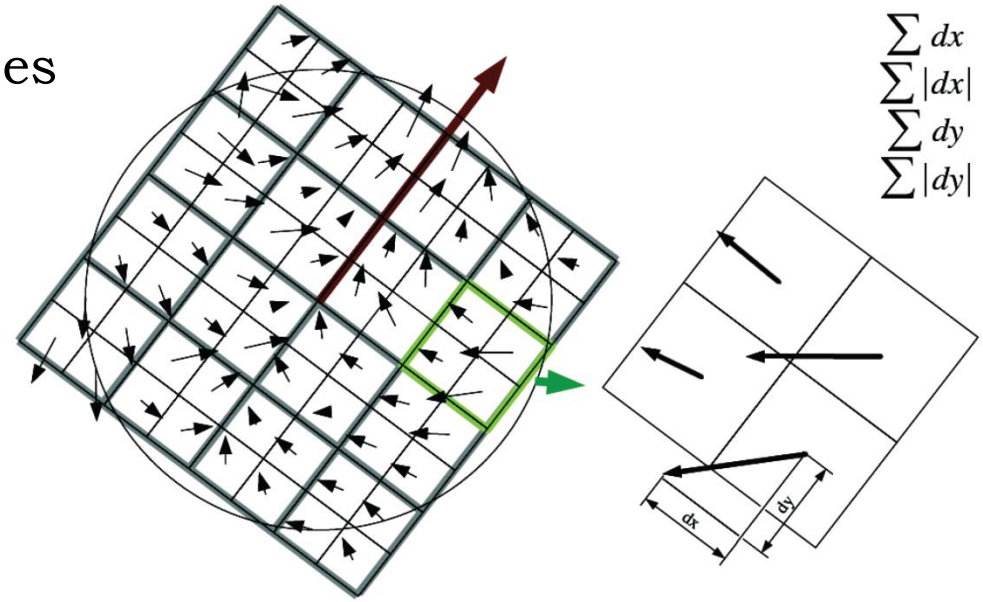
$$\mathcal{L} : \{(\mathbf{p}_i, \mathbf{p}_j) \mid \|\mathbf{p}_j - \mathbf{p}_i\| > 13.67t\}$$

Descriptors

- SURF feature vector: similar in concept to SIFT

- First order Haar wavelet responses instead of image gradient used in SIFT





Credit: Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., "Speeded-Up Robust Features (SURF)"

Feature matching

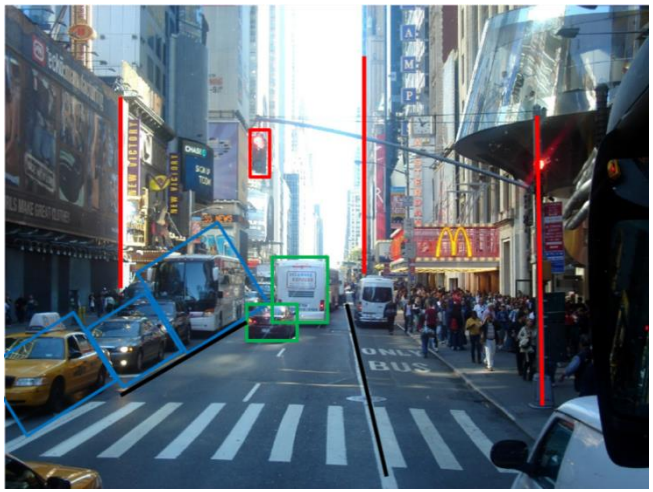
- Matching: comparison between two databases to extract likely common features descriptors
- Problem: due to noise, motion, etc.. the same feature won't have exactly same description in the two databases
- Depending on the application, we will have a large subset of matching features:


Feature matching

- Matching: comparison between two databases to extract likely common features descriptors

- Problem: due to noise, motion, etc.. the same feature won't have exactly same description in the two databases

- Depending on the application, we will have a large subset of matching features, or a very small one (cluttered scenes):

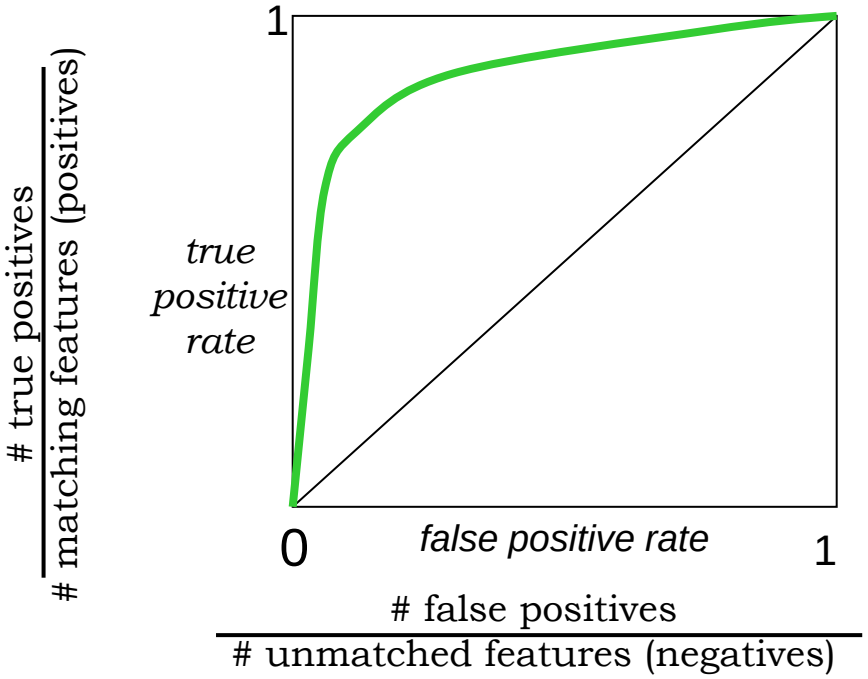


Feature matching

- The best candidate match for each keypoint is found by identifying its nearest neighbor in the database of keypoints
- Extensive search is potentially slow: use of approximated methods that return a closest neighbor with high probability OR use indexing/hashing strategies.
- Speed increases when matches are known to be close in the image space (small difference between images): search around the image coordinates of each point
- There could be false positives (matching of non-corresponding features) and false negatives (rejected matching of corresponding features). This problem is addressed by applying robust estimation procedures (introduced later in this course), e.g., RANSAC

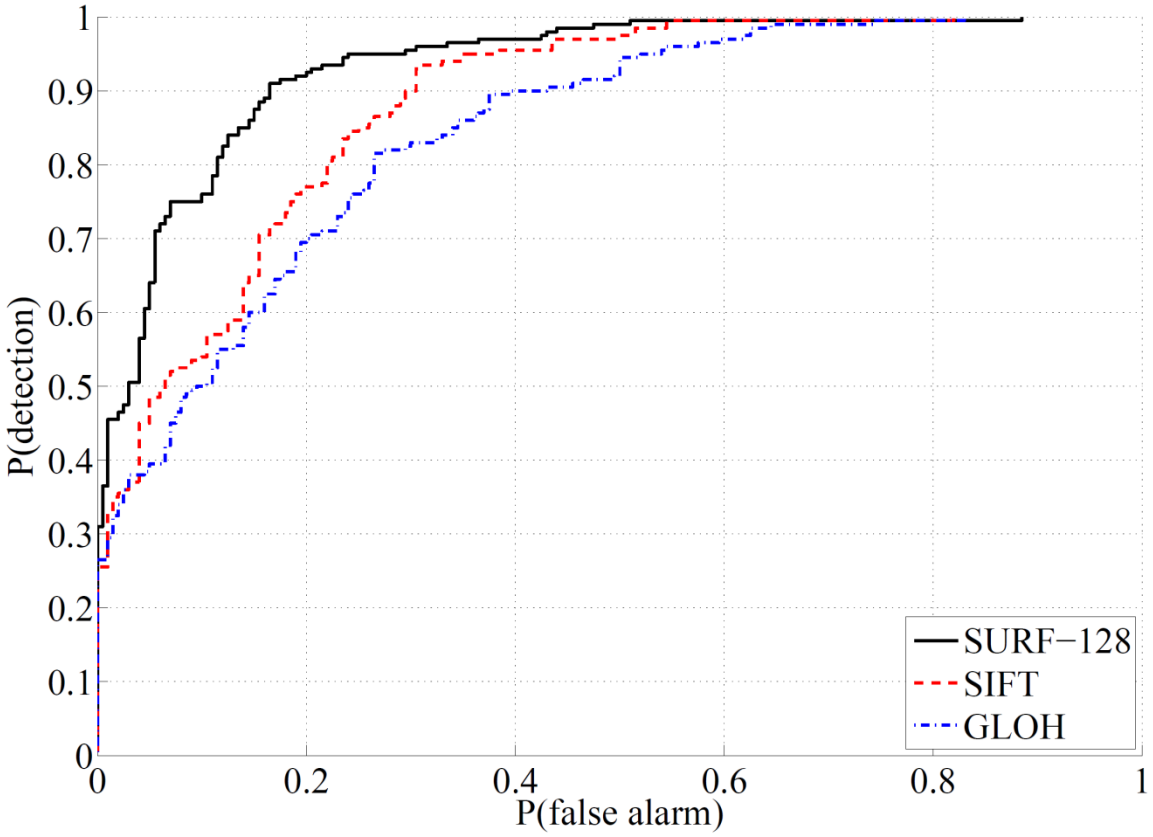
Feature matching

- Simplest approach: SSD (Sum of Squared Distances) between feature vectors
- Refinement: $\frac{SSD_{\text{best}}}{SSD_{2^{\text{nd}}\text{best}}}$ (small for ambiguous matches)
- Evaluation of matching strategy: ROC (Receiver Operator Characteristic) curve
- The larger the area under the curve, the better the performance



Feature matching

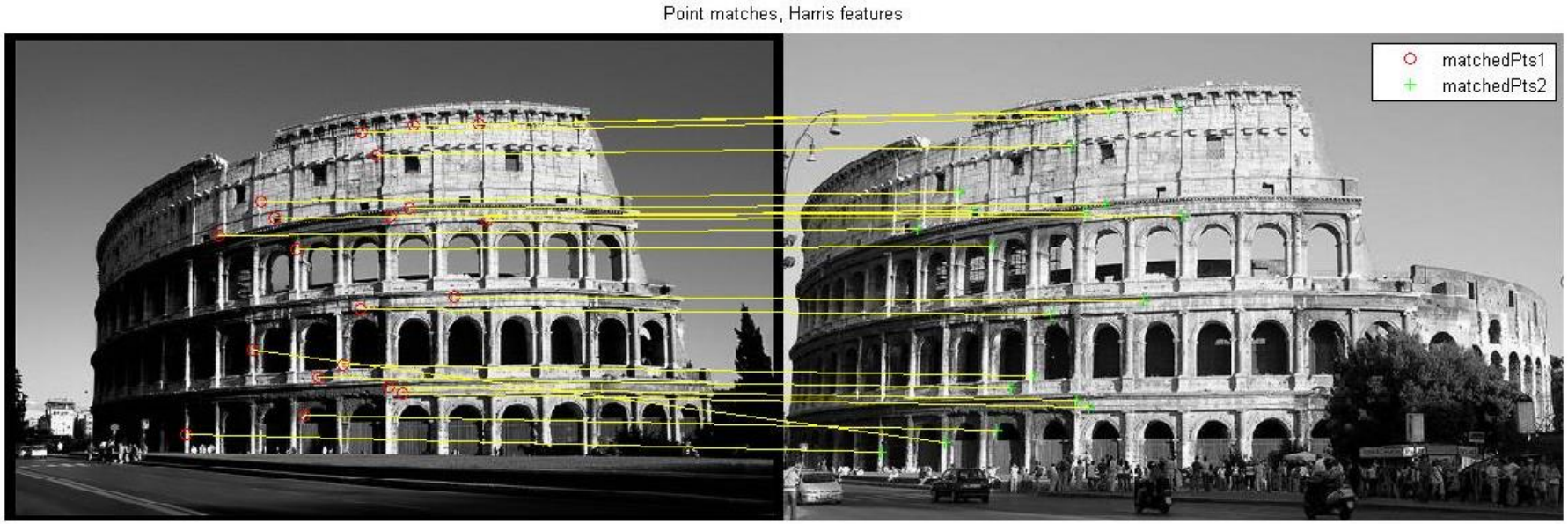
- Example: SURF vs SIFT



Credit: Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., "Speeded-Up Robust Features (SURF)"

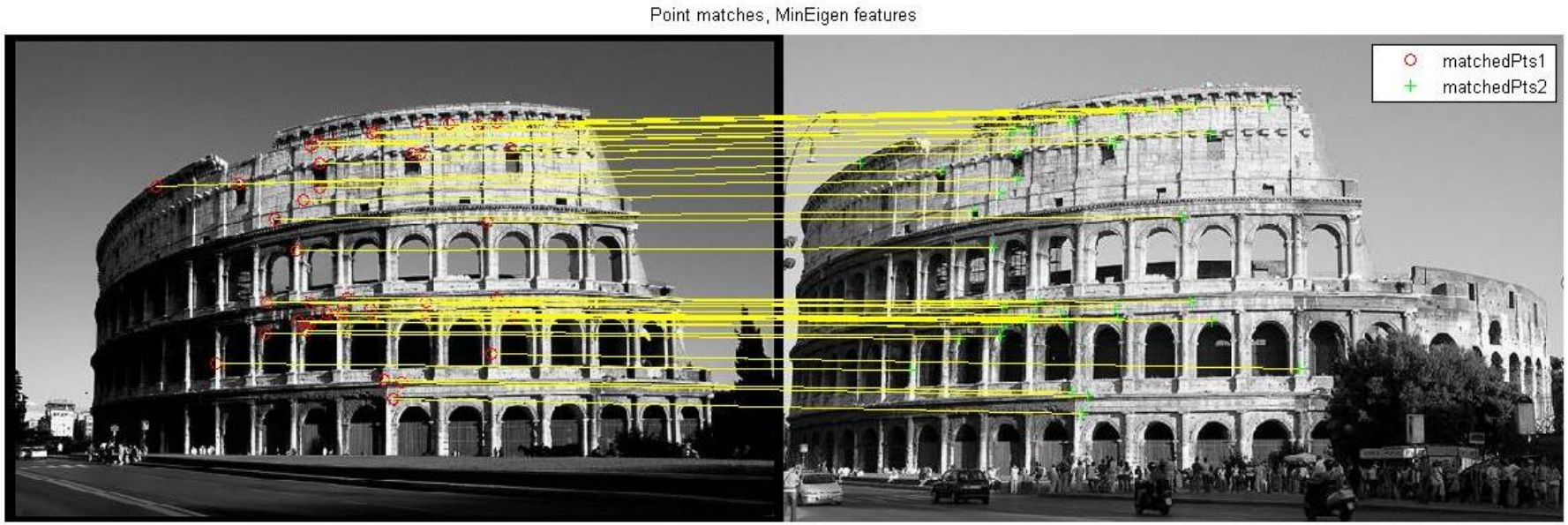
Some feature matching examples

- Harris



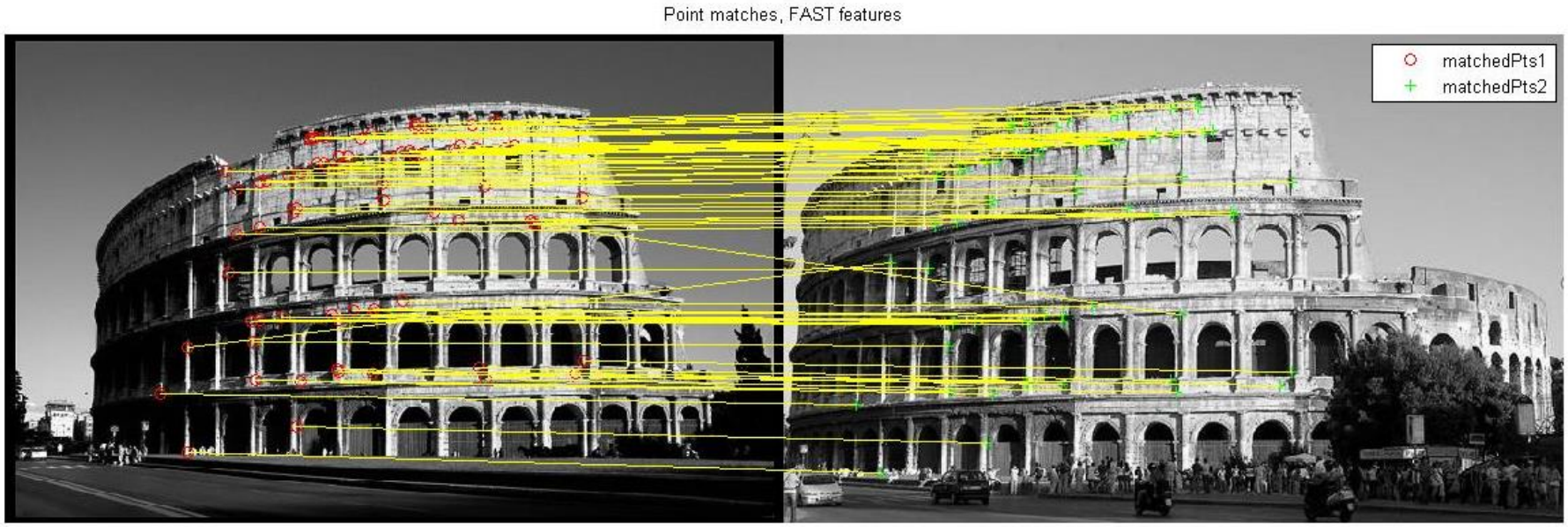
Some feature matching examples

- Minimum eigenvalue



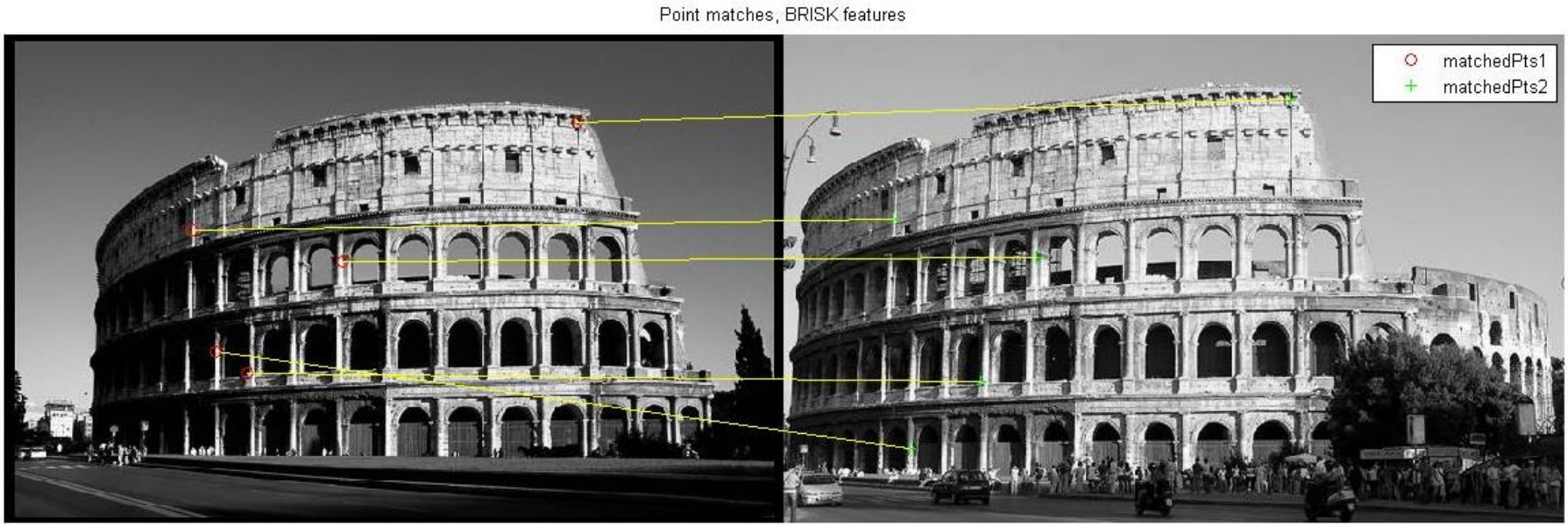
Some feature matching examples

- FAST



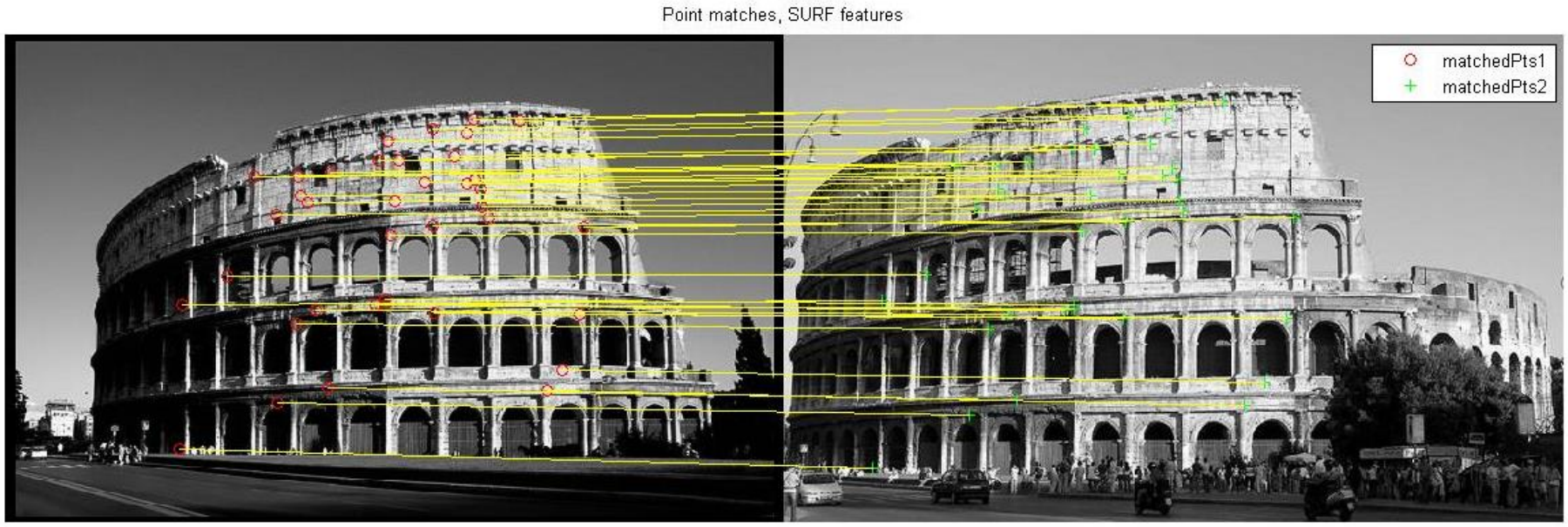
Some feature matching examples

- BRISK



Some feature matching examples

- SURF



Feature tracking

- Instead of independently finding features in all candidate images and then matching, find a set of likely feature locations in subsequent images
- Main application: video sequences (small displacements between subsequent features)
- Benefits from a-priori knowledge of camera motion
- Most renowned algorithm: KLT (Kanade-Lucas-Tomasi)

Final notes

- Computer vision classes:

Dr. M. Kleinsteuber
Computer Vision [(EI2223 (old) / EI7120 (new)]
Lehrstuhl für Datenverarbeitung

Prof. Cremers
Computer Vision Group
Informatik IX