

Farbbasierte Verfolgung

Sehseminar: Farbwahrnehmung und Farbbildverarbeitung

Sommersemester 2007

Matthias Schneider
Universität Ulm
89069 Ulm, Germany
matthias.schneider@uni-ulm.de

1. MOTIVATION

Verfolgung von Objekten ist ein wichtiger Bestandteil in vielen Anwendungen der Computer Vision. Darunter fallen beispielsweise Fahrassistenzsysteme, Überwachung, intelligente Videokompression, medizinische Anwendungen wie Gebärdenspracherkennung und Mensch-Maschine Systeme wie etwa *perceptual user interfaces*. Das wichtigste Merkmal eines guten *Tracking-Algorithmus* ist Robustheit gegenüber Verdeckungen, Farb- und Helligkeitsänderungen sowie Störungen im Bild.

Die Verfahren die hier besprochen werden, können grob in zwei Klassen aufgeteilt werden: Verfahren die Farbinformationen verwenden und Verfahren die Farbe nicht als Merkmal verwenden. Der Hauptaugenmerk dieser Ausarbeitung liegt auf farbbasierten Verfahren.

Ansätze, die auf Farbinformationen verzichten, verwenden wiederum verschiedene Verfahren zur Erkennung und Verfolgung von Objekten. Diese sind jedoch in Hinsicht auf ihre Robustheit oft auf bestimmte, eng umrissene Szenarien beschränkt. Daher muss ein weiteres Merkmal herangezogen werden, das diese Schwäche ausgleicht.

Farbbasierte Verfahren können auf eine Eigenschaft der Farbe zurückgreifen, die bei *Tracking*-Verfahren erheblich bessere Ergebnisse erzielt: Bei Beobachtungen über einen längeren Zeitraum führt sie im Gegensatz zu Kanten, Ecken oder anderen Merkmalen zu einer besseren Stabilität.

Im Folgenden werden zunächst Verfahren, die ohne Farbe operieren, besprochen. Anschließend erfolgt eine ausführliche Besprechung der farbbasierten Verfahren.

2. VERFAHREN OHNE FARBE

Nach [1] kann man Verfahren ohne Farbmerkmale in drei Klassen unterteilen.

A priori Wissen über die Form des Objektes wird verwendet, beispielsweise das Heck eines Autos. Damit lassen sich jedoch nur Objekte verfolgen, die ihre Form nicht grundlegend ändern und damit auf Szenarien wie etwa die Erkennung von anderen Fahrzeugen auf Autobahnen beschränkt sind.

Stereoinformationen, generiert durch die Analyse korrespondierender Bilder, werden verwendet um über Tiefeninforma-

tionen auf die Form des Objektes schließen zu können. Hier entsteht aber oft das Problem der *sparese depth map*, also eine zu spärlich gefüllte Karte mit Tiefeninformationen. Dadurch kann eine erfolgreiche Erkennung eines Objektes erschwert bzw. nicht erreicht werden.

Die dritte Klasse erkennt Objekte anhand von Informationen über ihre Bewegung. Methoden mit einer stationären Kamera evaluieren den Unterschied zwischen neuen Frames und dem statischen Hintergrund. Jedoch sind Verfahren, die nicht-stationäre Kameras erlauben, hier von größerer Bedeutung. Es wird zwischen kontinuierlichen und diskreten Methoden unterschieden.

Kontinuierliche Methoden sind stark von der Stetigkeit des Verschiebungsvektorfeldes abhängig. Unstetigkeiten verhindern hier eine verlässliche Erkennung der Bewegung.

Diskrete Methoden hingegen stützen sich auf die Detektion von Kanten, Ecken oder lokale Intensitätsmaxima oder -minima. Diese *Features* werden einander in aufeinander folgenden Bildern zugeordnet und daraus Informationen über die Bewegung berechnet. Die hier entstehenden Vektorfelder sind zwar sehr genau, oft jedoch nicht dicht genug um ausreichend Objekte zu extrahieren.

3. VERFAHREN MIT FARBE

Die Verfahren *Color Blob Flow* und *Color Cluster Flow* von Heisele & Ritter [2][1], die eine einfache Farbsegmentierung als Basis für die Verfolgung verwenden, werden zuerst besprochen. Neuere Methoden wie das *mean-shift*-Verfahren bzw. *Kernel-based Tracking* werden anschließend behandelt.

3.1 Color Blob Flow

Ähnlich wie bei diskreten Methoden, die aus den Eigenschaften eines Bildes ein Vektorfeld berechnen welches die Bewegung beschreibt (*optical flow*), stützt sich das CBF (*Color Blob Flow*) Verfahren auf die Berechnung eines Verschiebungsvektors für je einen Farbbereiche (*color blob*). Auf diese Weise lässt sich das Problem stark vereinfachen, da nur noch wenige solcher Vektoren berechnet und Bereiche zugeordnet werden müssen.

Der erste Schritt zur Berechnung des *Color Blob Flow* ist die Segmentierung der Farbe. Hier wird auf ein Algorithmus[3] zurückgegriffen, der einen guten Ausgleich zwischen Berechnungszeit und Qualität der Quantisierung bietet. Die un-

terschiedlichen Farbbereiche werden bestimmt, indem eine feste Anzahl an Referenzvektoren $(r, g, b)^T$ bestimmt wird, die die Farbverteilung im Bild optimal repräsentieren. Jeder Bildpunkt des Bildes wird nun unter Verwendung einer Abstandsmetrik (hier: Euklidischer Abstand) durch den am nächst gelegenen Referenzvektor ersetzt. Wichtig ist, daß die Anzahl der Farbbereiche nicht zu hoch gewählt wird, da sonst viele verschiedene Farbbereiche für eigentlich zusammenhängenden Oberflächen von Objekten als Repräsentanten entstehen.

Als nächster Schritt werden die Farbbereiche, die bisher nur aus Pixeln gleicher Farbe bestehen, zu zusammengehörigen Flächen zusammengefasst. Der hier eingesetzte Algorithmus bestimmt in einem Durchlauf alle wichtigen Merkmale dieser Flächen: Die genaue Umrandung der einzelnen Farbbereichen, darauf aufbauend den Flächeninhalt und Schwerpunkt, sowie die rechteckige Umrandung (*Bounding-Box*). Damit sind alle topografischen Eigenschaften der Farbbereiche bestimmt und können im nächsten Schritt weiterverwendet werden.

Im Folgenden wird nun eine Technik entwickelt, die Farbbereiche in zwei aufeinander folgenden Bildern korrekt zuordnet (*matching*). Die im vorhergehenden Schritt erlangten Daten können hier nun verwendet werden. Da Farbe, wie schon angesprochen, meistens unabhängig von der Bewegung des Objektes ist, eignet sie sich besonders gut zur Bestimmung von zusammengehörigen Farbbereichen. Diese Eigenschaft alleine ist aber nicht ausreichend. Daher ziehen Heisele & Ritter die Eigenschaften Flächeninhalt und Seitenverhältnis der *Bounding-Box* des jeweiligen Farbbereiches hinzu. Dadurch können noch stabilere Ergebnisse erzielt werden, vorausgesetzt die Abstände zwischen den Einzelbildern sind hinreichend klein, da ansonsten die beiden letzteren Eigenschaften durchaus ihre Ausprägungen zu stark ändern können.

Für die Berechnung des Unterschieds zwischen zwei Farbbereichen werden folgenden Gleichungen verwendet:

Farbunterschied:

$$D_{C_{AB}} = \|w_A - w_B\| \text{ mit } w = (r, g, b)^T$$

Unterschied im Flächeninhalt:

$$D_{F_{AB}} = \left\| \frac{F_A - F_B}{F_A} \right\|$$

Unterschied von Seitenverhältnissen:

$$D_{S_{AB}} = \left\| \frac{S_A - S_B}{S_A + S_B} \right\| \text{ mit } S = \frac{\text{Breite}}{\text{Höhe}}$$

Die gesamte Abweichung lässt sich nun, abhängig von den Gewichtungsfaktoren w_i , mit

$$D_{AB} = w_C D_{C_{AB}} + w_F D_{F_{AB}} + w_S D_{S_{AB}}$$

berechnen. Die Faktoren müssen experimentell je nach Einsatzort (Autobahn, Stadtverkehr etc.) bestimmt werden.

Ob nun zwei unterschiedliche Farbbereiche in zwei aufeinander folgenden Bildern einander zugeordnet werden können, wird über die Gleichung

$$Z_{AB} = \min\{D_{AX} | X \in \Phi, \|g_A - g_X\| \leq d\}$$

bestimmt. Es wird also ein Farbbereich A in Bild P mit einem Farbbereich B in Bild Q assoziiert, wenn der Abstand der beiden Schwerpunkte g kleiner ist als alle anderen Abstände zwischen A und einem beliebigen Farbbereich X aus der Menge Φ aller Farbbereiche aus Q. d wird dabei wieder je nach Einsatzort bestimmt.

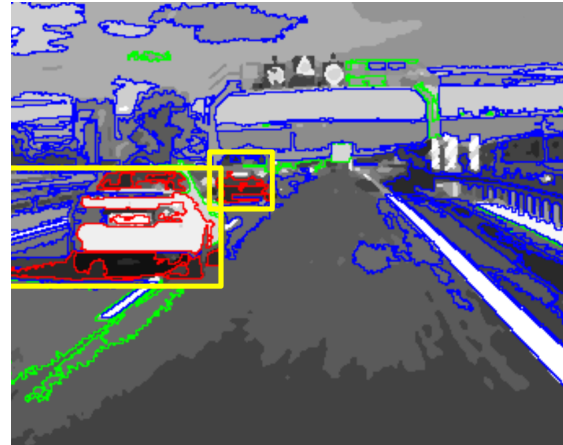


Figure 1: Verkehrsszene mit Color Blob Flow. Blau umrandete Bereiche entsprechen erfolgreich verfolgte Bereiche. Bereiche mit gleicher Bewegung sind gelb markiert. Quelle: [1]

Dies sagt aber noch nicht aus, ob diese Farbbereiche auch die gleiche Bewegung haben. Nur wenn sie diese Bedingung erfüllen, und dies auch über eine gewissen Zeitspanne (d.h. mehrere Bilder pro Sekunde) aufrecht erhalten können, kann man die einzelnen Bewegungen im Bild erfolgreich segmentieren.

Das Maß für gleiche Bewegung wird bestimmt, indem die Korrelation von zwei Schwerpunkten über eine ausreichend große Zeitspanne (in der Praxis meist > 6 Frames) beobachtet wird. Wenn nun alle der folgenden Bedingungen erfüllt werden, kann man auf gleiche Bewegung und damit gleiche Objekte schließen:

- $korrel_{AB} > korrel_{min}$ und B überlappt mit A,
- $T_{AB} > T_{min}$, mit T als Zeit der Beobachtung,
- $\min(p_A, p_B) > p_{min}$ mit p als Pfadlänge der Schwerpunkte während der Beobachtung,

wobei der Index *min* den jeweils kleinsten Wert der entsprechenden Variable während der gegebenen Zeitspanne angibt.

Mit diesem Verfahren wird auch der Rechenaufwand bedeutend verringert, da jeweils nur komplette Farbbereiche verfolgt werden müssen. Für die Anzahl der Farbbereiche hat sich laut Heisele & Ritter ein Wert von 16 für die meisten Szenen als passend herausgestellt.

Dieses Verfahren bildet die Basis für das im kommenden Abschnitt vorgestellte Verfahren des *Color Cluster Flow*.

3.2 Color Cluster Flow

Das Verfahren des CBF zeigt jedoch Schwächen, wenn nicht-starre Objekte verfolgt werden sollen. Ein Fußgänger oder ein Radfahrer ändert beispielsweise seine Form ständig durch Eigenbewegung. Dadurch ändern sich Flächeninhalt sowie Seitenverhältnis der Farbbereiche von Bild zu Bild zu stark um weiterhin erfolgreiches *Matching* zu gewährleisten.

Das Verfahren *Color Cluster Flow* ermöglicht es auch nicht-starre Objekte erfolgreich zu verfolgen.

Das ursprüngliche Bild wird hierbei in Cluster unterteilt, die Pixel ähnlicher Farbe und ähnlicher Position enthalten. Jeder dieser Cluster wird durch dessen Schwerpunkt im Farb- und Merkmalsraum repräsentiert und im Folgenden Prototyp des Clusters genannt. Dieser Prototyp wird in jedem darauffolgenden Bild neu berechnet und passt somit die Cluster automatisch der neuen Situation im Bild an. Somit ist im Gegensatz zur CBF Methode kein explizites Tracking nötig. Eine Anzahl von $k = 64$ Cluster hat sich als passend für Verkehrsszenen herausgestellt.

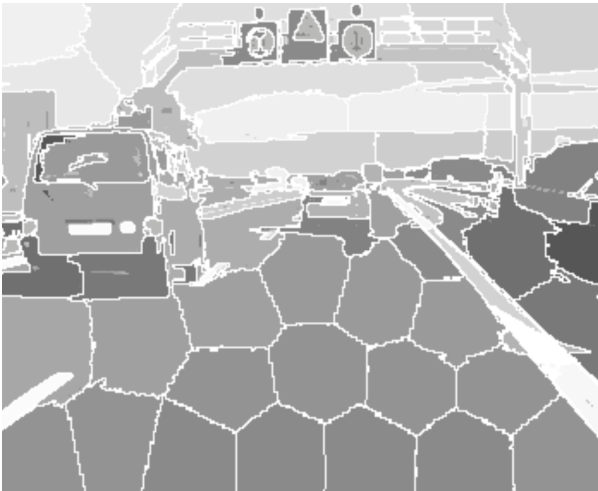


Figure 2: Resultat des Clustervorgangs (64 Prototypen). Quelle: [2]

Der erste Clustervorgang einer Szene wird mit einem Algorithmus berechnet [4], welcher mit einem Cluster beginnt, der die gesamte Fläche abdeckt. In jedem Schritt wird nun der Cluster geteilt, der die höchste Abweichung zum vorherigen Bild zeigt. Die Teilung wird durchgeführt, indem eine Gerade durch den Cluster gezogen wird, die senkrecht zur Richtung der Abweichung steht und durch den Schwerpunkt (Prototypen) des Cluster führt.

Wie sich in Abbildung 2 zeigt, werden große Flächen zu ka-

chelförmigen Einheiten zusammengefasst, während kleinere Details wie zum Beispiel Teile des Autos deren Form widerspiegeln.

Für jedes weitere Bild dienen nun die Prototypen des vorherigen Bildes als *Seed* des *k-means*-Clusteralgorithmus[3], der die neuen Prototypen berechnet. Im ersten Schritt des Algorithmus wird jeder Pixel f_{ij} des neuen Bildes dem Cluster $C_k(t+1)$ zugeteilt, der dem Prototyp $p_k(t)$ am nächsten ist. Danach wird der neue Prototyp p_k anhand des neu gebildeten Clusters $C_k(t+1)$ berechnet. Dies geschieht durch Bestimmung des Mittelwerts aller Pixel in diesem Cluster. Dieser Wert wird dann als neuer Prototyp verwendet.

Heisele et. al konnten durch Evaluierung von verschiedenen Straßenszenen zeigen, dass ein einziger Durchlauf des Algorithmus ausreicht um die Prototypen korrekt an die neue Situation im Bild anzupassen.

Eine Schwäche des CCF Verfahrens tritt auf, wenn das zu verfolgende Objekt zu stark von einem anderen Objekt verdeckt wird bzw. es für kurze Zeit sogar komplett hinter diesem verschwindet. Es ist dann keine Korrespondenz mehr zwischen dem alten und dem wieder erscheinenden Objekt herzustellen.

Verbesserungen versprechen Verfahren, die sich auf die Analyse des Merkmalsraums eines Bildes stützen. Das *Mean-Shift* Verfahren, das im folgenden Abschnitt vorgestellt wird, ist ein solches Verfahren.

3.3 Mean-Shift Verfahren

Beim *Tracking* von Objekten wird oft nicht direkt auf dem Bild bzw. auf den Bildsequenzen gearbeitet, sondern es werden die für die Verfolgung interessanten Merkmale wie Farben, Kanten, Ecken usw. in einen Merkmalsraum projiziert und dieser anschließend nach Verteilungen bzw. Häufigkeiten analysiert. Hier ist es notwendig, dass der Merkmalsraum zuvor geglättet wurde. Da es sich bei den verschiedenen Merkmalen eines Bildes meistens um jeweils auch verschiedene Verteilungsarten handelt, wird die Glättung mithilfe eines Kernel-Dichte-Schätzers[5] durchgeführt. Damit können beliebige Verteilungen abgeschätzt werden, da a priori keine Annahmen über die Verteilung vorgenommen werden müssen. Dadurch eignet sich dieses Verfahren besonders für Echtzeitsysteme. Der Kernel repräsentiert hier einen mehrdimensionalen Bereich innerhalb des Merkmalsraums. Dieser Bereich wird durch einen Punkt x im Zentrum und n -viele Beobachtungen in einem gewissen Radius um dieses Zentrum herum beschrieben. Der Radius wird auch "Bandbreite" des Kernels genannt.

Die Aufgabe des *Mean-Shift* Verfahrens[6] ist nun, in diesem Merkmalsraum nach Maxima zu suchen, welche die zu beobachtenden Merkmale durch z.B. bekannte Verteilungen oder Unstetigkeiten (d.h. Kanten) repräsentieren.

Allgemeine gradientenbasierte Verfahren berechnen zuerst den Gradienten und verschieben danach den Kernel in Richtung eines Maximums der Verteilung. Die Länge dieser Verschiebung bzw. die Größe dieses Vektors zu bestimmen ist eine Problematik bei herkömmlichen Verfahren. Wird der Vektor zu lang gewählt, kann keine Verfolgung mehr durch-

geführt werden. Ist der Vektor zu kurz, resultiert dies in einem zu hohen Berechnungsaufwand. Beim *Mean-Shift* Verfahren muss diese "Schrittgröße" nicht explizit bestimmt werden. Sie wird automatisch anhand der lokalen Steigung der Verteilung angepasst. Auch der Gradient muss beim *Mean-Shift* Verfahren nicht berechnet werden, da der *Mean-Shift* Vektor nachweisbar in Richtung des Gradienten verläuft.

Die einzelnen Schritte des *Mean-Shift* Verfahrens sind also wie folgt:

- Wahl der Bandbreite des Kernels
- Wahl der Startposition des Kernels
- Berechnung des *Mean-Shift* Vektors und Verschiebung des Kernels in diese Richtung
- Wiederholung bis Maximum erreicht

Beim *Tracking* mit *Mean-Shift* wird nun über ein Maximum (also ein Merkmal des Bildes, z.B. Farbe oder Kanten) im Merkmalsraum nach ähnlichen Kandidaten gesucht. Dabei werden zwei Modelle bestimmt: Das Zielmodell, das das zu verfolgende Objekt beschreibt und viele Kandidatenmodelle, die sich in jedem neuen Bild ergeben und mit dem Zielmodell verglichen werden. Kandidaten, die eine starke Korrelation zum Zielmodell aufweisen, sind mit hoher Wahrscheinlichkeit in der Nähe der Position des Zielmodells platziert.

Comaniciu et al. nehmen in [5] und [6] an, dass sich das zu verfolgende Objekt zwischen zwei aufeinanderfolgenden Bildern nur wenig verschiebt. Daher verwenden sie als Startposition für die Suche des Maximums im aktuellen Bild die Maximumposition im vorhergehenden Bild.

3.4 CamShift Verfahren

Das *CamShift* Verfahren (*Continuously Adaptive Mean Shift*) wird in [7] vorgestellt. Er basiert auf *Mean-Shift*, kann jedoch mit sich ändernden Farbverteilungen umgehen. Dies ist beim *Mean-Shift* Verfahren nur bei großen Farb- und Größenunterschieden möglich. Das Verfahren stützte sich in seiner ursprünglichen Form auf zweidimensionale Rot-Grün Histogramme, die jedoch wegen ihrer starken Abhängigkeit der Beleuchtung durch den *Hue*-Wert das HSV-Farbraum ersetzt wurden. Der *Hue*-Wert gibt nur den Farbwert wieder und ist unabhängig von der Sättigung und der Helligkeit.

Der Algorithmus findet vor allem Einsatz beim *Tracking* von Gesichtern. Hier wird ein Histogramm des *Hue*-Werts berechnet und die Verteilungsdichte der Farbwerte betrachtet, die dem Hautfarbton am ähnlichsten sind. Dies ist aus dem Grund erfolgreich, weil die verschiedenen Hautfarbwerte ähnliche *Hue*-Werte zeigen.

CamShift ist darüber hinaus sehr robust gegen Rauschen, da randomisiertes Rauschen mit einer geringen Wahrscheinlichkeit die Farbwerte von Haut annimmt. Experimente haben gezeigt, dass Szenen die zu einem Drittel aus Rauschen bestehen, noch erfolgreiche Resultate lieferten.

Schwierigkeiten mit diesem Verfahren entstehen beispielsweise, wenn der *Hue*-Wert nicht ausreicht, um das Objekt

von seinem Hintergrund zu trennen oder ein Objekt mehrere sehr verschiedene *Hue*-Werte aufweist.

4. ZUSAMMENFASSUNG

Die vorgestellten Verfahren bieten alle eine robuste Möglichkeit um Objekte über eine Folge von Bildern zu verfolgen. Die Hinzunahme des Merkmals Farbe ergibt bei allen farbbasierten Verfahren eine Erhöhung der Erkennungsrate und bietet sich somit als zentrales Merkmal zur Verfolgung an.

Die Verfahren *Color Blob Flow (CBF)* und *Color Cluster Flow (CCF)* wenden einfache mathematische Verfahren an um Objekte von einer nicht-stationären Kamera aus zu verfolgen. Dabei werden nur einzelne Farbbereiche statt Kanten oder Ecken verfolgt. CCF kann im Vergleich zu CBF auch mit nicht-starren Objekten wie Fußgängern oder Fahrradfahrern umgehen und erreicht eine höhere Performanz, da die zusammengehörigen Farbbereiche nicht explizit zugeordnet werden müssen. Beide Verfahren zeigen für ihre Einsatzgebiete ausreichende Ergebnisse, haben aber Schwächen bei starker Verdeckung und unerwartetem Verschwinden des Objektes.

Das *Mean-Shift* Verfahren hingegen kann auch Objekte, die für kurze Zeit komplett Verdeckt sind, erfolgreich verfolgen. Es analysiert einen Merkmalsraum, der die zu verfolgenden Merkmale wie z.B. Farbe enthält. Es ist mathematisch aufwändiger zu realisieren als CBF und CCF, bietet aber eine sehr gute Performanz und eignet sich daher für Echtzeitsysteme.

Das *CamShift* Verfahren basiert auf *Mean-Shift* und bietet sich besonders beim *Tracking* von Gesichtern an.

5. REFERENCES

- [1] Bernd Heisele and W. Ritter. Obstacle detection based on color blob flow. In *Proceedings Intelligent Vehicles Symposium '95*, pages 282–286, 1995.
- [2] Bernd Heisele, Ulrich Kressel, and W. Ritter. Tracking non-rigid, moving objects based on color cluster flow. In *Conference on Computer Vision and Pattern Recognition CVPR '97*, pages 257–260. IEEE Computer Society, 1997.
- [3] Xiaolin Wu and Ian H. Witten. A fast K-means type clustering algorithm. Technical Report 85/197/10, University of Calgary, June 1985.
- [4] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, January 1980.
- [5] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.
- [6] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.
- [7] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. In *OpenCV*, 1998.