Bauhaus-Universität Weimar

Image Analysis and Object Recognition

Assignment 2

Image Filtering and
Interest Points

SS 2017

(Course notes for internal use only!)

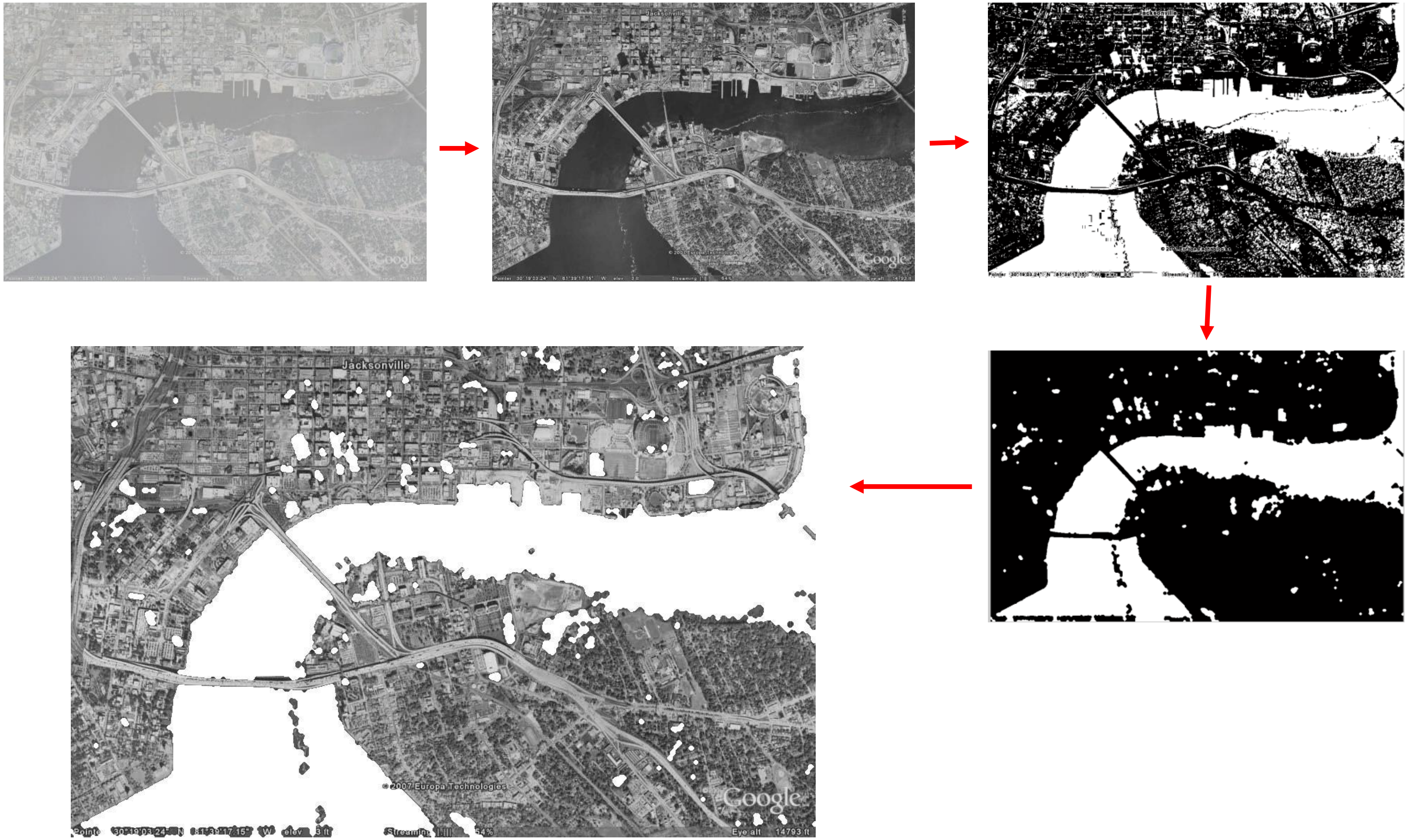
Assignment 1

Topic: basic information extraction

- Extract “regions of interest” from an image
  - Getting familiar with MATLAB
  - Image enhancement → histogram stretching
  - Global thresholding → derive a binary image
  - Morphological operators → dilation and erosion, opening and closing

Assignment 1 (A-C)

Workflow:



Assignment 1

A: Image enhancement

- Computing a grayscale image *GI* from rgb *image*:
  - `GI = mean(image, 3); % equal weights` → ***preferred***
  - `GI = rgb2gray(image); % unequal weights`

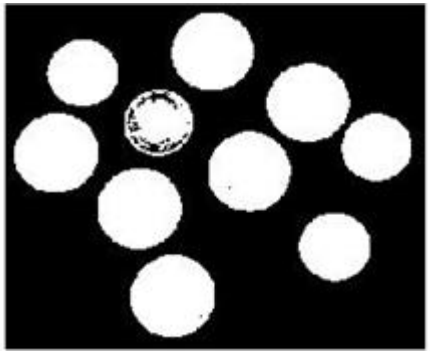
- Get the maximum value of an 2d-array:
  - `Maxi = max(max(GI));`
  - `Maxi = max(GI(:));`

- Histogram stretching:
  - `SI = (GI - Mini)/(Maxi-Mini);`

  → For-loops not necessary

Assignment 1

B: Global Thresholding



- Finding a threshold: trial and error *or* use function `graythresh`
- Apply threshold: using operators "`<, >, <=,` …" or function `im2bw`

```
mask = image < threshold;
```

→ For-loops not necessary

Assignment 1

C: Morphological filtering: Erosion and Dilation

```
% result arrays
result_erode = mask*0; result_dilate = mask*0;

% array with structuring element
radius_se = 4; se = strel('disk', radius_se)
filtering_array = getnhood(se);
size_image = size(mask);

% erosion and dilation
for i = radius_se+1:(size_image(1)-radius_se)
    for j = radius_se+1:(size_image(2)-radius_se)

        % get the current mask chip which is covered by the se
        mask_chip = mask( (i-radius_se):(i+radius_se), (j-radius_se):(j+radius_se) );

        % derive product (element-wise) of chip and se
        prod = mask_chip .* filtering_array;

        % erosion (AND)
        if sum(sum(prod)) == sum(filtering_array(:))
            result_erode(i,j) = 1;
        end

        % dilation (OR)
        if sum(sum(prod)) >= 1
            result_dilate(i,j) = 1;
        end
    end %j
end %i
```

```
filtering_array =

     0     0     1     1     1     1     1     0     0
     0     1     1     1     1     1     1     1     0
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     0     1     1     1     1     1     1     1     0
     0     0     1     1     1     1     1     0     0
```

Assignment 1

- Reference pixel is always the center pixel of the mask!
- Wrong implementation:

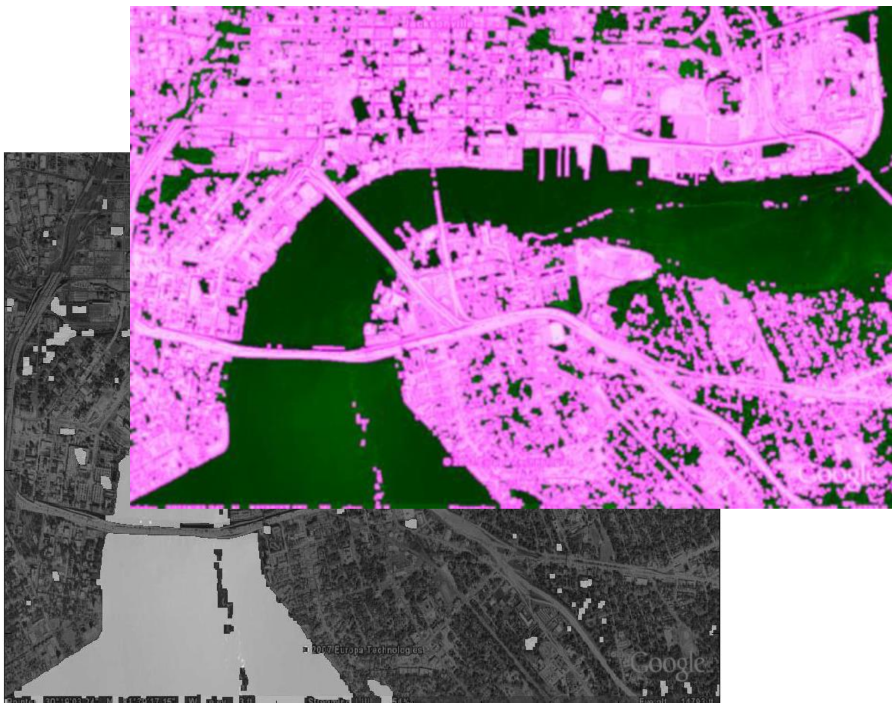
```
function dilation = dilation(img)

    % initialize empty matrix with the size of the image
    custom_img = false(size(img));
    width = 5;
    field = getnhood(strel('square',width));
    m = floor(size(field,1)/2);
    n = floor(size(field,2)/2);
    array = padarray(img,[m,n]);

    for x = 1:size(array,1)-(2*m)
        for y = 1:size(array,2)-(2*n)
            temp_img = array(x:x+(2*m), y:y+(2*n));
            custom_img(x,y) = max(max(temp_img&field));
        end
    end
```

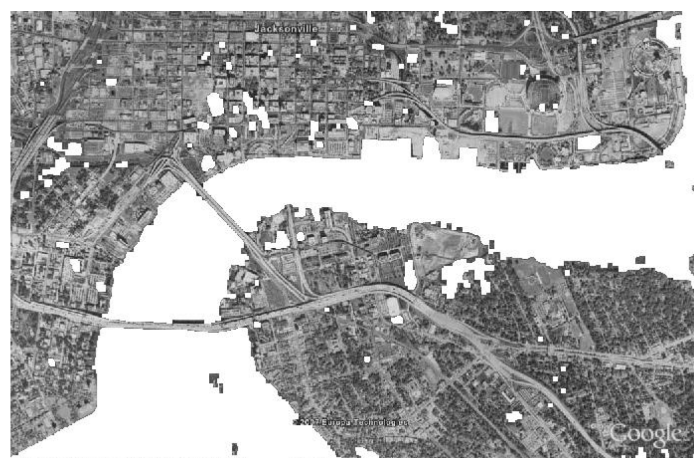
```
filtering_array =

     0     0     1     1     1     1     1     0     0
     0     1     1     1     1     1     1     1     0
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1
     0     1     1     1     1     1     1     1     0
     0     0     1     1     1     1     1     0     0
```

Assignment 1: some results

C: Morphological filtering

- Application of opening and closing subsequently
- Watermask was desired





Matlab function `imfuse`

Summary: Binary image processing

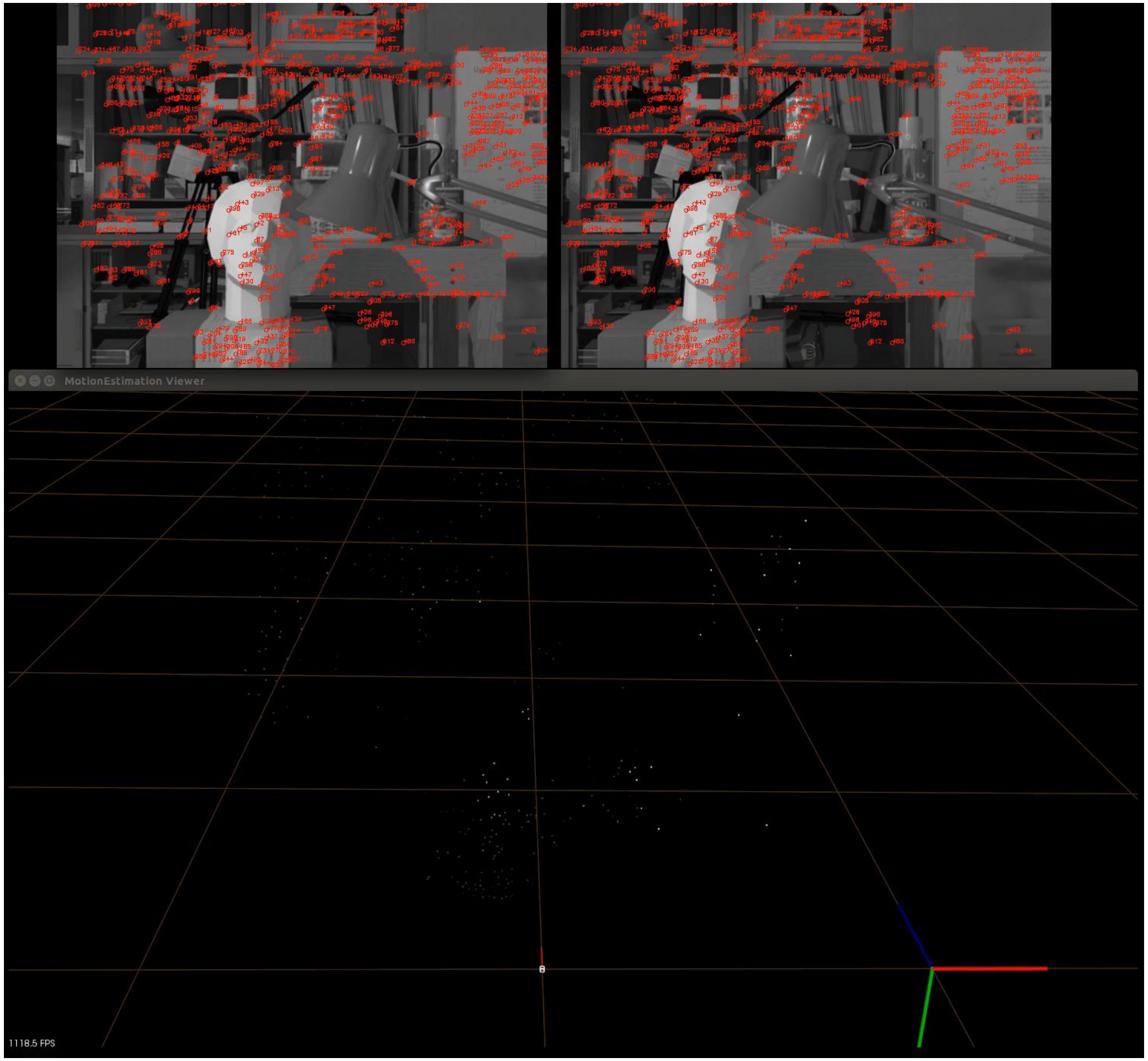
- Pro’s:
  - Easy techniques and fast to compute
  - Binary images are easy to store
  - Can be useful in constrained scenarios with well known conditions
- Con’s:
  - Hard to extract the “clean” object silhouettes
  - Influence of noise
  - Not suitable for more complex problems

Assignment 2

- **Task A:** Image-filtering (GoG)
- **Task B:** Interest points (Förstner)
- Aims
  - Learn how to do image filtering
  - Deriving edge information (intensity changes)
  - Reducing noise and deriving edge information **simultaneously** using GoG-filtering
  - Using edge information to identify “points of interest” in images
- Relevant for
  - Understanding filtering
  - Edge detection and image smoothing
  - Finding corresponding points in images

Assignment 2

Corresponding points for stereo visual odometry



Other Applications:

- Image Stitching
- Camera Calibration
- 3D-Reconstruction
- Object Detection
- Object Tracking
- …..

Assignment 2

Task A: Gradient of Gaussian Image-filtering

Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)

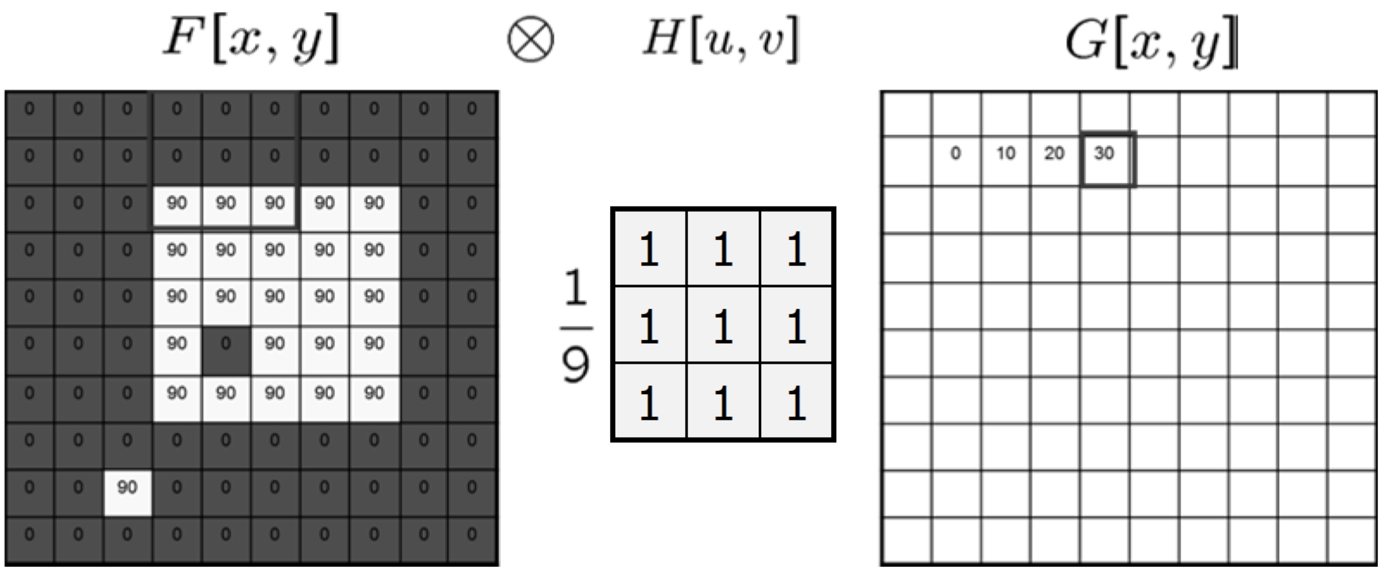


Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)

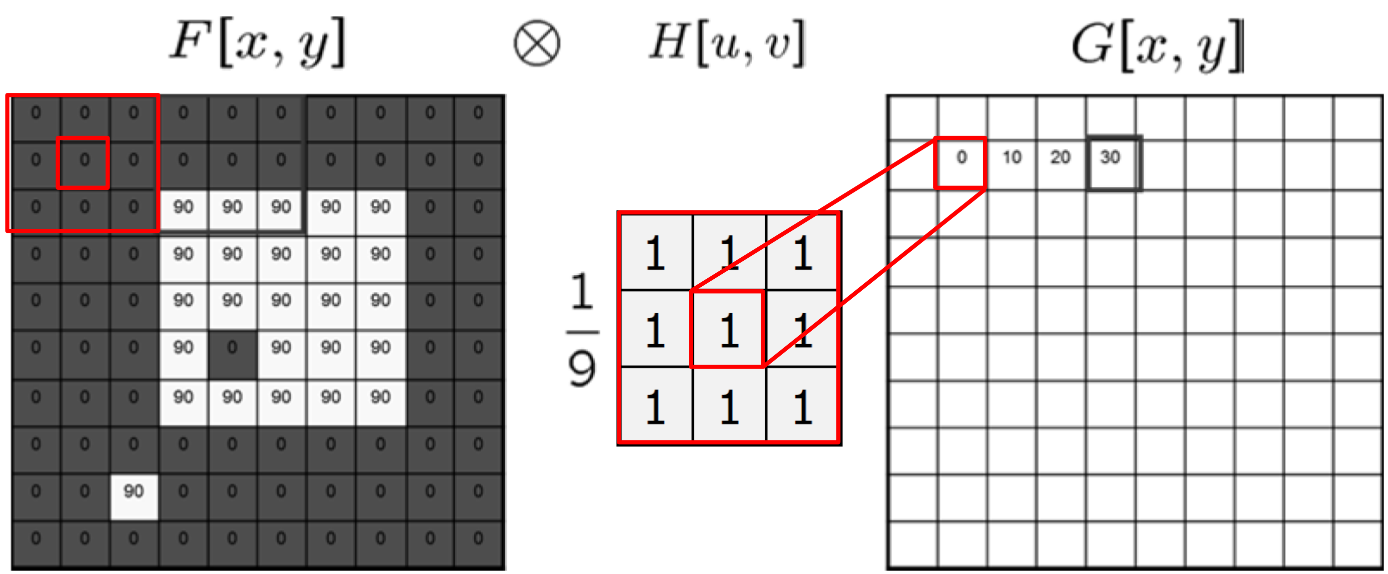


Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)

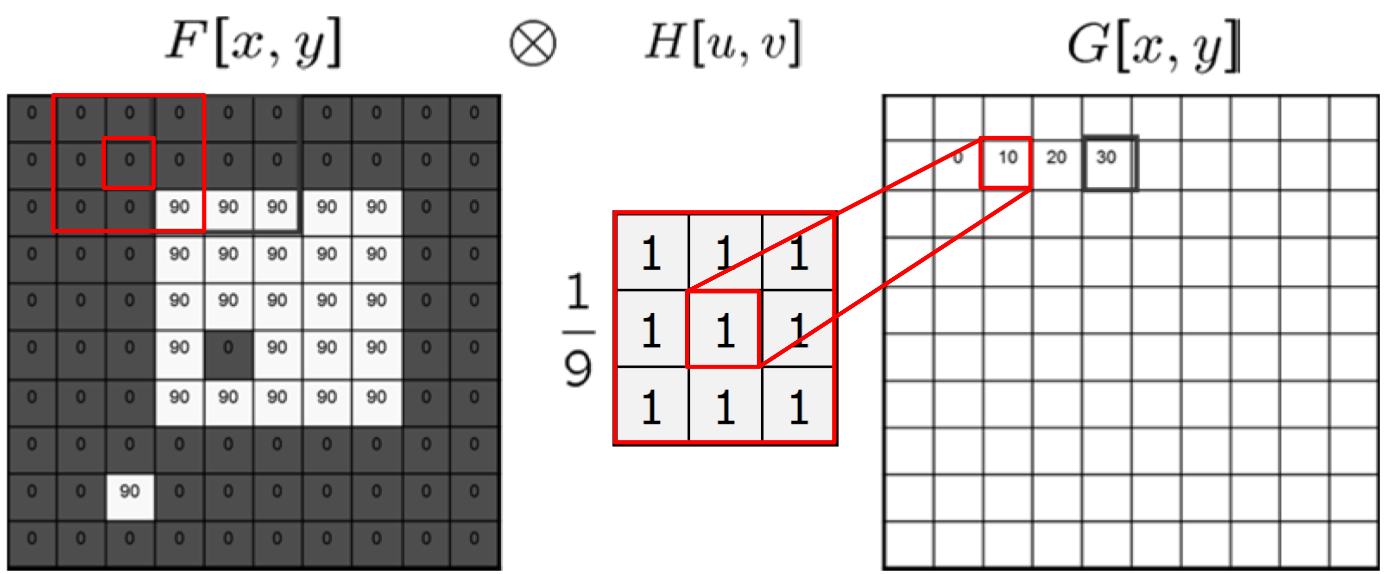


Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)

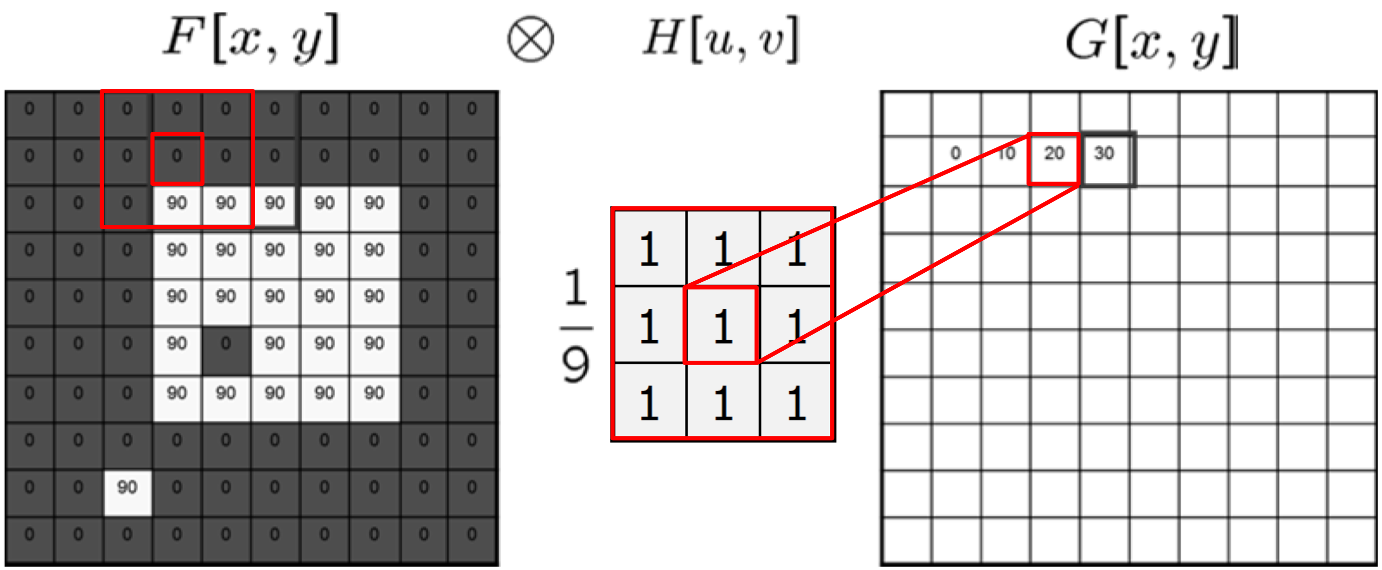
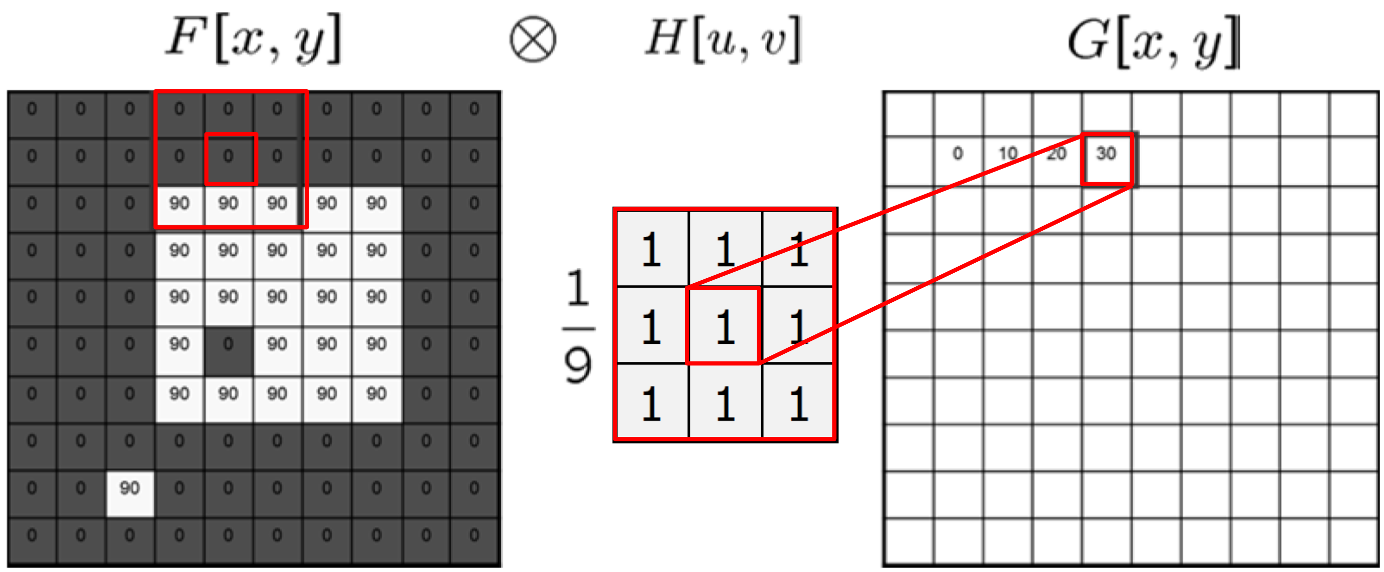


Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)

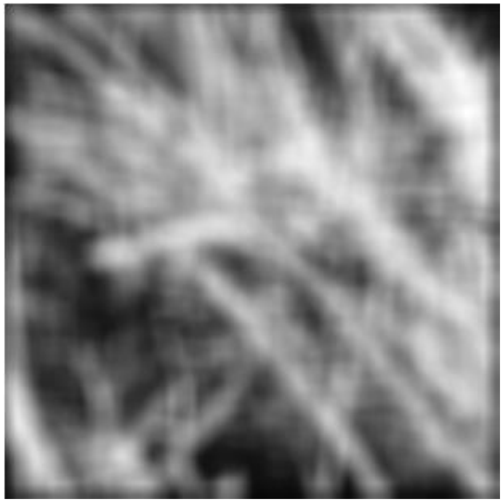


$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]\, F[i+u, j+v] \;\rightarrow\; k = 1$$

$G = H \otimes F$ → **Cross-correlation**

Image Filtering

- Replace each pixel with a linear combination of its neighbors
  - Filter Mask H : contains weights for the linear combination
  - Example: Moving average (image smoothing)



$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]\, F[i+u, j+v]$ → $k = 1$

$G = H \otimes F$ → **Cross-correlation**

Image Filtering

- Replace each pixel with a linear combination of its neighbors
- Filter kernel H: coefficients or weights

- **Cross-correlation:**

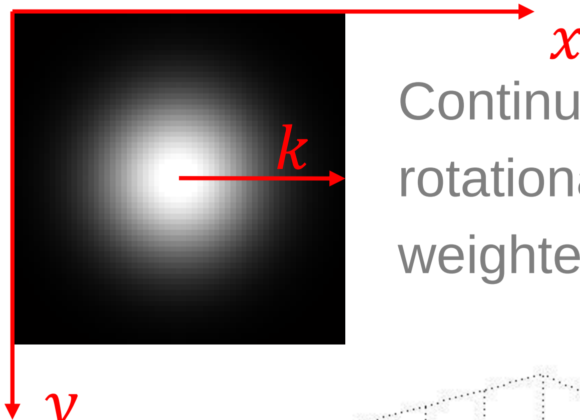
  $$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] \boxed{F[i+u,j+v]}$$
  $$G = H \otimes F$$

  - Check similarity of two signals
- **Convolution:**

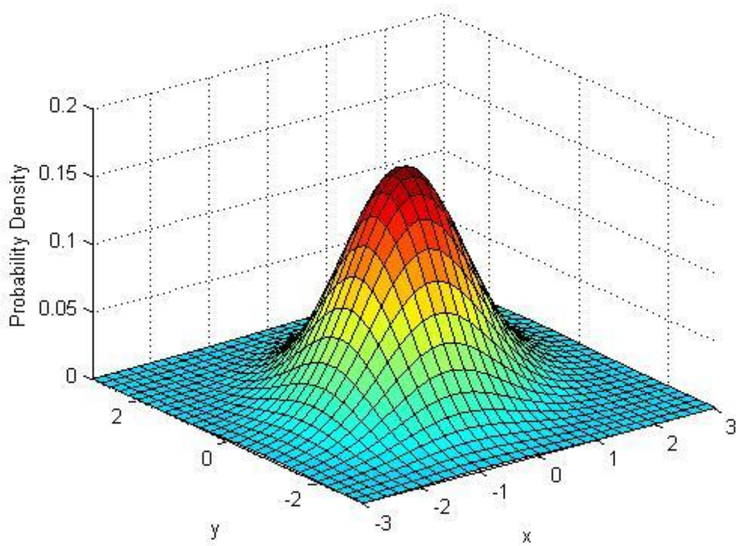
  $$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] \boxed{F[i-u,j-v]}$$
  $$G = H \star F$$

  - Apply filter H on image F for: information extract for processing tasks
  - Can easily be done in frequency-domain
  - Associative (independent of application sequence)
  - Symmetric filter kernel → Correlation = Convolution

2D Gaussian Filter

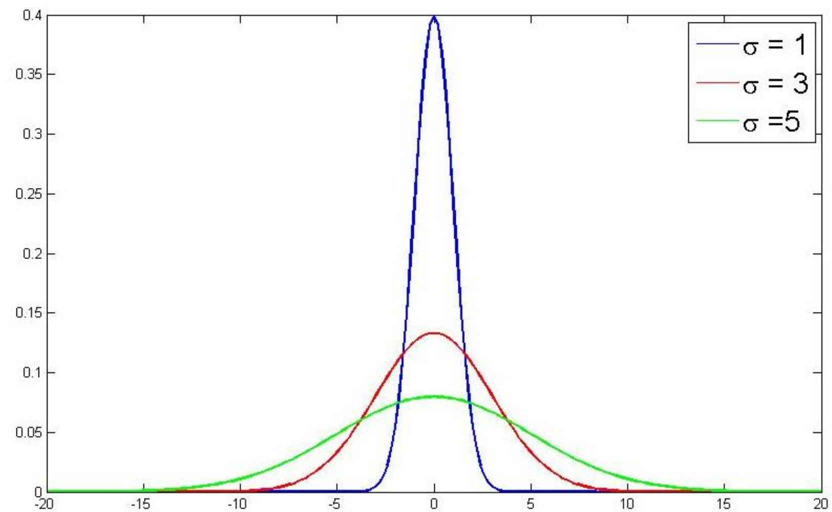


Continuous,
rotationally symmetric
weighted average



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

Effect of standard deviation σ



Mask radius $k = 2\sqrt{2}\sigma \approx |3\sigma|$

2D Gaussian Filter

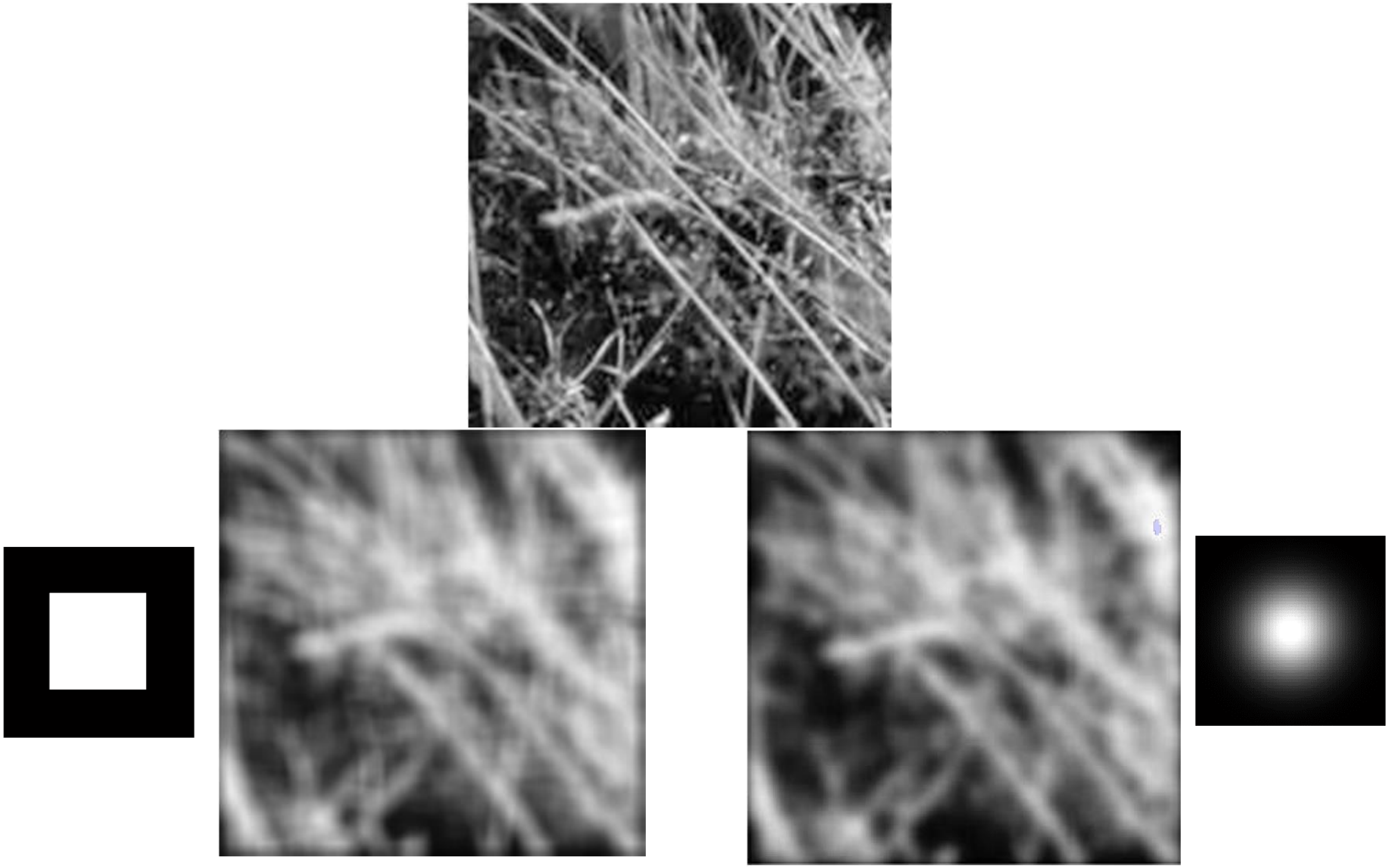
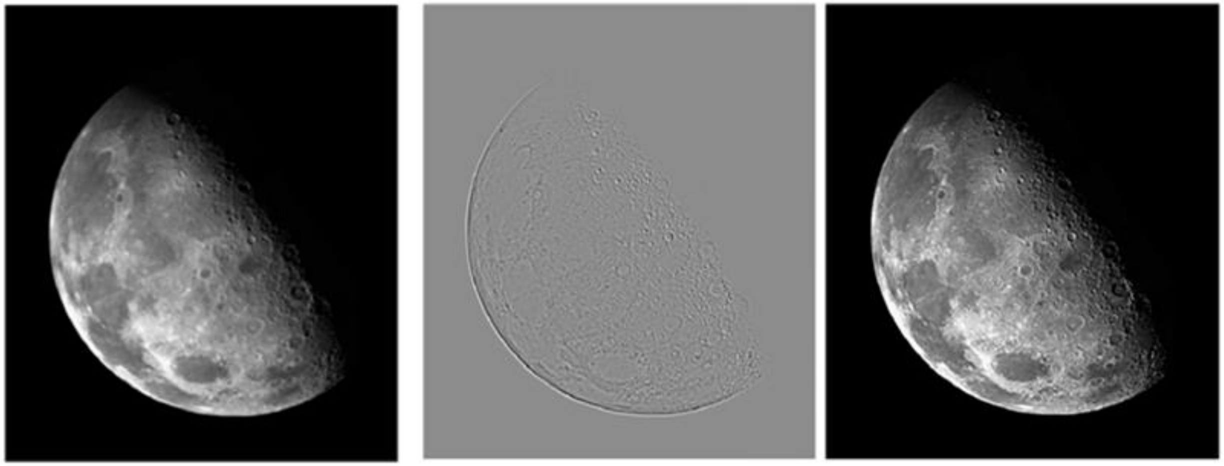


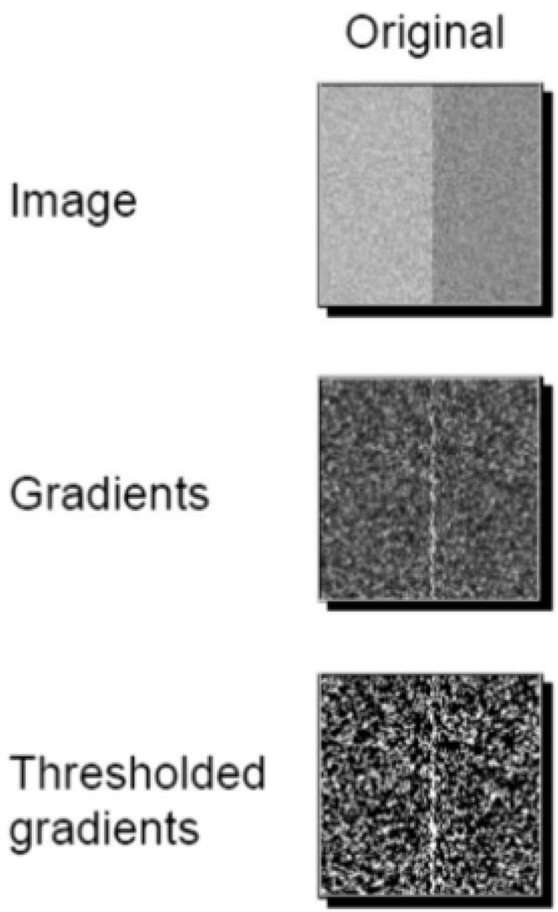
Image Sharpening

- Mean and Gaussian filter
  - Remove high-frequency components from images
  - Low-pass filter
- Smoothing → integration
- Sharpening → differentiation
  - Edge detection
  - Image enhancement



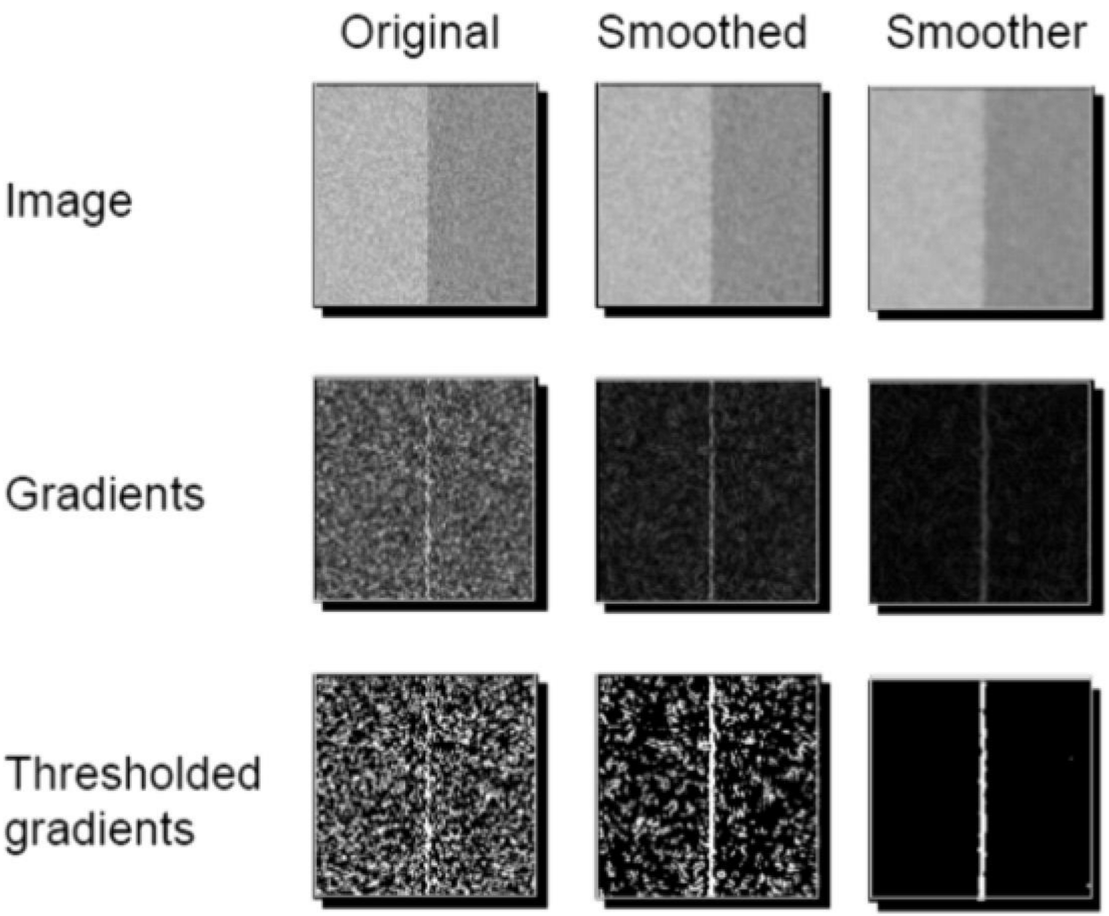
Sharpening and Smoothing

Benefits of smoothing in edge detection



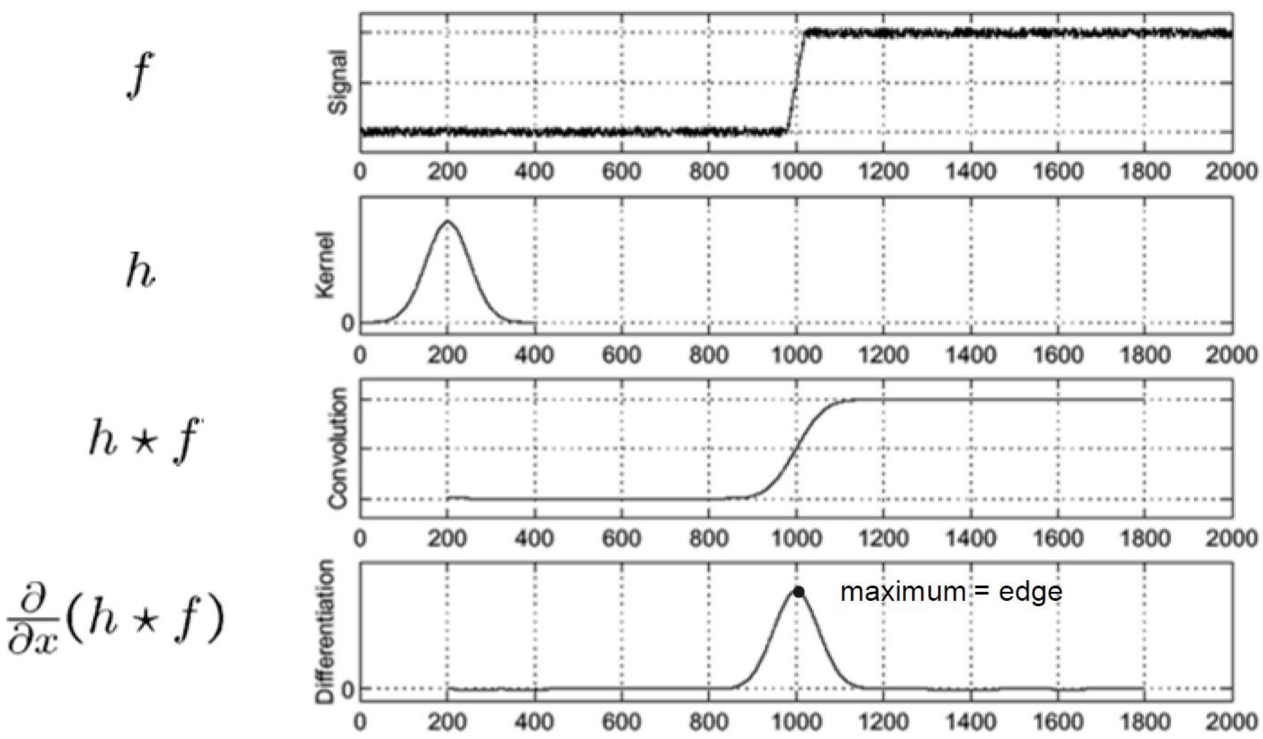
Sharpening and Smoothing

Benefits of smoothing in edge detection



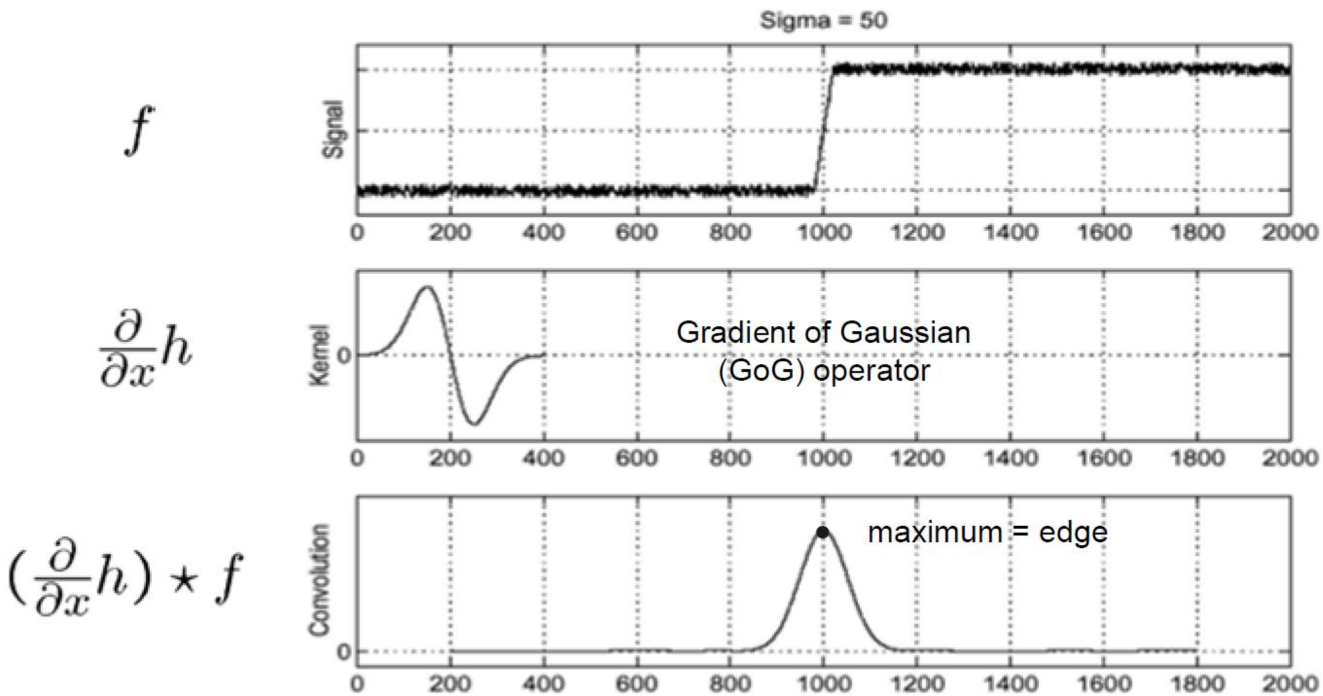
Sharpening and Smoothing

- Smoothing before computing the differentiation
  → Two independent filter operations (convolutions)



Sharpening and Smoothing

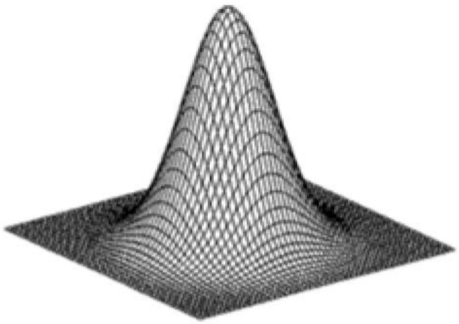
- Differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = \left(\frac{\partial h}{\partial x}\right) \star f$

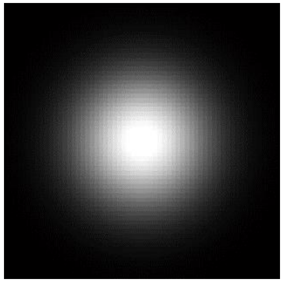


2D GoG filtering

- Gaussian filter

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

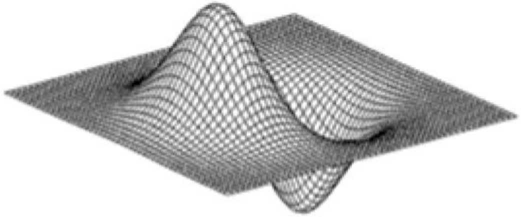


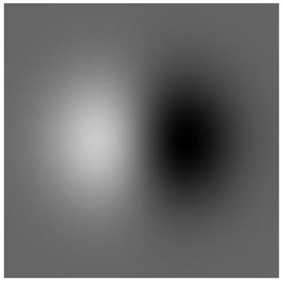


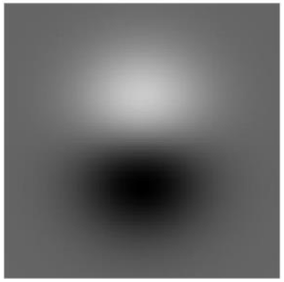
- Gradient of Gaussian

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

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





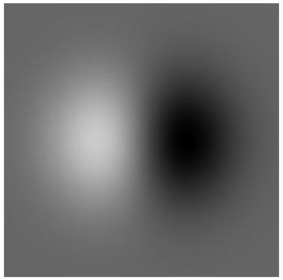


2D GoG filter computation

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

1) Define standard deviation, e.g. $\sigma = 0.5$
2) "Size" of filter kernel from center pixel: $r = |3 \cdot \sigma| = 2.0$
3) Define 2 Arrays c_x and c_y with $(r \cdot 2 + 1)$ columns and rows for local coordinates

$$c_x = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}; \quad c_y = c_x^T$$



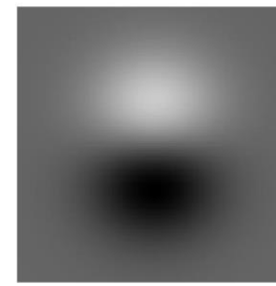
4) Compute filter using c_x and c_y for x and y

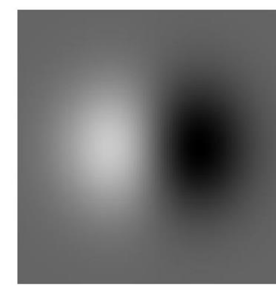
$$G_x = \frac{\partial G(x,y,\sigma)}{\partial x} = \begin{bmatrix} 0.0000 & 0.0001 & 0.0000 & -0.0001 & -0.0000 \\ 0.0002 & 0.0466 & 0.0000 & -0.0466 & -0.0002 \\ 0.0017 & 0.3446 & 0.0000 & -0.3446 & -0.0017 \\ 0.0002 & 0.0466 & 0.0000 & -0.0466 & -0.0002 \\ 0.0000 & 0.0001 & 0.0000 & -0.0001 & -0.0000 \end{bmatrix}; \quad G_y = \frac{\partial G(x,y,\sigma)}{\partial y} = \frac{\partial G(x,y,\sigma)}{\partial x}^T$$

Task A: GoG filtering

Input image:

Compute grayscale image and scale it to double [0,…,1] (`mean, mat2gray`).





a. Compute GoG-filter masks for filtering in x- and y- direction
b. Apply the two filters $G_x$and G_y on the input image using *for-loops* → Convolution, result: I_x and I_y
c. Compute the gradient magnitude image using equation

$$G = \sqrt{(I_x)^2 + (I_y)^2}$$

Plot and export the resulting image G (by-product!).

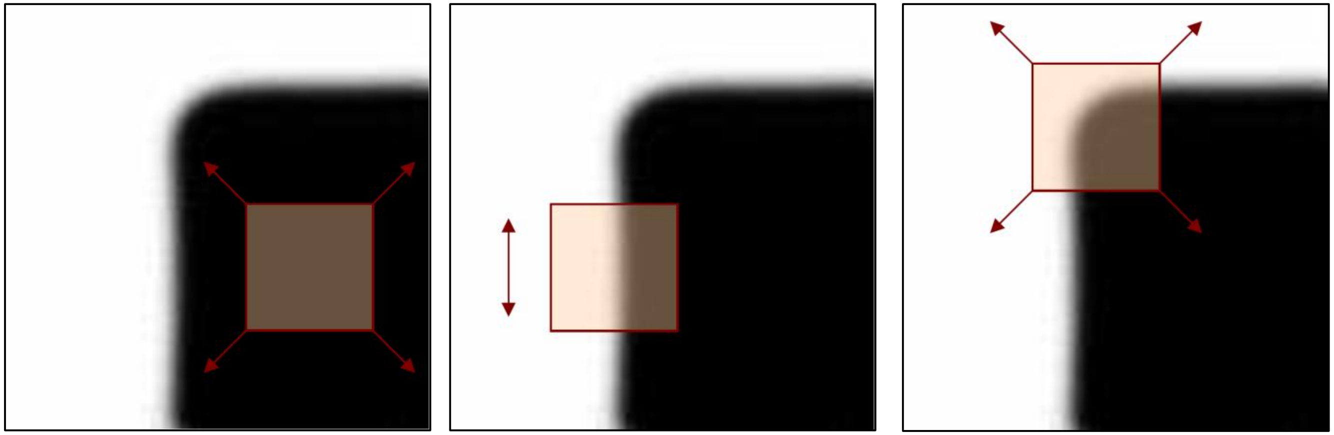
Bauhaus-Universität Weimar

Assignment 2

Task B: Förstner Operator

Corners as distinctive interest points

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity



Flat region:
no changes in all directions

Edge:
no change along edge direction

Corner:
significant change in all directions

Auto-correlation Matrix

- Identification of corners
- Input: First derivatives in x- and y-direction I_x and I_y (result of A.b.)

Grayscale image

I_x (GoG)

I_y (GoG)

Auto-correlation Matrix M

- 3 Input Arrays: I_x^2, I_y^2 and $I_x I_y$
- Computation of M for each pixel:

Definition: $w_N = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ → weights of local Neighborhood N

$$M = \sum_{x,y \in N} w_N(x,y) \cdot \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = w_N \star \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

→ $M = \sum_{x,y \in N} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ **… M contains the sum of all values of I_x^2, I_y^2 and $I_x I_y$ in the local neighborhood N**

Auto-correlation Matrix M

Do for each pixel in the image (except edges):

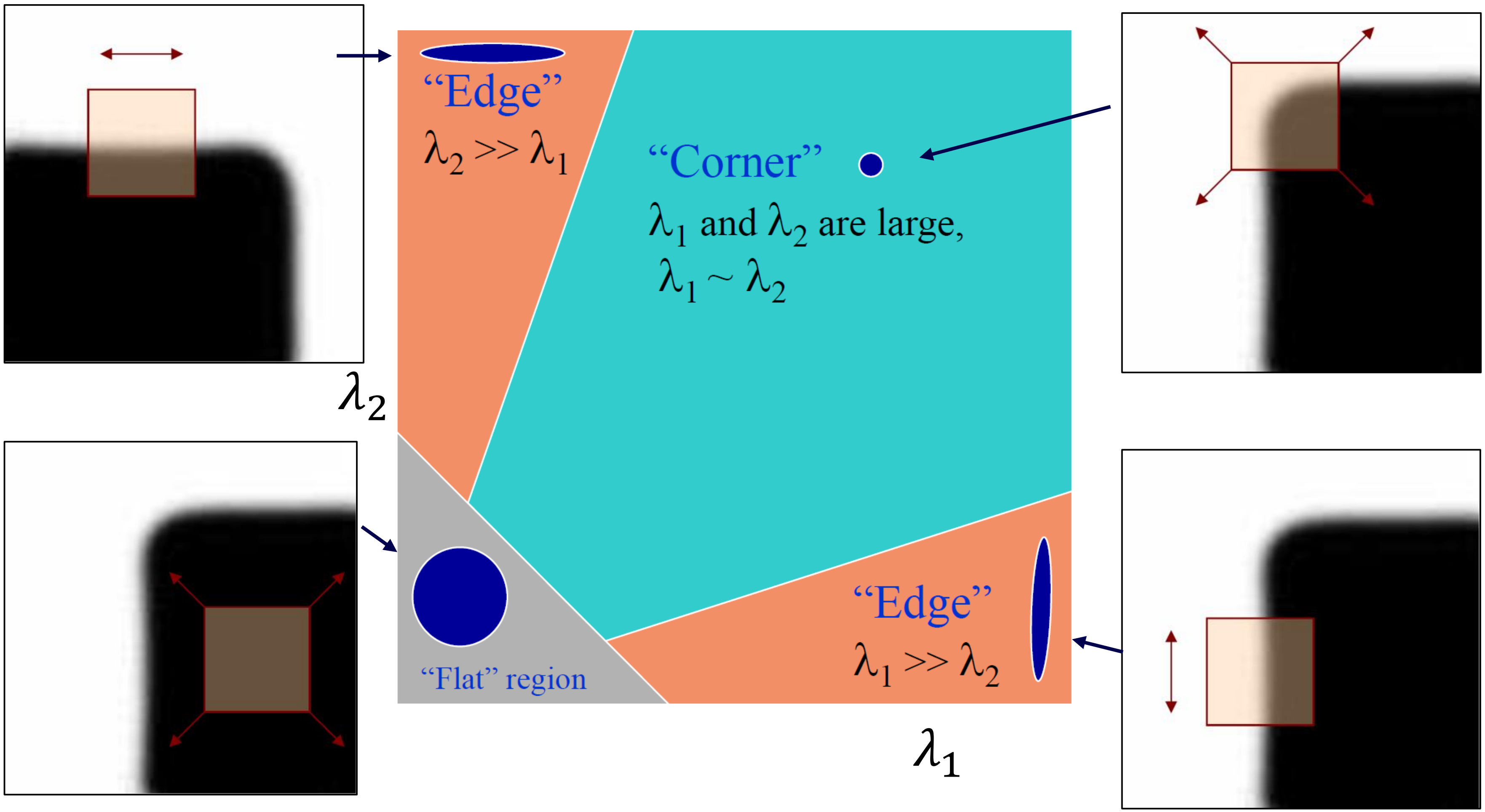
1) Extract local image chip (covered by w) from I_x^2, I_y^2 and $I_x I_y$
2) Compute M for each pixel:
   → summarize three local values I_x^2, I_y^2 and $I_x I_y$ in w_N
   → $\overline{I_x^2} = \sum_N I_x^2$, also for $\overline{I_y^2}$ and $\overline{I_x I_y}$
3) Build M

$$M = \begin{bmatrix} \overline{I_x^2} & \overline{I_x I_y} \\ \overline{I_x I_y} & \overline{I_y^2} \end{bmatrix}$$

Equal to: Convolve I_x^2, I_y^2 and $I_x I_y$ with w_N and then compute M for each pixel

Auto-correlation Matrix *M*

Use Eigenvalues of *M* to detect corners



Förstner Interest Operator

- Corneness:

$$w = \frac{trace(M)}{2} - \sqrt{\left(\frac{trace(M)}{2}\right)^2 - det(M)}, \qquad w > 0$$

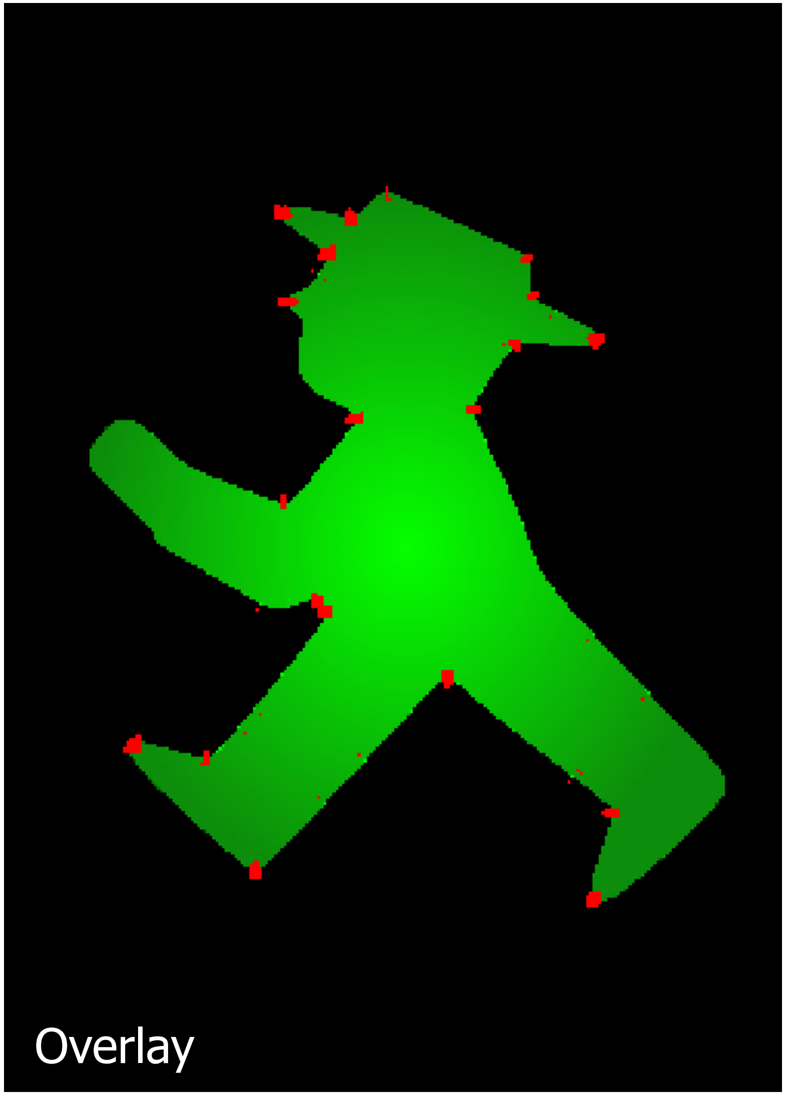
- Roundness

$$q = \frac{4 \cdot det(M)}{trace(M)^2}, \qquad 0 \leq q \leq 1$$

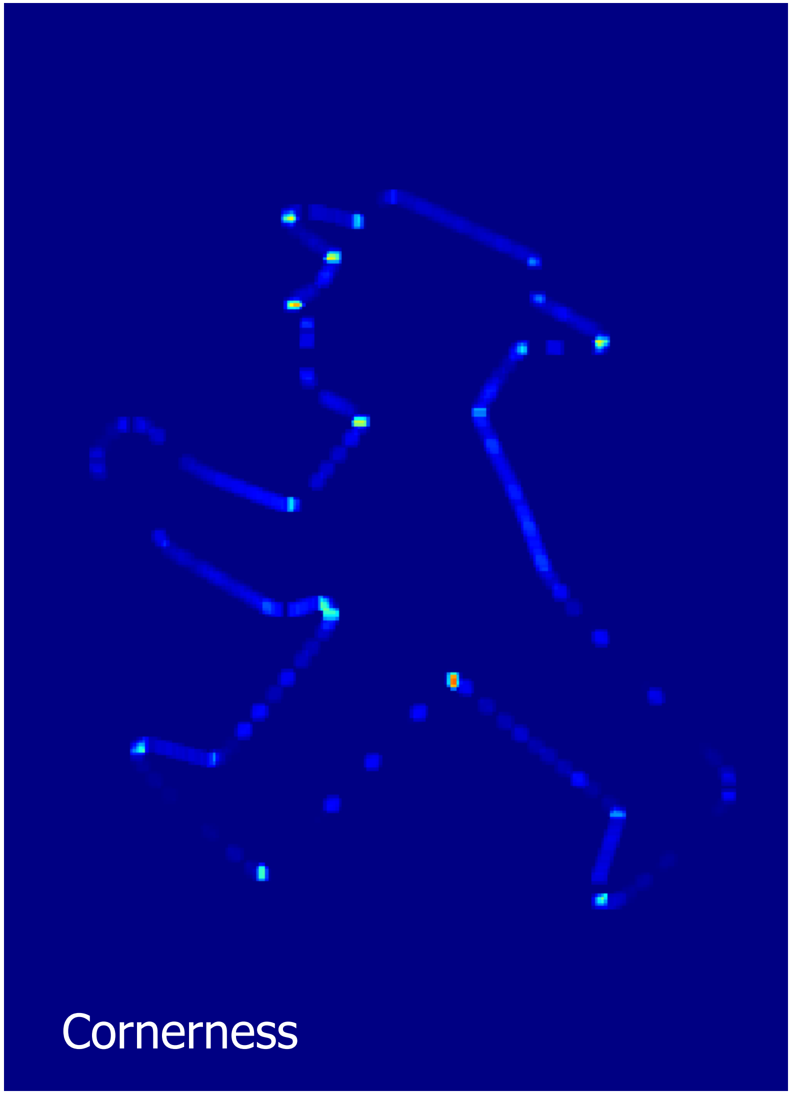
- Find corner point candidates M_C

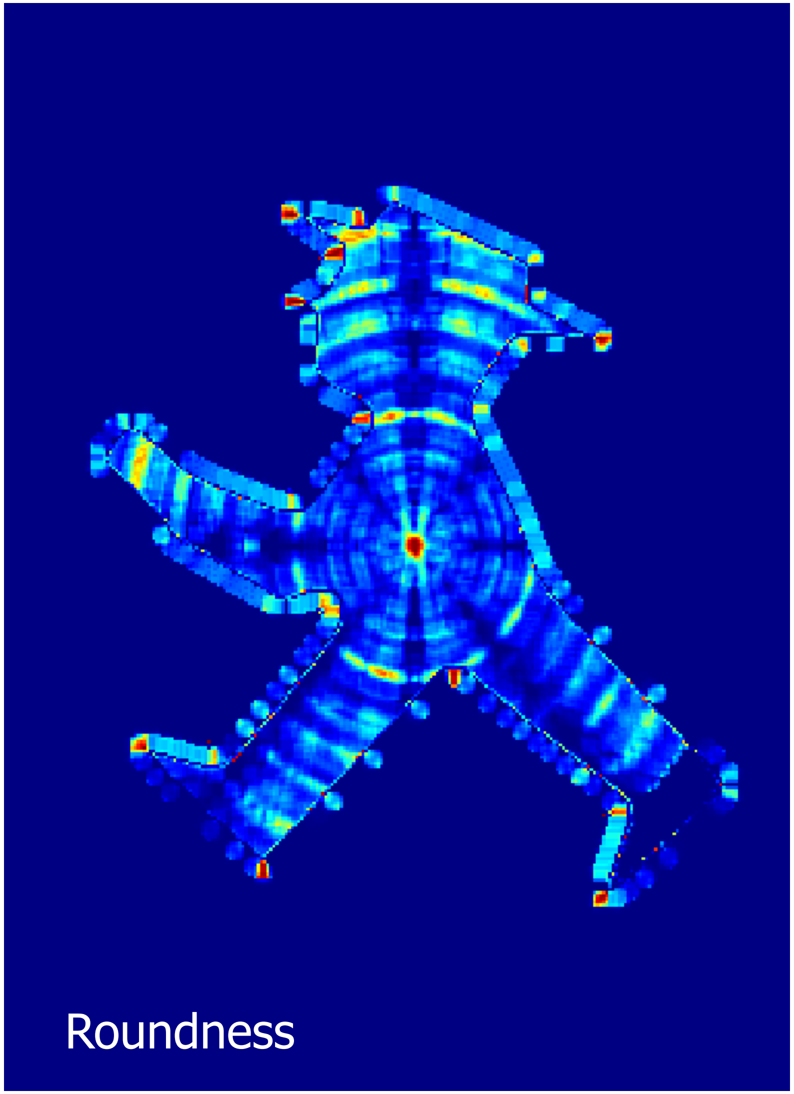
$$M_C = w > t_w \;\&\; q > t_q$$
$$t_w = [0.001, \dots, 0.01], t_q = [0.5, \dots, 0.75]$$

Overlay of original image and M_c

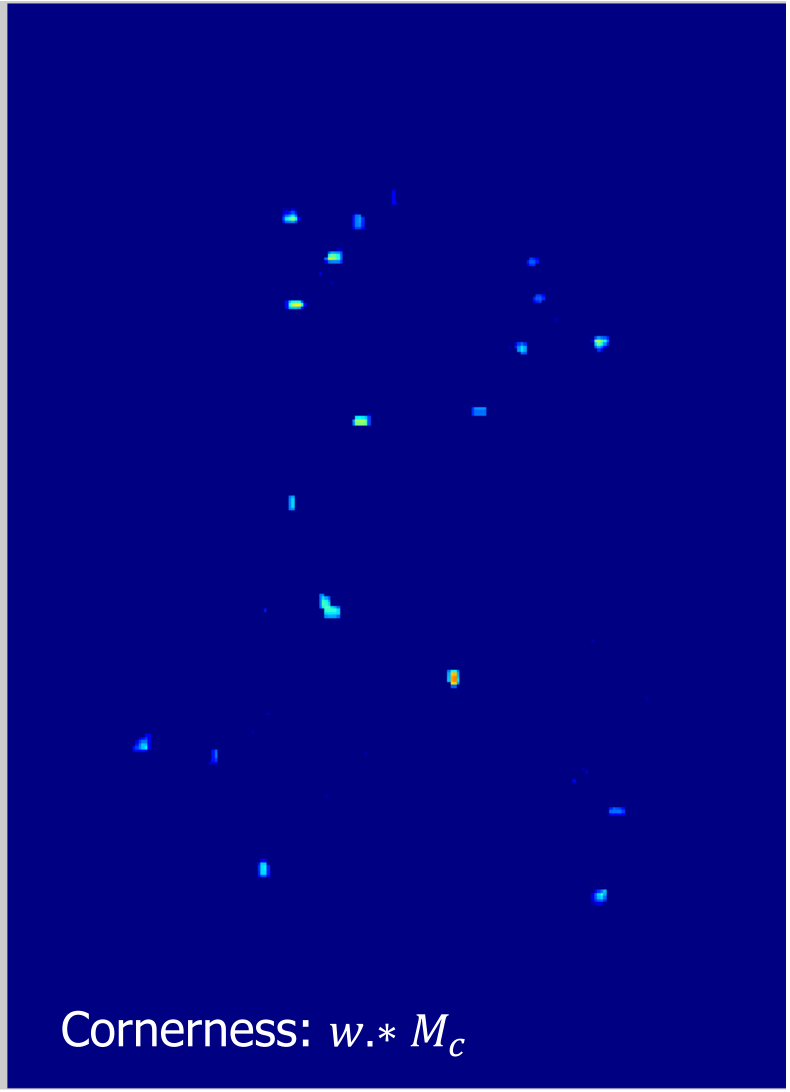


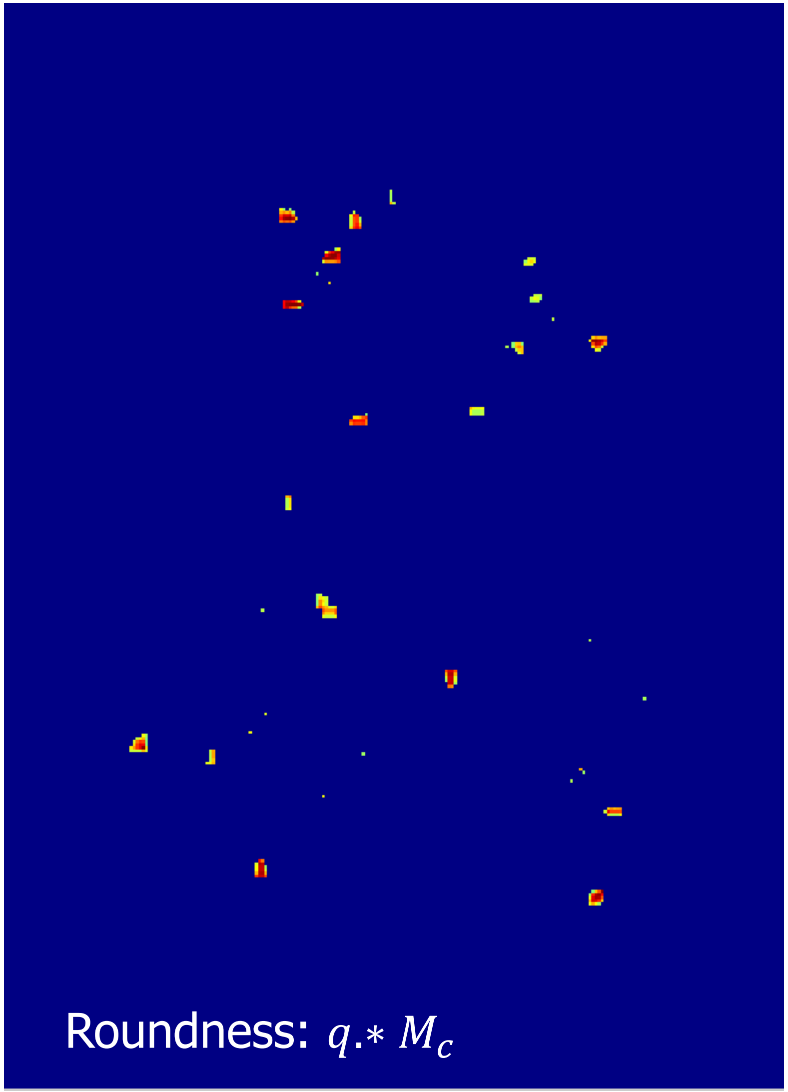
Thresholded regions of w and q



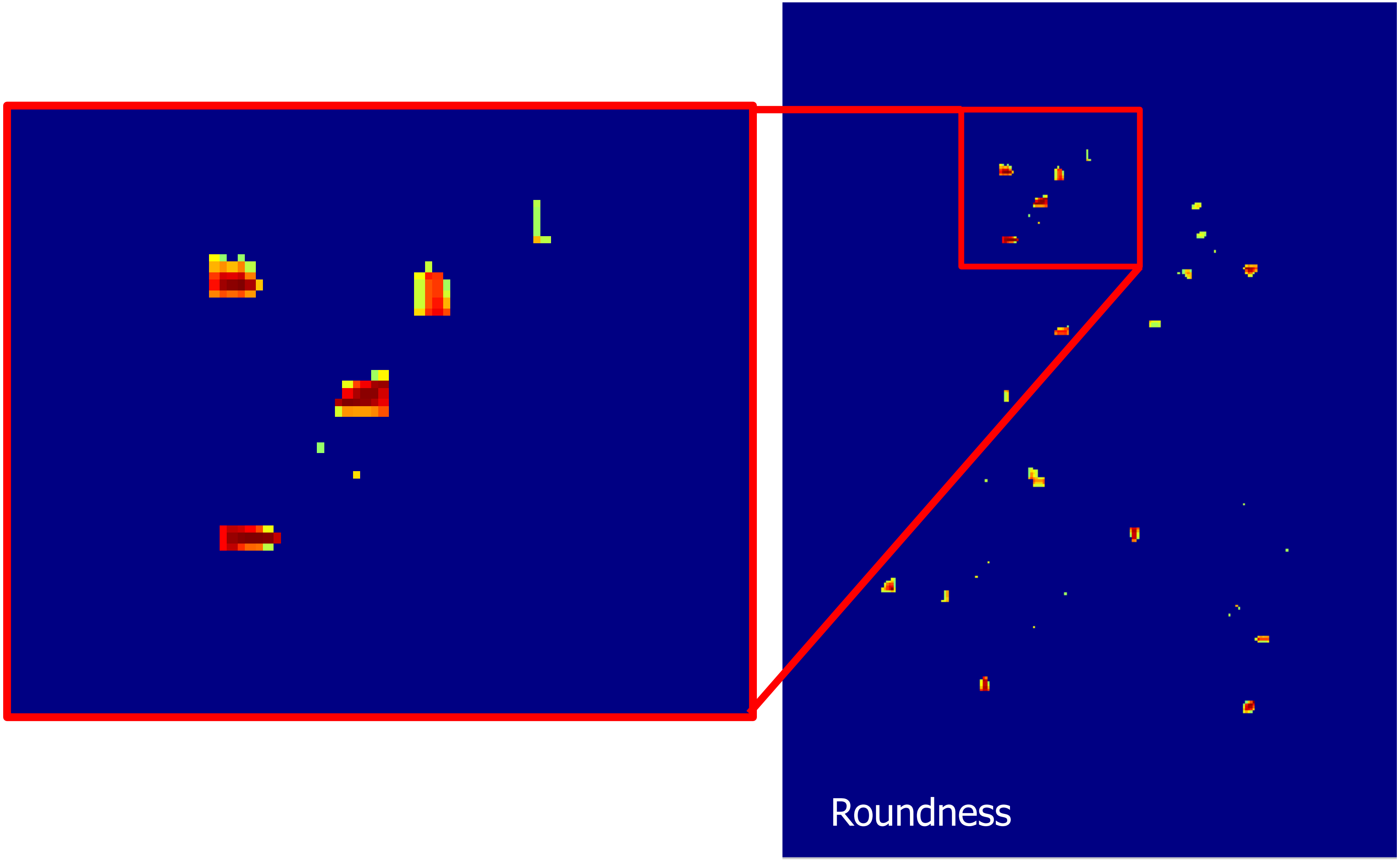


Thresholded regions of w and q



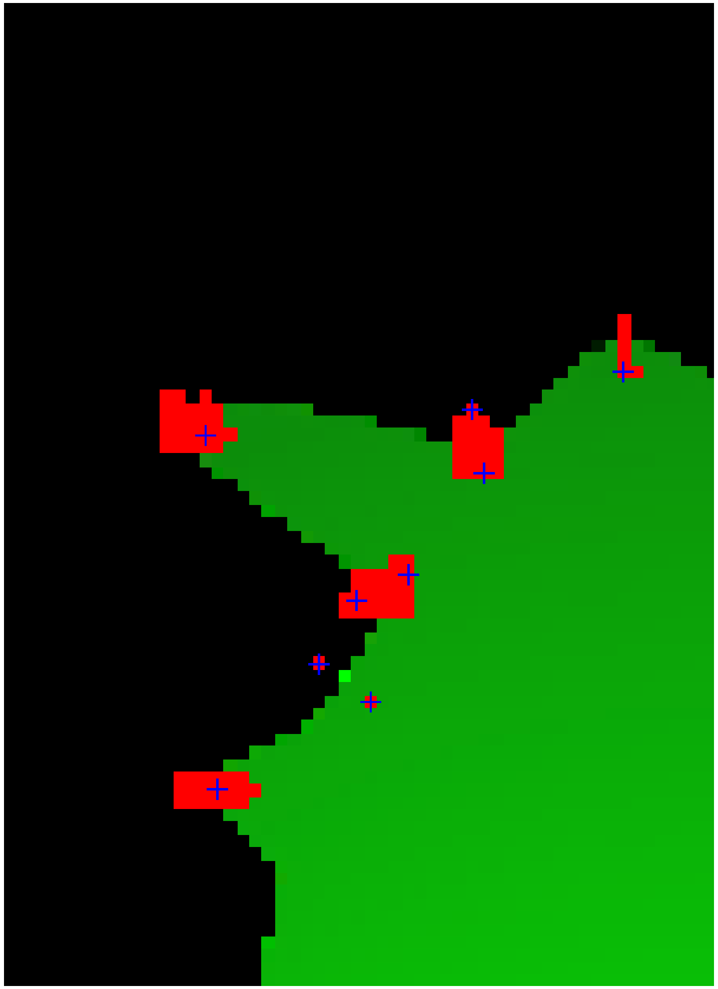


Thresholded regions of w and q



Extract interest points

- Use product $w .* q$:
  - Apply MATLAB function `imregionalmax` to detect local maxima
  - Output: binary mask with peaks
  - Use functions `find` and `plot` to derive and plot the points of interest



Task B: Förstner Operator

Idea: Use GoG-images to identify Förstner points

a. Compute the autocorrelation matrix M for each pixel using a 5x5 moving window
b. Instead of storing M for each pixel, compute the cornerness w and roundness q from M and store these values in matrices W and Q. Plot the arrays
c. Derive a binary mask M_c of potential interest points by simultaneously applying thresholds, e.g. $t_w = 0.0004$ and $t_q = 0.5$, on W and Q
d. Multiply W and Q with the resulting mask M_c of step c ($\bar{W} = W \cdot M_c$, $\bar{Q} = Q \cdot M_c$) and apply the function `imregionalmax` to $\bar{W} \cdot \bar{Q}$ in order to derive the points of interest
e. Plot an overlay of the initial input image and the detected points

Bauhaus-Universität Weimar

Thank you!

Questions?