ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
INSTITUT FÜR INFORMATIK
Lehrstuhl für Mustererkennung und Bildverarbeitung
Prof. Dr.-Ing. Hans Burkhardt

Georges-Köhler-Allee
Geb. 052, Zi 01-029
D-79110 Freiburg
Tel. 0761 - 203 - 8260

Übungen zur Vorlesung
Grundlagen der Bilderzeugung und Bildanalyse WS 05/06

Musterlösung 2

Aufgabe 2.1: Affine Transformation, Rotationsmatrizen

a) Da die Punkte $\mathbf{x}'$, $\mathbf{y}'$ und $\mathbf{z}'$ das Ergebnis der Ausübung der gleichen affinen Transformation, beschrieben durch eine 2×2 Matrix $\mathbf{A}$ und einen 2-Vektor $\mathbf{t}$, auf die Punkte $\mathbf{x}$, $\mathbf{y}$ und $\mathbf{z}$ sind, haben wir:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{t} \quad \wedge \quad \mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{t} \quad \wedge \quad \mathbf{z}' = \mathbf{A}\mathbf{z} + \mathbf{t} \quad \Rightarrow$$
$$\mathbf{x}' - \mathbf{y}' = \mathbf{A}(\mathbf{x} - \mathbf{y}) \quad \wedge \quad \mathbf{x}' - \mathbf{z}' = \mathbf{A}(\mathbf{x} - \mathbf{z})$$

Als Matrix zusammengefasst:

$$\left[\begin{array}{cc} \mathbf{x}' - \mathbf{y}' & \mathbf{x}' - \mathbf{z}' \end{array} \right] = \mathbf{A} \left[\begin{array}{cc} \mathbf{x} - \mathbf{y} & \mathbf{x} - \mathbf{z} \end{array} \right] \quad \Rightarrow$$

$$\mathbf{A} = \left[\begin{array}{cc} \mathbf{x}' - \mathbf{y}' & \mathbf{x}' - \mathbf{z}' \end{array} \right] \left[\begin{array}{cc} \mathbf{x} - \mathbf{y} & \mathbf{x} - \mathbf{z} \end{array} \right]^{-1} = \begin{bmatrix} 7 & 2 \\ -4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 9 & 2 \\ 2 & 6 \end{bmatrix}$$

Die Translation $\mathbf{t}$ gewinnt man etwa aus $\mathbf{t} = \mathbf{x}' - \mathbf{A}\mathbf{x} = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$.

b) Wir suchen eine orthogonale Transformation $\mathbf{R}$, die die Matrix $\mathbf{A}$ in eine reelle Diagonalform $\mathbf{D}$ überführt. Die Matrix $\mathbf{D}$ wird genau die Eigenwerte der Matrix A auf der Diagonale sitzen haben. Wir suchen also $\lambda \in \mathbb{R}$ und $\mathbf{v} \in \mathbb{R}^2$ so, dass $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. Dies kann nur für die λ gelten, die das charakeristische Polynom $\det(\mathbf{A} - \lambda\mathbf{I})$ verschwinden lassen.

$$\begin{aligned} & \det(\mathbf{A} - \lambda\mathbf{I}) = (9 - \lambda)(6 - \lambda) - 4 = 0 \qquad (1) \\ \Rightarrow \quad & \lambda^2 - 15\lambda + 50 = 0 \\ \Rightarrow \quad & \lambda_{1/2} = 7.5 \pm \sqrt{56.25 - 50} = 7.5 \pm 2.5 \\ \Rightarrow \quad & \lambda_1 = 10, \qquad \lambda_2 = 5 \end{aligned}$$

Nun können auch leicht die entsprechenden Eigenvektoren mit Hilfe der Gleichung $(\mathbf{A} - \lambda_i\mathbf{I})\mathbf{v}_i = 0$ bestimmt werden. Man erhält

$$\mathbf{v}_1 = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \end{pmatrix},$$

wobei die $\mathbf{v}_i$ bereits normiert worden sind, d.h. $|\mathbf{v}_i| = 1$. Die Koordinatentransformation, welche die Basisvektoren $(1, 0)^T$ und $(0, 1)^T$ auf die Eigenvektoren $\mathbf{v}_1$ und $\mathbf{v}_2$

abbildet ist gerade durch die Matrix $\mathbf{R}$ gegeben, die $\mathbf{v}_1$ und $\mathbf{v}_2$ als Spaltenvektoren enthält, also

$$\mathbf{R} = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix}, \qquad bzw.\ \mathbf{R}^{-1} = \mathbf{R}^T = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix}$$

Die inverse Transformation $\mathbf{R}^T$ entspricht gerade der Transformation, die das Standardkoordiantensystem in das Eigensystem der Matrix $\mathbf{A}$ überführt, also ergibt sich

$$\mathbf{A} = \mathbf{RDR}^T = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix} \cdot \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix}$$

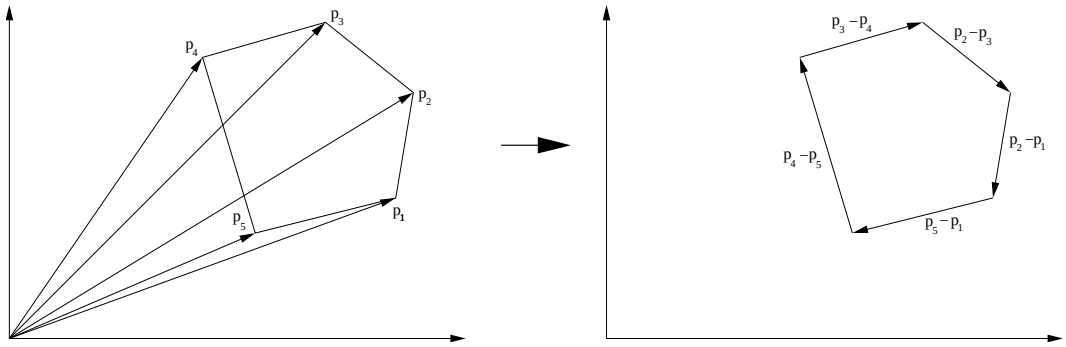
Liest man $\mathbf{RDR}^T$ von rechts nach links: Mit $\mathbf{R}^T$ wird in das Eigensystem von $\mathbf{A}$ gewechselt. Dort ist $\mathbf{A}$ nur eine komponentenweise Multiplikation, d.h. diagonal. Und schliesslich wird wieder in das Standardsystem zurücktransformiert.

Gleichung (1) ist nur reell lösbar falls die Matrix $\mathbf{A}$ symmetrisch ist, d.h. eine Darstellung in obiger Form ist nur möglich falls $\mathbf{A} = \mathbf{A}^T$. Dies ist auch sofort daran zu sehen, dass die geforderte Normalform $\mathbf{RDR}^T$ symmetrisch ist, was leicht nachzurechnen ist: $(\mathbf{RDR}^T)^T = ((\mathbf{R})^T)^T\mathbf{D}^T\mathbf{R}^T = \mathbf{RDR}^T$.

Aufgabe 2.2: Rotation, Kongruenzen, Invarianten

a) Um invariant bezüglich Translationen zu sein, wähle Darstellung

$$T(\mathbf{x}) = (p_1 - p_2, p_2 - p_3, \ldots, p_{n-1} - p_n, p_n - p_1)$$



b) Betrachte die Länge zwischen p_i und p_{i+1}

$$d_i = |p_i - p_{i+1}|$$

und den relativen Winkel α_i' zwischen $\underbrace{\overline{p_i p_{i+1}}}_{\mathbf{a}_i}$ und $\underbrace{\overline{p_{i-1} p_i}}_{\mathbf{b}_i}$

$$\cos(\alpha_i') = \frac{\mathbf{a}_i \mathbf{b}_i}{|\mathbf{a}_i||\mathbf{b}_i|}$$

Achtung! Orientierung geht verloren. Um dieses zu beheben betrachte

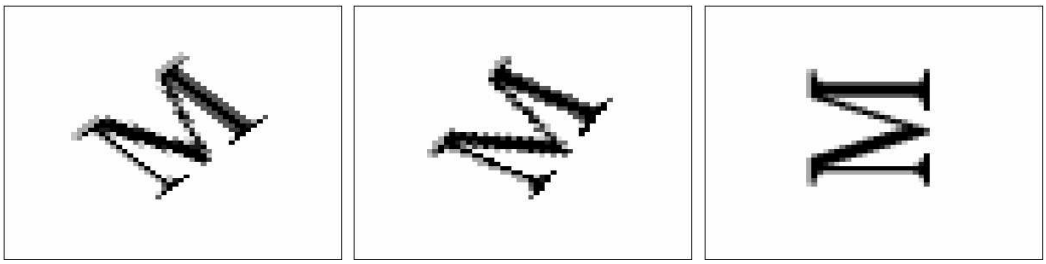
$$\alpha_i = \begin{cases} \alpha_i' & , \text{falls } orient_i = 1 \\ 2\pi - \alpha_i' & , \text{falls } orient_i = -1 \end{cases} \quad \text{mit } orient_i := sg(\det([\mathbf{a}_i, \mathbf{b}_i]))$$

Damit ist $T(\mathbf{x}) = \left[\begin{pmatrix} d_1 \\ \alpha_1 \end{pmatrix}, \ldots, \begin{pmatrix} d_n \\ \alpha_n \end{pmatrix} \right]$ eine vollständige, invariante Darstellung bezüglich Kongruenz.

Aufgabe 2.3: Programmieraufgabe: Rotation Folgende Funktion rotiert ein gegebenes Bild mittels der 'Nächste Nachbar'-Regel, d.h. gebrochene Koordinaten werden per `round` auf ganzahlige Koordinaten gerundet. Um 'Löcher' im Ergebnisbild zu vermeiden, wird dir Rotation rückwärts berechnet. Für jeden Pixel im Ergebnisbild wird die nächstgelegene rotierte Koordinate im Ursprungsbild festgestellt und der Grauwert an dieser Stelle übernommen.

```
function B = rotImg(A, phi)
// function p = rotImg(A, phi)
//
// Rotation eines Bildes mit 'Nächste Nachbar'-Regel
//
// input:    A            Bild beliebiger Größe
//           phi          Rotationswinkel
//
// output:   B            gedrehtes Bild (doppelt so groß wie Orginal)
//

// Konstruiere die Rotationsmatrix
R = [cos(phi) sin(phi); -sin(phi) cos(phi)];

// Hole Größen des Bildes
m = size(A,1);
n = size(A,2);
mhalbe = round(m/2);
nhalbe = round(n/2);

// Das Ergebnisbild sei doppelt so groß
B = ones(m*2,n*2)*255;

// Erweitere das Orginal auf Ergebnisgröße
As = ones(m*2,n*2)*255;
As(mhalbe:(m+mhalbe-1),nhalbe:(n+nhalbe-1)) = A

// Das Rotationszentrum
center = [m;n];

// Schleife über das Ergebnisbild
for x = 1:(2*m),
   for y = 1:(2*n),
      // rotiere Koordinate um center
      Z = round(R' * ([x ; y]-center) + center);
      // nur wenn legale Koordinate berechnet wurde
      if Z(1) > 0 & Z(1) <= 2*m & Z(2) > 0 & Z(2) <= 2*n,
         B(x,y) = As(Z(1),Z(2));
      end;
   end;
end;
```

Die Ergebnisse angewendet auf das gegebene Bild für Winkel von 45,60 und 90 Grad.



Offenbar enstehen durch die `round` Operation zahlreiche unschöne Artefakte