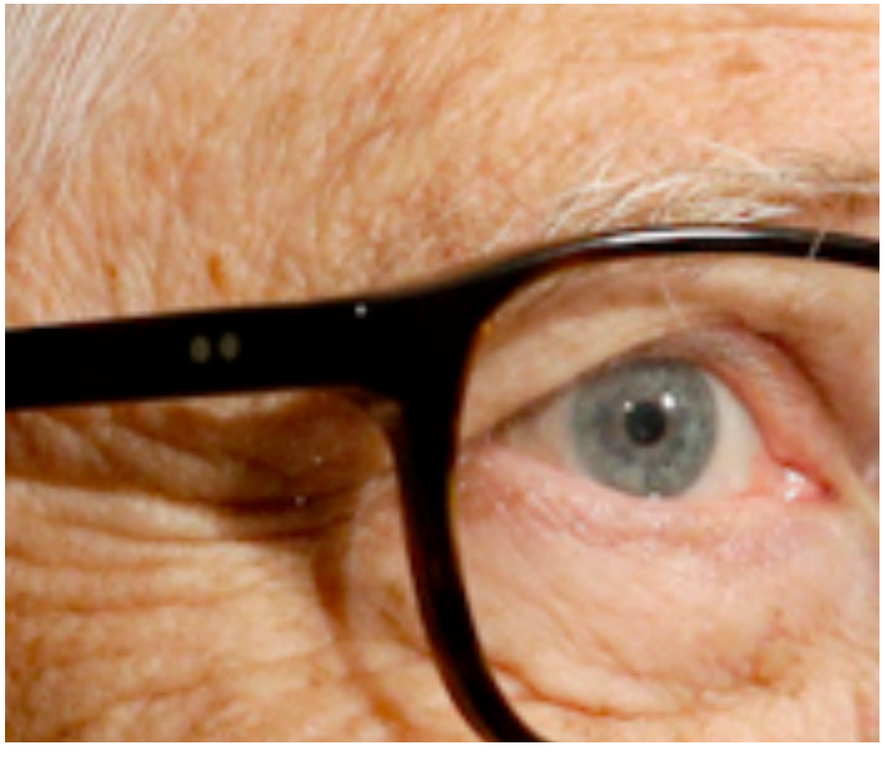
Interest Points

CS 652: Computer Vision

Features

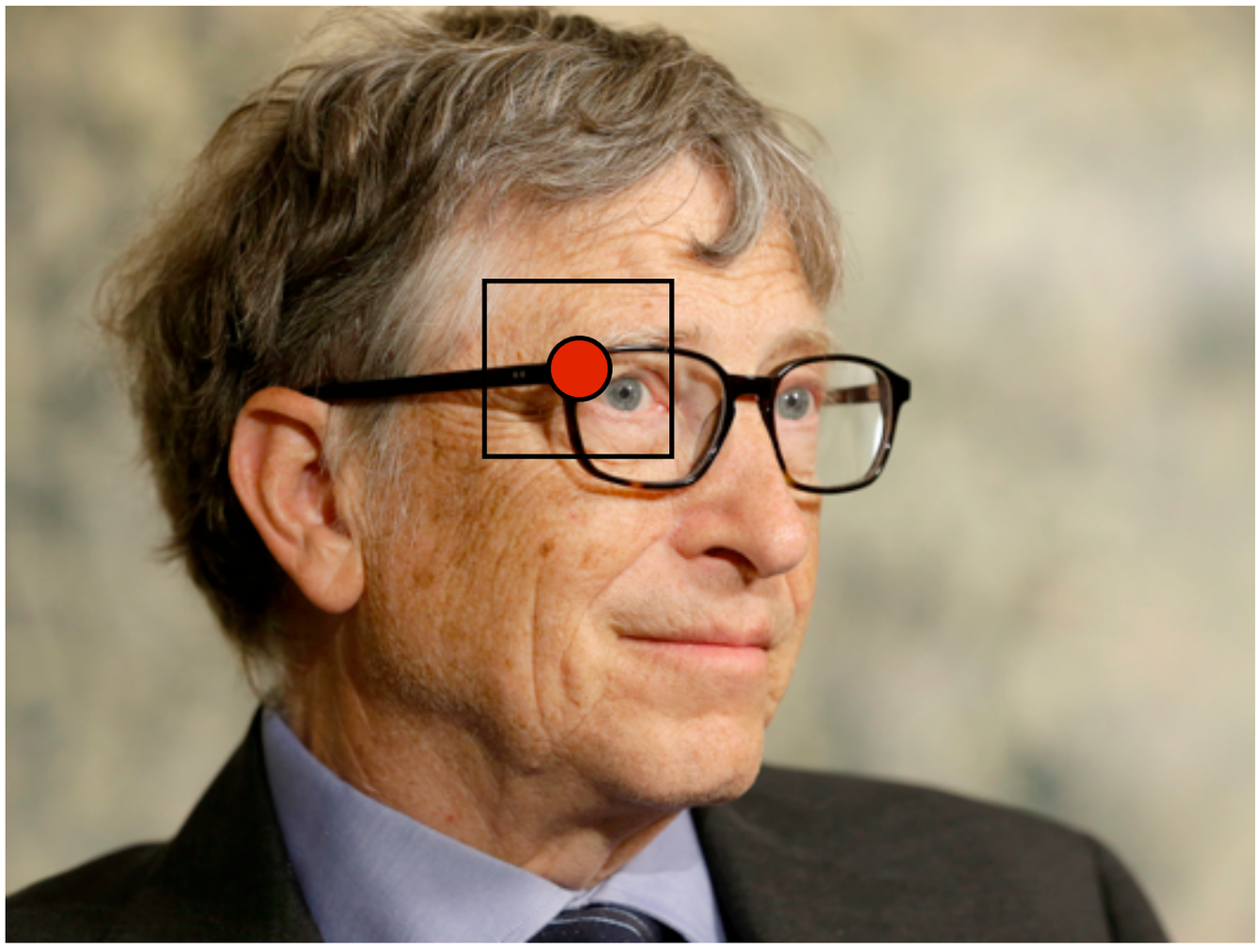
Example:
Square of size 31 x 31 around Gates's eye is a feature.
Helps to perform the task of recognizing Gates.



Features

Location of a point is a feature

Corner of the eye is a feature.



(Actually it is more like the region around the corner of the eye, as the region contains information that helps identify Gates).

Corner Points

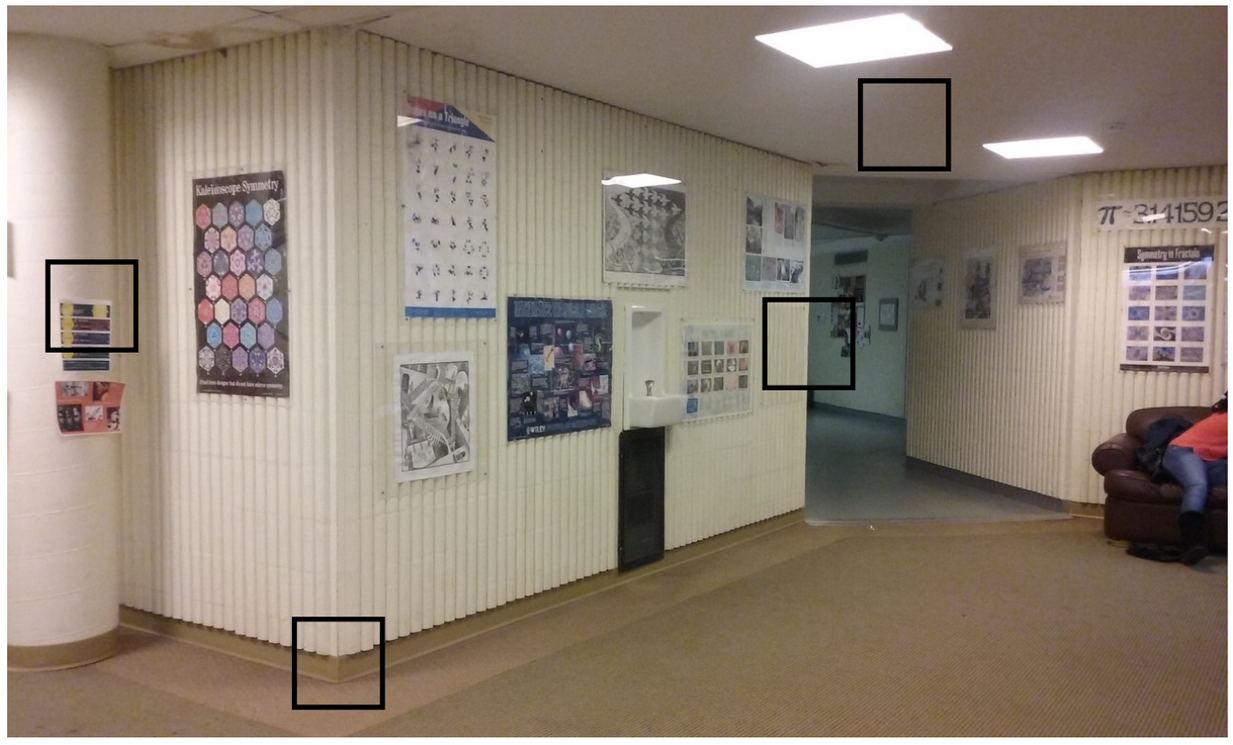
Interesting because of uniqueness

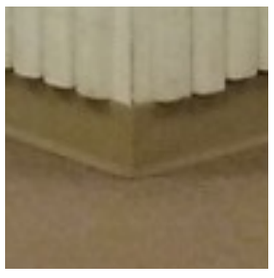
Motivation: Automated Correspondences

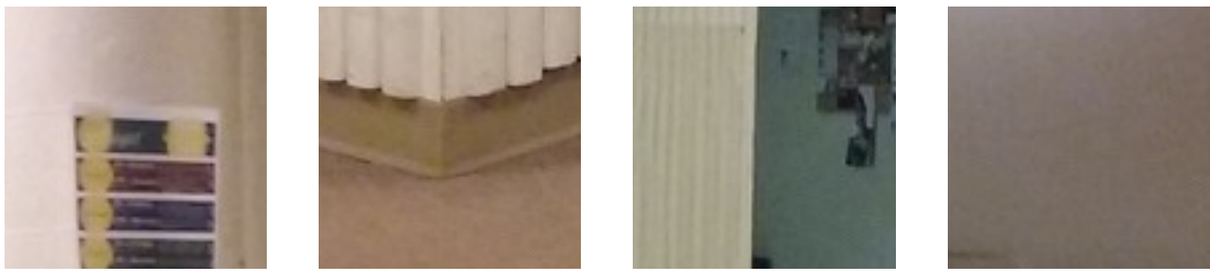
Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences





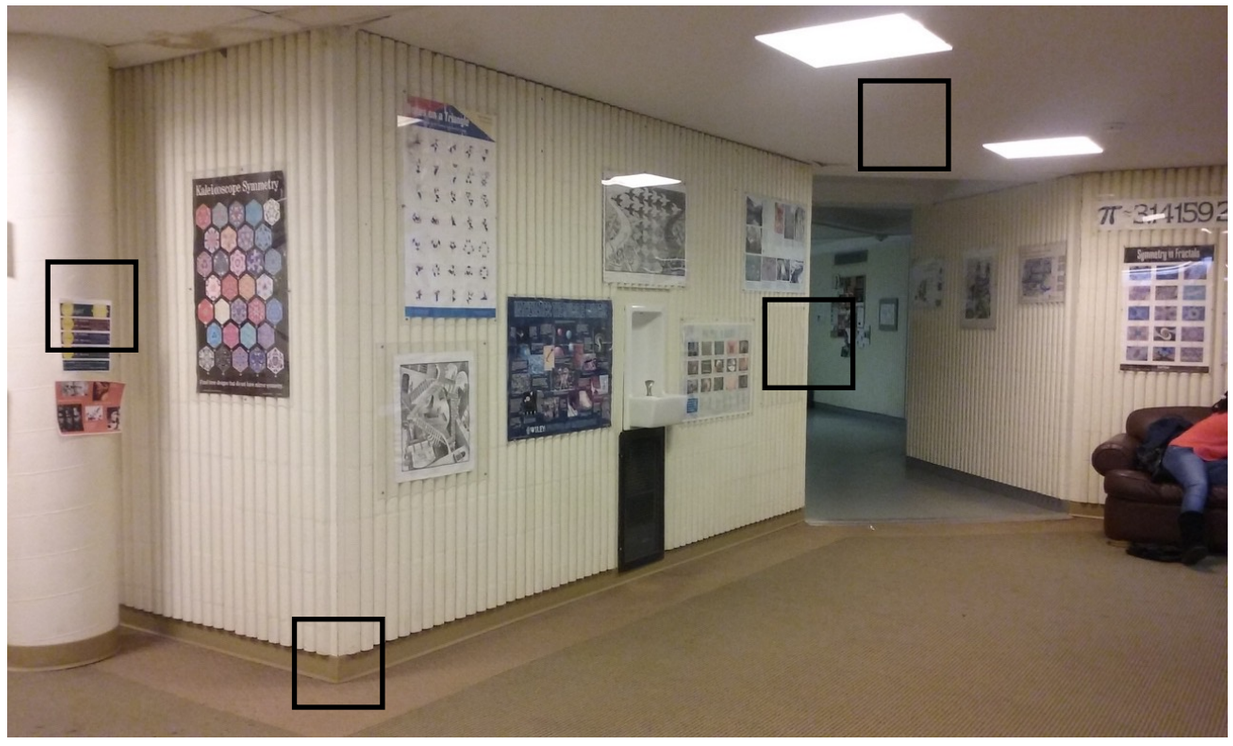


Which patch on the right matches the one on the left?

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences

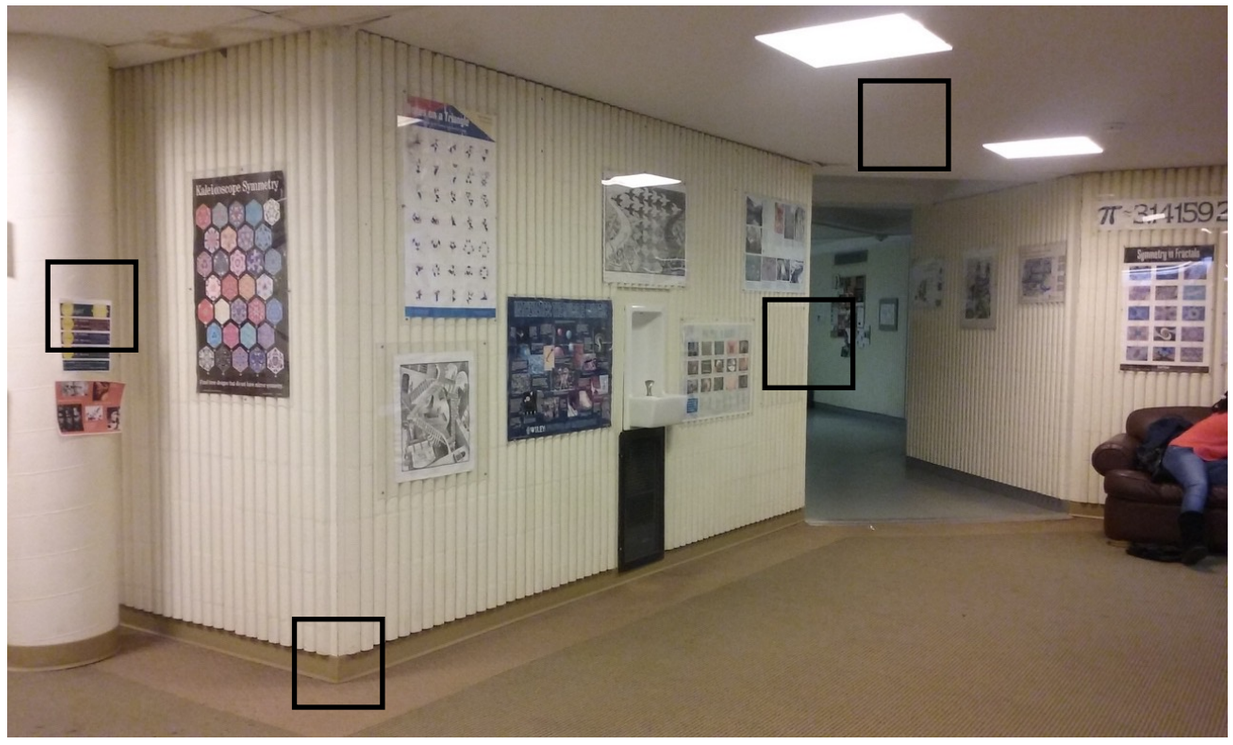


Which patch on the right matches the one on the left?

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences

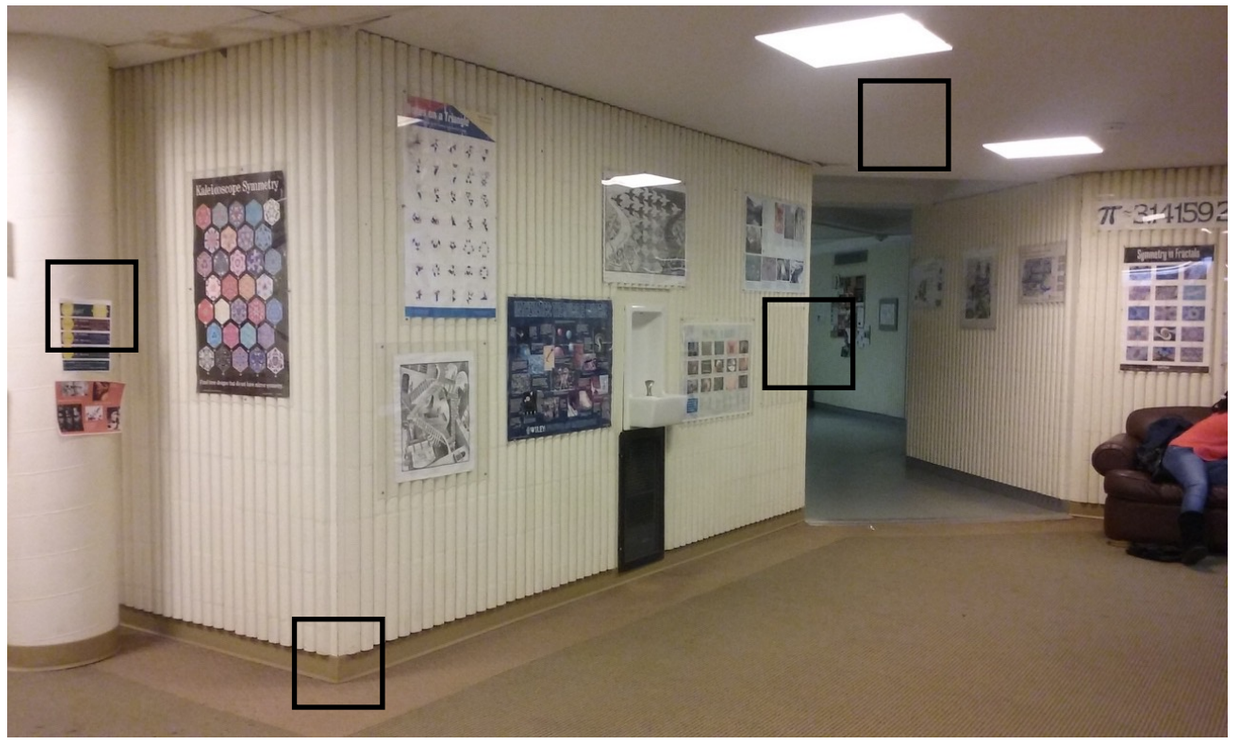


How can you match the patches?

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences

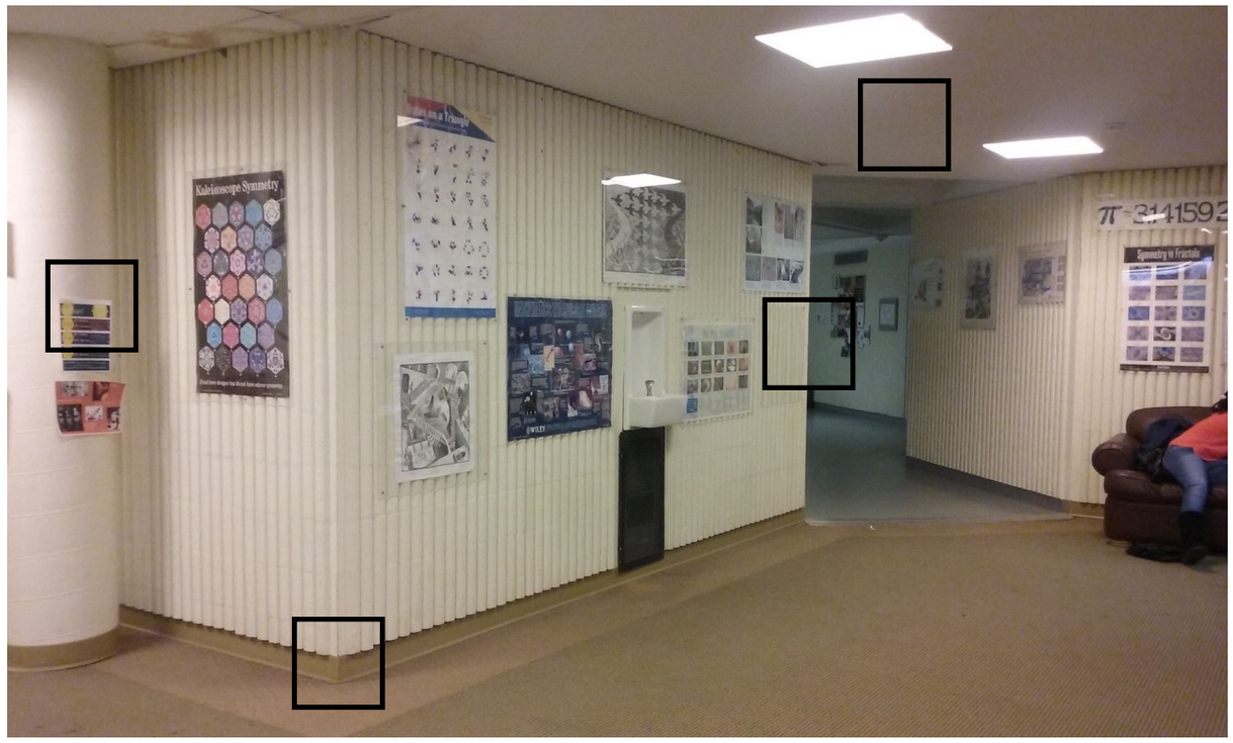


SSD (if light intensity is similar), Normalized cross-correlation.

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences



SSD (if light intensity is similar), Normalized cross-correlation.

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences

What if I want to match this patch (uniform)?

Corner Points

Interesting because of uniqueness

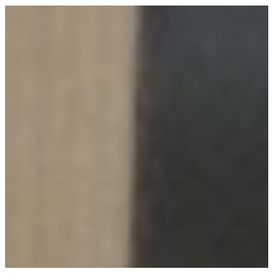
Motivation: Automated Correspondences

It has many matches in the second image!

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences

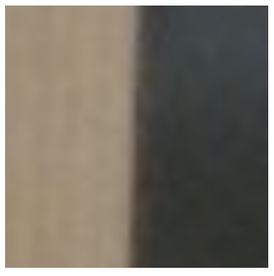


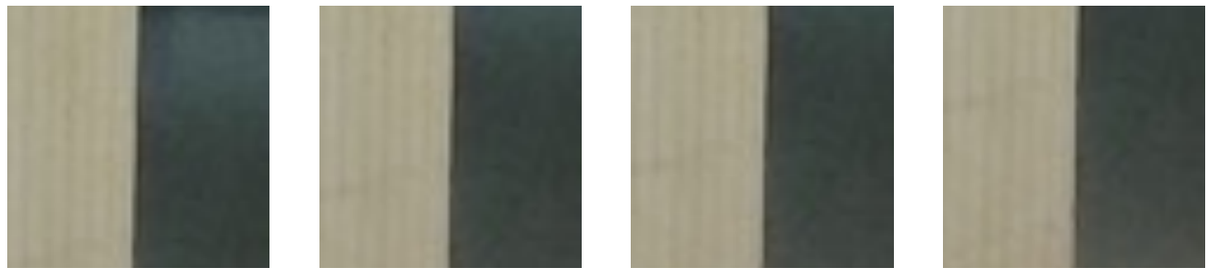
What about this patch (edge)?

Corner Points

Interesting because of uniqueness

Motivation: Automated Correspondences



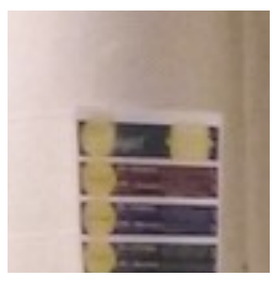


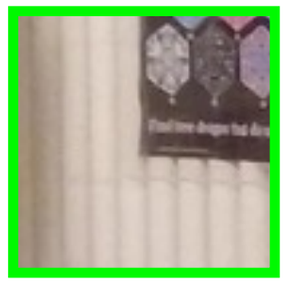
Still not unique. Could match anywhere along the edge.

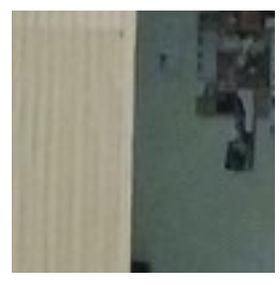
Corner Points

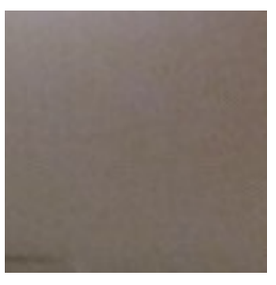
Interesting because of uniqueness

Motivation: Automated Correspondences









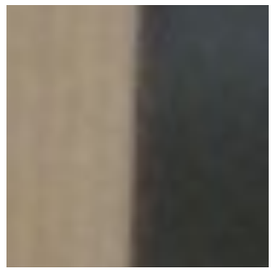
Corner patch has only one match!

Corner Points

Interesting because of uniqueness

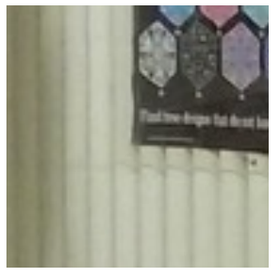
Uniform Patch

Can match anywhere in uniform region.



Edge Patch

Can match anywhere along edge.



Corner Patch

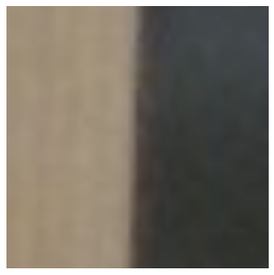
Matches only at one corner.

Corner Points

Interesting because of uniqueness

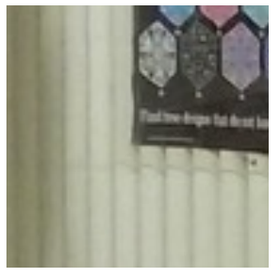
Uniform Patch

Can match anywhere in uniform region.



Edge Patch

Can match anywhere along edge.

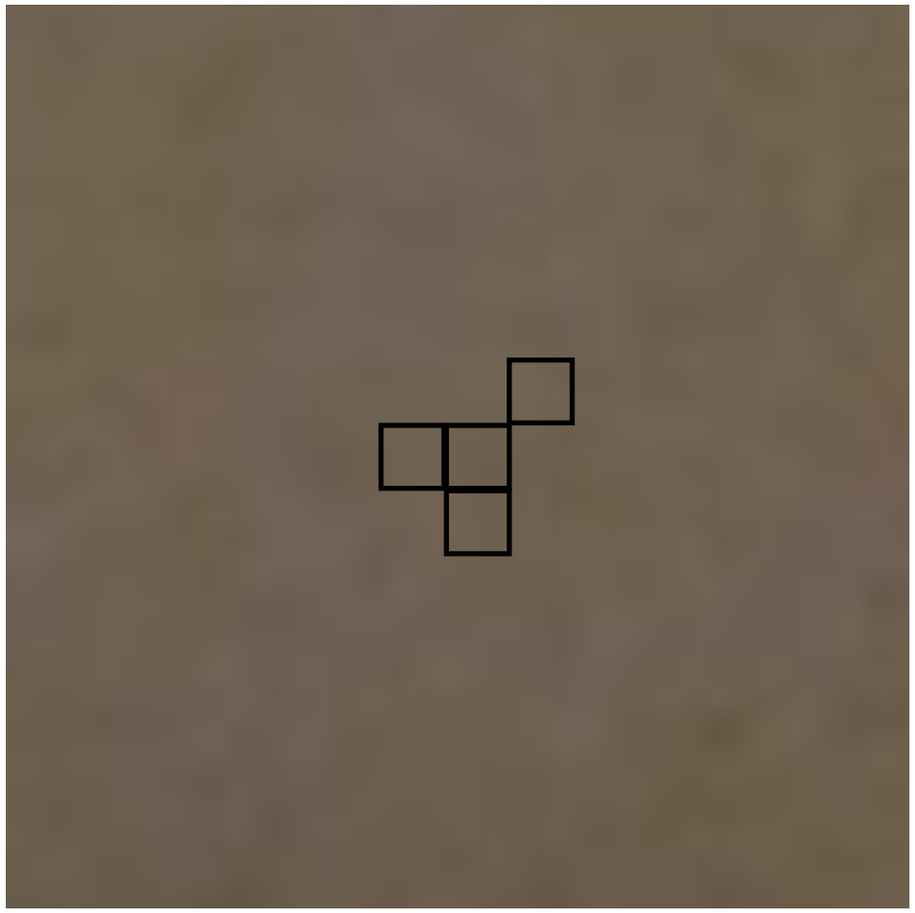


Corner Patch

Matches only at corner.

Let us find corners in an image!

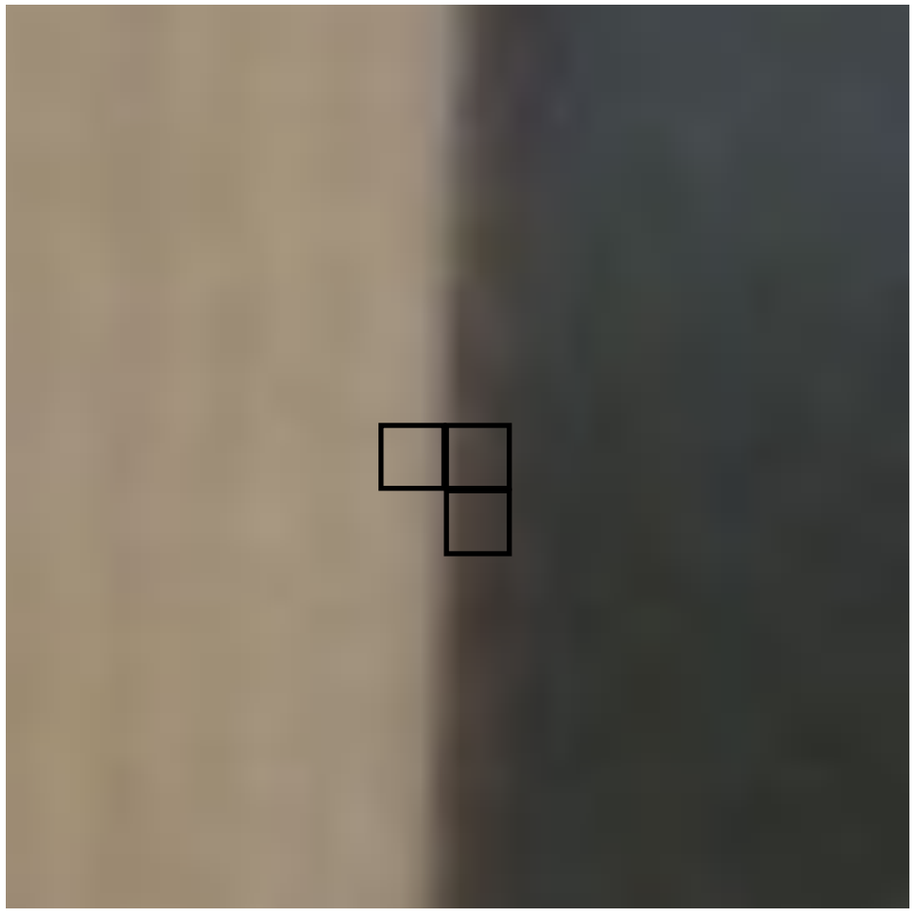
What happens at a corner?



Uniform Patch

No change in intensity in any direction.

What happens at a corner?



Edge Patch

Intensity changes across the edge.

Intensity does not change **along** the edge.

What happens at a corner?

Corner Patch

Intensity changes in all directions.

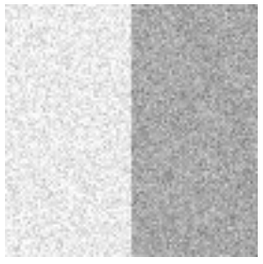
Harris Corner Detector

Mathematical approach to measure
directional change in intensity

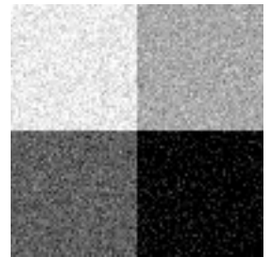
$\mathbf{I}$

Uniform

Edge



Corner



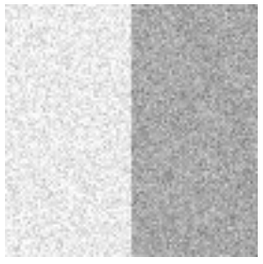
Harris Corner Detector

How can we measure **change** in intensity?

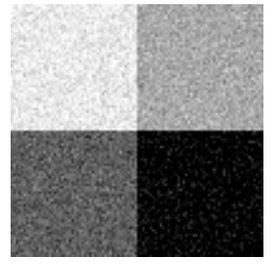
$\mathbf{I}$

Uniform

Edge



Corner



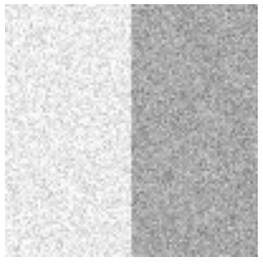
Harris Corner Detector

Find the gradients of the image!

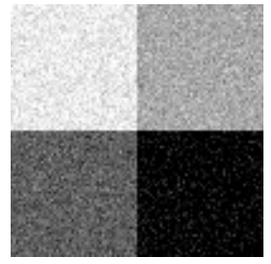
$\mathbf{I}$

Uniform

Edge

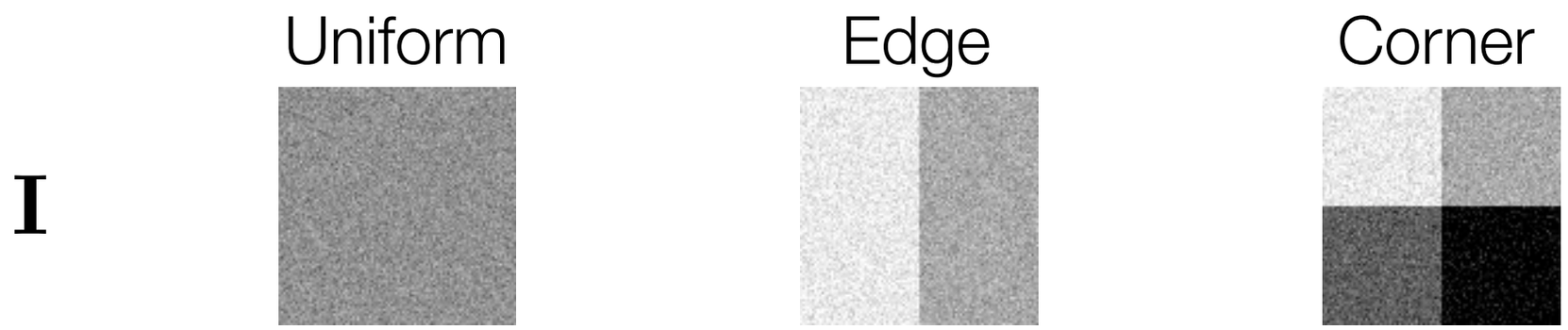


Corner



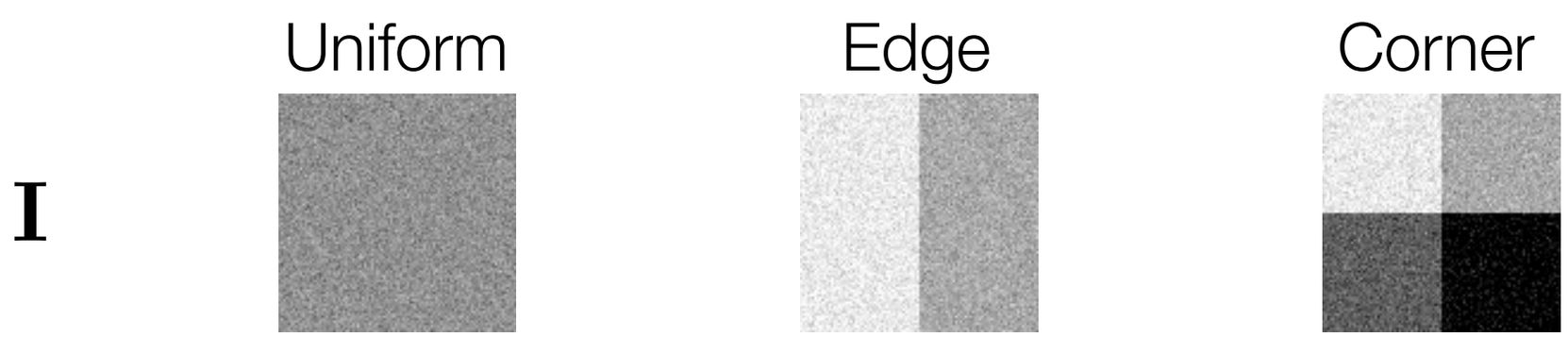
Harris Corner Detector

First Derivative in x using Sobel filter in x



Harris Corner Detector

First Derivative in *y* using Sobel filter in *y*

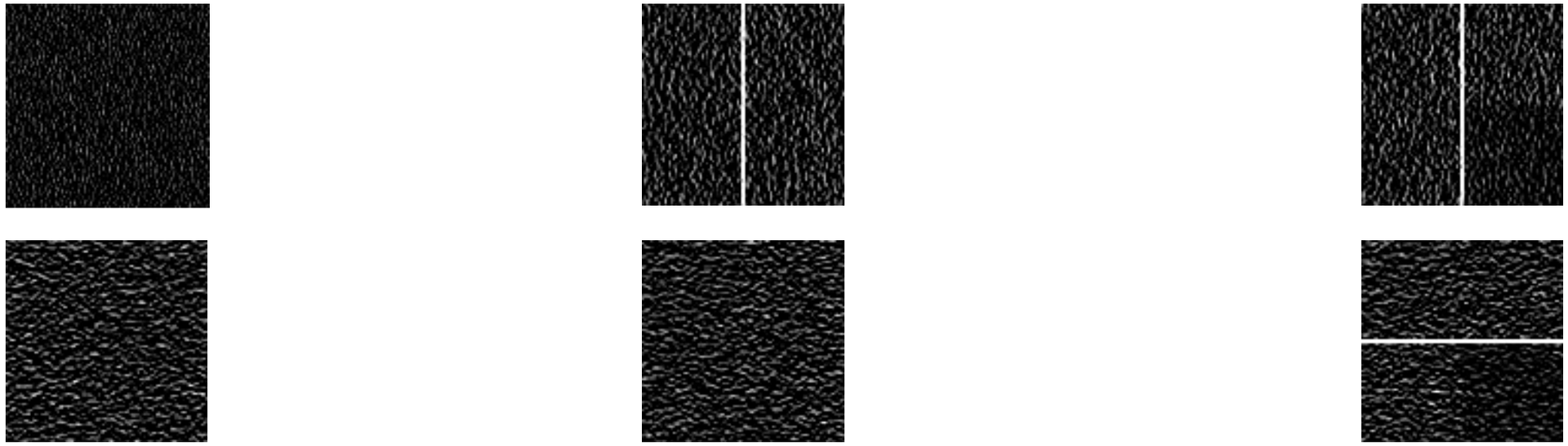


Harris Corner Detector

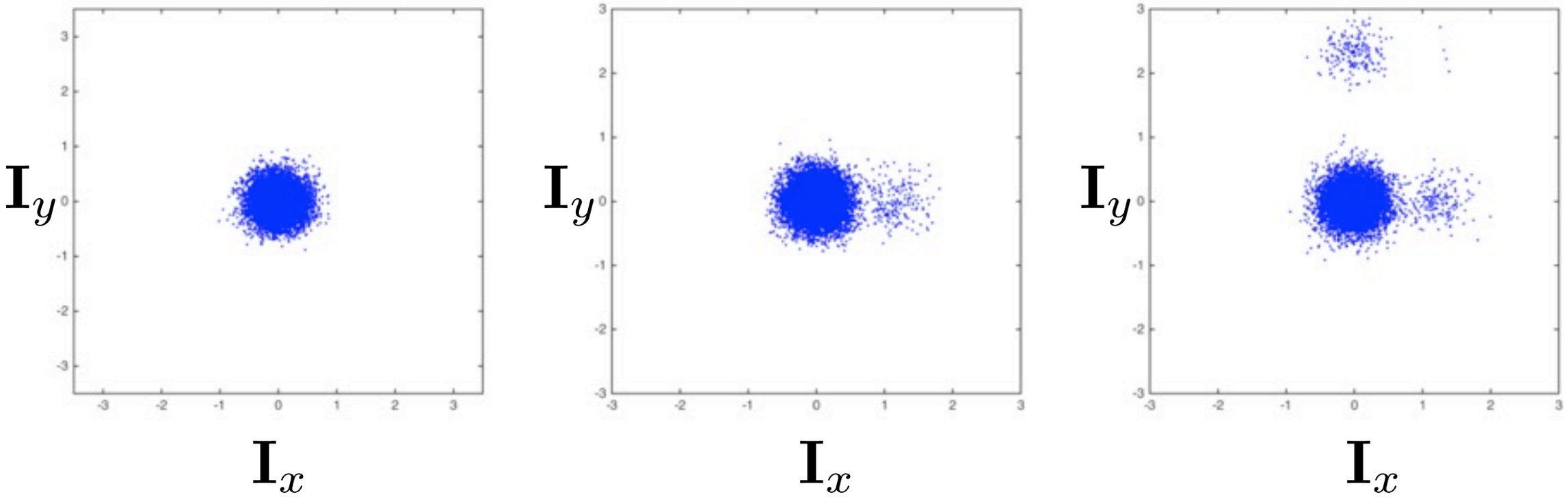
Plot pixel count for first derivative in *x* and *y*

$\mathbf{I}_x$

$\mathbf{I}_y$

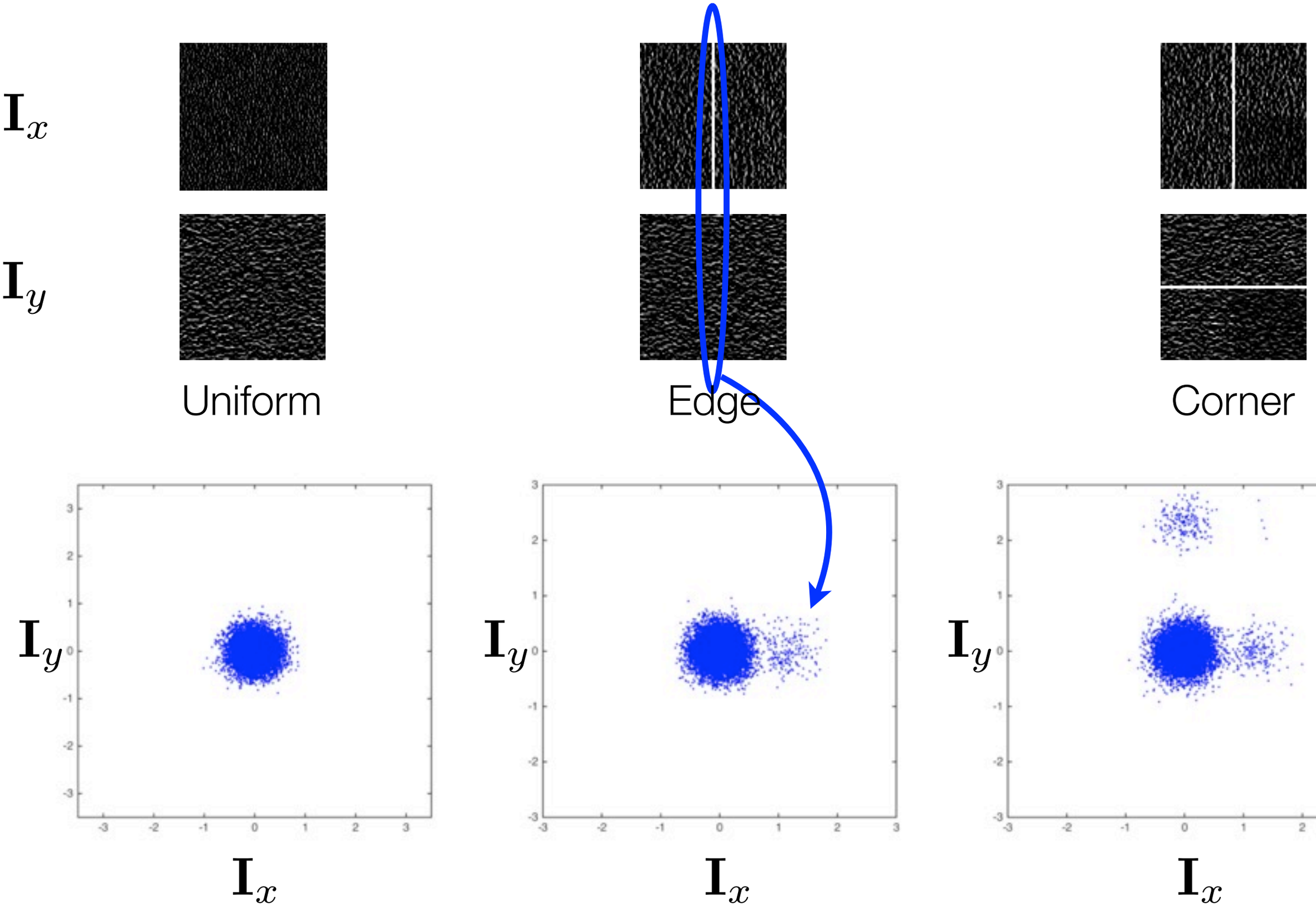


Uniform | Edge | Corner



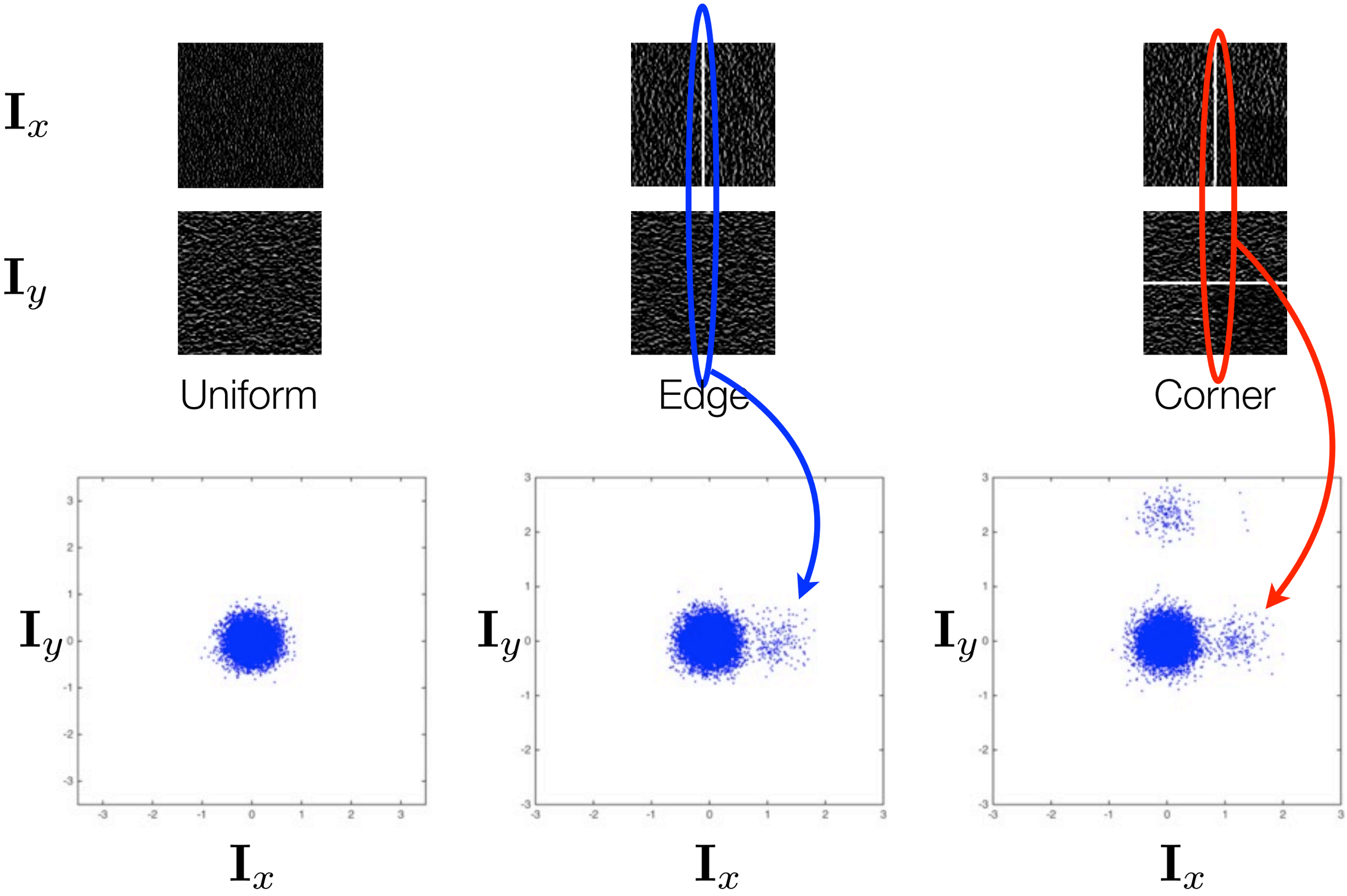
Harris Corner Detector

Plot pixel count for first derivative in *x* and *y*



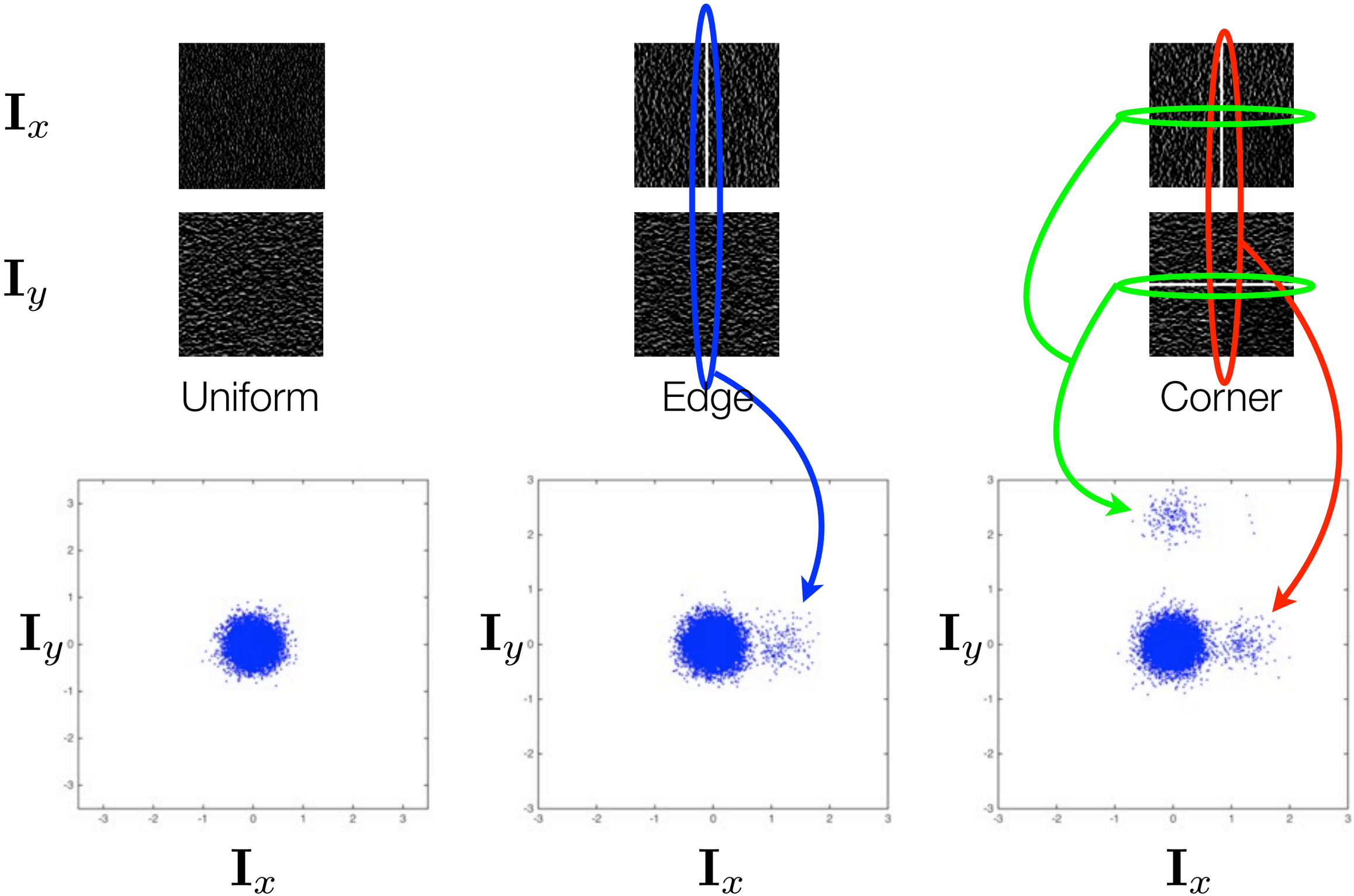
Harris Corner Detector

Plot pixel count for first derivative in *x* and *y*



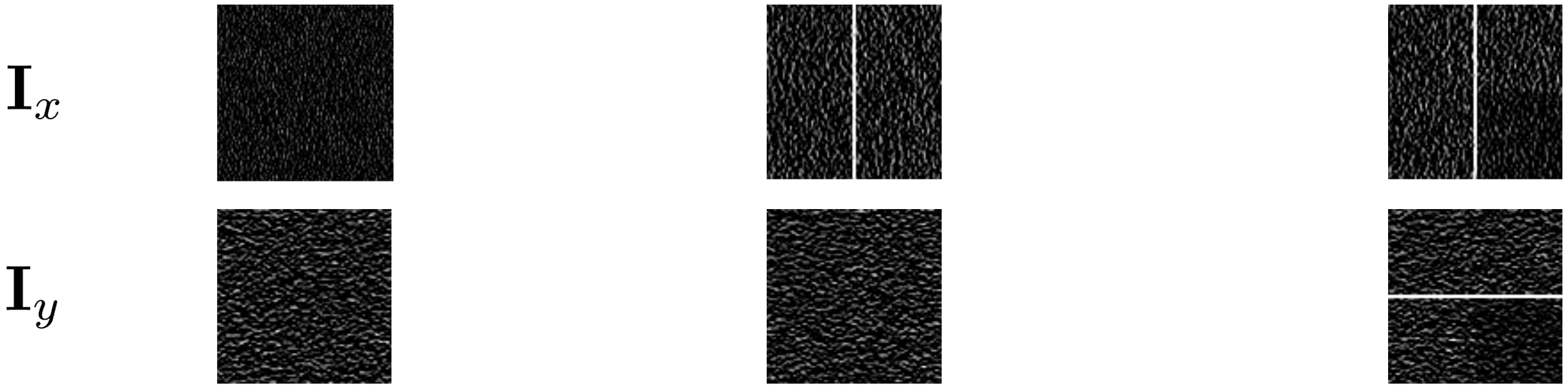
Harris Corner Detector

Plot pixel count for first derivative in *x* and *y*



Harris Corner Detector

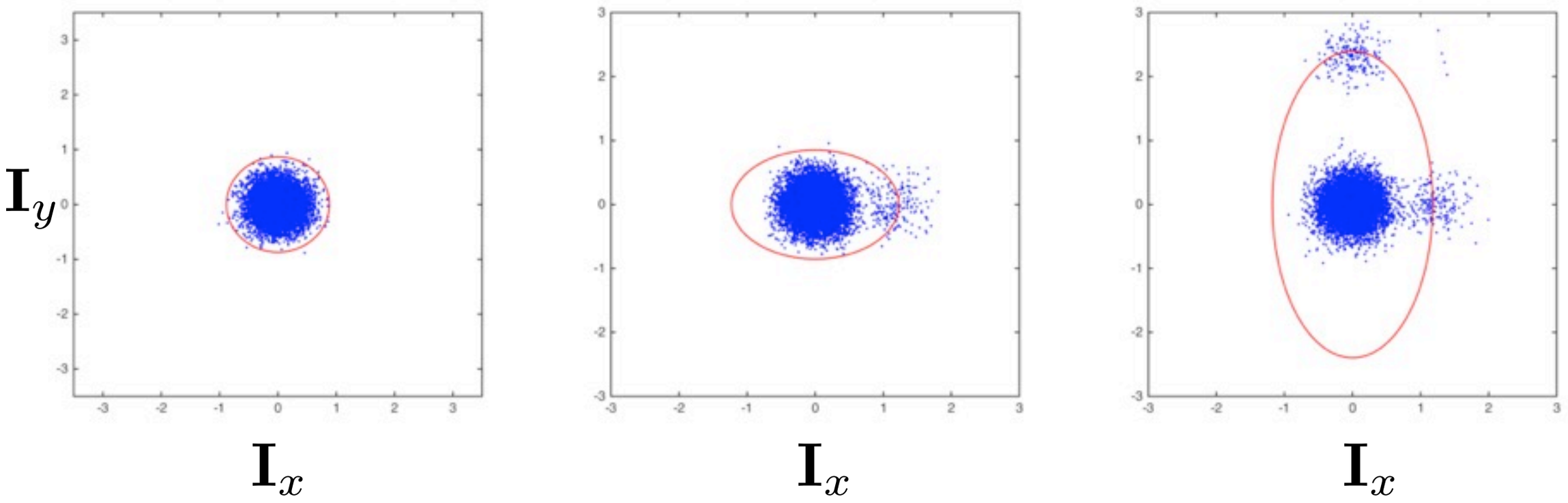
Plotted points fall within an ellipse



Uniform

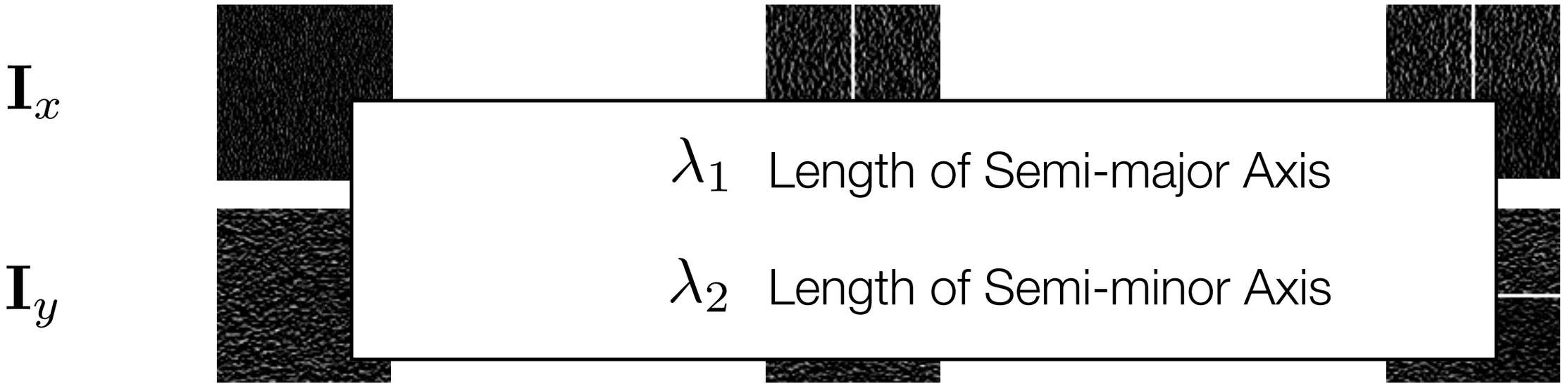
Edge

Corner

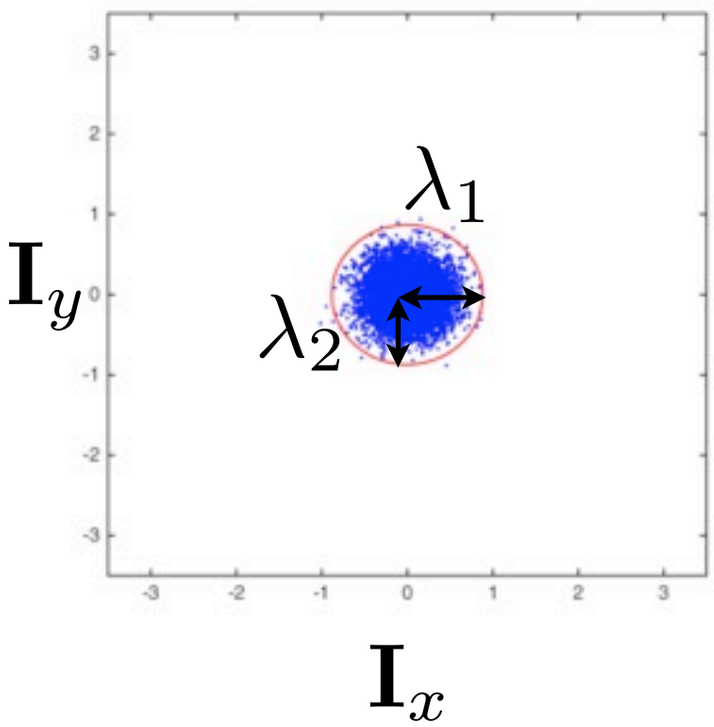


Harris Corner Detector

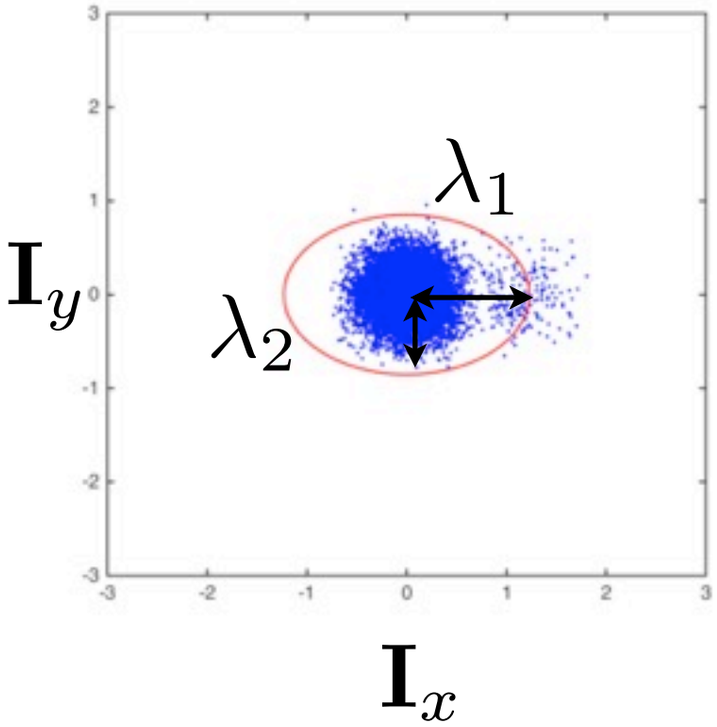
Plotted points fall within an ellipse



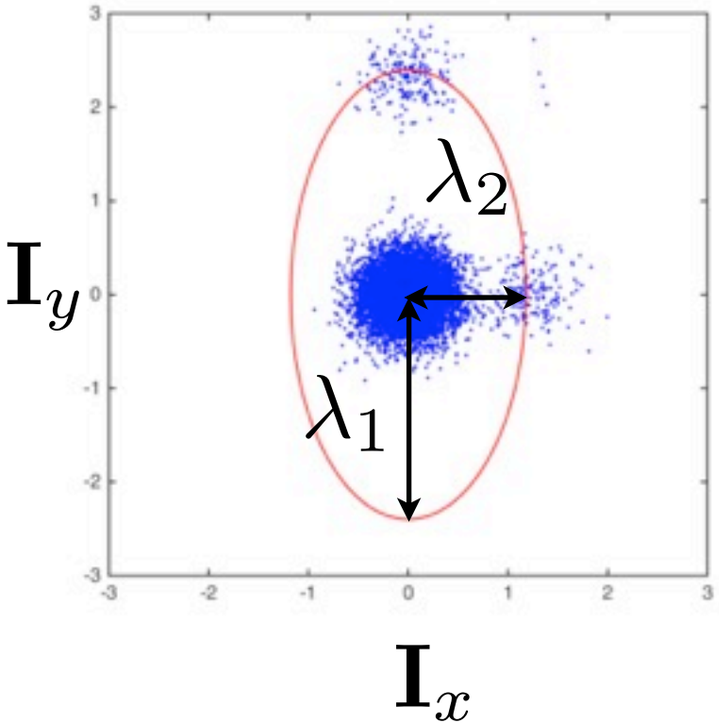
Uniform



Edge

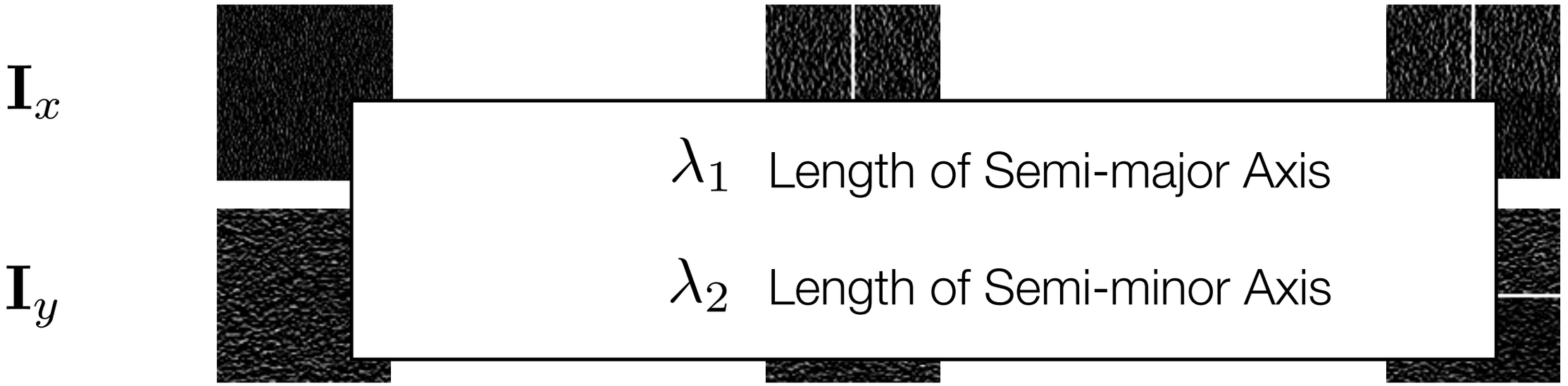


Corner



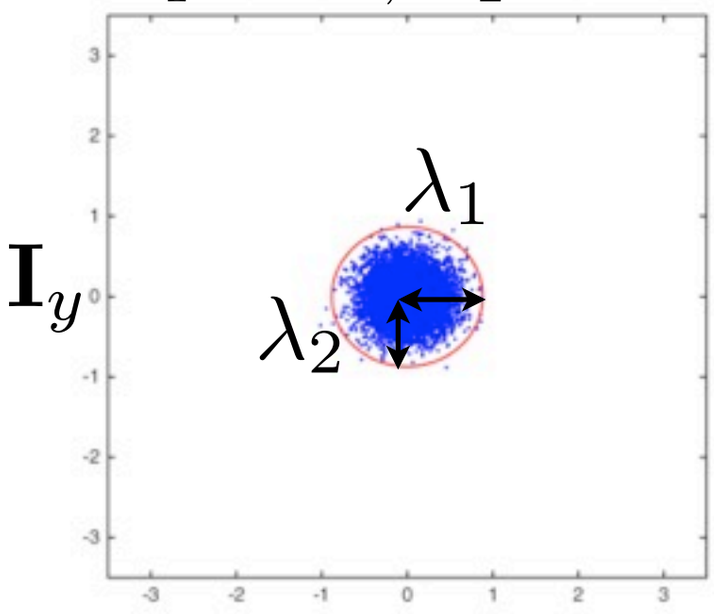
Harris Corner Detector

Plotted points fall within an ellipse



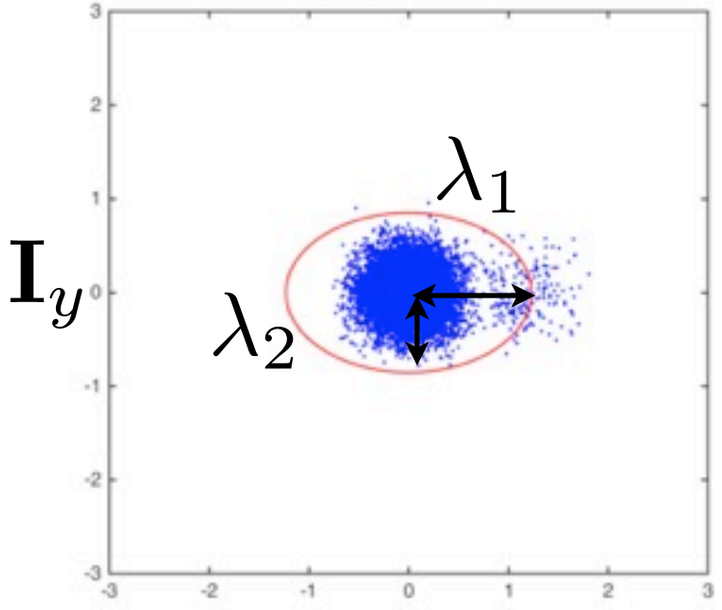
Uniform

λ_1 small, λ_2 small



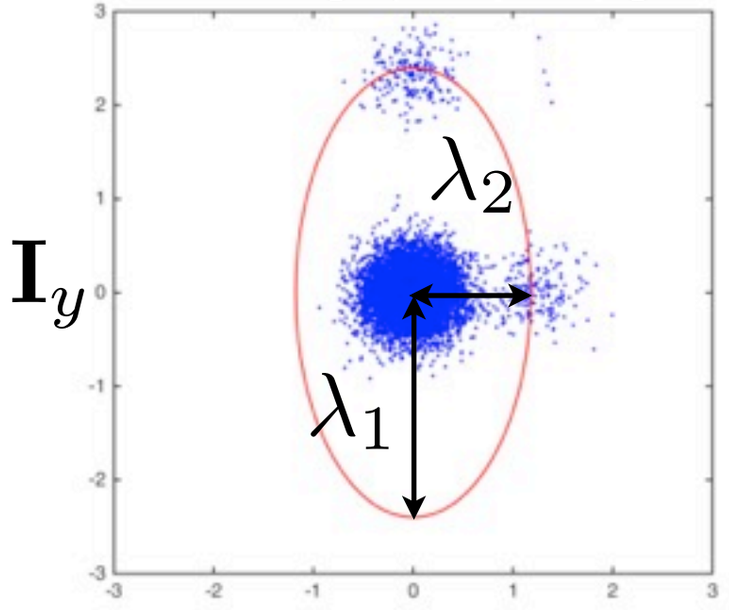
Edge

λ_1 large, λ_2 small



Corner

λ_1 large, λ_2 large

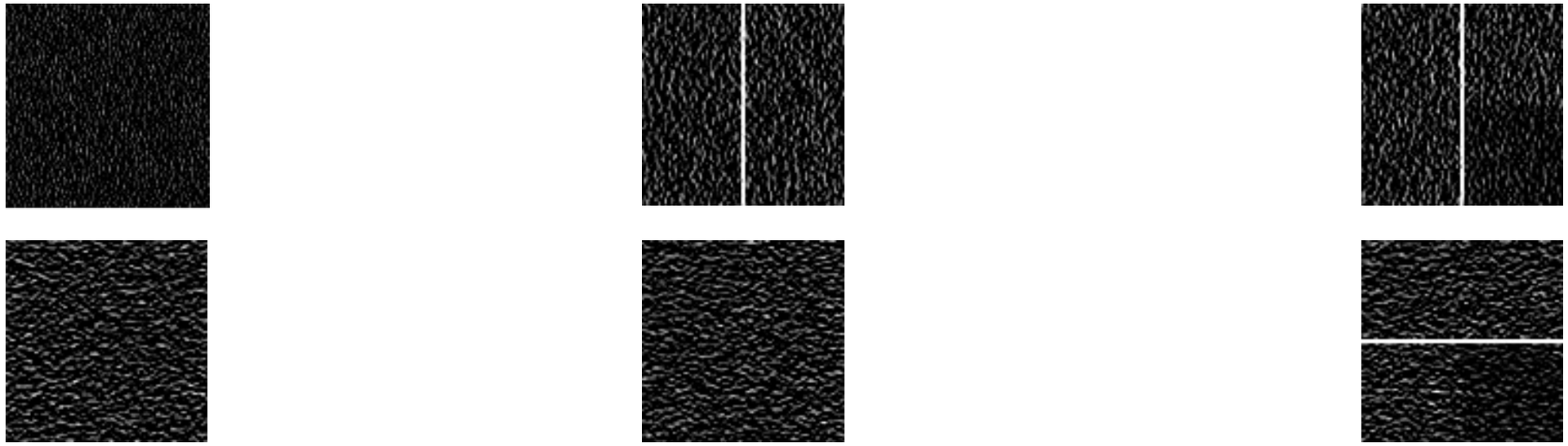


Harris Corner Detector

Measuring semi-major and semi-minor axes of ellipse

$\mathbf{I}_x$

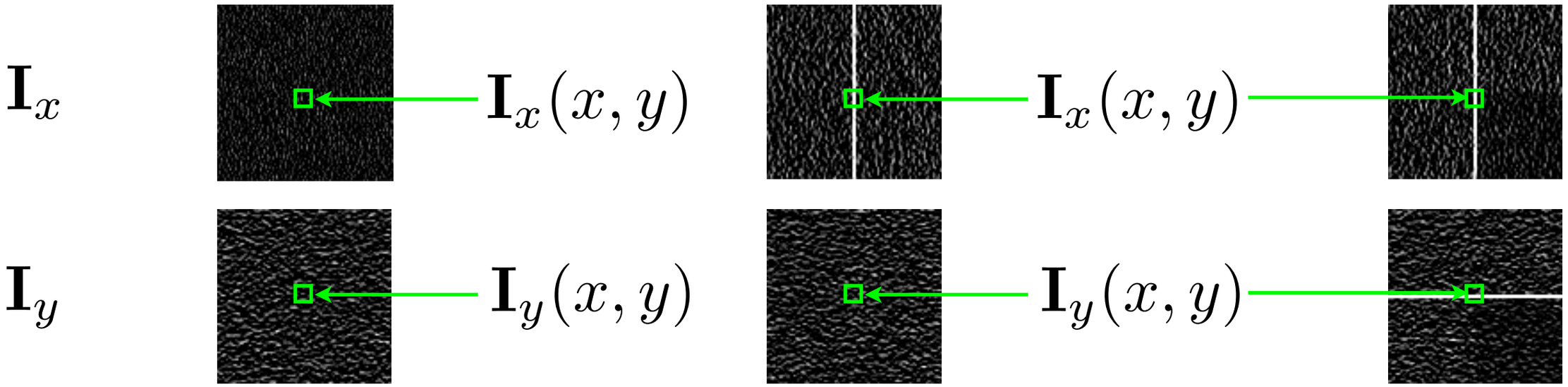
$\mathbf{I}_y$



Uniform Edge Corner

Harris Corner Detector

Measuring semi-major and semi-minor axes of ellipse



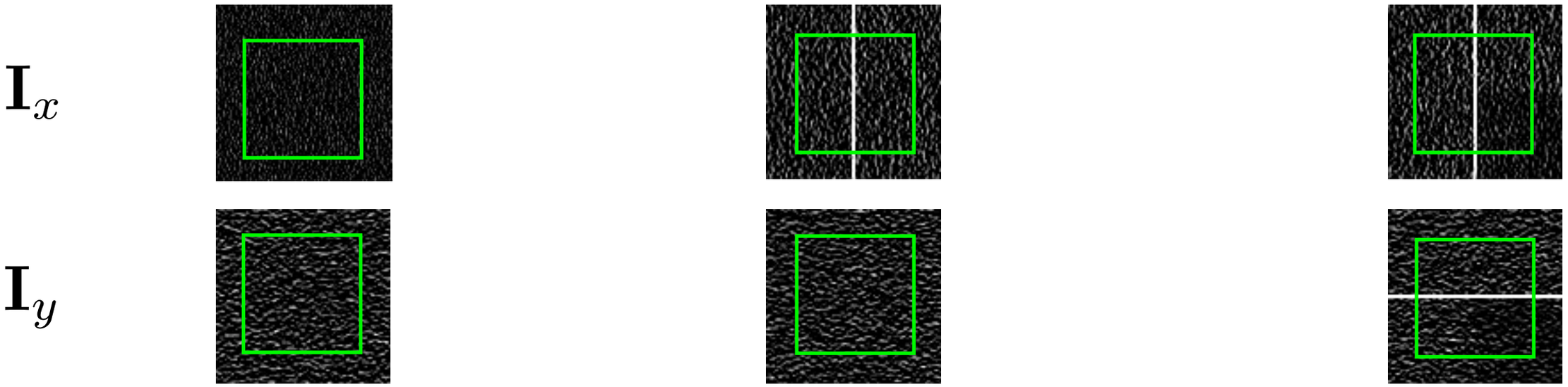
Uniform Edge Corner

For a single pixel of interest, calculate Harris matrix (or structure tensor).

$$\mathbf{M}^{xy} = \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix}$$

Harris Corner Detector

Measuring semi-major and semi-minor axes of ellipse

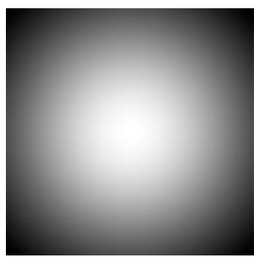


Uniform Edge Corner

For a single pixel of interest, calculate Harris matrix (or structure tensor).

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Better to sum over several pixels surrounding pixel of interest, weighted by weighting matrix **W**.



W is Gaussian

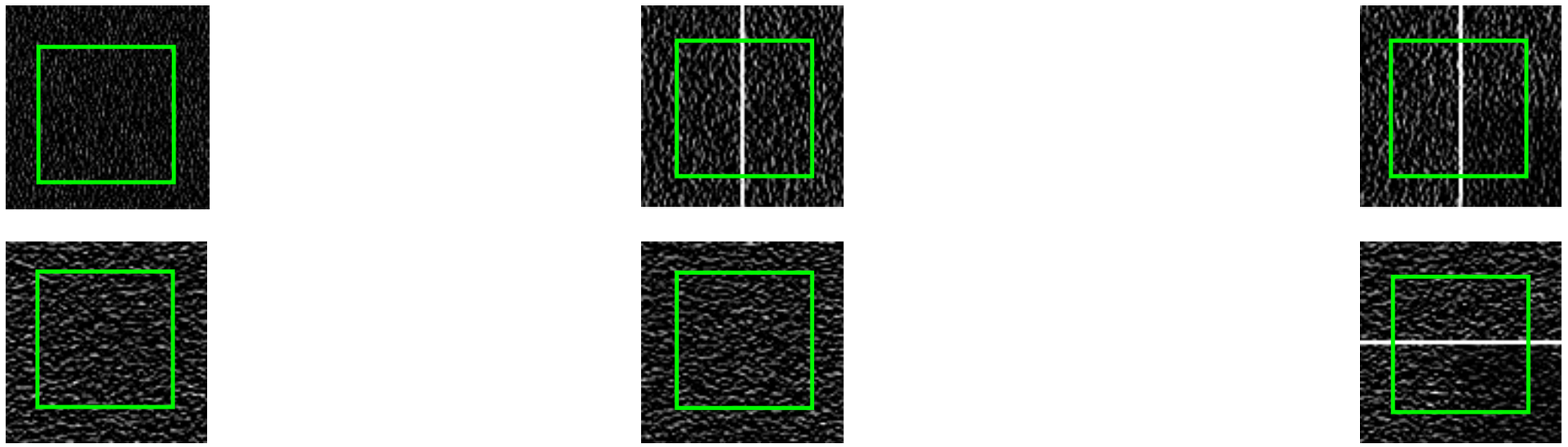
W is Uniform

Harris Corner Detector

Measuring semi-major and semi-minor axes of ellipse

$\mathbf{I}_x$

$\mathbf{I}_y$



Uniform Edge Corner

For a single pixel of interest, calculate Harris matrix (or structure tensor).

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Eigenvalues of $\mathbf{M}^{xy}$ provide major axis λ_1 and minor axis λ_2.

Harris Corner Detector

Find Eigenvalues of Harris Matrix

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Eigenvalues of $\mathbf{M}^{xy}$ provide major axis λ_1 and minor axis λ_2.

When both eigenvalues are large, we have a corner.

When one of the eigenvalues is large, we have an edge.

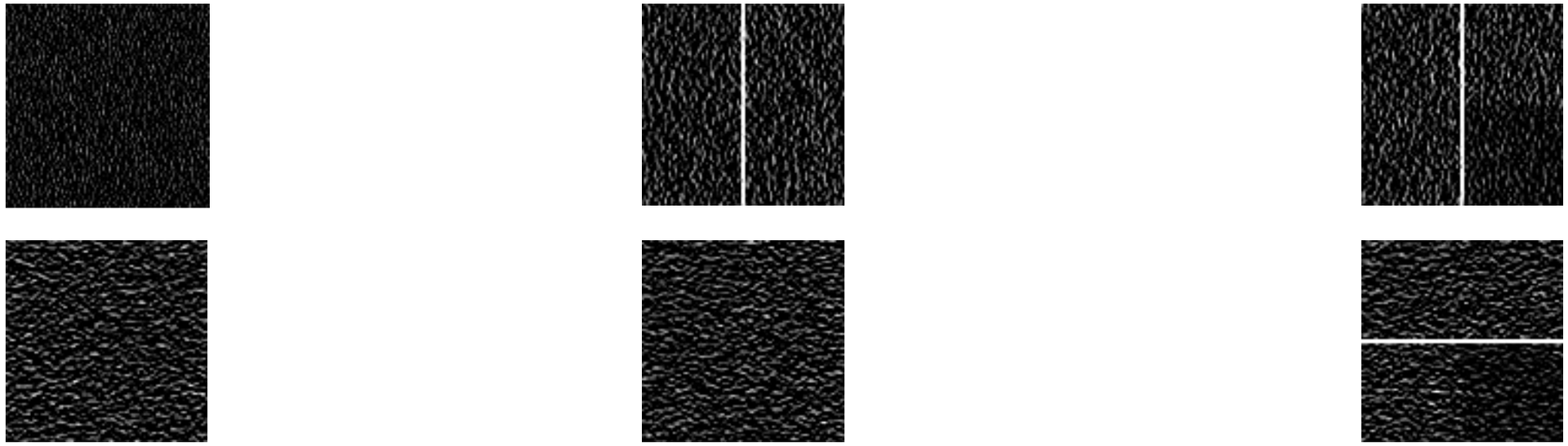
Eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ of the matrix provide directions of ellipse axes.

Harris Corner Detector

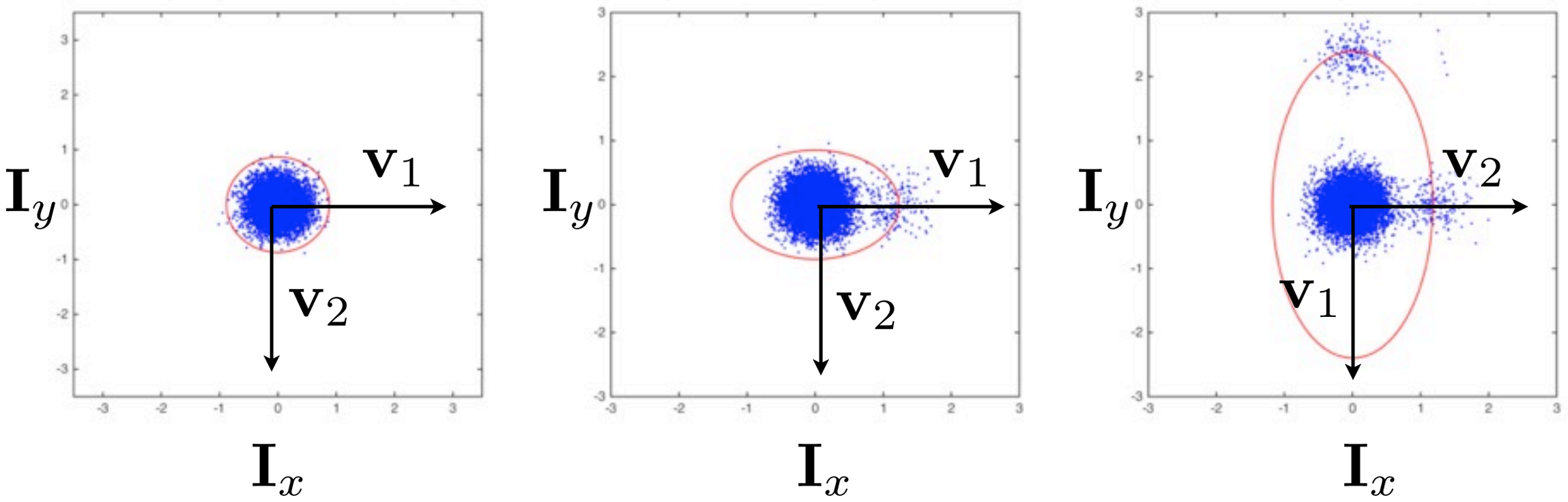
Plotted points fall within an ellipse

$\mathbf{I}_x$

$\mathbf{I}_y$

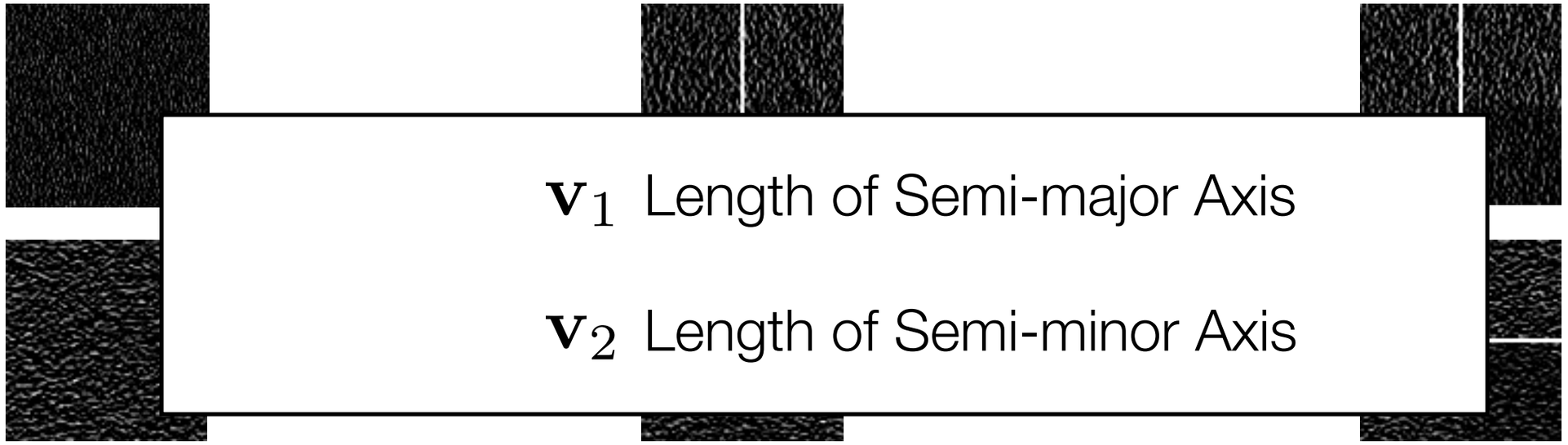


Uniform | Edge | Corner

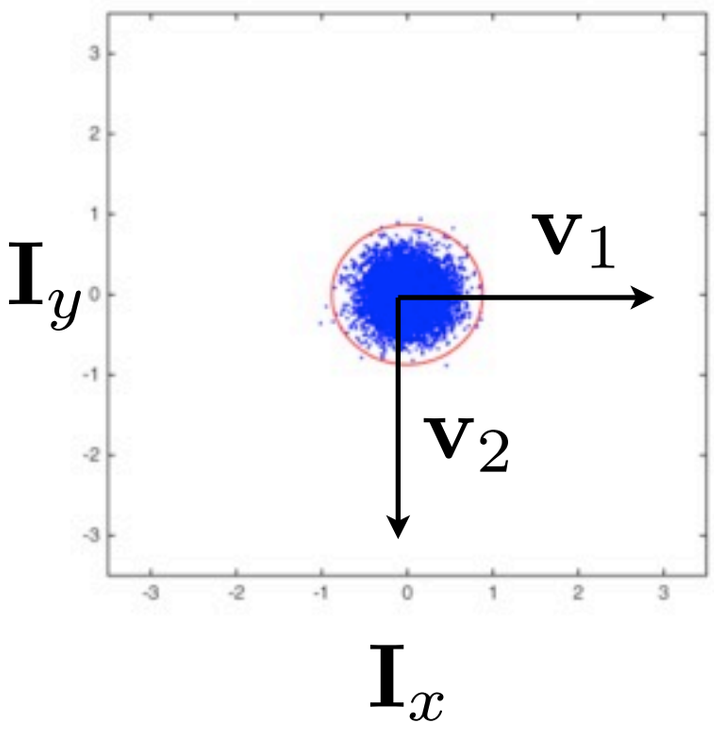


Harris Corner Detector

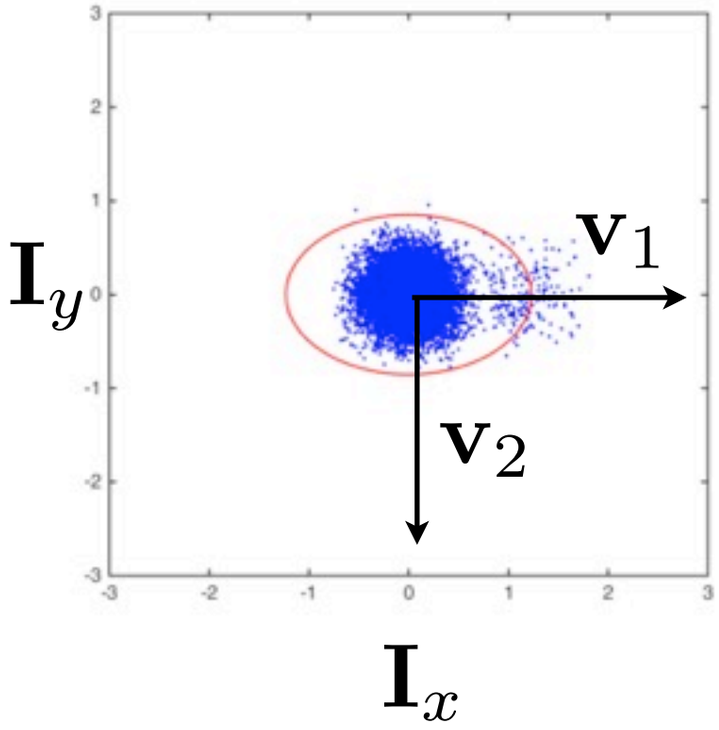
Plotted points fall within an ellipse



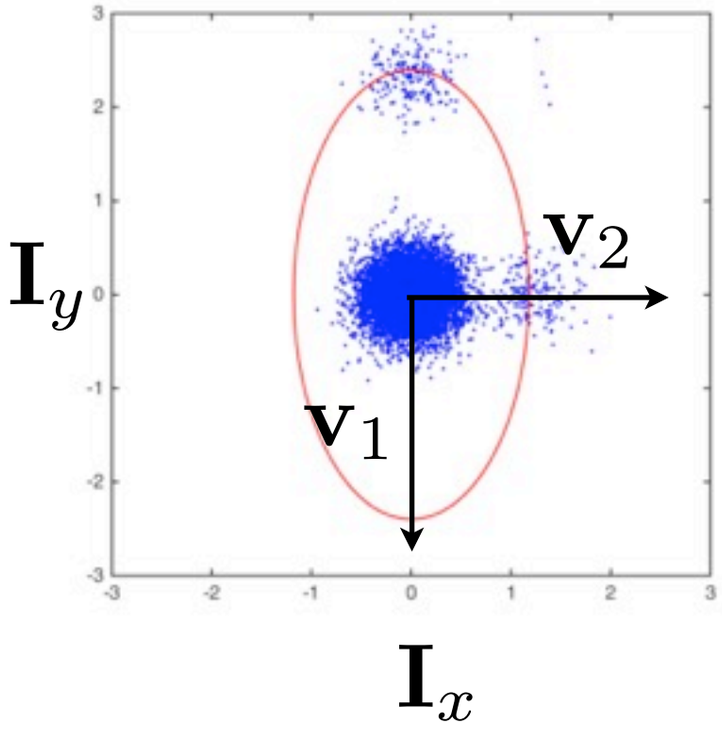
Uniform



Edge

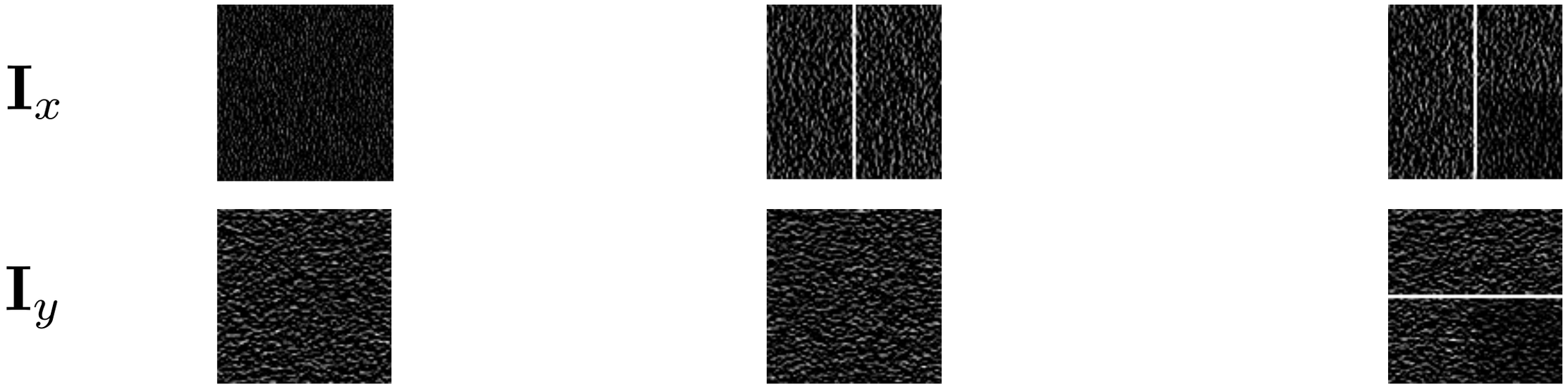


Corner



Harris Corner Detector

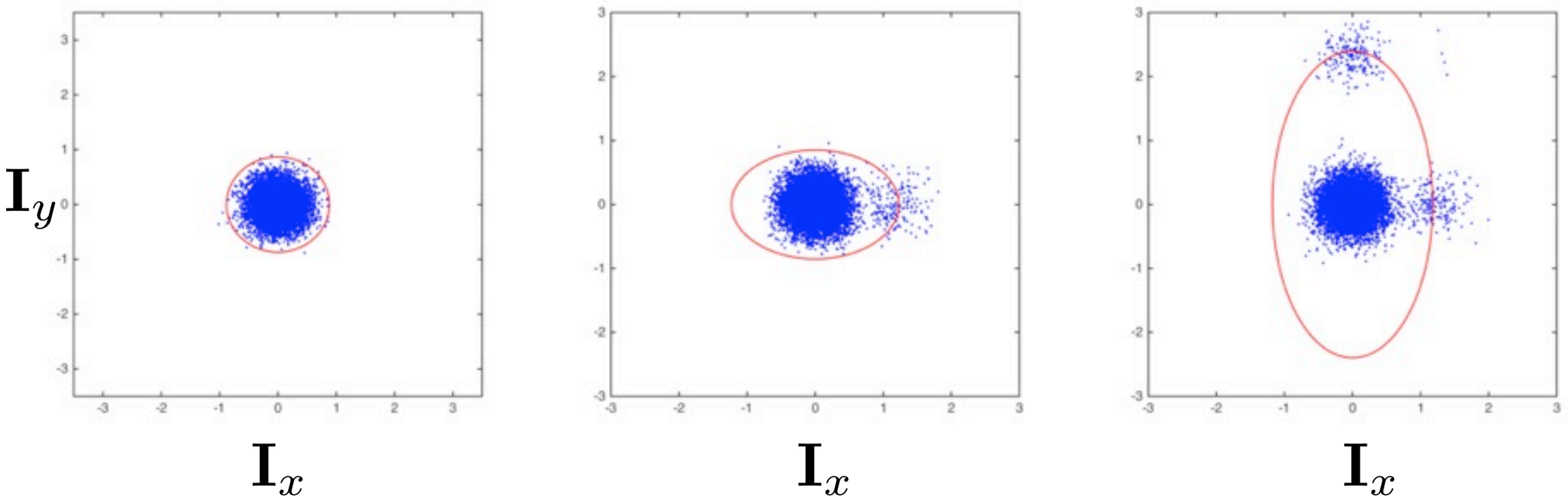
Plotted points fall within an ellipse



Uniform

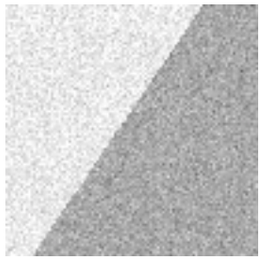
Edge

Corner



Harris Corner Detector

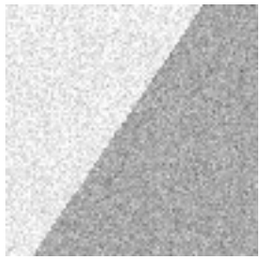
How does the ellipse look for this image?
What are the directions of the ellipse axes?



$\mathbf{I}$

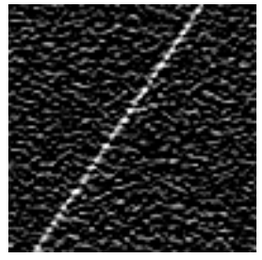
Harris Corner Detector

How does the ellipse look for this image?
What are the directions of the ellipse axes?



$\mathbf{I}$

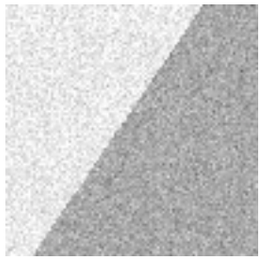
$\mathbf{I}_x$



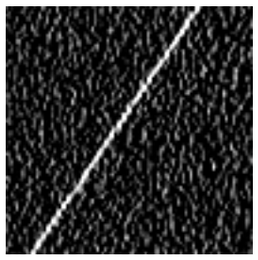
$\mathbf{I}_y$

Harris Corner Detector

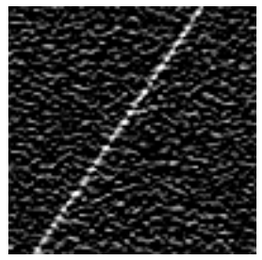
How does the ellipse look for this image?
What are the directions of the ellipse axes?



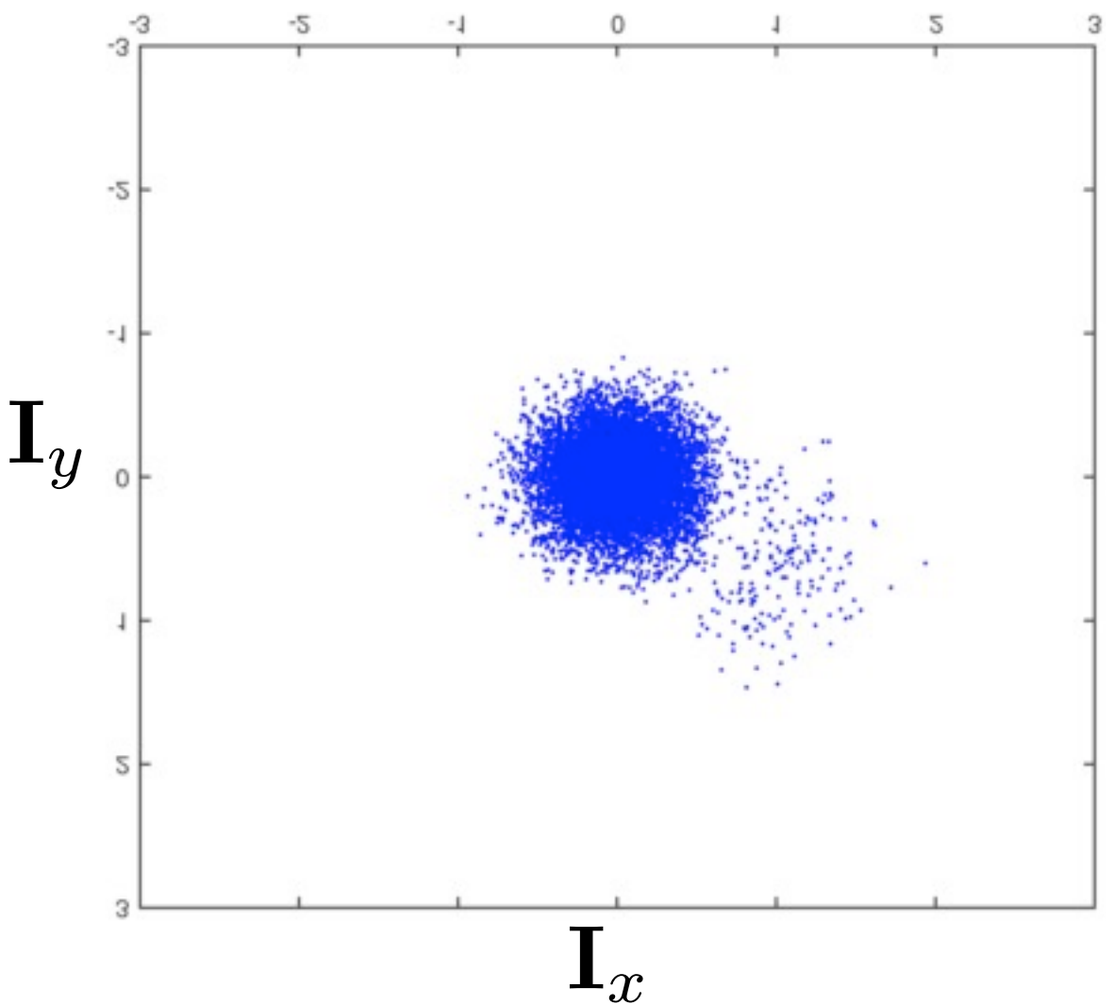
$\mathbf{I}$



$\mathbf{I}_x$



$\mathbf{I}_y$

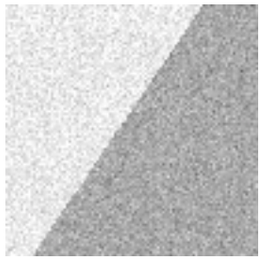


$\mathbf{I}_y$

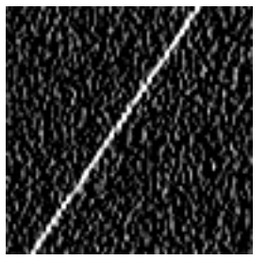
$\mathbf{I}_x$

Harris Corner Detector

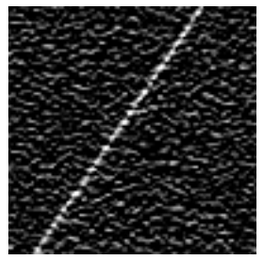
How does the ellipse look for this image?
What are the directions of the ellipse axes?



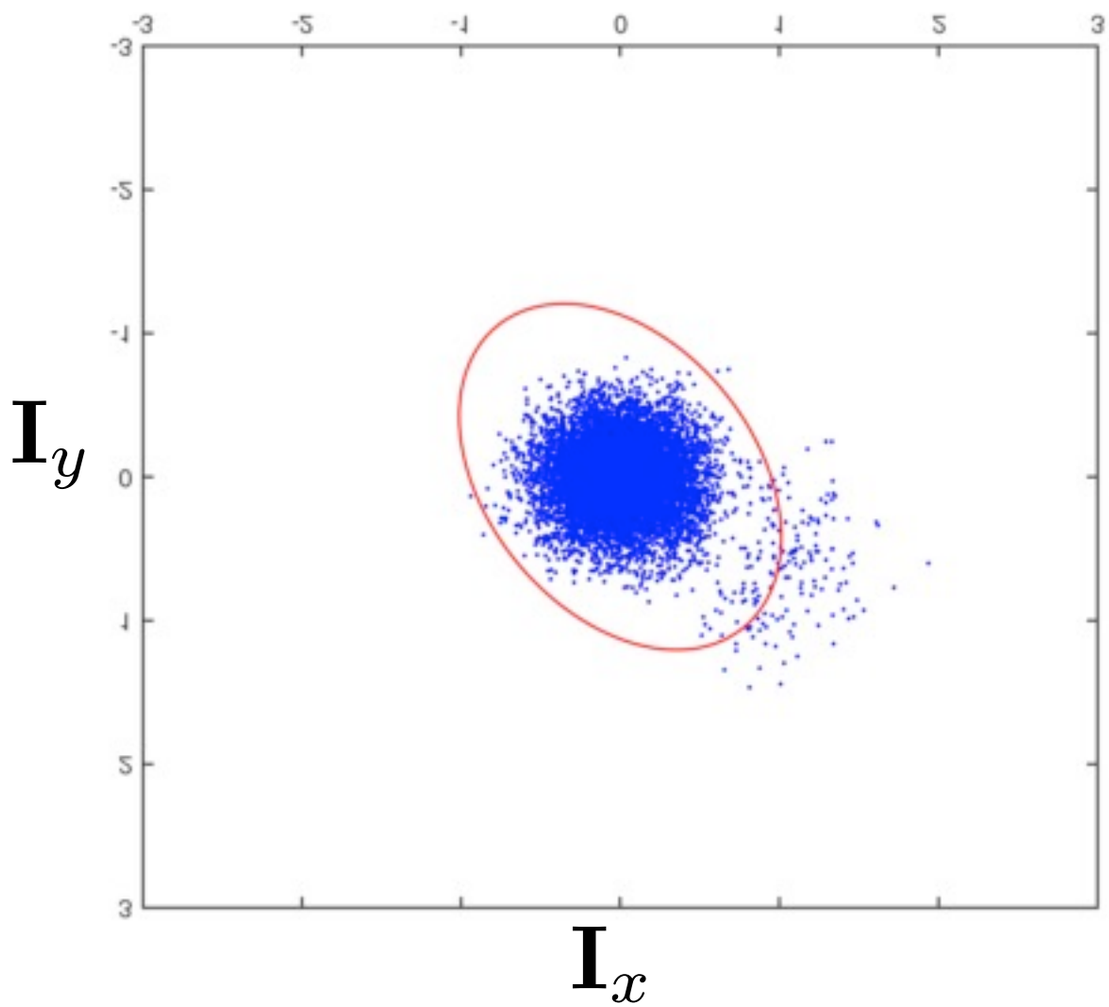
$\mathbf{I}$



$\mathbf{I}_x$

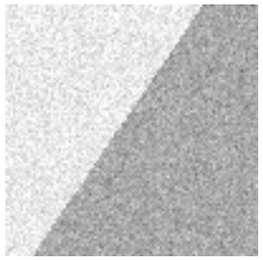


$\mathbf{I}_y$

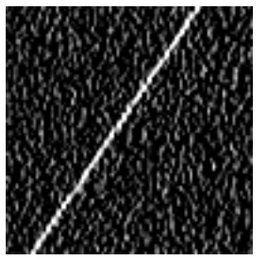


Harris Corner Detector

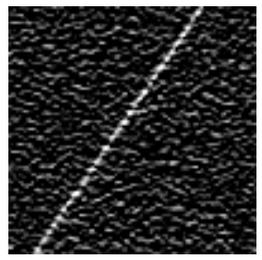
How does the ellipse look for this image?
What are the directions of the ellipse axes?



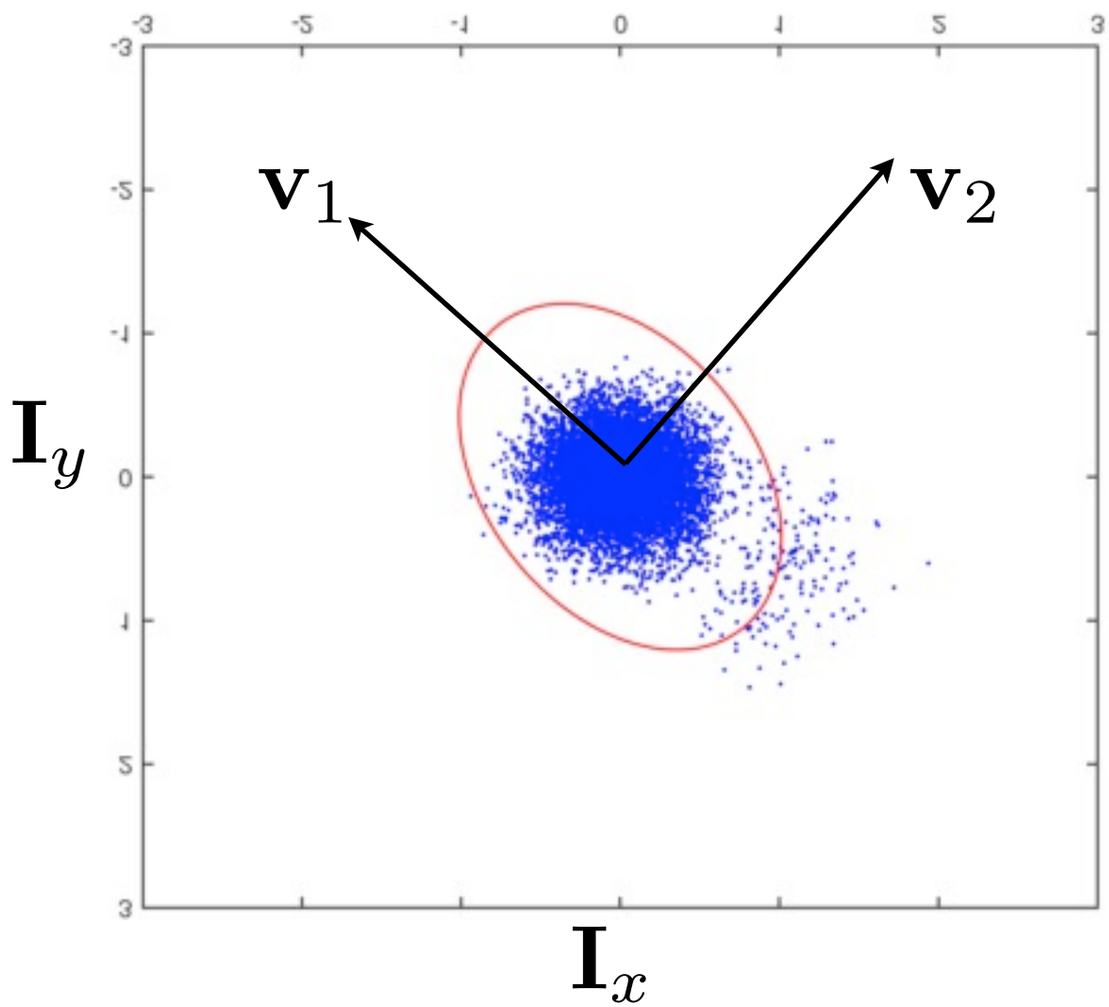
$\mathbf{I}$



$\mathbf{I}_x$

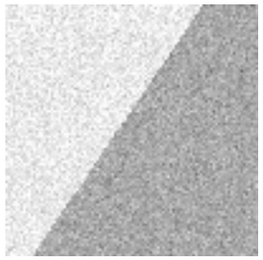


$\mathbf{I}_y$

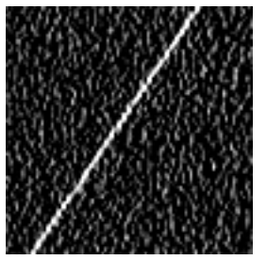


Harris Corner Detector

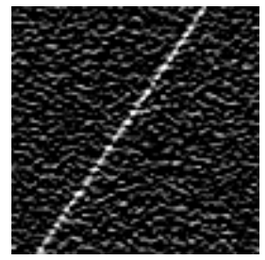
How does the ellipse look for this image?
What are the directions of the ellipse axes?



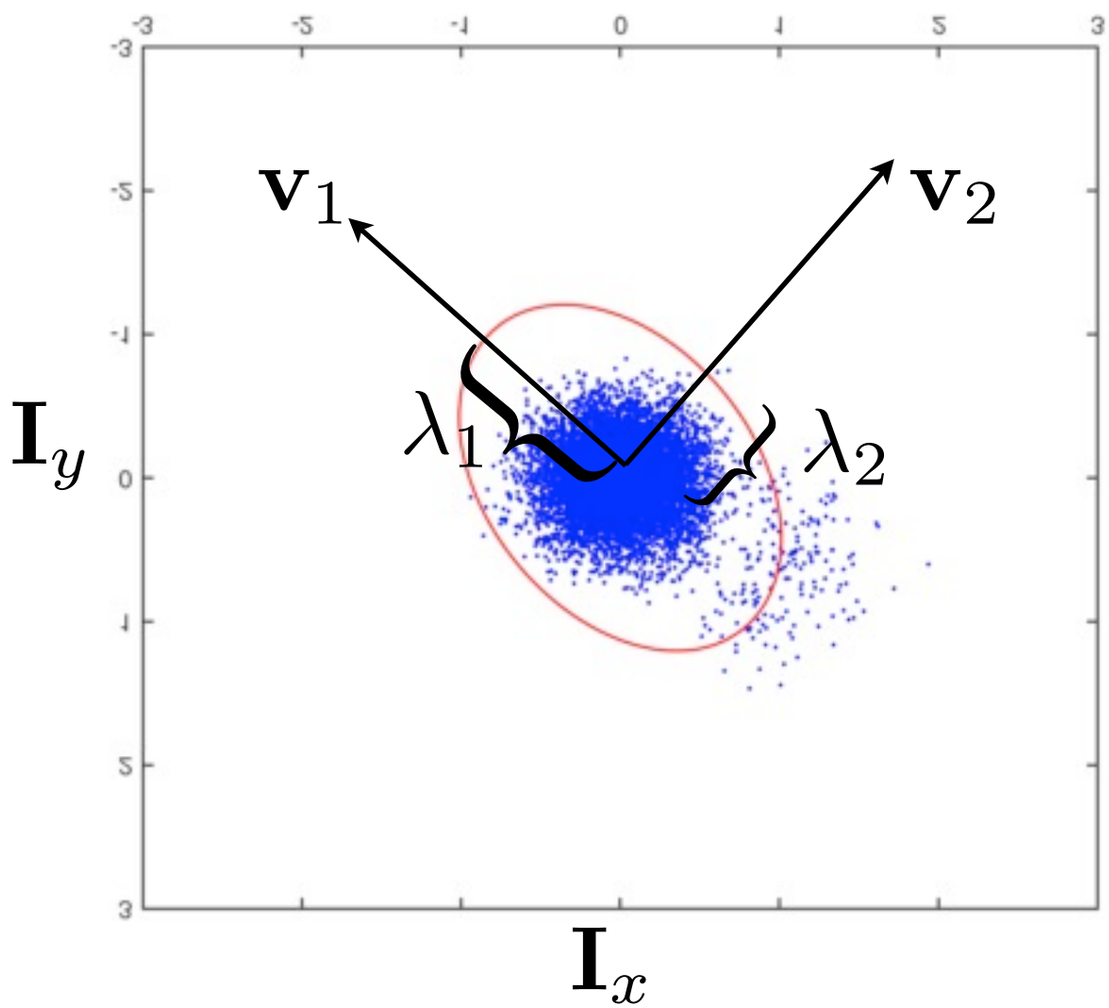
$\mathbf{I}$



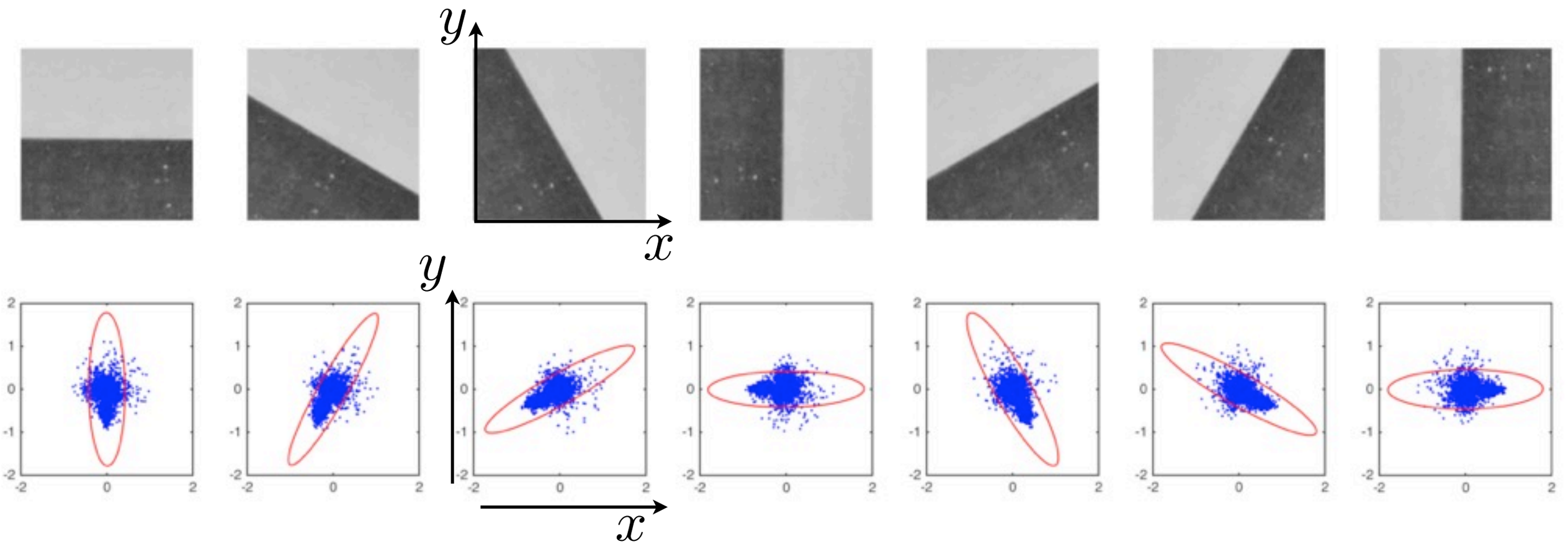
$\mathbf{I}_x$



$\mathbf{I}_y$



Examples for Other Edges



Harris Corner Detector

Find Eigenvalues of Harris Matrix

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Eigenvalues of $\mathbf{M}^{xy}$ provide major axis λ_1 and minor axis λ_2.

Instead of comparing two values, it would be better to have a single measure to compare.

Several measures are used in computer vision.

$$\min(\lambda_1, \lambda_2)$$

$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Retain the point if one of the above measures is greater than a threshold.

Harris Corner Detector

Find Eigenvalues of Harris Matrix

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Eigenvalues of $\mathbf{M}^{xy}$ provide major axis λ_1 and minor axis λ_2.

Instead of comparing two values, it would be better to have a single measure to compare.

Several measures are used in computer vision.

$$\min(\lambda_1, \lambda_2)$$

$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Determinant of Harris Matrix

Trace of Harris Matrix

Retain the point if one of the above measures is greater than a threshold.

Harris Corner Detector

Harris Corner Detector

Result of *min(lambda1,lambda2) = lambda2*

Harris Corner Detector

Thresholding (Retain all where *lambda2* > .1 * maximum)

Harris Corner Detector

Points marked on image

Matching Between Two Images

Extract patches from both images.
For every pair of patches, compute SSD (template matching).
For every patch in first image, find patch with min(SSD) in second image.

Matching Between Two Images

Extract patches from both images.
For every pair of patches, compute SSD (template matching).
For every patch in first image, find patch with min(SSD) in second image.

Matching Between Two Images

Extract patches from both images.
For every pair of patches, compute SSD (template matching).
For every patch in first image, find patch with min(SSD) in second image.

Matching Between Two Images

Extract patches from both images.
For every pair of patches, compute SSD (template matching).
For every patch in first image, find patch with min(SSD) in second image.

Issue

What happens when rotation, scale, and lighting are different?

Issue

What happens when rotation, scale, and lighting are different?

SSD with patches is influenced by changes in rotation, scale, and lighting

SIFT: Scale Invariant Feature Transform

Iconic work in Computer Vision by David Lowe.

First published an initial paper in 1999.
http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

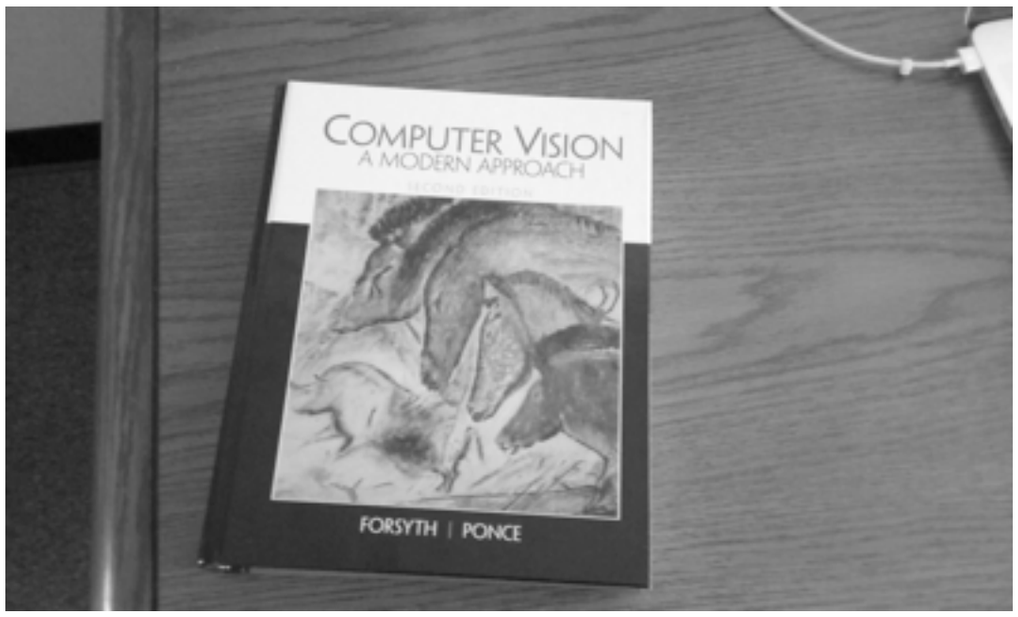
Comprehensive paper in 2004.
http://www.cs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf

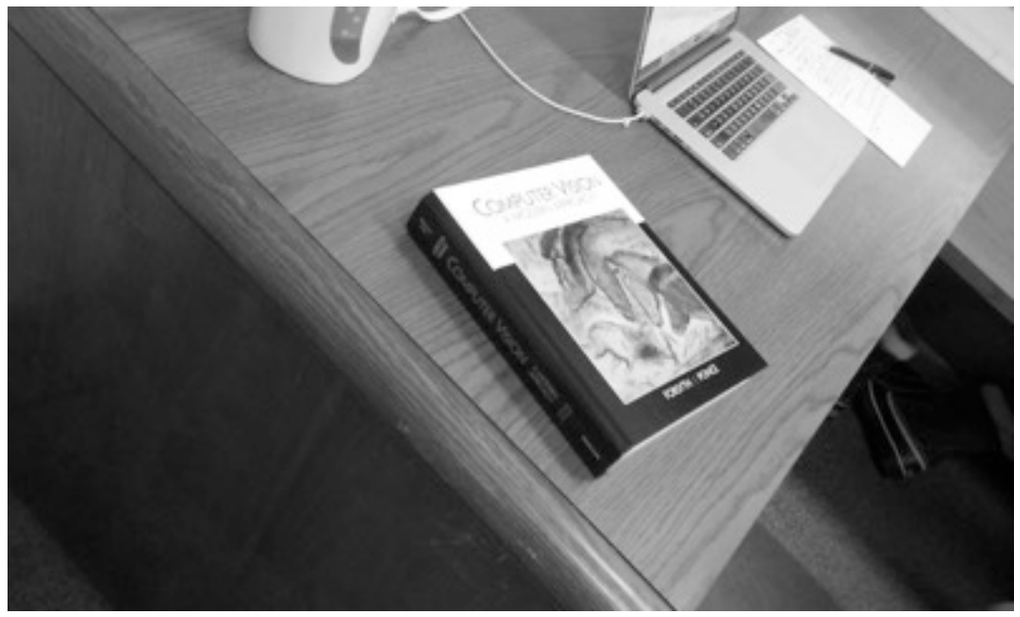
We will cover the basics of SIFT,
however, you should read the papers for details
on parameter setting and implementation.

SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance



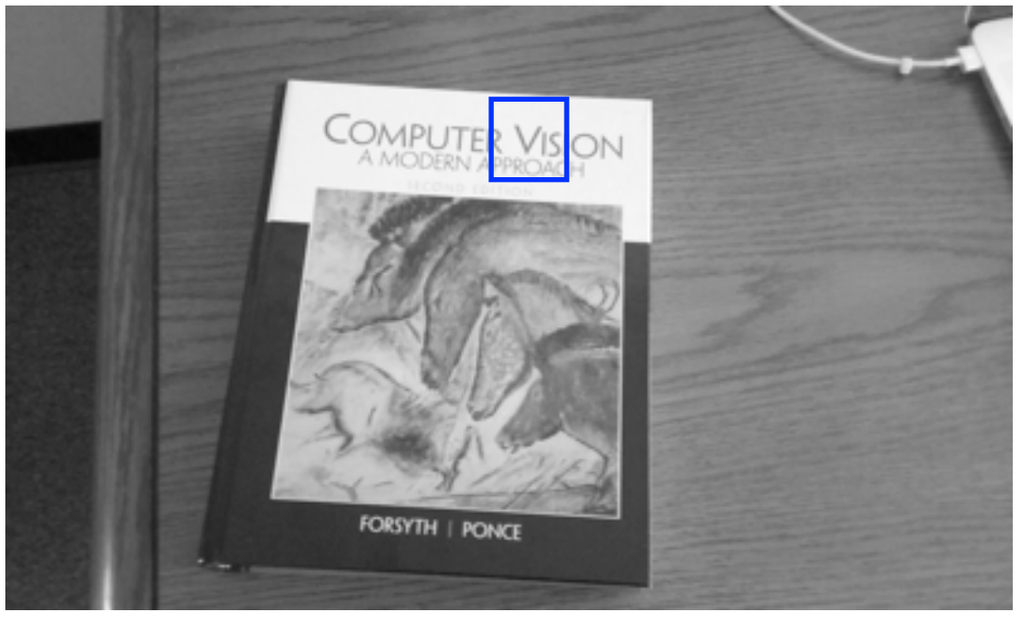


SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance

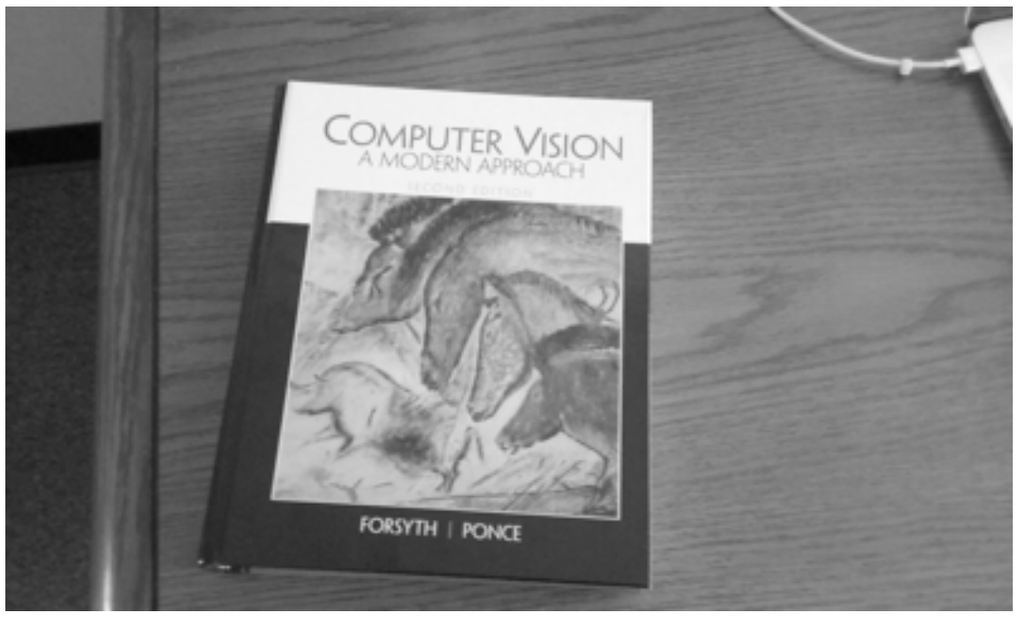
V in left image is larger than in right image,
but conceptually it is still the same point.

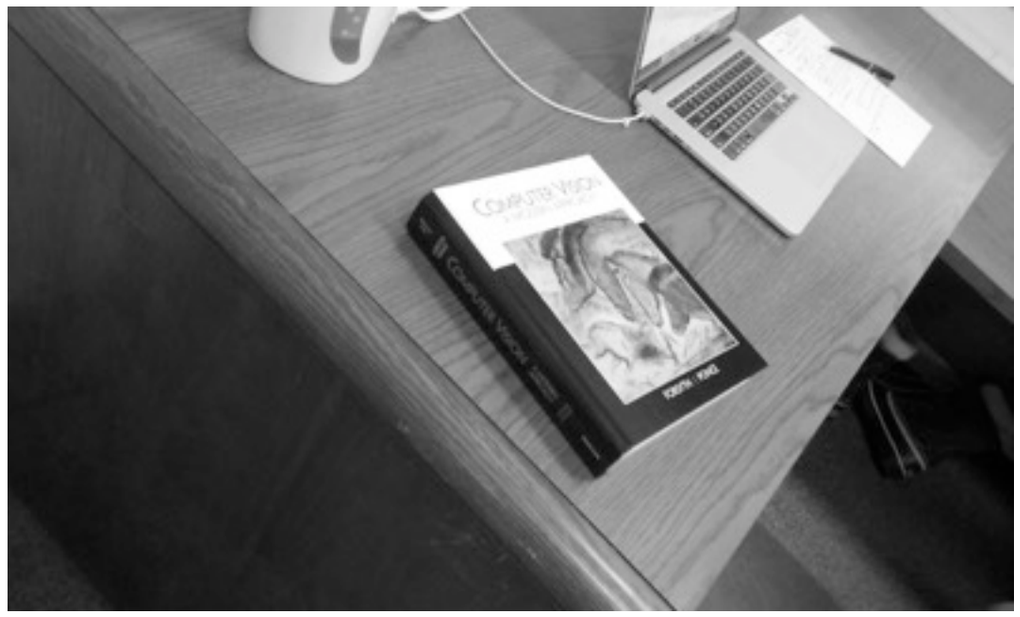


SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance



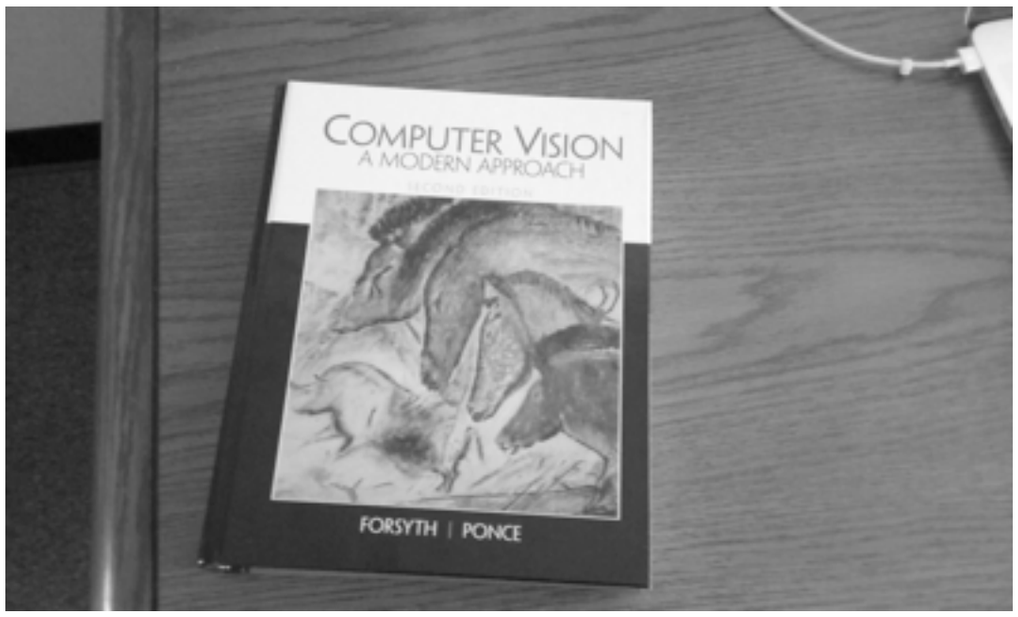


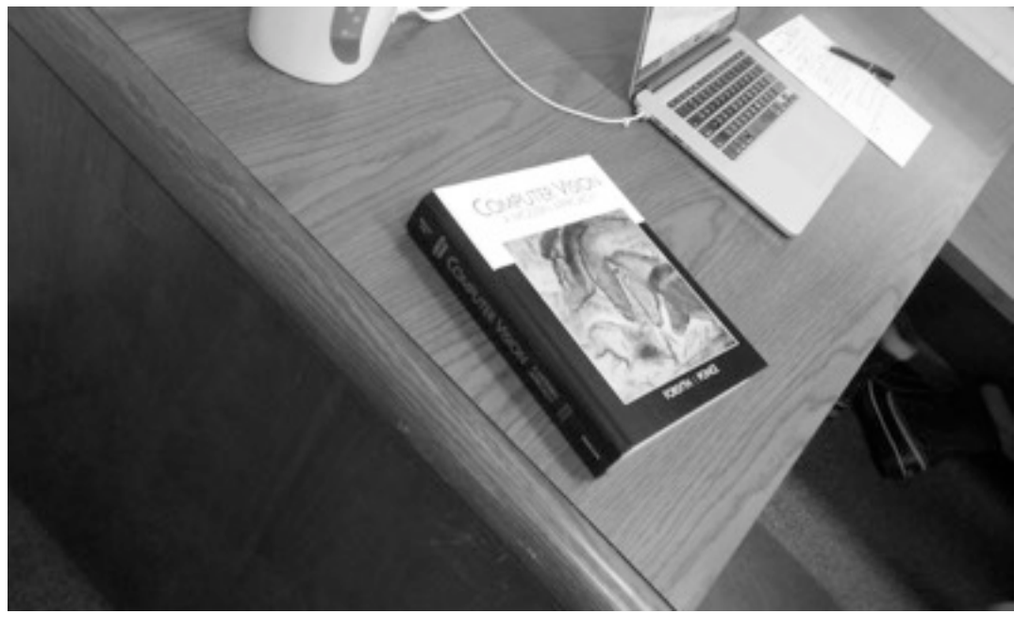
SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance

Rotation Invariance





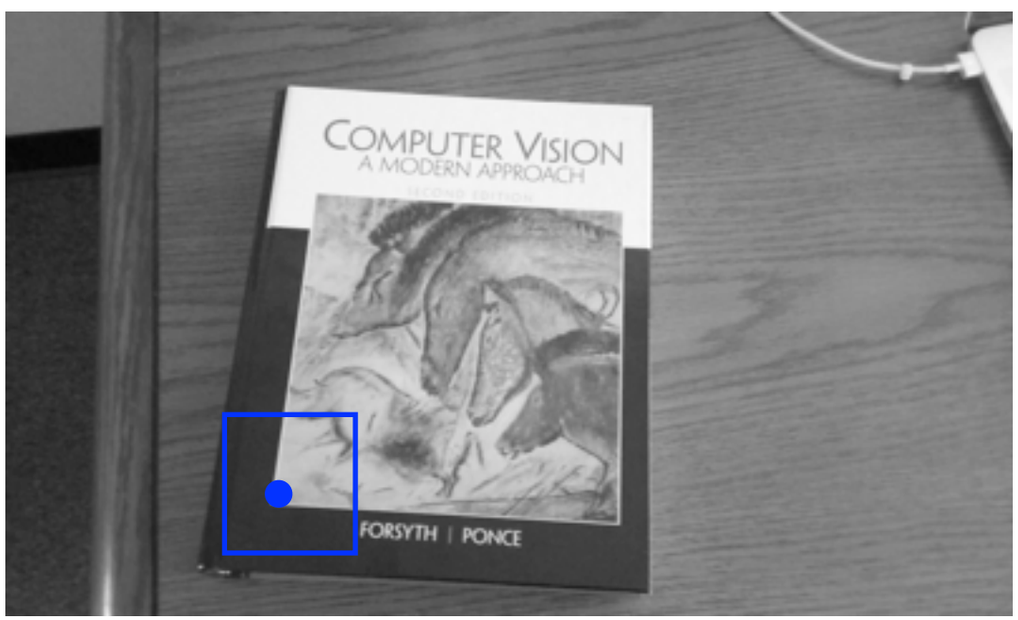
SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance

Rotation Invariance

Corner is rotated,
but conceptually still the same point.

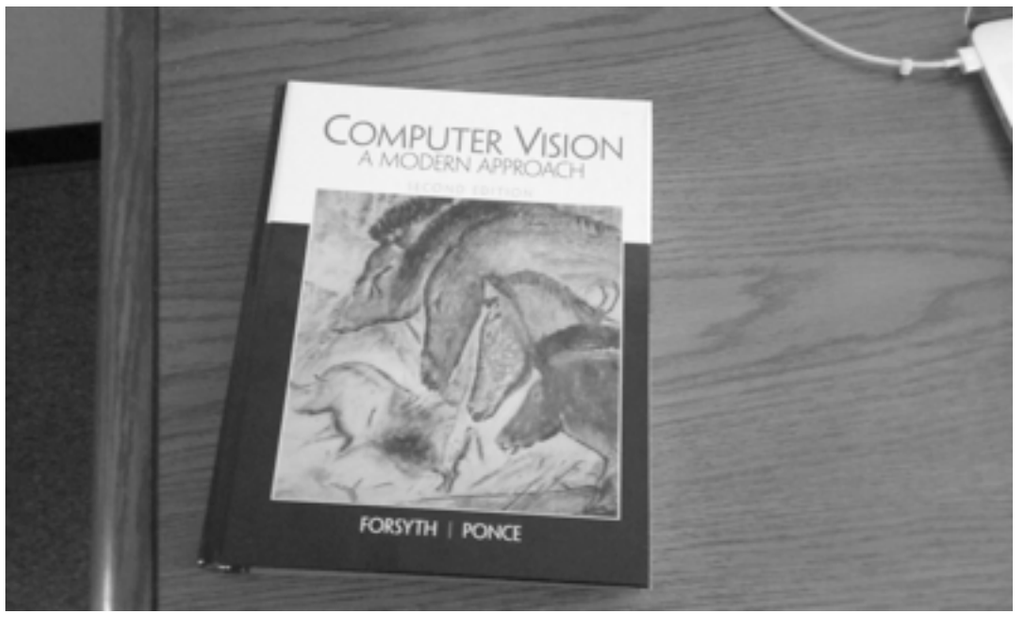


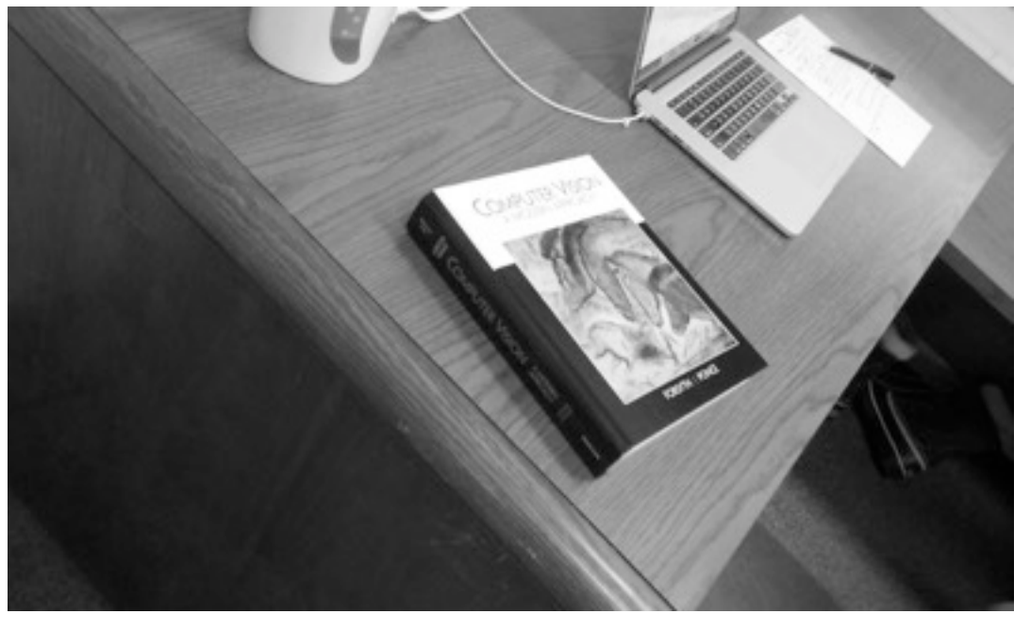
SIFT: Scale Invariant Feature Transform

What Characteristics Should Interest Points Have?

Scale Invariance

Rotation Invariance





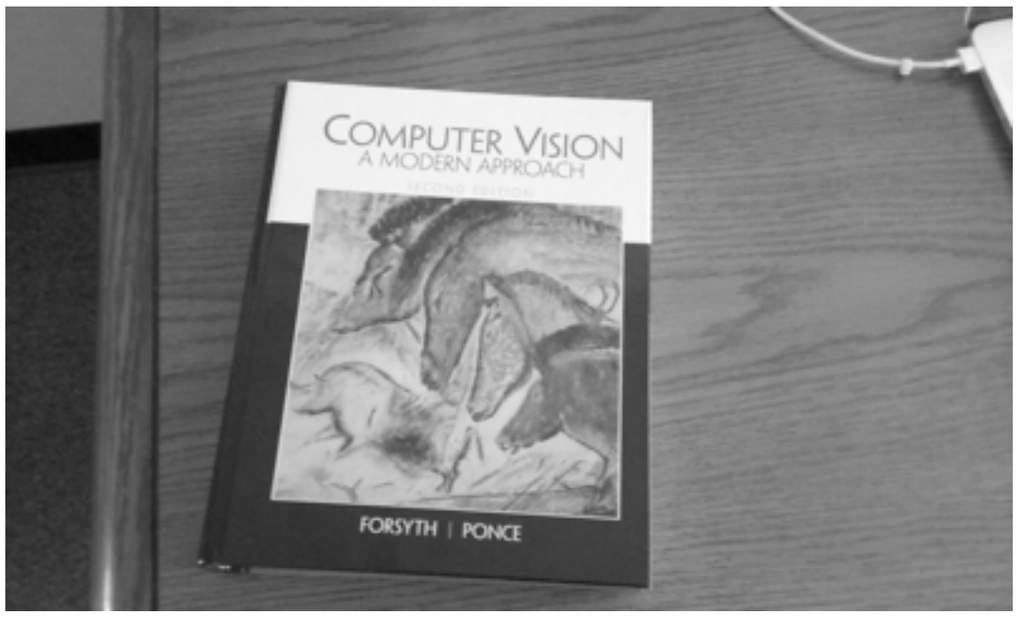
SIFT: Scale Invariant Feature Transform

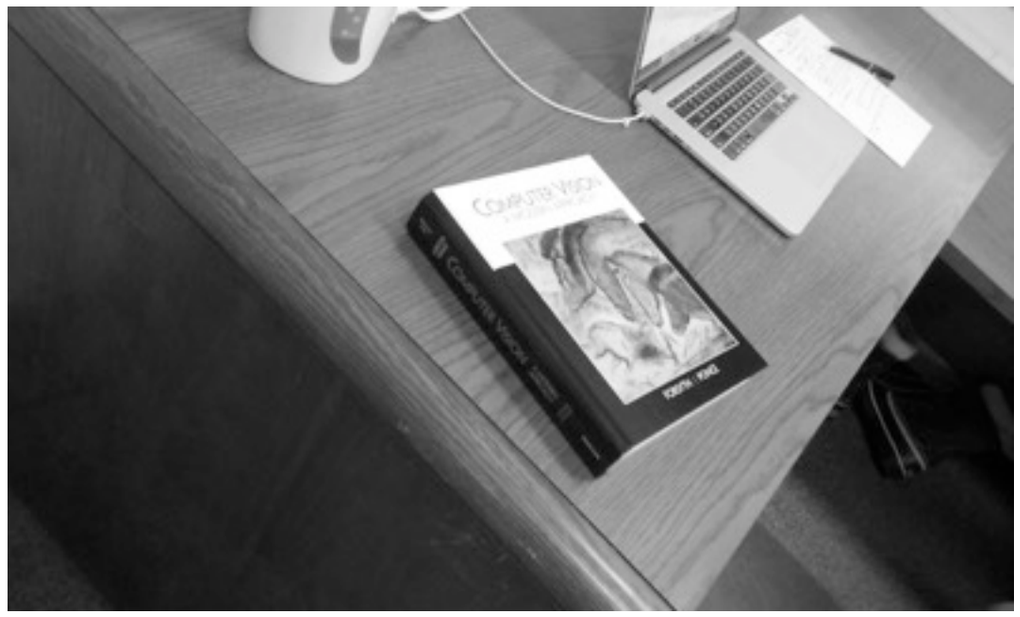
What Characteristics Should Interest Points Have?

Scale Invariance

Rotation Invariance

Illumination Invariance





SIFT: Scale Invariant Feature Transform

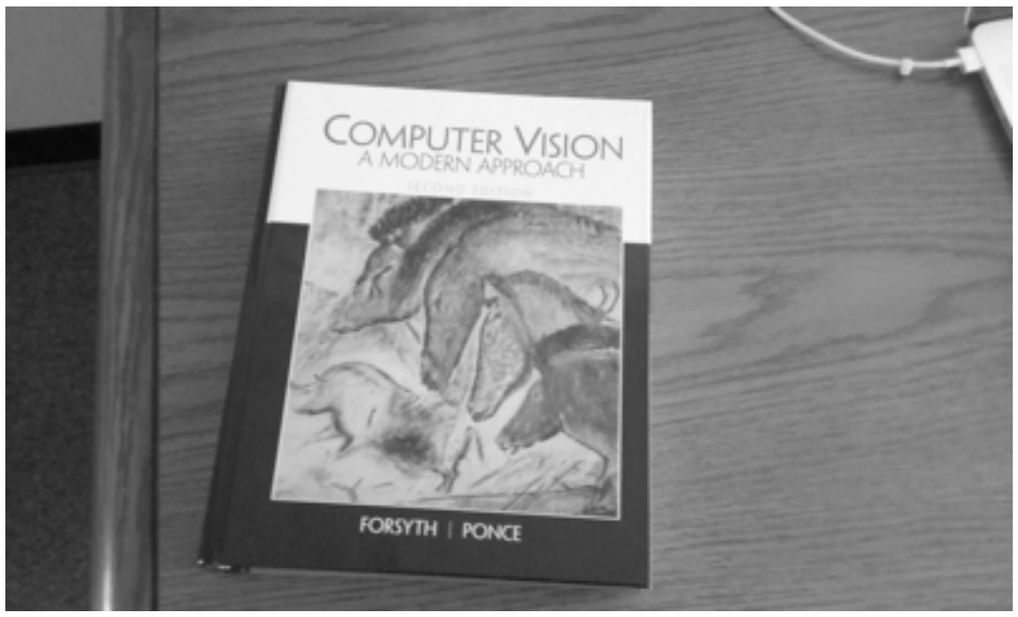
What Characteristics Should Interest Points Have?

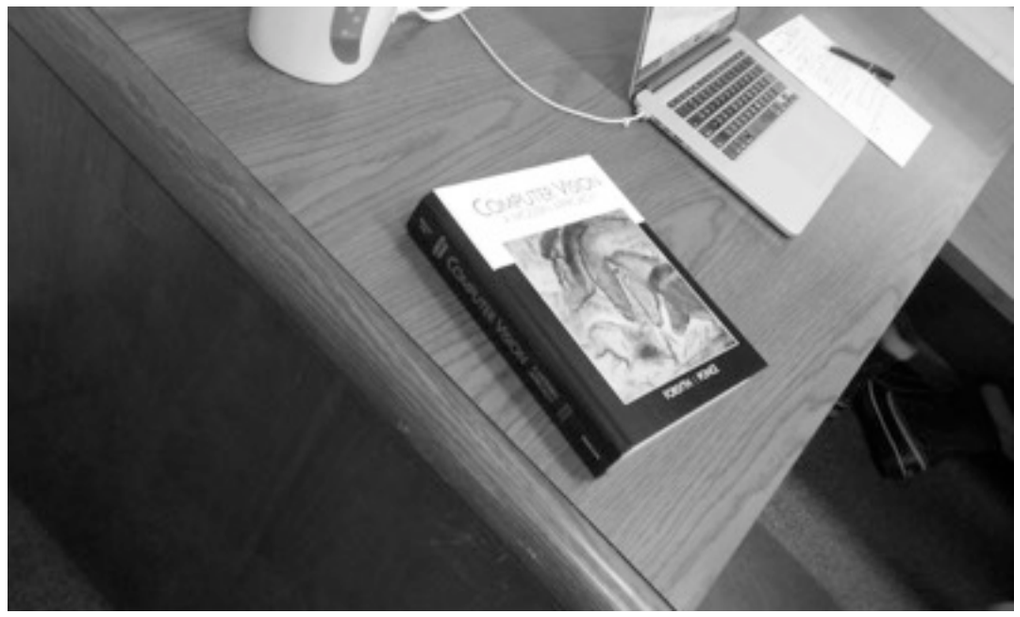
Scale Invariance

Rotation Invariance

Illumination Invariance

Contrast in right image is higher than in left image,
but points still belong to the same book





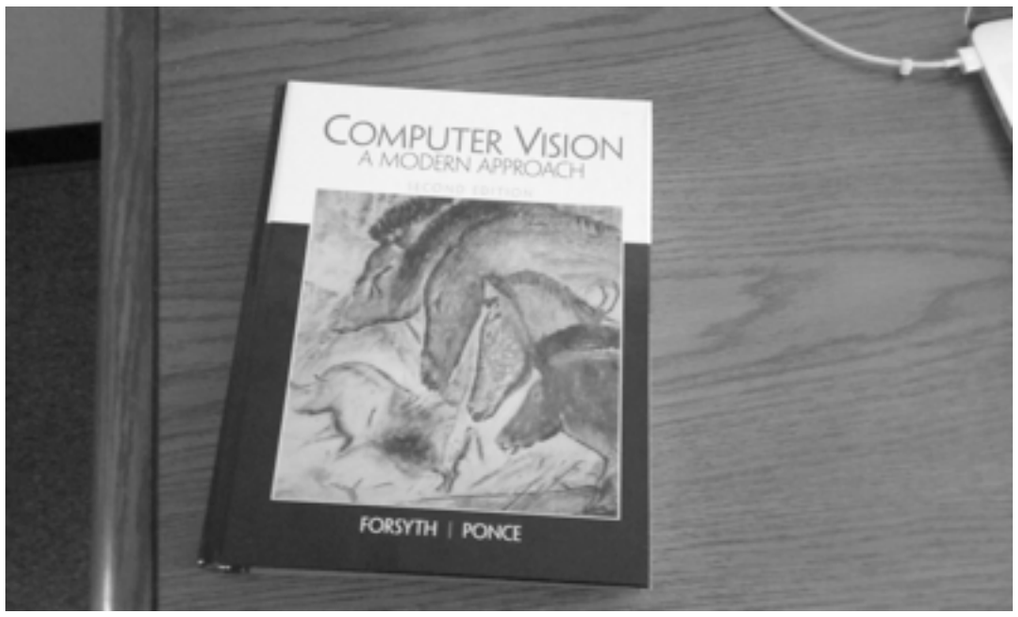
SIFT: Scale Invariant Feature Transform

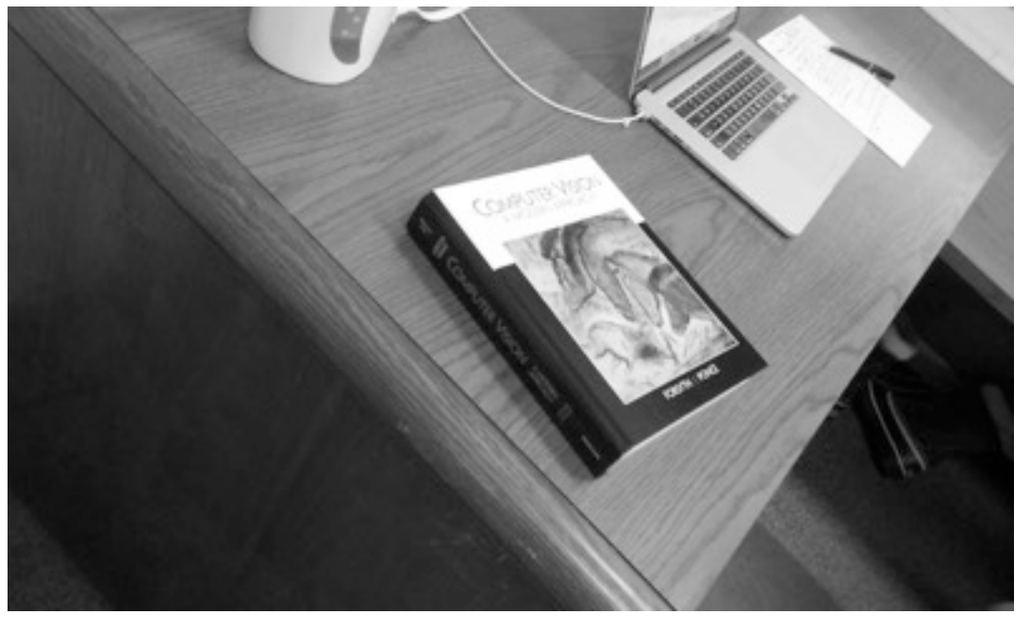
What Characteristics Should Interest Points Have?

Scale Invariance

Rotation Invariance

Illumination Invariance





SIFT: Scale Invariant Feature Transform

Recollect Gaussian Pyramids

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

$\sigma = 16$

$\sigma = 32$

SIFT: Scale Invariant Feature Transform

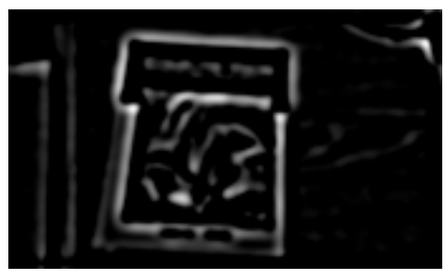
We can similarly form Laplacian of Gaussian (LoG) Pyramids

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

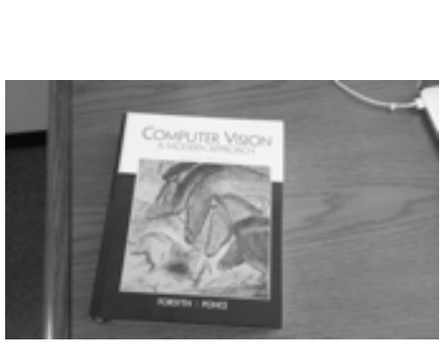
$\sigma = 16$



$\sigma = 32$

SIFT: Scale Invariant Feature Transform

We can similarly form Laplacian of Gaussian (LoG) Pyramids



$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

$\sigma = 16$

$\sigma = 32$

Rescaled to lie between 0 and 1.

SIFT: Scale Invariant Feature Transform

One way to build LoG: Filter with LoG filter

$\sigma = 2$

$\sigma = 4$

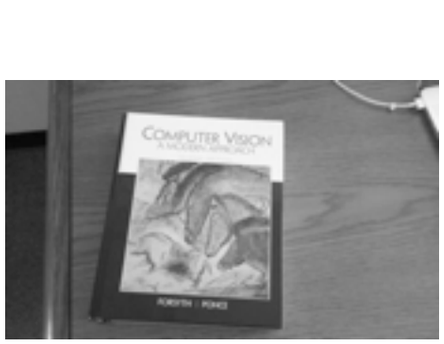
$\sigma = 8$

$\sigma = 16$

$\sigma = 32$

SIFT: Scale Invariant Feature Transform

One way to build LoG: Filter with LoG filter



$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

$\sigma = 16$

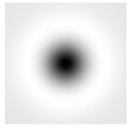
$\sigma = 32$

SIFT: Scale Invariant Feature Transform

One way to build LoG: Filter with LoG filter

Image

$\otimes$



$\sigma = 16$

$=$

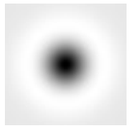
LoG

SIFT: Scale Invariant Feature Transform

Another way: Approximate by Difference of Gaussian (DoG)

Image

$\otimes$



$\sigma = 16$

$=$

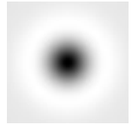
LoG

Image

SIFT: Scale Invariant Feature Transform

Another way: Approximate by Difference of Gaussian (DoG)

$\otimes$



$\sigma = 16$

Image

Filter of same size

$=$

LoG

$\otimes$

$=$

$\sigma = 16$

$\otimes$

$=$

$\sigma = 16 \times 1.2$

Image

Gaussians

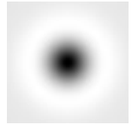
Slightly bigger filter

SIFT: Scale Invariant Feature Transform

Another way: Approximate by Difference of Gaussian (DoG)

Image

$\otimes$



$\sigma = 16$

Filter of same size

$=$

LoG

$\otimes$

$\sigma = 16$

$=$

$\otimes$

$\sigma = 16 \times 1.2$

$=$

$-$

DoG

Image

Gaussians

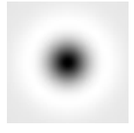
Slightly bigger filter

SIFT: Scale Invariant Feature Transform

Another way: Approximate by Difference of Gaussian (DoG)

Image

$\otimes$



$\sigma = 16$

Filter of same size

$=$

LoG

$\otimes$

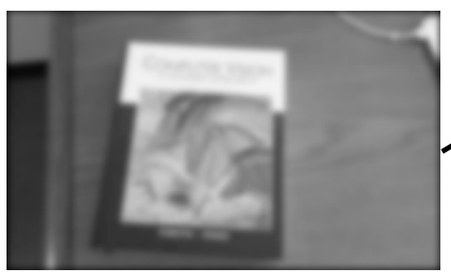
$\sigma = 16$

$=$

$\otimes$

$\sigma = 16 \times 1.2$

$=$



Image

Gaussians

$-$

DoG

Slightly bigger filter

Lowe found stability with multiplicative factors of up to 2.

SIFT: Scale Invariant Feature Transform

Difference of Gaussian (DoG) pyramid

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

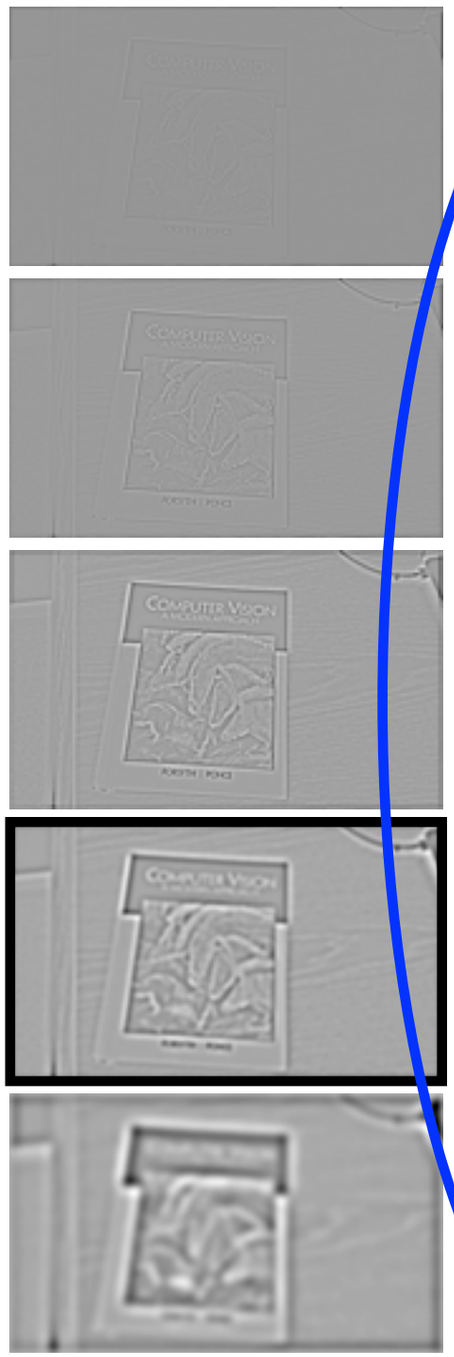
$\sigma = 16$

$\sigma = 32$

SIFT: Scale Invariant Feature Transform

Difference of Gaussian (DoG) pyramid

These are power of 2, and start at 2.

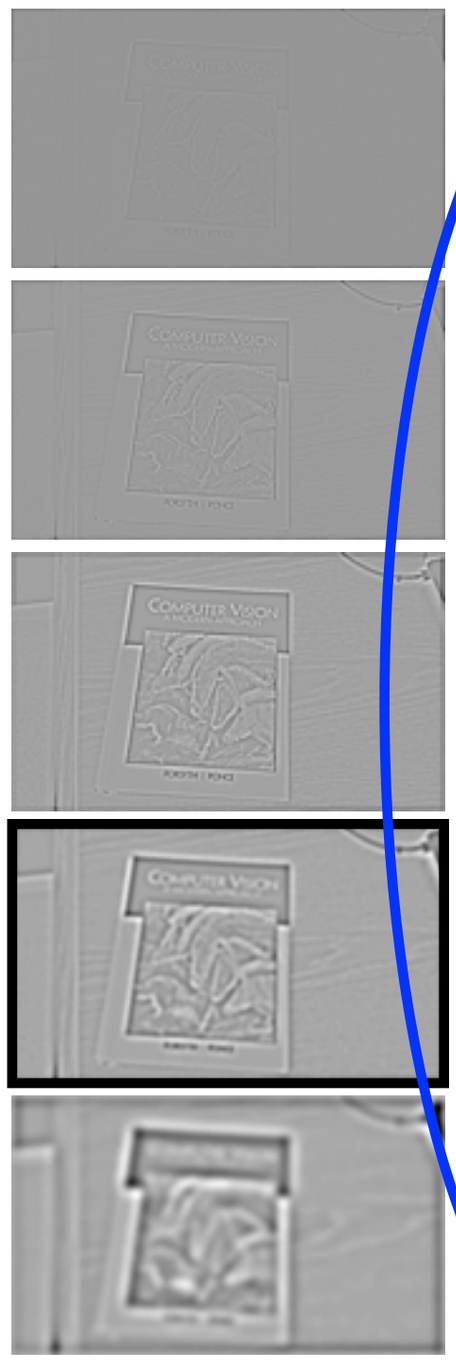


SIFT: Scale Invariant Feature Transform

Difference of Gaussian (DoG) pyramid

These are power of 2, and start at 2.

Can also be powers of sqrt(2),
or powers of any other number.



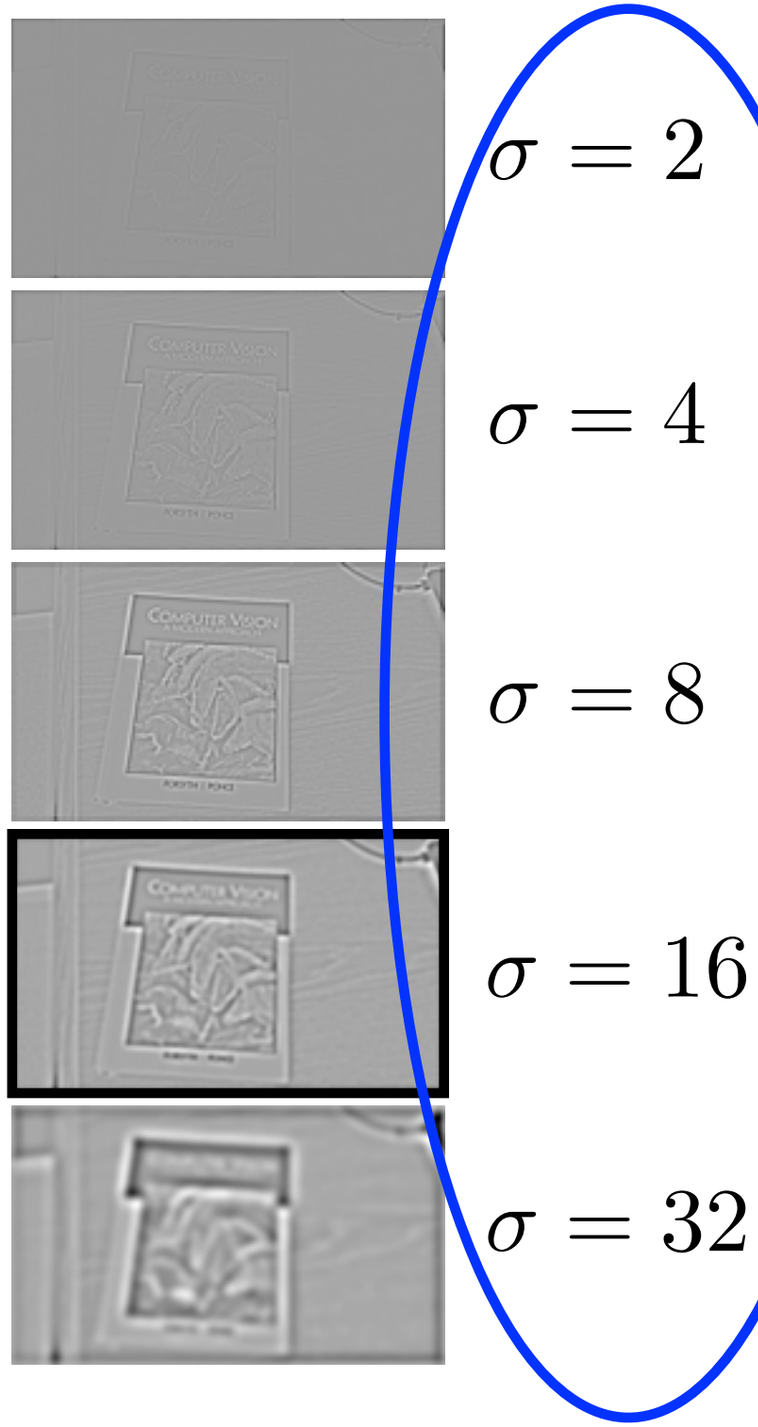
SIFT: Scale Invariant Feature Transform

Difference of Gaussian (DoG) pyramid

These are power of 2, and start at 2.

Can also be powers of sqrt(2),
or powers of any other number.

Need not start at 2.



SIFT: Scale Invariant Feature Transform

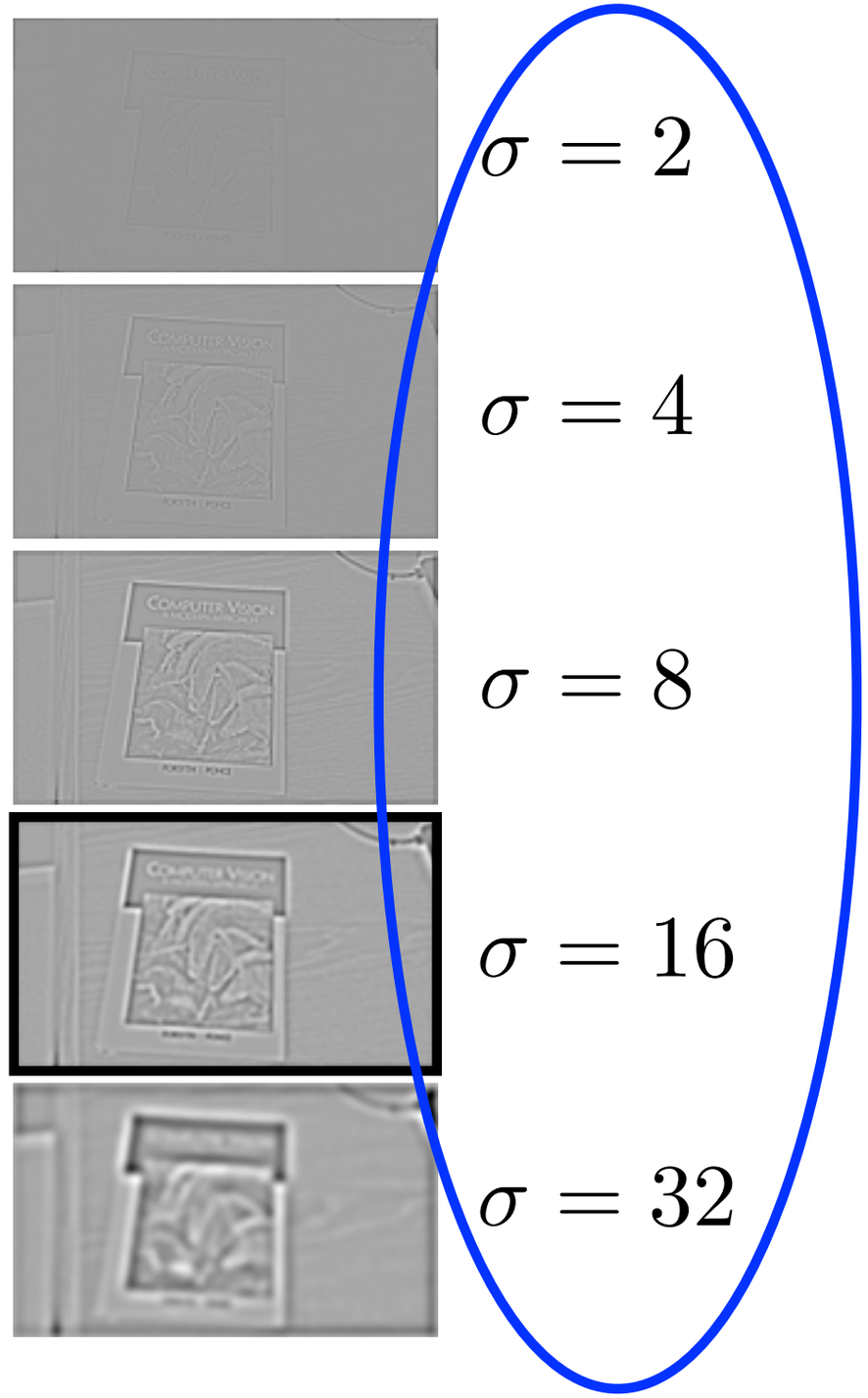
Difference of Gaussian (DoG) pyramid

These are power of 2, and start at 2.

Can also be powers of sqrt(2),
or powers of any other number.

Need not start at 2.

Both starting sigma and power
are determined empirically.



SIFT: Scale Invariant Feature Transform

Why use Laplacian or Difference of Gaussian?

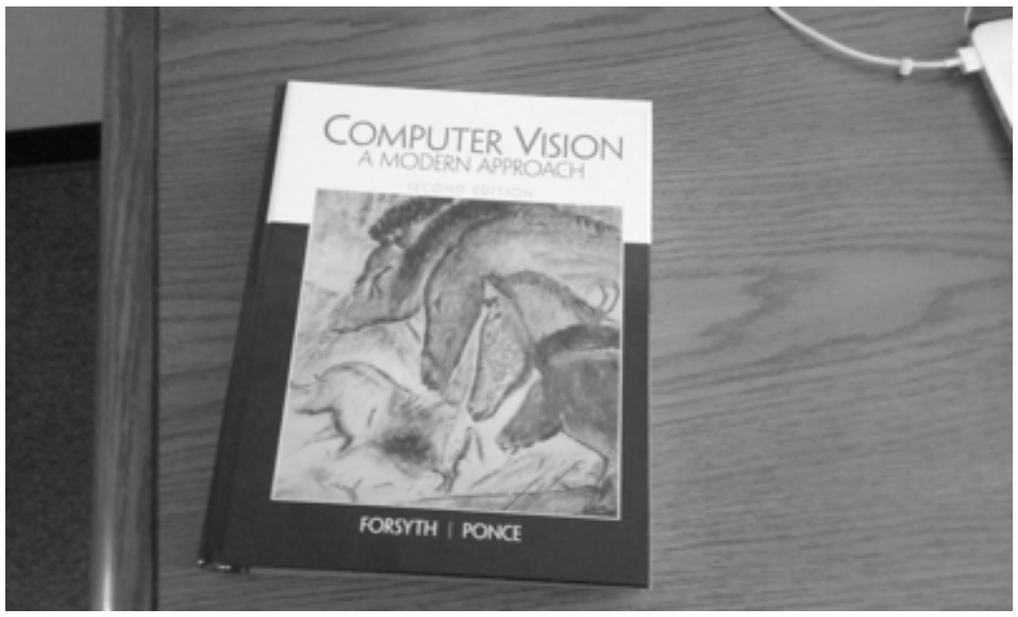
SIFT: Scale Invariant Feature Transform

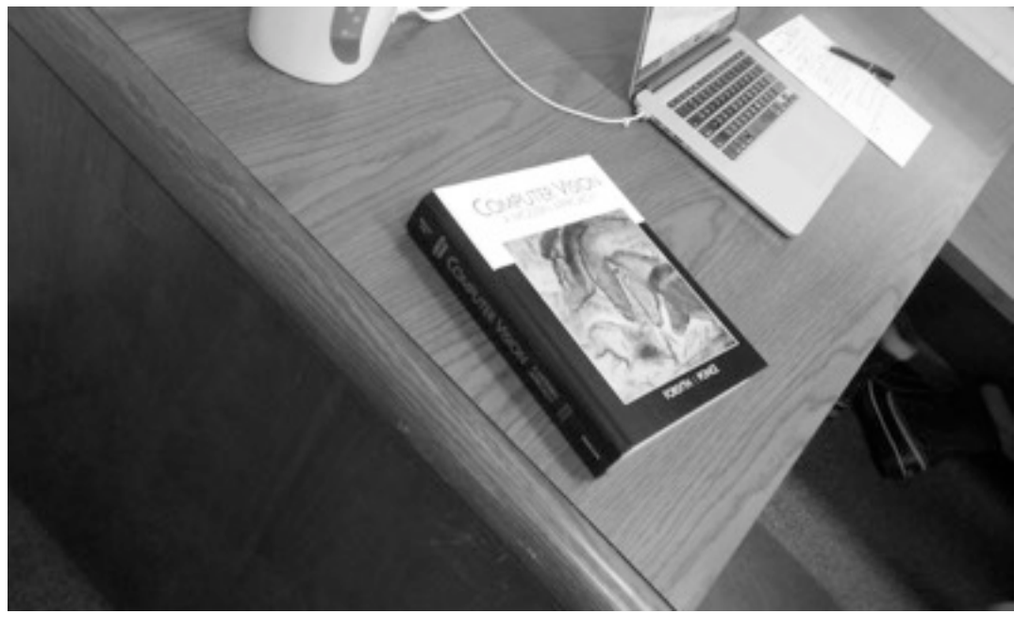
Why use Laplacian or Difference of Gaussian?

Scale Invariance

Rotation Invariance

Illumination Invariance





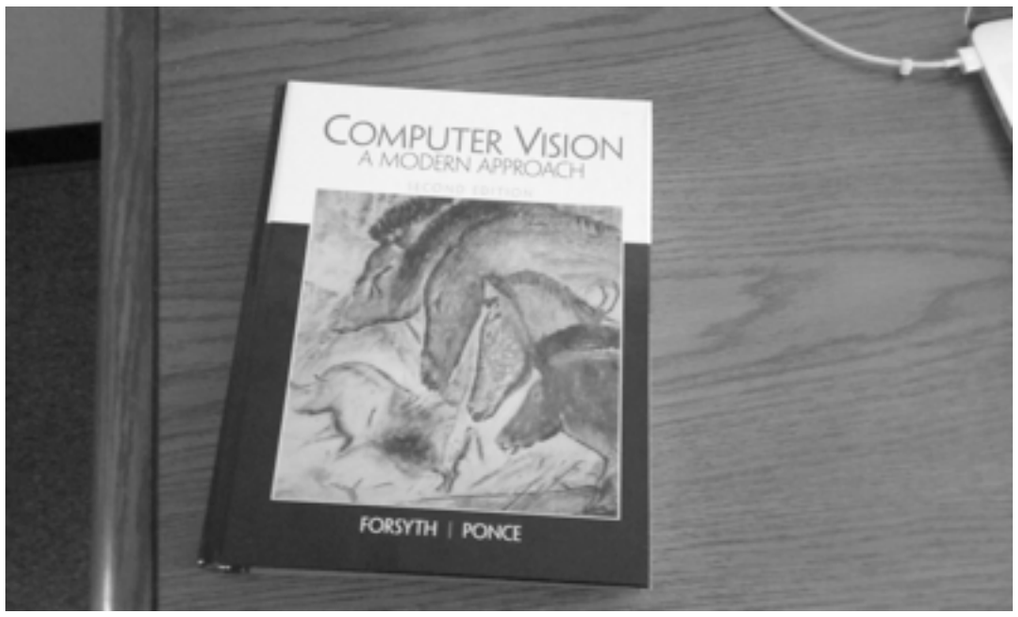
SIFT: Scale Invariant Feature Transform

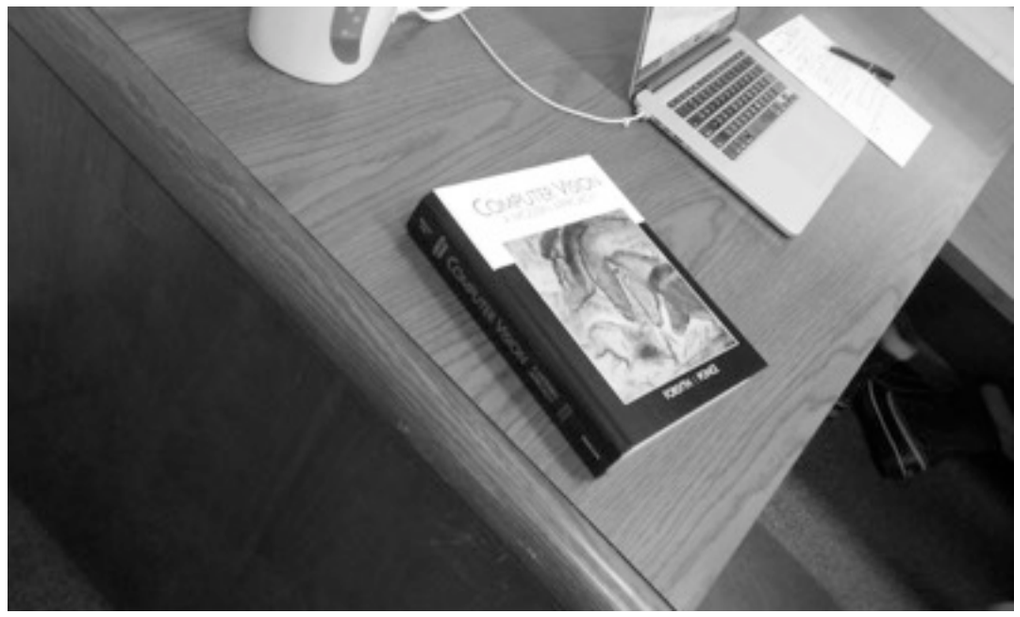
Why use Laplacian or Difference of Gaussian?

Scale Invariance

Rotation Invariance

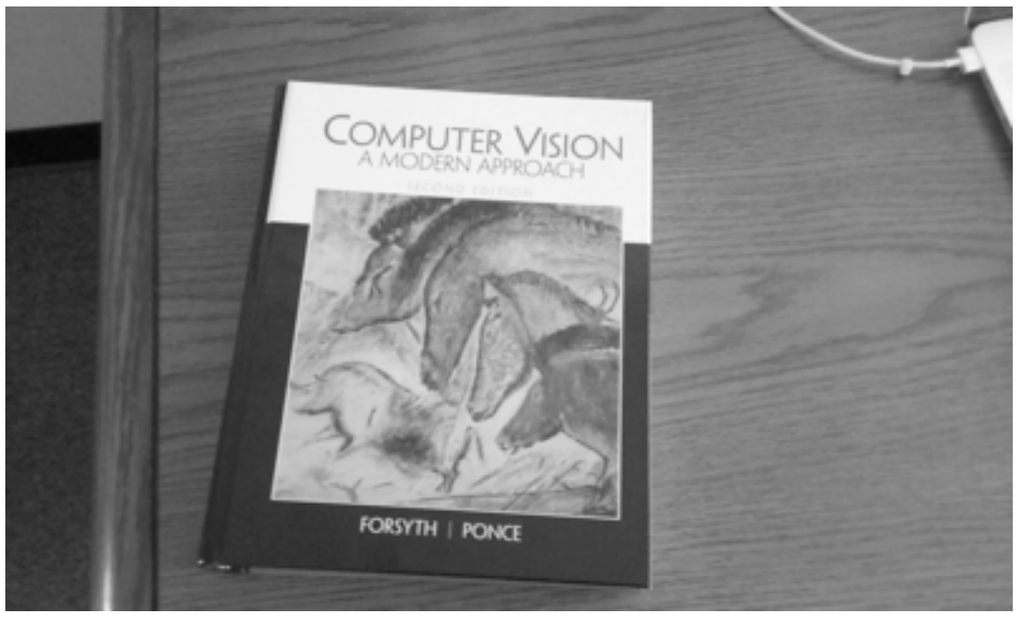
Illumination Invariance

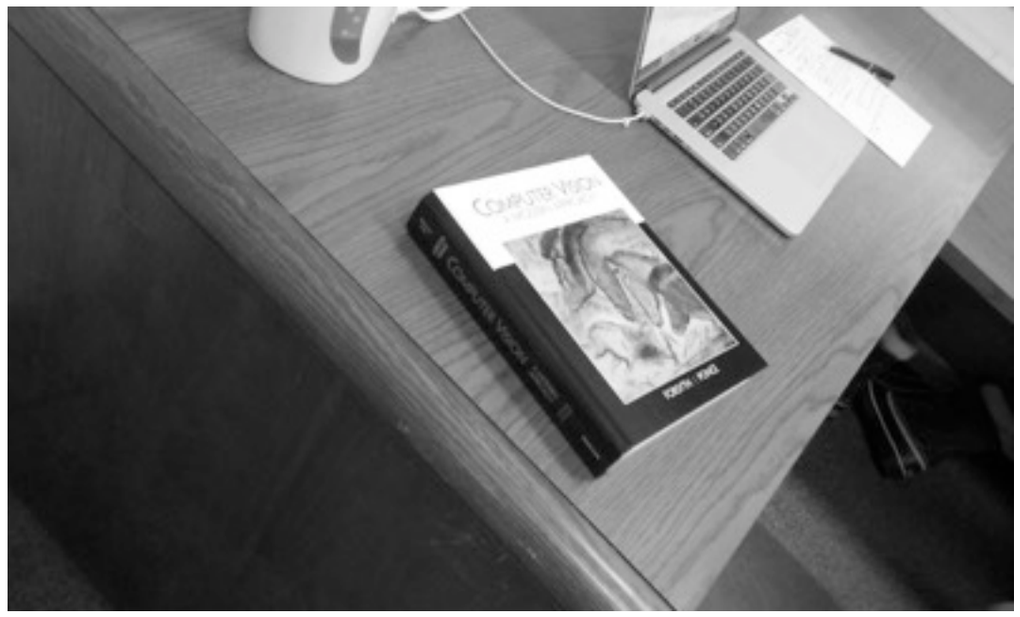




SIFT: Scale Invariant Feature Transform

Laplacian or Difference of Gaussian provides Illumination Invariance

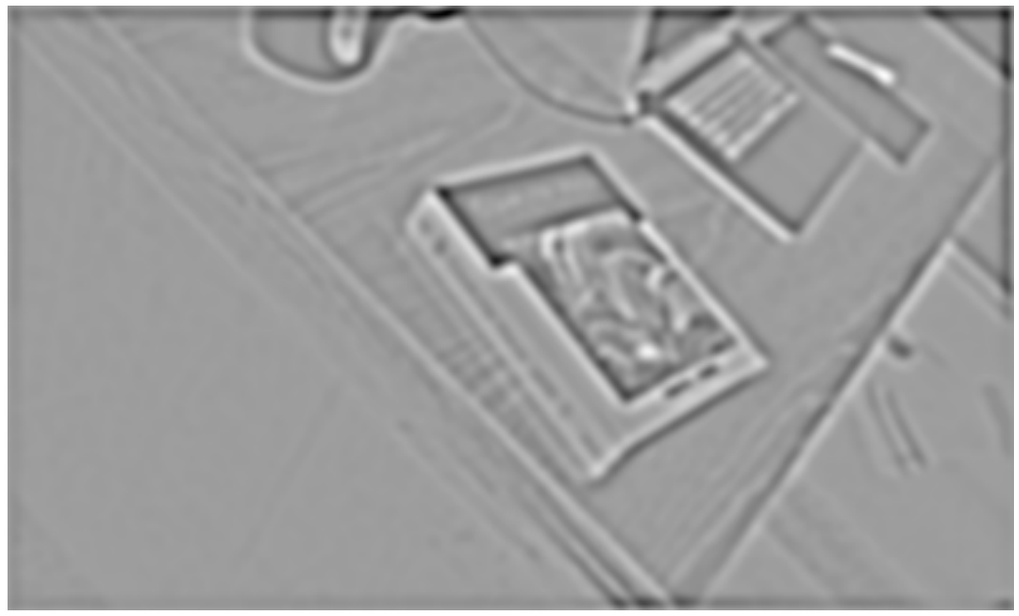




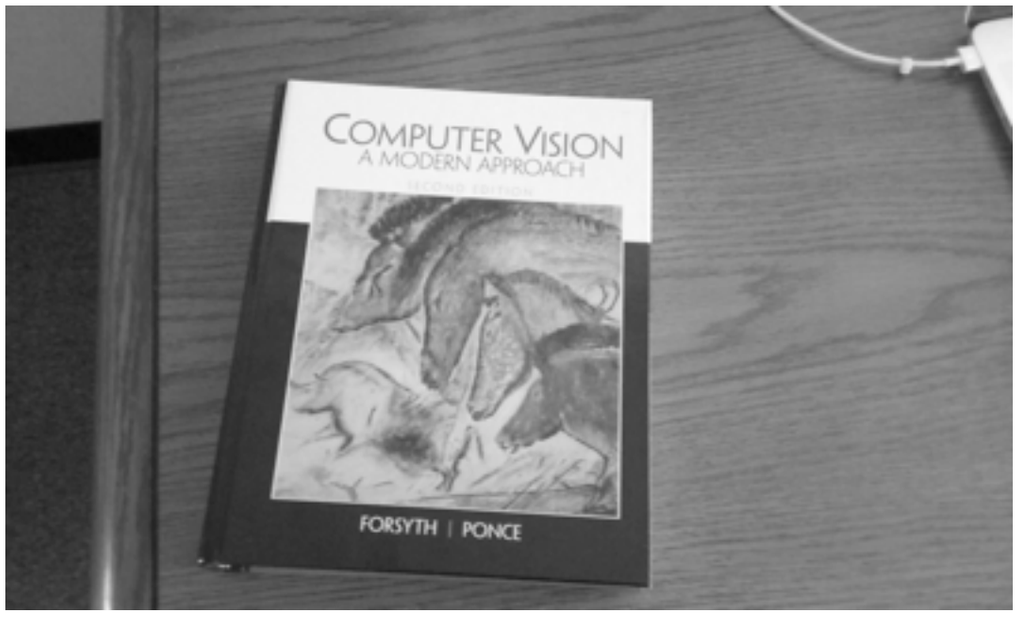
Left image is darker than right image.

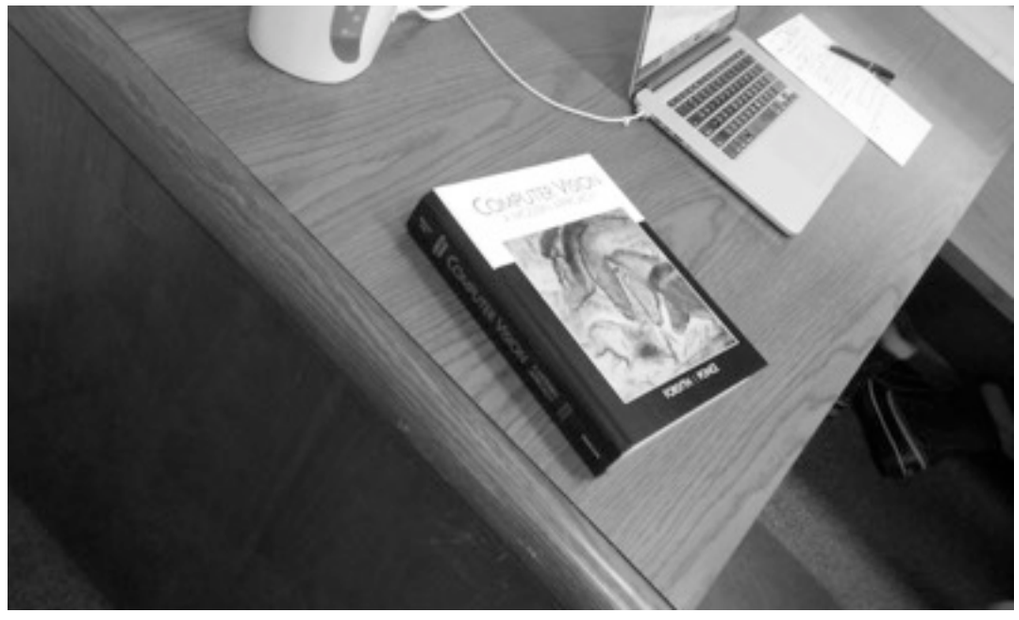
SIFT: Scale Invariant Feature Transform

Laplacian or Difference of Gaussian provides Illumination Invariance



Laplacians of Gaussian have similar brightness.

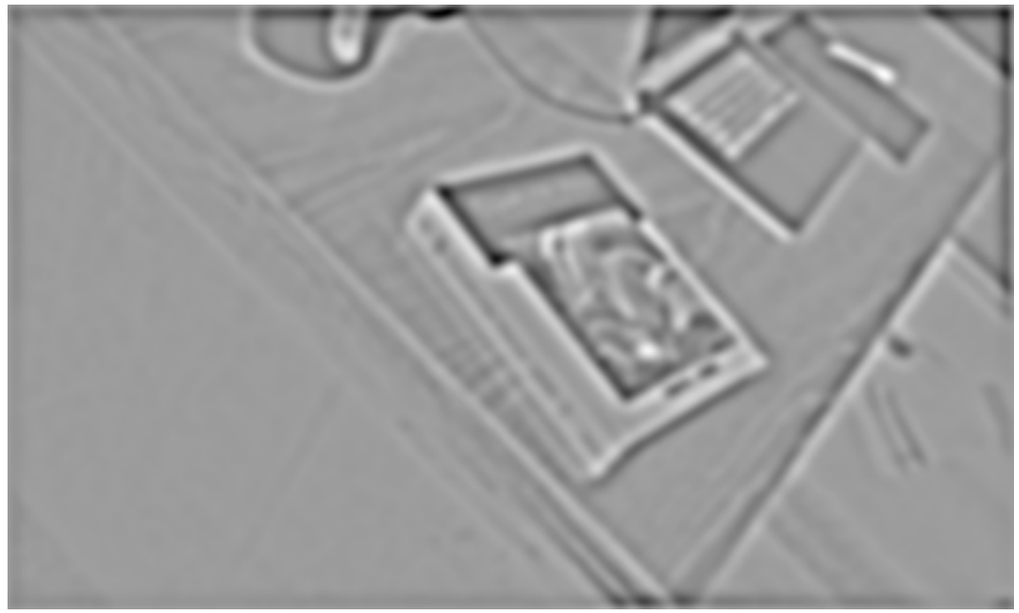




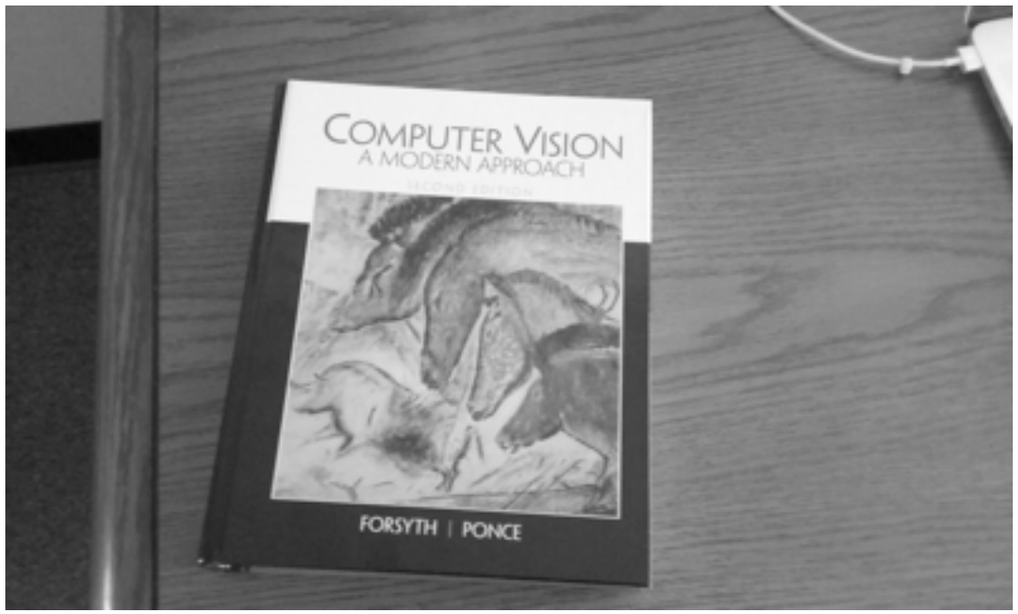
Left image is darker than right image.

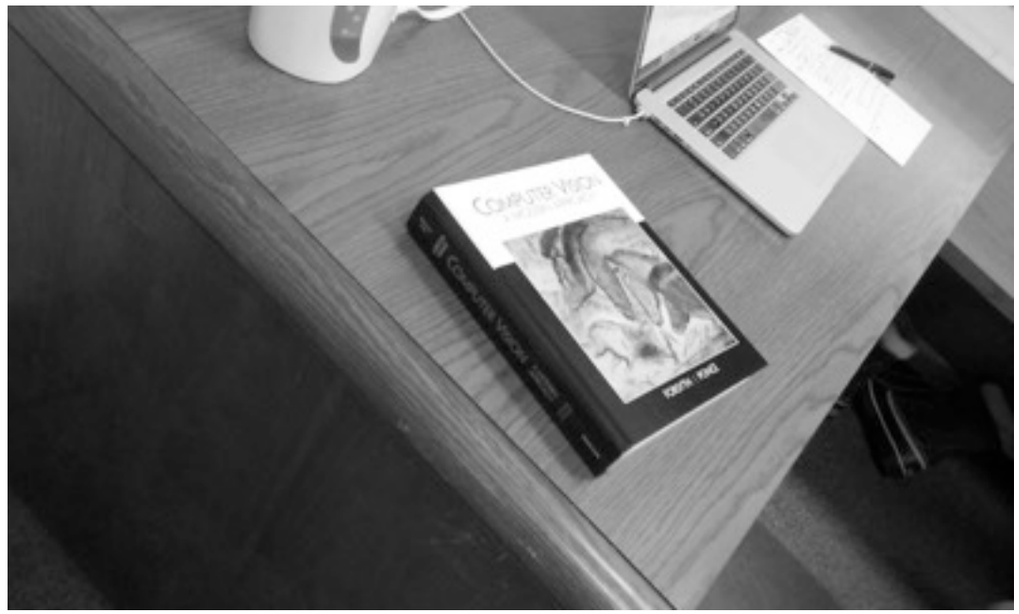
SIFT: Scale Invariant Feature Transform

Laplacian or Difference of Gaussian provides Illumination Invariance



Differences of Gaussian have similar brightness.





Left image is darker than right image.

SIFT Interest Points

Local Extrema in Difference of Gaussian

Start with DoG Pyramid.

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

$\sigma = 16$

$\sigma = 32$

SIFT Interest Points

Local Extrema in Difference of Gaussian

Let us consider a particular DoG.

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

$\sigma = 16$

$\sigma = 32$

SIFT Interest Points

Local Extrema in Difference of Gaussian

Let us consider a particular DoG.

$\sigma = 16$

SIFT Interest Points

Local Extrema in Difference of Gaussian

Get DoG for a standard deviation
slightly smaller and slightly larger.

$\sigma = 16/1.2$

$\sigma = 16$

$\sigma = 16 \times 1.2$

SIFT Interest Points

Local Extrema in Difference of Gaussian

Get DoG for a standard deviation
slightly smaller and slightly larger.

$\sigma = 16/1.2$ $\sigma = 16$ $\sigma = 16 \times 1.2$

Lowe found stability with
multiplicative factors of up to 2.

SIFT Interest Points

Local Extrema in Difference of Gaussian

For every pixel in the center DoG (example pixel shown),

$\sigma = 16/1.2$

$\sigma = 16$

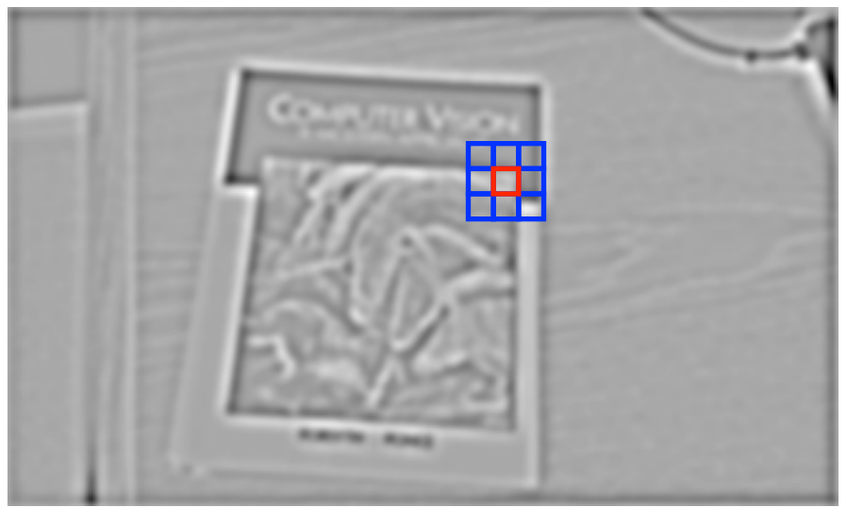
$\sigma = 16 \times 1.2$

SIFT Interest Points

Local Extrema in Difference of Gaussian

For every pixel in the center DoG (example pixel shown), Consider 8 surrounding,

$\sigma = 16/1.2$



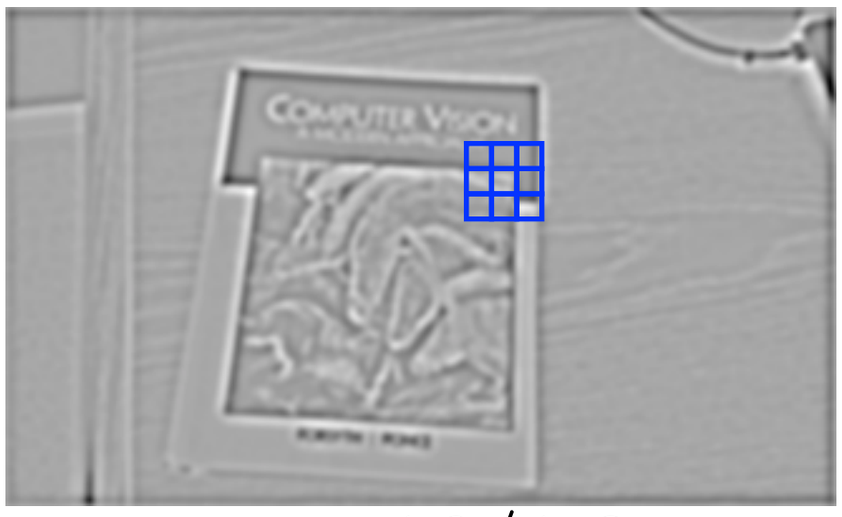
$\sigma = 16$

$\sigma = 16 \times 1.2$

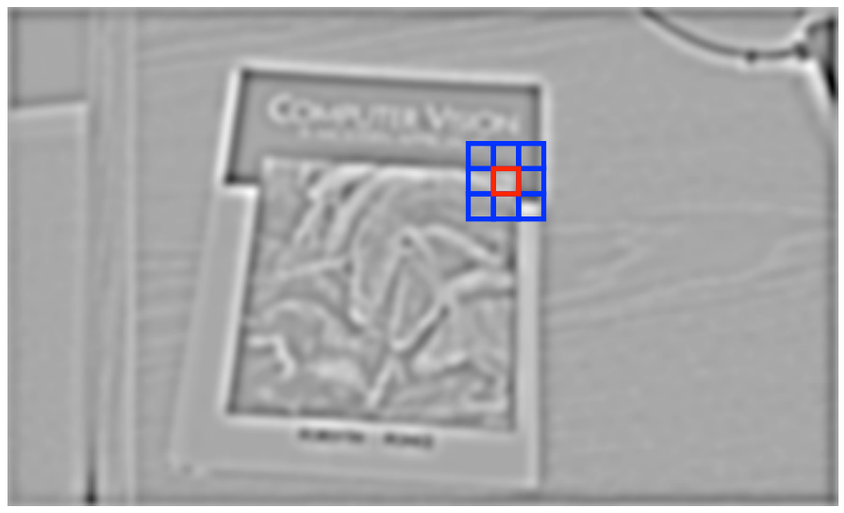
SIFT Interest Points

Local Extrema in Difference of Gaussian

For every pixel in the center DoG (example pixel shown),
Consider 8 surrounding, 9 below,



$\sigma = 16/1.2$



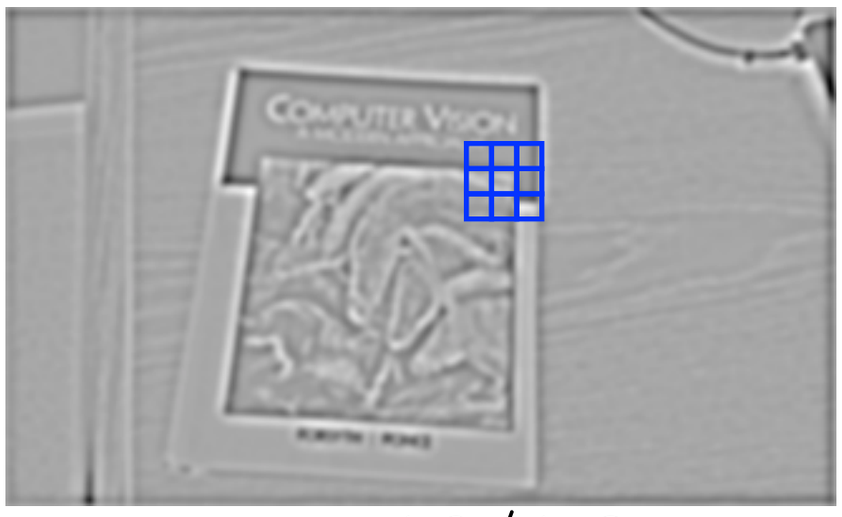
$\sigma = 16$

$\sigma = 16 \times 1.2$

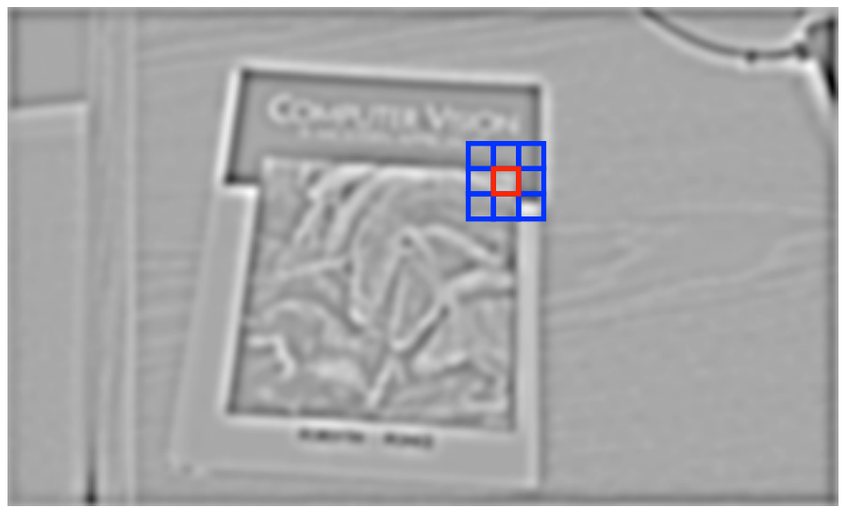
SIFT Interest Points

Local Extrema in Difference of Gaussian

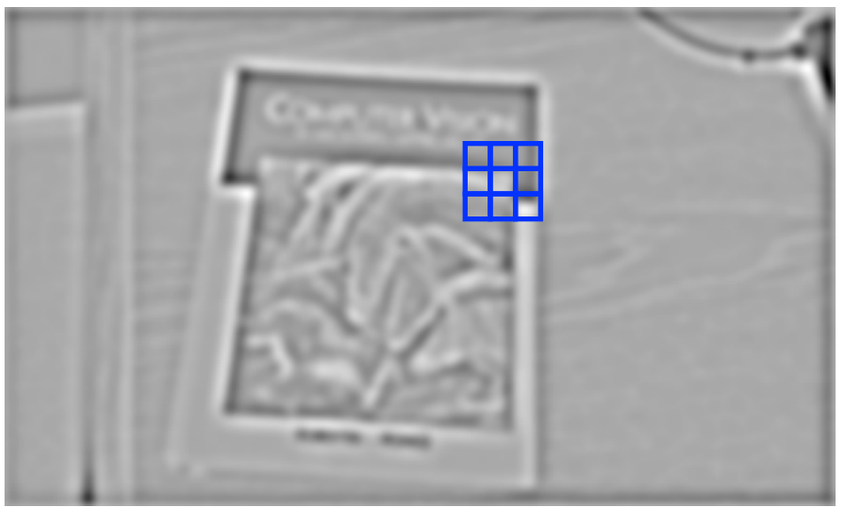
For every pixel in the center DoG (example pixel shown),
Consider 8 surrounding, 9 below, and 9 above.



$\sigma = 16/1.2$



$\sigma = 16$



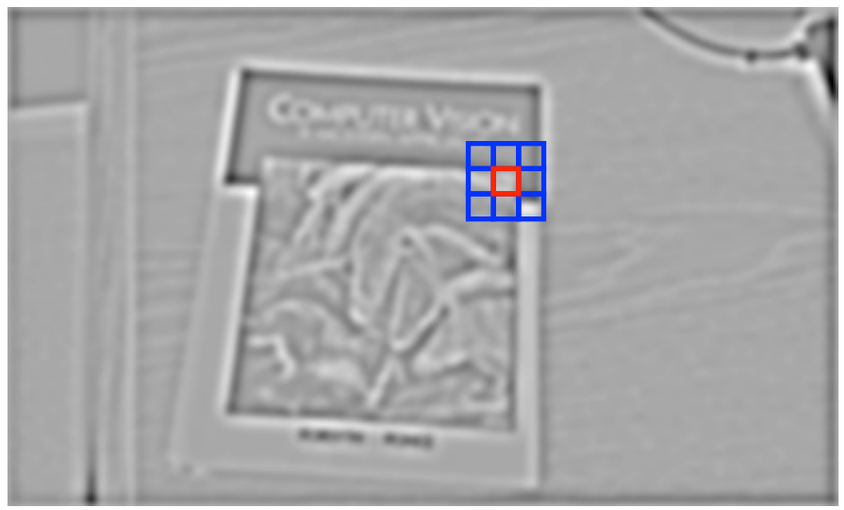
$\sigma = 16 \times 1.2$

SIFT Interest Points

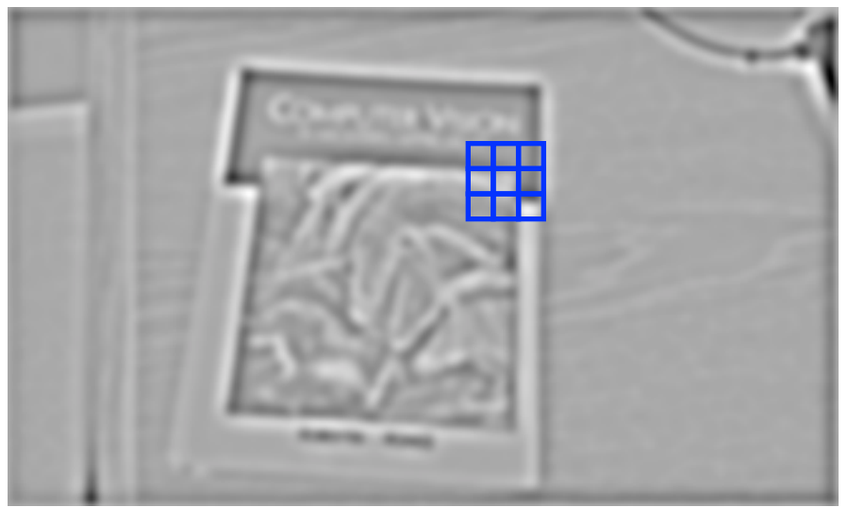
Local Extrema in Difference of Gaussian

For every pixel in the center DoG (example pixel shown),
Consider 8 surrounding, 9 below, and 9 above.

$\sigma = 16/1.2$



$\sigma = 16$



$\sigma = 16 \times 1.2$

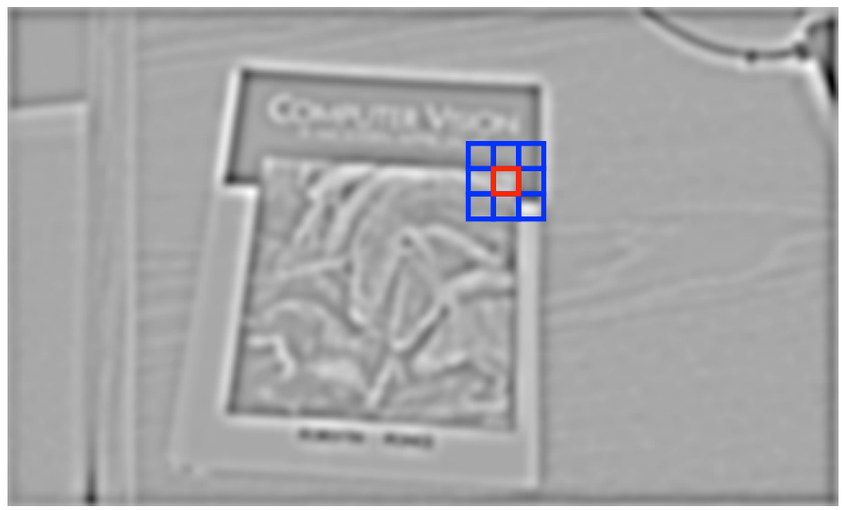
Retain pixel if its DoG value is
the **maximum** or **minimum** of all 26 pixels.

Maxima and minima are also called extrema.

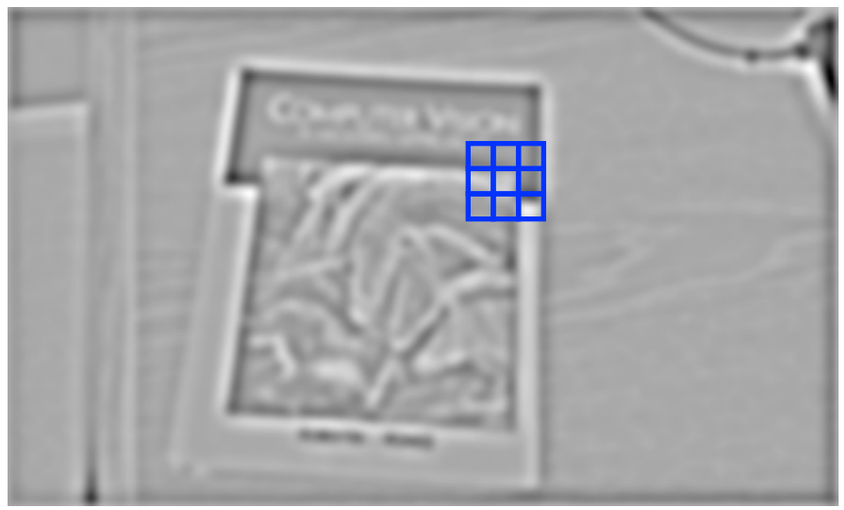
SIFT Interest Points

Local Extrema in Difference of Gaussian

$\sigma = 16/1.2$



$\sigma = 16$



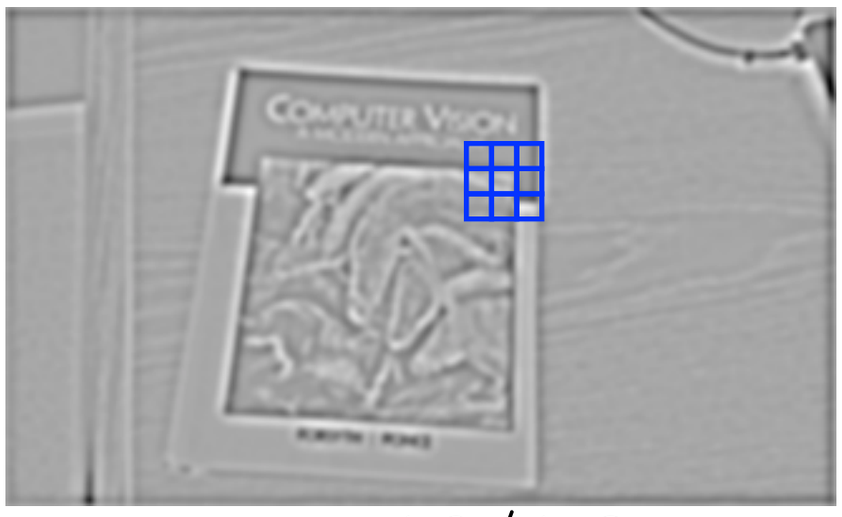
$\sigma = 16 \times 1.2$

Extrema in **scale-space**.

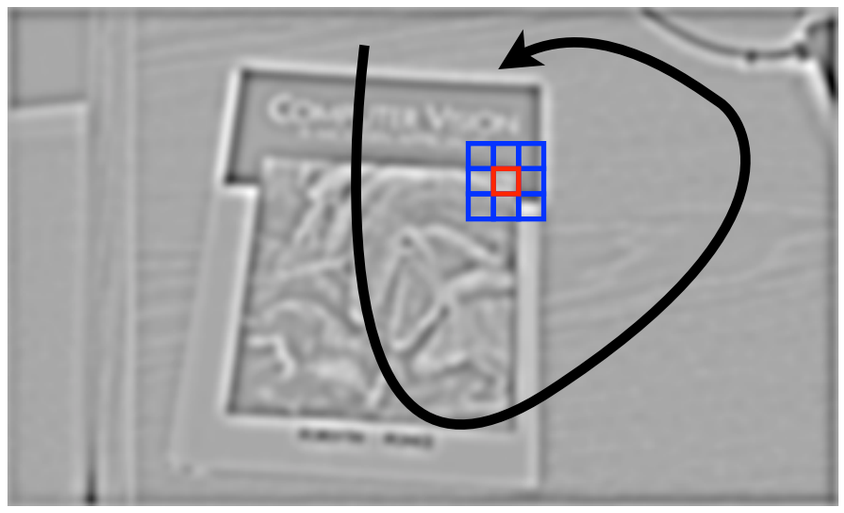
SIFT Interest Points

Local Extrema in Difference of Gaussian

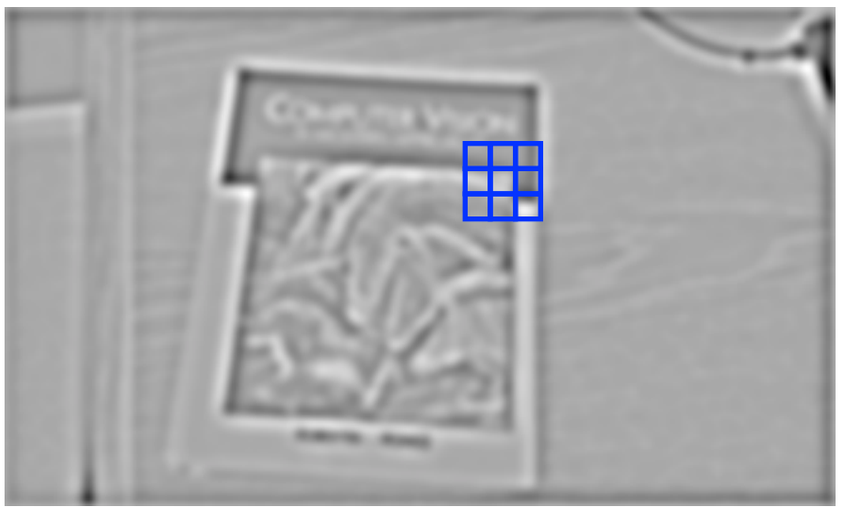
By searching in x and y, we obtain extrema in space.



$\sigma = 16/1.2$



$\sigma = 16$



$\sigma = 16 \times 1.2$

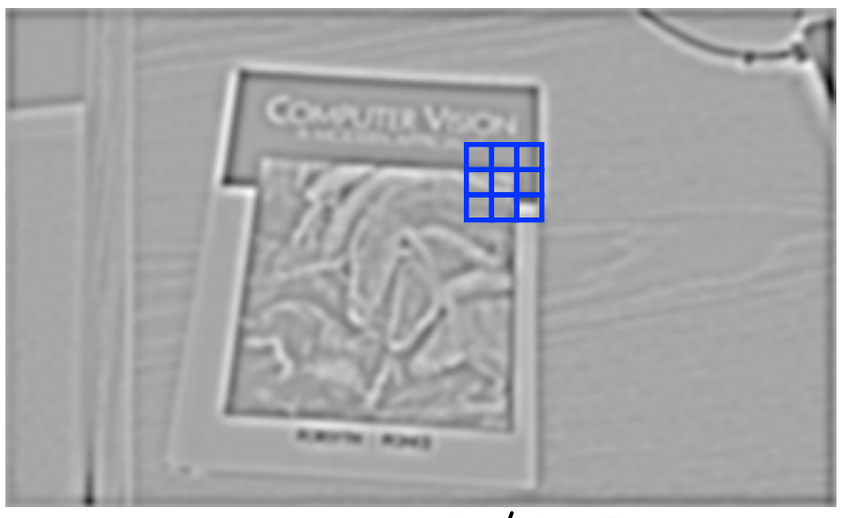
Extrema in **scale-space**.

SIFT Interest Points

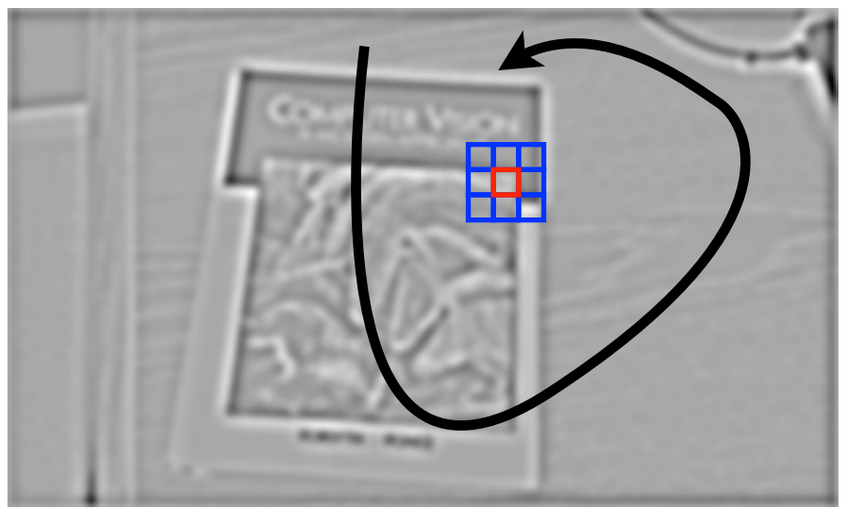
Local Extrema in Difference of Gaussian

By searching in x and y, we obtain extrema in space.

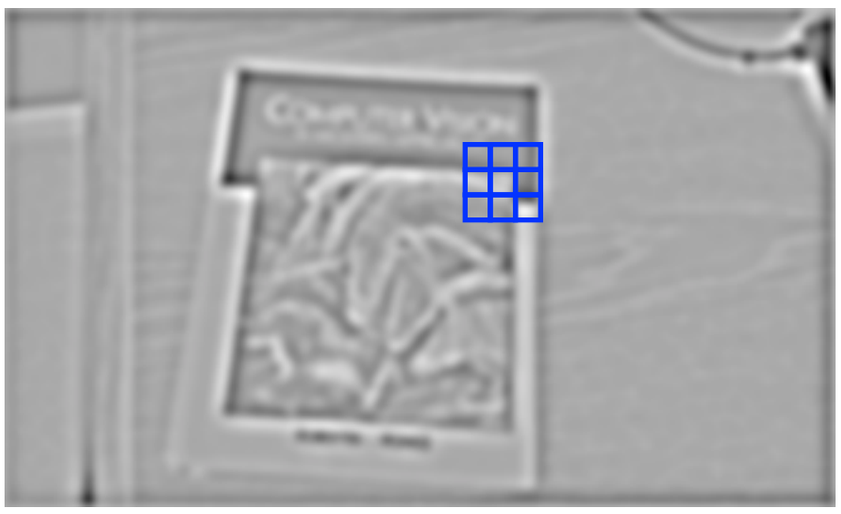
By searching in standard deviation, we obtain extrema in scale.



$\sigma = 16/1.2$



$\sigma = 16$

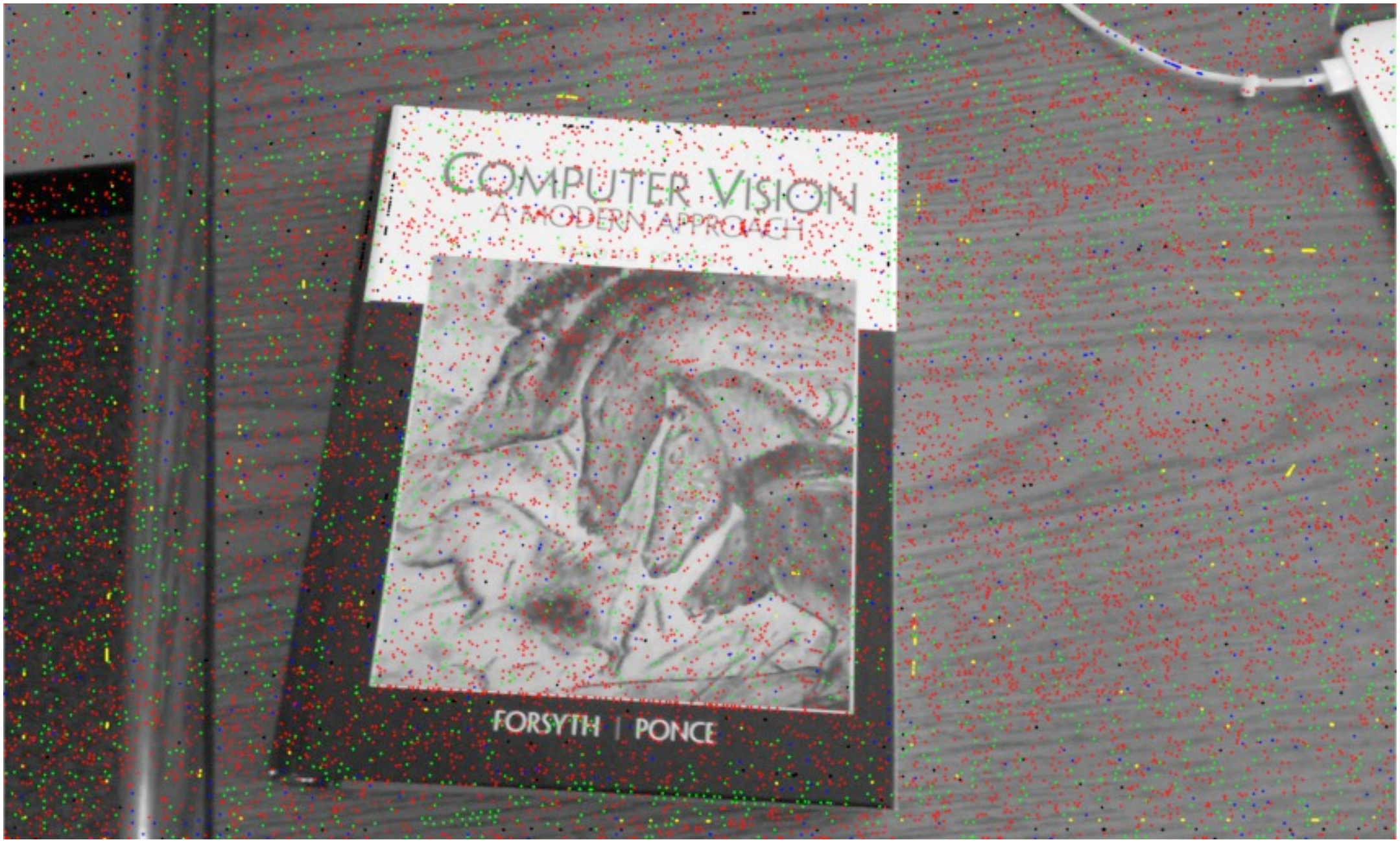


$\sigma = 16 \times 1.2$

Extrema in **scale-space**.

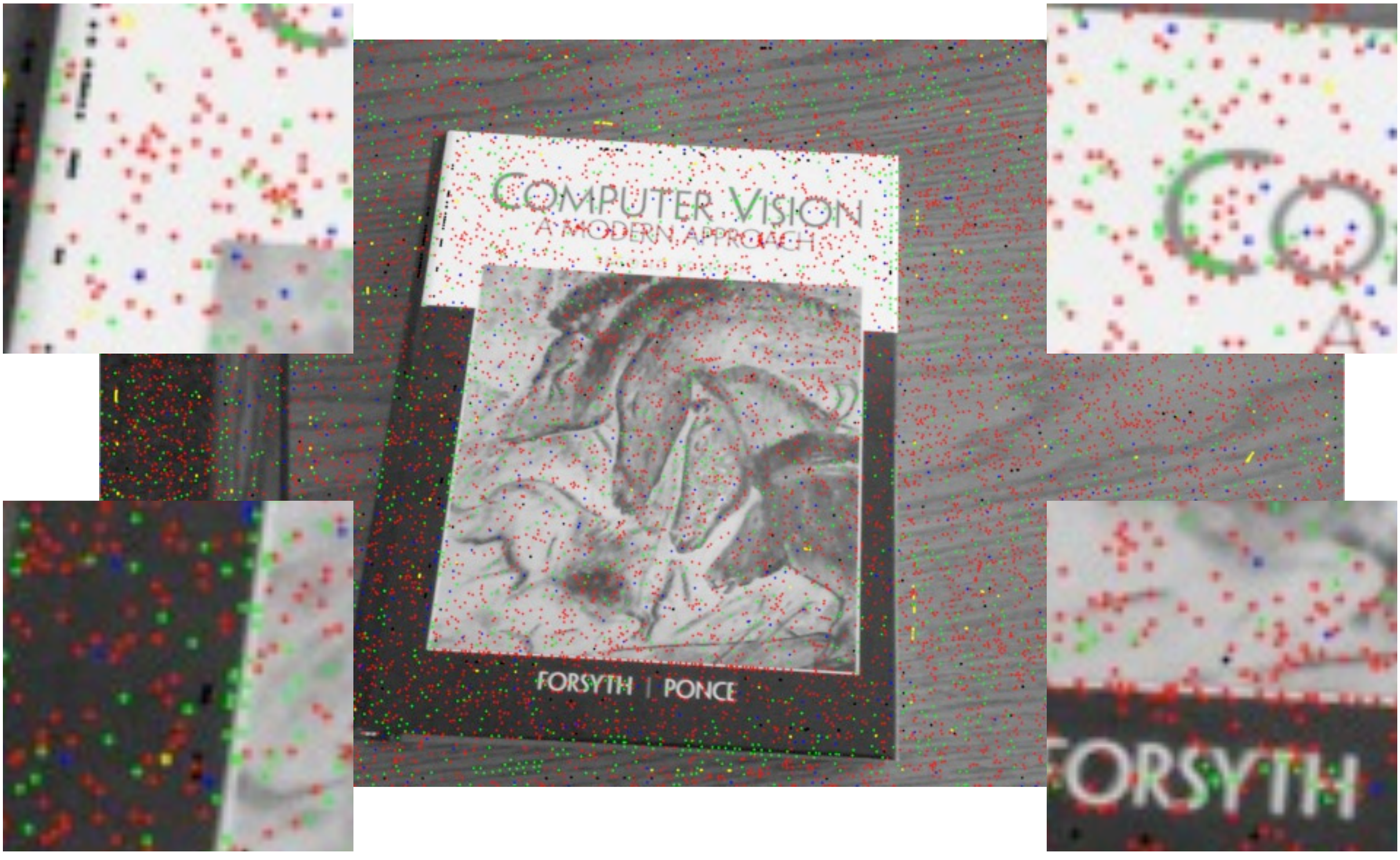
SIFT Interest Points

Local Extrema in Difference of Gaussian



SIFT Interest Points

Local Extrema in Difference of Gaussian



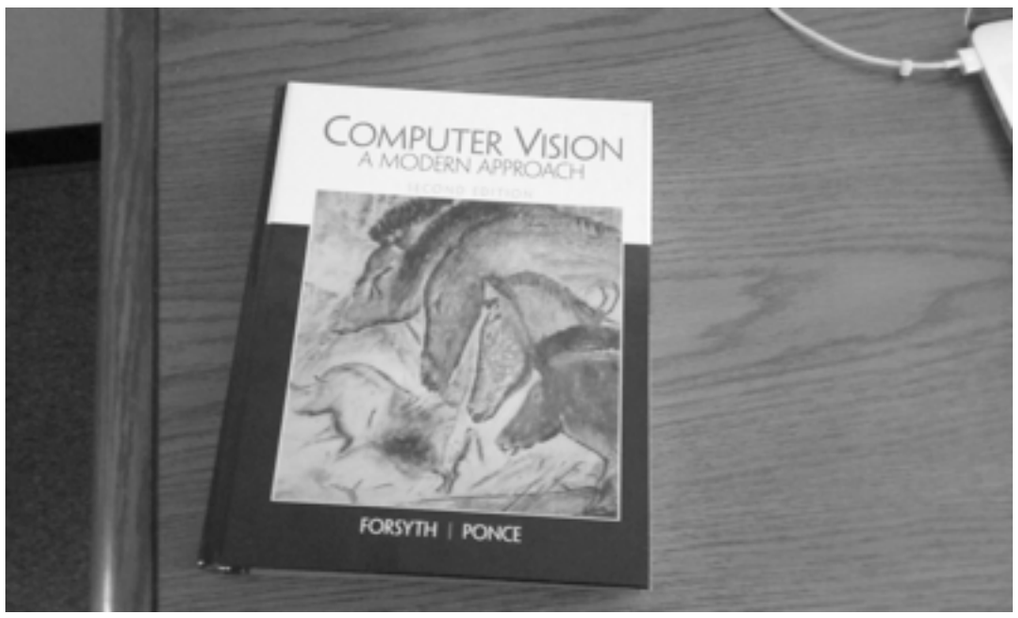
SIFT Interest Points

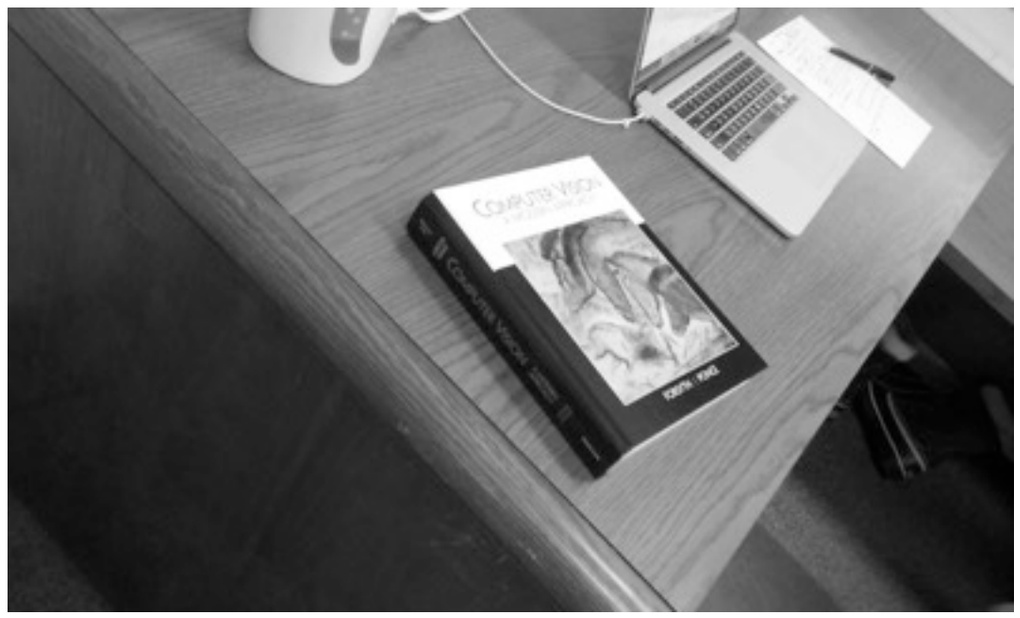
Why Local Extrema in Difference of Gaussian?

Scale Invariance

Rotation Invariance

Illumination Invariance





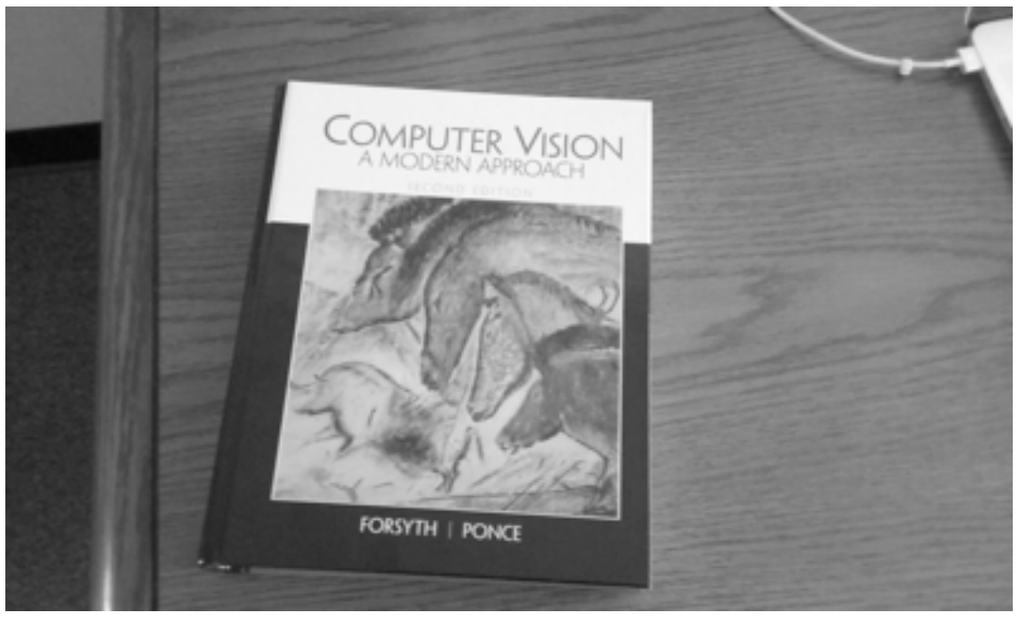
SIFT Interest Points

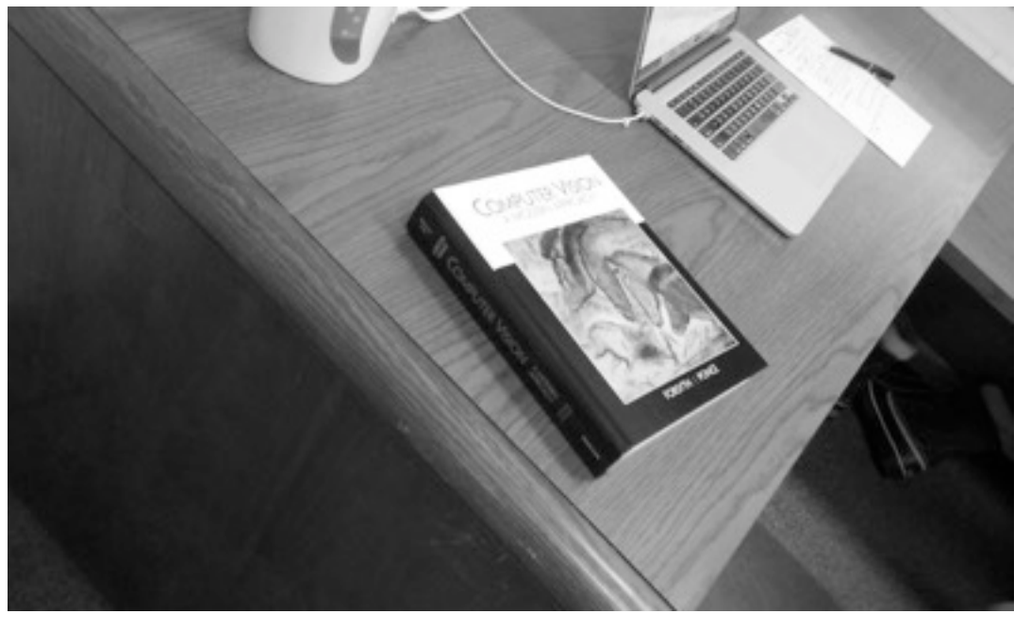
Why Local Extrema in Difference of Gaussian?

Scale Invariance

Rotation Invariance

Illumination Invariance





SIFT Interest Points

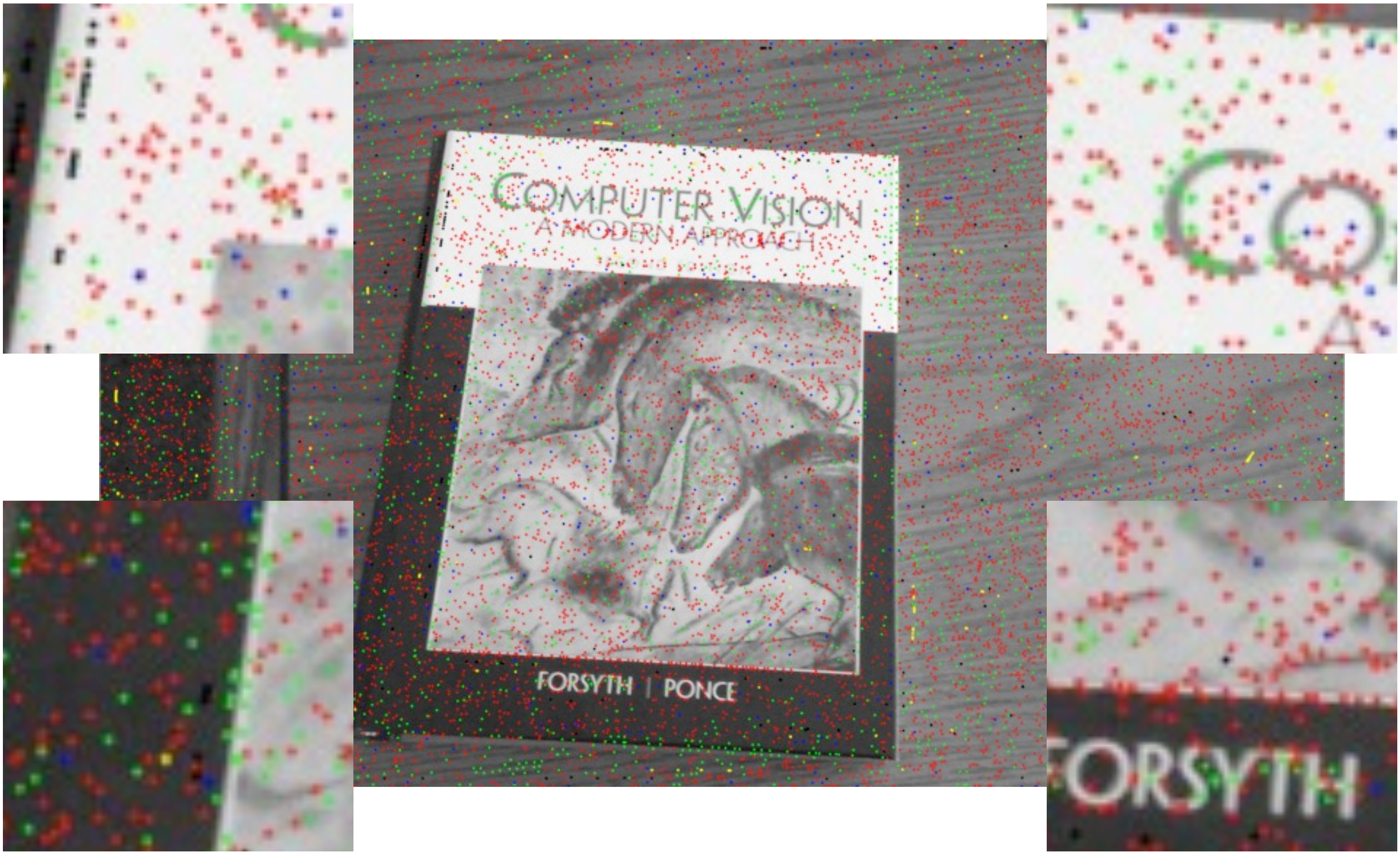
Why Local Extrema in Difference of Gaussian?

Extrema in DoG are stable over scale changes.

You can go closer or farther from the object and still detect those points.

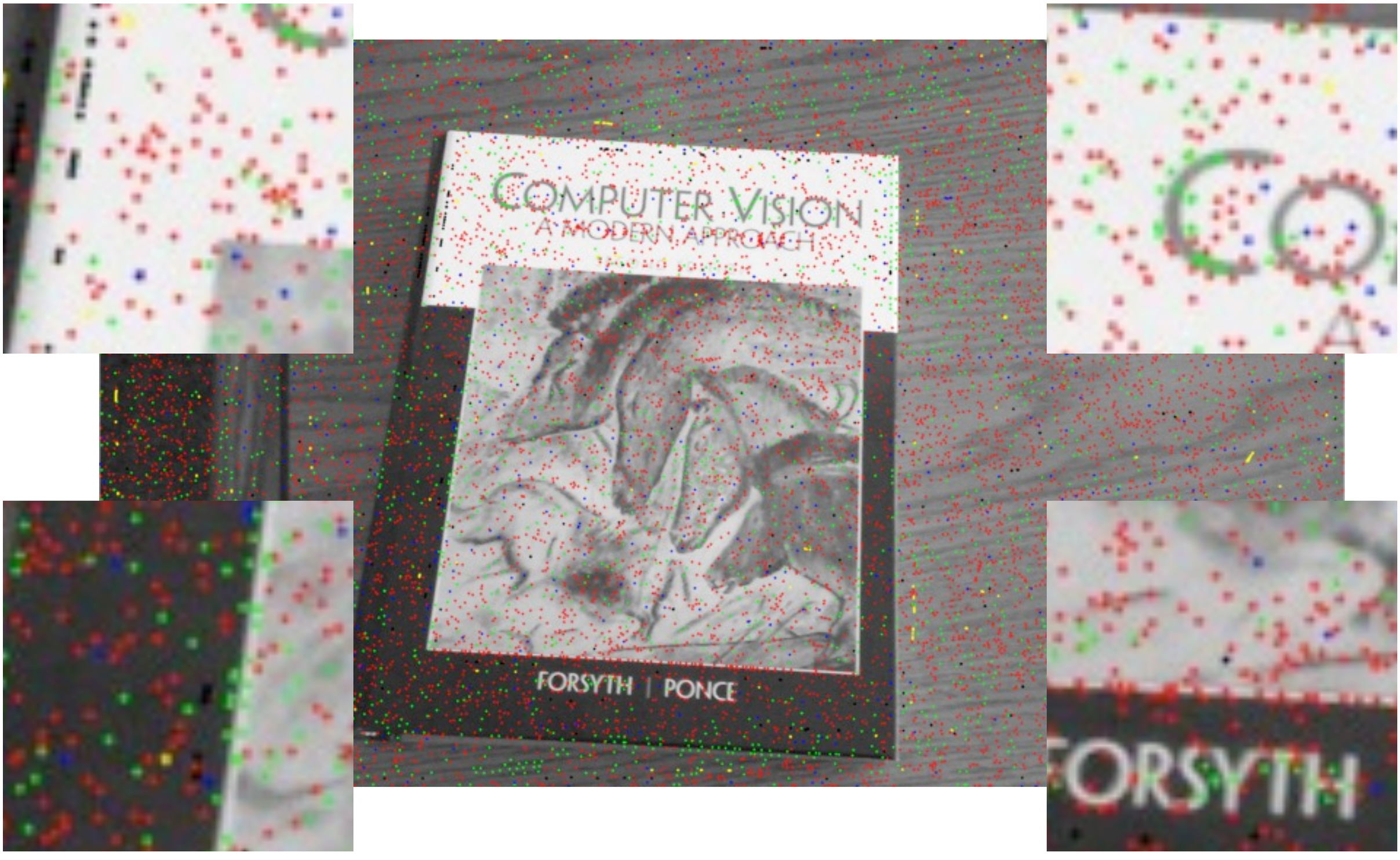
SIFT Interest Points

Local Extrema in Difference of Gaussian



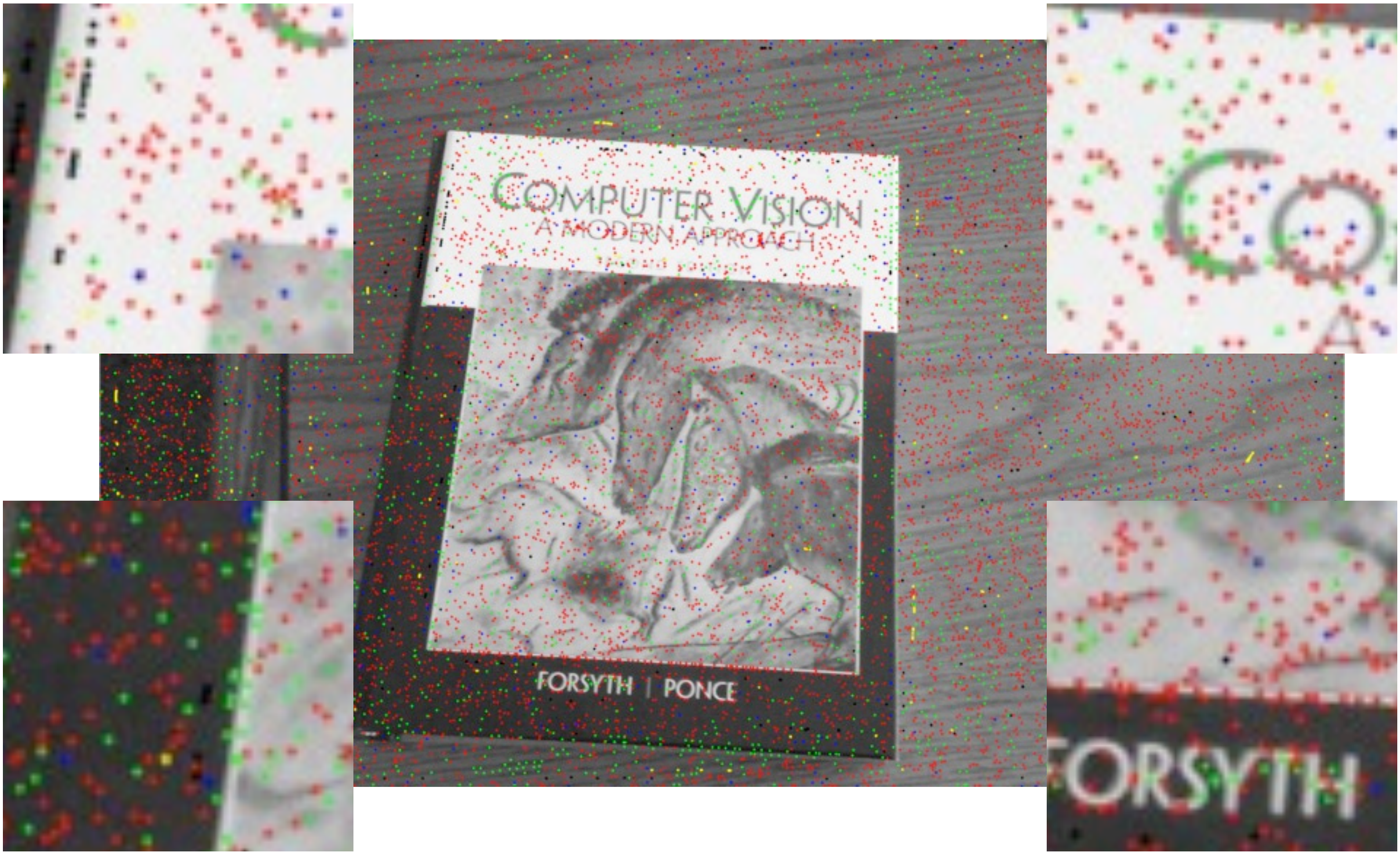
SIFT Interest Points

Some of these are not necessary (edge points, uniform region points)



SIFT Interest Points

Use Corner Detector To Eliminate Them



SIFT Interest Points

Use Corner Detector To Eliminate Them

Earlier we saw Harris corner detector.

Corner is found where eigenvalues of Harris matrix are large.

$$\mathbf{M}^{xy} = \sum_x \sum_y \begin{bmatrix} \mathbf{I}_x^2(x,y) & \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) \\ \mathbf{I}_x(x,y)\mathbf{I}_y(x,y) & \mathbf{I}_y^2(x,y) \end{bmatrix} \mathbf{W}(x,y)$$

Gradient (or first derivatives) of image

SIFT Interest Points

Use Corner Detector To Eliminate Them

For SIFT, Lowe recommends using Hessian matrix.

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

Second derivatives of DoG!

SIFT Interest Points

Use Corner Detector To Eliminate Them

For SIFT, Lowe recommends using Hessian matrix.

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

Second derivatives of DoG!

D_{xx}: Sobel filter in x followed by Sobel filter in x

D_{xy}: Sobel filter in x followed by Sobel filter in y

D_{yy}: Sobel filter in y followed by Sobel filter in y

SIFT Interest Points

Use Corner Detector To Eliminate Them

For SIFT, Lowe recommends using Hessian matrix.

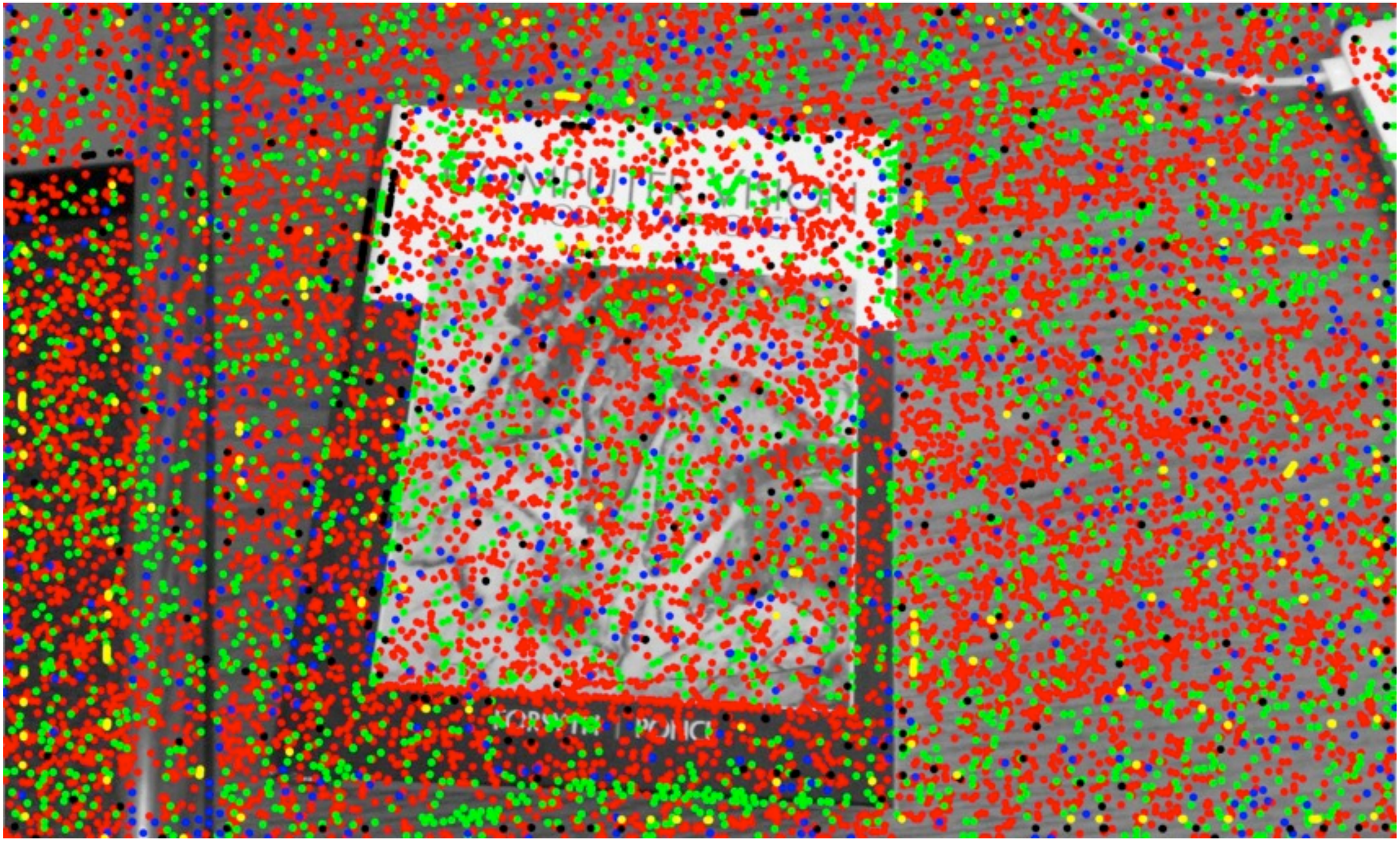
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Second derivatives of DoG!

Find eigenvalues of **H**.
Large eigenvalues indicate corners.

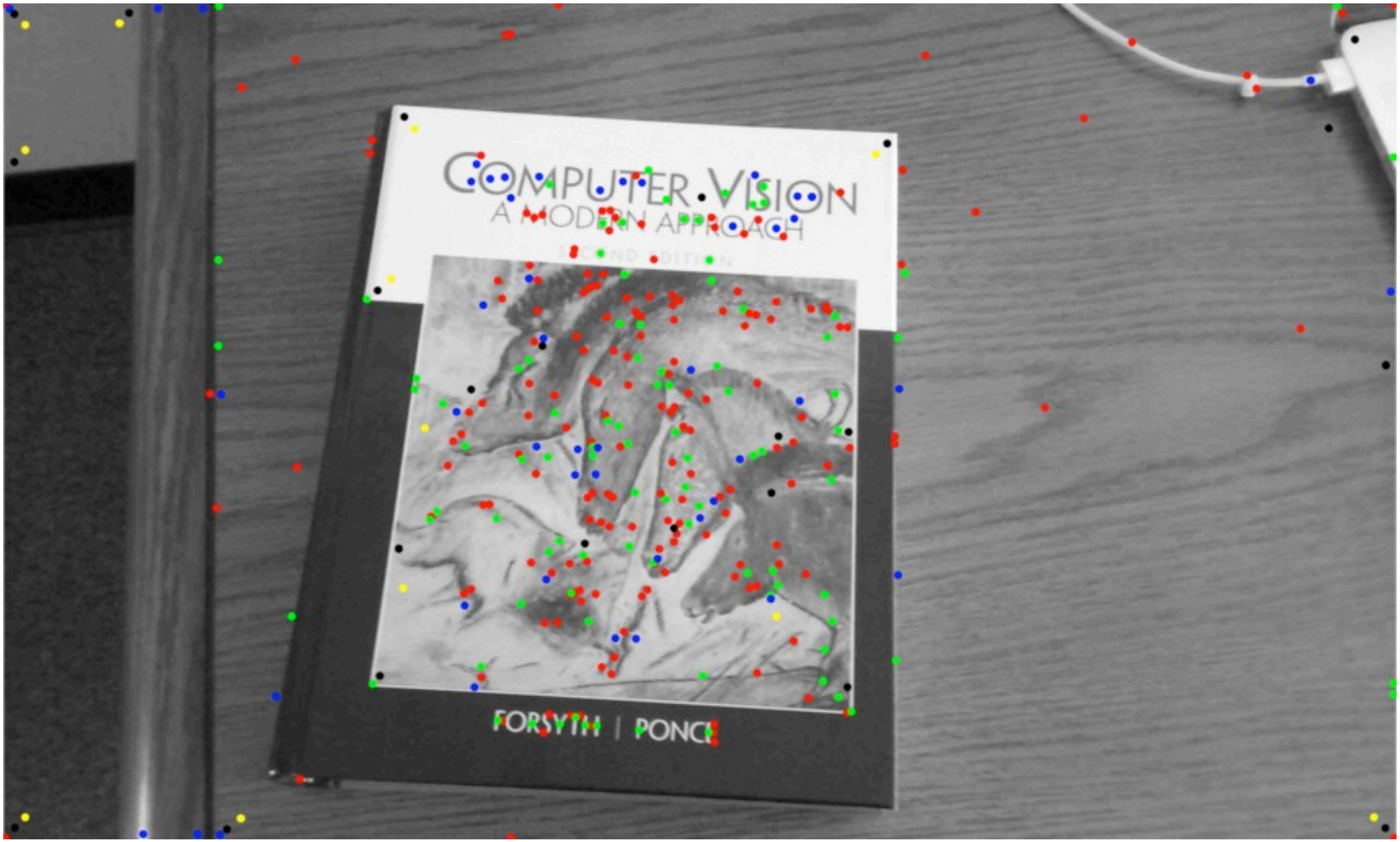
SIFT Interest Points

Only Extrema of DoG in Scale-Space



SIFT Interest Points

Extrema of DoG in Scale-Space + Hessian Corner Detector



Fewer points, largely on corners.

SIFT Interest Points

Scale Invariance

Rotation Invariance

Illumination Invariance

SIFT Interest Points

Scale Invariance

Rotation Invariance

Illumination Invariance DoG

SIFT Interest Points

Scale Invariance	Extrema of DoG scale-space
Rotation Invariance	
Illumination Invariance	DoG

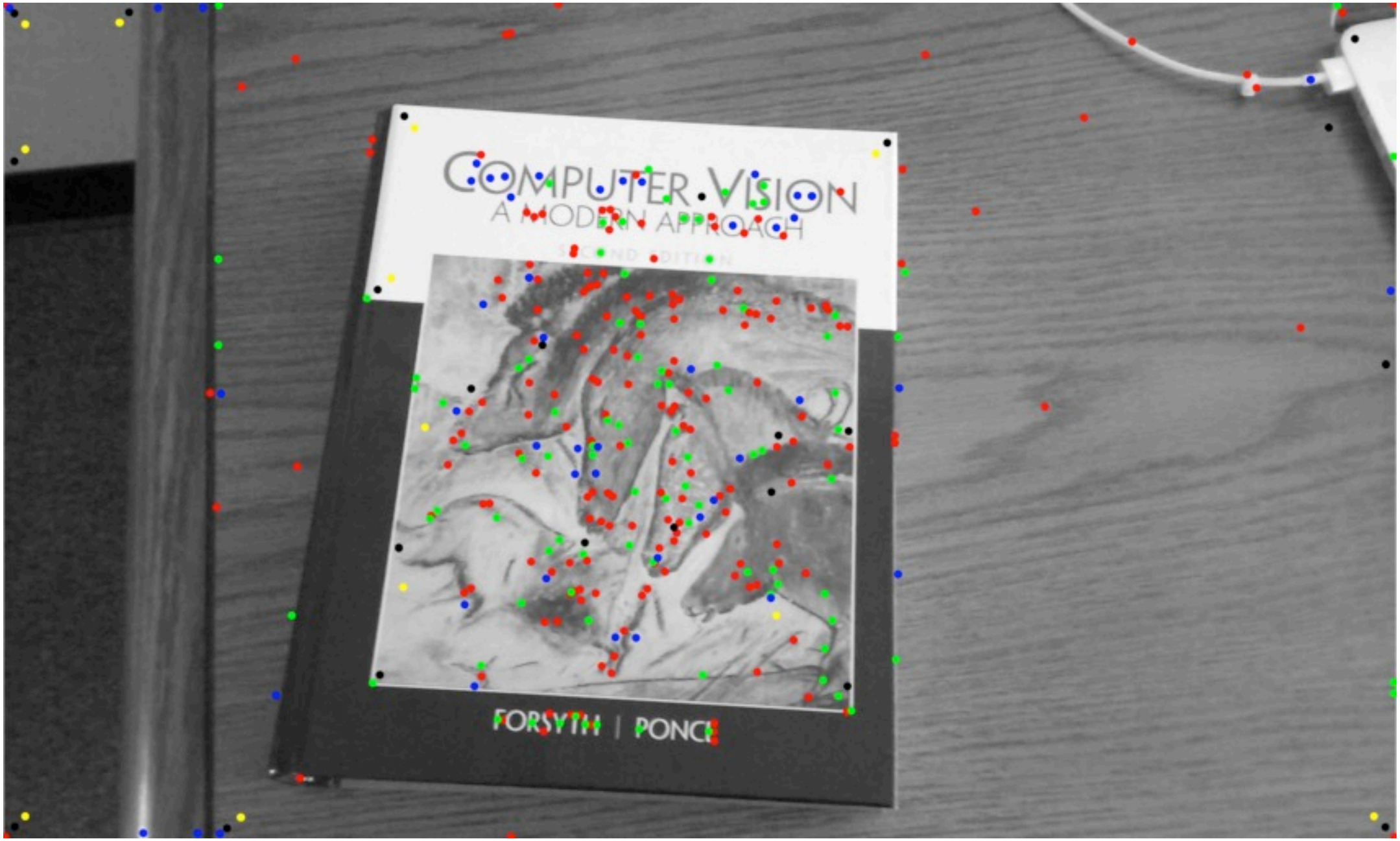
SIFT Interest Points

Assign Orientation To Each Interest Point

Scale Invariance	Extrema of DoG scale-space
Rotation Invariance	
Illumination Invariance	DoG

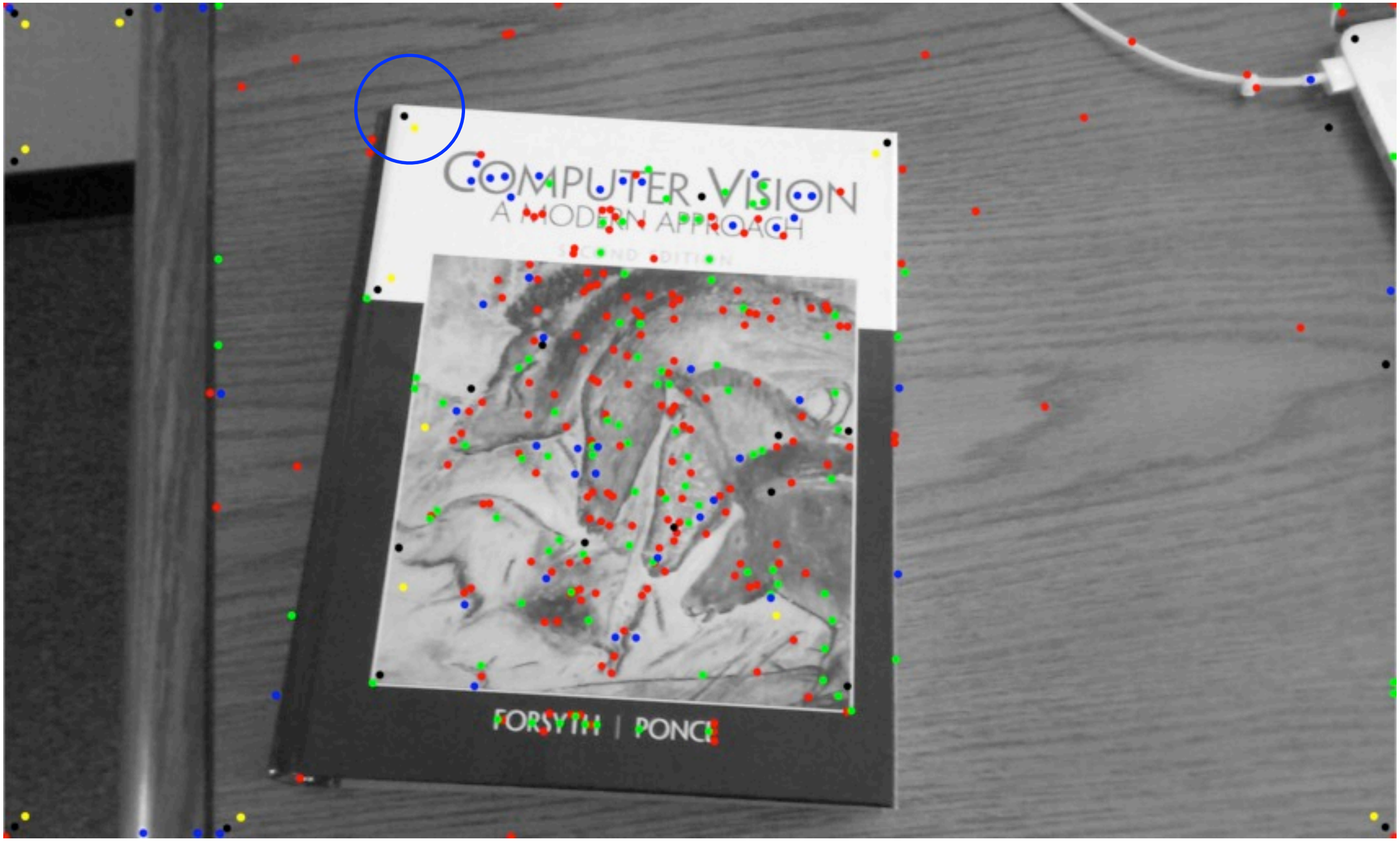
SIFT Interest Points

Assign Orientation To Each Interest Point



SIFT Interest Points

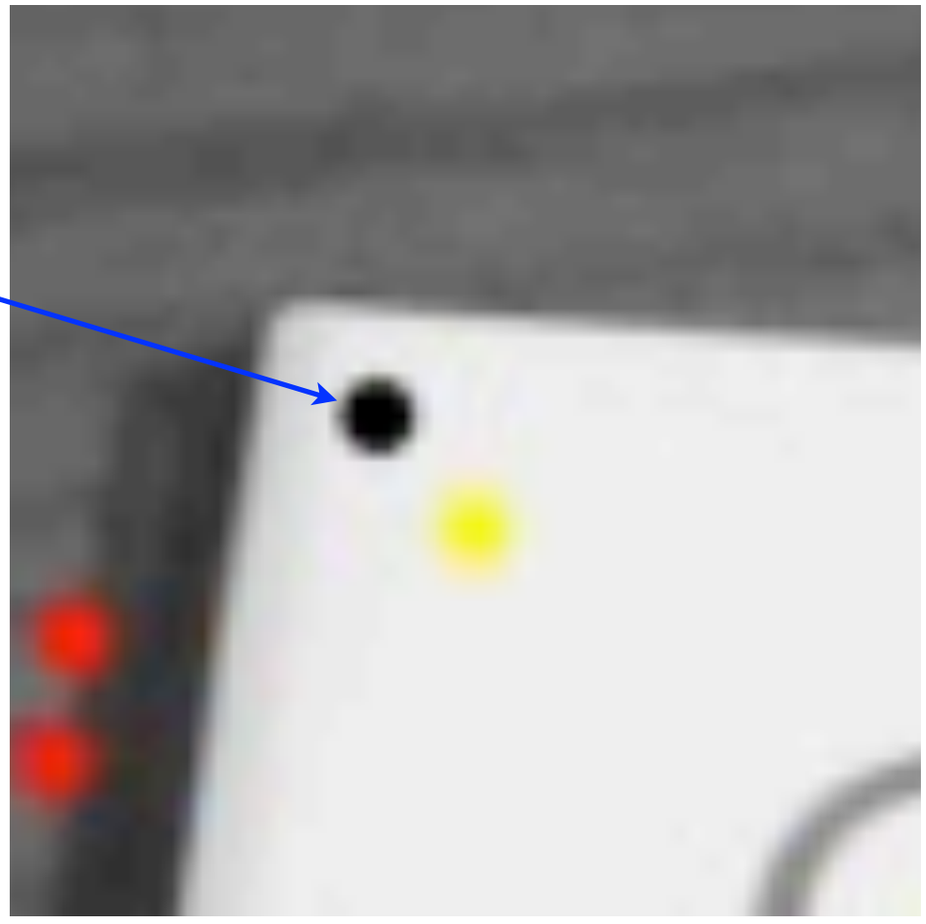
Assign Orientation To Each Interest Point



SIFT Interest Points

Assign Orientation To Each Interest Point

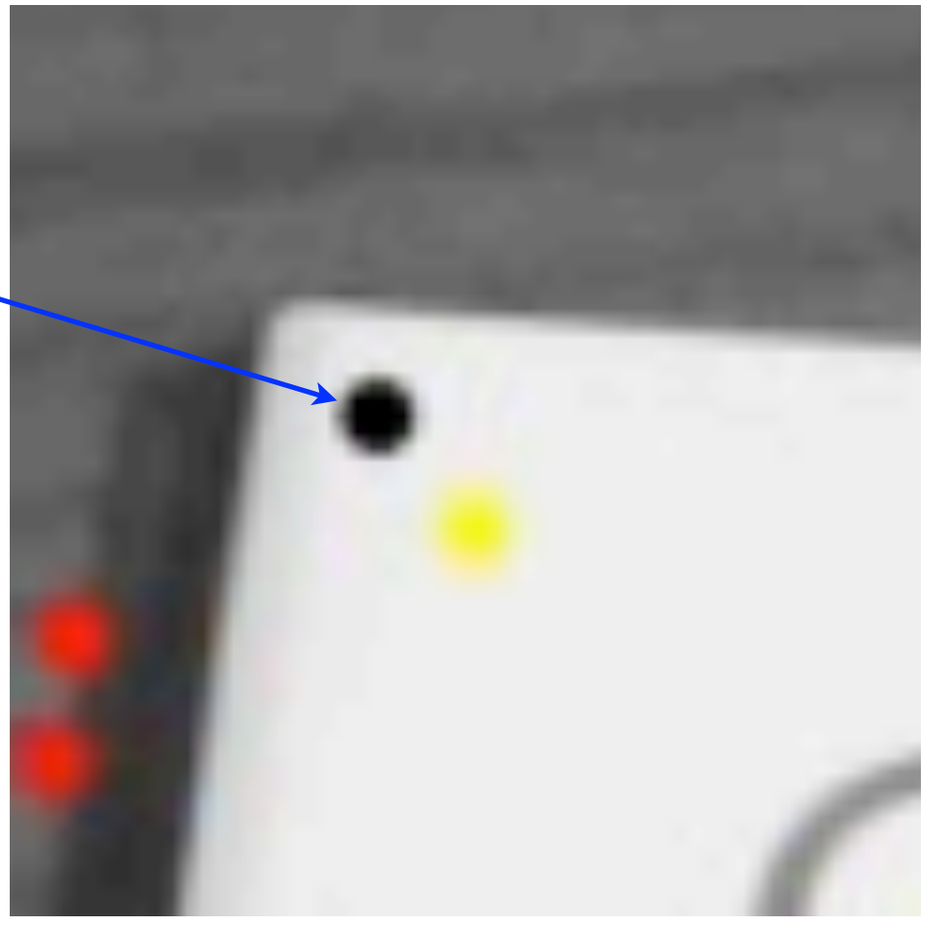
x=737
y=209



SIFT Interest Points

Select Gaussian at Scale of the Interest Point
Rescale by 1/sigma

x=737
y=209



$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

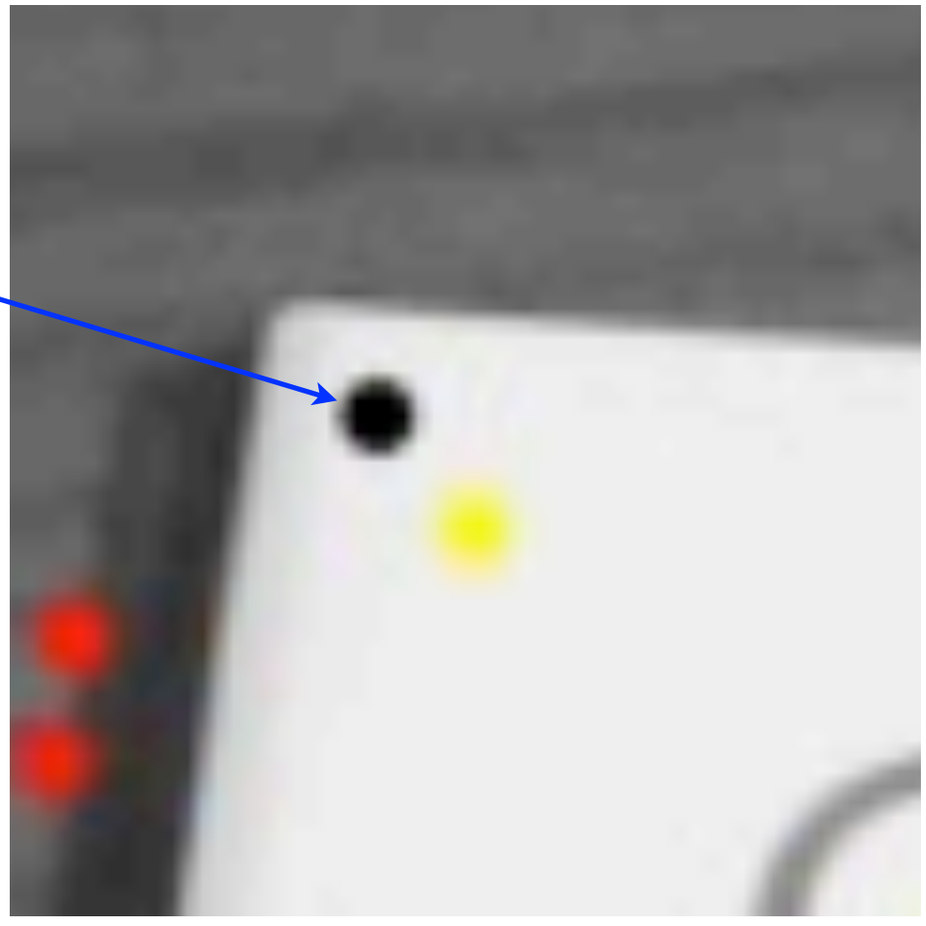
$\sigma = 16$

$\sigma = 32$

SIFT Interest Points

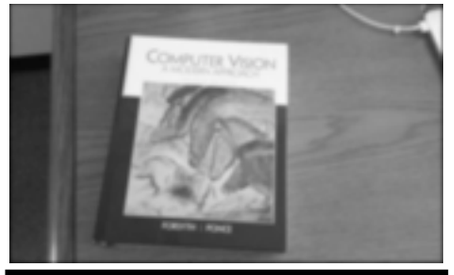
Select Gaussian at Scale of the Interest Point
Rescale Gaussian By 1/sigma

x=737
y=209

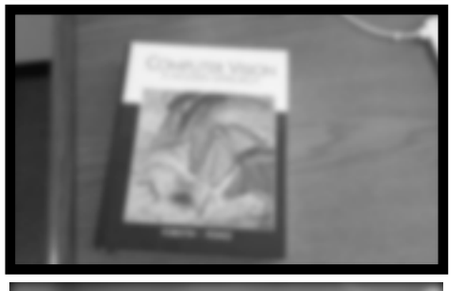


$\sigma = 2$

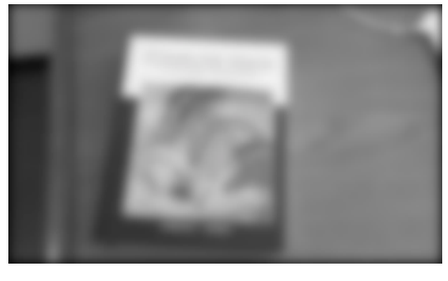
$\sigma = 4$



$\sigma = 8$



$\sigma = 16$

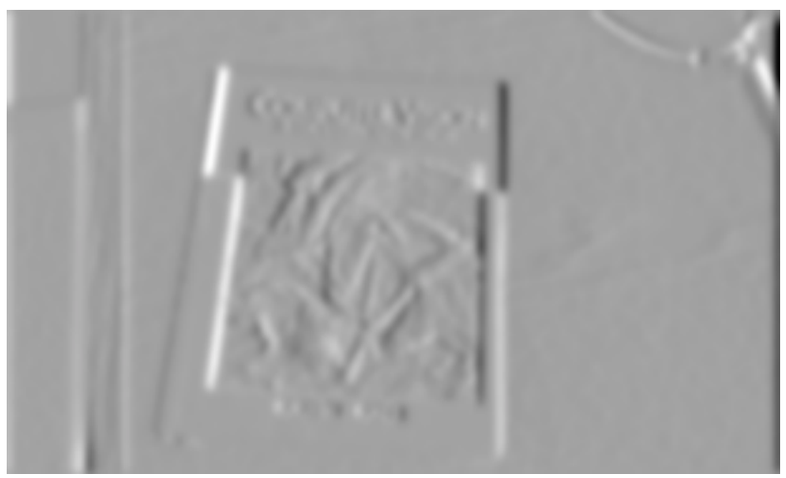


$\sigma = 32$

SIFT Interest Points

Compute Gradient Magnitude and Orientation

$\mathbf{G}$

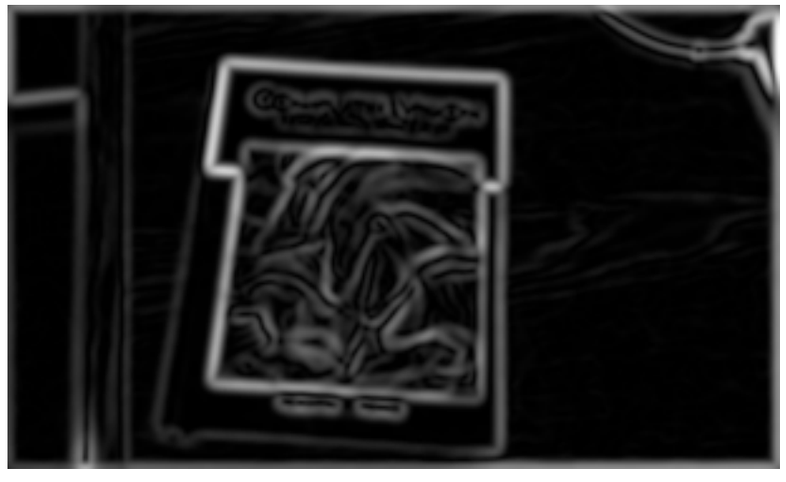


$\mathbf{G}_x$

$\mathbf{G}_y$

$$\text{Gradient Magnitude} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

$$\text{Gradient Orientation} = \tan^{-1} \frac{\mathbf{G}_y}{\mathbf{G}_x}$$



Gradient Magnitude

SIFT Interest Points

Create Orientation Histogram In Neighborhood of Point

Image

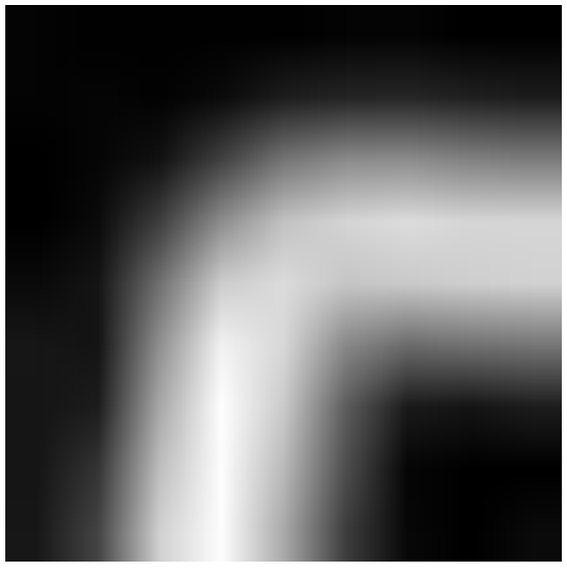
Neighborhood of Point

SIFT Interest Points

Create Orientation Histogram In Neighborhood of Point

Image

Neighborhood of Point



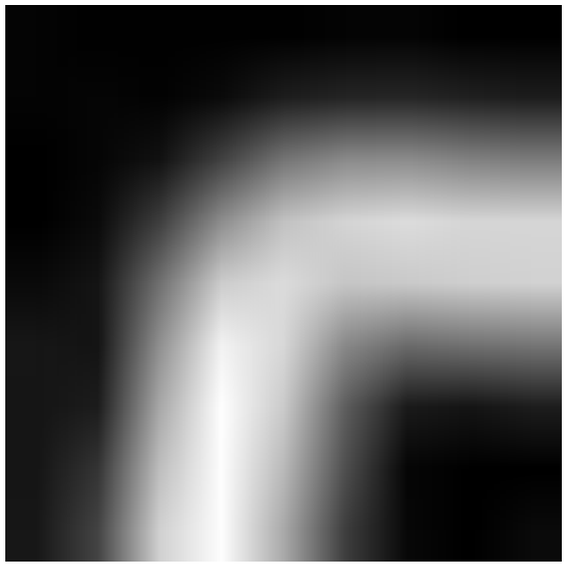
Gradient Magnitude
in Neighborhood

SIFT Interest Points

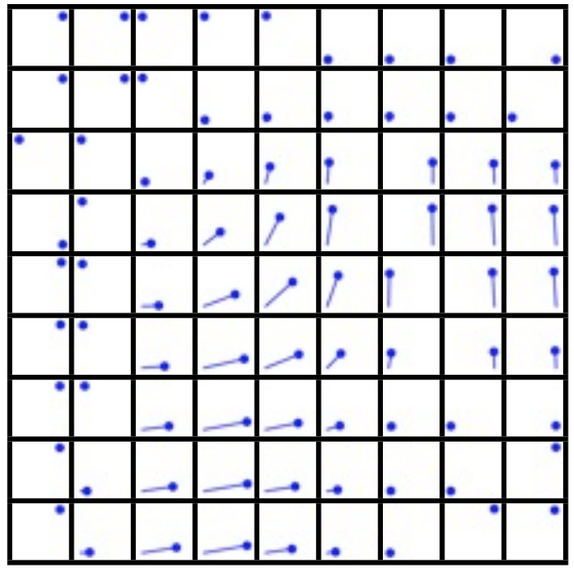
Create Orientation Histogram In Neighborhood of Point

Image

Neighborhood of Point



Gradient Magnitude in Neighborhood



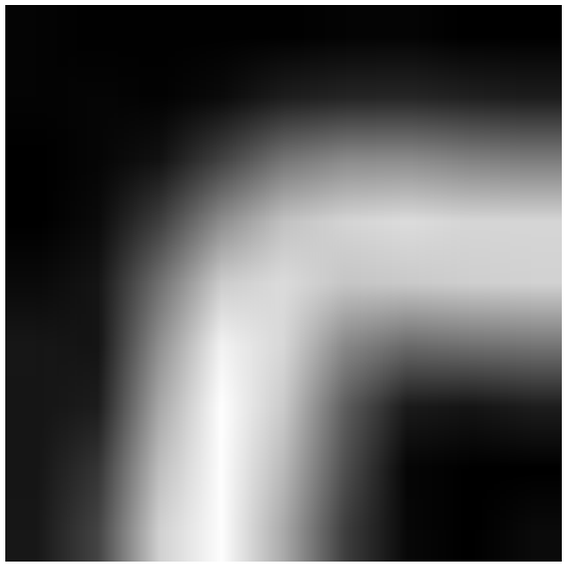
Gradient Orientations in Weighted by Gradient Magnitude

SIFT Interest Points

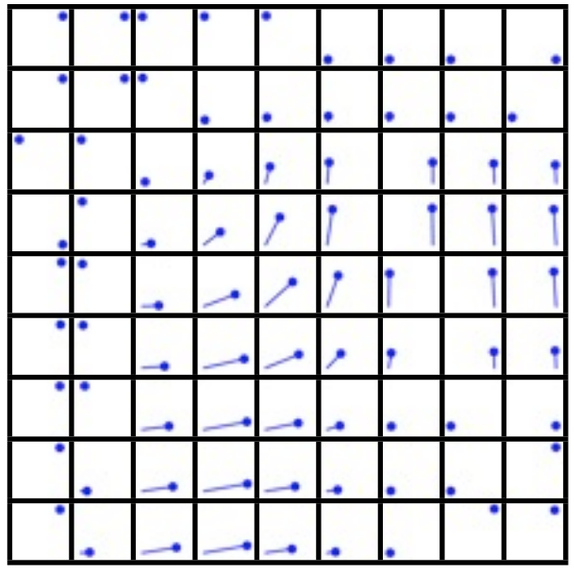
Create Orientation Histogram In Neighborhood of Point

Image

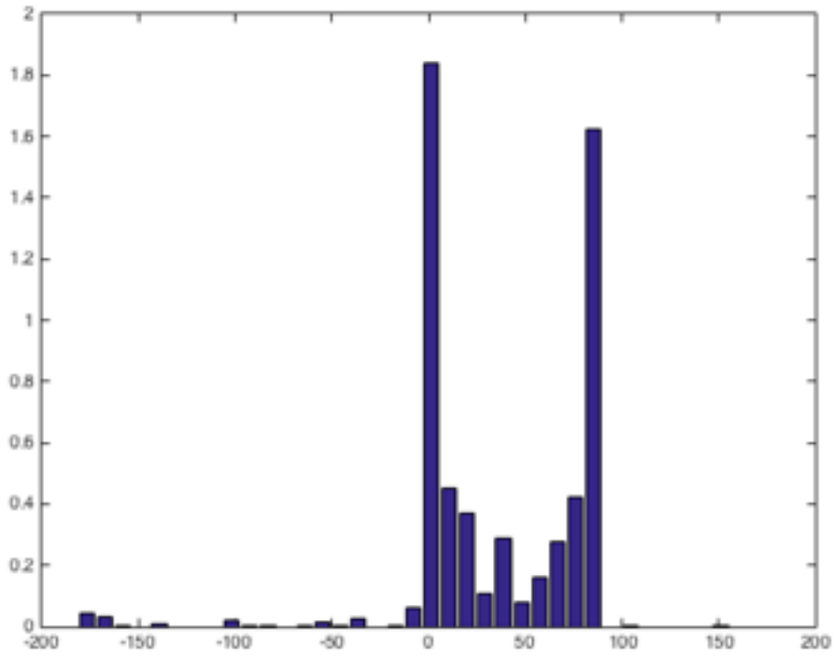
Neighborhood of Point



Gradient Magnitude in Neighborhood



Gradient Orientations in Weighted by Gradient Magnitude

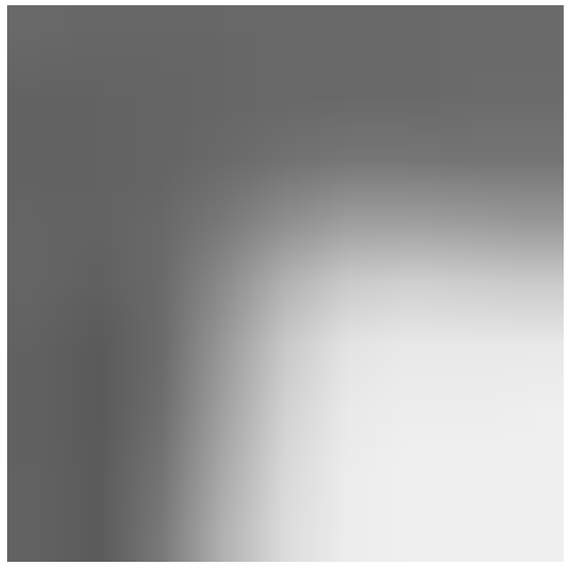


Histogram of Gradient Orientations (36 bins)

SIFT Interest Points

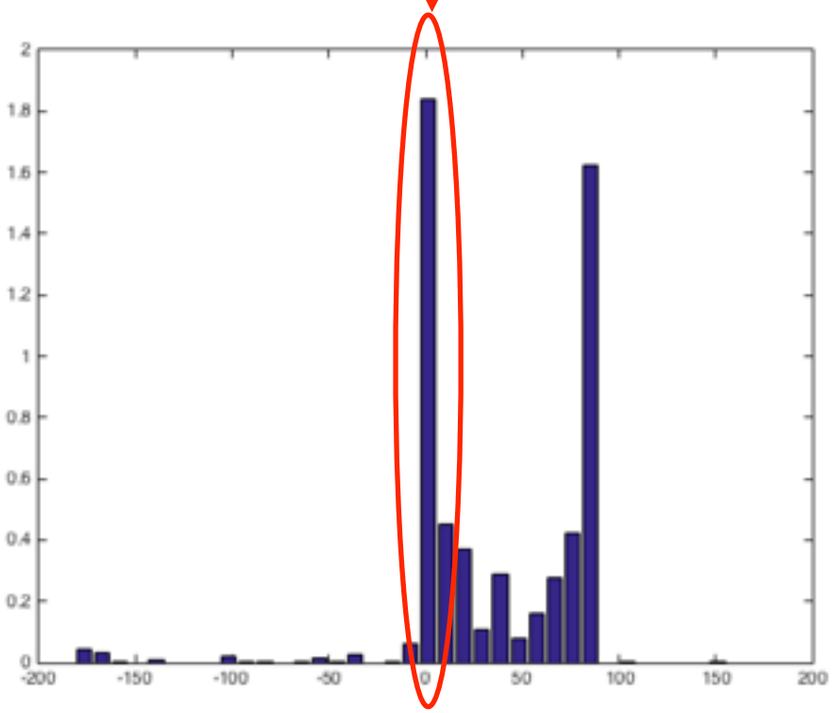
Create Orientation Histogram In Neighborhood of Point

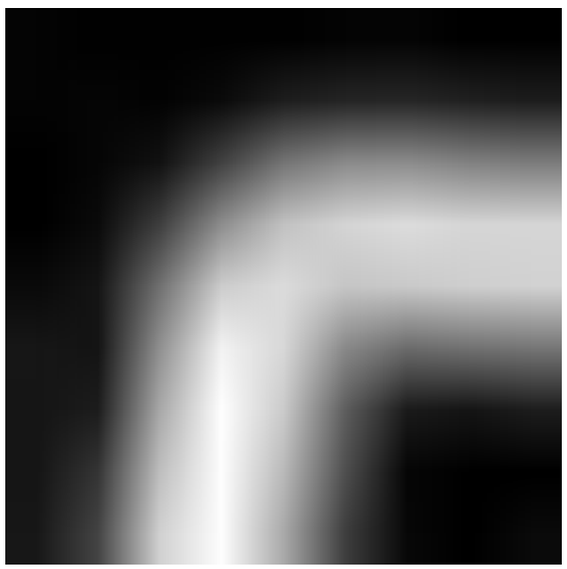
Image



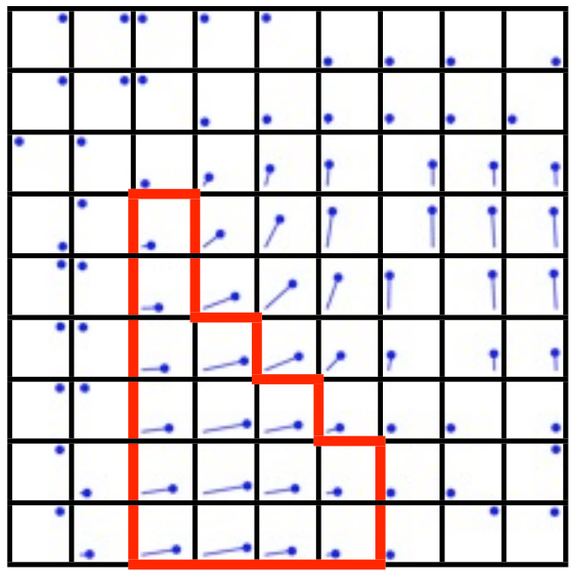
Neighborhood of Point

Sum of weights
(gradient magnitudes)
of elements with
orientations near 1.3 degrees.





Gradient Magnitude
in Neighborhood



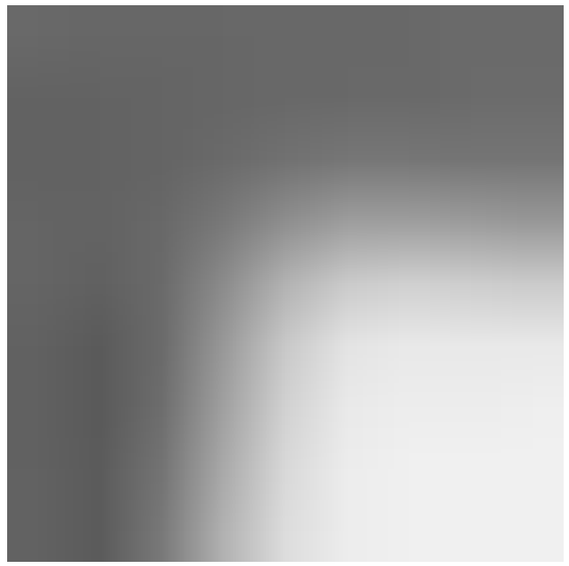
Gradient Orientations in
Weighted by
Gradient Magnitude

Histogram of
Gradient Orientations
(36 bins)

SIFT Interest Points

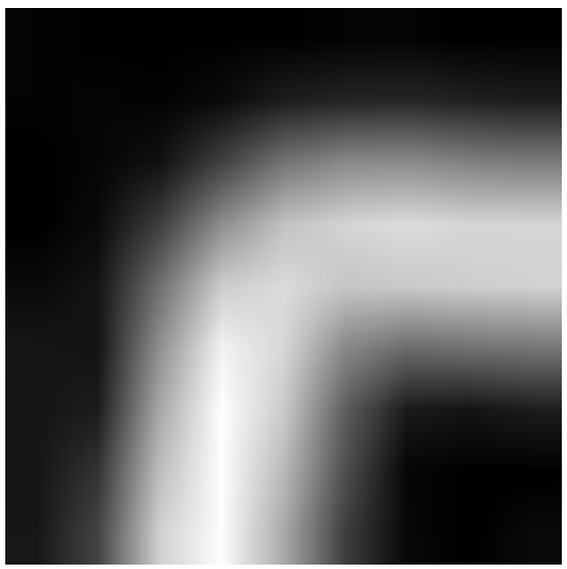
Create Orientation Histogram In Neighborhood of Point

Image

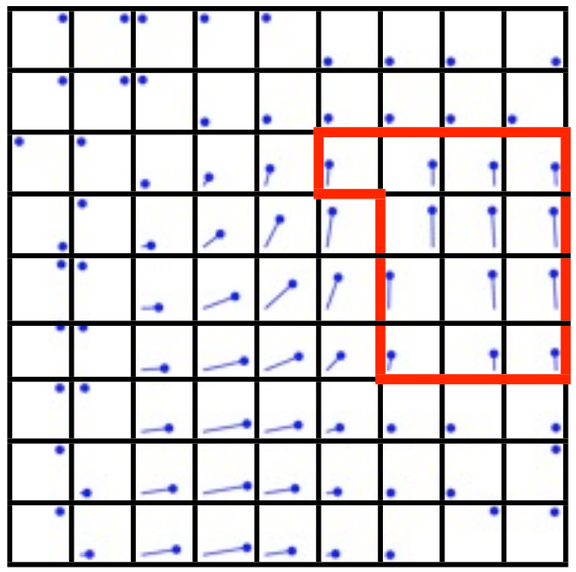


Neighborhood of Point

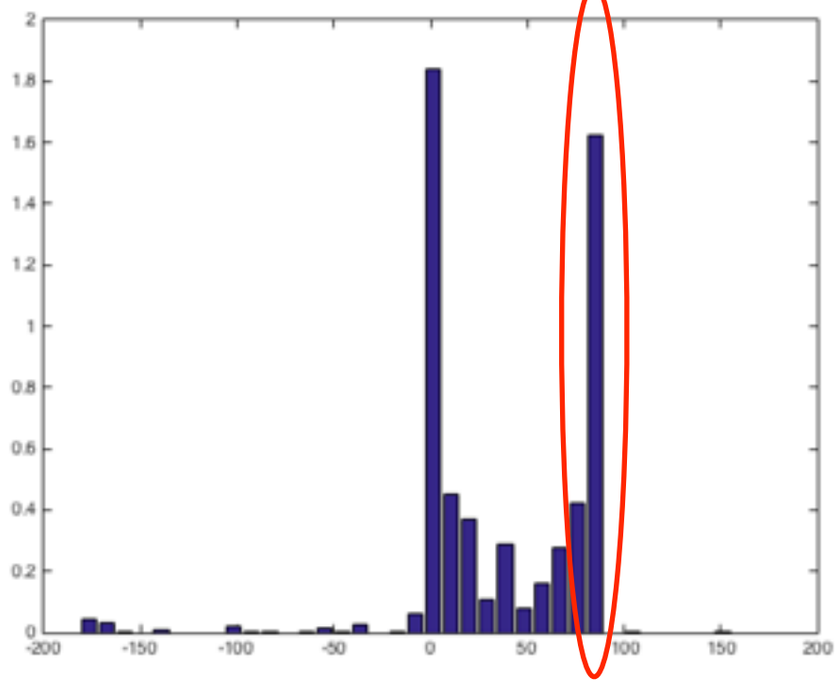
Sum of weights
(gradient magnitudes)
of elements with
orientations near 85.6 degrees.



Gradient Magnitude
in Neighborhood



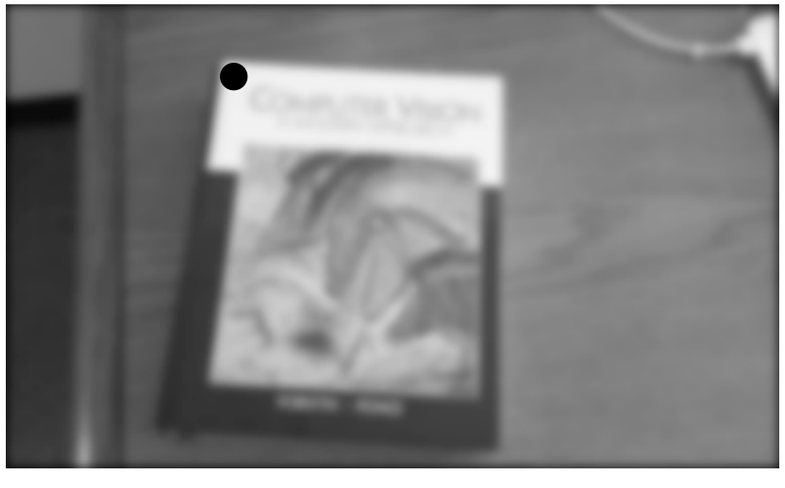
Gradient Orientations in
Weighted by
Gradient Magnitude



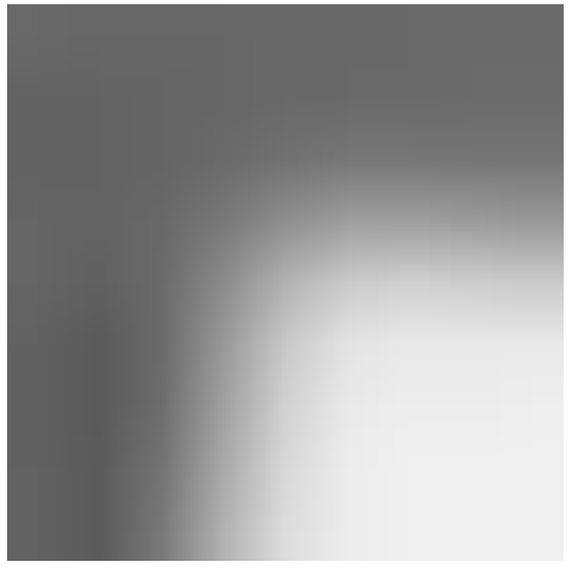
Histogram of
Gradient Orientations
(36 bins)

SIFT Interest Points

Create Orientation Histogram In Neighborhood of Point



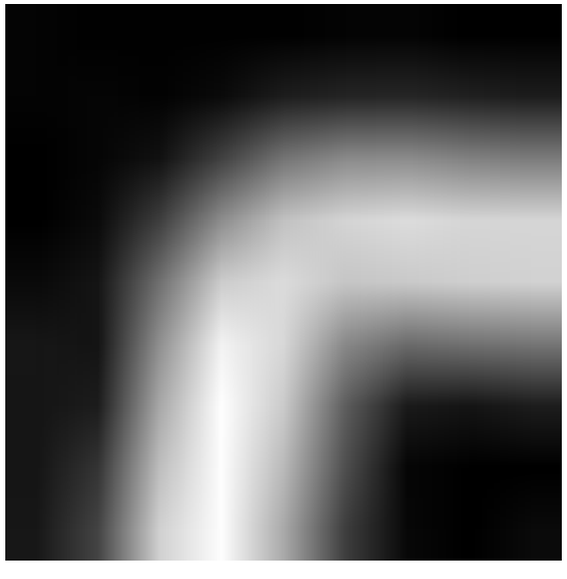
Image



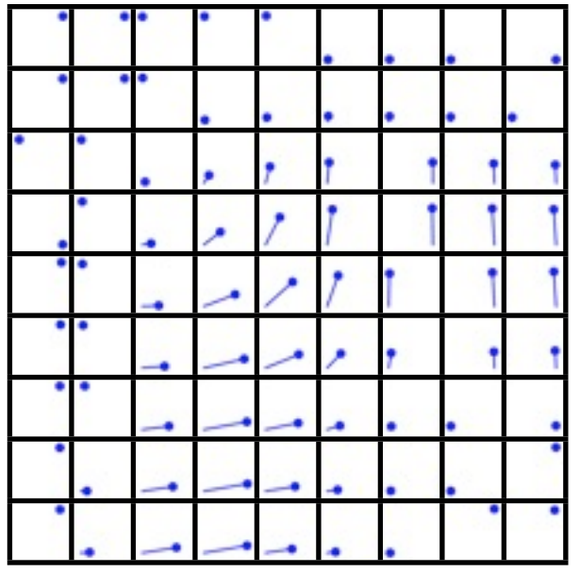
Neighborhood of Point

Retain maximum bin
and bins with counts
within 0.8 of maximum bin

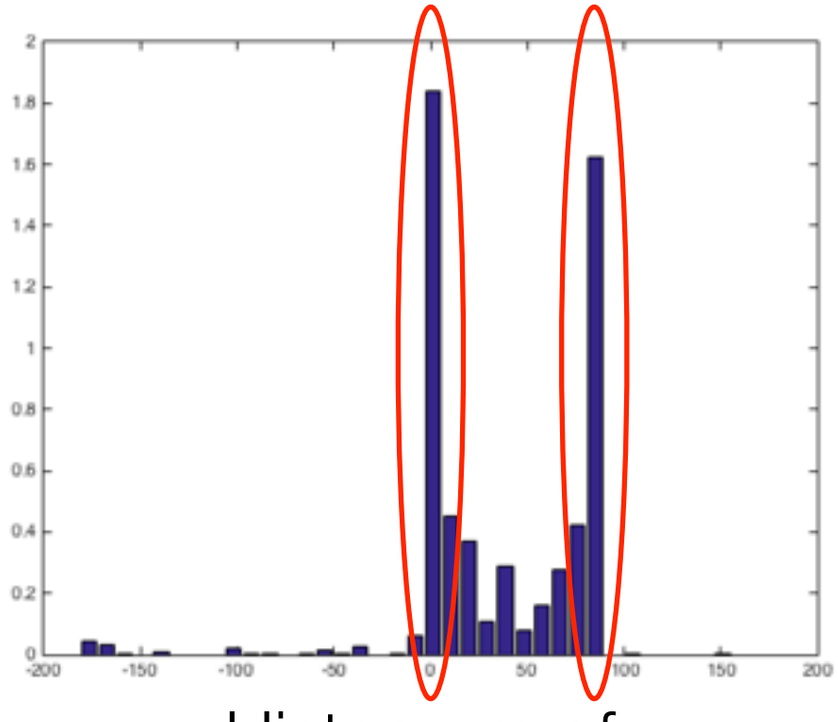
1.3 85.6



Gradient Magnitude
in Neighborhood



Gradient Orientations in
Weighted by
Gradient Magnitude



Histogram of
Gradient Orientations
(36 bins)

SIFT Interest Points

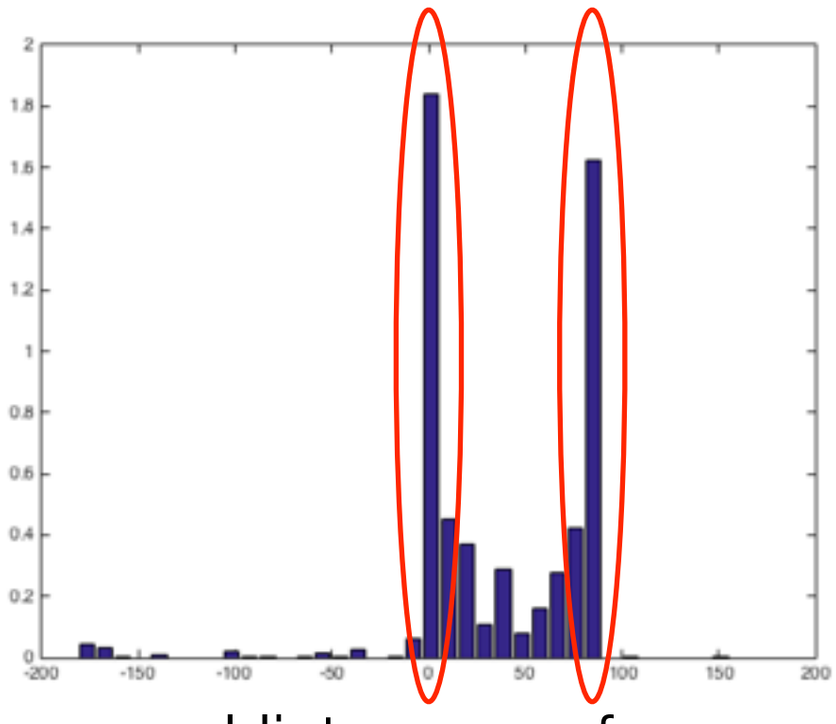
Save copies of the points and scale for every retained orientation

$x = 737, y = 209, \sigma = 16, \theta = 1.3$

$x = 737, y = 209, \sigma = 16, \theta = 85.6$

Retain maximum bin
and bins with counts
within 0.8 of maximum bin

1.3 85.6



Histogram of
Gradient Orientations
(36 bins)

SIFT Interest Points

Save copies of the points and scale for every retained orientation

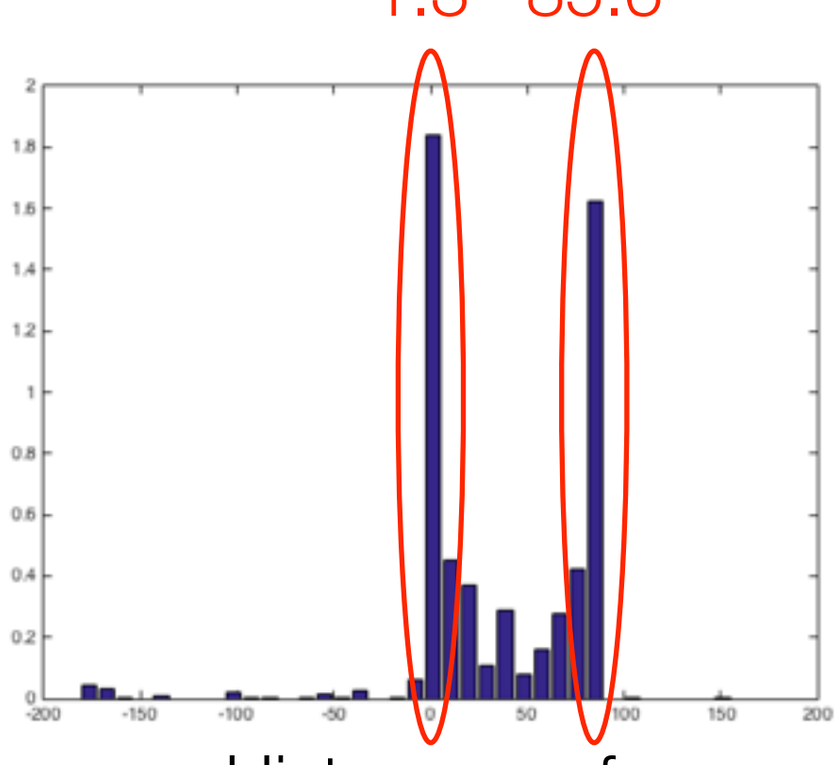
$x = 737, y = 209, \sigma = 16, \theta = 1.3$

$x = 737, y = 209, \sigma = 16, \theta = 85.6$

Retain maximum bin
and bins with counts
within 0.8 of maximum bin

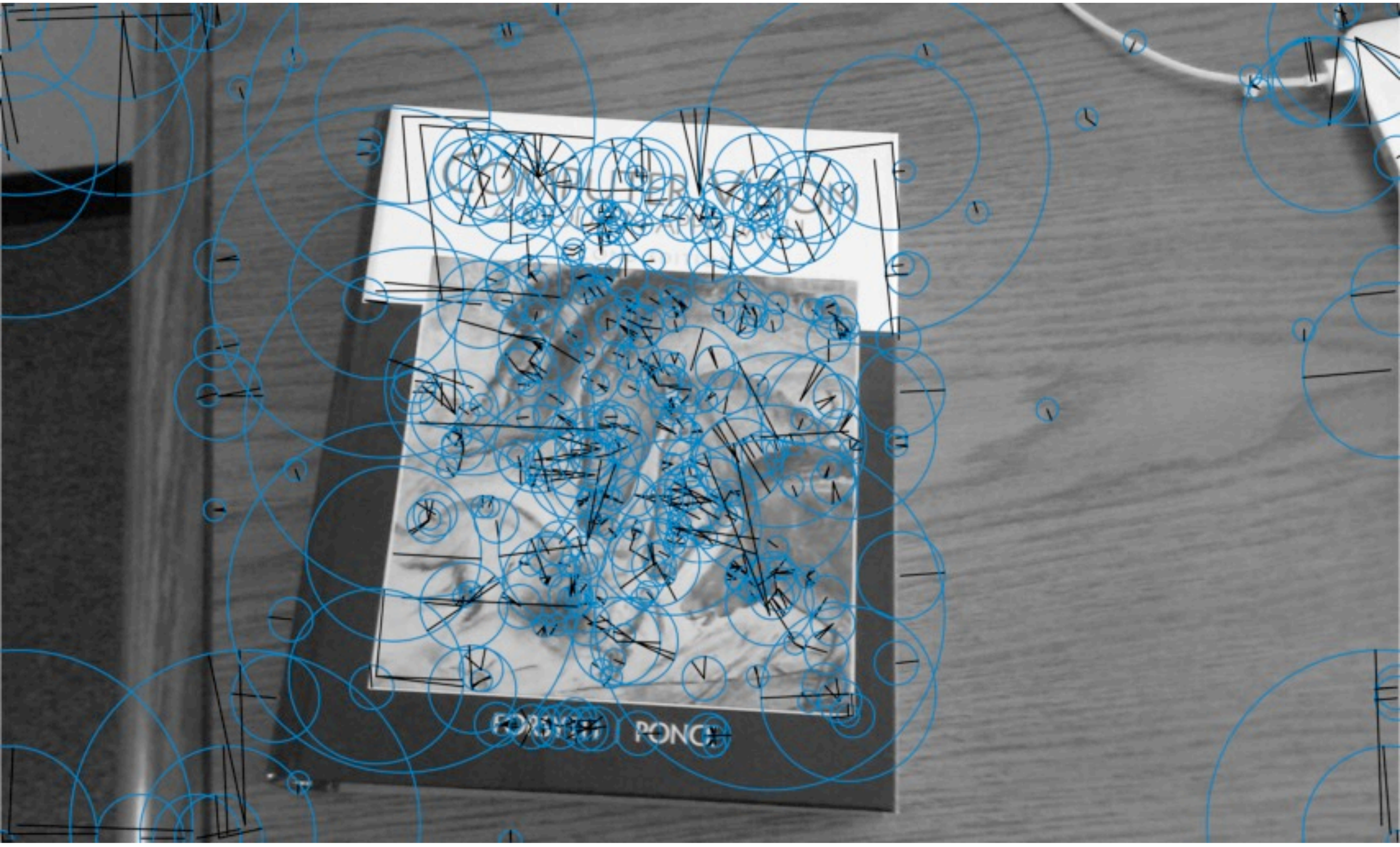
Saving multiple copies helps
ensure rotation invariance
by providing many candidates
for potential rotations.

(Hedge your bets in many different orientations.)



Histogram of
Gradient Orientations
(36 bins)

SIFT Interest Points




UNGA

SIFT Descriptor

Make for every (point,scale,orientation) tuple

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

$$x = 737, y = 209, \sigma = 16, \theta = 85.6$$

SIFT Descriptor

Make for every (point,scale,orientation) tuple

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

$$x = 737, y = 209, \sigma = 16, \theta = 85.6$$

A descriptor ‘sums up’ the information about a feature.

You use descriptors of features to match features to each other.

SIFT Descriptor

Resize Gaussian at Scale of Point by 1/sigma.
Rotate Resized Gaussian so that Theta is 0.

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

Gaussian

Resized Gaussian
Rotated by -1.3 degrees

SIFT Descriptor

Resize Gaussian at Scale of Point by 1/sigma.
Rotate Resized Gaussian so that Theta is 0.

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

Gaussian

Resized Gaussian
Rotated by -1.3 degrees

Why resize and rotate? So that we get a descriptor that is invariant to scale and rotation.

SIFT Descriptor

In the rotated Gaussian, Get Gradient Magnitudes and Orientations in 16x16 Region around Point.

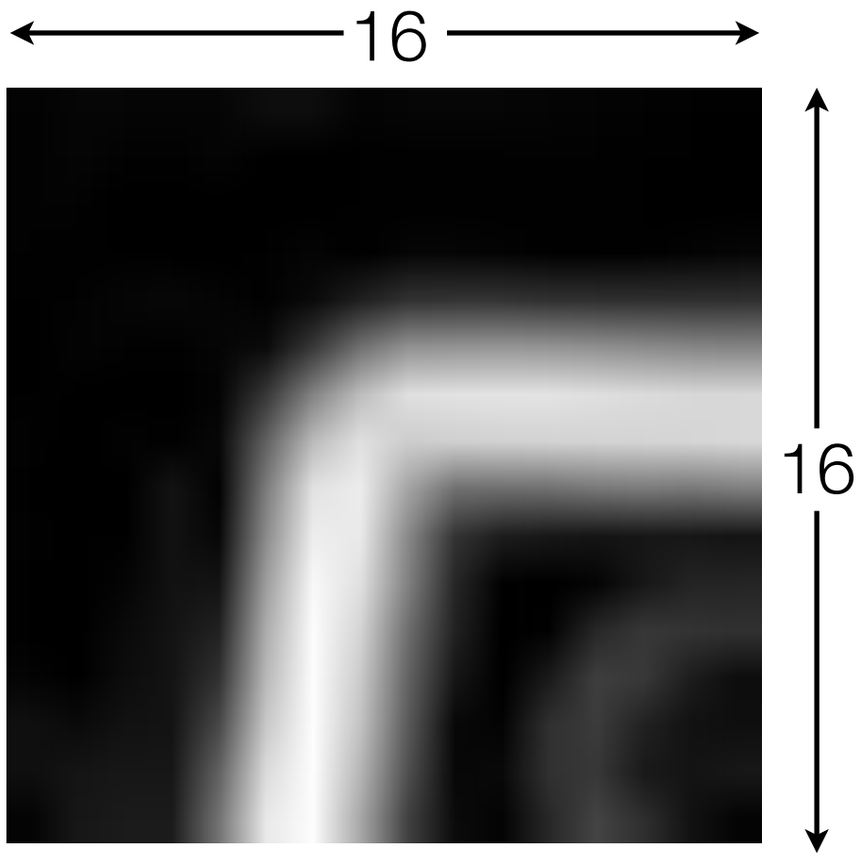
$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

Resized Gaussian
Rotated by -1.3 degrees

SIFT Descriptor

In the rotated Gaussian, Get Gradient Magnitudes and Orientations in 16x16 Region around Point.

$x = 737, y = 209, \sigma = 16, \theta = 1.3$



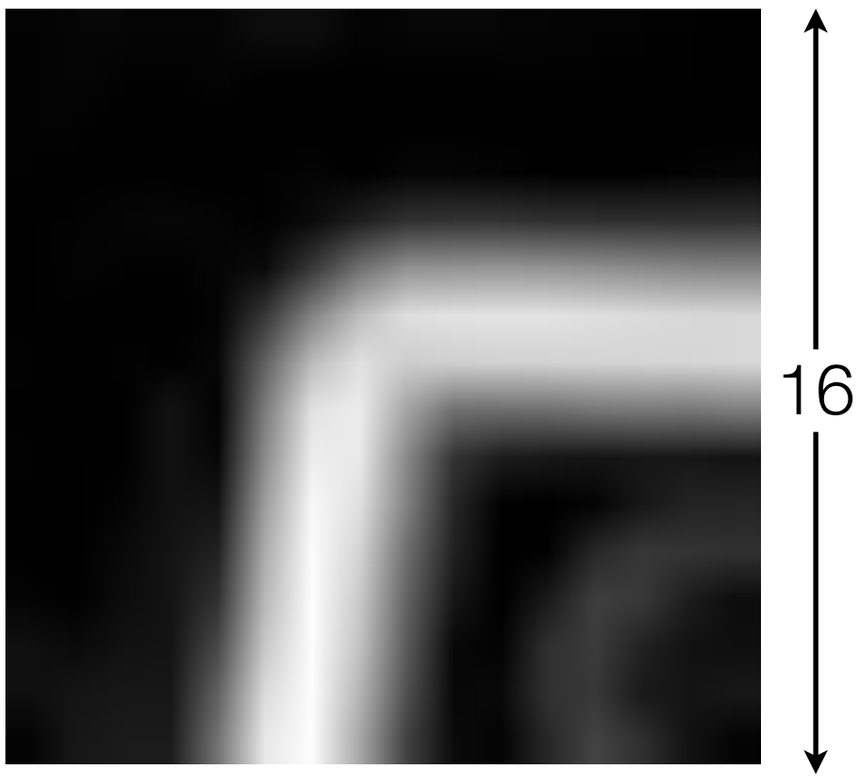
Gradient Magnitude

Resized Gaussian
Rotated by -1.3 degrees

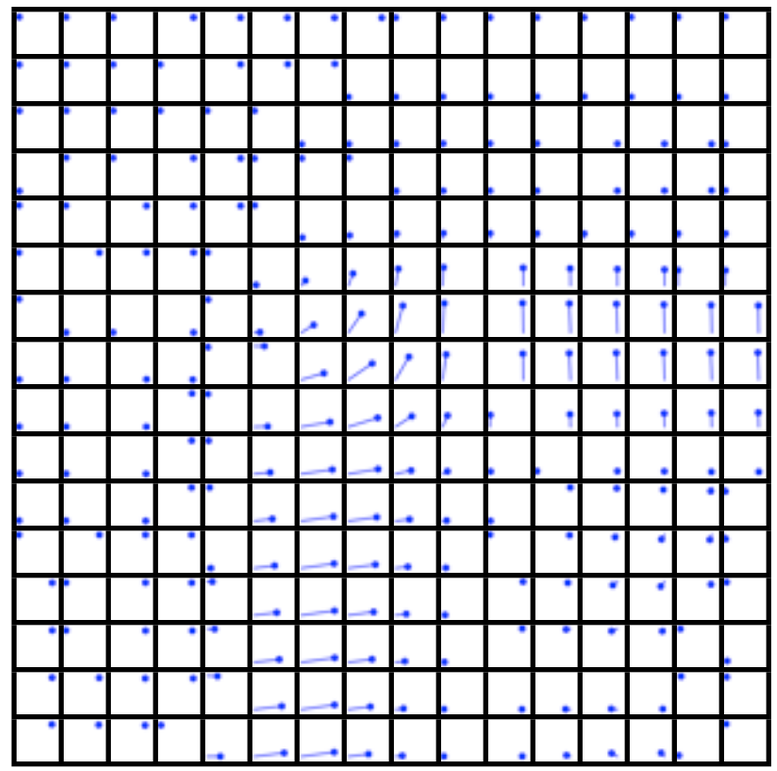
SIFT Descriptor

In the rotated Gaussian, Get Gradient Magnitudes and Orientations in 16x16 Region around Point.

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$



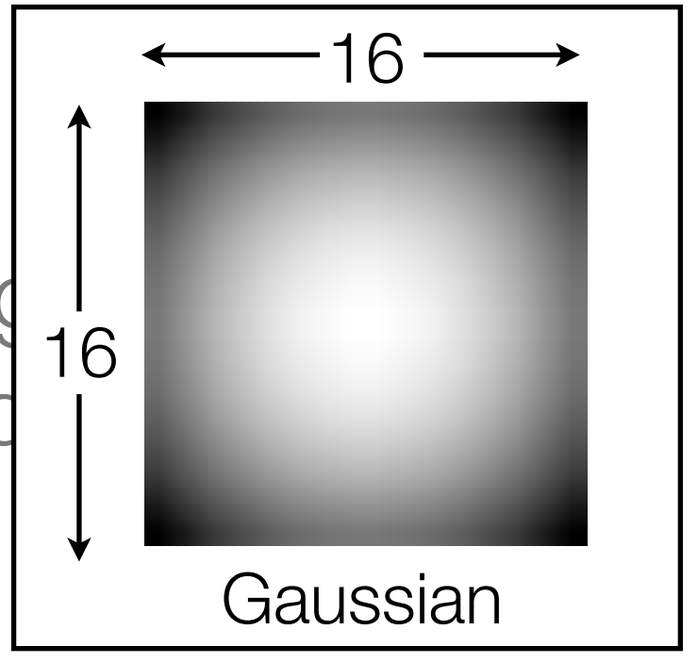
Gradient Magnitude



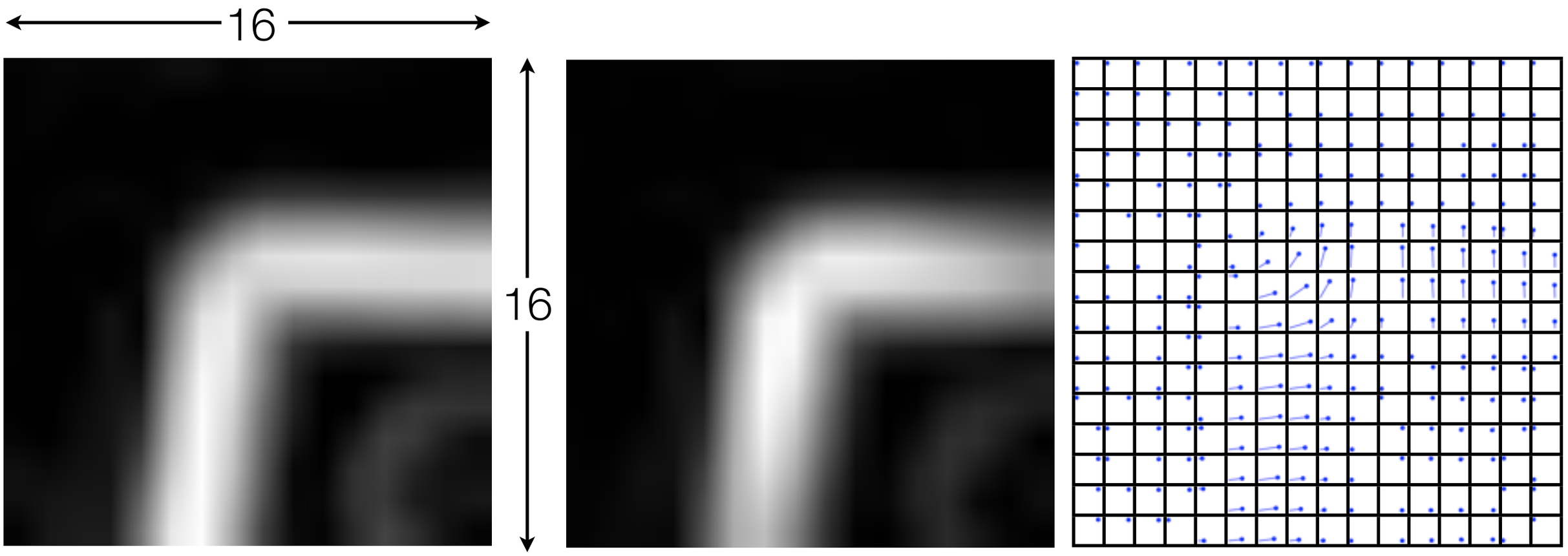
Gradient Orientations Weighted by Gradient Magnitudes

SIFT Descriptor

In the rotated Gaussian, Get Gradient Mag
Orientations in 16x16 Region around



$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$



Gradient Magnitude

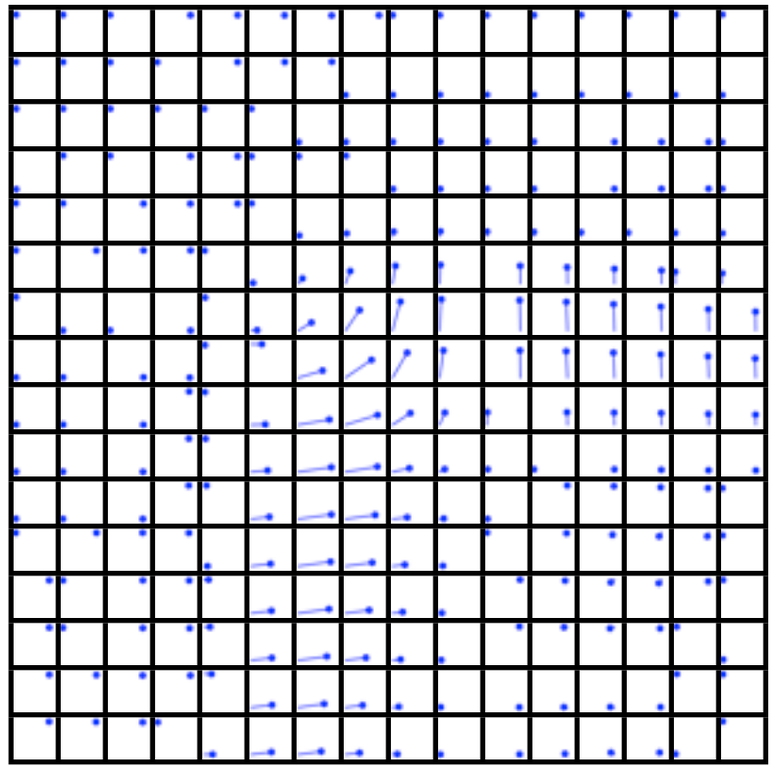
Gradient Magnitude Weighted by Gaussian

Gradient Orientations Weighted by Gradient Magnitudes and Gaussian

SIFT Descriptor

In the rotated Gaussian, Get Gradient Magnitudes and Orientations in 16x16 Region around Point.

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

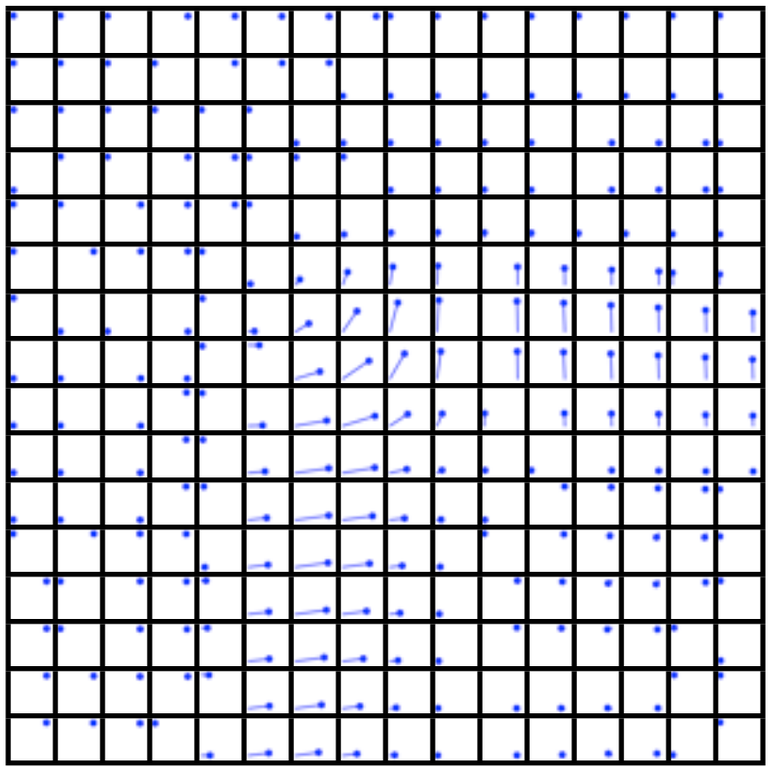


Gradient Orientations Weighted by Gradient Magnitudes and Gaussian

SIFT Descriptor

Divide Orientations Into 4 x 4 Blocks

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

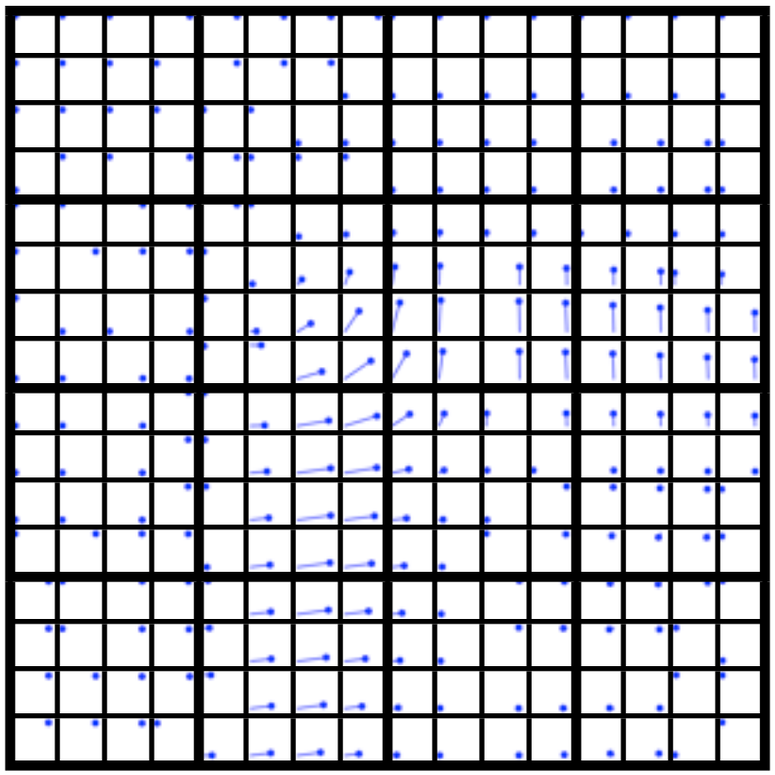


Gradient Orientations Weighted by Gradient Magnitudes and Gaussian

SIFT Descriptor

Divide Orientations Into 4 x 4 Blocks

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$



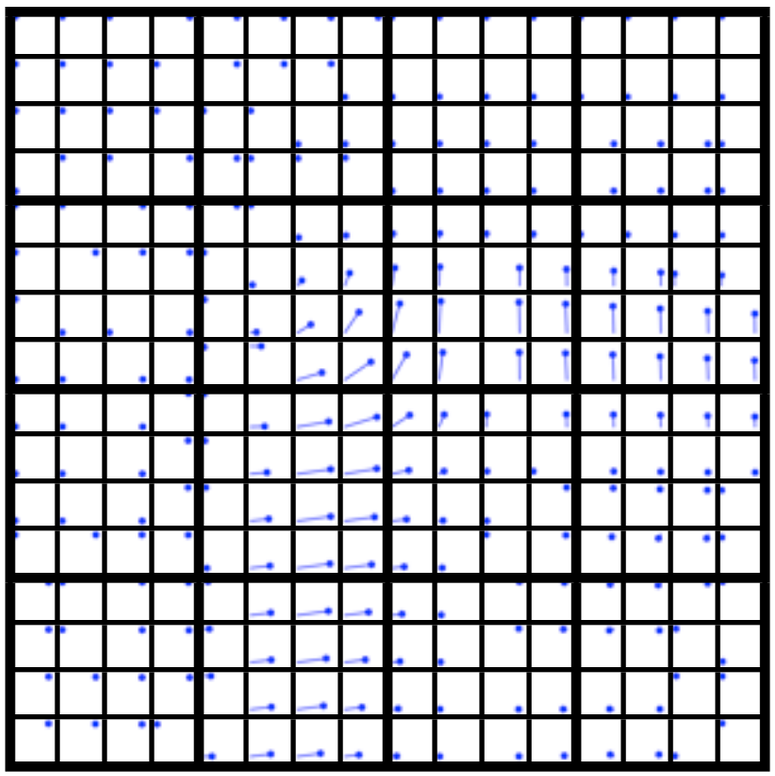
Gradient Orientations Weighted by Gradient Magnitudes and Gaussian

SIFT Descriptor

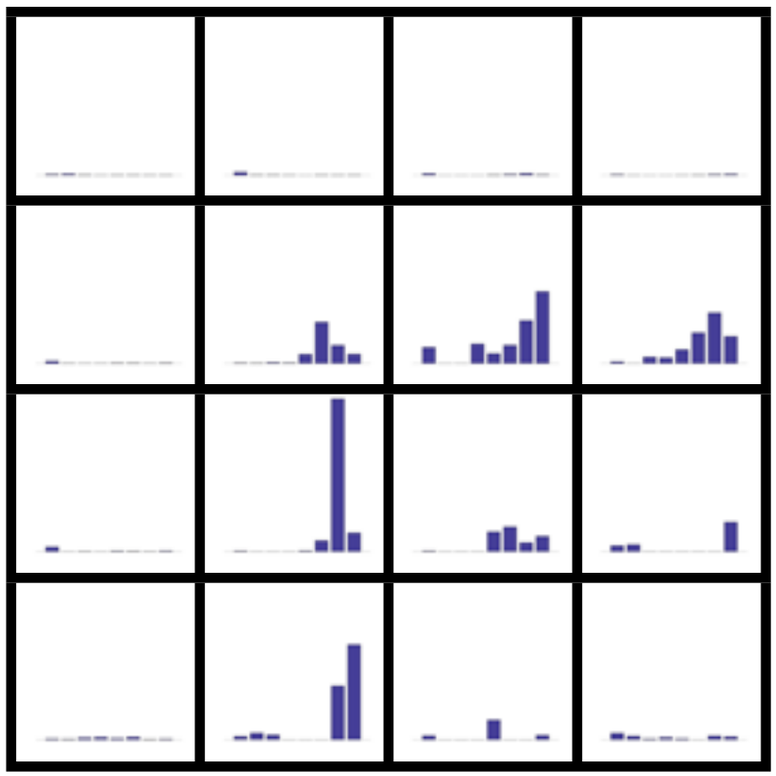
Build Histogram of Gradient Orientations For Each Block

$$x = 737, y = 209, \sigma = 16, \theta = 1.3$$

4 blocks horizontally x
4 blocks vertically x
8 bins per block =
128 values



Gradient Orientations Weighted by Gradient Magnitudes and Gaussian



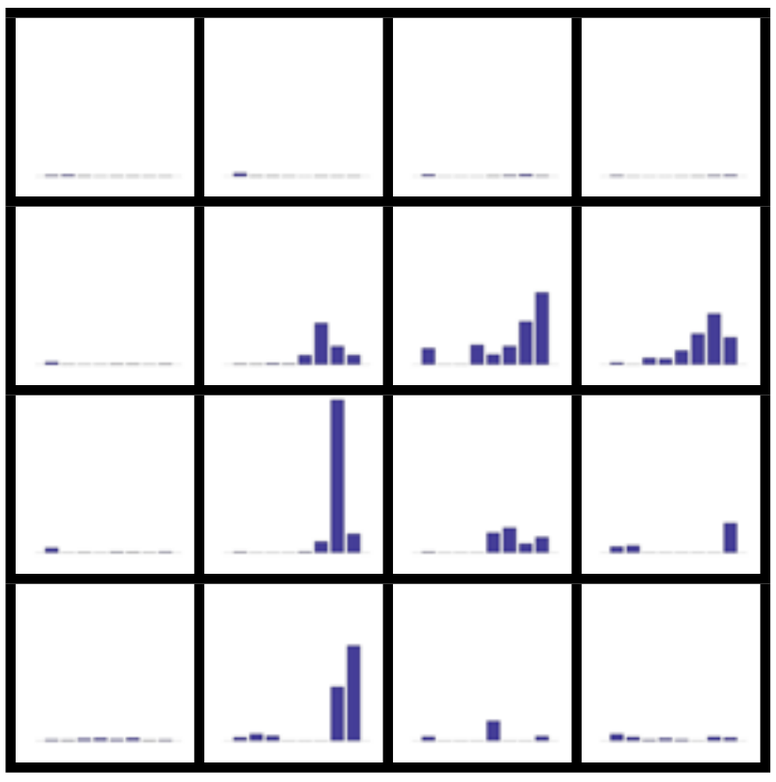
Histogram of Gradient Orientations Per Block

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$

128 values

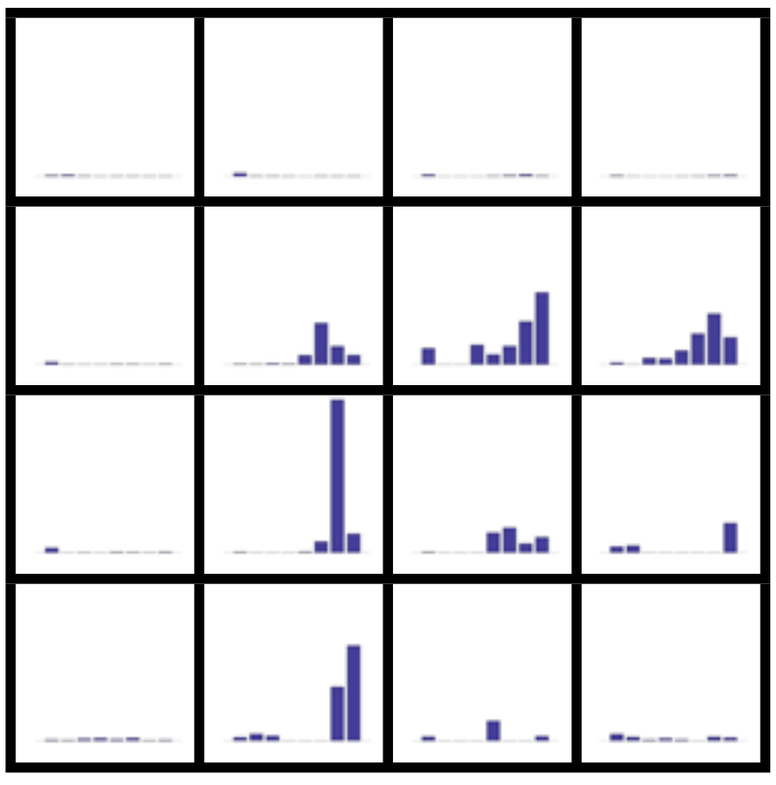


Histogram of Gradient Orientations Per Block

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$



Histogram of Gradient Orientations Per Block

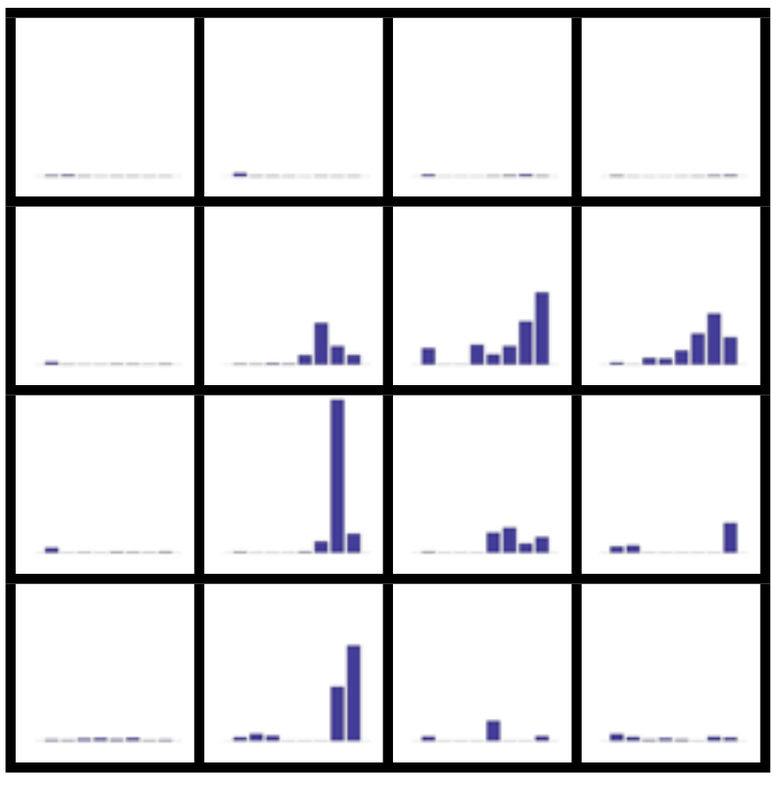
128 values

Vector **d** with 128 Values.

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$



Histogram of Gradient Orientations Per Block

128 values

Vector **d** with 128 Values.

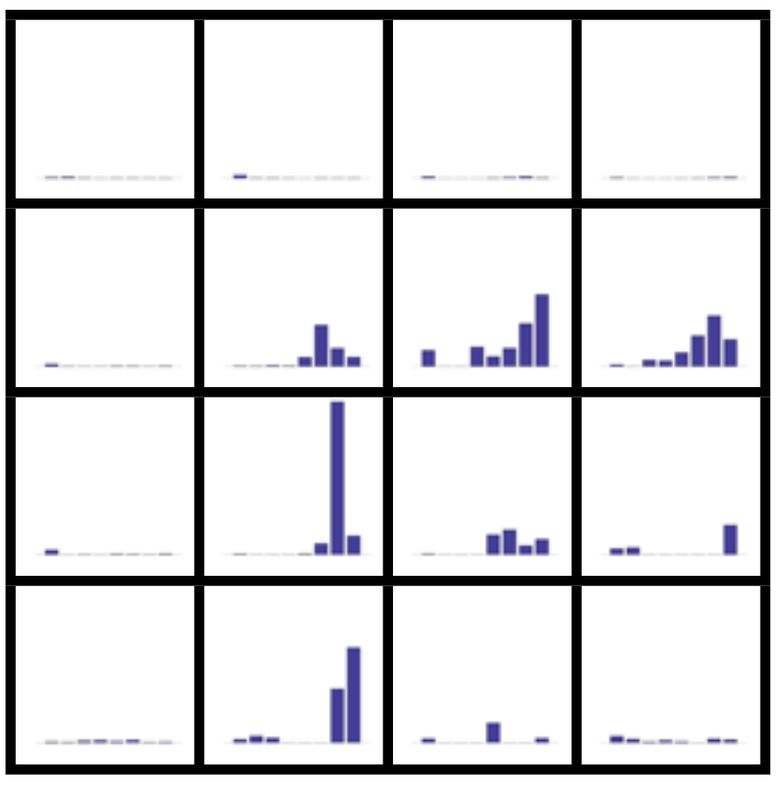
Normalize vector to have unit length: provides illumination invariance.

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$

128 values



Histogram of Gradient Orientations Per Block

Vector **d** with 128 Values.

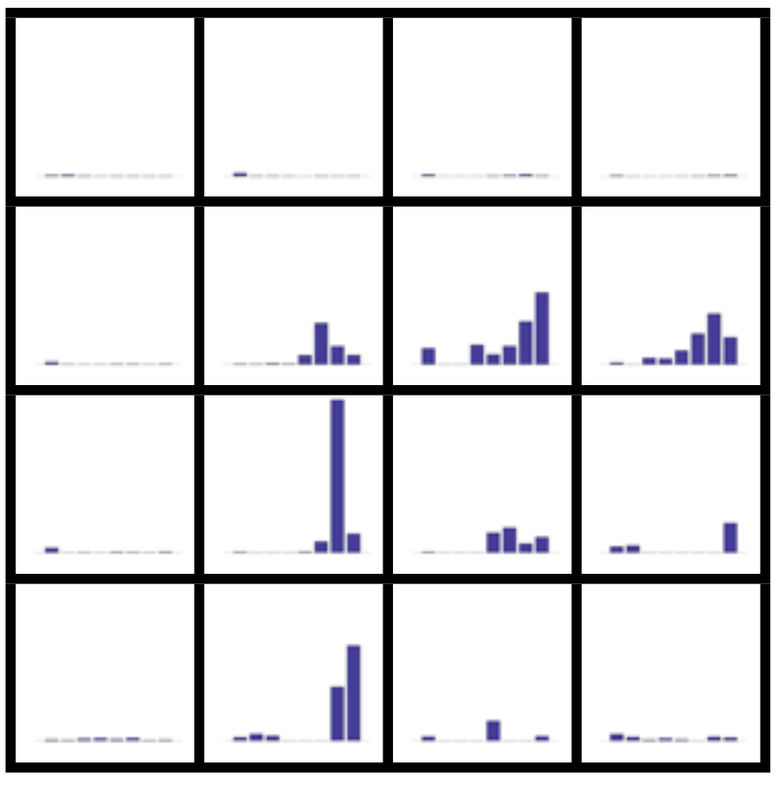
Normalize vector to have unit length:
provides illumination invariance.

Force all values to be <= 0.2:
removes effects of camera saturation.

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$



Histogram of Gradient Orientations Per Block

128 values

Vector **d** with 128 Values.

Normalize vector to have unit length:
provides illumination invariance.

Force all values to be <= 0.2:
removes effects of camera saturation.

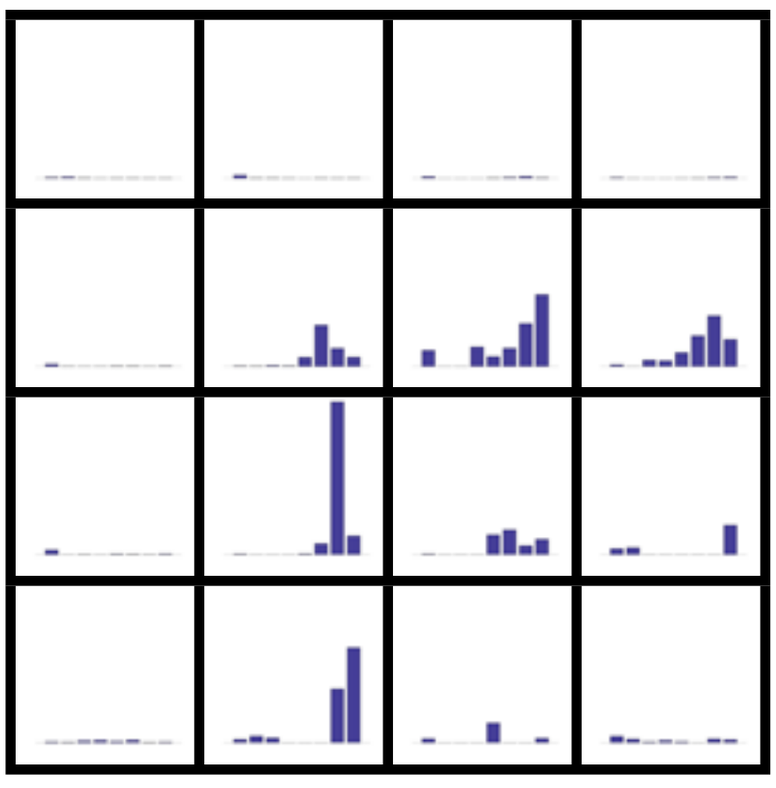
Renormalize vector to have unit length:
maintains illumination invariance.

SIFT Descriptor

Build Histogram of Gradient Orientations For Each Block

$x = 737, y = 209, \sigma = 16, \theta = 1.3$

128 values



Histogram of Gradient Orientations Per Block

Vector **d** with 128 Values.

Normalize vector to have unit length:
provides illumination invariance.

Force all values to be <= 0.2:
removes effects of camera saturation.

Renormalize vector to have unit length:
maintains illumination invariance.

SIFT Descriptor

Matching Descriptors Between Two Images

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Matching Descriptors Between Two Images

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

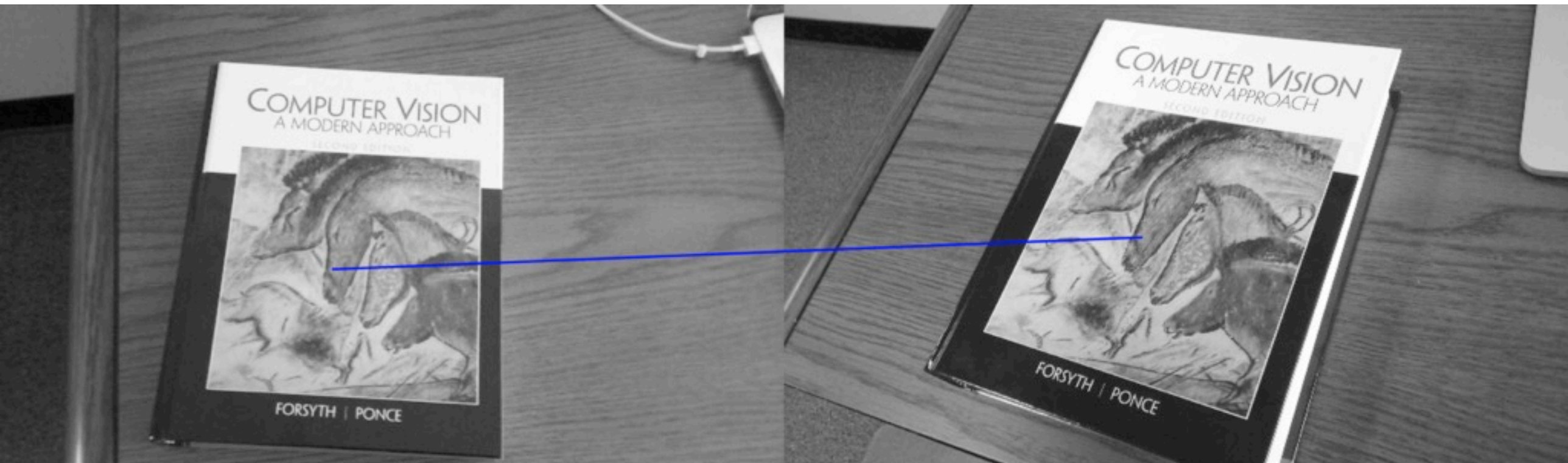
Get its descriptor $\mathbf{d_{i1}}$,

Matching Descriptors Between Two Images

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Get its descriptor $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{j2}}$ in second image with closest descriptor $\mathbf{d_{j2}}$ to $\mathbf{d_{i1}}$,



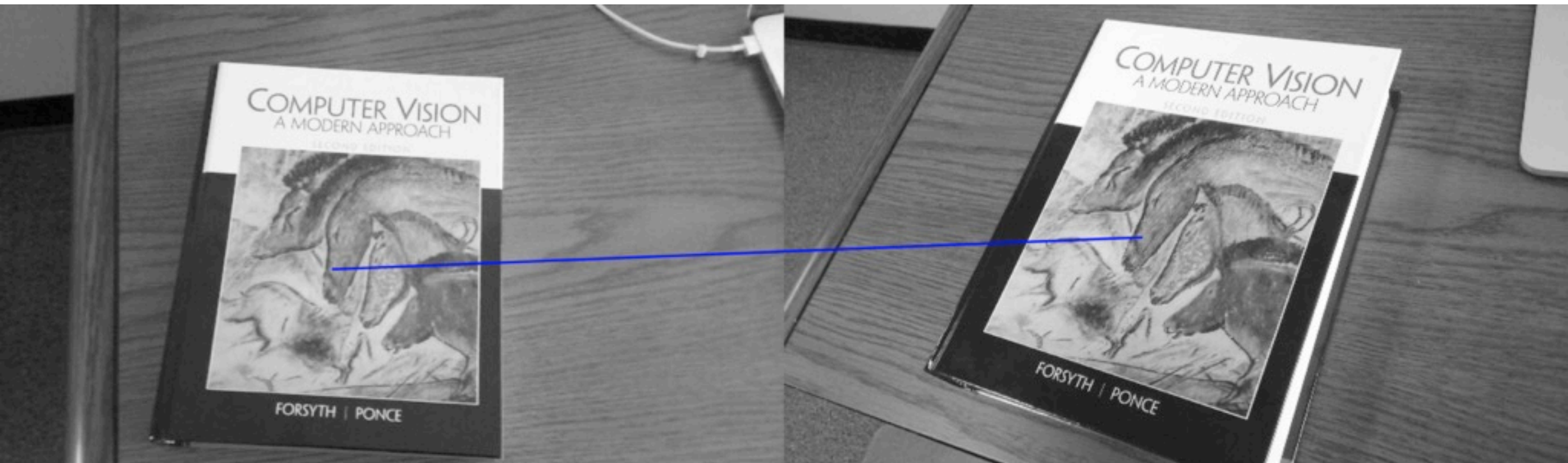
Matching Descriptors Between Two Images

Nearest Neighbor.

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Get its descriptor $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{j2}}$ in second image with closest descriptor $\mathbf{d_{j2}}$ to $\mathbf{d_{i1}}$,



Matching Descriptors Between Two Images

Nearest Neighbor.

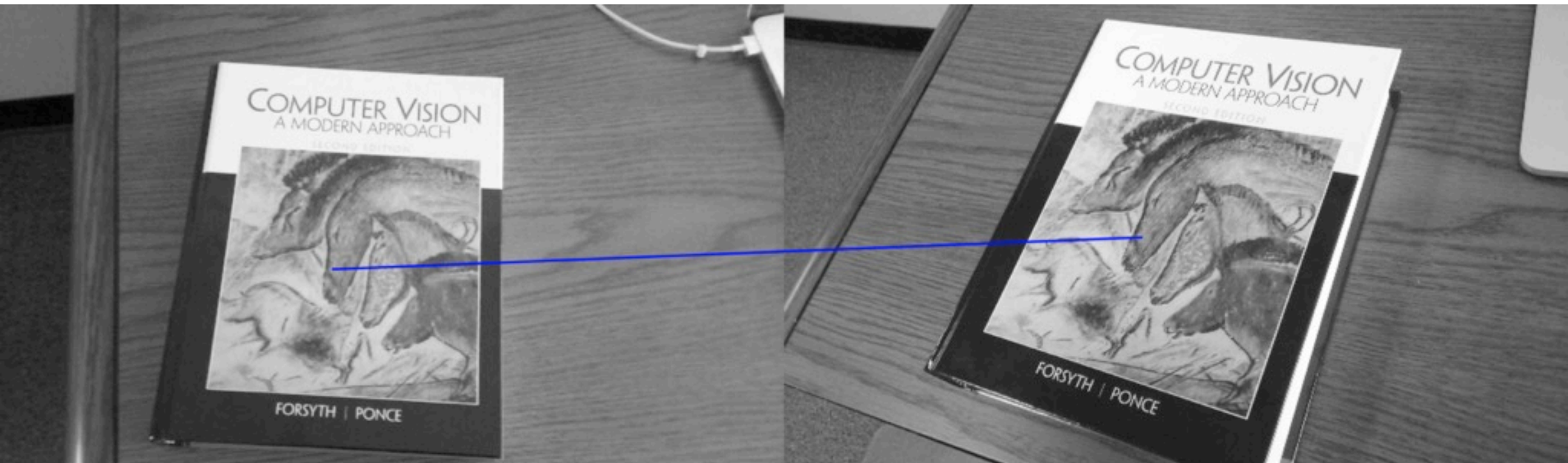
Use SSD (sum-squared distance).

Also called Euclidean distance.

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Get its descriptor $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{j2}}$ in second image with closest descriptor $\mathbf{d_{j2}}$ to $\mathbf{d_{i1}}$,



Matching Descriptors Between Two Images

Nearest Neighbor.

Use SSD (sum-squared distance).

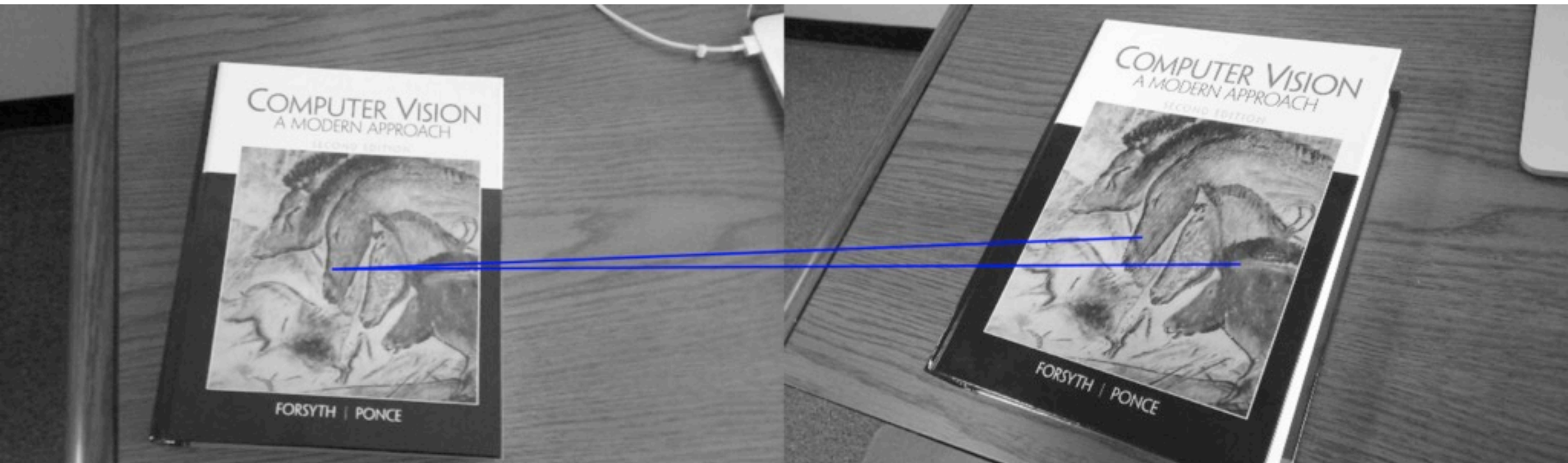
Also called Euclidean distance.

For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Get its descriptor $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{j2}}$ in second image with closest descriptor $\mathbf{d_{j2}}$ to $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{k2}}$ in second image with second closest descriptor $\mathbf{d_{k2}}$ to $\mathbf{d_{i1}}$,



Matching Descriptors Between Two Images

Nearest Neighbor.

Use SSD (sum-squared distance).

Also called Euclidean distance.

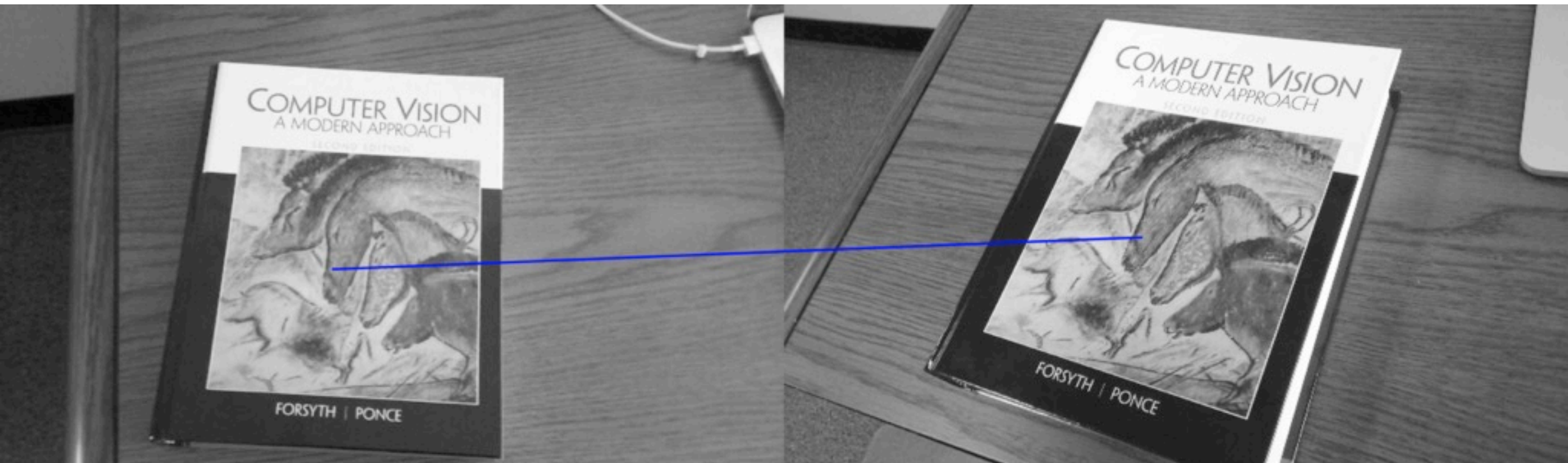
For an interest point $\mathbf{p_{i1}}$ in first image $\mathbf{I_1}$,

Get its descriptor $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{j2}}$ in second image with closest descriptor $\mathbf{d_{j2}}$ to $\mathbf{d_{i1}}$,

Find interest point $\mathbf{p_{k2}}$ in second image with second closest descriptor $\mathbf{d_{k2}}$ to $\mathbf{d_{i1}}$,

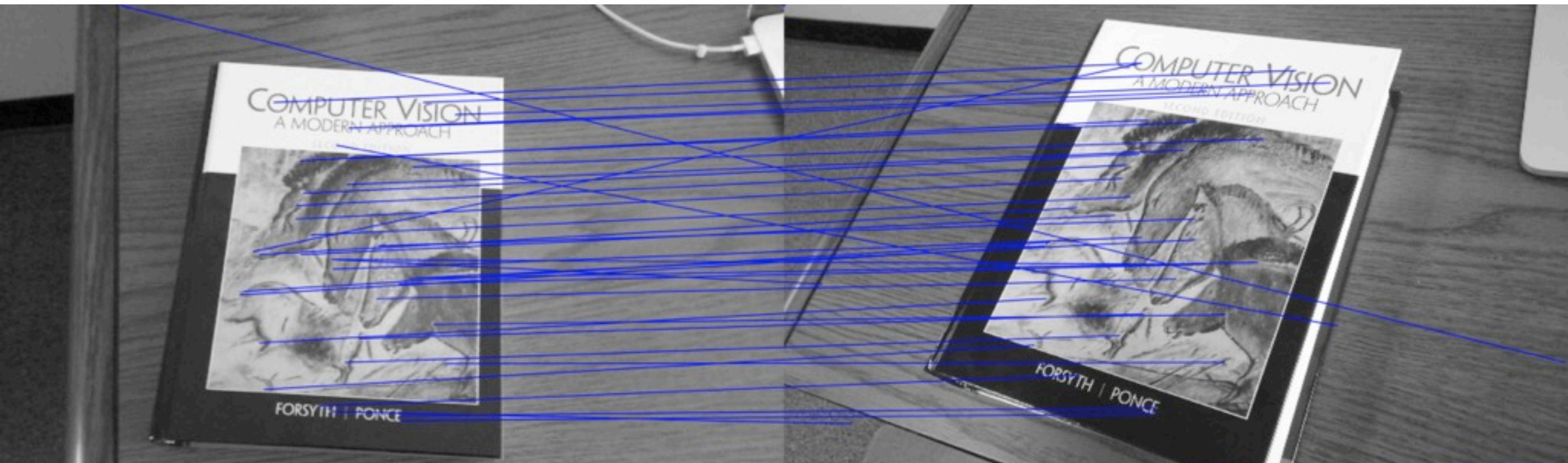
Keep $\mathbf{p_{j2}}$ as match if $SSD(\mathbf{d_{i1}}, \mathbf{d_{j2}})/SSD(\mathbf{d_{i1}}, \mathbf{d_{k2}}) <$ threshold (0.8 in Lowe2004).



Matching Descriptors Between Two Images

Several true positives.

A few false positives.



Matching Descriptors Between Two Images

Several true positives.

A few false positives.

Used in tasks such as point matching for panorama reconstruction, object detection (if several points match between a source and a target image then target image is likely to contain the object in the source image).