Abschlussbericht

Team SIFT

Vorlesungsbegleitendes Projekt im Verlauf der Veranstaltung Grundlagen der Computer Vision

Teilnehmer

Tim Jagla

Patrick Nierath

Betreuer

Sebastian Schäfer

SS2012

I. Einleitung

Im Zuge der Veranstaltung *Grundlagen der Computer Vision* im Sommersemester 2012 gab es ein übungsbegleitendes Projekt, bei dem eine Software geschrieben werden sollte, welche eine festgelegte Menge von Verkehrsschildern in zur Verfügung gestellten Bilddaten erkennt [3].
Die Vorgehensweise und der genutzte Algorithmus sind hierbei frei wählbar und eigenständig umzusetzen gewesen. Die einzige Limitierung waren die zu evaluierenden Testdaten, welche gestellt wurden, um eine Vergleichsmöglichkeit zwischen den verschiedenen Gruppen zu gewährleisten und die Qualität des Ergebnisses bewerten zu können.
Unser Ziel war eine möglichst erfolgreiche Erkennung mit einem Verfahren zu gewährleisten, welches noch nicht in verschiedenen Veranstaltungen behandelt worden und damit eine neue Erfahrung ist.
Die Wahl fiel auf SIFT. Als einer der frühen merkmalsbeschreibenden Algorithmen bietet Dieser Vorteile bezüglich der mathematischen Nachvollziehbarkeit und Präzision mit dem Nachteil der Laufzeitkomplexität.
Zur Umsetzung unseres Programmes haben wir C++ sowie folgende Software und Bibliotheken benutzt:

- Microsoft Visual Studio 2010 (Ultimate/Professional)
- GanttProject (Projektplanung)
- SVN
- OpenCV
- Ursprünglich Qt und QML

II. Verfahren

Der Grund, warum wir uns für SIFT entschieden haben, war zum einen das Interesse sich mit etwas Neuem uns bis dahin Unbekanntem zu beschäftigen und zum anderen die Vorteile des Verfahrens gegenüber diversen in der Vorlesung vorgestellten Ansätzen, Templates in Bildern zu erkennen. SIFT, ausformuliert **S**cale **I**nvariant **F**eature **T**ransform, wird per Definition als invariant gegenüber den großen Problemfaktoren beim Template-Matching beschrieben:

- Rotation
- Skalierung
- Beleuchtungssituation
- Betrachtungsperspektive

Nach anfänglichen Schwierigkeiten den Algorithmus selbst zu implementieren haben wir uns aus Zeitgründen für die vorhandenen Klassen- und Datenstrukturen in der OpenCV Bibliothek entschieden.
Der Algorithmus wurde von David G. Lowe entwickelt und im Jahre 1999 veröffentlicht. Um die Komplexität besser zu erfassen, wird das Verfahren oft in mehrere Teile untergliedert.

Der erste Schritt im Algorithmus ist das Erstellen sogenannter *Scale Spaces*. Hierfür wird das Ausgangsbild in ein Grauwertbild umgewandelt und für bessere Ergebnisse um den Faktor zwei hochskaliert.
Dieses Bild wird einem Gauß-Filter unterzogen und progressiv weichgezeichnet. Ist eine sogenannte *Oktave* fertiggestellt, wird im nächsten Schritt die Größe des Ausgangsbildes halbiert um dieses darauf erneut zu *blurren*. Illustriert ist dieser Teil des Verfahrens in Abb. 1.

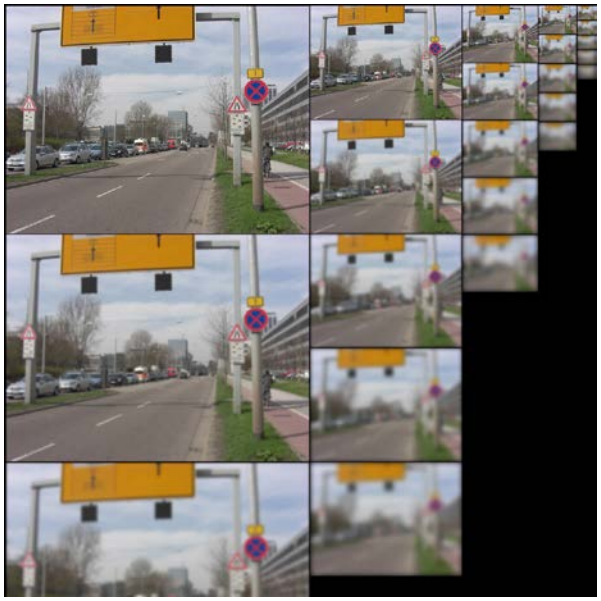

Abbildung 1 *Die verschiedenen Oktaven mit ihren jeweils weichgezeichneten Ebenen, aufgezeigt anhand eines Bildes aus den gegebenen Testdaten.*

Im nächsten Schritt würde man die Bilder eigentlich einem *Laplace-of-Gaussian (LoG)* Filter unterziehen, um Ecken und Kanten in dem Bild zu lokalisieren. Da dieser Prozess allerdings relativ rechenaufwändig ist, wird bei SIFT ein einfaches Subtraktionsverfahren verwendet. Der dabei entstehende Fehler ist vernachlässigbar.
Von der eben erstellten erweiterten Gauß-Pyramide werden paarweise Differenzbilder gebildet, *Difference-of-Gaussian (DoG)* Bilder genannt. Diese *DoG* Bilder ähneln durch *LoG*

entstandenen und dienen als Grundlage für die skalierungsinvariante Merkmalserkennung des Algorithmus. In Abbildung 2 wird die Vorgehensweise deutlicher.

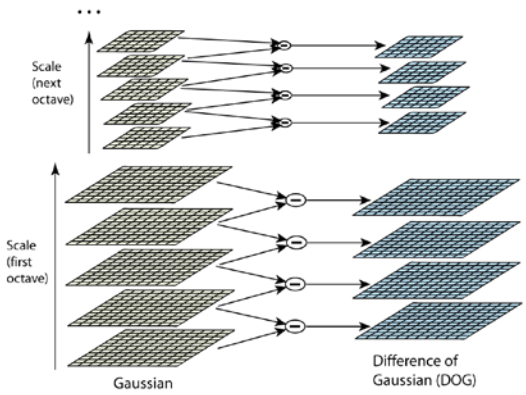

Abbildung 2 *Oktaven und ihre jeweiligen DoG Bilder, erstellt anhand von Subtraktion der weichgezeichneten Bilder.* [1]

Der dritte Schritt umfasst nun die Suche nach den Minima und Maxima unseres Bildes anhand der *DoG* Bilder.
Um einen *Keypoint* als einen solchen zu verifizieren werden zum einen dessen benachbarte Pixel betrachtet sowie die jeweiligen Nachbarn in der höheren und der tieferen Ebene in der *DoG* Pyramide. Folglich in Bildern gleicher Auflösung. Abbildung 3 veranschaulicht dieses Konzept.

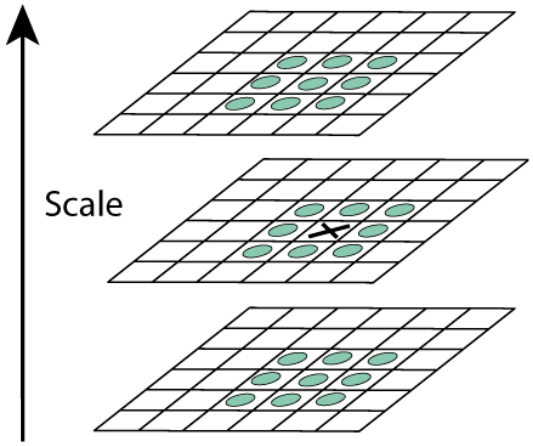

Abbildung 3 *Es handelt sich um drei DoG Bilder ein und derselben Oktave, also gleicher Auflösung. Für den mit einem Kreuz markierten Pixel wird durch einen einfachen Vergleich mit dessen 26 Nachbarn entschieden, ob es sich um einen Extremwert handelt.* [1]

Auf diese Art und Weise erhalten wir eine idealerweise große Menge *Keypoints*. Nicht alle davon sind jedoch relevant für den Algorithmus:
Viele der Punkte liegen auf Kanten, an Ecken oder aber in flachen, kontrastarmen Regionen. Jene Punkte werden, um die Präzision des Algorithmus zu erhöhen, herausgefiltert.

Die Pipeline bis zu diesem Punkt wird in Abbildung 4 noch einmal zusammengefasst.

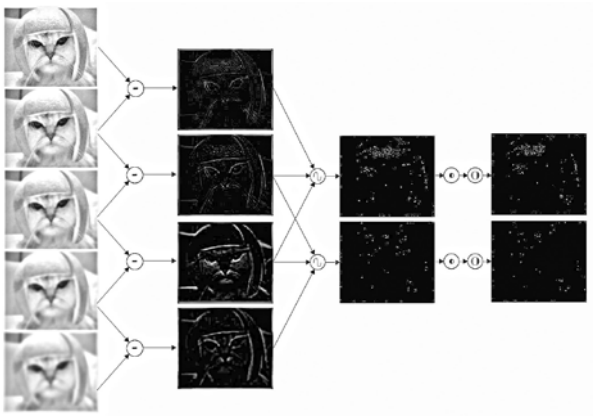

Abbildung 4 *(von links) Eine Oktave von Bilddaten, die jeweiligen DoG Bilder, daraus generierte Extremwertbilder sowie die gefilterten Extremwertbilder.* [4]

Um nun auch noch die gewünschte Invarianz gegenüber Rotation abzusichern, bestimmen wir für jeden *Keypoint* ein sogenanntes Orientierungs-Histogramm aus dem sich ein normalisierter und damit vergleichbarer *Descriptor* berechnen lässt. In Abbildung 5 wird das Konzept grundlegend erläutert.

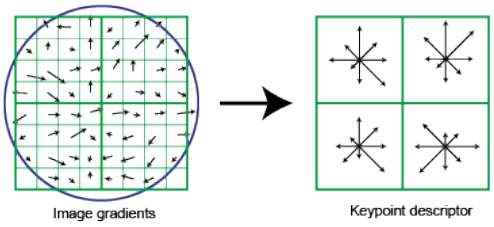

Abbildung 5 *(links) Für jeden Keypoint werden aufgeteilt in vier Sektionen die Größe und Orientierung umliegender Gradienten bestimmt. Die Ergebnisse werden nach Distanz zum Keypoint über ein Gauß-Fenster gewichtet. Die Relevanz nimmt nach außen hin ab, symbolisiert durch den blauen Kreis. (rechts) Aus den Ergebnissen werden Orientierungs-Histogramme generiert, welche eine Art Zusammenfassung der Ergebnisse aus der Abbildung links darstellen. Die Länge eines jeden Vektors korrespondiert zu der Summe der Größen der Gradienten in die jeweilige Richtung.* [1]

Dieser Fingerprint eines jeden *Keypoints* kann nun über verschiedene Ähnlichkeitsmaße mit anderen *Keypoints* verglichen und auf

Übereinstimmungen geprüft werden. Dieser Prozess wird *Matching* genannt.

III. Matching

Das Matching ist in unserer Implementation verhältnismäßig einfach ausgefallen und dadurch Hauptursache für die mittelmäßige Spezifität der Endergebnisse.
Um die gesuchten Verkehrszeichen als solche zu klassifizieren trainieren wir einen *Flann-Based-Matcher* auf die sechs gesuchten Schilder. Matches aus einem *Brute-Force-Matcher* haben sich als für unsere Anforderungen zu unpräzise herausgestellt.
Für jeden Deskriptor x im zu evaluierendem Bild werden die nächsten n = 2 *Nearest-Neighbor* Deskriptoren in der Datenbank des *Flann-Based-Matcher* gesucht. Dabei entsteht eine Liste von sortiert nach Relevanz, also Qualität der Übereinstimmung. Das Evaluationskriterium für den Fund eines Verkehrsschildes ist bei uns:

```
//retrieving the two nearest neighbor
//descriptors for the current descriptor x
actDescrPair = AllNearestNeighborPairs[x]

//evaluate whether the first entry of the
//current descriptor pair is a good one
if ( actDescrPair[0].distance <
     0.6 * actDescrPair[1].distance )
{
        //add the good matching descriptor to
        //the list of good matching
        //descriptors called “GoodMatches”
        GoodMatches.add(actDescrPair[0]);
}
```

Nun kann die Liste *GoodMatches* zum Beispiel insofern ausgewertet werden, dass die Anzahl der auf ein bestimmtes Verkehrszeichen zutreffenden Deskriptoren gezählt und mit einem Schwellenwert belegt werden. Ab einer gewissen Anzahl *GoodMatches* mit einem bestimmten Schild, eventuell eingegrenzt auf eine Region oder Distanz zueinander, wurde das Verkehrszeichen also zuverlässig gefunden. Der am besten beschreibende Deskriptor wird als korrekte Position des Fundes angenommen.

IV. Evaluation

Für die Evaluierung der Testdaten haben wir unsere Ergebnisse in eine Textdatei geschrieben. Diese hat in etwa das Format von der Textdatei für die Übersicht der Testdaten, sodass wir auf einen Blick genau sagen konnten, welche Schilder wir im jeweiligen Testbild gefunden haben, und sie somit schnell und einfach manuell auswerten konnten.

Die folgende Tabelle gibt einen Überblick über unsere Ergebnisse.

	Schilder	KP	FN	S (%)	# FP / DS
1	Stopp, 30, 120	1	0	100	2
2	Stopp	0	2	0	1
3	Gefahr	1	1	50	0
4	60, 120	1	1	50	1
5	30	1	0	100	0
6	Gefahr, Vorfahrt	1	1	50	1
7		0	0	100	0
8	Gefahr	1	1	50	0
9	Stopp	1	0	100	0
10		0	2	0	0
11	30	1	1	50	0
12	Gefahr, Vorfahrt	1	2	33,3	1
13	120	0	0	100	1
14		0	3	0	0
15	Stopp	0	2	0	1
16	Stopp, 120	1	0	100	1
17	Stopp, 30, 60, 120, G	1	0	100	4
18	120	0	1	0	1
19	Stopp	1	0	100	0
20	120	0	1	0	1
21	30	0	1	0	1
22	Stopp, Vorfahrt, Gefa	1	0	100	2
23	120	1	1	50	0
24	Gefahr	0	0	100	1
25	Stopp, 60	1	1	50	1
26	Stopp, 120	1	0	100	1
27		0	2	0	0
28		0	1	0	0
29	60	1	0	100	0
30		0	2	0	0
Ø		0,57	0,87	52,8	0,7

Tabelle 1 *Legende:*
K = Korrekt ↔ F = Falsch
P = Positiver Fund ↔ N = Negativer Fund
S = Sensitivität
DS = Datensatz

Die erste Spalte von Tabelle 1 enthält lediglich die Nummer des entsprechenden Bildes aus den vorgegebenen Evaluationsdaten. In der zweiten Spalte haben wir zusammengefasst, welche Schilder im jeweiligen Bild von unserem Algorithmus gefunden wurden. Die anderen Spalten stellen dann die Werte gemäß der

Beschreibung in Tabellenkopf und Legende dar.

Aus den Ergebnissen wird ersichtlich, dass wir im Durchschnitt eine Sensitivität von 52,77% haben, also immerhin bei mehr als der Hälfte der Testfälle die Schilder korrekt erkennen.
Fehlerkennungen treten durchschnittlich 0,7-mal pro Bild auf.

Generell wird in Tabelle 1 deutlich, dass die Resultate schwanken. Die Ergebnisse reichen von „sehr gut“ bis „ungenügend“.
Besonders erstaunlich und gut gelungen ist die Auswertung des folgenden Bildes:

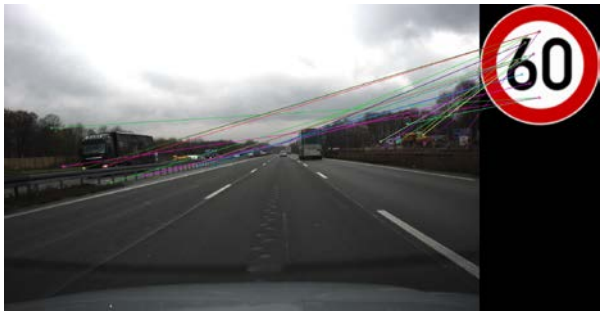

Abbildung 5 *Die zwischen dem Schild und der Szene gefundenen Matches. Viele davon sind ausgesprochen minderwertig und prädestiniert dazu, herausgefiltert zu werden.*

Abbildung 6 *Matches nach unserem in Abschnitt „III. Matching“ beschriebenem Filterungsverfahren. Ausschließlich der Match mit dem rein optisch nur schwer erkennbaren Verkehrszeichen übersteht die Filterung.*

Nun stellt sich die Frage, warum der Algorithmus trotz seiner vielen Vorzüge nicht immer funktioniert hat? Das hat mehrere Gründe:
Ein Aspekt ist, dass wir gewisse Feinheiten nicht abfangen bzw. im Rahmen dieses Projektes nicht weiter bearbeiten konnten. Da unser Algorithmus auf Feature-Vektoren beruht, gibt es gerade bei den Geschwindigkeitsbegrenzungsschildern ein paar Probleme.
So werden markante Stellen der Zahl null sehr gut gefunden, allerdings kommt diese in jeder Geschwindigkeitsbegrenzung vor und erschwert die Klassifizierung der Schilder. Deswegen erkennen wir zum Beispiel im folgenden Bild fälschlicherweise je ein 30, 60 und ein 120 km/h Höchstgeschwindigkeit Schild.

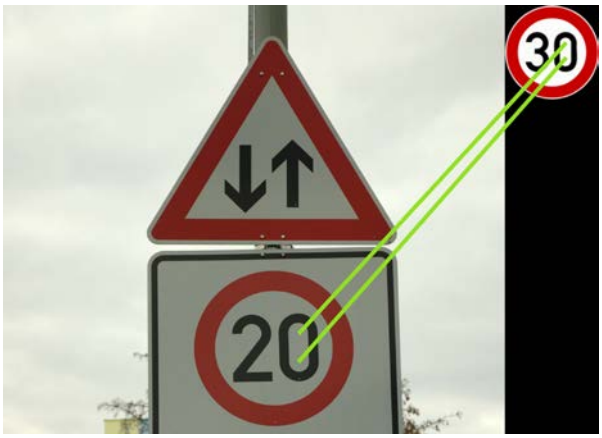

Abbildung 7 *Ein für unser Matching hochproblematisches Bild.*
Durch die null werden Übereinstimmungen mit allen Geschwindigkeitsbegrenzung Schildern gefunden und darüber hinaus noch aus unbekannten Gründen ein Stopp Schild.

Ein weiteres Problem ist unser eigentlicher Vorteil der Invarianz gegenüber Rotationen. Dadurch ergeben sich viele Verwechslungen und falsch positive Matches mit Schildern, die die Grundform des Gefahrenzeichens haben. Die nächste Abbildung gibt Aufschluss über die Problematik:

Abbildung 8 *Die Rotationsinvarianz ist Ursache für die sichtbare Problematik. Weil der untere Teil des Trainingsdatensatzes mit der Seite des Gefahrenzeichens in der Szene als übereinstimmend gewertet wird, können sich in anderen Situationen schwere Verwechslungen zwischen Gefahr und Vorfahrt Beachten Schildern ergeben.*

Man muss SIFT als Basisalgorithmus zugestehen, dass es für diese Art der Problemstellung noch erheblich mehr Potential besitzt, als das, was wir im Rahmen der Veranstaltung umsetzen konnten.

Große Schwierigkeiten entstanden zum Beispiel bereits bei der Suche der *Keypoints*. Weil der Algorithmus in der Szene oft nur ein bis zwei

Extrempunkte auf einem Verkehrszeichen findet sind es auf den Trainingsdaten selbst erheblich mehr und detailliertere, wie in Abbildung 5 in Ansätzen zu erkennen.
Auf dieser Grundlage einen verlässlichen Match zwischen den Schildern zu finden hat sich als extrem kompliziert herausgestellt. Außerdem gibt es Situationen in denen Verkehrszeichen auf dem Bild keinerlei *Keypoints* zugeordnet bekommt. In diesem Fall können die betroffenen Schilder unmöglich klassifiziert werden, wie in Abbildung 9 zu sehen.

Abbildung 9 *In dieser Szene bekommt nur das linke 120 km/h Schild einen einzelnen Keypoint zugeordnet. Matches zwischen dem Trainingsdatensatz und der Szene sind an dieser Stelle unwahrscheinlich. Das rechte Schild hat keinen einzigen Keypoint und ist damit im weiteren Verlauf unmöglich zu erkennen.*

Das Verändern der von David G. Lowe empfohlenen Standardparametern des SIFT Algorithmus hat aus unserer Sicht keine Verbesserung, geschweige denn signifikante Veränderungen gebracht.
Positiv anerkennen muss man also, dass der Algorithmus stabile und reproduzierbare Ergebnisse liefert, selbst unter Parameterveränderungen.
Als erschwerender Faktor stellte sich allerdings heraus, dass der *Matcher* unabhängig von den SIFT Parametern ist und deswegen ebenfalls optimiert werden kann. Hier sind die Veränderungen signifikanter, es ist jedoch schwer präzisere Übereinstimmungen ohne längere Erfahrung mit der Thematik hervorzurufen. Die Standardparameter haben sich als Garant für die verlässlichsten *Matches* herausgestellt.

Die Performanz des Verfahrens ist überproportional abhängig von der Größe der Bilddaten und der Menge daraus extrahierten *Keypoints*. Unsere Auswertung der 30 Datensätze hat auf einem 2,1 GHz Dual-Core Prozessor etwa drei Minuten gedauert. Der eigene Code-Anteil ist allerdings fernab von jeglicher Optimierung.

V. Schluss

Abschließend können wir sagen, dass der SIFT-Algorithmus durchaus geeignet ist um Verkehrszeichen auf Bilddaten zu erkennen.
Aufgrund der Komplexität des Verfahrens ist es aber zum einen nicht einfach mit ihm umzugehen und zum anderen erfordert es gerade im Bereich des *Matchings* eine bessere Implementation um die aufgezeigten Sonderfälle zu behandeln.
Beim *Matching* herrscht allgemein das größte Verbesserungspotenzial. Um die Spezifität des Verfahrens zu erhöhen existieren verschiedene herausragende Implementationen mit *Machine-Learning* Algorithmen. [2]

Um die Quantität und Qualität von *Keypoints* und *Descriptors* zu erhöhen muss mit einem zusätzlichen *Upscaling* der Evaluations- und Trainingsdaten experimentiert werden. [5]

Eine größere Trainingsdatenbank wäre nützlich um Verkehrszeichen etwas verlässlicher auf den Evaluationsdatensätzen zu erkennen.

Erweitert man das bisherige Verfahren um einen *Bag-of-Words* Klassifikator (im Zusammenhang mit SIFT also *Bag-of-Visual-Words*) ließen sich die Verkehrszeichen eventuell eindeutiger klassifizieren. [6]
Die ohnehin bereits hohe Laufzeitkomplexität würde allerdings erheblich ansteigen.

VI. Quellenangaben

[1] D. G.. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, 2004
[2] P. Sermanet, Y. LeCun, *Traffic Sign Recognition with Multi-Scale Convolutional Networks*, 2011
[3] K. Tönnies, *Grundlagen der Computer Vision*, Lecture, [Letzter Zugriff: 4.7.2012], website: http://wwwisg.cs.uni-magdeburg.de/bv/index.php?article_id=23&clang=0
[4] U. Sinha, *SIFT: Scale Invariant Feature Transform*, [Letzter Zugriff: 6.7.2012], website: http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/
[5] D. G.. Lowe, *Object Recognition from Local Scale-Invariant Features*, 1999
[6] Roy S., [Letzter Zugriff: 29.6.2012], website: http://www.morethantechnical.com/2011/08/25/a-simple-object-classifier-with-bag-of-words-using-opencv-2-3-w-code/