Bildverarbeitung SoSe 2012
Übung 8

Max Michels, Sebastian Kürten

1 Aufgabe 1

Der Canny-Kantenerkennungsalgorithmus arbeitet in etwa folgendermaßen:

- Anwenden eines Gauß-Weichzeichners mittels Konvolution. Dabei tritt einer der Konfigurationsparameter auf, nämlich die Größe des Gauss-Filters.
- Dann werden mit verschiedenen Verfahren Gradienten ermittelt um Kanten in verschiedenen Ausrichtungen zu verstärken.
- Es werden lokale Nicht-Maxima verworfen
- Auf Basis zweier Schwellwerte werden alle Pixel als Kante, nicht-Kante oder zwischen-Kanten klassifiziert.
- Alle zwischen-Kanten Pixel werden erneut überprüft, ob diese an Kantenpixel grenzen, und im diesem Fall zu Kantenpixeln umklassifiziert.

Wir haben zufällig Konfigurationsparameter für t und σ durchprobiert, bis sinnvolle Ergebnis erreicht wurden. Jedoch konnte keine mit der Musterlösung vergleichbare Konfiguration gefunden werden, die genauso glatte Kanten produziert. Als geeignete Parameter wurden, z.B. $t = [2, 9]$ und $\sigma = 0.3$ identifiziert. Scheinbar sind die Ergebnisse aber auch von der verwendeten Octave-Version abhängig.

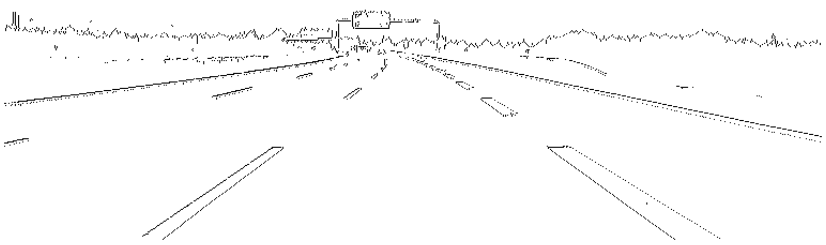

Figure 1: Canny Kantenerkennung (invertiert)

2 Aufgabe 2

Die Hough-Transformation wurde wie folgt implementiert. Sie erhält als Parameter ein Eingabebild und die Anzahl an Werten in die der Winkel diskretisiert werden soll. Hier ist zum Beispiel 512 zu wählen. Der Koordinatenursprung wurde an der linken oberen Ecke des Bildes belassen. (Wir haben ihn auch mal testweise in die Mitte des Bildes verschoben)

Listing 1: Hough-Transformation

```
function O = hough(I, alphasteps)
% create a hough transform of the input image
    [h, w] = size(I)
    % maximal distance possible (diagonal)
    maxd = ceil(sqrt(h*h+w*w))
    % resulting hough space
    O = zeros(maxd, alphasteps);

    % discrete angles
    steps = [1:alphasteps];
    radians = steps / alphasteps * pi;

    % find non-zero entries
    [bys,bxs] = find(I);

    % move the origin to the center of the image
    %xs = bxs - w / 2;
    %ys = bys - h / 2;

    % keep the origin at the top left corner of the image
    xs = bxs;
    ys = bys;

    for p = 1:size(xs, 1)
        x = xs(p);
        y = ys(p);
        % calculate values for d
        ds = abs(round(x * cos(radians) + y * sin(radians)));
        % select only valid alpha positions
        alphas = find(ds > 0 & ds < maxd);
        % recalculate index of the (alpha,d) entry
        i  = (alphas-1) * maxd + ds(alphas);
        % cumulate values
        O(i) += 1;
    end;
end;
```

Die Hauptdatei 'a2.m' berechnet für das Eingabebild die Hough-Transformation, entfernt daraus alle lokalen Nicht-Minima, wählt dann aus dem Akkumulatorbild alle Punkte, die den Schwellwert 80 überschreiten aus und zeichnet die Geraden, die durch die entsprechenden Werte d und α an der jeweiligen Stelle gegeben sind in das ursrprüngliche Bild ein. Zwischendurch werden Bilder produziert.

Listing 2: Hauptdatei a2.m

```
B = imread("lanes-bw.png");
I = imread("lanes-bw-canny.png");

% discretetize angle with 'alphasteps' steps
alphasteps = 512;
% perform hough transformation
O = hough(I, alphasteps);

```

```
% create an image of the result
O1 = O;
O1(find(O1 > 255)) = 255; % cut off too big values
imwrite(uint8(255-O1), "lanes-hough.png");

% supress non maxima
O2 = nms(O);

% create another image of the local maxima
O3 = O2;
O3(find(O3 > 255)) = 255; % cut off too big values
imwrite(uint8(255-O3), "lanes-hough-nms.png");

% find global maxima with thresholding
[ds,alphas] = find(O2 > 80);

% draw lines on the original image
imshow(B);
hold on;
[h,w] = size(B);
alphas = alphas / alphasteps * pi;
ys = ds./sin(alphas);
yws = (ds-w*cos(alphas))./sin(alphas);
xs = ds./cos(alphas);
n = size(ys,1);
for k = 1:n
    x = [0 w];
    y = [ys(k) yws(k)];
    plot(x, y);
end;
print("lanes-bw-result-own.png");
```

Listing 3: Non-maxima supression nms.m

```
function O = nms(I)
% non maximum supression
    [h, w] = size(I);
    O = zeros(h, w);
    I = padZero(I);
    for y = 2:h-1
        for x = 2:w-1
            P = I(y-1:y+1,x-1:x+1);
            m = max(P(:));
            if I(y,x) ≥ m
                O(y-1,x-1) = I(y,x);
            end;
        end;
    end;
end;
```

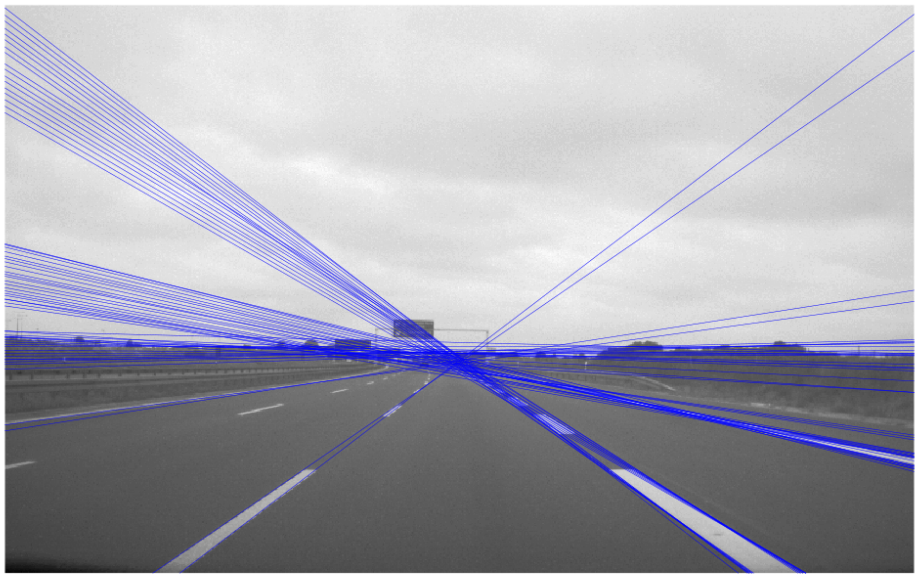

Figure 2: Originalbild mit eingezeichneten Geraden

Figure 3: Hough-Raum bei Koordinatenursprung oben links

Figure 4: Hough-Raum bei Koordinatenursprung mitte