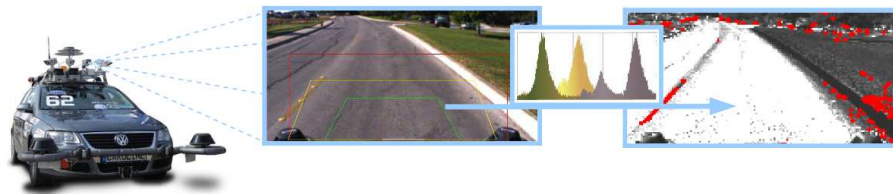


Echtzeiterkennung von befahrbaren Bereichen in urbanen Szenarien

Kai Berger, Christianz Linz, Christian Lipski, Timo Stich, Marcus Magnor

Institut für Computergrafik,
Mühlenpfordtstraße 23, 38106 Braunschweig
{ berger,lipski,linz,stich,magnor } @cg.tu-bs.de

Zusammenfassung. Unser Artikel beschreibt ein Echtzeitverfahren zur kamerabasierten Fahrbereichserkennung, welches in urbanen bzw. ländlichen Fahr Szenarien eingesetzt wird. In dem Eingabebild eines monokularen Kamerasystems, welches auf dem Dach eines Automobils in Fahrtrichtung montiert ist, wird pro Zeitschritt ein kleiner Bereich vor der Motorhaube als befahrbar vorausgesetzt. Der Algorithmus berechnet die vorherrschenden Farben innerhalb dieses befahrbaren Bereiches und vergleicht die Farben mit den Farbwerten jedes Pixels im Eingabebild: Je ähnlicher die Pixelfarben zu den vorherrschenden sind, desto höher ist die Wahrscheinlichkeit, dass die entsprechenden Bereiche befahrbar sind. Um den Algorithmus auch im städtischen Umfeld einsetzen zu können, muss ein vorverarbeitendes Modul vorgeschaltet werden, welches Fahrspuren, Schatten und überbelichtete Bereiche ausmaskiert und in der endgültigen Befahrbarkeitskarte als unbekannt (rot) markiert. Weiterhin wird ein dynamisches Suchpolygon vorgestellt, um den Algorithmus unabhängig von weiteren Eingabesensoren zu gestalten.



1 Einleitung

Das im folgenden beschriebene Verfahren ist ein Teil der Software, die im autonomen Fahrzeug *Caroline* verwendet wurde. *Caroline* hat im Finale der *DARPA Urban Challenge 2007* teilgenommen. Der vorgestellte Algorithmus ist unablässig in Situationen, in denen die Fahrbahnfläche von anderen Sensoren nicht ausreichend klassifiziert werden kann, z.B. auf Straßen ohne Fahrspurmarkierungen bzw. ohne ausreichend hohe Bordsteinkanten oder auf Feldwegen.

Um ein besseres Verständnis der Aufgabe des vorgestellten Algorithmus zu bekommen, ist ein kurzer Überblick über die im Fahrzeug verwendete Sensorik notwendig.

Diese unterteilt sich in aktive Sensoren (drei Laserscanner, zwei Radar-Sensoren und zwei Lidar-Sensoren), welche große Objekte, wie Mauern, Autos oder Straßensperren wahrnehmen sollen und passive Sensoren (7 Kameras), welche Straßenmarkierungen [?] und die Befahrbarkeit des Untergrundes [4] erkennen sollen. In einer nachgeschalteten Sensorfusion werden die einzelnen Informationsquellen in einer 2D-Karte fusioniert. Zwei Laserscanner werden von einem Bodenerkennungsmodul zur Rekonstruktion der Oberflächengeometrie ausgelesen, um kleine Hindernisse oder unebenes Terrain zu umfahren. Fahrspurmarkierungen und Stopplinen werden von einem Modul zur Fahrspurerkennung mit Hilfe vierer Farbkameras wahrgenommen. Die künstliche Intelligenz wertet daraufhin alle vorliegenden Informationen aus und berechnet eine optimale Trajektorie, welche an die Lenk- und Beschleunigungsaktorik des Fahrzeugs gesendet wird.

2 Stand der Technik

Die wesentliche Grundlage für die folgenden Betrachtungen bildet der Algorithmus auf der folgenden Seite, welcher von Thrun et al. [1] vorgestellt und ausführlich in der DARPA Grand Challenge im Oktober 2005 ausgetestet wurde. Die Idee ist, in einem gegebenen Kamerabild einen kleinen Bereich als befahrbar anzunehmen und die darin enthaltenen Farben mit dem gesamten Bild zu vergleichen um ähnliche Farben als befahrbar zu markieren. Allerdings ist der Algorithmus nur für Geländefahrten entwickelt worden um den Anforderungen der Grand Challenge zu genügen. Der im folgenden vorgestellte Algorithmus erhält zusätzlich als Eingabe die Tiefenwerte eines Laserscanners. In einer Höhenkarte werden die Scanlinien normalerweise über die Zeit ausgewertet. Um die Datenmenge, die zwischen Laserscanner und dem Algorithmus ausgetauscht wird, gering zu halten, wird pro Zeitschritt ein Hüllpolygon definiert. Dieses umfasst den Bereich vor dem Auto in der Höhenkarte, welcher als befahrbar betrachtet wird. Das übertragene Polygon wird dann in die Bildkoordinaten der Eingabekamera transformiert und gegebenenfalls geclippt, so dass es nun den befahrbaren Bereich im Bildraum markiert. Alle Farben innerhalb dieses Bereiches werden nun betrachtet und statistisch ausgewertet, um die vorherrschenden Farbtöne zu bestimmen, z.B. straßengrau oder grasgrün. Diese Farbtöne werden nun mit allen Pixeln im aktuellen Kamerabild verglichen, wobei ein im verwendeten Farbraum anwendbares Distanzmaß benutzt wird. Ist die resultierende Distanz eines Pixels kleiner als ein vorgegebener Schwellwert, wird der Pixel und damit der Bereich in der Welt, den er abbildet, als befahrbar angenommen. Zusammengefasst erweitert der Algorithmus den Bereich vor dem Auto, über den eine Aussage zur Befahrbarkeit des Untergrundes getroffen werden kann, von einigen wenigen Metern auf über 50 Meter. In der städtischen Umgebung stößt das beschriebene Verfahren an seine Grenzen und muß weiterentwickelt werden. Beim intensiven Testen des Algorithmus in städtischer Umgebung sind neue Probleme festgestellt worden, da die Straßen nun andersgefärbte Fahrbahnmarkierungen besitzen und große Gebäude weite Schatten über die Straße werfen. So kommt es vor, dass sich gelbe Fahrbahnmarkierungen, wie sie z.B. in den USA vorkommen, nicht

Algorithmus 1 : Der grundlegende Algorithmus zur Berechnung befahrbarer Regionen

Data : Das Bild der Eingabekamera I_{kamera} und das Polygon des Laserscanners $P_{Scanner}$

Result : Ein klassifiziertes Bild $I_{befahrbar}$, segmentiert in befahrbare und nicht befahrbare Regionen.

```
1 begin
2   Hole das Bild der Eingabekamera  $I_{kamera}$  und das Laserscannerpolygon
    $P_{Scanner}$  welches einen Bereich vor dem Auto als definitiv befahrbar
   markiert.
3   Kopiere  $I_{kamera}$  nach  $I_{befahrbar}$ 
4   Erstelle eine Liste der aktuellen Farben von den Bildpixeln von  $I_{befahrbar}$ 
   die innerhalb  $P_{Scanner}$  sind
5   Berechne daraus  $n$  vorherrschende Farben und verbessere diese mit
   Cluster-Methoden, wie dem Expectation Maximisation Algorithmus[2], [3]
   /* Füge diese  $n$  Farbverteilungen in eine Liste von  $m > n$ 
   Farbverteilungen */
6   if Die Listengröße kleiner  $m$  ist then
7     | Füge den Wert in die Liste ein
8   else
9     /* Vergleiche jede Farbverteilung  $i$  von den  $n$  Farbverteilungen
   mit jeder Verteilung  $j$  der  $m$  gespeicherten
   Farbverteilungen
   Zum Vergleich wird die Mahalanobisdistanz verwendet:
    $d(i, j) = (\mu_i - \mu_j)^\perp * (\Sigma_i + \Sigma_j)^{-1} (\mu_i - \mu_j)$  */
10    foreach Farbverteilung  $j$  do
11      | Speichere die kleinste Farbverteilung  $d(i, j)$  und die Position  $j$ 
12    end
13    if  $d(i, j)$  ist kleiner oder gleich einem Schwellwert  $\phi$  then
14      /* Farbverteilung  $j$  wird angepasst an Farbverteilung  $i$  */
15       $\mu_j \leftarrow \frac{1}{\alpha_i + \alpha_j} (\alpha_i * \mu_i + \alpha_j * \mu_j)$ 
16       $\Sigma_j \leftarrow \frac{1}{\alpha_i + \alpha_j} (\alpha_i * \Sigma_i + \alpha_j * \Sigma_j)$ 
17       $\alpha_j \leftarrow \alpha_i + \alpha_j$ 
18      /* Dabei ist  $\alpha$  die Gewichtung der Verteilung je nach
   Anzahl der zu Grunde liegenden Farben */
19    else
20      | Lösche die Farbverteilung  $k$  mit der kleinsten Gewichtung  $\alpha_k$  und
   füge dafür Verteilung  $i$  ein
21    end
22  end
23  foreach Gespeicherte Farbverteilung  $j$  do
24    | Multipliziere das Gewicht  $\alpha_j$  mit einem Faktor  $\gamma < 1$ 
   /* Dadurch wird das Gewicht einer Farbverteilung mit jedem
   Frame geringer */
25  end
26  foreach Pixel des Bildes  $I_{befahrbar}$  do
27    foreach Gespeicherte Farbverteilung  $j$  do
28      | Berechne die Verteilung  $p(x, \mu_j, \Sigma_j)$  mit der Verteilungsfunktion
29      if  $p(x, \mu_j, \Sigma_j)$  ist größer als ein Schwellwert  $\chi$  für ein gegebenes  $j$ 
30      then
31        | Klassifiziere den Pixel  $x$  als befahrbar
32      else
33        | Klassifiziere den Pixel  $x$  als nicht befahrbar
34      end
35    end
36  end
37 end
```

innerhalb des Bereiches befinden, der vom Laserscanner als befahrbar erkannt wird ($P_{Scanner}$). Infolgedessen sind sie im Kamerabild als nicht befahrbar markiert. Eine durchgezogene Linie in der Mitte der Fahrbahn verhindert z.B. einen Fahrspurwechsel, während Stoplinien als blockierende Objekte vor dem Auto erkannt werden, Abb. 1.



(a) Eingabebild

(b) Normale Befahrbarkeitskarte

Abb. 1. Die Befahrbarkeitskarte (b) wird vom grundlegenden Algorithmus zur Berechnung befahrbarer Regionen erstellt. Ein weißer Pixel ist befahrbar, ein schwarzer ist unbefahrbar. Eine gelbe Fahrspur (a) allerdings wird als unbefahrbar (b, schwarz) erkannt, weil die Farbe nicht im Suchpolygon auftritt.

Desweiteren erweisen sich großflächige Schatten von höheren Gebäuden als problematisch, während schmale Schatten von Bäumen die Fahrbahnfarbe im Kamerabild nur leicht beeinflussen. Die großflächigen Gebäudeschatten wurden in Gänze als unbefahrbar interpretiert, Abb. 2. Sobald das Auto sich allerdings innerhalb dieser Schattenfläche befindet, passt sich die Belichtungsregelung der Kamera an die verminderte Helligkeit an mit der Folge dass alle Flächen ausserhalb des Schattens überbelichtet erscheinen und ebenfalls als nicht befahrbar erkannt werden, Abb. 3.



(a) Eingabebild

(b) Normale Befahrbarkeitskarte

Abb. 2. Große dunkle Schatten (a, links) unterscheiden sich zu stark von der Straßenfarbe (b, dunkel).

Bei Fahrten am frühen Vormittag und am Nachmittag erweist sich auch der eigene Schatten des Fahrzeugs als Störfaktor, sobald sich die Sonne hinter dem Auto befindet. In diesem Fall taucht der Eigenschatten im Kamerabild vor dem Fahrzeug auf und wird entweder als nicht befahrbar oder als einzig befahrbarer Bereich im gesamten Bild markiert, Abb. 4.



(a) Eingabebild

(b) Normale Befahrbarkeitskarte

Abb. 3. Sobald sich die Kamera im Schatten befindet, wird die Belichtung nachgeregelt (a). Unglücklicherweise erscheinen Bereiche ausserhalb des Schattens überbelichtet und werden deshalb als nicht befahrbar erkannt (b, dunkel).



(a) Eingabebild

(b) Normale Befahrbarkeitskarte

Abb. 4. Der Eigenschatten des Fahrzeugs verursacht Probleme (a), zum Beispiel, wenn der Schatten in dem Suchbereich für die befahrbaren Farben liegt (b, weiß).

Schlussendlich ist es problematisch, in Gegenden zu testen, in denen kaum ein Höhenunterschied zwischen der Straße und dem seitlich angrenzendem Gelände, z.B. Grasnarbe oder Sand, festzustellen ist. Die vorgenommenen Änderungen werden ausführlich in Abschnitt 3 beschrieben, es kann aber bereits festgestellt werden, dass anstelle des im Algorithmus verwendeten EM-Verfahrens [3] auch das KMeans-Verfahren angewendet werden sollte. Ebenso können die Abstandsberechnungen in unterschiedlichen Farbräumen, wie RGB, aber auch YUV, L^*a^*b oder HSV durchgeführt werden, ohne dass der vorgestellte Algorithmus eingeschränkt wird.

3 Der Algorithmus für urbane Szenarien

Um die oben vorgestellten Probleme zu bewältigen, wurde ein zusätzliches System entwickelt, welches die Bilder der Eingabekamera vorbearbeitet. Bevor die Eingabebilder an Algorithmus 1 gegeben werden, werden sie an die folgenden Präprozessoren verteilt: Präprozessor für dunkle Bereiche, Präprozessor für überbelichtete Bereiche, Präprozessor für Fahrbahnmarkierungen und Präprozessor für Eigenschatten-Pixel. Jeder Präprozessor liefert nach erfolgter Bearbeitung eine Bitmaske, welche die Pixel ausmaskiert, die jeweils als kritisch bewertet werden. Die ausmaskierten Pixel werden in der endgültigen Befahrbarkeitskarte nicht wie bisher mit einem Farbwert besetzt, sondern als *unbekannt* gekennzeichnet.

Der Präprozessor für dunkle Bereiche maskiert Pixel aus, die zu dunkel und damit potentiell im Schatten sind. Das Eingabebild wird in den HSV-Farbraum konvertiert, der Helligkeitswert des Pixels mit einem gegebenen Schwellwert verglichen. Wenn der Wert kleiner ist, wird der Pixel in der Ausgabemaske auf 1 gesetzt, ansonsten auf 0.

Der Präprozessor für überbelichtete Bereiche maskiert überbelichtete Pixel aus. Dazu wird das Eingabebild in den HSV-Farbraum konvertiert. Anschließend wird der Helligkeitswert jedes Pixels mit einem vorgegebenem Schwellwert verglichen. Wenn der Wert größer ist, wird der Pixel in der Ausgabemaske auf 1 gesetzt, ansonsten auf 0.

Der Präprozessor für Fahrbahnmarkierungen sucht Pixel, die im RGB-Farbraum nahe an gelben Farbtönen sind, und somit potentiell zu gelben Fahrspuren, wie sie in den USA auftreten, gehören. Wenn der Grün-Wert des Pixels größer als der Rot-Wert und der Blau-Wert ist, so wird er nicht als gelblich betrachtet. Ebenso wenig gelblich ist er, wenn der Rot-Wert größer als die Summe des Blau- und Grün-Wertes ist. In allen anderen Fällen wird das Verhältnis $\frac{\min(R,G)}{B} - 1$ betrachtet. Ist es größer als ein gegebener Schwellwert, so wird der Pixel in der Ausgabemaske mit 1 gesetzt, ansonsten mit 0. Um lediglich schmale Fahrbahnschienen herauszufiltern und große gelbe Bereiche (z.B. sandige Flächen am Fahrbahnrand) unberührt zu lassen, wird eine Kopie der Ausgabemaske noch geglättet, mit einer Dilatation versehen und schließlich von der Ausgabemaske subtrahiert.

Der Präprozessor für Eigenschatten-Pixel maskiert die Pixel aus, die potentiell zu dem eigenen Schatten des Autos gehören. Dazu wird eine feste Punktmenge $p(x)$ im Bild definiert, welche sich im Bild am Rand der Motorhaube befinden. Pro Frame wird dann per *FloodFill*-Operation ein zusammenhängender Bereich im Bild gesucht, welcher einen geringen Helligkeitswert aufweist. Um zu verhindern, dass der gefundene Bereich beliebig groß ist, wird die Summe gefundener Pixel mit einem Schwellwert verglichen, der die maximale Fläche des Autoschattens darstellt. Die gefundene Fläche wird in der Ausgabemaske markiert.

Wie bereits erwähnt, lässt sich das Eingabepolygon von einem Laserscanner dann gut verwenden, wenn der tatsächlich befahrbare Bereich durch relativ hohe Objekte, wie z.B. Sträucher oder Sandhügel, begrenzt ist. Im urbanen Szenario kann der Höhenunterschied zwischen Straße und Bordstein oder Grasnarbe sehr gering sein. Würde der Schwellwert der Befahrbarkeitserkennung des Laserscanners darauf eingestellt, können Probleme auftreten, wenn die Straße einen Hügel entlangläuft und der Höhenunterschied größer als der Schwellwert ist. Da außerdem der Laserscanner eine physikalisch basierte Fahrbahnerkennung liefert, würde er grüne Grasflächen neben der Straße als potentiell befahrbar erkennen, obwohl dieses weder wünschenswert noch von Vorteil ist. Aus diesem Grund wurde

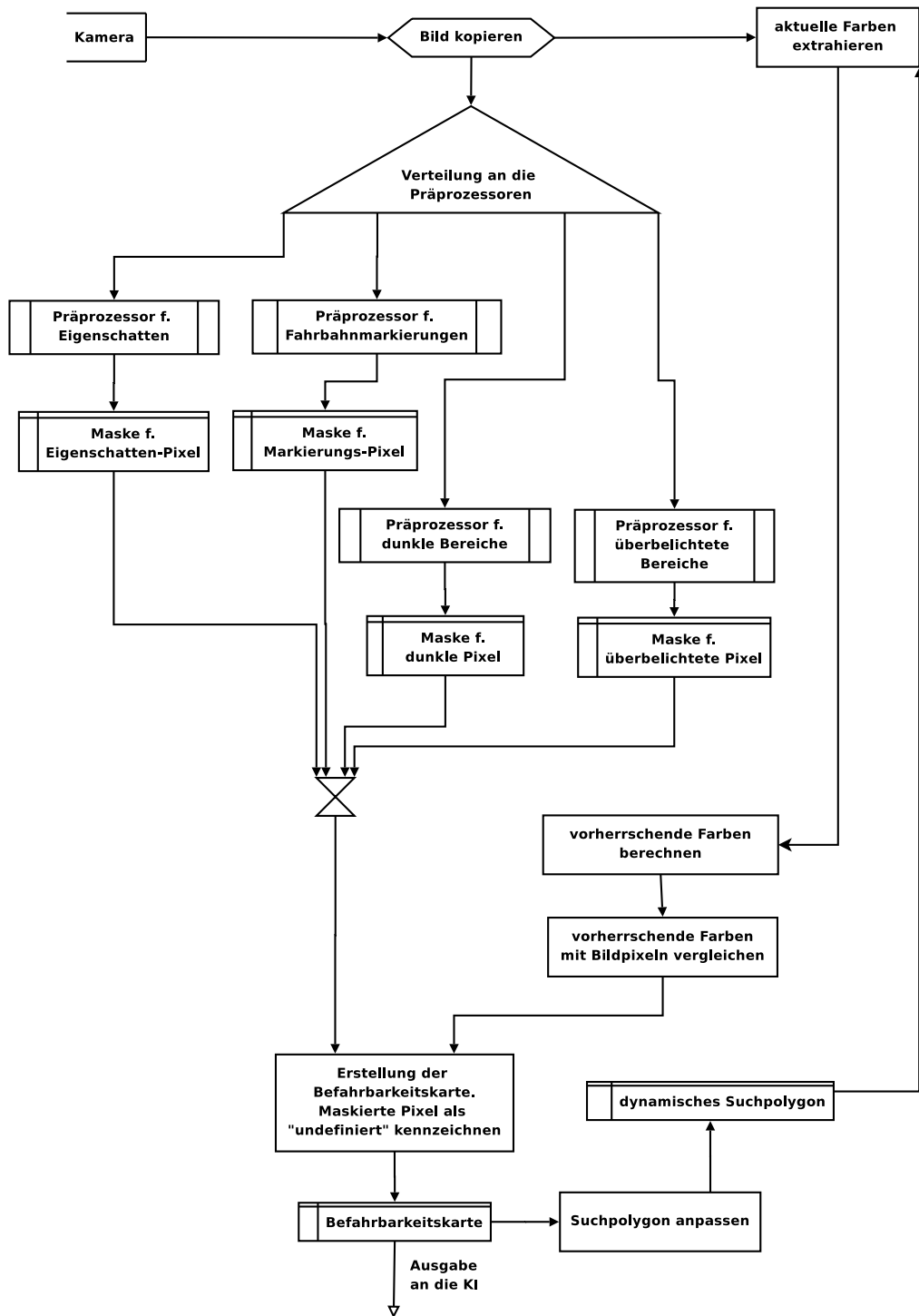
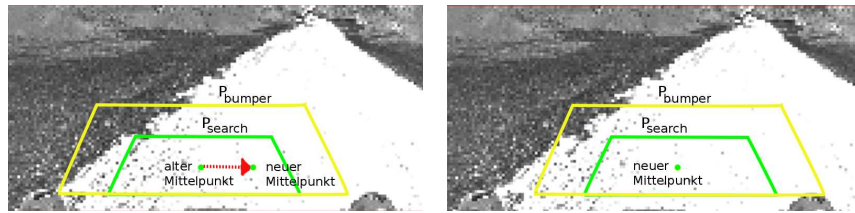


Abb. 5. Die Verarbeitungsschritte des Algorithmus



(a) Das dynamische Suchpolygon zur Zeit t

(b) Das dynamische Suchpolygon zur Zeit $t + 1$

Abb. 6. Das dynamische Suchpolygon im Frame t (a, grünes Trapez) wird zum Frame $t + 1$ nach rechts verschoben (b), weil der berechnete Schwerpunkt im gelben Polygon rechts vom Mittelpunkt des Suchpolygons liegt.

ein dynamisches Suchpolygon entwickelt, welches eine feste Form hat, aber die Möglichkeit hat sich innerhalb des Eingabebildes in einem eingegrenzten Bereich zu bewegen. Jede Bewegungsrichtung für einen Frame wird aus der resultierenden Befahrbarkeitskarte des letzten Frames berechnet. Dazu wird ein größeres Polygon P_{bumper} , Abb. 6, mit ähnlicher Form angenommen, welches das Suchpolygon stets umgibt. Innerhalb dieses Polygons werden die Befahrbarkeitswerte jedes Pixels aufsummiert, um den numerischen Schwerpunkt zu berechnen. Geringe Befahrbarkeitswerte im linken Bereich von P_{bumper} , Abb. 6, bewirken so eine Verschiebung der Polygone nach rechts, während geringe Werte im rechten Bereich umgekehrt eine Verschiebung nach links bewirken.

4 Versuchsergebnisse

Das beschriebene Verfahren wurde auf der Basis der *OpenCV*-Bibliothek [8] von *Intel* implementiert und auf einem *Intel Dual Core Car PC* mit Linux-Betriebssystem installiert. Über USB spricht das System eine *IDS uEye* - Kamera mit 640×480 Pixel Bildauflösung an. Für das Verfahren werden die Eingabebild auf eine 160×120 Pixel Auflösung herunterskaliert um eine Mindest-Framerate von 10 fps zu erreichen. Abzüglich Himmel und Motorhaube arbeitet das Verfahren in einem Bereich von 160×75 Pixel. Der beschriebene Algorithmus 1 arbeitet im Gegensatz zu [1] im L^*u^*v -Farbraum, um vorherrschende Helligkeitsunterschiede im Eingabebild für die Farbsegmentierung besser herauszufiltern. Das autonome Fahrzeug *Caroline* fuhr mit laufendem Fahrbereichserkennungs-Algorithmus 20-35 km/h in den Testkursen bis einschließlich zum Finale, spezielle Grenzen in Bezug auf die Echtzeit wurden nicht festgestellt. Abb. 7 zeigt die Unterschiede zwischen normaler Befahrbarkeitserkennung und der Erkennung mit vorangestelltem Präprozessor. Der große Schatten des linken Gebäudes ist zu unterschiedlich vom Farbton der Straße und wird deshalb als nicht befahrbar klassifiziert. Der Präprozessor für dunkle Bereiche hingegen klassifiziert die schattigen Pixel als unbekannt. Die Versuchsergebnisse für überbelichtete Bereiche sind in Abb. 8 dargestellt: Das normale Verfahren erkennt die überbelichteten Bereiche hinter dem Schatten als unbefahrbar, der Präprozessor für überbelichtete Bereiche klassifiziert sie als unbekannt. Damit Fahrspuren möglichst schnell und klar

vom menschlichen Betrachter erfasst werden, weisen sie meist einen hohen Farbgradienten zur Umgebung, der Straße, auf. Diese Tatsache erweist sich für das normale Erkennungsverfahren als Nachteil, da die Fahrspuren als unbefahrbar deklariert werden, Abb. 9. Der Präprozessor für Fahrbahnmarkierungen jedoch maskiert die gelben Pixel aus und deklariert sie als unbekannt. Obwohl Probleme mit dem Schatten des Fahrzeuges nur auftreten, wenn die Sonne hinter dem Auto steht, führen Sie jedoch dann zu unbefriedigenden Ergebnissen, Abb. 10. Der Präprozessor für Eigenschatten-Pixel hingegen sucht den zusammenhängenden dunklen Bereich vor dem Auto und deklariert ihn als unbekannt.



(a) Eingabebild (b) Normale Befahrbarkeitskarte (c) Mit Präprozessor für dunkle Bereiche

Abb. 7. Versuchsergebnisse mit dem Präprozessor für dunkle Bereiche. Das mittlere Bild (b) zeigt die Klassifikation ohne Präprozessor. Im rechten Bild (c) ist der betreffende Bereich als unbekannt klassifiziert (rot).



(a) Eingabebild (b) Normale Befahrbarkeitskarte (c) Mit Präprozessor für überbelichtete Bereiche

Abb. 8. Versuchsergebnisse mit dem Präprozessor für überbelichtete Bereiche. Das mittlere (b) Bild zeigt die Klassifikation ohne Präprozessor. Im rechten Bild (c) ist der überbelichtete Bereich markiert (rot).

5 Zusammenfassung

Es wurde ein kamerabasierter Farbsegmentierungsalgorithmus vorgestellt, welcher die Herausforderungen einer städtischen Umgebung meistert. Unter der Annahme, dass eine kleine Region direkt vor dem Auto befahrbar ist, gelingt es dem Algorithmus über die Zeit die Flächen im Eingabebild zu verfolgen, welche eine ähnliche Farbe aufweisen. Das Verfahren maskiert Artefakte im Bild aus, wie

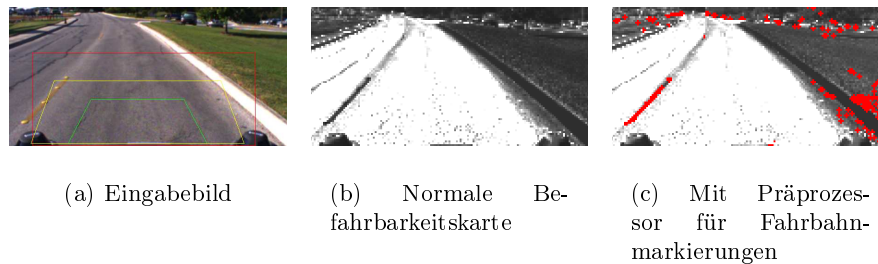


Abb. 9. Versuchsergebnisse mit dem Präprozessor für Fahrbahnmarkierungen. Das mittlere (b) Bild zeigt die Klassifikation ohne Präprozessor. Im rechten Bild (c) sind die Fahrbahnmarkierungen markiert (rot).

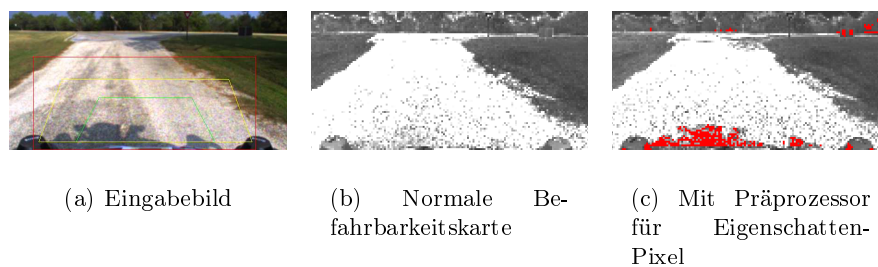


Abb. 10. Versuchsergebnisse mit dem Präprozessor für Fahrbahnmarkierungen. Das mittlere (b) Bild zeigt die Klassifikation ohne Präprozessor. Im rechten Bild (c) ist der Eigenschatten des Autos markiert (rot).

z.B. weiße oder gelbe Fahrspuren, Schatten oder überbelichtete Bereiche. Selbst wenn das autonome Fahrzeug durch Fehlentscheidungen im Begriff ist einen befahrbaren Bereich zu verlassen, gelingt es dem Verfahren, den befahrbaren Bereich weiter zu verfolgen.

Literaturverzeichnis

1. Thrun S. et al.: Stanley: The Robot That Won The DARPA Grand Challenge, Journal of Field Robotics, 661–692 (2006)
2. Duda, Richard O. and Hart, Peter E. : Pattern Classification and Scene Analysis (1973)
3. Bilmes Jeff: A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models Technical Report, University of Berkeley, ICSI-TR-97-021, (1997)
4. Kai Berger, Christian Lipski, Christian Linz, Timo Stich, Marcus Magnor The area processing unit of Caroline - Finding the way through DARPA's Urban Challenge 2nd Workshop Robot Vision 260–274 (2008)
5. Christian Lipski, Björn Scholz, Kai Berger, Christian Linz, Timo Stich, Marcus Magnor, A Fast and Robust Approach to Lane Marking Detection and Lane Tracking Proc. IEEE Southwest Symposium on Image Analysis and Interpretation (2008)
6. Open Source Computer Vision Library
<http://www.intel.com/research/mrl/research/opencv>