Institut für Informationsverarbeitung (TNT)
Appelstraße 9a, 30167 Hannover
www.tnt.uni-hannover.de

Prof. Dr.-Ing. Bodo Rosenhahn
Felix Winkler, M.Sc.

WiSe 2022/23 **25.01.2023**

Praktische Übung: Matlab für die medizinische und industrielle Bildinterpretation

Versuch 11: Tracking in Videos

Aufgabe 1 - Block Matching

Beim Block Matching wird zu einem Bildbereich

$$I_t^w(x,y) = \begin{pmatrix} I_t(x-w, y-w, :) & \cdots & I_t(x-w, y+w, :) \\ \vdots & \ddots & \vdots \\ I_t(x+w, y-w, :) & \cdots & I_t(x+w, y+w, :) \end{pmatrix}$$

in einem Frame I_t der ähnlichste Bereich im folgenden Frame I_{t+1} gesucht, um einen Punkt zu verfolgen. Die Suche nach einem passenden Bildbereich ist aufwendig, da die Bewegungsrichtung der Kamera und des Objekts im Allgemeinen nicht bekannt ist. Diverse Verfahren versuchen das Bild möglichst effektiv zu durchsuchen.

a) Schreiben Sie eine Funktion `TDLS(M, xStart, yStart, w)`, die das Verfahren „*Two Dimensional Logarithmic Search*“ umsetzt. Im Video $M = \{I_t\}_{t=1}^n$ wird der Punkt (x_{start}, y_{start}) verfolgt, wobei Bildbereiche mit der Fenstergröße w verglichen werden. Als Distanzmaß bietet sich z.B. die Summe der quadrierten Differenzen über die drei Farbkanäle an:

$$SSD(I_t^w(x,y), I_{t+1}^w(\tilde{x},\tilde{y})) = \sum_{i,j=-w}^{w} \sum_{c=1}^{3} \left(I_t^w(x+i, y+j, c) - I_{t+1}^w(\tilde{x}+i, \tilde{y}+j, c)\right)^2$$

Um die Suche effizient zu gestalten, werden iterativ Bildbereiche im Abstand S verglichen, wobei der Abstand schrittweise verkleinert wird. Sie können sich bei der Implementierung an folgenden Pseudocode halten:

Input: Video $M \in \{0, \ldots, 255\}^{m \times n \times c \times t}$
Input: Startpunkt (x_{start}, y_{start})
Input: Fenstergröße w
Output: Verfolgter Pfad p

Initialisiere Pfad: $p_1 \leftarrow (x_{start}, y_{start})$
for $i = 2, \ldots, t$ **do**
    $I_{old} \leftarrow M_{i-1}$
    $I \leftarrow M_i$
    Initialisiere Pfad im Frame $p_i \leftarrow p_{i-1}$
    Initialisiere Schrittweite $s \leftarrow 8$
    while $s > 1$ **do**
        Vergleiche alte Region $I^w_{old}(p_{i-1})$ mit neuem Frame (Mittelpunkt und vier Nachbarblöcke: $I^w(p_i + S),\ S \in \{(0,0), (s,0), (-s,0), (0,s), (0,-s)\}$)
        Verschiebe Mittelpunkt auf den ähnlichsten Block $p_i \leftarrow p_i + \hat{S}$
        if keine Verschiebung ($\hat{S} = (0,0)$) **then**
            Halbiere Schrittweite $s \leftarrow s/2$
        end if
    end while
    Vergleiche alte Region $I^w_{old}(p_{i-1})$ mit allen Nachbarblöcken: $I^w(p_i + S),\ S \in \{-1, 0, 1\}^2$
    Verschiebe Mittelpunkt auf den ähnlichsten Block $p_i \leftarrow p_i + \hat{S}$
end for

b) Testen Sie ihr implementiertes Verfahren (siehe `tracking.m`).

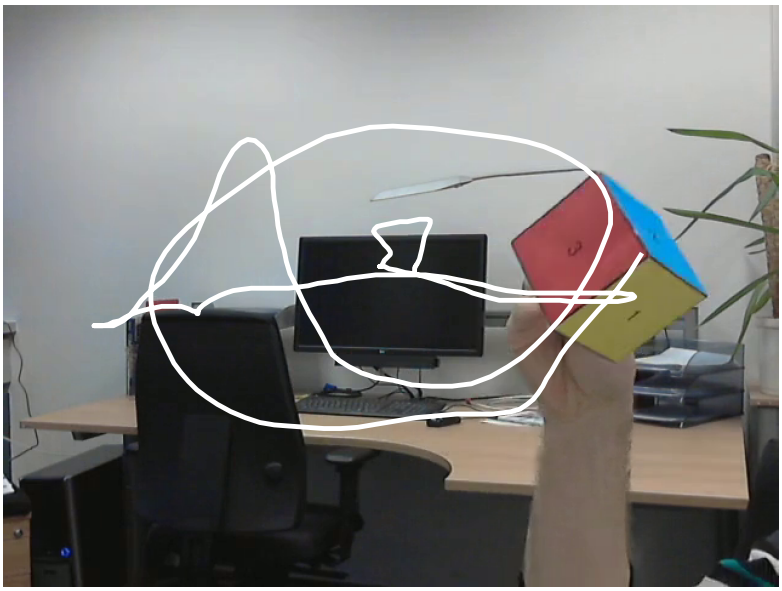

Abbildung 1: Trajektorie der zugewandten Ecke des Würfels

Aufgabe 2 - Partikelfilter

Im Gegensatz zum Blockmatching wird ein Objekt mit mehreren Punkten (Partikel) gleichzeitig verfolgt. Die Partikel werden auf der Basis eines angenommenen oder gelernten Bewegungsmodells verschoben und die Ähnlichkeit zum gesuchten Objekt bestimmt. Je ähnlicher der Partikel, desto wahrscheinlicher wird der Partikel im Verlauf weiter verfolgt.
Das Tracking mit einem Partikelfilter umfasst damit die folgenden Schritte:

- Verschiebung der Partikel mit einem Bewegungsmodell
- Evaluierung der verschobenen Partikel
- Resampling der Partikel (neue Partikel werden auf der Basis der Ähnlichkeit gezogen)

Im Folgenden soll die Funktion *particleFilterTracking.m* vervollständigt werden, bei der ein Objekt anhand der Farbe verfolgt werden soll.

a) Vervollständigen Sie die Funktion `updateParticles(X, V)`, die die Partikel geeignet verschiebt. Hier beschreibt X die Position der Partikel und V gibt die Geschwindigkeit der Partikel an. Zusätzlich wird üblicherweise normalverteiltes Rauschen zu der Bewegung addiert. Damit gilt

$$X_{t+1}^i = X_t^i + V_t^i + n_x \qquad \text{mit} \qquad n_x \sim \mathcal{N}(0, \sigma_x^2)$$

für alle Partikel $i = 1, \ldots, n$. Zusätzlich wird die Geschwindigkeit der Punkte angepasst

$$V_{t+1}^i = \frac{1}{2}\left(V_t^i + (X_{t+1}^i - X_t^i)\right) + n_v \qquad \text{mit} \qquad n_v \sim \mathcal{N}(0, \sigma_v^2).$$

b) Vervollständigen Sie die Funktion `resampleParticles(X, V, L)`, die zufällig aus der Verteilung der Partikel neue Partikel zieht. Je größer die Likelihood der Partikel, desto wahrscheinlicher sollen diese gezogen werden. Bestimmen Sie dazu die Verteilungsfunktion F der normierten Likelihoods

$$F(i) = \frac{\sum_{j=1}^{i} L^j}{\sum_{j=1}^{n} L^j}.$$

Ziehen Sie jetzt aus dieser Verteilungsfunktion die neuen Partikel, indem Sie n zufällige Zahlen $z^i \in [0, 1]$ aus der Gleichverteilung ziehen und auf F abbilden, d.h. für die Zufallszahl z_i ziehen Sie den Partikel j genau dann, wenn

$$z^i > F(j-1) \wedge z^i \leq F(j).$$

c) Testen Sie ihr implementiertes Verfahren (`tracking.m`).

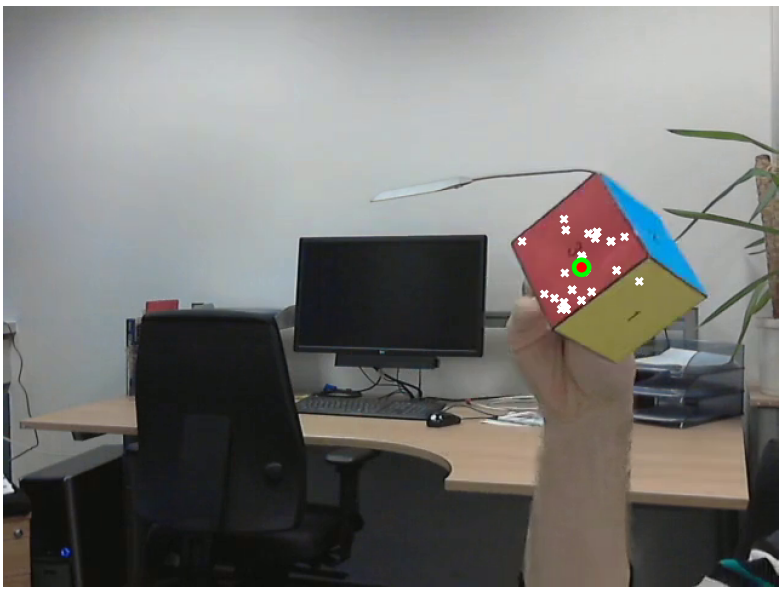

Abbildung 2: Tracking der roten Würfelseite mit dem Partikelfilter

d) Der implementierte Tracker verfolgt momentan rote Bereiche im Bild. Dazu wird zur Bestimmung der Likelihood der Abstand zu Rot bestimmt und der Abstand als normalverteilt mit Mittelwert $\mu = 0$ und einer zu wählenden Varianz σ_c^2 angenommen. Machen Sie sich mit der Bestimmung der Likelihood vertraut und passen Sie diese so an, dass eine beliebige Farbe verfolgt wird, die zunächst im ersten Frame angeklickt wird.

e) Testen Sie beide Verfahren auch mit Hilfe der Webcam (`trackingWebcam.mlapp`). Falls notwendig, passen Sie die Parameter der Verfahren an. Sind die beide Methoden echtzeitfähig?

Kontrollfragen

a) Nennen Sie die wesentlichen Bestandteile beider Tracker.

b) Welche Probleme treten jeweils bei den Verfahren auf? Wie können diese verhindert werden?

c) Nennen Sie Vor- und Nachteile beider Verfahren.