Hochschule Osnabrück
Fakultät Ingenieurwissenschaften und Informatik
Labor für Digital- und Mikroprozessortechnik
Prof. Dr.-Ing. B. Lang, Prof. Dr.-Ing. M. Weinhardt

Praktikum Bildverarbeitung, Übung 3

Sobel-Filter

Ziel dieser Praktikumsübung ist es, den Sobelfilter als Beispiel für einen linearen Filter zu programmieren. Das resultierende Gradientenbild soll dann in die Polarkoordinatendarstellung konvertiert werden. Aus der Polarkoordinaten-darstellung soll eine Art Histogramm ermittelt werden, welches die Gradientenbeträge über den Winkeln aufsummiert. Weiterhin sollen die Gradienten der Kanten auf einen Punkt breite Linien am Ort der maximalen Steigung reduziert werden.

In Aufgabe 1 werden die Gradienten im Bild in x- und in y-Richtung errechnet.

In Aufgabe 2 werden die Gradienten in die Polarkoordinatendarstellung überführt.

In Aufgabe 3 wird das Betragshistogramm über den Winkeln errechnet.

In Aufgabe 4 werden die Kanten auf einen Punkt breite Linien reduziert

Im vorbereiteten Programmrahmen wird als Quellbild ein Farbbild `rgb` ermittelt und in ein Luminanzbild `src` mit `double`-Pixeln gewandelt.

Bereiten Sie alle Aufgaben vor dem Praktikumstermin gründlich vor und erstellen Sie soweit wie möglich schon vor dem Praktikumstermin Programmcode.

Version: April 2014, geändert M. Weinhardt

Aufgabe 1: Sobel-Gradientenbild mit Steigungen in x- und y-Richtung errechnen

Im Programmrahmen wird eine Funktion `compute_Sobel_xy` aufgerufen, die aus dem Luminanzbild `src` ein Gradientenbild `sobel_xy` aus x- und y-Ableitung pro Pixel nach dem Sobel-Algorithmus berechnet und zurückgibt. Jedes Pixel des Sobelbilds besteht dann aus einem `pair<double,double>`-Objekt, in dessen `first`-Komponente der Wert der x-Ableitung und in dessen `second`-Komponente der Wert der y-Ableitung abgelegt wird.

Anschließend werden die x- und die y-Ableitungen des Gradientenbilds getrennt und in zwei Bitmap-Bildern *sobel_x.bmp* und *sobel_y.bmp* ausgegeben.

⇒ Erstellen Sie den Code der Funktion `compute_Sobel_xy`.

⇒ Berechnen Sie Gradientenbilder aus verschiedenen Quellbildern und diskutieren Sie mit den Betreuern, ob die erzeugten Bitmap-Bilder korrekt sind.

Aufgabe 2: Umwandlung eines Gradientenbilds in die Polarkoordinatendarstellung

Das Gradientenbild `sobel_xy` aus Aufgabe 1 wird mit der Funktion `convert_Gradient_to_Polar` in ein Gradientenbild `sobel_mp` umgerechnet, bei dem die einzelnen Pixel von der xy-Darstellung in die Polarkoordinatendarstellung umgerechnet werden, in dessen `first`-Komponente der Betrag und in dessen `second`-Komponente der Winkel des Gradienten abgelegt wird.

Anschließend wird das Gradientenbild in einer Pseudofarbdarstellung ausgegeben, in welcher die Winkelwerte durch Farben und die Beträge durch die Farbintensität kodiert sind. Der Betrag 0 wird somit durch weiße Pixel dargestellt.

Weiterhin werden die Beträge und die Winkel des Gradientenbilds getrennt und in zwei Bitmap-Bildern *sobel_m.bmp* und *sobel_p.bmp* ausgegeben.

⇒ Erstellen Sie den Code der Funktion `convert_Gradient_to_Polar`.

⇒ Berechnen Sie Gradientenbilder aus verschiedenen Quellbildern und diskutieren Sie mit den Betreuern, ob die erzeugten Bitmap-Bilder korrekt sind.

Aufgabe 3: Berechnung eines Winkelhistogramms

Aus der Polarkoordinatendarstellung kann nun ein spezielles Histogramm berechnet werden, bei dem die Amplitudenwerte über den Winkeln aufsummiert werden. Dazu wird der Wertebereich [0..2·π) der Winkel in eine feste Anzahl von Winkelsegmente aufgeteilt, die mit aufsteigenden Winkelwerten durchnummeriert werden.

Es bietet sich an, die Winkel in 256 Segmente aufzuteilen. Die Segmentnummer seg zu einem Winkel φ errechnet sich dann wie folgt:

$$seg = \left\lfloor \frac{\varphi}{2 \cdot \pi} \cdot 256 \right\rfloor \%256$$

Dabei kennzeichnen die Klammern $\lfloor \ldots \rfloor$ die "Floor"-Funktion, die einen reellen Wert auf die nächstkleinere ganze Zahl abrundet. Weiterhin bezeichnet der Operator % die Modulo-Operation.

Zur Berechnung dieses Histogramms wird ein Feld $Hist$ mit der gewünschten Größe von 256 Segmenten als vector<double>-Container definiert und alle Elemente zunächst zu 0 gesetzt. Anschließend muss zu jedem Pixel des Gradientenbilds die Segmentnummer seg zum Winkel φ ermittelt, mit dieser das zugehörige Element des Vektors $hist[seg]$ ausgewählt und dazu der Betrag m des Gradienten akkumuliert werden:

$$hist[seg] = hist[seg] + m$$

In diesem Histogramm beschreibt der Eintrag $hist[seg]$ die Summe der Beträge aller Kantenpixel im Bild, deren Gradientenrichtung im Segment seg und somit im Winkelintervall $seg \cdot 2 \cdot \frac{\pi}{256} \leq \varphi < (seg + 1) \cdot 2 \cdot \pi/256$ liegt.

Im vorgegebenen Programmrahmen erfolgt die Berechnung des Winkelhistogramms mit der Funktion `create_AngleHistogram`. Ihr wird ein Gradientenbild in Polarkoordinatendarstellung übergeben und es liefert das daraus errechnete Winkelhistogramm zurück.

Im Hauptprogramm wird das errechnete Histogramm in einem Bitmap-Bild dargestellt und in der Datei *HistImg.bmp* abgespeichert.

⇒ Erstellen Sie den Code der Funktion `create_AngleHistogram`.

⇒ Berechnen Sie Winkelhistogramme aus verschiedenen Quellbildern und diskutieren Sie, welche Kanten im Bild Maxima im Histogramm erzeugen.

Aufgabe 4: Berechnung eines NMS-Bildes (optional)

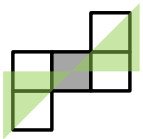
Innerhalb der Kanten des Bilds sollen nun noch die Positionen mit der stärksten Steigung gesucht werden. Diese findet man, wenn man die Nachbarpixel in Gradientenrichtung untersucht. Sind die Beträge der Gradienten aller dieser Nachbarn kleiner als der Betrag der Ableitung des aktuellen Pixels, weist er die stärkste Steigung in seiner Umgebung auf und sein Gradient wird in das Ergebnisbild übernommen. Besitzt einer der betrachteten Nachbarpixel einen größeren Betrag, wird der Gradient im Ergebnisbild zu 0 gesetzt. Dieses Vorgehen wird als "Non Maximum Supression" (NMS) bezeichnet.

Zur konkreten Berechnung eines NMS-Bilds teilt man den Gradientenwinkel φ in 8 Segmente ein:

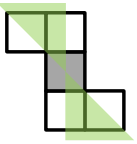
$$seg_{NMS} = \left\lfloor \frac{\varphi}{2 \cdot \pi} \cdot 8 \right\rfloor \%8$$

In Abhängigkeit vom Segment ergeben sich für den grau hinterlegten, aktuellen Pixel folgende Nachbarpixel:

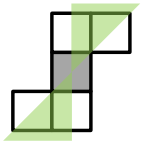
Segmente 0 und 4:
(0..π/4
und π..5·π/4)



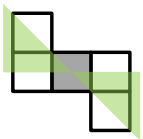
Segment 2 und 6:
(2·π/4.. 3·π/4
und 6·π/4..7·π/4)



Segment 1 und 5:
(π/4..2·π/2
und 5·π/4..6·π/4)



Segment 3 und 7:
(3·π/4.. π
und 7·π/4..2·π)



Im vorgegebenen Programmrahmen erfolgt die Berechnung des NMS-Bilds mit der Funktion `create_NMS_Gradient`. Ihr wird ein Gradientenbild in Polarkoordinaten-darstellung übergeben und es liefert das daraus errechnete NMS-Bild zurück.

Im Hauptprogramm wird das errechnete NMS-Bild in einem Bitmap-Bild in Pseudofarbdarstellung gewandelt und in der Datei *Sobel_nms.bmp* abgespeichert.

⇒ Erstellen Sie den Code der Funktion `create_NMS_Gradient`.

⇒ Berechnen Sie das NMS-Bild aus verschiedenen Quellbildern und inspizieren Sie die Linien mit stärkster Kantensteigung.