Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Seminar Computer Vision im WS 2004/2005

Segmentation by Fitting a Model

Segmentierung durch Modellanpassung

Carsten Dolch

Mittwoch, den 22.12.2004

Segmentation by Fitting a Model

- Einführung und Motivation
- Linienanpassuung
- Kurvenanpassung
- Anpassung als probabilistisches Reduktionsmodell
- Robustheit
- Anwendungsbeispiel: automatische Verkehrsüberwachung
- Zusammenfassung

Einführung und Motivation

- häufig bekannt: welche Formen in einem Bild vorkommen
- Beispiele:
  - Buch $\Rightarrow$ Rechteck
  - Apfel $\Rightarrow$ Ellipse
  - Straße $\Rightarrow$ gerade oder kurvige Linie
- Aber: Modell ist immer nur eine grobe Vorgabe, deshalb: Anpassung des Modelles

mathematisches Modell (Rechteck, Ellipse, Kurve, ...)

$\Downarrow$

Anpassung an die gemessenen Bildpunkte

$\Downarrow$

Segmentierung des Bildes in die entsprechenden Bereiche

Die Hough-Transformation

- Suche nach Linien in einem Bild (keine Anpassung eines Modelles)
- Grundlage:

  jede Linie, die durch den Punkt (x, y) geht, ist darstellbar als:

  $$x \cos\theta + y \sin\theta + r = 0, \theta \in [0, 2\pi]$$

  $\Rightarrow$ für einen Punkt (x, y) beschreibt (θ, r) genau eine Linie, die durch diesen Punkt geht

  $\Rightarrow$ alle Geraden, welche auf der Kurve

  $$r = -x_0 \cos\theta - y_0 \sin\theta$$

  liegen, führen durch den Punkt (x_0, y_0)
- $r \leq R$
- Bestimme, welche Punkte eine Linie bilden:
  - Aufstellung eines Akkumulatorarrays mit den Dimensionen θ und r
  - bestimme für jeden Punkt die durch ihn gehenden Linien und erhöhe einen Zähler im entsprechenden Arrayfeld (θ, r) um eins
  - die Linie mit den meisten Stimmen gewinnt

Probleme in der Praxis

- **Quantisierungsfehler**
  - Wahl der Gittergröße häufig schwierig
  - zu grob: in einem Korb befinden sich die Stimmen sehr verschiedener Linien
    zu fein: in keinem Korb befinden sich genug Stimmen, da ähnliche Linien keine gemeinsamen Stimmen bekommen
  - Lösungsansätze:

    $\Rightarrow$ Anpassung durch „Try and Error“

    $\Rightarrow$ Falls alle Nachbarn eine Stimme haben, so bekommt das zentrale Feld auch eine
- **Rauschen**
  - Verfahren verbindet auch relativ weit voneinander enfernt liegende Punkte
  - eigentlich ein Vorteil, aber:

    in einer gleichverteilten Menge von Bildpunkten werden viele Phantomlinien gefunden
  - Texturen können mehr Stimmen bekommen als eventuell vorhandene Linien
  - Lösungsansätze:

    $\Rightarrow$ Entfernung möglichst vieler irrelevanter Bildpunkte, zum Beispiel durch Kantendetektor

Linienanpassung - Methode der kleinsten Quadrate (Least Squares)

- Anpassung der durch $y = ax + b$ repräsentierten Linie an die Datenpunkte (x_i, y_i)
- Finde die Parameter a und b, für die die Gleichung bei gegebenem x_i den besten Wert für y_i vorhersagt

  $\Rightarrow$ minimiere den Ausdruck:

$$\sum_i (y_i - ax_i - b)^2$$

- Lösung des Problems durch Differenzierung:

$$\begin{pmatrix} \overline{y^2} \\ \overline{y} \end{pmatrix} = \begin{pmatrix} \overline{x^2} & \overline{x} \\ \overline{x} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}.$$

- Aber:
  - Fehler hängt vom gegebenen Koordinatenrahmen ab
  - vertikale Verschiebung der Punkte zu der Linie wird als Fehler gemessen

    $\Rightarrow$ nahezu vertikale Linien verursachen große Fehler

    $\Rightarrow$ vertikale Linien werden gar nicht erkannt
- **Fazit:** leichte Berechnung, aber dafür schlechte Anpassungsergebnisse

Methode der insgesamt kleinsten Quadrate (Total Least Squares)

- Erweiterung der vorherigen Darstellung zu:

$$ax + by + c = 0$$

- Berechnung des zur Linie senkrechten Abstandes zwischen der Linie und einem Datenpunkt (u, v) durch:

$$abs(ax + by + c = 0) \text{ mit } a^2 + b^2 = 1$$

- wie vorher: Minimierung des Abstandes aller Datenpunkte zu der Linie

$$\Downarrow$$

Minimiere:

$$\sum_i (ax_i + by_i + c)^2 \text{ mit } a^2 + b^2 = 1$$

- Berechnung einer Maximum-Likelihood-Lösung durch Maximierung dieses Ausdruckes führt zu dem folgenden Eigenwertproblem:

$$\begin{pmatrix} \overline{x^2} - \overline{x}\,\overline{x} & \overline{xy} - \overline{x}\,\overline{y} \\ \overline{xy} - \overline{x}\,\overline{y} & \overline{y^2} - \overline{y}\,\overline{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

- Lösung: zwei senkrecht zur Linie stehende Linien, von denen eine die Gesuchte ist

Linienanpassung - Die inkrementelle Anpassung

- Punkte liegen nur selten isoliert $\Rightarrow$ enthalten Informationen über ihre Orientierung
- Verwendung zusammenhängender Kurven von Kantenpunkten
- Anpassung der Linie an diese Punkte

Algorithmus: inkrementelle Anpassung

```
Schreibe alle Punkte in der Reihenfolge der Kurve in die KurvenListe
Leere die LinienPunkteListe
Leere die LinienListe
until zu wenige Punkte sind in der KurvenListe
    Verschiebe die ersten Punkte von der KurvenListe in die LinienPunkteListe
    Passe die Linie an die Punkte in der LinienPunkteListe an
    while die angepasste Linie ist gut genug do
        Verschiebe den nächsten Punkt von der KurvenListe in die LinienPunkteListe und passe diese an
    end
    Verschiebe den letzten Punkt von der LinienPunkteListe in die KurvenListe
    Passe die Linie an
    Füge die Linie zur LinienListe hinzu
end
```

Linienanpassung - Punktzuweisung mittels K-Means

- Anwendung, falls Punkte isoliert liegen
- Modell:
  - es existieren k Linien
  - jede generiert eine Teilmenge der Datenpunkte
- Minimiere:

$$\sum_{l_i \in \text{Linien}} \sum_{x_j \in \text{Linie } i} dist(l_i, x_j)^2$$

- Aber: zu komplex!!!

Algorithmus: Anpassung mit K-Means

Wähle k Linien

or

Wähle ein Zuweisung von Linien zu Punkten und passe diese Linien unter Benutzung der Zuteilung an

until Konvergenz

Teile jedem Punkt die am nächsten liegende Linie zu

Passe die Linien an

end

Kurvenanpassung - Implizite Kurven

- ähnliches Prinzip wie bei Geraden

  $\Rightarrow$ minimiere den quadratischen Abstand zwischen Kurve und Datenpunkten

- allgemeine Form von impliziten Kurven: $\phi(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0$
- finde den zu (d_x, d_y) am nächsten liegenden Punkt (u, v) auf der Kurve
- Eigenschaften von (u, v):
  1. $\phi(u, v) = 0$
  2. $\mathbf{s} = (d_y, d_y) - (u, v)$ ist normal zur Kurve, also dem Tangentialvektor **T** an dieser Stelle
- der Kürzeste dieser Vektoren ist der Gesuchte
- Aber: ist ϕ vom Grad, d so ergeben sich bis zu d^2 viele Lösungen
- $\Rightarrow$ Verwendung von Distanzapproximationen
  1. $dist((d_x, d_y), \phi(x, y)) = \phi(d_x, d_y)$ (algebraische Distanz)
  2. $\frac{\phi(d_x,d_y)}{|\nabla\phi(d_x,d_y)|}$ (normalisierte algebraische Distanz)

Kurvenanpassung - Parametrische Kurven

- allgemeine Form von parametrischen Kurven: $(x(t), y(t)) = (x(t;\theta), y(t;\theta)), t \in [t_{min}, t_{max}]$
- Verfahren ähnlich dem bei impliziten Kurven
- suche den durch τ definierten am nächsten zu (d_x, d_y) liegenden Punkt
- $\mathbf{s}(\tau) = (d_x, d_y) - (x(\tau), y(\tau))$ ist normal zum Tangentialvektor **T**
- nur eine zu lösende Gleichung
- Aber: $x(\tau)$ und $y(\tau)$ sind fast immer Polynome

  $\Rightarrow$ evtl. große Anzahl möglicher zu findender Wurzeln
- weiterer Nachteil: parametrische Kurven mit unterschiedlichen Koeffizienten können dieselbe Kurve repräsentieren

  z.B.: $x(t), y(t)$ mit $t \in [0, 1]$ entspricht $x(2t), y(2t)$ mit $t \in [0, 1/2]$

Anpassung als probabilistisches Reduktionsmodell

- dem Modell zur Anpassung sollte ein Fehlermodell der zu erwartenden Abweichungen zugrunde liegen

  $\Rightarrow$ Anpassung von Linien an Punkte, deren Quelle bekannt ist

- simuliere k Messwerte (x_i, y_i) durch das Modell:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + n \begin{pmatrix} a \\ b \end{pmatrix} \text{ mit } n \sim N(0, \sigma),\ au + bv + c = 0, \text{ und } a^2 + b^2 = 1$$

$\Rightarrow$ **Likelihood-Funktion**: $P(\text{Messwerte}|a, b, c) = \prod_i P(x_i, y_i|a, b, c)$

$\Rightarrow$ **Log-Likelihood-Funktion**:

$$-\frac{1}{2\sigma^2} \sum_i (ax_i + by_i + c)^2 \text{ mit } a^2 + b^2 = 1$$

- Maximum-Likelihood-Berechnung führt zur *Total Least Squares*-Berechnung

Robustheit

- bisher: quadratische Fehlertherme

  $\Rightarrow$ starke Gewichtung von Ausreißern

  $\Rightarrow$ unter Umständen verfälschtes Ergebnis

- Warum entstehen Ausreißer?

  $\Rightarrow$ Fehler bei Datenmessung und -übertragung

  $\Rightarrow$ zugrunde liegende Modelle fehlerhaft

- Lösungsmöglichkeiten:

  1. Einführung eines expliziten Ausreißermodelles
  2. Rauschen erhält stärker gewichtete Ränder
  3. Suche nach gut erscheinenden Punkten

M-Schätzer

- Kriterium: der beste Schätzer ist der, welcher sich am besten auf der schlechtesten Verteilung verhält, möglichst nah am parametrischen Modell
- Schätzung durch Minimierung von:

$$\sum_i \rho(r_i(\mathbf{x}_i, \theta); \sigma)$$

  mit:
  - $r_i(\mathbf{x}_i, \theta)$ als Residualfehler am i-ten Datenpunkt
  - θ als die Parameter des Modelles
- gängige Wahl:

$$\rho(u, \sigma) = \frac{u^2}{\sigma^2 + u^2}$$

  Bild der Funktion
- Diskussion anhand der Einflussfunktion:

$$\frac{\partial \rho}{\partial \theta}$$

M-Schätzer

Unter Umständen auftretende Probleme:

- Berechnung der Extrema nicht linear $\Rightarrow$ iterative Lösung
- vernünftige Abschätzung von σ (Skalierung)

Häufig verwendet:

$$\sigma^{(n)} = 1,4826 \ \mathrm{median}_i |r_i^{(n)}(x_i; \theta^{n-1})|$$

Algorithmus:

For $s = 1$ **to** $s = k$

Drucke eine Teilmenge von r verschiedenen, gleichverteilt gewählten Punkten

Passe an diese Menge von Punkten, unter Benutzung von Maximum-Likelihood (gewöhnlich kleinste Quadrate), an um θ_s^0 zu erhalten

Schätze σ_s^0 unter Benutzung von θ_s^0 ab

Until Konvergenz (gewöhnlich: $|\theta_s^n - \theta_s^{n-1}|$ ist klein)

Mache einen Minimierungsschritt durch Anwendung von θ_s^{n-1}, σ_s^{n-1} um θ_s^n zu erhalten

Berechne nun σ_s^n

end

end

Melde die beste Anpassung an die Menge, unter Benutzung des Medians der Reste als Kriterium

Random Sample Consensus (RANSAC)

- Random Sample Consensus = Zufallsprobenübereinstimmung
- Suche nach Datenpunkten, die „gut“erscheinen
- Iterativer Prozess:

1. wähle eine kleine Teilmenge aller Punkte
2. passe bezüglich dieser Teilmenge an
3. nun sieht man, wie viele der anderen Punkte zu dem daraus resultierenden Objekt passen
4. wiederhole diesen Prozess so lange, bis eine hohe Wahrscheinlichkeit besteht, dass das gesuchte Objekt gefunden wurde

RANSAC-Algorithmus

Algorithmus:

Bestimme:

n - die kleinste Anzahl an benötigten Punkten

k - die Anzahl der benötigten Iterationen

t - die Grenze, mittels welcher bestimmt wird, ob ein Punkt nahe genug bei der Linie liegt

d - die Anzahl der benötigten in der Nähe liegenden Punkte, um zuzusichern, dass das Modell gut passt

Until k Iterationen wurden durchgeführt

Drucke eine Teilmenge von n Punkten, gleichverteilt aus den Daten gewählt

Passe an diese Menge von n Punkten an

For each Datenpunkt außerhalb der Teilmenge

Prüfe die Distanz zwischen Punkt und Linie gegen t. Falls die Distanz zwischen dem Punkt und der Linie kleiner als t ist, dann liegt der Punkt nahe

end

Falls d oder mehr Punkte existieren, welche eine bestimmte Fehlertoleranz nicht überschreiten, passe die Linie unter Benutzung aller dieser Punkte wieder an

end

Benutze die beste Anpassung aus der Sammlung mit dem Anpassungsfehler als Kriterium

RANSAC - Parameterbestimmung

Modellkompatibilität - d

- y: Wahrscheinlichkeit für Ausreißer in der Auswahl
- wähle d so, dass y^d klein ist

Anzahl der Iterationen - k

- $$\begin{aligned} E[k] &= 1P(\text{eine gute Probe bei einmal drucken}) + 2P(\text{eine gute Probe bei zweimal drucken}) + \ldots \\ &= w^n + 2(1-w^n)w^n + 3(1-w^n)^2 + \ldots = w^{-n} \end{aligned}$$
- Sicherheit einer guten Probe $\Rightarrow$ mehr als $E[k] = w^{-n}$ Proben durch Hinzunahme von ein oder zwei Standardabweichungen $SD(k)$
- $SD(k) = \frac{\sqrt{1-w^n}}{w^n}$

Wann liegt ein Punkt nahe? - t

- Teil des Modellierungsprozesses
- Abschätzung durch versuchweises Anwenden

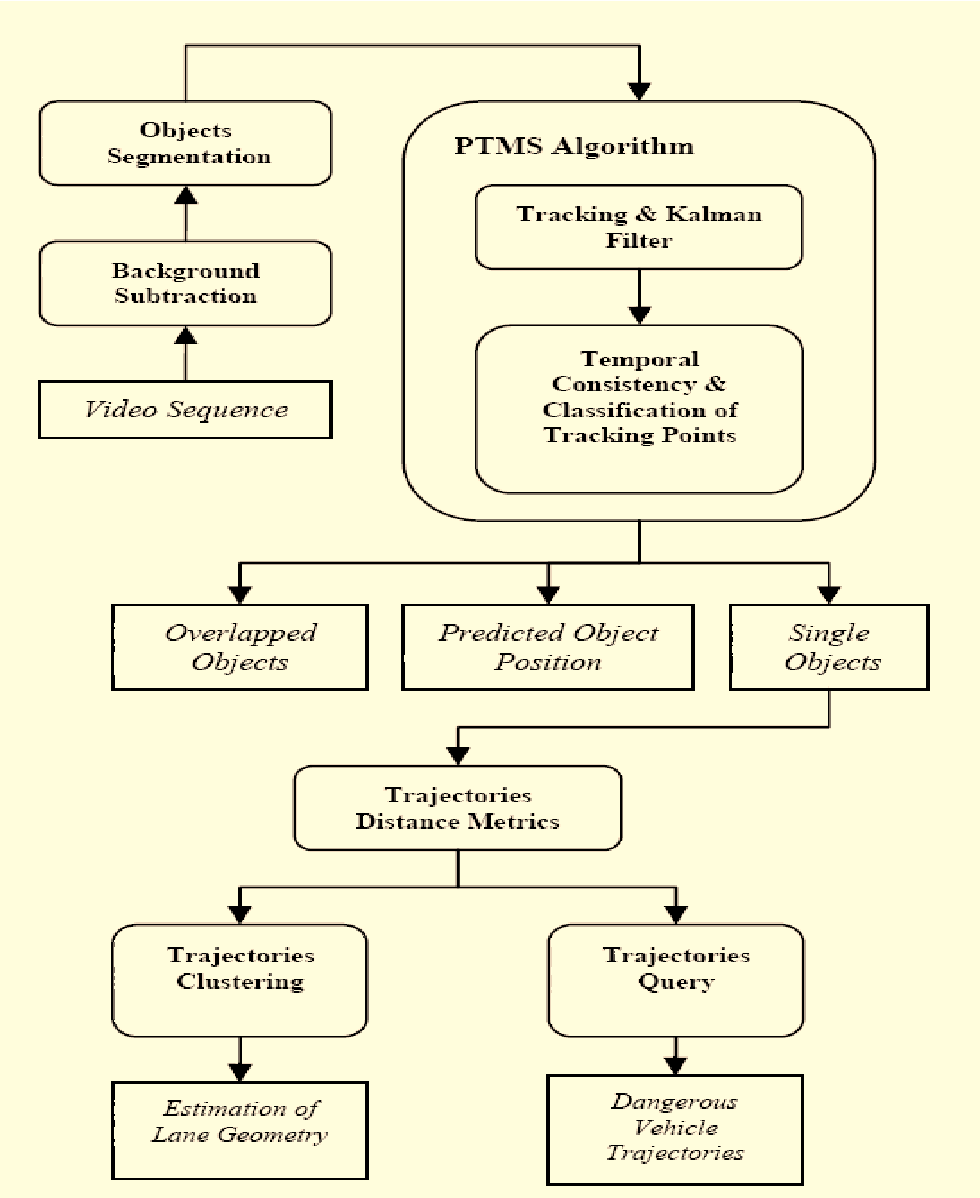
Beispiel: automatische Verkehrsüberwachung auf Autobahnen

- Autobahnen bieten sich an, da stark strukturierte Umgebung
- Untersuchung von Filmaufnahmen besser als von statischen Bildern, da:
  - Benutzung von nichtstationären Pan-Tilt-Zoom-Kameras
  - Blickwinkel- und Skalenunabhängigkeit
  - weniger empfindlich gegenüber sensorischem Rauschen und Lichtveränderungen
- Voraussetzung: ungefähre Kenntnis der Fahrbahnbreite

Vorgehensweise:

1. Erkennung von sich bewegenden Fahrzeugen und Überdeckungen
2. Analyse der Bewegungen der Fahrzeuge
3. Abschätzung der Fahrwege und Fahrbahnen

Gliederung


Objects Segmentation
Background Subtraction
Video Sequence
PTMS Algorithm
Tracking & Kalman Filter
Temporal Consistency & Classification of Tracking Points
Overlapped Objects
Predicted Object Position
Single Objects
Trajectories Distance Metrics
Trajectories Clustering
Trajectories Query
Estimation of Lane Geometry
Dangerous Vehicle Trajectories

Predictive Trajectory Merge-And-Split Algorithmus (PTMS)

Vorbereitung:

- Entfernung des Hintergrundes
- Segmentierung der Objekte (Blob $\geq K_{min}$)
- Verfolgung der Bewegung mittels Kalman-Filter
- Annahme: jeder Blob besteht aus einer gewissen Anzahl von Objekten (Kindern)
- jeder Blob wird mit einem Kindelement initialisiert

Algorithmus:

- Algorithmus vergleicht aufeinanderfolgende Frames
- Falls neue Blobs aus einem alten Blob hervorgehen, so wird die Anzahl der Kindelemente dieses alten Blobs entsprechend verringert
- Falls sich mehrere Blobs zu einem vereinen, so wird dieser mit der entsprechenden Anzahl Kinder initialisiert

Problem: Blobs, die von Anfang an aus mehreren Objekten bestehen, werden als ein Objekt erkannt
$\Rightarrow$ mit muster- oder modellbasiertem Ansatz Anfangsblobs untersuchen

Bewegungsbahnerkennung durch Referenzsuche

Verwendete Metriken:

- Mengen von Polynomen der Bewegungsbahnmodelle:

  $T_1 = \{t_1(x) : x \in [a, b]\}$

  $T_2 = \{t_2(x) : x \in [c, d]\}$

- Differenzpolynom:

  $d(x) = t_1(x) - t_2(x) : x \in [a, b] \cup [c, d]$

- Distanzmessungen $d(T_1, T_2)$:

  $d_{max}(T_1, T_2) = \sup_{x \in I}\{d(x)\}$

  $dc_{min}(T_1, T_2) = \inf_{x \in I}\{d(x)\}$

  $dc_{mean}(T_1, T_2) = \sqrt{\frac{1}{l_2 - l_1} \int_{l_1}^{l_2} \{d(x)\}^2 dx}$

Verfahren:

- spezifiziere Referenzsuche T_Q
- bestimme nach Ähnlichkeit sortierte Liste der ähnlichen Bewegungsbahnen mittels der Metriken

Abgewandelter K-Means Cluster Algorithmus

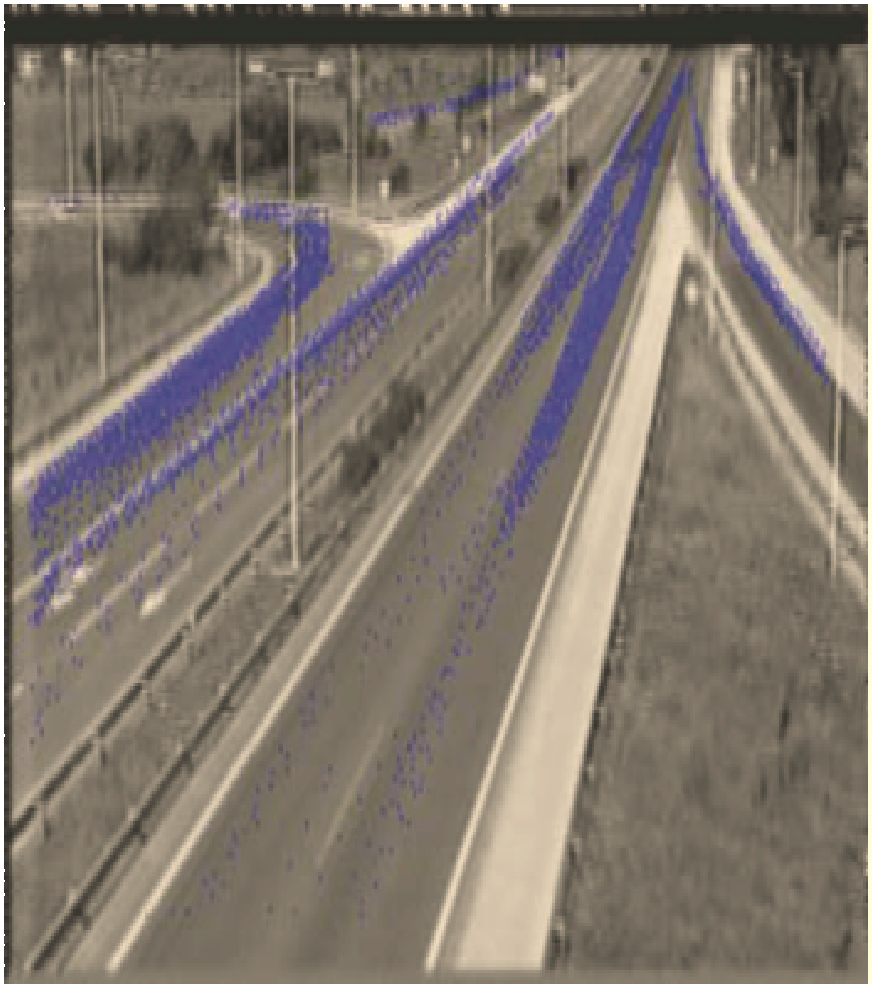
Erstelle eine Startmenge von Clusterbahnen:

1. Erstelle eine lineare Referenzbahn t_R am Rande des Bildes parallel zur Hauptorientierung des Verkehrsflusses.
2. Erstelle eine Menge mit einer Bahn t_J, die die größte Distanz von t_R unter der Benutzung der dc_{min}-Metrik hat.
3. Finde die nächste Bahn t_{J+1} $(J = 1, 2, ...)$ mit dem größten Abstand unter der Anwendung von dc_{min} aller in der Menge vorhandener Bahnen mit einer Länge oberhalb einer bestimmten Grenze.
4. Falls die gefundene Bewegungsbahn eine Distanz dc_{min} zur nächsten Bahn hat, die geringer ist als die abgeschätzte Fahrbahnbreite, so verwerfe die Bewegungsbahn.
5. Wiederhole dies von Schritt 3 an so oft, bis keine weiteren Bahnen mehr hinzugefügt werden können.

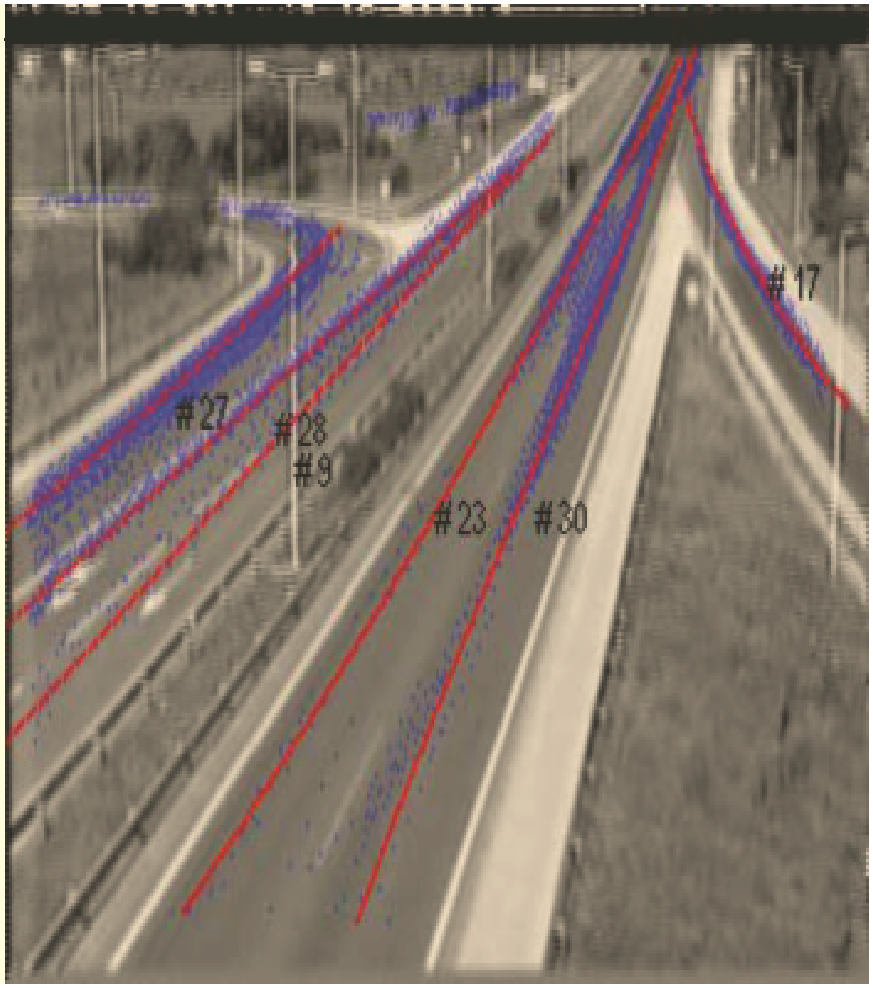
Führe nun den Clusteringschritt durch:

- fasse eine Bewegungsbahn und einen Cluster zusammen, wenn:
  - $d_{max} < \tau$
  - $|t| > L_{min}$
- füge die Bahn dann dem Cluster T_J hinzu, der $min(d_{max}{}'s)$ liefert
- Cluster ist auf 20 Bahnen beschränkt
- der Cluster wird mit RANSAC nach fünf betrachteten Bahnen neu modelliert

Beispiel für den abgewandelten K-Means Cluster Algorithmus



Gemessene Daten



Fahrbahnen nach Clusterung

Anwendung Verkehrsüberwachung

Erkennen des Überfahrens einer durchgezogenen Linie

- gebe die durchgezogene Linie als Referenz an
- durchsuche die Bewegungsbahnen in der Datenbank nach solchen die $dc_{min} < 0.5$ Pixel haben

$\Rightarrow$ mit hoher Wahrscheinlichkeit haben die gefundenen Bewegungsbahnen die durchgezogene Linie gekreuzt