SSY097 - Image Analysis

Lecture 3 - Scale-Invariant Local Features

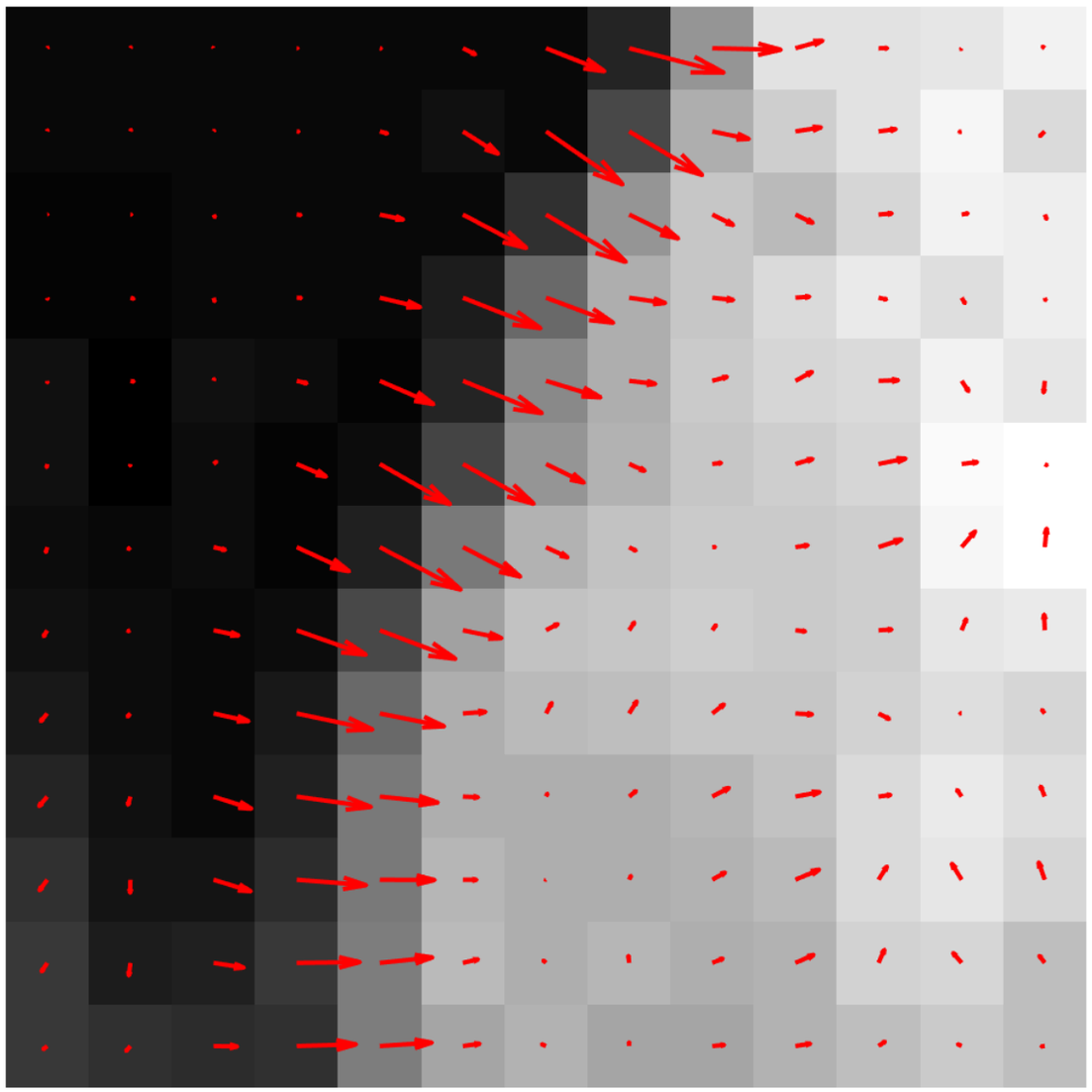
Torsten Sattler
(slides adapted from Olof Enqvist)

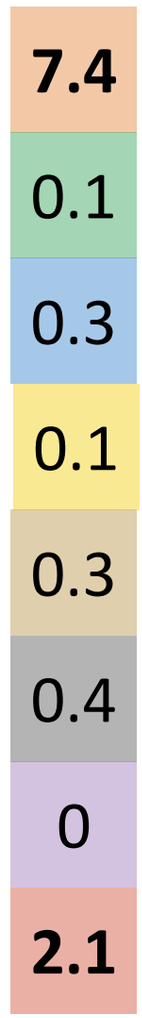
Last Lecture

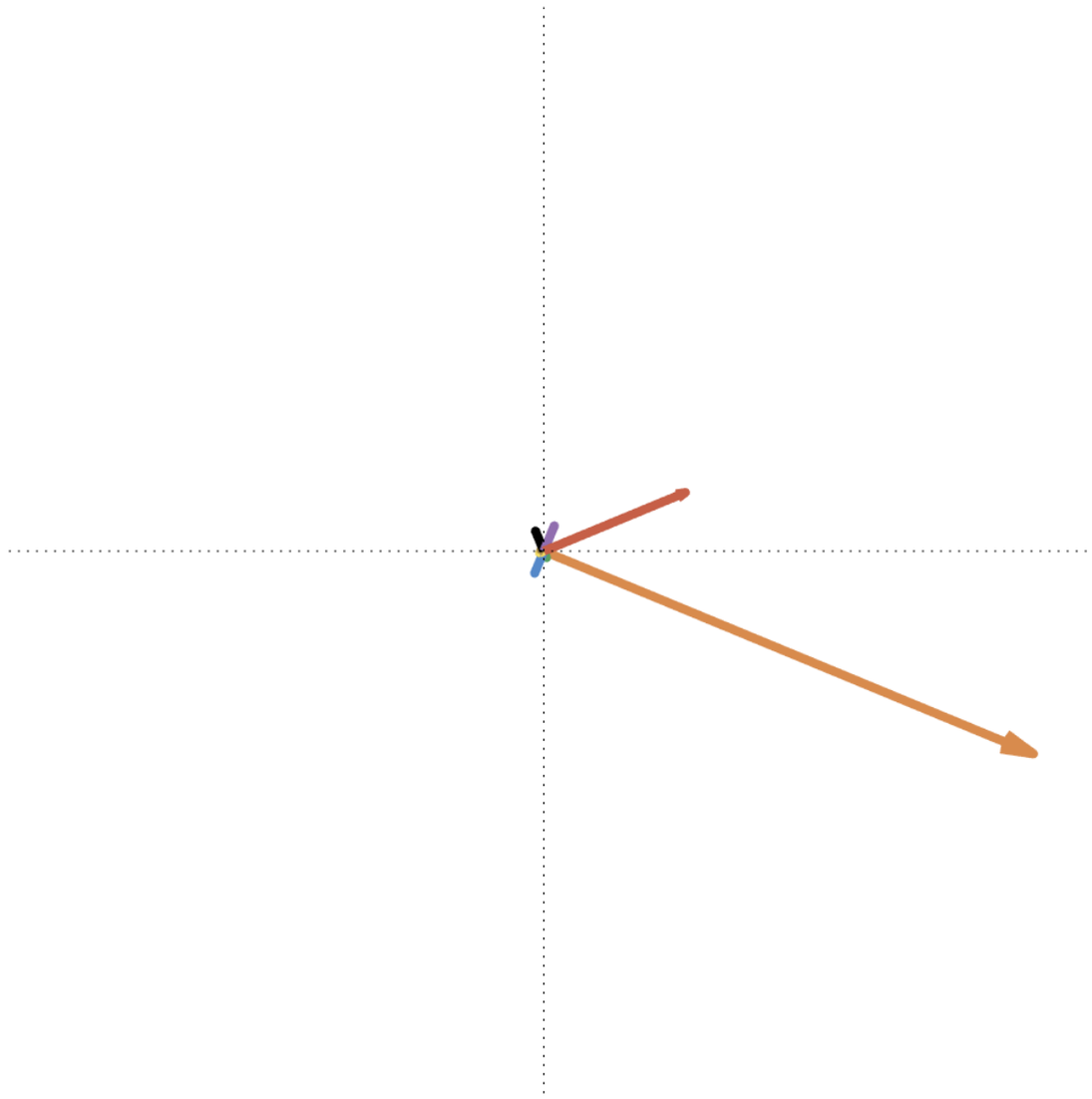
Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	**Filtering, gradients, scale**	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	TBA	
Mar. 9	TBA	

Last Lecture

Gradient histograms

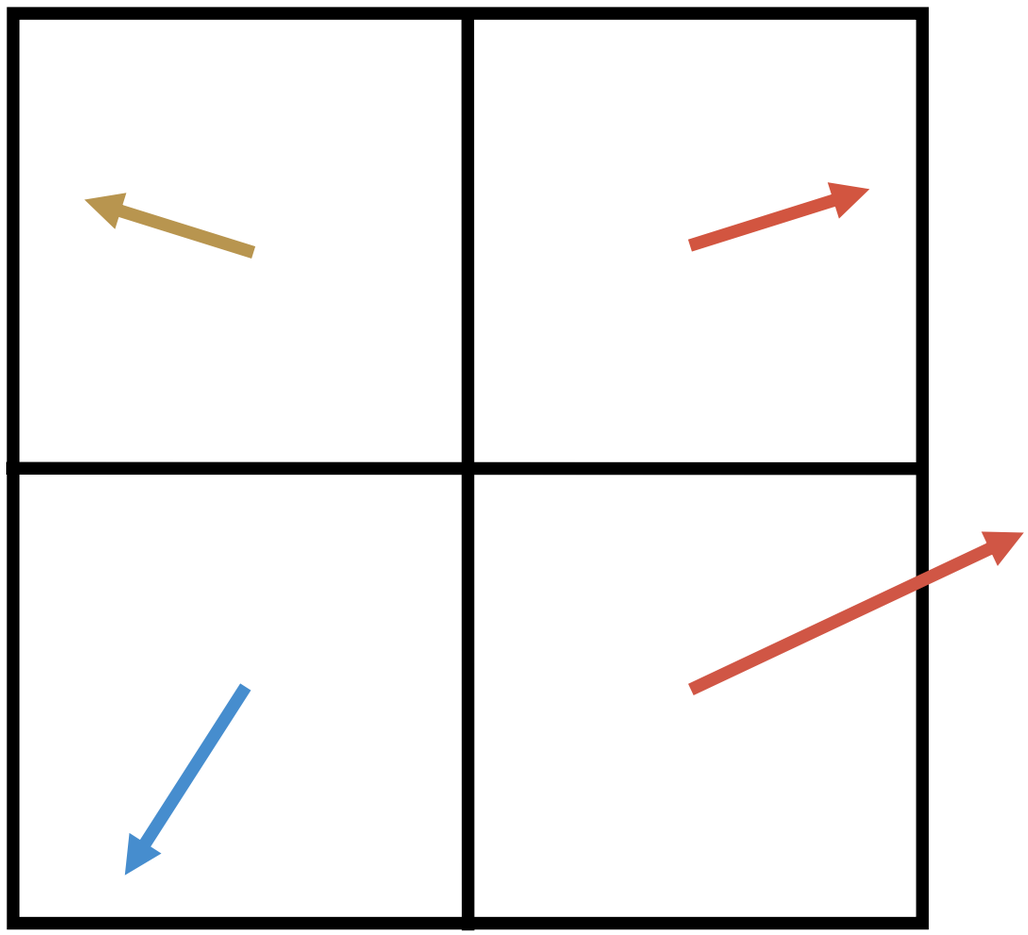






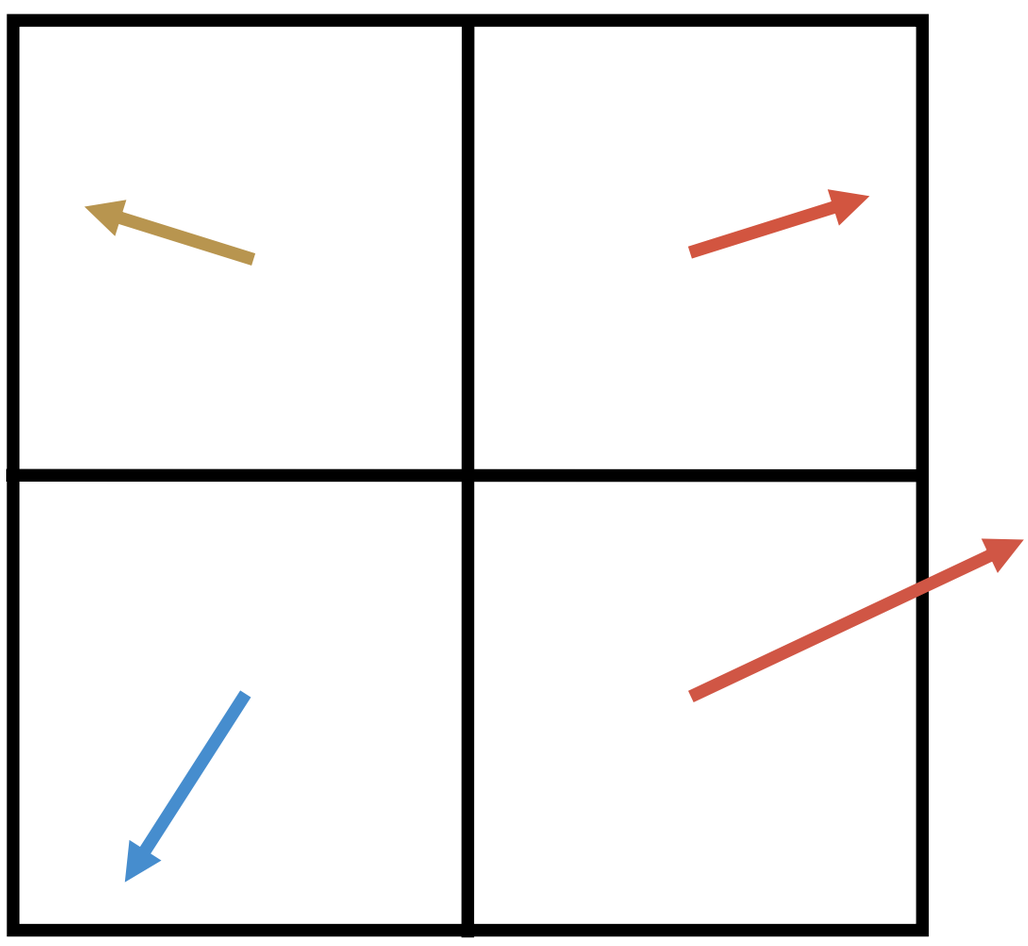
Last Lecture

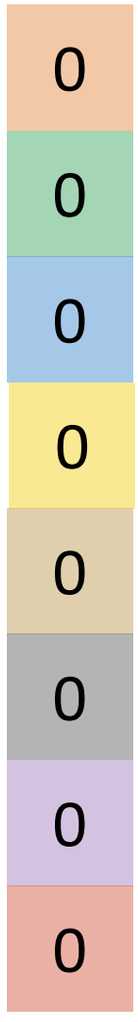
Gradient histograms

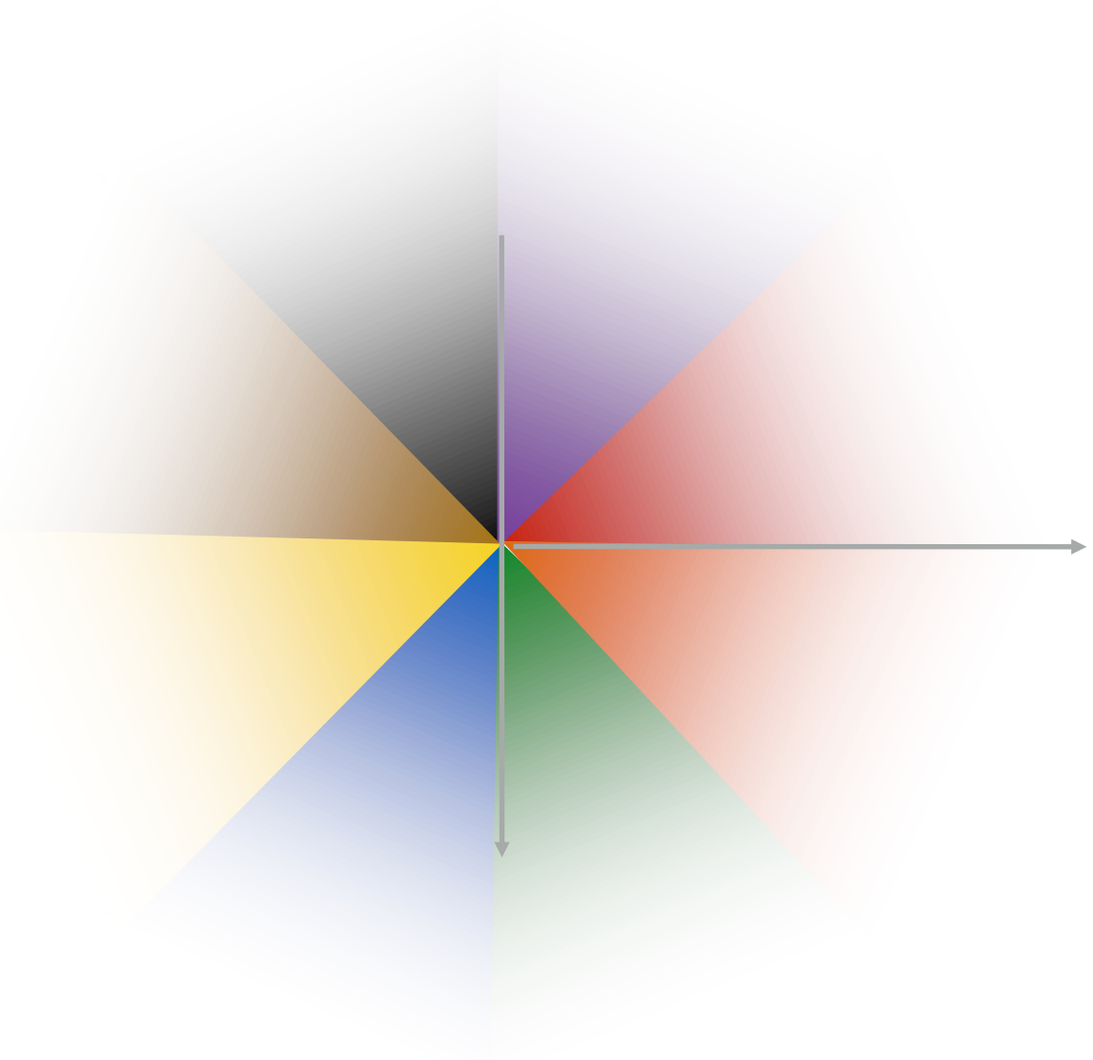


Last Lecture

Gradient histograms

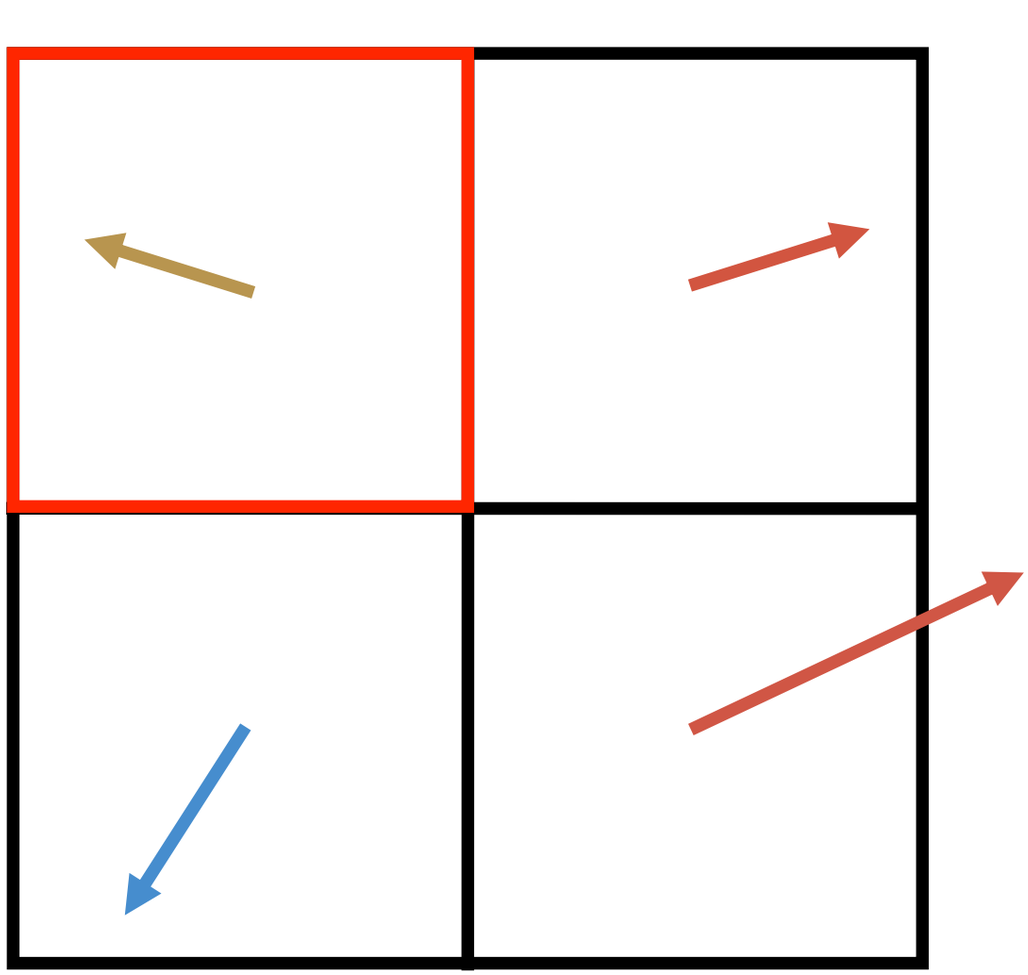


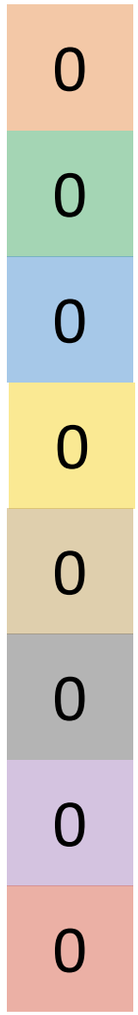


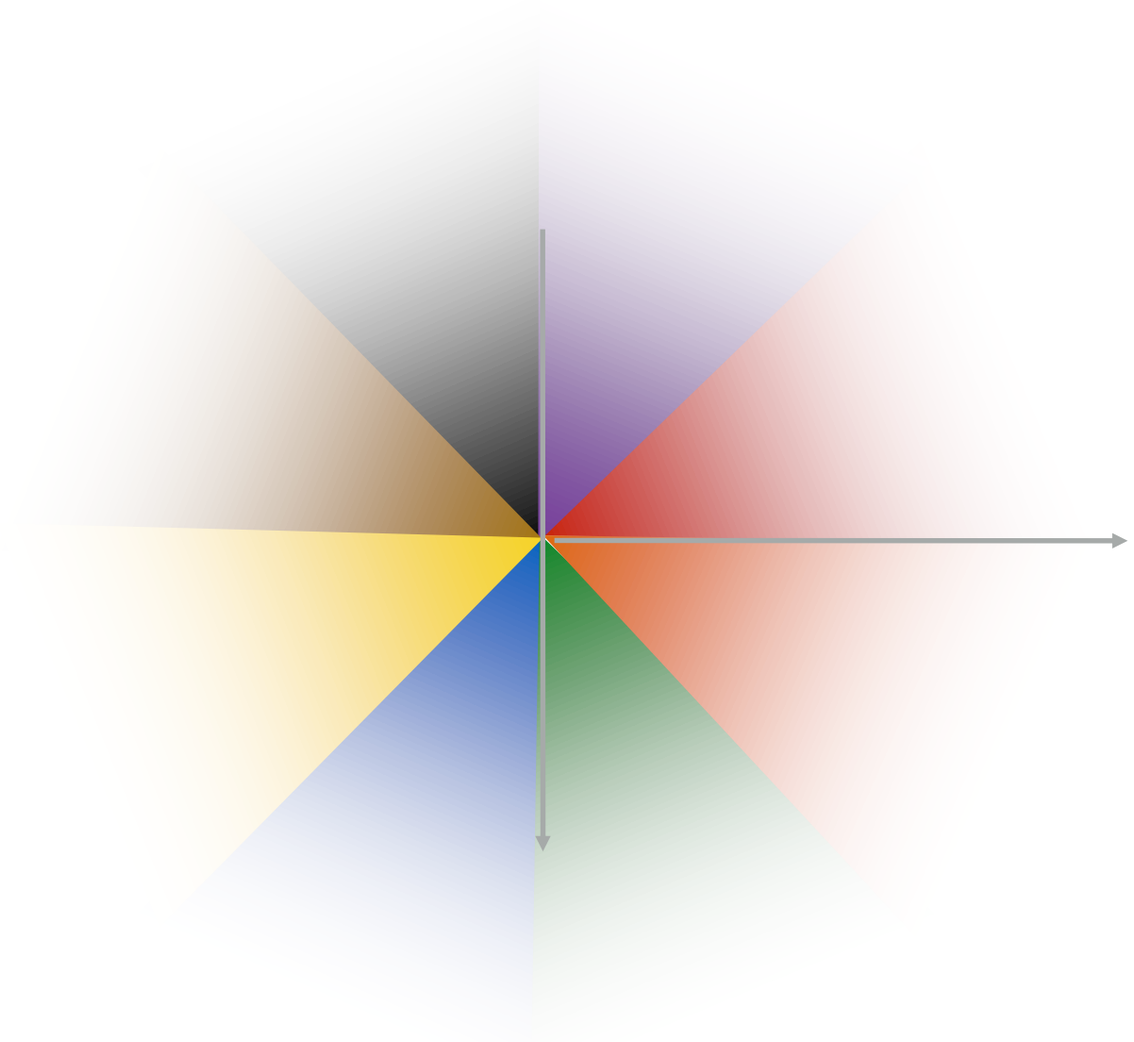


Last Lecture

Gradient histograms

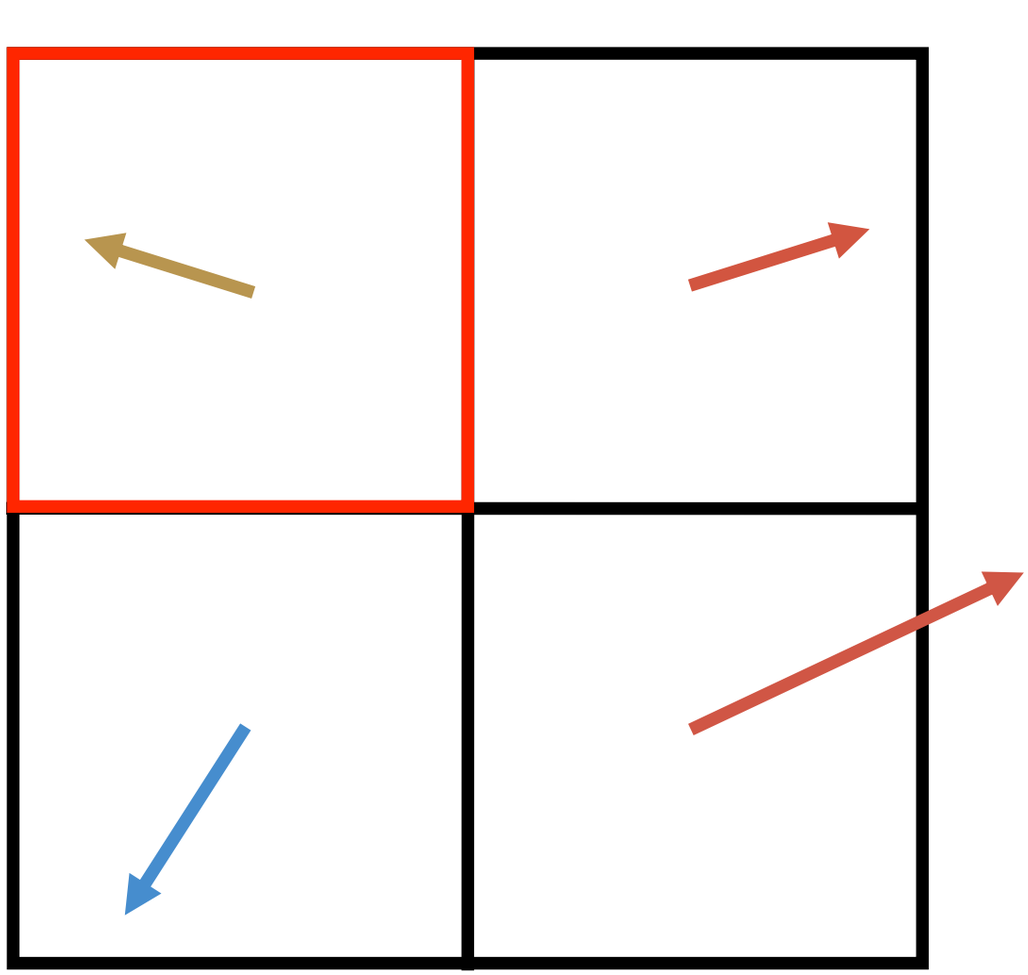


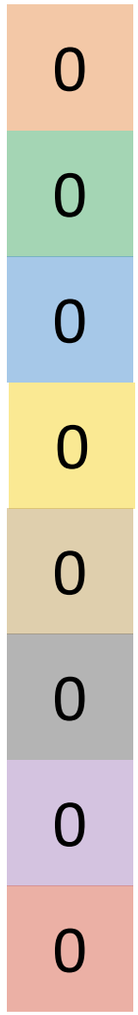


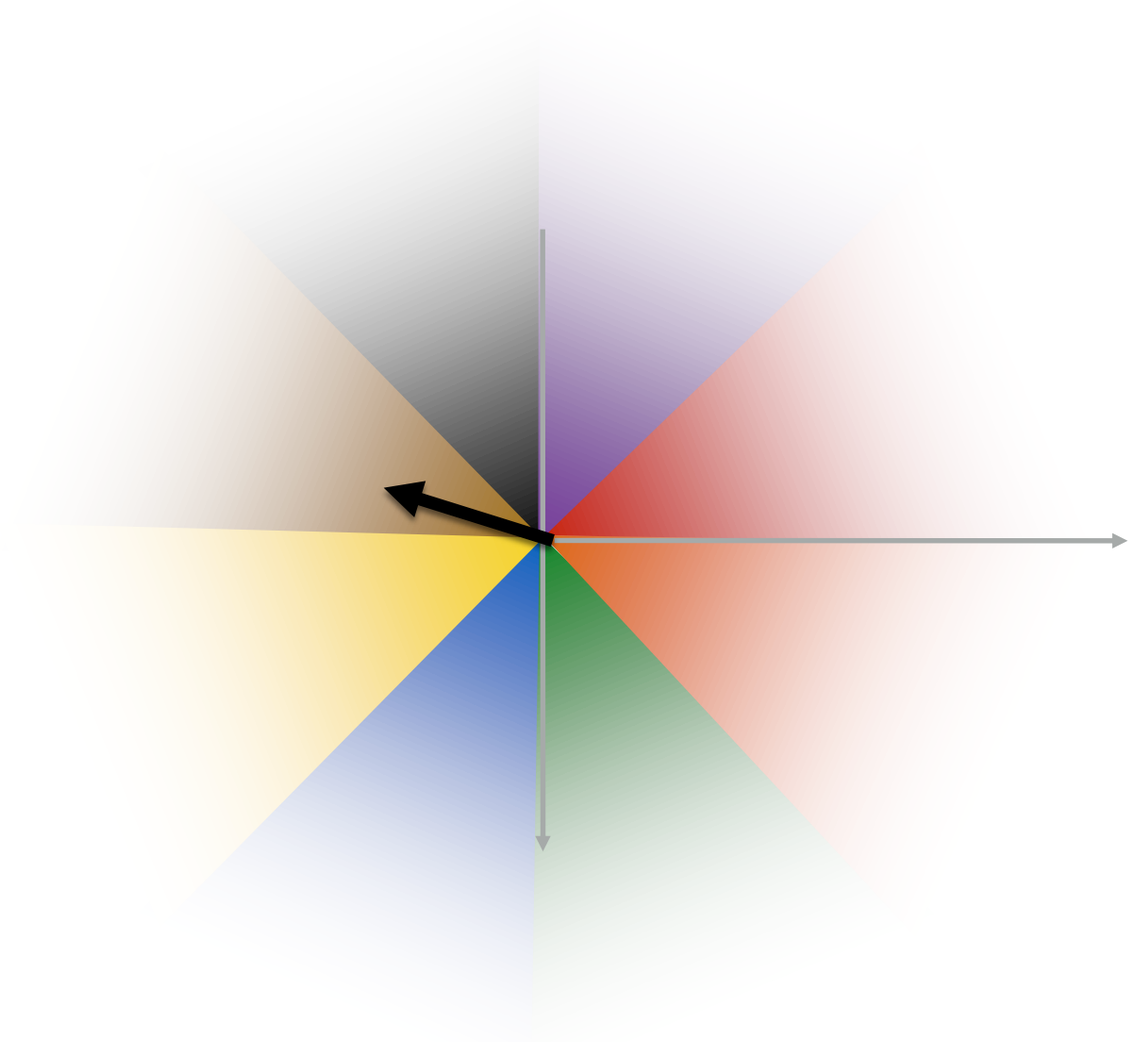


Last Lecture

Gradient histograms

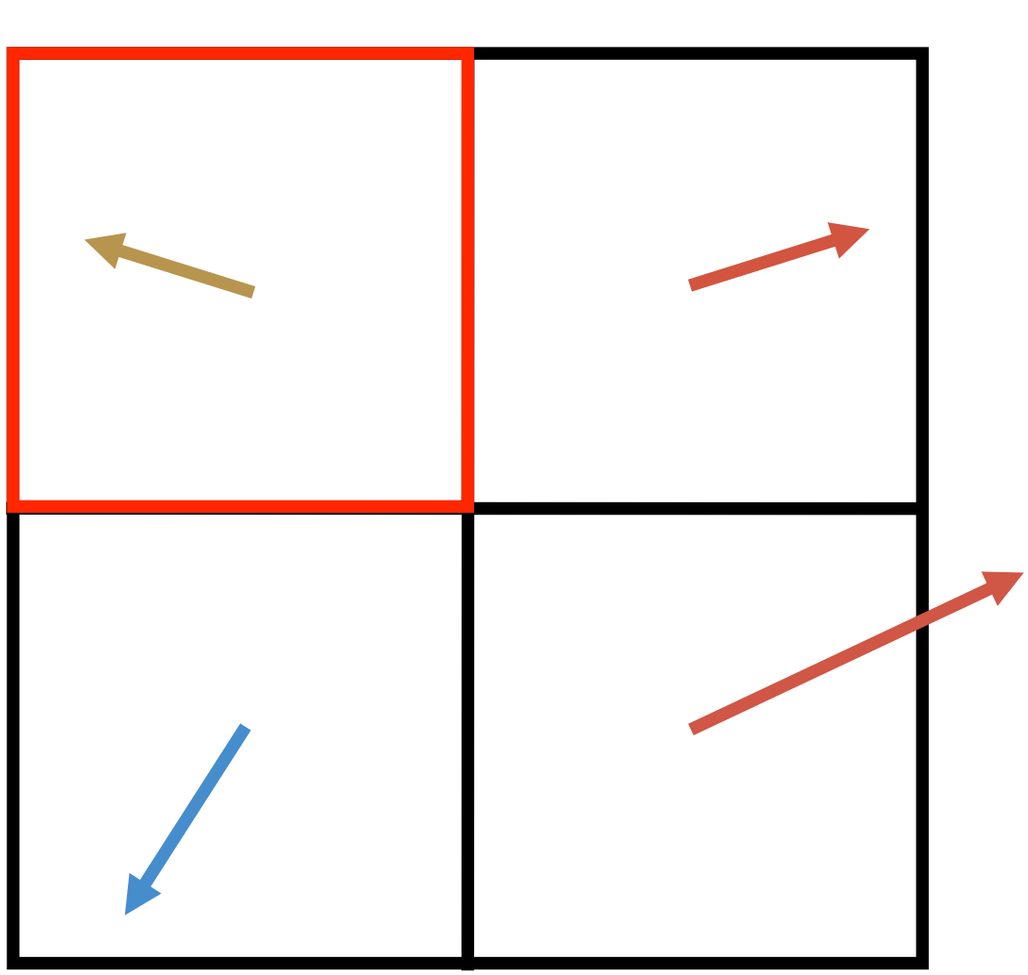


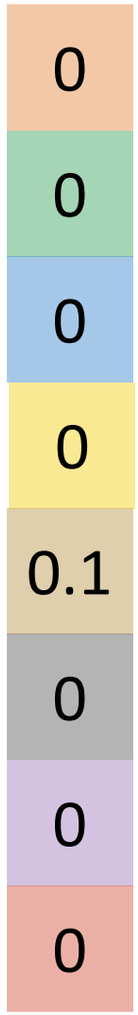


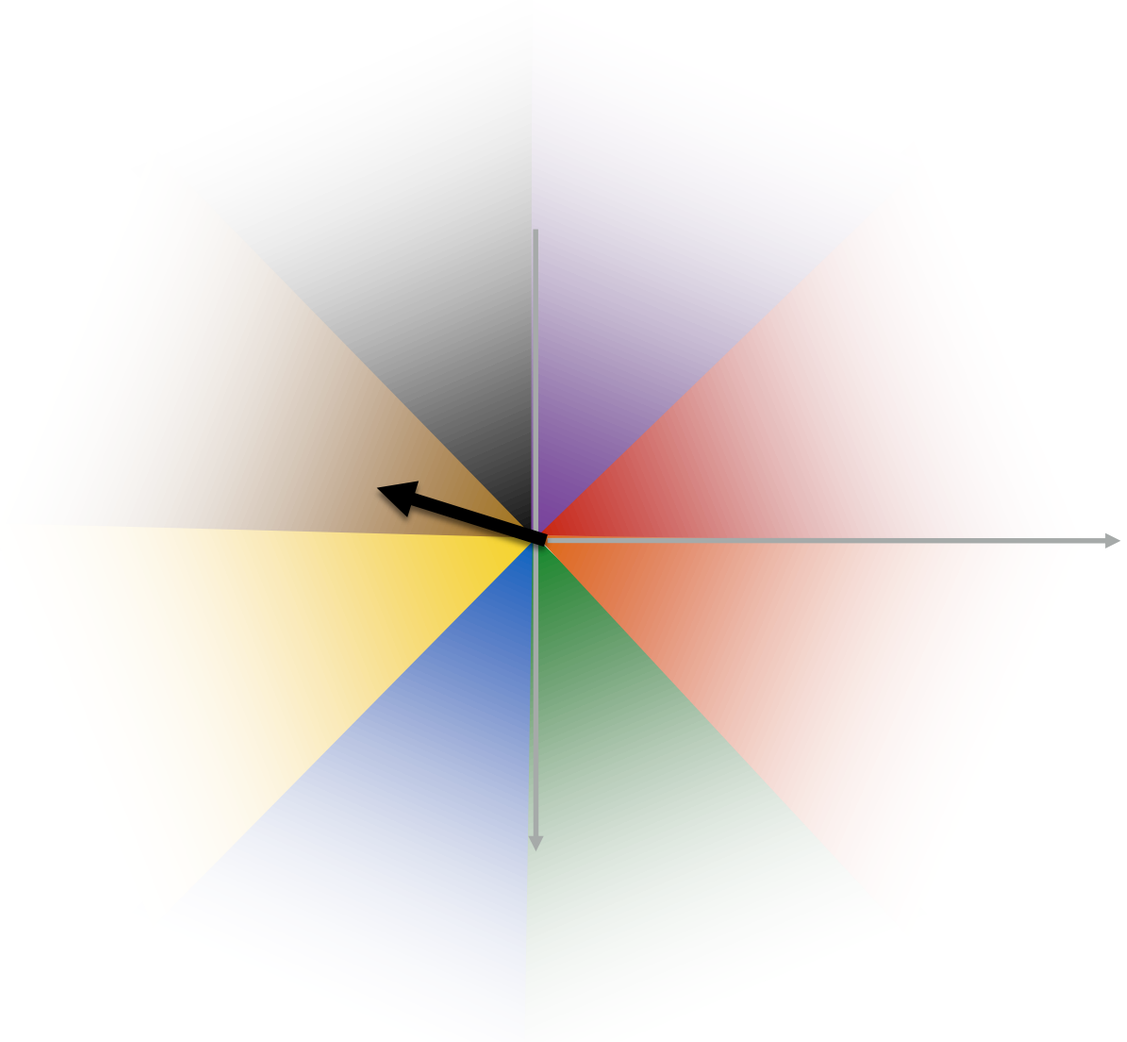


Last Lecture

Gradient histograms

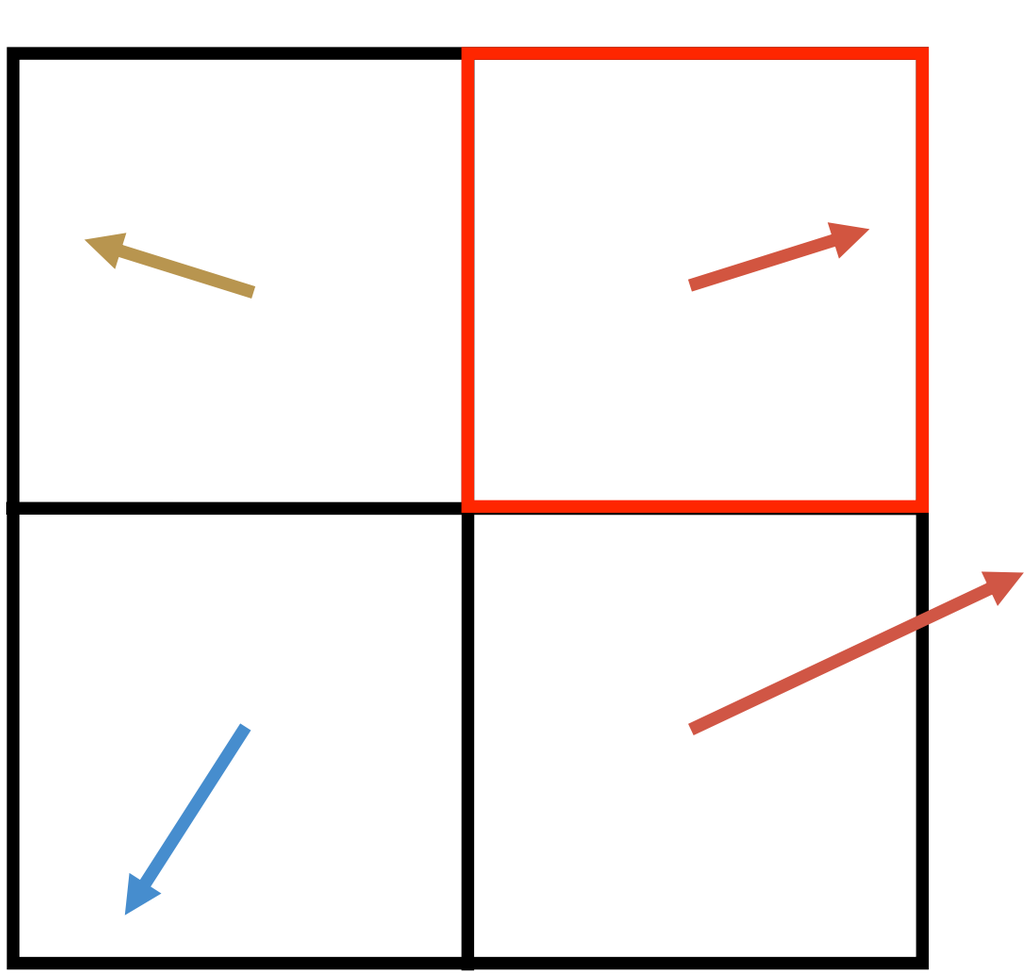


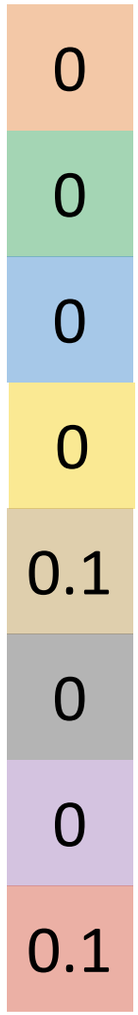


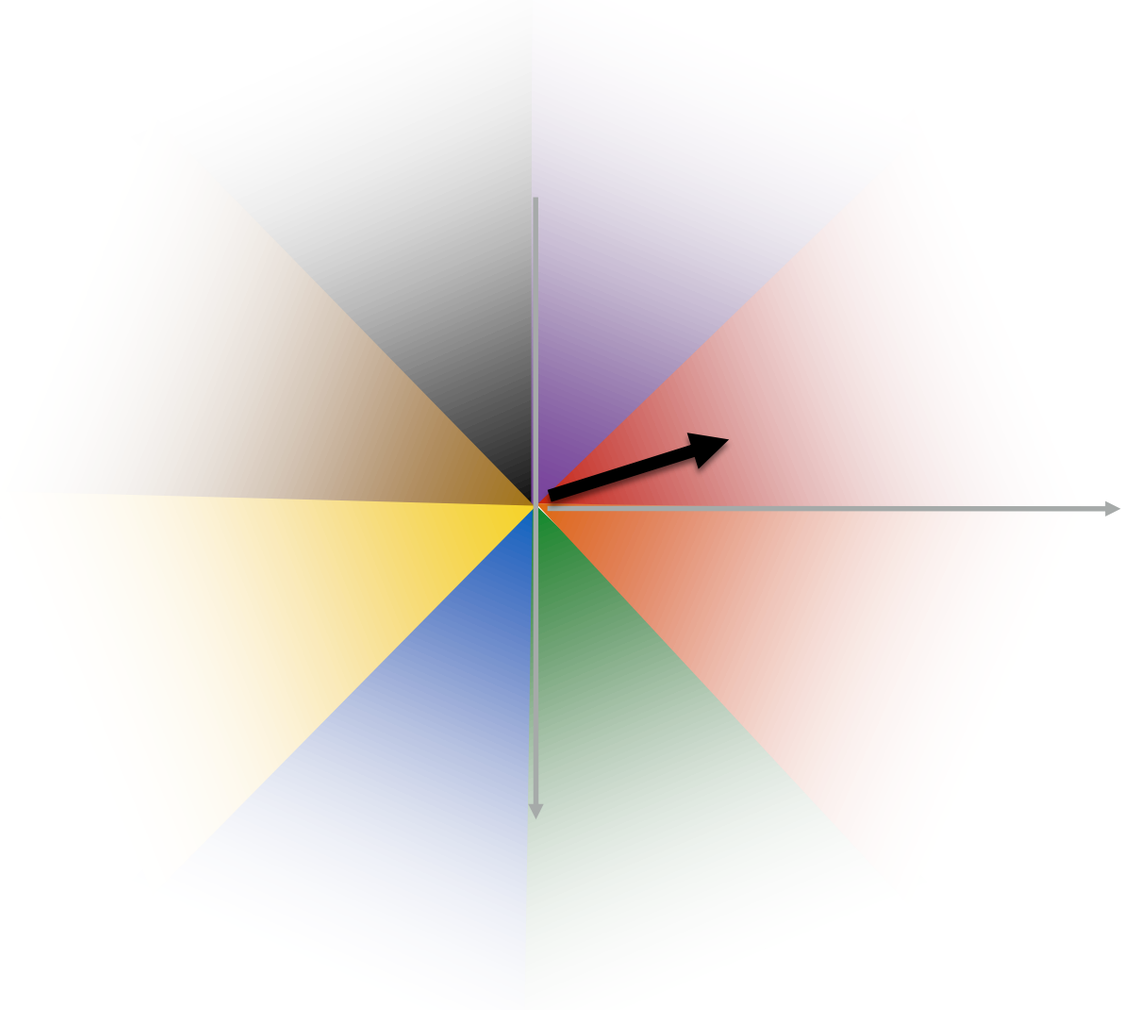


Last Lecture

Gradient histograms

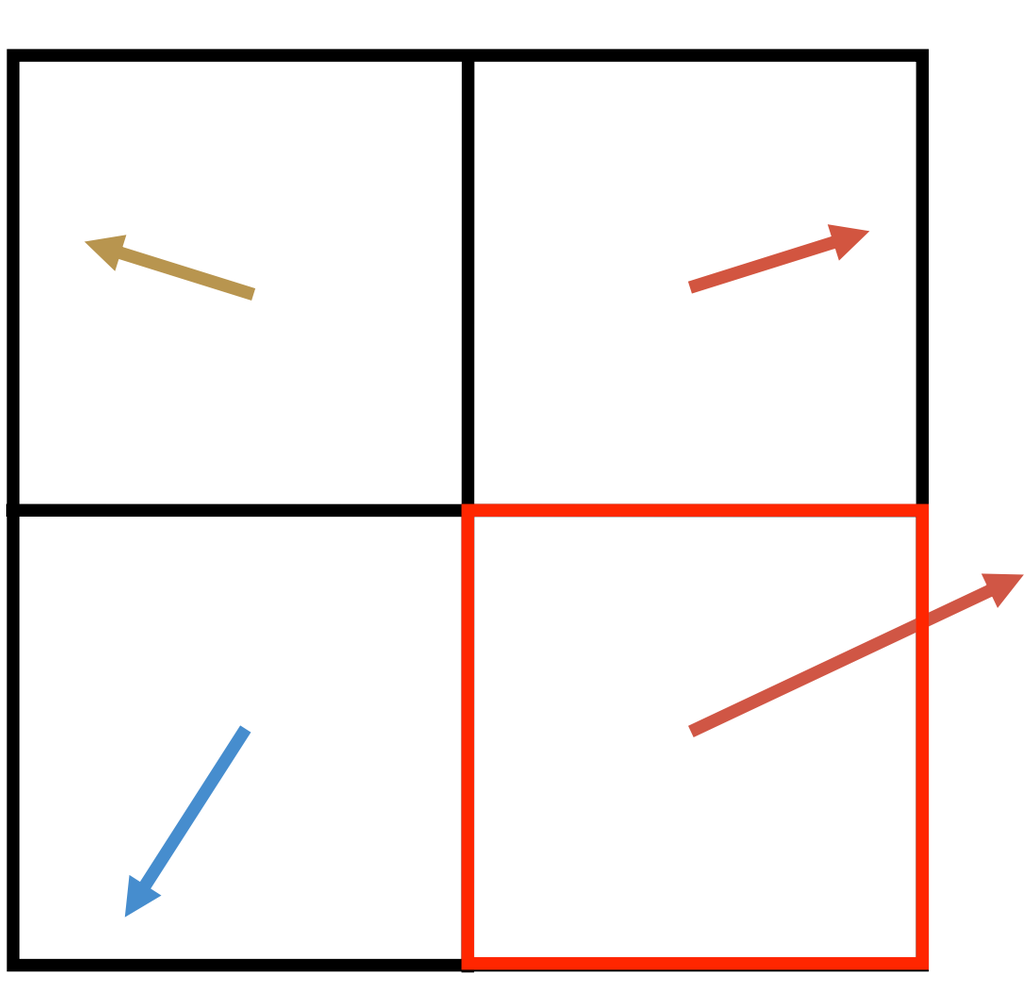


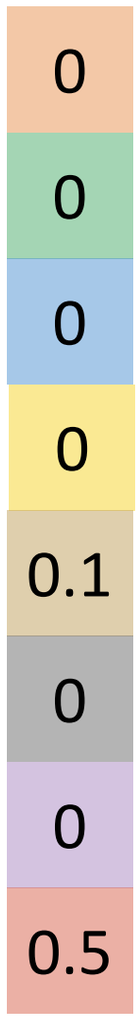


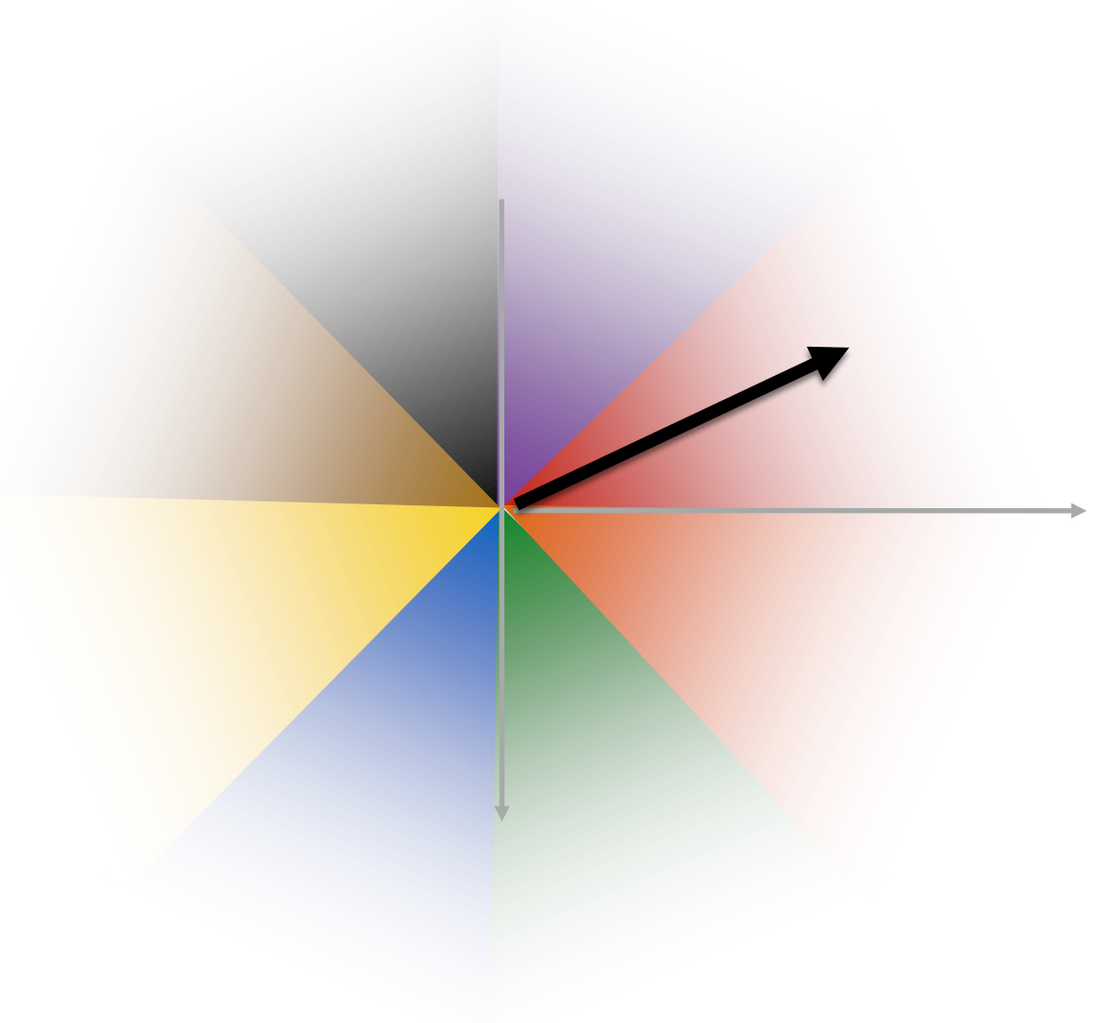


Last Lecture

Gradient histograms

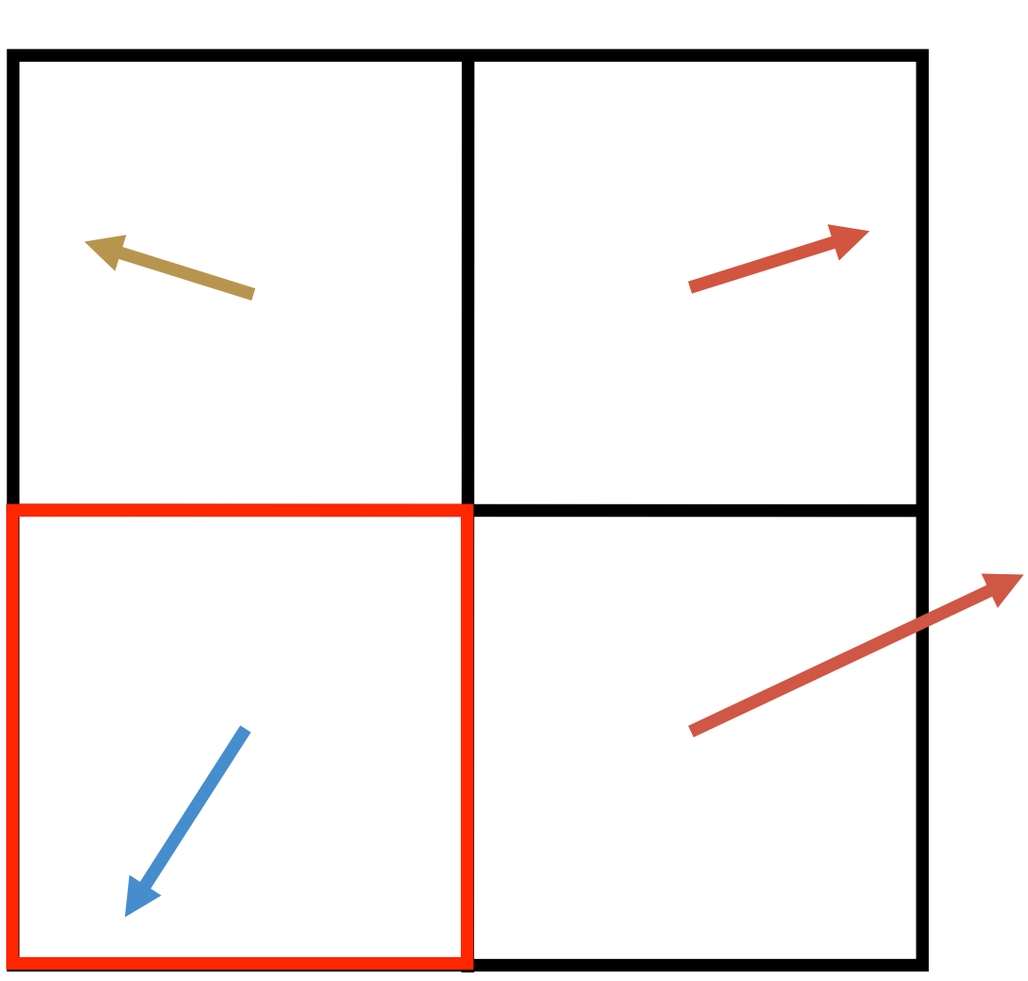


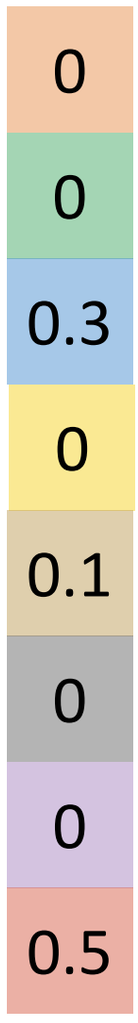


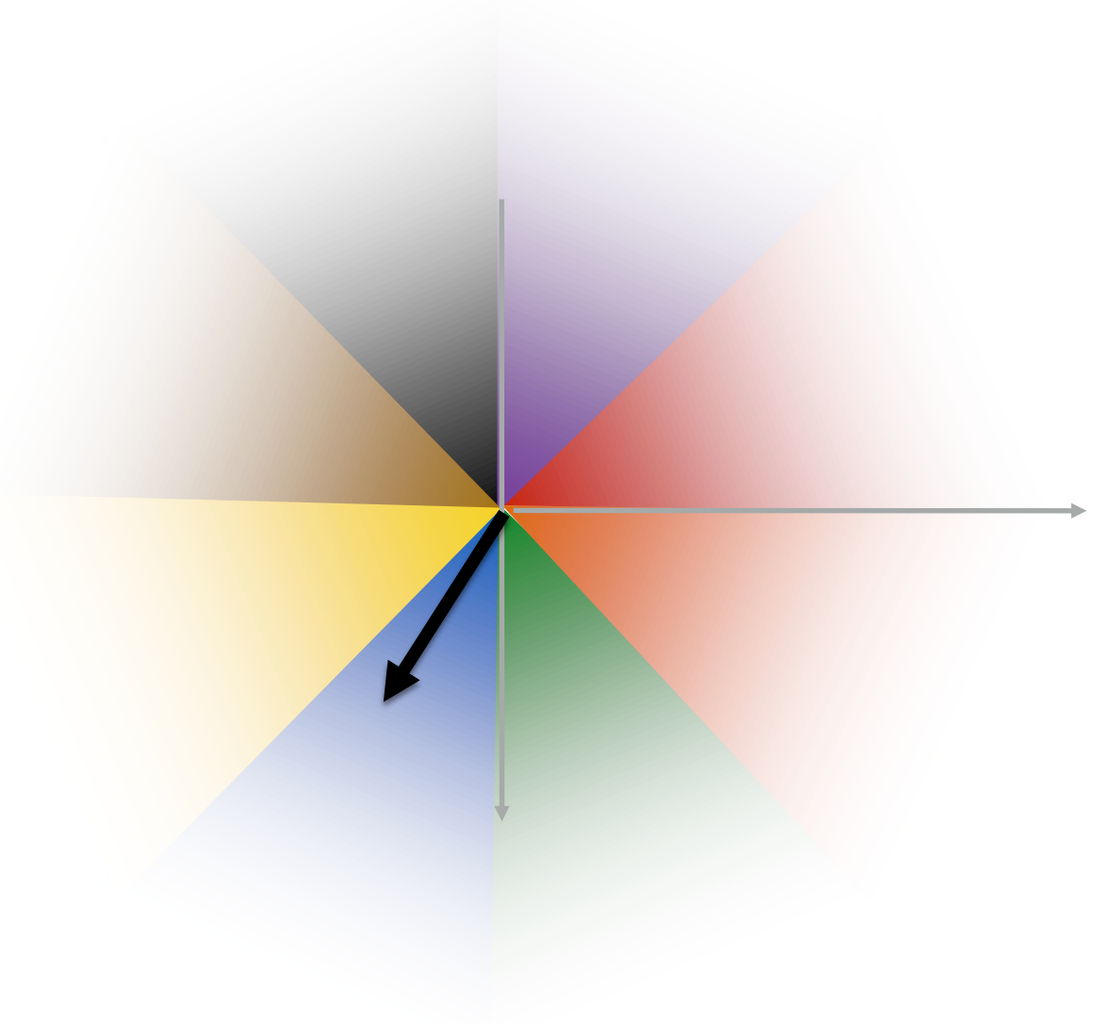


Last Lecture

Gradient histograms

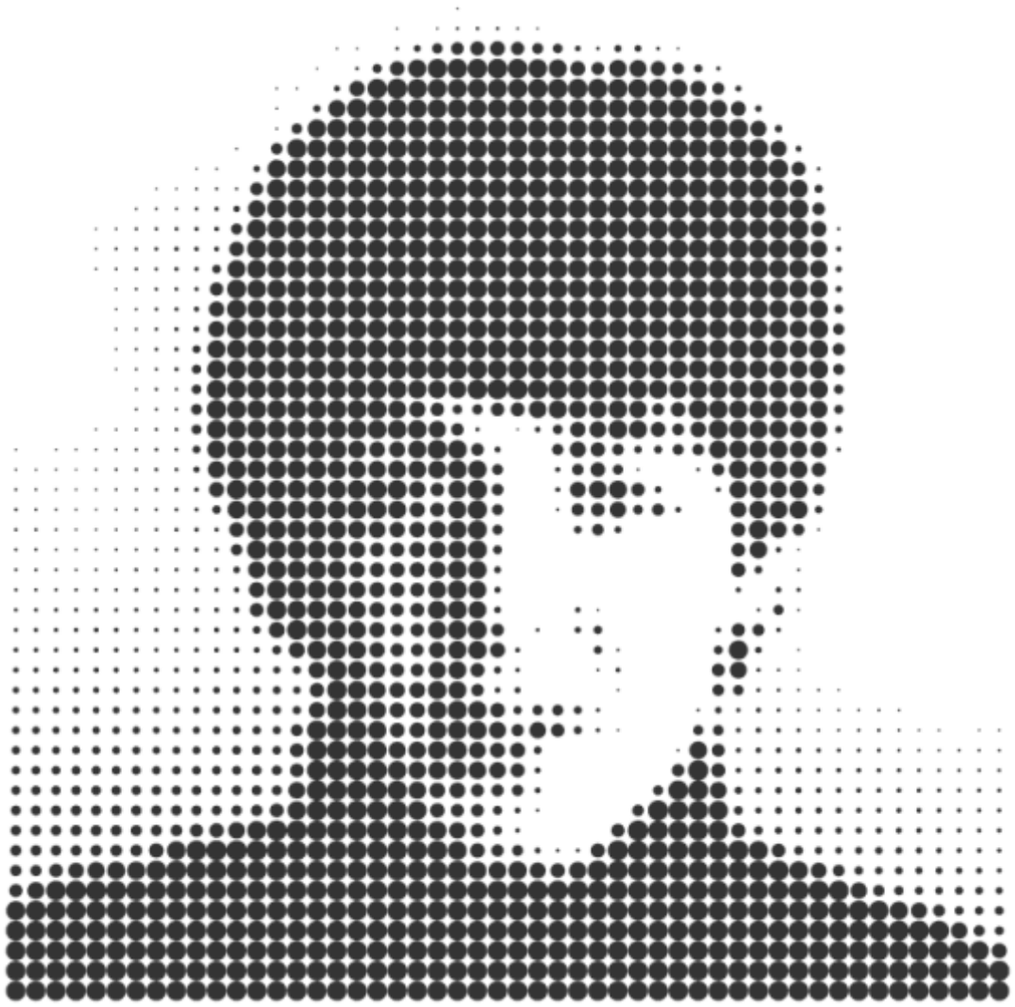






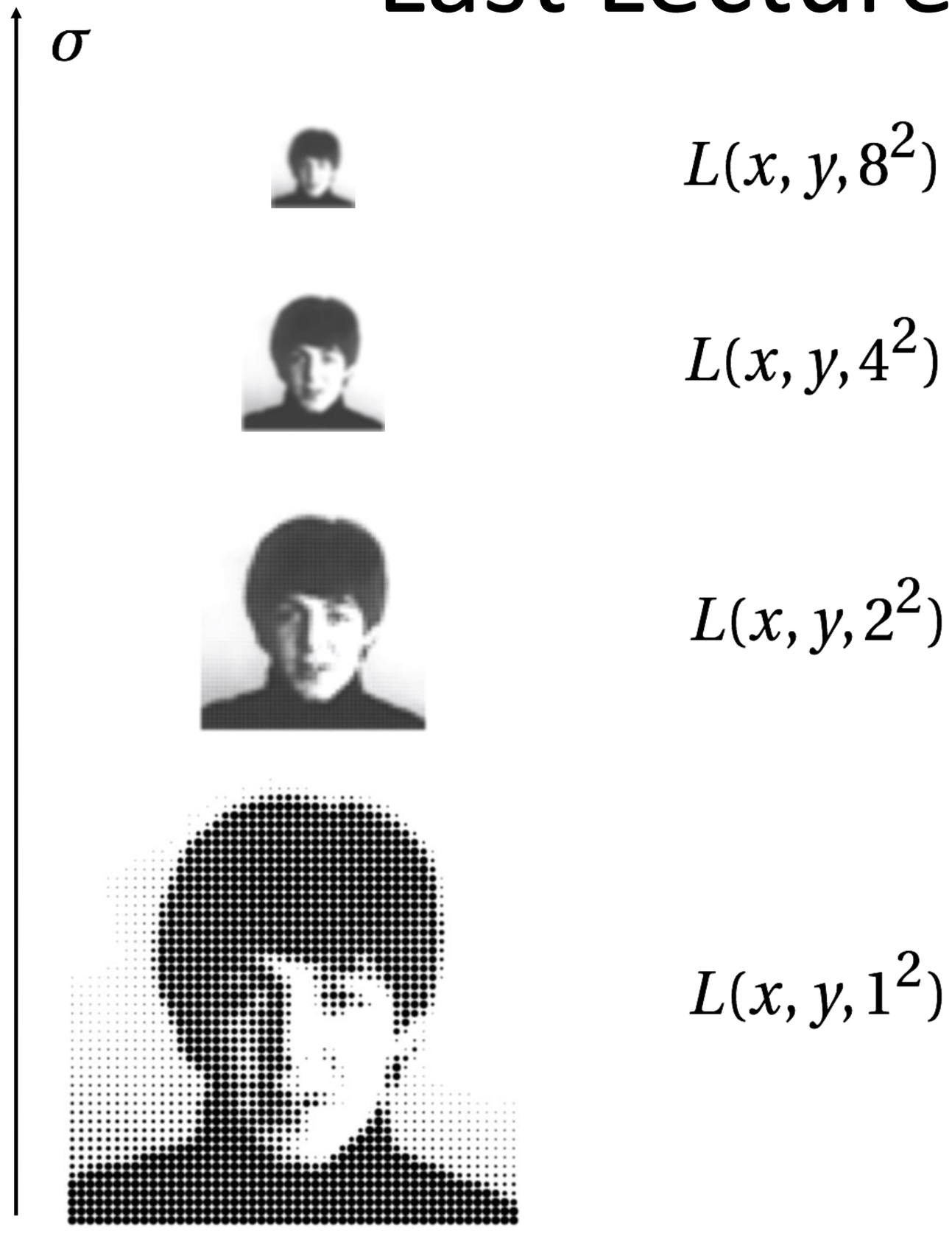
Last Lecture

Object detection at different scales



Last Lecture

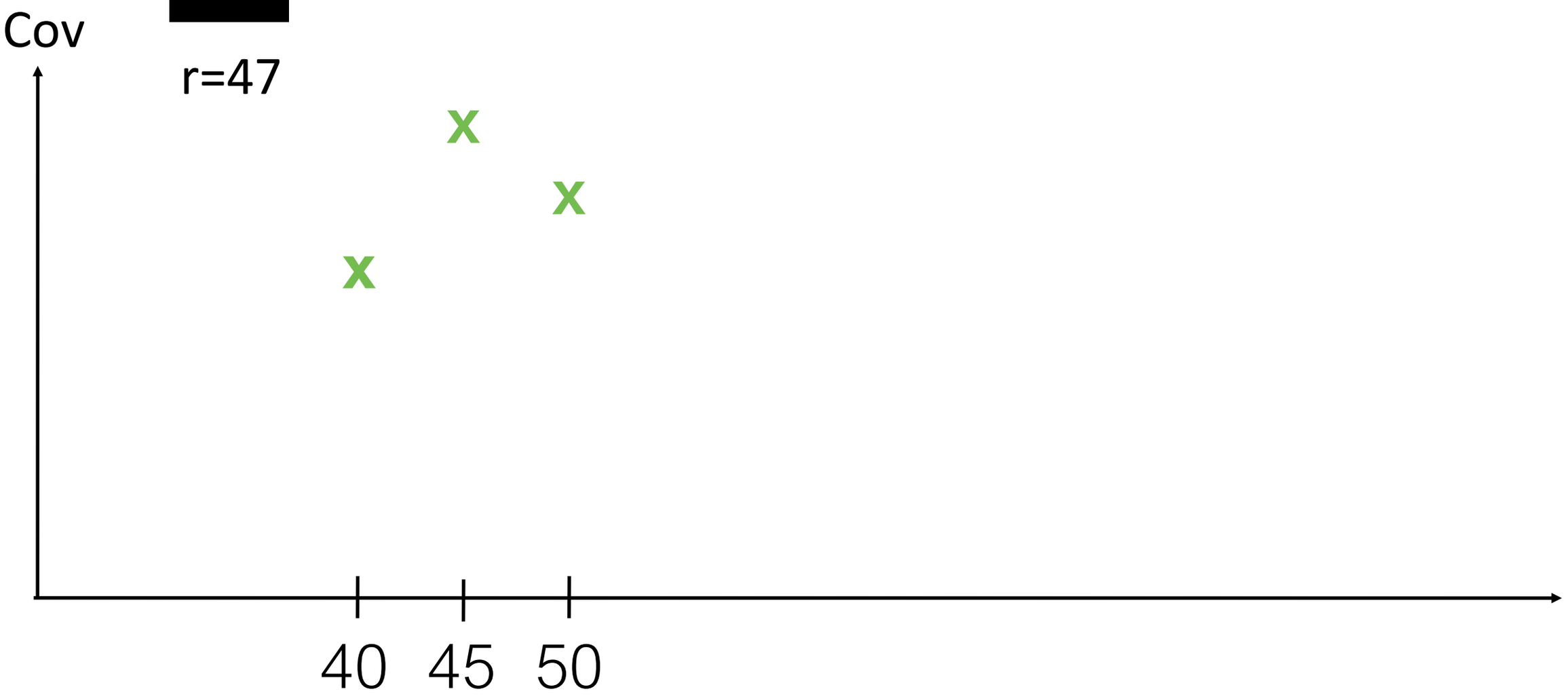
Scale space



Last Lecture

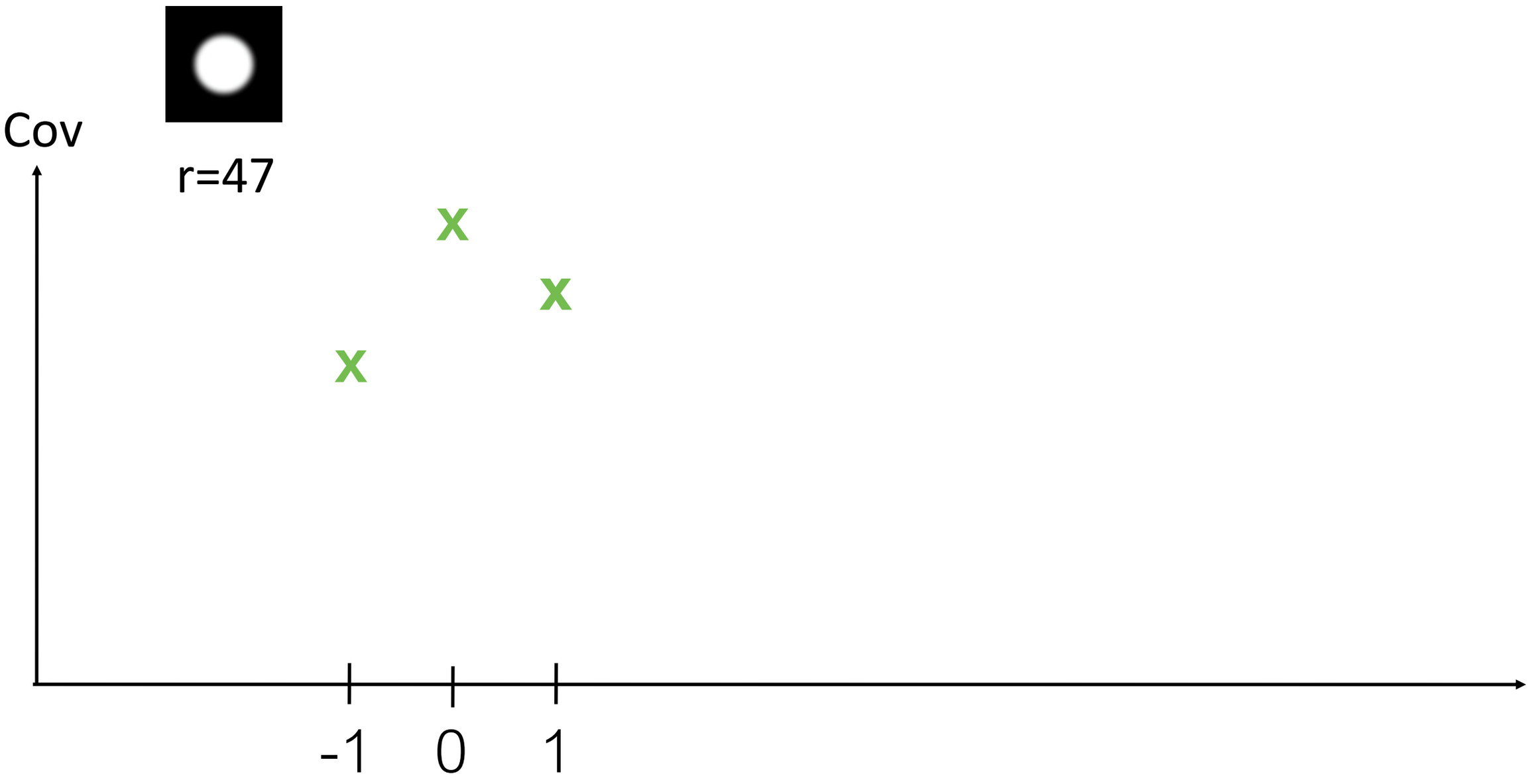
Scale refinement

r=47



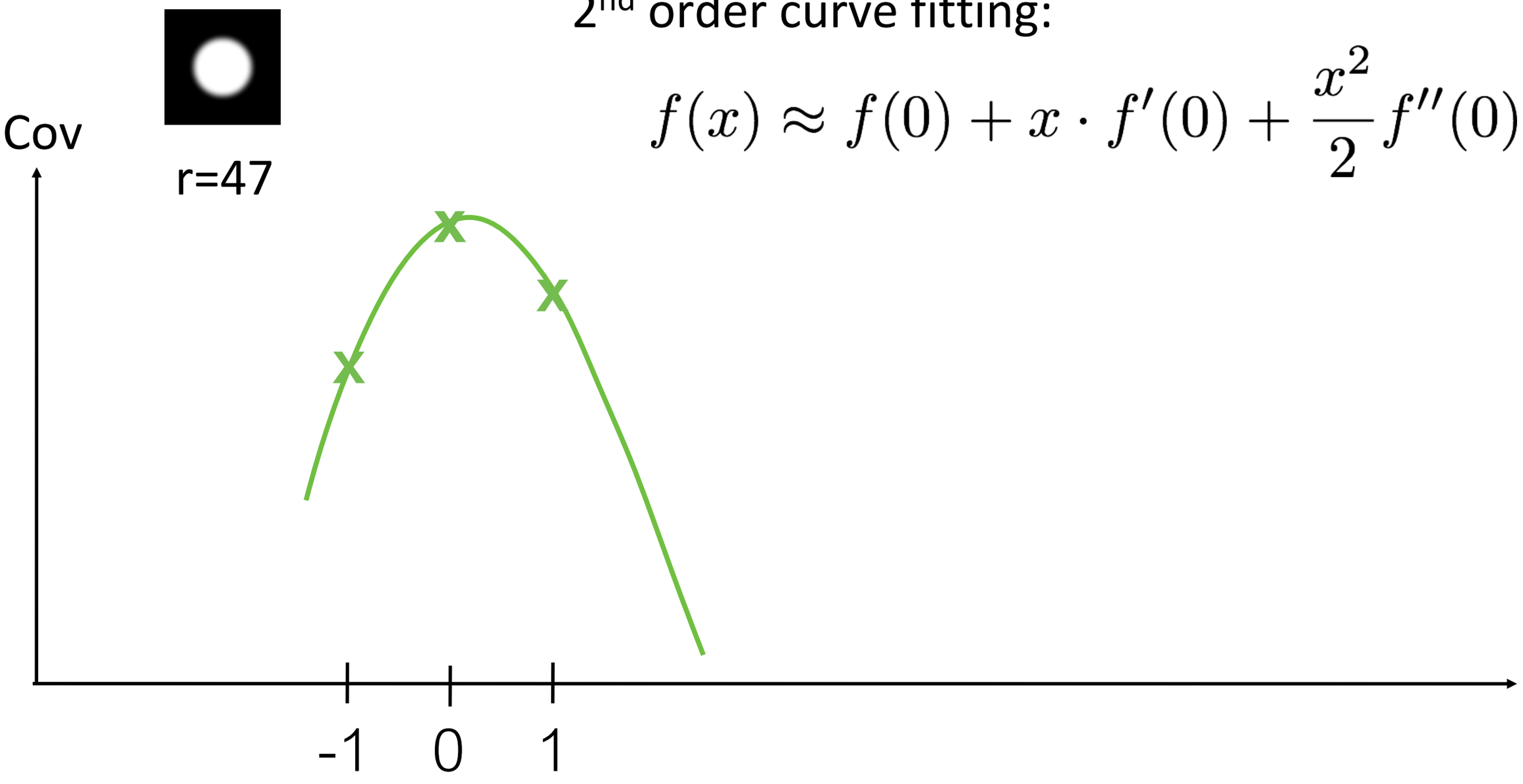
Last Lecture

Scale refinement



Last Lecture

Scale refinement

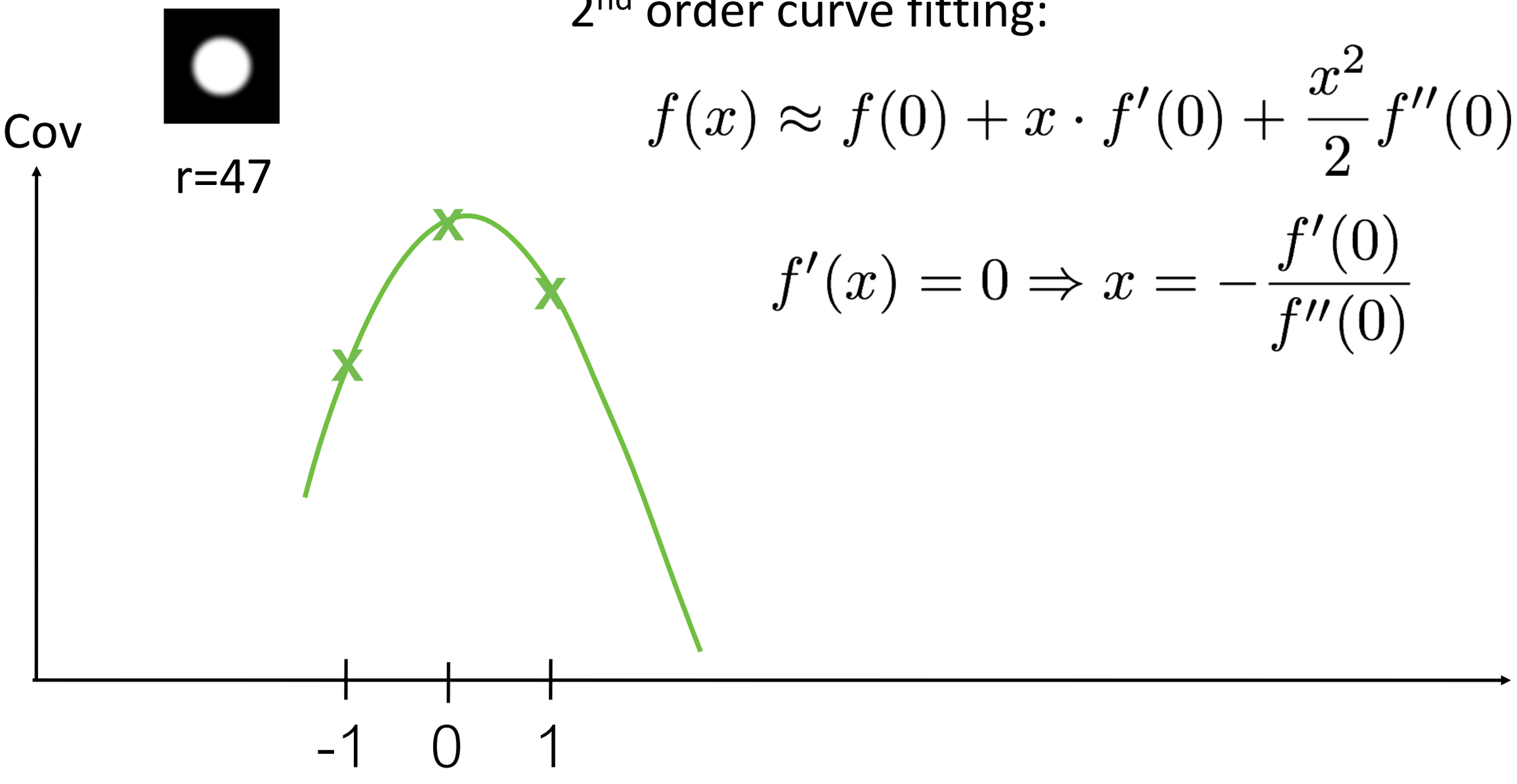


2nd order curve fitting:

$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

Last Lecture

Scale refinement



2nd order curve fitting:

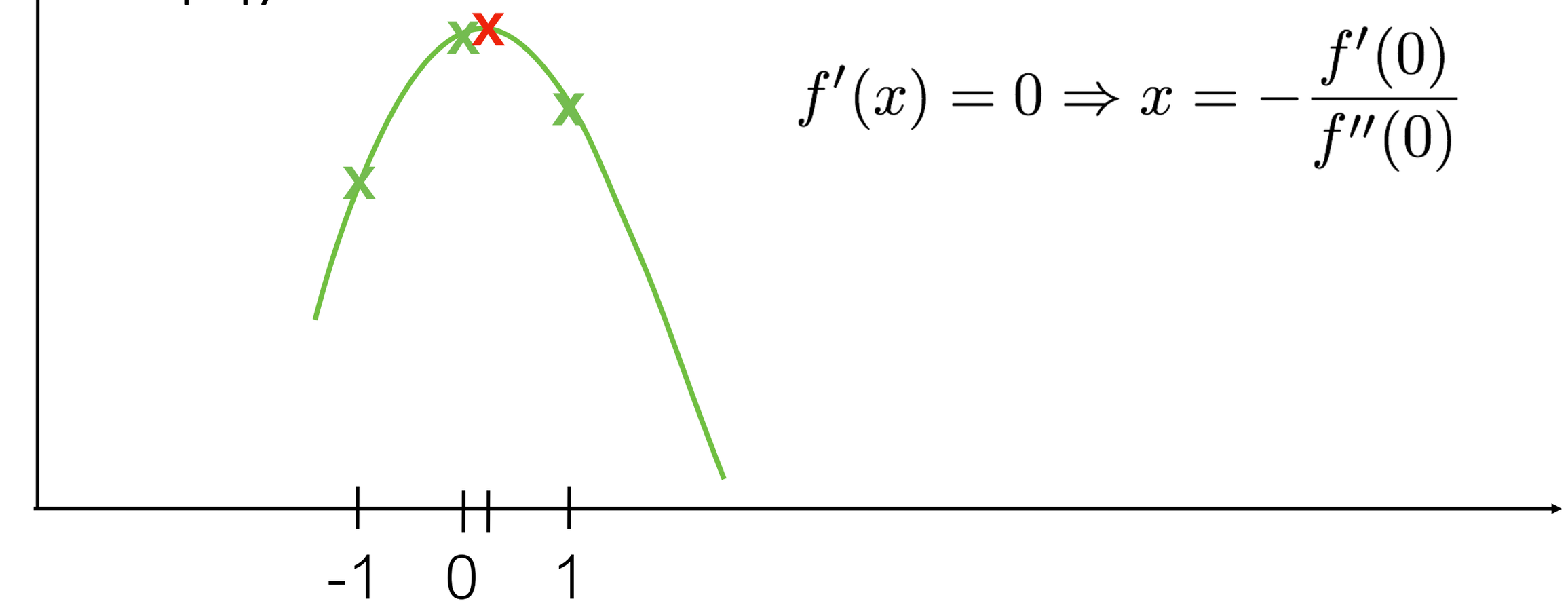
$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

$$f'(x) = 0 \Rightarrow x = -\frac{f'(0)}{f''(0)}$$

Last Lecture

Scale refinement

r=47



2nd order curve fitting:

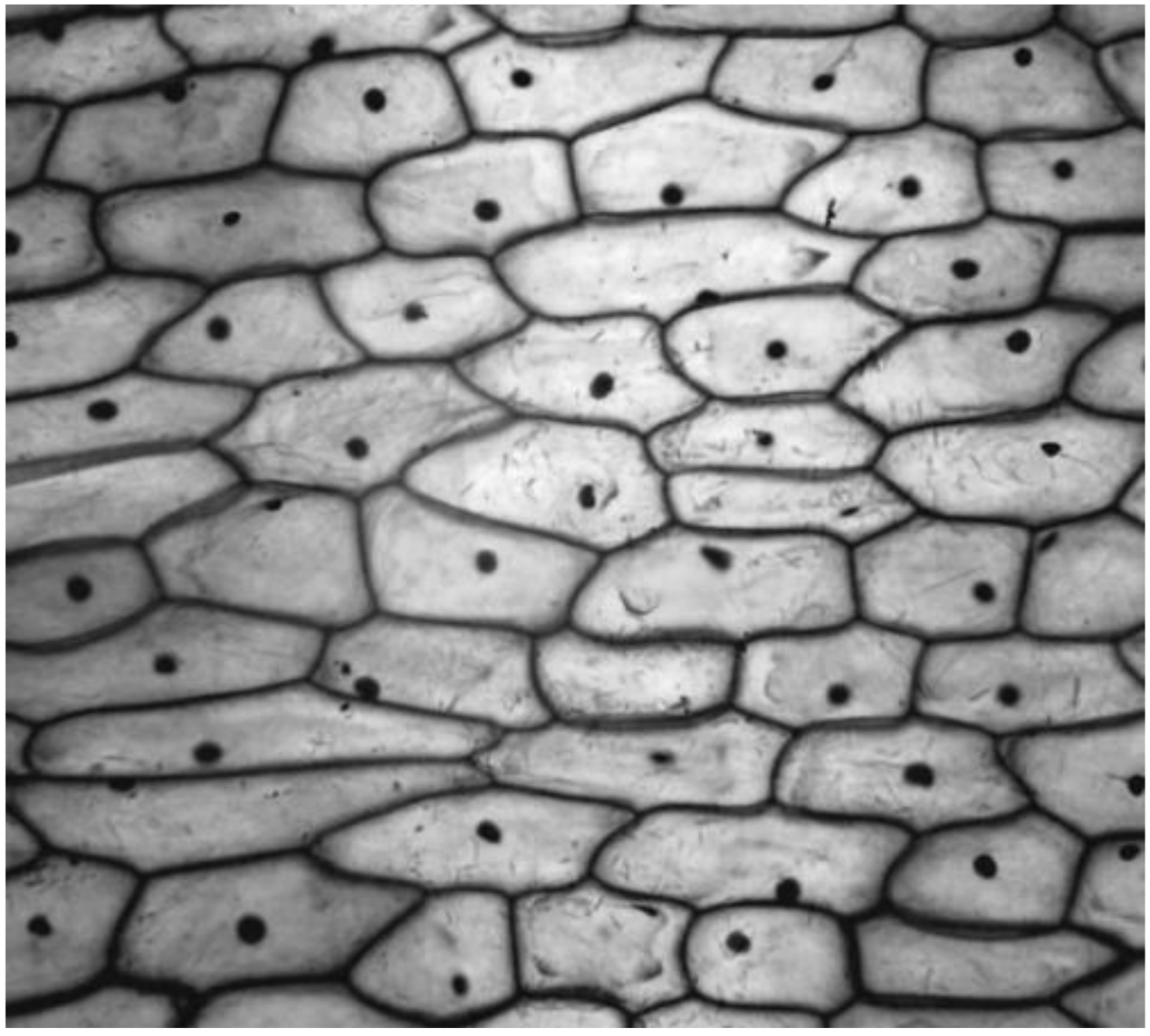
$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

$$f'(x) = 0 \Rightarrow x = -\frac{f'(0)}{f''(0)}$$

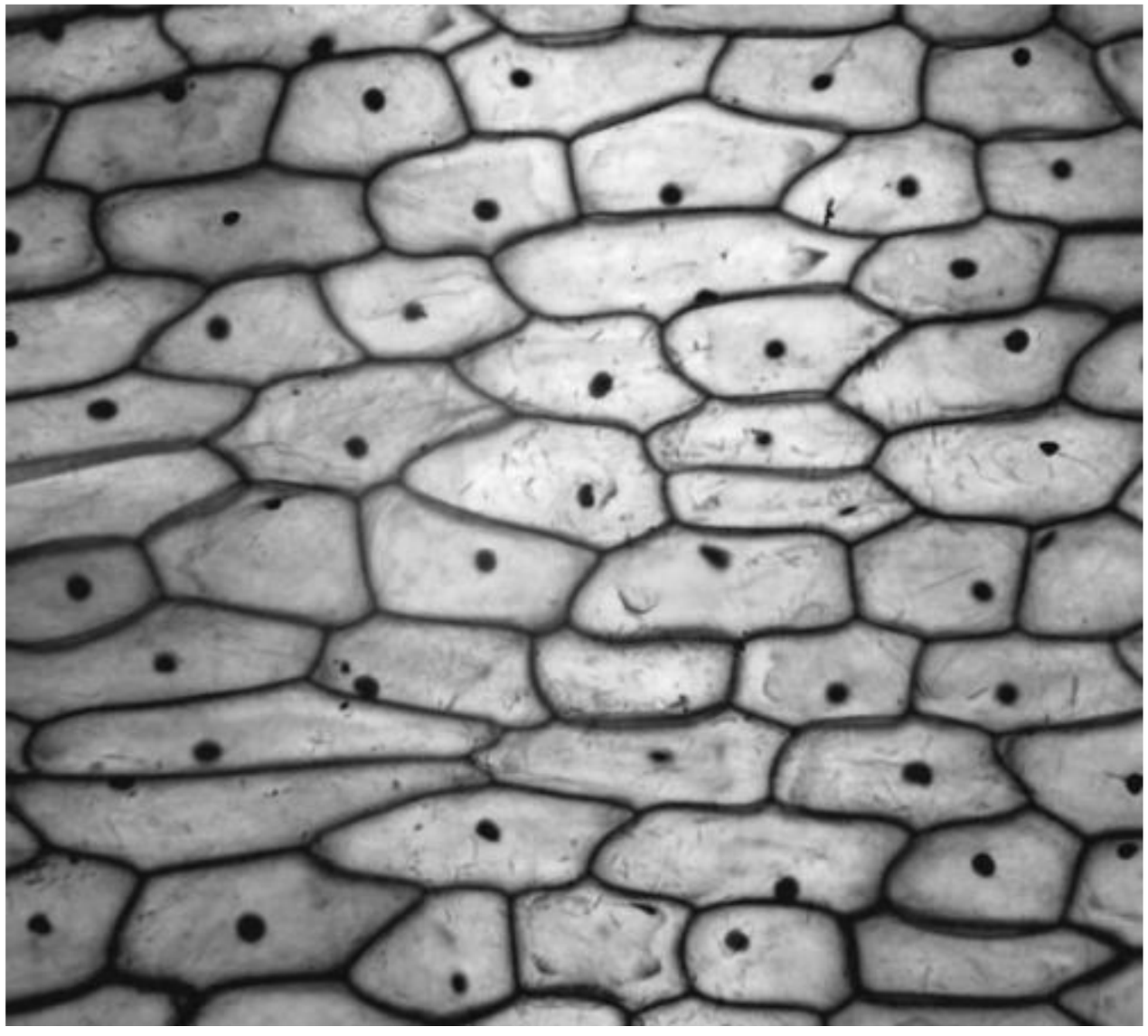
Today

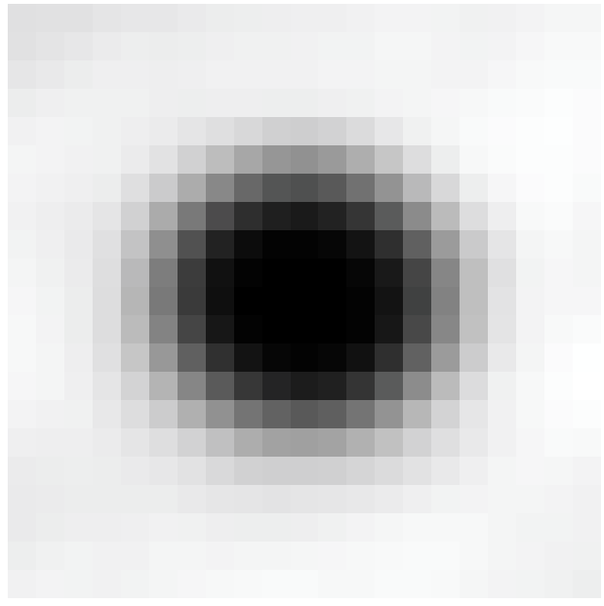
- Scale-Invariant Local Features (SIFT)
- Geometric Invariances

Object Detection via Classification

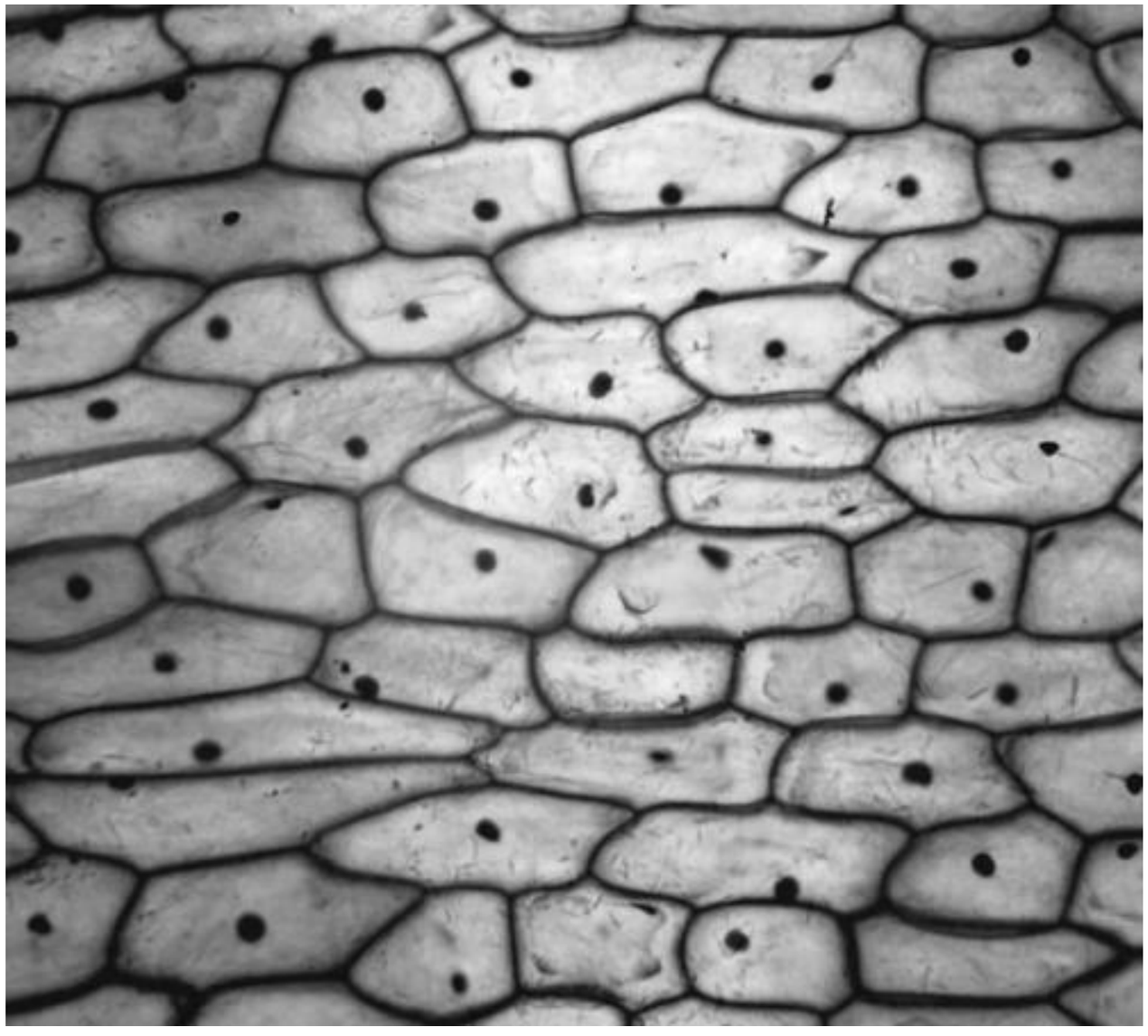


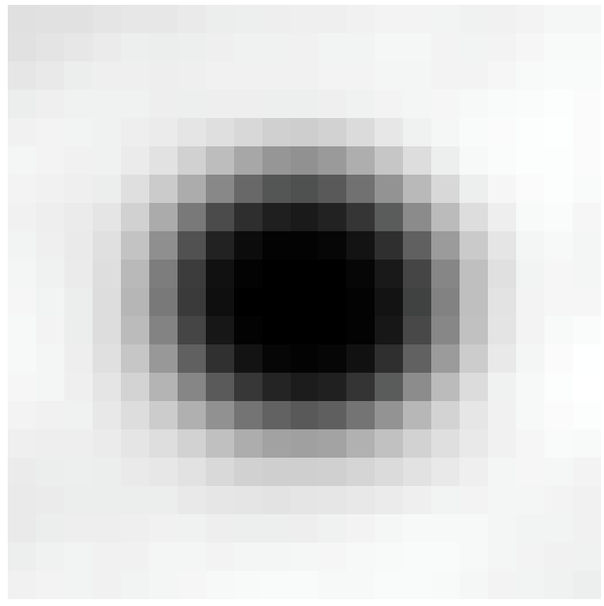
Object Detection via Classification





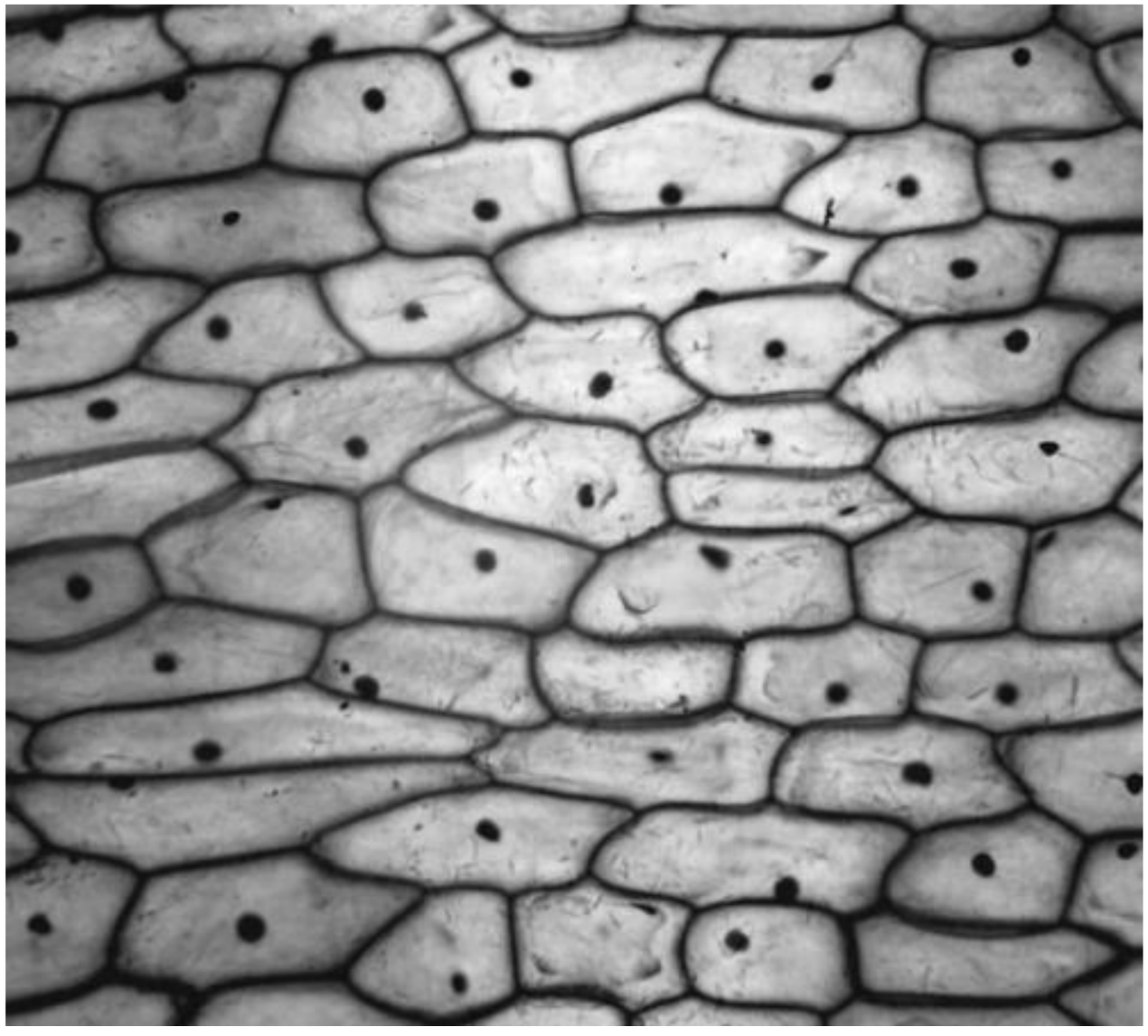
Object Detection via Classification

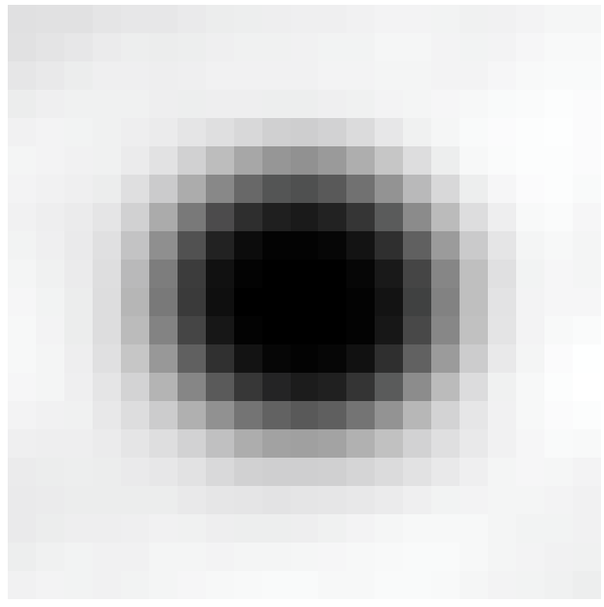




Linear filter
= example object

Object Detection via Classification





Linear filter
= example object

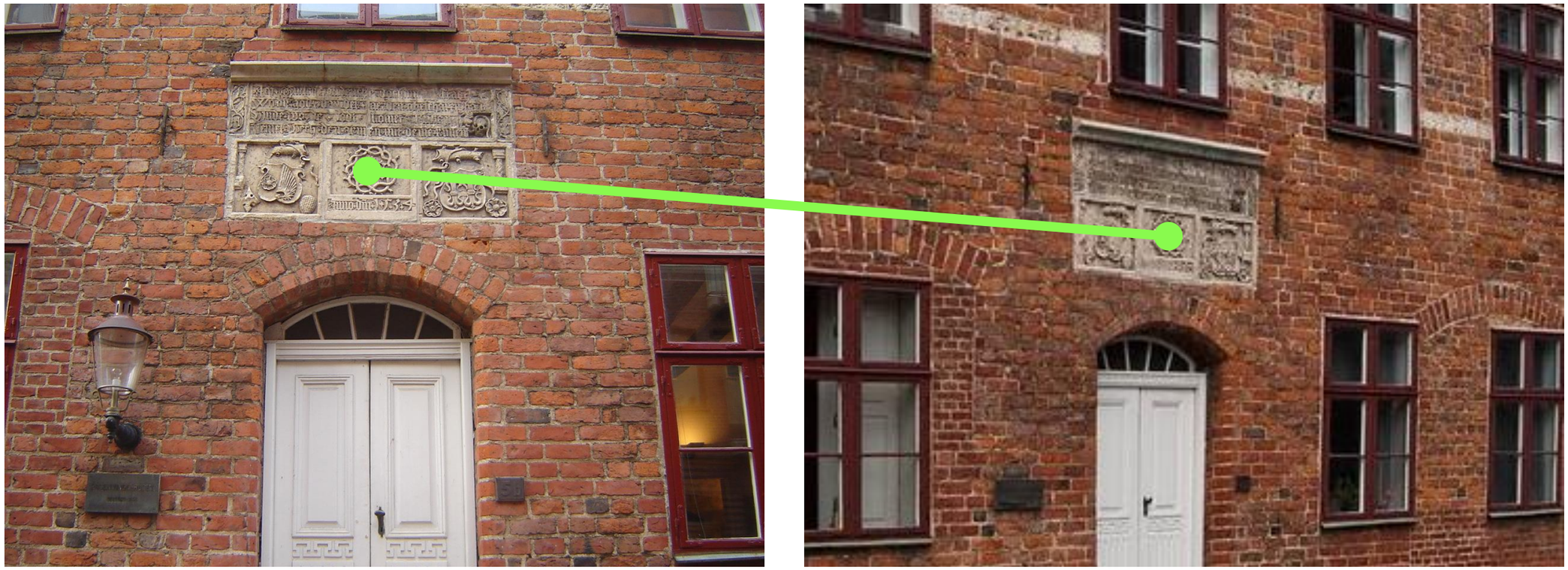
Know the object in advance

Local Features

Establish pixel-level correspondences between two images

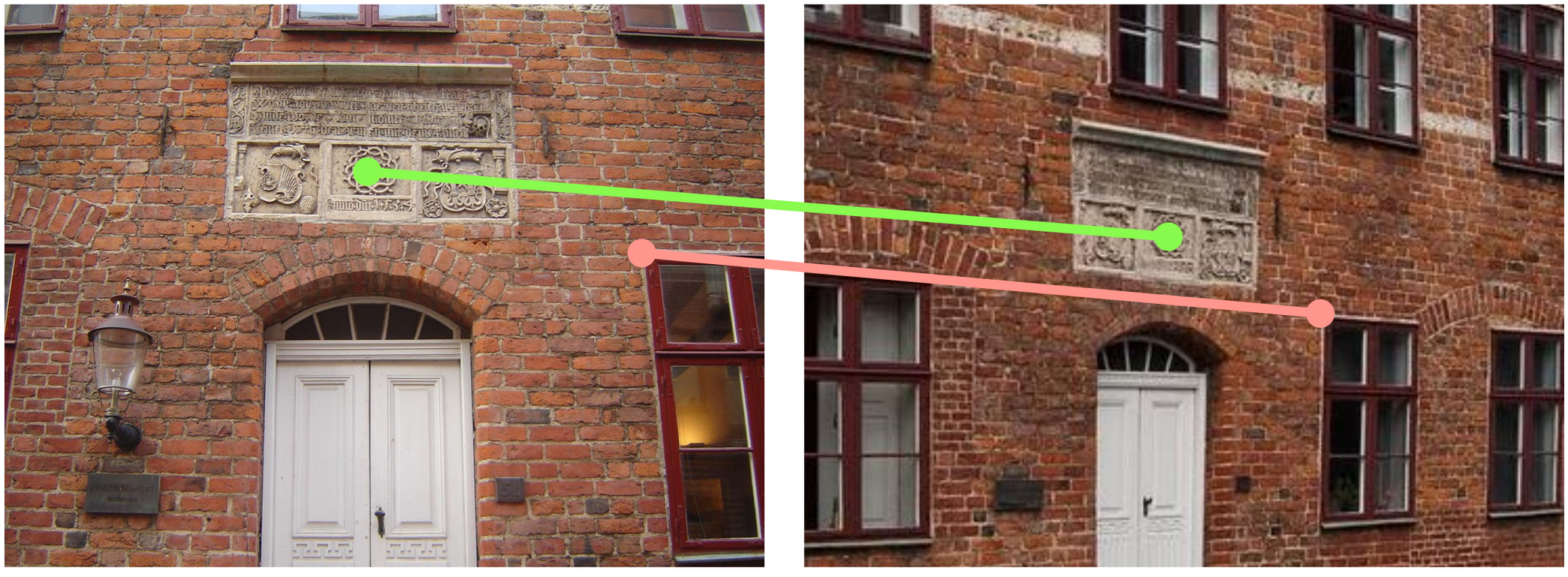
Local Features

Establish pixel-level correspondences between two images



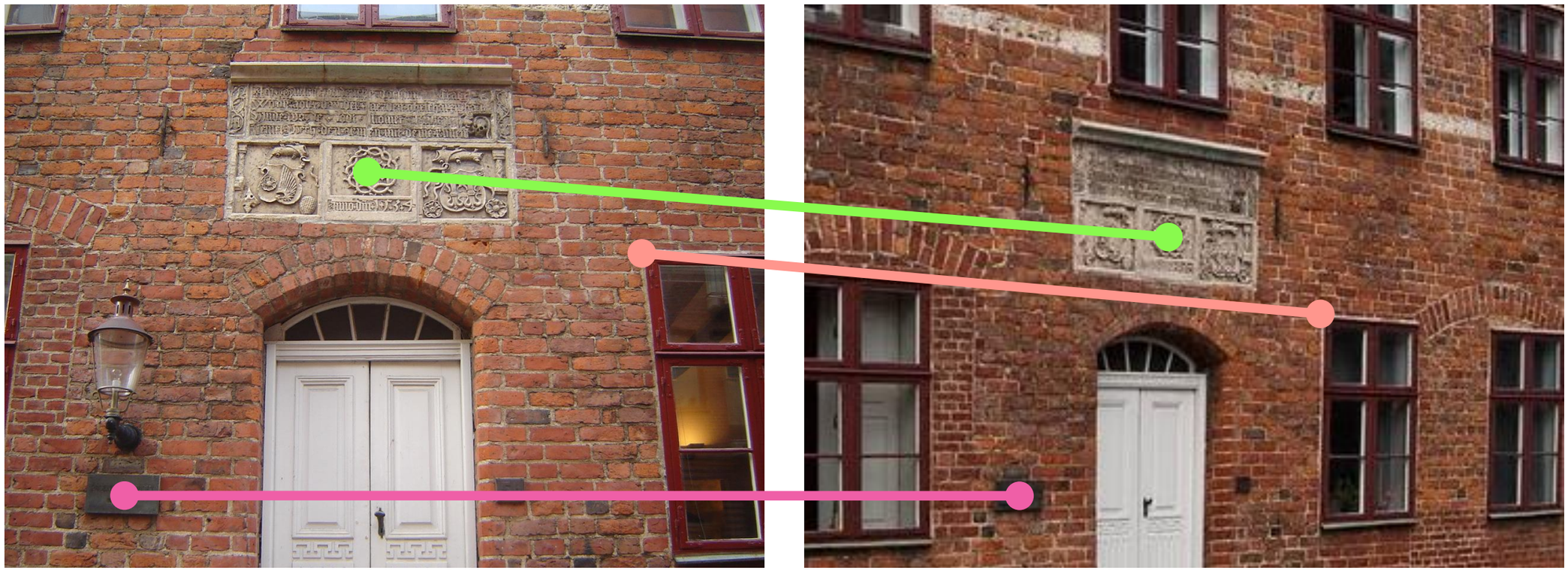
Local Features

Establish pixel-level correspondences between two images



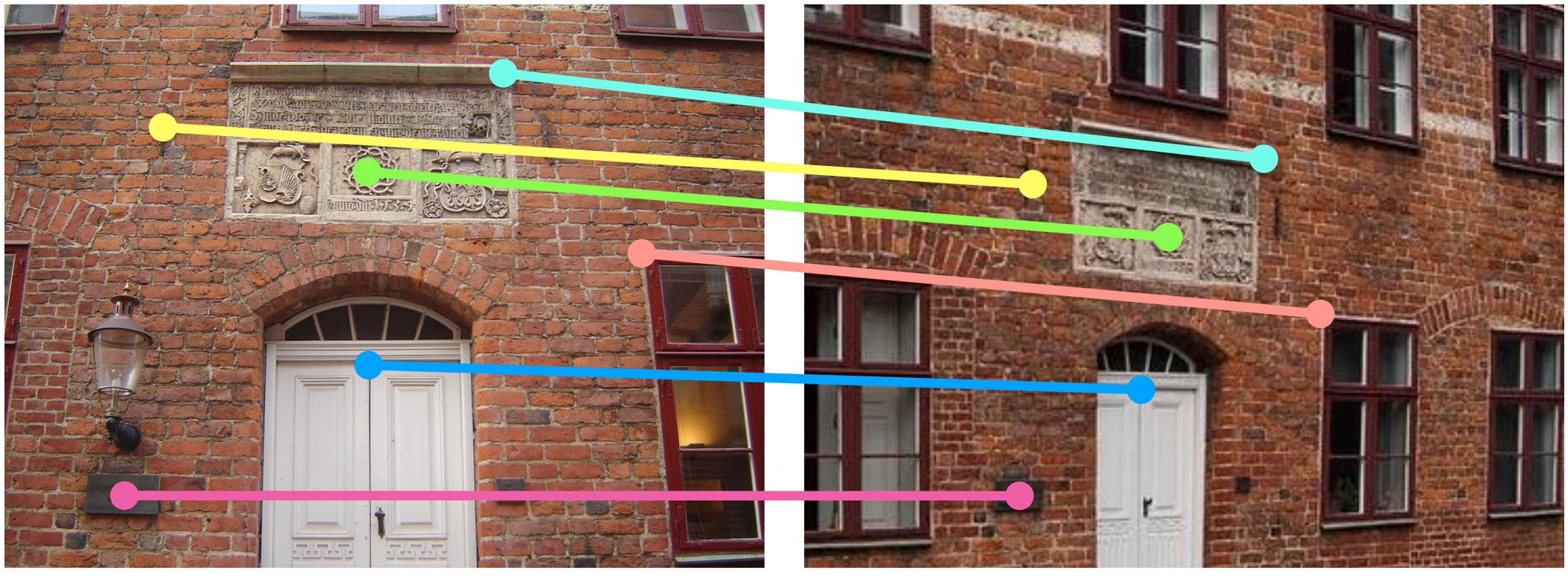
Local Features

Establish pixel-level correspondences between two images



Local Features

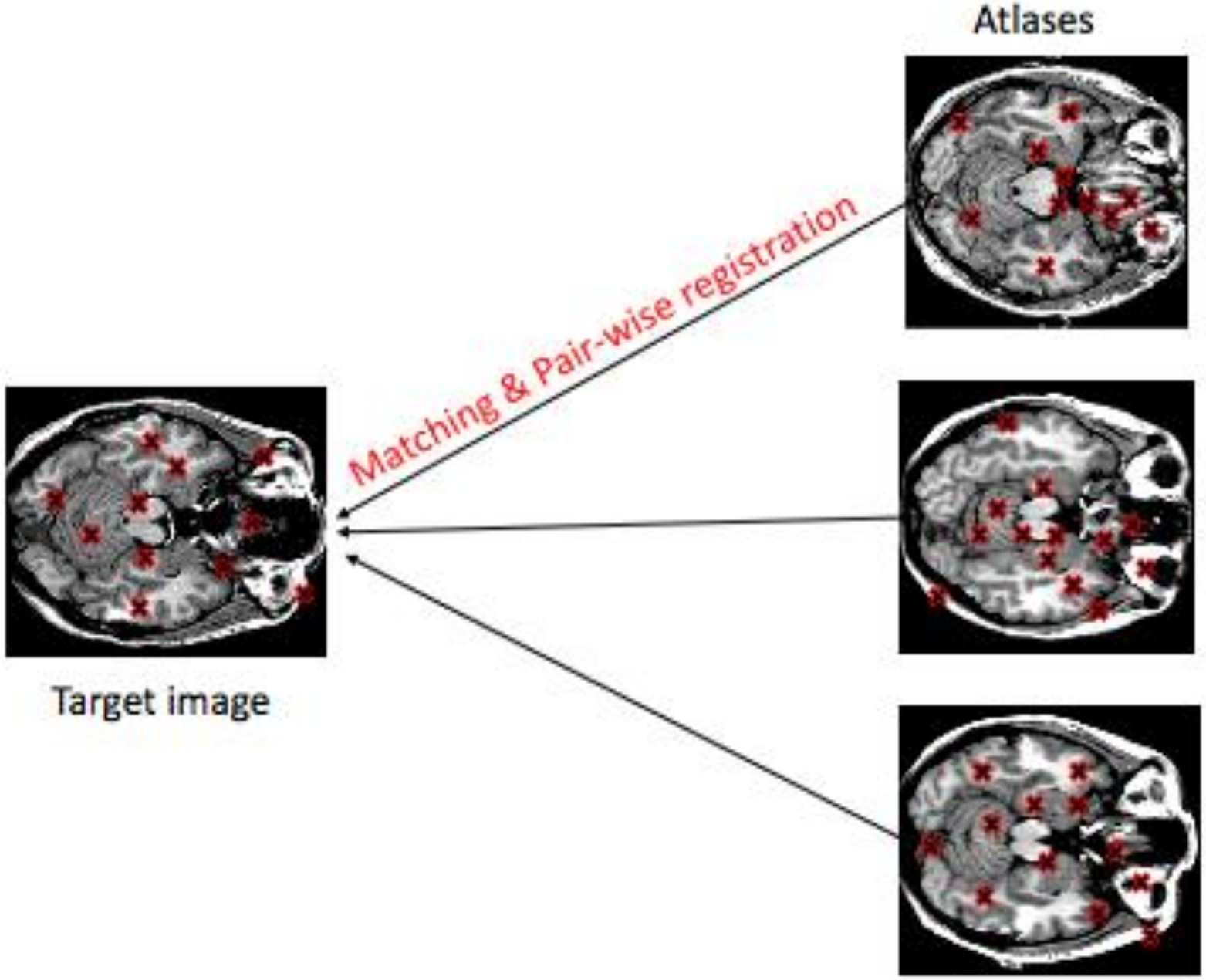
Establish pixel-level correspondences between two images



Applications: Image Alignment

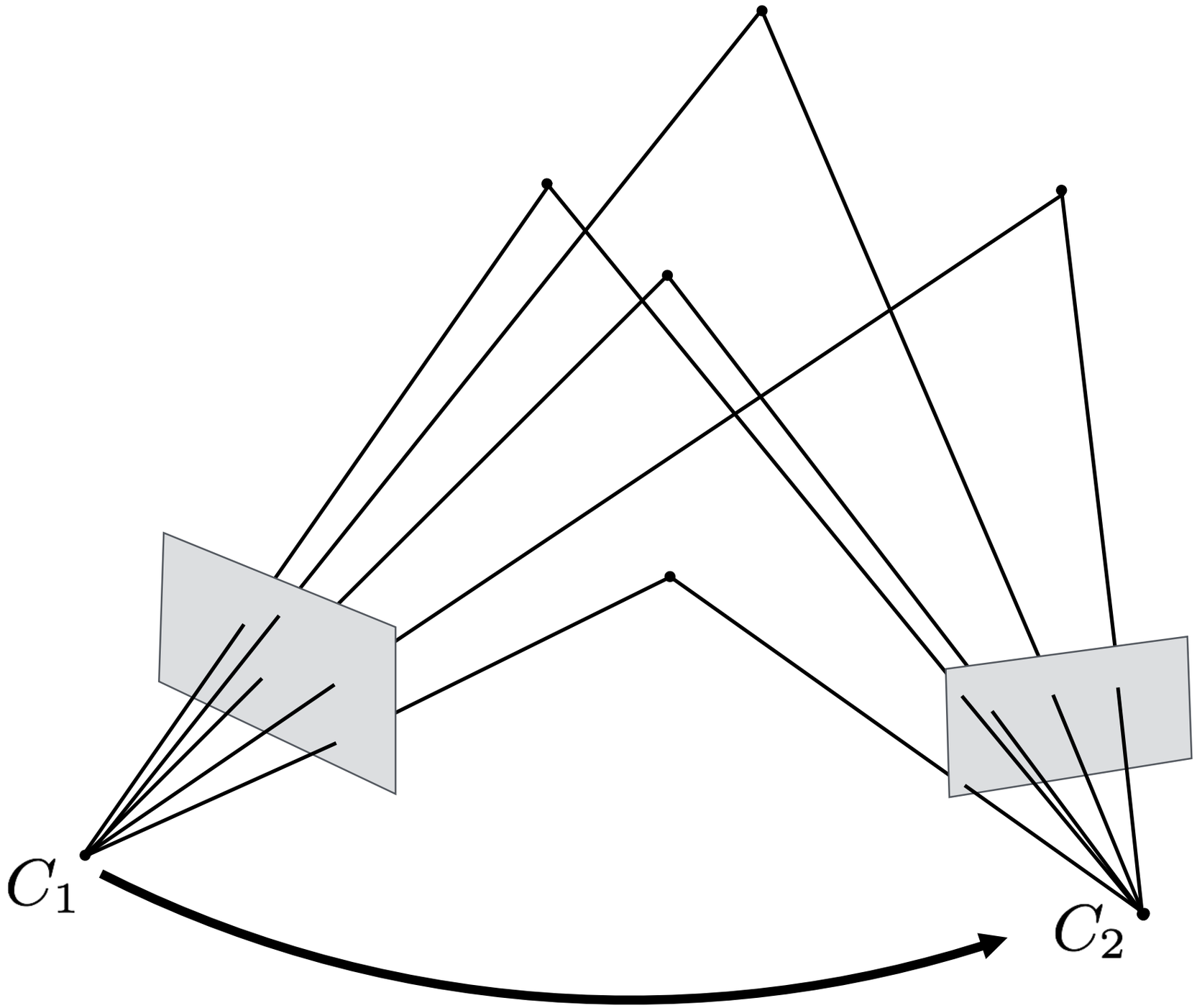
Lecture 8

Applications: MRI Image Alignment



Alvén et al.

Relative Pose Estimation



Lecture 9, 10

Applications: 3D Reconstruction

Structure-from-Motion Revisited

Johannes L. Schönberger, Jan-Michael Frahm

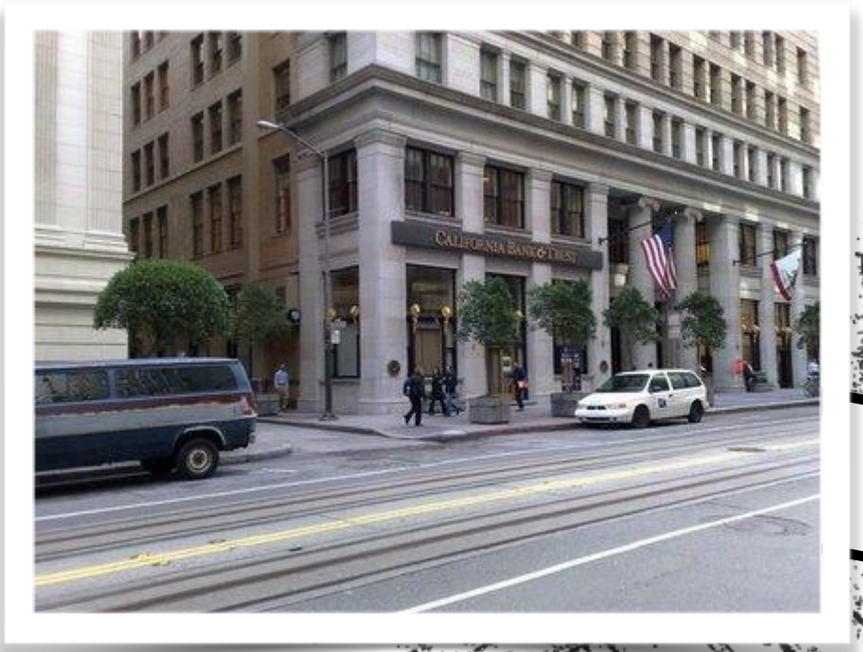
CVPR 2016

Code available at:
https://github.com/colmap/colmap

[Schönberger & Frahm, Structure-from-Motion Revisited, CVPR 2016]

Lecture 10

City-Scale Localization



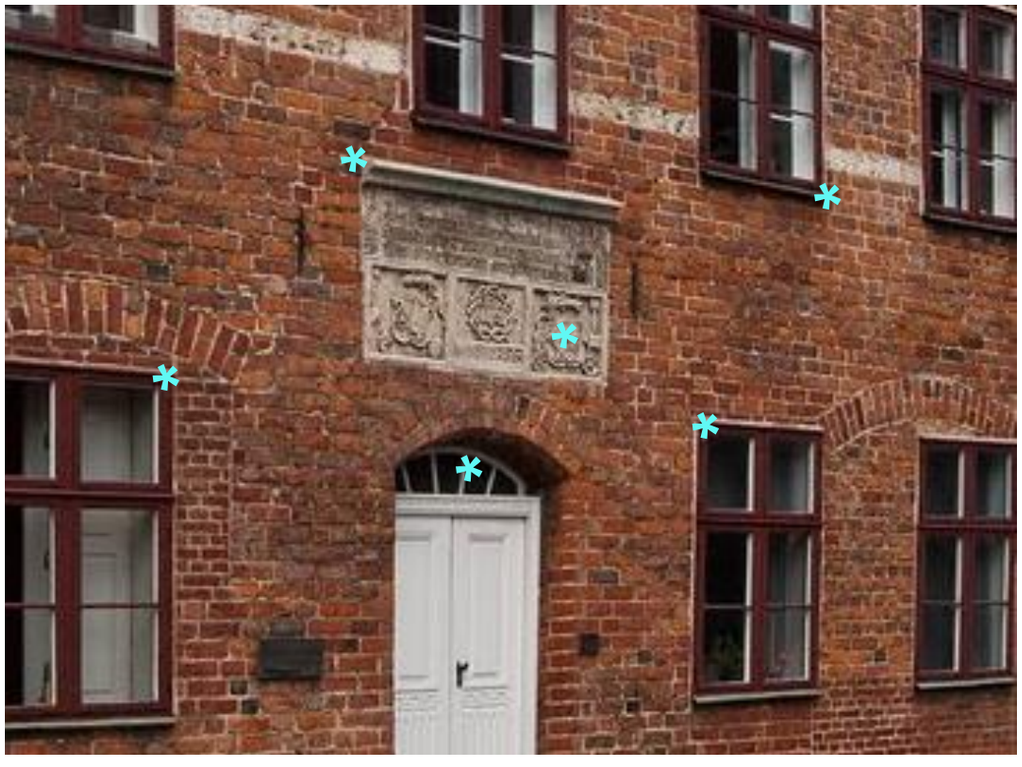
[Svärm, Enqvist, Kahl, Oskarsson, City-Scale Localization for Cameras with Known Vertical Direction, IJCV 2017 / CVPR 2014]
[Zeisl, Sattler, Pollefeys, Camera Pose Voting for Large-Scale Image-Based Localization, CVPR 2015]

Local Features

Two stage process:

Local Features

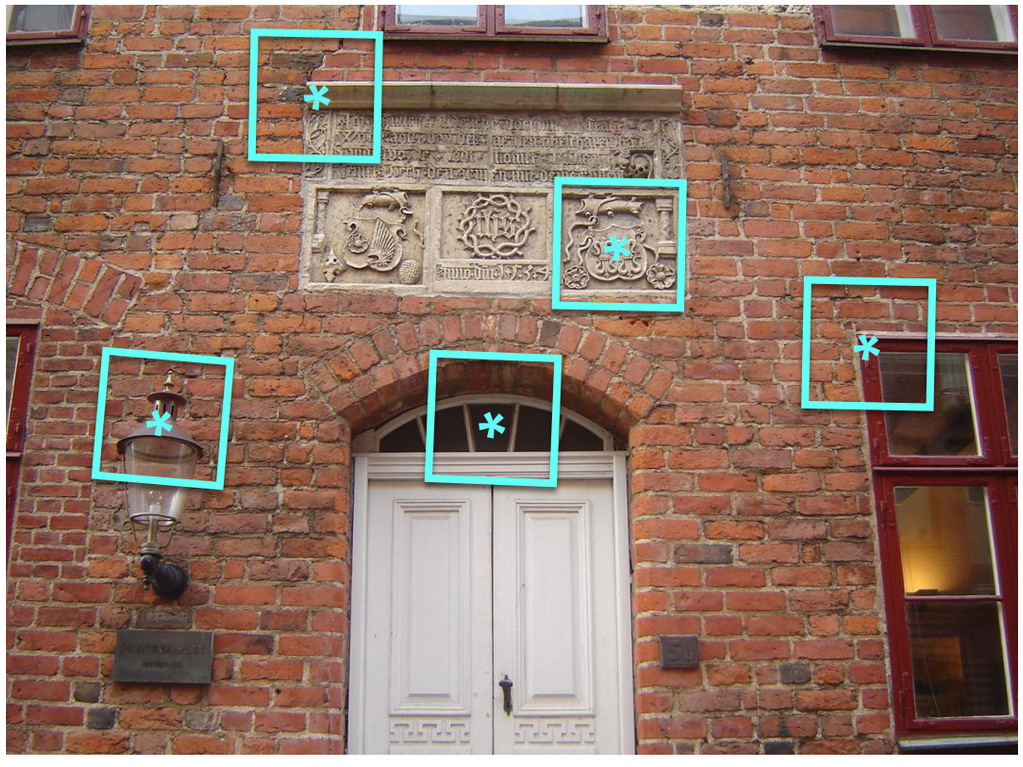
Two stage process:

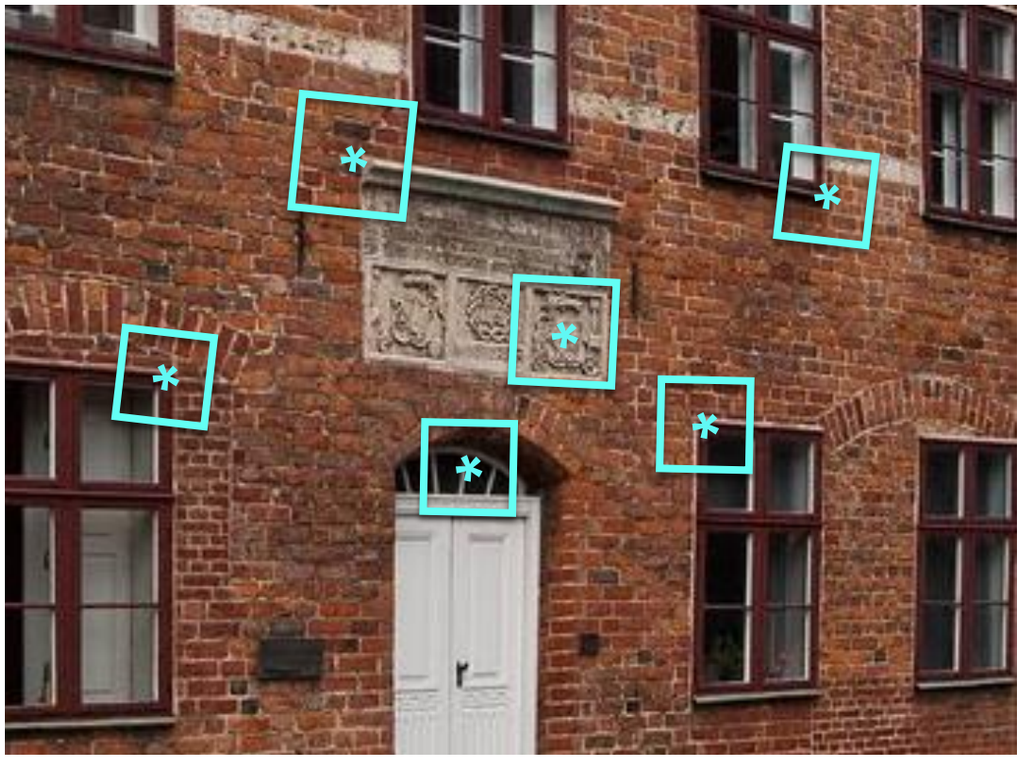


1. **Keypoint Detector**: Find “interesting points” in the images

Local Features

Two stage process:





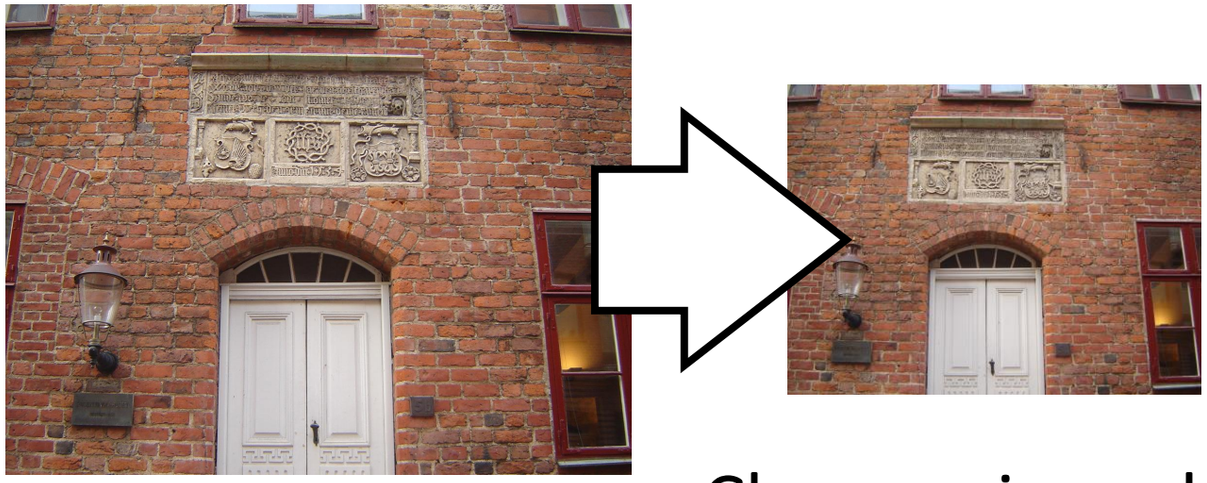
1. **Keypoint Detector**: Find “interesting points” in the images
2. **Keypoint Descriptor**: Describe regions around these points

Local Features

Desired: Robustness / invariance to (among others)

Local Features

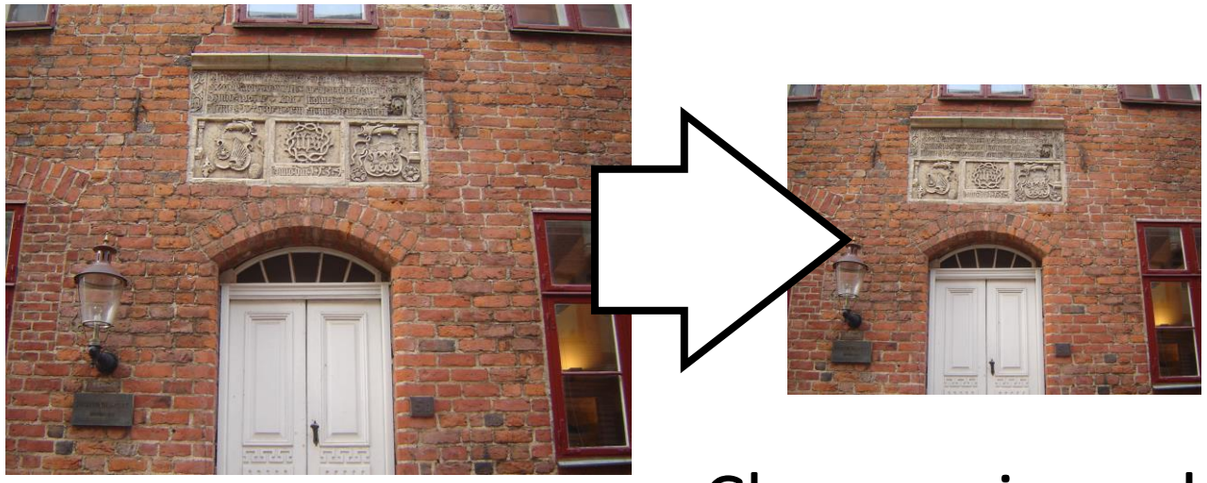
Desired: Robustness / invariance to (among others)



Changes in scale

Local Features

Desired: Robustness / invariance to (among others)

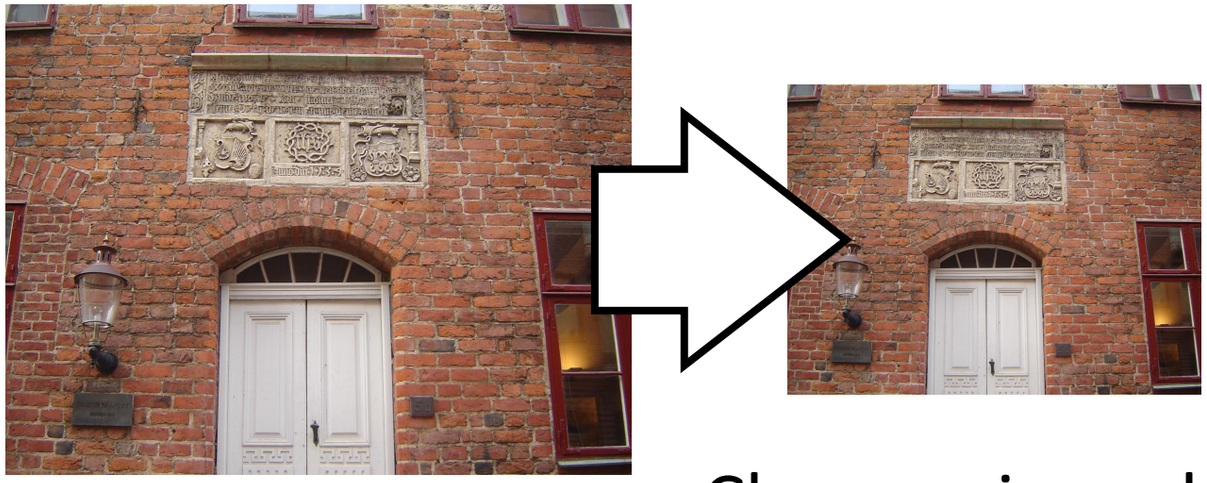


Changes in scale

in-plane rotation

Local Features

Desired: Robustness / invariance to (among others)



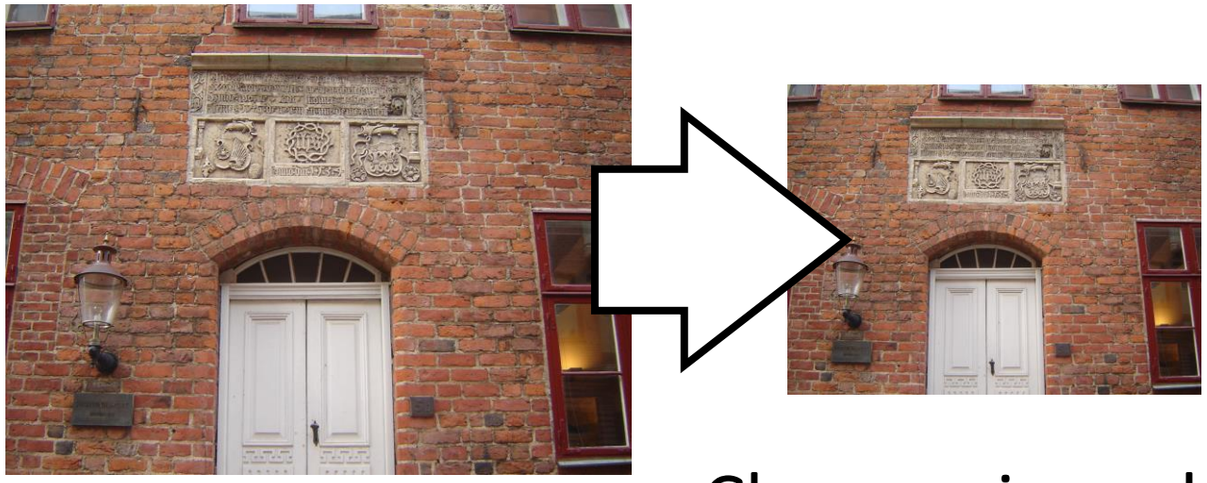
Changes in scale

in-plane rotation

Changes in brightness

Local Features

Desired: Robustness / invariance to (among others)



Changes in scale

in-plane rotation

Changes in brightness

Changes in viewpoint

Scale-Invariant Feature Transform (SIFT) Features

[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Scale-Invariant Feature Transform (SIFT) Features

- Find a set of *interest* points

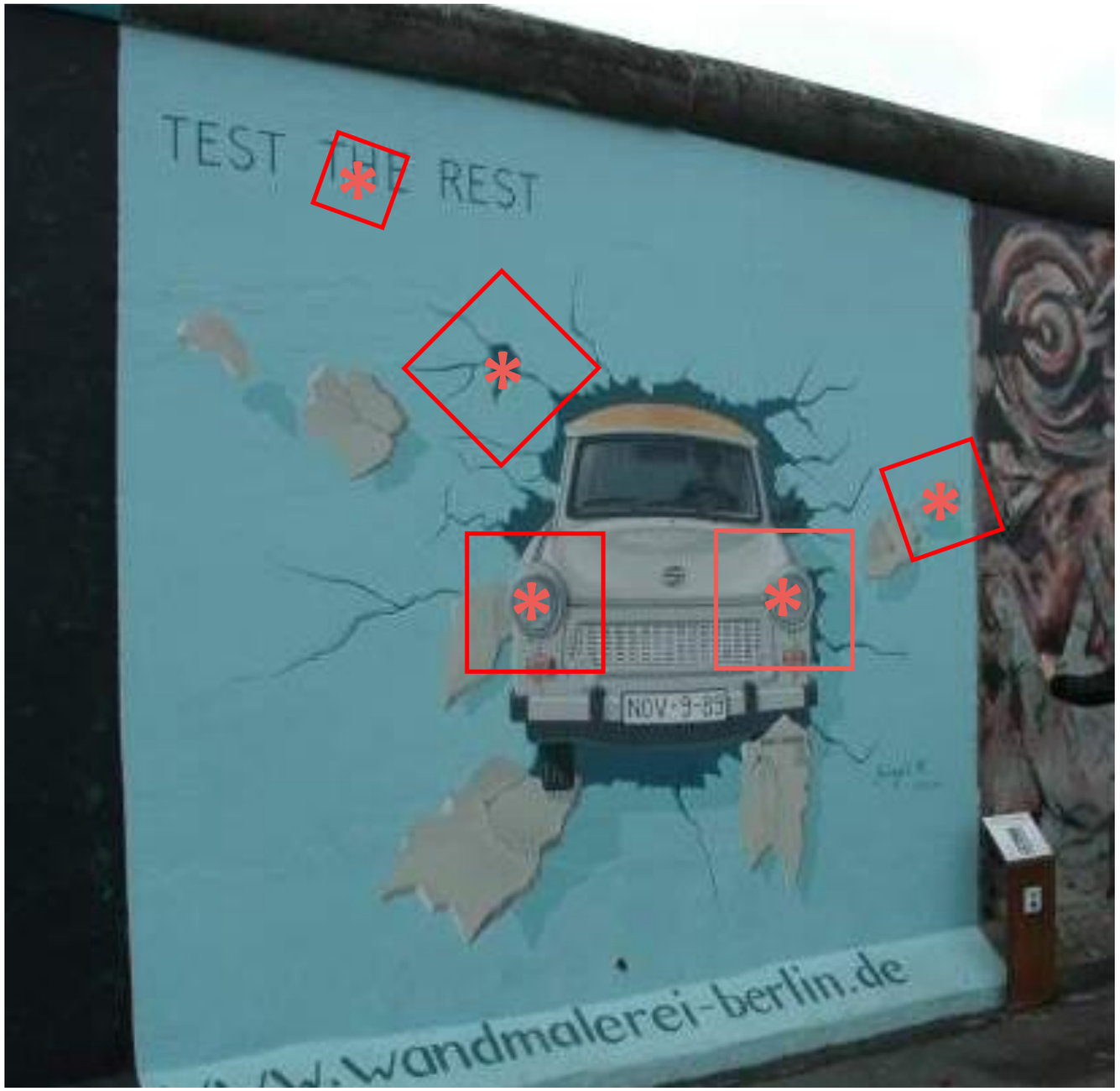
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Scale-Invariant Feature Transform (SIFT) Features

- Find a set of *interest* points
  - At different scales
  - Robust to brightness changes

[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

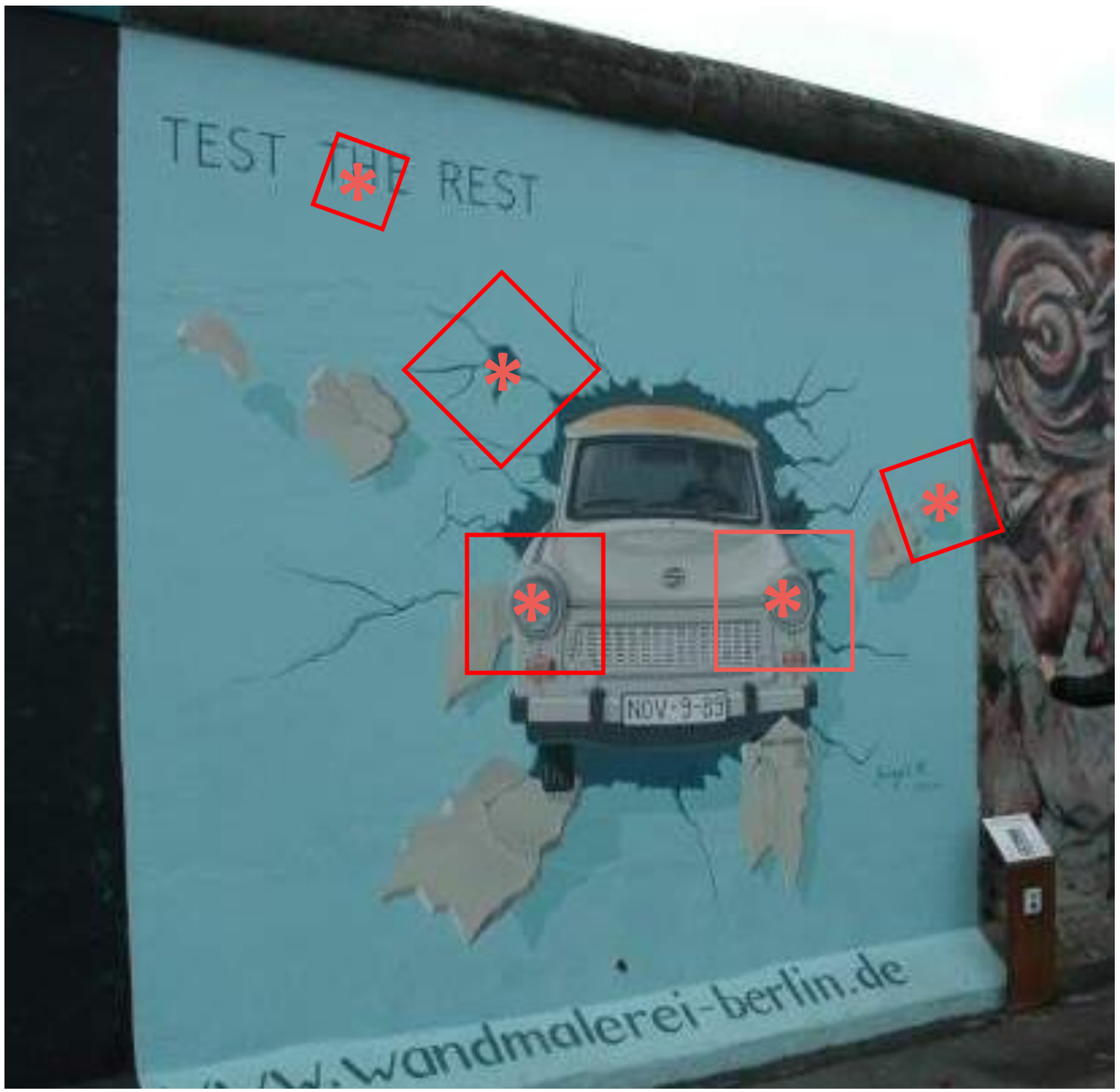
Scale-Invariant Feature Transform (SIFT) Features



- Find a set of *interest* points
- Extract scaled & oriented patch around interest points

[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

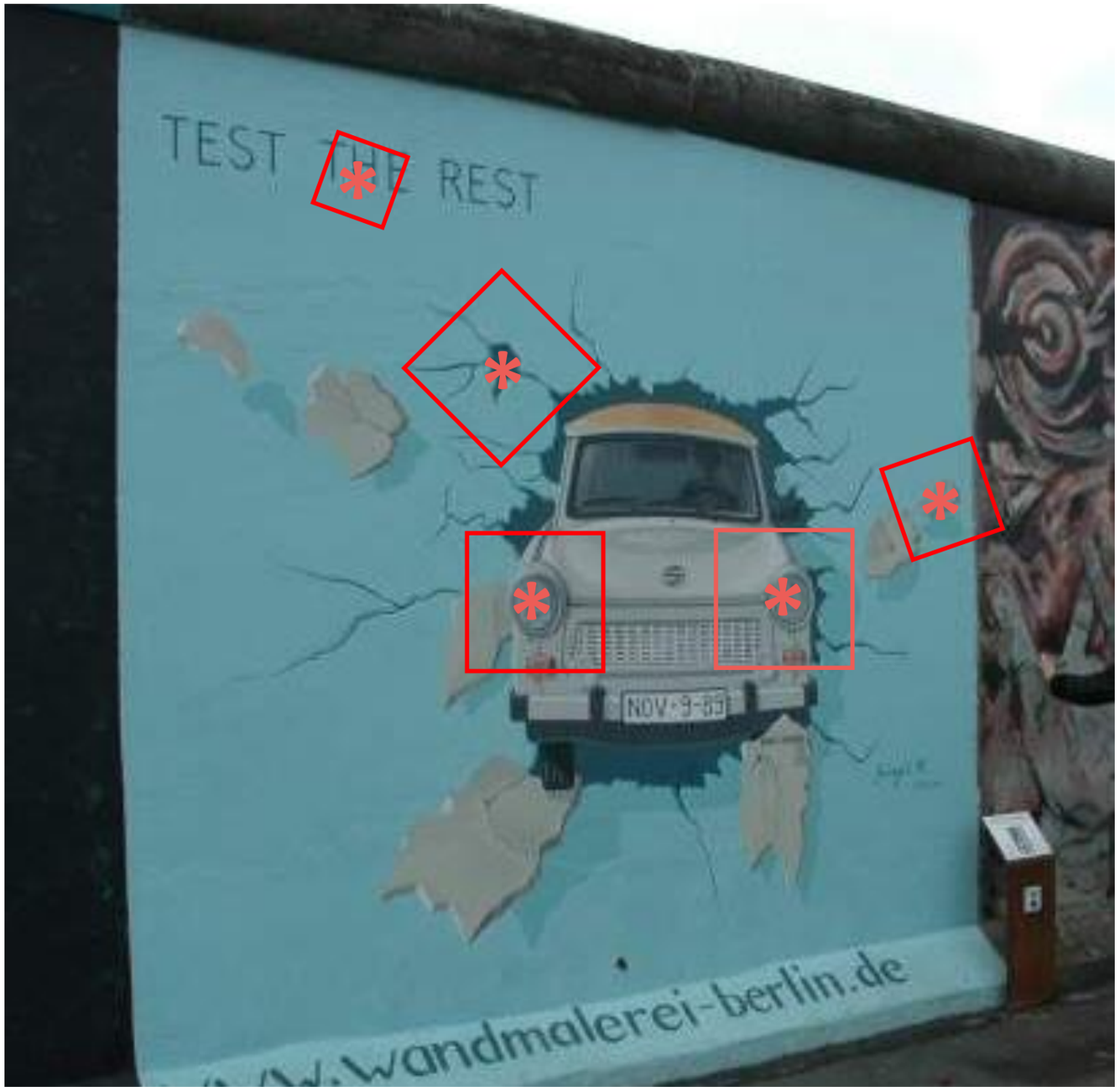
Scale-Invariant Feature Transform (SIFT) Features



- Find a set of *interest* points
- Extract scaled & oriented patch around interest points
  - Invariance to rotations
  - Invariance to scale changes

[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

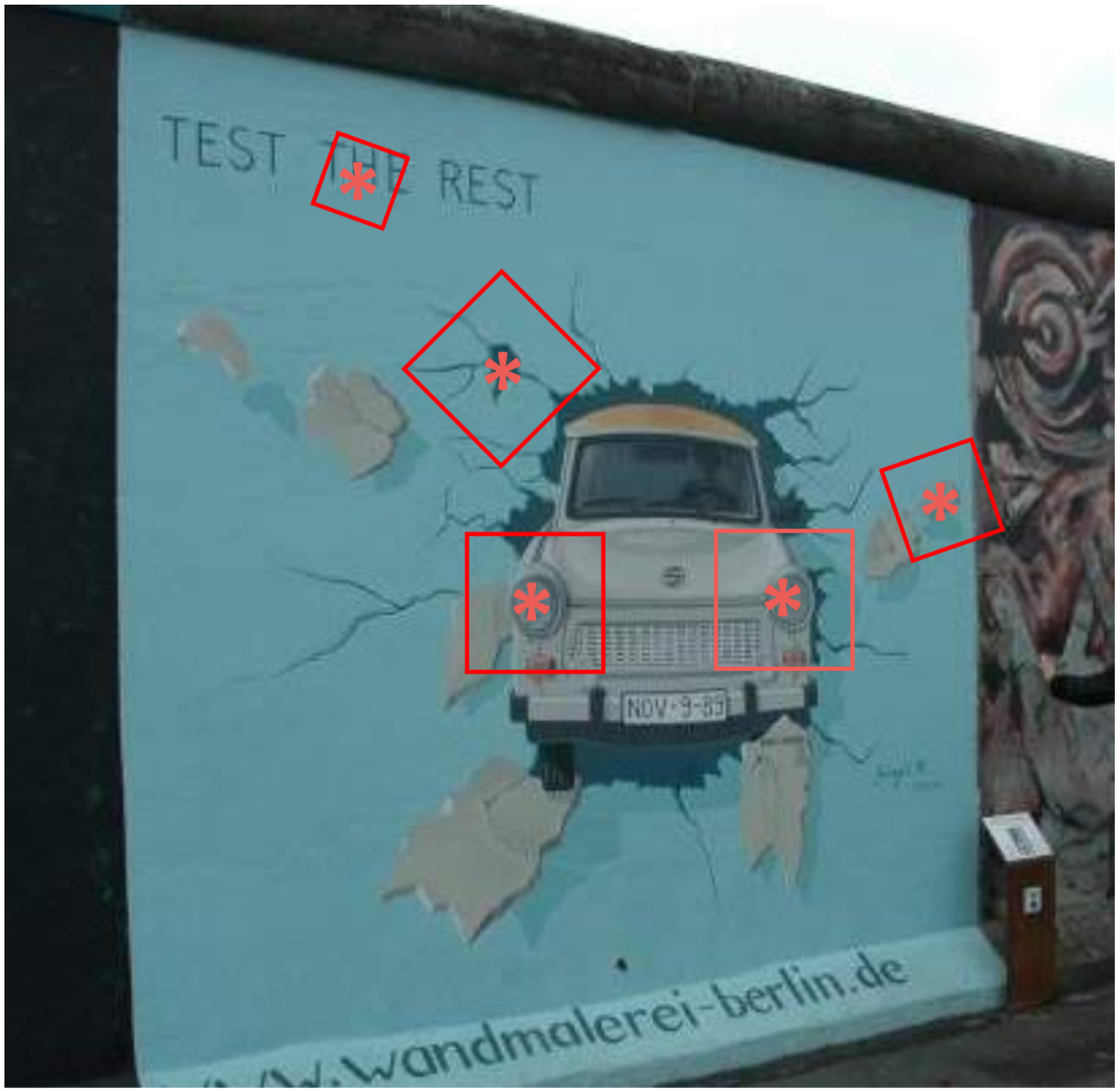
Scale-Invariant Feature Transform (SIFT) Features



- Find a set of *interest* points
- Extract scaled & oriented patch around interest points
- Compute SIFT descriptor for each patch

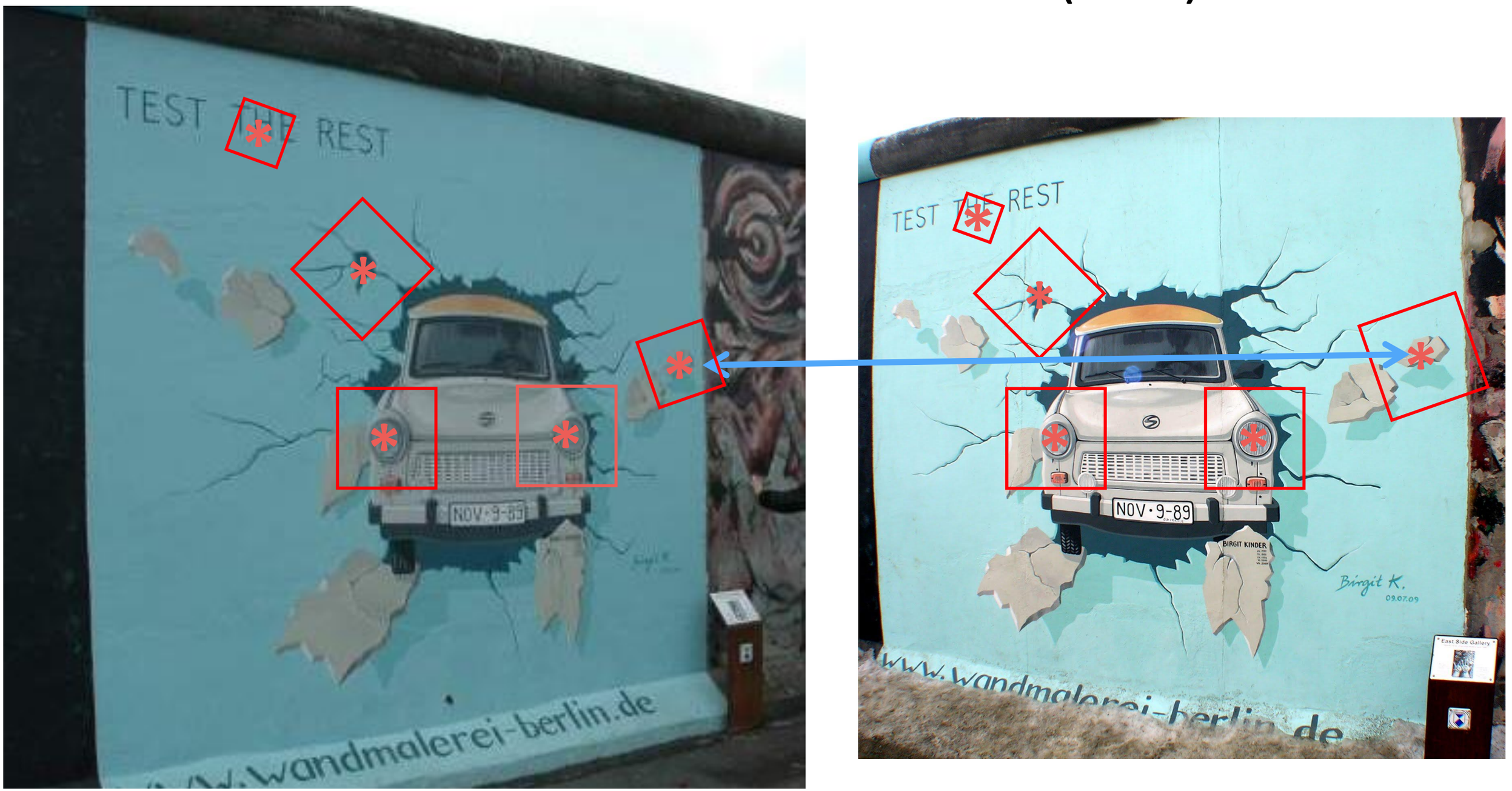
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Scale-Invariant Feature Transform (SIFT) Features



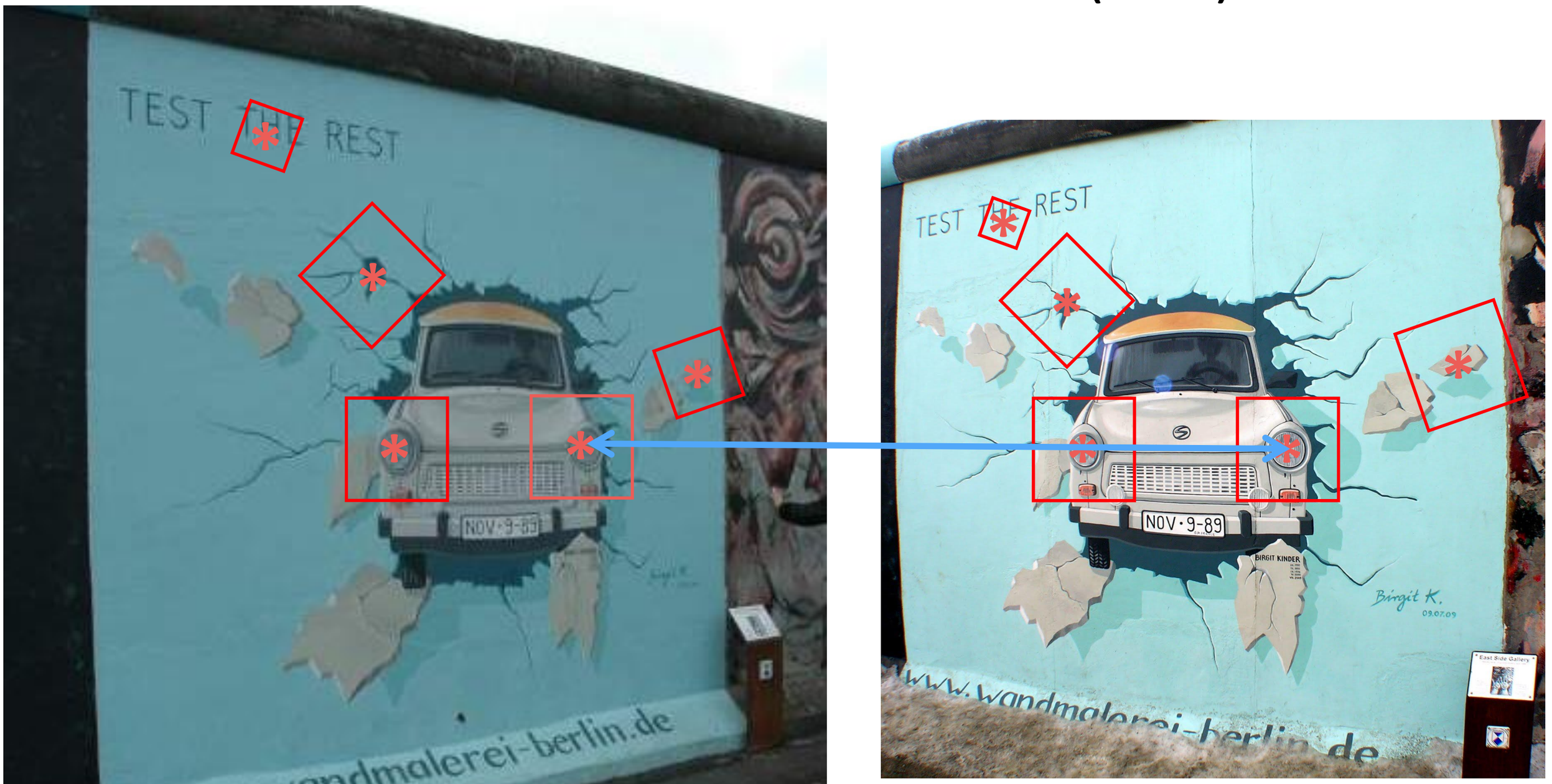
Feature Matching: Use descriptor similarity (L_2 distance) to find corresponding points in different images

Scale-Invariant Feature Transform (SIFT) Features



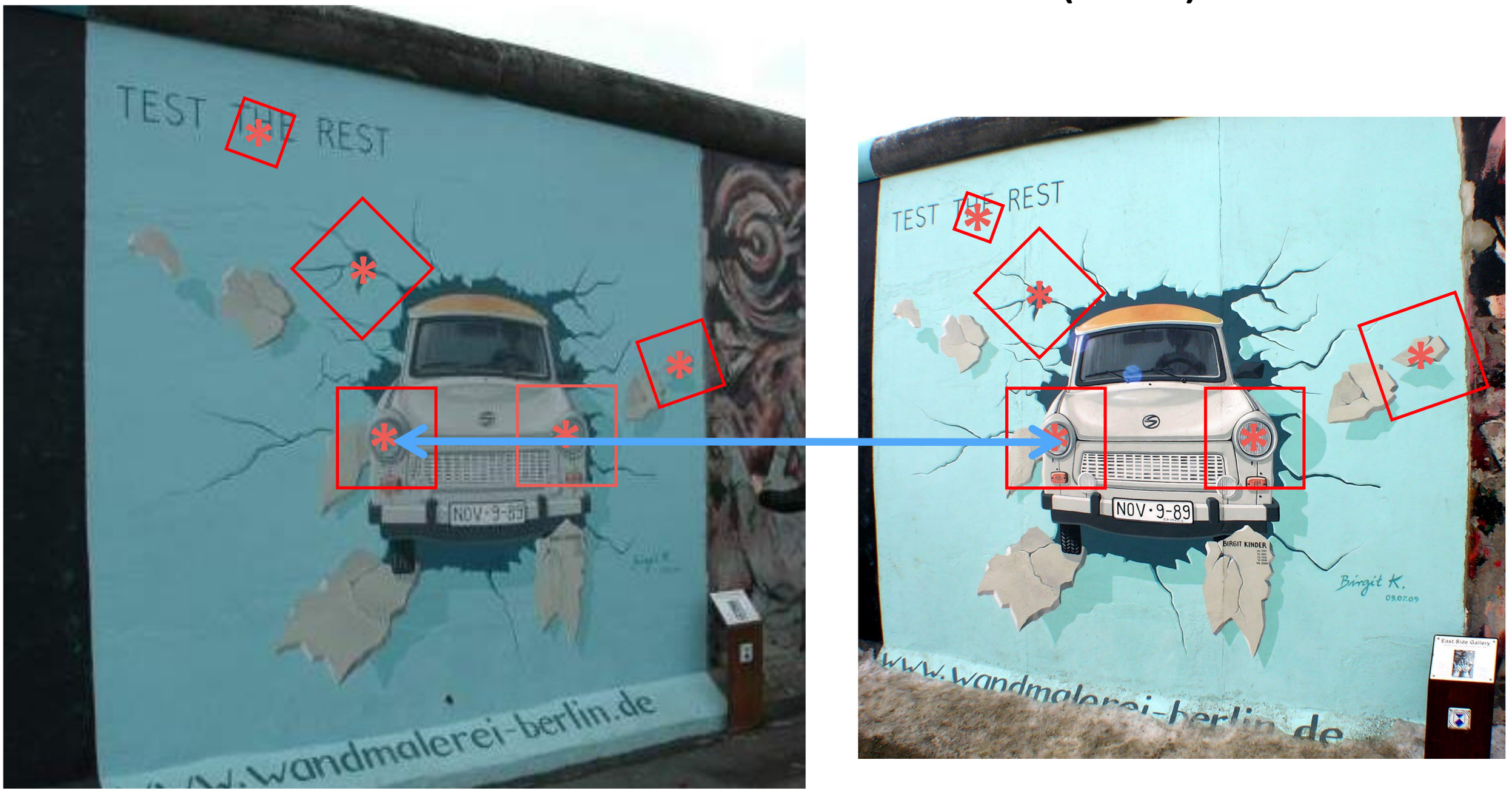
Feature Matching: Use descriptor similarity (L_2 distance) to find corresponding points in different images

Scale-Invariant Feature Transform (SIFT) Features



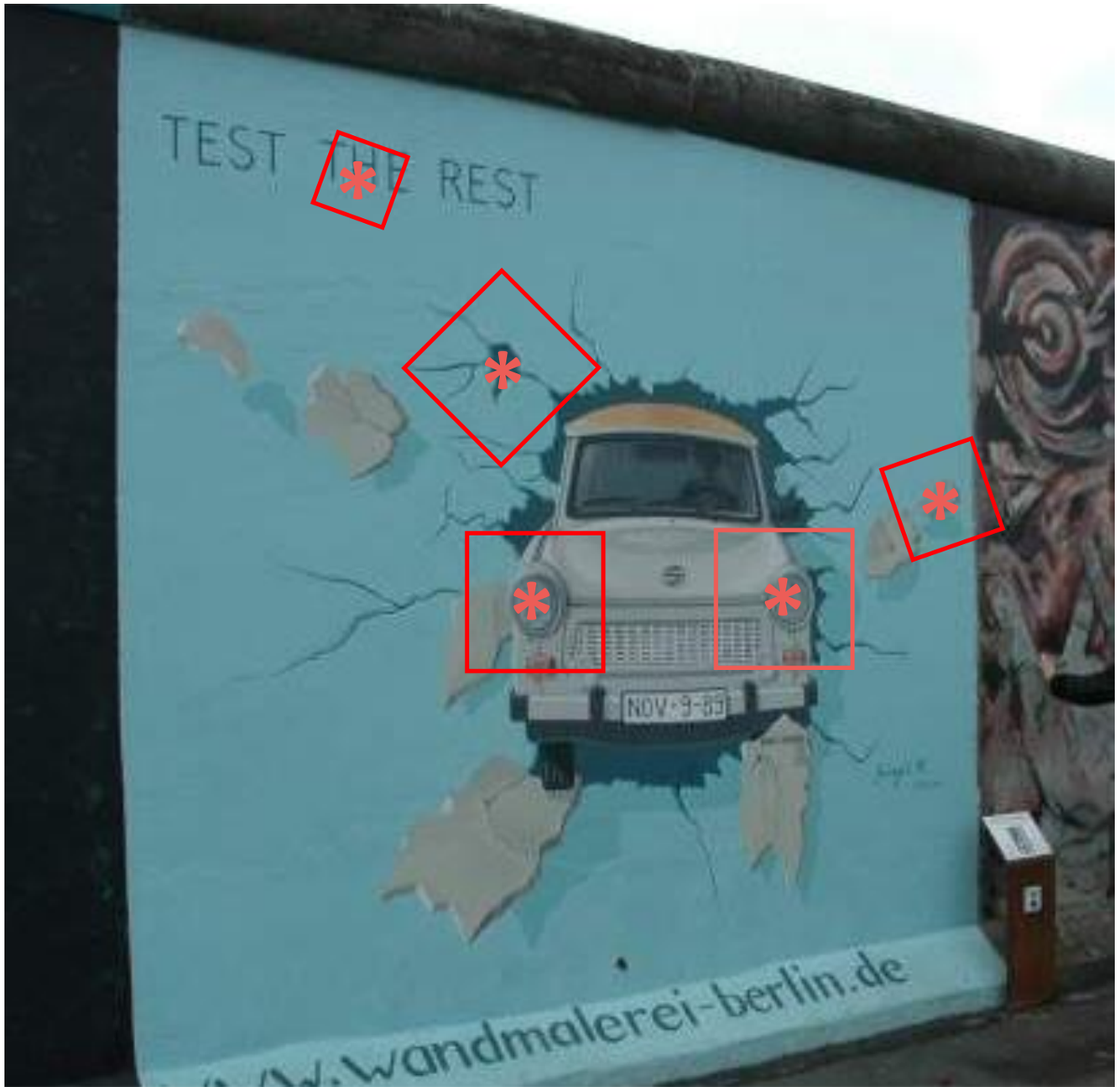
Feature Matching: Use descriptor similarity (L_2 distance) to find corresponding points in different images

Scale-Invariant Feature Transform (SIFT) Features



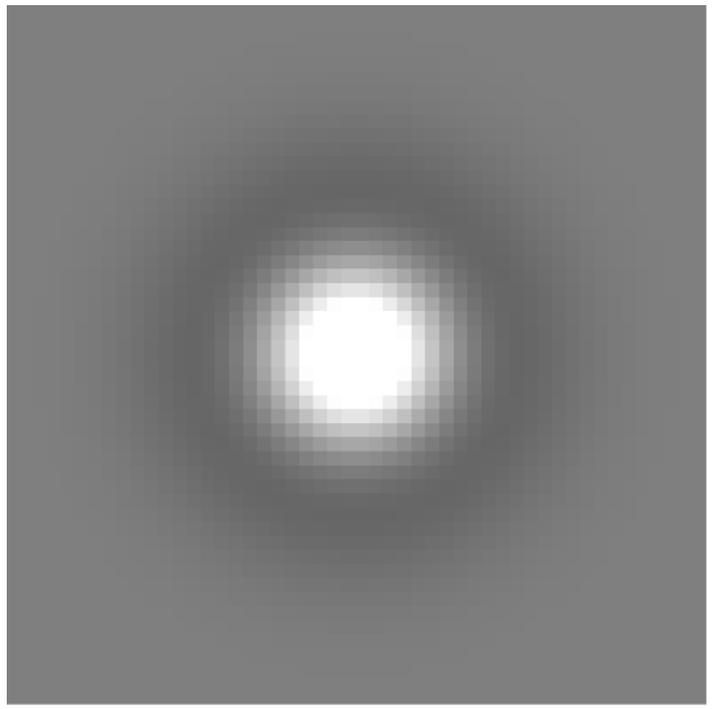
Feature Matching: Use descriptor similarity (L_2 distance) to find corresponding points in different images

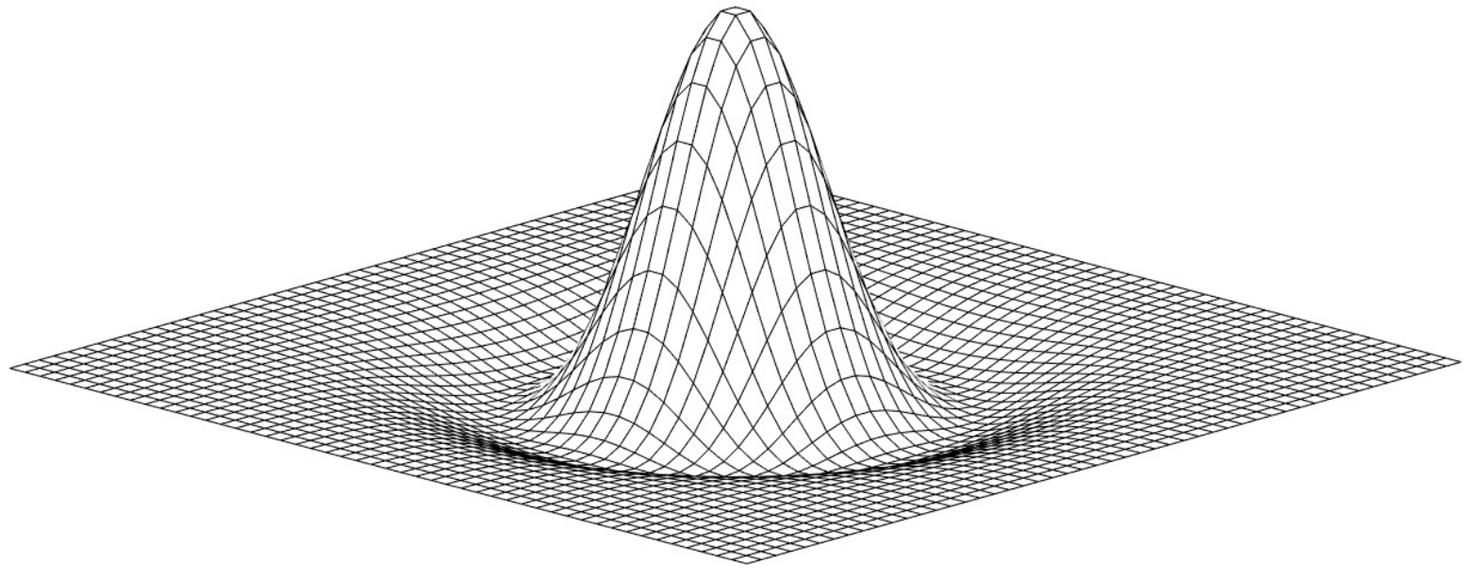
Scale-Invariant Feature Transform (SIFT) Features



- **Find a set of *interest* points**
- Extract oriented patch around each interest point
- Compute SIFT descriptor for each patch

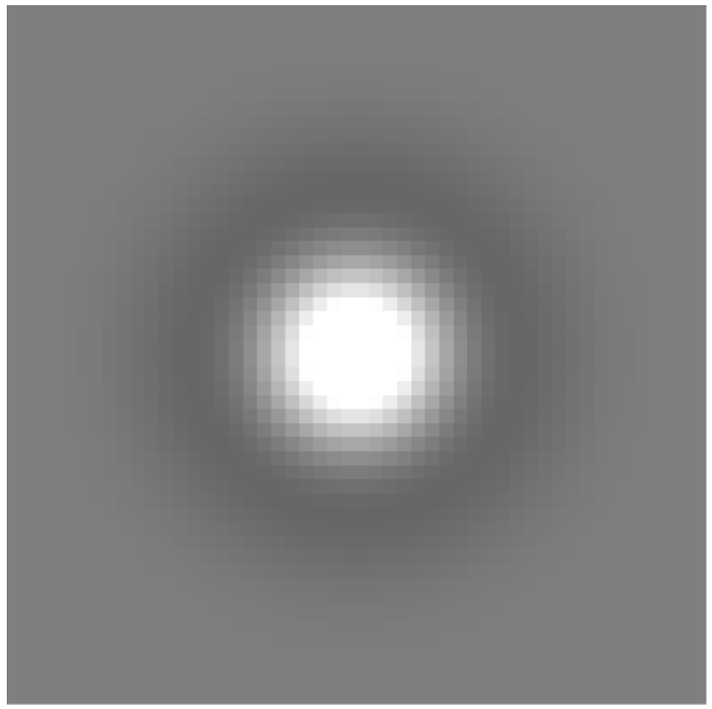
Keypoint Detector: Difference of Gaussians (DoG)

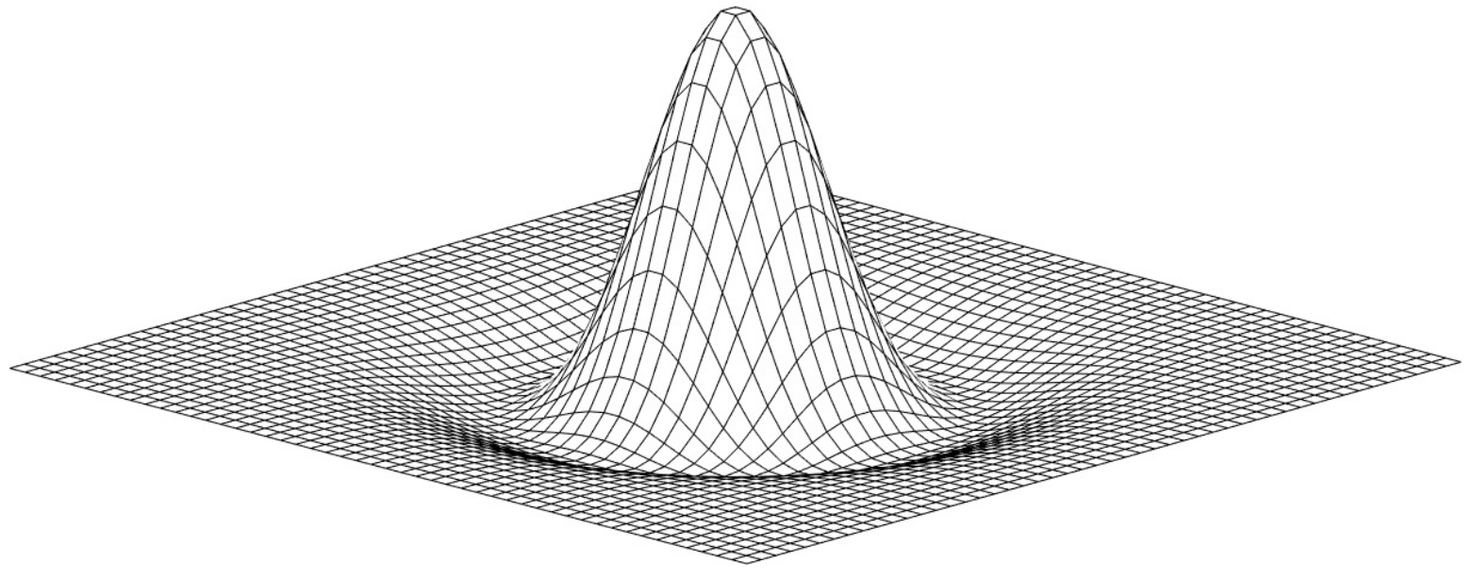




- Apply DoG filter over the image
- Keypoints: Local extrema of DoG response
- Why? Efficient approximation to Laplacian of Gaussians
  - Produces very stable detections [Mikolajczyk, Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France, 2002]

Keypoint Detector: Difference of Gaussians (DoG)





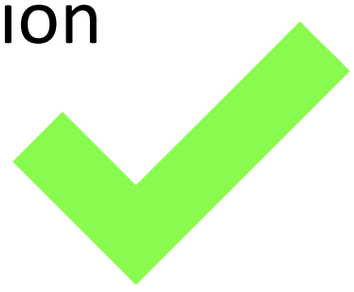
- Difference of two Gaussian filters
- Intuitive explanation of extrema:
  - Local maxima at bright blobs
  - Local minima at dark blobs

Robustness / Invariances?

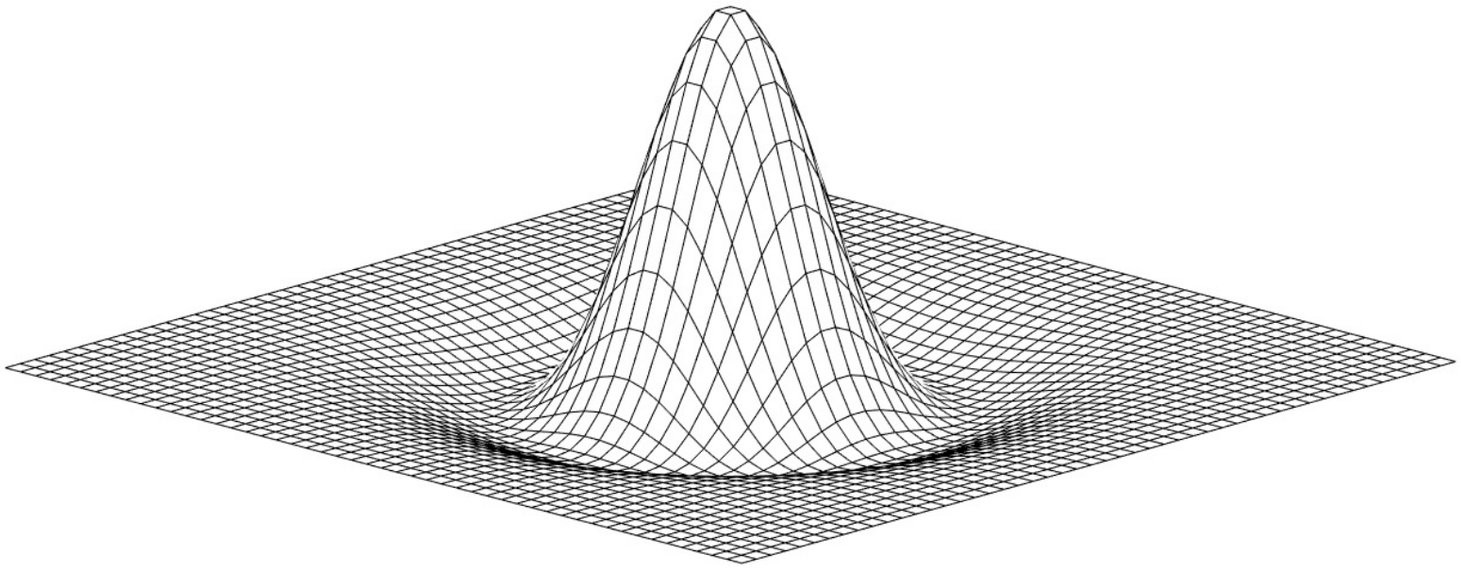
in-plane rotation

Robustness / Invariances?

in-plane rotation



Robustness / Invariances?



Filter is rotation symmetric

in-plane rotation

Robustness / Invariances?

Changes in brightness

Robustness / Invariances?

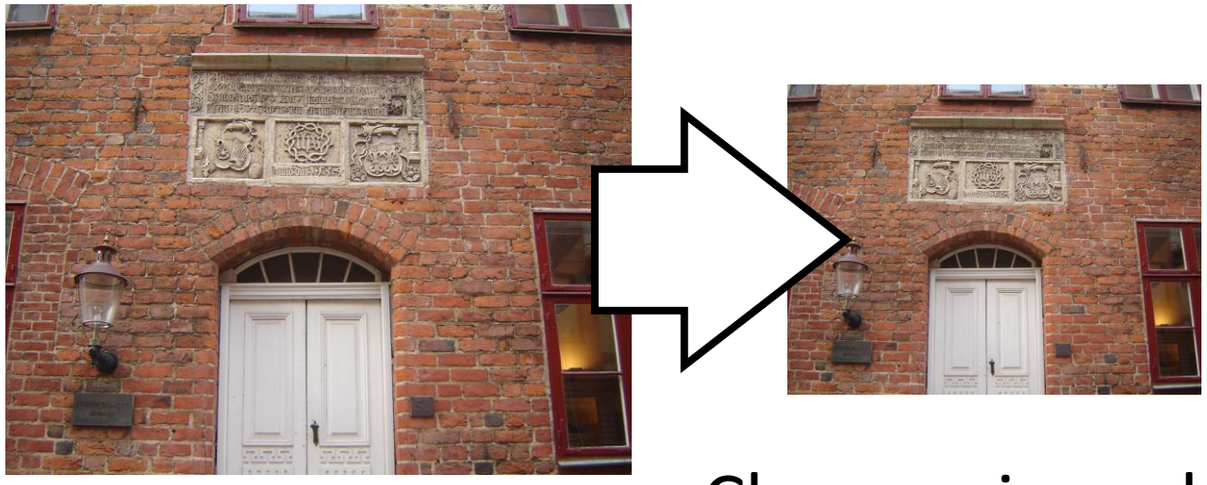
Local minima / maxima unaffected* by:

- Additive changes in brightness
- Multiplicative changes in brightness

* As long as inside dynamic range of camera

Changes in brightness

Robustness / Invariances?



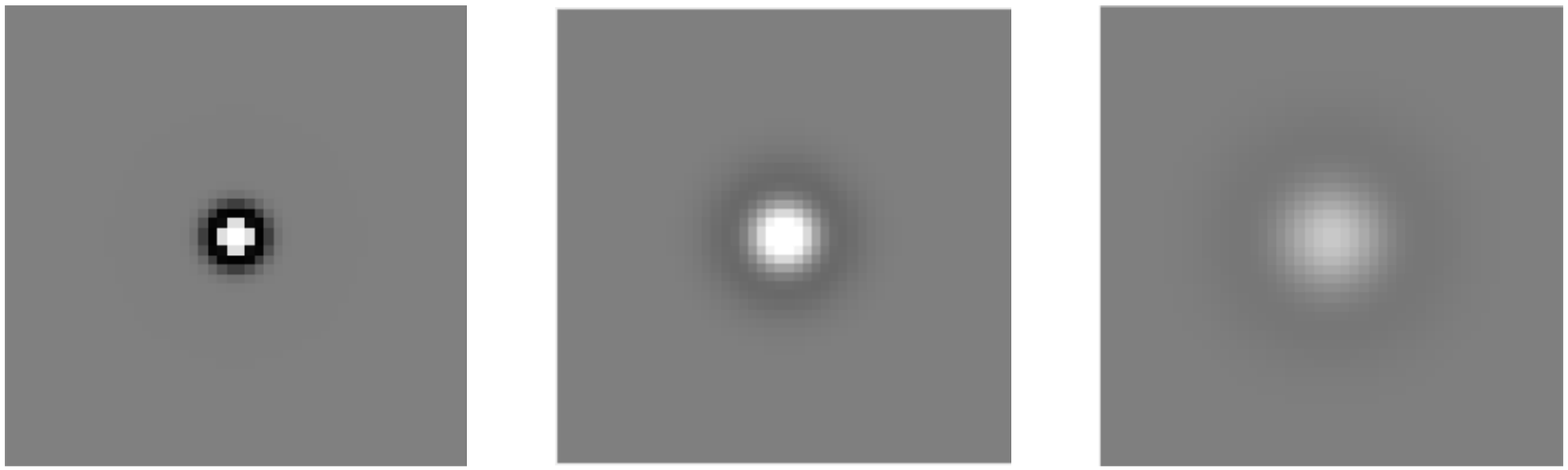
Changes in scale

The Need for Multi-Scale Detections

The Need for Multi-Scale Detections

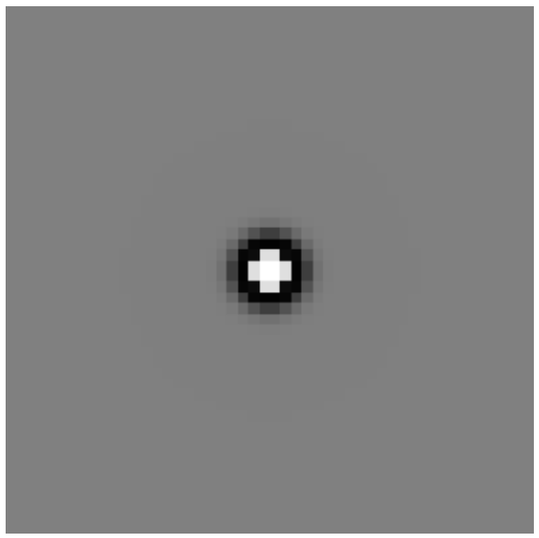
Difference of Gaussians (DoG)

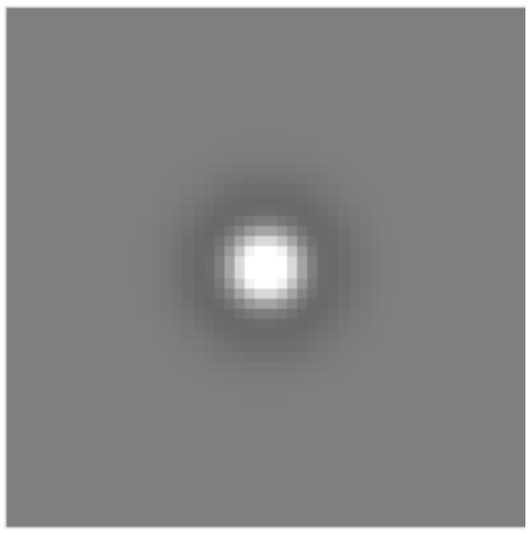
Use filters of different size

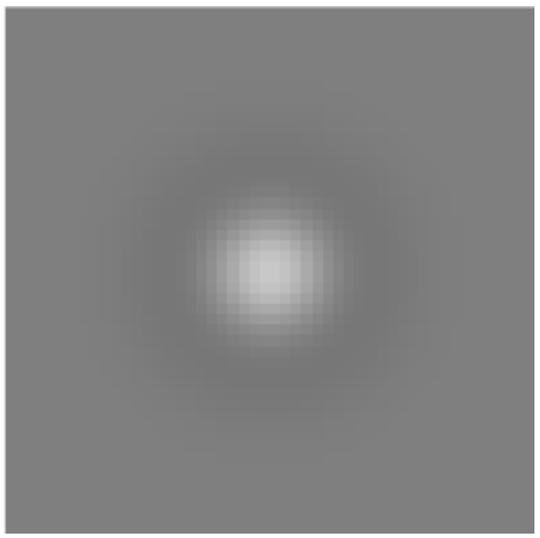


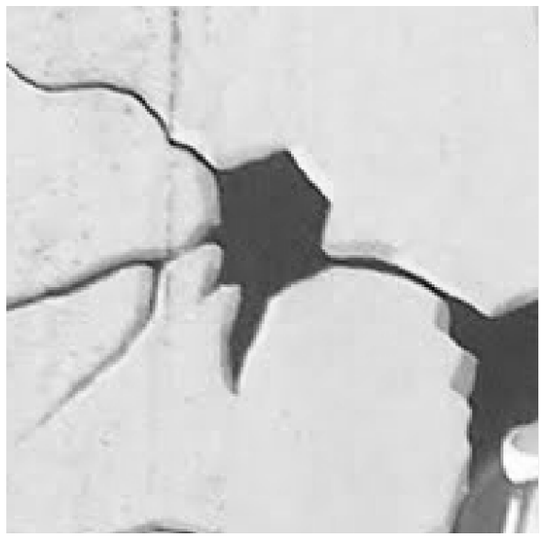
Difference of Gaussians (DoG)

Use filters of different size



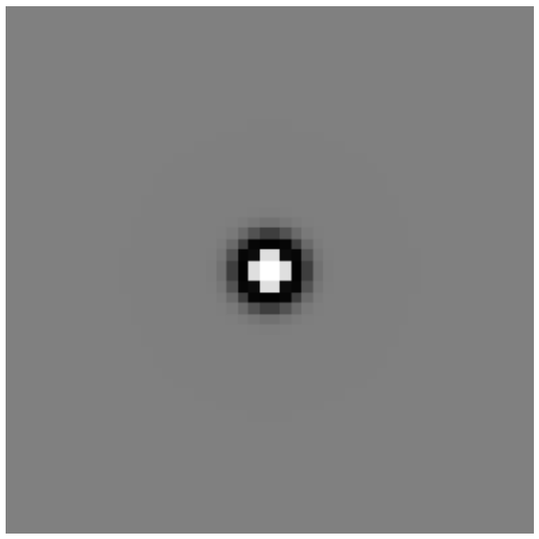


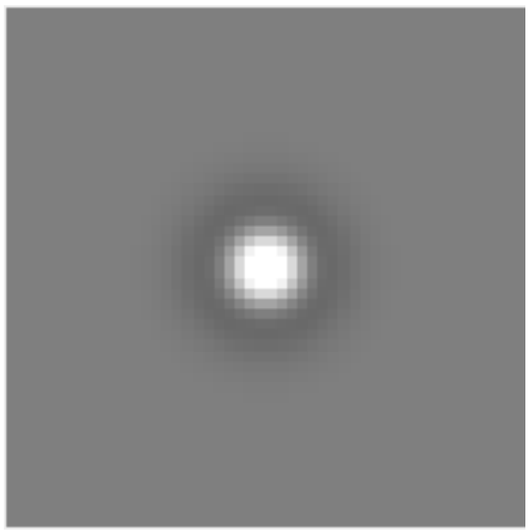


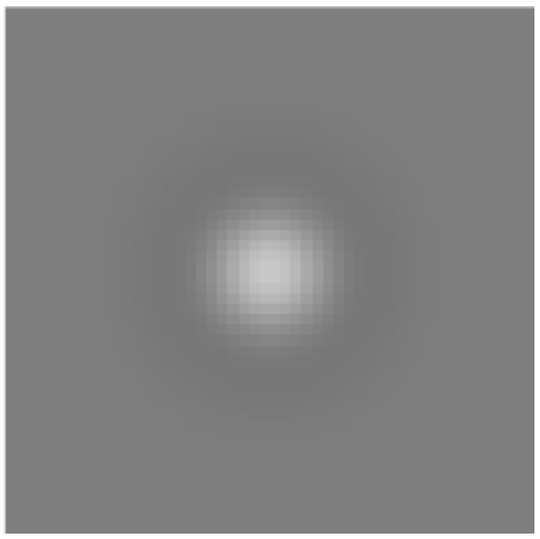


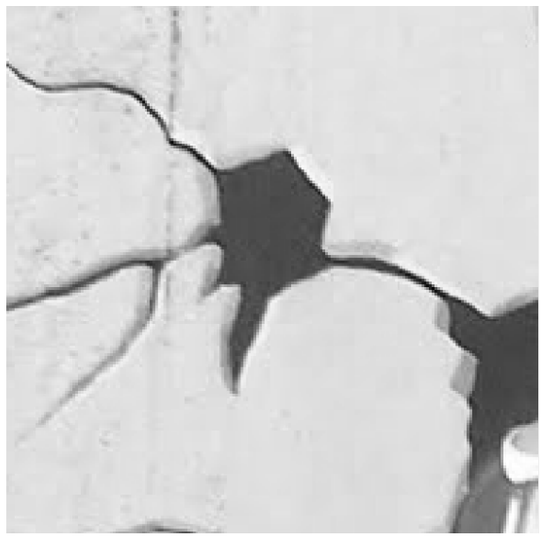
Difference of Gaussians (DoG)

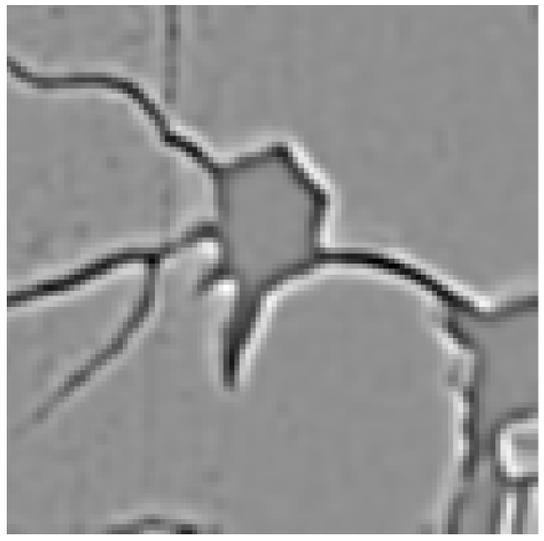
Use filters of different size





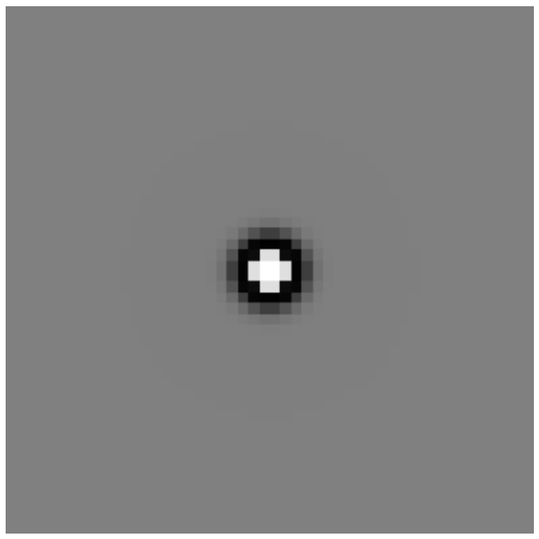


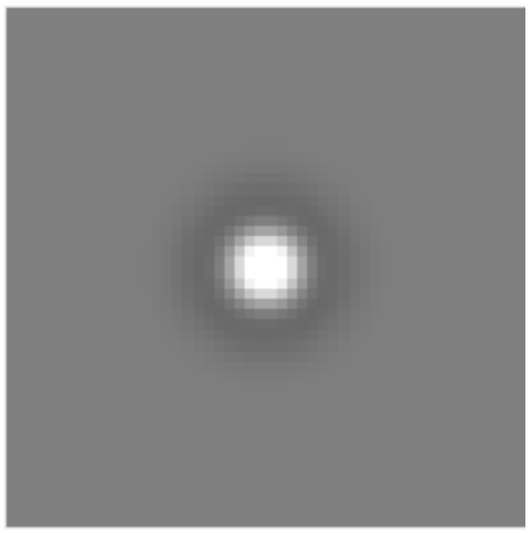


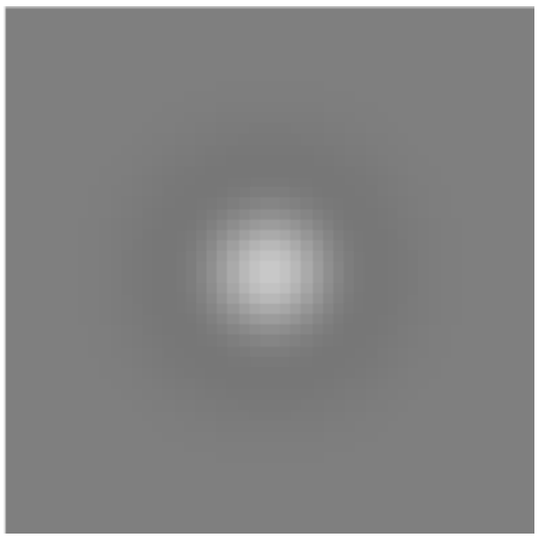


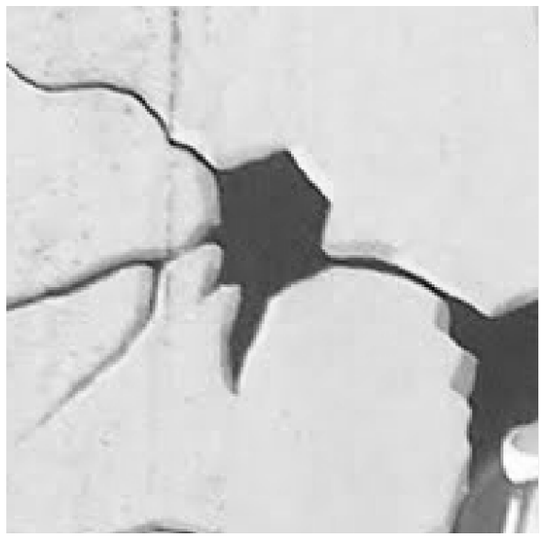
Difference of Gaussians (DoG)

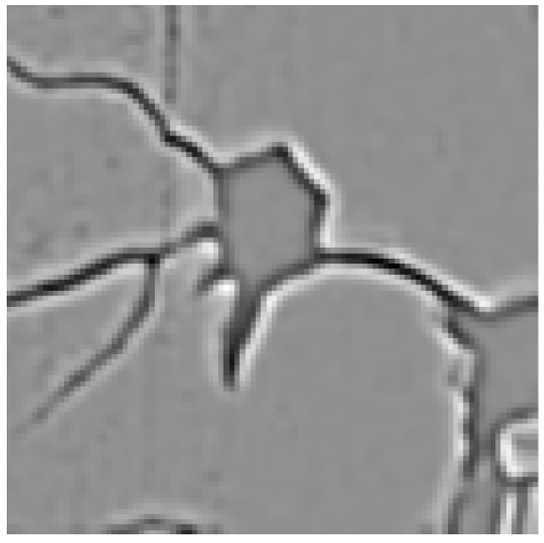
Use filters of different size

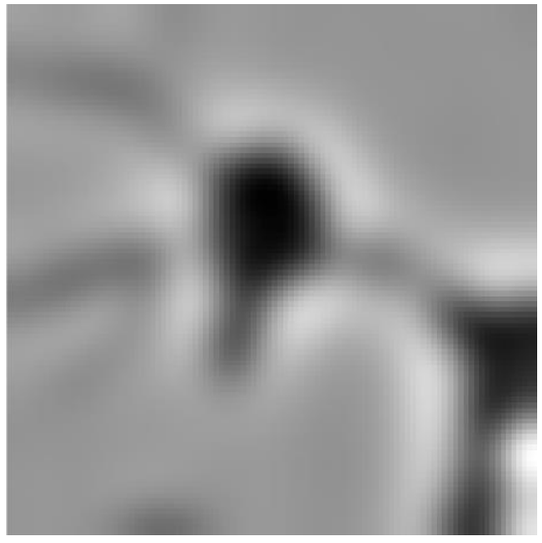






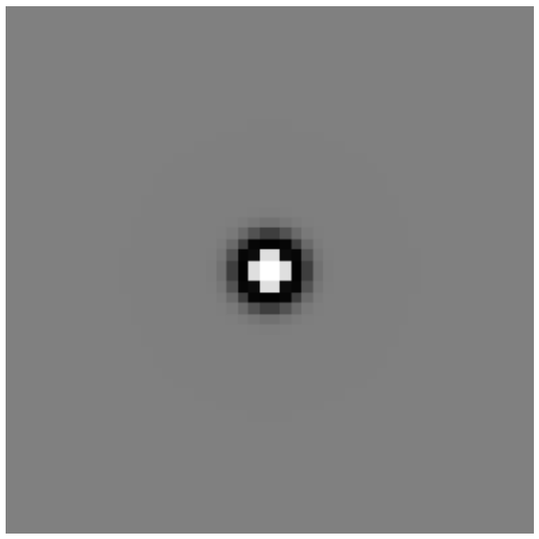


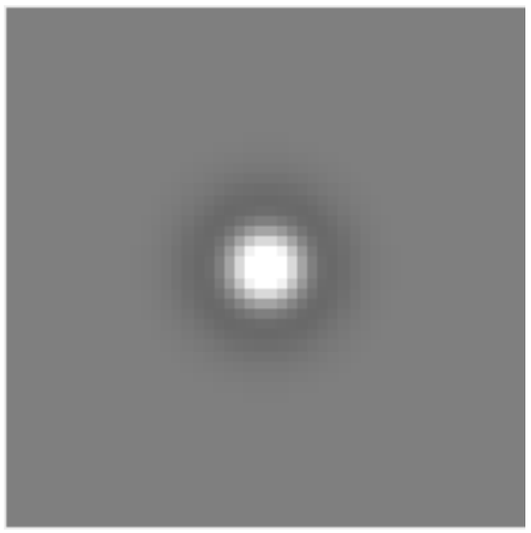


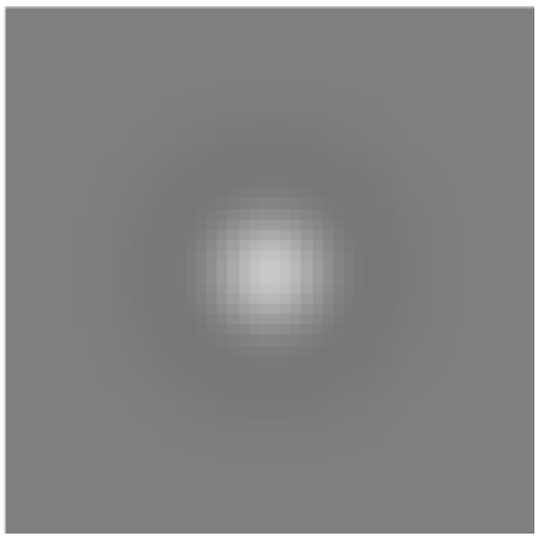


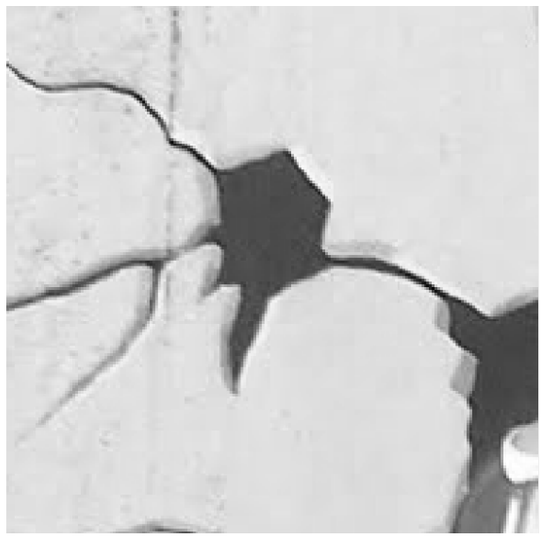
Difference of Gaussians (DoG)

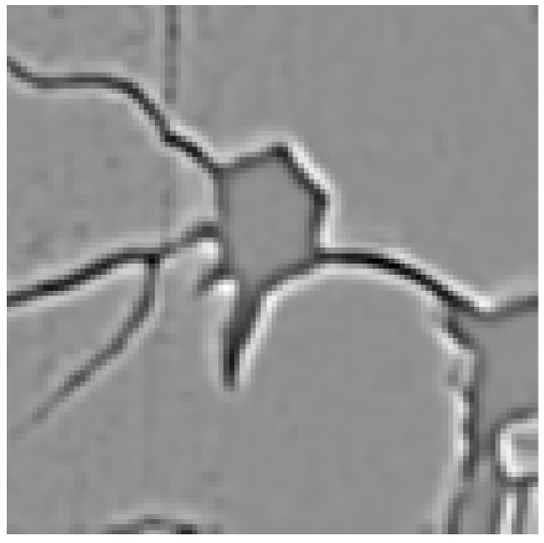
Use filters of different size

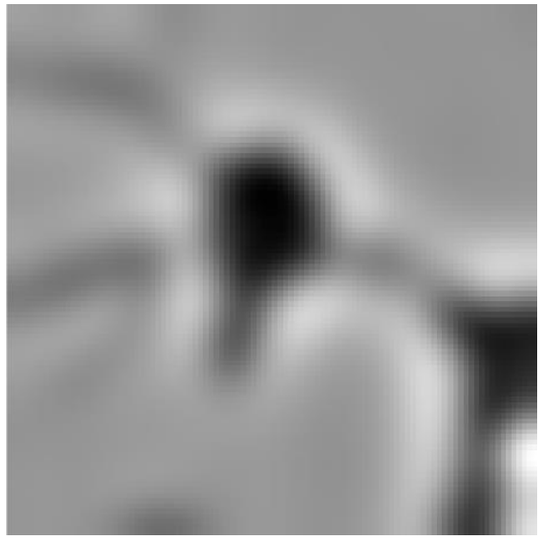


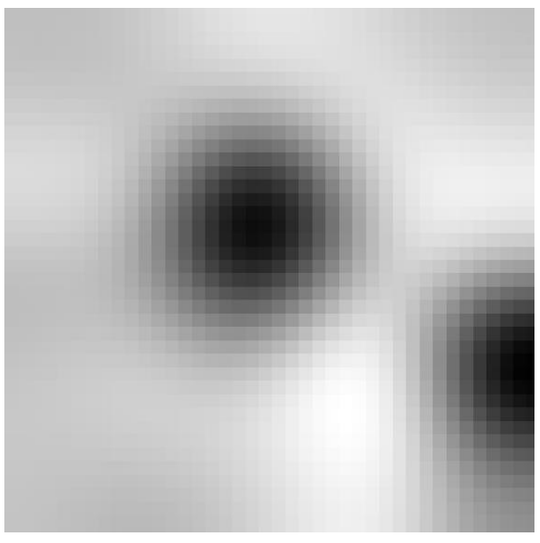






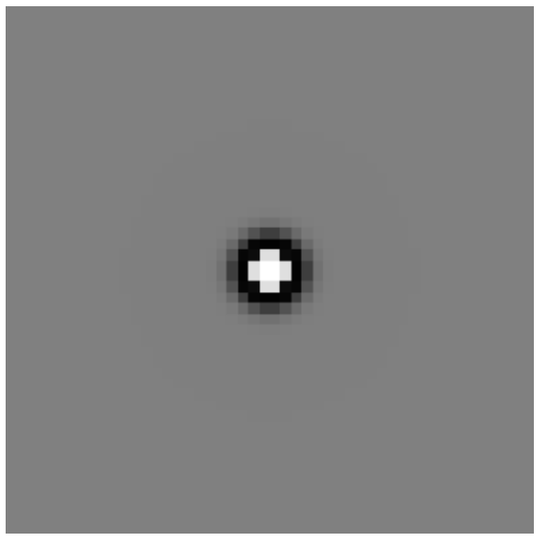


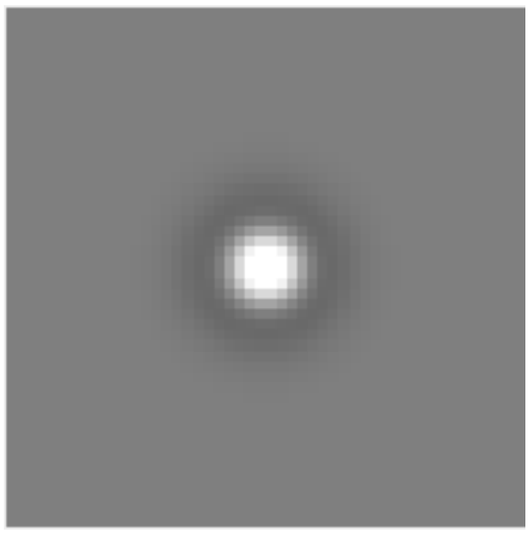


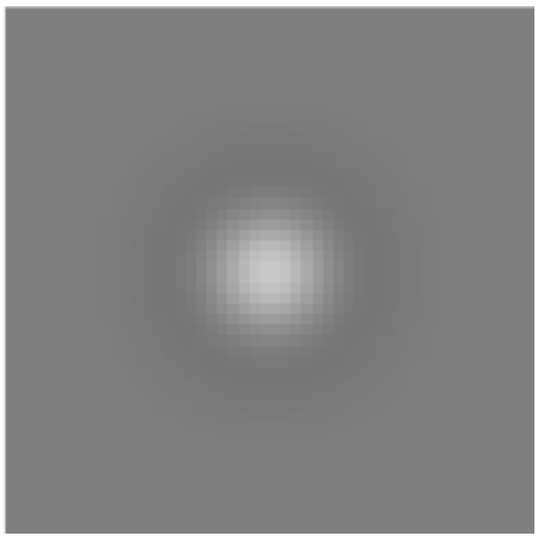


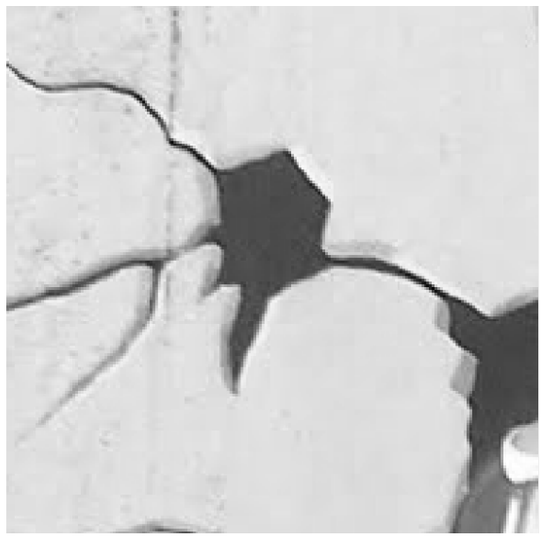
Difference of Gaussians (DoG)

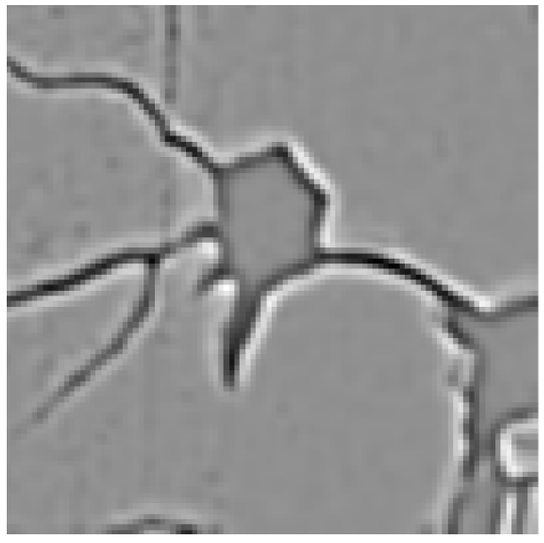
Use filters of different size

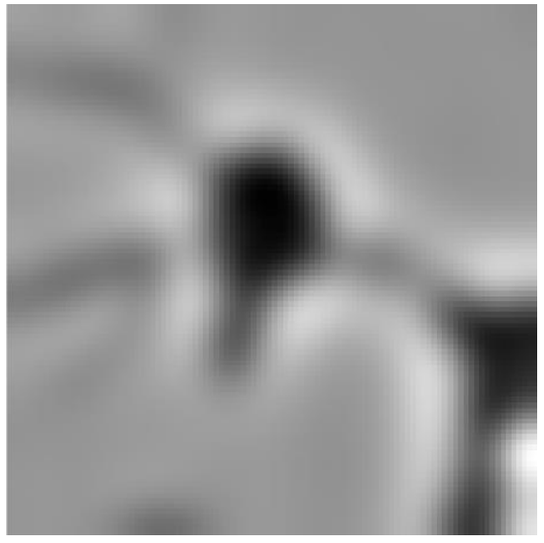


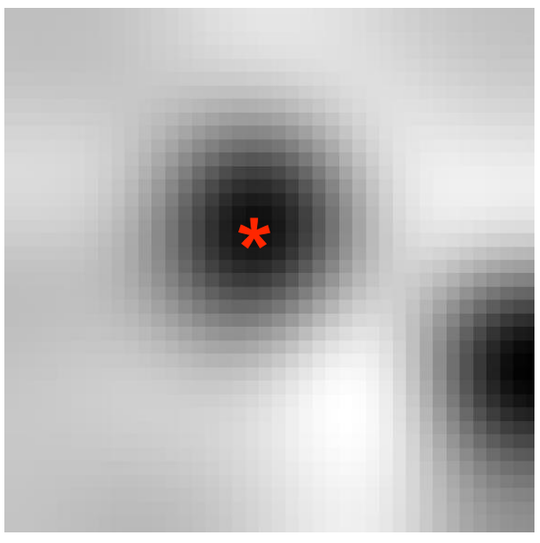




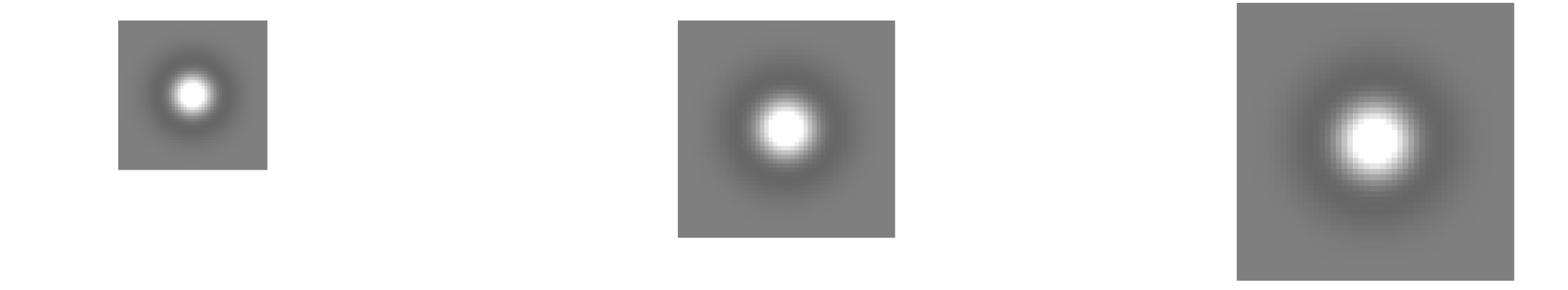




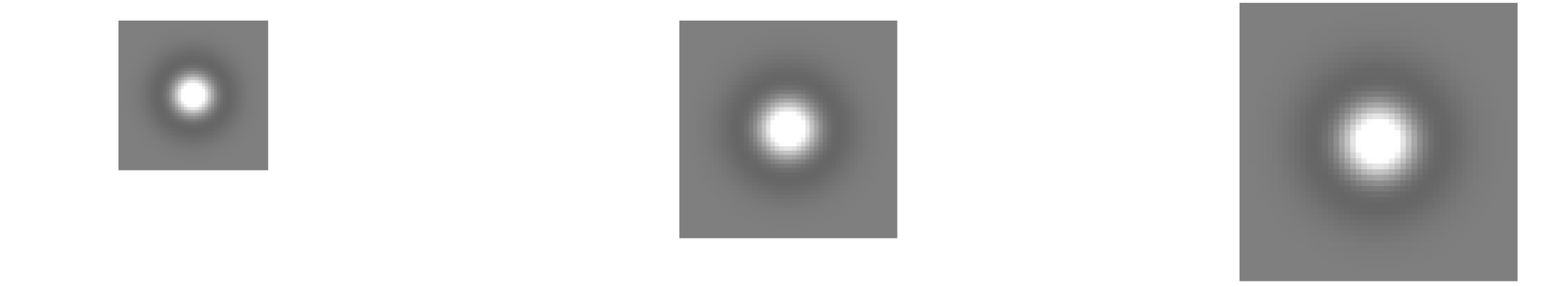




Detect Scale and Position



Detect Scale and Position

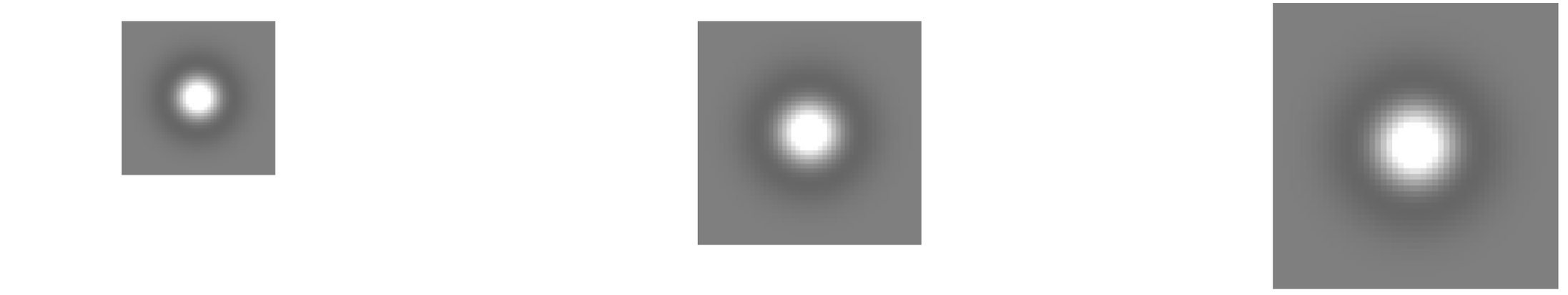


1	0	1	2	3	4	2
2	7	8	2	3	3	3
2	3	1	4	5	6	2
2	7	12	15	20	7	2
2	8	12	25	19	11	3
3	9	11	18	16	7	1
1	2	3	2	1	3	1

2	1	2	2	2	3	3
1	2	3	2	2	3	4
5	11	9	8	9	8	1
1	11	15	20	22	9	5
2	12	16	35	22	11	4
3	9	15	18	21	7	1
2	2	9	8	7	9	1

3	3	3	3	3	3	3
3	4	4	4	4	4	3
3	5	6	6	6	5	2
2	7	12	12	12	7	2
2	5	12	20	13	11	3
3	9	12	13	13	9	1
1	2	5	5	5	3	1

Detect Scale and Position



1	0	1	2	3	4	2
2	7	8	2	3	3	3
2	3	1	4	5	6	2
2	7	12	15	20	7	2
2	8	12	25	19	11	3
3	9	11	18	16	7	1
1	2	3	2	1	3	1

2	1	2	2	2	3	3
1	2	3	2	2	3	4
5	11	9	8	9	8	1
1	11	15	20	22	9	5
2	12	16	35	22	11	4
3	9	15	18	21	7	1
2	2	9	8	7	9	1

3	3	3	3	3	3	3
3	4	4	4	4	4	3
3	5	6	6	6	5	2
2	7	12	12	12	7	2
2	5	12	20	13	11	3
3	9	12	13	13	9	1
1	2	5	5	5	3	1

Non-maximum suppression in image **and scale**!

Sub-Pixel Refinement

2nd order curve fitting (1D):

$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

Sub-Pixel Refinement

2nd order curve fitting (1D):

$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

2nd order surface fitting (3D):

$$f(x, y, \sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Sub-Pixel Refinement

2nd order curve fitting (1D):

$$f(x) \approx f(0) + x \cdot f'(0) + \frac{x^2}{2} f''(0)$$

2nd order surface fitting (3D):

$$f(x, y, \sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Hessian matrix:

$$\mathtt{H}(0,0,0) = \begin{pmatrix} f''_{xx}(0,0,0) & f''_{xy}(0,0,0) & f''_{x\sigma}(0,0,0) \\ f''_{xy}(0,0,0) & f''_{yy}(0,0,0) & f''_{y\sigma}(0,0,0) \\ f''_{x\sigma}(0,0,0) & f''_{y\sigma}(0,0,0) & f''_{\sigma\sigma}(0,0,0) \end{pmatrix}$$

Sub-Pixel Refinement

2nd order surface fitting (3D):

$$f(x,y,\sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathrm{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Sub-Pixel Refinement

2nd order surface fitting (3D):

$$f(x,y,\sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathrm{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Set derivative to 0:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \nabla f(0,0,0) + \mathrm{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Sub-Pixel Refinement

2nd order surface fitting (3D):

$$f(x,y,\sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathrm{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Set derivative to 0:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \nabla f(0,0,0) + \mathrm{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Solve small linear system:

$$\begin{pmatrix} x \\ y \\ \sigma \end{pmatrix} = -\mathrm{H}(0,0,0)^{-1} \nabla f(0,0,0)$$

Sub-Pixel Refinement

2nd order surface fitting (3D):

$$f(x,y,\sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Set derivative to 0:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \nabla f(0,0,0) + \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

offset from pixel

Solve small linear system:

$$\begin{pmatrix} x \\ y \\ \sigma \end{pmatrix} = -\mathtt{H}(0,0,0)^{-1} \nabla f(0,0,0)$$

Sub-Pixel Refinement

2nd order surface fitting (3D):

$$f(x,y,\sigma) \approx f(0,0,0) + \begin{pmatrix} x & y & \sigma \end{pmatrix} \cdot \nabla f(0,0,0) + \frac{1}{2} \begin{pmatrix} x & y & \sigma \end{pmatrix} \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

Set derivative to 0:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \nabla f(0,0,0) + \mathtt{H}(0,0,0) \begin{pmatrix} x \\ y \\ \sigma \end{pmatrix}$$

offset from pixel

Solve small linear system:

$$\begin{pmatrix} x \\ y \\ \sigma \end{pmatrix} = -\mathtt{H}(0,0,0)^{-1} \nabla f(0,0,0)$$

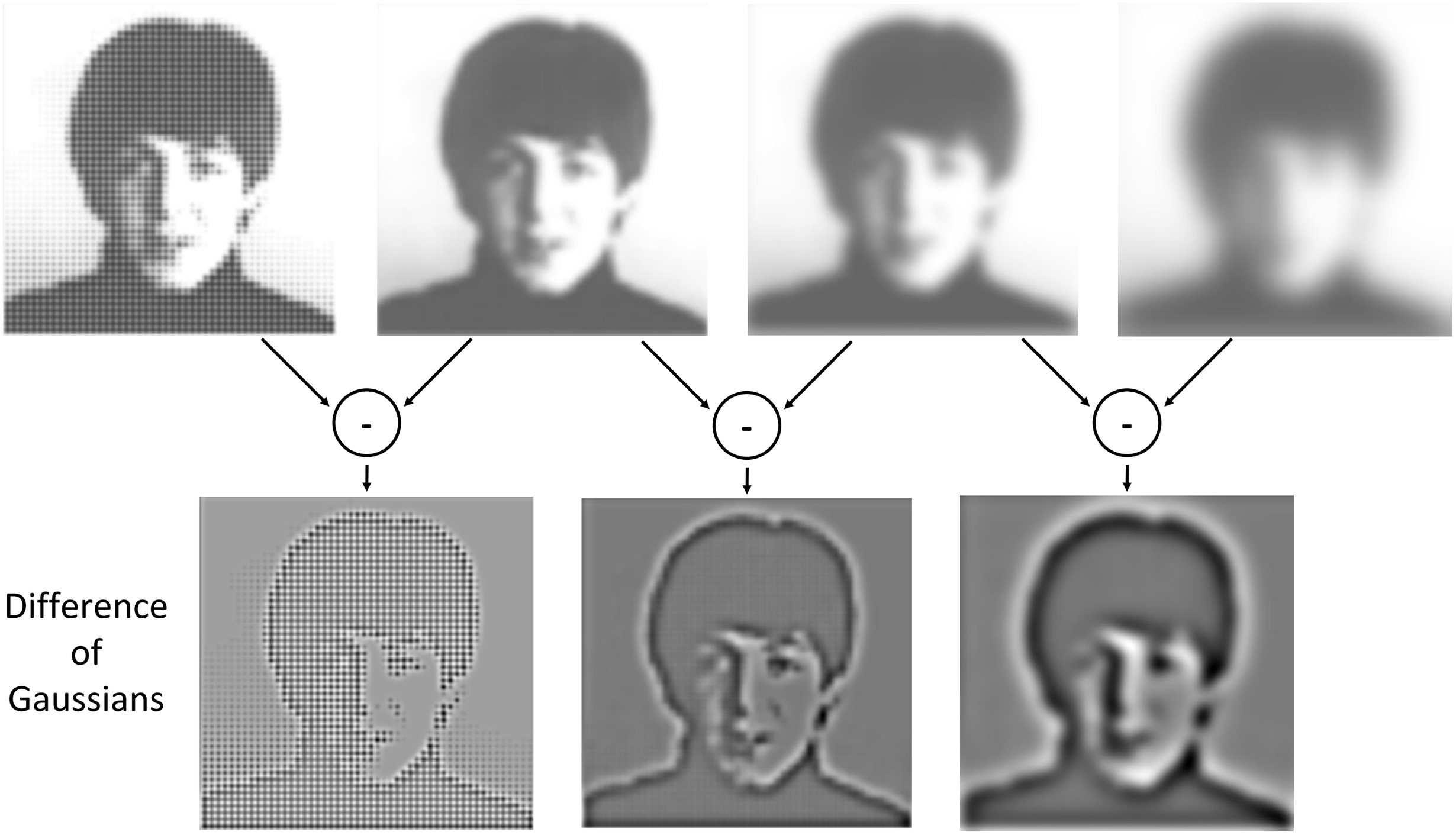
Potentially repeat for a few iterations (centered on current estimate!)

Efficient implementation

Gaussian filtering

Efficient implementation

Gaussian filtering



Difference of Gaussians

Efficient implementation

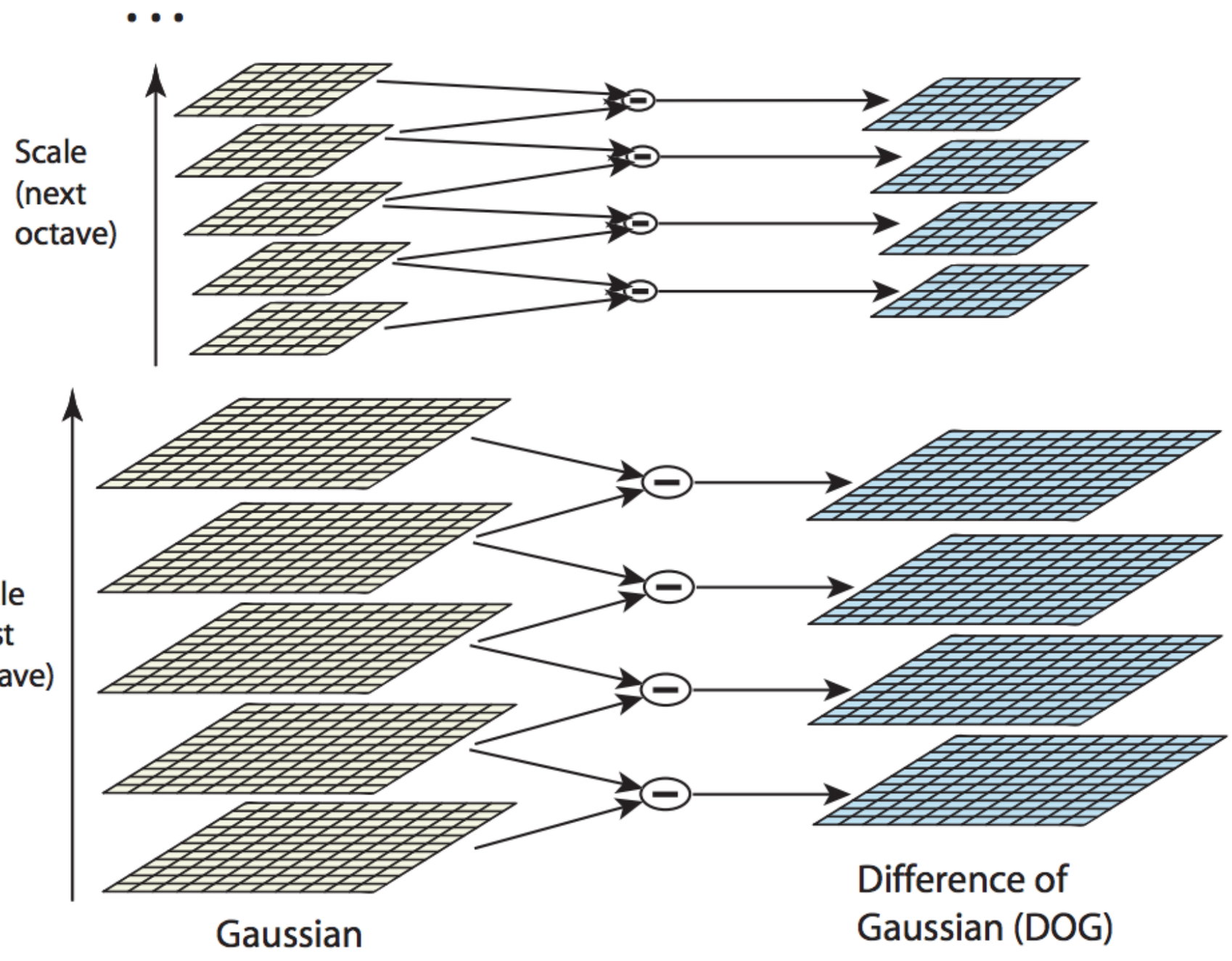
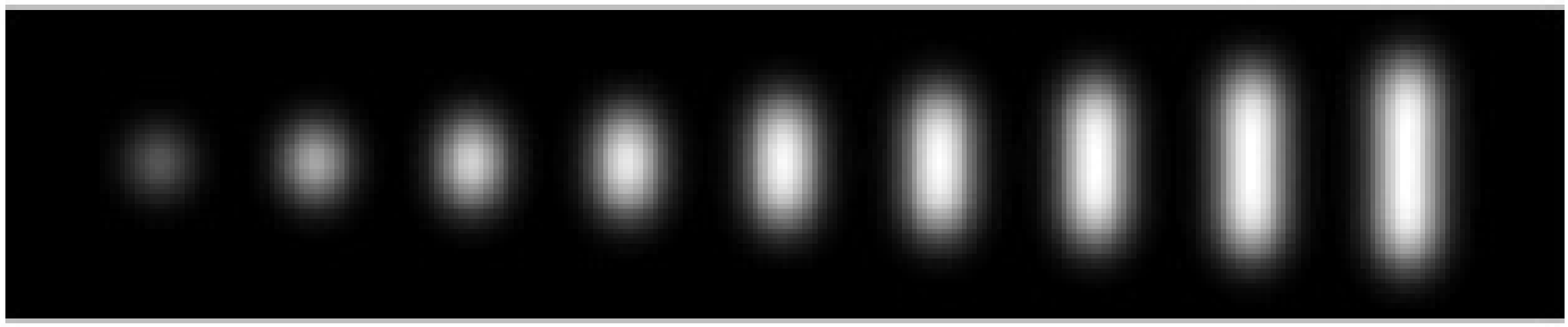


illustration from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Avoiding Edges

Avoiding Edges

Avoiding Edges

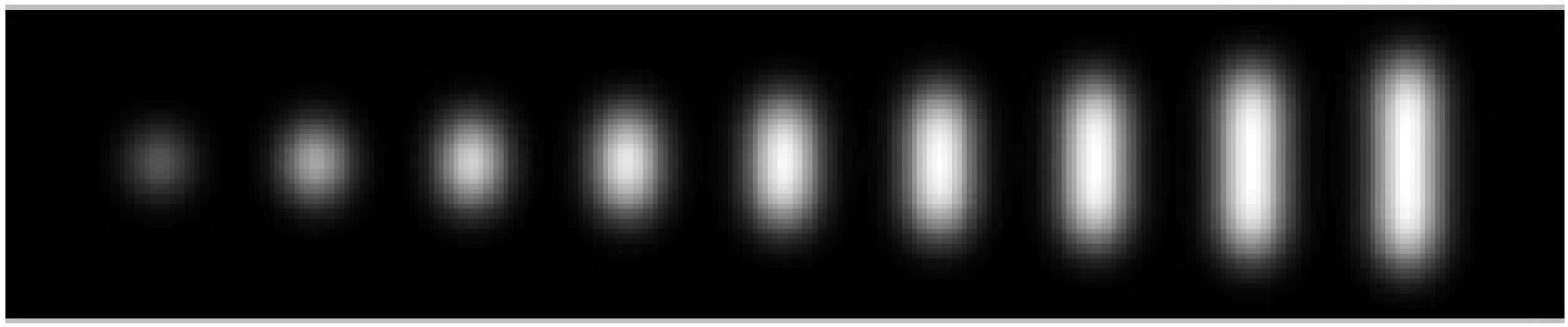


well-localized blob

edge

image source: vlfeat.org

Avoiding Edges



well-localized blob

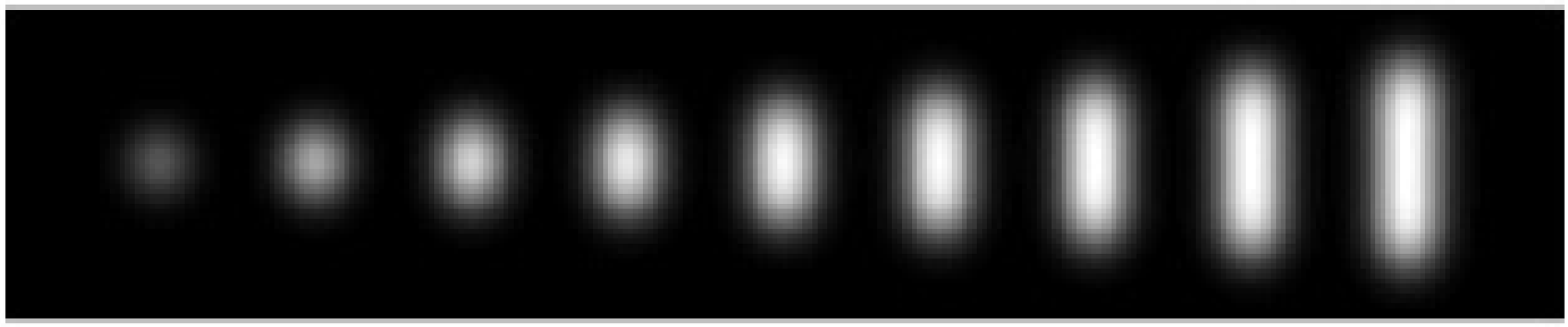
edge

Consider 2D Hessian around detected keypoint (after refinement):

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

image source: vlfeat.org

Avoiding Edges



well-localized blob

edge

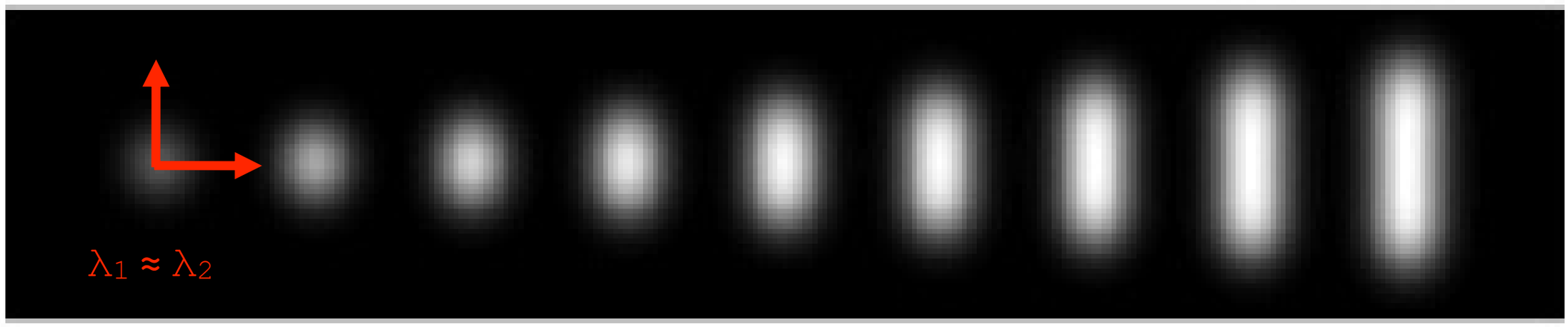
Consider 2D Hessian around detected keypoint (after refinement):

$$\mathtt{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$$

Eigenvalues $\lambda_1 \geq \lambda_2$ of $\mathtt{H}$ proportional to *principal curvature*

image source: vlfeat.org

Avoiding Edges



well-localized blob

edge

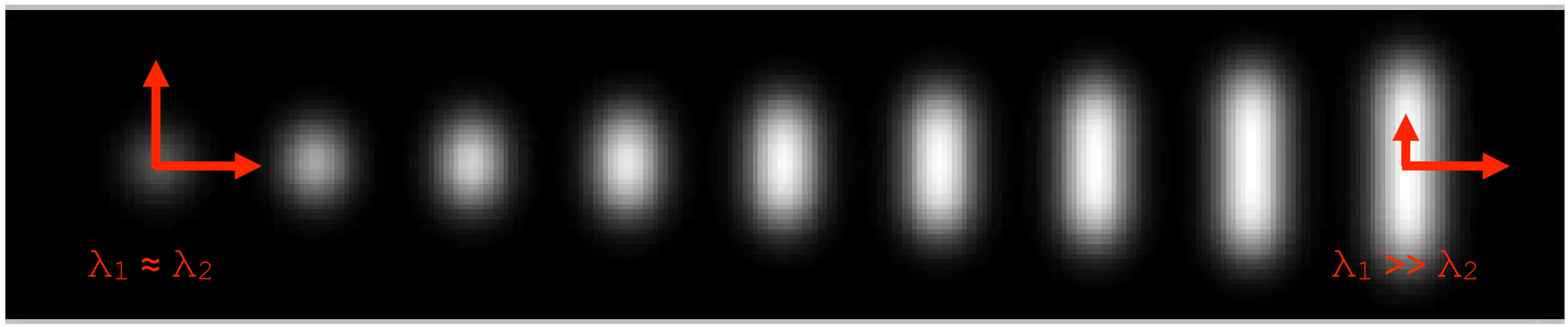
Consider 2D Hessian around detected keypoint (after refinement):

$$\mathtt{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$$

Eigenvalues $\lambda_1 \geq \lambda_2$ of $\mathtt{H}$ proportional to *principal curvature*

image source: vlfeat.org

Avoiding Edges



well-localized blob

edge

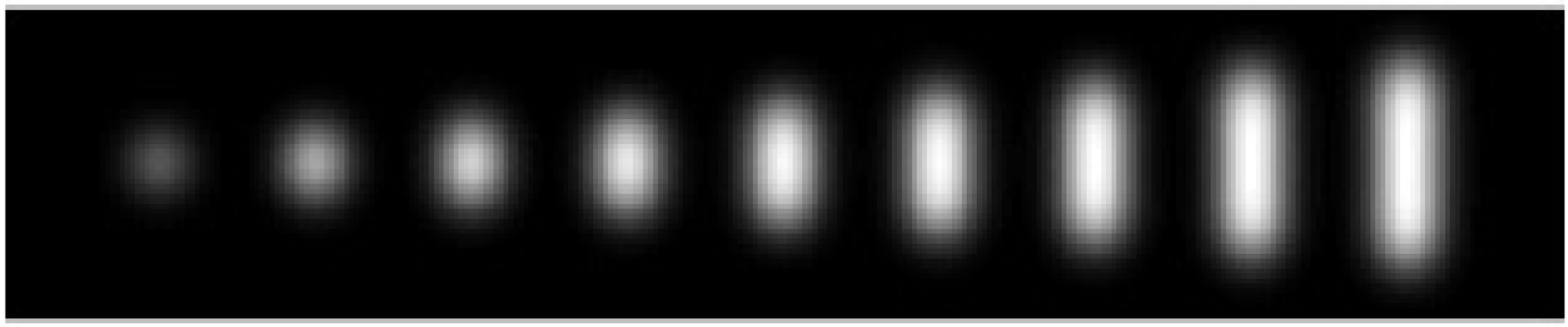
Consider 2D Hessian around detected keypoint (after refinement):

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

Eigenvalues $\lambda_1 \geq \lambda_2$ of H proportional to *principal curvature*

image source: vlfeat.org

Avoiding Edges



well-localized blob

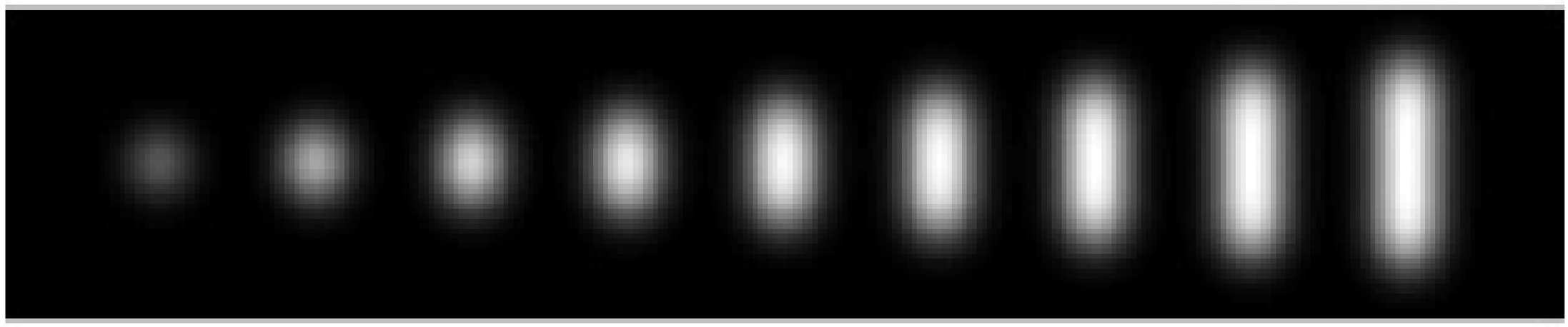
edge

Detecting edges: Look at ratio of Eigenvalues ($\lambda_1 \geq \lambda_2$)

$$\frac{\lambda_1}{\lambda_2} = r \qquad\qquad \frac{(r+1)^2}{r} < \tau$$

image source: vlfeat.org

Avoiding Edges



well-localized blob

edge

Detecting edges: Look at ratio of Eigenvalues ($\lambda_1 \geq \lambda_2$)

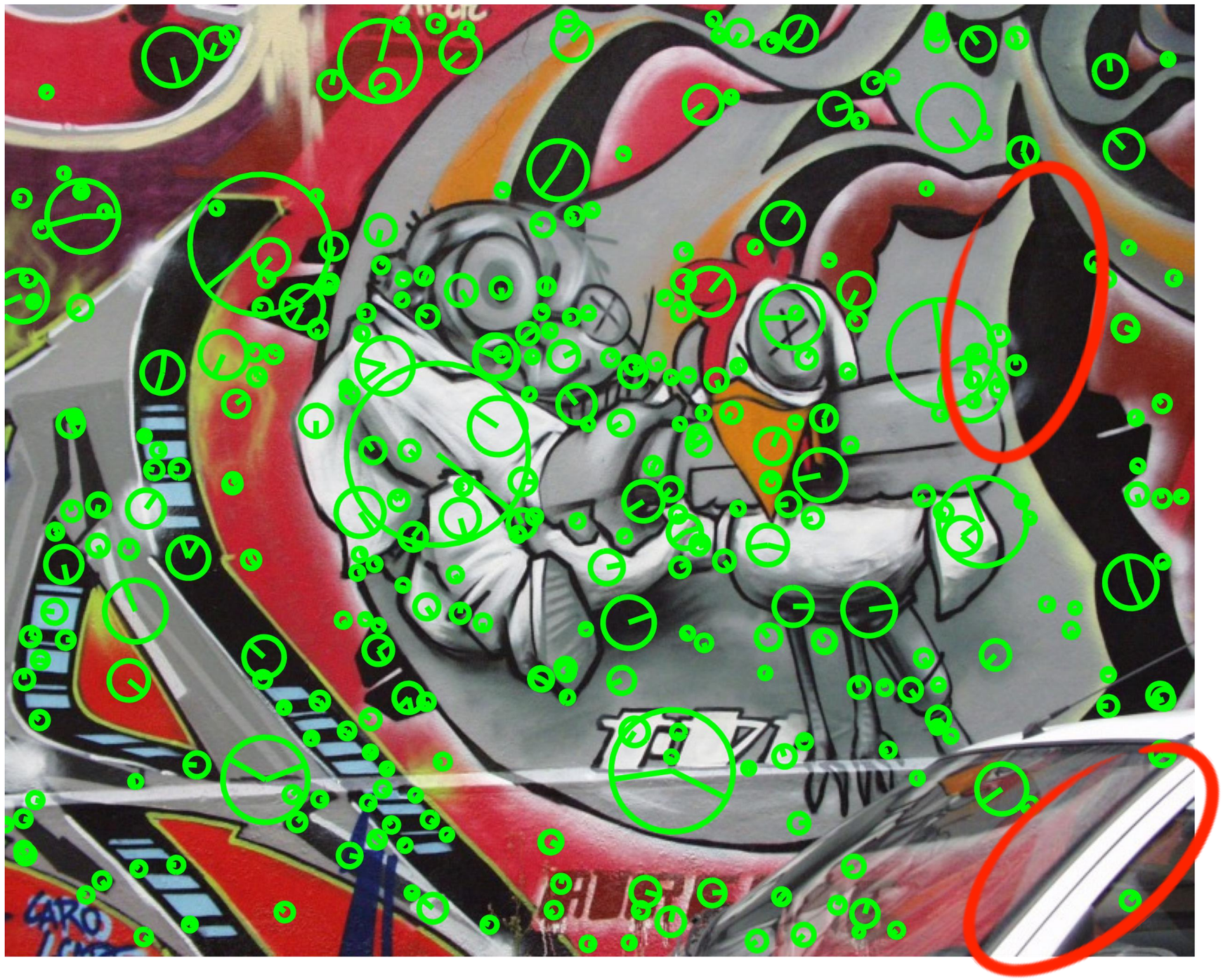
$$\frac{\lambda_1}{\lambda_2} = r \qquad\qquad \frac{(r+1)^2}{r} < \tau$$

No need to explicitly compute Eigenvalues (see Lowe's paper)

image source: vlfeat.org

Avoiding Edges

Avoiding Edges

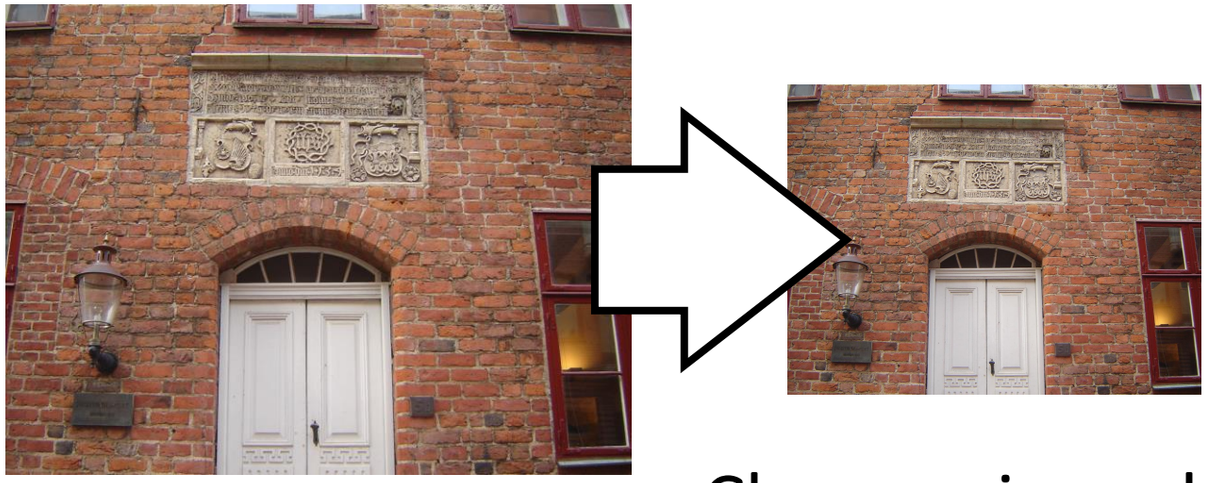


Invariances?

in-plane rotation

Changes in brightness

Invariances?

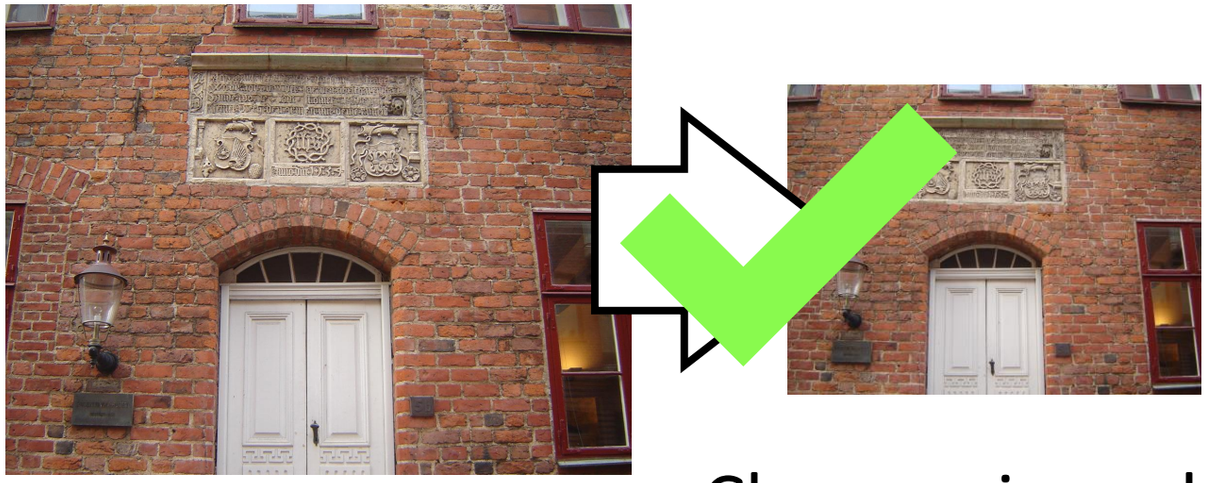


Changes in scale

in-plane rotation

Changes in brightness

Invariances?

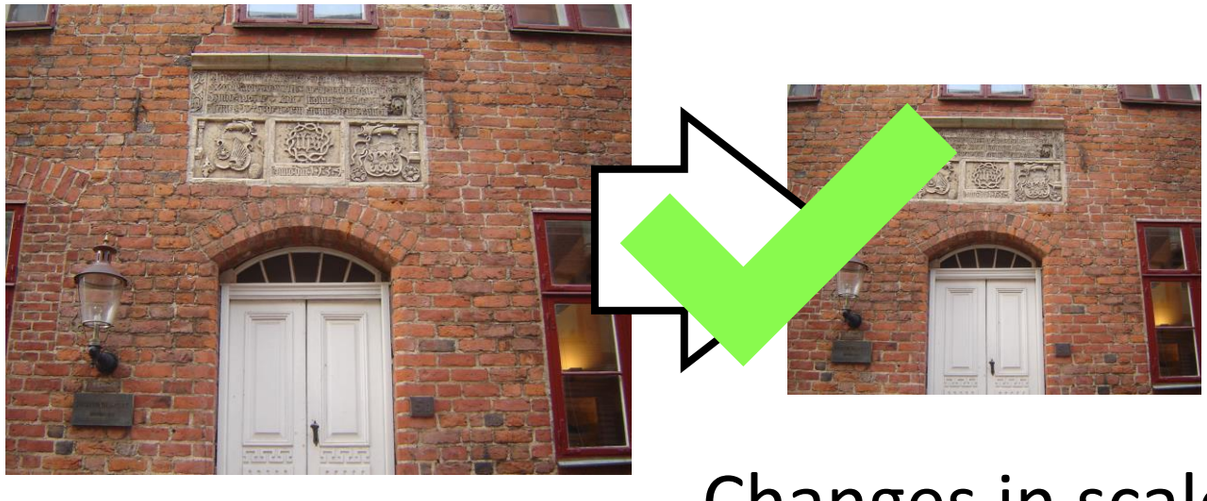


Changes in scale

in-plane rotation

Changes in brightness

Invariances?



Changes in scale

in-plane rotation

Changes in brightness

Changes in viewpoint

Invariances?

Changes in scale

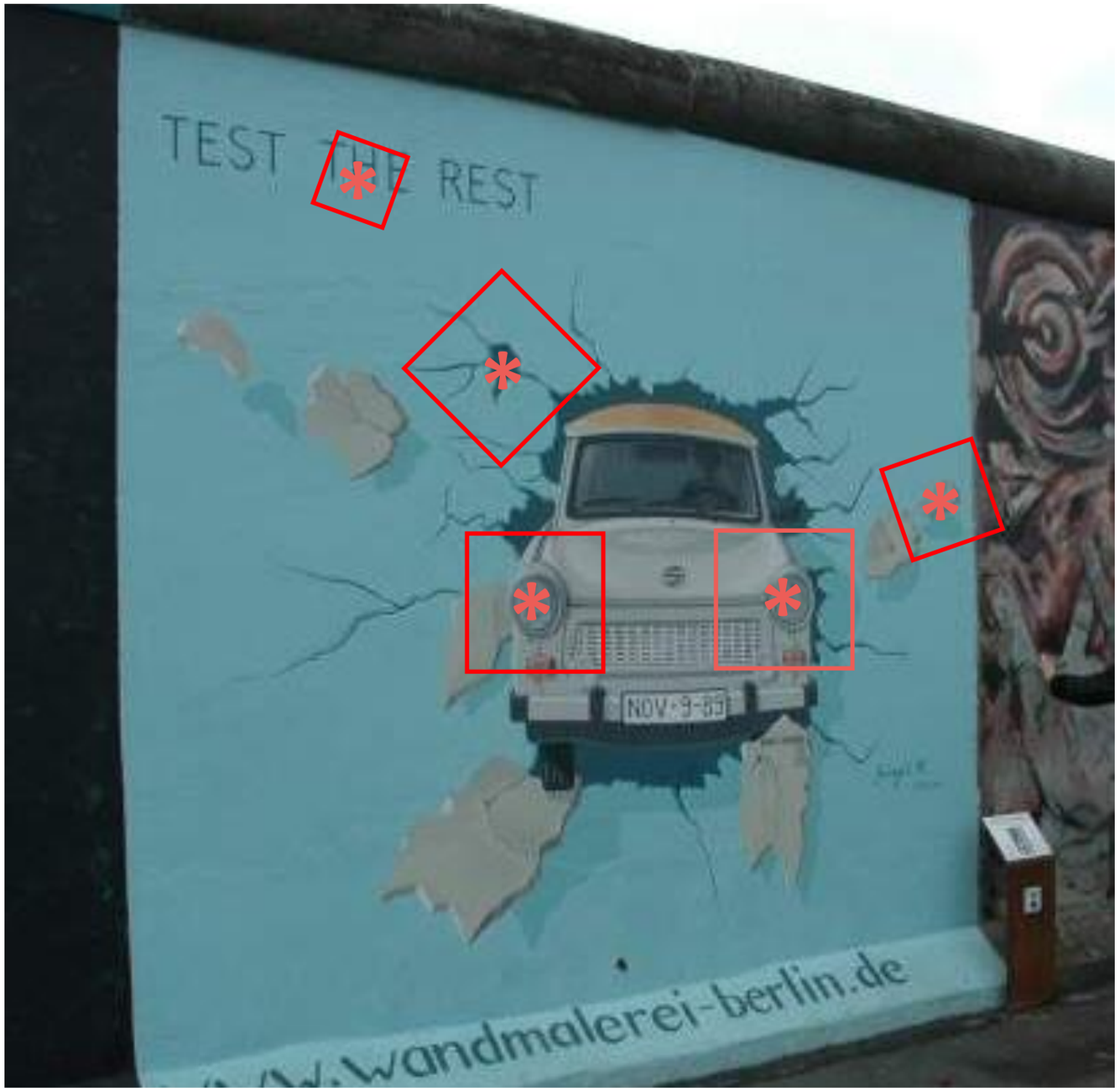
in-plane rotation

Changes in brightness

Not invariant, but reasonably robust

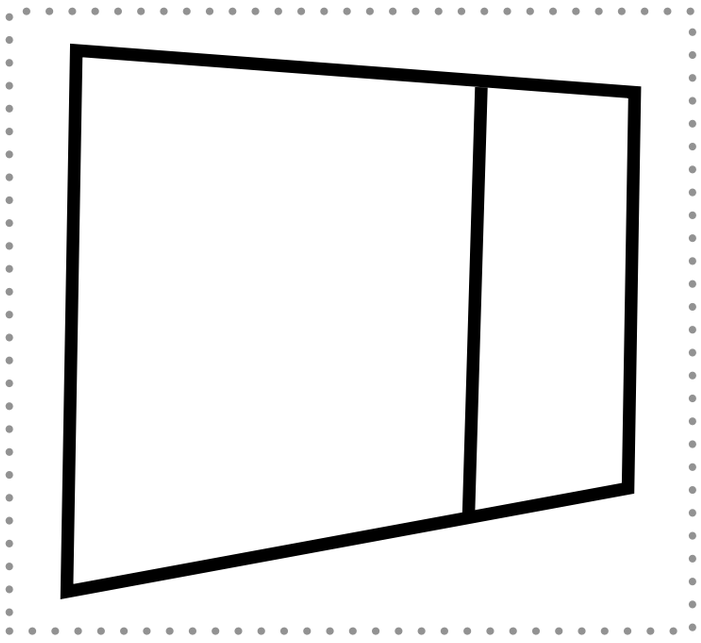
Changes in viewpoint

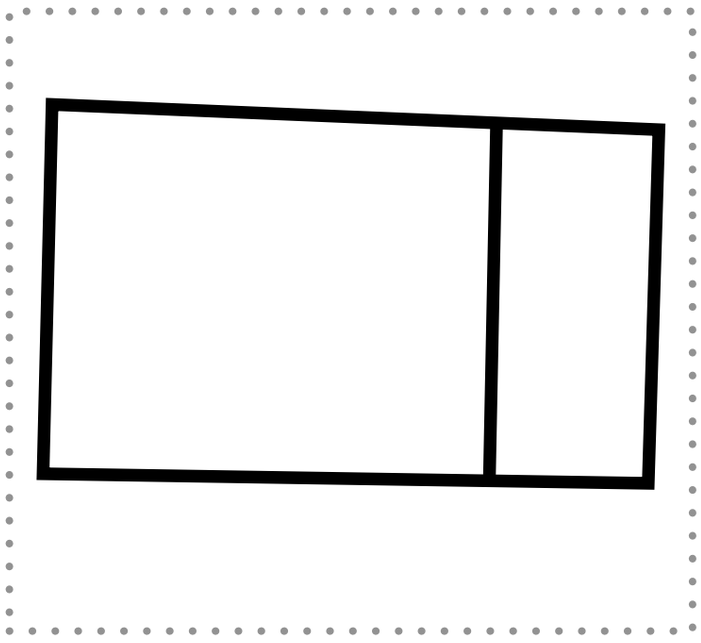
Scale-Invariant Feature Transform (SIFT) Features

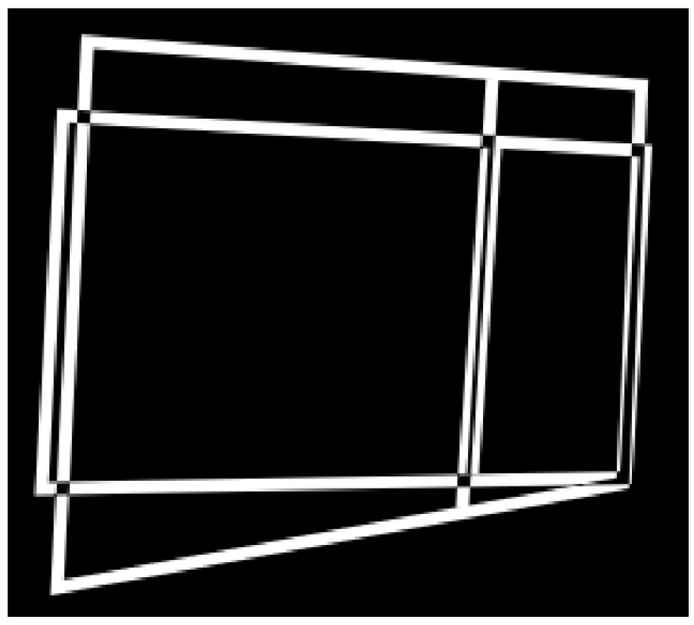


- Find a set of *interest* points
- Extract scaled & oriented patch around interest points
- **Compute SIFT descriptor for each patch**

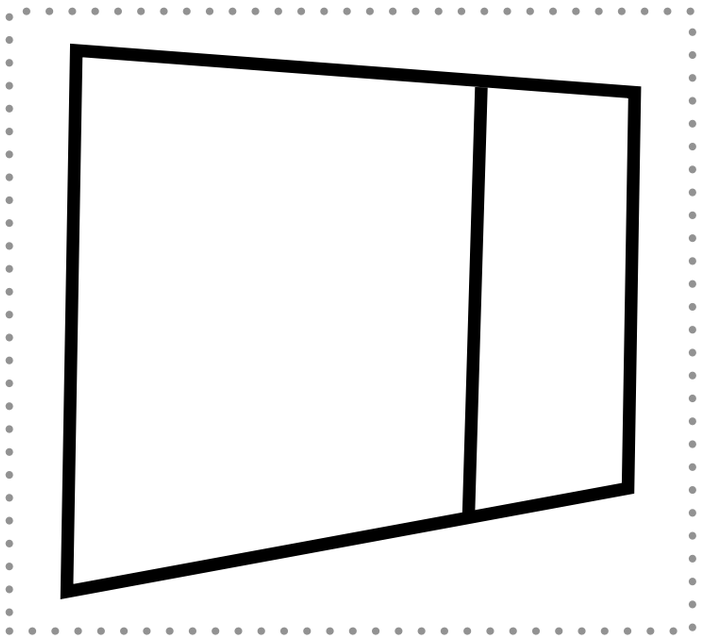
Perspective Distortion

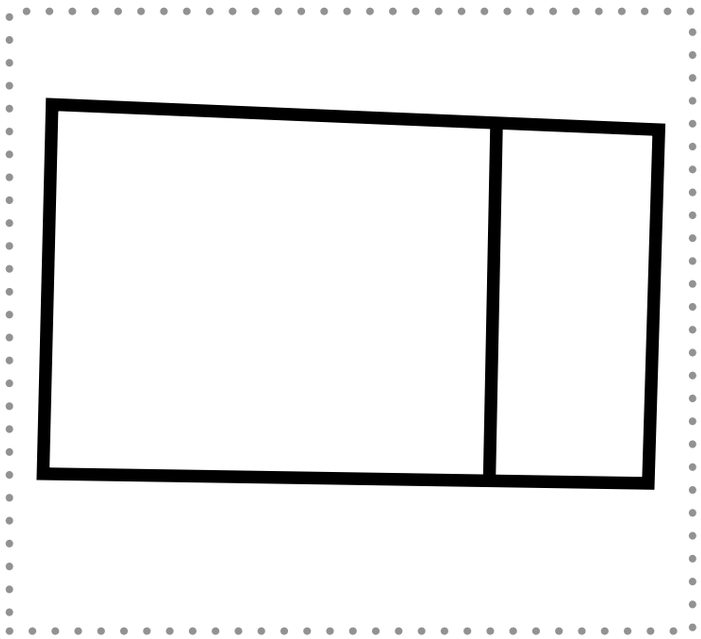


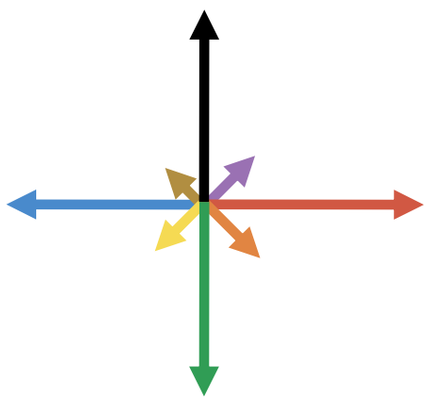


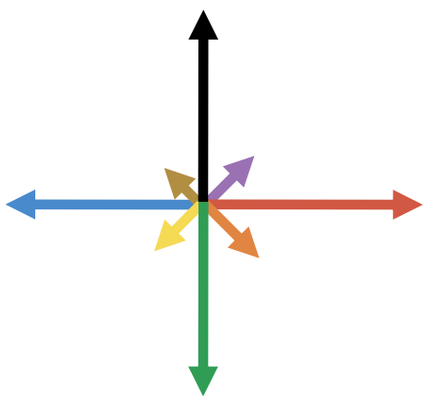


Perspective Distortion

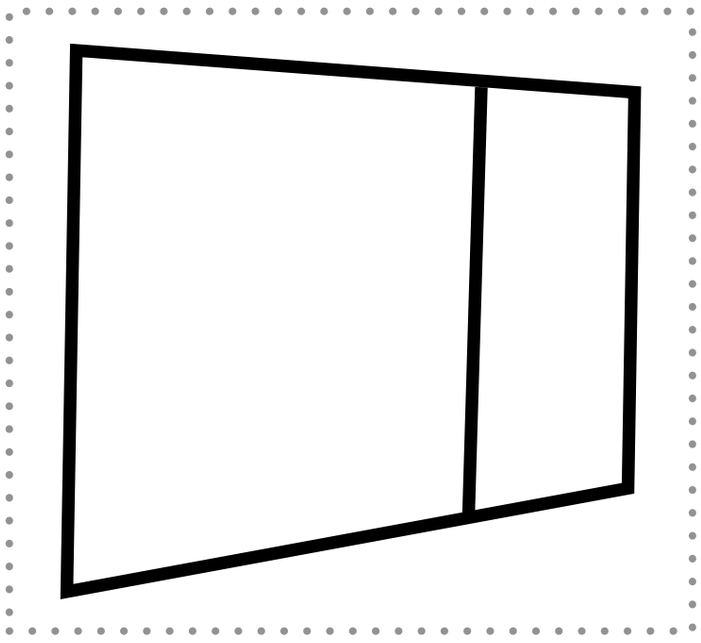


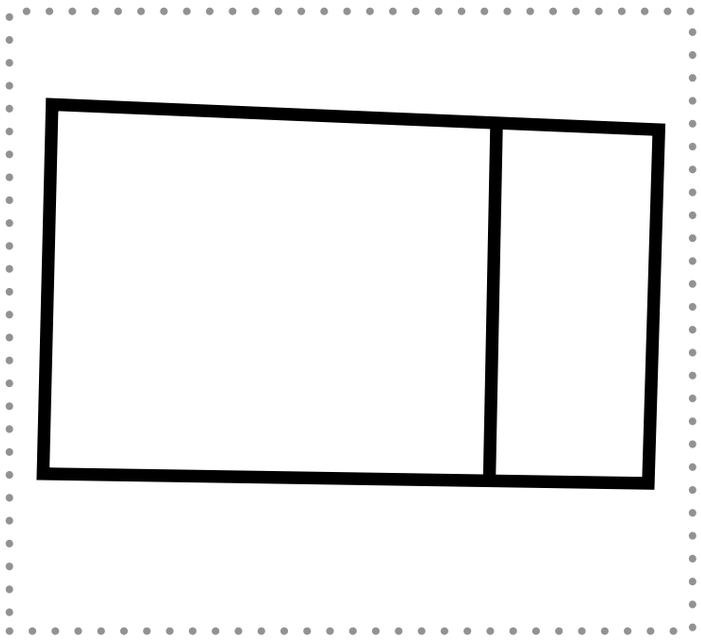


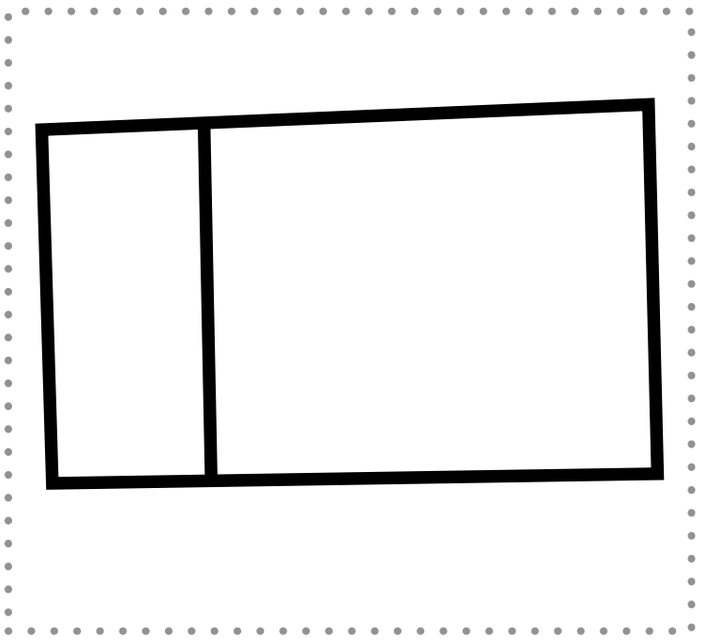


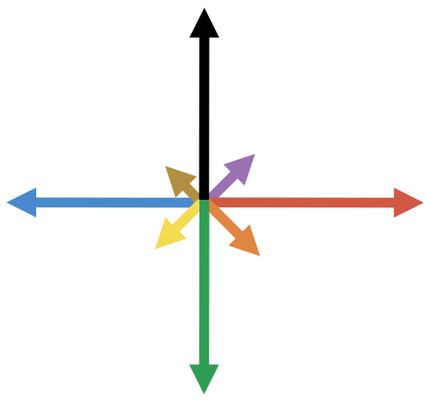


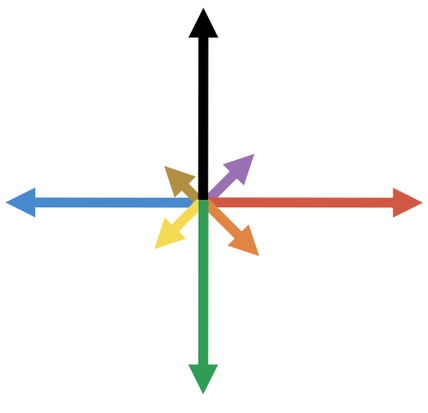
Perspective Distortion

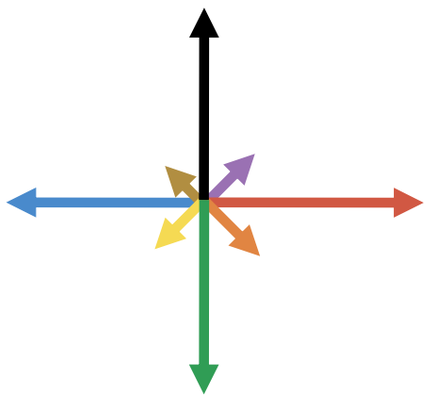




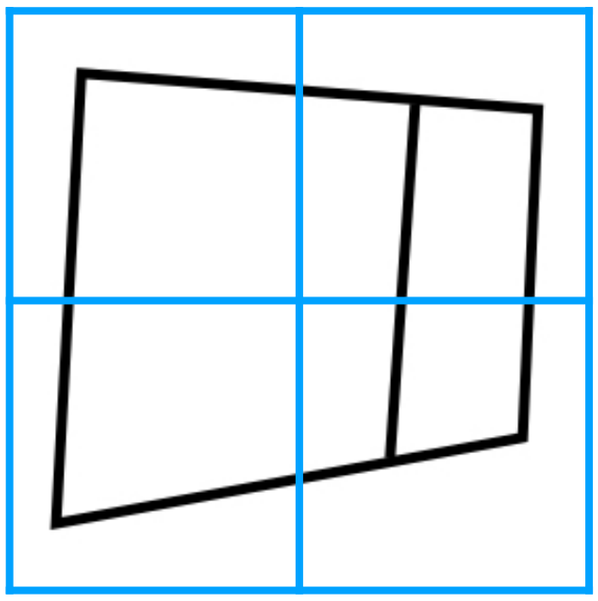






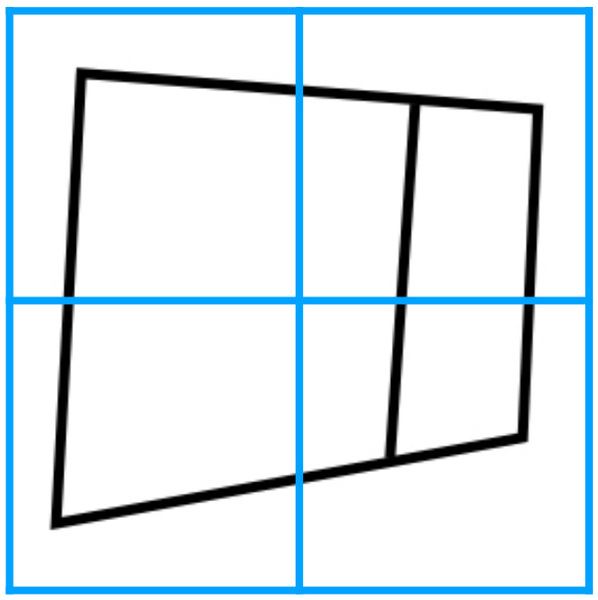


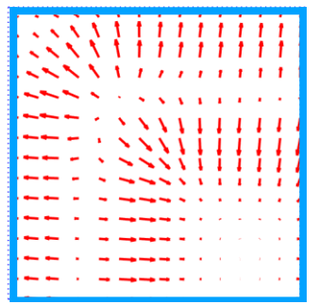
SIFT Descriptor (Idea)



Split patch in 2x2 regions
(SIFT actually uses 4x4 regions)

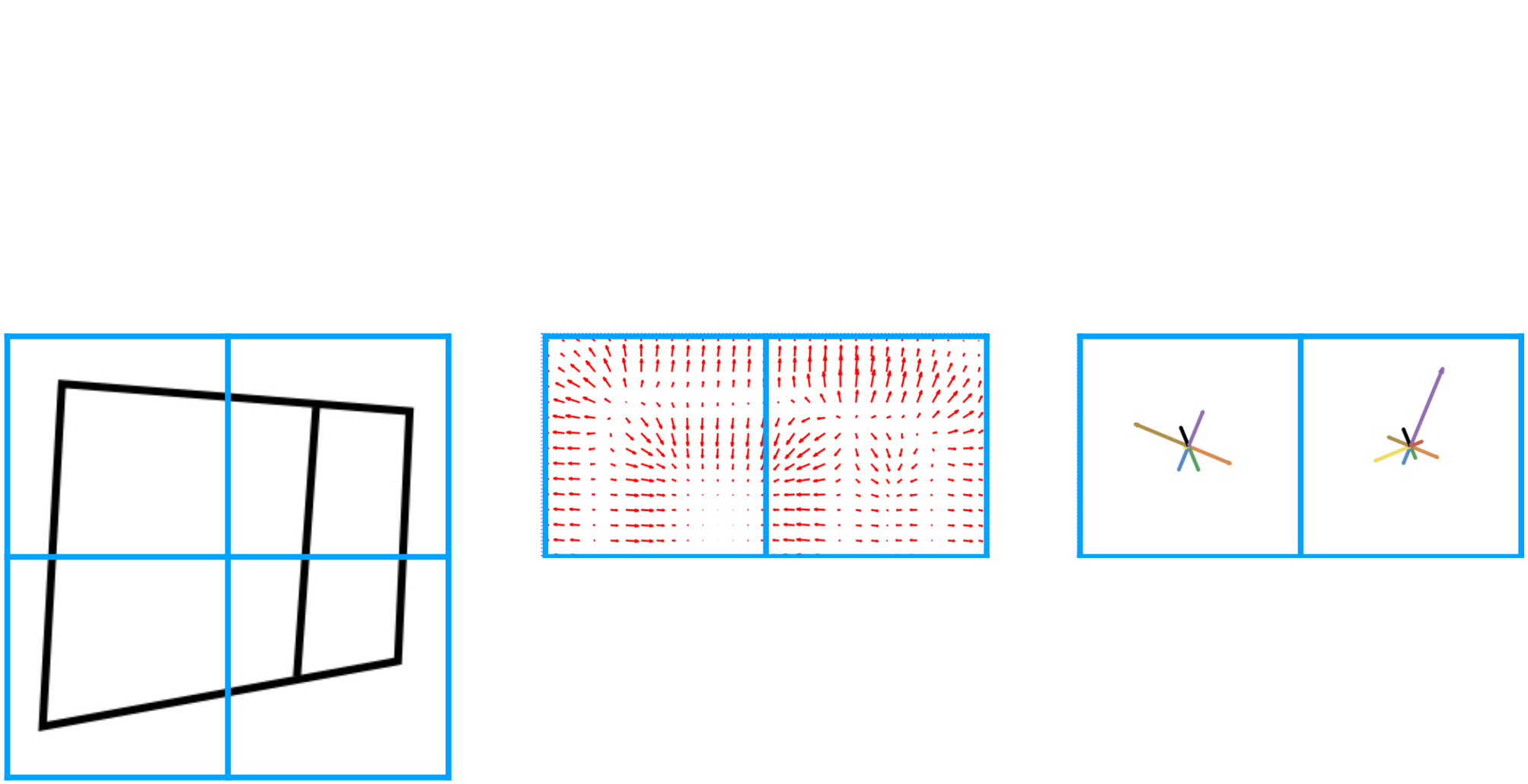
SIFT Descriptor (Idea)





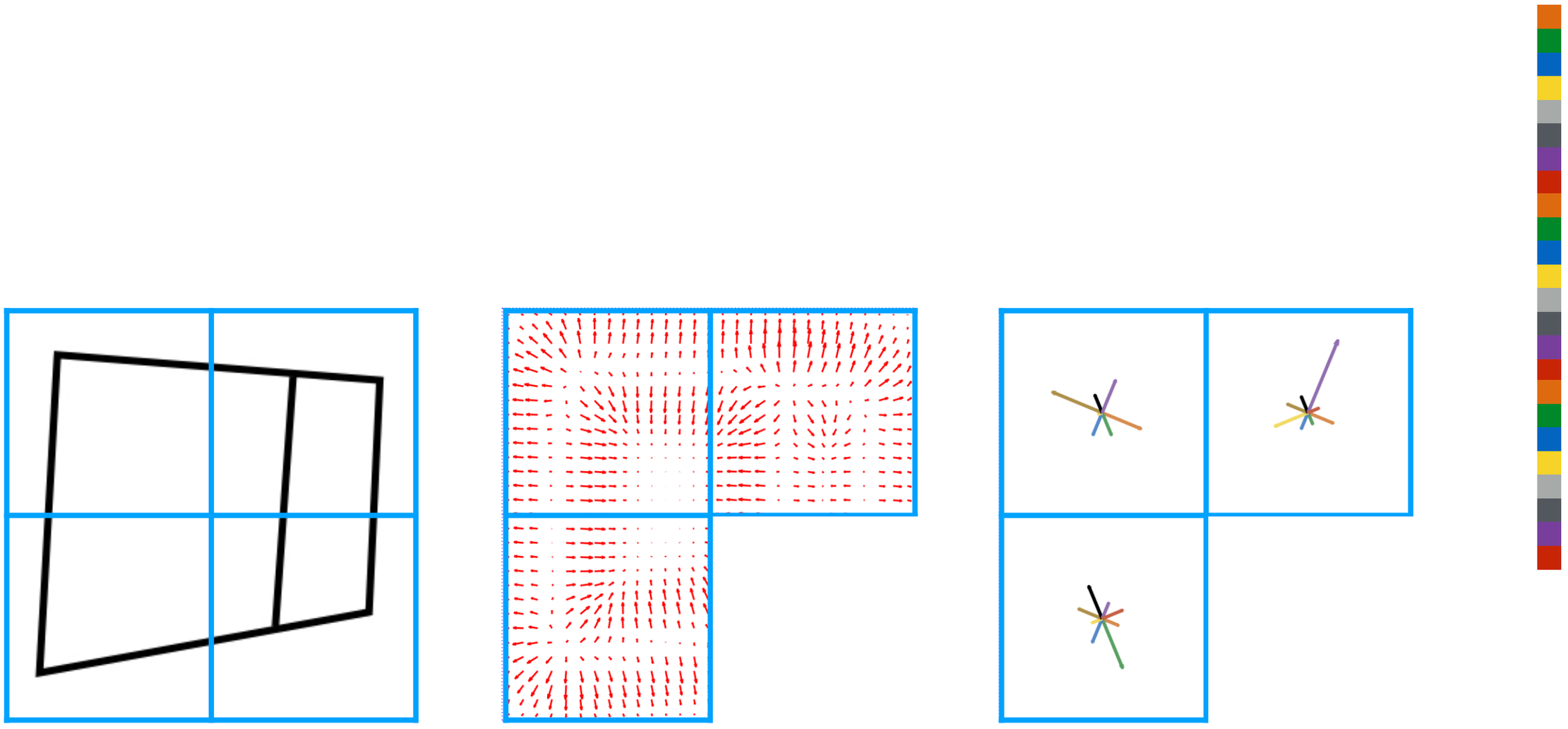
Compute a 8-bin gradient histogram for each region

SIFT Descriptor (Idea)



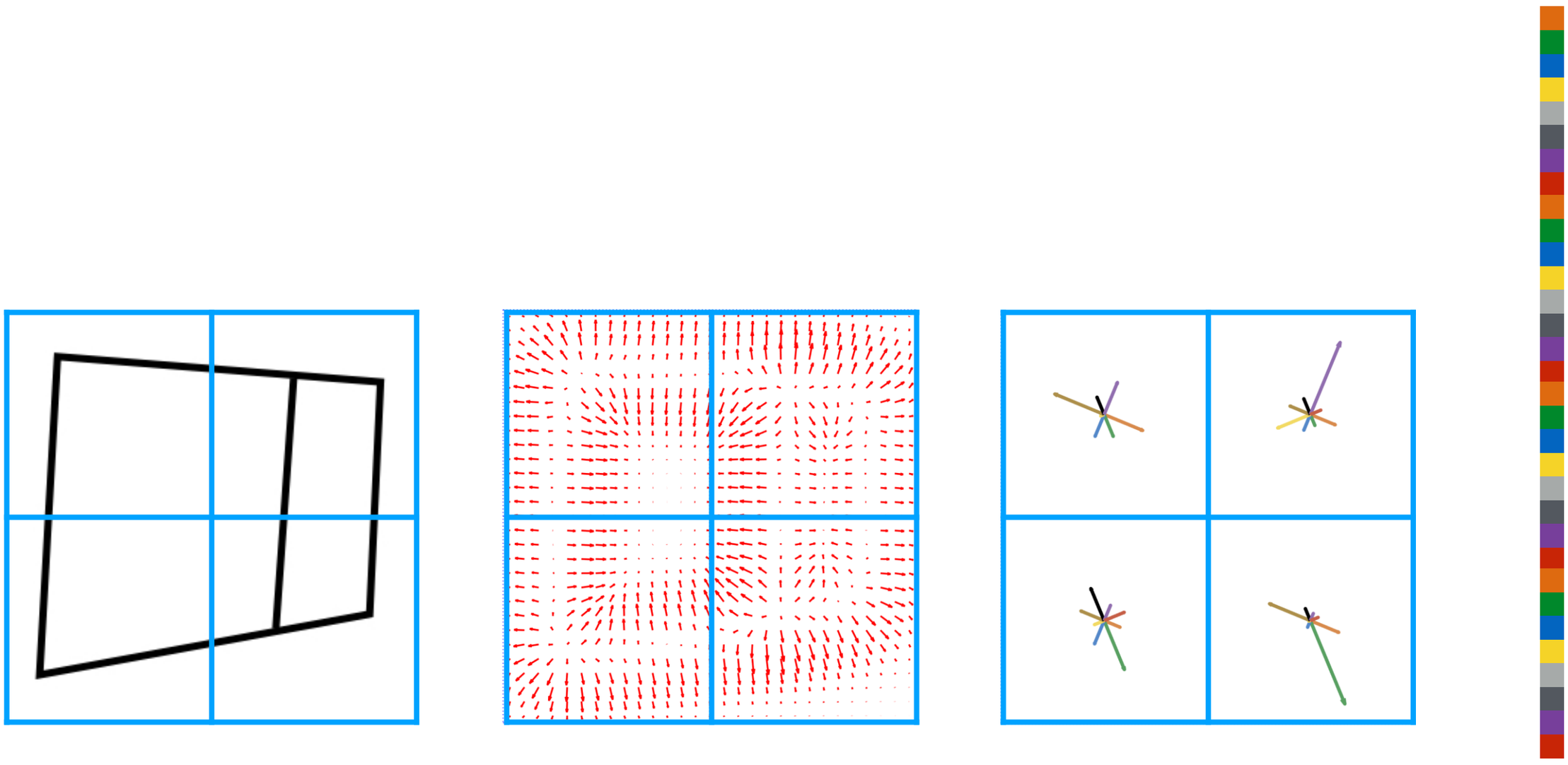
Compute a 8-bin gradient histogram for each region

SIFT Descriptor (Idea)

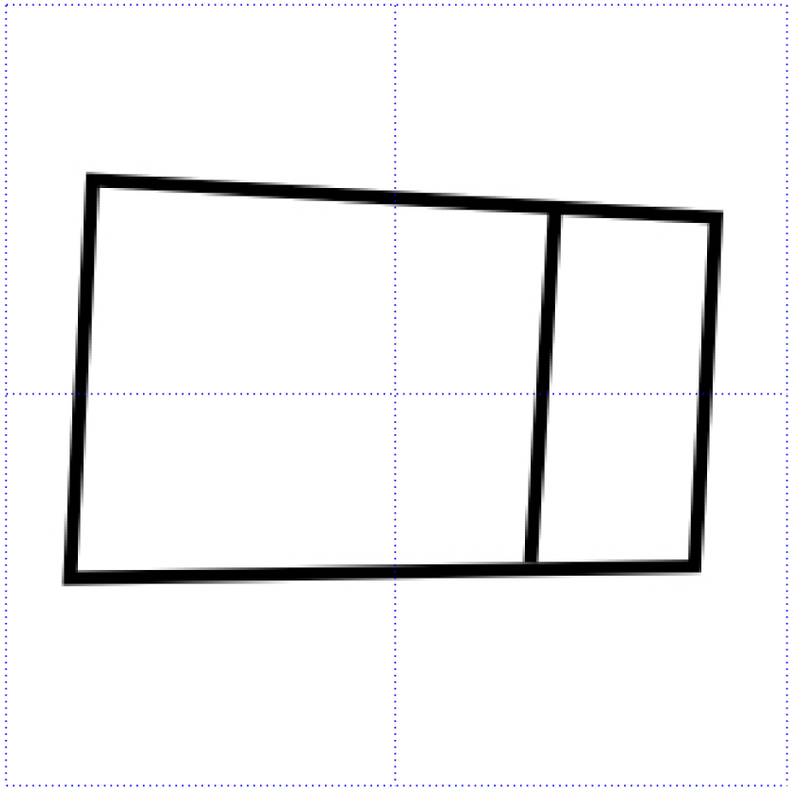


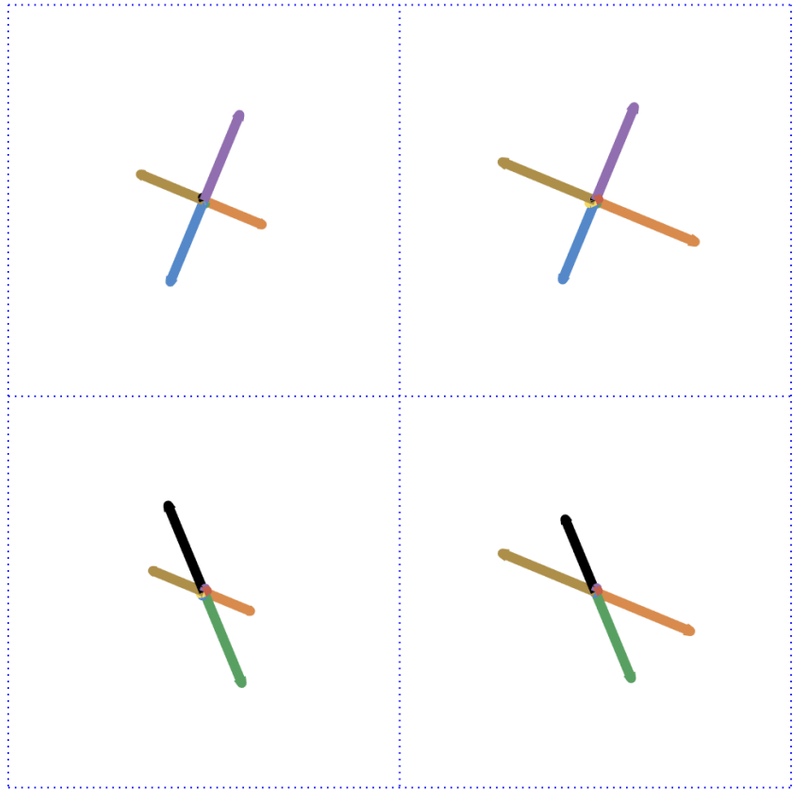
Compute a 8-bin gradient histogram for each region

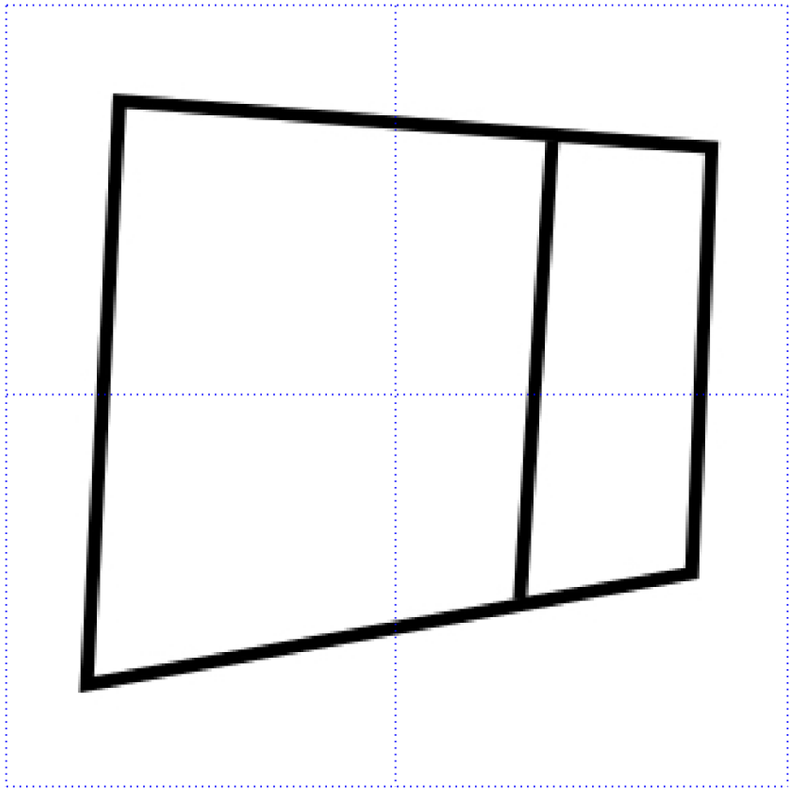
SIFT Descriptor (Idea)

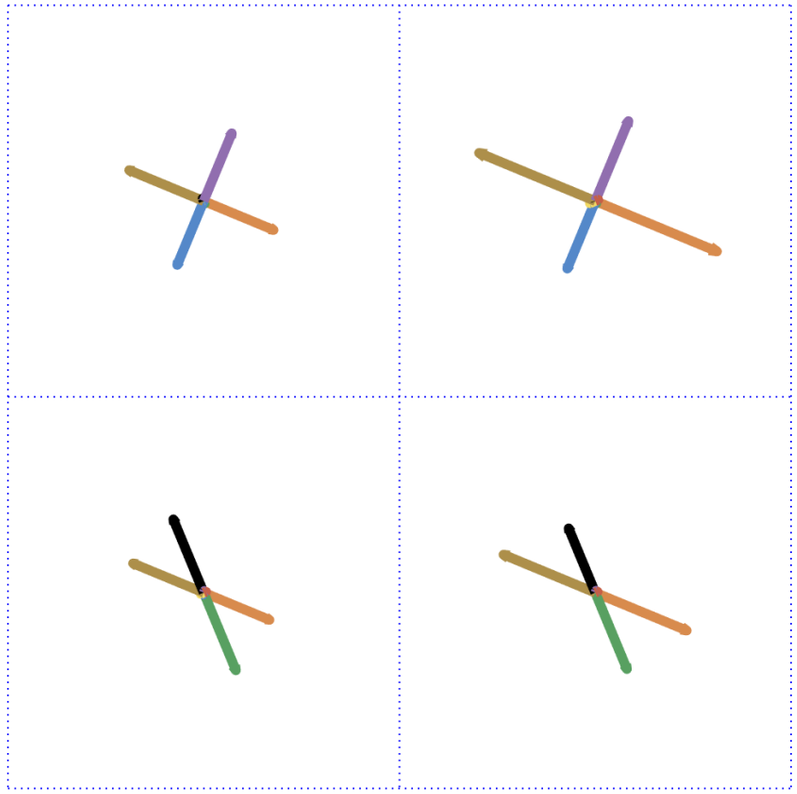


L_2 Normalize the descriptor







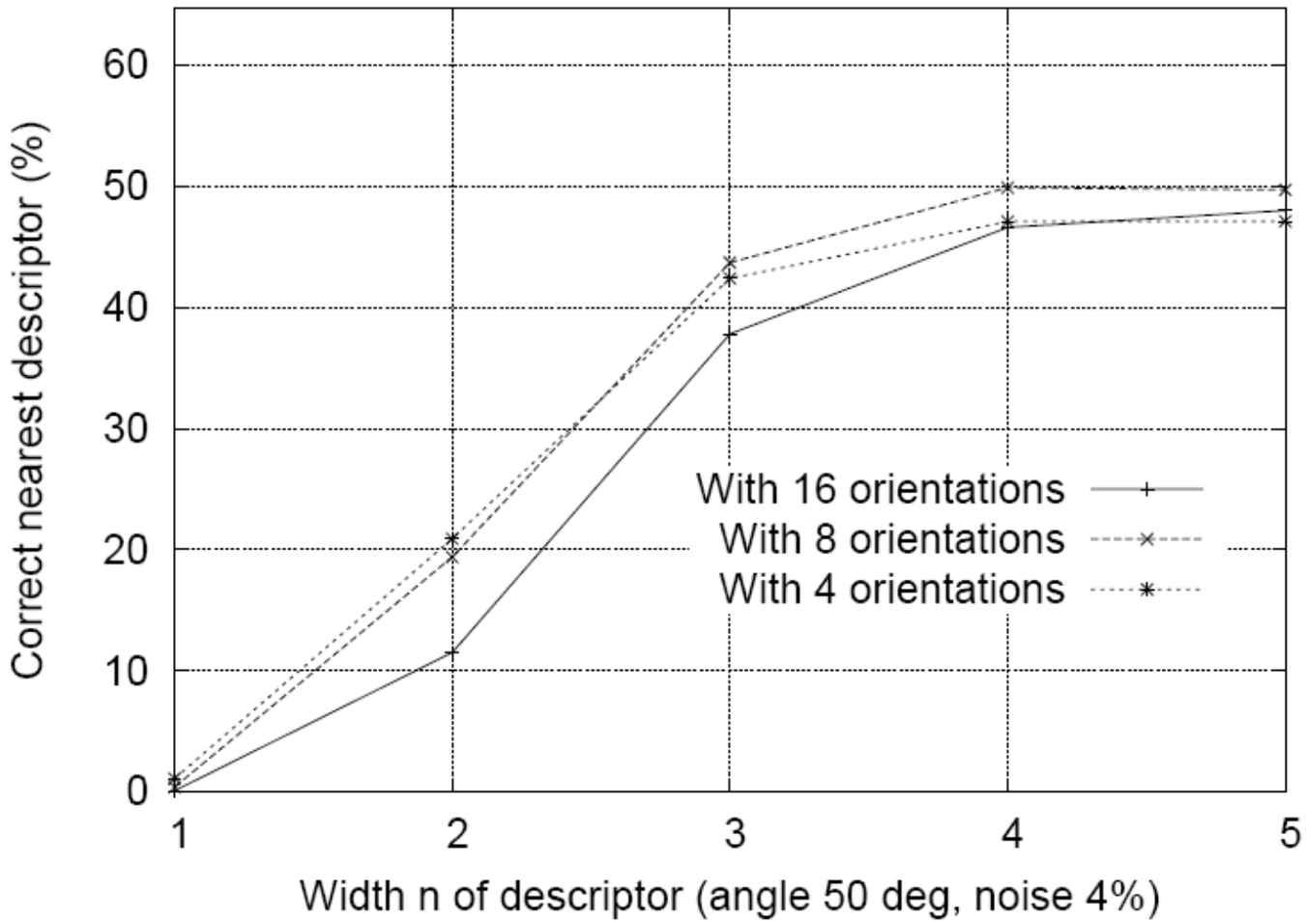


SIFT Descriptor (Idea)

Why 4x4 regions and 8 bins per histogram?

SIFT Descriptor (Idea)

Why 4x4 regions and 8 bins per histogram?



Careful parameter tuning

illustrations from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Invariances?

Changes in brightness

Invariances?

- Additive changes in brightness

Changes in brightness

Invariances?

- Additive changes in brightness
  - Do not affect the gradient

Changes in brightness

Invariances?

Additive changes in brightness

- Do not affect the gradient

Changes in brightness

Invariances?

Additive changes in brightness

- Do not affect the gradient

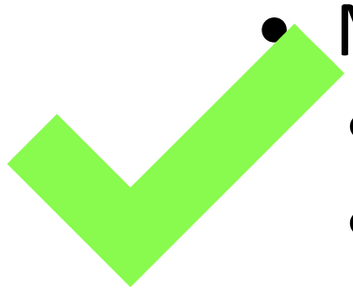
- Multiplicative changes in brightness
  - Affect gradient magnitude but not orientation

Changes in brightness

Invariances?

Additive changes in brightness

- Do not affect the gradient



- Multiplicative changes in brightness
  - Affect gradient magnitude but not orientation
  - L2-normalize the descriptor

Changes in brightness

Invariances?

- Additive changes in brightness
  - Do not affect the gradient
- Multiplicative changes in brightness
  - Affect gradient magnitude but not orientation
  - L2-normalize the descriptor
- Robustness against non-uniform changes
  - L2-normalize
  - Clamp every entry >0.2 to 0.2
  - Re-normalize
  - Intuition: Gradient orientations more important than magnitudes

Changes in brightness

Invariances?

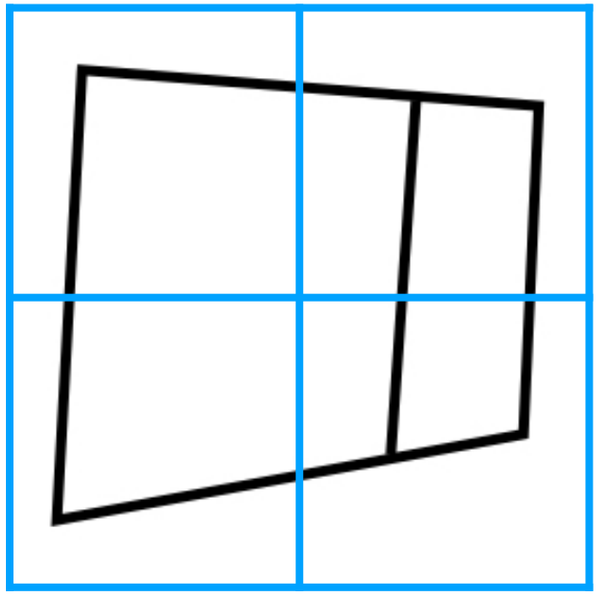
Changes in brightness

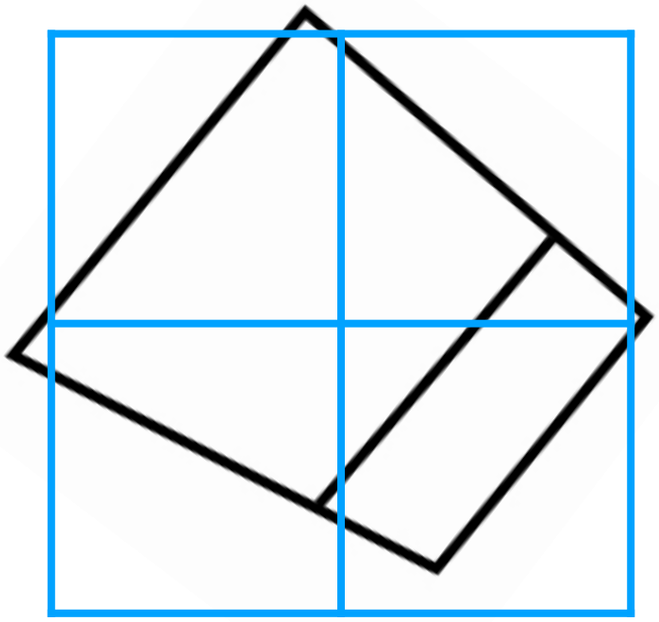
Invariances?

in-plane rotation

Changes in brightness

Invariances?

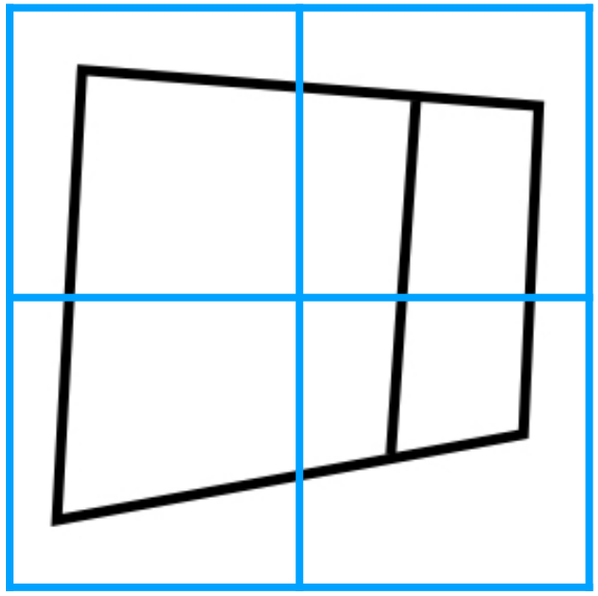


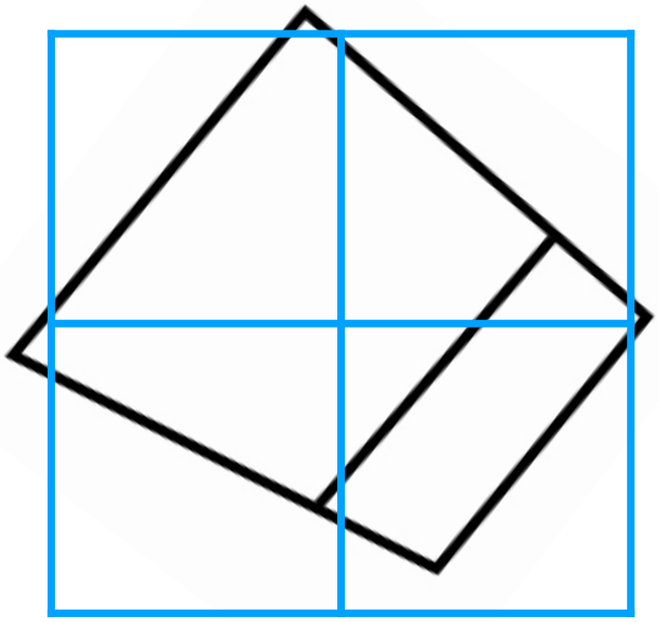


in-plane rotation

Changes in brightness

Invariances?

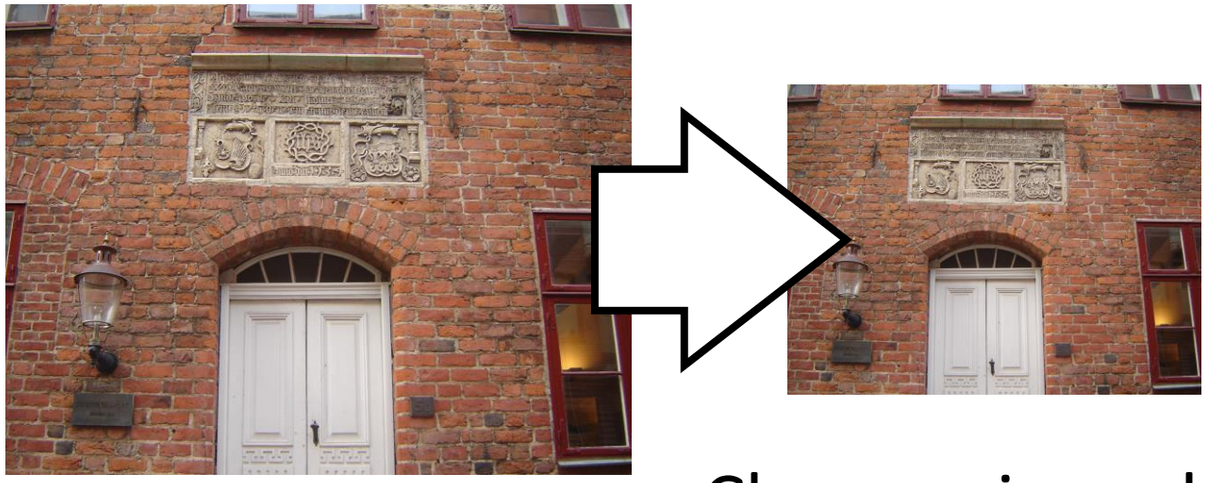




in-plane rotation

Changes in brightness

Invariances?

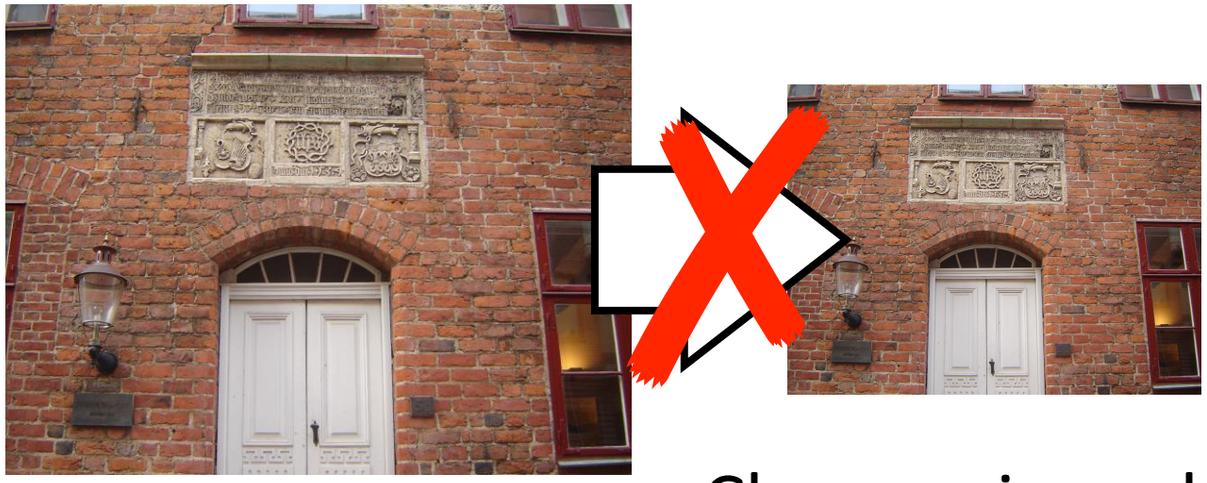


Changes in scale

in-plane rotation

Changes in brightness

Invariances?

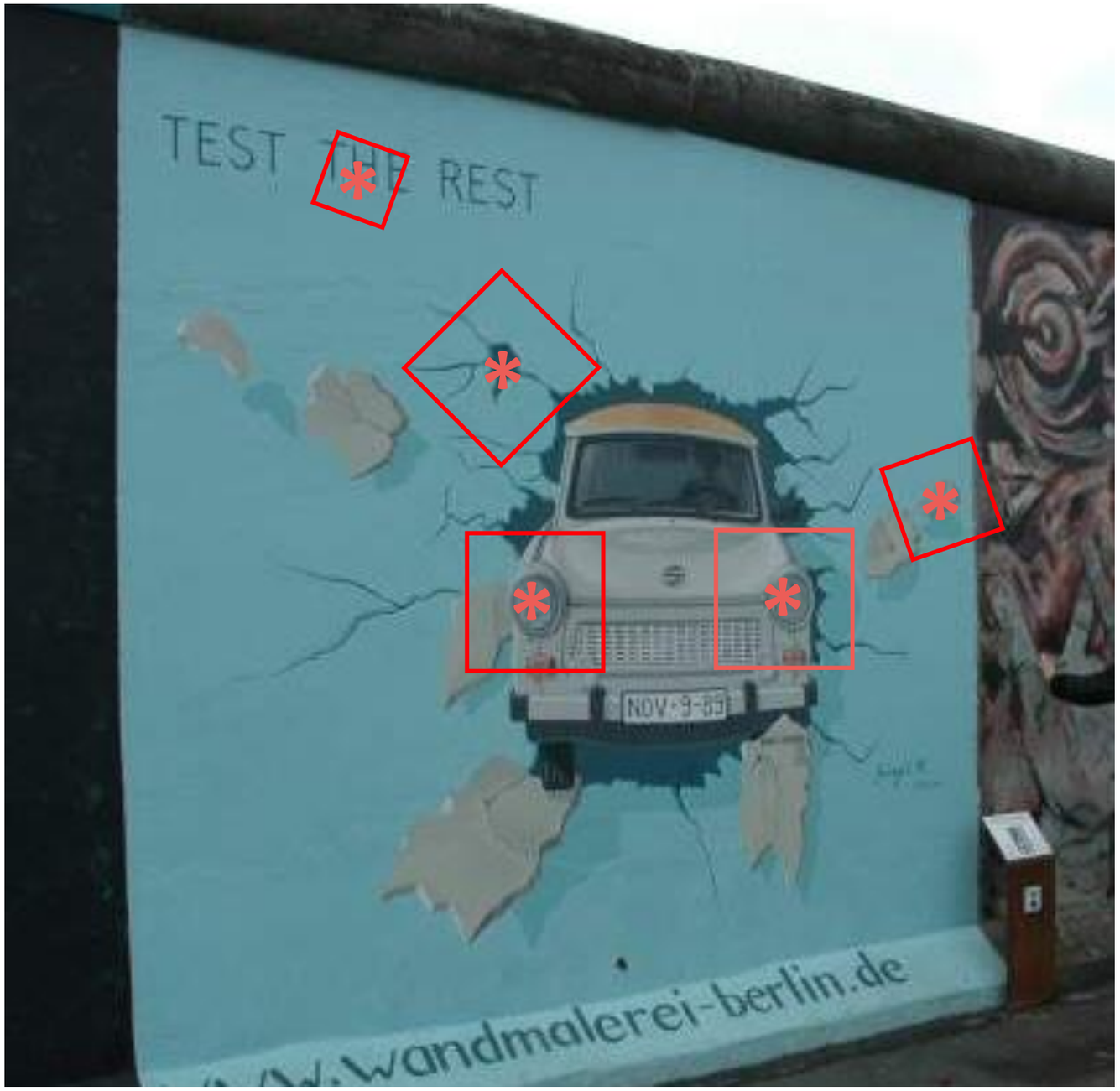


Changes in scale

in-plane rotation

Changes in brightness

Scale-Invariant Feature Transform (SIFT) Features



- Find a set of *interest* points
- **Extract scaled & oriented patch around interest points**
- Compute SIFT descriptor for each patch

Scale Invariance

Detect a size, s, for each point

Scale Invariance

Detect a size, s, for each point

Scale Invariance

Detect a size, s, for each point
Scale patch size to fit.
Compute gradients in $L(x, y, s^2)$

Scale Invariance

size=5

size=0.5

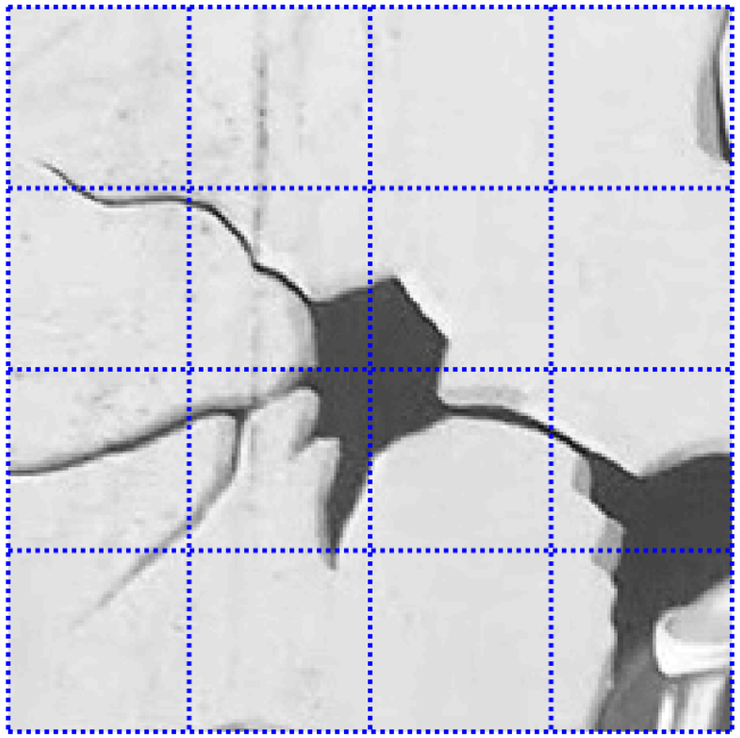
Scale Invariance

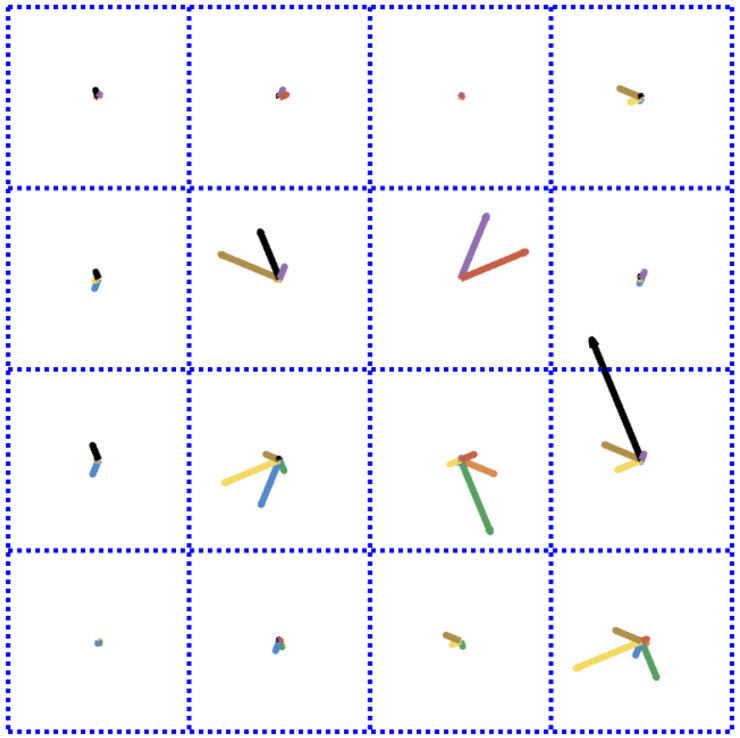
size=5

size=0.5

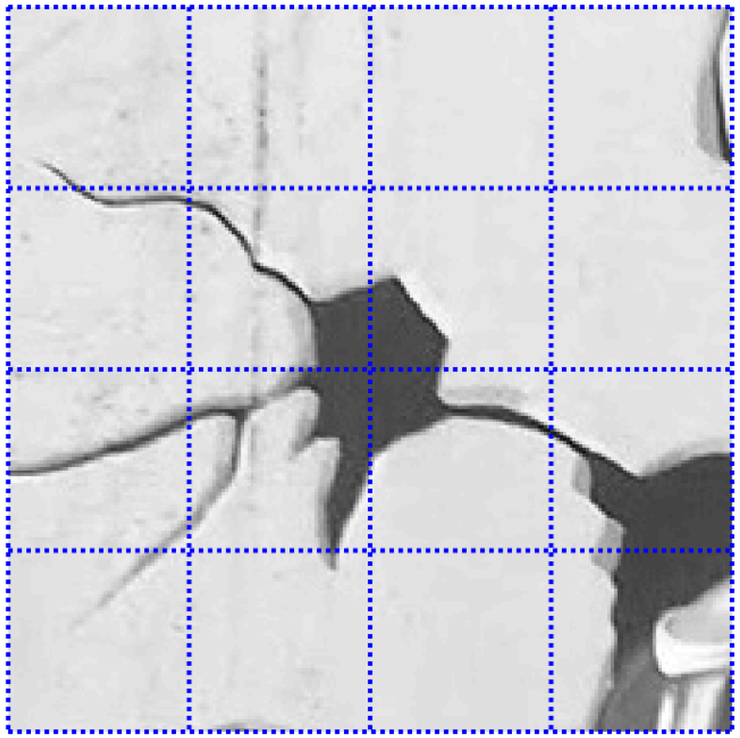
Map to canonical patch before descriptor extraction

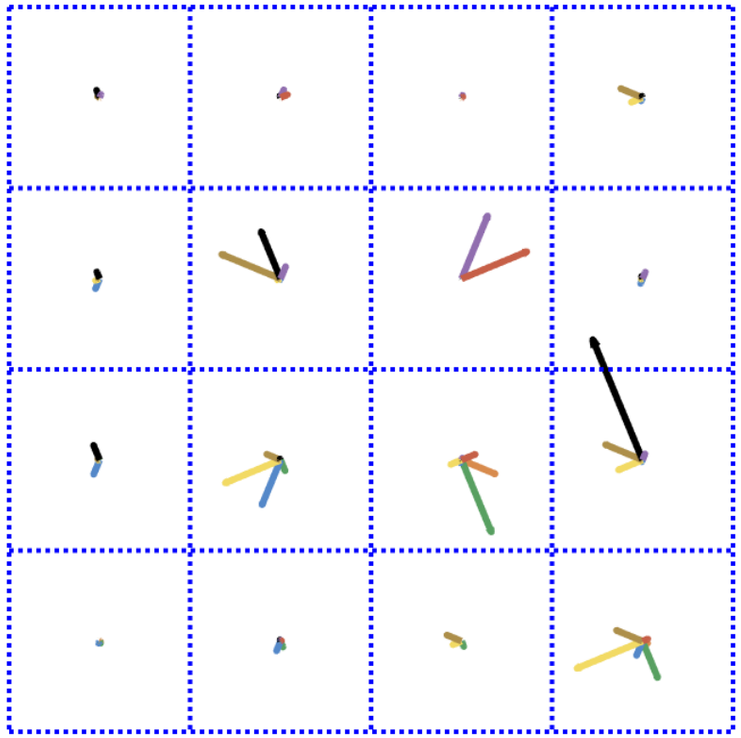
Canonical Patches

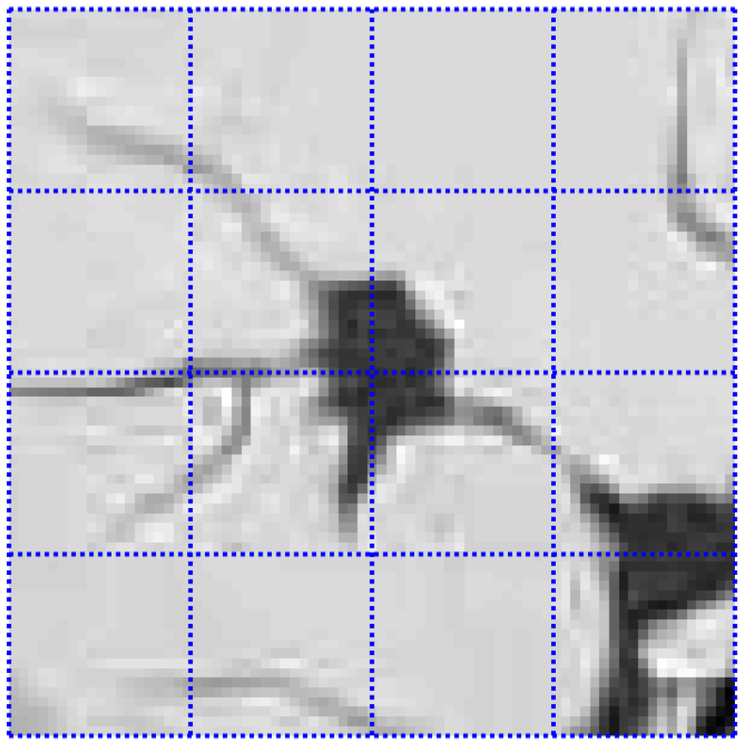


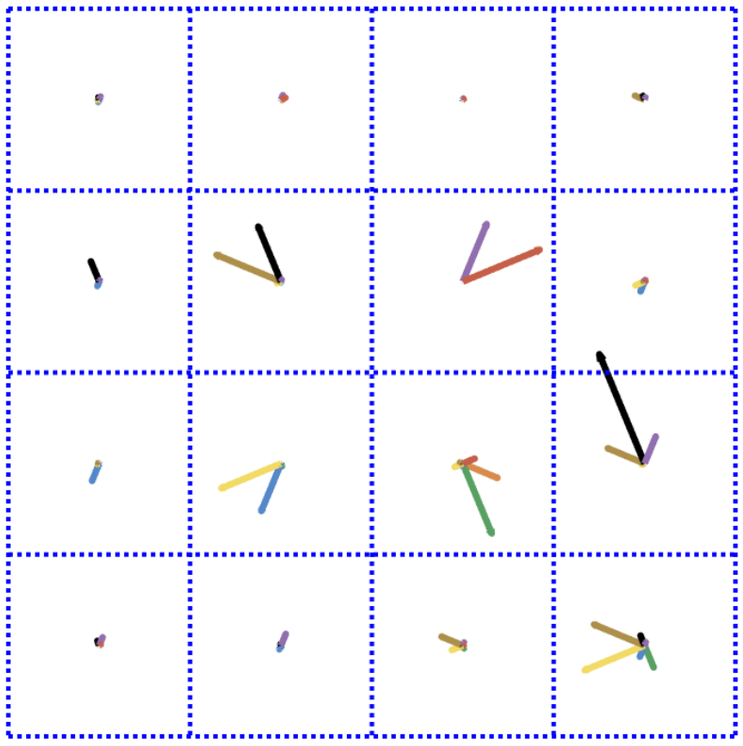


Canonical Patches

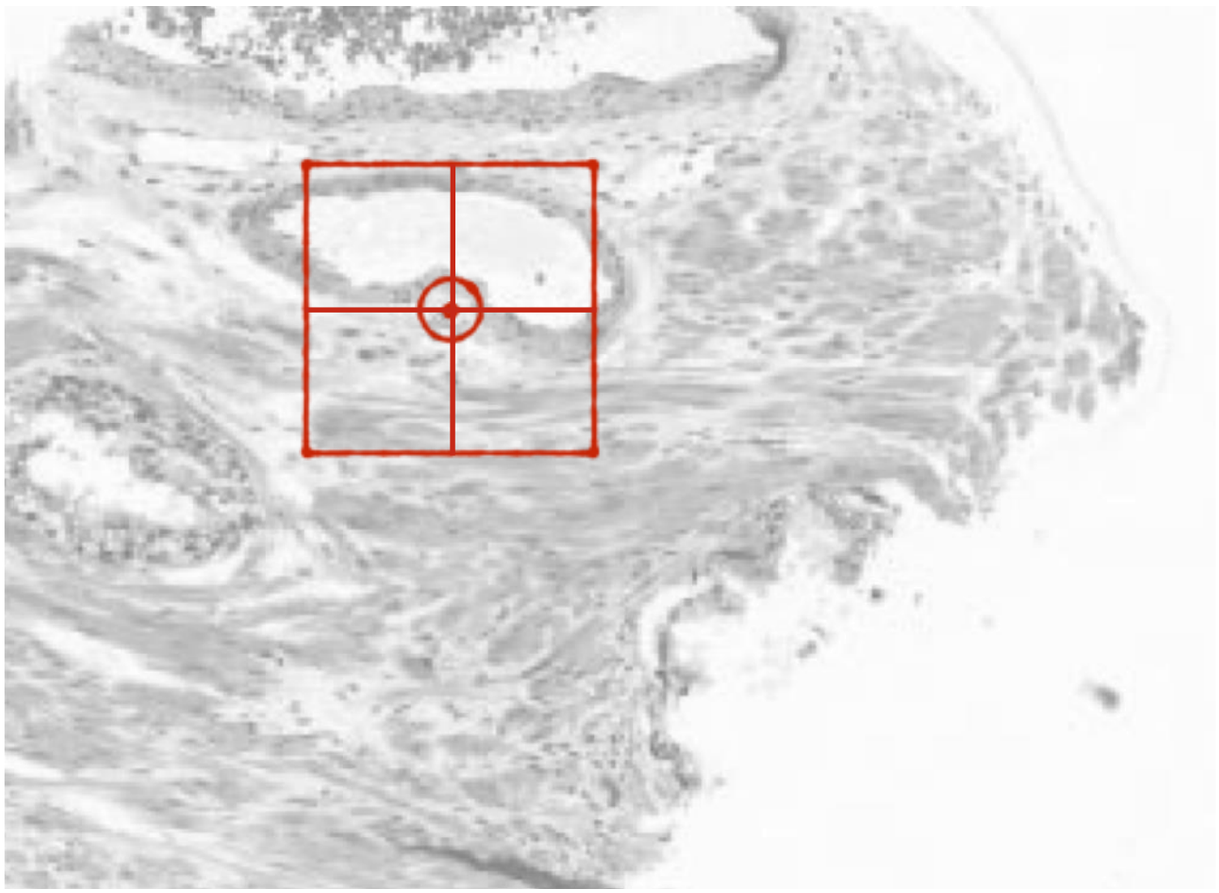


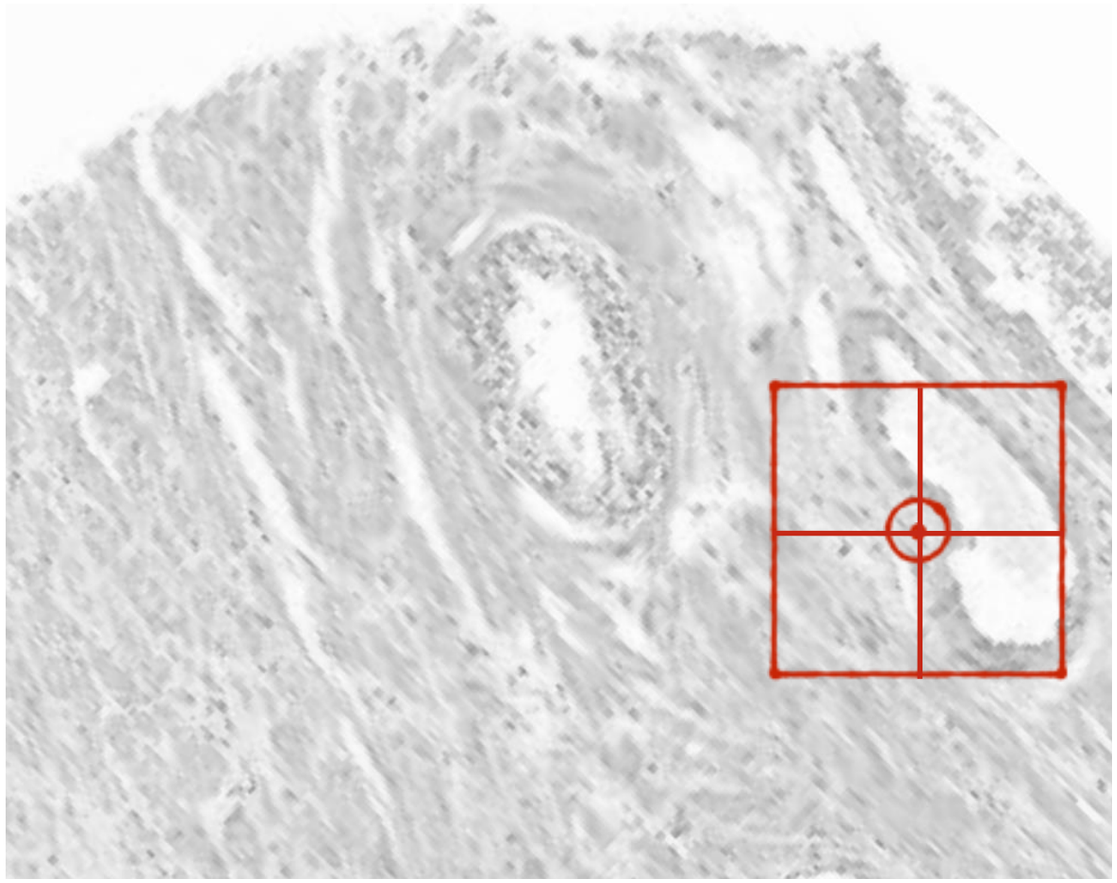




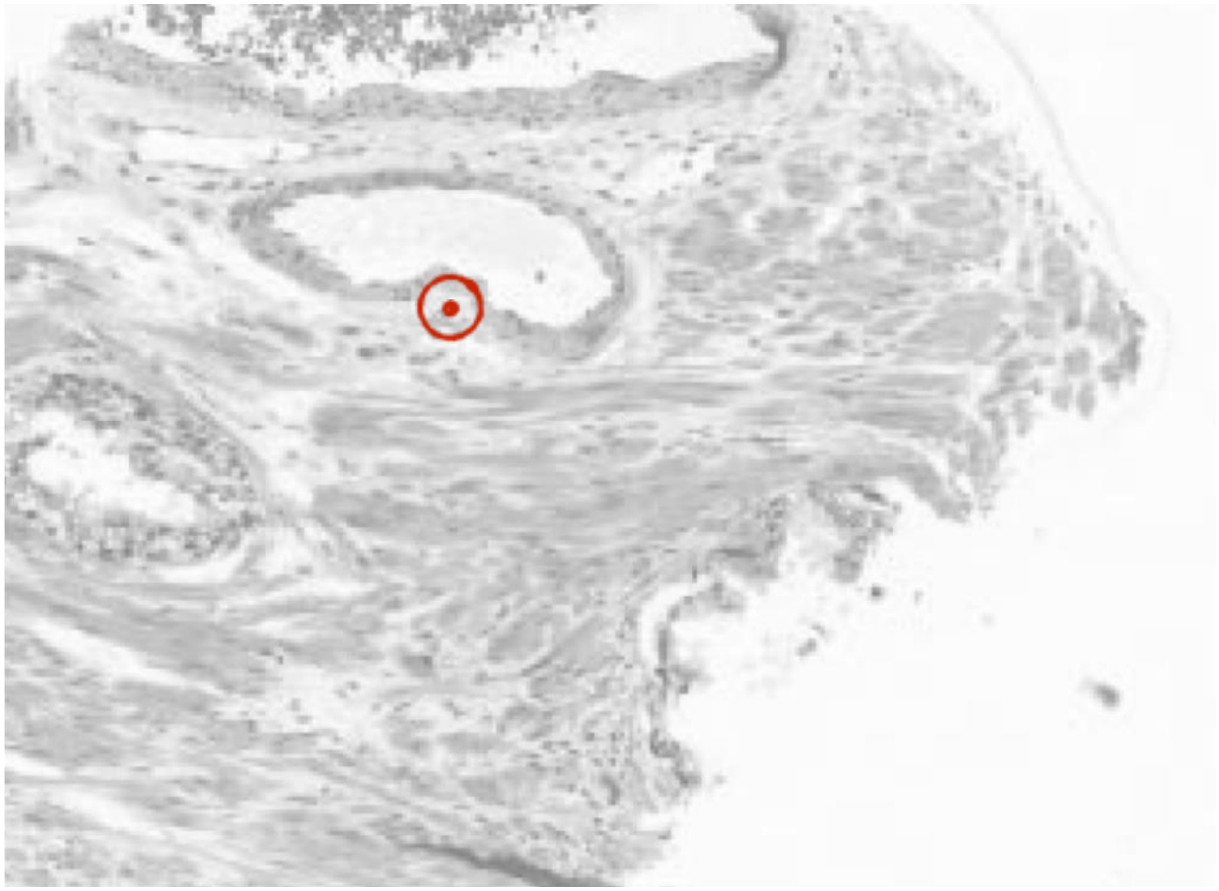


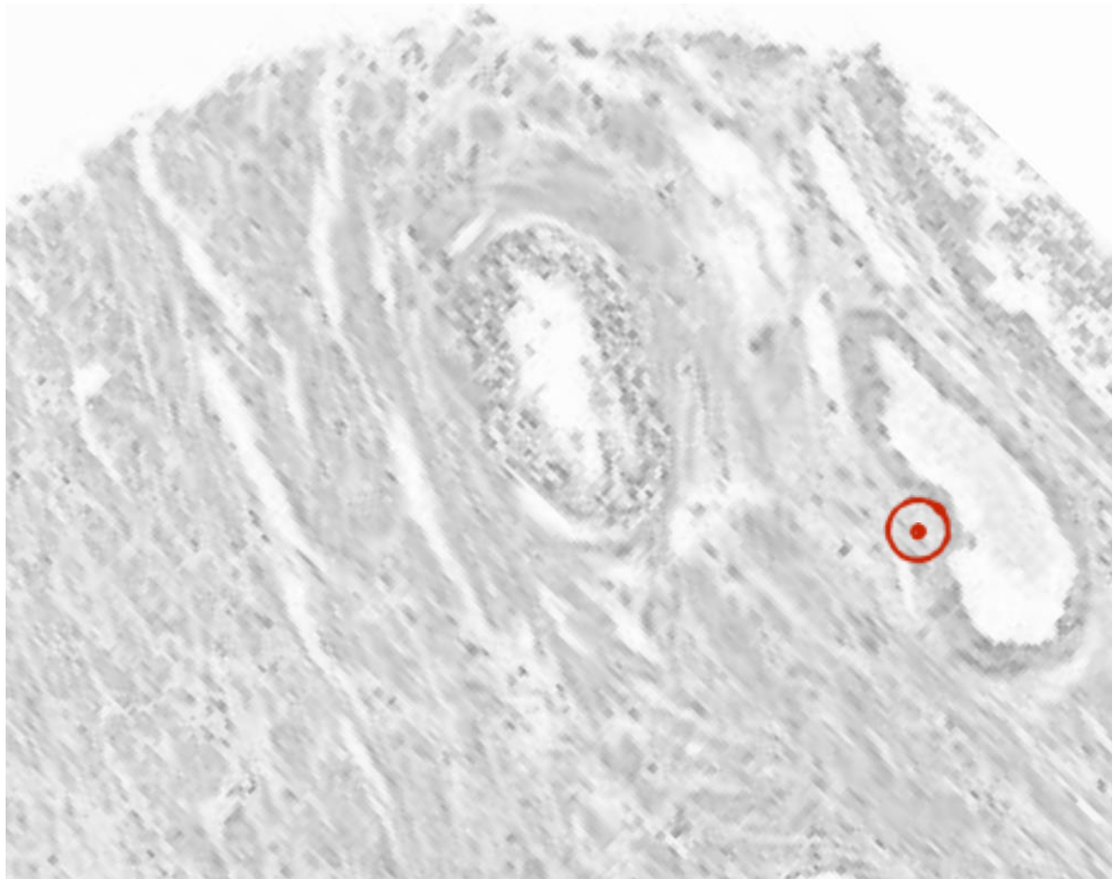
Challenge: Rotations





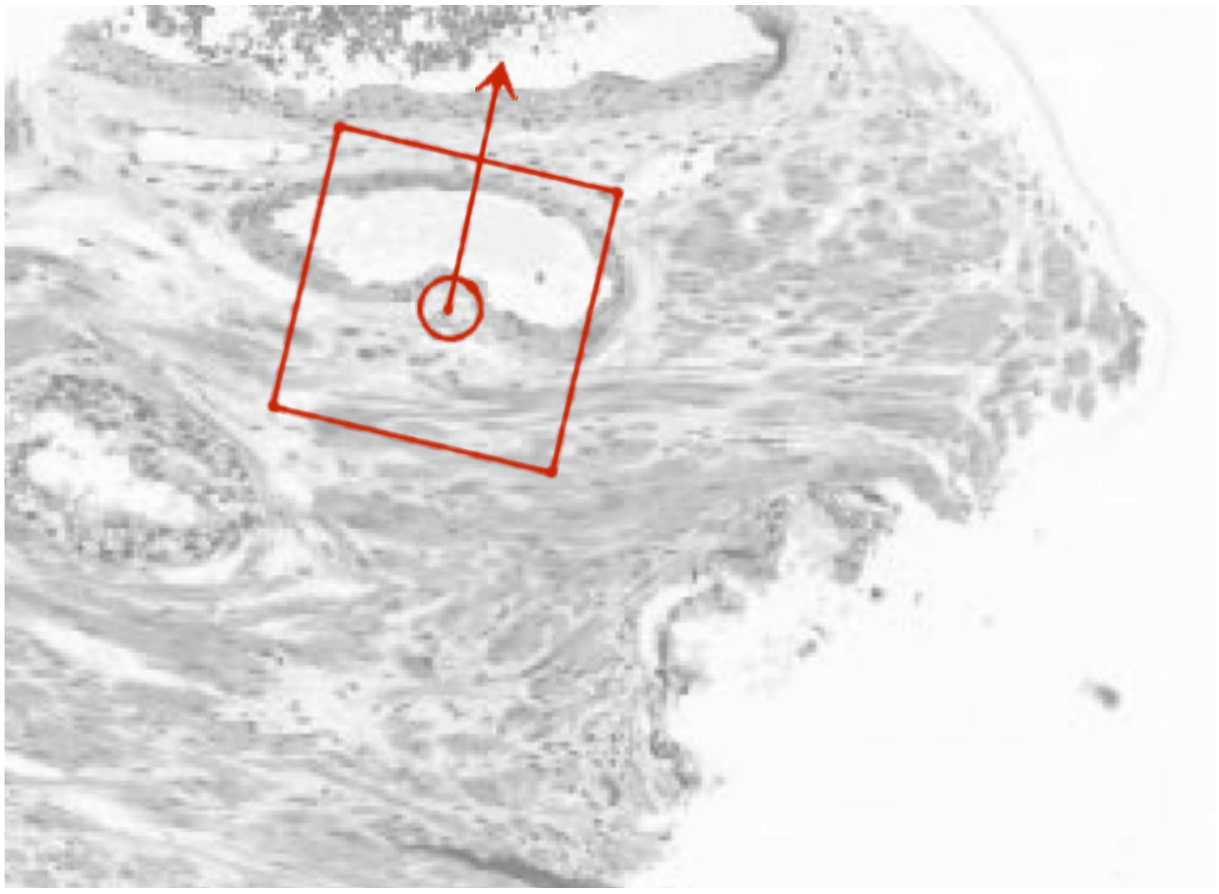
Challenge: Rotations

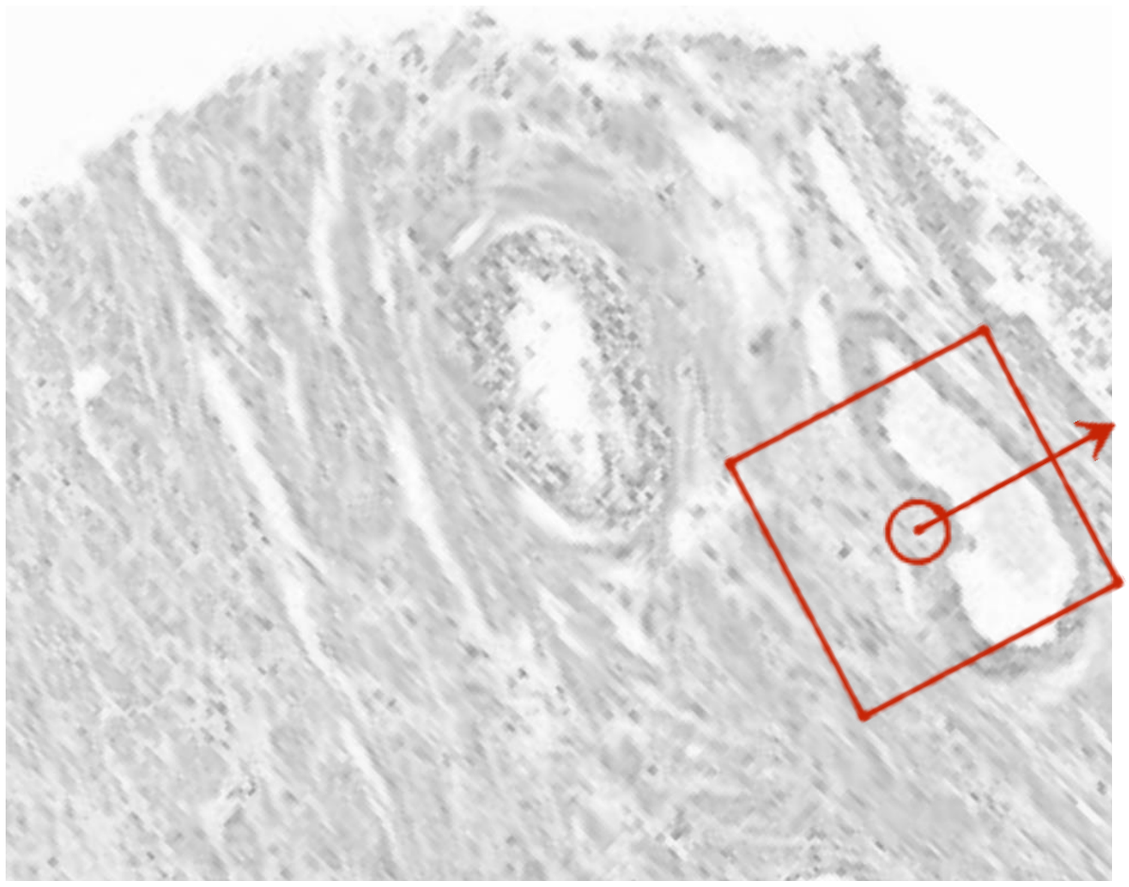




Estimate a dominant direction

Challenge: Rotations





Estimate a dominant direction

Align patches to this direction prior to descriptor extraction

Estimating the Dominant Orientation

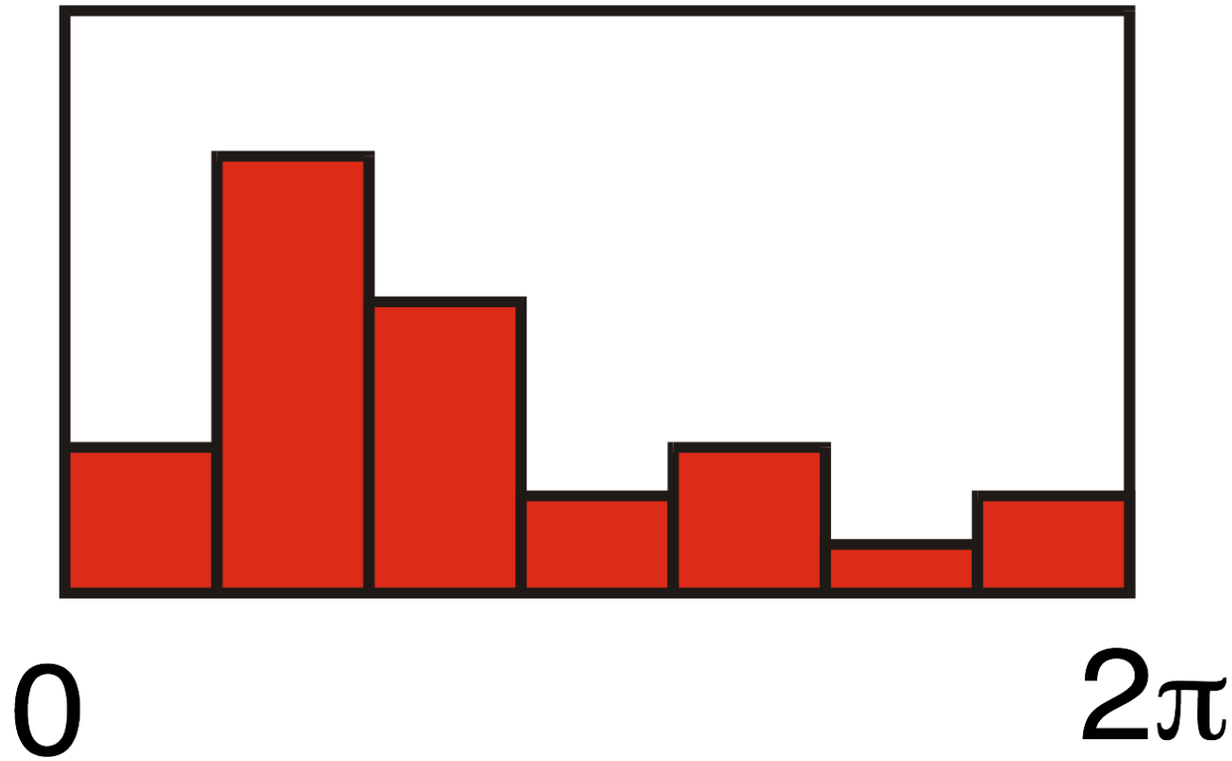
Estimated gradients

illustrations from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Estimating the Dominant Orientation

Estimated gradients

Gradient histogram

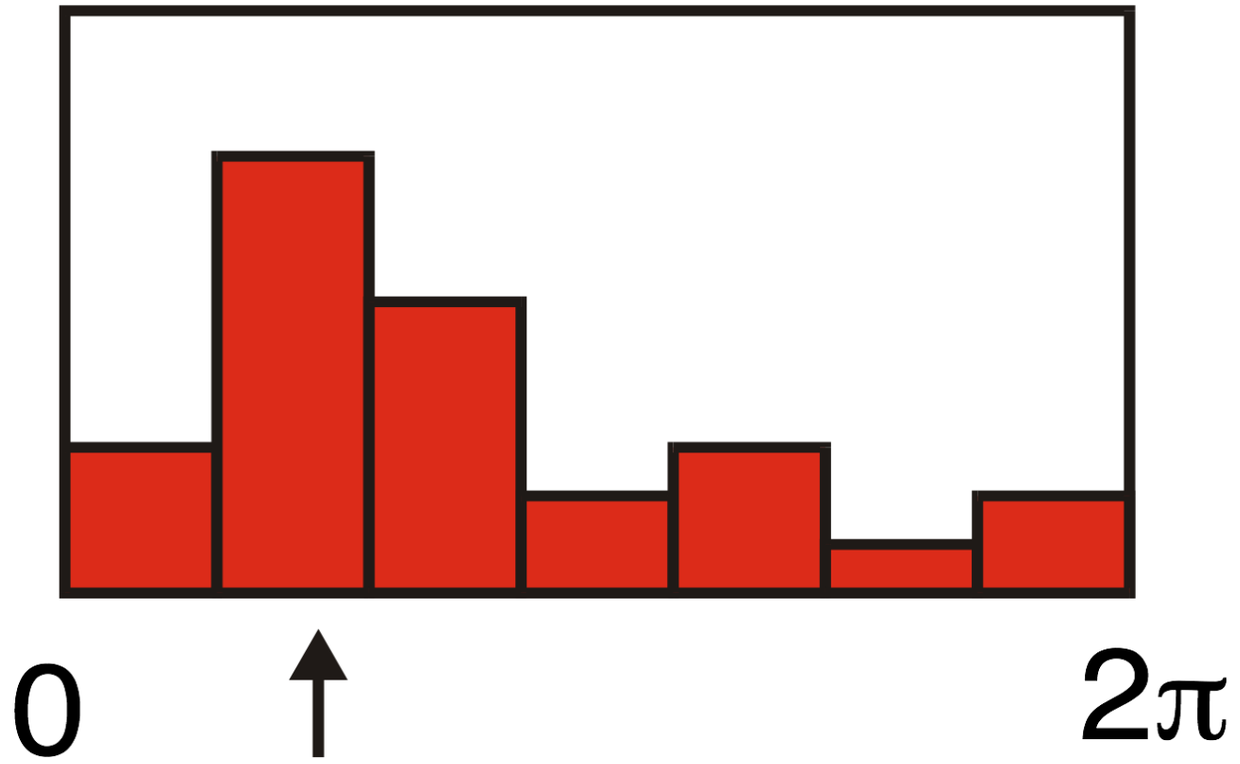


illustrations from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Estimating the Dominant Orientation

Estimated gradients

Gradient histogram

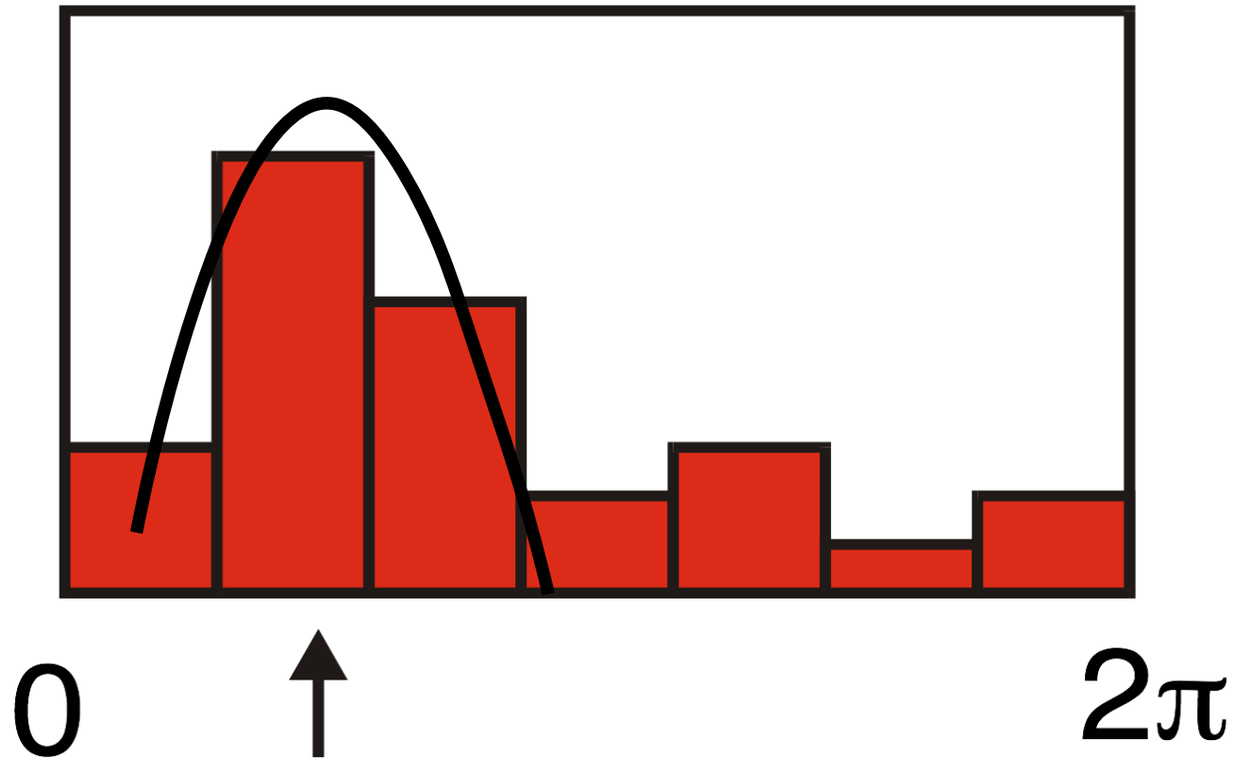


illustrations from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Estimating the Dominant Orientation

Estimated gradients

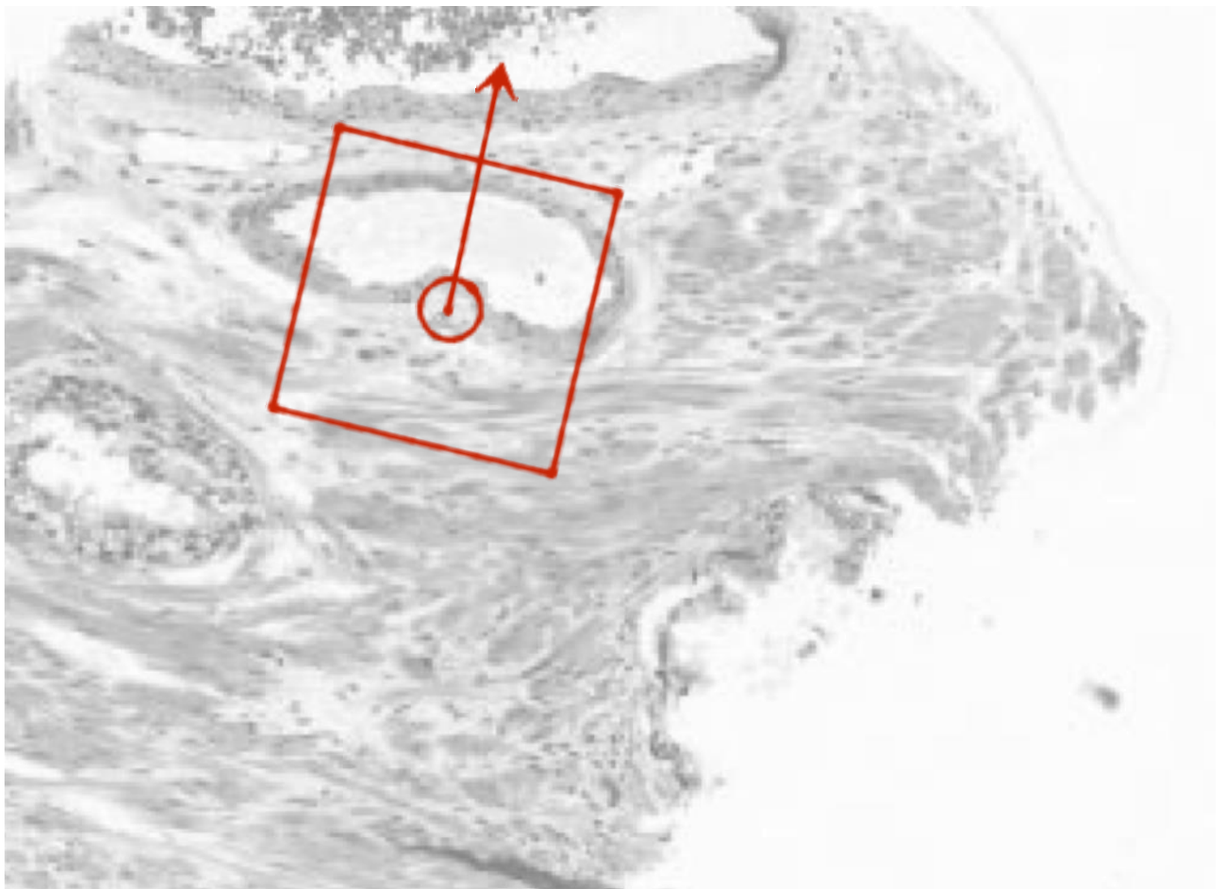
Gradient histogram

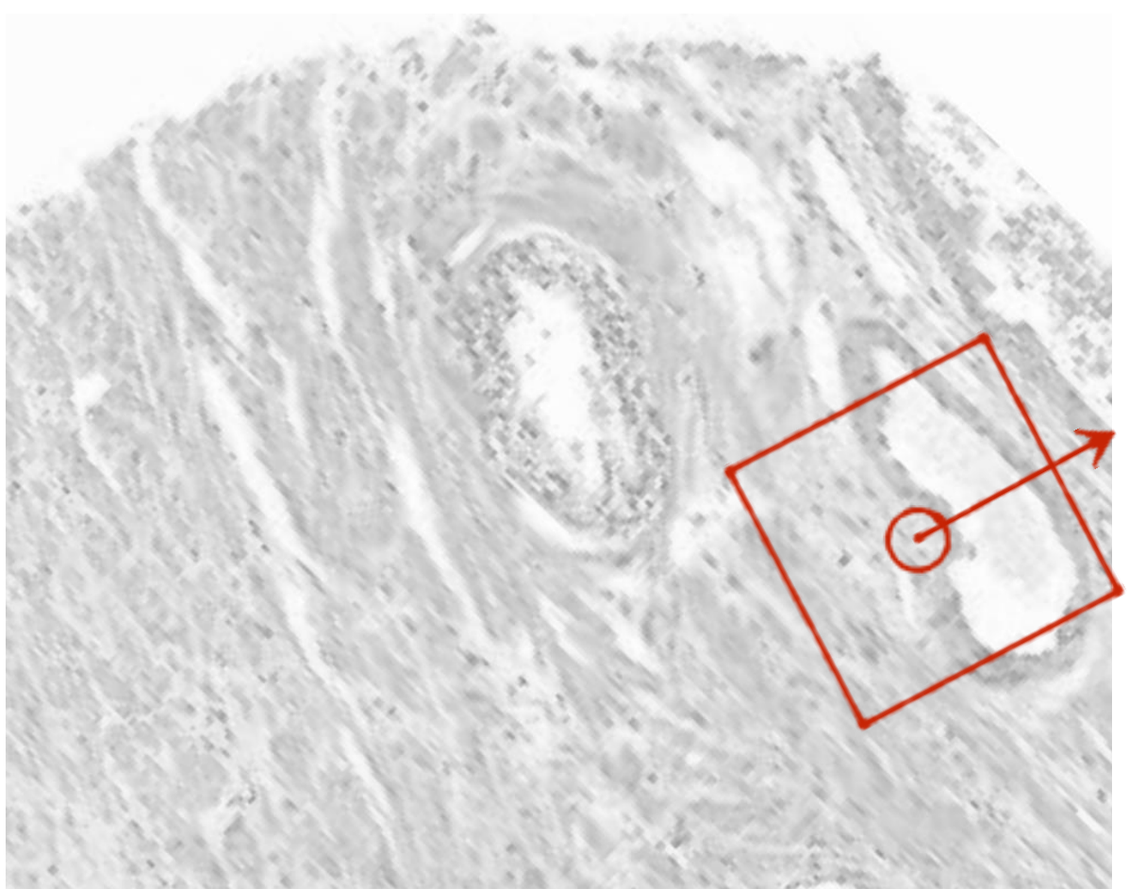


Dominant direction
refine by curve fitting

illustrations from
[Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004]

Estimating the Dominant Orientation

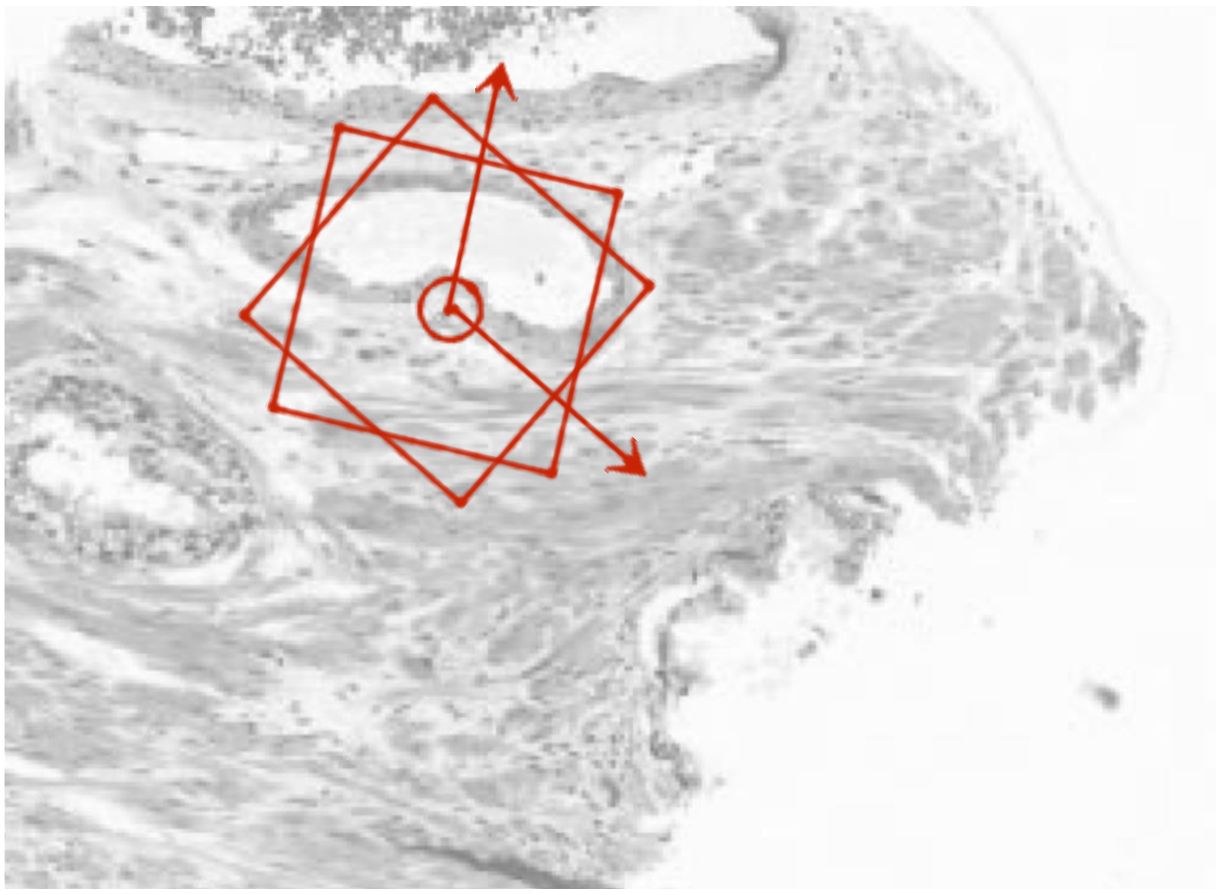


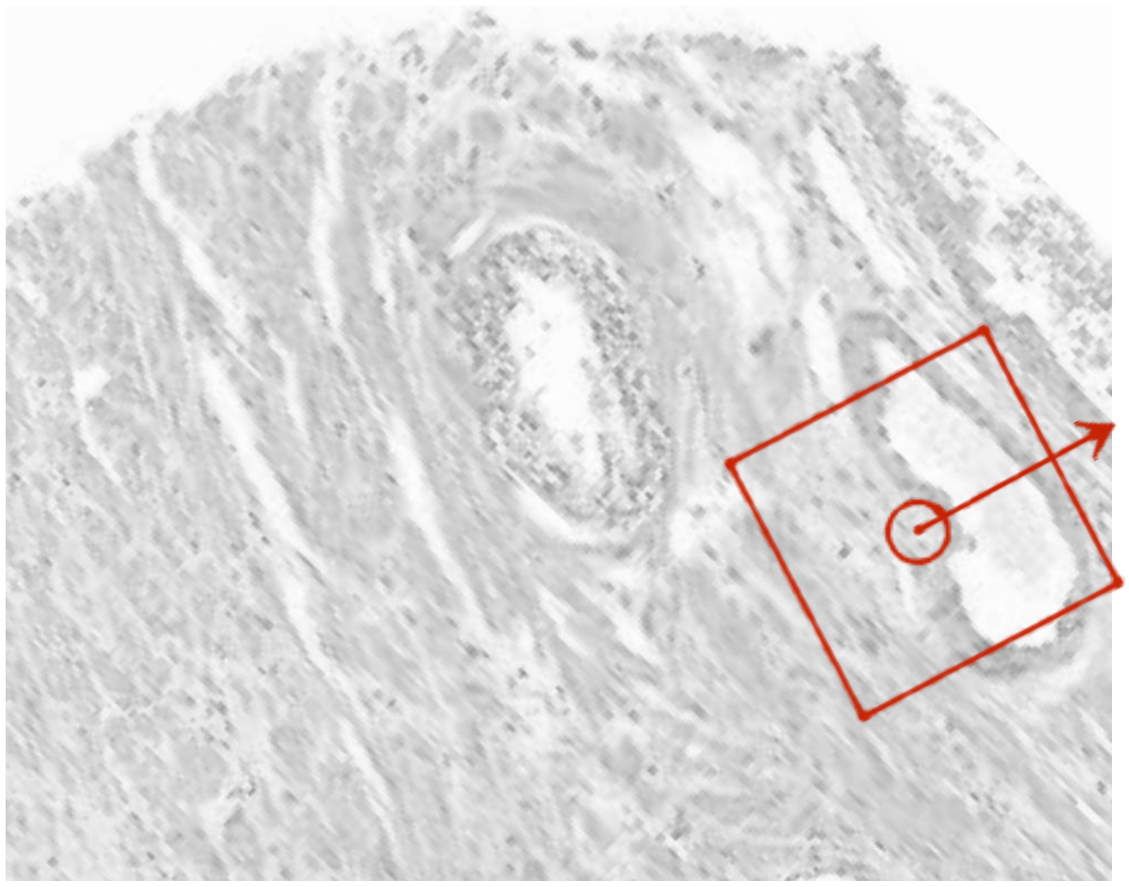


Estimate a dominant direction

Align patches to this direction prior to descriptor extraction

Estimating the Dominant Orientation





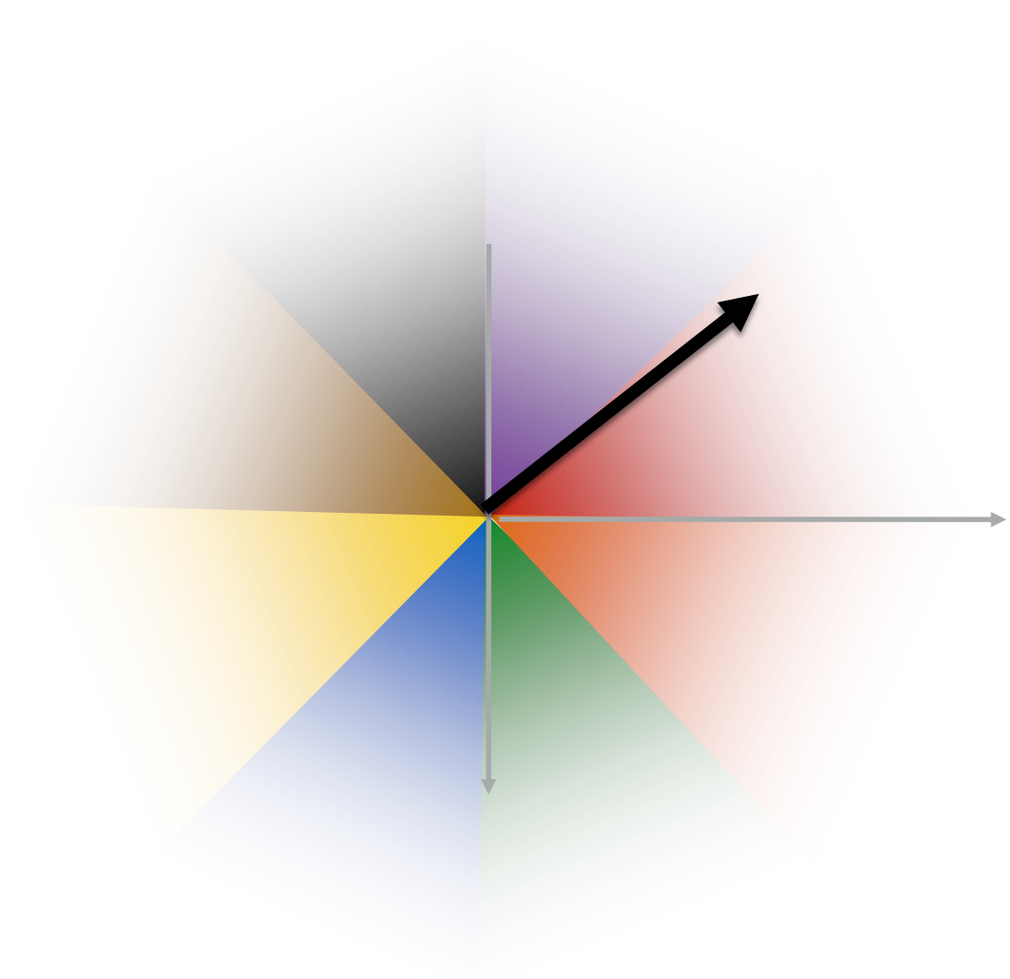
Estimate a dominant direction

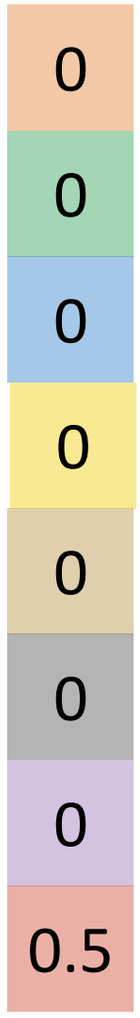
Align patches to this direction prior to descriptor extraction

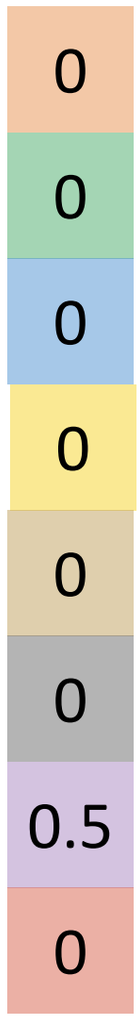
Can create multiple patches per keypoint (multiple dominant directions)

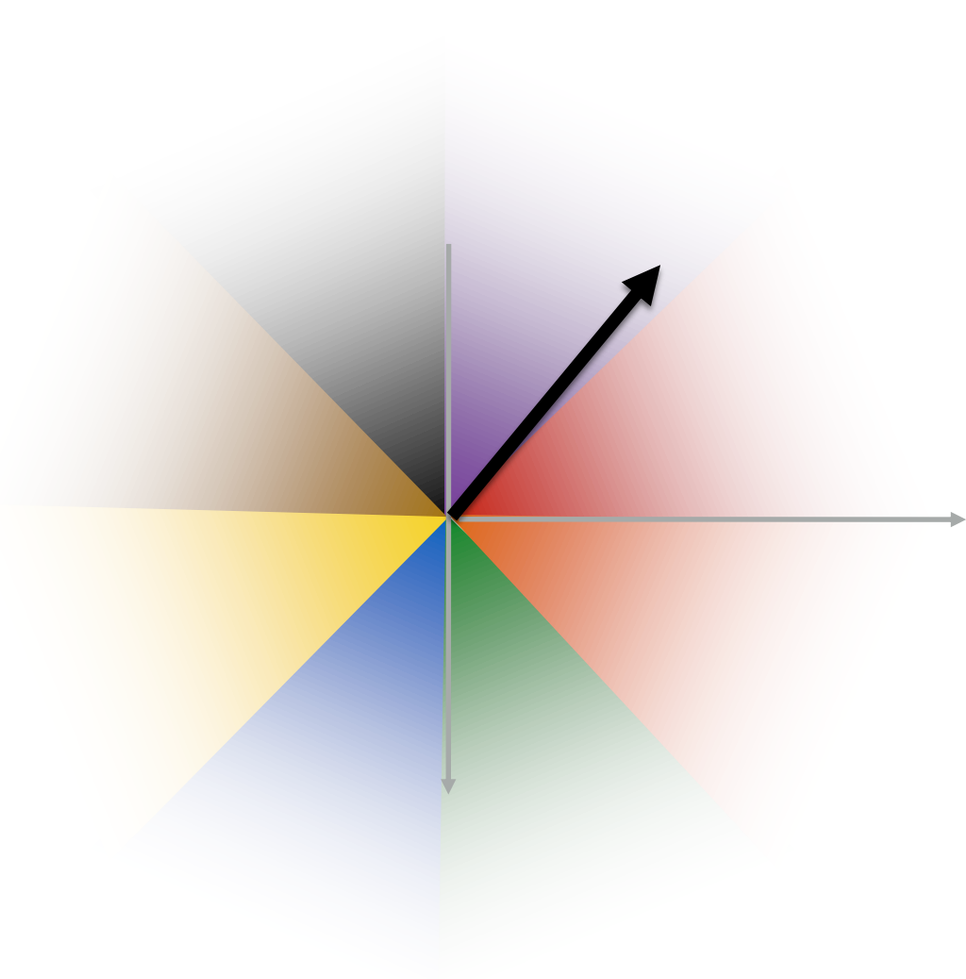
Estimating the Dominant Orientation

Boundary effects in descriptor due to small errors in orientation





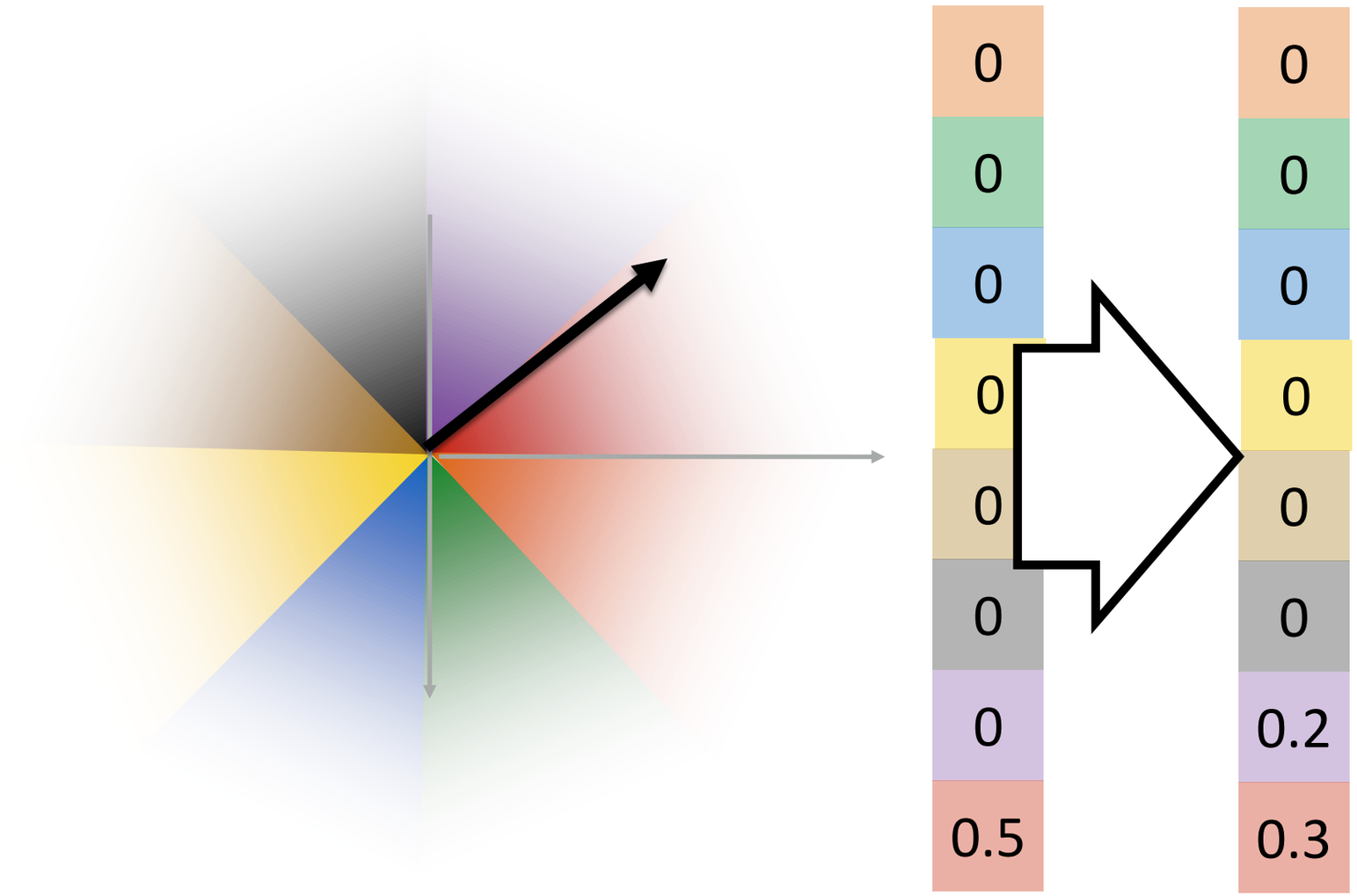




slide credit: Marc Pollefeys, Kevin Köser

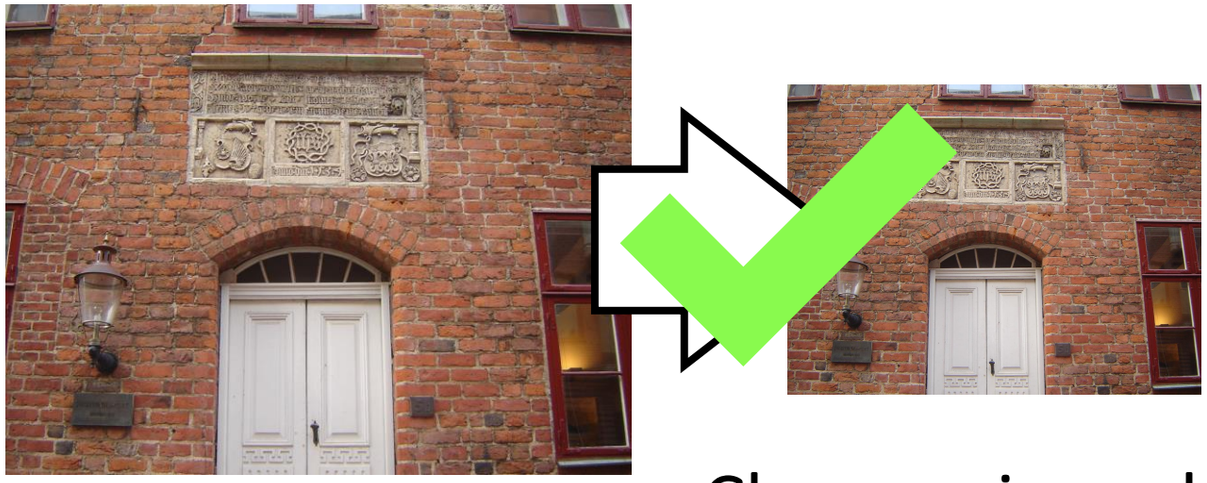
Soft-Binning

Distribute weights to multiple bins via linear interpolation



slide credit: Marc Pollefeys, Kevin Köser

Invariances?

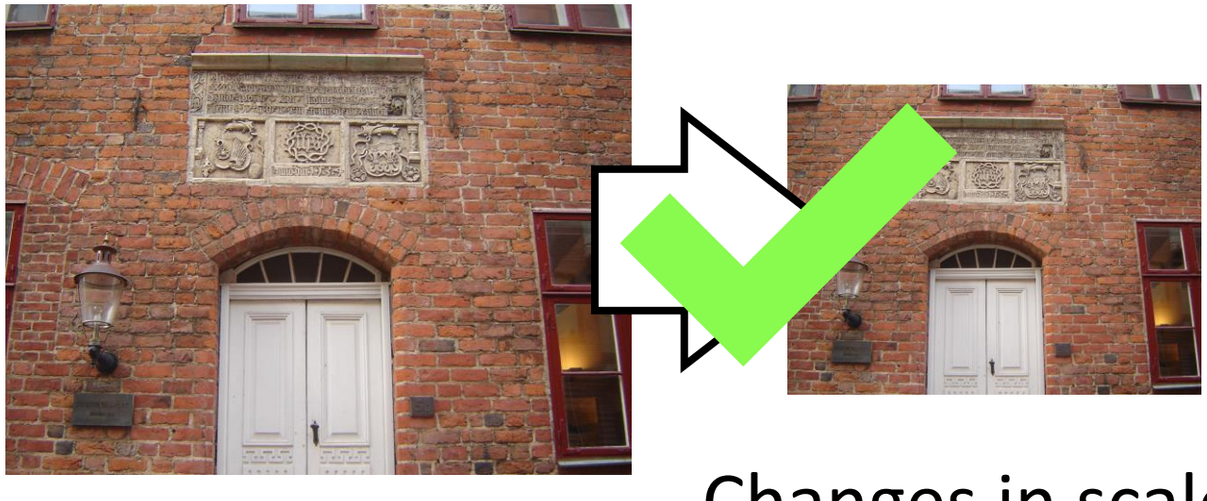


Changes in scale

in-plane rotation

Changes in brightness

Invariances?



Changes in scale

in-plane rotation

Changes in brightness

Changes in viewpoint

Invariances?

Changes in scale

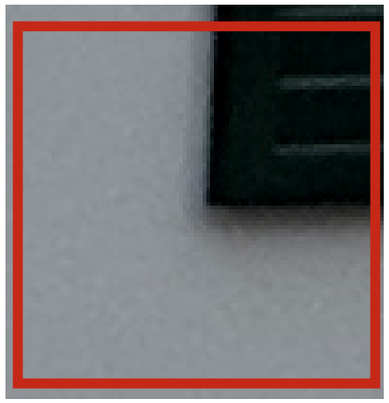
in-plane rotation

Changes in brightness

Not invariant, but reasonably robust

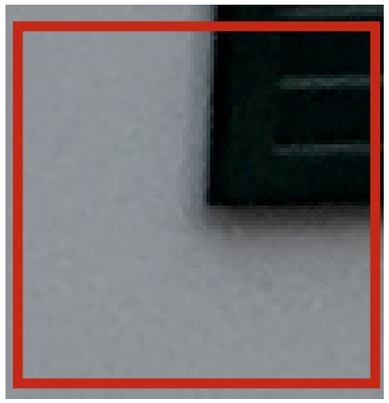
Changes in viewpoint

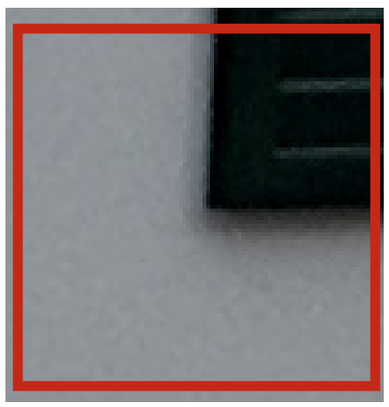
How Much Invariance Do We Want?

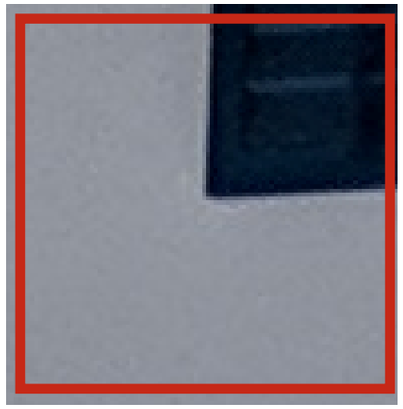


Invariance can also reduce performance

How Much Invariance Do We Want?







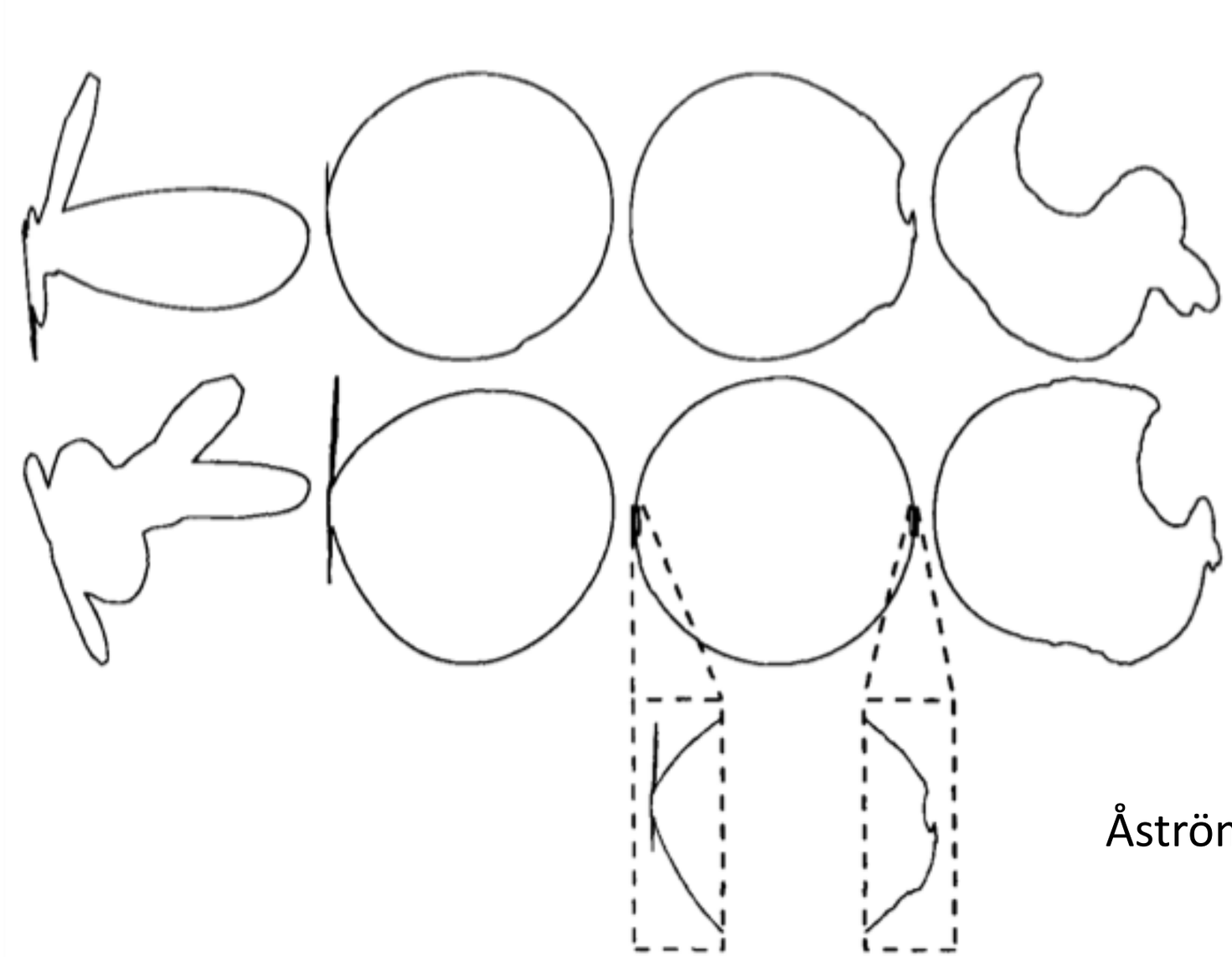
Invariance can also reduce performance

Is Your Visual System Rotation Invariant?

Is Your Visual System Rotation Invariant?

Thatcher illusion

Invariance vs. Robustness

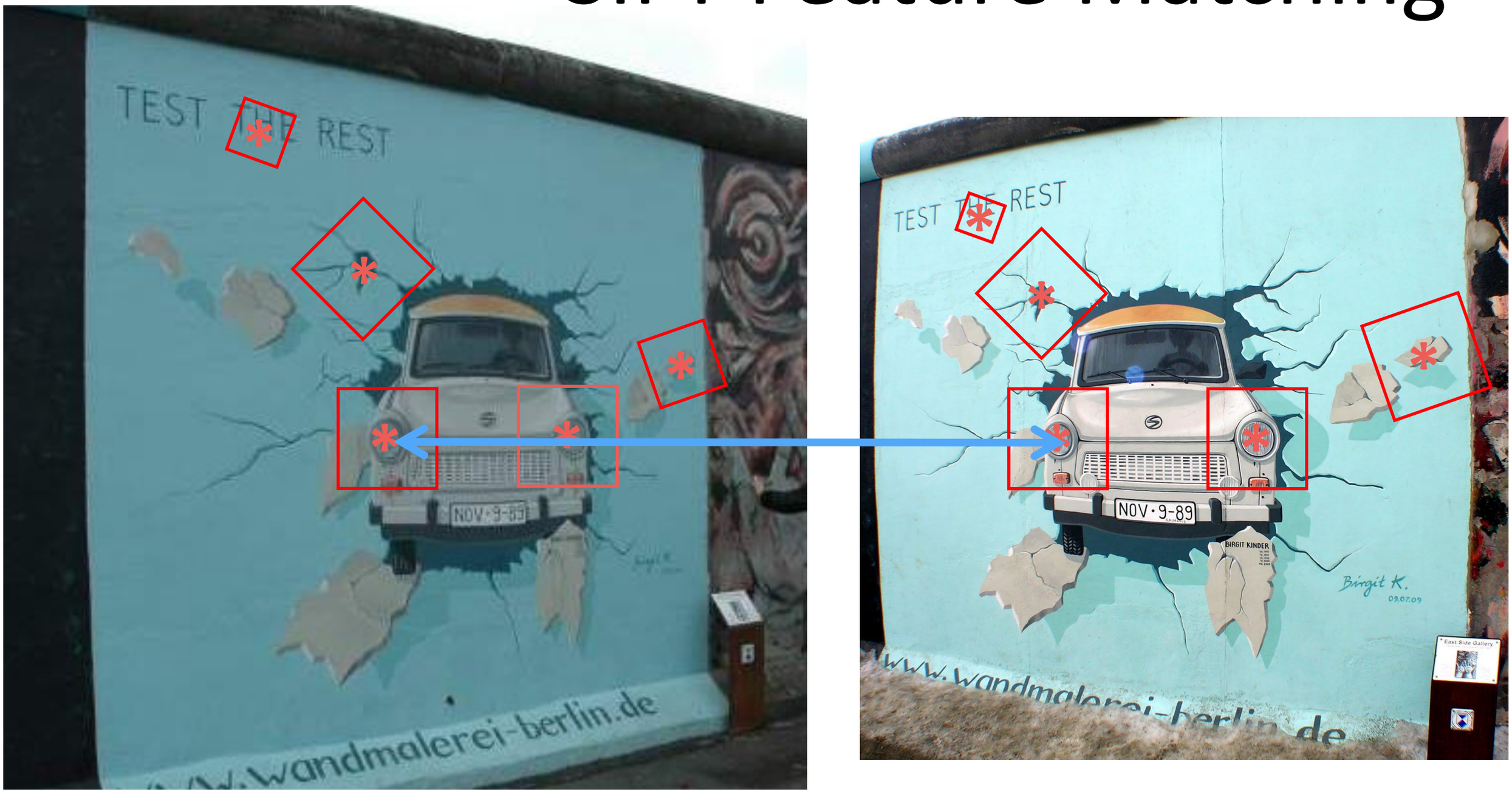


Åström-95

Invariance = no influence on result

Robustness = we can deal with a moderate amount

SIFT Feature Matching

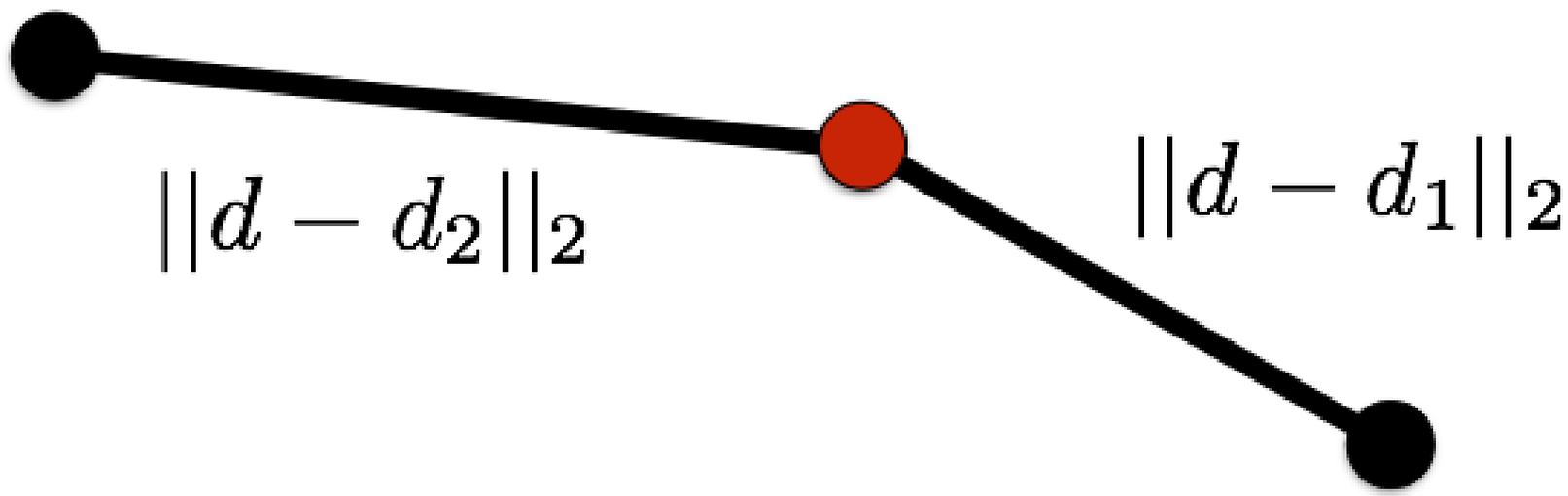


Use descriptor similarity (L_2 distance) to find corresponding points in different images

L_2 Similar = Correct?

query:

Lowe's Ratio Test



Accept match with 1st neighbor if $\dfrac{||d - d_1||_2}{||d - d_2||_2} < 0.8$

Faster Matching

- Exhaustively comparing all descriptors is expensive

Faster Matching

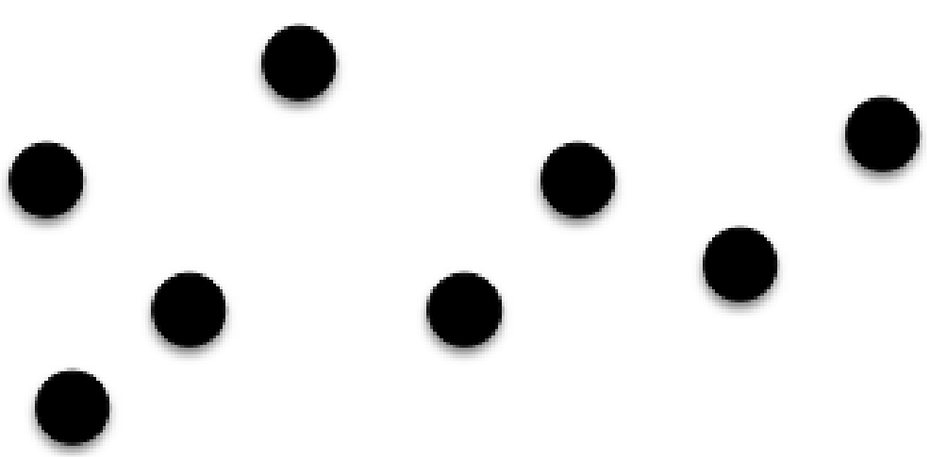
- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance

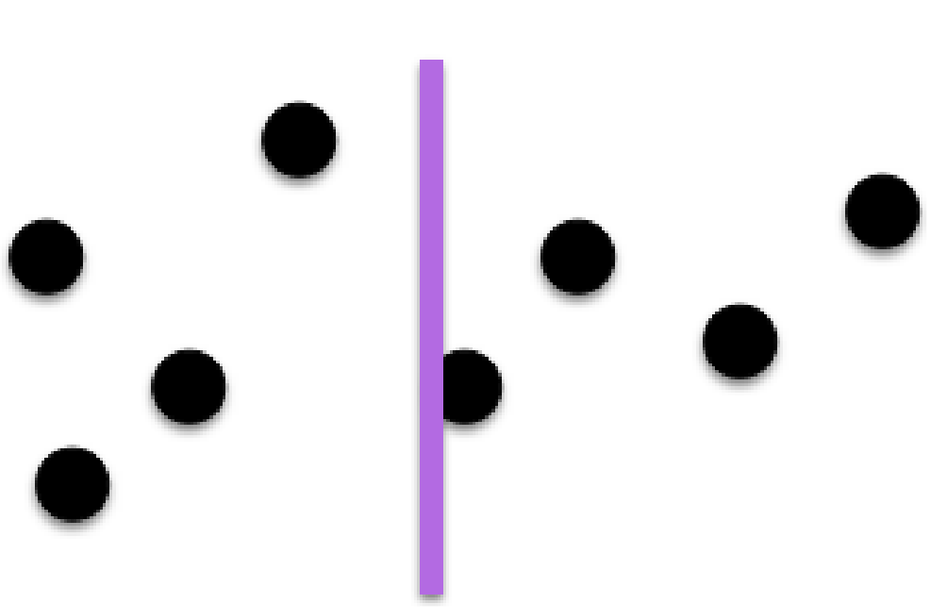


Descriptor Space

kd-tree

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance

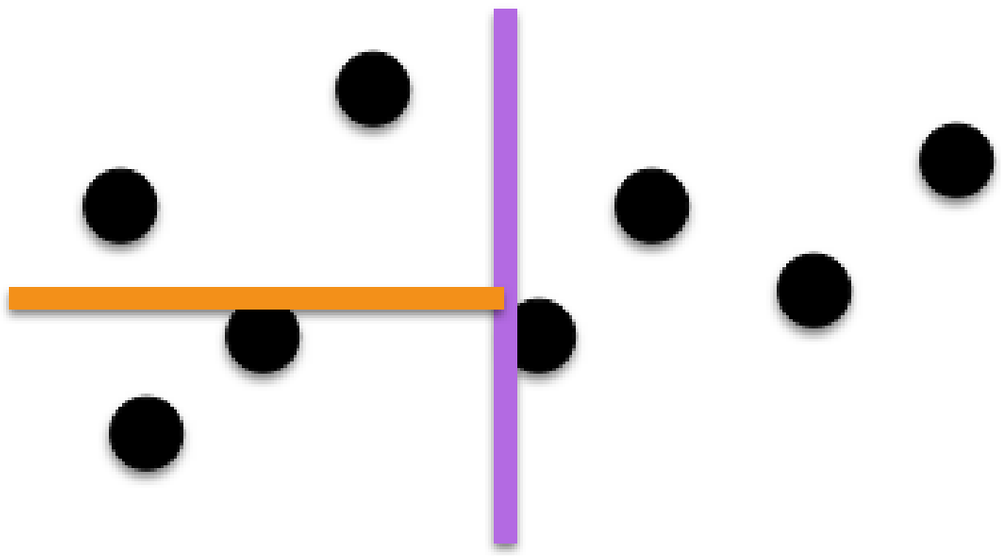


Descriptor Space

kd-tree

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance

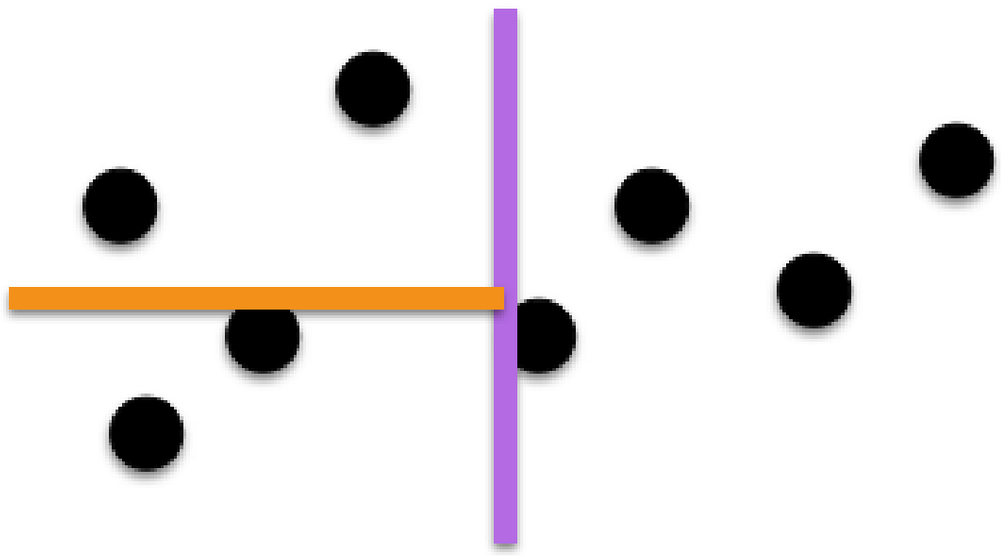


Descriptor Space

kd-tree

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance

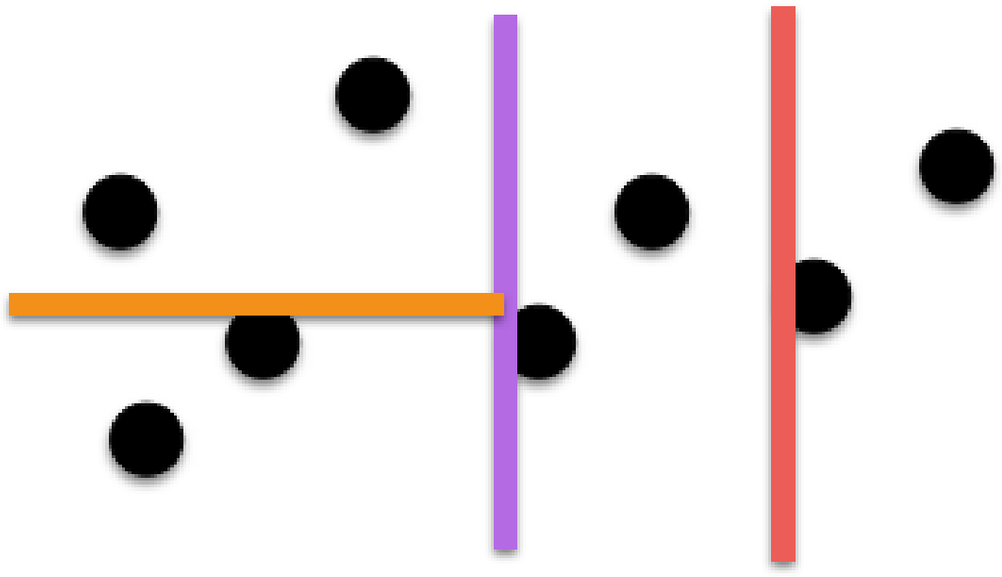


Descriptor Space

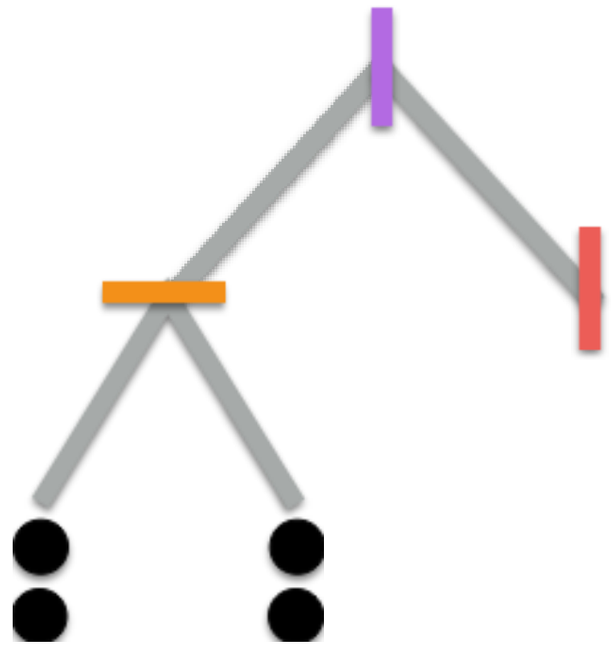
kd-tree

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance



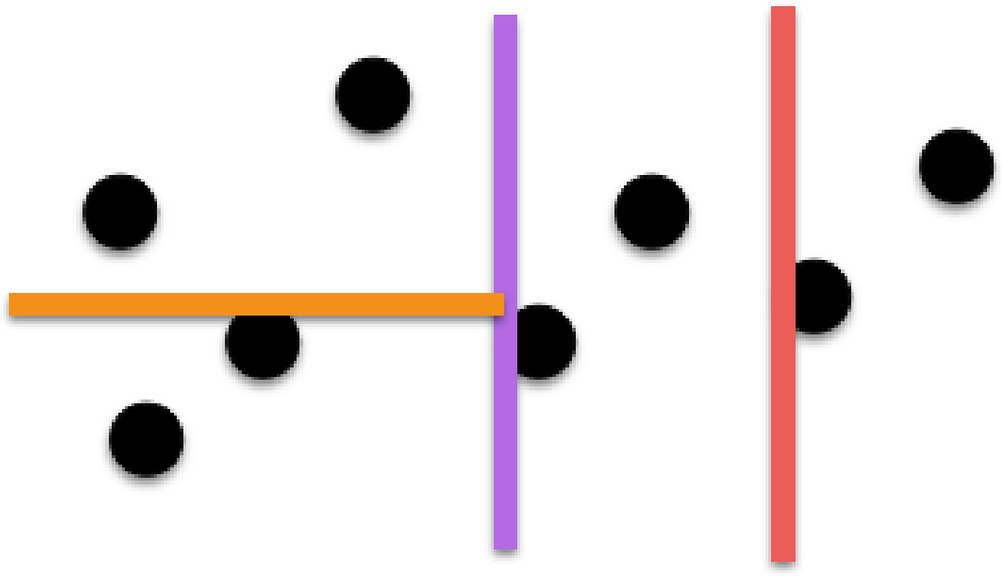
Descriptor Space



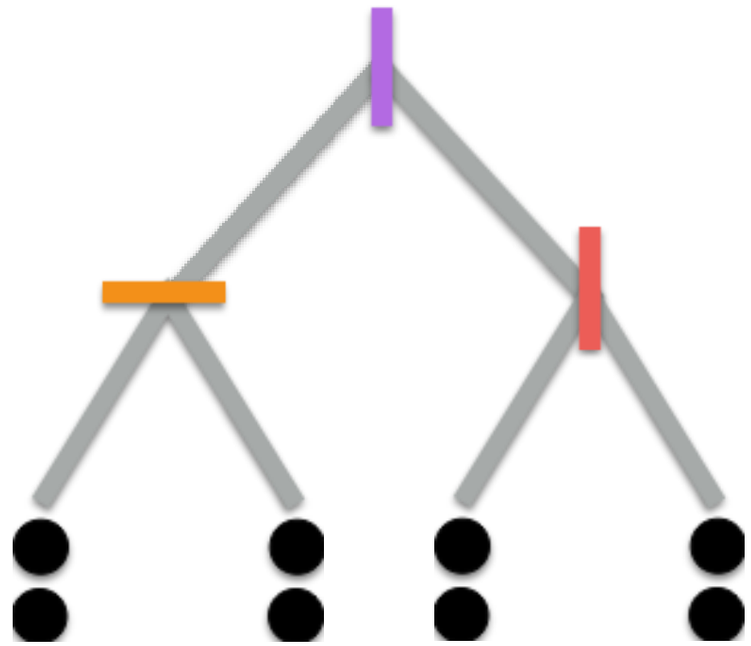
kd-tree

Faster Matching

- Exhaustively comparing all descriptors is expensive
- What about spatial acceleration schemes, e.g., kd-trees?
- kd-tree: Iteratively split descriptor space along dimension with largest variance



Descriptor Space



kd-tree

Faster Matching

- Curse of dimensionality: No acceleration scheme (asymptotically) faster than exhaustive search

Faster Matching

- Curse of dimensionality: No acceleration scheme (asymptotically) faster than exhaustive search
- Intuition: Consider SIFT (128 dimensions)
  - Around 16,000 features lead to kd-tree of depth 14
  - By the time you reach a leaf node, you have only considered 14 out of the 128 dimensions
- High chance that the other dimensions are very different
- High chance that the best leaf node does not contain nearest neighbor

- Need to essentially search through all leaves

Faster Matching

- Curse of dimensionality: No acceleration scheme (asymptotically) faster than exhaustive search
- Intuition: Consider SIFT (128 dimensions)
  - Around 16,000 features lead to kd-tree of depth 14
  - By the time you reach a leaf node, you have only considered 14 out of the 128 dimensions
  - High chance that the other dimensions are very different
  - High chance that the best leaf node does not contain nearest neighbor
  - Need to essentially search through all leaves
- **Approximate search**: Search only through fixed number of leaves

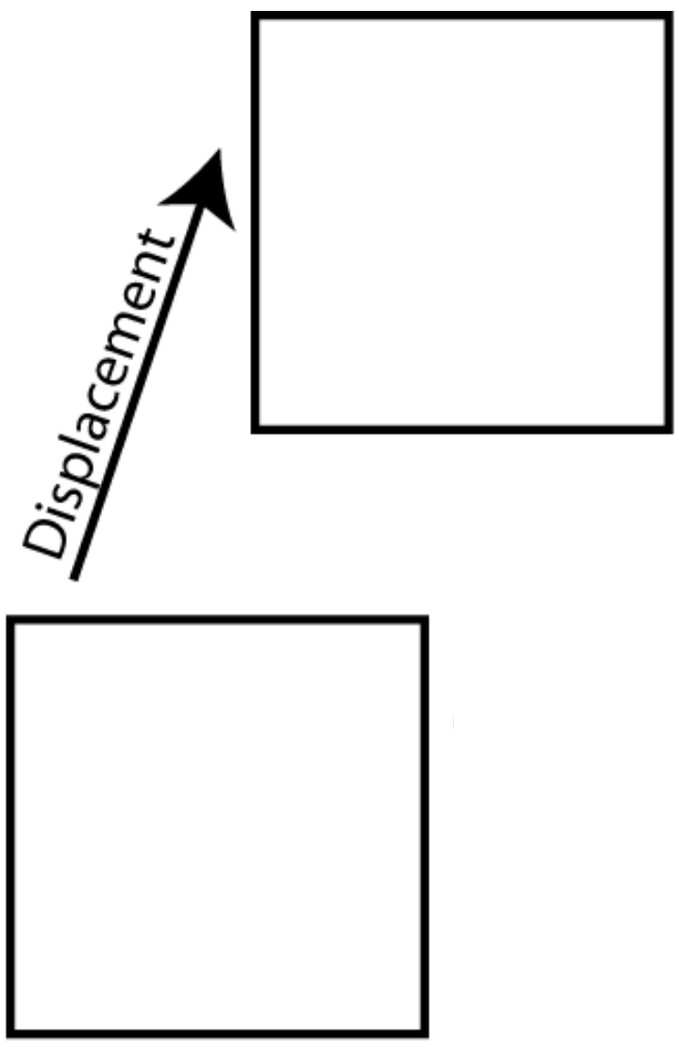
SIFT Features

- Skipped some details:
  - Down-weighting of pixels far away from center of spatial bin
  - Bilinear interpolation between spatial bins
  - For further details, read Lowe's 2004 IJCV paper

Today

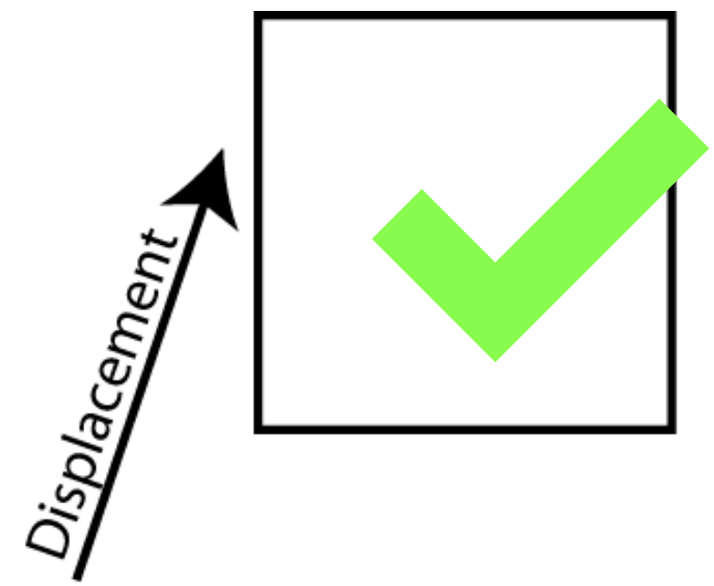
- Scale-Invariant Local Features (SIFT)
- Geometric Invariances

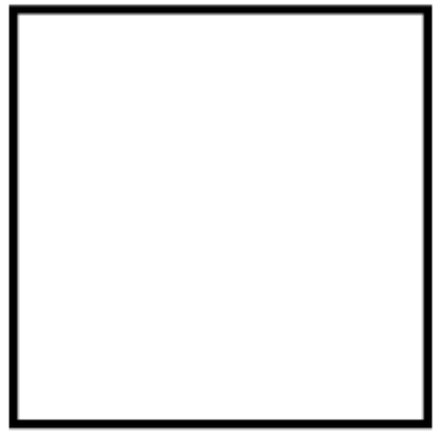
Patch Transformations



slide credit: Marc Pollefeys

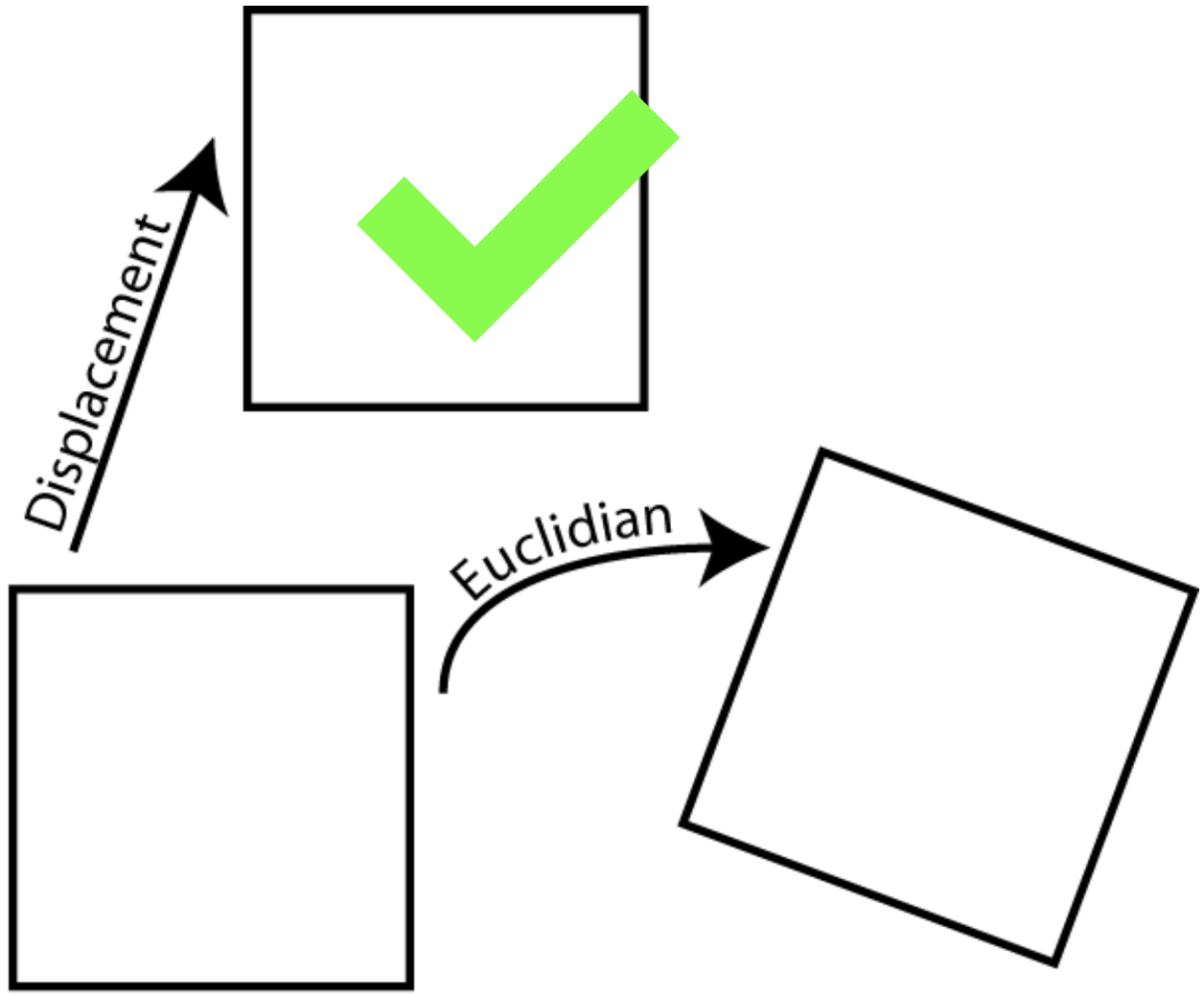
Patch Transformations





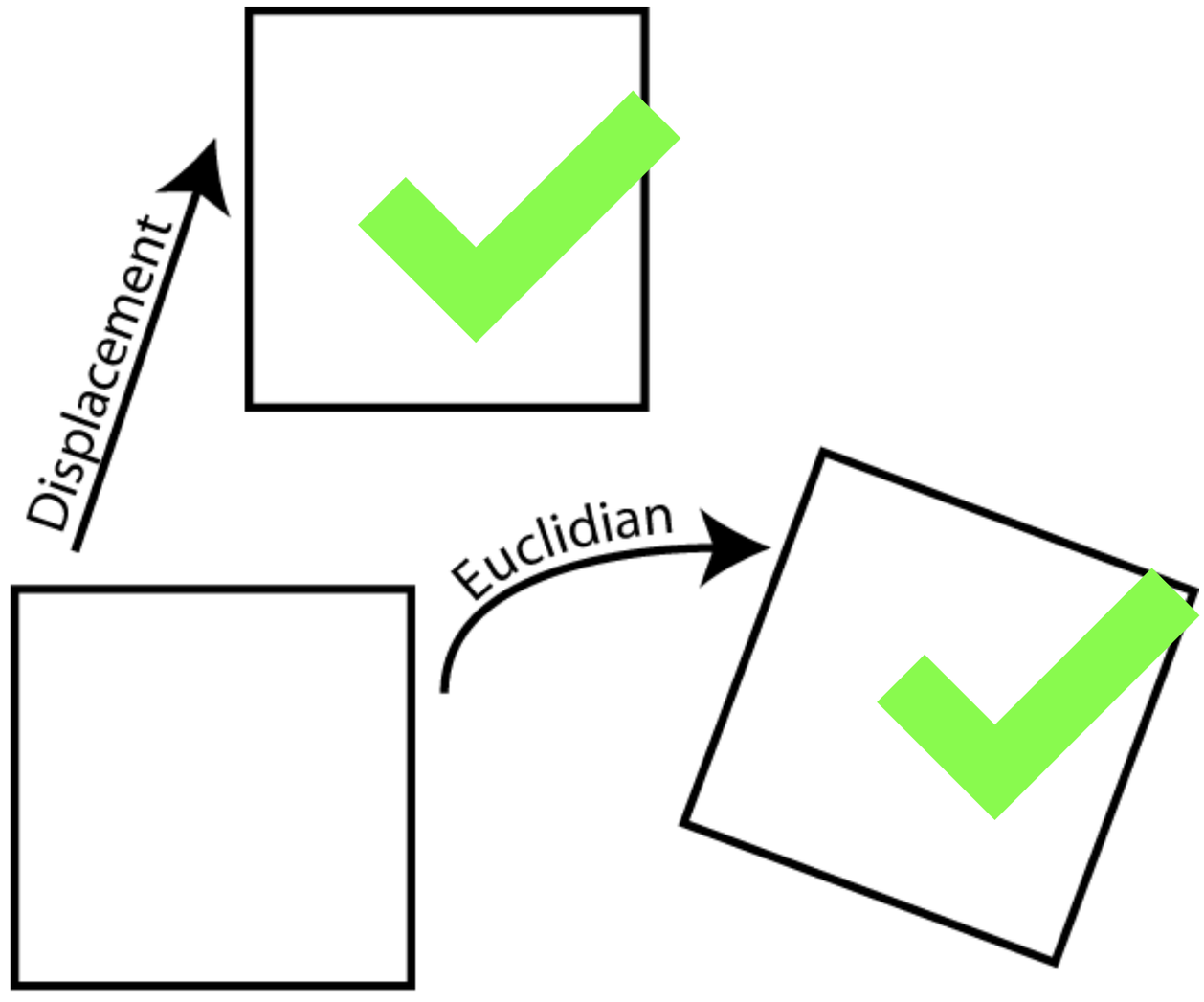
slide credit: Marc Pollefeys

Patch Transformations



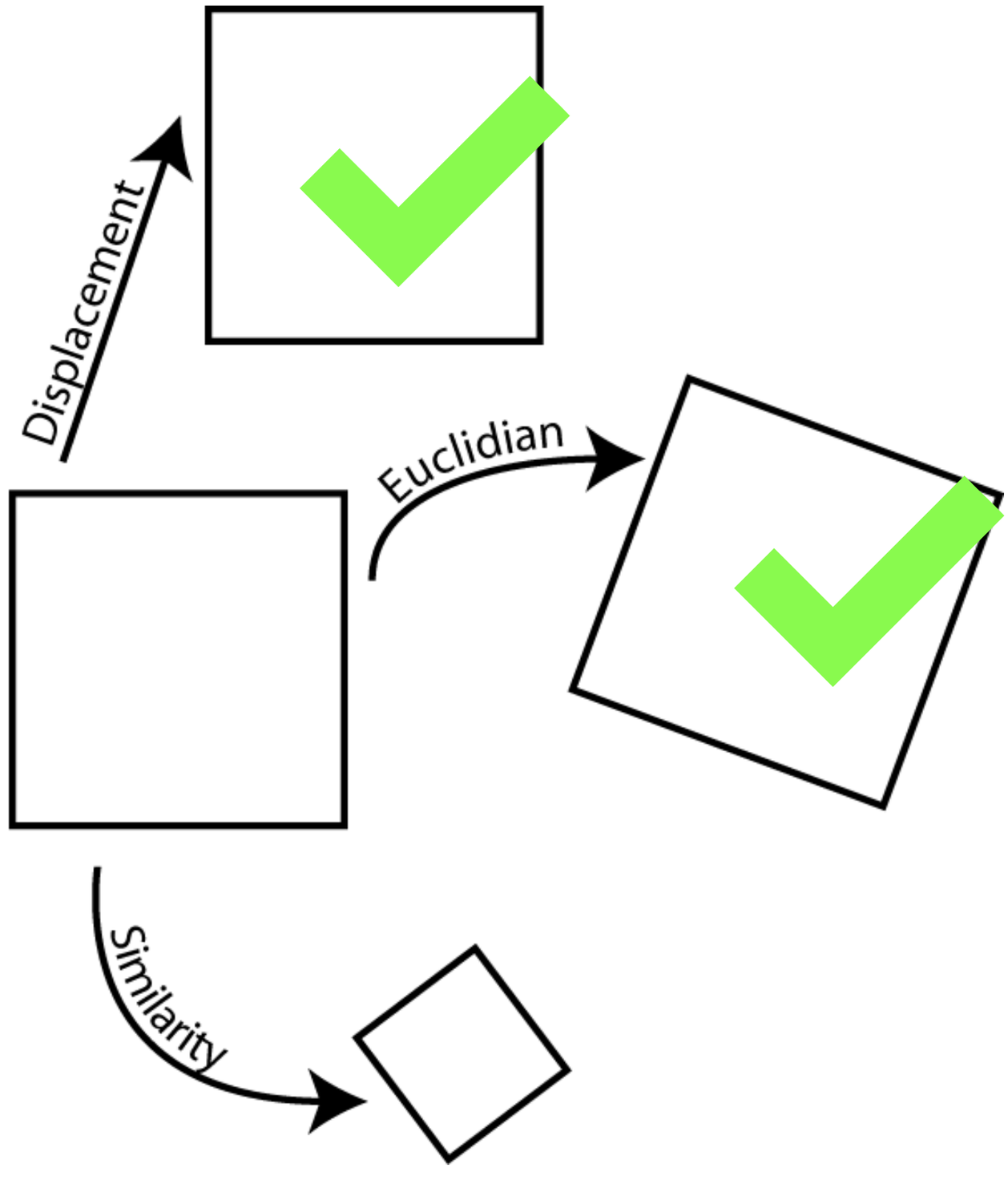
slide credit: Marc Pollefeys

Patch Transformations



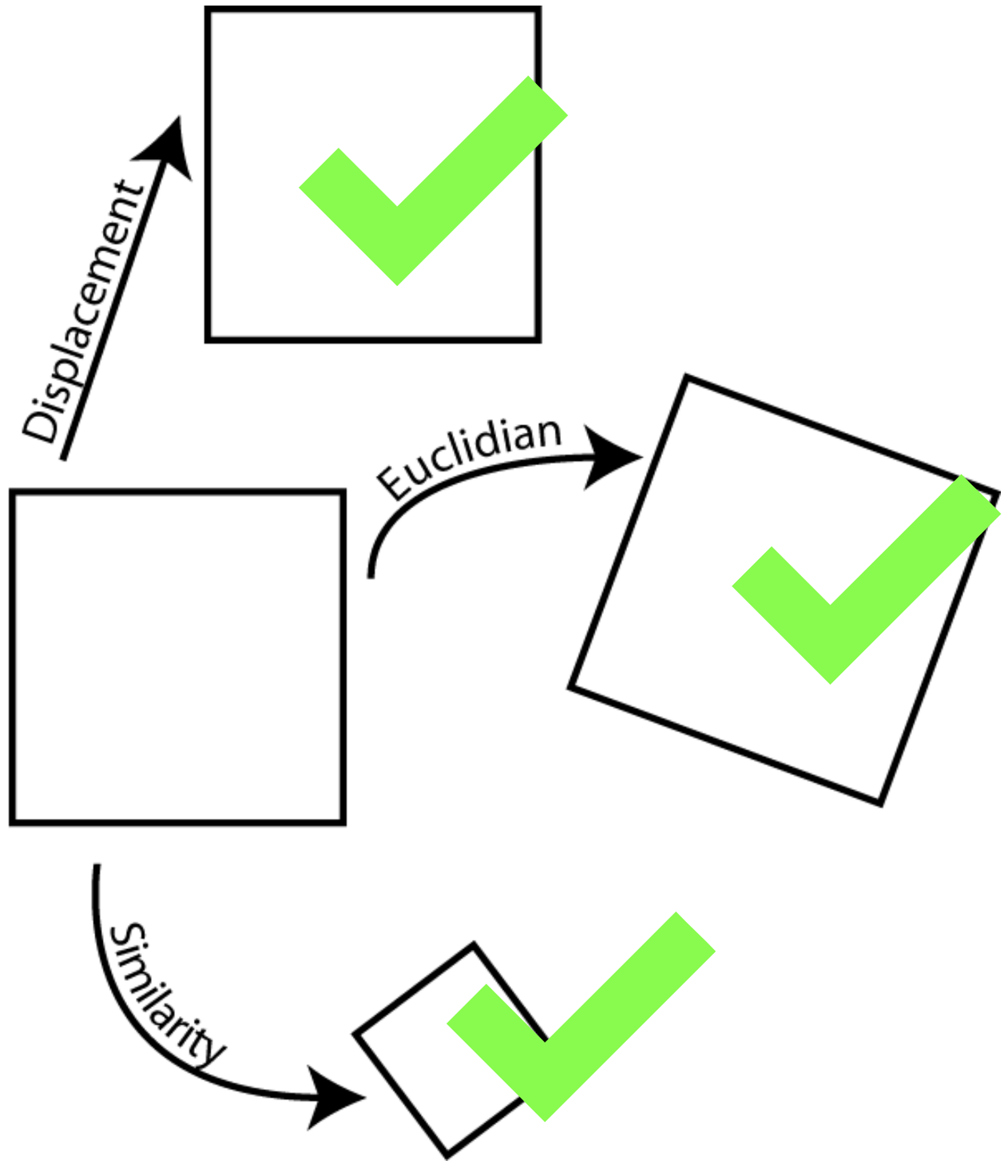
slide credit: Marc Pollefeys

Patch Transformations



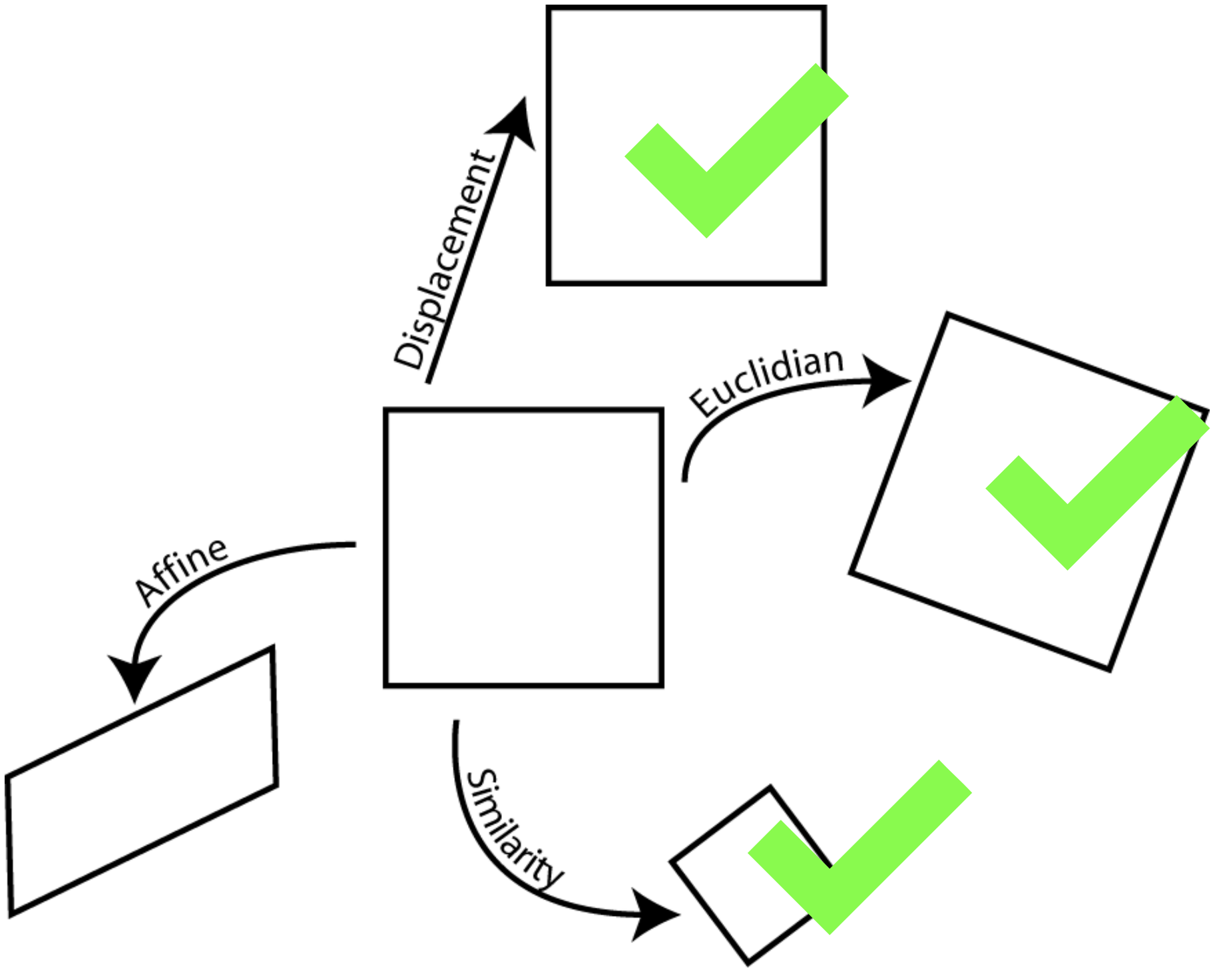
slide credit: Marc Pollefeys

Patch Transformations



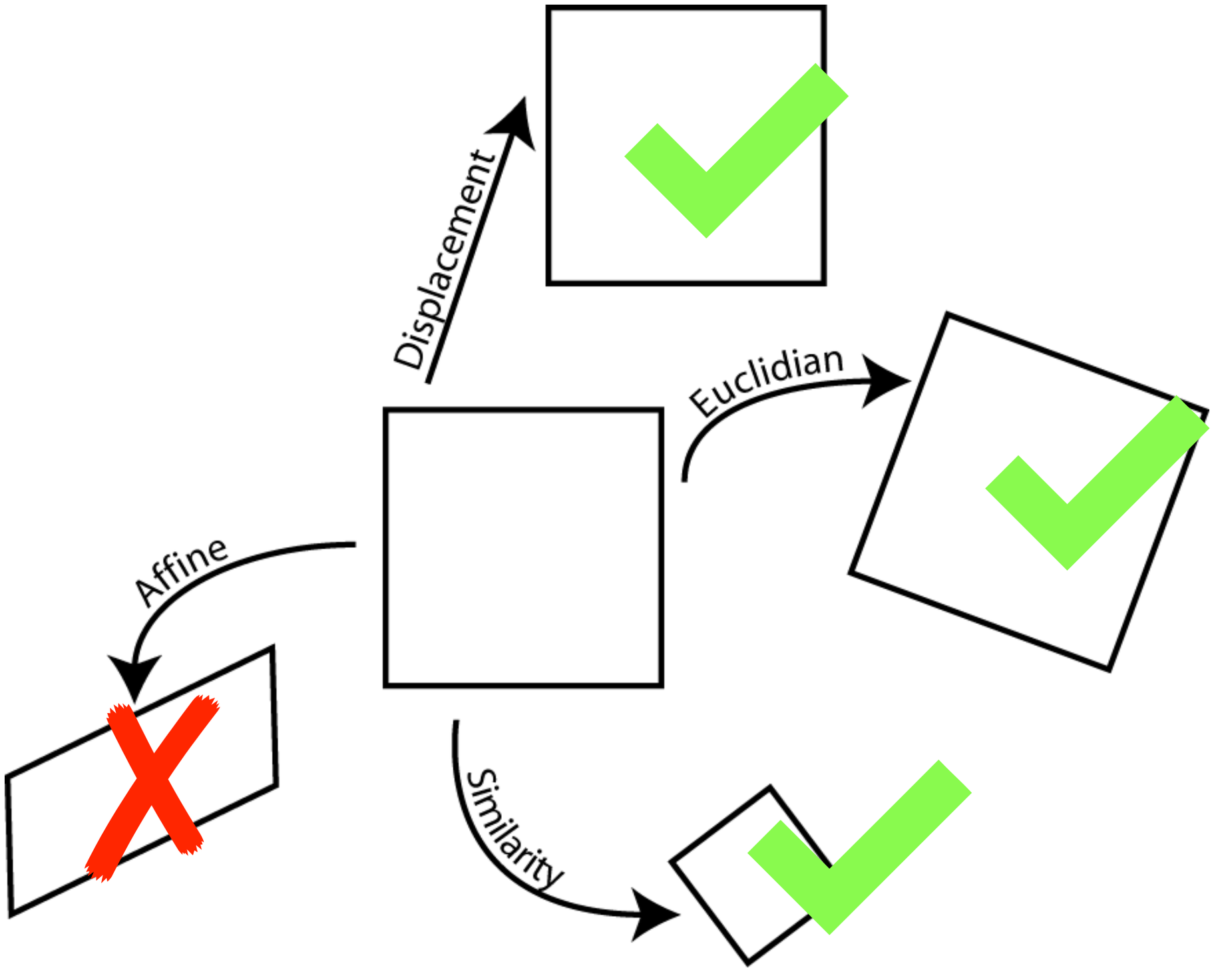
slide credit: Marc Pollefeys

Patch Transformations



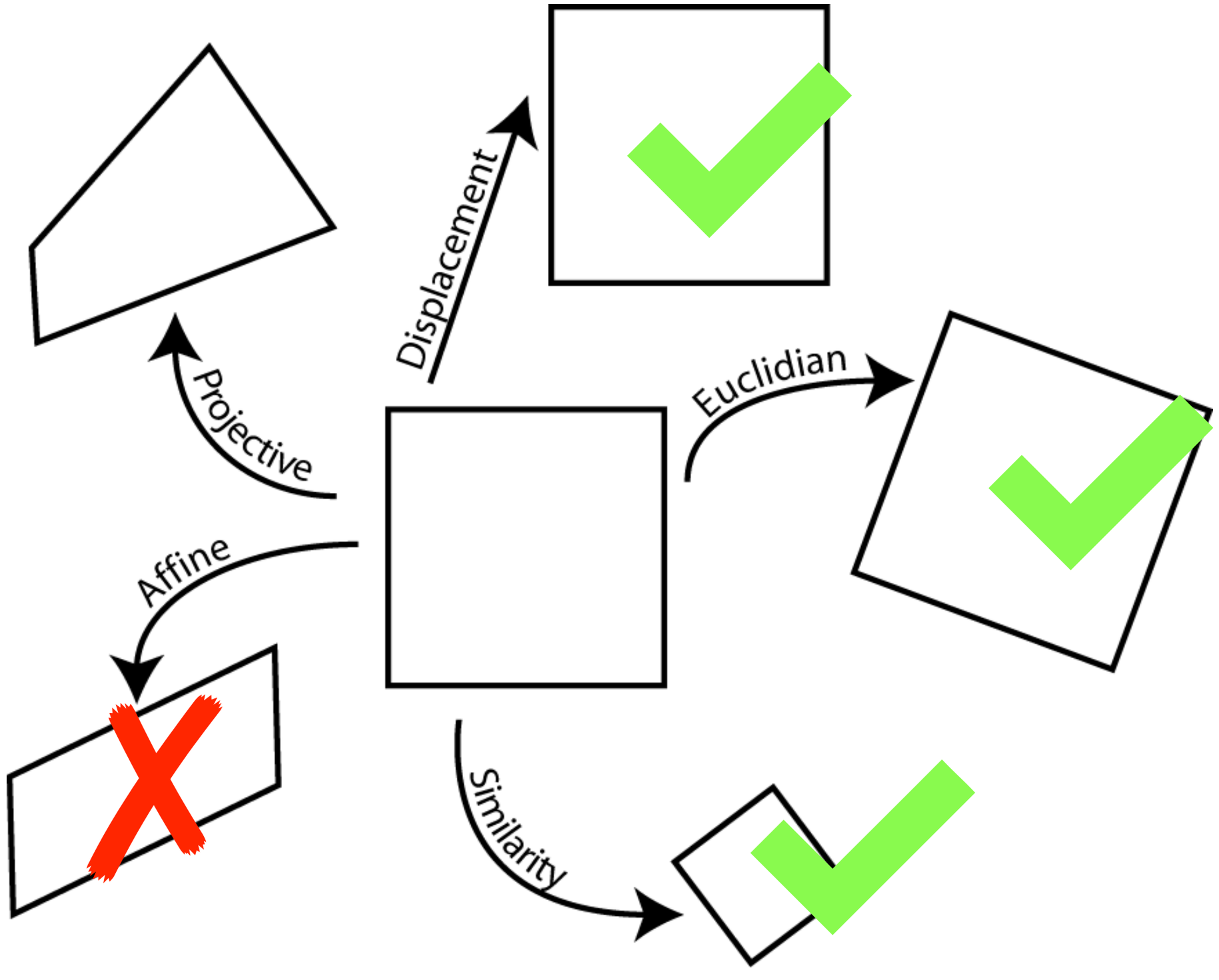
slide credit: Marc Pollefeys

Patch Transformations



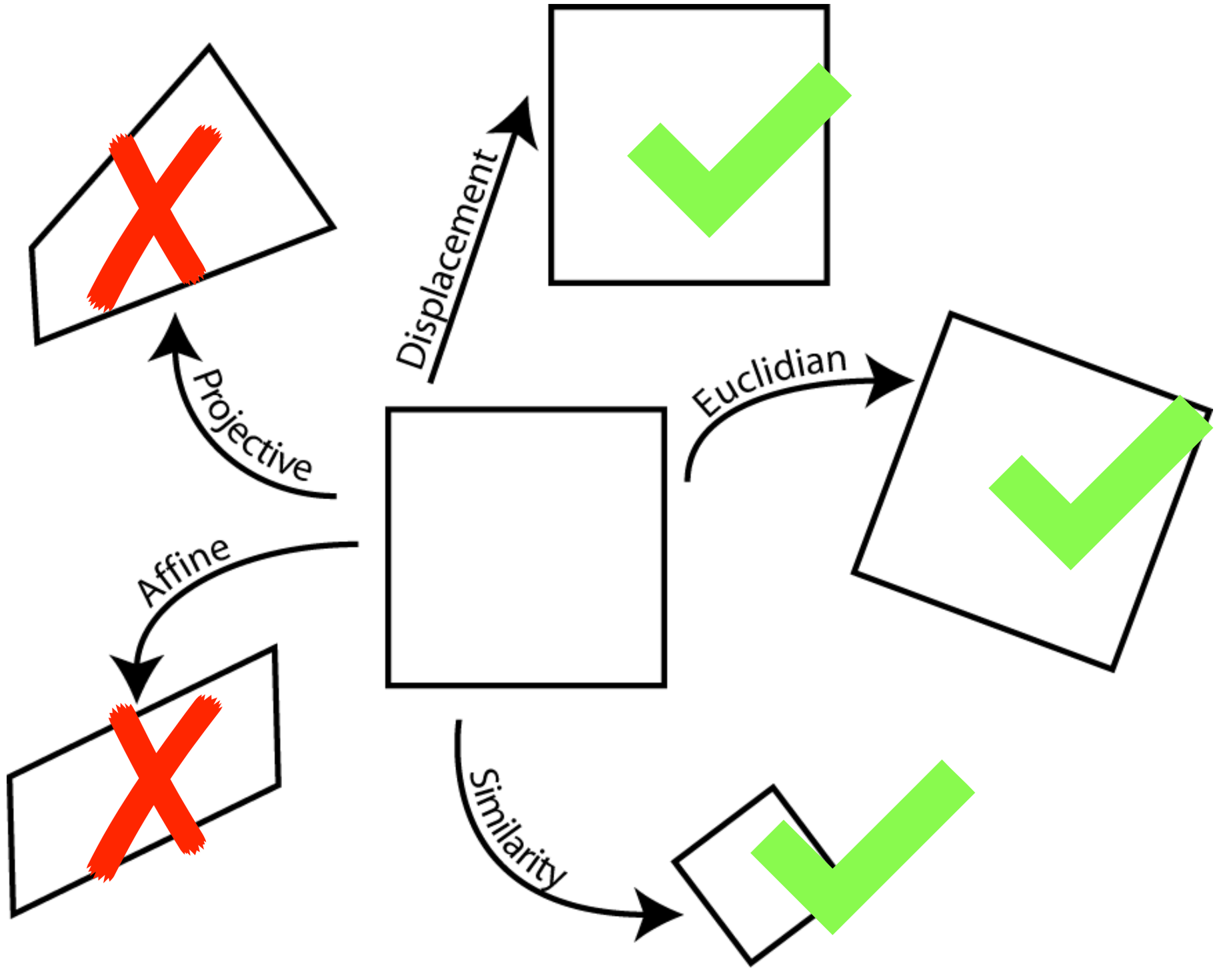
slide credit: Marc Pollefeys

Patch Transformations



slide credit: Marc Pollefeys

Patch Transformations



slide credit: Marc Pollefeys

Projective Transformations

Affine transformations can locally approximate perspective transformations quite well

slide credit: Marc Pollefeys, Kevin Köser

Affine Invariant Features

- Detect regions rather than points
- Fit ellipse into regions
- Warp ellipse to circle (= remove affine distortion)
- Compute SIFT descriptor (including orientation assignment)
- Example: MSER detector [Matas et al., Robust Wide Baseline Stereo from Maximally Stable Extremal Regions, BMVC 2002]

Maximally Stable Extremal Regions (MSER)

Extremal regions: Much brighter than their surroundings (based on thresho

slide credit: Marc Pollefeys, Kevin Köser

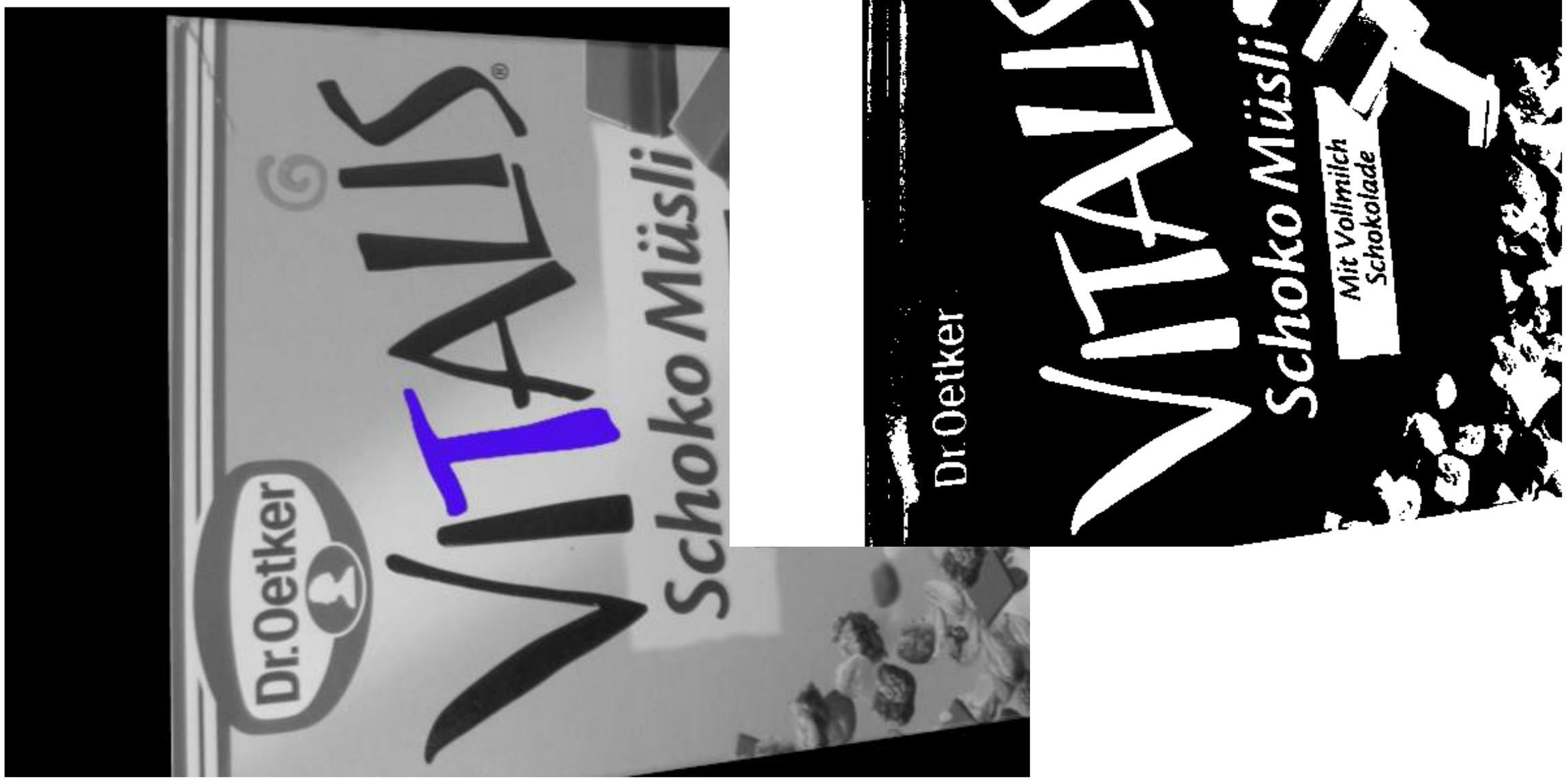
Maximally Stable Extremal Regions (MSER)

Extremal regions: Much darker than their surroundings (based on threshold

slide credit: Marc Pollefeys, Kevin Köser

Maximally Stable Extremal Regions (MSER)

Maximally stable regions: Connected components whose size does not change around a threshold



slide credit: Marc Pollefeys, Kevin Köser

From Regions to Normalized Patches

Fit ellipse into region (via PCA)

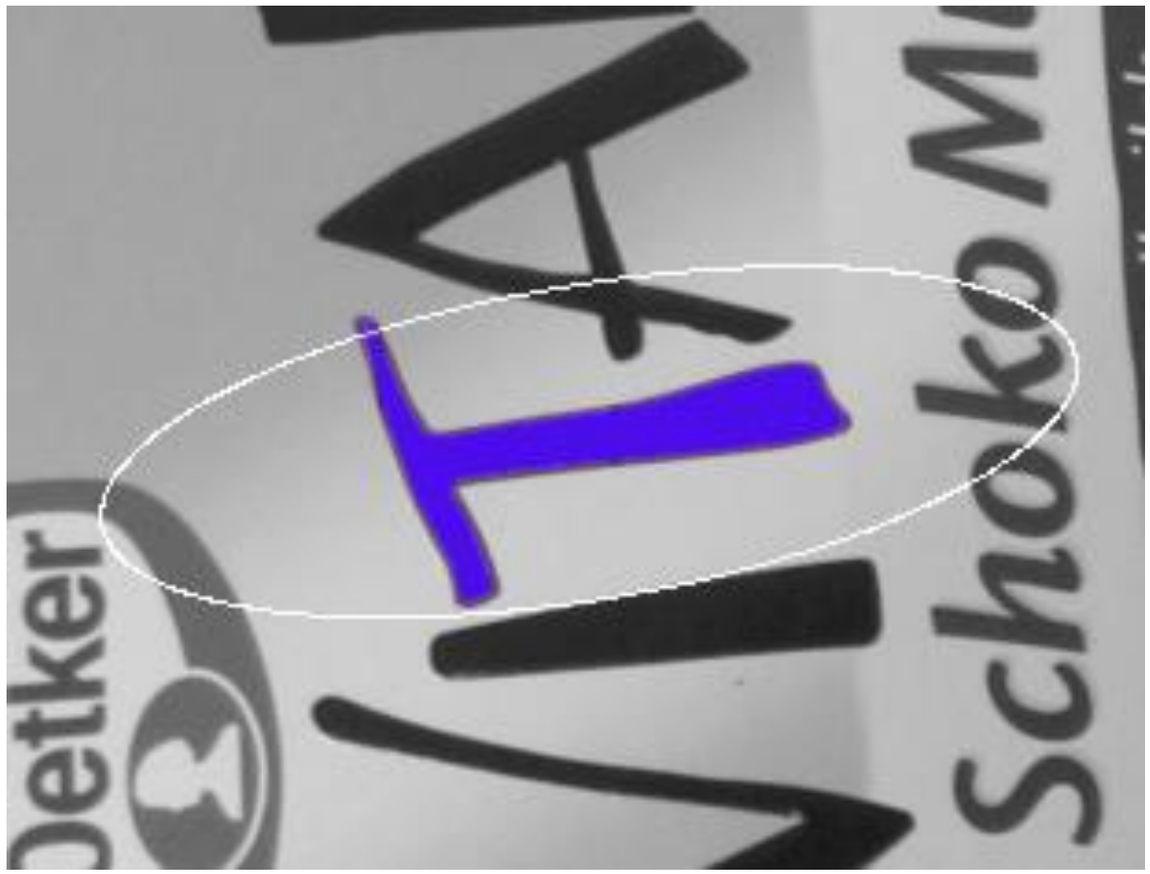
slide credit: Marc Pollefeys, Kevin Köser

From Regions to Normalized Patches

Fit ellipse into region (via PCA)

slide credit: Marc Pollefeys, Kevin Köser

From Regions to Normalized Patches



Warp ellipse to unit circle

slide credit: Marc Pollefeys, Kevin Köser

From Regions to Normalized Patches

Rotate into canonical coordinate system
based on dominant orientation

slide credit: Marc Pollefeys, Kevin Köser

Lessons Learned

- Main lessons from this lecture
  - SIFT feature detector: Difference of Gaussians over scale space
  - Extraction of oriented and scale-dependent patches
  - SIFT feature descriptor: Concatenation of gradient histograms
- Next lecture: Learning classifiers

Last Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	**Learning a classifier**	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	TBA	
Mar. 9	TBA	