Digital Image Processing

Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

Digital Image Processing

(One) **Goal of Computer Vision:**

Automatic Understanding of digital Images!

Image is distorted?

→ Image restoration (e.g. Wiener filter)

Image has still bad quality?

→ Image enhancement (e.g. Histogram equalization)

How to describe an image (or the image content)?

→ Just pixel intensity?!

Image Features

Digital Image Processing

Automatic Image Analysis

Image → Image

Image → Features

Features → “Understanding”

[DIP Winter term]

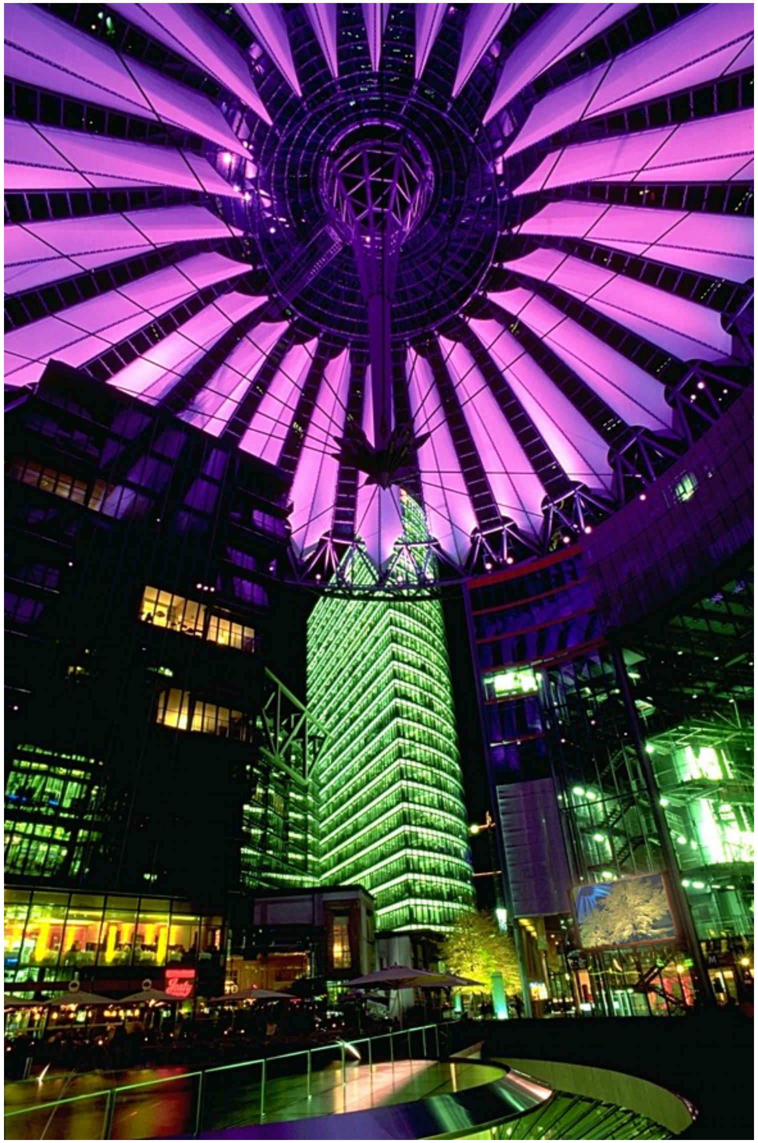
[AIA Summer term]
[PCV Winter term]

How to get a meaningful image description?

How to use this image description to infer information about image content?

Image Features

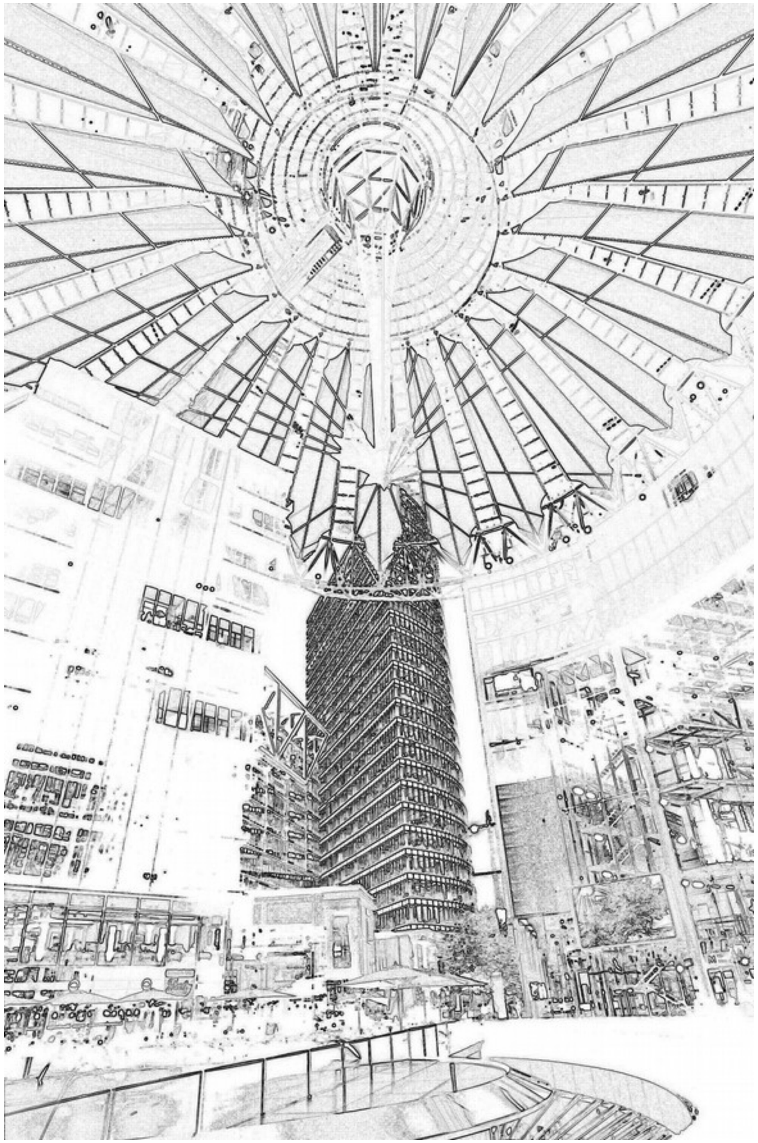
Color



Intensity

Image Features

Edges



Segments

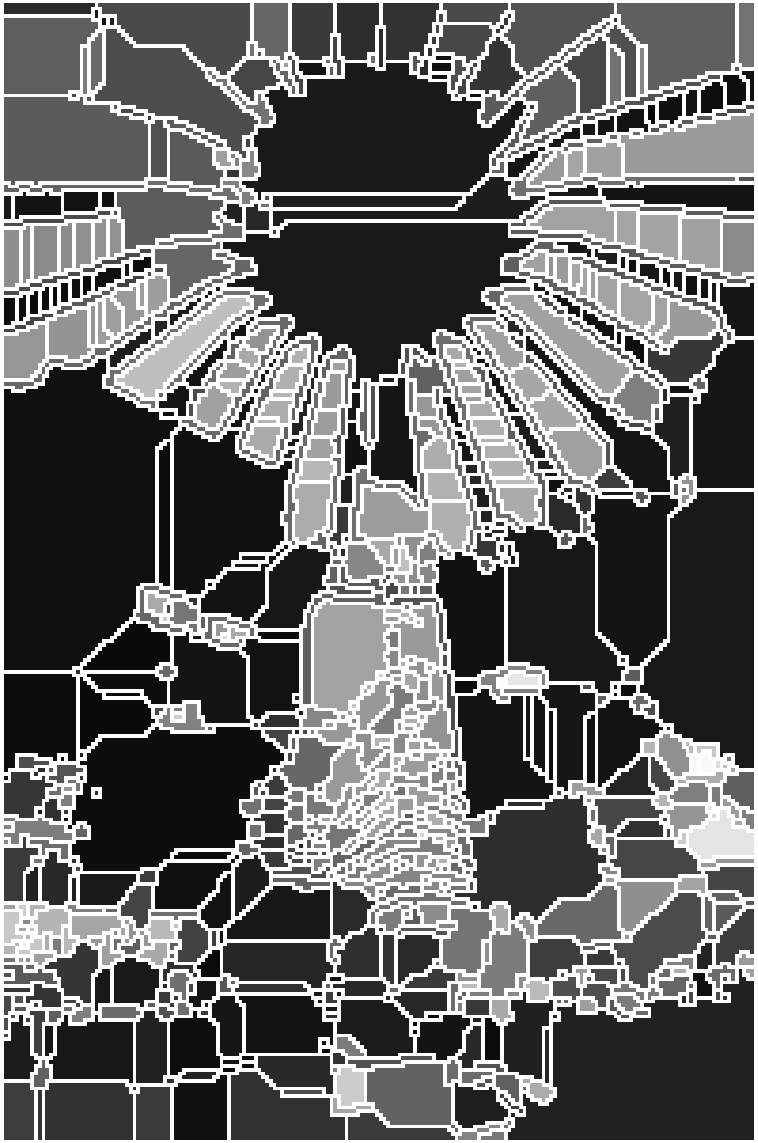
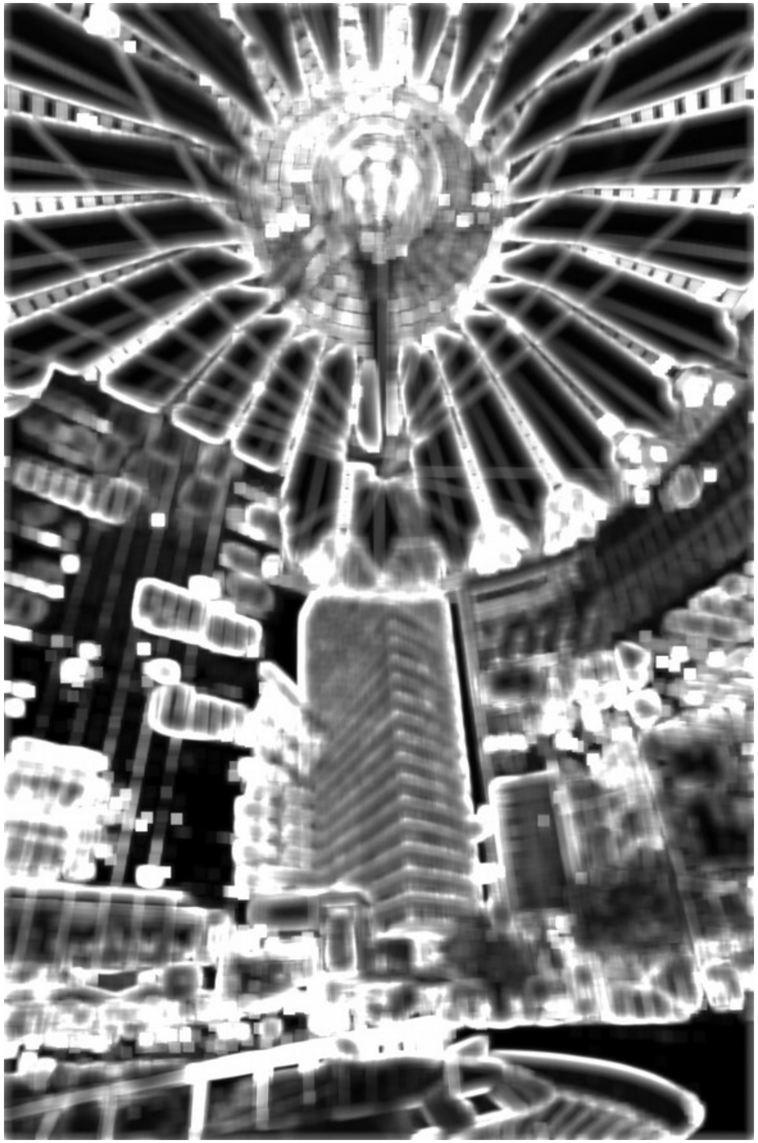


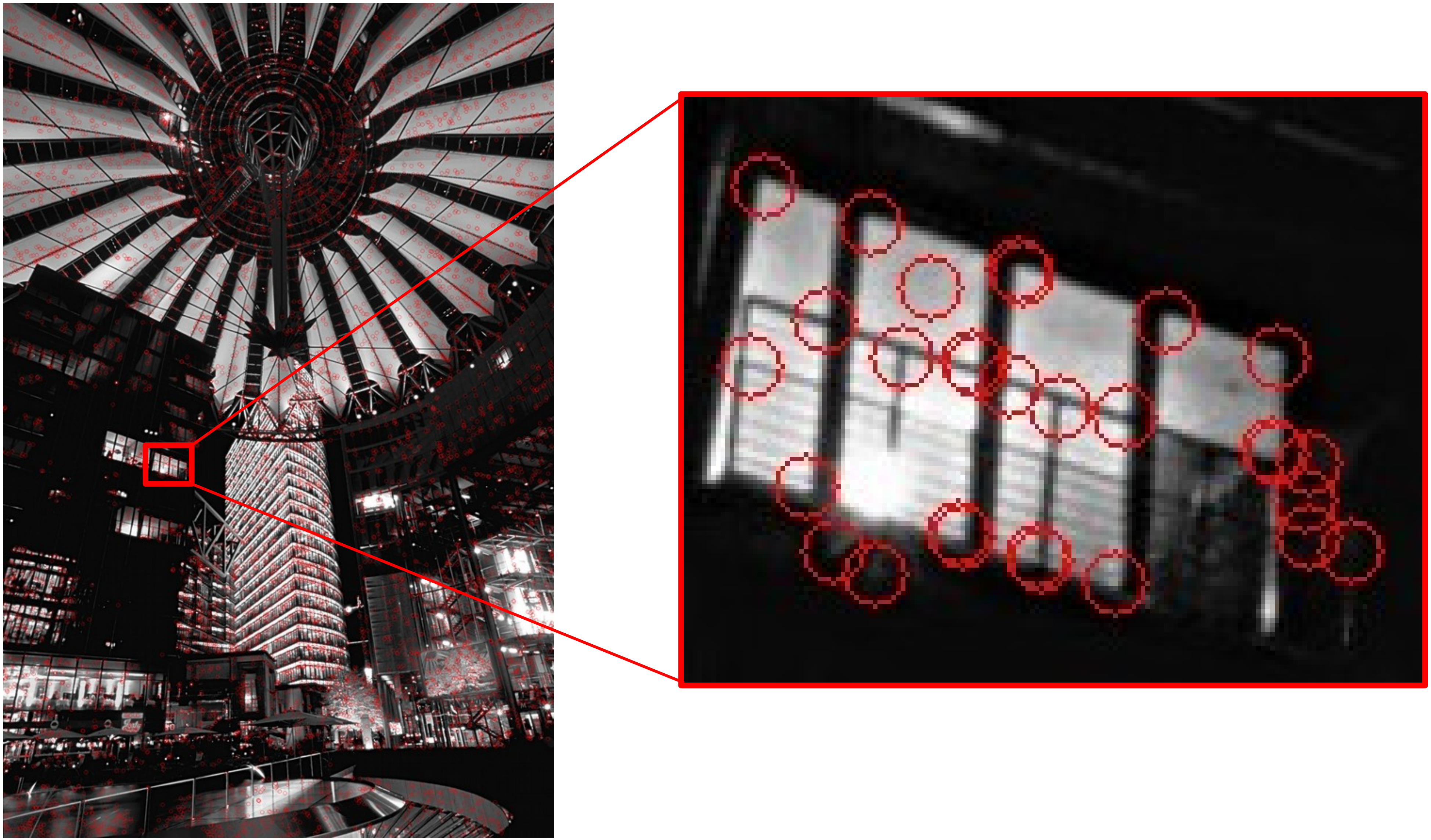
Image Features

Texture

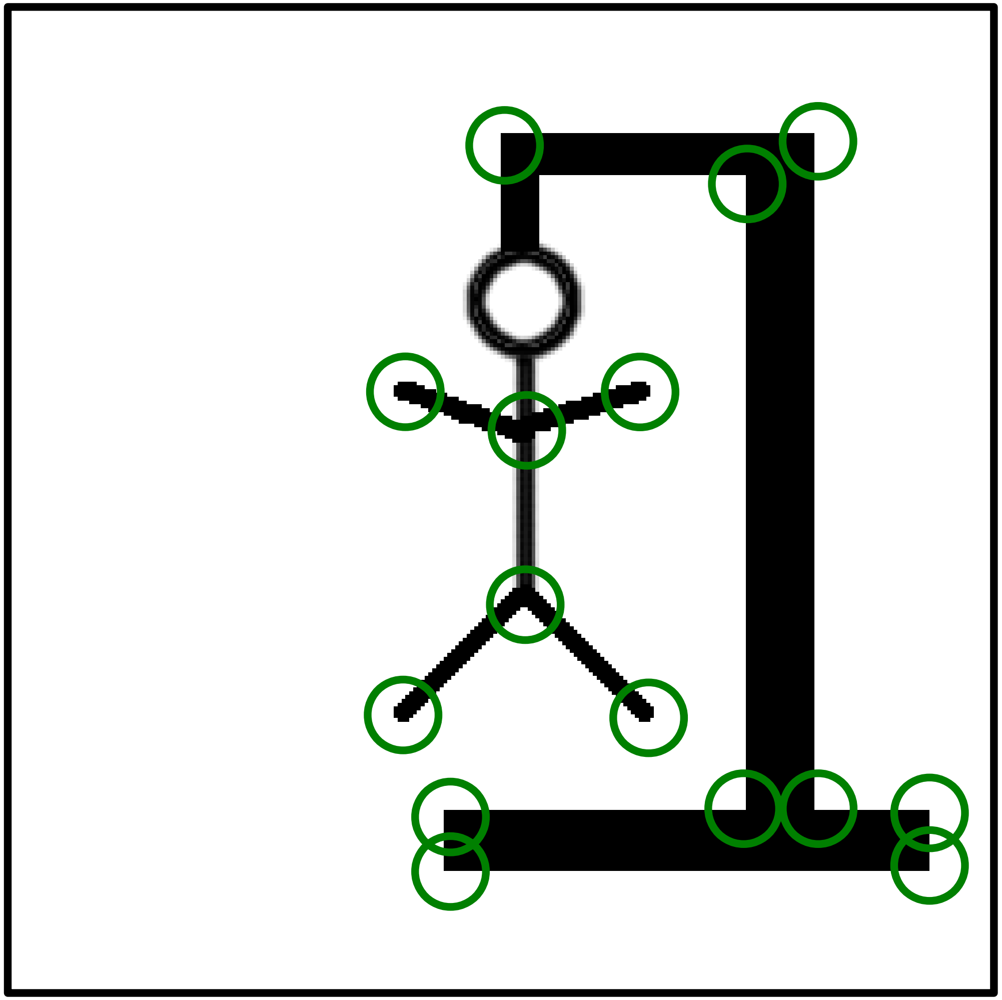


Interest Points

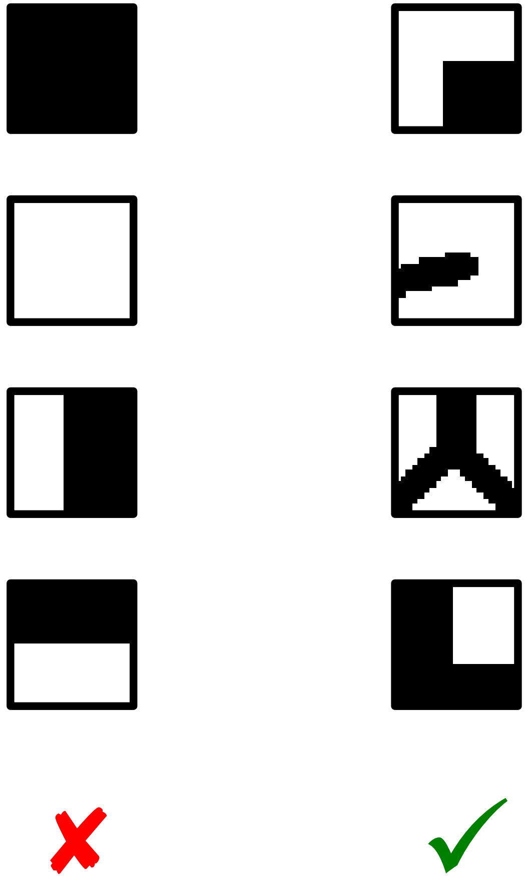
Interest Points



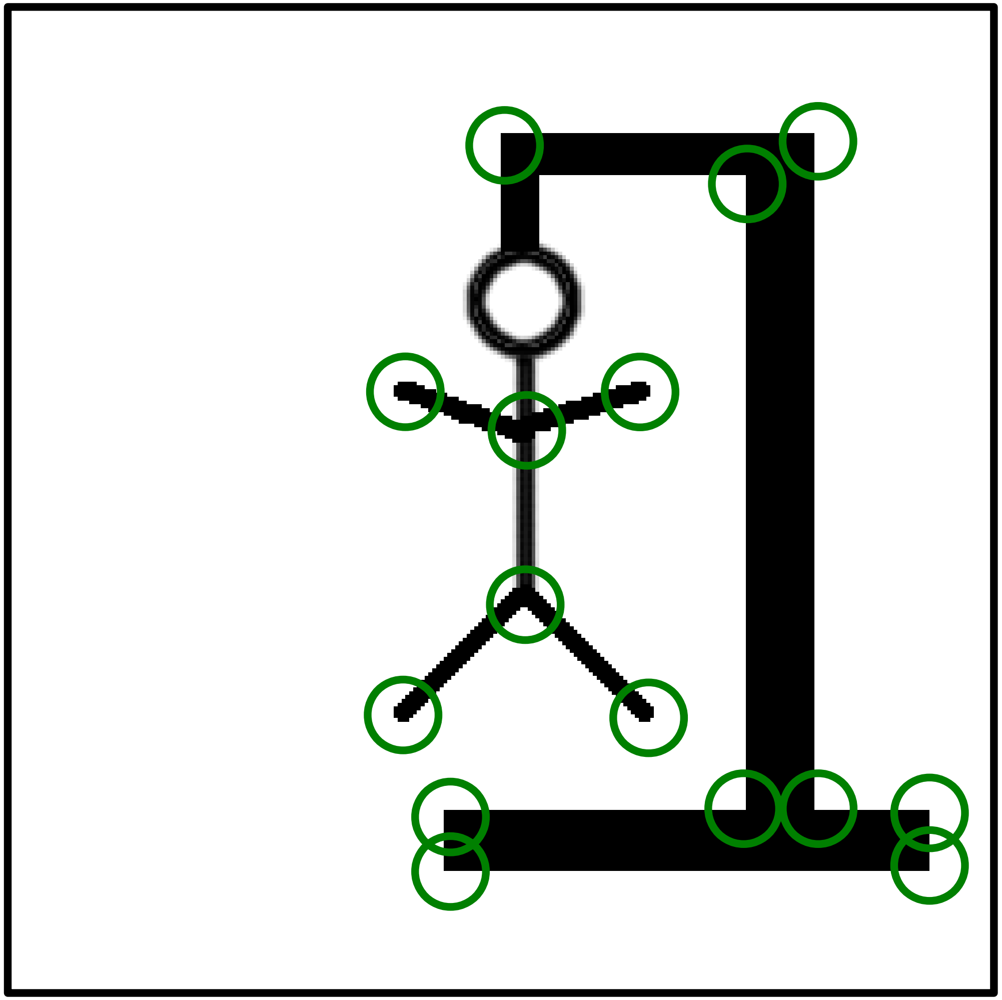
Interest Points



K _ Y P _ I N T ?



Interest Points

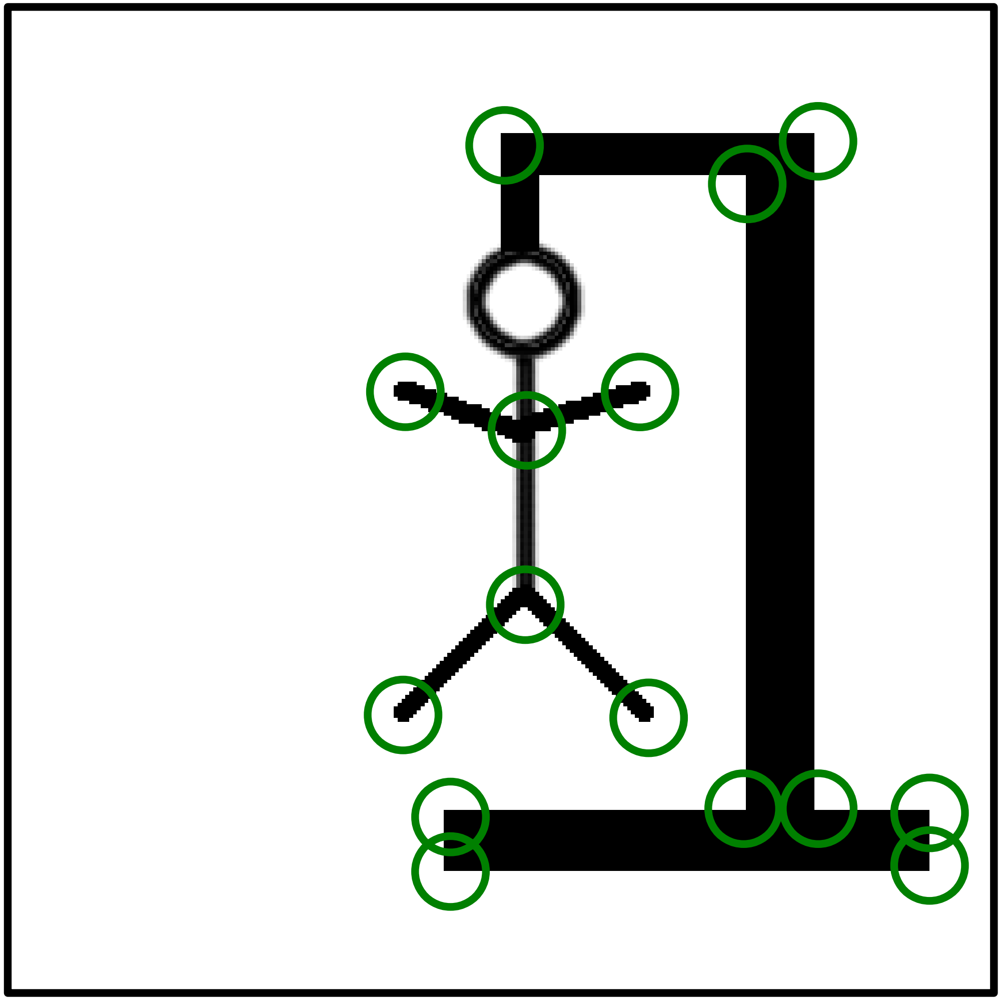


K _ Y P _ I N T ?

Keypoint Detector:

- clear mathematically definition
- well-defined position
- rich local information contents
- stable under perturbations
- reliable

Interest Points



K _ Y P _ I N T ?

Keypoint Detectors:

→ Harris-Stephens

→ Förstner

→ Shi-Tomasi

→ SUSAN

→ FAST

→ SIFT

→ SURF

→ ....

Basics

- Directional Gradients
- Covariance Matrices

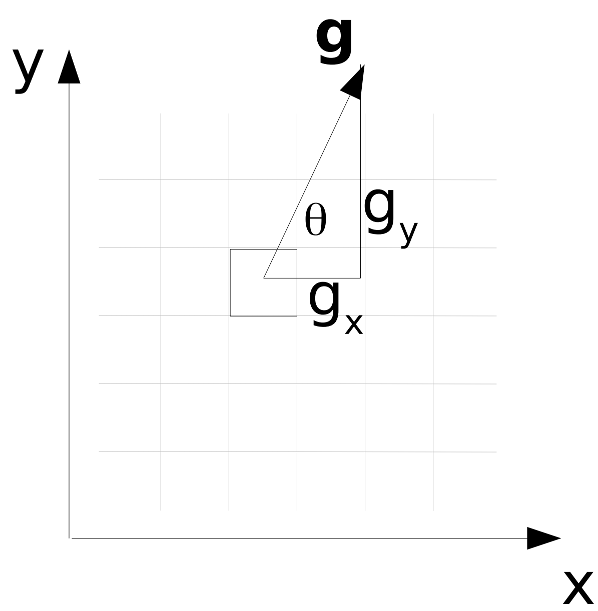
Directional Gradients

- Often only gradient magnitude is computed:
  - Use e.g. a radially symmetric filter
  - No information concerning the direction of gradients

- Now: **Directional gradients**
  - Convolution with suitable filters, e.g. G_x and G_y
  - **Image** $\otimes$ $\mathbf{G_x}$ $\rightarrow$ Gradient in x direction
  - **Image** $\otimes$ $\mathbf{G_y}$ $\rightarrow$ Gradient in y direction

- Each pixel is associated with a gradient vector $\mathbf{g} = (g_x, g_y)^T$

Directional Gradients



- Gradient magnitude: $|g| = \sqrt{g_x^2 + g_y^2}$

- Gradient direction: $\theta = \tan^{-1}\left(\frac{g_y^2}{g_x^2}\right)$

  → Direction in which intensity increases quickest

Directional Gradients

- Commonly Used: Composition of differential operator and low-pass
- E.g. derivatives of the normal distribution:

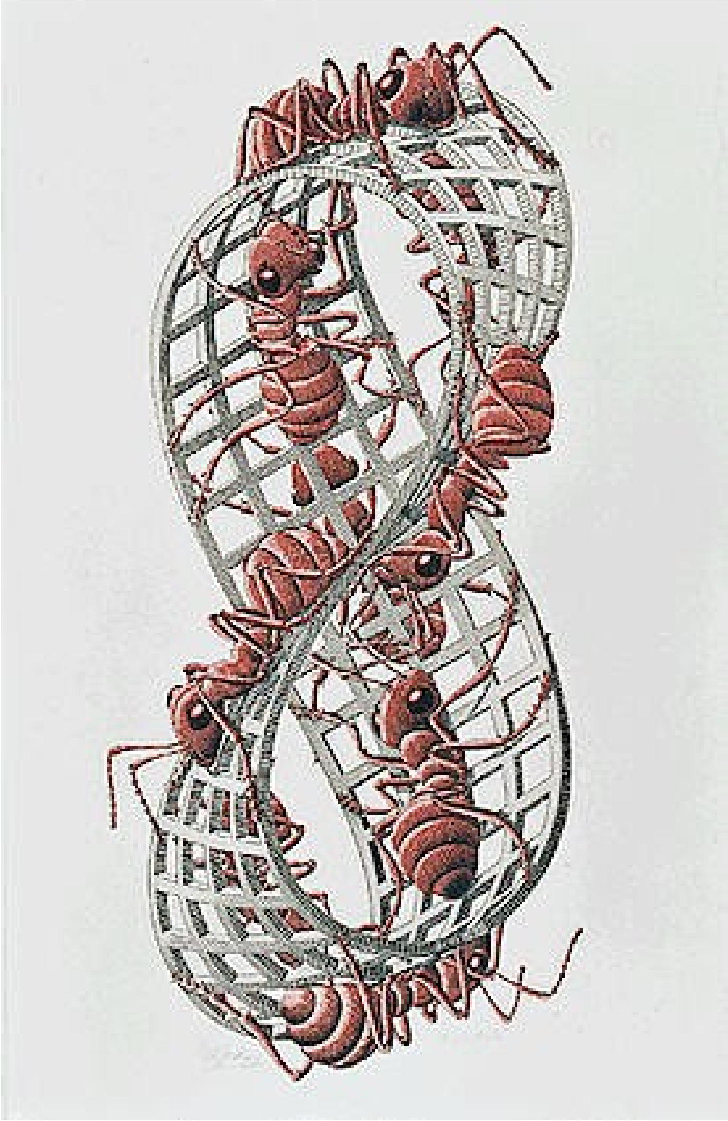
$$G(x,y;\sigma)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$G_x(x,y)=\frac{\partial}{\partial x}G(x,y;\sigma)=\frac{-x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)=\frac{-x}{\sigma^2}G(x,y;\sigma)$$

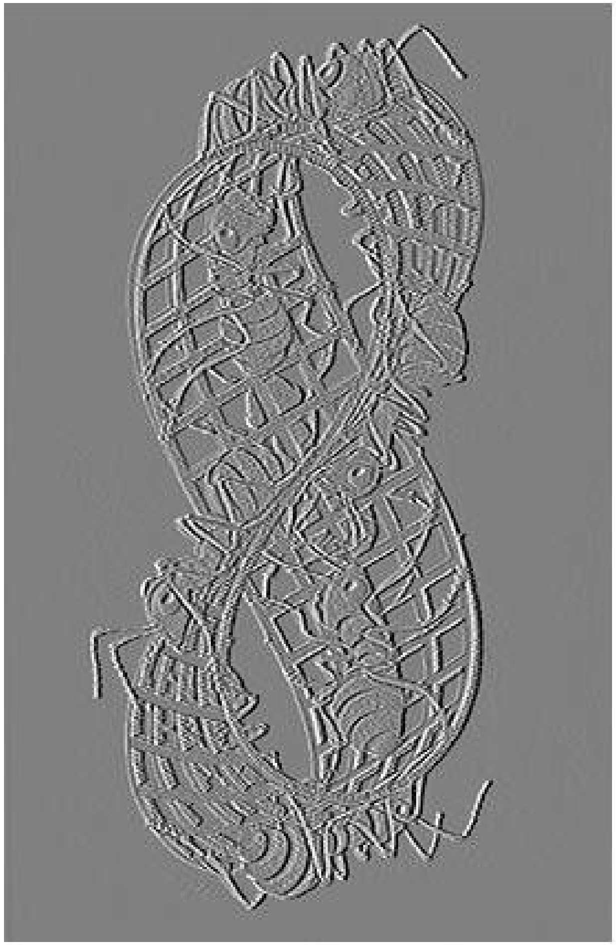
$$G_y(x,y)=\frac{\partial}{\partial y}G(x,y;\sigma)=\frac{-y}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)=\frac{-y}{\sigma^2}G(x,y;\sigma)$$

- σ: Scale and noise sensitivity
  - σ small: Small structures discernible, noise/texture preserved
  - σ large: Large structures emphasized, noise suppressed

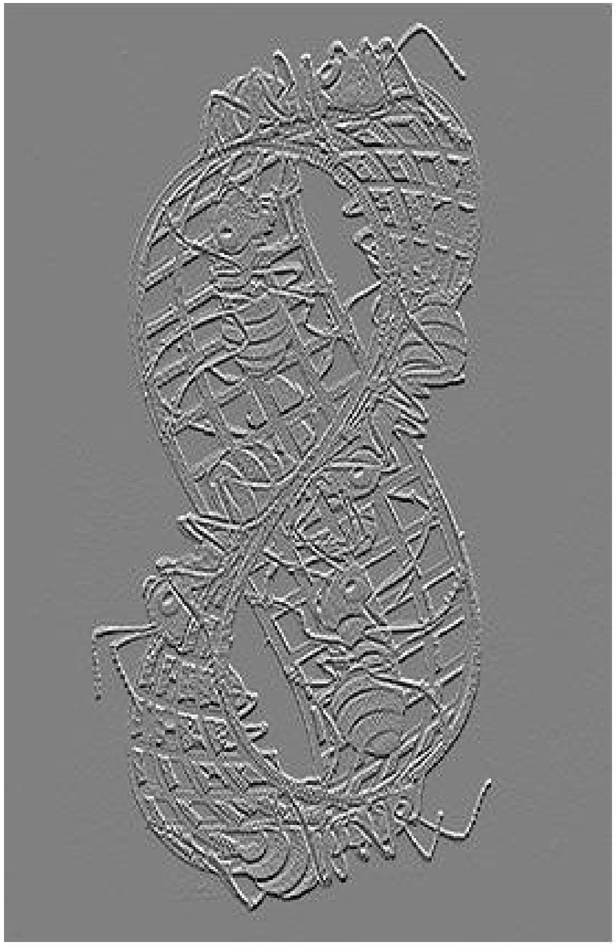
Directional Gradients



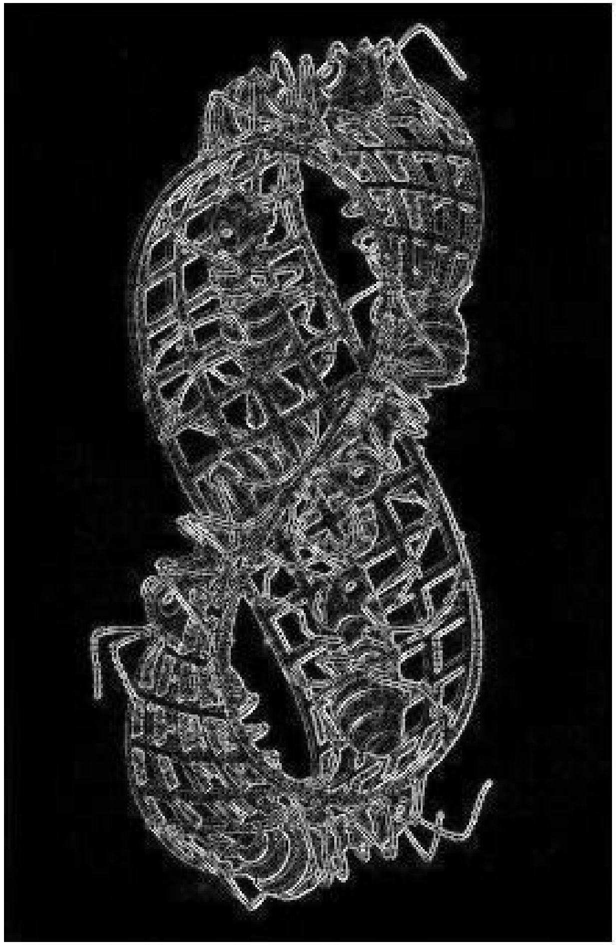
Directional Gradients



g_x



g_y



$|\mathbf{g}|$

Directional Gradients

g_x

g_y

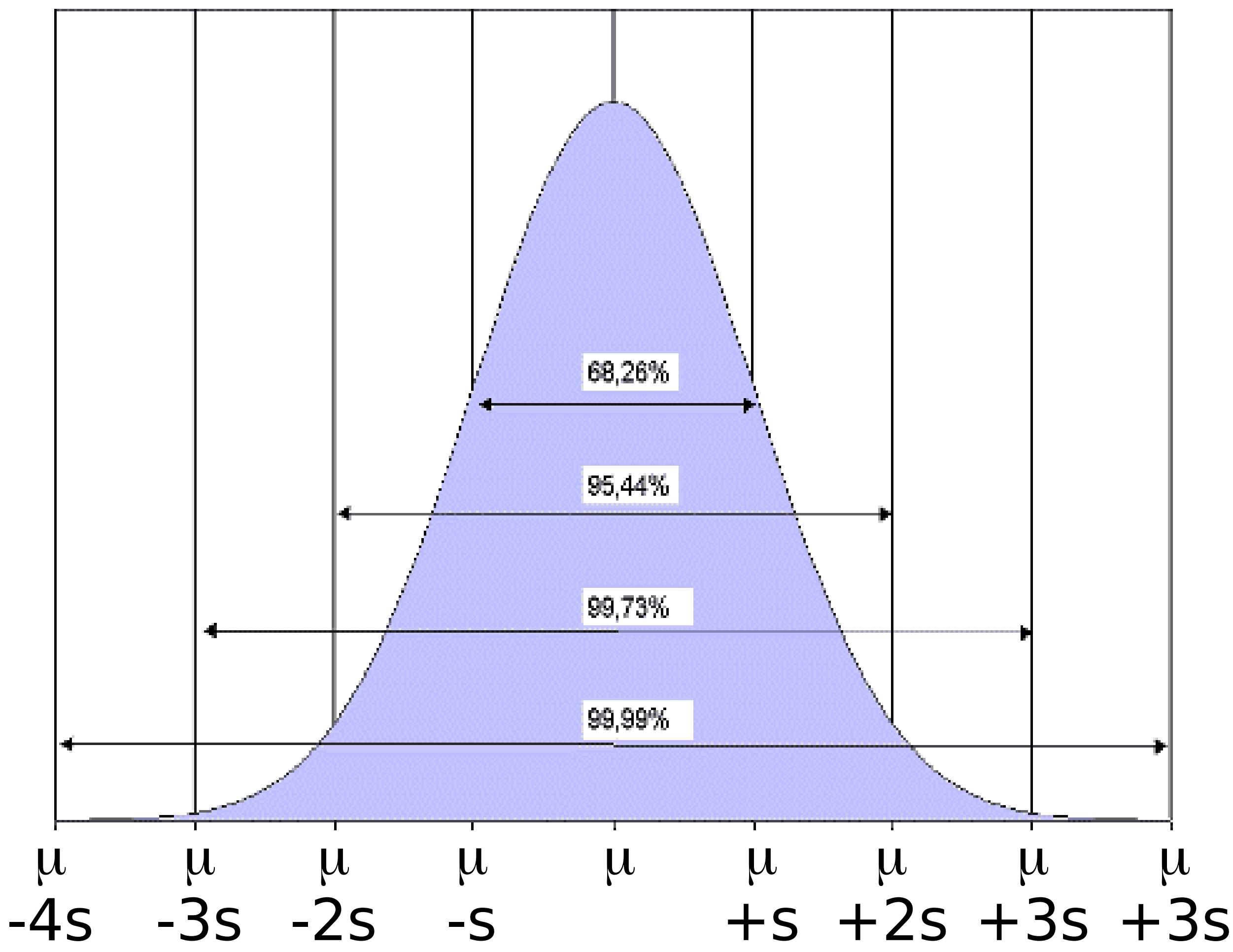
$|\mathbf{g}|$

Basics

- Directional Gradients
- Covariance Matrices

Covariance Matrices

- Variance of scalars $\{x_1,x_2,x_3,\ldots,x_N\}$: Measures dispersion around mean μ



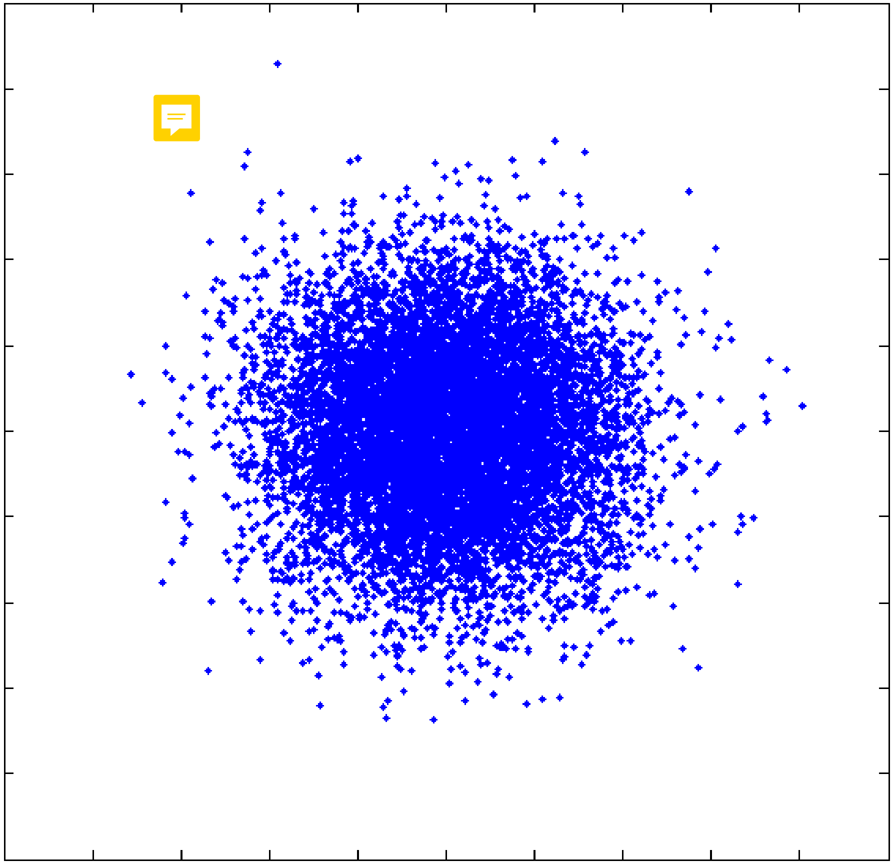
Covariance Matrices

- For sets of vectors $\{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\ldots,\mathbf{x}_N\}$ ($\mathbf{x}_j$ M-dimensional):
  - Mean $$\mu=\frac{1}{N}\sum_{j=1}^{N} x_j$$
  - Covariance $$\Sigma=\frac{1}{N}\sum_{j=1}^{N}(x_j-\mu)(x_j-\mu)^T$$
- For $\mathbf{x}_j = (x_{j,1}, x_{j,2}, x_{j,3}, \ldots, x_{j,M})^T$:

$$\Sigma=\frac{1}{N}\begin{bmatrix} \sum_{j=1}^{N}(x_{j,1}-\mu_1)^2 & \sum_{j=1}^{N}(x_{j,1}-\mu_1)(x_{j,2}-\mu_2) & \cdots & \sum_{j=1}^{N}(x_{j,1}-\mu_1)(x_{j,M}-\mu_M) \\ \sum_{j=1}^{N}(x_{j,2}-\mu_2)(x_{j,1}-\mu_1) & \sum_{j=1}^{N}(x_{j,2}-\mu)^2 & \cdots & \sum_{j=1}^{N}(x_{j,2}-\mu_2)(x_{j,M}-\mu_M) \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

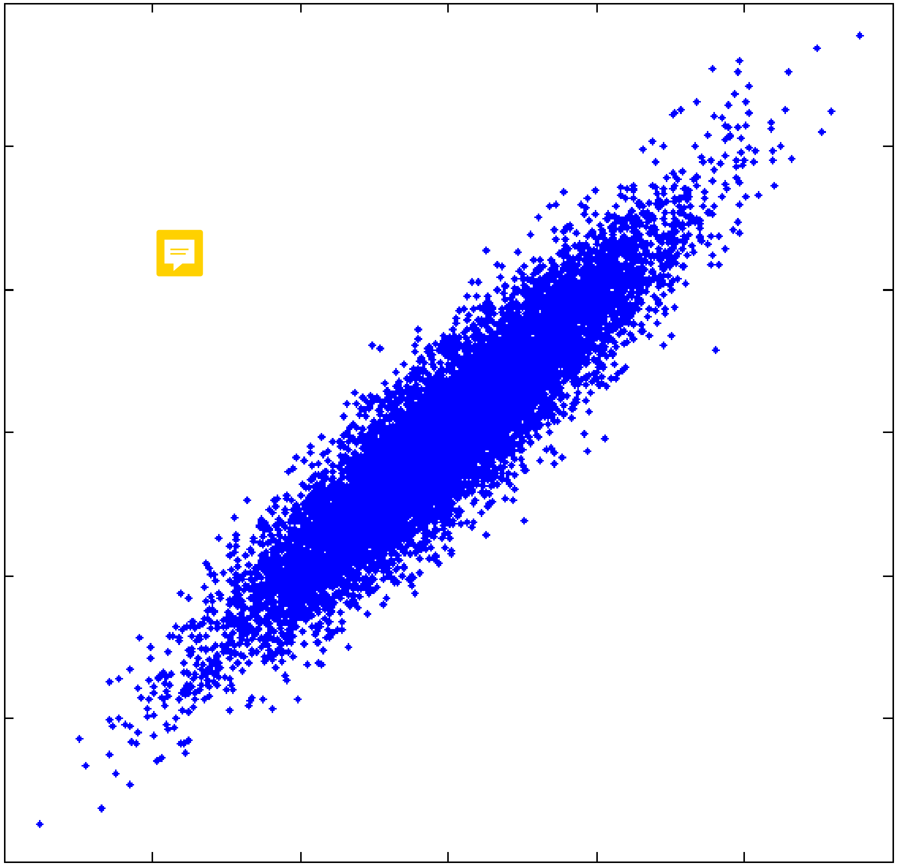
  - Diagonal: Variance along individual dimensions
  - Otherwise: Correlation between dimensions

Covariance Matrices



$$\Sigma = \begin{pmatrix} 0.9976 & -0.0187 \\ -0.0187 & 0.9700 \end{pmatrix}$$

Both dimensions almost uncorrelated

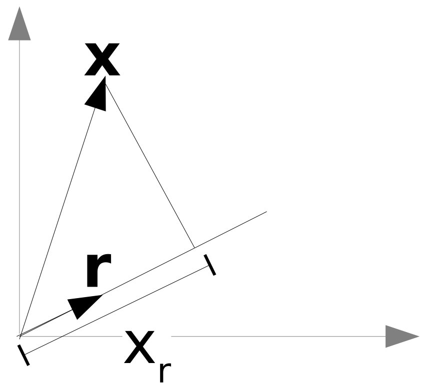


$$\Sigma = \begin{pmatrix} 0.4998 & 0.4625 \\ 0.4625 & 0.5054 \end{pmatrix}$$

Both dimensions strongly correlated

Covariance Matrices

- Component of a vector **x** in direction **r**:



- **r** = (cos θ, sin θ)^T
- $x_r = \mathbf{r}^T \mathbf{x}$
- x_r: Scalar, component of **x** along **r**

- Mean and variance of vectors $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N\}$ along direction **r**:

$$\mu_r = \frac{1}{N}\sum_j x_{r,j} = \frac{1}{N}\sum_j r^T x_j = r^T \frac{1}{N}\sum_j x_j = r^T \mu$$

$$\Sigma_r = \frac{1}{N}\sum_j (x_{r,j} - \mu_r)(x_{r,j} - \mu_r)^T = \frac{1}{N}\sum_j (r^T x_j - r^T \mu)(r^T x_j - r^T \mu)^T$$

$$= \frac{1}{N}\sum_j r^T (x_j - \mu)(x_j - \mu)^T r = r^T \Sigma r$$

- The covariance matrix determines the variance in all directions!

Covariance Matrices

Task: Find directions with maximal variance, i.e.: $r^T \Sigma r = max$

Solution: $\Sigma = \mathbf{V}\,\mathbf{D}\,\mathbf{V}^T$

$$\begin{pmatrix} \uparrow & \uparrow & \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots \\ \downarrow & \downarrow & \end{pmatrix} \qquad \begin{pmatrix} l_1 & 0 & \cdots \\ 0 & l_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Eigenvectors Eigenvalues (diagonal)

- In the eigenbasis, dimensions of vectors **x** are not correlated

- Variance along in the directions $\mathbf{v}_1$, $\mathbf{v}_2$ ... : l_1, l_2, ...
- Standard deviations in other directions form an ellipse with major/minor axes along eigenvectors with deviations given by eigenvalues

Structure Tensor

- For each pixel, the structure tensor **A** is defined as:

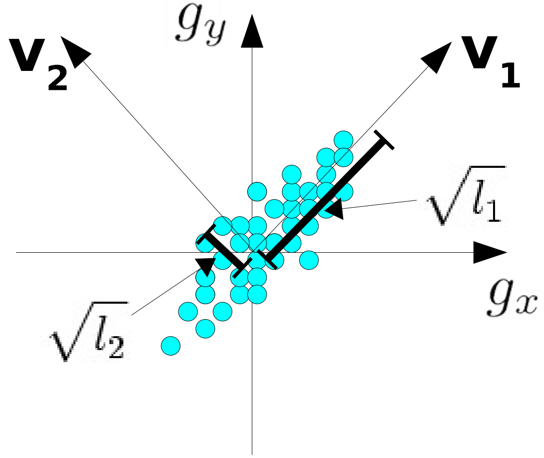
$$\mathbf{A} = \sum_W \mathbf{g}\mathbf{g}^T = \begin{pmatrix} \sum_W g_x^2 & \sum_W g_x g_y \\ \sum_W g_y g_x & \sum_W g_y^2 \end{pmatrix}$$

- *W* denotes the neighborhood of the pixel considered
  - ➔ In this exercise: Gaussian window with std-dev N_W around the pixel
- **A** is a covariance matrix computed assuming $\mu = 0$
  - ➔ **A** describes the distribution of gradients around **g** = $(0,0)^T$

- For covariance **Σ**: $\mathbf{r}^T \mathbf{\Sigma} \mathbf{r} =$ Variance along **r**

- For the structure tensor **A**: $\mathbf{r}^T \mathbf{A} \mathbf{r} =$ (Squared) gradient magnitude along **r**

Structure Tensor



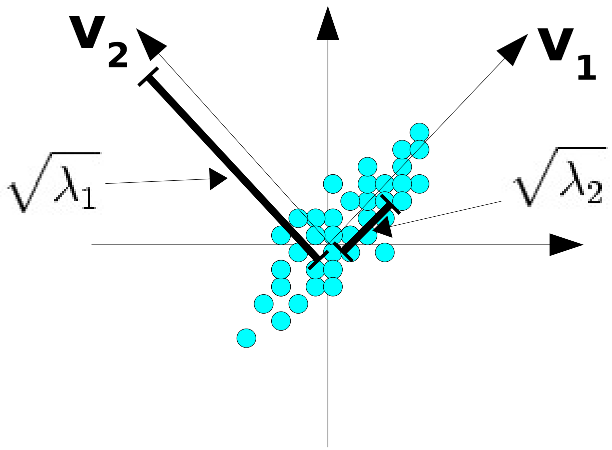
$$\mathbf{A} = \mathbf{V}\,\mathbf{D}\,\mathbf{V}^T$$

$$\mathbf{V} = \begin{pmatrix} \uparrow & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 \\ \downarrow & \downarrow \end{pmatrix} \qquad \mathbf{D} = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix}$$

$$l_1 \geq l_2$$

- $\mathbf{v}_1$: Direction with the greatest gradient magnitude (max. eigenvalue l_1)
  - ➔ Gradient direction $\mathbf{v}_1$ dominates neighborhood W
- l_1: Total (squared) gradient magnitude along direction $\mathbf{v}_1$
- $\mathbf{v}_2$: Direction with the smallest gradient magnitude
  - ➔ Gradient direction $\mathbf{v}_2$ is rare in neighbourhood W
- Gradient magnitude as a function of direction describes an ellipse with major/minor axes along $\mathbf{v}_1$ und $\mathbf{v}_2$

Structure Tensor



$$\mathbf{A}^{-1} = (\mathbf{V}\ \mathbf{D}\ \mathbf{V}^T)^{-1} = \mathbf{V}\ \mathbf{D}^{-1}\ \mathbf{V}^T$$

$$\mathbf{D}^{-1} = \begin{pmatrix} \frac{1}{l_2} & 0 \\ 0 & \frac{1}{l_1} \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

- Eigenvectors remain unchanged
- Eigenvalues are inverted
- Small eigenvalues λ_1 und λ_2 indicate strong gradients in the neighborhood
- If λ_1 and λ_2 are large, the image is homogeneous

Förstner Operator

- The structure tensor can be used to derive salient information:
- Weight **w**: Strength of gradients in the neighborhood

$$w = \frac{1}{\operatorname{tr}\left(\mathbf{A}^{-1}\right)} = \frac{1}{\lambda_1 + \lambda_2} = \frac{\det(\mathbf{A})}{\operatorname{tr}(\mathbf{A})} \qquad w > 0$$

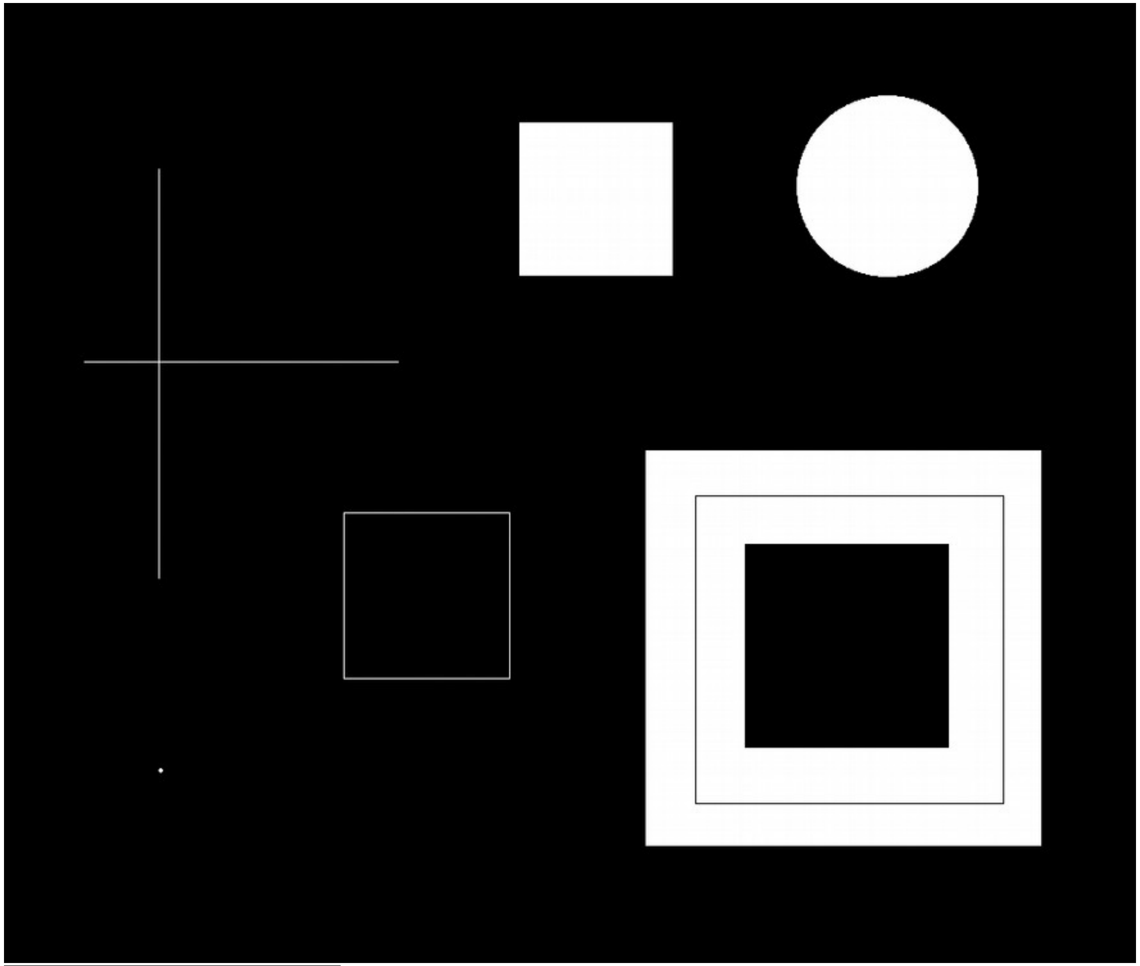
  - → w large: λ_1 und λ_2 small, i.e. strong gradients in the neighborhood
  - → $w_{min} = 0.5, \ldots, 1.5 \cdot \bar{w}$, $\bar{w}$ is the mean of w over whole image

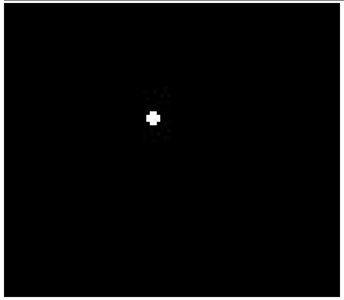
- Isotropy **q**: Measures the uniformity of gradient directions in the neighbourhood

$$q = 1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 = \frac{4\det\left(\mathbf{A}\right)}{\operatorname{tr}\left(\mathbf{A}\right)^2} \qquad 0 \leq q \leq 1$$

  - → **q** small: Gradients occur primarily in one direction
  - → $q_{min} = 0.5, \ldots, 0.75$

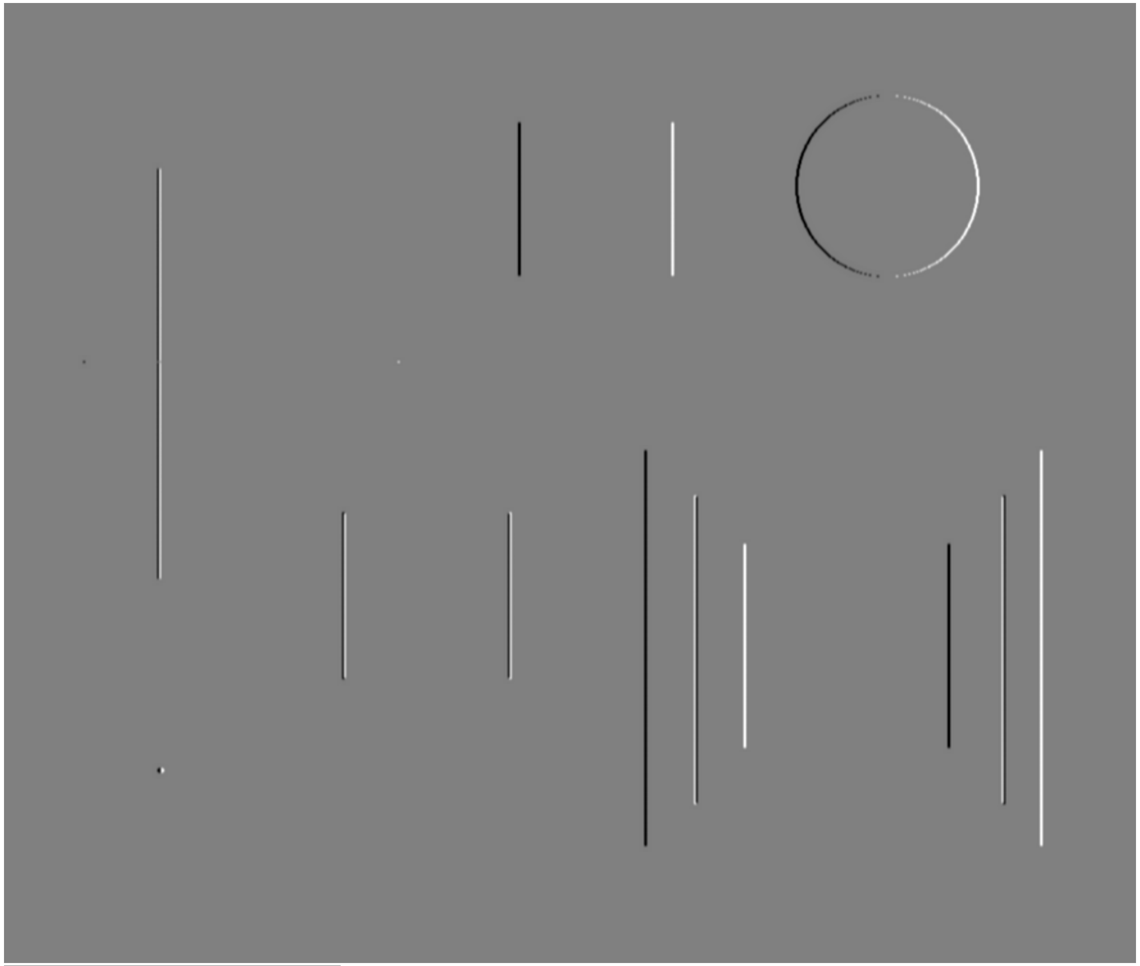
Förstner Operator



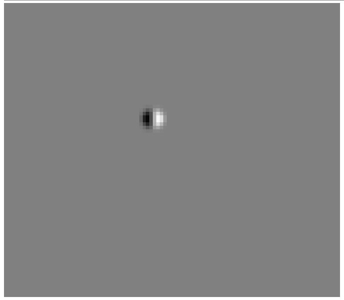


Original image

Förstner Operator

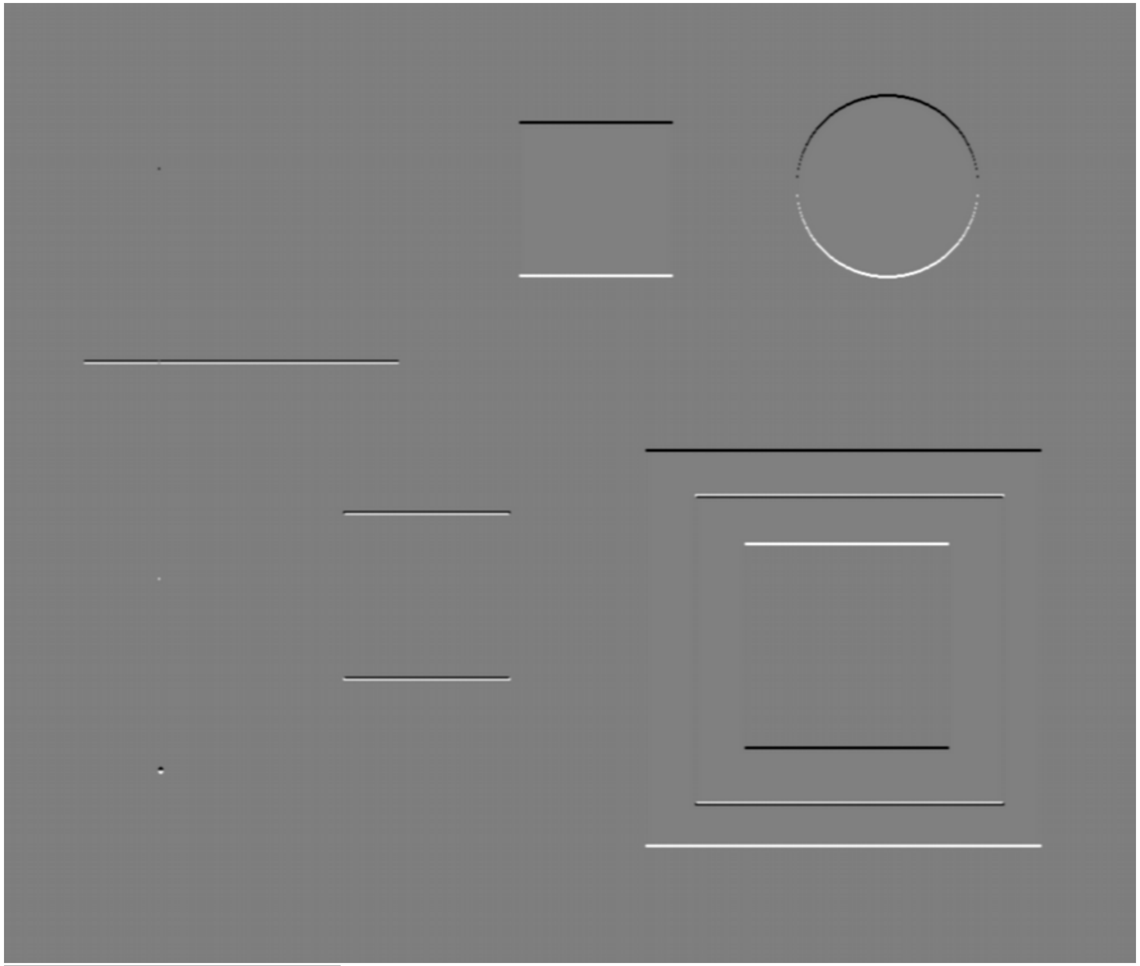


1. Gradient in x-direction

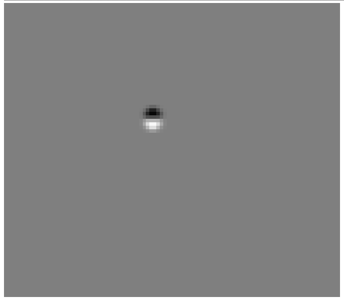


Gradient in x-direction

Förstner Operator

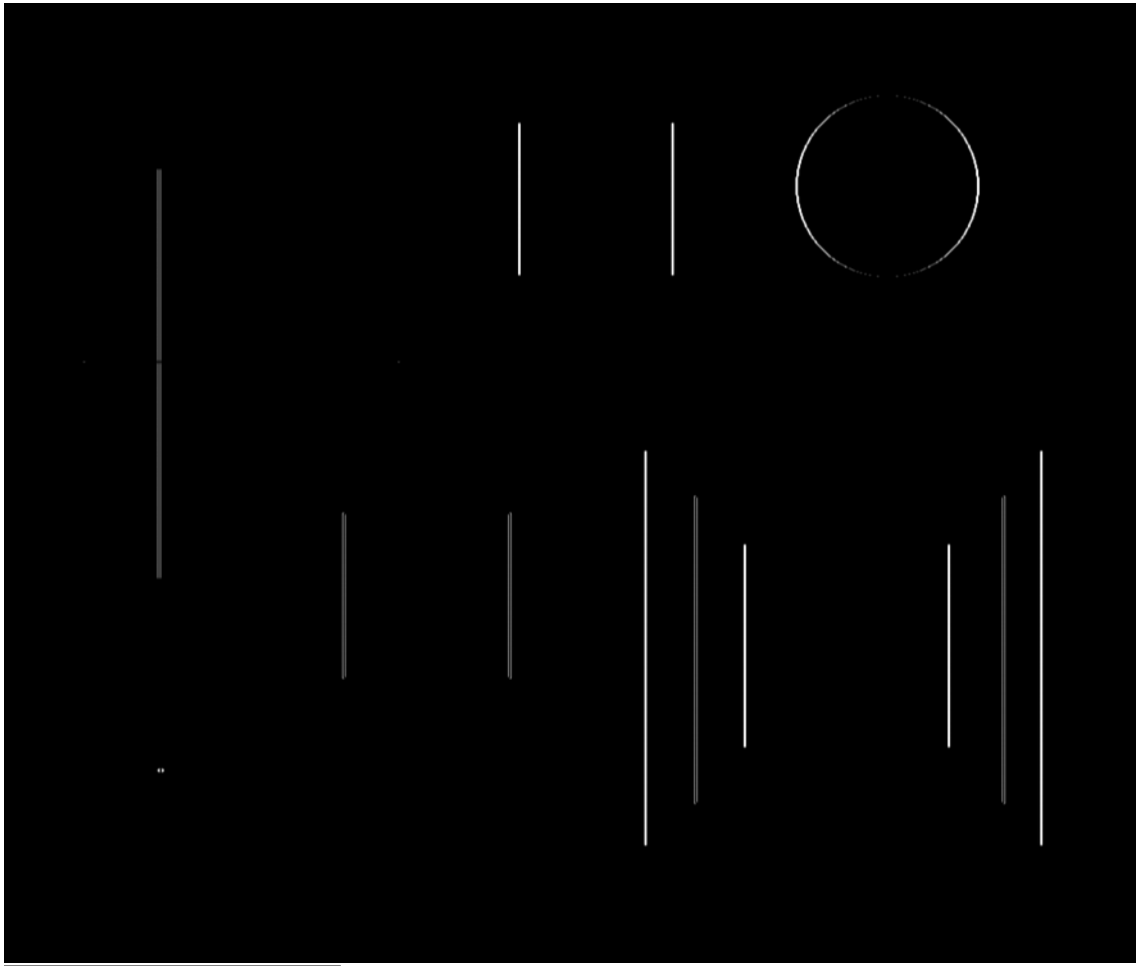


1. Gradient in x- and y-direction

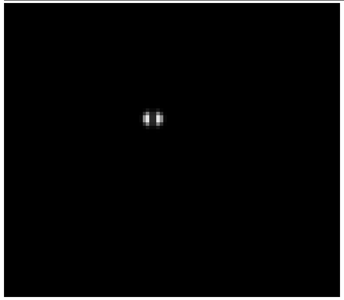


Gradient in y-direction

Förstner Operator

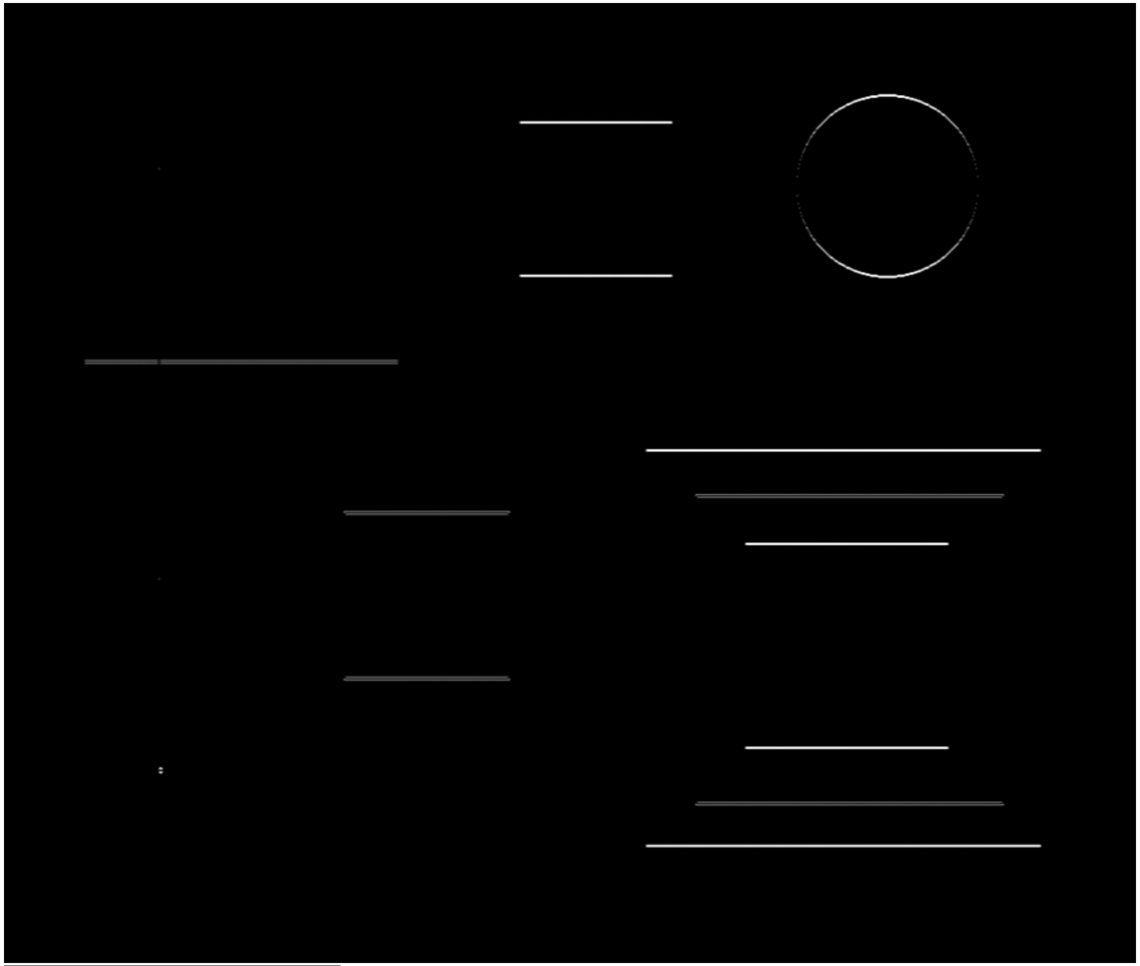


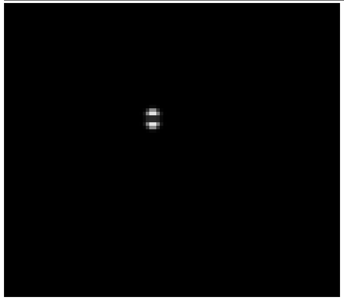
1. Gradient in x- and y-direction
2. $g_x \cdot g_x$



$g_x \cdot g_x$

Förstner Operator

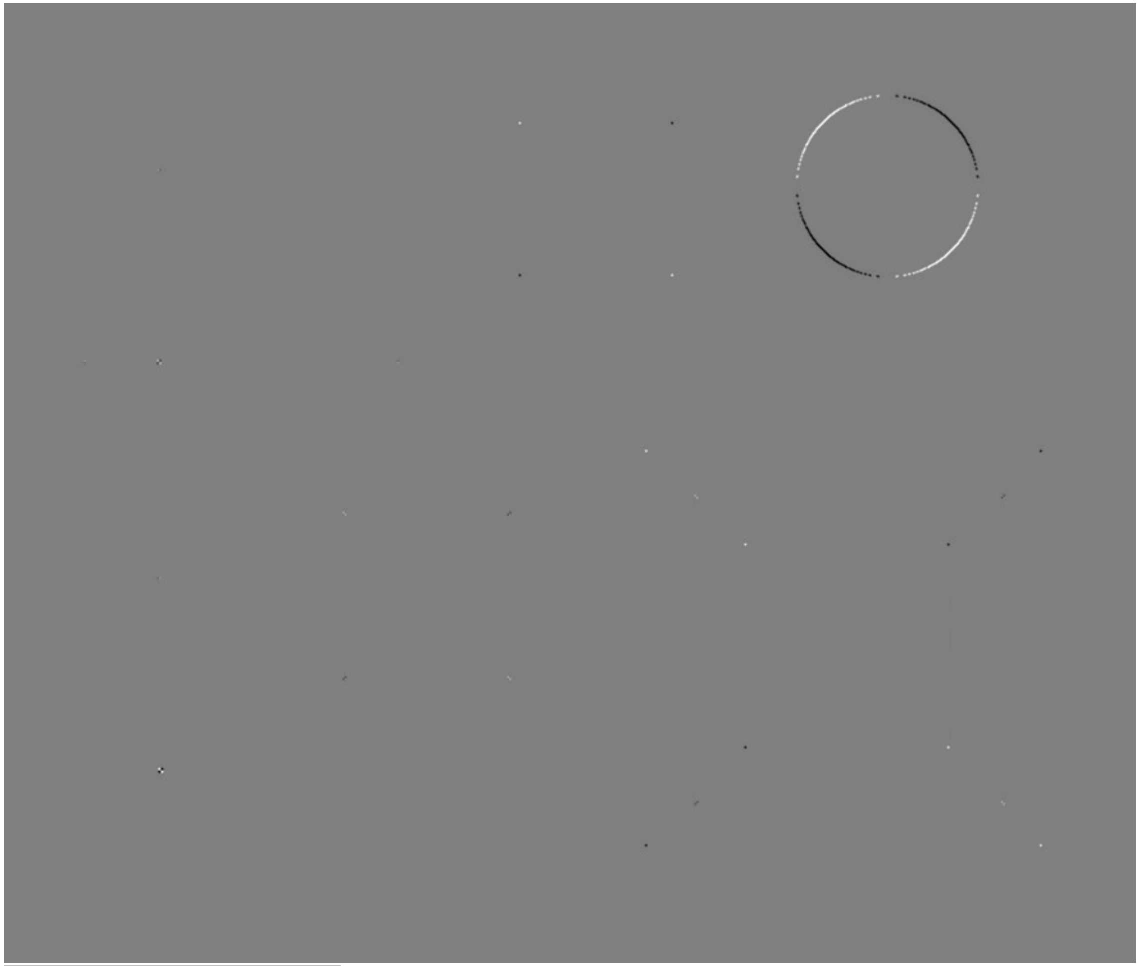




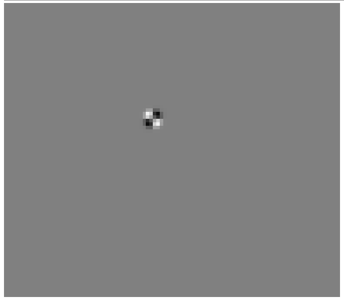
$g_y \cdot g_y$

1. Gradient in x- and y-direction
2. $g_x \cdot g_x, g_y \cdot g_y$

Förstner Operator

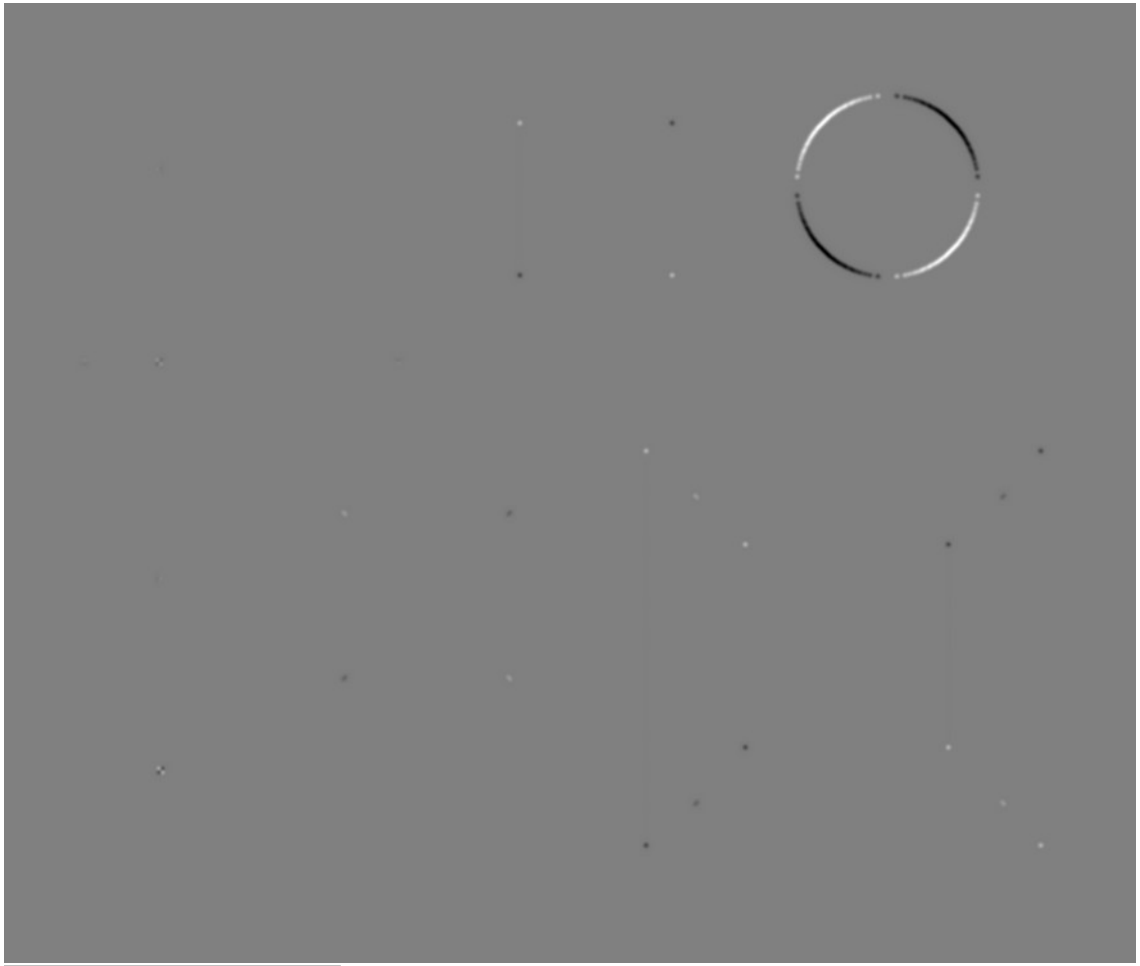


1. Gradient in x- and y-direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$

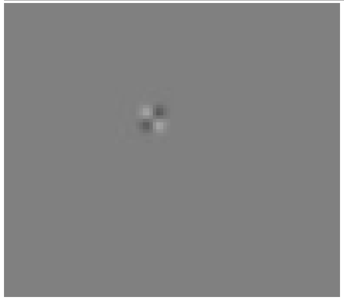


$g_x \cdot g_y$

Förstner Operator

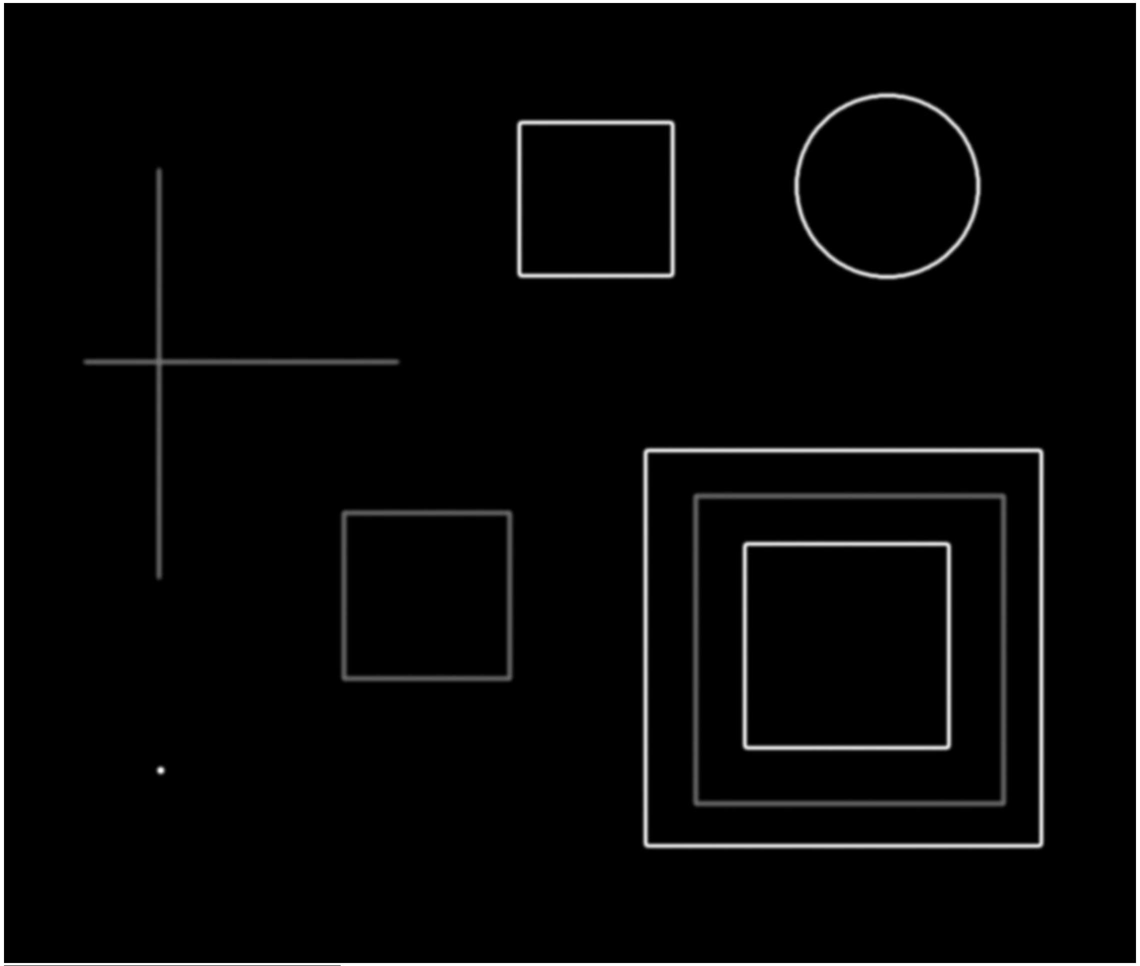


1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)

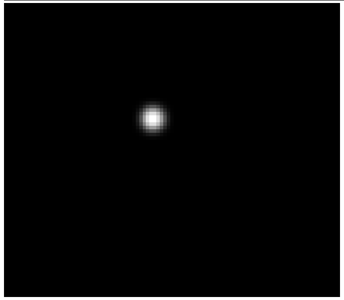


Averaged (smoothed) $g_x \cdot g_y$

Förstner Operator

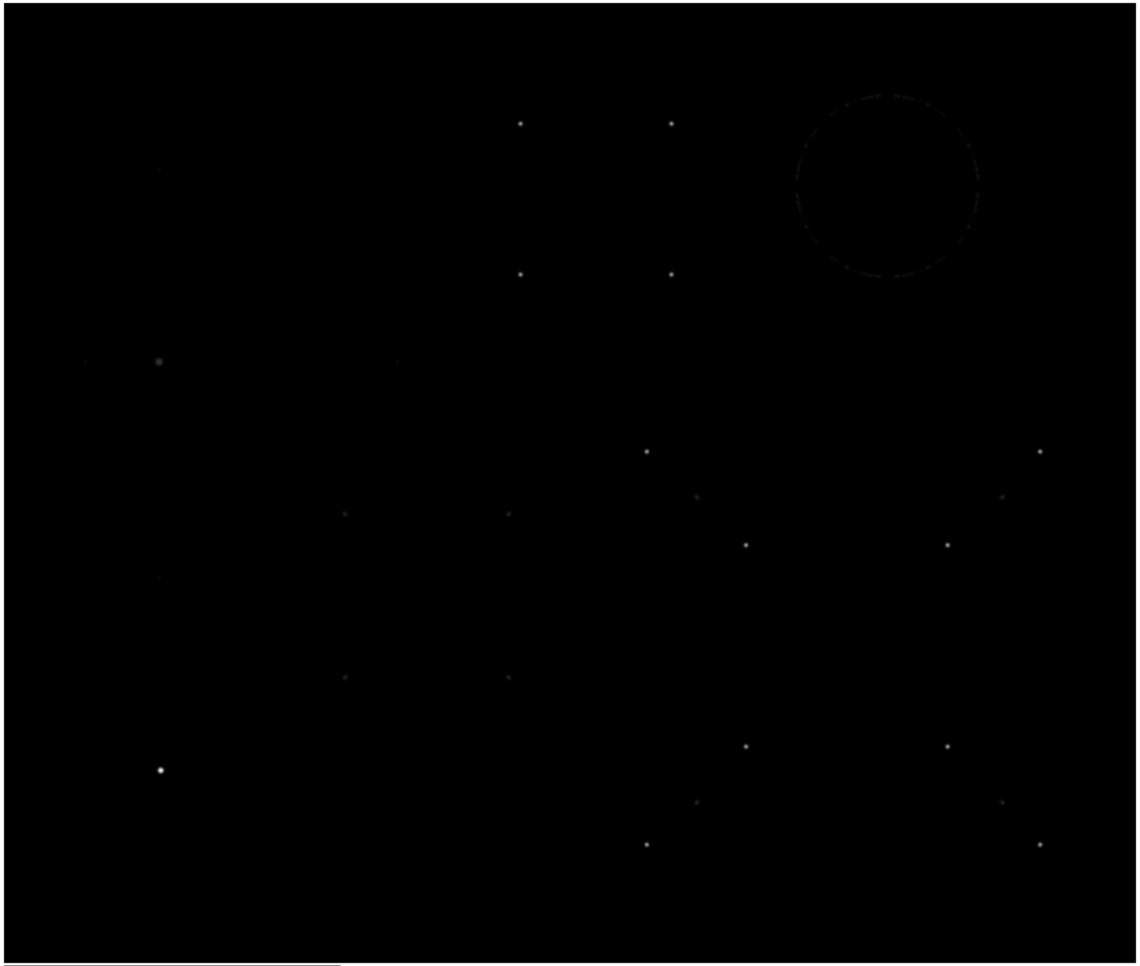


1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)
4. Trace of structure tensor

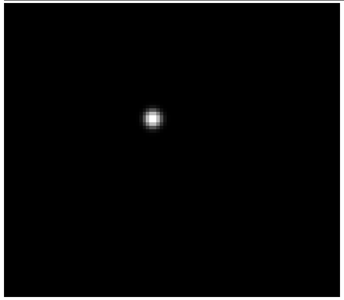


tr(A)

Förstner Operator

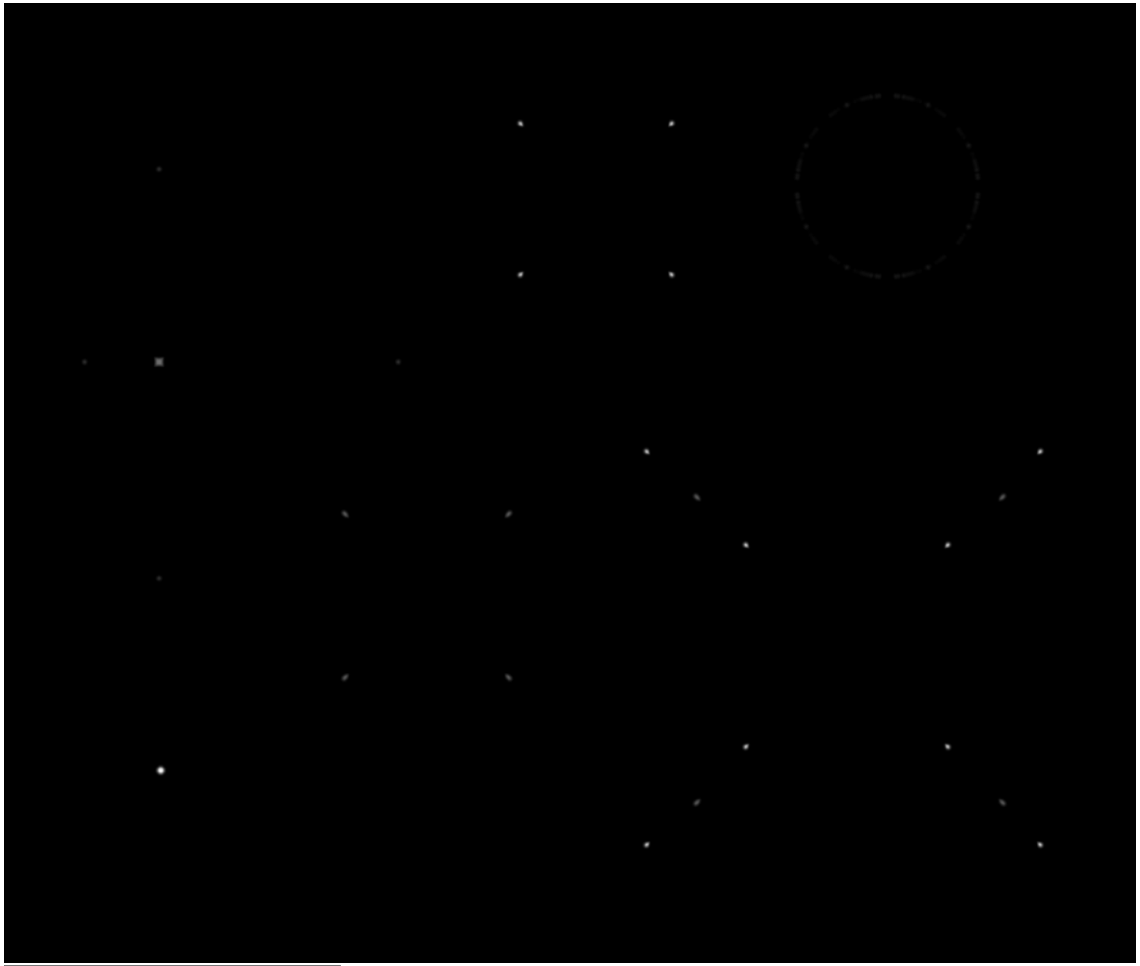


1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)
4. Trace of structure tensor
5. Determinant of structure tensor

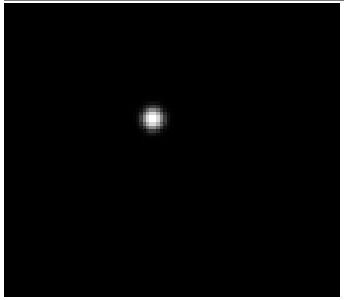


|A|

Förstner Operator

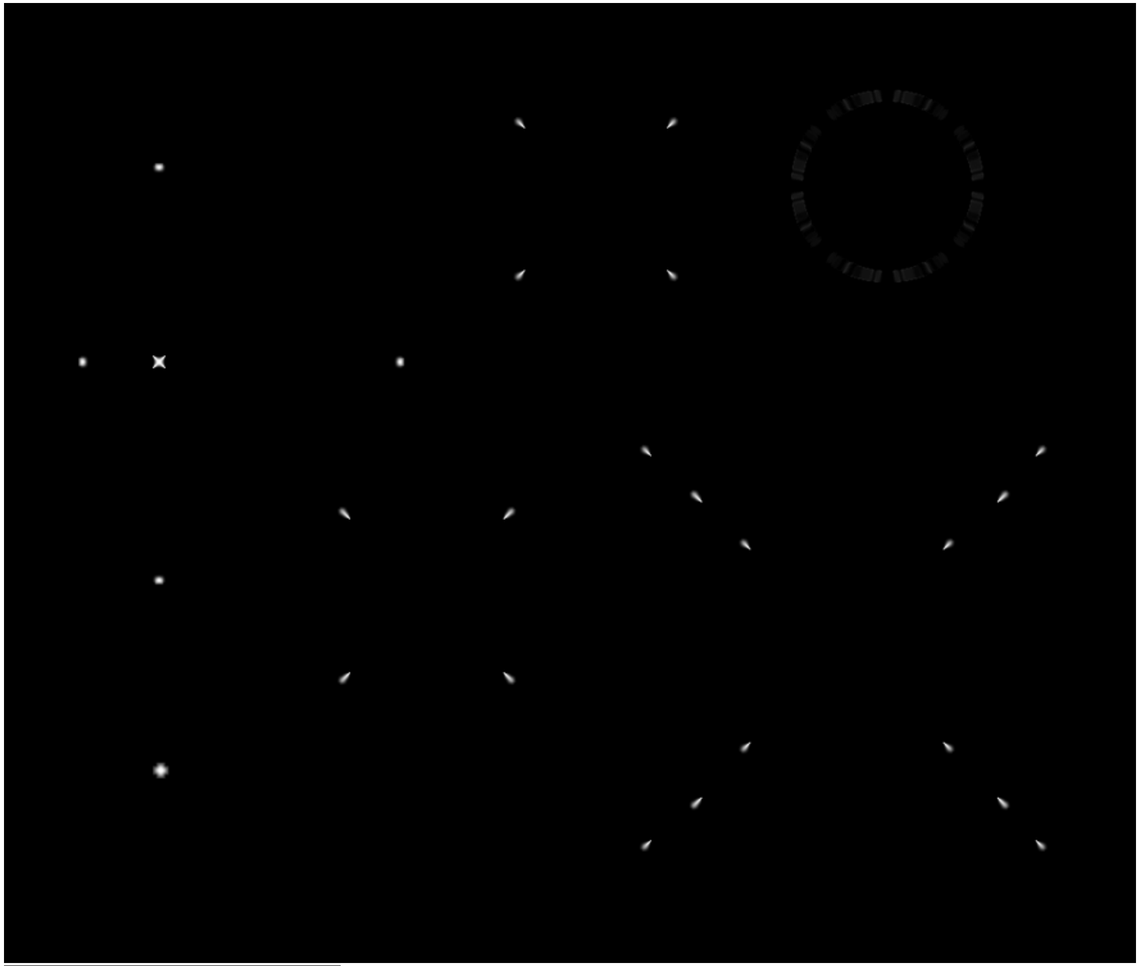


1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)
4. Trace of structure tensor
5. Determinant of structure tensor
6. weight calculation

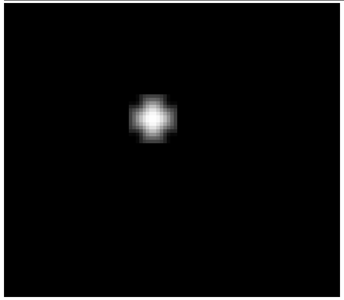


Weight w

Förstner Operator

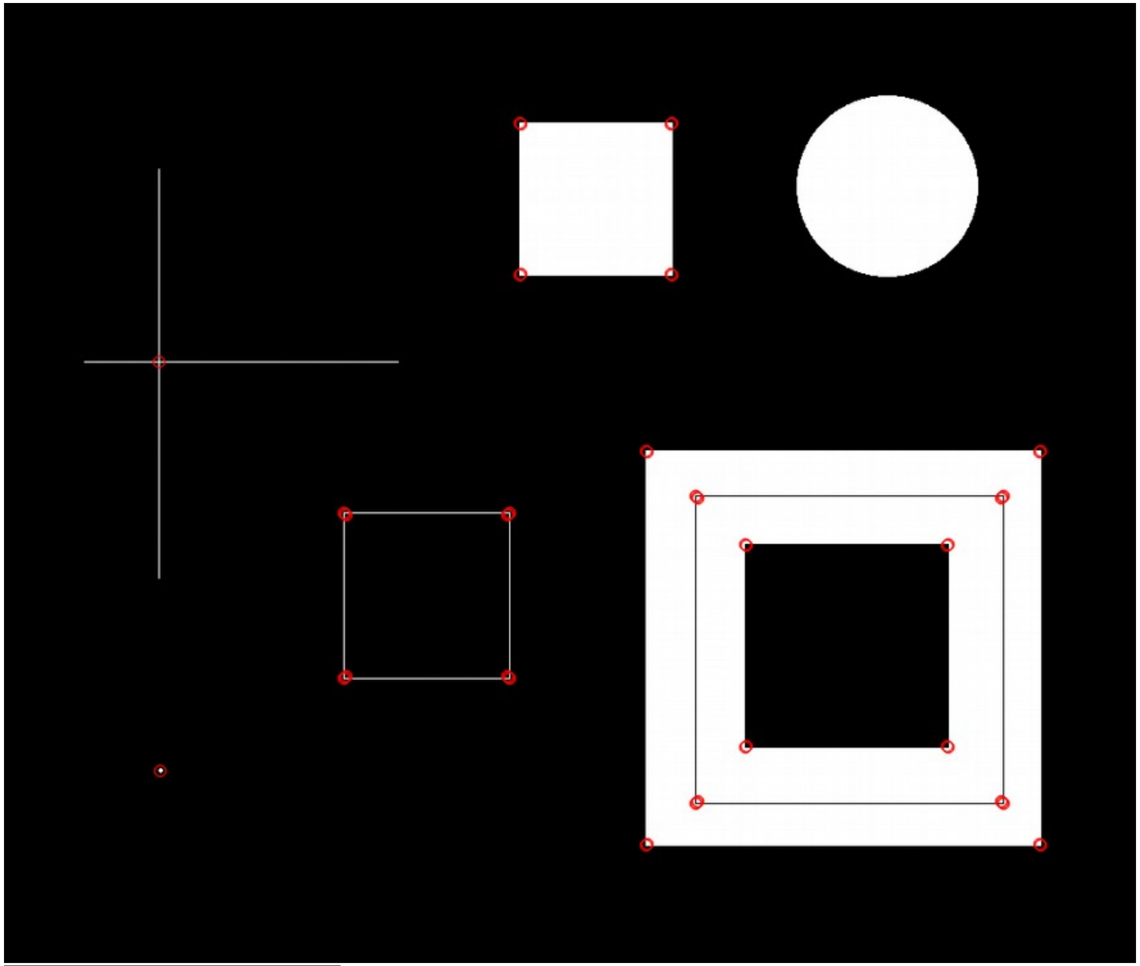


1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)
4. Trace of structure tensor
5. Determinant of structure tensor
6. weight calculation
7. isotropy calculation

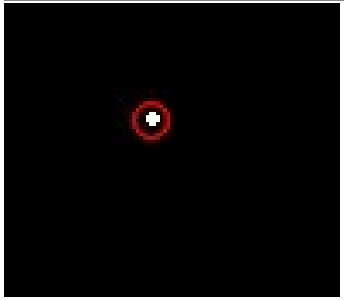


Isotropy q

Förstner Operator



1. Gradient in x- and y direction
2. $g_x \cdot g_x, g_y \cdot g_y, g_x \cdot g_y$
3. Average (Gaussian Window)
4. Trace of structure tensor
5. Determinant of structure tensor
6. weight calculation
7. isotropy calculation
8. Keypoint extration
   → weight > weight threshold
   → isotropy > isotropy threshold
   → weight is local maximum



Keypoints

5. Exercise – Given

```
int main(int argc, char** argv)
```

- Loads image, extracts and shows keypoints
- argv[1] == path to image
- argv[2] == scale of kernel (std-dev) for directional gradients
- argv[3] == scale of kernel (std-dev) for neighborhood

```
unsigned getOddKernelSizeForSigma(float sigma)
```

`sigma` Standard deviation
`return` Kernel size to use (always odd)

- Institutionally mandated "correct" kernel size
- Makes unit testing easier for me

```
bool isLocalMaximum(const cv::Mat_<float>& img, int x, int y)
```

`img` input image
x,y pixel location. Note: x == col, y == row
`return` `true` if value at (x,y) in img is locally maximal

- Checks if all neighbors are smaller

5. Exercise – ToDo

```
cv::Mat_<float> createGaussianKernel1D(float sigma)
```

`sigma` std-dev of filter kernel
`return` 1D gaussian kernel (horizontal layout)

- Computes 1D Gaussian kernel for separable convolutions
- Compute kernel size using **`getOddKernelSizeForSigma`**
- Copy/Adapt from previous homework

```
Mat separableFilter(Mat& src, Mat& kernelX, Mat& kernelY)
```

`src` Image to filter
kernelX 1D kernel to apply horizontally (kernel in horizontal layout)
kernelY 1D kernel to apply vertically (kernel in horizontal layout)
`return` Filtered image (same size)

- Computes separable convolution
- Note that different kernels can be used for horizontal and vertical passes
- Copy/Adapt from previous homework

5. Exercise – ToDo

```
cv::Mat_<float> createFstDevKernel1D(float sigma)
```

`sigma` std-dev of filter kernel (first derivative of Gaussian)
`return` the created kernel

- Generates kernel that corresponds to the first derivative of a Gaussian

$$G_x(x) = \frac{\partial}{\partial x} G(x;\sigma) = \frac{-x}{2\pi\sigma^4} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

```
void calculateDirectionalGradients(cv::Mat& img, float sigmaGrad,
                     cv::Mat_<float>& gradX, cv::Mat_<float>& gradY)
```

```
img       input image
sigmaGrad std-dev of Gaussians
gradX     Output matrix for x-components of per pixel gradients
gradY     Output matrix for y-components of per pixel gradients
```

- Computes directional gradients via **separable** convolution
- For x-component: Convolve horizontally with derivative of Gaussian and vertically with normal Gaussian
- For y-component: The other way around

5. Exercise – ToDo

```
void calculateStructureTensor(Mat& gradX, Mat& gradY, float sigma,
                              Mat& A00, Mat& A01, Mat& A11)
```

`gradX,gradY`	Input directional gradients
`sigma`	Std-dev for the Gaussian blur to compute the neighborhood sum.
`A00,A01,A11`	Output per pixel structure tensor matrix

- Computes the structure Tensor for each pixel
- Neighborhood summation through convolution with Gaussian kernel
- Output tensor matrix elements as separate matrices

$$\mathbf{A} = \sum_W \mathbf{g}\mathbf{g}^T = \begin{pmatrix} \sum_W g_x^2 & \sum_W g_x g_y \\ \sum_W g_y g_x & \sum_W g_y^2 \end{pmatrix}$$

Use Gaussian blur instead of plain summation

5. Exercise – ToDo

```
void calculateFoerstnerWeightIsotropy(Mat& A00, Mat& A01, Mat& A11,
                                      Mat& weight, Mat& isotropy)
```

`A00,A01,A11`	`Input` per pixel structure tensor matrices
`weight`	`Output` per pixel "Förstner weight"
`isotropy`	`Output` per pixel "Förstner isotropy"

- Computes per pixel the weight and isotropy
- Prevent division by zero:
  - […] / std::max(trace, 1e-8f)
  - […] / std::max(trace * trace, 1e-8f)

$$w = \frac{1}{\mathrm{tr}\left(\mathbf{A}^{-1}\right)} = \frac{1}{\lambda_1 + \lambda_2} = \frac{\det(\mathbf{A})}{\mathrm{tr}(\mathbf{A})} \qquad w > 0$$

$$q = 1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 = \frac{4\det\left(\mathbf{A}\right)}{\mathrm{tr}\left(\mathbf{A}\right)^2} \qquad 0 \leq q \leq 1$$

5. Exercise – ToDo

```
vector<Vec2i> getFoerstnerInterestPoints(Mat& img,
                                        float sigmaGrad, float sigmaNeighborhood,
                                        float minWeight, float minIsotropy)
```

`img`	input image
`sigmaGrad`	std-dev of filter kernels for directional gradients
`sigmaNeighborhood`	std-dev of filter kernel for neighborhood summation
`minWeight`	Minimum weight of interest points as fraction of average weights
`minIsotropy`	Minimum isotropy of interest points
`return`	found keypoint locations **(column, row)**

- Computes directional gradients, structure tensors, weights and isotropies
- Extracts pixel locations where:
  - weight is larger than computed weight threshold
  - isotropy is larger than minIsotropy
  - weight is local maximum
- Use isLocalMaximum(…) to check if weight is local maximum

$$w_{min} = 0.5, \ldots, 1.5 \cdot \bar{w}, \bar{w} \text{ is the mean of w over whole image}$$

6. Exercise – Testcode

main.cpp contains a piece of test code for exercise 6:

```
#define TEST_FOR_DIP6

#ifdef TEST_FOR_DIP6
#include <immintrin.h>
void testForDip6() {
    ..................
}
#else
void testForDip6() {
}
#endif
```

This checks if stuff that we will need for exercise 6 works on your PC. If this fails during compilation or runtime, comment out "`#define TEST_FOR_DIP6`" and write me an email.

Next Meeting

Deadline: 21.01.2020

Next Meeting: 21.01.2019

No theory questions but do remember to include input/output images in submission.