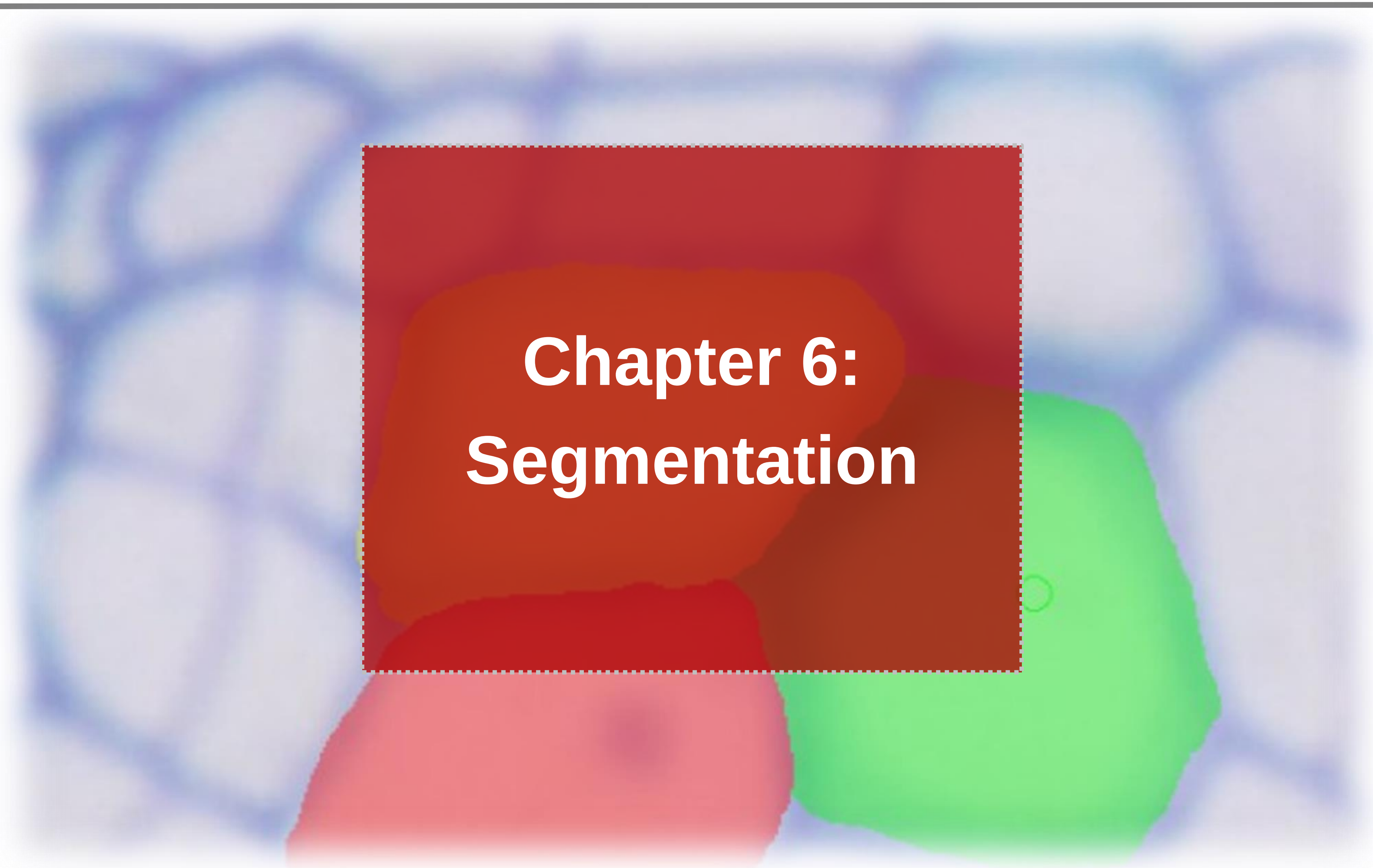
Solution Exercise 5: Morphology



Chapter 6: Segmentation

Segmentation is one of the basic steps in image processing. It links the low level image processing steps with the high level ones.

Image Acquisition → Image Preprocessing → Segmentation → Feature Extraction → Classification

Definition:

- Separation of the image in partial areas (regions/ segments) with same properties

Application:

- Separation of foreground from background and vice versa
- Extraction of objects

Properties:

- Complete: every pixel belongs to at least one segment
- Overlap free: every pixel belongs at most to one segment
- Coherence: every (foreground-) segment builds a coherent (connected) object

- Using model information in the segmentation process for higher robustness

Template Matching

(1) Define a pattern p(k,l) (called Template) in a way that shape and orientation of the segment is similar in the image g(x,y)

(2) Determine normalized cross correlation function:

$$c(x,y) = \frac{\sum_k \sum_l g(x+k, y+l) p(k,l)}{\sqrt{\sum_k \sum_l g(x+k, y+l)^2} \sqrt{\sum_k \sum_l p(k,l)^2}}$$

indices k and l are only applied in regions, where g(x+k,y+l) and p(k,l) are overlapped

(3) The segments are on the local extremal points of c(x,y)

Properties:

- very easy to apply
- suffers from noise and invariances (rotation, scaling, illumination, etc.)
- only possible, when it is clear how the searched-for object looks like (in reality often not)

Example: Template Matching

Hough Transformation

Main Idea:

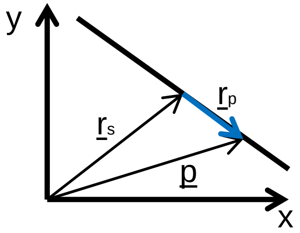
- Detect geometrical shapes (e.g. straight lines, circles, ellipse) in images by transferring them into a special parameter space

Theoretical Background:

- Search straight lines in a binary image b(x,y)
- Using this line equation:

$$\underline{p}^T \cdot \underline{n} = d$$

- Hessian normal form
  - $\underline{n}$ – normal vector ⊥ on the straight line with the length $||n||=1$ and the angle φ to x-axis → $\underline{n}\begin{pmatrix}\cos\phi \\ \sin\phi\end{pmatrix}$
  - $\underline{p}$ – position vector of all line points
  - d – distance of the line to the point of origin



$$\underline{p} = \underline{r}_s + \underline{r}_p$$

$$|d| = ||\underline{r}_s||$$

  - in consequence: $\mathrm{x}\cos\phi + \mathrm{y}\sin\phi = \mathrm{d}$

- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

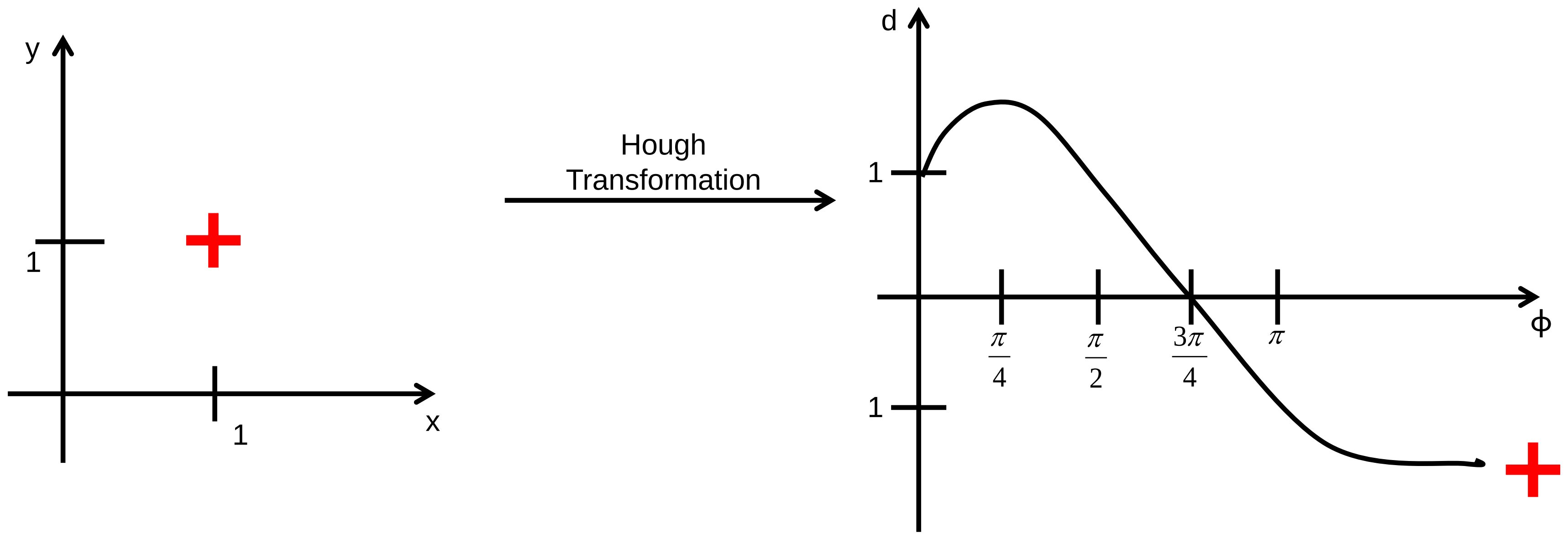
➢ $$H_k : x_k \cos\phi + y_k \sin\phi = d$$

Algorithm:

(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

cartesian coordinate system

parameter space

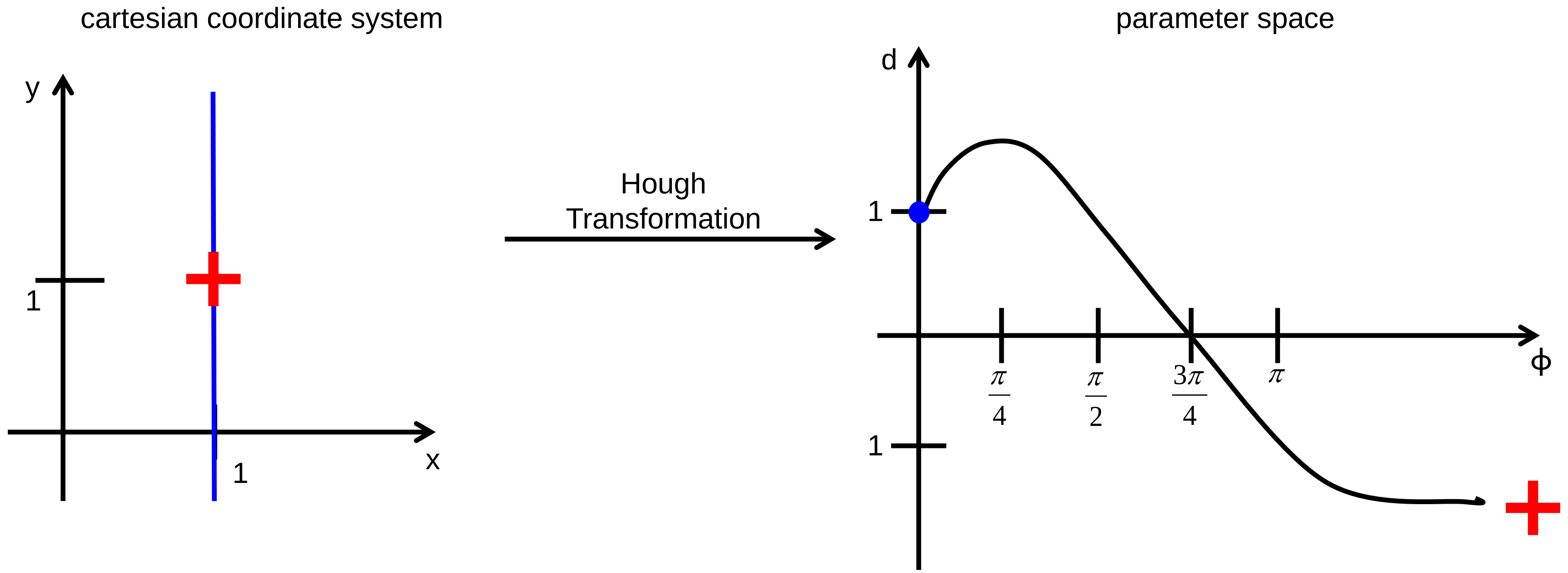


- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

➢ $H_k : x_k \cos\phi + y_k \sin\phi = d$

Algorithm:

(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

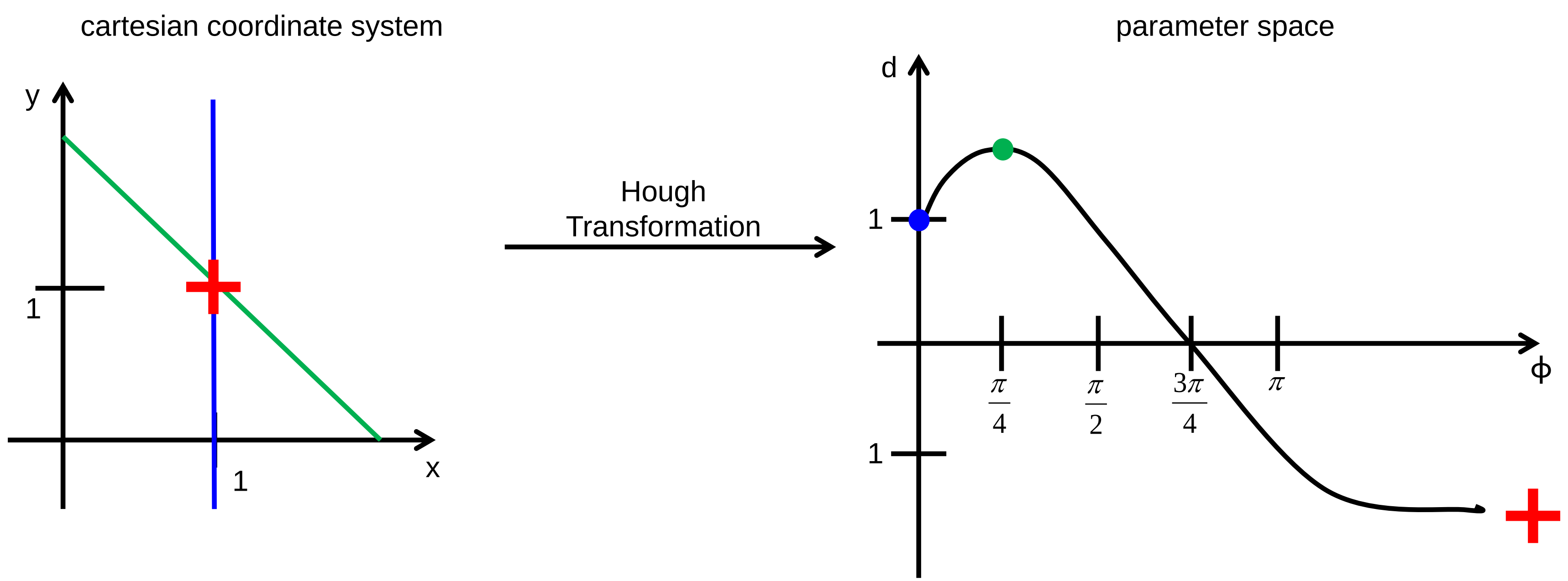


- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

➢ $H_k : x_k \cos\phi + y_k \sin\phi = d$

Algorithm:

(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

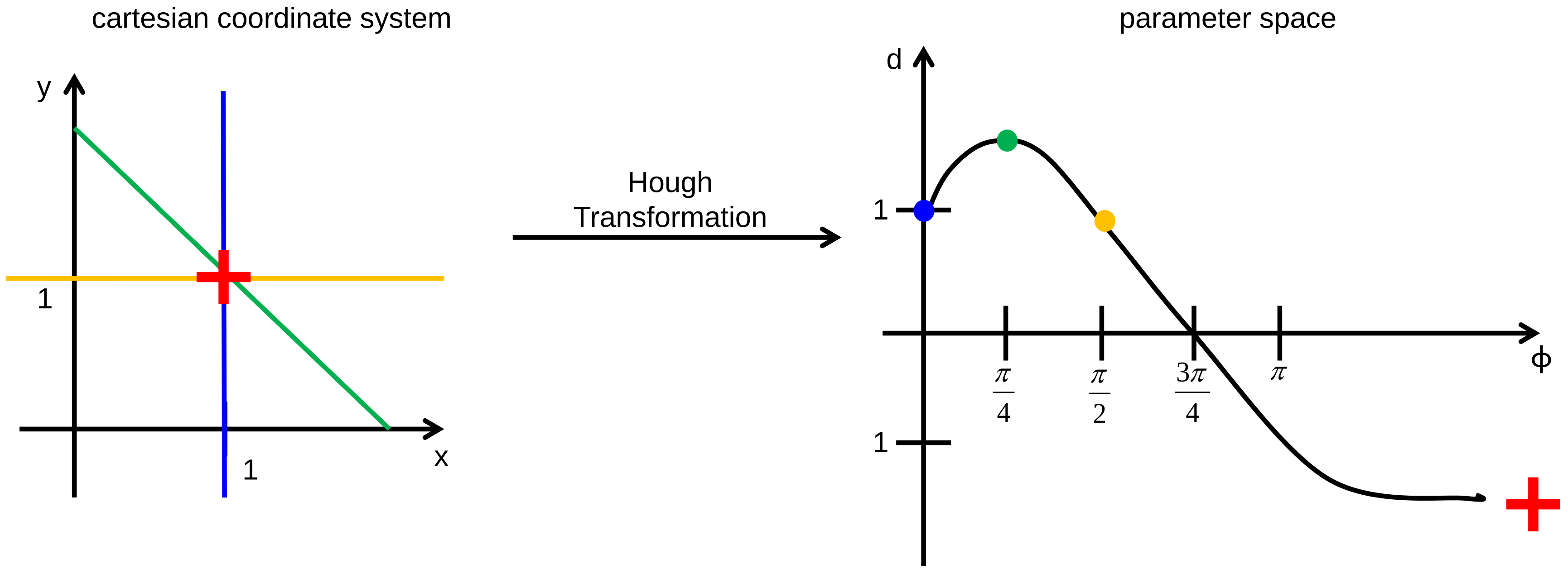


- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

➢ $$H_k : x_k \cos\phi + y_k \sin\phi = d$$

Algorithm:

(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

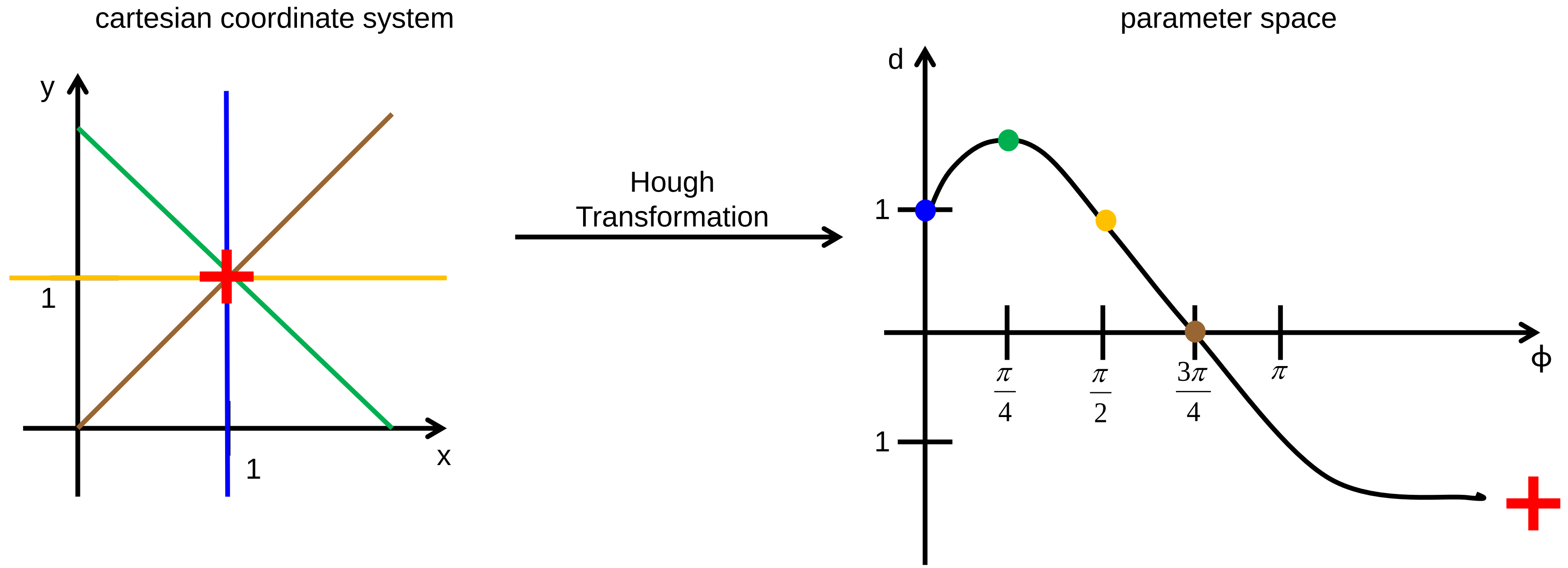


- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

➢ $$H_k : x_k \cos\phi + y_k \sin\phi = d$$

Algorithm:

(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

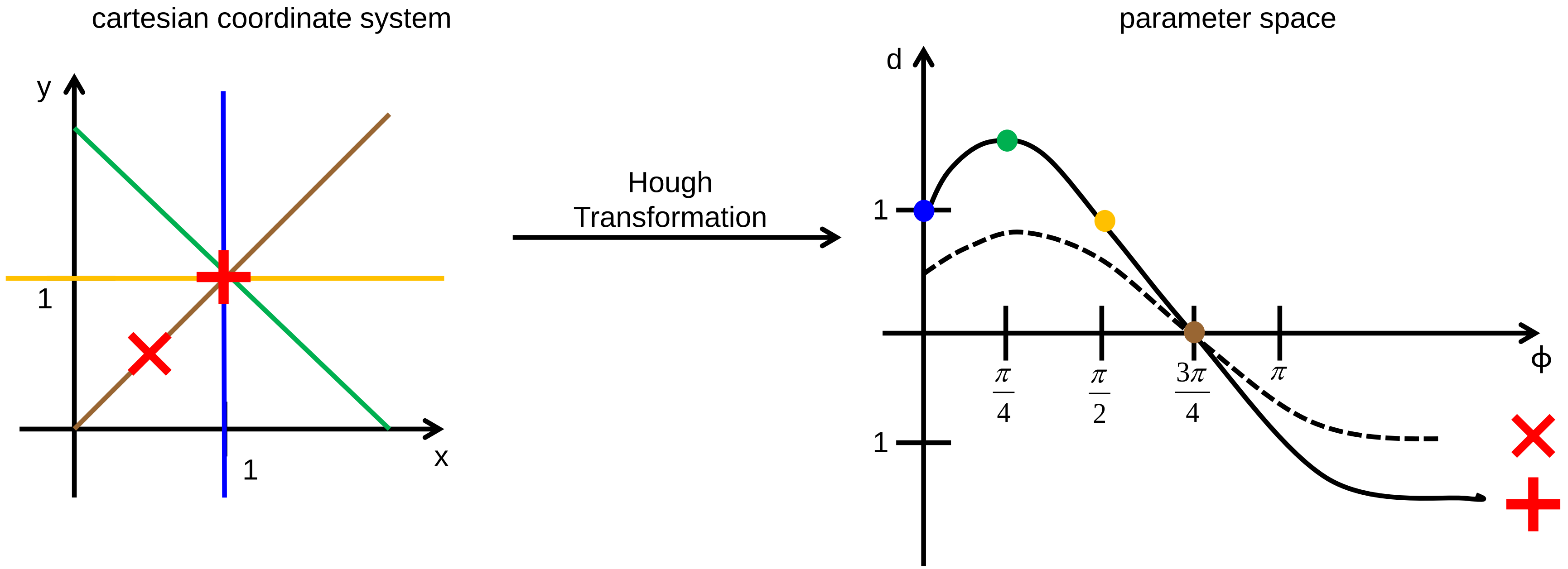


- coordinate transformation:
  - All straight lines touching the points $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in the cartesian coordinate system will be represented in the (φ,d) - coordinate system with a curve

➢ $$H_k : x_k \cos\phi + y_k \sin\phi = d$$

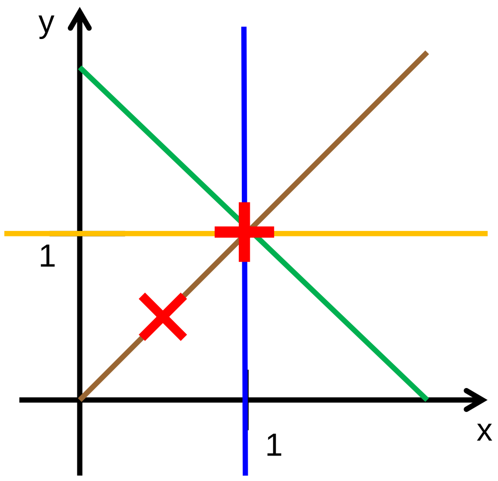
Algorithm:

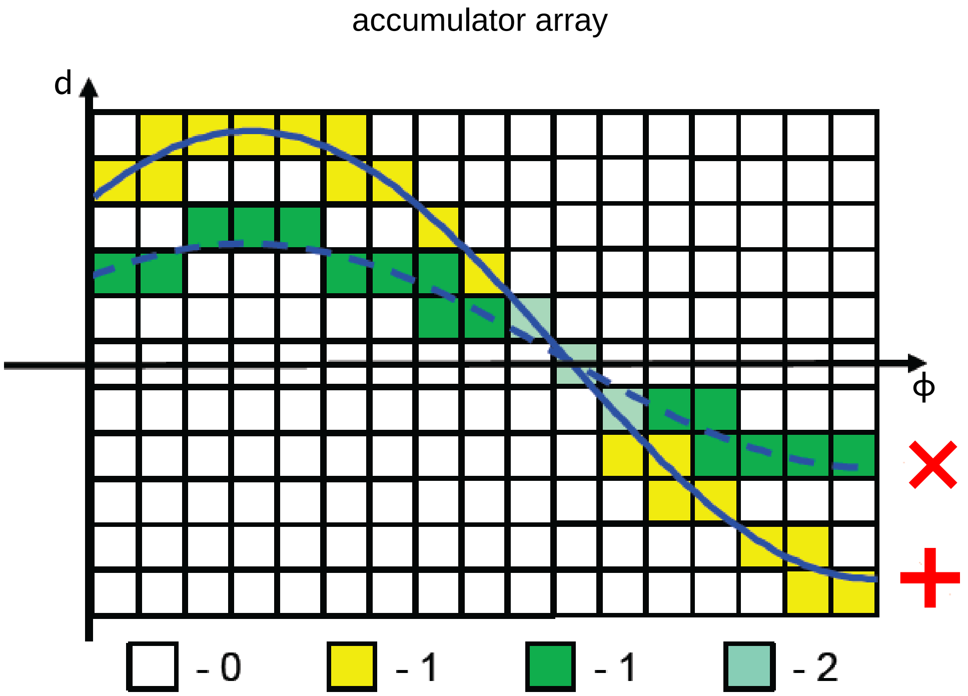
(1) Transformation of all foregroundpixels $\underline{p}_k = \begin{pmatrix} x_k \\ y_k \end{pmatrix}$ in g(x,y) to the corresponding curves of H_k in the (φ,d) – parameter space

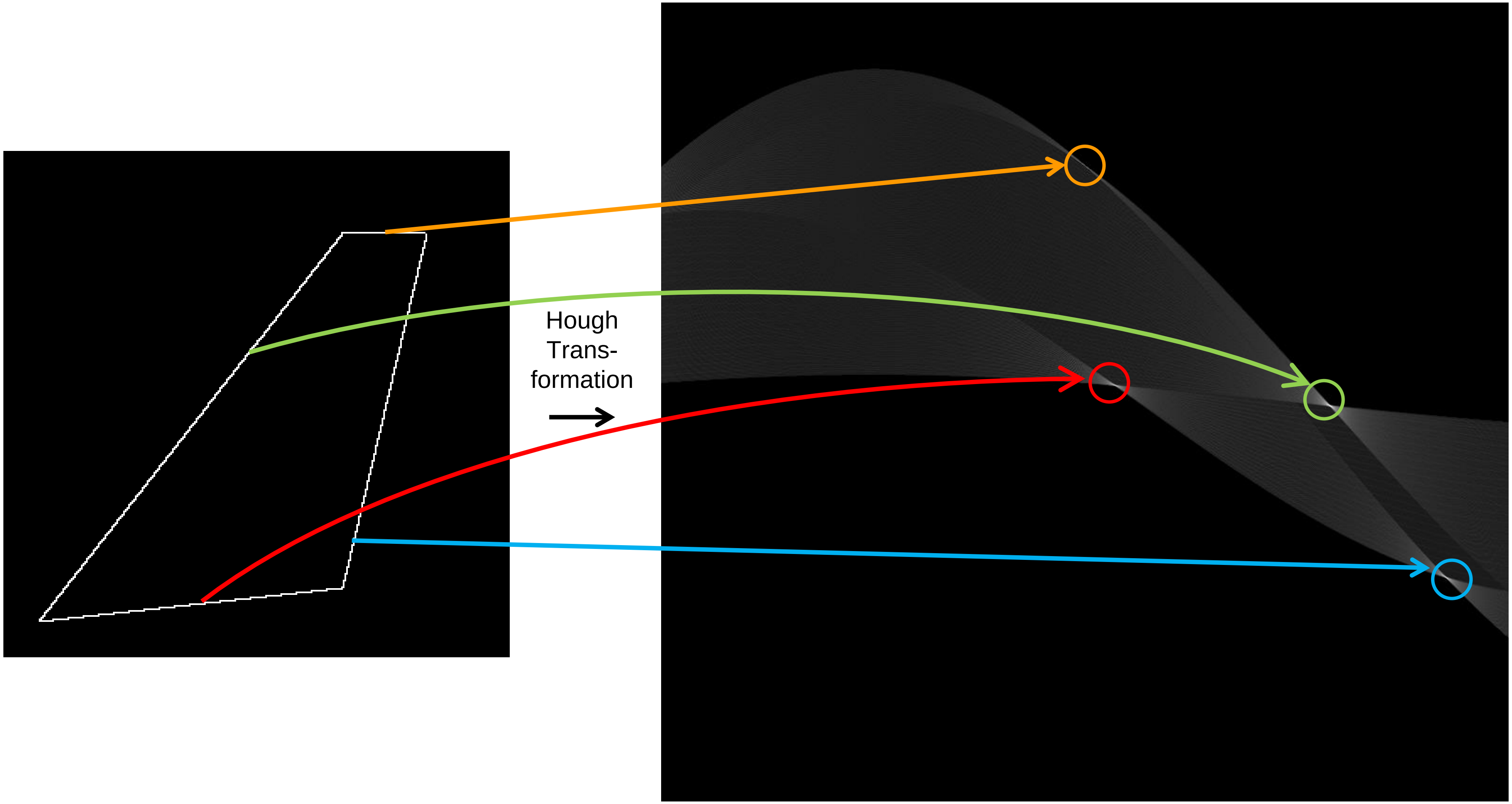


(2) Search for points in the (φ,d) - parameter space, where many curves H_k are intersecting

- for easy implementation use accumulator array
- every foreground pixel p_k crossing the curve H_k results in increasing the bins of the accumulator array by 1
- straight lines are at the maxima in the accumulator array







[Source: Tönnies, „*Grundlagen der Bildverarbeitung*"]

Sixth Exercise

- Implement Template Matching via Cross Correlation and Hough Transformation for straight lines

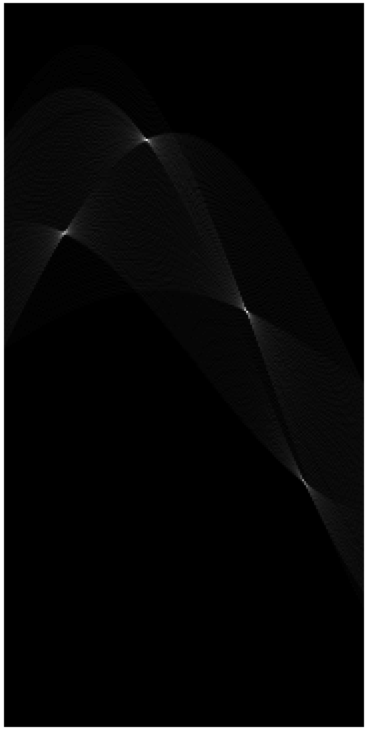
Expected Output (Template Matching)

Cross Correlation Image

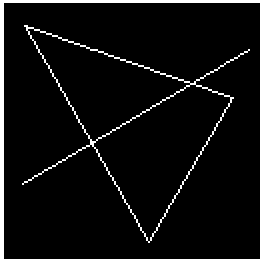
Matched Template

Expected Output (Hough Transformation)

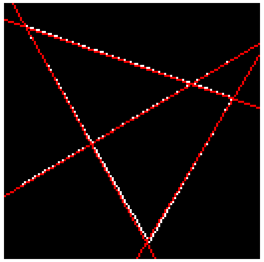
Hough Transformation



Straight Lines



Detected Lines



That is all for today.