

ABSCHLUSSPROJEKT  
BILDVERARBEITUNG

IM RAHMEN DER LEHRVERANSTALTUNG  
„OPTISCHE SYSTEME“ (M3.2 133132)

Prüfer Prof. Dr. Dieter Maier

Autoren: Gustav Willig 197332

Daniel Husfeldt 197349

WINTERSEMESTER 2017/2018

**Inhaltsverzeichnis**

[Einleitung 1](#_Toc502350252)

[Aufgaben- und Zielstellung 1](#_Toc502350253)

[Programmablauf 3](#_Toc502350254)

[Bildvorverarbeitung 5](#_Toc502350255)

[Erstellung eines Grauwertbilds 5](#_Toc502350256)

[Gaußfilterung 5](#_Toc502350257)

[Binarisierung 5](#_Toc502350258)

[Morphologische Operationen 6](#_Toc502350259)

[Objekterkennung und Objektextraktion 7](#_Toc502350260)

[Konturenfindung 7](#_Toc502350261)

[Merkmale der Verkehrsschilder 7](#_Toc502350262)

[Objektspeicherung 9](#_Toc502350263)

[Unterscheidung der Schildarten 10](#_Toc502350264)

[Buchstabenextraktion 13](#_Toc502350265)

[Hierarchie 13](#_Toc502350266)

[Buchstabenspeicherung 13](#_Toc502350267)

[Buchstabenerkennung 14](#_Toc502350268)

[Buchstaben-Templates 15](#_Toc502350269)

[Abgleich mit den Templates 18](#_Toc502350270)

[Anzeige der Ergebnisse 20](#_Toc502350271)

[Arbeitsbedingungen des Bildverarbeitungssystems 21](#_Toc502350272)

[Fazit und Ausblick 21](#_Toc502350273)

**Abbildungsverzeichnis**

[Abbildung 1: Ortsschild 2](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350274)

[Abbildung 2: Pfeilwegweiser 2](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350275)

[Abbildung 3: Pfeilwegweiser innerorts 2](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350276)

[Abbildung 4: Pfeilwegweiser Autobahn 2](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350277)

[Abbildung 5: Programmablaufplan 4](#_Toc502350278)

[Abbildung 6: Ortsschild nach Bildvorverarbeitung 6](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350279)

[Abbildung 7: Ortsschild als RGB-Farbbild 6](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350280)

[Abbildung 8: Erklärung zum Formfaktor 7](#_Toc502350281)

[Abbildung 9: Erläuterung der OpenCV-Funktion „inRange“ [5] 11](#_Toc502350282)

[Abbildung 10: Ergebnis der Objekterkennung und Objektextraktion 12](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350283)

[Abbildung 11: Ergebnis der Buchstabenextraktion 14](#_Toc502350284)

[Abbildung 12: Ortsschild "abcdefghij" 15](#_Toc502350285)

[Abbildung 13: Extrahierter Buchstabe "b" 16](#_Toc502350286)

[Abbildung 14: Fertiges Template für den Buchstaben "b" 17](#_Toc502350287)

[Abbildung 15: Extrahierter Buchstabe 19](#_Toc502350288)

[Abbildung 16: Template 19](#_Toc502350289)

[Abbildung 17: XNOR-Bild 19](#_Toc502350290)

[Abbildung 18: Endergebnis im Konturenbild 20](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350291)

[Abbildung 19: Schieberegler 20](file:///C:\Users\Amelie\Dropbox\Optische_Systeme_2\05_Dokumentation\Willig_Husfeldt_Dokumentation_Bildverarbeitung.docx#_Toc502350292)

# Einleitung

Der Mensch fällt täglich tausende von Entscheidungen und beurteilt dabei verschiedenste Situationen anhand von Bilddaten, die er über sein Auge wahrnimmt. Innerhalb von Sekundenbruchteilen, können hierbei komplexe Objekte aus einer Szene extrahiert und klassifiziert werden. Dies ist auch für viele technische Anwendungen von hoher Bedeutung.

Daher kommen Bilddatenverarbeitungssysteme heute in vielen technischen Systemen zum Einsatz und finden in nahezu allen Industriezweigen Anwendung. Von der visuellen Inspektion im Rahmen von Qualitätskontrollen über die Orientierung von autonomen Systemen, die Identifikation von Personen oder die Erkennung von Zeichen, sind vielerlei Anwendungsmöglichkeiten denkbar. [1]

Im Rahmen der Vorlesung „Optische Systeme“ bzw. „Ausgewählte Kapitel der Bildverarbeitung“ soll eine praktische und industrienahe Aufgabenstellung als Abschlussprojekt bearbeitet werden, um Erfahrungen in der Implementierung von Bildverarbeitungssystemen zu sammeln.

# Aufgabenstellung und Zielsetzung

Das Abschlussprojekt soll mithilfe der freien Bildverarbeitungsprogrammbibliothek OpenCV und unter Verwendung der Programmiersprache C++ realisiert werden.

Das Projektthema lautet: **„Erkennung von Ortsschildern und Pfeilwegweisern“**. Hierbei sollen verschiedene Schilder erkannt und klassifiziert werden. Zudem soll aus den Ortstafeln bzw. Pfeilwegweisern die Schrift ausgelesen werden können. Die nachfolgenden Abbildungen geben einen Überblick über die unterschiedlichen Schildtypen. Abbildung 1 zeigt ein typisches Ortsschild. In Abbildung 2, Abbildung 3 und Abbildung 4 sind klassische Pfeilwegweiser dargestellt, wobei sich der Pfeilwegweiser mit der Aufschrift „Berlin“ auf der Autobahn befindet. Der Pfeilwegweiser mit der Aufschrift „Bahnhof“ bezieht sich auf Ziele, die innerorts zu erreichen sind. Der gelbe Pfeilwegweiser weist zu Zielen, für die eine Überlandfahrt auf der Landstraße notwendig wird.

Das zu realisierende Bildverarbeitungssystem soll problemlos ausgegeben können, ob sich das Schild innerorts, auf der Landstraße oder auf der Autobahn befindet.

|  |  |
| --- | --- |
| Ähnliches Foto  Abbildung : Ortsschild | Abbildung : Pfeilwegweiser |
| Abbildung : Pfeilwegweiser innerorts | Bildergebnis für autobahnschild blau  Abbildung : Pfeilwegweiser Autobahn |

Zusätzlich gelten die folgenden Rahmenbedingungen: Die Schilder sollen zum einen natürlich erkannt, klassifiziert und ausgelesen werden, wenn sie dem Bildverarbeitungssystem als Datei vorliegen. Zum anderen soll das System auch dann stabil funktionieren, wenn die Schilder auf einem weißen Hintergrund per Webcam eingelesen werden. Als optionale Anforderung wird eine Erkennung und Klassifizierung von Schildern in realen Verkehrssituationen festgesetzt.

# Programmablauf

Zur besseren Übersicht soll zunächst der grobe Programmablauf anhand des Programmablaufplans in Abbildung 5 kurz dargestellt werden. Die einzelnen Schritte des Programmablaufs werden in diesem Kapitel lediglich kurz genannt. Eine detaillierte Erläuterung erfolgt erst im weiteren Verlauf dieser Dokumentation.

Das Bildverarbeitungssystem beginnt mit dem Einlesen eines Frames. Hierbei muss durch den Benutzer vor dem Start des Programmes der Dateipfad gewählt werden, unter dem das gewünschte Bild auf der Festplatte abgelegt wurde. Alternativ kann auch die Webcam als Bildquelle eingestellt werden. Ein Umschalten zwischen der Webcam und einem gespeicherten Frame ist hierbei auch noch während der Programmlaufzeit per Schieberegler möglich.

Nach dem Einlesen des gewünschten Bildes findet die Bildvorbearbeitung statt, die sich aus der Umwandlung in ein Grauwertbild, einer Gaußfilterung, der Binarisierung und einer Behandlung mit morphologischen Filtern (Erodieren und Dilatieren) zusammensetzt.

An die Bildvorbearbeitung schließt sich die Objekterkennung und Objektextraktion an. Hierzu werden zunächst alle Konturen im Bild gefunden und abgespeichert. Die einzelnen Konturen werden anhand von definierten Merkmalen untersucht und nur dann auch als relevantes Objekt abgespeichert, wenn die Merkmale darauf hinweisen, dass es sich um ein Verkehrsschild handelt.

Anschließend wird das Verkehrsschild anhand seiner Farbe als Schild auf der Landstraße, Autobahn bzw. als Schild innerorts klassifiziert.

Aus den gefundenen Objekten bzw. Verkehrsschildern werden im nächsten Schritt alle Buchstaben extrahiert und abgespeichert. Anhand von Templates erfolgt die Erkennung bzw. Klassifizierung der extrahierten Buchstaben.

Im letzten Schritt wird dem Benutzer als Ergebnis die Art des erkannten Verkehrsschilds sowie der erkannte Text angezeigt. Das Programm startet automatisch einen neuen Durchlauf, wenn es nicht vom Benutzer durch Eingabe einer beliebigen Taste unterbrochen wird.



Abbildung : Programmablaufplan

# Bildvorverarbeitung

Um das eingelesene Bild für die weiteren Verarbeitungsstufen der Merkmalsextraktion und Klassifikation aufzubereiten, ist zunächst eine Bildvorverarbeitung notwendig. [2]

## Erstellung eines Grauwertbilds

Im ersten Schritt wird das eingelesene dreikanalige RGB-Farbbild in ein einkanaliges Grauwertbild umgewandelt. Dazu wird die Funktion cvtColor mit dem Parameter   
COLOR\_BGR2GRAY eingesetzt. Die Umwandlung ist notwendig, da die nachfolgenden Bildvorverarbeitungsstufen nur mit einem Grauwertbild möglich sind.

## Gaußfilterung

Zur Eliminierung von statistischen Messfehlern und zur Glättung des Bildinhaltes wird das Grauwertbild im Rahmen der Bildvorverarbeitung einer Gaußfilterung unterzogen. Der Gaußfilter gehört zur Gruppe der linearen Lokaloperatoren. [2] Durch Ausprobieren wurde herausgefunden, dass ein 3x3-Gaußfilter mit der Standardabweichung 1 für unseren Anwendungsfall gut geeignet ist und das Bildrauschen angemessen vermindert.

## Binarisierung

Um das mit dem Gaußfilter bearbeitete Bild für eine Kantenerkennung weiter vorzubereiten, wird ein Binärbild erstellt. Im Binärbild können alle Pixel entweder den Wert 0 (schwarz) oder 255 (weiß) annehmen. Für die Binarisierung wurde zunächst ein einfaches Schwellenwertverfahren mit dem Befehl setTo implementiert. Im weiteren Verlauf des Projekts hat sich jedoch herausgestellt, dass dieses Verfahren bei Verwendung der Webcam unter schlechten Beleuchtungssituationen zu falschen Ergebnissen bei der Binarisierung führt.

Deswegen haben wir entschieden, für die Binarisierung auf die folgenden zwei Methoden zurückzugreifen:

* adaptiveThreshold: Bei der OpenCV-Funktion adaptiveThreshold handelt es sich um ein Verfahren mit einem adaptiven Schwellwert. Als Methode für den adaptiveThreshold wird der Parameter ADAPTIVE\_THRESH\_GAUSSIAN\_C gewählt. Dieser verwendet ein Gaußfenster mit wählbarer Blockgröße zur Gewichtung der Werte der Nachbarpixel und berechnet daraus einen adaptiven Schwellwert. [3]
* CV\_THRESH\_OTSU: Mit der OpenCV-Funktion CV\_THRESH\_OTSU wird ein globaler Grauschwellwert ermittelt, mit dem das Bild binarisiert wird.

## Morphologische Operationen

Um Fehler bzw. Fehlzuweisungen zu eliminieren, die bei der Binarisierung auftreten, können in Bildverarbeitungssystemen sogenannte morphologische Filter zum Einsatz kommen. [4] Im vorliegenden Projekt wurden die eigenen Funktionen dilatieren und erodieren implementiert. Für eine möglichst gute Bildvorverarbeitung hat es sich als hilfreich herausgestellt, das Verfahren des sogenannten „Closing“ (Fermenture) anzuwenden. Hierbei wird das Binärbild im ersten Schritt dilatiert und anschließend sofort wieder erodiert. [4] Dadurch wird erreicht, dass Lücken im Rand der Schilder sowie in der Beschriftung bzw. den Buchstaben geschlossen werden. Die Maskengröße für die selbst implementierten morphologischen Filter wird über die Parameter dilation\_size und erosion\_size gesteuert und ist im Programm per Schieberegler variabel einstellbar.

In den nachfolgenden Abbildungen ist das Ergebnis zu erkennen, das durch die Bildvorverarbeitung erreicht wird. Im linken Bild (Abbildung 7) ist ein Ortsschild als originales RGB-Farbbild dargestellt. Rechts daneben (Abbildung 6) ist das Bild nach seiner Bildvorverarbeitung zu erkennen. Es handelt sich um das gaußgefilterte und mit morphologischen Filtern bearbeitete Binärbild.

|  |  |
| --- | --- |
| Abbildung : Ortsschild als RGB-Farbbild | Abbildung : Ortsschild nach Bildvorverarbeitung |

# Objekterkennung und Objektextraktion

Nachdem die Bildvorverarbeitung abgeschlossen ist, liegt ein sauber aufbereitetes Binärbild vor, welches nun für die Merkmalsextraktion und Klassifikation verwendet werden kann.

## Konturenfindung

Im ersten Schritt werden die Konturen im vorbearbeiteten Binärbild mithilfe der OpenCV-Funktion findContours gefunden und im Vektor vector<vector<Point>>contours abgespeichert. Zusätzlich wird die Hierarchie, d.h. die Beziehung der Konturen untereinander, im Vektor vector<Vec4i> hierarchy abgelegt.

## Merkmale der Verkehrsschilder

Nachdem alle Konturen aus dem Binärbild extrahiert wurden, müssen diejenigen Konturen selektiert werden, die ein Ortsschild oder einen Pfeilwegweiser darstellen. In der Bildverarbeitung ist hierzu eine präzise Beschreibung von Objekten über Merkmale notwendig.

Für die Objektselektion wurden daher verschiedene Bedingungen bzw. Merkmale definiert, welche die Schilder genau klassifizieren. Folgende Merkmale kommen zum Einsatz:

* **Formfaktor**

Ein wesentliches Merkmal, das für die Realisierung des Bildverarbeitungssystems genutzt wurde, ist der sogenannte Formfaktor. Dieser ist definiert als das Verhältnis der Fläche eines Objektes zum Umfangsquadrat und berechnet sich gemäß folgender Formel (siehe Abbildung 8). Er ist somit ein Maß für die „Kreisähnlichkeit“. [4]

|  |  |
| --- | --- |
| Abbildung : Erklärung zum Formfaktor |  |

Der Vorteil des Formfaktors ist, dass er translationsinvariant, skaleninvariant und rotationsinvariant ist. [4]

Es hat sich gezeigt, dass der Formfaktor ein vollkommen ausreichendes Mittel darstellt, um Pfeilwegweiser und Ortsschilder zu klassifizieren. Erfahrungsgemäß bewegt er sich hierbei meist im Bereich zwischen 0,6 und 0,8. Die gewünschte obere und untere Grenze für den Formfaktor kann innerhalb des Programms (auch während der Laufzeit) per Schieberegler eingestellt werden.

* **Anzahl der Konturen (=Buchstaben) innerhalb eines Schildes**

Ein weiteres Merkmal zur Klassifizierung der Schilder ist die Anzahl der Buchstaben innerhalb eines Schildes. Um diese zu bestimmen, wurde die eigene Funktion count\_childs implementiert. An die Funktion count\_childs werden der Hierarchie-Vektor, sowie die Kontur-Identifikationsnummer übergeben. Die Funktion zählt, wieviele Kinder („Childs“) der jeweiligen Kontur („Parent“) zugewiesen sind. Übertragen auf das Projekt entspricht das „Parent“ dem Verkehrsschild und die „Childs“ entsprechen den Buchstaben innerhalb des Schildes. Definitionsgemäß muss ein Verkehrsschild mindestens 3 Buchstaben enthalten, um vom Bildverarbeitungssystem als Schild erkannt zu werden. Dies wird im Code durch die Zeile   
bool n\_childs\_smaller\_min = object.n\_childs > 3 abgeprüft.

* **Minimale Schildgröße**

Die nächsten beiden Merkmale, die zur Klassifizierung der Konturen verwendet werden, beziehen sich auf die Größe der Kontur bzw. des Schildes. Es kam teilweise das Problem auf, dass kleine Objekte im Hintergrund als Ortsschild klassifiziert wurden, da sich ihr Formfaktor im Bereich um 0,7 bewegte. Um dieser Fehlklassifikation entgegenzuwirken, wird als Merkmal für die Schilder definiert, dass diese mindestens 1% der Bildfläche einnehmen müssen.

* **Maximale Schildgröße**

Analog zum obigen Merkmal wird vorsichtshalber auch die maximale Schildgröße auf 90% der gesamten Bildfläche begrenzt.

Innerhalb einer for-Schleife werden alle Konturen durchlaufen. Hierbei werden die oben genannten Bedingungen/Merkmale abgeprüft. Falls eine Kontur alle Bedingungen erfüllt, wird diese als relevantes Objekt bzw. Verkehrsschild klassifiziert, erhält eine grüne Umrahmung mit einer Bounding-Box und wird in einer eigenen Objektstruktur abgespeichert. Diese Objektstruktur wird im folgenden Kapitel „Objektspeicherung“ nochmal ausführlich erläutert.

## Objektspeicherung

Beim Durchlaufen aller Konturen werden anhand der definierten Merkmale die relevanten Konturen als Verkehrsschild klassifiziert. Um auch später noch auf diese relevanten Objekte zugreifen zu können, müssen sie abgespeichert werden.

Die Anforderung hierbei ist, dass zu jedem Zeitpunkt im Programmverlauf alle Daten der Kontur, wie beispielsweise der Formfaktor oder die Hierarchie, zur Verfügung stehen. Um dieser Anforderung gerecht zu werden, wurde eine innovative Kombination aus einer Struktur und einem Vektor als Objektspeicher gewählt. Die Struktur sieht wie folgt aus und bündelt alle Objekteigenschaften.

struct object {

unsigned short Contour\_ID; //Kontur-Identifikationsnr.

unsigned short Object\_ID; //Objekt-Identifikationsnr.

vector<Vec4i> aiHierarchy; //Hierarchie-Vektor

int n\_childs; //Anzahl Buchstaben im Schild

double contour\_length; //Konturlänge

double area; //Fläche

double form\_factor; //Formfaktor

string plate\_type; //Typ des erkannten Schildes

Rect box; //Bounding box

int x; //x-Koordinate

int y; //y-Koordinate

};

Den Variablen innerhalb der Struktur kann jederzeit ein neuer Wert zugewiesen werden. Nachfolgend wird beispielsweise eine Kontur-ID gesetzt:

object.Contour\_ID = 53;

Selbstverständlich sollen nur diejenigen Konturen abgespeichert werden, welche auch als Schild klassifiziert wurden. Dies wird folgendermaßen mithilfe eines Vektors vom Struktur-Typ object realisiert:

vector<object> object\_vector; //Vektor des structs „object“

Durch Verwendung des push\_back-Operators kann die gesamte Struktur nun ganz leicht auf dem Vektor abgelegt werden. Dies wird schließlich durch folgende Codezeile beispielhaft durchgeführt:

object\_vector.push\_back(object)

## Unterscheidung der Schildarten

Wie bereits in der Aufgabenstellung und Zielsetzung beschrieben, soll das Bildverarbeitungssystem weiße Pfeilwegweiser (innerorts), gelbe Pfeilwegweiser (Überlandfahrt notwendig) und blaue Pfeilwegweiser (Autobahn) unterscheiden können.

Die Unterscheidung erfolgt anhand der Farbe im HSV-Farbraum. Mithilfe folgender Parameter kann im HSV-Farbraum eine Farbe exakt definiert werden: [4]

* Hue: Farbwert bzw. Winkel auf dem Farbkreis von 0°-360°
* Saturation: Sättigungswert von 0% (neutralgrau) - 100% (Farbe in Sättigung)
* Value: Dunkelstufe von 0% (dunkel) - 100% (hell)

Nachdem die Kontur des Verkehrsschilds auf dem Objektspeicher-Vektor abgelegt wurde, kann mit der Unterscheidung der Schildarten begonnen werden. Zu diesem Zweck wird das Verkehrsschild zunächst mithilfe der abgespeicherten „Bounding-Box“ aus dem originalen RGB-Farbbild extrahiert. Anschließend erfolgt eine Umwandlung des RGB-Bildausschnitts in den HSV-Farbraum mit der bereits bekannten Funktion cvtColor und dem Parameter   
COLOR\_BGR2HSV.

Durch Verwendung der OpenCV-Funktion inRange wird das HSV-Farbbild dahingehend untersucht, ob es ein definiertes Spektrum an Farbwerten enthält. Dies soll für die blauen Farbanteile nachfolgend genauer erläutert werden:

In der untenstehenden Codezeile wird der Funktion inRange als ersten Parameter das HSV-Farbbild übergeben.

inRange(hsv, Scalar(110,50,50), Scalar(130,255,255), mask\_blue);

Mit dem zweiten und dritten Parameter werden der Funktion 2 Blautöne als HSV-Farbwerte zur Verfügung gestellt. Sie überprüft nun für jeden Pixel, ob sich der Farbwert innerhalb der Spannweite der beiden Blautöne befindet. Falls dies der Fall ist, wird in der Maske für die Blauanteile (mask\_blue) eine 1 bzw. ein weißer Pixel hinterlegt. Dieser Zusammenhang wird durch die Abbildung 9 nochmal grafisch anhand eines blauen Balls verdeutlicht.



Abbildung : Erläuterung der OpenCV-Funktion „inRange“ [5]

Anschließend wird mit der OpenCV-Funktion countNonZero die Anzahl der weißen Pixel in der Maske gezählt. Mithilfe der folgenden Gleichung wird die Anzahl der weißen Pixel auf die Gesamtzahl der Pixel im Bild normiert. Somit ergibt sich für den Koeffizienten immer eine Zahl zwischen 0 und 1.

Das Vorgehen wird analog für die Gelb- und Weißanteile im Verkehrsschild wiederholt. Durch Bestimmung des Maximums der drei Koeffizienten kann eine Aussage über die Art des Schildes getroffen werden. Ist beispielsweise der Koeffizient mit den Blauanteilen am höchsten, so handelt es sich um ein Autobahnschild wie in Abbildung 4 (siehe Seite 2).

Das Ergebnis der Klassifizierung wird dem Benutzer des Bildverarbeitungssystems am unteren Rand des erkannten Schildes in gelber Schrift angezeigt.

In der folgenden Abbildung wird zusammenfassend das Ergebnis der Objekterkennung und Objektextraktion angezeigt. Dem Programm wurde hierbei das links dargestellte, leere Ortsschild (ohne Beschriftung) übergeben (siehe Abbildung 10). Das Ortsschild wird im Konturenbild durch die beschriebenen Merkmale erkannt bzw. klassifiziert und daher mit einer grünen Bounding-Box umrahmt. Die Schildart „Ortsschild/Pfeilwegweiser“ wird angezeigt, da es sich um ein gelbes Schild handelt.

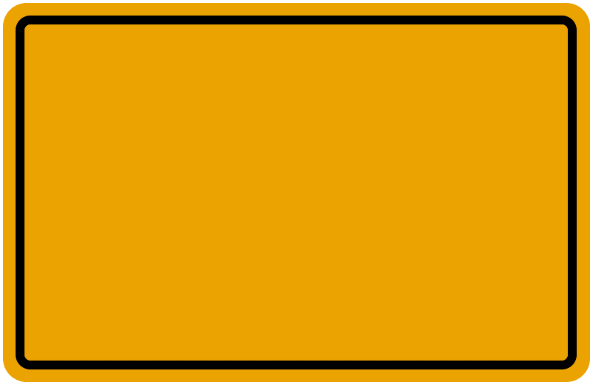




Abbildung : Ergebnis der Objekterkennung und Objektextraktion

# Buchstabenextraktion

Im vorigen Kapitel wurde beschrieben, wie das Bildverarbeitungssystem die Extraktion und Klassifizierung der Verkehrsschilder vornimmt. Dieses Kapitel beschäftigt sich nun mit der Extraktion des Textes bzw. der Buchstaben aus den gefundenen Schildern.

## Hierarchie

In OpenCV stellt jede Kontur Informationen über Ihre Hierarchie-Struktur bereit. Dies geschieht über den Hierarchie-Vektor, welcher die folgenden 4 Werte umfasst: [6]

Hierarchie = [Next, Previous, First\_Child, Parent]

Bei der Extraktion der Buchstaben aus den Verkehrsschildern spielt der Hierarchie-Vektor daher eine entscheidende Rolle. Es wird zunächst innerhalb der Kontur des Verkehrsschildes („Parent“) die erste Buchstabenkontur („First\_Child“) extrahiert. Anschließend werden so lange die nächsten Buchstabenkonturen („Next“) extrahiert, bis innerhalb des Verkehrsschilds keine weiteren Konturen mehr vorkommen.

Es hat sich herausgestellt, dass trotz sorgfältiger Bildvorverarbeitung mit morphologischen Filtern teilweise Konturen innerhalb der Schilder übrigbleiben, die keine Buchstaben sind. Hierbei handelt es sich fast ausschließlich um Ansammlungen von maximal 3x3 Pixeln. Der Effekt ist zum Großteil auf, bei der Binarisierung, eintreffende Zuweisungsfehler zurückzuführen.

Aus diesem Grund werden nur diejenigen Konturen innerhalb der Verkehrsschilder als Buchstaben erkannt, deren Bounding-Box eine Mindestgröße von 5x5 Pixeln übersteigt.

## Buchstabenspeicherung

Um auch später noch im Rahmen der Buchstabenerkennung auf die einzelnen Buchstaben zugreifen zu können, bedarf es einer Speicherung der zugehörigen Konturen. Hierfür wurde ein ähnlicher Ansatz gewählt, wie er bereits im Kapitel „Objektspeicherung“ beschrieben wurde.

Die Konturen der einzelnen Buchstaben werden hierbei im Vektor   
vector<vector<Point>>letter\_contours abgelegt, während ihre Bounding-Boxen auf dem Vektor vector<Rect>letter\_boxes Platz finden.

In der folgenden Abbildung ist das Ergebnis der Buchstabenextraktion im Konturenbild erkennbar. Alle Buchstaben im dargestellten Ortsschild wurden dabei mithilfe von blauen Bounding-Boxen extrahiert und abgespeichert.

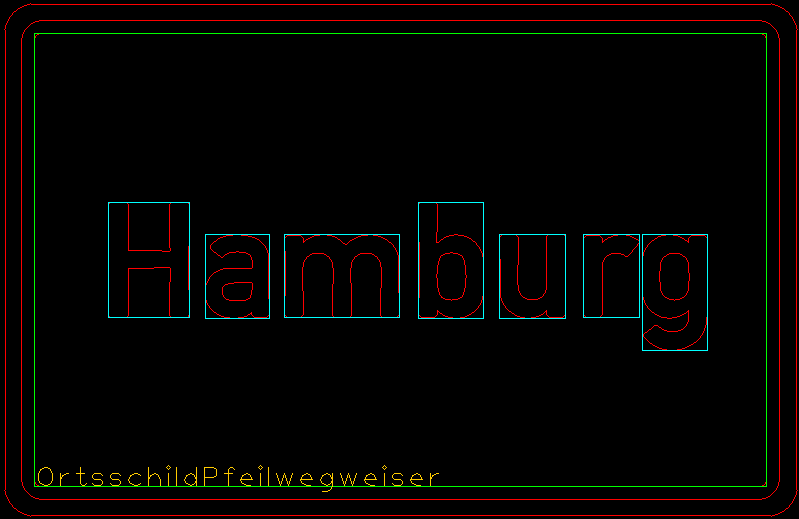


Abbildung : Ergebnis der Buchstabenextraktion

# Buchstabenerkennung

Im vorigen Kapitel wurden alle Buchstaben aus den Verkehrsschildern extrahiert und abgespeichert. Die nächste Herausforderung stellt nun dar, die extrahierten Buchstaben zu klassifizieren, sodass der Text aus den Schildern korrekt ausgelesen werden kann. Um dies zu realisieren, gibt es unter anderem die folgenden zwei Möglichkeiten:

* Verwendung einer Bibliothek für die OCR („**o**bject **c**haracter **r**ecognition“)
* Erstellung von Templates mit den relevanten Buchstaben und Zahlen und anschließender Abgleich der extrahierten Buchstaben mit den Templates

Im Rahmen des Projekts haben wir uns bewusst für die Methode der Templates entschieden, da sich hier die Möglichkeit bietet, die gesamte Buchstabenerkennung von Grund auf selbst zu implementieren und zu gestalten.

## Buchstaben-Templates

Der erste Schritt zur Realisierung einer Buchstabenerkennung mittels Templates ist die Erstellung einer möglichst realitätsnahen Datenbank, die alle möglichen Buchstaben und Zahlen enthält. Hierzu gehören 26 Großbuchstaben, 26 Kleinbuchstaben, sowie 9 Zahlen und einige Sonderzeichen. (ä, ö, ü etc.) Es ist dabei essentiell, die gleiche Schriftart zu verwenden, die auch in den Verkehrsschildern vorkommt. Dadurch wird eine möglichst hohe Erkennungsrate sichergestellt. Es hat sich als sehr hilfreich erwiesen, dass alle Verkehrsschilder in Deutschland eine einheitliche Schriftart nach DIN1451 verwenden. [7] Um möglichst realitätsnahe Templates für die Buchstabenerkennung zu erstellen, wurde die folgende Herangehensweise gewählt:

Im Internet existieren frei verfügbare „Schild-Generatoren“, mit deren Hilfe Ortstafeln mit beliebigem Text erstellt werden können. Mit einem solchen Generator wurde beispielsweise das Schild in Abbildung 12 mit der Aufschrift „abcdefghij“ als .png-Datei generiert. Selbstverständlich wurde das nachfolgend beschriebene Vorgehen nicht nur mit den Buchstaben „a“ bis „j“ durchgeführt, sondern es wurden nacheinander Schilder mit allen Buchstaben und Zahlen generiert.



Abbildung : Ortsschild "abcdefghij"

Die .png-Dateien mit den Schildern lassen sich leicht in das bis hierher realisierte Bildverarbeitungssystem einlesen. Die Schilder und die Buchstaben werden dabei sofort erkannt und mit Bounding-Boxen versehen. Die Buchstaben werden daraufhin als Binärbild auf der Festplatte abgespeichert.

In Abbildung 13 wird beispielhaft das Binärbild des Kleinbuchstabens „b“ dargestellt. An diesem soll nun erläutert werden, wie die weitere Aufbereitung bis zum Template vonstattengeht.



Abbildung : Extrahierter Buchstabe "b"

Wichtig für den späteren Abgleich der Templates mit den extrahierten Buchstaben ist, dass deren Größe exakt übereinstimmt. Das bedeutet zunächst, dass alle Templates dieselbe Größe (Breite und Höhe in Pixeln) besitzen müssen. Für die Templates musste daher eine fest definierte Höhe und Breite gefunden werden. Hierbei hat es sich als sinnvoll erwiesen, die Höhe und Breite des größten Buchstabens im Alphabet zu wählen. Das war in diesem Fall der Großbuchstabe „W“, der eine Breite von 74 Pixeln (template\_width) und eine Höhe von 69 Pixeln (template\_height) besitzt.

Um auch alle anderen Buchstabentemplates auf die Höhe von 69 Pixeln zu skalieren, ohne sie hierbei zu verzerren, muss mit einem Skalierungsfaktor gearbeitet werden. Er berechnet sich gemäß folgender Gleichung:

Die neue Breite des Templates ergibt sich dann mithilfe des Skalierungsfaktors nach folgendem Zusammenhang:

Mithilfe der OpenCV-Funktion resize und der berechneten neuen Breite wird das Template nun verzerrungsfrei auf die gewünschte Höhe von 69 Pixeln skaliert.

Für den späteren Abgleich der Templates mit den, aus den Verkehrsschildern extrahierten Buchstaben, reicht es nun nicht aus, dass deren Höhe übereinstimmt. Aus diesem Grund wird als nächstes die Breite betrachtet. Hierbei fällt auf, dass der Kleinbuchstabe „b“ zum Beispiel deutlich schmaler ist, als der Großbuchstabe „W“. Um den Kleinbuchstaben „b“ trotzdem verzerrungsfrei auf die Breite von 74 Pixeln (template\_width) anzupassen, wird dieser links im Template platziert und die nicht verwendeten Pixel werden mit 0 (schwarz) aufgefüllt. In   
OpenCV wird hierzu ein leeres Mat mit der Funktion zeros und den Parametern  
template\_width sowie template\_height initialisiert. Anschließend wird das leere Mat in einer Schleife Pixel für Pixel mit dem Bildinhalt des Buchstabens „b“ befüllt. Abbildung 14 zeigt das Ergebnis und das damit fertiggestellte Template für den Kleinbuchstaben „b“. Das Template besitzt wie gewünscht die Breite von 74 Pixeln und die Höhe von 69 Pixeln.



Abbildung : Fertiges Template für den Buchstaben "b"

Das beschriebene Vorgehen wurde analog für alle Groß- und Kleinbuchstaben des Alphabets, sowie für die Zahlen angewendet. Es ergibt sich eine sehr umfangreiche Template-Datenbank, die im Projektverzeichnis unter dem Ordnernamen „Template“ abgelegt wurde.

## Abgleich mit den Templates

Nachdem nun alle zu erkennenden Zeichen in einer einheitlichen Höhe und Breite als Datenbank angelegt sind, kann mit der eigentlichen Buchstabenerkennung begonnen werden. Grundgedanke hierbei ist es, die aus den Verkehrsschildern extrahierten Buchstaben nacheinander mit allen Templates aus der Datenbank abzugleichen.

Das Template mit der höchsten Übereinstimmung wird ausfindig gemacht und dem extrahierten Buchstaben der entsprechende Wert zugewiesen. Im folgenden Abschnitt soll genauer erläutert werden, wie der Abgleich zwischen den extrahierten Buchstaben und den Templates erfolgt.

Dazu wird jeder aus dem Verkehrsschild extrahierte Buchstabe zunächst gleichermaßen skaliert, wie es im vorigen Kapitel beschrieben wurde. Somit liegt der Buchstabe nun analog zu den Templates in der Größe template\_height\*template\_width vor.

Der Abgleich zwischen den extrahierten Buchstaben und den Templates erfolgt durch eine bitweise XNOR-Verknüpfung der beiden Bilder. Dieser Zusammenhang soll auch durch die folgenden drei Abbildungen verdeutlicht werden. In Abbildung 15 ist der Buchstabe „b“ zu erkennen, der aus einem zufällig ausgewählten Verkehrsschild extrahiert wurde. In Abbildung 16 ist das Template dargestellt, für welches das Programm die höchste Übereinstimmung gefunden hat. Mithilfe der beiden OpenCV-Funktionen bitwise\_xor und bitwise\_not wurde eine XNOR-Verknüpfung von Buchstabe (Abbildung 15) und Template (Abbildung 16) durchgeführt. Das Ergebnis der XNOR-Verknüpfung wird in Abbildung 17 dargestellt.



Abbildung : Extrahierter Buchstabe



Abbildung : Template



Abbildung : XNOR-Bild

Im Falle einer Übereinstimmung zwischen dem extrahierten Buchstaben und dem Template wird im XNOR-Bild ein weißes Pixel erzeugt. Durch einfaches Zählen der weißen Pixel im XNOR-Bild mit der OpenCV-Funktion countNonZero kann die Übereinstimmung zwischen dem extrahierten Buchstaben und dem Template bestimmt werden. Im Projekt wurde diese Zahl gemäß folgender Gleichung auf die Größe der Templates normiert.

Damit ergibt sich als Korrelationskoeffizient immer eine Zahl zwischen 0 und 1. Im Programm werden alle Templates schrittweise durchgegangen. Das Template, das bei der XNOR-Verknüpfung mit dem zu bestimmenden Buchstaben den größten Korrelationskoeffizienten erzeugt, wird ausfindig gemacht und das entsprechende Zeichen zurückgegeben.

# Anzeige der Ergebnisse

In den letzten Kapiteln wurden die Einzelfunktionalitäten des Bildverarbeitungssystems zur Schilderkennung detailliert erläutert. Weniger Beachtung ist hierbei bisher der Ergebnisanzeige und der Bedienung des Systems zugekommen. In Abbildung 19 sind die Schieberegler dargestellt, mit denen einige Parameter sowie Einstellungen geändert werden können. Hierzu gehören unter anderem Parameter für den adaptiven Schwellwert bei der Binarisierung, Einstellungen für die morphologischen Filter, Parameter für den Formfaktor etc. Außerdem ist auch die Extraktion und Erkennung von Buchstaben per Schieberegler aktivierbar bzw. deaktivierbar. In Abbildung 18 ist das Ergebnis des Programms für das Ortsschild von Hamburg dargestellt. Das Ortsschild wird als solches erkannt, mit einer grünen Bounding-Box versehen und klassifiziert. Unten links wird der Schildtyp „Ortsschild/Pfeilwegweiser“ angezeigt. Alle Buchstaben innerhalb des Schildes werden extrahiert, was auch durch die hellblauen Bounding-Boxen erkennbar ist. Zudem werden die extrahierten Buchstaben klassifiziert und das Ergebnis der Buchstabenerkennung wird in Gelb über jedem Zeichen angezeigt.

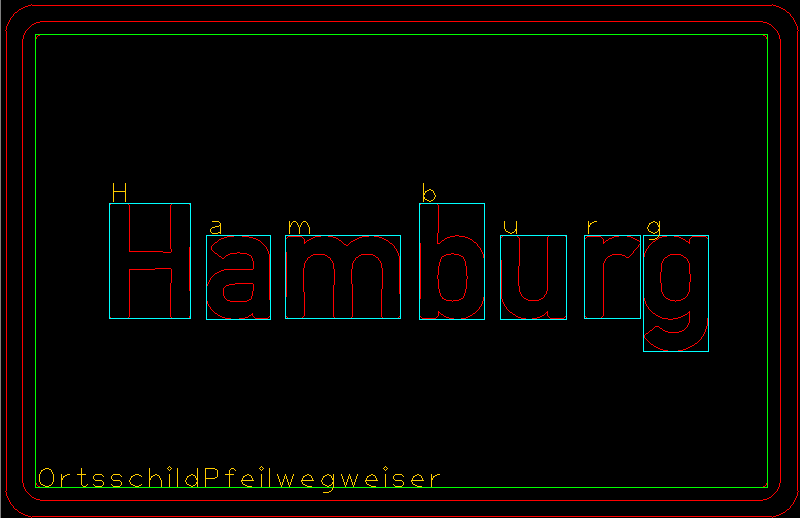


Abbildung : Endergebnis im Konturenbild

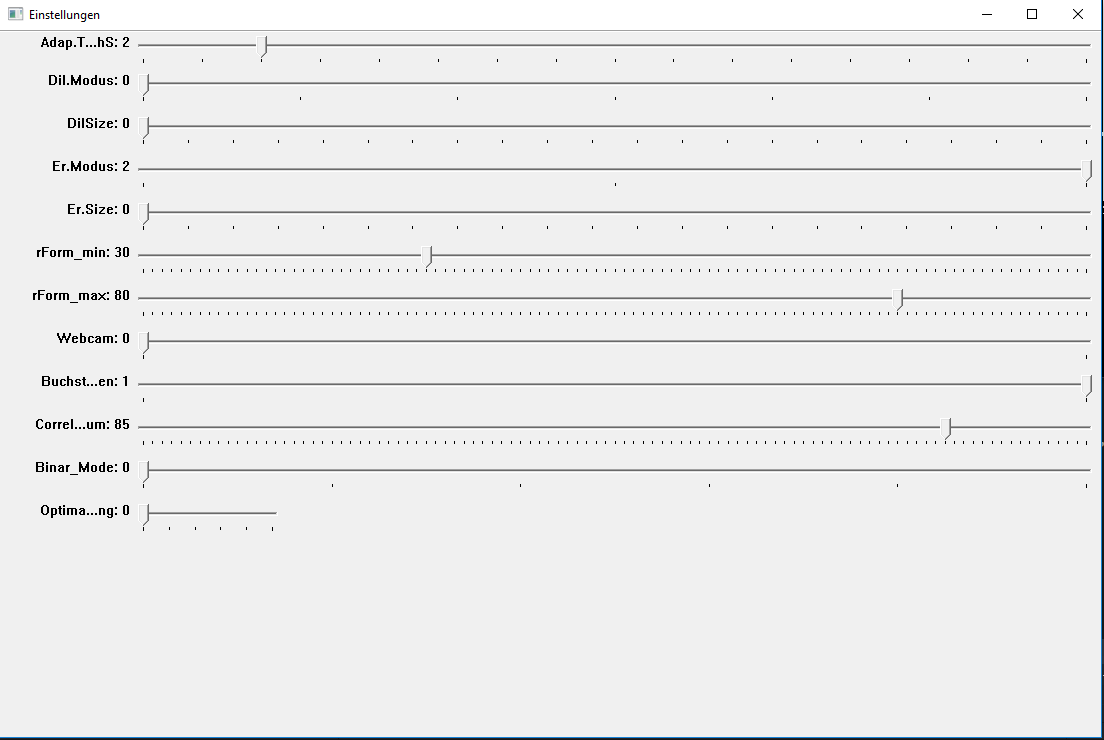


Abbildung : Schieberegler

# Arbeitsbedingungen des Bildverarbeitungssystems

Für die Präsentation der Ergebnisse des Bilddatenverarbeitungslabors sollen die Arbeitsbedingungen festgehalten werden, unter denen das Programm problemlos funktioniert und korrekte Ergebnisse liefert. Es gelten die folgenden Restriktionen.

* Das Verkehrsschild darf maximal 90% der gesamten Bildfläche einnehmen
* Das Verkehrsschild muss mindestens 10% der gesamten Bildfläche einnehmen
* Die Buchstaben in den Schildern müssen größer als 5x5 Pixel sein
* Die Verkehrsschilder dürfen nicht verdreht werden
* Beleuchtung mit farbigem Licht ist unzulässig
* Ortsnamen unter 3 Zeichen sind unzulässig
* Das Ausschalten der Beleuchtung ist unzulässig

# Fazit und Ausblick

Alle Pflichtziele, die zu Anfang des Bildverarbeitungs-Projekts gesetzt wurden, konnten erfolgreich umgesetzt werden. Es wurde ein Bildverarbeitungssystem implementiert, das unter Beachtung der Arbeitsbedingungen jegliche Ortsschilder und Pfeilwegweiser aus Bildern extrahiert, klassifiziert und deren Text ausliest.

Mit etwas zusätzlicher Zeit hätte noch eine Sortierung der Buchstaben sowie die Ausgabe des Schildinhalts über die Konsole umgesetzt werden können.

Natürlich haben wir gemerkt, dass das System bei Verwendung von Bildern mit realen, unübersichtlichen Verkehrssituationen an seine Grenzen stößt. Um das Programm so zu erweitern, dass die Komplexität des realen Straßenverkehrs beherrscht werden kann, wären wahrscheinlich Jahre an weiterem Entwicklungsaufwand nötig.

Zum Schluss kann festgehalten werden, dass während des Projektes das, in der Vorlesung vermittelte Wissen, praktisch angewandt und besser verstanden wurde. Letztlich sind wir sehr froh, dass wir ein solch industrienahes und ergebnisreiches Projektthema bearbeiten durften. Wir sehen das Bilddatenverarbeitungslabor als vollen Erfolg und als wichtigen Beitrag zu unserer persönlichen und fachlichen Entwicklung.

**Literaturverzeichnis**

|  |  |
| --- | --- |
| [1] | Klaus D. Tönnies, Grundlagen der Bildverarbeitung, München: Pearson Verlag, 2005. |
| [2] | K. Peschke, H. Burkhardt und R. Klaus, „Bildvorverarbeitung,“ [Online]. Available: https://lmb.informatik.uni-freiburg.de/lectures/praktika\_brox/bvpraktikum/BVAnl\_bildvor.pdf. [Zugriff am 28. Dezember 2017]. |
| [3] | OpenCV, „OpenCV Dokumentation - Thresholding,“ [Online]. Available: https://docs.opencv.org/3.3.1/d7/d4d/tutorial\_py\_thresholding.html. [Zugriff am 28. Dezember. 2017]. |
| [4] | Prof. Dr. Dieter Maier, Vorlesungsskript Bilddatenverarbeitung, Hochschule Heilbronn, 2017. |
| [5] | OpenCV, „OpenCV-Dokumentation - HSV-Farbraum,“ [Online]. Available: https://docs.opencv.org/3.1.0/df/d9d/tutorial\_py\_colorspaces.html. [Zugriff am 28. Dezember. 2017]. |
| [6] | OpenCV, „OpenCV-Dokumentation - Hierarchie,“ [Online]. Available: https://docs.opencv.org/trunk/d9/d8b/tutorial\_py\_contours\_hierarchy.html. [Zugriff am 28. Dezember 2017]. |
| [7] | Typografie.info, „Normschrift auf Verkehrsschildern,“ [Online]. Available: http://www.typografie.info/3/Schriften/fonts.html/din-1451-r214/. [Zugriff am 2017 Dezember 29.]. |

**Selbständigkeitserklärung**

Wir versichern, dass wir die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt haben. Wir haben uns anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

|  |  |
| --- | --- |
| Gustav Willig |  |
|  |  |
| Daniel Husfeldt |  |