Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Introduction to Image Processing:

Global Operations: Hough Transform

Marcus Hudritsch (hsm4)

Hough Transform

- Die Hough-Transformation wird gebraucht, um geometrische Strukturen wie **Geraden**, **Kreise** und **Ellipsen** in einem binarisierten Bild zu detektieren.
- Auch die Hough-Transformation ist eine rechenintensive Aufgabe.
- Sie aber sehr **robust** und **wenig anfällig gegen Rauschen** ist.

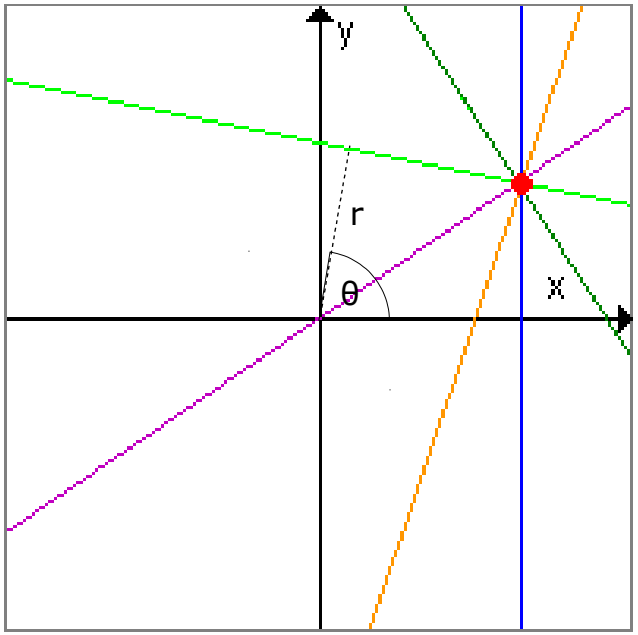
Hough Transformation von Linien

- Für eine Gerade wird nicht die übliche Geradengleichung

  $$ax+by+c=0$$

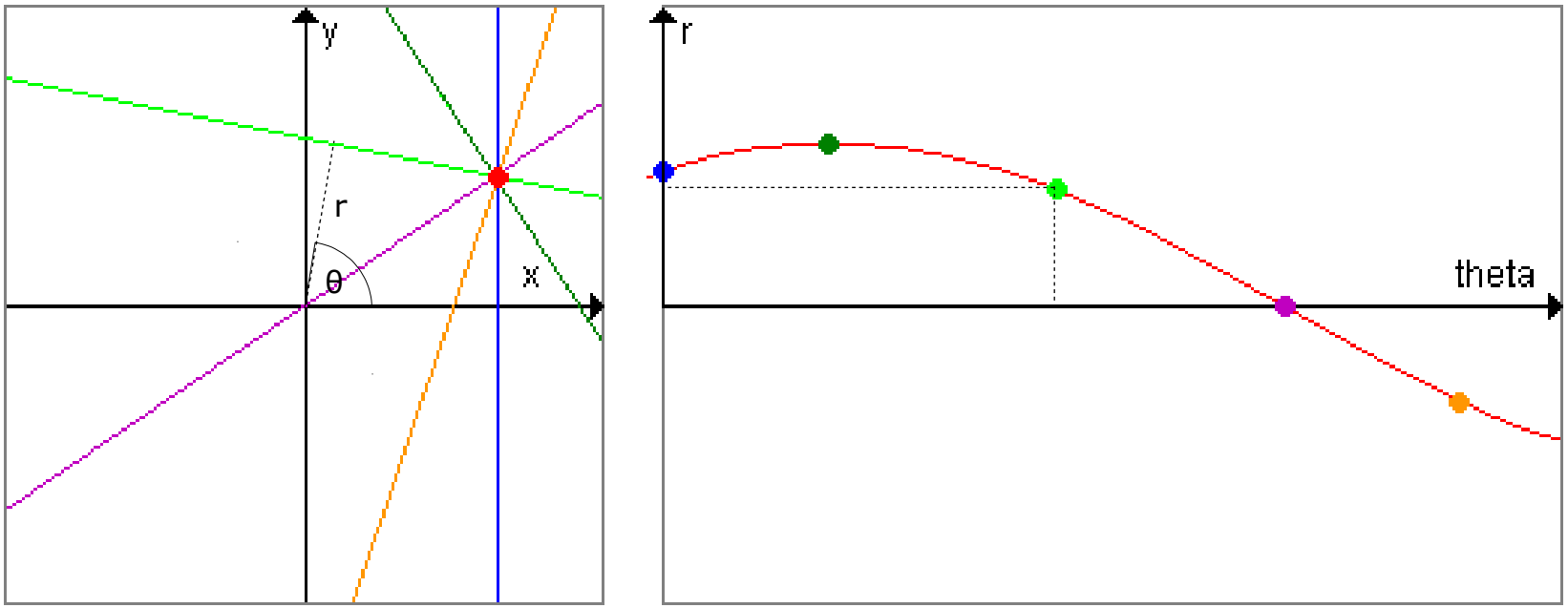
  im kartesischen Koordinatensystem verwendet, sondern die unübliche **Hesse'sche Normalform**: $r=x\cos(\varphi)+y\sin(\varphi)$

- ***r*** ist darin der Abstand der Geraden zum Ursprung und ***theta*** ist der Winkel zur X-Achse.



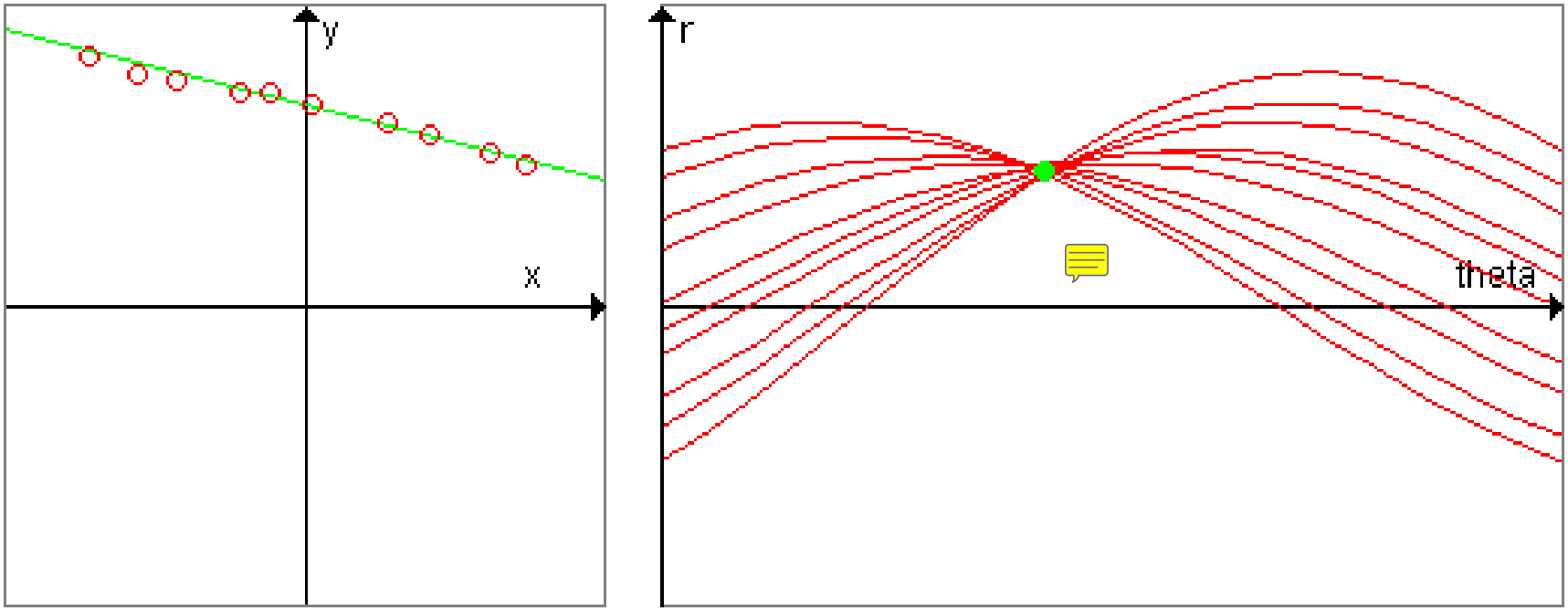
Hough Transformation von Linien

- Für den roten Punkt ergeben sich unendlich viele Geraden. Diese Geraden in den **Hough-Raum** mit ihren Parametern *r* und *theta* so ergibt sich eine rote Kurve auf der ein Punkt einer Geraden entspricht:



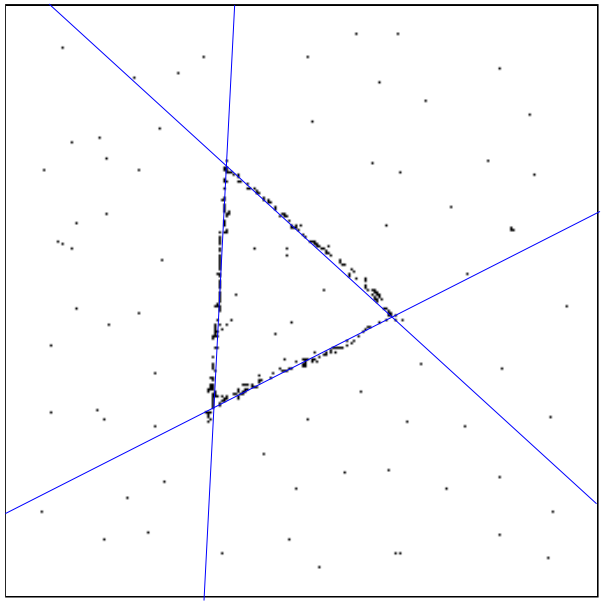
Hough Transformation von Linien

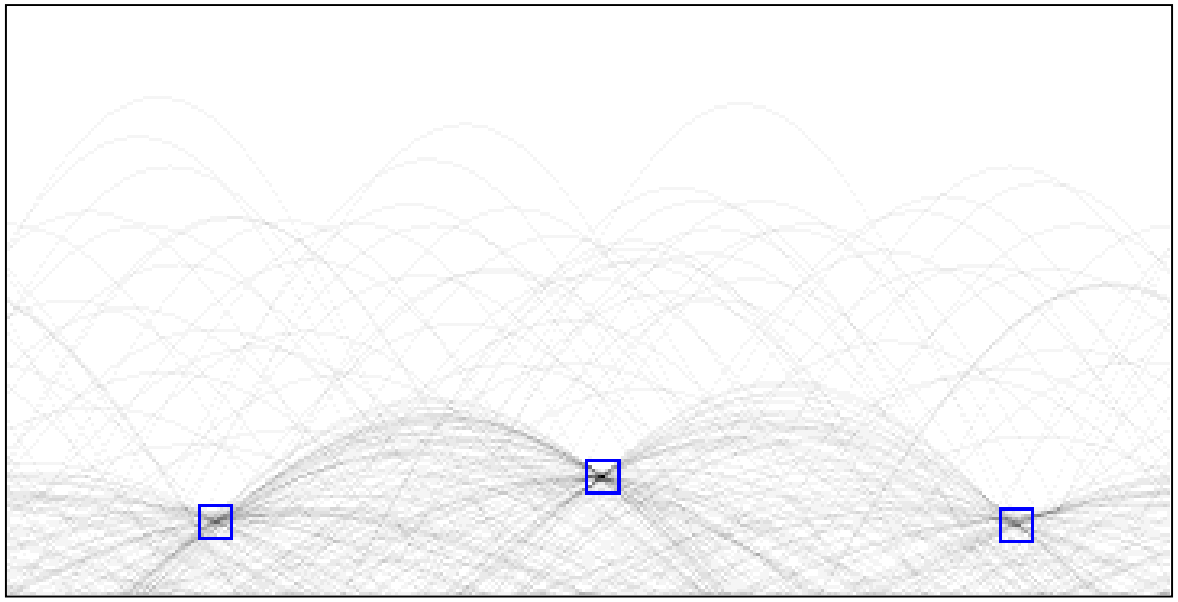
- Umgekehrt entspricht der grüne Punkt im Hough-Raum einer Geraden im kartesischen Raum. Durch diesen Punkt führen unendlich viele Parameterkurven mit unterschiedlichen ***r*** und ***theta***.
- Alle diese Parameterkurven entsprechen einem Punkt auf derselben grünen Geraden im kartesischen Raum.



Hough Transformation von Linien

- Um **eine Gerade zu detektieren** wird für jeden schwarzen Punkt im Quellbild eine Kurve in einem **diskreten Hough-Raum** gezeichnet.
- Dabei werden die Punkte auf der Kurve nicht einfach gesetzt sondern inkrementiert.
- Nur dort, wo sich viele Kurven kreuzen werden deutlichere Punkte sichtbar. Die Maxima entsprechen den Geraden:



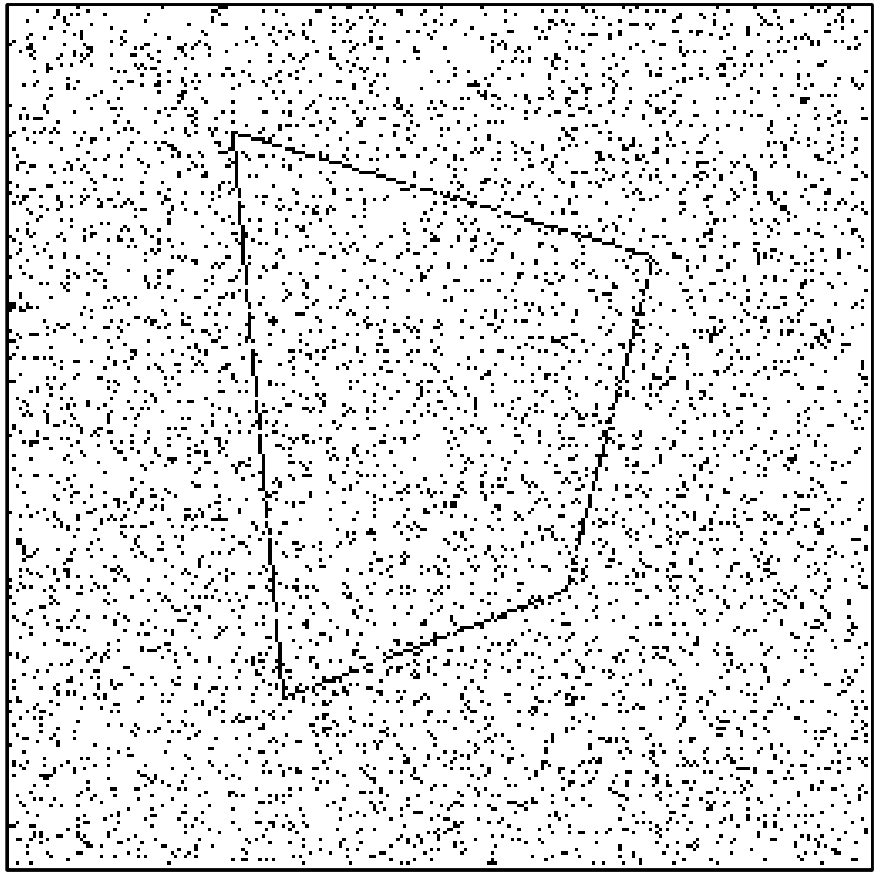


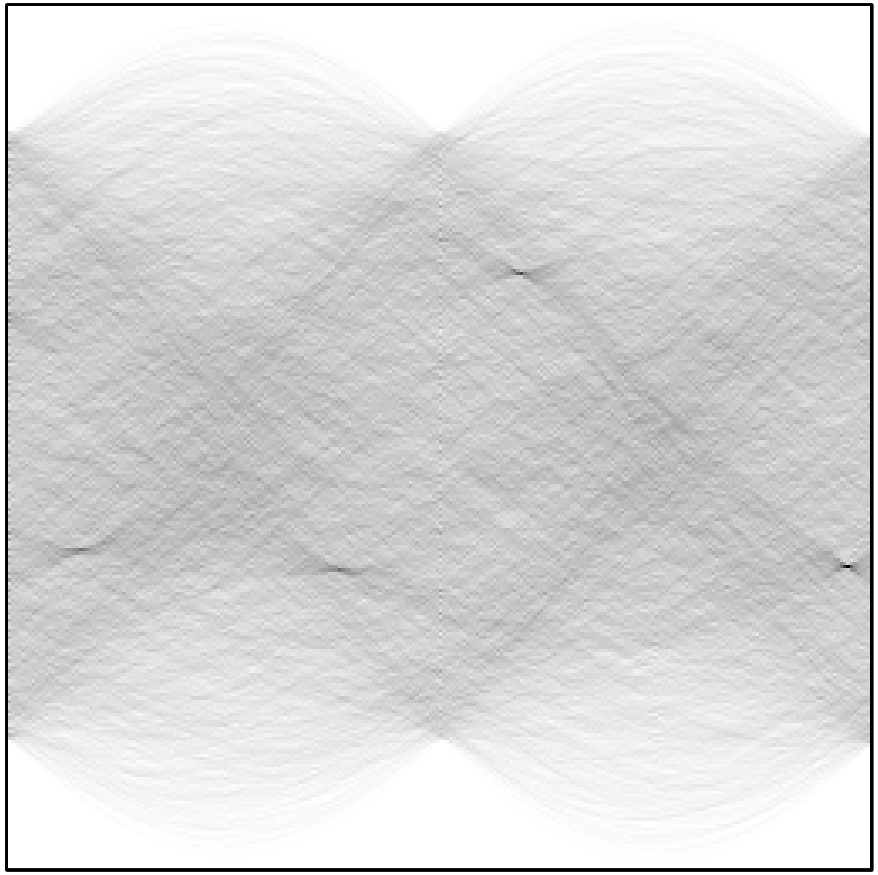
HT von Linien: Hough-Raum erstellen in Java:

```
final int nAng = 256;                    // number of angels
final int nRad = 256;                    // number of radii
int xC = ip.getWidth()/2;                // x-coordinate of image center
int yC = ip.getHeight()/2;               // y-coordinate of image center
double dAng = (Math.PI/nAng);            // step size of angle
double rMax = Math.sqrt(xC*xC + yC*yC); // max. radius (center to corner)
double dRad = (2*rMax)/nRad;             // step size radius
int[][] hough = new int[nAng][nRad];     // Hough accumulator space

// Fill Hough array
for (int y = 0; y < ip.getHeight(); y++)
{   for (int x = 0; x < ip.getWidth(); x++)
    {   if (ip.getPixel(x,y) > 0)
        {   int cx = x-xC, cy = y-yC;

            // Calculate for all theta and r and increment the hough array
            for (int t = 0; t < nAng; t++)
            {   double theta = dAng * t;
                int r = (int)((cx*Math.cos(theta) +
                               cy*Math.sin(theta)) / dRad) + nRad/2;
                if (r >= 0 && r < nRad) hough[t][r]++;
            }
        }
    }
}
```

HT von Linien: Detektieren der Linien

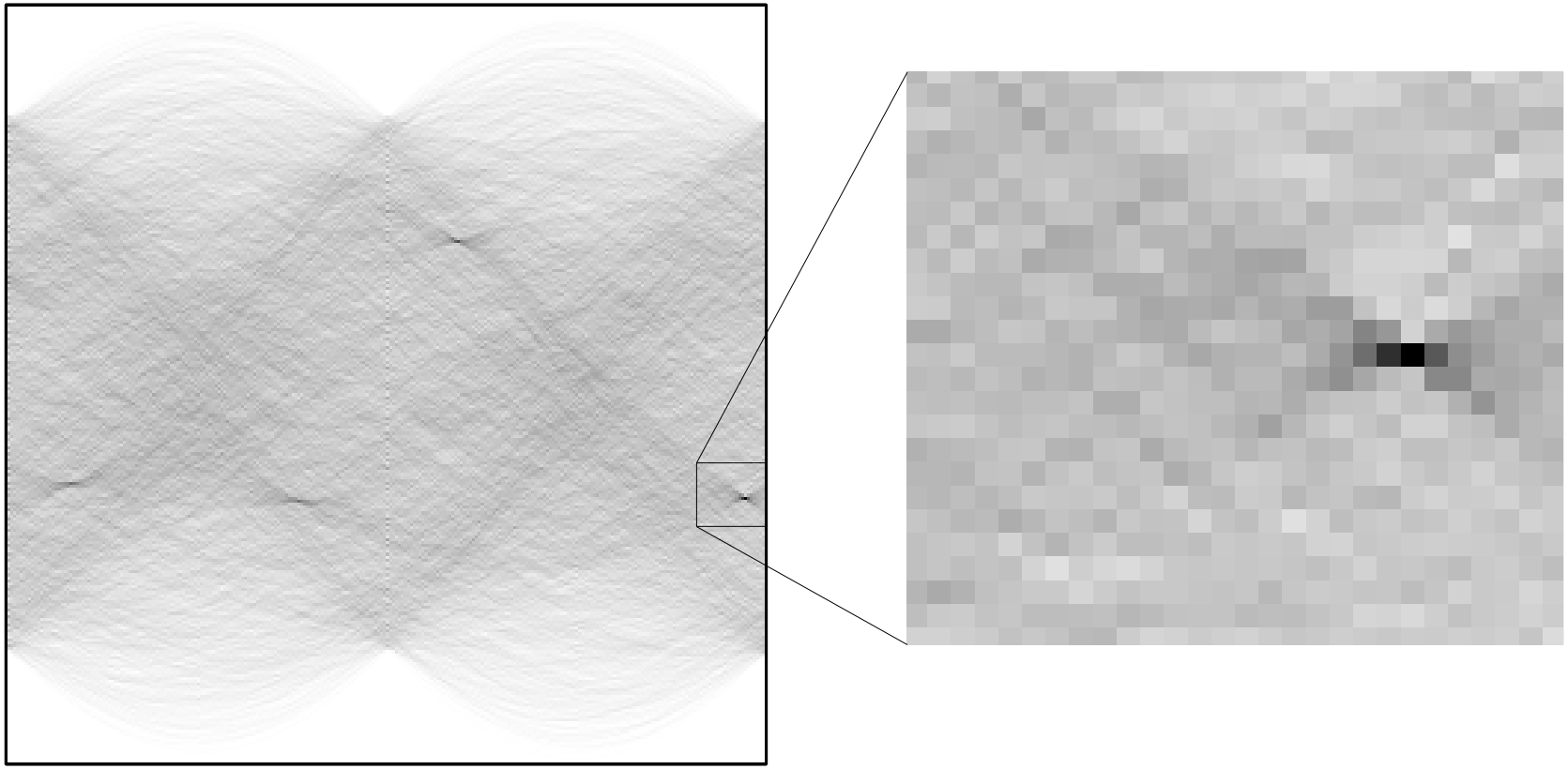
Bei einem verrauschten Bild erhalten wir auch einen flacheren Hough-Raum:





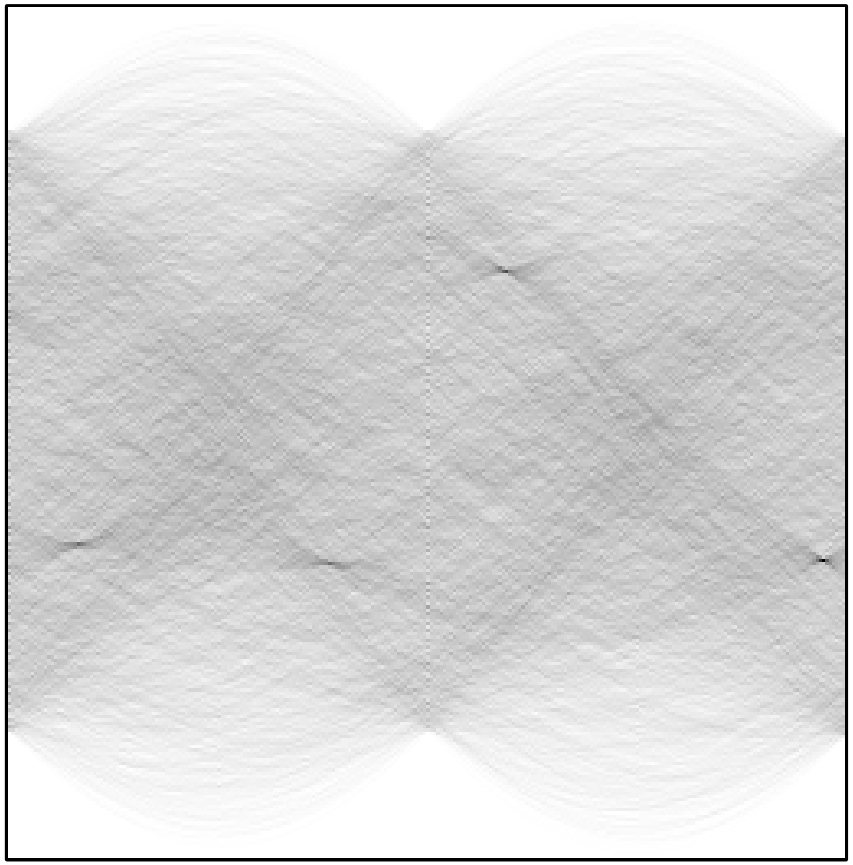
HT von Linien: Detektieren der Linien

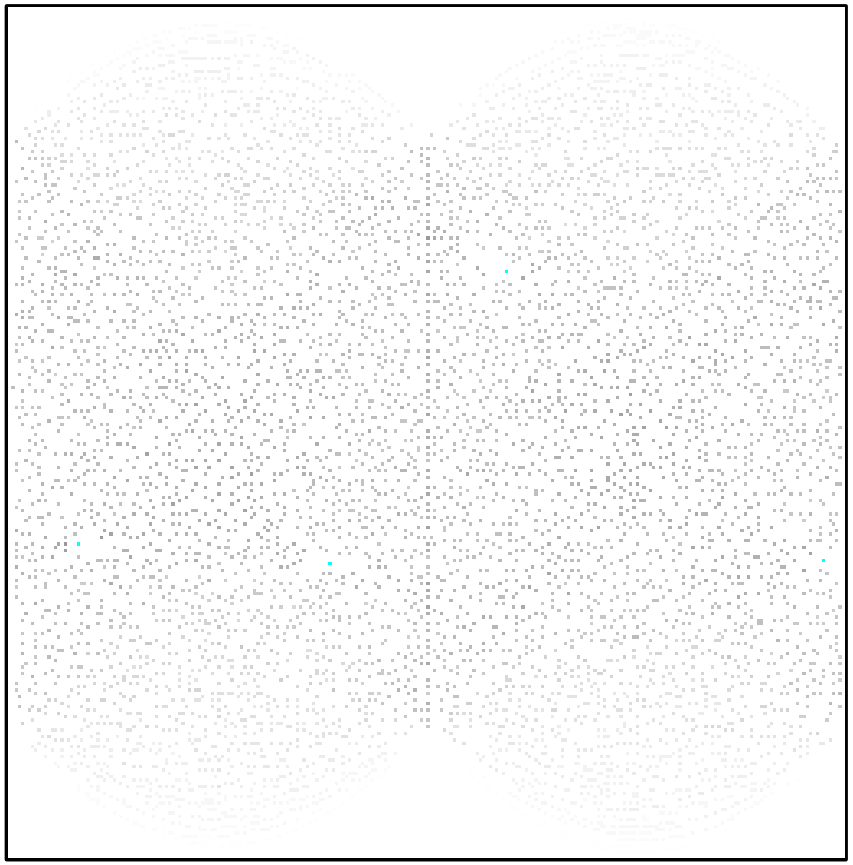
Es kann passieren, dass benachbarte Punkte eines Peaks höher sind als andere Peaks:



HT von Linien: Detektieren der Linien

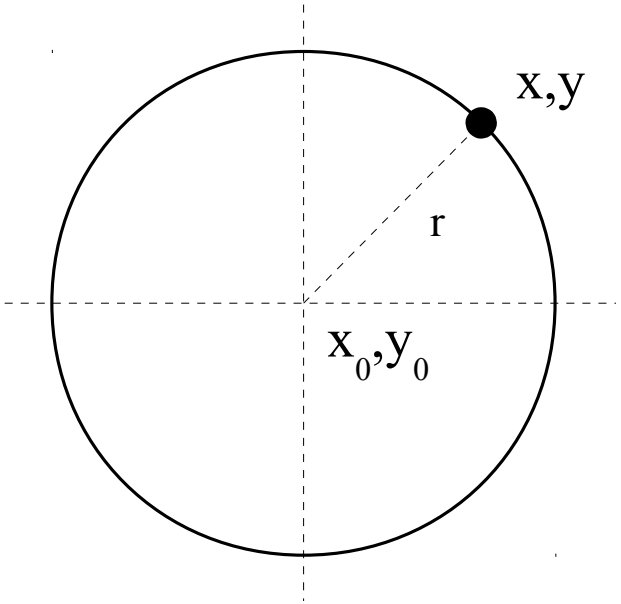
- Wir müssen zuerst eine ***Non-Maximum Suppression*** machen.
- Sie müssen dabei mit einer 3x3 Maske durch den Hough-Raum iterieren und alle nicht Punkte ausser dem Maximum löschen:





Hough Transformation von Kreisen

Ein Kreis ist durch sein Zentrum (x0, y0), seinen Radius r und durch die Kreisgleichung definiert:



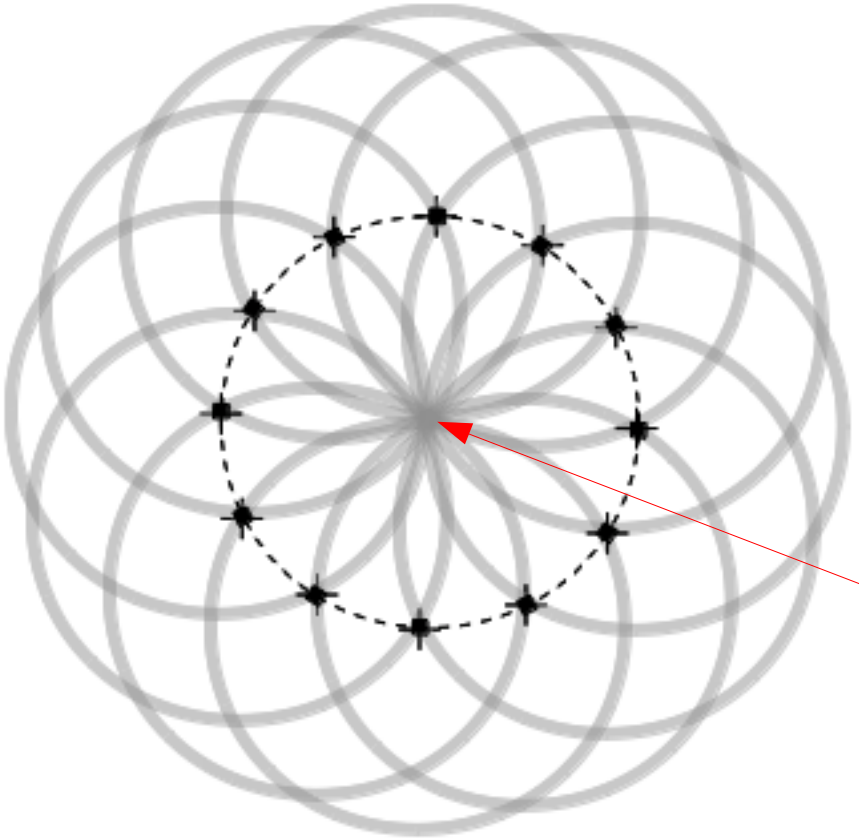
$$(x-x_0)^2+(y-y_0)^2-r^2=0$$

Hough Transformation von Kreisen

- Das Akkumulatorraum für Kreise ist damit dreidimensional.
  - In der Ebene ist er gleich gross wie das Bild.
  - In der Tiefe ist er gleich gross wie die Anzahl der Radien.

Hough Transformation von Kreisen

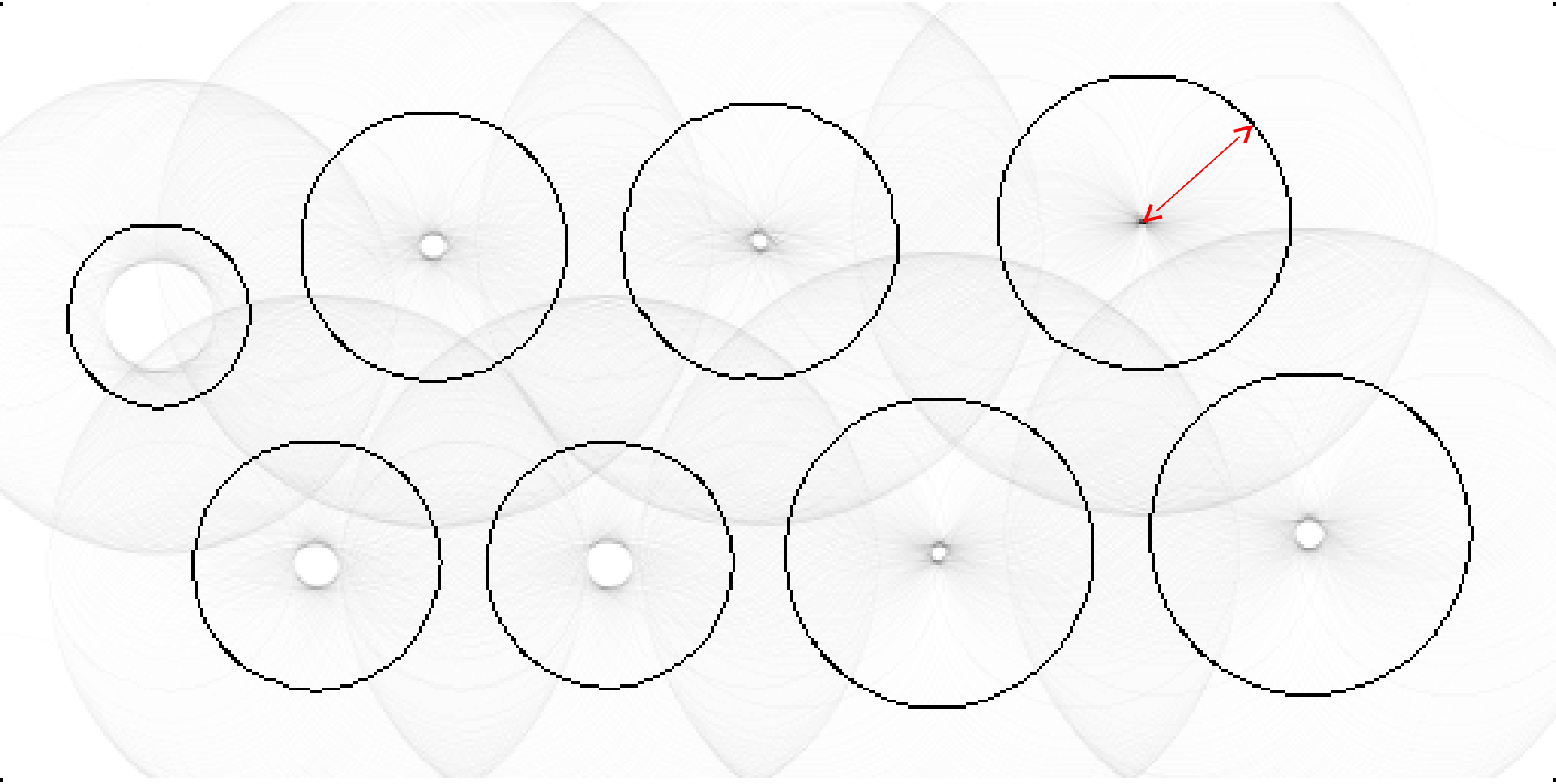
- In der Akkumulatorebene für einen bestimmten Radius wird für jedes schwarze Pixel ein Kreis gezeichnet.
- Dabei werden wiederum bestehende Werte nicht überschrieben, sondern auf akkumuliert.



Max. = Kreiszentrum

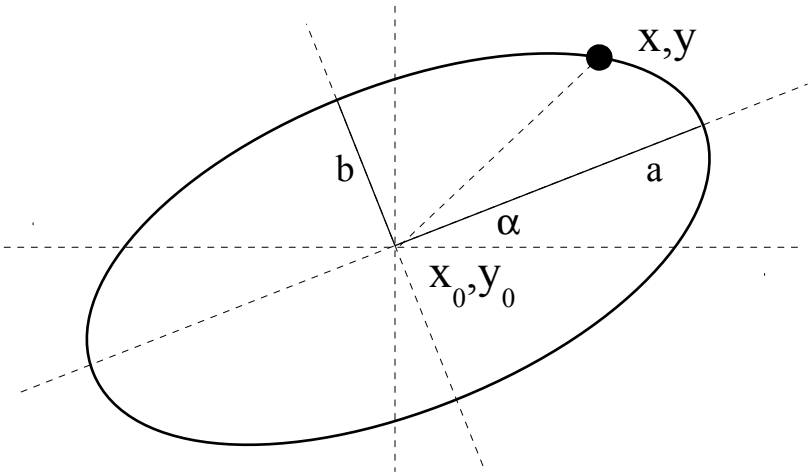
Hough Transformation von Kreisen

- Gesucht wurde ein Kreis mit einem Radius von 42 Pixeln.
- Quellbild und Akkumulatorbild übereinander:



Hough Transformation von Ellipsen

- Der Hough-Raum für Ellipsen ist 5-dimensional:
  - $\mathbf{x_0}$ und $\mathbf{y_0}$ für den Mittelpunkt
  - **a** und **b** für die Radien
  - **α** für den Rotationswinkel



$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} x_0 + a\cos t\cos\alpha - b\sin t\sin\alpha \\ y_0 + a\cos t\sin\alpha + b\sin t\cos\alpha \end{pmatrix} \quad \text{mit} 0 \le t \le 2\pi$$

Hough Transformation von Ellipsen

- Der Hough-Raum für Ellipsen ist 5-dimensional
- Dies kann schnell zu Speicherknappheit führen
- Bei einer Auflösung von 128 = 2^7 Schritten in jeder Dimension ergeben sich bereits 2^{35} Akkumulator-Zellen. Bei 4-Byte-Integers sind das 2^{37} Bytes bzw. 128 Gigabytes.

Übungen zur Hough Transformation

- Siehe Kapitel 7.3.4 und 7.3.5