Analyse von Straßenbestandsobjekten aus Laserpunktwolken durch Mustererkennung/ Objekterkennung einschließlich der Georeferenzierung

Fachveröffentlichung der
Bundesanstalt für Straßenwesen

bast

Analyse von Straßenbestandsobjekten aus Laserpunktwolken durch Mustererkennung/ Objekterkennung einschließlich der Georeferenzierung

Projektnummer
FE 02.0378/2014

Andreas Großmann
Steffen Scheller

LEHMANN + PARTNER GmbH
Erfurt

Alexander Reiterer
Dominik Störk

Fraunhofer-Institut für Physikalische Messtechnik IPM
Freiburg im Breisgau

Fachbetreuung
Gerd Kellermann

Herausgeber
Bundesanstalt für Straßenwesen
Brüderstraße 53, 51427 Bergisch Gladbach

Juni 2023

Kurzfassung

Als Grundlage des Forschungsprojektes FE 02.0378/2014/CGB „Analyse von Straßenbestandsobjekten aus Laserpunktwolken durch Mustererkennung/Objekterkennung einschließlich der Georeferenzierung“ wurden Messdaten der Mobile Mapping Fahrzeuge IRIS5 und IRIS12 der Firma LEHMANN+PARTNER GmbH (LP) eingesetzt. Die georeferenzierten Punktwolken entstanden mit Hilfe eines an den Fahrzeugen montierten CPS Laserscanners vom Projektpartner, dem Fraunhofer Institut für Physikalische Messtechnik, IPM, Freiburg.

Die Aufgabe des Forschungsprojektes bestand in der Erforschung von Methoden und Algorithmen zur automatisierten Mustererkennung von speziellen Objekten in dreidimensionalen Punktwolken. Die ausschließliche Analyse bzw. Extraktion von Objekten auf Grundlage der reinen Punktwolke erwies sich am Anfang des Projektes als nicht zielführend. Größter limitierender Faktor war die mit der Entfernung zunehmende Dichteverringerung in den Daten der Punktwolken. Dies führte zum Teil zu unterrepräsentativen Abbildungen von kleinen oder schmalen Objekten. Um dieses Defizit auszugleichen, wurden die georeferenzierten Bilddaten der Mobile Mapping Fahrzeuge als Analysehilfe verwendet. Die Bilddaten weisen ein vielfach höheres Auslösungsvermögen im Vergleich zur Punkwolke auf. Zur Analyse der Daten kamen verschieden Neuronale Netze zum Einsatz, die zunächst die Bildinformationen analysierten. Nach der Trainingsphase des Neuronalen Netzes wurden Objekte einer messtechnischen Aufnahme (Scene) detektiert. Durch die georeferenzierten Bilddaten konnten alle automatisch gefundenen Objektinformationen in die Punktwolke übertragen werden. Hierbei wurde eine weitaus größere Diversität der Extraktionsergebnisse als mit der Analyse der reinen Punktwolke erzielt. Die finale Lösung der automatischen Extraktion bestand in der Projektion der einzelnen Objekte vom Neuronalen Netz in die Punktwolke. Dadurch, dass jedes Objekt mehrfach in jeder Bildszene erfasst wurde, besitzt jeder Laserscanner Punkt mehrere automatisch generierte Objektlable. Mit Hilfe von Clusteranalysen und Mehrheitsentscheidungen konnte die Ausgangspunktwolke in einzelne Objekte vollautomatisch zerlegt werden.

Für die Verwendung in einem GIS oder für die OKSTRA konforme Speicherung mussten die Daten weiter aufbereitet werden. Hierzu wurden die einzelnen Objektklassen einer Repräsentationsklasse zugeordnet, sodass eine eindeutige Darstellung in einem GIS erfolgen konnte.

Zur Kontrolle und Validierung der Extraktionsergebnisse wurde eine unbekannte Teststecke (Rundkurs bei Köln-Rösrath) aufgenommen. Die Digitalisierung jedes Referenzobjektes erfolgte manuell. Der Vergleich der Objekte erfolgte aufgrund Lage, Ausprägung und Objekttyp. Hierbei stellte es sich heraus, dass insgesamt 66% der Objekte komplett oder teilweise extrahiert werden konnten. Nur 4% der automatisch detektierten Flächenobjekte konnte im Referenzdatensatz nicht gefunden werden. Bestes Extraktionsergebnis lieferte der Objekttyp „Markierungslinien“ mit 90% Übereinstimmung mit dem Referenzdatensatz auf dem Testabschnitt der BAB. Zuordnungsdefizite kamen hauptsächlich aus den Label-Zuordnungsfehlern, die sich zum Teil aus der Abbildungsgeometrie schmaler und kleiner Objekte in der Punktwolke ergaben. An dieser Stelle besteht noch erhebliches Potential zur Verbesserung der Extraktionsergebnisse, welche im Rahmen dieser Forschungsarbeit nicht weiter optimiert werden konnte.

Das Forschungsvorhaben zeigte, dass aus einer Punktwolke unter Zuhilfenahme von georeferenzierten Bildern der gleichen Szene vollautomatisch Objekte zu extrahieren sind. Im Ergebnis liegen georeferenzierte Objekte, die in einem Geoinformationssystem abgebildet werden können, vor.

Short version

As a basis of the research project FE 02.0378 / 2014 / CGB "Analysis of Road Stock Objects from Laser Point Clouds by Pattern Recognition / Object Recognition including Georeferencing" measurement data with the Mobile Mapping Vehicle IRIS5 and IRIS12 of the company LEHMANN + PARTNER GmbH (LP) were collected. The geo-referenced point clouds were created with a CPS laser scanner from the project partner, Fraunhofer Institute for Physical Measurement, IPM, Freiburg.

The task of the research project was the investigation of methods and algorithms for automated pattern recognition of special objects in three-dimensional point clouds. The initial analysis or extraction of objects based on the pure point cloud proved to be ineffective at the beginning of the project. The biggest limiting factor was the increasing density reduction with the distance in the data of the point cloud. This partly led to under representative images of small or narrow objects. To compensate for this deficit, the georeferenced image data of the mobile mapping vehicles were additional used to support the analysis. The image data have a much higher resolution compared to the point cloud. To analyze the data, various neural networks were used, which first analyzed the image information. After the training phase of the neural network objects of a picture-based recording (scene) were detected. Due to the georeferenced image data, all automatically founded object information could be transferred into the point cloud. Here, a much greater diversity of the extraction results than with the analysis of the pure point cloud could be achieved. The final solution of the automatic extraction was the projection of the individual objects from the neural network into the point cloud. Due to the fact that each object has been recorded several times in each image scene, each laser scanner point now has several automatically generated object labels. With the help of cluster analyzes and majority decisions, the starting point cloud could be decomposed into individual objects fully automatically.

For use in a GIS or for OKSTRA compliant storage, the data had to be further processed. For this purpose, the individual object classes were assigned to a representation class, so that a clear representation could take place in a GIS.

For the control and validation of the extraction results, data at an unknown test section were collected (test circuit close to Cologne-Rösrath). The digitization of each reference object was done manually. The objects were compared on the basis of location, characteristics and object type. It turned out that a total of 66% of the objects could be extracted completely or partially. Only 4% of the automatically detected area objects could not be found in the reference data set. The best extraction result was provided by the object type road marking with 90% agreement with the reference data set on the motorway.

The final mapping deficits mainly come from label mappings errors, from the measured scene into the point cloud. The reason for that, are the geometrical density deficit inside the point cloud, so that narrow and small objects get a position error from the projection. At this point, there is still considerable potential for improving the extraction results, which could not be further optimized in the context of this research work.

The research project showed that objects can be extracted fully automatically from a point cloud with the aid of georeferenced images of the same scene. As a result, georeferenced objects that can be mapped in a geoinformation system are available.

Summary report

1. Task

As a basis of the research project "Analysis of roadside objects from laser point clouds by pattern recognition / object recognition including georeferencing", measurement data of the mobile mapping vehicle IRIS5 and IRIS12 of the company LEHMANN + PARTNER GmbH (LP) were used. The geo-referenced point clouds were created with a CPS laser scanner from the project partner Fraunhofer Institute for Physical Measurement, IPM, Freiburg.

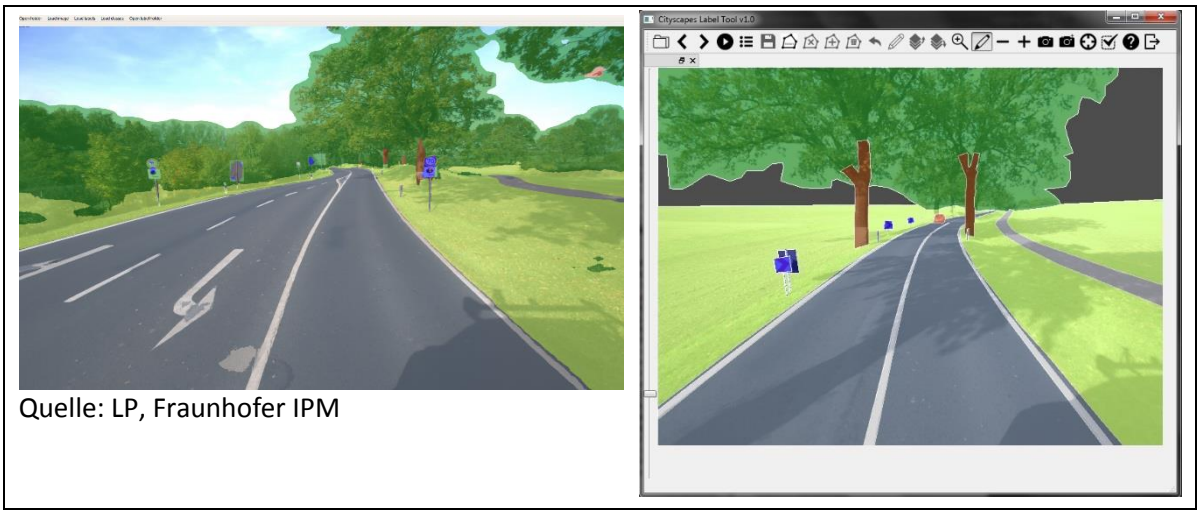
Figure 1 Picture left: Mobile Mapping Vehicle IRIS12 - Picture right: Clearance Profile Scanner CPS for recording the test data

2. Research methodology

The task of the research project was the investigation of methods and algorithms for automated pattern recognition of special objects in three-dimensional point clouds. The initial analysis or extraction of objects based on the pure point cloud proved to be ineffective at the beginning of the project. The biggest limiting factor was the increasing density reduction with the distance in the data of the point cloud. This partly led to under representative images of small or narrow objects. To compensate for this deficit, the georeferenced image data of the mobile mapping vehicles were used as an aid to analysis. The image data have in comparison to the point cloud a much higher resolution. To analyze the data, various neural networks were used, which first analyze the image information. So-called training images were labeled for the detection of the previously defined street objects. This means that each area in a picture-based recording (scene) have to be clearly assigned to an object. In retrospect, this annotation proved to be particularly difficult, as each operator generalized the different object classes differently. As a result, smaller objects, some in the background of a scene, were labeled differently. This influenced the neural network a lot. Extensive tests were carried out by the project partner IPM on which network architecture is best suited for object recognition in the road area. Numerous tests have been carried out in this area and solutions have been discarded during processing. After the training phase of the neural network objects of a scene could be detected. Due to the georeferenced image data, all automatically found object information could be transferred to the point cloud. Here, a much greater diversity of the extraction results could be reached than with the analysis of the pure point cloud.

3. Investigative Results

The final solution of the automatic extraction was the projection of the individual objects from the picture based neural network data into the point cloud. Each object has been captured multiple times in each image scene, so each laser scanner point now has several automatically generated object labels. With the help of cluster analyzes and majority decisions, the labeling of point cloud could be done, fully automatically.



Quelle: LP, Fraunhofer IPM

Figure 2: **Left picture: Segmentation of an image from the test dataset: delineator, road, pavement, marker, vegetation, treetop, and sky are reliably segmented (FCN-based network, 60,000 iterations).**
Right picture: manually labeled training image for the neural network

For use in a geographic information system (GIS) or for OKSTRA compliant storage, the data had to be further processed. For this purpose, the individual object classes were assigned to a representation class, so that a clear representation could take place in a GIS.

For the control and validation of the extraction results, which is unknown to the neuronal network process, was recorded a test track with the mobile mapping vehicle IRIS12. The digitization of each reference object was manual digitized inside a georeferenced database. The automatically georeferenced extraction results were also imported into the database. The objects were compared on the basis of location, characteristics and object type. It turned out that a total of 66% of the objects could be extracted completely or partially. Of these, 26% of the objects with the correct object type and correct position, 14% with the correct position but the wrong object type and 26% could be incompletely detected. 34% of the objects have not been found in the automatically detected dataset compared to the reference dataset. It turned out that only 4% of the reference objects were not found in the area objects. The correctly extracted area objects could be recorded with a coverage rate of 50%, in the sum over all test areas. The best extraction result was provided by the object type Marking Lines with 90% agreement (reference data set on the test section of the A3 and A4 freeways).

The point objects could be verified in total only in 47% of the cases with the reference data set. The allocation deficits mainly result from the label allocation errors, which result in part from the calibration deficits of the measurement sensors and partly from the imaging geometry of narrow and small objects in the point cloud. There is still a lot of potential here to improve the extraction results, which could not be further optimized within the scope of this research project.

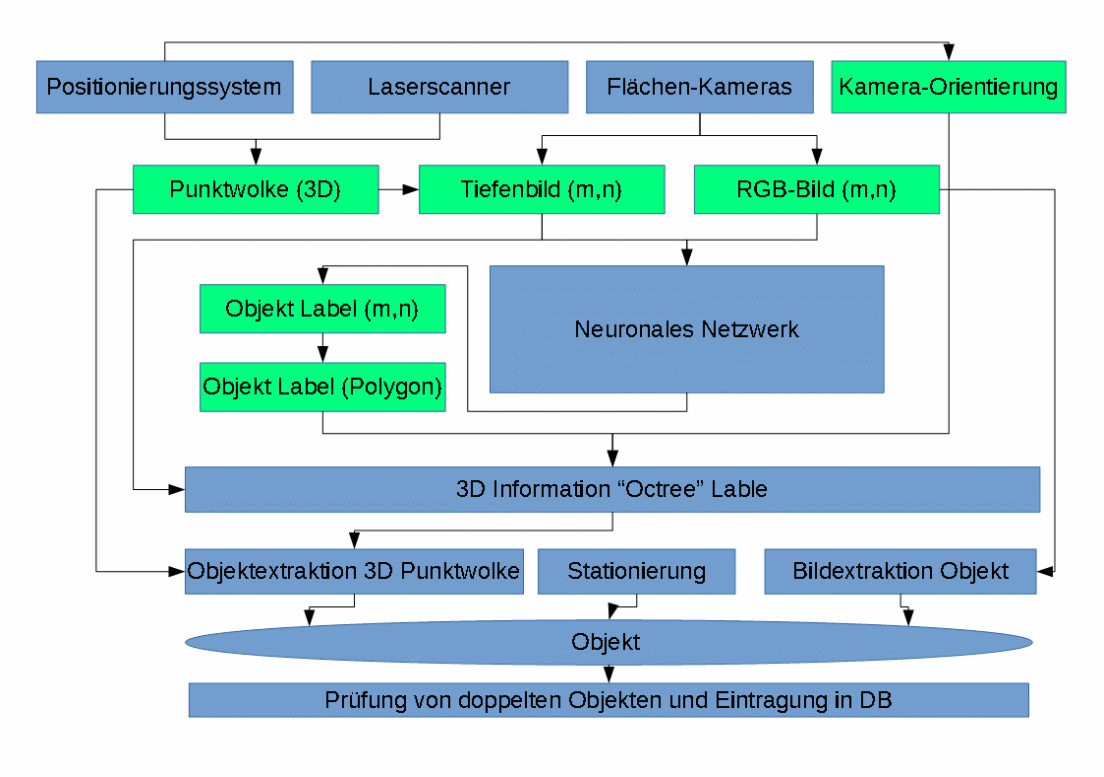


Figure 3: **Semantic flowchart of the evaluation procedure with intermediate steps and their relationships to each other.**

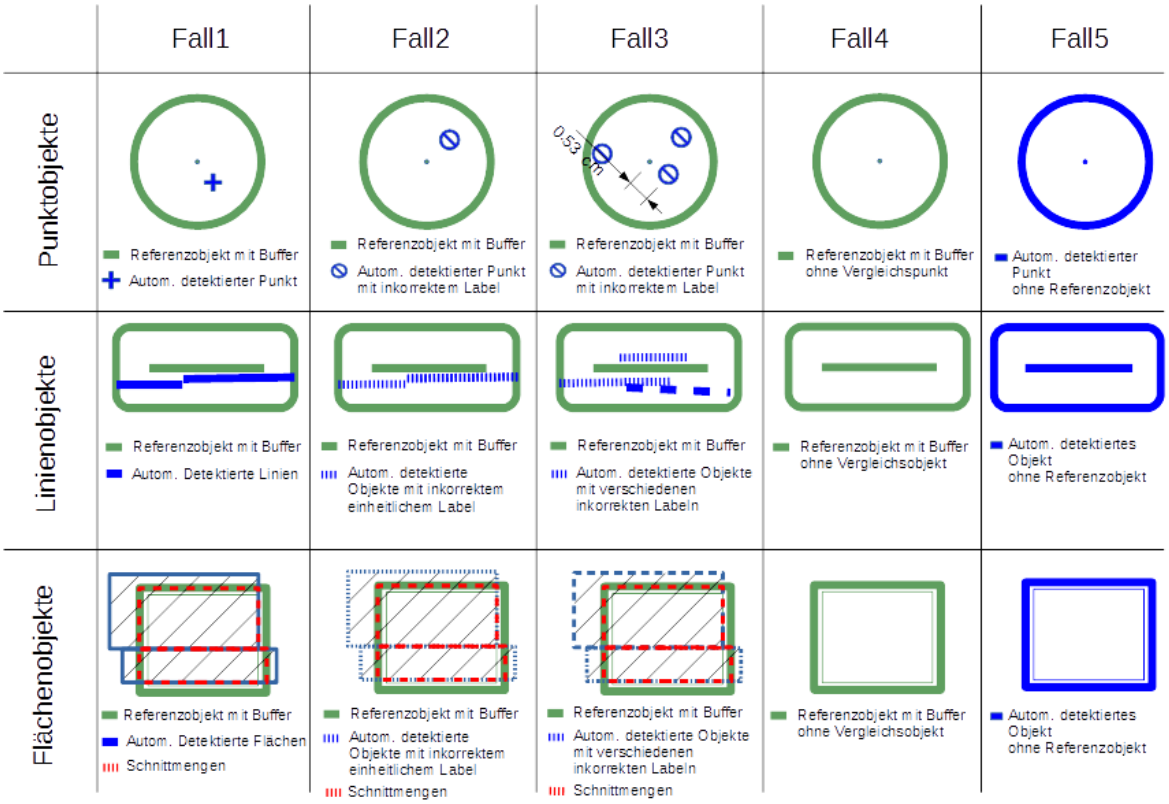


Figure 4: **Distinctions for the classification of the nominal-actual comparison between the reference data record and the automatically generated data record. Here, a distinction is made between point lines and area objects**

Table 1: **Comparison of the reference digitization with the prepared, automatically found objects of the neural network in the combined areas highway, country road and urban, the columns with the extraction results F1-F5 are described in Figure 4. The reference describes the underlying check quantity and the distance the mean geometric deviation of the individual objects**

ID	Objekttyp	Geometrie	F1 [%]	F2 [%]	F3 [%]	F4 [%]	Referenz	F5 [%]	Abstand [m]
1	Road sign	Point	27.23	5.6	25.95	41.22	393	62.62	0.78
2	kilometre-plate	Point	**80.00**	**0.00**	**20.00**	**0.00**	5	16.67	0.78
3	station-plate	Point	0.00	11.11	22.22	66.67	9	0.00	0.48
4	City plate	Point	0.00	20.00	40.00	40.00	5	0.00	0.39
5	Emergency phone	Point	0.00	0.00	0.00	100.00	4	100.00	-
6	Traffic light	Point	1.52	22.73	33.33	42.42	66	66.67	0.59
7	Street lamps	Point	0.00	24.14	39.66	36.21	116	28.57	1.76
8	pole	Point	19.17	11.66	30.83	38.34	386	65.13	0.77
9	reflector post	Point	14.83	4.10	5.05	76.03	317	44.00	1.00
10	crash barrier	Multi-Lin-estring	**56.12**	**2.04**	**38.78**	**3.06**	29260m	31.75	0.16
11	barrier	Multi-Lin-estring	1.69	15.25	42.37	40.68	340m	41.06	0.29
12	building	Polygon	**65.06**	**28.92**	**4.82**	**1.20**	6198 m^2	23.23	0.05
13	distributor	Point	0.00	26.47	29.41	44.12	34	100.00	1.01
14	kerb	Multi-Lin-estring	0.93	9.30	61.86	27.91	1152 m	55.96	0.23
15	manhole cover	Point	0.00	10.17	12.20	77.63	295	50.00	2.07
16	street marking	Multi-Lin-estring	**73.50**	**0.40**	**8.92**	**17.17**	39839 m	12.83	0.15
17	Tree trunk	Point	42.33	3.43	16.70	37.53	437	59.48	1.07
18	street	Polygon	**56.13**	**30.06**	**5.52**	**8.29**	222079 m^2	45.02	0.09
20	sidewalk	Polygon	**53.62**	**15.32**	**27.23**	**3.83**	29773 m^2	33.30	0.03
23	vegetation	Polygon	**55.22**	**29.22**	**10.25**	**5.31**	174322 m^2	42.60	0.05
24	ground	Polygon	48.57	0.00	50.00	1.43	9777 m^2	0.25	0.02
26	marking	Polygon	18.37	20.41	59.18	2.04	5004 m^2	24.82	0.02
28	duct	Point	2.70	6.76	9.46	81.08	148	0.00	1.61
30	Concrete bar-rier	Multi-Lin-estring	0.00	60.00	20.00	20.00	474m	74.57	0.14

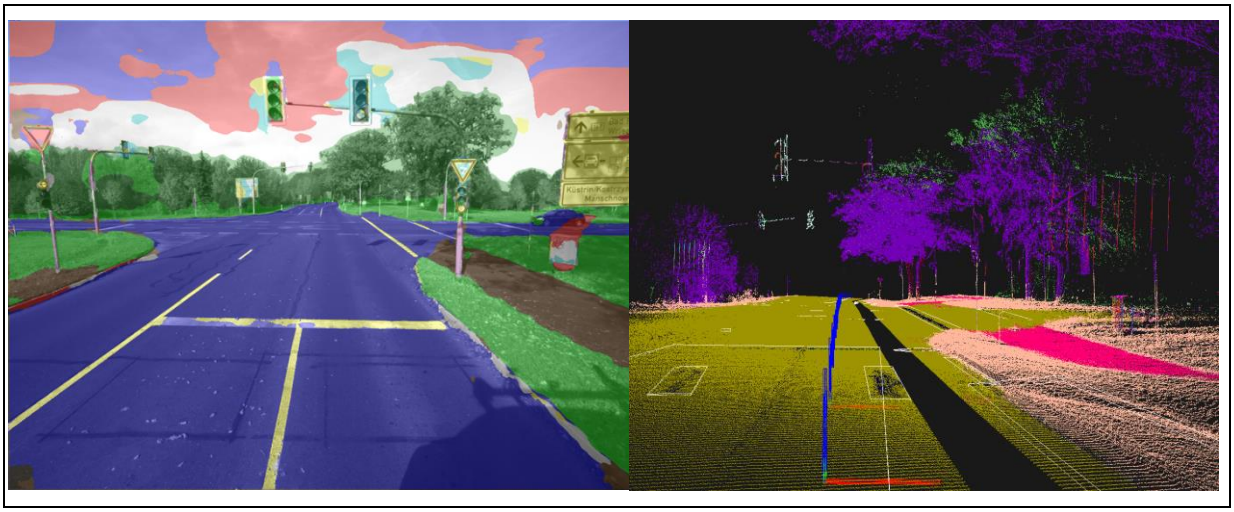


Figure 5: Image on the left: automatically detected object classes of a camera image Image on the right: transferred label attributes into the 3-dimensional point cloud

Figure 6: superimposed representation of reference digitization and automatically extracted point, line and area objects. All automatically found objects are labeled.

4. Conclusions

The research project showed that it is possible to fully automatically extract objects from a point cloud using georeferenced images of the same scene. The result is georeferenced objects that can be mapped in a geographic information system.

Inhalt

1 Problemstellung und Zielsetzung

1.1 Aufgabenstellung

Laserscanmessungen dienen heute der Erstellung von Grundplänen für den Um- und Ausbau von Straßen. Im Einsatz sind terrestrische Laserscanner (TLS), mobile Laserscanner (MLS) und luftgestützte Laserscanner (ALS). Der Vorteil dieser Messsysteme ist der geringe Einfluss auf den fließenden Verkehr. So lassen sich Staus einhergehend mit kostenintensiven Fahrbahn- bzw. Fahrstreifensperrungen reduzieren bzw. verhindern.

Bei den Laserscanmessungen entstehen sehr große Punktmengen (mehrere Milliarden Punkte je nach Messabschnitt) und damit sehr große Datenmengen. Diese Datenmengen erfordern neue Techniken bei der Auswertung.

Moderne Softwaresysteme sind in der Lage, diese Laserscandaten einzulesen und als Punktwolke am Bildschirm darzustellen. Daraus lassen sich Objekte, die vom Laserscanner während der Aufnahme abgetastet wurden, schemenhaft als Bild erkennen. Aus den ausgewählten angeklickten Punkten können die exakten Koordinaten entnommen werden. Darauf aufbauend können durch Bildschirmarbeit Objekte, wie z.B. Schilder, erfasst werden. Über den Fußpunkt lassen sich die Lagekoordinaten der Objekte ermitteln und zusammen mit visuellen Erkenntnissen (z.B. die Art des Schildes) diese Informationen über eine Eingabemaske in einer Straßendatenbank ablegen. Gleiches gilt für weitere Objekte, bspw. Durchfahrtshöhen von Bauwerken.

Während solche Punktwolken durch Messfahrzeuge mit einem Laserscanner schnell und einfach zu erstellen sind, ist die manuelle Auswertung heute noch sehr aufwändig, insbesondere wenn mit diesem Verfahren bspw. eine Bestandserfassung von größeren Straßennetzen erfolgen soll.

1.2 Zielsetzung

Mit dem Forschungsvorhaben sollen Methoden und Algorithmen entwickelt werden, die in der Punktwolke nach vorab definierten Mustern von Bestandsobjekten suchen und erkannte Objekte protokollieren und in einer Datei ggf. OKSTRA-konform abspeichern.

Eine Methode, die aus Punktwolken Bestandsobjekte analysieren kann, würde die notwendigen Vorarbeiten bei der Planung und beim Betrieb einer Straße gegenüber der bisherigen Arbeitsweise erheblich vereinfachen und beschleunigen. Mit einer Laserscanbefahrung mit anschließender Auswertung lassen sich Bestandsobjekte schnell erfassen und Änderungen im Bestand mit höherer Qualität nachweisen.

2 Literaturanalyse

2.1 Kinematische Messverfahren

Im letzten Jahrzehnt wurde ein Großteil der Laserscannerdaten im Straßenraum mit Mobile Mapping Systemen aufgenommen. Der große Vorteil dieser Systeme ist die Erfassung der Daten im fließenden Verkehr.

Das folgende Kapitel soll einen kurzen Überblick über die Entstehung und die verwendeten Komponenten in einem Mobile Mapping System geben. Hierbei wird auch auf die Sensoren eingegangen, da diese Grundvoraussetzung für die Datengrundlage und der darauf aufbauenden Mustererkennung sind. Die Qualität, Messdichte und die Kalibrierung haben direkten Einfluss auf die zu erwartenden Ergebnisse.

Tabelle 1 Frühe Mobile Mapping Systeme (nach Li et al., 1999).

MM System	Development Institution	Navigation Sensors	Mapping Sensors
GPSVan™	Center for Mapping, OSU	GPS/Gyro/Wheel Counter GPS/INS (second generation of the system)	2CCD, voice recorder
VISAT	University of Calgary	GPS/INS/ABS	8 b/w CCD and 1 color SVHS
GeoVan	Geospan Corp., USA	GPS/DR	10 VHS, voice recorder
GPS Vision	Lambda Tech. Inc., USA	GPS/INS	2 color CCD
KISS	Univ. of Bundeswher Munich and GeoDigital, Germany	GPS/IMU/Inclination Odometer/Barometer	1 SVHS, 2 b/w CCD, voice recorder
ON-SIGHT	Transmap Corp., USA	GPS/INS	4 color CCD
RGIAS	Rowe Surveying and Eng., Inc., USA	GPS	video, laser range finder
TruckMap	John E. Chance Inc., USA	GPS/Gyro/WADGPS	Laser range finder, 1 video camera
WUMMS	Wuhan University, China	GPS	3 video cameras, laser range finder
ROMDAS	Highway and Traffic Consultants Ltd., New Zealand	GPS	digital video camera
DDTI	Digital Data Technologies, Inc., USA	GPS	touch-screen, voice recorder
POS/LV™ 420	Applanix Corporation, Canada	GPS/INS/ distance measuring instrument (DMI)/ GPS Azimuth Measurement Subsystem (GAMS)	CCD, video

2.1.1 Mobile Mapping Systeme

Die ersten Mobile Mapping Systeme, die Straßendaten erfassen, wurden von Airborne/Luft-gestützten Systemen zur Kartografie abgeleitet. Erste wissenschaftliche Veröffentlichungen von Messfahrzeugen mit einem integrierten IMU/GPS-System wurden bei (Krakiwsky, 1991) und (Harris, C G & J M Pike 1988) beschrieben. Die ersten Messfahrzeuge besaßen ein einfaches GPS-

System, Videokameras und konnten Objekte entlang der Trajektorie mit einer Genauigkeit von 2-20 m lokalisieren.

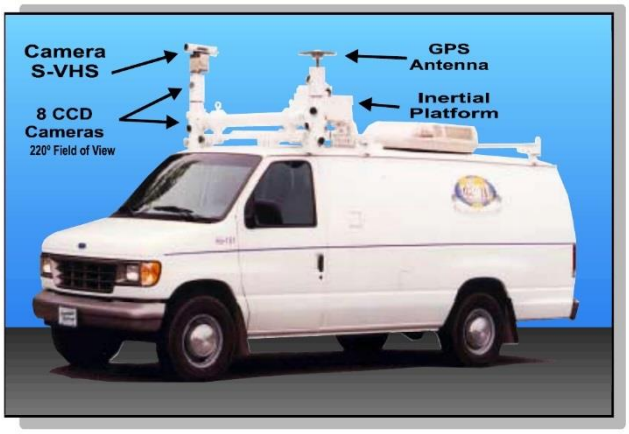


Bild 1: VISAT Van ca. 1995 (El-Sheimy et al. 1999).

Ein Mobile Mapping System besteht i.d.R. aus einem Positionierungssystem, welches das Primärsystem darstellt. Die Position wird durch die Kombination eines Kreiselsystems, einem GPS-System und einem Wegstreckensensor ermittelt. Dies geschieht mathematisch meist durch einen Kalman-Filter, der eine kontinuierliche Vorwärtslösung in Realtime berechnet. Alle weiteren Sensoren sind Sekundärsensoren, die entweder durch den Weg oder die Zeit mit dem Positionierungssystem synchronisiert sind. Diese Sensoren können Videokameras, CCD/CMOS Einzelbildkameras, Laserscanner, Laserdistanz-sensoren oder Georadarsensoren sein. Aktuelle Systeme werden im Kapitel 2.1.8 aufgezeigt.

2.1.2 Positionierungssystem

Ein hochgenaues Positionierungssystem besteht aus folgenden Komponenten:

- Inertial Measurement Unit (IMU)

Mit heutigem Stand der Technik besitzt es durchschnittlich eine Messfrequenz von 200 Hz und eine Drift von 1 Grad je Stunde. Die IMU ist neben dem GPS die Hauptkomponente des Systems. Eine beispielhafte Spezifikation ist der nachfolgenden Abbildung zu entnehmen.

- GPS-Receiver

Zur absoluten Positionierung werden ein oder zwei GPS-Receiver eingesetzt. Diese sollten sowohl Signale des GPS-Systems als auch von GLONAS verarbeiten. Dies garantiert eine höchstmögliche Empfangsrate. Es werden die Satelliten - Rohdaten mit einer Frequenz von 10 Hz aufgezeichnet (im RHINEX Format).

Tabelle 2: Typische Spezifikationen eines Kreiselsystems (Applanix 2015)

Performance – Accelerometer	
Bias Repeatability	300 µg to 3.0 milli-g, 1σ
Scale Factor Accuracy	300 to 5,000 ppm, 1σ
Performance – Gyro	
Bias Repeatability	1°/hr to 3°/hr, 1σ
Scale Factor Accuracy	100 to 500 ppm, 1σ
Random Walk	0.07° to 0.15°/√hr Power Spectral Density (PSD) level

- Wegstreckensensor

Zur Stützung der Orientierung in Längsrichtung und zur eindeutigen Erkennung von Stillstand (zero – velocity update – ZUPT) sollte ein Hodometer mit einer Auflösung von 4000 Pulsen je Meter (ca. 1 mm) eingesetzt werden.

Das Positionierungssystem ist die wesentliche Komponente des Fahrzeuges. Alle weiteren Sensoren werden mit diesem System synchronisiert. Das System sollte auf eine Aufnahmegeschwindigkeit von 120 km/h ausgelegt sein. Somit muss die Synchronisierungsgenauigkeit besser als 10^{-5} s betragen. Vor Einsatz bzw. vor Projektbeginn ist eine Kalibrierung der Hebelarme zu den einzelnen Komponenten erforderlich. Hierbei werden alle Rotationen und Translationen messtechnisch bestimmt. Die Parameter sollten im Nachgang über eine spezielle Software mit Hilfe einer Bündelblockausgleichung auf Signifikanz geprüft werden.

Ergebnis der gemeinsamen Lösung der einzelnen Sensoren ist eine dreidimensionale Trajektorie. Diese beschreibt die Fahrlinie des Fahrzeuges im übergeordneten Koordinatensystem über eine zeitliche Abfolge georeferenzierter Punkte mit einem begleitenden Dreibein.

Die Auswertung der Positionsdaten kann in drei verschiedene Klassen eingeteilt werden:

- Realtime Trajektorie
- RTK gestützte Trajektorie
- Postprozessierte Trajektorie

Die Realtime-Trajektorie wird i.d.R. online im Fahrzeug durch einen vorwärts gerichteten Kalman Filter berechnet. Alle Kalibrierungsfehler und GPS-Fehlereinflüsse und Driftverhalten der IMU durch GPS-Ausfall werden als Genauigkeits- oder

Positionsfehler der Trajektorie sichtbar. Diese Lösung hat die geringste Genauigkeit.

Mit einer zusätzlichen Stützung zur Messzeit durch ein RTK Signal (Realtime kinematic – Einspeisen von Korrekturdaten, wie SAPOS oder VRSNow) kann die GPS-Positionsgenauigkeit gesteigert werden.

Die beste Fehlerelimination erreicht man hingegen mit einer postprozessierten Trajektorie. Dazu werden alle Sensorrohdaten und mindestens drei Basisstationen genutzt (Berechnung als Netzausgleichung). In Abhängigkeit der GPS-Bedingungen können Genauigkeiten besser 20 cm erreicht werden.

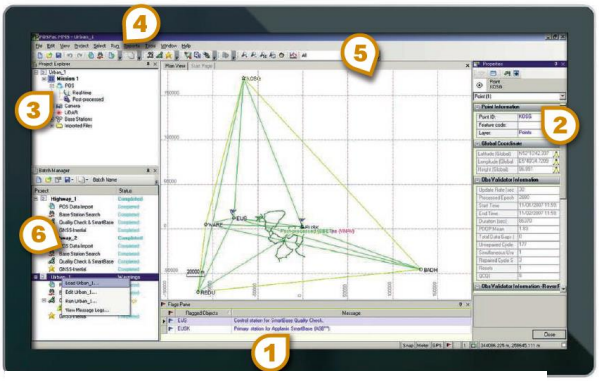


Bild 2: Beispiel einer Post-Prozessierungssoftware Applanix POS Pac.

2.1.3 Laserscanner

Unter Laserscanning wird ein punktuelles, zeitlich versetztes, flächenmäßiges Abtasten von Oberflächen mit einem Laserstrahl verstanden. Das Ergebnis sind ungeordnete dreidimensionale Punktkoordinaten. Diese werden durch polares Anhängen an die Standpunktkoordinaten, ähnlich wie bei einem Tachymeter, berechnet. Für jeden Punkt werden ein horizontaler und vertikaler Richtungswinkel und eine Strecke bestimmt. Die Streckenmessung erfolgt über eine reflektorlose Laufzeitmessung eines Laserstrahls. Zusätzlich zu den Koordinaten wird die Intensität des reflektierten Laserstrahls gespeichert. Diese ist abhängig von der Richtung des einfallenden Laserstrahls und vom Reflexionsgrad des Materials. Mit diesem Messverfahren ist es möglich, in wenigen Sekunden mehrere Millionen Messpunkte aufzunehmen. Der Nachteil ist, dass alle Punkte ungeordnet sind und auf einer zufälligen Abtastung des Objektes beruhen. Die Gesamtheit der Messergebnisse eines Laserscanners wird im Weiteren als Punktwolke bezeichnet.

2.1.4 Entfernungsmessung mittels Laserlicht

Die elektronische Streckenmessung beruht auf dem Prinzip, dass ein am Anfangspunkt der Strecke stehender Sender eine sich mit konstanter Geschwindigkeit ausbreitende Welle aussendet. Diese wird am Endpunkt der Strecke reflektiert und an der Sendeposition wieder empfangen. Dieser Trägerwelle wird ein eindeutiges Signal so aufmoduliert, dass eine Bestimmung der zurückgelegten Strecke möglich ist. Aus der Signallaufzeit t und der Gruppengeschwindigkeit der Trägerwelle c_{Gr} ergibt sich die Streckenlänge zwischen Anfangs- und Endpunkt s.

$$s' = \frac{c_{Gr} t}{2}$$

Für die Trägerwelle wird in der Regel eine Wellenlänge von 0,4 µm < λ < 1,3 mm verwendet.

Für die Signalmodulation kommen zwei Verfahren zur Anwendung, eine Phasen- und eine Impulsmodulation (Deumlich und Staiger, 2002). Bei Laserscannern mit hohen Abtast-geschwindigkeiten und großer Reichweite wird auch eine Impulsmodulation verwendet.

Das Verfahren sendet einen energiereichen Impuls aus. Dieser wird durch die Reflektion am Objekt an der Empfangseinheit wieder detektiert. Die Bestimmung der Laufzeit erfolgt durch die zeitliche Differenzbildung zwischen ausgesendetem und reflektiertem Impuls (Bild 2.1).

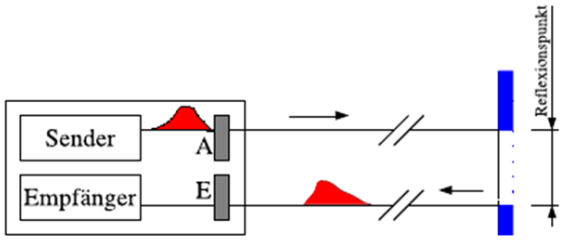


Bild 3: Prinzip des Impulsverfahrens, reflektorlos (in Anlehnung an [Deumlich und Staiger, 2002]).

Die genaue Detektion des zurückkommenden Signals kann auf verschiedene Weisen erfolgen, über einen Schwellwert, einen Nulldurchgang oder durch eine Rekonstruktion des empfangenen Impulses mittels einer Rasteraufteilung (multi pulse) [Ulrich u. a., 2005]. Es werden drei Arten von Reflexionsverhalten an einem Objekt unterschieden. Diese sind in der Abbildung 2.2 skizziert.

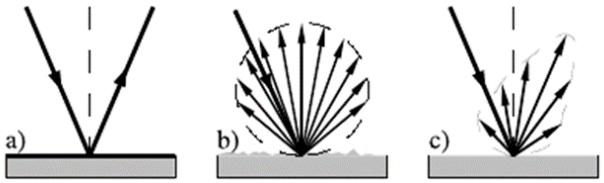


Bild 4: Arten von Reflexionen a) Spiegelung, b) diffuse Reflexion, c) diffuse Reflexion mit Spiegelung [Kern, 2002].

Die Laserscanner IMAGER 5003 der Zoller+Fröhlich GmbH, Fraunhofer IPM PPS/CPS und Laserscanner LS der Firma FARO verwenden das Phasenvergleichsverfahren, mit dem eine genauere Messung mit geringerer Reichweite möglich ist.

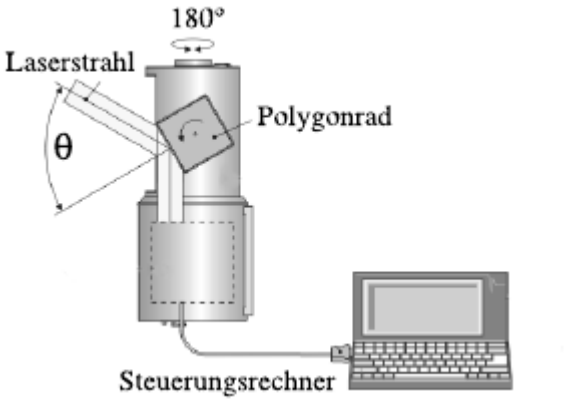


Bild 5: Prinzipskizze eines Laserscanners der Baureihe LMS der Firma Riegl [Riegl, 2005].

2.1.5 Terrestrische Laserscanner zur Erfassung von Objekten

Die Reflexionseigenschaften eines Objektes oder einer Fläche können sehr verschieden sein. Sie sind in erster Linie von der Materialbeschaffenheit abhängig, d.h., aus welchem Material besteht die Oberfläche, welche Rauigkeit besitzt sie und wie stark wird die Trägerwelle des Messsignals reflektiert oder absorbiert. Die Stärke des reflektierten Messsignals an der Empfangseinheit des Laserscanners wird als Intensitätswert eines Punktes gespeichert. Mit allen aufgenommenen Punkten eines Standpunktes ergibt sich ein Intensitätsbild.

2.1.6 Aufbau und Funktionsweise Laserscanner

Mit dem Ziel, ein regelmäßiges Raster auf einer Objektoberfläche zu erhalten, muss ein Laserstrahl entsprechend abgelenkt werden. In der vertikalen Ebene wird dies über einen beweglichen Spiegel oder ein rotierendes Polygonrad realisiert. Der horizontale Versatz ergibt sich durch eine horizontale Drehung des gesamten Spiegels oder des Polygonrades. In Bild 5 ist die Prinzipskizze der Laserscannerbaureihe LMS der Firma Riegl graphisch veranschaulicht.

Die Prinzipskizze zeigt einen Panoramascanner mit einem Gesichtsfeld von 360° × 88°. Das bedeutet, dass ein horizontales Gesichtsfeld von 360° und ein vertikales von 88° aufgespannt wird. Weitere Bauformen sind in der Literatur von Luhmann (2002) zu finden.

Die Abtastung der Umgebung erfolgt über die Veränderung des Vertikal- und Horizontalwinkels. Mit der Streckenmessung vom Laserscanner zum Reflexionspunkt definiert sich ein Kugelkoordinatensystem {0; r, θ, φ}. Durch die Beschreibung eines solchen Koordinatensystems wird jedem Winkelpaar eine Entfernung zugeordnet.

Ausnahme stellt eine Mehrfachimpulsauswertung (multi pulse) dar. Hierbei wird ein ausgesendetes Signal, das an einer Kante geteilt wird, welches zu unterschiedlichen Laufzeiten führt, getrennt voneinander ausgewertet. Damit erhält eine Winkelposition mehrere Entfernungen [Kraus, 2004].

Die Berechnung der kartesischen Koordinaten erfolgt aus den zuvor genannten Kugelkoordinaten. Bei einer Laserscanneraufnahme sind die im Folgenden aufgelisteten Fehlereinflüsse zu beachten:

- Divergenz des Laserstrahls
- Atmosphärische Einflüsse
- Auftreffwinkel des Laserspots
- Messungen an Kanten

Die Laserspotgröße hat einen entscheidenden Einfluss auf die Extraktion von Kanten. Die Reflexion eines hinreichend großen Laserspots an einer Kante führt zu einer Verfälschung der Entfernungsmessung. In Bild 6 ist dieses Problem skizziert.

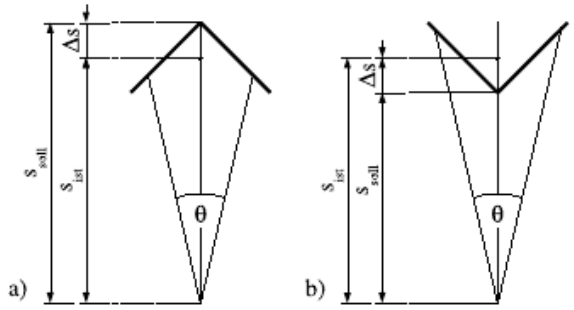


Bild 6: Prinzipskizze zu Messabweichungen a) an einer konkaven und b) an einer konvexen Kante.

Kanten mit konkaver Ausrichtung werden zu kurz und Kanten mit konvexer Ausrichtung zu lang gemessen. Diese Auswirkung verstärkt sich mit zunehmender Spotgröße des Laserstrahls und hat eine Glättung von Kantenbereichen in der Punktwolke zur Folge. Weitere Ausführungen zu diesem Thema sind in der Literatur von Kern (2002) zu finden.

2.1.7 Anforderungen Laserscanner

Die Abbildung der 3D-Oberfläche erfolgt mit Hilfe eines Laserscanners. Die Messdaten sind Grundlage für die Berechnung der Punktwolken.

Die minimalen Spezifikationen des Scanners sollten nach derzeitigem Stand der Technik sein:

- 2 Mio. Punkte pro Sekunde
- Erfassungsbreite: min. 30 m
- Laserklasse 1 – augensicher
- Genauigkeit der Längenmessung <7 mm
- Min. 1 mm Entfernungsauflösung
- Wasserdicht und staubdicht min. IP67 (unempfindlich gegen Witterungseinflüsse bei Überführungsfahrten – Qualitätssicherung der Messung)
- Aufnahme eines Intensitätsbildes min. 10bit

Bild 7: 3D Punktwolke des CPS Scanners mit 10° Sichtschatten der Referenzfaser zum Boden gerichtet. Die Referenzfaser dient der internen Kontrolle der Längen- und Winkelmessung pro Umdrehung des Laserscanners.

2.1.8 Mobile Mapping und Laserscanning

Der Unterschied vom terrestrischen Laserscanning zum Mobile Mapping System ist, dass der Vertikalwinkel r durch die Vorwärtsbewegung des Fahrzeuges ersetzt wird. Der Laserscanner misst hier eine Art Helix, wobei der Helixabstand von der Geschwindigkeit des Fahrzeuges abhängig ist. Die einzelnen Messpunkte werden zeitlich hochgenau dem Positionierungssystem zugeordnet, sodass durch eine Kalibrierung eine eindeutige Georeferenzierung erfolgen kann.

Die aktuellen hochgenauen Mobile Mapping Systeme unterscheiden sich hier im folgenden Maße:

- Genauigkeit des Positionierungssystems
- Reichweite und Anzahl des/der Laserscanner
- Messverfahren (Impulslaufzeit-/Phasen-vergleichsverfahren)
- Genauigkeit des Laserscanners
- Attribuierung der Geometriedaten (z.B. Intensitätsdaten, Farbinformationen)

Aktuelle Mobile Mapping Systeme mit hochgenauen Positionierungssystemen und Laserscanner (Zugriff 2016):

- ***RIEGL* VMX-450**

(http://www.riegl.com/nc/products/mobile-scanning/produktdetail/product/scannersystem/10/

- **StreetMapper 360**

(http://www.igi.eu/streetmapper.html) / Riegl Laserscanner-Daten

- **Dynascan mobile mapping**

(https://www.renishaw.com/en/dynascan-land-based-mobile-mapping-and-surveying--25586

- **topcon IP-S2 Compact +**

(http://topconcare.com/en/hardware/mobile-mapping/ip-s2/

- **trimble MX8**

(http://www.trimble.com/Imaging/Trimble-MX8.aspx

- **optech Lynx SG Mobile Mapper**

(http://www.teledyneoptech.com/en/products/mobile-survey/lynx-hs300/

- **Leica Pegasus Mobile Mapping Solution**

(https://leica-geosystems.com/products/mobile-sensor-platforms/capture-platforms/leica-pegasus_two

-**SITECO Roadscanner**

(http://www.sitecoinf.it/en/solutions/road-scanner)

- **LEHMANN+PARTNER GMBH (LP) I.R.I.S**

(http://www.lehmann-partner.de/technologie/system-iris)

Die Positionierungssysteme der hier aufgelisteten Fahrzeuge/Komponenten haben annähernd ähnliche Spezifikationen. Die Laserscanner sind i.d.R. von den Firmen Riegl, Z+F, SICK, Optec, FARO und Fraunhofer IPM Freiburg Germany.

Alle Systeme erfassen eine georeferenzierte Punktwolke, die sich zur Geometrie/Objekt-erkennung eignet. Als minimales Austauschformat kann eine Textdatei (Easting, Northing, Height + epsg code) oder eine LAS Datei dienen.

2.1.9 Photogrammetrische Bilder

Zur Dokumentation und Objekterfassung werden Messkameras eingesetzt. Diese sind photogrammetrisch kalibriert und erlauben nachträglich eine Objekterfassung bzw. unterstützen diese (zusammen mit den Laserscannerdaten).

Für eine eindeutige Georeferenzierung muss sichergestellt werden, dass jede Aufnahme/Bild einen eindeutigen Zeitstempel besitzt. Über diesen Zeitstempel ist es möglich, das aufgenommene Bild eindeutig einer Trajektorienposition zuzuordnen. Dies sollte über eine externe Triggerung erfolgen, da mindestens eine Genauigkeit für den Zeitstempel von 10^{-5}s erreicht werden muss. Anderenfalls wird das Bild einer falschen Station zugeordnet und alle weiteren Transformationen und Rotationen zur 3D-Messung entsprechen nicht den Genauigkeitsansprüchen (vgl. Scheller 2007).

Die geforderte Genauigkeit liegt bei 10 bis 50 cm im gemessenen Einzelbild je nach verwendeter Trajektorie.

2.1.10 Kalibrierung

Um eine eindeutige mathematische Beschreibung der einzelnen Sensoren sicherzustellen muss jeder Sensor, der zur Georeferenzierung von Objekten dienen soll, kalibriert werden. Es ist zu gewährleisten, dass alle Sensoren fachgerecht eingemessen sind. Dies ist Grundlage für alle Messungen bzw. Messergebnisse jedes einzelnen Sensors. Weiterhin sind alle Sensoren regelmäßig einer Überprüfung zu unterziehen, um ggf. Veränderungen in der Kalibrierung rechtzeitig zu erkennen.

Jede Kamera sollte einzeln in Bezug zum Fahrzeugkoordinatensystem eingemessen sein. Dies beinhaltet mindestens die Sensorgröße, Pixelgröße, Kamerakonstante c, die Hauptpunkt-verschiebungen in x und y Richtung, die Ver-zeichnungsparameter (z.B. nach Braun A1, A2, A3, B1, B2, C1, C2, C3). (Luhmann, T. (2002))

Bei allen Sensoren ist mindestens ein Boresight Alignement zu ermitteln. Dies beschreibt die Rotation rx, ry, rz und Translation tx, ty, tz vom Sensorkoordinatensystem zum übergeordneten Fahrzeugsystem. Hierbei muss sichergestellt werden, dass alle Parameter eindeutig beschrieben werden. Drehrichtungen, Drehreihenfolgen, Linkshand- vs. Rechtshandsystem. Jeder Hersteller verwendet hier seine eigenen Koordinaten-systemdefinitionen. Dies beginnt mit der Festlegung des Nullpunktes im Fahrzeug und der Festlegung der Koordinatenachsen. Die Sensoren besitzen in der Regel ein eigenes Koordinatensystem, welches in das jeweilige Fahrzeugsystem zu überführen ist.

Anforderungen an eine Sensor-Kalibrierungsdatei:

- Jeder Sensor besitzt eine Kalibrierungsdatei

- Kalibrierungsdatei sollte menschenlesbar sein
- alle Einheiten der Parameter sind dokumentiert
- ersichtliches Datum der letzten Kalibrierung

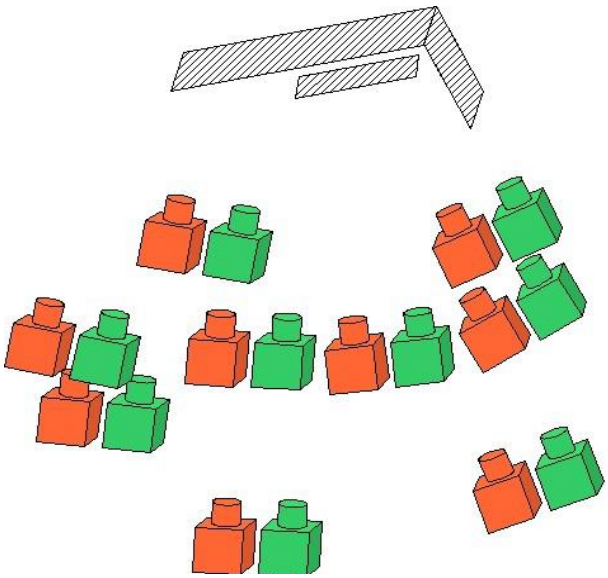


Bild 8: Kalibrierungspositionen eines Mobile Mapping Fahrzeuges vor einer Kalibrierungswand mit einzelnen georeferenzierten Messmarken (Scheller 2007).

Die einheitliche Kalibrierung der Sensoren ist Grundvoraussetzung für eine eindeutige Beschreibung der Sensoren und hat somit direkten Einfluss auf die Genauigkeit der erstellten Messungsdaten.

2.1.11 Datenablage und Datenformate

In den letzten Jahren haben sich unterschiedliche Austauschformate entwickelt, die eine Punktwolke speichern können. Diese Formate kommen aus verschiedenen Bereichen der Informatik oder der Geodäsie. Die weitverbreitetsten sind das Format LAS und E57. Das LAS wurde in Zusammenarbeit mit der " American Society for Photogrammmetrie and Remote Sensing" (ASPRS) erstellt. Dieses wird hauptsächlich für luftgestützte Laserscanner-erfassung verwendet.

Das Format E57 wurde seit 2006 vom „ASTM" Committee vorgeschlagen. Viele Firmen haben die Formate als Standard für Laserscannerdaten (vgl. Daniel Huber 2011).

Das einfachste menschenlesbare Austauschformat ist hingegen eine Textdatei. Dieses Format ist i.d.R. mit jeder Software einzulesen und lässt sich mit jedem Texteditor bearbeiten. Nachteile sind die extreme Größe und die lange Ladezeit/Speicherzeit durch die ASCII Interpretation der Datei.

Standard Austauschformate für Laserscannerdaten dateibasiert:

- *.txt – einfaches Format zeilenbasiert (variabel auch GeoJSON möglich)
- *.las - LASF - ASPRS LIDAR Data Exchange Format - open source - Binärformat (http://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html)
- *.e57 – Binärformat mit API (http://www.astm.org/COMMITTEE/E57.htm)
- *.pdc - Point Cloud Library (PCL) Binär/Textformat (http://pointclouds.org/documentation/tutorials/pcd_file_format.php)

Es besteht im Weiteren die Möglichkeit, die Laserscannerdaten in eine Datenbank zu überführen. Der Import der Daten dauert i.d.R. vergleichsweise lange. Die Verarbeitung der Punktwolken kann dann jedoch schneller in der Datenbank erfolgen. Mit den uns z.Z. bekannten Möglichkeiten ist dies eine gute Alternative für Datensätze kleiner 5TB.

2.1.12 Fazit

Die Georeferenzierung eines Mobile Mapping Fahrzeuges erfolgt mit Hilfe eines Positionierungssystems. Das Positionierungs-system kombiniert GPS Messungen, eine Wegstreckenmessung und Messungen einer IMU (inertial measurement unit). Die IMU ist hierbei das Primärsystem und erfasst Beschleunigungs-messungen sowie Winkelunterschiede in allen drei Koordinatenachsen. Alle weiteren Messsensoren werden in die Trajektorie des Positionierungs-systems eingerechnet (2.1.8). Über die zeitliche Kombination des Positionierungssystems und den einzelnen Messsystemen kann eine Georeferen zierung berechnet werden. Grundvoraussetzung hierfür ist eine eindeutige Kalibrierung der Sensoren und somit maßgebend für eine gute Datengrundlage für das weite Forschungsprojekt.

2.2 Mustererkennung

2.2.1 Aufgaben der Objekterkennung in der Analyse von Straßenszenen

Methoden der Muster- bzw. Objekterkennung werden für verschiedene Zielvorgaben entwickelt. Die beiden Begriffe Muster- und Objekterkennung können dabei im Rahmen dieses Berichts weitgehend synonym verwendet werden.

Die Komplexität der Erkennungsaufgabe nimmt mit der Art und Stärke der Variation in den Eingangsdaten zu. Je mehr einzelne Objekte in einer Punktwolke oder einem Kamerabild vorhanden sind und je stärker die einzelnen Objekte in ihrer Erscheinungsform variieren können, entscheidet, ob eine bestimmte Methode ausreichend „mächtig“ ist, eine zuverlässige Erkennung durchzuführen. Die Stärke des Rauschens in den Eingangsdaten sowie weiterer Störeinflüsse können die Mustererkennung zusätzlich erschweren.

Prinzipiell lassen sich folgende Aufgabenstellungen unterscheiden, deren Lösung zunehmend komplexe Methoden erfordert.

Objekterkennung (Object Recognition): Ergebnis der Erkennung ist eine Aussage darüber, ob bzw. wie stark die Eingangsdaten Ähnlichkeit zu zuvor bereits „gesehenen“ Objekten aufweisen. Am besten funktioniert die Klassifizierung, wenn als Eingabe ein einzelnes, mehr oder weniger isoliertes Objekt zur Verfügung steht. Im Kontext von Straßenszenen könnte dies z.B. ein Vorfahrtsschild sein. Typische Lösungen basieren auf dem Matching geeigneter Merkmale in den Bildern oder der Registrierung von Punktwolken. Typische Merkmale sind Farben, Kanten oder geometrische Formen. Häufig werden komplexere Deskriptoren wie z.B. SIFT (Scale-Invariant Feature Transform) oder HOG (Histogram of Gradients) eingesetzt.

Objektlokalisierung (Object Detection): Bestimmung der Position eines Objekts in Bild oder Punktwolke (z.B. als Bounding Box). Meist als Erweiterung zur Objekterkennung, welche selbst keine Aussage zur Position eines Objekts beinhaltet. Die Eingangsdaten enthalten dabei im Allgemeinen mehrere verschiedene Objekte, die sich auch gegenseitig überschneiden können.

Objektklassenerkennung (Object Class Recognition): Klassifizierung von Objekten, die ein abstrakteres Konzept repräsentieren und deshalb in ihrer Erscheinung stärker variieren. Ein typisches Beispiel ist die Klasse „Auto“, worunter Sportwagen ebenso fallen wie Geländewagen, Cabriolets oder Kleinstwagen. Bei den Straßenbestandsobjekten sind dies z.B. verschiedene Straßenabdeckungen (Schacht, Hydrant, etc.), Beleuchtungsanlagen, Bäume, Lichtsignalanlagen (wenn man die Gesamtanlage incl. Masten betrachtet). Methoden zur Objektklassenerkennung basieren i.d.R. auf einem Lernprozess. Aus einem Trainingsdatensatz wird dabei eine geeignete Auswahl an Merkmalen bestimmt anhand derer Instanzen der Objektklasse zuverlässig erkannt werden können.

Lokalisierung von Objektklassen (Object Class Detection): Ähnlich wie bei der Objektlokalisierung geht es um die Bestimmung der Position in Daten, welche mehrere Objekte enthalten.

Erkennung auf Instanzebene (Instance-Level Detection): Unterscheidung mehrerer Instanzen desselben Typs, die sich überschneiden.

Segmentierung: Aufteilung eines Datensatzes in Bereiche (oft als Regions of Interest (ROI) bezeichnet), welche in sich maximal ähnlich sind. Der einfachste Fall ist die Trennung zweier Regionen, etwa des Vorder- und Hintergrunds eines Bildes. Ähnlichkeitskriterien können z.B. wieder Farbe, Form oder Textur sein; bei Bildsequenzen oder Stereobildern auch sekundäre Informationen wie Tiefe oder Bewegung.

Semantische Segmentierung (Semantic Segmentation, Semantic Labeling): Zuordnung jedes Pixels bzw. Punktes zu einem bestimmten Objekttyp oder einer Objektklasse. Sich überschneidende Objekte desselben Typs werden dabei nicht unterschieden.

Semantische Segmentierung auf Instanzebene (Instance-Level Semantic Segmentation): Zusätzliche Unterscheidung sich überlappender Instanzen desselben Objekttyps bzw. von Instanzen einer Objektklasse bei der Segmentierung.

Im Rahmen dieses Forschungsprojekts war eine Klassifikation und Lokalisierung einfacherer Objekte, z.B. bestimmter Verkehrsschilder, sowie von Objektklassen erforderlich. Außerdem war die Erkennung einzelner Instanzen nötig, wenn sich Objekte desselben Typs überschneiden. Abschließend war eine georeferenzierte Positionierung der Objekte und die OKSTRA-konforme Dokumentation der Ergebnisse zu gewährleisten.

2.2.2 Mustererkennung in Punktwolken (3D)

Die Methoden zur Muster- bzw. Objekterkennung in Punktwolken werden im Folgenden übersichtsartig dargestellt. Der Schwerpunkt der Literaturanalyse liegt auf den Methoden, welche RGB-Bilder (ggf. zusammen mit Tiefenbildern) als Eingangsdaten nutzen, da diese in den Lösungsstrategien bevorzugt werden.

2.2.2.1 Erläuterungen zur Datenbasis

Eine Punktwolke stellt die Repräsentation einer Szene als (zunächst) unsortierte Menge von Punkten in 3D-Koordinaten dar (3-Punktwolke). Zusätzlich zur 3D-Information steht vorliegend auch der vom Laserscanner empfangene Intensitätswert zur Verfügung (4-Punktwolke).

Im Kontext von Laserscanning wird des Öfteren der Begriff 2½D verwendet. Somit wird der Tatsache Rechnung getragen, dass Informationen nur auf der dem Laserstrahl zugewandten Oberfläche eines Objekts aufgenommen werden, nicht aber auf der dem Sensor abgewandten Seite.

Die beiden vom Fraunhofer IPM entwickelten Laserscanner Pavement Profile Scanner (PPS) und Clearance Profile Scanner (CPS) zeichnen 2D-Profile mit Scanraten von 800 Hz bzw. 200 Hz auf. Die Samplingfrequenz beträgt 1 MHz bzw. 2 MHz. Die dritte Dimension der Daten ergibt sich aus der Vorwärtsbewegung des Messfahrzeugs. Aus den Messdaten wurde von LP eine georeferenzierte Punktwolke erzeugt.

Aufgrund des Messprinzips ergaben sich für die Nutzung der Punktwolken als Ausgangspunkt für eine Klassifizierung Herausforderungen hinsichtlich der Intensitätsinformation und der geometrischen Anordnung der Scanprofile.

2.2.2.2 Nutzung der Intensitätsinformation

Die von einem Laserscanner wahrgenommene Intensität eines Messpunktes variiert mit dessen Entfernung vom Sensor. Die Intensität eines Objekts ist als Merkmal deshalb nur eingeschränkt nutzbar. Hierin unterscheiden sich lasergenerierte Daten von bspw. Kamerabildern, bei denen die wahrgenommene Intensität unabhängig von den Objektdistanzen ist. Zusammen mit der Entfernungsinformation könnte man – unter zusätzlichem Berechnungs- bzw. Kalibrierungsaufwand – die Intensitätsinformation nutzen. Es ist allerdings dann noch mit einer relativ starken Variation zu rechnen, da die Menge des zurückreflektierten Lichts zusätzlich noch von den Witterungsbedingungen, z.B. nassen Oberflächen, abhängig ist. Auch retroreflektierende Flächen stellen für laserbasierte Messsysteme im Allgemeinen ein Problem dar, da die Empfangselektronik, zumindest kurzzeitig, „geblendet“ wird.

2.2.2.3 Nutzung der geometrischen Information

Die Messungen eines Laserscanners, die als Punktwolke vorliegen, sind ideal geeignet, um genaue Informationen über die räumliche Struktur von Objekten zu erfassen. Speziell die Ebenheit von Oberflächen lässt sich sehr exakt bestimmen.

Für die Identifikation voluminöser Objekte, wie z.B. Bäume oder Laternenmasten, kann die Geometrie ein wesentlicher Anhaltspunkt sein. Wenn „echte“ 3D-Daten vorliegen, können zylindrische Objekte bspw. durch die kreisförmige oder elliptische Anordnung von Punkten in einem Schnitt durch das Objekt oder durch die Analyse von Normalenvektoren erkannt werden.

Bei 2½D-Daten sind diese geometrischen Merkmale unvollständig ausgeprägt, was deren Definition und Auswertung erschwert. Auch das Volumen von Objekten ist hier als Merkmal nur eingeschränkt nutzbar, da nicht alle Seiten von Objekten sichtbar sind.

Aufgrund des Abstandes zwischen den Scanprofilen, der von der Geschwindigkeit des Messfahrzeugs abhängt, werden auf Objekten mit geringer räumlicher Ausdehnung, wie z.B. den Masten von Schildern, selbst bei einer Scanfrequenz von 200 Hz, wie beim CPS, nur wenige Profile aufgezeichnet. Auch eine Verkippung des Laserscanners wird hier keine wesentliche Verbesserung mit sich bringen.

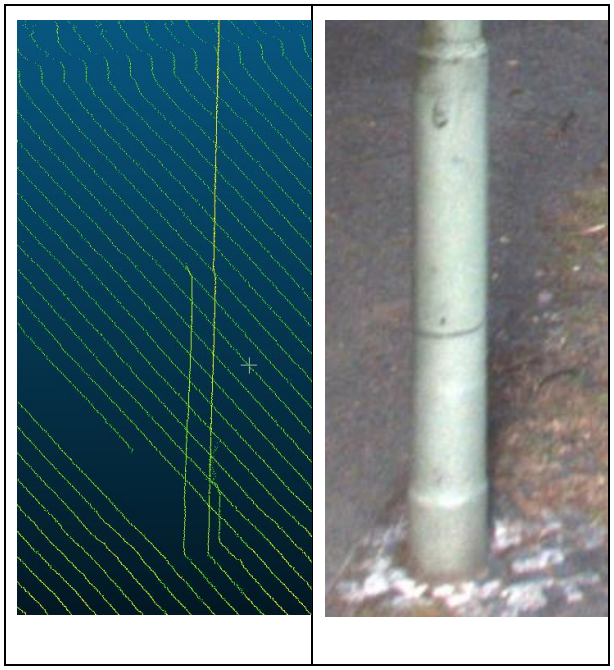


Bild 9: Der Mast einer Beleuchtungseinrichtung in der Punktwolke des CPS-Scanners (li.) und als RGB-Bild (re.).

Der Unterschied zwischen Kamerabild und Punktwolke ist exemplarisch in Bild 9 zu sehen. Obwohl die 3D-Daten entlang des Scanprofils sehr hoch aufgelöst sind, ist es dennoch mit dem RGB-Bild besser möglich, eine Unterscheidung von ähnlichen Objekten (Schild/Mast, Baumstamm) zu treffen.

2.2.2.4 Objektidentifikation in Punktwolken

Um Objekte in Punktwolken zu lokalisieren und zu klassifizieren gibt es eine Vielzahl verschiedener Ansätze. Für die Analyse von Straßenszenen werden häufig geometrische Eigenschaften von Objekten sowie a-priori-Wissen über den typischen Aufbau einer Straßenszene kombiniert. So entsteht ein mehrstufiges Vorgehen, indem z.B. zunächst über die Suche nach einer ebenen Fläche eine grobe Kategorisierung in Fahrbahn, Objekte auf der Fahrbahn sowie Objekte über der Fahrbahn vorgenommen wird (PU et al. 2011). Bordsteine lassen sich z.B. über Sprünge in den Höhenwerten bestimmen (vgl. ZHOU, VOSSELMANN 2012, dort im Vergleich zu Mobile und Airborne Laserscanning).

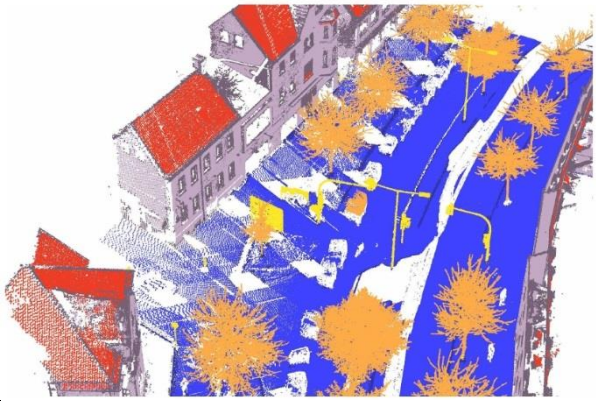
Andere Methoden zur Objekterkennung in 3D (bspw. die Bildverarbeitung) gehen stärker merkmalsbasiert vor. Hierzu zählen auch Ansätze, welche für komplexere Szenen mit Verdeckungen und einer größeren Anzahl an Objekten geeignet sind. Unterscheiden lassen sich hierbei globale und lokale Ansätze, je nach Art der verwendeten Merkmale.

Einen Überblick über Methoden, welche lokale Merkmale auf Oberflächen betrachten, bietet z.B. GUO et al. 2014. Eine verbreitete Vorgehensweise ist, zunächst Schlüsselpunkte (Keypoints) zu bestimmen, in deren Umgebung relevante Merkmale vermutet werden. Um diese Keypoints werden Merkmale, wie z.B. HOG (Histogram of Gradients), berechnet. Über die Keypoints und deren Merkmale ist dann anhand von Ähnlichkeitskriterien ein Matching mit anderen Datensätzen (bzw. Ausschnitten davon) möglich, um bekannte Objekte zu identifizieren.

Einen speziellen Deskriptor, mit dem Merkmale von Keypoints in Punktwolken beschrieben werden können, bietet der NARF-Ansatz (STEDER et al. 2011). Dieser überführt die Punktwolken dabei zunächst in Tiefenbilder.

2.2.2.5 Analyse geometrischer Eigenschaften und graphenbasierter Modelle

Ein Beispiel für einen Ansatz, welcher zwei verschiedene Methoden für die Klassifikation von Objekten in Straßenszenen kombiniert, ist in BREMER et al. 2013 zu finden. Es werden sieben verschiedene Klassen definiert: Boden, Bodeninventar, Wand, Wandinventar, Dach, künstlicher Masten, Baum (sowie eine weitere Klasse „Undefiniert“ für nicht klassifizierbare Punkte). Einen Eindruck der Ergebnisse, die erzielt wurden, zeigt Bild 10.



Quelle: BREMER et al. 2013

Bild 10: Klassifizierung einer Punktwolke anhand lokaler Merkmale, sowie einer graphenbasierten Klassifikation von pfahlartigen Objekten.

Wesentlicher Bestandteil für die Detektierung von Objekten ist hierbei die Suche nach bestimmten geometrischen Formen (Linie, Ebene, etc.). Dies geschieht anhand der Analyse von Eigenwerten

und Eigenvektoren einer Kovarianzmatrix für eine lokale Umgebung einzelner Punkte. Die Auswertung findet in BREMER et al. 2013 für zwei verschiedene Radien statt. Robustheit gegenüber Rotation und Skalierung der Objekte ist in gewissen Grenzen gewährleistet (vgl. GROSS, THOENESSEN 2006).

Die „finale“ Identifikation pfahlartiger Objekte (auch Bäume) erfolgt dann mit einem graphenbasierten Klassifikator, der fortschreitende Verzweigung analysiert.

Als Datenbasis für die Evaluation dient die Befahrung einer 200 m langen Strecke mit zwei im 45°-Winkel zur Fahrtrichtung ausgerichteten Optech Lynx-Laserscannern. Der durchschnittliche Abstand von Punkten beträgt 0,02 m. Die erzielte Genauigkeit (Accuracy) reicht von 0,69 für Bodeninventar (Ground Inventory) bis zu 0,99 für Boden (Ground) und Baum (Tree), ausgedrückt als der relative Wert korrekt zugeordneter Punkte.

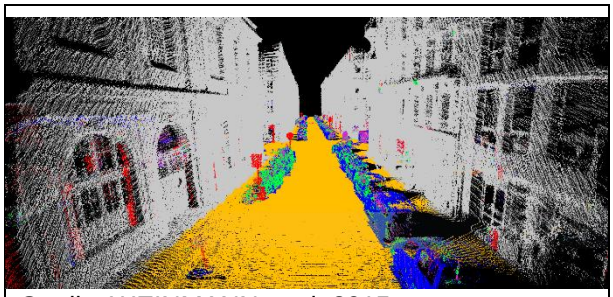
Die durchschnittliche Genauigkeit ist mit 0,94 angegeben. Es scheint im Datensatz jedoch wenig Verdeckungen z.B. durch ineinander verwachsene Bäume zu geben, was die mitunter hohen Genauigkeiten ein Stück weit relativiert.

2.2.2.6 Optimierung der Nachbarschaftsbestimmung

Einen vergleichbaren Ansatz verfolgten WEINMANN et al. 2015. Hier wurde ein Framework aus Bausteinen vorgestellt, welches eine flexiblere Auswahl der Nachbarschaftsgröße sowie eine Beschränkung auf relevante Merkmale erlaubte und eine Klassifikation einbezog, die auf überwachtem Lernen (Supervised Learning) basiert. Es wurden verschiedene Bausteine variiert sowie der Einfluss unterschiedlicher Nachbarschaftsgrößen untersucht.

Evaluiert wurden die verschiedenen Setups des Frameworks auf den beiden öffentlich verfügbaren Datensätzen „Oakland 3D Point Cloud Dataset“ und „Paris-rue-Madame-Database“. Als Objektklassen wurden für den Oakland-Datensatz Draht (Wire), Masten/Stamm (Pole/Trunk), Wand (Facade), Boden und Vegetation betrachtet. Für den Paris-Datensatz erfolgte die Definition der Objektklassen Wand (Facade), Boden, Auto, Motorrad, Verkehrsschild und Fußgänger.

Die erreichten Genauigkeiten (Precision) variierten entsprechend stark und reichten (für das jeweils beste Setup) von 0,09 für Draht über 0,89 für Wand bis zu 0,998 für Boden. Eine Klasse „Bodeninventar“ gab es hier nicht. Die beste erreichte Genauigkeit betrug 0,89, der beste erreichte Recall-Wert 0,85. Einen visuellen Eindruck hierzu vermittelt Bild 11.



Quelle: WEINMANN et al. 2015

Bild 11: Klassifizierte Punktwolke einer Straßenszene nach der Methode von WEINMANN et al. 2015. Problematisch erscheint z.B. die Identifikation von Verkehrsschildern (rot), welche fälschlicherweise in der Fassade von Häusern positioniert werden.

2.2.2.7 Fazit zur Objekterkennung in Punktwolken

Die verschiedenen Ansätze zur Objektidentifikation in Punktwolken beschränken sich meist auf wenige Objekte bzw. Objektklassen oder auf Objekte, welche in ihrer Erscheinung nur geringer Variation unterliegen. Bei stärker variierenden Objekten (bspw. Bäume) war es offensichtlich, dass ein komplexes Modell implementiert werden musste. Für die detaillierte Unterscheidung einer größeren Anzahl komplexerer Objektklassen, wie in diesem Projekt gefordert, erschien es deshalb notwendig, weitere Informationsquellen einzubeziehen, wie z.B. RGB-Bilddaten (vgl. zu dieser Einschätzung z.B. PU et al. 2011).

Speziell die Lücken in den Punktwolken, die durch den Abstand zwischen einzelnen Scanprofilen entstehen, erschwerten eine eindeutige Detektion von Objekten.

Kamerabilder enthielten darüber hinaus weitere Informationen, welche für die Klassifizierung hilfreich sein könnten, wie z.B. die Farbe von Verkehrsschildern.

Die Tiefeninformation aus der Punktwolke wiederum konnte einen wesentlichen Beitrag zur Auflösung der Mehrdeutigkeiten in Bildern leisten, wenn sich z.B. Objekte mit ähnlichen Merkmalen (teilweise) überdeckten.

Optimale Ergebnisse waren dann zu erwarten, wenn sowohl die Punktwolke als auch RGB-Bilder als Informationsquelle verwendet wurden. Um die gemeinsame Nutzung zu ermöglichen, müssten die Daten in angemessener Weise fusioniert bzw. zueinander in Bezug gesetzt werden.

Im Rahmen dieses Forschungsprojekts war die Lokalisierung und Klassifizierung einer Vielzahl verschiedener Objektklassen gefordert, deren Instanzen in ihrer Erscheinung stark variieren. Die Implementierung von merkmalsbasierten Detektoren in 3D erschien zwar für viele Objekte möglich, war jedoch für jede einzelne Objektklassen mit einem erheblichen Aufwand für die Merkmalsdefinition und die entsprechende Implementierung verbunden.

Vielversprechender erschien auch in dieser Hinsicht der Ansatz des Deep Learning, welcher geeignete Merkmale aus einem Trainingsdatensatz lernen und sowohl Lokalisierung als auch Klassifizierung übernehmen kann.

2.2.3 Mustererkennung in Bilddaten (2D)

2.2.3.1 Klassische Lösungsansätze

Als „klassische“ Methoden werden im Folgenden solche bezeichnet, welche nicht auf komplexen Lernalgorithmen wie dem Deep Learning beruhen.

Eine etablierte Vorgehensweise für die Erkennung „einfacher“ Objekte (Object Recognition) ist, ähnlich wie bei der Mustererkennung in 3D, Keypoints zu identifizieren und Merkmale aus der Umgebung zu berechnen. Dabei ist besonders darauf zu achten, die Merkmale robust gegenüber der zu erwartenden Variation zu gestalten. Dies betrifft insbesondere solche Variationen in der Erscheinung von Objekten, die auf die Beleuchtungssituation und den Blickwinkel des Betrachters zurückgehen oder die teilweise Verdeckung von Objekten. Anhand der so gewonnenen Merkmalsbeschreibung lassen sich Eingangsdaten über die Ähnlichkeit zu den gesuchten Objekten klassifizieren.

Die Erkennung von Objektklassen (Object Class Recognition) erfordert aufgrund der stärkeren Variation in der Erscheinung das „Lernen“ bzw. das Training eines Klassifikators, z.B. einer Support Vector Machine (SVM). Dieser bestimmt anhand eines Trainingsdatensatzes, der aus manuell ausgewerteten und annotierten Bildern besteht, welche Merkmale für eine bestimmte Objektklasse relevant sind, also in den Trainingsdaten besonders häufig auftreten. Merkmale, welche nur wenige Instanzen einer Objektklasse aufweisen oder die für eine eindeutige Bestimmung nicht hilfreich sind, werden vom Klassifikator als ungeeignet erkannt.

Für eine Vielzahl von Aufgaben im Bereich der Objekterkennung haben sich jedoch in den vergangenen 3 bis 4 Jahren künstliche neuronale Netze und das Deep Learning als besonders geeignet erwiesen (s.u.).

2.2.3.2 Bildsegmentierung zur Identifikation von Regions-of-Interest

In Bildern, welche eine Vielzahl von Objekten enthalten, ist es sinnvoll, zunächst interessante Bereiche zu identifizieren, welche dann jeweils separat klassifiziert werden. Die Lokalisierung ist dann am Ende als Bounding Box um die erkannten Objekte verfügbar, für die ROI, mit der das beste Ergebnis erzielt wurde.

Eine typische Vorgehensweise ist der Sliding-Window-Ansatz, welcher Bounding Boxes verschiedener Größe über das gesamte Bild bewegt und den jeweiligen Inhalt der Box analysiert bzw. klassifiziert. Die vielversprechendsten Ergebnisse werden festgehalten und am Ende ausgewählt. Um den Rechenaufwand in Grenzen zu halten, ist hierbei ggf. eine Beschränkung der Anzahl von Skalierungen und Positionen nötig, worunter die Qualität der Ergebnisse leiden kann.

Eine weitere Herangehensweise stellt die sog. Aufmerksamkeitssteuerung dar. Anhand vordefinierter Merkmale (z.B. Farbe, Kanten, Formen) werden Bereiche identifiziert, in denen einzelne Merkmale – oder Kombinationen von Merkmalen – besonders ausgeprägt sind. Diese Regionen werden dann für die genauere Analyse oder Klassifizierung ausgewählt.

2.2.3.3 Erkennung von Straßenbestands-objekten

Für die Identifikation einzelner Bestandsobjekte werden in der Literatur verschiedene Ansätze vorgestellt. Einige Methoden haben ihren Weg bis in den Alltag der Navigationssysteme gefunden, wo z.B. Verkehrsschilder automatisch erkannt werden, oft zusammen mit einer GPS-basierten Verortung und einer Karte, welche bekannte Positionen von Schildern enthält.

In der klassischen Bildverarbeitung wird meist eine Kombination verschiedener Filter verwendet, um am Vorhandensein einzelner Merkmale auf ein bestimmtes Objekt schließen zu können.

Komplexere Deskriptoren wie SIFT (Scale-Invariant Feature Transform) oder HOG (Histrogram of Gradients) verbessern die Robustheit der Ergebnisse. Ein Beispiel für die Anwendung von HOG ist z.B. ADAM, IOANNIDIS 2014, dort in Kombination mit einer Support Vector Machine (SVM) für die Klassifikation.

Der Einsatz von Hough-Algorithmen zur Erkennung kreisförmiger Objekte ist ebenfalls eine bewährte Methode. Zusammen mit einer zusätzlichen Erkennung von Ziffern auf Schildern mit Geschwindigkeitsbeschränkung mittels eines

neuronalen Netzes findet er sich z.B. in MOUTARDE et al. 2007. Neuronale Netze für die Klassifikation einzelner Schilder wurden in ähnlicher Weise z.B. schon von DE LA ESCELERA et al. 1997 verwendet.

Verschiedene Blickwinkel auf ein Objekt entsprechen einer affinen Transformation zwischen den jeweiligen 2D-Abbildungen. Es existieren Merkmale bzw. Deskriptoren, welche robust gegenüber solchen Transformationen sind, z.B. ASIFT (Affine SIFT) oder MSER (Maximally Stable Extremal Regions). Diese eignen sich deshalb besonders für das Matching von Merkmalen, wenn die Eingangsdaten affinen Transformationen unterliegen.

2.2.3.4 Künstliche neuronale Netze (KNN) und Deep Learning

Im Bereich der Objekterkennung sind die künstlichen neuronalen Netze (KNN) den klassischen Methoden jedoch inzwischen überlegen. Dies gilt vor allem dann, wenn es darum geht, eine Vielzahl verschiedener Objekte zu erkennen, deren Erscheinung zudem stark variiert. Sie sind außerdem robuster gegenüber Variationen, die typisch für Straßenszenen sind, z.B. Verdeckungen, Ausbleichen von Farben, Beschädigungen oder Verschmutzungen.

Interessant sind in diesem Zusammenhang auch die beiden Testdatensätze German Traffic Sign Recognition Benchmark (GTSRB, vgl. STALLKAMP et al. 2012) und German Traffic Sign Detection Benchmark (GTSDB, vgl. HOUBEN et al. 2013). Es geht dabei um die Klassifizierung bzw. Lokalisierung von Verkehrsschildern, wofür umfangreiche Trainings- und Testdatensätze bereitgestellt werden. Methoden, welche KNNs verwenden, schneiden am besten ab. Ausführlicher beschrieben wird ein solcher Ansatz z.B. in SERMANET; LECUN 2011. Einige Bilder, die auch diesen Ansatz vor Probleme stellen, zeigt Bild 12.

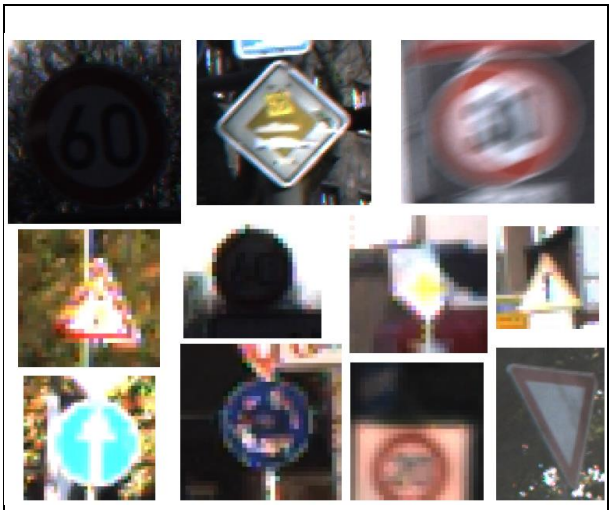


Bild 12: Einige besonders schwer zu klassifizierende Bilder von Verkehrsschildern aus dem GTSRB, zusammengestellt in SERMANET, LECUN 2011.

Die Idee, künstliche neuronale Netze (KNN) in der Bildverarbeitung einzusetzen, ist nicht neu – „einfache“ Aufgaben wie die Handschrifterkennung wurden bereits Ende der 1980er Jahre durch den Einsatz von KNN gelöst (sog. LeCun-Netzwerk). Die typische Architektur der damaligen Netze wird heute als „flach“ bezeichnet und umfasste max. 8-10 Schichten mit meist weniger als 20.000 Neuronen.

Im Bereich der Bildverarbeitung sind insbesondere die sog. Convolutional Neural Networks (CNNs oder ConvNets) von Interesse (vgl. z.B. ZEILER; FERGUS 2014). Diese machen sich die Tatsache zunutze, dass Informationen in Bildern meist lokal, d.h. auf einen kleineren Teilbereich des Bildes beschränkt sind. Deshalb können für die Analyse des Bildinhaltes Filter verwendet werden, welche nur einen kleinen Teil des Bildes betrachten, z.B. 7x7 oder 11x11 Pixel.

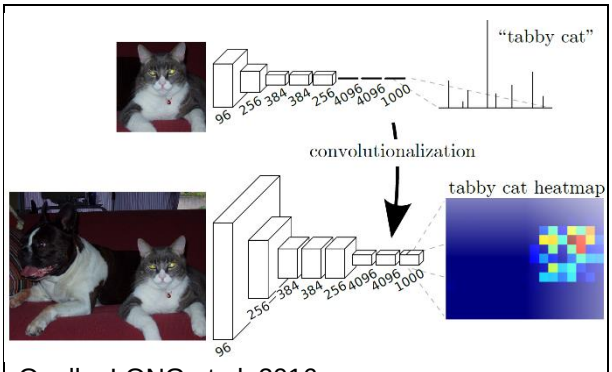
Dadurch ist eine wesentlich geringere Anzahl an Neuronen notwendig als dies für vollständig verbundene Netze der Fall ist, was eine erhebliche Einsparung an Rechenzeit und Speicherbedarf mit sich bringt. Dennoch wurde es erst in den vergangenen Jahren möglich, Netze zu entwickeln, die mit der Menge an Variation umgehen können, welche die Eingangsdaten für eine Erkennung komplexerer Objekte aufweisen. Dies ist insbesondere der Verfügbarkeit leistungsfähiger und hoch parallelisierter Grafikprozessoren (GPUs) zu verdanken.

Der Durchbruch für solche „tiefen“ Netze (Deep Networks) mit mehr als 10 Schichten und mehreren Millionen Neuronen wurde im Jahr 2012 durch das sog. AlexNet (KRIZHEVSKY et al. 2012) erreicht.

Dieses erzielte für die Klassifikation von Bildern aus dem ImageNet-Datensatz wesentlich bessere Ergebnisse als alle klassischen Methoden zur Objekterkennung. Den schematischen Aufbau eines Netzes zur Klassifikation zeigt der obere Teil von Bild 13. Deutlich sichtbar ist die Reduzierung der Auflösung (Lokalisierung, codiert durch die Fläche der einzelnen Schichten) mit zunehmender Tiefe des Netzwerks, bei gleichzeitigem Anwachsen des Merkmalsraums (Tiefe der Schichten, angegeben als Zahlenwert unter den einzelnen Schichten des Netzwerks).

Die Stärke der neuronalen Netze liegt insbesondere auch darin begründet, dass die Merkmale, die sie lernen, im Ggs. zu „manuell“ definierten Merkmalen (wie z.B. Farbe, Form, SIFT, HOG), mehrere Hierarchieebenen aufweisen. Sie repräsentieren dadurch zunehmend abstrakte Konzepte. Durch die Reduzierung der Auflösung am Übergang zwischen den Schichten des Netzwerks wird außerdem ein immer größerer Bereich des Bildes betrachtet. Am Ende steht i.d.R. eine Analyse des gesamten Bildes durch mehrere vollständig verbundene Schichten.

Die Gewichte der Verbindungen zwischen den Neuronen, welche eine Gewichtung der Ergebnisse der einzelnen Filter des Netzes vornehmen, werden beim Training des Netzes berechnet. Dieses erfolgt anhand von manuell annotierten Trainingsdaten. Die Gewichte werden dabei durch die sog. Backpropagation, eine spezielle Variante des Gradientenabstiegs (Gradient Descent), bestimmt. Die Berechnung ist sehr aufwändig, da je nach Komplexität der Aufgabe mehrere Millionen Gewichte aus der Analyse mehrerer Zehn- bis Hunderttausend Bilder gewonnen werden müssen.



Quelle: LONG et al. 2016

Bild 13: Schematischer Vergleich der Architektur zweier neuronaler Netze: oben: Netz zur Klassifikation; unten: Netz zur Berechnung der ungefähren Lokalisierung eines Objekts anhand der Aktivierungen der Merkmalskarte (Heatmap). Die Zahlen unter den Schichten repräsentieren die Anzahl an Merkmalskarten.

Die Gewichte enthalten am Ende die gesamte Information, welche zur Klassifikation von Objekten notwendig ist. Sie entscheiden darüber, wie stark ein durch einen Filter repräsentiertes Merkmal gewichtet wird und für eine Aktivierung von Neuronen in nachfolgenden Schichten sorgt.

Während das Training sehr rechenaufwändig ist, auch auf leistungsfähigen Grafikkarten mehrere Tage bis Wochen in Anspruch nehmen kann, führen die fertig trainierten Netze die Analyse eines Bildes i.d.R. schnell aus. Typische Werte sind Prozessierungszeiten von 5 ms pro Bild für einfache Aufgaben bis zu ca. 1 s für komplexere Analysen; natürlich weiterhin abhängig vom Grad der Optimierung der Algorithmen und der Leistungsfähigkeit der verwendeten Hardware.

Der Ansatz des Deep Learning wurde inzwischen bereits auf eine Vielzahl von Anwendungen übertragen und für verschiedene Arten von Eingangsdaten optimiert, so z.B. auch für die zusätzliche Verwendung der Tiefeninformation in RGB-D-Bildern (mehr dazu s.u.).

2.2.3.5 Objekterkennung (Object Recognition) mit Deep Learning

Neuronale Netze sind inzwischen so weit fortgeschritten, dass sie für manche Klassifikationsaufgaben, z.B. Teile des oben bereits erwähnten ImageNet-Benchmarks (vgl. RUSSAKOVSKY et al. 2015), besser abschneiden als menschliche „Tester“ (HE et al. 2015).

Das ist beeindruckend; man darf diese Aussage allerdings auch nicht überbewerten, da es für einen Computer leicht ist, auch weniger bekannte Objekte, wie spezielle Hunderassen, zu identifizieren, wenn er sie durch den Trainingsdatensatz lernen konnte (vgl. Bild 14).

Für das Training beim ImageNet stehen für jede der 1.000 zu erkennenden Klassen im Durchschnitt ca. 1.000 Bilder zur Verfügung, die das entsprechende Objekt aus verschiedenen Blickwinkeln, sowie in verschiedenen Beleuchtungen und Umgebungen zeigen.

Neuronale Netze (und die Ansätze der Computer Vision allgemein) haben andererseits Probleme, Dinge zu erkennen, die für einen Menschen trivial sind, z.B. bestimmte Objekte oder Handlungen, deren Bedeutung primär vom Kontext abhängig ist.

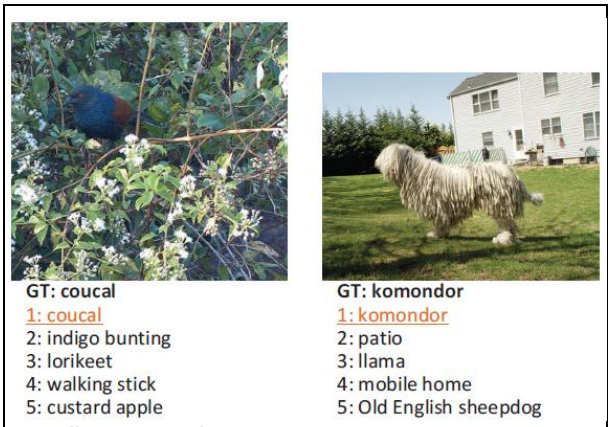


Bild 14: Ausgabe der fünf wahrscheinlichsten Objektnamen für zwei Bilder aus dem ImageNet-Datensatz, klassifiziert mit dem Ansatz von HE et al. 2015 (GT = Ground Truth). Speziell solche „ausgefallenen" Objekte können Netze zuverlässiger klassifizieren als menschliche Probanden.

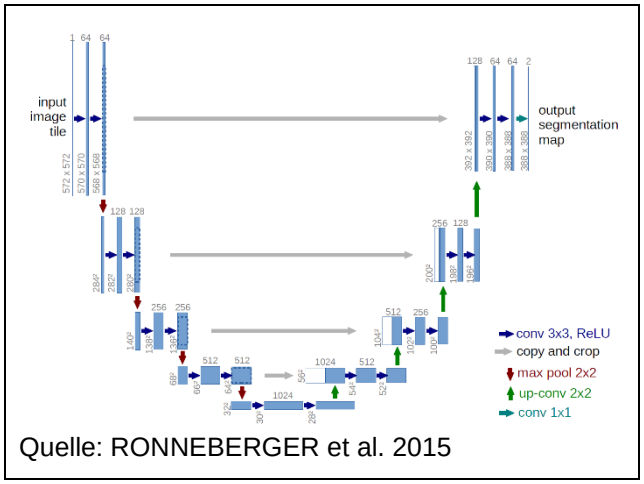


Bild 15: Schema der Architektur eines Up-Convolutional Networks für die semantische Segmentierung von Bildern. Der linke Teil übernimmt die Klassifizierung, während die Information über die Lokalisierung von Objekten verloren geht. Der rechte Teil stellt diese Information durch Upsampling wieder her.

Bei der Identifikation von Straßenbestandsobjekten können die Stärken neuronaler Netze jedoch genutzt werden; die gesuchten Objekte können weitgehend isoliert betrachtet werden. Der Kontext oder anderes High-Level-Wissen spielen eine untergeordnete Rolle.

2.2.3.6 Lokalisierung und semantische Segmentierung

Bei der semantischen Segmentierung geht es darum, jeden Pixel eines Bildes einer bestimmten Klasse zuzuordnen. Für realistische Szenen ist dies im Allgemeinen eine schwierige Aufgabe, speziell bei Verdeckungen, deformierbaren Objekten oder bei gleichzeitigem Vorhandensein vieler ähnlicher Objekte (vgl. z.B. Bild 17).

Netze für die semantische Segmentierung besitzen eine deutlich aufwändigere Architektur als Netze für die Klassifikation, deren Ausgabe lediglich ein „Score" (Wahrscheinlichkeit) je zu bestimmender Klasse ist; die Information über die Position eines Objekts im Bild geht verloren. Eine Architektur, die für diese Aufgabe vorgeschlagen wird, sind die sog. Up-Convolutional Networks.

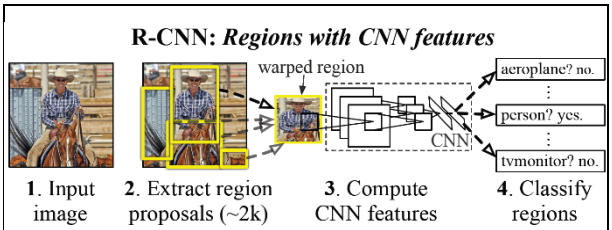
Diese schalten dem Klassifizierungsteil weitere Schichten nach, welche die Aktivierungen der Merkmale der Klassifikation „upsampeln", um am Ende eine Auflösung zu erzielen, die der Eingangsauflösung nahekommt.

Das Netz lernt wiederum anhand von Trainingsdaten, wie es dieses Upsampling durchführen kann. Dabei lernt es außerdem, Informationen über die Lokalisierung bestimmter Merkmale aus dem ersten Teil des Netzes zu nutzen. Einen schematischen Überblick über diese Netzarchitektur zeigt Bild 15. Ein Beispiel für ein solches Netz ist das U-Net (RONNEBERGER et al. 2015), welches z.B. für die Segmentierung biomedizinischer Aufnahmen erfolgreich eingesetzt wird.

Ein mehrstufiger Ansatz zur Objekterkennung und Bildsegmentierung wird von GIRSHICK et al. 2014 vorgeschlagen. Ausgehend von „klassischen" Methoden der Bildverarbeitung, welche interessante Regionen im Bild identifizieren, erfolgt die Berechnung von Merkmalen in den einzelnen Regionen durch ein neuronales Netz. Die Klassifizierung übernimmt eine Support Vector Machine (SVM). Einen Überblick über die Verarbeitungskette dieses als R-CNN bezeichneten Ansatzes zeigt Bild 16. Eine Weiterentwicklung dieses Netzwerks als „Mask R-CNN" erfolgte durch HE et al. 2017.

Andere Methoden gehen etwas anders vor und lassen das Netz eine Merkmalsverteilung auf dem gesamten Eingangsbild berechnen. So kombinieren z.B. HARIHARAN et al. 2014 ein Klassifikationsnetz zur Erzeugung von Merkmalskarten, eine SVM für die Klassifizierung und einen separaten Algorithmus zur Verfeinerung der finalen Segmentierung. Als Bezeichnung des Vorgehens wird Simultaneous Detection and Segmentation (SDS) gewählt. Es wird gezeigt, dass mit dieser Methode

Überdeckungen von Objekten besser erkannt und korrigiert werden.



Quelle: GIRSHICK et al. 2014

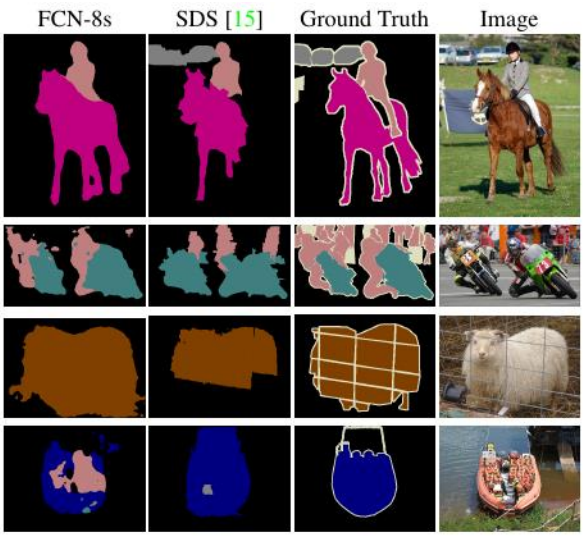
Bild 16: Auf die Extrahierung interessanter Regionen folgt eine Merkmalsbestimmung mit einen neuronalen Netz; die Klassifizierung wird schließlich von einer Support-Vector-Machine (SVM) durchgeführt.

Exemplarische Ergebnisse zweier aktueller Ansätze zur semantischen Segmentierung sind in Bild 17 zu sehen: HARIHARAN et al. 2014, im Bild mit SDS bezeichnet, sowie LONG et al. 2016, mit FCN-8s betitelt, wobei FCN für Fully Convolutional Network steht. LONG et al. können hier eher überzeugen, da sie eine detailliertere Segmentierung erreichen, aber auch hier treten Artefakte oder falsche Klassifizierungen von Bildbereichen auf.

Eine wesentliche Voraussetzung für die Netze zur semantischen Segmentierung besteht darin, eine ausreichende Menge an annotierten Trainingsdaten zur Verfügung zu haben, was nicht immer der Fall ist. Eine manuelle Annotierung ist vergleichsweise aufwändig, da jedes Objekt exakt nachgezeichnet werden muss. Ein klarer Vorteil dieses Ansatzes besteht jedoch darin, dass ein Ende-zu-Ende-Training möglich ist und nicht mehrere Algorithmen für jeden Teilschritt separat implementiert und optimiert werden müssen.

2.2.3.7 Semantische Segmentierung von Straßenszenen

Eine detaillierte Analyse von Straßenszenen, welche nicht auf wenige Objektklassen beschränkt ist, ist für viele Anwendungen von großem Interesse, wie auch im Rahmen dieses Forschungsprojekts zur Detektion von Straßenbestandsobjekten. Für die Automobilindustrie und speziell das autonome Fahren sind darüber hinaus noch bewegte Objekte sowie die Echtzeitfähigkeit bedeutend.

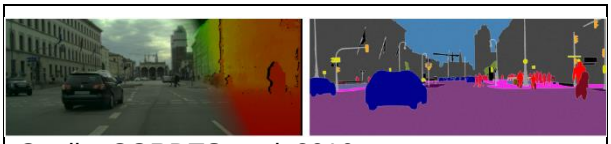


Quelle: LONG et al. 2016

Bild 17: Semantische Segmentierung teilt ein Bild in semantisch zusammengehörige Regionen auf, indem jeder Pixel eines Bildes einer bestimmten Klasse zugeordnet wird (hier durch die Farbe codiert). Ground Truth ist die manuelle Annotierung. Die Segmentierung komplexer Szenen stellt nach wie vor eine Herausforderung dar.

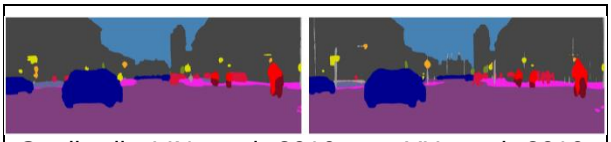
In der Forschung ist in diesem Bereich viel Entwicklung festzustellen. Seit kurzem ist z.B. der City-scapes-Datensatz verfügbar, welcher annotierte semantische Segmentierungen für Straßenszenen enthält (CORDTS et al. 2016). In den Trainingsdaten, welche Bilder der linken und rechten Kamera eines Stereo-Setups bereitstellen, sind u.a. die Klassen Verkehrsschild, Ampel, Gebäude, Schutzplanke (Guard Rail), Mast (Pole) und Vegetation definiert. Ein Beispielbild mit zugehöriger manueller Annotation zeigt Bild 18.

Einen ersten Eindruck über (vorläufige) Ergebnisse vermittelt Bild 19. Diese wurden mit den Methoden von LIN et al. 2016 sowie YU et al. 2016 erzielt; die Ergebnisse wurden vorab in CORDTS et al. 2016 veröffentlicht. Sie geben eine ungefähre Vorstellung dessen, was mit Deep Learning auf Daten erreicht werden kann, die ähnlich denen sind, die im Rahmen dieses Projekts zu den Straßenbestandsobjekten zur Verfügung stehen. Objekte, die in Bild 19 nicht oder nur unzuverlässig erkannt werden, weil sie weit von der Kamera entfernt sind, können ggf. in nachfolgenden Bildern zuverlässiger detektiert werden.



Quelle: CORDTS et al. 2016

Bild 18: Der Cityscapes-Datensatz stellt u.a. Stereo-RGB-Bilder bereit, aus denen auch Tiefeninformationen berechnet werden können (li. eine Überblendung von RGB-Bild und Tiefenkarte). Das rechte Bild zeigt eine semantische Segmentierung aus den manuell annotierten Trainingsdaten (die Klassen sind durch Farben codiert).



Quelle: li.: LIN et al. 2016; re.: YU et al. 2016; abgebildet in: CORDTS et al. 2016

Bild 19: Erste Ergebnisse auf dem Cityscapes-Datensatz; die hier referenzierten Veröffentlichungen von LIN et al. 2016 und YU et al. 2016 waren zu diesem Zeitpunkt noch vorläufig.

2.2.3.8 Nutzung von Tiefeninformation in neuronalen Netzen

Für das vorliegende Forschungsprojekt zu den Straßenbestandsobjekten standen sowohl Bildsequenzen als auch zwei Punktwolken (CPS- und PPS-Scanner) zur Verfügung. Die darin enthaltene Tiefeninformation können als zusätzliche Datenquelle die Segmentierung der Bilddaten unterstützen, da ähnliche Objekte, die sich in ihren Tiefenwerten unterscheiden, leichter separiert werden können.

Da die Daten vollständig kalibriert vorliegen, können die Kamerapositionen in der Punktwolke verortet werden. Die Verfolgung der Projektionsstrahlen der einzelnen Pixel erlaubt es, einen Tiefenwert aus der Punktwolke zu bestimmen.

Ergebnis der Fusionierung sind Bilddaten in RGB-D, also Farbbilder, welche zusätzlich mit einem Tiefenwert versehen sind. Ein pixelbasiertes Format zu wählen erscheint deshalb geboten, da es ein ideales Eingabeformat für KNNs darstellt und „kompatibel“ mit den RGB-Bildern ist.

Für Objekterkennung anhand solcher Daten wurden bereits Methoden entwickelt (z.B. GUPTA et al. 2014), welche darauf hoffen lassen, dass die Einbeziehung der Tiefeninformation zur Robustheit der vorgeschlagenen Datenprozessierung beitragen kann. Etwas grundsätzlichere Überlegungen zur gemeinsamen Nutzung von Kamera- und Laserscannerdaten im Kontext von Deep Learning bietet z.B. GIERING et al. 2015. Verweise auf weitere Methoden zur Objekterkennung mittels RGB-D-Bildern finden sich in GUO et al. 2014 (oder auch FILIPE, ALEXANDRE 2014).

2.2.3.9 Überlegungen zur Tiefe von neuronalen Netzen

Die zugrundeliegenden Methoden und Implementierungen bei der Nutzung künstlicher neuronaler Netze zur Objekterkennung sind einer ständigen Weiterentwicklung unterworfen. Im Folgenden werden einige Ansätze vorgestellt, welche – für bestimmte Anwendungen und unter bestimmten Voraussetzungen – eine Steigerung der zu erwartenden Performanz eines KNN erwarten lassen.

Insbesondere wurden bereits früh Experimente mit immer tieferen neuronalen Netzwerken durchgeführt, d.h. die Anzahl der Schichten im Netzwerk erhöht. Diese zeigen, dass unter bestimmten Voraussetzungen eine erhöhte Erkennungsgenauigkeit bei gleichzeitiger Verringerung der Komplexität des gelernten Modells erreicht werden kann. Die Reduzierung der Komplexität kann wiederum einen positiven Einfluss auf die Resistenz eines Netzes gegenüber Overfitting ausüben. Vorreiter bei der Analyse des Einflusses der Tiefe des Netzwerkes auf die Klassifizierungsleistung sind SZEGEDY et al. 2014 mit ihrem GoogLeNet.

Wird die Tiefe der Netzwerke noch weiter erhöht, zeigt sich jedoch, dass solche Netze dazu tendieren, langsamer oder sogar schlechter zu lernen. Dies liegt darin begründet, dass die Gradienteninformation bei der Backpropagation durch viele Schichten weitergereicht werden muss. Dabei wird pro Schicht jeweils nur die dort relevante Änderung des Gradienten berücksichtigt, sodass nach vielfacher Ableitung diese Änderung immer geringer ausfällt und im Extremfall keine nutzbare Information mehr enthält. Eine Möglichkeit, sehr tiefe Netze wieder schneller und zuverlässiger trainieren zu können, besteht in der Nutzung sog. Residual Functions. HE et al. 2015 haben für ihre Residual Networks (ResNets) z.B. Versuche mit Netzen mit einer Tiefe von bis zu 152 Schichten durchgeführt. Auf dem Cityscapes-Benchmark erzielen WU et al. 2016 eine gute Platzierung mit ihrem Residual Network „ResNet-38“.

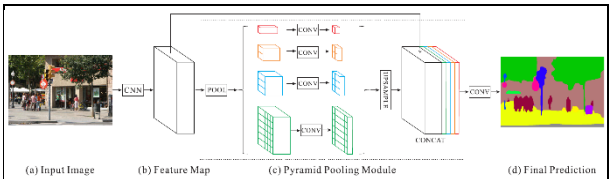
SIMONYAN; ZISSERMAN 2015 führten hierzu ihre eigenen Experimente durch und entwickelten die sogenannte VGG-Architektur, welche bei verschiedenen Benchmarks für die Klassifizierung von Bildern sehr erfolgreich war. Die Übernahme

der VGG16-Architektur für die pixelweise semantische Segmentierung von Bilddaten in den FCN-Netzen (Fully Convoluational Network) bei LONG et al. 2016 bildet auch die Basis für die von uns final verwendete Netzwerkarchitektur.

2.2.3.10 Weitere Ideen zur Anpassung künstlicher neuronaler Netze

Für spezielle Anwendungsfälle ist es z.B. möglich, die Erkennungsgenauigkeit durch komplexe Anpassungen der Netzarchitektur zu erhöhen. Ein Beispiel hierfür ist die Klassifizierung von Objekten in stark variierenden, in sich aber jeweils ähnlich aufgebauten Arten von Szenen, etwa in einer Fußgängerzone voller Menschen oder einer einsamen Szene am Meer. Das KNN kann lernen, welche Objektklassen typischerweise in solchen diversen Szenen vorkommen. So ist das Vorkommen von Fahrzeugen in einer Szene am Meer eher unwahrscheinlich, während Boote in einer Innenstadtumgebung selten auftreten. Dieses Vorwissen kann also genutzt werden, Verwechslungen insbesondere zwischen solchen Klassen zu reduzieren, welche in ihrer Erscheinung ähnlich sind, jedoch nur selten gemeinsam in einer bestimmten Situation auftreten.

Ein Beispiel für diesen Ansatz ist das PSPNet (Pyramid Scene Parsing Network, ZHAO et al. 2017). Hier wird zunächst die Gesamtszene klassifiziert und diese „globale“ Information dann als zusätzlicher Input für die pixelweise semantische Segmentierung der Bilddaten verwendet. Außerdem wird die Netzarchitektur durch ein „Pyramid Pooling Module“ erweitert, welches die zu verarbeitenden Informationen in verschiedenen Auflösungen analysiert (s. Bild 20). Diese Architektur ist allerdings nicht zur freien Nutzung verfügbar.



Quelle: ZHAO et al. 2017

Bild 20: Das PSPNet (ZHAO et al. 2017) erweitert die Netzwerkarchitektur um globale Informationen zur Art von Szene, welche im Bild zu sehen ist, sowie um ein pyramidenförmiges Pooling-Modul.

Hier muss jedoch, wie immer im Zusammenhang mit komplexen Erweiterungen der Netzwerk-architektur, berücksichtigt werden, dass diese Netze oftmals mit sehr hohem Aufwand daraufhin optimiert wurden, in einem bestimmten Benchmark durch die Erhöhung der Erkennungsgenauigkeit um wenige (Zehntel-) Prozent auf einen der vorderen Plätze zu gelangen. Es ist zudem immer abzuwägen, ob der Mehraufwand für das Labeln der Trainingsdaten und das Training des KNN dem zu erwarteten Zuwachs an Erkennungsgenauigkeit entspricht. Oder ob nicht bereits eine Erhöhung der Anzahl der gelabelten Trainingsdatensätze einen ausreichend hohen Zuwachs an Genauigkeit und eine ebenso erhöhte Generalisierungsleistung des Netzes erwarten lässt.

2.2.3.11 Fazit zur Mustererkennung in Bilddaten

Für die Detektion einer größeren Anzahl an Bestandsobjekten aus umfangreichen und hoch aufgelösten, heterogenen Datenquellen, wie in diesem Forschungsprojekt gefordert, erscheint die Nutzung eines auf Deep Learning basierten Ansatzes besonders vielversprechend. Insbesondere die Fähigkeit, sehr gut mit starker Variation in der Erscheinung von Objekten umgehen zu können, spricht für den Einsatz tiefer neuronaler Netze.

Semantische Segmentierung von Bilddaten, welche Klassifizierung und Lokalisierung in sich vereint ist für die Detektierung der Straßenbestandsobjekte in diesem Forschungsprojekt ein idealer Ansatz. In der Literatur finden sich vielversprechende Ergebnisse, welche mit vergleichbaren Ansätzen auf RGB-Bilddaten erzielt wurden. Vgl. hierzu speziell CORDTS et al. 2016 zum Cityscapes-Datensatz, oder auch ZHOU et al. 2017 zum ADE20K-Datensatz.

Architektur und Training eines solchen Netzes sind jedoch aufwändig. Dies betrifft insbesondere auch die Verfügbarkeit bzw. die Generierung eines geeigneten und ausreichend großen Trainingsdatensatzes. Generell gilt bei Anwendungen, welche auf künstlicher Intelligenz bzw. maschinellem Lernen basieren, dass der Gesamtaufwand für die Implementierung einer Objekterkennung nicht geringer ausfällt als bei „traditionellen“ Ansätzen. Arbeitsintensiv ist hier neben der Qualitätssicherung bezüglich des Trainingsdatensatzes auch die die Suche nach einem geeigneten Satz an Hyper-Parametern für das Training, sowie die Evaluierung verschiedener Architekturen des KNN.

2.2.4 Übersicht über mögliche Lösungsstrategien

Für das Forschungsprojekt wurden zwei verschiedene Lösungsstrategien untersucht, welche im Folgenden kurz skizziert.

2.2.4.1 Strategie 1: Semantische Segmentierung mittels Deep Learning

Ein Überblick über die anvisierte Prozessierungskette für „Strategie 1“ ist in Bild 21 zu sehen. Wesentlicher Bestandteil dieses Lösungsansatzes ist ein künstliches neuronales Netz (KNN), welches mittels Deep Learning für die semantische Segmentierung von RGB(-D)-Bilddaten trainiert wird. Die Informationen aus den Punktwolken der beiden Laserscanner könnte für jede Kameraposition in eine Tiefenkarte (D-Komponente) überführt werden, welche zusammen mit den RGB-Bildsequenzen der Kameras die Eingangsdaten für das Netz bildet.

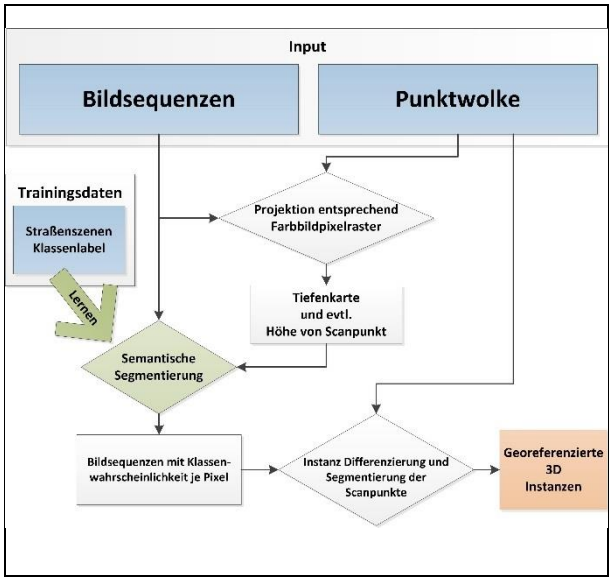


Bild 21: Übersicht über die Prozessierungskette bei „Strategie 1“: Durch eine semantische Segmentierung anhand von RGB- und Tiefeninformationen werden Position und Klasse von Objekten durch ein KNN bestimmt und anschließend in georeferenzierte Ergebnisse überführt.

Das Training des Netzes erfolgt iterativ in verschiedenen Schritten, welche in Bild 22 dargestellt sind.

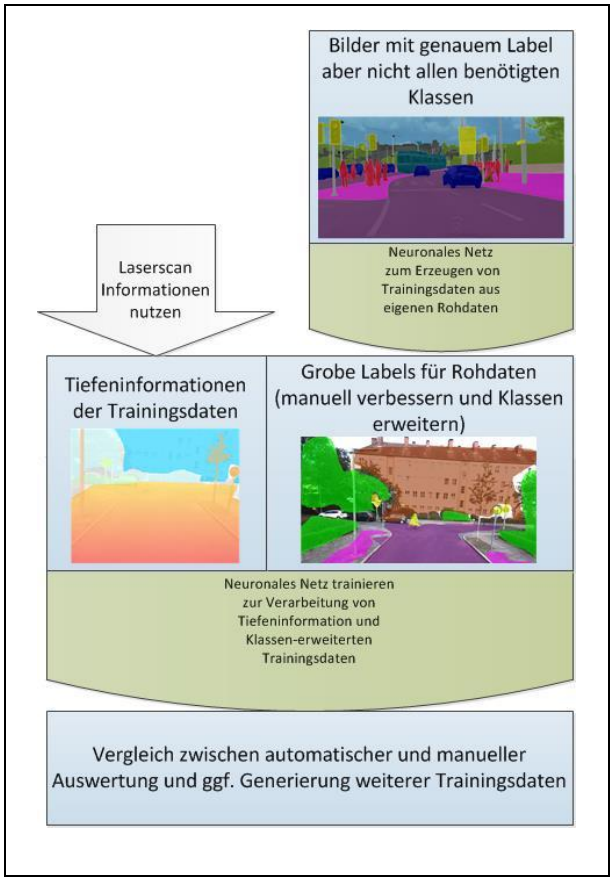


Bild 22: Übersicht über die Schritte für das Training des Netzes, das eine semantische Segmentierung anhand von RGB(-D)-Daten berechnet.

Bei „Strategie 1“, der für die Objekterkennung wesentliche Algorithmus, das neuronale Netz, Ende-zu-Ende trainierbar ist. Dies bedeutet, dass die Verarbeitungskette, wenn das Netz bereits trainiert ist, um neue Objektklassen erweitert werden kann. Hierzu müssen allerdings Trainingsdaten für die neue Objektklasse erzeugt werden und damit ein neuer Trainingsdurchlauf aufgesetzt werden. Die Merkmale der neuen Klassen werden dann wieder während des Trainings automatisch gelernt.

Eine Herausforderung bei diesem Lösungsansatz stellt, neben der Generierung einer geeigneten und ausreichend großen Menge an Trainingsdaten, die Implementierung und Evaluierung einer geeigneten Architektur des neuronalen Netzes dar. Im Projekt zeigte sich, dass dieser Ansatz auf Grund der Komplexität der Erzeugung der Trainingsdaten nicht praxistauglich ist und wurde daher nicht weiterverfolgt.

2.2.4.2 Deep Learning-basierte Klassifikation

In Bild 23 ist die Übersicht über die Bearbeitungsschritte der Prozessierungskette von „Strategie 2“ zu sehen.

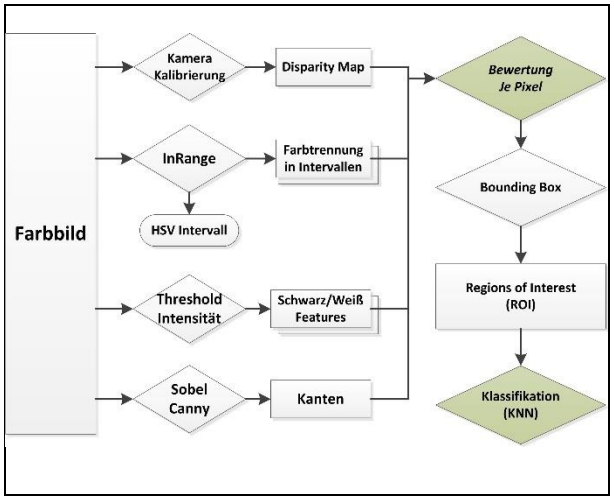


Bild 23: Übersicht über die Prozessierungskette von „Strategie 2“. Sie besteht aus zwei Hauptkomponenten: der Bestimmung von Bounding Boxes (ROIs) anhand vordefinierter Merkmale, die pixelweise ausgewertet werden. Es folgt eine Klassifizierung der ROIs durch ein KNN.

Dieser zweistufige Ansatz ist insofern vorteilhaft, dass das neuronale Netz zur Klassifizierung von Objekten relativ einfach gehalten werden kann, da es Bildausschnitte und nicht gesamte Bilder analysieren muss.

Die Berechnung der ROIs stellt in diesem Ansatz eine Herausforderung dar, da anhand eines vordefinierten Satzes an Merkmalen eine zuverlässige Auswahl erfolgen muss, damit keine Objekte übersehen werden. Andererseits würde eine große Zahl an fälschlicherweise als interessant deklarierten Bildausschnitten den Rechenaufwand beträchtlich erhöhen. Die Vielzahl an zu erkennenden Objekten stellt in dieser Hinsicht eine besondere Herausforderung dar.

2.2.4.3 Dokumentation der Ergebnisse

Anhand der Pixelkoordinaten der in den Bilddaten erkannten Objekte lassen sich die zugehörigen (georeferenzierten) Koordinaten in der Punktwolke bestimmen. Somit kann am Ende die Segmentierung der Punktwolke in Regions of Interest (ROIs) vorgenommen werden.

Für die Ablage der Ergebnisse soll, soweit möglich, eine OKSTRA-konforme Datenbankschnittstelle entworfen werden.

2.2.4.4 Nutzung der Point Cloud Library (PCL)

Die Software-Bibliothek Point Cloud Library (PCL) bietet eine Vielzahl von Funktionen zur Repräsentierung, Bearbeitung, Analyse und Speicherung von Punktwolken. Sie ist Open-Source und wird unter der 3-Clause BSD License bereitgestellt. Sie ist für die Nutzung mit C++ entwickelt; es existieren Schnittstellen (Bindings) z.B. für Python, die jedoch unter Umständen mit Einschränkungen bei der Benutzung verbunden sind.

Nützlich sind zunächst Funktionen, welche die unstrukturierte Punktwolke in Datenstrukturen wie z.B. octrees und kdtrees überführen. Diese sind Voraussetzung für eine praktikable räumliche Analyse, z.B. um Nachbarschaftsbeziehungen untersuchen zu können.

Darüber hinaus sind Algorithmen für verschiedene Bearbeitungsaufgaben verfügbar, darunter verschiedene Filter, um z.B. Ausreißer zu entfernen, oder Operatoren, welche ein Downsampling der Punktwolke erlauben.

Funktionen zur Objektidentifikation umfassen verschiedene Bereiche wie die Registrierung von Objekten bzw. Punktwolken, oder merkmals- und/oder Keypoint-basierte Ansätze wie NARF (STEDER et al. 2011). Im Bereich der Segmentierung können z.B. Ebenen, Zylinder als Basis dienen, genauso wie Algorithmen auf Basis von Farbinformation, Region Growing oder auch graphenbasierte Ansätze.

Die PCL wurde im vorliegenden Projekt nicht tiefergehend eingesetzt, da die Kombination 2D- und 3D-Daten direkt umgesetzt wurde und als sequenzieller Prozess abläuft.

2.3 OKSTRA Datenformat

Der OKSTRA ist eine Sammlung und Modellierung von Objekten aus dem Straßen- und Verkehrswesen. Ziel ist es, bereichsübergreifend und umfassend die Objekte einheitlich zu beschreiben, um ein gemeinsames Verständnis für diese Objekte herzustellen und insbesondere den Austausch von Daten zwischen verschiedenen Fachbereichen.

Die Koordinierung und Weiterentwicklung des OKSTRA obliegt der Projektgruppe OKSTRA des Bund-/Länder-Fachausschusses IT-Koordinierung.

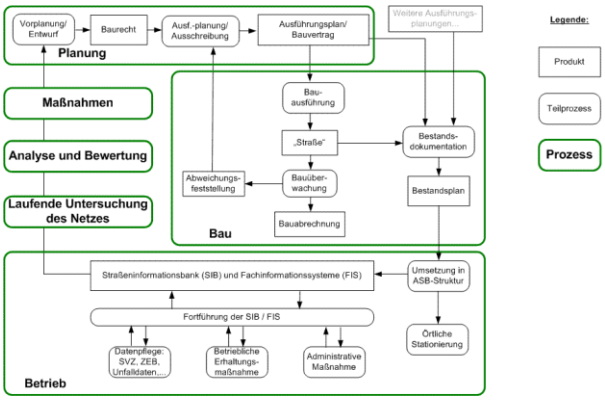


Bild 24: Mustergeschäftsprozess gemäß aktuellem Regelwerk.

Die Thematik im vorliegende Forschungsprojekt ist mittels Bild 24:Mustergeschäftsprozess gemäß aktuellem Regelwerk in die „Bestands-dokumentation" einzuordnen. Die Vermessung bildet somit den gerade aktuellen Stand des Straßenabschnittes ab.

Ziel war es, die extrahierten Objekte aus den Befahrungsdaten zu extrahieren und in eine OKSTRA- konforme Abbildung zu überführen.

Gemäß der ASB-Netz [11,22] bedarf jeder an der Straße ermittelte Tatbestand zur Speicherung und Weiterverarbeitung eines Ordnungsmerkmals. Das Merkmal muss so beschaffen sein, dass der mit ihm versehene Tatbestand jeweils eindeutig örtlich zugeordnet werden kann.

Die nach ASB definierten Objekte sind im OKSTRA entsprechend modelliert. Die Topologie wird einheitlich durch Abschnitte, Äste, Netzknoten und Nullpunkte und ggf. zusätzlich über Straßenelemente und Verbindungspunkte definiert. Die Geometrie einer Straße repräsentiert gemäß ASB jedoch die „Bestandachse". Die Bestandsachse verläuft grundsätzlich in der Straßenmitte und ist einem Abschnitt bzw. Ast eindeutig zuzuordnen. Die Fahrbahnachse verläuft in der Fahrbahnmitte und beschreibt die Geometrie jedes Straßenelements.

Alle Stützpunkte der Straßengeometrie werden über ein Punktobjekt verortet. Optional kann eine Höhenkoordinate angegeben werden. Die nach ASB definierten Objekte sind im OKSTRA entsprechend modelliert.

Alle folgenden Kapitel „OKSTRA Objektklassen" dienen als Dokumentation für das weite Forschungsprojekt und zeigen gegebenenfalls mögliche ableitbare Attribute aus der Objekterkennung.

2.3.1 OKSTRA Objektklasse Baum

Die Beschreibung eines Baumes erhält folgende Attribute. Diese sind nur zum Teil automatisch erfassbar.

Erbt von: *ASB_Objekt, Bewuchs, Punktobjekt, Zuordnungsobjekt_ASB, in Klammern gesetzte Attribute können nur bedingt abgebildet werden*

Name	Datentyp
Lage	Lage
(Baumgattung)	(Baumgattung)
(Baumart)	(Baumart)
Stammumfang	Meter
Stammdurchmesser	Meter
Kronendurchmesser	Meter
(Wurzelhalsdurchmesser)	(Meter)
Stammhoehe	Meter
Baumhoehe	Meter
Schiefstand	Schiefstand_Baum

2.3.2 OKSTRA Objektklasse Schutzplanke

Das Objekt Schutzplanke wird durch eine Vielzahl von Attributen beschrieben. Die geometrische Beschreibung erfolgt zum eine über die ASB und über eine georeferenzierte Lage.

Erbt von: *Fahrzeug_Rueckhaltesystem, Streckenobjekt, Zuordnungsobjekt_ASB*

Name	Datentyp
Abstand_von-_Station	Meter
Abstand_bis-_Station	Meter
Lage	Lage
Standort	Standort_Rueckhaltesystem
Gelaender	Dreiwertige_Logik
Mitwirkung_-Gelaender	Dreiwertige_Logik
Unterfahrschutz	Dreiwertige_Logik
Holmform	Holmform_Schutzeinr-_Stahl
Pfostenform	Pfostenform_Schutzeinr_Stahl
Art_Pfosten-befestigung	Art_Pfostenbefestigung_Schutzeinr_Stahl
Pfostenabstand	Meter

2.3.3 OKSTRA Objektklasse Schutzwand

Siehe Bauwerke Kapitel 2.3.11.

2.3.4 OKSTRA Objektklasse Schacht und Deckel

Erbt von: *OKSTRA_Objekt, Streckenobjekt,*

Name	Datentyp
Art	Art_Schacht
Lage	Lage_Schacht-_Strassenablauf
Angaben_zum_-Konus	Angaben_zum_Konus
(Schachttiefe)	(Meter)
Multigeometrie	Multigeometrie

2.3.5 OKSTRA Objektklasse Schild

Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste für Kennung:
'00', 'unbekannt'
'01', 'Einzelschild'
'02', 'Bestandteil eines Mehrfachschildes'

2.3.6 OKSTRA Objektklasse Stationszeichen

Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste für Kennung:
'00', 'unbekannt'
'01', 'Prismenkörper'
'02', 'Tafel (groß, unterhalb der Tagesmarkierung)'
'03', 'Tafel (klein, oberhalb der Tagesmarkierung)'
'04', 'Schild'
'99', 'sonstiges

2.3.7 OKSTRA Objektklasse Lichtsignalanlage

Die Lichtsignalanlage setzt sich aus mehreren Entitäten zusammen. Hier wurde nur beispielhafte Attributliste aufgeführt.

Erbt von: *OKSTRA_Objekt, Punktobjekt,*

Strassenbezugsobjekt

Name	Datentyp
Lage	Lage
Art	Art_Strassenausst_Punkt
Art_sonst	Art_Strausst_Punkt_sonst
Dauereinrichtung	Dreiwertige_Logik
Detaillierungsgrad	Detaillierungsgrad_ASB
Multigeometrie	Multigeometrie

2.3.8 OKSTRA Objektklasse Punkt

Als Punktrelation kommen im OKSTRA mehrere Beschreibungen zur Anwendung. Im Folgenden werden die möglichen Beschreibungen kurz aufgezeigt.

TYP: Art_Strassenausst_Punkt

Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste:
'06', 'Lichtsignalanlage'
'10', 'Verkehrsspiegel'
'11', 'Notrufsäule'
'14', 'Leitpfosten'
'15', 'Kilometerstein, Kilometertafel'
'19', 'Beleuchtung'
'20', 'Bauwerkstafel'
'22', 'Ortsdurchfahrtszeichen'
'99', 'Sonstiges'

Notrufsäule, Leitpfosten, Kilometertafel, Beleuchtung, Ortsdurchfahrtszeichen sind Punktobjekte, bei denen nicht zu rechnen ist, Attribute aus der Objekterkennung zu extrahieren.

Die folgenden beiden Schlüsseltabellen beschreiben die Verortung des jeweiligen Punktobjektes.

TYP: Datenerhebung_Pos_2D

Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste:
'0100', 'Aus GPS'
'1900', 'Aus sonstiger Vermessung ermittelt'
'2000', 'Aus Luftbildmessung oder Fernerkundungsdaten ermittelt'
'3000', 'Aus Netzvermessung ermittelt'

TYP: Datenerhebung_Pos_3D

Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste:
'1000', 'Höchste Positionsgenauigkeit'
'2000', 'Hohe Positionsgenauigkeit'
'3000', 'Mittlere Positionsgenauigkeit'
'9998', 'Nach Quellenlage nicht zu spezifizieren Messung

Mit folgendem Schlüssel wird das Koordinatensystem festgelegt:

Typ: Koordinatenreferenzsystem_3D
Erbt von: -

Name	Datentyp
Kennung	CharacterString
Langtext	CharacterString

Werteliste:
100', 'WGS84_X-Y-Z'
'110', 'ETRS89_Lat-Lon-h'
'120', 'ETRS89_X-Y-Z'

Die georeferenzierten Koordinaten werden letztendlich im "DataType" Koordinate gespeichert.

Typ: Koordinate
Erbt von: -

Name	Datentyp
x_Koordinate	Real
y_Koordinate	Real
z_Koordinate	Real

2.3.9 OKSTRA Objektklasse Bordstein- und Fahrbahnmarkierung

Bordsteine und Fahrbahnmarkierungen werden in OKSTRA als Profilelinie abgebildet.

Bordstein:

TYP: Art_Profillinie

Werteliste:

'111', 'Bordstein an Gehweg linke Seite links'
'112', 'Bordstein an Gehweg linke Seite rechts'
'113', 'Bordstein an Gehweg rechte Seite links'
'114', 'Bordstein an Gehweg rechte Seite rechts'

Markierung:

TYP: BR_Punkt

Sondertyp: Markierungspfeile

Name	Datentyp
Abstand	Meter
Station	Meter
Berechnung	Bedeutung_Berechnung
hat_Aufweitg-_Verbreit-_Verbind	Aufweitung_Verbreit_Verbindung
hat_Abstand-_Achse_Achse	Abstand_Achse_Achse

Werteliste:

1', 'Rechtsabbiegepfeil'
'2', 'Linksabbiegepfeil'
'3', 'Geradeauspfeil'
'4', 'Geradeaus und Links'
'5', 'Geradeaus und Rechts'

Die Fahrbahnmarkierungen als Linien Objekt können über den „DataTyp" Art_des_ZEB_Objektes beschrieben werden.

2.3.10 OKSTRA Objektklasse Verteilerkasten

Der Objekttyp Verteilerkasten kann über folgende Beschreibung erfasst werden.

TYP: Typ_Stueck_Ver_Entsorgungseinrichtung

Erbt von: -

Werteliste:
'01', 'Brunnen'
'02', 'Hydrant'
'03', 'Telefonzellen'
'04', 'Trafostation'
'05', 'Verteilerkasten'
'06', 'Kleinkläranlage'
'07', 'Sammelgrube'
'08', 'Schächte'
'99', 'Sonstige'

2.3.11 OKSTRA Objektklasse Bauwerk

Bauwerke werden als Objekt erfasst und erhalten folgende Kennung. Hierbei wird lediglich die Länge in Abschnittsrichtung erfasst.

Erbt von: Objekt_mit_ID

Name	Datentyp
Gesamtlaenge_Bruecken	Meter
Gesamtlaenge_Tunnel	Meter
Gesamtlaenge_Laermschutzbauw	Meter
Gesamtlaenge_Stuetzbauwerke	Meter

2.4 Zusammenfassung

Die Literaturrecherche hat gezeigt, dass Grundvoraussetzung für eine gesicherte Detektion von Objekten eine qualitativ hochwertige Datengrundlage ist. Durch die gemeinsame Verwendung der Daten von mehreren Sensoren kann deren Informationsgehalt nochmals angereichert werden. Dies wird in der Literatur auch als Sensorfusion bezeichnet. Hierbei werden aus jedem Sensor Attribute extrahiert und zusammengefasst, womit die Zuverlässigkeit und Robustheit der Objekterkennung im Vergleich zu einer separaten Betrachtung der einzelnen Sensordaten signifikant erhöht werden kann.

Für das vorliegende Forschungsprojekt war es von besonderem Interesse, auch Objekte erkennen und unterscheiden zu können, welche aufgrund der Eigenheiten der 3D-Erfassung mit Laserscannern in der Punktwolke nicht ohne weiteres zuverlässig detektiert werden können. Dies betrifft insbesondere kleine und längliche Objekte, auf denen oftmals nur eines oder wenige Scanprofile aufgezeichnet werden. Am Beispiel einer Beleuchtungseinrichtung wurde dies in Kap. 2.2.2.3 aufgezeigt. Anhand ihrer Geometrie können deshalb z.B. Lichtsignalanlagen, Beleuchtungseinrichtungen, Schilder und Masten nur mit großem Aufwand bezüglich der zugrundeliegenden mathematischen Modelle zuverlässig unterschieden werden. Dies gilt in ähnlicher Weise für Leitpfosten, Kilometertafeln und Stationszeichen, die auf Autobahnen zudem oft teilweise von Schutzplanken verdeckt sind. Auch die Hinzunahme des zusätzlichen Intensitätskanals in 4-Punktwolken (XYZI) erscheint nicht ausreichend, um diese Schwierigkeiten zu umgehen. Die Farb- und Texturinformation von RGB-Bildern ist jedoch geeignet, eine eindeutigere Klassifizierung zu ermöglichen.

Deshalb wird die Mustererkennung im Forschungsprojekt in zwei Schritten vorgenommen. Zunächst erfolgt eine Mustererkennung in 2D. Ergebnis ist eine semantische Segmentierung von Rasterdaten, also eine pixelweise Klassifikation der RGB pro Kameraposition.

Die Klasseninformation pro Pixel wird dann zurück in die Laserscanpunktwolke transferiert. Anschließend wird die 3D-Information aktiv genutzt, um das Ergebnis zu verbessern. Die 3D-Information wird dabei in Form von semantischen Heuristiken einbezogen. Anschließend wird die Klassifizierung und Extrahierung von Objektinstanzen in 3D fortgesetzt. Um einzelne Instanzen von Objekten, z.B. verschiedene Schilder, in der Szene zu detektieren, kann dann auf die Bildung Clustern z.B. anhand von Abstandskriterien aufgebaut werden.

Für 2D-Daten stehen außerdem die Klassifikationsalgorithmen aus dem Bereich des Deep Learning unmittelbar zur Verfügung, während der Einsatz von Deep Learning direkt auf Punktwolken noch nicht ausreichend erforscht ist. Dies haben auch eigene Experimente mit dem PointNet von QI et al. 2016 gezeigt. Eine Verschränkung von 2D-Klassifikation unter Nutzung von 3D-Information für Heuristiken stellt damit die beste derzeit umsetzbare Lösung dar.

Deep Learning als Methode des Machine Learning ist ein Ansatz, welcher die aufwändige und in ihrer Beschreibungsmächtigkeit begrenzte manuelle Definition und Implementierung mathematischer Modelle und Verfahren zur Analyse der Daten durch einen trainingsbasierten Algorithmus ersetzt. Hierbei werden aus den Trainingsdaten hierarchische Merkmale extrahiert, anhand derer

eine robuste und zuverlässige Objekterkennung ermöglicht wird. Dies trifft insbesondere für Objektklassen zu, welche in ihrer Erscheinung stark variieren, z.B. Bauwerke, Bäume und Vegetation, oder auch Fahrzeuge und Personen. Neuronale Netzwerke erweisen sich auch als vergleichsweise robust gegenüber unterschiedlichen Lichtverhältnissen und Schwankungen im lokalen Kontrast bei Schattenwurf.

Der Nachteil eines neuronalen Netzwerks besteht darin, dass die am Ende erreichbare Erkennungsgenauigkeit zu einem wesentlichen Teil davon abhängt, dass ein ausreichend großer und einheitlich erzeugter Trainingsdatensatz zur Verfügung steht. Die hierzu notwendige manuelle Annotierung der Trainingsdaten stellt wiederum einen hohen Aufwand dar.

Die Laserscanpunktwolke wird im Forschungsprojekt bei der Mustererkennung eingesetzt: Zum einen für Heuristiken um in Anschluss an den 2D-Klassifizierungsteil Verbesserungen in der 3D-Punktwolke zu erreichen, zum anderen um klassenspezifischer Clustering-Algorithmen georeferenzierte Objekt-instanzen in 3D extrahiert.

3 Methodik

3.1 Mustererkennung mittels Deep Learning

3.1.1 Nutzung der Information aus der Punktwolke

Für ein künstliches neuronales Netz (KNN), das RGB-Bilddaten als Input verarbeitet, stellen Überlagerungen von Objekten mit ähnlicher Farbe und Struktur eine Herausforderung dar. Ein Beispiel hierfür wäre ein rotes Fahrzeug, das einen Radweg mit rotem Fahrbahnbelag überquert. Ein weiteres Beispiel wären Abbildungen von Objekte z.B. auf spiegelnden Flächen oder Wertetafeln: ein KNN kann schwer unterscheiden, ob es die Projektion eines „realen“ Objekts, oder lediglich dessen Abbild.

Um solche Fälle robuster klassifizieren zu können, könnte zur Erweiterung des im Forschungsprojekt implementierten Ansatzes auch die in der Punktwolke vorhandene geometrische Information über die Objekte genutzt werden. Die Abbildung eines Objekts auf einer Werbetafel wäre dann anhand ihrer „untypisch“ flachen Geometrie zu erkennen.

Für einen solchen Ansatz erschien es geboten, die Information aus der Punktwolke derart in ein rasterartiges Format zu bringen, dass zu jedem RGB-Kamerabild ein zugehöriges Tiefenbild (Tiefenkarte) entsteht. Zusätzlich zu den Intensitäten der drei Farbkanäle (i.d.R. zunächst Integer-Werte) steht als weiterer Datenkanal pro Pixel dann ein Entfernungs- bzw. Tiefenwert (zunächst als Fließkommazahl) zur Verfügung.

Die Positionen der Kameras in der Punktwolke bei der Aufnahme der einzelnen RGB-Bilder können aus der von LP berechneten Trajektorie des Messwagens anhand der Kalibrationsdaten des Messfahrzeuges IRIS5 bestimmt werden.

Eine einfache Methode zur Generierung von Tiefenbildern wäre, für jede Kameraposition eine Lochkamera in der Punktwolke zu positionieren und alle Punkte in einer bestimmten Umgebung auf eine virtuelle Bildebene zu projizieren. Der Abstand zur Bildebene ist dann die Tiefeninformation. Es ist jedoch ersichtlich, dass bei dieser Vorgehensweise viele „Lücken“ entstehen, bzw. die Auflösung des Tiefenbildes abstandsabhängig und inhomogen ausfällt. Grundsätzlich stellte sich außerdem die Frage, wie damit umgegangen werden soll, dass in der Punktwolke zunächst alle Objekte „durchsichtig“ erscheinen.

Eine ebenfalls vorstellbare Vorgehensweise, welche die Generierung „dichter“ Tiefenkarten erlaubt, wäre die Vermaschung der Punktwolke mit anschließendem Raytracing für jeden Pixel der virtuellen Bildebene. Jeder Pixel enthält den Entfernungs- bzw. Tiefenwert des Schnittpunkts des verfolgten Strahls mit der nächsten Oberfläche der vermaschten Punktwolke. Es ergaben sich jedoch auch hier gewisse Schwierigkeiten. In Bereichen mit starkem Rauschen oder mit geringer Punktdichte war es kaum möglich, ohne a-priori-Wissen eine geeignete Oberfläche zu bestimmen. Werden die Parameter so gewählt, dass große Lücken, wie z.B. auf der Windschutzscheibe eines Autos, geschlossen werden, gehen an anderer Stelle durch Glättung lokale Informationen verloren. Für die Versuche mit der Vermaschung wurden die Algorithmen der Bibliothek MeshLab verwendet (CIGNONI et al. 2008).

Für die Erstellung von Tiefenbildern wurde ein von LP vorgeschlagener Algorithmus entwickelt, der unten näher beschrieben wird. In die Objekterkennung durch das KNN wurden die Tiefenbilder jedoch in diesem Forschungsprojekt nicht mehr eingebunden.

Als Alternative (oder Ergänzung) zu Tiefenbildern sind auch Disparitätsbilder vorstellbar. Wenn ein stereokalibriertes Kamerasetup vorhanden ist, kann aus zwei zueinander gehörigen Bildern die Disparitätsinformation berechnet werden. Über die Optimierung einer Energiegleichung wird ein (möglichst) dichtes Disparitätsbild erzeugt (z.B. mittels Semi-global Matching, vgl. HIRSCHMÜLLER 2005). Abhängig vom Kamerasetup ist z.B. die Disparität für Objekte, die sich nahe an der Kamera befinden, größer als für solche, welche weiter entfernt sind. Die Disparität liefert damit einen Anhaltspunkt für die Entfernung bzw. Tiefe von Objekten, ist jedoch i.d.R. mit relativ starkem Rauschen behaftet und weist Lücken auf, die z.B. aufgrund von Verdeckungen entstehen. Ein Beispiel aus dem Cityscapes-Datensatz zeigt Bild 37.

Disparitätsdaten können, wie Tiefenbilder, mit geringem Aufwand als zusätzlicher Eingangskanal für das neuronale Netz zur semantischen Segmentierung einbezogen werden. Erkenntnisse aus den Versuchen mit der Disparität werden in Kap. 3.3.1 dargestellt.

3.1.2 Algorithmus zur Erstellung von Tiefenbildern

Der folgende Algorithmus setzt ein voll kalibriertes Messfahrzeug voraus. Hierbei müssen alle verwendeten Sensoren eindeutig referenziert sein. Das bedeutet, jeder Sensor ist mathematisch durch eine Translation, eine Rotation und ggf. eine Abbildungsbeschreibung definiert. Mit dieser Grundvoraussetzung können die einzelnen Messsensoren im Fahrzeugkoordinatensystem oder im globalen Koordinatensystem das gleiche Objekt an der theoretisch gleichen Stelle abbilden.

Die zur Berechnung verwendeten Sensoren sind zum einen ein 2D-Laserscanner und zum anderen bildgebende Flächensensoren (Kameras).

Die Daten des Laserscanners werden durch die gemessene Trajektorie des Positionierungssystems, der Kalibrierung/Boresight Alignment des Sensors und den gemessenen Rohdaten zu einer dreidimensionalen Punktwolke verortet. Diese Punktwolke ist Grundvoraussetzung für die folgenden Schritte. Die dreidimensionalen Punktwolken liegen im globalen Koordinatensystem ETRS89 Zone 32N oder 33N mit einer WGS84 Höhe vor.

Im zweiten Schritt werden die zur Berechnung verwendet Kameras eindeutig mit dem Positionierungssystem verortet, sodass eine mathematische Beschreibung des Sensors im übergeordneten Koordinatensystem erfolgen kann. Wenn keine radiometrischen Kameras am Messsystem vorhanden sind, können auch virtuelle Kameras in einem festgelegten Abstand definiert werden. Bei diesem Ansatz kann allerdings nur auf die Intensität des Laserscanners und die Tiefeninformation zurückgegriffen werden.

$$x' = x'_0 \frac{r_{11}(X+X_0)+r_{21}(Y+Y_0)+r_{31}(Z+Z_0)}{r_{13}(X+X_0)+r_{23}(Y+Y_0)+r_{33}(Z+Z_0)} + dx' \P$$

$$y' = y'_0 \frac{r_{12}(X+X_0)+r_{22}(Y+Y_0)+r_{32}(Z+Z_0)}{r_{13}(X+X_0)+r_{23}(Y+Y_0)+r_{33}(Z+Z_0)} + dy' \P$$

Über die Kollinearitätsgleichung und die Verzeichnungsparameter analog BROWN 1971 können zu jedem Pixel die dreidimensionalen Objektstrahlen berechnet werden. Diese beschreiben den Verlauf eines Lichtstrahls durch das Objektiv in den Objektraum. Jeder Pixel ist hiermit verzeichnungsfrei mathematisch beschreibbar.

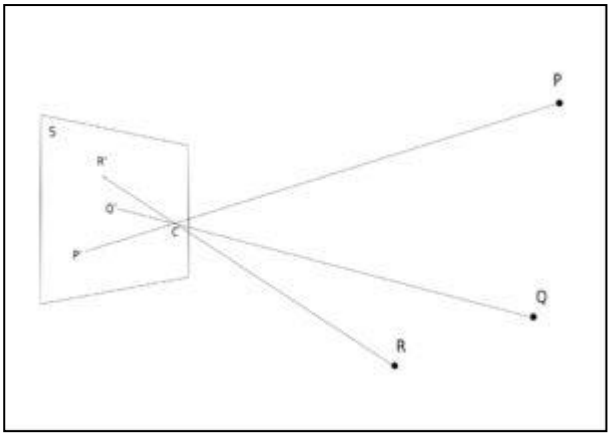


Bild 25: Rückprojektion von Objektpunkten auf einen Sensorebene.

Durch die Verschneidung von den einzelnen Objektstrahlen von jedem Pixel der Kamera lassen sich jetzt minimale Entfernungen berechnen. Diese Verschneidung wird über die PCL Bibliothek vorgenommen. Hierzu wird die Punktwolke an der Stelle der Kameraposition eingegrenzt, sodass nur noch relevante Teile der Punktwolke zur Berechnung herangezogen werden. Die Erzeugung des Tiefenbildes erfolgt analog SUNG 1991. Mit Hilfe eines Octree werden die einzelnen Bildstrahlen im Tiefenbild registriert. Als Ergebnis wird eine Bildmatrix mit realen Abständen in Bezug zur Punktwolke berechnet.

Hierbei zeigte sich, dass noch Anpassungen vorgenommen werden müssen. Zum einen darf die Filterung nur minimale Distanzen zur Kamera verwenden und zum anderen gibt es derzeit noch Defizite beim Interpolieren von Lücken in der Punktwolke.

Zur Speicherung der Daten wird ein spezielles HDR-Grafikdateiformat verwendet. Dieses Format wurde 1999 als offenes Format von der US-amerikanischen Firma Industrial Light & Magic entwickelt und später mit einer OpenSource Lizenz herausgegeben. Vorteil dieses Formates ist die Speicherung von Gleitkommazahlen pro Pixel. Zum Auslesen/Speicherung/Umwandeln wurde eine spezielle C++ Klasse auf Grundlage von OPEN-CV entwickelt, sodass die Austauschbarkeit und Wiederverwendbarkeit gewährleistet ist.

3.1.3 Nutzung der RGB-Bilddaten

Obwohl Algorithmen zur Objekterkennung in Punktwolken existieren, werden, sofern Bilddaten verfügbar sind, oft Methoden der Bildverarbeitung bevorzugt. Dies liegt unter anderen in deren Komplexität, aber auch in der Tatsache begründet, dass die Information in Bildern „dicht“ (Dense) vorliegt. Vor allem für die Erkennung geometrisch ähnlicher Objekte, wie z.B. Ampeln, Beleuchtungsanlagen oder Masten von Verkehrsschildern, kann dies für eine zuverlässige Erkennung erforderlich sein.

Wie in „Strategie 1“ ausgearbeitet, wird der zentrale Algorithmus für die Objekterkennung im weiteren Projektverlauf ein auf Deep Learning basierender Ansatz sein, welcher als Eingangsdaten die RGB-Bilddaten, verwenden wird, um eine semantische Segmentierung der Bildinformationen zu berechnen.

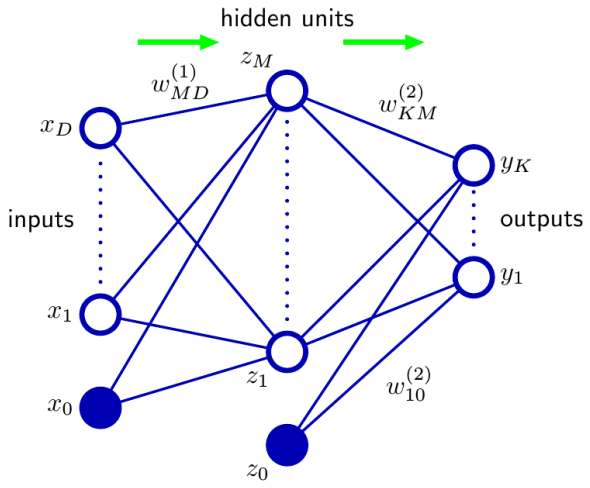
Deep Learning unterscheidet sich von den „klassischen“ Methoden der Objekterkennung vor allem dadurch, dass die zu erkennenden Objekte (oder Objektklassen) nicht vom Entwickler durch einen geeigneten Satz manuell definierter und implementierter Merkmale beschrieben werden. Die Merkmalsrepräsentation eines Objektes wird vom neuronalen Netz dadurch „gelernt“, dass eine ausreichende Menge manuell annotierter Trainingsdaten prozessiert wird.

Die Funktionsweise eines solchen Netzes wird im Folgenden näher beschrieben. Vgl. hierzu auch die grundlegende Einführung der Begriffe künstliches neuronales Netz, Deep Learning, Convolutional Neural Network (ConvNet), Objekterkennung und semantische Segmentierung in Kap. 2.2.4.

3.1.4 Hintergrund: Das Multilayer Perceptron

Die meisten aktuellen Ansätze des Deep Learning basieren auf einer Netzarchitektur, die als Multilayer Perceptron (MLP) bekannt wurde (RUMELHART et al. 1986, BISHOP 2006). Gleichzeitig wurde die sog. Backpropagation als mathematisches Hilfsmittel für den eigentlichen Lernvorgang (das Training) vorgestellt.

In Bild 26 ist die Architektur eines einfachen MLP dargestellt.



Quelle: BISHOP 2006, S. 228

Bild 26: Multilayer Perceptron mit Eingabe- und Ausgabeschicht sowie einer zusätzlichen versteckten Schicht (Hidden Unit). Die Neuronen einer Schicht sind als nicht ausgefüllte blaue Kreise dargestellt, die vollständig eingefärbten Neuronen sind die sog. Bias-Neuronen, deren Wert konstant ist und nicht von den Aktivierungen in vorhergehenden Schichten abhängt. Die Gewichte an den Verbindungen (Links) werden im Training gelernt.

Jede Schicht besteht aus einer Menge von Neuronen. Die Anzahl der Eingabeneuronen wird passend zu den Eingangsdaten gewählt, die Anzahl der Ausgabeneuronen passend zum gewünschten Ergebnis; sie entspricht also z.B. der Anzahl der zu erkennenden Objektklassen. Zwischen Ein- und Ausgabeschicht befindet sich eine beliebige Zahl versteckter Schichten (Hidden Units). Die Schichten sind im Bild vertikal angeordnet, die grünen Pfeile symbolisieren die Richtung, in der die Information in einem Vorwärtsdurchgang (Forward Pass, Feed-forward oder Forward Propagation), also dem „normalen“ Anwendungsfall, durch das Netz propagiert wird.

Jedes Neuron ist mit allen Neuronen aus der vorhergehenden Schicht verbunden. Die Aktivierung eines Neurons wird somit aus allen Aktivierungswerten in der vorhergehenden Schicht bestimmt. Diese werden mit den Gewichten der einzelnen Verbindungen (Links) zu diesem Neuron multipliziert, die während des Trainings des Netzes gelernt wurden. Zusätzlich gibt es in jeder Schicht ein sog. Bias-Neuron, dessen Wert konstant ist und nicht von den Aktivierungen in der vorhergehenden Schicht abhängt.

Der Aktivierungswert eines Neurons a_j in der ersten Schicht wird aus den Eingangsdaten durch folgende Formel bestimmt:

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$$

Es wurde ein zusätzliches Eingangsneuron x_0 eingeführt, damit die Summenformel für jede Schicht in dieser einfachen Weise ausgedrückt werden kann; es hat den konstanten Aktivierungswert 1. Die hochgestellte (1) bezeichnet die Tiefe der versteckten Schicht im Netzwerk, das Gewicht w_{ji} bezieht sich auf die Verbindung zwischen der i-ten Eingangsvariable und dem j-ten Neuron in Schicht 1.

Als nächstes wird jeder Aktivierungswert durch eine nichtlineare Aktivierungsfunktion h (Activation Function) transformiert:

$$z_j = h(a_j)$$

Die Wahl der Aktivierungsfunktion hängt von der Anwendung ab und stellt einen der Hyperparameter dar, welche die Performanz des neuronalen Netzes beeinflussen. Typische Aktivierungsfunktionen für Neuronen in verstecken Schichten sind z.B. Rectified Linear Unit (ReLU), logistische Sigmoidfunktion (Logistic Sigmoid Function) oder Tangens Hyperbolicus (tanh). Die Aktivierungsfunktion muss i.d.R. in vorher festgelegten Bereichen differenzierbar sein, damit während des Trainings die Ableitung bzw. der Gradient bestimmt werden kann (s.u.).

Bild 27 zeigt eine ReLU-Aktivierungsfunktion, sowie eine Variante davon, die Softplus Function genannt wird.

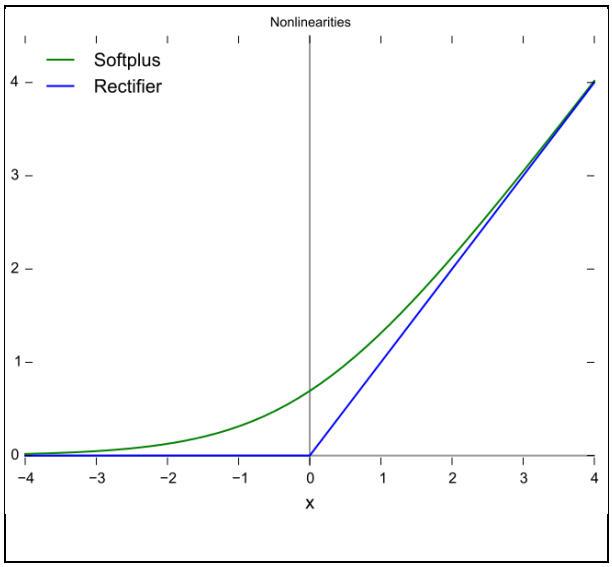


Bild 27: Beispielhafte Darstellung der Aktivierungsfunktion Rectified Linear Unit (auch als ReLU oder Rectifier bezeichnet), sowie einer als Softplus Function bezeichneten Variante.

Während die Aktivierungen mit der ReLU beliebig hohe (positive) Werte annehmen können, beschränkt die in Bild 28 dargestellte Logistic Sigmoid Function den Wertebereich auf das Intervall (0,1).

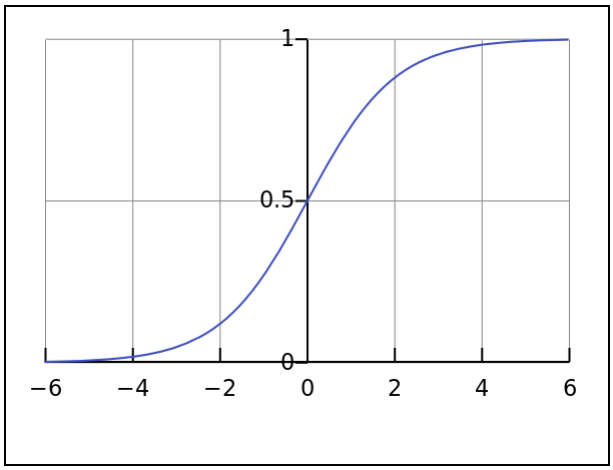


Bild 28: Darstellung der Logistic Sigmoid Aktivierungsfunktion.

Die Aktivierungen in nachfolgenden versteckten Schichten werden analog berechnet.

Für die Ausgabeschicht wird die Aktivierungsfunktion wieder je nach Anwendung ausgewählt: bei einer Regressionsaufgabe entspricht der Ausgabewert der Aktivierung der Neuronen, die Aktivierungsfunktion ist hier die Identität. Bei Klassifizierungsaufgaben wird i.d.R. eine Logistic Sigmoid Function oder für Multiklassenprobleme eine Softmax-Aktivierungsfunktion verwendet.

Zusammengefasst berechnen sich die Ausgabewerte aus den Eingabewerten für das MLP in diesem Beispiel mit der Formel:

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

Das σ steht für die Aktivierungsfunktion der Ausgangsschicht.

Obwohl das MLP von seinem Aufbau her einfach erscheint, lassen sich prinzipiell viele Aufgaben damit lösen. So kann jede begrenzte kontinuierliche Funktion mit beliebiger Genauigkeit durch ein MLP mit lediglich einer versteckten Schicht approximiert werden (CYBENKO 1989).

Ausgehend von diesem einfachen Netzaufbau wurde eine Vielzahl von Netzarchitekturen für verschiedene Anwendungen entwickelt. Für die Verarbeitung von Bilddaten sind die sog. Convolutional Neural Networks besonders geeignet.

3.1.5 Convolutional Neural Networks

Die klassische Architektur eines voll verbundenen Neuronalen Netzes auf ein Bild anzuwenden, indem jedes Pixel einem eigenen Eingangsneuron zugeordnet wird, stößt auf verschiedene Probleme. Problematisch ist insbesondere die hohe Anzahl an dafür benötigten Neuronen bzw. Parametern, um jeden Intensitätswert der Bildmatrix mit allen Neuronen der jeweils nächsten Schicht zu verbinden.

Außerdem werden durch die Fixierung der Zuordnung jeden Pixels zu einem „eigenen" Eingangsneuron Informationen entsprechend ihrer Position auf dem Bild unterschiedlich verarbeitet. Es ist jedoch für die Verarbeitung von Bildern wichtig, die Lokalität der enthaltenen Informationen zu beachten, sodass Bildecken miteinander weniger in Verbindung stehen als direkt benachbarte Pixel. Somit sollten Auswertungen anhand von Methoden erfolgen, welche über das gesamte Bild gleichermaßen arbeiten.

Ein Prinzip aus der Bildverarbeitung, welches die beschriebene Problematik löst, sind Filter. Diese lassen sich außerdem gut in die Architektur von neuronalen Netzen einfügen. Ein Filter besteht aus dem Produkt zweier Funktionen, was auch als Faltung (Convolution) bezeichnet wird.

Um einen gefilterten Pixelwert zu berechnen, wird das lokale Umfeld eines Pixels durchlaufen (zwei Schleifenfunktionen) und eine gewichtete Summe aus den dortigen Intensitätswerten gebildet. Dieses Umfeld kann unterschiedlich groß sein. Ein einfaches Beispiel, das auch bei Neuronalen Netzen häufig Verwendung findet, ist ein 3x3-Filter. Dieser würde demnach aus neun Gewichten (plus einem Bias- bzw. Schwellwert) bestehen, mit denen die entsprechenden Nachbarwerte multipliziert werden. Diese Gewichte lassen sich während des Lernprozesses (Training) anpassen; die variablen Filter erlauben es, mit einer geringen Anzahl an Parametern das gesamte Bild zu durchlaufen (Weight Sharing).

Als Ergebnis lässt sich das Vorkommen bestimmter Merkmale bzw. die Stärke ihrer Ausprägung im Bild in Form eines neuen „Bildes" herausfiltern. Diese werden auch als Merkmalskarten (Feature Maps) bezeichnet. In einem Farbbild (RGB-Kanäle) oder in Bildern mit mehr Kanälen werden Gewichte bzw. Filter für jeden Farbkanal benötigt.

Verschiedene gefilterte Bilder lassen sich zu einem Bild zusammenfassen, welches ähnlich zu einem RGB-Bild wieder aus mehreren (oder vielen) Kanälen besteht. Die Filter werden dann parallel angewandt. Genauso kann dies jedoch sequenziell geschehen, also weitere Filter auf das gefilterte Bild angewandt werden. Die Fähigkeit der Informationsgewinnung steigt entsprechend der Tiefe des Netzwerks und der Anzahl der Filter.

3.1.6 Semantische Segmentierung

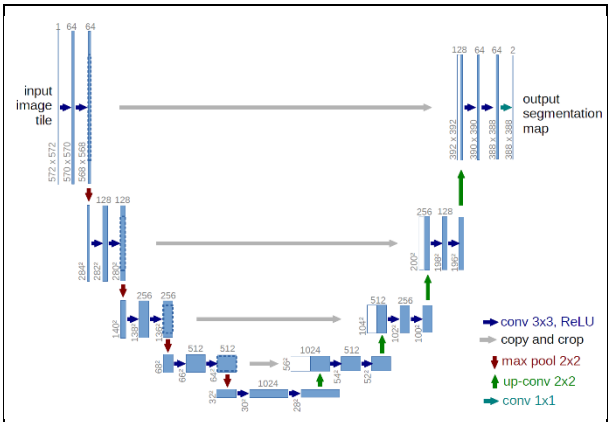
Convolutional Neural Networks (ConvNets) zur Analyse von Bilddaten eignen sich nicht nur für die Klassifizierung eines Gesamtbildes (eine Klasse pro Bild), sondern auch für eine pixelweise Klassifizierung. Diese wird auch als semantische Segmentierung bezeichnet.

Vgl. hierzu die Ausführungen zur semantischen Segmentierung von Straßenszenen aus Bilddaten in Kap. 2.2.3.6 und 2.2.3.7.

3.1.7 Erste Versuche mit einer Netzarchitektur

Die Filter im Neuronalen Netz werden nun so angeordnet, dass Informationen aus Bildern extrahiert und Pixel bzw. Regionen im Bild klassifiziert werden können. Diese Anordnung und die Art und Weise der zusätzlichen Verarbeitungsschritte im Netz (z.B. Nichtlinearitäten wie die ReLU) werden als Netzarchitektur bezeichnet.

Für dieses Forschungsprojekt wurde zunächst auf die U-Net-Architektur zurückgegriffen (RONNEBERGER et al. 2015). Das U-Net besteht aus einem ersten Teil, welcher unter fortlaufender Auflösungsreduzierung eine Klassifikation von Bildbereichen berechnet, und einem zweiten Teil, welcher die Information über die Klassifizierung wieder im Ursprungsbild lokalisiert, also den entsprechenden Pixeln zuordnet. Auf diesen zweiteiligen, U-förmigen Aufbau, wie in Bild 29 dargestellt, bezieht sich auch die Namensgebung. Die Auflösungen, welche im Bild in vertikaler Schrift angegeben sind, wurden für die Daten, wie sie für die Straßenbestandsobjekte vorliegen, angepasst.



Quelle: RONNEBERGER et al. 2015

Bild 29: Architektur des U-Net: Im linken Teil erfolgt die Klassifikation von Bildinhalten, wobei die Lokalisierung im Bild durch die Auflösungsreduzierung weitgehend verlorengeht. Im rechten Teil wird die Lokalisierung durch die Up-Convolutions und die Informationen aus dem linken Teil mit der Klassifikation verknüpft.

Je nach Anzahl der Kanäle der Eingabedaten wird die erste Schicht im U-Net beim Lernprozess automatisch angepasst. Der Einfachheit halber ist in Bild 29 lediglich ein Kanal dargestellt. Je mehr Klassen als Ausgabe detektiert werden sollen, desto mehr Kanäle benötigt die letzte Schicht des Netzes, welche vor dem Lernprozess definiert werden muss, nämlich ein Kanal je Klasse.

Alle Filter (mit einer Ausnahme) werden in der Größe 3x3 angewandt, in Verbindung mit einer ReLU als Aktivierungsfunktion. Das Bild wird in jeder Schicht des Netzes in zwei Ebenen nacheinander gefiltert und anschließend in einem MaxPooling-Schritt zusammengefasst. Dabei wird der Maximalwert von je 2x2 Pixel weitergegeben, womit die Seitenlänge des Bildes halbiert wird. In der nächsten Schicht des Netzwerks wird dadurch ein größerer Bildbereich analysiert. Das MaxPooling bewirkt außerdem eine Fokussierung auf die wesentlichen im Bild vorhandenen Merkmale.

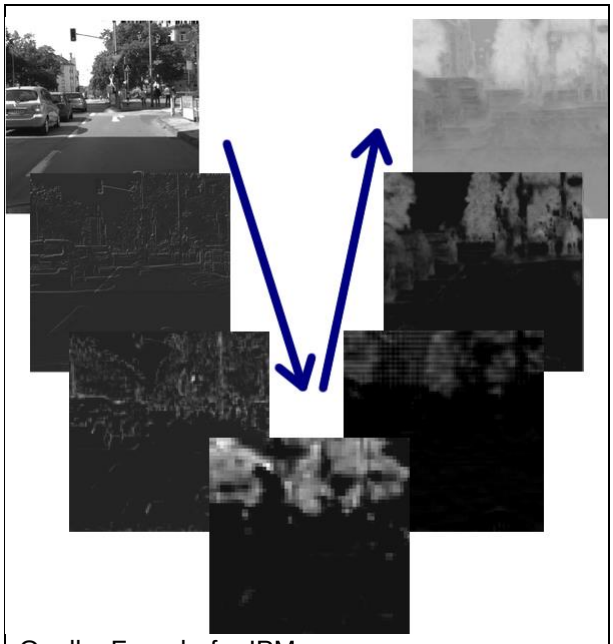
Filterung und Zusammenfassung (MaxPooling) werden vier Mal wiederholt, bis ein „Bild“ (bzw. eine Feature Map) entsteht, in dem immer 16 Pixel des Eingabebildes zusammenfasst sind. Je Stufe werden, beginnend mit 64 Filtern, immer doppelt so viele Filter angewandt, bis auf der untersten Ebene 1024 Merkmalskanäle vorhanden sind.

In der zweiten Hälfte des U-Nets sind die Stufen ähnlich angeordnet, nur, dass anstatt der Zusammenfassung von Pixeln jedes Pixel nun auf 2x2 Pixel „projiziert“ wird (mit jeweils individuellen Gewichten). Durch diese Up-Convolution-Layer wird die Seitenlänge somit jeweils wieder verdoppelt. Zusätzlich werden nach dieser Schicht die Kanäle aus der entsprechend gegenüberliegenden ersten Hälfte des U-Nets „angehängt“ (graue Pfeile). Dadurch werden Klasseninformationen wieder entsprechenden Pixeln zugeordnet.

Am Ausgang des Netzes kommt, als einzige Ausnahme, ein 1x1 Filter zum Einsatz. Die dort ausgegebene Anzahl an Kanälen entspricht der Anzahl an zu erkennenden Klassen.

Entsprechend der Symmetrie des Netzes sollte die Bildeingabegröße der Ausgabegröße entsprechen. Jedoch verursacht jeder Filter einen Offset, da durch die 3x3 Filter jeweils immer ein Pixel am Rand „verloren“ geht, da dort sonst der Bildrand beim Filtern überschritten werden würde. Dies verursacht in den Feature Maps der untersten Ebene des Netzes große Verluste, da die Auflösung hier nur noch sehr gering ist.

Ein Beispiel für die Klassifikation von Pixeln, welche zur Klasse „Vegetation“ gehören, ist Bild 30 zu entnehmen.



Quelle: Fraunhofer IPM

Bild 30: Darstellung einzelner Feature Maps für die Klasse „Vegetation“ bei der Auswertung eines Bildes aus dem Cityscapes-Datensatz.

Die ersten Filter des Netzes werten Merkmale wie z.B. Kanten aus, während in tieferen Schichten die Informationen aus immer größeren Bildbereichen in eine Wahrscheinlichkeit für eine bestimmte Klasse überführt wird (zur Verdeutlichung dargestellt als Helligkeitswert). Im rechten Teil ist zu sehen, wie die reduzierte Auflösung durch Up-Convolutions schrittweise wieder erhöht wird.

Im Netzwerk werden 18 Convolutional Layer verwendet, welche aus insgesamt fast 6000 Filtern bestehen. Zusätzlich werden weitere 1000 Filter für die Up-Convolutions angewandt. Insgesamt verfügt das Modell, welches durch das Netz repräsentiert wird, damit über ca. 31 Millionen Parameter bzw. Gewichte. Zur Speicherung der Parameter werden bei 32-bit je Gewicht ca. 124 MB Speicherplatz beansprucht.

Die final verwendete Netzarchitektur wird in Kap. 5.3.5 beschrieben.

3.1.8 Lernen durch Backpropagation

Im Gegensatz zu den klassischen Methoden der Objekterkennung werden die Merkmale, die ein Objekt beschreiben, nicht vom Programmierer vorgegeben, sondern anhand von Trainingsdaten gelernt und durch die Filter des Convolutional Network repräsentiert. Den Parametern der Filter entsprechen im Netzwerk die Gewichte an den Verbindungen zwischen den Neuronen. Bei einer Filtergröße von 3x3 Pixeln wird der entsprechende Filter durch die Gewichte der neun von diesem Filter ausgehenden Neuronen-Verbindungen definiert.

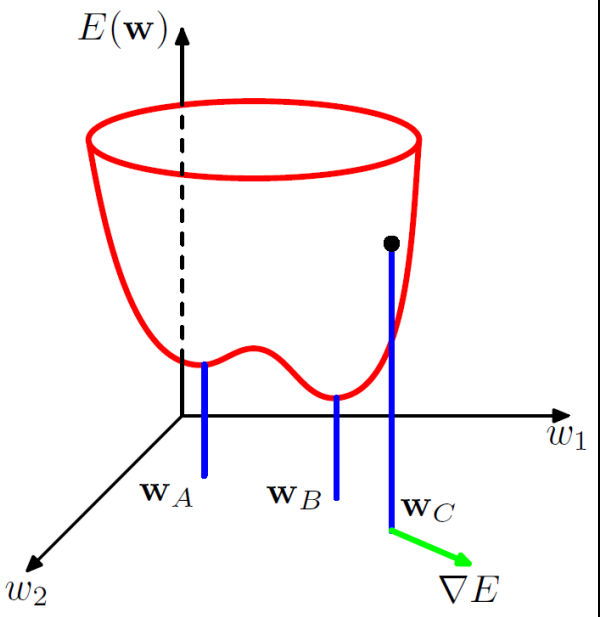
Netze für Deep Learning bauen hierbei aufgrund ihrer vergleichsweise „tiefen“ Architektur (hohe Anzahl an Schichten und Neuronen) und durch die hierarchische Abfolge von Filteroperationen und die sukzessive Vergrößerung des betrachteten Bildbereichs, besonders beschreibungskräftige und gleichzeitig flexible Merkmale auf.

Die Merkmale bzw. Filter werden beim Training so bestimmt, dass bei erneuter Prozessierung der Trainings- oder Testdaten ein möglichst kleiner Restfehler in der Objekterkennung erreicht wird. Einen Lernalgorithmus derart zu trainieren, wird auch als Supervised Learning bezeichnet.

Für das Training wird i.d.R. eine Least-Squares-Fehlerfunktion E definiert, die in Abhängigkeit von allen Gewichten **w** des Netzwerks und allen „richtigen“ Ergebnissen aus den N Trainings-Samples $\mathbf{t_n}$ eine Aussage darüber erlaubt, wie zuverlässig die Ergebnisse **y** für alle Datenpunkte $\mathbf{x_n}$ ausfallen:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2$$

Diese Fehlerfunktion hängt von den Gewichten **w** ab; somit kann der Fehler iterativ minimiert werden, indem die Ableitungen nach allen Gewichten des Netzes über alle hierarchischen Schichten des Netzes hindurch bestimmt werden und anschließend alle Gewichte in negative Richtung des Gradienten verändert werden. Eine prinzipielle Darstellung für zwei Gewichte ist in Bild 31 zu sehen.



Quelle: BISHOP 2006, S. 236

Bild 31: Geometrische Darstellung des Fehlers E(**w**) (Loss Function) als Funktion zweier Gewichte w_1 und w_2. $\mathbf{w}_A$ ist ein lokales Minimum, $\mathbf{w}_B$ das globale Minimum. Die Information aus dem Gradient ∇E wird dazu verwendet, die Gewichte nach und nach so zu verändern, dass man sich von Punkt $\mathbf{w}_C$ aus einem Minimum annähert.

Geeignete Parameter für das Training zu finden, ist nicht trivial. Es ist darauf zu achten, dass man nicht in lokalen Minima der Fehlerfunktion endet. Eine ungeeignete Wahl der Schrittgröße beim Gradientenabstieg kann dazu führen, dass die Minimierung nicht konvergiert. Eine zu kleine Schrittgröße wiederum kann zu einer inakzeptablen Verlängerung des ohnehin rechenaufwändigen Trainingsprozesses führen. Das sukzessive Berechnen einer Folge von Ableitungen nach der Kettenregel, wie bei tiefen Netzen erforderlich, kann zu numerischen Problemen wie z.B. dem sog. Vanishing Gradient führen. Für das Training von neuronalen Netzen wurden deshalb verschiedene Varianten des Gradientenabstiegsverfahrens (Gradient Descent) entwickelt (s. Kap. 3.1.9).

Für einen Lernschritt wird bei dieser Art des Trainings der Fehler nach dem Vorwärtsdurchlauf des Netzes für eine bestimmte Menge an Trainingsdaten berechnet. Die daraus abgeleitete Information über die nötige Anpassung der Parameter wird, ausgehend von der Ausgabeschicht, „rückwärts“ durch die Schichten

des Netzes zurückpropagiert, weshalb man beim Training auch von „Error Backpropagation“ spricht.

Beim Lernen kann die Minimierung des Restfehlers zunächst „einseitig“ ausfallen, d.h. das Netzwerk optimiert nicht die Erkennung aller Objekte gleichzeitig, sondern lernt zunächst die Objekte optimal zu erkennen, mit denen es den Restfehler besonders schnell minimieren kann. Dies ist eine mathematische Eigenart des Lernvorgangs, da die Veränderung der Gewichte stets dem steilsten Gradient nachfolgt. Das Caffe-Framework erlaubt es jedoch, den Einfluss einer Objektklasse auf den Restfehler höher einzustellen, womit das Training für einzelne Klassen beschleunigt werden kann.

3.1.9 Varianten des Gradientenabstiegs

Ziel des Lernprozesses ist es, den Fehler, also die Abweichung zwischen Vorgabe und momentaner Ausgabe, zu minimieren. Dies geschieht über die Anpassung der einzelnen Netzgewichte. Anhand der Ableitung der Fehlerfunktion nach den Netzparametern kann durch deren Steigung das Minimum der Fehlerfunktion angestrebt werden.

Für große Mengen an Trainingsdaten kann es erforderlich sein, die Anpassung der Netzparameter nicht erst nach Prozessierung des gesamten Trainingsdatensatzes, sondern für Teilmengen davon vorzunehmen. Dieses Vorgehen wird als Stochastic Gradient Descent bezeichnet, die Submengen der Trainingsdaten als Batches oder Mini-Batches. Für die Verarbeitung von Bilddaten sind selbst große Mengen an Arbeitsspeicher (z.B. die 12 GB einer TitanX-GPU) nicht ausreichend, um alle Trainingsdaten gemeinsam zu prozessieren.

Ein elementarer Ansatz zum Anpassen der Gewichte ist die Gradient Descent-Methode (RUMELHART et al. 1986, BISHOP 2006). Darin werden die Gewichte $\boldsymbol{w}$ entsprechend einer Lernrate η und dem Gradienten ∇E_n iterativ angepasst:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

Um die Robustheit und Geschwindigkeit des Lernprozesses zu steigern, wurden verschiedene Methoden entwickelt. Ein erster Schritt ist die Verwendung eines Momentums (Impuls), mit welchem vorige Aktualisierungen „gemerkt“ werden und die Suche somit weniger im lokalen Minima stecken bleibt.

Eine andere Erweiterung ist die individuelle Anpassung der Lernrate je Netzparameter. Hier erhalten Aktualisierungen, welche nur sehr selten vorkommen, einen größeren Einfluss als bei ständig aktualisierten Parametern. Dieses Vorgehen gehört zu den als adaptiv bezeichneten Methoden. ADAM (Adaptive Moment Estimation) ist eine Variante davon, welche sich als sehr erfolgreich bei ConvNets erweist (KINGMA; BA 2014).

Das Netzwerk für die Straßenbestandsobjekte wurde zunächst mit einem einfachen stochastischen Gradientenabstieg trainiert, dann auf ADAM umgestellt, was den Lernprozess, wenn auch nicht wesentlich, beschleunigte.

Lernrate, Momentum und weitere Einstellmöglichkeiten gehören zu den Hyperparametern, welche vor dem Training für das Netz festgelegt werden müssen.

3.1.10 Anmerkungen zu den Hyperparametern

Als Hyperparameter werden solche Parameter bezeichnet, die nicht unmittelbar zu dem Modell gehören, das der eigentlichen Objekterkennung zugrunde liegt. Beim Deep Learning sind dies z.B. die Anzahl der Layer, die Dimensionen der Filtermasken, die Auswahl an Aktivierungsfunktionen und weiteren Nichtlinearitäten, wie z.B. die MaxPooling-Operationen.

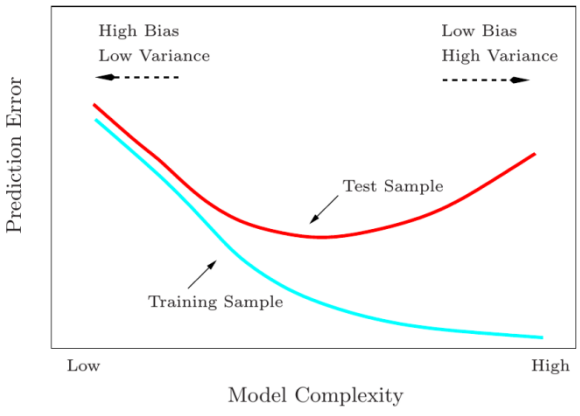
Die Wahl der Hyperparameter erfolgt meist anhand von Erfahrungswerten und ist eine nicht-triviale Aufgabe, da es sehr viele Variationsmöglichkeiten gibt, die nur schwer zu evaluieren und miteinander zu vergleichen sind. Vgl. hierzu auch unten die Ausführungen zur Komplexität und zur Gefahr des Under- oder Overfittings.

Die wesentlichen Hyperparameter, wie die oben beschriebenen Filterdimensionen, wurden aus der U-Net-Implementierung übernommen (RONNEBERGER et al. 2015) und wo nötig für unsere Aufgabenstellung angepasst. Die Werte dieser Parameter werden beim verwendeten Deep Learning-Framework Caffe in Textdateien abgelegt oder alternativ direkt im Quellcode gesetzt.

3.1.11 Komplexität und Under- oder Overfitting

Die gewählte Netzarchitektur, speziell die Anzahl der Schichten, Neuronen, Filtergrößen und Gewichte, bestimmt die Komplexität des Modells, welches durch das Netz repräsentiert wird. Je komplexer das Modell, desto mehr Variation in den Eingangsdaten kann „verkraftet“ werden. Andererseits steigt mit der Komplexität die Anfälligkeit für Overfitting. Dies bedeutet, dass das Netz Dinge, die sehr ähnlich sind wie das, was es

bereits beim Training „gesehen“ hat, sehr gut klassifizieren kann, aber zunehmend schlechter wird, wenn „Unbekannte“ Eingangsdaten (z.B. die Testdaten) bearbeitet werden. Man spricht dann davon, dass der Lernalgorithmus schlecht generalisiert. Dieser Zusammenhang wird in Bild 32 schematisch dargestellt.



Quelle: FRIEDMANN et al. 2009, S. 38

Bild 32: Darstellung des prinzipiellen Zusammenhangs zwischen der Komplexität des Modells (bzw. Netzes) und der Anfälligkeit für Overfitting.

3.1.12 Trainingsdaten und Data Augmentation

Um ein Netz zu trainieren, Objekte zuverlässig zu erkennen, die in ihrer Erscheinung stark variieren, werden sehr viele Trainingsdaten benötigt. Man denke z.B. daran, Menschen in verschiedenen Posen, unterschiedlich gekleidet, von vorne oder hinten betrachtet, mit verschiedenen Hautfarben, in verschiedenen Altersstufen, bei verschiedenen Beleuchtungen, in verschiedener Entfernung, von anderen Gegenständen teilweise verdeckt, zu erkennen. Die verschiedenen zu erwartenden Variationsmöglichkeiten müssen in den Trainingsdaten vorhanden sein, damit das Netz geeignete Merkmale lernen kann, die es erlauben, alle verschiedenen Erscheinungen gut zu erkennen.

Erfahrungswerte legen nahe, dass für Objekte, welche in ihrer Erscheinung relativ wenig variieren, für jede zu erkennende Klasse ca. 50-100 Trainingsbilder ausreichen. Für viele Objekte im Straßenumfeld ist diese Voraussetzung gegeben. Auch sind auf jedem unserer Bilder i.d.R. mehrere Klassen gleichzeitig zu sehen, allerdings oft weiter entfernt und daher sehr klein. Eine genaue Aussage über den Umfang der benötigten Trainingsdaten zu treffen ist schwierig; nicht zuletzt, da der „Lernbedarf“ für jede Klasse unterschiedlich hoch ist.

Um mit dieser relativ niedrigen Zahl von Bildern auszukommen, kann ggf. eine Technik eingesetzt werden, die als Data Augmentation bezeichnet wird. Diese erlaubt es, aus einem Trainingsbild durch gezielte Variation bestimmter Parameter einen ganzen Satz von Trainingsdaten zu erzeugen.

Gängige Transformationen für die Augmentierungen, die entsprechend parametrisiert und dann einzeln oder gemeinsam angewendet werden können, umfassen:

- Skalierung
- Spiegelung
- Ausschnitte bilden: zufällige Auswahl von Teilbildern; dabei werden Objekte auch teilweise beschnitten
- Subtraktion des Mittelwertes

Das auf dem Caffe-Framework basierende U-Net nutzt weitere Möglichkeiten der Augmentierung, darunter:

- Verschiedene Verzeichnungen
- Rotation
- Farbton (Value Augmentation)

Ein Beispiel für die Veränderung von Farbtönen im Training zeigt Bild 33.

Quelle: links: Cityscapes-Datensatz (CORDTS et al. 2016), rechts: Bearbeitung Fraunhofer IPM

Bild 33: Mit Data Augmentation können Trainingsbilder „mehrfach“ verwendet werden, indem z.B. unterschiedliche Lichtverhältnisse bzw. Farbwerte simuliert werden.

Für Objekte im Straßenraum macht es Sinn, nicht alle Operationen auszuwählen: eine wellenförmige Verzeichnung wird hier im Allgemeinen nicht relevant sein, sondern könnte im Gegenteil für „Verwirrung“ sorgen und die Erkennungsgenauigkeit negativ beeinflussen.

Die Wirksamkeit von Data Augmentation konnte während der vergangenen Projektphase bereits demonstriert werden. Die Erkennungsleistung des neuronalen Netzes, das mit Cityscapes-Daten trainiert wurde, konnte auf den Bilddaten von der Lehmann + Partner GmbH gesteigert werden, indem die Value Augmentation für das Training

aktiviert wurde. Die unterschiedlichen Farbeigenschaften der verschiedenen Kameras haben das Netz danach weniger vor Herausforderungen gestellt.

Zur weiteren Unterstützung des Trainings ist es außerdem denkbar, „synthetische“ Datensätze zu generieren. Hierzu werden Instanzen von Objekten zu Bildern bzw. Collagen montiert. Es muss allerdings gegeben sein, dass die Anordnung einer realistischen Szene entspricht, d.h. dass z.B. die Straße auch hier niemals im Himmel verortet sein darf. Zusätzlich erhält man bei dieser Vorgehensweise unmittelbar die korrekte Annotierung als Ground Truth; die aufwändige manuelle Auswertung entfällt oder wird zumindest erleichtert.

Dieses Vorgehen kann auch dann interessant sein, wenn bestimmte Klassen in den Trainingsdaten nur selten vorkommen und deshalb nicht ausreichend genau gelernt werden können.

3.2 Frühzeitige Versuche mit dem neuronalen Netz

Um das neuronale Netz zu trainieren, werden manuell annotierte Trainingsdaten benötigt. Dies bedeutet, dass eine größere Menge an Bilddaten pixelgenau mit Klassenlabeln versehen werden muss. Nur so ist es möglich, dass das neuronale Netzwerk ausreichend robuste Merkmale für die Klassifizierung neuer, unbekannter Daten lernen kann.

Da dieser aufwändige Prozess für die Messfahrten der Lehmann + Partner GmbH im Projektverlauf erst zu einem späten Zeitpunkt abgeschlossen werden kann, wurde für die ersten Implementierungsschritte auf den für Forschungszwecke freigegebenen Cityscapes-Datensatz zurückgegriffen (CORDTS et al. 2016).

Somit können bereits zu einem frühen Zeitpunkt im Projekt erste Aussagen zur Erkennungsleistung eines neuronalen Netzes auf den vom IRIS5 aufgezeichneten Daten getroffen werden, wenn auch mit der Beschränkung auf die Menge an Objektklassen, die im Cityscapes-Datensatz definiert ist.

3.2.1 Training mit dem Cityscapes-Datensatz

Die Cityscapes-Daten unterscheiden sich von den Messdaten des IRIS5 in der Auflösung der RGB-Bilder, sowie in der Perspektive der beiden Kameras (die dort auf Höhe des Rückspiegels eines „normalen“ PKW angebracht wurden). Punktwolken stehen hier nicht zur Verfügung, dafür jedoch u.a. die Trajektorie der Messfahrten, sowie aus den Stereoaufnahmen berechnete Disparity-Bilder.

Im Zuge der manuellen Auswertung der IRIS5-Daten in Arbeitspaket AP500 wird das Training des neuronalen Netzwerkes dann mit den entsprechenden Daten fortgeführt.

Mit Hinblick auf die Verbesserung der Algorithmen ist es jedoch durchaus von Interesse, zu analysieren, wie gut das mit Cityscapes-Daten trainierte Netz die aus anderer Perspektive und mit anderen Kameras aufgenommenen Daten der IRIS5 auswerten kann. Für erste Erkenntnisse diesbezüglich vgl. Kap. 3.3.

Die Annotierung des Cityscapes-Datensatzes enthält bereits eine Reihe der Straßenbestandsobjekte, die für das vorliegende Forschungsprojekt relevant sind. Dies sind:

- Straße, Gehweg
- Parkmöglichkeit
- Bauwerk (Gebäude), Mauer, Zaun
- Schutzplanke (nur sehr selten abgebildet)
- Brücke, Tunnel
- Mast/Pfosten, Mastgruppe
- Schild (Verkehrszeichen)
- Lichtsignalanlage
- Vegetation, Terrain/Bewuchs

Daneben gibt es weitere statische und dynamische Objekte aus Straßenszenen, u.a. Schiene, Person, Auto/Fahrzeug, Fahrrad, LKW, Bus oder Himmel. „Undefinierte“ Objekte werden als Boden oder schlicht „dynamisch“ oder „statisch“ gelabelt.

Ein Beispiel für ein manuell annotiertes Trainingsbild ist in Bild 34 zu sehen: Jede Klasse ist in einer Farbe codiert dargestellt, z.B. rot für PKW, gelb für Schilder, grün für Vegetation und hellblau für Bauwerke.

Quelle: Cityscapes-Datensatz (CORDTS et al. 2016)

Bild 34: Trainingsdaten aus dem Cityscapes-Datensatz: Jede Klasse ist hier zur Veranschaulichung in einer Farbe dargestellt; die manuell erstellte semantische Segmentierung ist mit dem Originalbild überblendet.

Da fehlende Klassen, speziell z.B. die Fahrbahnmarkierungen, ohnehin in der manuellen Annotierung in AP500 für die Daten des IRIS5 generiert werden sollen, erfolgte für die ersten Implementierungen eine Beschränkung auf die im Cityscapes-Datensatz vorhanden Objektklassen.

Es zeigt sich, dass bei der Annotierung von Daten sehr sorgfältig vorgegangen werden muss, da immer wieder uneinheitliche, widersprüchliche oder schlicht falsche Label auftreten. Solche Effekte können die Erkennungsgenauigkeit negativ beeinflussen, vor allem bei Objektklassen, welche vergleichsweise selten vorkommen. Einige diesbezügliche Anmerkungen zum Cityscapes-Datensatz sind in Kap. 3.2.2 zusammengefasst.

3.2.2 Anmerkungen zur Annotierung im Cityscapes-Datensatz

Bei der Arbeit mit dem Cityscapes-Datensatz (CORDTS et al. 2016) sind verschiedene Aspekte aufgefallen. Manche können als positive Beispiele gewertet werden, andere erscheinen stellenweise inkonsequent umgesetzt und können das Training eines Netzes negativ beeinflussen.

Diese Erkenntnisse sollten für die manuelle Annotierung von Messdaten des IRIS5 im Rahmen von AP500 durch die Lehmann + Partner GmbH berücksichtigt werden. Einige werden im Folgenden dargestellt, zusammen mit weiteren allgemeinen Anmerkungen zum Datensatz.

Die am häufigsten vorkommende Klasse ist Road (Straße). Private Seitenstraßen werden jedoch immer wieder nicht als Straße gelabelt. Die Oberfläche bzw. Textur der Straßen kann sehr unterschiedlich sein. So gibt es neben verschiedenen Asphaltsorten, Maserungen und Farbtönen auch Innerstadtbereiche mit Kopfsteinpflaster. Je nach Verkehrszone werden gepflasterte Fahrbahnen nicht mehr als Straße gelabelt, sondern zählen als „Ground‘ (Boden). Straßenmarkierungen werden zur Straße gerechnet und nicht separat gelabelt. Die Randbereiche der Straße verlaufen i.d.R. bis zur Unterkante des Bordsteines bei Bürgersteigen.

Die Klasse „Boden“ wird auch für Verkehrsinseln oder offene Plätze verwendet. Auch Bordsteine sind, wenn sie nicht an einen Bürgersteig grenzen, sondern beispielsweise an eine Grünfläche, so gelabelt. Die Klasse Boden dient allgemein als Sammelklasse im Straßenbereich, wenn Fahrbahn, Parkplatz, Bürgersteig oder Schienen nicht zutreffen.

Die Klasse Parking (Parkmöglichkeiten) beschreibt Regionen seitlich der Fahrbahn, in denen Parkplätze zur Verfügung stehen. In einigen Fällen heben sich diese Bereiche durch abweichende Texturierung gegenüber asphaltierter Straße ab. Es ist aber auch möglich, dass Parkzonen lediglich mit Markierungen angedeutet sind. Gelegentlich werden Parkbereiche aber auch „fälschlicherweise“ als Straße gelabelt.

Als Sidewalk (Gehweg) werden alle Bereiche markiert, die ausschließlich für Fußgänger gedacht sind. Die klassischen Bürgersteige an der Seite der meisten städtischen Straßen gehören dazu, aber auch Bereiche von Verkehrsinseln, welche für Personen gedacht sind. Es kann in einigen Städten vorkommen, dass breitere Fußgängerbereiche mit Straßencafés als Boden gelabelt werden. Wenn es erlaubt ist, dass Fahrzeuge auf dem Gehweg parken, bleibt dieser meist als Bürgersteig gelabelt und wird nicht als Parkmöglichkeit klassifiziert.

Ein weiterer Objekttyp zur Klassifizierung von Untergrund ist „Rail Track“ (Bahngleis/Schienen). In Städten mit Straßenbahnen werden so die Schienen bzw. der Straßenbahnbereich gelabelt. Bei Mitbenutzung durch den normalen Straßenverkehr bleibt der Bereich als Schienen gelabelt.

Straßenbahnen erhalten eine eigene Klasse „Train“. Dabei wird der Stromabnehmer auf dem Dach der Bahnen eingeschlossen, ohne den Hintergrund, der durch die Öffnungen hindurch sichtbar ist, zu unterscheiden. Pixel, welche als „Train“ klassifiziert wurden, kommen selten vor: im fein annotierten Cityscapes-Datensatz etwa gleich oft wie die zugehörigen Gleisstrecken mit jeweils nur ca. 0,2%.

Meist vorkommende Klasse an Fahrzeugen sind PKW mit einem Vorkommen von etwa 6% der Pixel. Spezielle Klassen für Fahrzeuge sind „Truck“ (LKW), „Bus“ (Bus), „Caravan“ (Wohnwagen) und

„Trailer“ (Anhänger). Diese Typen haben ein Pixelvorkommen im Datensatz von unter einem Prozent. Zusätzlich ist zu beachten, dass die Größe solcher Fahrzeugklassen aus vielen Pixeln je Instanz besteht, da sie sehr nahe an die Kameras herankommen. LKW werden inklusive ihres Anhängers gelabelt, die Klasse „Anhänger“ ist auf kleinere PKW-Anhänger beschränkt.

Zweirädrige Fahrzeuge werden in die Klassen „Motorcycle“ (Motorrad) und „Bicycle“ (Fahrrad) unterteilt. Das Vorkommen beider Typen liegt weit unter einem Prozent. Die Fahrer der Zweiräder werden in einer eigenen Klasse als „Rider“ (Fahrer) annotiert und kommen dadurch ebenfalls selten vor. Werden Fahrräder transportiert, z.B. an Wohnmobilen, so werden sie trotzdem separat gelabelt. Segways werden als Fahrräder markiert.

Ein spezieller Fahrzeugtyp ist die „Ego-vehicle“-Klasse. Da die Bilder des Datensatzes aus dem Fahrerraum eines PKW aufgenommen wurden, ist die Motorhaube des Messfahrzeuges auf den Bildern im unteren Bereich zu sehen. Da es auf allen Bildern zu sehen ist, ist das Vorkommen an Pixeln entsprechend hoch. Der relativ flache Winkel der Perspektive und die klare Lackierung der Motorhaube sorgen immer wieder für starke Spiegelungen der Umgebung.

Kommen Personen als Fußgänger bzw. ohne Fahrzeug vor, werden diese als „Person“ gekennzeichnet. Werbeplakate, welche Personen abbilden, fallen jedoch nicht unter diese Kategorie. Kleine Gegenstände wie Taschen zählen noch zu der jeweiligen Person, Kinderwägen oder hinterhergezogene Koffer jedoch nicht mehr. Letztere fallen unter die Kategorie „Dynamic“ (dynamisch), in der viele bewegliche Objekte zusammengefasst werden; weitere Beispiele sind Mülltonnen, Café-Stühle sowie Sonnenschirme. Beide Letzteren sind jedoch teilweise auch als „Static“ (statisch) gekennzeichnet, was bedeutet, dass diese Objekte fest an ihren Ort gebunden sind. Markante Beispiele hierfür sind große Blumentöpfe oder Verteilerkästen.

Zur Unterscheidung der Pflanzen und Grünbereiche gibt es die Klasse „Vegetation“. Hauptvertreter sind Bäume mit Stamm und Blättern. Aber auch Hecken und Büsche zählen dazu. Ist eine Blätterwand bzw. -dach nicht deckend (kann man also dahinterliegende Objekte erkennen), so wird dennoch das Label Vegetation über die volle Fläche angewandt. Auch Hohlräume zwischen Ästen werden zum Baum gezählt. Die Ränder um Blätter sind grob umrissen, weswegen rund um einen Baum i.d.R. ein gewisser „Offset“ mit zur Vegetation gelabelt wird. Eine weitere Klasse um die Natur zu klassifizieren ist „Terrain“ (Boden). Primär sind damit Rasenflächen gekennzeichnet, wie sie beispielsweise auf Grünstreifen vorkommen. Es kommt aber auch vor, dass angrenzende Bordsteine mit gelabelt werden.

Eine nicht zuletzt wegen der nahezu horizontalen Ausrichtung der Kamera recht häufig vorkommende Klasse ist „Sky“ (Himmel). Dazu zählen auch Wolken. Vereinzelt verlaufen Drähte und Versorgungsleitungen durch den Himmel, werden aber nicht gesondert gelabelt.

Als „Tunnel“ sind die Innenwände und Decken auch von ausgedehnteren Straßenunterführungen gekennzeichnet. „Brücken“ werden nur bei seitlicher Ansicht oder von unten gekennzeichnet; Bilder, welche auf einer Brücke aufgenommen wurden, markieren diese nicht als solche. Beide Typen kommen im Datensatz selten vor.

Um Objekte der Kategorie Bauwerke zu markieren, wird hautsächlich die Klasse Building verwendet. Gebäude aller Typen, von Wohnhäusern bis zu Geschäften, fallen darunter, womit diese Klasse einen hohen Anteil an Pixeln im Datensatz belegt. Einzelne Bauteile wie Fenster werden nicht weiter separiert. Lediglich aushängende Werbetafeln werden extra klassifiziert. Ebenfalls zu den Bauwerken zählen überdachte Bushaltestellen. Sind diese aus Glas oder bestehen Gebäude aus großen gläsernen Etagen, wird das Label Gebäude dennoch verwendet, auch wenn der Hintergrund durch das Objekt hindurch gut sichtbar ist.

Weitere Klassen für kleinere Konstruktionen sind Wall (Mauer) und Fence (Zaun). Letzterer ist meist als Gartenzaun, Brückengeländer oder Baustellenabsperrung zu finden. Die Hohlräume in der Zaunstruktur werden ebenfalls als Zaun gelabelt, auch wenn Objekte im Hintergrund gut erkennbar sind. Als Mauer oder Wand hingegen werden undurchlässige Flächen markiert.

Die Klasse Guard Rail (Schutzplanke) ist ein in diesem Datensatz selten vorkommender Klassentyp.

Eine wesentliche Klasse zur Deutung von Straßenszenen sind Schilder, welche in einer allgemeinen Klasse Traffic Signs vertreten sind. Darunter fallen alle Typen von Straßenschildern, bis hin zu Informationstafeln von Bundestrassen. Ebenfalls dazu zählen Leitpfosten und Leitbaken, sowie Schilder für Straßennamen. Die Rückseite von Verkehrsschildern ist jedoch mit der Klasse „Static“ markiert.

Schilder sind meist an einer Metallstange befestigt. Diese wird von einer eigenen Klasse „Pole“

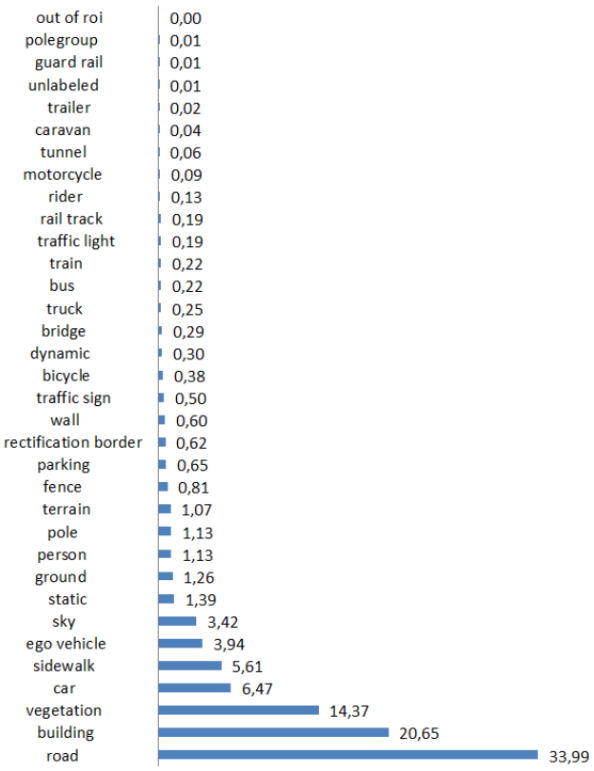
(Mast/Pfosten) repräsentiert. Darunter fallen z.B. auch Masten von Beleuchtungseinrichtungen. Das Vorkommen von Masten auf den Bildern ist hoch, dennoch ist die Pixelanzahl durch deren geringe Ausmaße relativ niedrig. Ein spezieller Klassentyp von Masten ist Polegroup (Mast-Gruppe), worunter Masten zusammengefasst sind, die nur schwierig als einzelne Instanzen zu erkennen sind.

Mit der Klasse Traffic Light werden Lichtsignalanlagen gelabelt. Die Einteilung erfolgt inklusive Rückseite und Lichtpunkt.

Wenn einem Pixel keine explizite Klasse zugeordnet werden kann oder soll, wie z.B. bei der groben Annotation für nicht klassifizierte Bereiche, wird das Pixel der Klasse Unlabeld (unmarkiert) zugeschlagen.

Zumindest im Cityscapes-Datensatz mit feiner (d.h. vollständiger) Klassifizierung aller Pixel treten gelegentlich auch fehlerhafte Zuordnungen auf. So werden Klassen von Objekten teilweise nicht richtig zugeordnet, wie beispielsweise die fehlerhafte Markierung einer Lichtsignalanlage als Bauwerk (Hannover0/48765). Aber auch Bauwerke können nicht erkannt und entsprechend zum Himmel zugeordnet werden (Bochum0/3245). Ein gröberer Schnitzer ist beispielsweise die Zuordnung eines kompletten Himmels zur Klasse der Schilder (Hannover0/23975). Unauffälliger ist etwa die Zuordnung eines Werbeschildes zur Klasse der Straßenschilder (Straßburg0/5912). Vermehrt sind im Datensatz Objekte uneinheitlich klassifiziert worden, wie bereits bei der Darstellung der einzelnen Klassen erwähnt. So werden beispielsweise Seitenstraßen teilweise als Boden und teilweise als Straße klassifiziert. In einem anderen Fall werden auf einigen Bildern Nummernschilder von Autos als Schilder gelabelt, obwohl diese sonst mit zum Auto-Objekt gezählt werden (Straßburg0/14743).

Den Anteil des Vorkommens der verschiedenen Klassen im Trainingsdatensatz zeigt Bild 35.



Quelle: Fraunhofer IPM

Bild 35: Häufigkeit des Vorkommens einzelner Klassen in den Trainingsdaten des Cityscapes-Datensatzes. Unterrepräsentierte Klassen können vom neuronalen Netz nur schwer „gelernt“ werden.

3.3 Zwischenergebnisse

3.3.1 Analyse von Testdaten aus dem Cityscapes-Datensatz

Bereits früh im Projektverlauf wurden verschiedene Versuche mit Implementierungen eines neuronalen Netzes unternommen. Als Trainingsdaten wurden ca. 3.000 Bilder aus dem Cityscapes-Trainingsdatensatz verwendet, die Evaluierung erfolgte mit Bildern aus dem Testdatensatz.

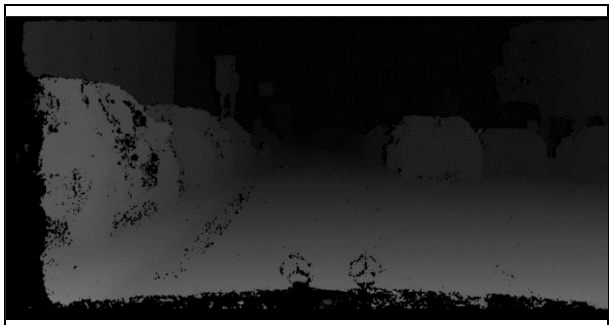
Zunächst wurden Training und Evaluierung ausschließlich mit RGB-Bilddaten von einer der beiden Kameras durchgeführt. Bild 36 zeigt ein exemplarisches Ergebnis.

Quelle: Ursprungsbild: CORDTS et al. 2016; Klassifizierungs-Overlay: Fraunhofer IPM

Bild 36: Semantische Segmentierung eines Bildes aus den Testdaten des Cityscapes-Datensatzes nach 200.000 Trainingsiterationen mit RGB-Bilddaten. Die pixelweise Klassifizierung wurde zur Veranschaulichung in Farben kodiert und dem Originalbild überlagert.

Es fällt auf, dass die Ergebnisse für die Klassen Gebäude, PKW, Straße und Vegetation in weiten Bereichen sehr akkurat ausfallen. Allerdings zeigen sich auch Unstimmigkeiten, vor allem im Nah- und Fernbereich. Dies könnte damit zusammenhängen, dass im Nahbereich die Auswirkungen der Perspektive am stärksten ausfallen, oder daran, dass abgeschnittene Objekte, wie der PKW am linken Bildrand, für das Netz schwieriger zu erkennen sind. Verkehrsschilder und Masten werden nicht vollständig klassifiziert.

Im Cityscapes-Datensatz stehen zusätzlich zu den RGB-Bildern auch Disparitätsbilder zur Verfügung (Bild 37), welche Informationen über die Entfernung von Objekten zur Kamera enthalten. Sie können damit analog zu Tiefenbildern interpretiert werden.



Quelle: CORDTS et al. 2016

Bild 37: Beispiel eines Disparity-Bildes aus dem Cityscapes-Datensatz, das zusätzlich zu dem RGB-Bild verwendet wurde. Die Einbeziehung dieser Information kann, ähnlich wie für Tiefenbilder zu erwarten, die Erkennung verschiedener Objektklassen erleichtern.

Die Nutzung dieser Daten als weiterer Eingangskanal für das neuronale Netz führte zu einer konsistenteren Klassifizierung im Nahbereich (Bild 38).

Quelle: Ursprungsbild: CORDTS et al. 2016; Klassifizierungs-Overlay: Fraunhofer IPM

Bild 38: Semantische Segmentierung eines Bildes aus den Testdaten des Cityscapes-Datensatzes nach 200.000 Trainingsiterationen mit RGB-Bilddaten und der Disparity als weiterem Eingangskanal. Die pixelweise Klassifizierung wurde zur Veranschaulichung in Farben kodiert und dem Originalbild überlagert.

Im Fernbereich werden nun die Verkehrsschilder und Masten besser erkannt, gleichzeitig steigt aber die Zahl falsch klassifizierter Pixel auf dem Gebäude. Die quantitative Analyse in Kap. 3.3.3 zeigt tatsächlich gerade für die eher „kleinen“ Objekte eine Verbesserung.

Dieses Experiment lässt wieder darauf schließen, dass die Verwendung von Tiefen- und/oder Disparitätsinformation für die Klassifizierung eine sinnvolle zukünftige Erweiterung zu den in diesem Forschungsprojekt implementierten Algorithmen darstellen kann.

3.3.2 Analyse von Bildern aus Testdaten von LP

Das mit dem Cityscapes-Datensatz trainierte neuronale Netz wurde nun auf Bilder angewendet, die mit dem IRIS5 aufgenommen wurden und für die zu diesem Zeitpunkt noch keine manuell annotierten Daten verfügbar waren.

Es ist zu erwarten, dass die Unterschiede bei der Datenaufnahme (Höhe und Ausrichtung der Kamera, Kameracharakteristik) die Objekterkennung deutlich erschweren. Erste Versuche bestätigten diese Vermutung. Eine Verbesserung brachte jedoch bereits die Verwendung der Data Augmentation für die Farbwerte beim Training des Netzes.

Bild 39 zeigt die Aufnahme einer Straßenszene durch IRIS5. Die erhöhte Kameraposition ist deutlich an der Perspektive sichtbar.

Quelle: Lehmann + Partner GmbH

Bild 39: Beispielbild vom Messfahrzeug IRIS5 LP

Die semantische Segmentierung dieses Bildes, mit dem Originalbild überlagert, ist in Bild 40 zu sehen.

Quelle: LP, Fraunhofer IPM

Bild 40: Überlagerung von semantischer Segmentierung und Originalbild mit farblicher Kodierung der erkannten Objekte, z.B. gelb für Verkehrsschilder, rot für PKW, hellgrün für Vegetation und hellblau für Gebäude. Erkennungsleistung des neuronalen Netzes nach 303.000 Trainingsiterationen mit Cityscapes-Trainingsdaten bei Anwendung auf Daten des IRIS5 der Lehmann + Partner GmbH.

Interessant ist, dass, obwohl das Netz mit Cityscapes-Daten trainiert wurde, die Erkennungsleistung für dieses Bild des IRIS5 nur wenig negativ beeinflusst wird. Allerdings wurde das Netz für diese Auswertung länger trainiert (303.000 vs. 200.000 Iterationen).

Die Auswertung anderer Bilder zeigt jedoch auch, dass die Erkennung auf den IRIS5-Daten noch nicht optimal arbeitet (Bild 41), wobei Unstimmigkeiten gerade im Nahbereich auch bei der Auswertung von Cityscapes-Daten auftreten.

Quelle: LP, Fraunhofer IPM

Bild 41: Überlagerung von semantischer Segmentierung und Originalbild mit farblicher Kodierung der Klassen, z.B. rot für PKW, grün für Vegetation, blau für Straße. Erkennungsleistung des neuronalen Netzes nach 303.000 Trainingsiterationen mit Cityscapes-Trainingsdaten bei Anwendung auf Daten des IRIS5 der Lehmann + Partner GmbH.

Der starke Schattenwurf auf der Häuserwand sowie die Position der Kamera fast senkrecht über dem Randstein stellen das Netz hier vor zusätzliche Herausforderungen. Ebenso scheint das hohe Aufkommen an Vegetation die Interpretation der weiter entfernten Bereiche zu erschweren.

3.3.3 Quantitative Interpretation

Die Berechnung der Genauigkeit, mit der Objekte bei der semantischen Segmentierung erkannt wurden, bezieht sich auf die Anzahl der korrekt klassifizierten Pixel. Genauigkeitsberechnungen wurden anhand der Ergebnisse auf Testbildern berechnet, also auf Daten, welche das Netzwerk zuvor (d.h. während des Trainings) noch nicht „gesehen“ hat.

Als Formel zur Berechnung der Genauigkeit des Ergebnisses verwenden wir die sog. „Intersection over Union“-Metrik (vgl. z.B. EVERINGHAM et al. 2014):

$$\frac{true\ positive}{true\ positive + false\ positive + false\ negative}$$

Ein Nachteil dieser Berechnungsweise ist allerdings, dass Objekte „bevorzugt“ werden, die (im Schnitt) eine große Fläche im Bild einnehmen.

Es ergibt sich außerdem durch Pixel am Randbereich von Objekten immer eine Unsicherheit, da bei solchen Pixeln oft nicht klar ist, ob sie noch zum Objekt oder bereits zu einem

anderen Objekt oder zum Hintergrund gehören. Wenn Objekte (teil-) transparent sind, tritt dieses Problem nochmals verstärkt in Erscheinung. So ist z.B. nicht klar, ab welcher Größe andere Objekte innerhalb einer Öffnung zwischen den Ästen eines Baumes separat identifiziert werden sollten. Bild 42 zeigt dieses Problem exemplarisch.

Quelle: CORDTS et al. 2016, Auswertung durch Fraunhofer IPM

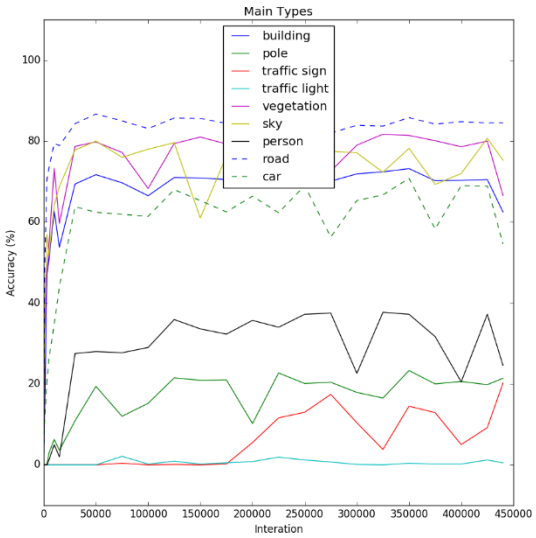
Bild 42: Die Genauigkeit der semantischen Segmentierung hängt z.B. in Randbereichen von Bäumen stark von der Art und Weise der manuellen Annotierung der Trainingsdaten ab: in den grünen Bereichen stimmen das Klassifikationsergebnis des neuronalen Netzes und die manuelle Annotierung überein, rote Bereiche markieren Unstimmigkeiten.

Eine Herausforderung für die Erkennung von Objekten aus Bilddaten stellen auch Bereiche dar, die reflektierend sind. Ein in einem Schaufenster reflektierter Baum wird vom neuronalen Netz oft als solcher klassifiziert , obwohl die Fläche als Teil eines Gebäudes erkannt werden sollte. Dasselbe gilt in abgeschwächtem Maß für Objekte, welche auf spiegelnden Autos oder nassen Flächen reflektiert werden. Anzumerken ist jedoch, dass solche reflektierenden Oberflächen für optische Messungen im Allgemeinen problematisch sind und deshalb z.B. auch in der Punktwolke wahrscheinlich nicht, oder nur unzuverlässig, erfasst sind.

Die Berechnung der Genauigkeit der Ergebnisse hängt deshalb auch signifikant davon ab, auf welche Weise – und mit welcher Sorgfalt – die manuelle Auswertung der Trainings- und Testdaten durchgeführt wurde. Eine weitere Auffälligkeit ist, dass die Erkennungsgenauigkeit für Klassen, die besonders häufig vorkommen, bzw. in einer besonders hohen Zahl an Pixeln sichtbar sind, überdurchschnittlich hoch ist. Vgl. hierzu auch Kap. 3.2.2 mit weiteren Bemerkungen zum Cityscapes-Datensatz.

Auffällig ist ebenfalls, dass Klassen, die häufig vorkommen, besser – oder zumindest schneller – gelernt werden. Dieser Effekt könnte jedoch abnehmen, wenn das Training des Netzes noch über die 400.000 Iterationen hinaus verlängert wird, auf die es aus Zeitgründen bisher beschränkt wurde.

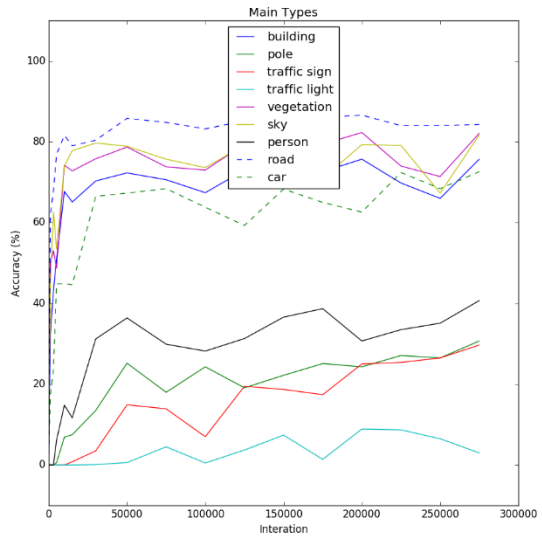
Einen ersten Eindruck davon, welche Genauigkeiten bisher erreicht werden können, vermittelt Bild 43 für die Auswertung mit dem fein-gelabelten Teil des Cityscapes-Datensatzes nach bis zu 450.000 Trainingsiterationen. Diese Anzahl an Iterationen war nach ca. vier Tagen erreicht; durch noch längeres Training kann die Genauigkeit voraussichtlich nochmals gesteigert werden.



Quelle: Fraunhofer IPM

Bild 43: Erkennungsgenauigkeit auf Cityscapes-Testdaten nach Training auf fein gelabelten Daten mit Data Augmentation und ADAM-Solver nach knapp 450.000 Trainingsiterationen.

Mit der Nutzung der Disparity-Bilder als weiterem Eingangskanal ergibt sich eine signifikante Erhöhung der Erkennungsgenauigkeit, wie in Bild 44 zu sehen ist.



Quelle: Fraunhofer IPM

Bild 44: Verbesserung der Erkennungsgenauigkeit mit Nutzung der Disparity-Information als weiterem Eingangskanal auf fein gelabelten Cityscapes-Daten nach knapp 300.000 Trainingsiterationen.

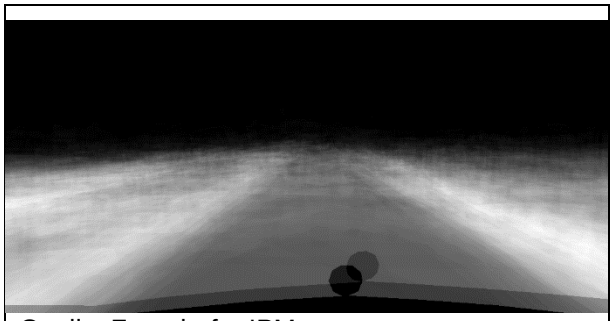
Die Verbesserung betrifft hier insbesondere die beiden für die Analyse von Straßenbestandsobjekten sehr interessanten Objekte Mast und Verkehrszeichen.

Um die Erkennungsgenauigkeit künftig weiter zu verbessern, stehen neben der Nutzung der Tiefeninformation verschiedene Ideen zur Verfügung, mit denen die im Forschungsprojekt implementierten Algorithmen ggf. erweitert werden könnten, z.B.:

- Nutzung weiterer Informationen aus der Punktwolke als zusätzliche Eingangskanäle für das neuronale Netz (z.B. Intensität oder Höhenwert)
- Berechnung von Disparity-Bildern, alternativ oder zusätzlich zu Tiefenkarten, aus den Bildern von zwei Frontkameras
- Einbeziehung der aktuellen Orientierung (Heading) des Messfahrzeugs zur Vorhersage der wahrscheinlichen Position von Objekten im Bild
- Gewichtung der Ausgabe des Netzes durch a-priori-Wissen vor der endgültigen Klassifizierung, z.B. über die Wahrscheinlichkeit des Vorkommens von bestimmten Objekten in bestimmten Bildbereichen (s.u.).

3.4 Gewichtung der Ergebnisse mit A-Priori-Wissen über die Position

Für einzelne Klassen wäre es denkbar, eine Gewichtung der Ergebnisse durch die wahrscheinliche Position im Bild vorzunehmen. So ist es z.B. eher unwahrscheinlich, einen Gehweg oberhalb einer bestimmten Position im Bild zu finden.



Quelle: Fraunhofer IPM

Bild 45: Wahrscheinlichkeit der Position von Gehwegen im Bild, dargestellt als Helligkeitswert. Auf der linken Seite ist die Position weniger eindeutig, da hier ggf. mehrere Fahrspuren zwischen Fahrzeug und Gehweg vorhanden sind. Die Information wurde aus der manuellen Annotierung von Trainingsdaten berechnet.

Aus den manuell annotierten Trainingsdaten lässt sich die wahrscheinliche Position von Objekten leicht bestimmen, indem pixelweise das Vorkommen einer Klasse summiert und dann durch die Zahl aller Trainingsbilder geteilt wird.

Die vom Netz berechneten Klassifikationsergebnisse können mit diesem a-priori-Wissen „korrigiert“ werden, z.B. durch pixelweise Multiplikation der Wahrscheinlichkeitswerte. Eine Wahrscheinlichkeitsverteilung wie in Bild 45 sollte hierfür jedoch geglättet werden. Ebenso sollten Toleranzschwellen implementiert werden, um in Randbereichen nicht durch zu restriktives Vorgehen eigentlich korrekte Klassifizierungen zu verwerfen. Dann könnten auf diese Weise Ausreißer reduziert werden, die z.B. durch die Spiegelung von Objekten entstehen.

Auch diese Idee zur Gewichtung der Ergebnisse stellt eine Möglichkeit dar, die im Forschungsprojekt implementierten Algorithmen künftig um weitere Funktionalität zu erweitern.

3.5 Weitere Auswertungsmöglichkeiten

Neben der Aussage darüber, wo ein bestimmtes Objekt im Bild zu finden ist, lassen sich aus den Ergebnissen der semantischen Segmentierung weitere Erkenntnisse gewinnen. Von Interesse ist z.B. die Frage, wie sicher das Ergebnis der Klassifizierung ist. Hierfür ist zunächst der vom Netz berechnete Wert pro Pixel für die wahrscheinlichste Klasse ausschlaggebend, wie er z.B. in Bild 46 als Helligkeitswert für die Klasse „Straße“ dargestellt ist.

Quelle: Fraunhofer IPM

Bild 46: Ergebnis einer semantischen Segmentierung für die Objektklasse „Straße“: die Wahrscheinlichkeit, dass ein Pixel zu dieser Klasse gehört, ist hier zur Verdeutlichung als Helligkeitswert dargestellt.

Daneben kann aber auch die Differenz zwischen der wahrscheinlichsten und der zweit-wahrscheinlichsten Klasse betrachtet werden: Eine große Differenz besagt, dass das Ergebnis relativ eindeutig ausgefallen ist. Wenn aufgrund von a-priori-Wissen das Auftreten eines Objekts in einem bestimmten Bereich unwahrscheinlich ist, kann dann bei einem „knappen“ Ergebnis die Wahl z.B. auf die zweitwahrscheinlichste Objektklasse fallen.

Abzuwägen wäre bei einer Implementierung diese Möglichkeit noch, ob das Ziel ist, False Positives oder False Negatives zu minimieren. False Positives sind Bereiche, die einer falschen Klasse zugeordnet wurden, z.B. dort, wo das KNN sich bei der Klassifizierung „unsicher“ war. Anhand der Wahrscheinlichkeitsverteilung bei der Ausgabe können solche Bereiche gefiltert werden. Dabei können jedoch False Negatives entstehen, wenn die Ergebnisse zu stark gefiltert werden, da ja auch unsichere Ergebnisse durchaus korrekt sein können. Dies gilt insbesondere dort, wo ungünstige Belichtungs- oder Kontrastverhältnisse anzutreffen sind, oder wo Objekte beschädigt oder durch Vandalismus in ihrer Erscheinung verändert sind.

3.6 Anmerkungen zu Infrastruktur und Rechenaufwand

Das Training eines komplexen neuronalen Netzes erfordert einen erheblichen Aufwand an Rechenzeit und Arbeitsspeicher. Demgegenüber ist die Bearbeitung von Bildern mit einem fertig trainierten Netz, also ein einfacher Forward Pass durch das Netzwerk, je nach Anwendung deutlich schneller möglich.

Für das Training des neuronalen Netzes verwenden wir eine leistungsstarke Workstation mit Intel Core i7-6700K-Prozessor (4 echte / 8 logische Kerne mit 4 GHz) mit 32 GB Arbeitsspeicher, die mit einer leistungsfähigen Grafikkarte ausgestattet ist. Diese verfügt über eine TITAN X-GPU von NVIDIA, welche 3072 CUDA-Rechenkerne und 12 GB Arbeitsspeicher besitzt. Ein solcher Rechner ist im Preissegment von deutlich unter 10.000 € zu beschaffen.

Das für das Deep Learning eingesetzte Caffe-Framework könnte für die hier gestellte Aufgabe ohne eine solche GPU nicht in vertretbarer Zeit trainiert werden. Selbst mit dieser Hardware wurden für ca. 400.000 Trainingsiterationen etwa vier bis fünf Tage benötigt.

Auf Softwareseite ist für das Caffe-Framework zusätzlich zu CUDA die Verwendung von cuDNN (NVIDIA CUDA Deep Neural Network Library) sinnvoll, um eine weitere Beschleunigung des Trainings erreichen zu können.

Caffe ist primär auf Linux ausgelegt, wurde jedoch in wesentlichen Teilen auf Windows portiert. Es steht unter der BSD 2-Clause-Lizenz. Die wichtigsten Abhängigkeiten für die Kompilierung des Caffe-Frameworks sind (s. http://caffe.berkeleyvision.org/installation.html):

- CUDA (für den GPU-Modus)
- BLAS via ATLAS, MKL, oder OpenBLAS
- Boost
- protobuf, glog, gflags, hdf5

Optional:

- OpenCV

- IO-Bibliotheken: lmdb, leveldb (leveldb benötigt snappy)
- cuDNN

Es gibt (optionale) Programmier-Interfaces für Python (Pycaffe) und Matlab (Matcaffe):

- Python und numpy sowie boost-provided boost.python
- MATLAB mit mex-Compiler

4 Definition von Teststrecken

Im Projekt wurden 850 km an Testdaten erfasst. Diese werden mit einem Messfahrzeug der Firma Lehmann + Partner erfasst. Hier kommen Laserscanner des Fraunhofer IPM zum Einsatz. Die Georeferenzierung erfolgt mit Hilfe eines Positionierungssystems. Das Positionierungssystem kombiniert GPS Messungen, eine Wegstreckenmessung und Messungen einer IMU (inertial measurment unit). Die IMU ist hierbei das Primärsystem und erfasst Beschleunigungsmessungen sowie Winkelunterschiede in allen drei Koordinatenachsen. Alle weiteren Messungen werden mit Hilfe eines Kalman-Filters bestmöglich in die Trajektorie eingerechnet. Eine weitere Genauigkeitssteigerung wird durch eine Post-Prozessierung erreicht. Hierbei wird minimal eine GPS-Basisstation oder virtuelle Referenzstation mit den Messdaten im Nachgang neu berechnet. Durch die Vorwärts- und Rückwärtsfilterung können viele GPS- und kreiseltypische Fehlereinflüsse und Driftverhalten eliminiert werden.

Besonders GPS-Abschattungen und Multipath-Effekte beinträchtigen die Positionierungsgenauigkeit erheblich. Die folgende Streckenauswahl sollte realitätsnahe Strecken beinhalten und auch messtechnisch problematische Strecken enthalten.

Charakteristik:

- Streckencharakteristik
- Straßenmöblierung
- Vegetation
- Verkehr
- Inner- bzw. Außerorts

Straßenverbindungsfunktion lt. Richtlinien für integrierte Netzgestaltung (*RIN*)

- Autobahnen - außerhalb und innerhalb bebauter Gebiete,
- Landstraßen - außerhalb bebauter Gebiete,
- anbaufreie Hauptverkehrsstraßen - im Vorfeld und innerhalb bebauter Gebiete, anbaufrei, Hauptverkehrsstraße,
- angebaute Hauptverkehrsstraßen - innerhalb bebauter Gebiete, angebaut, Hauptverkehrsstraße,
- Erschließungsstraßen - innerhalb bebauter Gebiete, angebaut, Erschließungsstraße.

Die Streckenauswahl kann unterteilt werden in verschiedene Charakteristiken. Zum einen sind dies die Ortsdurchfahrten, die Bundeslandstraßen, Kreisstraßen und Autobahnen. Die Strecken werden sich vorrangig auf Autobahnen und Landstraßen mit Ortsdurchfahrten beziehen.

Tendenziell ist festzuhalten, dass Autobahnabschnitte mit geringer Sichtabschattung der GPS- Satelliten (sky plot) die besten absoluten Genauigkeiten und Wiederholbarkeiten erreichen werden. Eine Ortsdurchfahrt mit dichter Bebauung wird eine geringere absolute Genauigkeit der Georeferenzierung erreichen.

Des Weiteren hat der Auftreffwinkel des Laserstrahls auf das Objekt einen Einfluss auf die Messgenauigkeit. Weiterhin ist zu beachten, dass es sich bei allen Laserscannern um polare Messsysteme handelt. Dies bedeutet, dass das Messsystem mindestens einen Winkel und eine Strecke (vgl. 2.1.6) misst. Hierdurch kommt es aufnahmebedingt mit wachsender Distanzmessung zu einer Dichteverkleinerung, mit welcher ein Objekt erfasst werden kann. Um alle möglichen Fehlereinflüsse abbilden, testen und auch ggf. trainieren zu können, ist die zu erfassende Strecke auf ~850 km angesetzt worden.

Vorschlag von Strecken, die zur Objekterkennung bzw. zu Trainingsdaten verwendet werden können:

Autobahnen:

- A4 (Gotha - Weimar) (120km)
- A4 (Magdala-Rüdersdorf) (120km)
- A6 (Kreuz Weinsberg - Kreuz Wiesloch/Rauenberg) (100 km)
- A24 (Pankow-Neuruppin) (140km)
- Stadtautobahn Berlin inkl. Zubringer (~240km)

  davon:

  - A10 – 20km
  - A100 - 88km
  - A114 - 34km
  - A115 - 26km

Bundesfernstraßen/Landstraßen:

- Blaues Netz Brandenburg (~580Km)
    - Ohne Ortsdurchfahrten

Tabelle 3 Streckenabschnitte im Netz Brandenburg.

Von	Bis	Straßen-name
Bad Liebenwerda	Elsterwerda	B101
Herzberg (Elster)	Trebbin	B101
Elsterwerda	Cottbus	B169
Cottbus	Spremberg	B97
Weißenberg	Eisenhüttenstadt	B112
Duben	Frankfurt (Oder)	B87
Herzberg (Elster)	Döbrichau	B87
Ludwigsfelde	Ludwigsfelde	B101N
Perleberg	Wittenberge	B189
Neuruppin	Frankfurt (Oder)	B167
Frankfurt (Oder)	Eisenhüttenstadt	B112
Fürstenwalde/Spree	Beeskow	B168
Neustrelitz	Gransee	B96
Gransee	Oranienburg	B96
Strassburg	Schwedt (Oder)	B166
Lawitz	Schlagsdorf	B112

- Sachsen Kreisstraßennetz (150km)
    - Mit Ortsdurchfahrten

Innerorts:

- Hauptstraßen Berlin 20km
- Hauptstraßen Dresden 30km

5 Auswertung der Testdaten

Vorliegend wurde, welche Schritte notwendig waren, um allgemeine Befahrungsdaten eines Mobile Mapping Fahrzeuges für das Training eines neuronalen Netzes und die anschließende Batch-Prozessierung von Mengendaten aufzubereiten. Hierbei ergaben sich einige Probleme, die im Folgenden erläutert werden.

5.1 Voraussetzungen zur Erstellung der Trainingsdaten für ein neuronales Netzwerk

5.1.1 Festlegung einer Ordnerstruktur

Zur Erstellung der Trainingsdaten waren eine Punktwolke und ein RGB-Bild erforderlich. Um die Daten zu prozessieren wurde eine einheitliche Ordnerstruktur festgelegt. Die übergeordnete sachliche und zeitliche Zuordnung einer Messung wurde wie folgt definiert:

#Projektname/Messfahrzeug/Messtag/Messung

Die Formatierung wurde wie folgt festgelegt:

#15111_Testdaten/IRIS5/2017010/012

Alle weiteren Programme verwendeten bzw. unterstützen diese Ordnerstruktur vollständig oder in Teilen.

Eine Messung wurde im Weiteren wie folgt definiert: Eine Messung besteht aus einer kontinuierlichen Aufnahme von Sensoren ohne Unterbrechungen. Hierbei war zu unterscheiden zwischen weggesteuerten und zeitgesteuerten Sensoren. Der Laserscanner ist hierbei ein zeitgesteuerter Sensor und nimmt kontinuierlich Messdaten ohne Berücksichtigung der Geschwindigkeit der Messplattform auf. Die Bildsensoren werden wegabhängig ausgelöst und liefern in einem festen Intervall Bilder in Abhängigkeit der Messplattformgeschwindigkeit.

Jeder Sensortyp wurde in einen Ordner der jeweiligen Messfahrt gespeichert. Hierbei wurde die Punktwolke in einen CPS- und die Bilddaten in einen CCD-Ordner gespeichert. Alle weiteten Zwischenergebnisse wurden in dergleichen Strukturtiefe abgelegt, wie zum Beispiel das Tiefenbild, die Labels und die extrahierten Objekte. Die enthaltenen Ordner wurden wie folgt benannt:

- CCD (Bilddateien)
- CPS (Punktwolke als Textdateien)
- DEPT (Tiefenbilder)
- LABELS (segmentierte Ergebnisse oder manuelle Trainingsannotierungen)
- OBJECTS/OBJEKTE (extrahierte Objekte in der Messfahrt)

5.1.2 Batchprozess zur Erstellung des Tiefenbildes

Das Tiefenbild wurde aus einem Punktwolkenabschnitt, den Orientierungs-parametern und den Kalibrierparametern berechnet. Die Berechnung soll in Zukunft in einem separaten Tool erfolgen. Der Arbeitsschritt wird z.Z. über Funktionen im Programm PhoML-Viewer realisiert. Hierzu wurden folgende Frameworks unter Windows implementiert: PCL 1.7.1 und OPENCV 2.4.10.

Wie bereits in Kap. 3.1.2 beschrieben, wurde der Algorithmus zur Berechnung des Tiefenbildes implementiert und automatisiert. Hierbei ergaben sich folgende Anpassungen in der Herangehens-weise. Der verwendete Octree aus der PCL-Bibliothek wurde auf 10x10 cm begrenzt. Dies hatte Einfluss auf die Abbildung der Größe der einzelnen Objekte und der Lückenfüllung im Tiefenbild. Weitere Ausführungen sind unter Kap 5.1.3 zu finden. Andererseits wurde aufgrund der Rechenleistung und der Auflösungsunterschiede zwischen Bild und Punktwolke nur jedes 5te Pixel im Ausgangsbild für eine Tiefenberechnung genutzt. Grundlage hierzu war die Erkenntnis, dass benachbarte Pixel in 70% der Fälle den gleichen Tiefenwert erhalten haben. Das bedeutete, dass der gleiche Punkt in der Punktwolke für die Berechnung verwendet wurde.

Die Batch-Prozessierung der Tiefenbilder erfolgte durch Einlesen der in Kap. 5.1.1 beschriebenen Ordnerstruktur. Hierbei wurden Kalibrierungsdaten und auch alle Bilddaten eingelesen. Das Programm diente auch als einfacher Viewer mit Testfunktionalitäten und Rückprojektionen von Objekten aus der Datenbank. Nach Auswahl des Unterpunktes „batch compute depth pictures“ wurden alle eingelesenen Daten in der Struktur verarbeitet. Hierbei wurde zu jedem Bild ein adäquates Tiefenbild erzeugt. Die beiden Bilder können zu Kontrollzwecken oder für Testmessungen abwechselnd in den Viewer geladen werden.

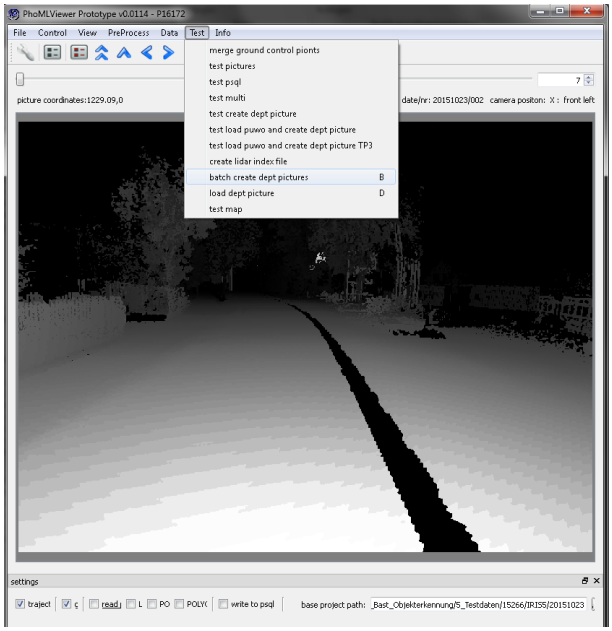


Bild 47: PhoML-Viewer mit Tiefenbilddarstellung und Auswahl der Batch-Prozessierung.

5.1.3 Genauigkeitsbetrachtung zum Tiefenbild

Durch die zuvor beschriebene Funktionalität im oben genannten Programm ließ sich die Qualität bzw. die Deckungsgleichheit des Tiefenbildes in Bezug zum RGB-Bild und zur Punktwolke realisieren. Hierbei zeigte es sich, dass Objekte, die in 3 bis 4 m Entfernung zum Fahrzeug erfasst wurden, eine ähnliche Auflösung in der Punktwolke wie im Tiefenbild aufweisen. Hier wurde das Objekt mit der Auflösung des Tiefenbildes realitätsnah erfasst. Die Rückrechnung in die Punktwolke ergab eine mittlere Oberflächenabweichung kleiner 1 cm. Dies entsprach in etwa der Genauigkeit des verwendeten Laserscanners. Weiter entfernte Objekte oder Objekte, die teilweise nur mit wenigen horizontalen oder vertikalen Punkten vom Laserscanner abgebildet wurden, wurden durch den verwendeten Algorithmus, den 10 cm großen Octree, im Tiefenbild vergrößert dargestellt. Zum einen verbesserte das die Messbarkeit und die Überlagerung mit dem RGB-Bild, andererseits ergab sich hieraus auch ein gewisser Lageversatz für schwach repräsentierte Objekte in der Punktwolke (Bild 48). Die Rückprojektion ergab eine Vergrößerung und einen Lageversatz, weil die erfasste Oberfläche des Objektes nur einen geringen Teil des Objektes wiederspiegelte. Bei Rückprojektion des Objektes in die Punktwolke wurde dies deutlich. Hiermit ergab sich generell für Objekte, die nicht hinreichend als Oberfläche repräsentiert sind, eine Lageabweichung der resultierenden Messung. Gleiches galt auch für die Punktwolke an sich. Ist eine Oberfläche nicht

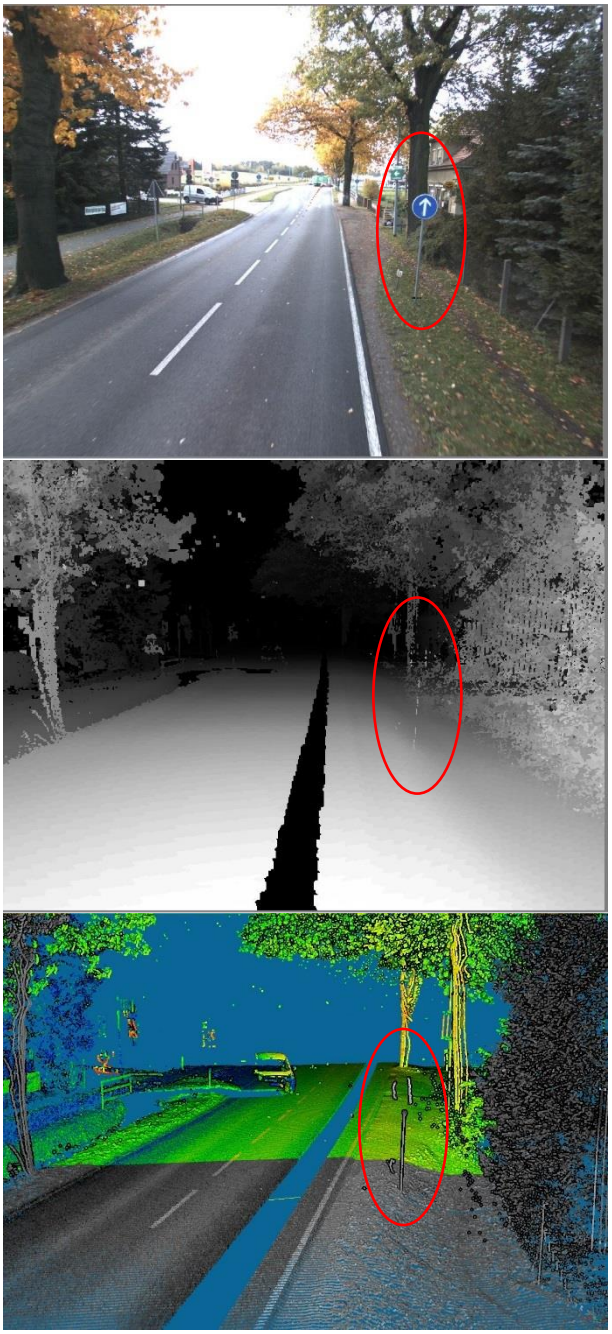


Bild 48: Darstellung einer schlechten Konditionierung der Oberfläche eines Objektes (Verkehrsschild).

hinreichend abgebildet, ergab sich zwangsläufig eine Fehlinterpretation der Lage des detektierten Objektes. Die im Projekt verwendeten Daten zeigten, dass dieser Fehler mit etwa der doppelten Punktdichte/Abstand abzuschätzen war. Bei hinreichender Punktdichte konnte dieser Fehler vernachlässigt werden.

Tabelle 4: Typische Standardabweichungen in den verwendeten Testdaten (Abgriff Trajektorien-Daten und Wiederholungsmessungen).

Absolute Genauigkeiten	**Standardabweichung (1Sigma) in mm**
Positionierungssystem	50 - 200
Laserscannerpunkt	60 - 260

Tabelle 5: Typische Standardabweichung in relativem Bezug auf der Sensorplattform.

Relative Genauigkeiten	**Standardabweichung (1Sigma) in mm**
Kamerakalibrierung	3 mm pro m (Abweichung des Bildstrahls ausgehend vom Kamerahauptpunkt)
Laserscannerpunkt	1-10

5.2 Labeln der Bilddaten

Grundlage für das Neuronale Netz war die Erstellung der Trainingsdaten. Dieses erfolgte durch die Markierung eines Objekts im Bild. Im Laufe der Projektbearbeitung stellte sich heraus, dass es nicht ausreichend war, nur Teile von einem Bild zu markieren, sondern immer das komplette Bild objektweise zu markieren ist. Hieraus ergaben sich dann neue Anforderungen an das Werkzeug/Software, mit denen das ganze Bild lückenlos markiert werden musste. Folgende Anforderungen waren umzusetzen: Jedes Pixel sollte eine Zuordnung erhalten oder einer „Default"-Klasse zugeordnet werden. Überlappende Bereiche von Objektklassen sollten immer gleich abgetrennt werden. Zwischen benachbarten Objekten dürfen keine Lücken entstehen. Eine Softwareneu-entwicklung wurde aufgrund des Projektzeitplanes verworfen. Die Open Source Software zum Cityscapes-Datensatz enthielt einen großen Teil der Funktionen, die sich durch die oben genannten Forderungen ergaben. Aus diesem Grund wurde das Tool für dieses Forschungsprojekt angepasst. In dieser Software war es möglich, Bereiche zu digitalisieren und einer Objektklasse zuzuordnen. Das Tool erlaubte es weiterhin, jedes markierte Objekt auf verschiedene Layer, in den Vorder- oder Hintergrund, zu schieben. Die sich daraus ergebenden Verschnitte konnte dieses Tool bereits berechnen, so dass Objektgrenzen durch Verschnitt zweier Polygone lückenfrei möglich waren. Somit konnten kleine Objekte über größeren Objekten digitalisiert werden. Die Verwendung dieser Layer-Funktionalität erwies sich als die effektivste Methode, ein komplettes Messbild in einzelne Objektklassen zu unterteilen.

Die Genauigkeit der zu erfassenden Objekte wurde ständig mit Fraunhofer IPM abgestimmt.

Die Software musste für das vorliegende Forschungsprojekt angepasst werden. Dieses Programm ist in Python geschrieben und benötigte zum Ausführen einen Interpreter für die API analog Java. Um die Änderungen nachvollziehbar zu machen und ggf. Updates vom Hauptprojekt mit einfließen zu lassen, wurde eine Versionsverwaltung mittels Git aufgesetzt. Weiterhin ermöglichte es die eingebundene Software, jederzeit den aktuellen Stand zu verteilen oder bei Problemen auch eine Version zurück zu gehen. Hinzugekommen waren eine Installations-anleitung und ein Erfassungsleitfaden für das aktuelle Projekt. Somit wurde sichergestellt, dass jeder Mitarbeiter/in mit der aktuellen Programmversion und mit dem neusten Erfassungsleitfaden arbeitet.

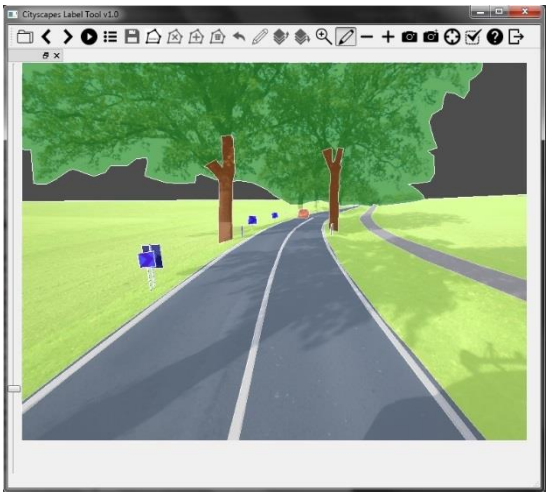


Bild 49: Angepasstes Cityscapes Programm mit aktuell gelabelter Landstraße.

5.2.1 Aufgetretene Probleme beim Labeln und Genauigkeitsanforderungen

Die Erfassung der Labels erfolgt grafisch. Hierbei zeigte es sich, dass es trotz der layerweisen Schnittfunktionalität zu erheblichen Lücken in der Erfassung eines Bildes kam. Sukzessive wurden Reihenfolgen (Hintergrund, Vordergrund) für die Digitalisierung festgelegt. Dann wurden Pufferbereiche eingeführt, sodass es bei den

Überlappungsbereichen theoretisch nicht mehr zu Lücken kommen konnte.

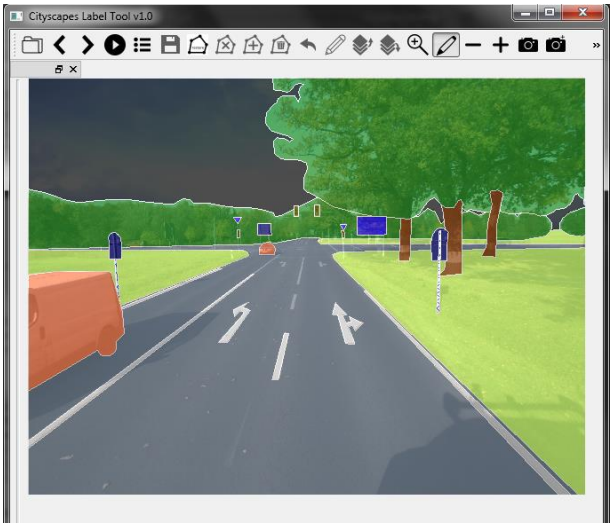


Bild 50: Gelabeltes Bild mit falscher Layerreihenfolge: Markierung Mitte fehlt.

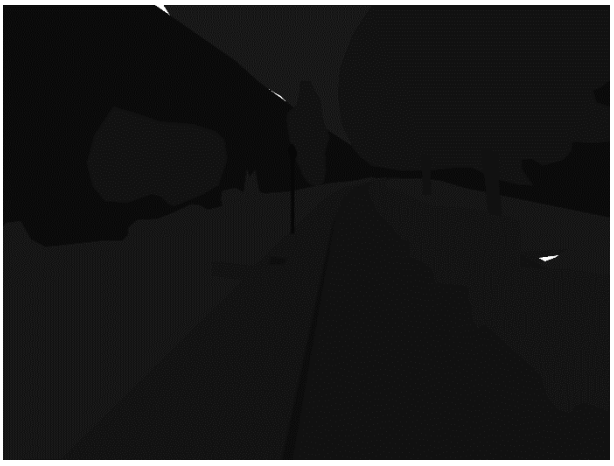


Bild 51: Lücken in der Annotierung weiß – fehlerhaftes Bild.

Festgelegte Reihenfolge bei der Digitalisierung:

- Himmel
- Straßenflächen
- Gebäude
- Vegetation
- Gehweg
- Baumkrone
- Bordstein
- Alle restlichen Objekte

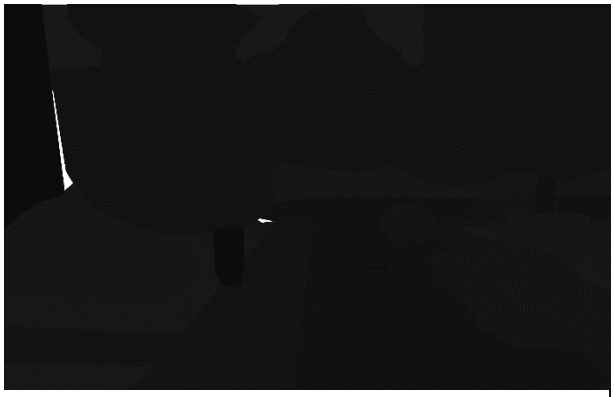


Bild 52: Bei der Annotierung fehlender Überlappungsbereich. Anpassung der Objektflächen ist hier notwendig.

Durch vermehrte falsche Digitalisierung wurde der Bearbeitungsmodus des Tools überarbeitet und angepasst. Des Weiteren wurden Lupenfunktionalität und Snapping-Funktionalität umprogrammiert.

Für eine schnellere Bearbeitung und Kontrolle wurde die programminterne Ordnerstruktur analog Kap. 5.1 angepasst. Somit benötigte das Programm nur den Eingangsordner für den Programstart. Alle weiteren Strukturen legte das Programm bei Bedarf eigenständig an. Dies verhinderte Speicherfehler bei der Bearbeitung oder wie anfangs getestet, wurde die Speicherstruktur vom Originalprojekt verwendet, um diese dann im Nachhinein umzukopieren. Hierbei entstanden anfangs große Probleme bei der Überarbeitung/Kontrolle der gelabelten Daten.

Abgeänderte Funktionen:

- Bearbeitung/Löschen von Stützpunkten
- Snapping von Punkten im Änderungsmodus
- Automatische Umwandlung von JSON in 8 Bit-PNG
- Umwandlung von PNG in JSON zur Korrektur von Ausgabedaten
- Verbesserung der Zoomfunktionalität
- Anpassen der Ordnerstruktur
- Anpassen der Labelklassen und IDs

Nach erfolgter Digitalisierung bekam jedes Polygon bei der Bearbeitung eine eindeutige ID zugewiesen. Diese ID wurde in allen weiteren Prozessschritten beibehalten und spiegelte die jeweilige Klasse wieder. Folgende Festlegungen wurden hier getroffen.

Tabelle 6 Festlegung der Label-IDs, die verwendet wurden, mit den dazugehörigen Farbwerten 8bit (Rot/Grün/Blau).

ID	Objekttyp	Farbe [R,G,B]
255	Unlabeled	0,0,0
1	Schild	0,0,255
2	Kilometertafel	0,0,50
3	Stationszeichen	0,0,100
4	Ortsdurchfahrtszeichen	255,255,0
5	Notrufsäule	255,0,0
6	Lichtsignalanlage	100,100,0
7	Beleuchtung	255,0,255
8	Mast	50,50,50
9	Leitpfosten	100,100,100
10	Schutzplanke	150,150,150
11	Schutzwand	20,20,20
12	Bauwerk	80,80,80
13	Verteilerkasten	40,80,120
14	Bordstein	120,80,40
15	Deckel	80,120,80
16	Fahrbahnmarkierung	200,200,200
17	Baumstamm	139,69,19
18	Straße	112,128,144
19	Baumkrone	34,139,34
20	Gehweg	163,163,163
21	Fahrzeug	255,127,80
22	Person	139,62,47
23	Vegetation	202,255,112
24	Boden	139,115,85
25	Himmel	0,0,0
26	Sperrfläche	200,200,200
27	Kennzeichen	50,100,200
28	Schacht	200,100, 50
29	Schildrückseite	0,0,100
30	Betonleitwand	150,150,150

Einige der in Tabelle 6 aufgeführten Objekttypen waren nicht Teil des Forschungsprojekes. Diese dienten als Platzhalter, um die Hauptobjekttypen besser bestimmen zu können.

Nach Fertigstellung der Trainingslabels wurde jedes einzelne Element mit dem gleichen Namen wie das korrespondierende Bild in eine Textdatei exportiert, auf dem es digitalisiert wurde. Die Textdatei besaß eine spezielle Formatierung namens JSON. Diese Beschreibungssprache regelt die Ablage der Metadaten und Attribute, ähnlich dem Format XML.

Nach Speicherung der JSON-Datei wurden die einzelnen Polygone in einem separaten 8 Bit-Grauwertbild gespeichert. Dieses Bild diente als Ground Truth-Eingangsdatei beim Training des neuronalen Netzwerks. Der Dateiname wurde wie bei der JSON-Datei vom ursprünglichen Bild beibehalten.

5.2.2 Zusammenfassung der bereits verarbeiteten Daten

Die für das Training des KNN erzeugten Annotierungen wurden aus folgenden Datensätzen generiert:

Strecke/Datensatz	Anzahl an Annotierungen
A001 (Autobahn A6)	179
B013 (Außerorts B87)	56
B002 (Außerorts)	62
B002 (Außerorts)	21
I080 (Innerorts Berlin)	39
I083 (Innerorts Berlin)	51

5.3 Training des künstlichen neuronalen Netzes

5.3.1 Umsetzung einer zweiten Netzwerkarchitektur

Bereits vor Abschluss der Erstellung der manuellen Annotierungen wurden die Ergebnisse der semantischen Segmentierung der Bilddaten mit einem ersten Teil der Annotierungen überprüft. Dieser umfasste 183 manuelle Annotierungen. Hierbei wurden bereits alle für das Forschungsobjekt definierten Objektklassen einbezogen. Die entsprechenden Zwischenergebnisse werden in Kap. 5.3.4 vorgestellt.

Um eine bessere Beurteilung bezüglich der Empfindlichkeit des Netzwerks für Overfitting treffen zu können (vgl. Kap. 3.1.11), wurde dabei eine weitere Netzarchitektur für das Training implementiert. Diese basiert auf dem sog. Fully

Convolutional Network (FCN-8s, LONG et al. 2016), und wurde dann auch für das finale Training und die finale Prozessierung der Daten beibehalten (vgl. Kap. 5.3.5).

Zunächst erfolgte ein Vergleich der Netzarchitekturen und eine Beschreibung der wichtigsten Parameter, die für das Training verwendet wurden.

5.3.2 Vergleich der Netzarchitekturen

Die beiden verwendeten Netzarchitekturen unterschieden sich in Aufbau und Abfolge der einzelnen Schichten, sowie in der Tiefe des Netzes. Die durch das Netz repräsentierten Modelle wiesen damit auch eine unterschiedliche Komplexität auf. Das U-Net verfügt über ca. 30 Millionen Parameter in 8 bzw. 9 Schichten (Convolutional Layer), während das in LONG et al. 2016 beschriebene, aus dem VGG-Netz (SIMONYAN; ZISSERMANN 2014) entwickelte FCN ca. 134 Millionen Parameter in 16 Schichten besitzt. Die beiden Netze unterscheiden sich auch bezüglich der Größe des Receptive Field, also der Umgebung, die bei der Klassifizierung eines Pixels in Betracht gezogen wird.

Von Interesse war neben der erreichbaren Erkennungsgenauigkeit auch die Frage, wie stark sich beim FCN Overfitting zeigen wird, insbesondere bei einem Trainingsdatensatz in der zur Verfügung stehenden Größe.

Eine direkte Vergleichbarkeit der Ergebnisse verschiedener Netze war jedoch im Allgemeinen nicht ohne Weiteres gegeben, da z.B. die Anzahl an benötigten Trainingsiterationen von der Lernrate abhing, die während des Trainings automatisch angepasst wurde. Beide Netze nutzen hierfür zwar den ADAM-Solver, doch wurde dieser Algorithmus wiederum von der Netzarchitektur, der Anzahl der Trainingsbilder oder der Batch-Größe beim Training beeinflusst.

5.3.3 Vorbereitung der Daten für das Training

Zu dem Zeitpunkt, an dem das Training des U-Net angestoßen wurde, standen 138 Trainingsbilder zur Verfügung. Für das FCN wurden 427 annotierte Datensätze einbezogen. Von diesen wurden mit 42 Bildern knapp 10 % zufällig ausgesondert und als Test-Datensatz zurückgehalten. Die Trainings- und Testdaten wurden für die Eingabe in das Netzwerk jeweils im hdf5-Format in entsprechende Pakete zusammengefasst.

Das U-Net verwendete Bilder in halbierter Auflösung sowohl für das Training als auch für die Klassifizierung. Der Speicher der Grafikkarte erlaubte damit eine Batch-Größe von vier Bildern, d.h., es können vier Bilder gleichzeitig geladen und für einen Lernschritt verwendet werden. Durch die höhere Komplexität des FCN ist es bei dessen Architektur nur möglich, mit einzelnen Bildern zu arbeiten, deren Auflösung mit einem Faktor von max. 0,6 skaliert wurde.

Die in der Datenaugmentierung verwendeten Transformationen beim U-Net wurden, wie bereits beschrieben, parametrisiert. Das später begonnene Training des auf der FCN-Architektur basierenden Netzes verwendete weder Datenaugmentierung, noch eine Initialisierung der Gewichte anhand eines vorhergehenden Trainings auf einem anderen Datensatz. Diese wurden also „von Grund auf“ neu trainiert.

5.3.4 Zwischenergebnisse

Qualitative Darstellung der Ergebnisse: U-Net

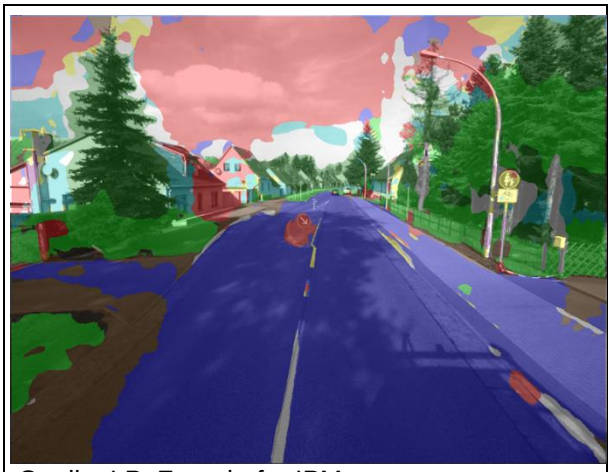
Das U-Net wurde auf dem Cityscapes-Datensatz vortrainiert. Die Netzarchitektur wurde für die neuen Objektklassen erweitert. Das Training für die neuen Klassen war auf 32.000 Iterationen auf einem sehr kleinen Trainingsdatensatz von 138 Bildern beschränkt, bevor die Abbildungen generiert wurden. Um Overfitting entgegenzuwirken, wurde mit Datenaugmentierung gearbeitet.

Ein vergleichsweise gutes Segmentierungsergebnis zeigt Bild 53. Hier zeigte sich erneut, dass Objekte wie Straße, Fahrbahnmarkierungen, Vegetation, Baumkronen und Bordsteine bereits gut erkannt werden. Problematisch erscheinen insbesondere noch Lichtsignalanlage, Masten und Schilder. Dieses Bild stammte nicht aus dem Trainingsdatensatz, ist jedoch in räumlicher Nähe zu einem Trainingsbild aufgenommen worden und damit nicht als echtes Testbild zu werten.

Quelle: LP, Fraunhofer IPM

Bild 53: Segmentierung eines Bildes, das in räumlicher Nähe zu einem Bild aus dem Trainingsdatensatz aufgenommen wurde (U-Net-basiertes Netzwerk, 32.000 Iterationen).

Die Klassifikation eines Bildes, das nicht in einem Bereich aufgenommen wurde, für den Trainingsdaten erzeugt wurden, zeigt Bild 54. Die Klassifikation ist hier insgesamt deutlich weniger zuverlässig. Dies zeigte, dass der Lernprozess des Netzwerks noch keine zufriedenstellende Generalisierung für neue Daten aufwies.



Quelle: LP, Fraunhofer IPM

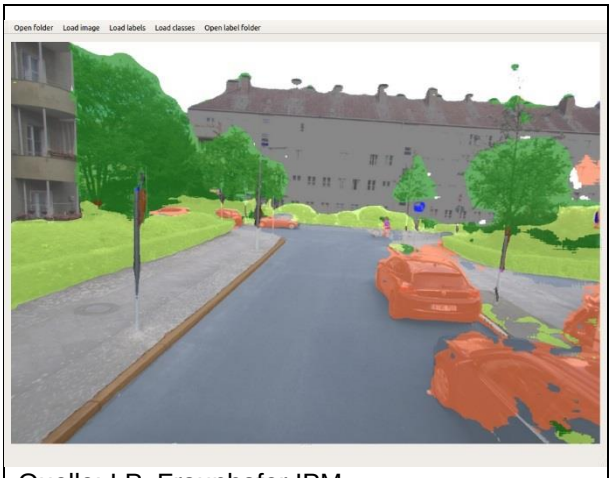
Bild 54: Segmentierung eines Bildes, für das keine ähnlichen Trainingsbilder existieren (U-Net-basiertes Netzwerk, 32.000 Iterationen).

Beide Bilder zeigten jedoch, dass das neuronale Netzwerk gut mit Bereichen zurechtkommt, in denen durch Schattenwurf die lokalen Kontrast- und Farbverhältnisse sehr uneinheitlich waren. Solche Lichtverhältnisse erforderten in „klassischen“ Algorithmen der Objekterkennung oft eine aufwändige Sonderbehandlung.

Qualitative Darstellung der Ergebnisse: FCN

Das Training des FCN-Netzes erfolgte mit 427 Trainings-Annotierungen, wovon 10% als Testdatensatz zurückgehalten wurden. Die folgenden Abbildungen wurden nach 60.000 Trainingsiterationen erzeugt, wobei keine Data Augmentation eingesetzt wurde. Aus Speichergründen wurden die Bilddaten mit einem Faktor von 0,6 skaliert.

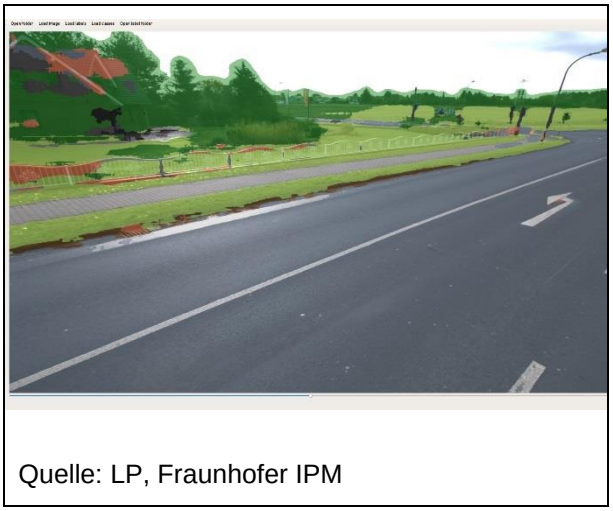
Bild 55 zeigt die semantische Segmentierung eines Bildes aus dem Testdatensatz, das an einer Position aufgenommen wurde, in deren Nähe auch manuelle Annotierungen generiert wurden, die im Training verwendet wurden. Die meisten Objektklassen wurden gut erkannt, insbesondere auch der Bordstein. Nicht erkannt wurde der auf dem Gehweg befindliche Deckel. Die Segmentierung der Bauwerke und des Himmels fiel bereits nach dieser kurzen Trainingsphase ohne Datenaugmentierung besser aus als beim U-Net. Kleine und weiter entfernte Objekte wie Masten, Schilder oder der Fahrradfahrer werden noch nicht zuverlässig erkannt. Im Randbereich der Bilder traten immer wieder problematische Stellen auf, wie hier bei der Segmentierung des PKW, obwohl Fahrzeuge generell sehr gut erkannt wurden.



Quelle: LP, Fraunhofer IPM

Bild 55: Segmentierung eines Bildes aus dem Testdatensatz, das in räumlicher Nähe zu einem Bild aus dem Trainingsdatensatz aufgenommen wurde (FCN-basiertes Netzwerk, 60.000 Iterationen).

Es zeigt sich, dass die Klassifizierungsleistung auf Bildern, für die keine „ähnlichen“ Trainingsdaten existierten, geringer ausfällt (Bild 56). Das Gebäude wurde hier sehr uneinheitlich klassifiziert. Andere Klassen wie Straße, Fahrbahnmarkierung, Gehweg, Mast oder Bordstein wurden jedoch auch hier überwiegend gut erkannt.



Quelle: LP, Fraunhofer IPM

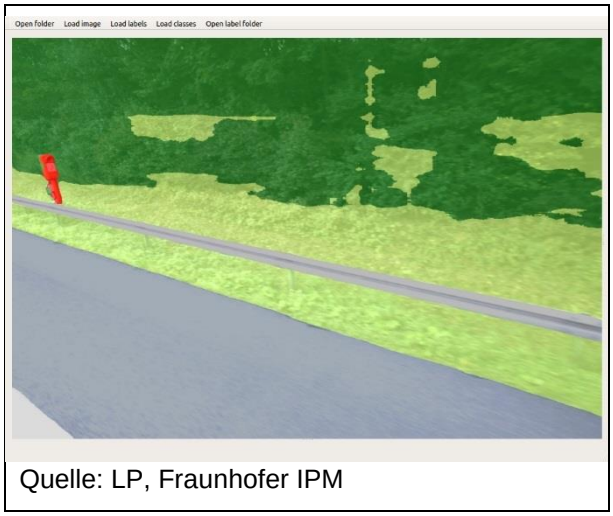
Bild 56: Segmentierung eines Bildes, das nicht in räumlicher Nähe zu einem Bild aus dem Trainingsdatensatz aufgenommen wurde (FCN-basiertes Netzwerk, 60.000 Iterationen).

Die Erkennungsgenauigkeit auf BAB-Strecken erschien insgesamt höher, da die Variation in den Eingangsdaten hier generell geringer ausfällt. Wiederum wurden Straße, Fahrbahnmarkierung und Fahrzeug, aber auch die Betonleitwand (Bild 57) zuverlässig erkannt. Auch die Schutzplanke im Hintergrund wurde sehr gut segmentiert.

Quelle: LP, Fraunhofer IPM

Bild 57: Segmentierung eines Testbildes zum Autobahn-Datensatz (FCN-basiertes Netzwerk, 60.000 Iterationen).

Da für die Seitenkameras mit ihrem nach unten geneigten Blickwinkel vergleichsweise wenige Trainings-Annotierungen zur Verfügung standen, fiel die Erkennungsgenauigkeit auf diesen Bildern geringer aus. Ein dennoch gutes Ergebnis zeigte Bild 58, bei dem die Notrufsäule deutlich erkannt wurde. Dies war jedoch nicht auf allen Testbildern der Fall. Außerdem war zu beachten, dass sehr ähnliche Seitenansichten in den Trainingsdaten enthalten waren.



Quelle: LP, Fraunhofer IPM

Bild 58: Segmentierung eines BAB-Bildes aus dem Testdatensatz: Die Notrufsäule wurde sehr gut segmentiert, ebenso die Schutzplanke (FCN-basiertes Netzwerk, 60.000 Iterationen).

Kleine Objekte wurden weniger zuverlässig oder zumindest erst später erlernt. Ein positives Beispiel waren allerdings die Leitpfosten, welche generell sehr gut segmentiert wurden. Hilfreich war dabei, dass die Erscheinung eines Leitpfostens wenig variiert, lediglich sehr schiefe oder stark verschmutzte Instanzen erschweren die Erkennung. Ähnlich große Objektlassen wie Kilometertafeln oder Stationszeichen variieren ebenfalls wenig in ihrer Ausdehnung und Geometrie, jedoch in der Beschriftung. Dies führte zu Overfitting, wenn zu wenige Instanzen dieser Klassen im Trainingsdatensatz vorhanden waren.

Quelle: LP, Fraunhofer IPM

Bild 59: Segmentierung eines Bildes aus dem Testdatensatz: Leitpfosten, Straße, Gehweg (Radweg), Vegetation, Baumkrone und Himmel werden zuverlässig segmentiert; Schilder erscheinen noch problematisch (FCN-basiertes Netzwerk, 60.000 Iterationen).

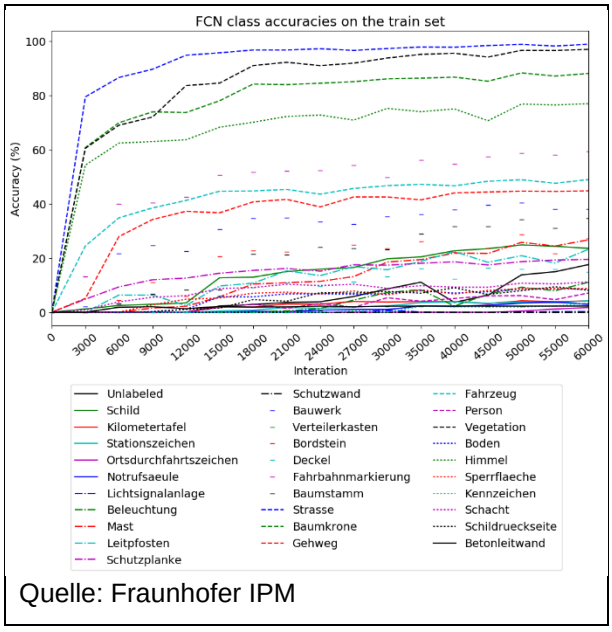
Quantitative Analyse der Ergebnisse des FCN

Während des Trainings ließ sich die Entwicklung der Erkennungsgenauigkeit für einzelne Objektklassen beobachten. Von besonderem

Interesse war der Vergleich der Klassifizierungsleistung auf dem Trainings- und dem Testdatensatz (vgl. Kap. 3.1.11). Hier wurden erste Anhaltspunkte detektiert, inwiefern beim Training Over- oder Unterfitting auftritt. Dies lieferte wiederum Rückschlüsse auf die Erkennungsgenauigkeit für neue, unbekannte Daten.

Im Folgenden wurde dieser Vergleich für die ersten 60.000 Iterationen des FCN dargestellt, das mit 427 Trainings-Annotierungen, minus 10% für den Testdatensatz, ohne Datenaugmentierung trainiert wurde. Verwendet wurde die in Kap. 3.3.3 eingeführte Genauigkeitsmetrik Intersection-over-Union (IoU).

Die folgende Abbildung zeigt diese für den Trainingsdatensatz während der ersten 60.000 Iterationen beim FCN-Netzwerk.

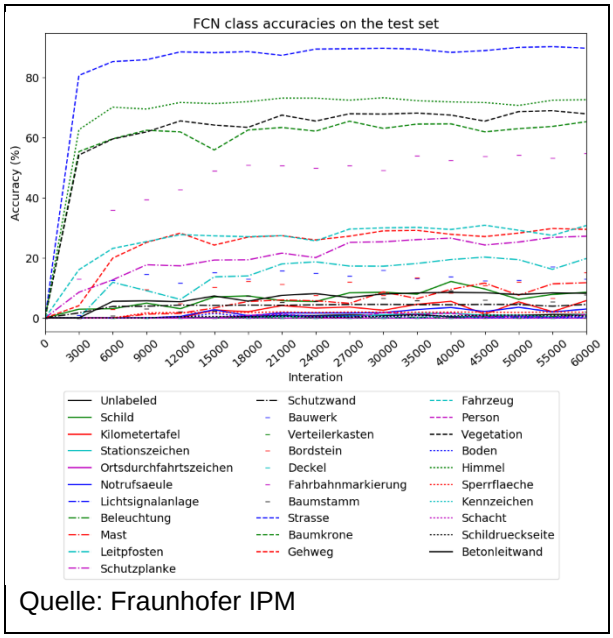


Quelle: Fraunhofer IPM

Bild 60: Erkennungsgenauigkeit pro Klasse des FCN bis 60.000 Iterationen auf den 385 Bildern des Trainingsdatensatzes.

Nach einem schnellen Erlernen der Merkmale für großflächige Objekte wie Straße, Himmel oder Vegetation bedarf es einer längeren Zeit, bis auch weitere Objektklassen, z.B. Masten, zuverlässiger erkannt wurden.

Die Erkennungsgenauigkeit auf dem Testdatensatz zeigt Bild 61.



Quelle: Fraunhofer IPM

Bild 61: Erkennungsgenauigkeit pro Klasse des FCN bis 60.000 Iterationen auf den 42 Bildern des Testdatensatzes.

Die 5 im Trainingsdatensatz am besten erkannten Klassen wurden ähnlich gut erkannt, was insbesondere auch für die Fahrbahnmarkierungen galt. Allerdings zeigte sich, dass die Unterscheidung von Vegetation und Baumkrone hier schwerer fiel; eine einheitliche Differenzierung dieser Klassen fällt jedoch bereits bei der Annotierung nicht leicht.

Eine ähnliche gegenseitige Beeinflussung ergab sich auch bei teiltransparenten Objekten, wie Vegetation und Baumkrone. Waren andere Objekte dahinter sichtbar, hing die erreichbare Erkennungsgenauigkeit auch von der manuellen Annotierung ab, wie in Kap. 3.3.3 bereits dargestellt.

Die Erkennungsleistung der weiteren Klassen fiel hinter die auf den Trainingsdaten erreichte Genauigkeit zurück. Manche Klassen wurden im Testdatensatz jedoch sogar besser erkannt als in den Trainingsdaten. Hierbei handelte es sich um einen zufälligen Effekt, der durch die begrenzte Größe der Datensätze entstand. Des Weiteren dadurch, dass die Reihenfolge, in denen das Netzwerk die Merkmale bestimmter Klassen lernte, nicht immer gleich war.

Im Vergleich der beiden Grafiken wurde sichtbar, dass die Erkennungsgenauigkeit auf dem Trainingsdatensatz tendenziell auch nach 60.000 Iterationen weiter stieg, während die Verbesserung auf dem Testdatensatz weniger deutlich ausfiel. Dies war ein Hinweis darauf, dass Overfitting an den Trainingsdatensatz zu erwarten war.

Für verhältnismäßig kleine oder im Datensatz selten vorkommende Objekte, wie Stationszeichen, Kilometertafeln, Ortsdurchfahrtstafeln oder Notrufsäulen war unklar, wie sich die Erkennungsgenauigkeit im weiteren Verlauf des Trainings entwickeln würde. Möglicherweise würde eine geeignete Datenaugmentierung hier eine Steigerung ermöglichen.

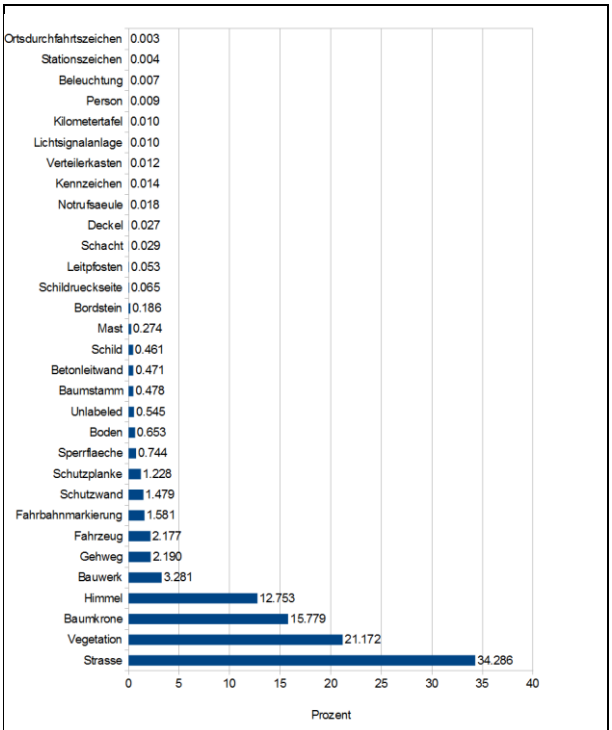
Der teilweise wellenförmige Verlauf der Genauigkeiten war primär auf die kleinen Batch-Größen beim Training sowie die Art und Weise der Gradientenberechnung beim Training zurückzuführen.

5.3.5 Finales Training

Die auf LONG et al. 2016 basierende Netzarchitektur wurde für das finale Training beibehalten. Dasselbe galt für die Hyperparameter, mit denen das frühere Training aufgesetzt wurde. Dadurch wurde die Vergleichbarkeit des finalen Trainings mit dem vollständigen Satz an Annotierungen gegenüber dem vorläufigen Training des FCN-Netzes mit nur 427 Trainingsbeispielen gegeben.

5.3.5.1 Analyse der Trainingsdaten

Im finalen Trainingsdatensatz standen insgesamt 941 manuell annotierte Bilder zur Verfügung, von denen 92 Bilder nach dem Zufallsprinzip für den Testdatensatz entnommen wurden. Das Training fand demnach mit 849 Annotierungen statt. Die Klassenverteilung im gesamten Datensatz zeigt Bild 62.

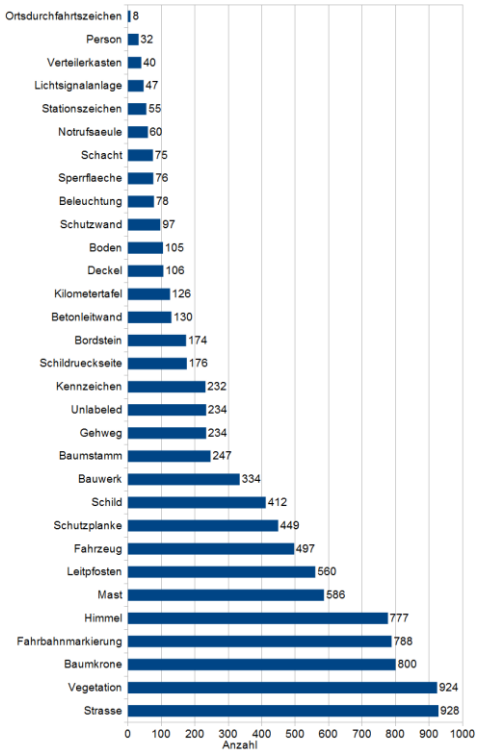


Quelle: Fraunhofer IPM

Bild 62: Pixel-Anteil der einzelnen Klassen in den 941 manuellen Annotierungen

Es war direkt ersichtlich, dass die Verteilung äußerst uneinheitlich ausfiel. Dies stellte eine generelle Herausforderung bei der semantischen Segmentierung von Bilddaten dar, da es selten vorkam, dass die Bereiche, die einzelne Klassen in den Bildern belegen, über den gesamten Datensatz ähnlich groß waren.

Auch die Auswertung der Anzahl der Bilder, auf denen die einzelnen Klassen vertreten waren, war für eine Abschätzung, wie gut die Klasse gelernt werden kann, interessant. Bild 63 zeigt, dass insbesondere die Klassen Ortsdurchfahrtszeichen, Verteilerkasten, Lichtsignalanlage, Stationszeichen und Notrufsäule auf nur wenigen Bildern vertreten waren.



Quelle: Fraunhofer IPM

Bild 63: Anzahl der manuell annotierten Bilder, auf denen die einzelnen Klassen vorhanden sind.

Für Objekte, die selten vorkamen und zusätzlich noch klein waren, ergaben sich damit besonders ungünstige Voraussetzungen im Training des KNN. Insbesondere waren dies, neben den eben schon erwähnten seltenen Klassen, die Beleuchtungen, die Kilometertafeln, die Kennzeichen, sowie Deckel und Schächte. Bei diesen Klassen hing die Zuverlässigkeit der Erkennung dann nicht nur davon ab, ob das Netzwerk in der Lage war, geeignete Merkmale zu extrahieren, sondern auch davon, ob ihr Einfluss beim Lernprozess von den prominenteren Klassen überdeckt wurde.

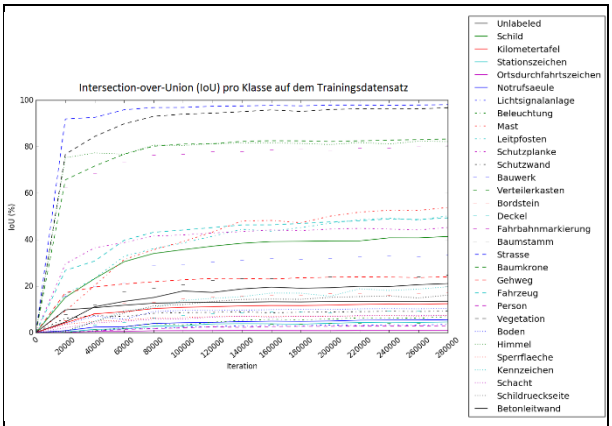
Dass auch solche Klassen gut gelernt werden können, zeigte sich bei der Analyse der Ergebnisse z.B. für Leitpfosten. Diese treten zwar häufig auf, fallen aufgrund ihrer geringen Größe beim Training wenig ins Gewicht.

Es war auch ersichtlich, dass die in beiden Kategorien vorne liegenden Klassen tatsächlich auch bei der IoU-Genauigkeit besonders gut abschneiden. Der Lernprozess korrelierte demnach wie erwartet stark mit der Dominanz der Klassen im Trainingsdatensatz.

5.3.5.2 Analyse der Trainingsergebnisse

Quantitative Analyse

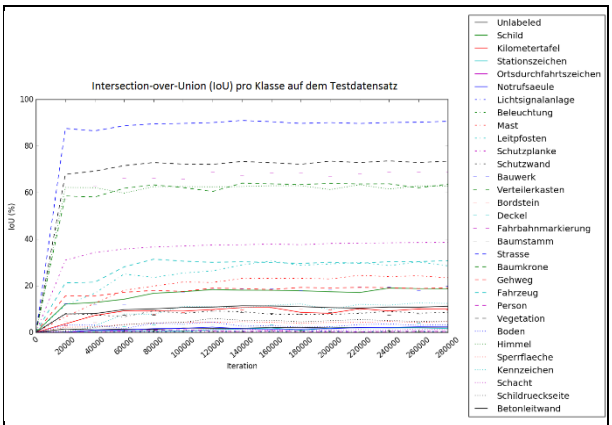
Das Training wurde 280.000 Iterationen lang durchgeführt, was mit einer Titan X-GPU etwa 5 Tage in Anspruch nahm. Die resultierenden pixelbasierten Genauigkeiten nach der Intersection-over-Union-Metrik (IoU) auf dem Trainingsdatensatz zeigt Bild 64.



Quelle: Fraunhofer IPM

Bild 64: Erkennungsgenauigkeit pro Klasse des FCN bis 280.000 Iterationen auf dem Trainingsdatensatz.

Die für die Generalisierungsleistung des KNN relevante Genauigkeit auf dem Testdatensatz ist in Bild 65 dargestellt.



Quelle: Fraunhofer IPM

Bild 65: Erkennungsgenauigkeit des FCN pro Klasse, bis 280.000 Iterationen auf dem Testdatensatz.

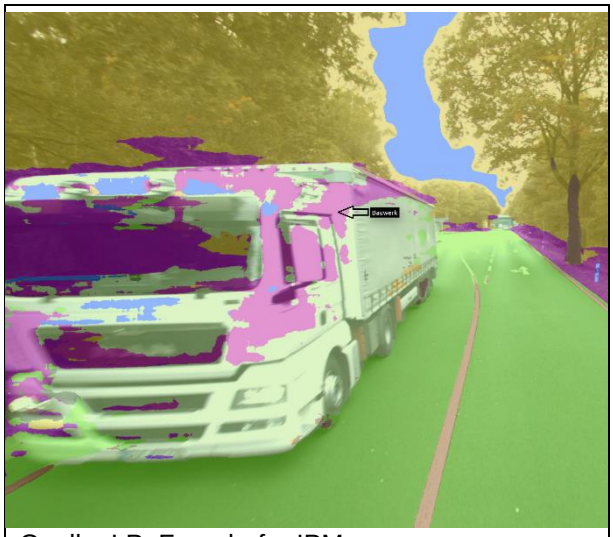
Wie zu erwarten, fiel die Genauigkeit auf dem Testdatensatz insgesamt geringer als auf dem Trainingsdatensatz. Auffällig war wiederum, dass die Klassen, welche den höchsten Anteil an Pixeln ausmachten, mit deutlichem Abstand am besten gelernt wurden. Dies waren vor allem Straße und Baumkrone, aber auch Vegetation und (mit Einschränkung) Bauwerke.

Klassen, die gut gelernt wurden, obwohl ihr Pixelanteil nicht überproportional groß ausfiel, waren z.B. Fahrbahnmarkierung, Schutzplanke und Leitpfosten. Dies war darauf zurückzuführen, dass sie in ihren Merkmalen, im Vergleich zu komplexeren Klassen wie den Fahrzeugen, eindeutig bestimmt wurden.

Fahrzeuge und Bauwerke/Gebäude wurden generell gut erkannt, was sich jedoch nicht unbedingt als herausragender Wert in der IoU niederschlug, wohl aber bei der qualitativen Kontrolle in der Durchsicht der erzeugten semantischen Segmentierungen auffiel. Gerade bei Gebäuden wurde die IoU durch unsichere Übergangsbereiche zu großflächigen vorgelagerten Objekten, wie Büschen oder Bäumen, reduziert, wo eine eindeutige Klassifizierung aufgrund der fließenden Übergänge oft nicht möglich war (vgl. Kap. 3.3.3).

Qualitative Analyse

Umgekehrt fiel auf, dass immer wieder Objekte im Straßenumfeld fälschlicherweise als Gebäude klassifiziert wurden, wenn bestimmte Merkmale vorhanden waren, die ebenfalls auf Gebäude zutrafen. Dies wirkte sich negativ auf die IoU der Gebäude-Klasse aus. Wesentlich war z.B., dass bei LKW oft ein Teil als Bauwerk klassifiziert wurde, da die visuelle Erscheinung in diesen Bereichen ähnliche Merkmale aufwies, wie etwa rechtwinklig konstruierte, großflächige Elemente. Eine weitere Ursache hierfür war, dass für die Klassifizierung nur Teile des Bildes, das sog. Receptive Field, um ein bestimmtes Pixel vom KNN in Betracht gezogen wurde. Wenn ein nah beim Befahrungsfahrzeug befindlicher LKW einen großen Teil des Bildes ausfüllte, fielen die unteren, eher „fahrzeug-typischen“ Bereiche des LKW für die Klassifizierung des oberen Fahrzeugteils nicht mehr in das Receptive Field. Allgemein wurden Objekte, welche sehr groß oder sehr klein abgebildet wurden, deshalb auf den entsprechenden Bildern oft weniger zuverlässig erkannt.



Quelle: LP, Fraunhofer IPM

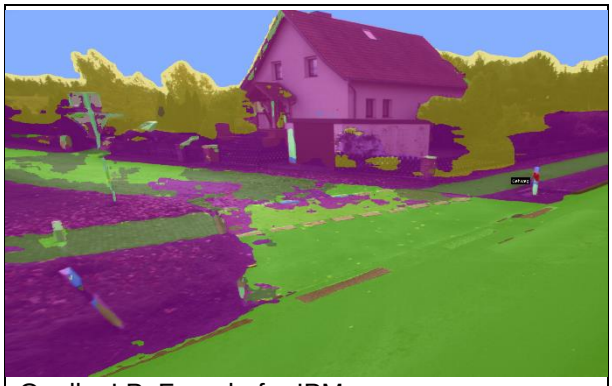
Bild 66: Selten im Trainingsdatensatz vorkommende Objekte, die zudem noch nahe zum Messfahrzeug aufgenommen wurden, stellen eine Herausforderung dar: Der obere Teil des LKW wird auf einem Bild aus dem Testdatensatz in Teilen als Bauwerk klassifiziert (Training nach 180.000 Iterationen).

Klassen wir Vegetation, Baumstamm/Baumkrone aber auch Fahrzeuge und Schilder, können nur anhand komplexer Merkmale zuverlässig erkannt werden, da sie in ihrer Erscheinung (Form, Farbe, Beschriftung etc.) stark variieren. Hier zeigte sich die Stärke von Deep Learning: Durch die hierarchische Verknüpfung eines großen Satzes an Merkmalen konnten auch solche Klassen relativ zuverlässig erkannt werden, wenn sie im Trainingsdatensatz ausreichend häufig und mit ausreichender Variation vertreten waren.

Interessant war auch die Betrachtung der IoU für die Klasse „Gehweg“. Diese kam vergleichsweise häufig vor und belegte somit einen größeren Anteil an Pixeln im Trainingsdatensatz. Dennoch fiel die IoU mit knapp 20 % vergleichsweise gering aus. Dies lag darin begründet, dass oft fließende Übergänge und eine große Ähnlichkeit mit der Klasse „Straße“ vorhanden waren und es entsprechend häufiger zu Verwechslungen kam. Tatsächlich wurden vergleichsweise oft Bereiche, welche als Gehweg annotiert wurden, vom KNN als Straße klassifiziert. Umgekehrt war das eher selten der Fall. Aufgrund der dominierenden Größe der Straßenflächen wurde die hervorragende IoU der Klasse „Straße“ von diesen Verwechslungen nur unwesentlich beeinträchtigt.

Dieser Effekt ist in Bild 67 in den Übergangsbereichen von Gehweg und Straße zu sehen, wo ein Teil der Straße als Gehweg klassifiziert wurde. Dort ist auch ersichtlich, dass Bereiche, in denen die Straße stark von Laub

verdeckt ist, als Vegetation oder auch als Gehweg klassifiziert wurden, da hier die Merkmale dieser Klassen gegenüber denen der Straße überwiegen. Offensichtlich sind Gehwege häufiger mit Laub bedeckt, als es bei der Straße der Fall ist.



Quelle: LP, Fraunhofer IPM

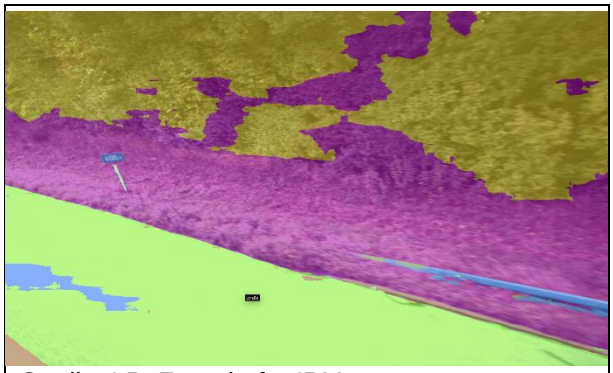
Bild 67: Der Übergang von Straße und Gehweg ist in der Ausgabe des KNN teilweise fließend (Training nach 180.000 Iterationen); eine größere Herausforderung stellen auch die Bereiche dar, welche stark mit Laub bedeckt sind.

Es zeigt sich, dass Über- und Unterbelichtung, wie in Bild 68 dargestellt, vom KNN in einem gewissen Rahmen toleriert werden.

Quelle: LP, Fraunhofer IPM

Bild 68: Die Straßenoberfläche ist auf diesem Bild aus dem Testdatensatz deutlich überbelichtet.

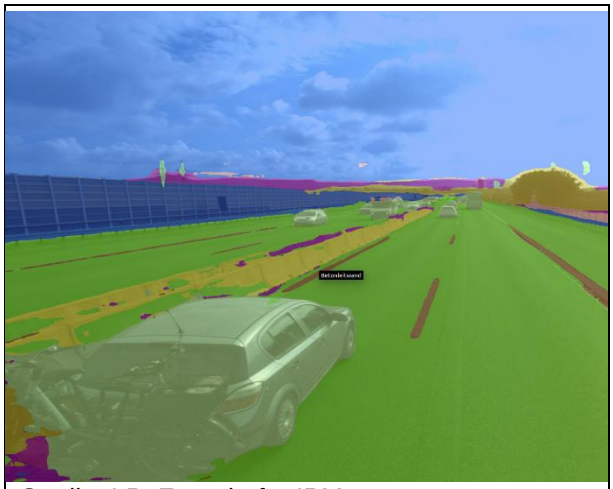
Die zugehörige Segmentierung des KNN (Bild 69) zeigt eine korrekte Klassifizierung der Oberfläche als „Straße“, außer in den Bereichen, in denen praktisch kein Kontrast mehr vorhanden war. In solchen komplett weißen Bereichen wurde vom KNN in der Regel die Klasse „Himmel“ ausgegeben.



Quelle: LP, Fraunhofer IPM

Bild 69: Die semantische Segmentierung der überbelichteten Straße und der Fahrbahnmarkierung (Ecke links unten) durch das KNN ist, bis auf die gänzlich weißen Bildbereiche, korrekt (Training nach 180.000 Iterationen).

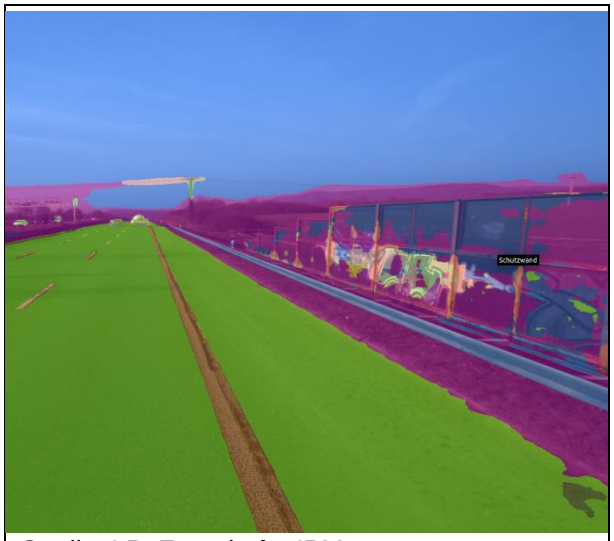
Der Einfluss von Vandalismus, z.B. in Form von Graffiti, spielt bei der Objekterkennung durchaus eine Rolle. Allerdings kann auch hier in gewissen Grenzen eine Toleranz durch das KNN gelernt werden. Folgendes Bild zeigte eine sehr gut erkannte Schutzwand entlang einer mehrspurigen Straße.



Quelle: LP, Fraunhofer IPM

Bild 70: Die Schutzwand im linken Bereich wird auf diesem Testbild korrekt erkannt (Training nach 180.000 Iterationen).

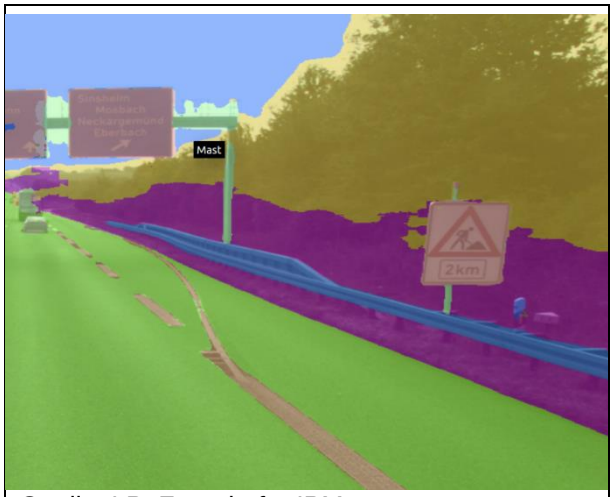
Eine mit Graffiti besprühte Schutzwand wurde oft nur in Teilen korrekt erkannt, wobei durchaus auch besprühte Bereiche noch richtig klassifiziert werden können (vgl. Bild 71).



Quelle: LP, Fraunhofer IPM

Bild 71: Die besprühte Schutzwand auf diesem Bild aus dem Testdatensatz wird nur in Teilen korrekt klassifiziert (Training nach 180.000 Iterationen). Die besprühten Bereiche waren dabei teilweise korrekt, teilweise mit einer falschen Klasse segmentiert.

Dass sehr kleine Objekte zum Teil nur ungenügend gelernt wurden, lag auch im mathematischen Algorithmus des Gradientenabstiegs beim Training begründet, welcher generell großflächige und häufig vorkommende Objekte bevorzugt. Eine besonders problematische Klasse war in diesem Trainingsdatensatz das Stationszeichen (vgl. Bild 72).



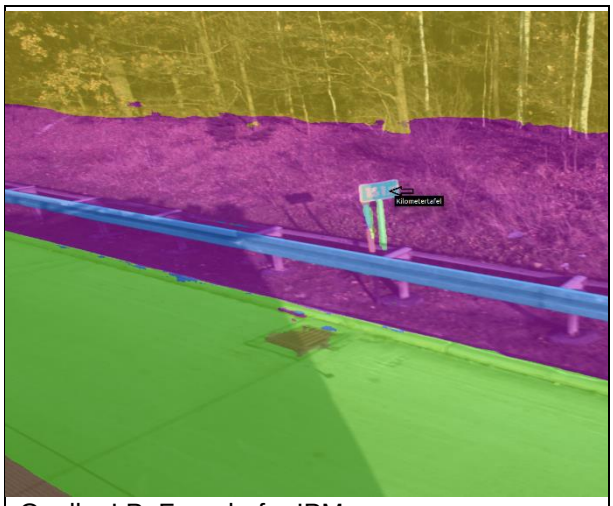
Quelle: LP, Fraunhofer IPM

Bild 72: Das Stationszeichen auf der rechten Seite wurde nicht erkannt (Bild aus dem Testdatensatz; Training nach 180.000 Iterationen).

Um Objekte, die sowohl klein sind, als auch selten im Trainingsdatensatz vorkommen, zuverlässiger erkennen zu können, waren aufwändige, klassenspezifische Anpassungen des Trainings-vorgangs notwendig. Dies konnte eine stärkere Gewichtung der entsprechenden Klassen bei der Berechnung der pixelweisen Loss-Funktion beim Training sein.

Eine weitere Möglichkeit war die häufigere Wiederholung von Bildern mit seltenen Objekten, wodurch jedoch das „Zufallsprinzip“ beim Training ein Stück weit außer Kraft gesetzt wurde, welches für ein optimales Training eine wichtige Rolle spielte. Prinzipiell konnte auch ein „synthetischer“ Datensatz erzeugt werden, in dem außer den kleinflächigen alle Objekte ausgeblendet wurden. Hier wäre experimentell zu bestimmen, ob ein tatsächlicher Zugewinn an Genauigkeit erreicht werden kann.

Dennoch zeigte die qualitative Analyse der Ausgabe des KNN, dass kleine und selten vorkommende Objekte durchaus immer wieder, zumindest teilweise, erkannt wurden.



Quelle: LP, Fraunhofer IPM

Bild 73: Der rechte Teil der Kilometertafel wird auf diesem Testbild korrekt erkannt (Training nach 180.000 Iterationen), der linke Teil ist als Schild klassifiziert. Auch der Deckel wurde zu einem großen Teil der korrekten Klasse zugeordnet.

Die wirksamste Methode, um der Unterrepräsentierung bestimmter Klassen entgegenzuwirken, lag jedoch in einer gezielten Steuerung des Prozesses der manuellen Annotierung von Daten, wenn ein größerer Trainingsdatensatz aufgebaut wurde, als es für das Forschungsprojekt möglich war. So sollten verstärkt Bilder für das Labeling ausgewählt werden, welche die seltenen und/oder kleinen Objekte enthalten, ohne jedoch dabei die Diversität des Datensatzes insgesamt als Ziel aus den Augen zu verlieren.

5.3.6 Herausforderungen und weitere Entwicklungsmöglichkeiten

Hyperparameter

Wie bereits in Kap. 3.1.10 dargestellt, hing der Lernerfolg eines Netzes nicht nur von den Trainingsdaten, sondern von einer Vielzahl von sog. Hyperparametern ab. Diese beziehen sich auf die Architektur des Netzes, aber auch auf mathematische Funktionen und Algorithmen, die beim Training zum Einsatz kommen. Die ist bspw.die Art und Weise, wie die Lernrate festgelegt wurde.

Um verschiedene Sätze von Hyperparametern zu testen, wurde jeweils ein Neu-Training des Netzwerks erforderlich. Deshalb war immer nur eine bestimmte Auswahl an Parametern tatsächlich getestet worden.

Data Augmentation

In der Regel wurden die Algorithmen für die Augmentierung anwendungsspezifisch festgelegt und implementiert, da nicht alle Transformationsarten in den Eingangsdaten auftraten. Im Kontext der Straßenbestandsobjekte war eine wellenförmige Verzeichnung für Klassen wie Mast, Verteilerkasten oder Gebäude wenig sinnvoll, könnte aber bspw. für Fahrbahnmarkierungen Vorteile mit sich bringen.

Die Augmentierung konnte „außerhalb“ des Netzes stattfinden, indem die Trainingsdaten unter Anwendung der relevanten Transformationen auf der Festplatte vervielfältigt wurden. So wurden Bilder, z.B. in bestimmten Winkelintervallen, rotiert, skaliert, zugeschnitten oder perspektivisch verzerrt. Durch die Kombination der möglichen Parameter entstanden jedoch schnell unüberschaubare Datenmengen.

Sinnvoller war es deshalb, eine Augmentierung direkt in die Trainingsalgorithmen des Netzes zu integrieren, wo sie dann während des Trainings im Arbeitsspeicher vorgenommen wird. Dies erfordert jedoch einen höheren Programmieraufwand als eine simple Vervielfachung der Trainingsdaten.

Manuelle Annotierung und Klassifizierungsgenauigkeit

Es zeigte sich, dass die manuelle Annotierung in mehrerlei Hinsicht nicht trivial war. Insbesondere war es schwierig, eine konsistente Annotierung zu gewährleisten. So war es beispielsweise nicht klar, bis zu welcher Entfernung oder „Sichtbarkeit“ ein Zaun als separates Objekt (bzw. als Unlabeled) markiert werden sollte. Dasselbe galt z.B. auch für die Objektklasse „Fahrbahnmarkierungen“.

Auch mit genau definierten Handlungsanweisungen werden Annotierungen verschiedener Menschen, aber auch ein und derselben Person, immer gewisse Abweichungen bezüglich der Einheitlichkeit aufweisen. In gewissen Grenzen ist dieser Effekt unproblematisch. Werden jedoch in einem vergleichsweise kleinen Datensatz Objekte, die nur wenige Pixel umfassen oder selten vorkommen, uneinheitlich gelabelt, macht sich dies durchaus in der erreichbaren Erkennungsgenauigkeit für diese Objektklassen bemerkbar.

Generalisierung und Overfitting

Der für das Forschungsprojekt erzeugte Trainingsdatensatz umfasste zunächst 138, dann 427 und am Ende 941 annotierte Bilder. Um Overfitting zuverlässig zu vermeiden, wäre eine Anzahl an Trainingsbildern erforderlich, wie sie im Cityscapes-Datensatz zur Verfügung steht, d.h. idealerweise in der Größenordnung von mind. 5.000 Annotierungen. Um optimale Klassifizierungsergebnisse erzielen zu können, wäre eine noch größere Anzahl sinnvoll, da im Forschungsprojekt eine höhere Diversität an Umgebungstypen vorlag (Stadt, Bundes-/Landstraßen, Autobahn) als im Cityscapes-Datensatz (ausschließlich Stadt).

Der vorliegende Datensatz ermöglichte zwar eine Einschätzung, welche Erkennungsgenauigkeit auf einem Testdatensatz erreicht werden kann, der von den zur Verfügung stehenden Annotierungen separiert wurde. Es ist jedoch davon auszugehen, dass mit einem umfangreicheren Trainingsdatensatz eine signifikant bessere Generalisierung auf neuen, unbekannten Daten erzielt werden könnte. Dies gilt insbesondere auch für Daten, welche mit einem anderen Messfahrzeug aufgezeichnet wurden.

Unvollständig oder fehlerhaft klassifizierte Bildbereiche erschwerten in der Folge die Detektierung von einzelnen Objekten in 3D nach Übertragung der Klassenlabel in die Punktwolke (vgl. Kap. 5.4.4).

5.3.7 Anmerkungen zur Infrastruktur für das Training

Der rechenaufwändige Prozess des Trainings war dann erneut durchzuführen, wenn neue Trainingsdaten in Form von manuell generierten Annotierungen zur Verfügung standen oder eine neue Netzarchitektur evaluiert werden sollte. Für die Auswertung neu aufgezeichneter Messdaten war dieser Schritt nicht notwendig. Er wurde deshalb nicht in das Programm integriert, welches die Batch-Verarbeitung von Messdaten ermöglicht.

Um das neuronale Netz zu trainieren, wurden zunächst die erforderlichen Parametrisierungen (Netzarchitektur, Hyperparameter) in entsprechenden Textdateien abgelegt. Die Trainings- und Testdaten wurden im hdf5-Format gebündelt.

Um das Training zu starten, werden Skripte verwendet, welche die Python-Schnittstelle der Deep-Learning-Bibliothek Caffe ansprechen. Es wurden verschiedene zusätzliche Skripte ebenfalls in Python entwickelt, welche eine detailliertere Analyse des Lernprozesses und die abschließenden Genauigkeitsanalysen sowie die zugehörigen Visualisierungen ermöglichen. Die wichtigsten Abhängigkeiten sowie die verwendete Infrastruktur wurden in Kap. 3.6 aufgelistet.

Der Trainingsprozess war sehr rechenintensiv, weshalb er nur mit einer leistungsstarken GPU realisierbar war. Das Training neuronaler Netze findet für gewöhnlich in Linux-basierten Umgebungen statt. Das Caffe-Framework steht jedoch auch in einer Portierung für Windows zur Verfügung (https://github.com/BVLC/caffe/tree/windows).

5.4 Objektextrahierung und Batch-Prozessierung

Für die Batch-Prozessierung bei der Mustererkennung kamen verschiedene Prototypenprogramme zum Einsatz.

Die für die Batch-Prozessierung relevanten Programme umfassen:

- Den PhoML-Viewer zur Erzeugung der Tiefenbilder.
- Die prototypische Software zur semantischen Segmentierung von Bilddaten und Tiefenbildern, Rückprojektion der erzeugten Klassenlabel in die Punkwolke sowie Klassifizierung bzw. Extraktion georeferenzierter Objekte in 3D (ohne GUI).
- Programm zur Speicherung der Ergebnisse und Übertragung in die Datenbank.

Das Training eines Neuronalen Netzwerks ist lediglich einmalig durchzuführen. Nach einem erfolgreichen Training können mit dem fertig gelernten Modell beliebig viele Mengendaten durch die prototypische Software prozessiert werden, sofern sie von derselben Messplattform stammen und die Ordnerstruktur in geeigneter Weise aufbereitet ist. Die Skripte für Vorbereitung, Durchführung und Analyse des Trainings des Neuronalen Netzwerks wurden in der Python-Skriptsprache programmiert.

Das folgende Ablaufdiagramm stellt den Datenfluss zwischen den verschiedenen Teilen der prototypischen Prozedur im Überblick dar.

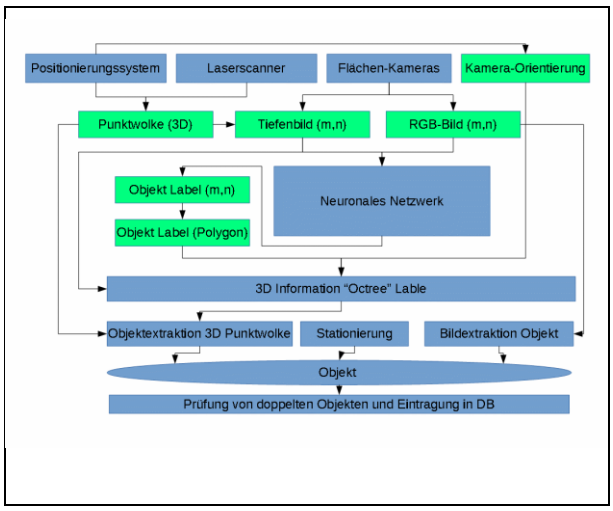


Bild 74: Ablaufplan mit Zwischenschritten und deren Beziehungen zueinander.

Der Mittelteil des in Bild 74 dargestellten Ablaufs entspricht der Darstellung der Prozessierung der Daten im neuronalen Netzwerk (vgl. Bild 21 in Kap. Kap. 2.2.4.1 zu „Strategie 1“). Hier findet die Klassifizierung in 2D auf RGB-Daten statt. Ergebnis dieses Zwischenschritts war die „semantische Segmentierung“. Danach wurden die Klassenlabel zurück in die Punktwolke übertragen, um dort die Klassifizierung/Extrahierung einzelner Objekte in 3D abzuschließen (dort als „Instanz-Differenzierung und Segmentierung der Scanpunkte“ bezeichnet).

5.4.1 Erstellen der Tiefenbilder

Wie bereits in Kap. 5.1.2 dargelegt, erfolgte die Berechnung automatisch. Hierbei können für einzelne Messfahrten, Messtage oder eine komplette Messkampagne Tiefenbilder berechnet werden. Voraussetzung hierfür sind eine Punktwolke, eine Index-Datei jedes Punktwolkenabschnittes und jeweils ein kalibriertes Messbild. Alle Tiefenbilder werden dann in einen separaten Ordner “DEPT“ erzeugt. Die Zuordnung erfolgt über identische Dateinamen, jedoch mit anderen Dateisuffixen.

5.4.2 Semantische Segmentierung der Bilddaten

Die Auswertung neuer Messdaten, die in der vordefinierten Ordnerstruktur vorliegen, begannen mit der Segmentierung der RGB-Bilder durch das Neuronale Netzwerk. Die erzeugten Segmentierungsmasken wurden bei Bedarf unter Beibehaltung des Dateinamens des Eingangsbildes mit dem Dateisuffix „_label“ abgelegt. Sie stellten ein Zwischenergebnis dar, das in der weiteren Prozessierung für die Übertragung der Klassenlabel in die Punktwolke verwendet wurde. Da sie nur temporär benötigt wurden, sind sie nicht Bestandteil der Ergebnisordner.

5.4.3 Übertragung der Klassenlabel in die Punktwolke

Vorbemerkungen

Da für die Objekterkennung keine hoch genaue und hoch aufgelöste geometrische Abtastung der Straßenoberfläche benötigt wurde, war die Implementierung im Prototyp auf die Punktwolken des CPS-Scanners beschränkt. Dieser erfasste die komplette Umgebung mit einer für das Forschungsprojekt hinreichenden Genauigkeit. Für die Berechnungszeit zeigte sich zudem, dass eine sehr hohe Punkdichte zu einer überproportionalen Verlängerung führte. Prinzipiell ließen sich die beschriebenen Algorithmen jedoch auf die gemeinsame Nutzung der beiden Punktwolken des CPS- und des PPS-Scanners übertragen.

Grundlegendes Vorgehen

Wenn zusätzlich zu den segmentierten Bildern ein Ordner mit Punktwolken geladen wurde und die Informationen zur Trajektorie vorhanden waren, konnte die Rückprojektion der Klassenlabel der segmentierten Bilder in die Punktwolke ausgeführt werden.

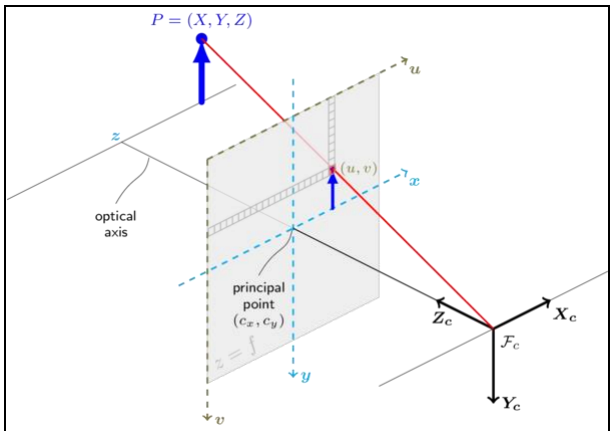
Ausgehend von der Trajektorie des Messfahrzeugs, welche die Positionen der aufgenommenen Bilder bestimmt, und der Kalibrierung der Kamera-positionen und -ausrichtungen (bezogen auf den Wagenursprung) wurde jeweils eine virtuelle Kamera in der Punktwolke platziert. Die internen Kameraparameter und ein Kameramodell dienten dazu, die Klassenlabel von den einzelnen Pixeln der Segmentierungsmasken auf Punkte in der Punktwolke zu übertragen.

Das Kameramodell basierte auf der Erzeugung eines Bildes mit einer Lochkamera, mit der Möglichkeit der Erweiterung durch Verwendung der intrinsischen Kameraparameter (vgl. z.B. http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$
$$y' = y/z$$
$$u = f_x * x' + c_x$$
$$v = f_y * y' + c_y$$

Folgende Abbildung veranschaulicht den Zusammenhang zwischen 3D-Weltkoordinaten und 2D-Bildkoordinaten:



Quelle: opencv.org

Bild 75: Zusammenhang der Bildkoordinaten und 3D-Koordinaten, festgelegt über die Parameter des Kameramodells.

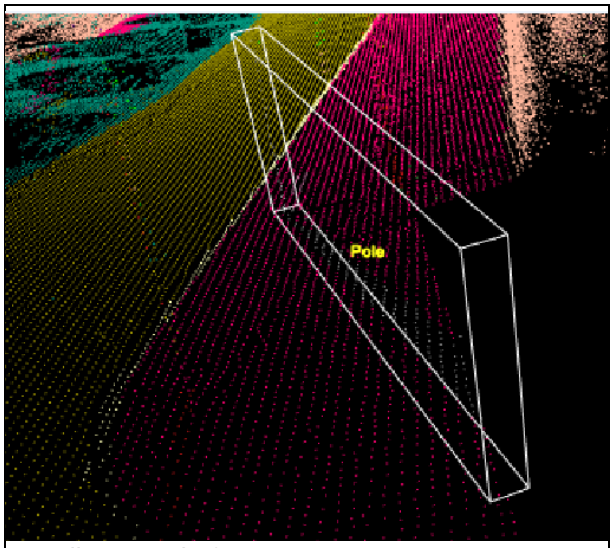
Der Ansatz besteht darin, von den Punkten der Punktwolke aus die Bildstrahlen auf die Pixel des zugehörigen semantisch segmentierten Bildes zu verfolgen. Fallen mehrere Punkte in einen Pixel, erhält derjenige mit dem geringsten Abstand zur Kamera das entsprechende Klassenlabel.

Bereits bei der Rückprojektion der perfekten Ground-Truth-Segmentierungen traten Effekte auf, die bei der Verarbeitung von „echten“ Segmentierungsmasken mit stellenweiser Fehlklassifikationen noch stärker in Erscheinung traten. Um z.B. den Mehrdeutigkeiten bei der Übertragung der Klassenlabel zu begegnen, wurden weitere Algorithmen entwickelt, welche bei der Rückprojektion größere Volumenelemente berücksichtigen konnten.

5.4.4 Herausforderungen bei der Rückprojektion

Verfehlen von Objekten

Bei dem o.b. Algorithmus zeigten sich bereits bei Segmentierungen aus der Ground Truth vor allem bei schmalen, stehenden Objekten, besonders am Bildrand, signifikante Abweichungen zwischen der Segmentierung im 2D-Bild und den passenden Punkten in der Punktwolke: Label werden in der Punktwolke „neben“ ihren eigentlichen Ort zurückprojiziert. Ein Beispiel zeigt Bild 76.



Quelle: Fraunhofer IPM

Bild 76: Mast-Label wurden hier bei der Rückprojektion am eigentlichen Objekt vorbei auf den Gehweg abgebildet (Label aus der manuellen Annotierung; Auflösung der Punktwolke auf 10 cm reduziert; Label farblich codiert).

Mittels einer Anpassung der Kalibrierungsparameter durch visuelle Korrektur der Überlagerung von RGB- und Tiefenbild konnte dieser Effekt ein Stück weit reduziert werden. Dennoch waren bspw. in Bild 82 „False Positives“ zu erkennen, wo Label von Baumstämmen noch auf einen Teil der Dächer von PKW übertragen wurden.

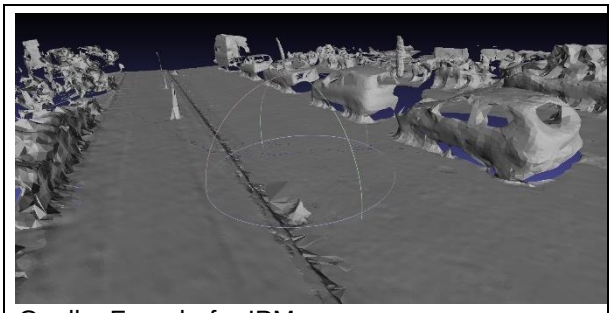
Problematisch war insbesondere, dass die Label, welche nicht das eigentliche Objekt trafen, an anderer Stelle „False Positives“ für die entsprechende Klasse erzeugen konnten. Diesem Effekt wurde durch Anwendung der Statistik entgegengewirkt, wenn Punkte mehrere Label enthielten. Außerdem konnten Parameter bei der Extrahierung einzelner Objektinstanzen bestimmte Arten von „False Positives“ verhindern. Mehr zu beiden Vorgehenswiesen folgt im Weiteren.

Transparenz und Lücken von Objekten in 3D

Objekte erschienen in der Punktwolke „transparent“, d.h., es war zunächst keine Information über den Zusammenhang von Punkten bzw. zu Oberflächen von Objekten vorhanden. Dies verursacht einen ähnlichen Effekt, wie er durch das Verfehlen von Objekten an deren Rand entsteht: Klassenlabel werden fälschlicherweise auf Objekte im Hintergrund übertragen.

Dies fiel vor allem dort ins Gewicht, wo häufige, größere Lücken vorhanden waren, z.B. im Bereich von Vegetation oder Bäumen. Auch auf Verkehrsschildern oder spiegelnden Oberflächen war die Punktdichte nicht immer ausreichend hoch.

Die Anwendung eines Algorithmus zur Vermaschung der Punktwolke hat gezeigt, dass dieser Ansatz keine einfache Lösung für die genannten Probleme bietet. So wurden z.B. je nach Auflösung der Vermaschung kleinere ungewollte Lücken geschlossen, dafür jedoch auch Oberflächen generiert, welche in der Realität nicht vorhanden waren und ihrerseits wieder eine korrekte Rückprojektion verhinderten (vgl. Bild 77).



Quelle: Fraunhofer IPM

Bild 77: Beispiel einer Vermaschung der Punktwolke.

Schließen von Lücken durch Volumenelemente

Eine Möglichkeit, beide bisher genannten Effekte zu reduzieren, wurde in der Verwendung von Volumenelementen gefunden (z.B. Octree-Datenstruktur).

Vom Kameraursprung wird über die Mitte jedes Pixels ein Strahl in die Punktwolke hinein bis zum ersten besetzten Volumenelement verfolgt, das dieser schneidet. In diesem Volumenelement wird der Punkt mit dem Klassenlabel des jeweiligen Pixels markiert, welcher dem Strahl am nächsten liegt. Über zwei Punkte x_1 und x_2 auf dem Strahl berechnet sich der Abstand von Punkt x_0 zum Strahl wie folgt:

$$d = \frac{|(\mathbf{x}_2 - \mathbf{x}_1)\times(\mathbf{x}_1 - \mathbf{x}_0)|}{|\mathbf{x}_2 - \mathbf{x}_1|} = \frac{|(\mathbf{x}_0 - \mathbf{x}_1)\times(\mathbf{x}_0 - \mathbf{x}_2)|}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

Bedingt durch die Octree-Datenstruktur werden hierbei zunächst nur Nachbarn innerhalb des jeweiligen Volumenelements betrachtet, weshalb nicht gewährleistet ist, dass immer derjenige Punkt das Label erhält, welcher sich am nächsten zum verfolgten Strahl befindet. Eine exaktere Übertragung ließe sich über eine zusätzliche radiusbasierte Suche in der Nachbarschaft erzielen. Haben mehrere benachbarte Pixel in der Segmentierungsmaske dasselbe Label, was für korrekte Klassifizierungen i.d.R. gegeben ist, dürfte dieser Effekt jedoch nicht ins Gewicht fallen.

Werden die Volumenelemente zu groß gewählt, kann dies bei einem komplexen Aufbau der Szene die Menge der Punkte, welche Label erhalten, signifikant reduzieren. Es gibt dann nur noch wenige Punkte im Vordergrund, die jeweils viele Label erhalten. Sie erhalten jedoch auch die Label, welche eigentlich zu Objekten im Hintergrund gehören.

Volumen mit einer Seitenlänge von 5 bis 10 cm ergeben bei der Übertragung der Label aus der Ground Truth einen sinnvollen Kompromiss. Schmale Objekte wie die Masten von Schildern werden damit jedoch teilweise noch verfehlt. Für solche Objekte könnte es sinnvoll sein, die Volumengröße klassenspezifisch anzupassen und Punkte, welche z.B. als zur Straße gehörig identifiziert wurden, auszublenden.

Am Ende wurde in der prototypischen Implementierung statt dieses Volumenraster-basierten Ansatzes das geometrisch konsistentere Lochkamera-Modells verwendet.

5.4.5 Höhere Sicherheit durch mehrere Label

Aufgrund der räumlichen Nähe der Kamera-aufnahmen sowie der Verwendung mehrerer Kameras, waren Objekte in der Punktwolke meist auf mehreren Bildern zu sehen. Bei der Rückprojektion erhalten einzelne Punkte in der Punktwolke somit mehrere Label. Fehlklassifikationen in 2D können auf diese Art und Weise in gewissem Rahmen korrigiert werden.

Je mehr Label pro Punkt zur Verfügung stehen, desto sicherer kann eine korrekte Entscheidung getroffen werden. Die Anzahl der Label pro Punkt ist abhängig von der Auflösung der Punktwolke bzw. vom Algorithmus, der für die Rückprojektion verwendet wird. Aufgrund der mit zunehmender Entfernung eines Objektes von der Kamera wachsenden Unsicherheit bei der Klassifizierung sollte außerdem ein Schwellwert oder eine Gewichtung für die Objektentfernung eingeführt werden. Dies reduziert ggf. die Anzahl der Klassenlabel pro Punkt, erhöht aber deren Sicherheit und erlaubt damit eine zuverlässigere Entscheidung für die finale Klassifizierung eines 3D-Punktes.

Die Wahl des endgültigen Klassenlabels wurde als Mehrheitsentscheidung implementiert. Es wäre an dieser Stelle auch denkbar, die Statistik über einen gewissen Nachbarschaftsbereich zu betrachten, oder die Wahl der Label durch einen Filter über die Nachbarschaft nochmals zu korrigieren. Weitere Kriterien für die Entscheidung könnten z.B. die Sicherheit sein, mit der das Pixel vom Neuronalen Netz klassifiziert wurde, der Abstand zur Kameraposition oder a-priori-Wissen über die Wahrscheinlichkeit eines bestimmten Objektes (vgl. Kap. 3.4).

Ergebnis der Rückprojektion der Label war eine Punktwolke, bei der für jeden Punkt die Information XYZ und Klassenlabel verfügbar waren.

5.4.6 Übertragung der RGB-Information

Die Algorithmen, welche für die Rückprojektion der Klassenlabel verwendet wurden, erlaubten auch die Übertragung der RGB-Information eines Pixels auf die Punkte in der Punktwolke. Um möglichst vielen Punkten einen Farbwert zu übertragen, war der auf dem Lochkamera-Modell basierende Algorithmus besser geeignet. Werden mehrere Farbwerte auf einen Punkt abgebildet, bietet es sich an, anstatt einer Mehrheitsentscheidung den Mittelwert der Farbwerte zu bilden. Eine Gewichtung über den Abstand zur Kamera sorgt für eine Reduzierung der durch Kalibrierungsabweichungen entstehenden Fehler.

Die XYZ-RGB-Punktwolke stellt ein „Nebenprodukt“ der Prozessierung dar. Die darin enthaltene Information wurde im Forschungsprojekt nicht für die Klassifizierung verwendet. Es waren jedoch Erweiterungen der Algorithmen denkbar, welche sich diese Information zunutze machen könnten. In den folgenden Abschnitten sind mehrere Beispiele der kolorierten Punktwolke dargelegt.

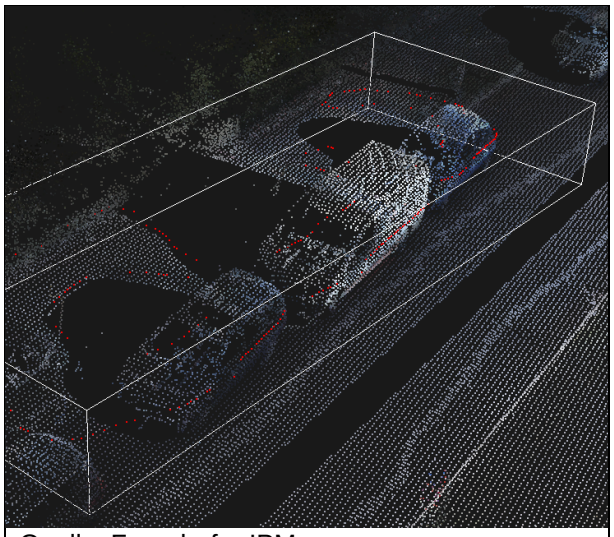
5.4.7 Extrahierung von Objekten in 3D

Um in der Punktwolke eine Menge von Punkten zu einer Instanz eines Objekts zusammenzufassen, waren verschiedene Ansätze möglich, wie z.B. Region Growing oder andere Clustering-Algorithmen. Anhand von Nachbarschafts-beziehungen ist hierbei zu entscheiden, welche

Punkte zu einem einzelnen Objekt gehören. Werden die Parameter zu restriktiv gewählt, werden Objekte zerschnitten; im anderen Fall besteht die Gefahr, z.B. zwei dicht aufeinander geparkte PKW oder eng stehende Schilder nicht voneinander trennen zu können.

Um diese Aufgabe zu lösen, wurde der Euclidean Cluster Extraction-Algorithmus aus der PCL gewählt. Die wesentlichen Parameter waren eine Mindestanzahl von Punkten pro Cluster, sowie ein Abstandskriterium. Dieses sorgte für die Trennung einzelner Objektinstanzen, wenn die nächsten Nachbarpunkte mit demselben Label eine gewisse Distanz überschreiben. Über die Minimalanzahl lassen sich sehr kleine Objekte, welche mit hoher Wahrscheinlichkeit „False Positives“ sind, eliminieren. Die Parameter wurden klassenspezifisch gewählt.

Dieses Vorgehen zeigte zufriedenstellende Ergebnisse für die meisten relevanten Objektklassen. Nicht zuverlässig separiert werden konnten jedoch bspw. Baumkronen, wenn diese ineinander verwachsen waren oder räumlich sehr eng platzierte Objekte ein und derselben Klasse. Bild 78 zeigt die Punktwolke um ein Fahrzeug-Objekt, in dem mehrere PKW nicht korrekt getrennt wurden.



Quelle: Fraunhofer IPM

Bild 78: Bounding Box um mehrere Fahrzeuge, die nicht korrekt separiert wurden (Label aus der manuelle Annotierung; Auflösung der Punktwolke auf 10 cm reduziert).

5.4.8 Repräsentierung von Objekten in 3D

Eine mögliche Repräsentierung der einzelnen 3D-Objekte ergibt sich durch die oben beschriebenen Cluster. Diese können selbst wieder als eigenständige Punktwolken betrachtet werden, in der alle Punkte dasselbe Label haben.

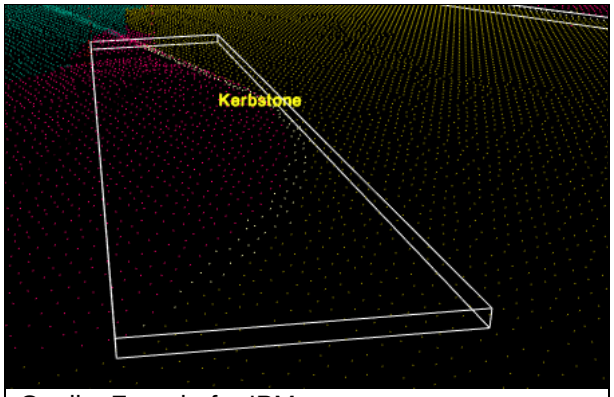
Für die Speicherung der Ergebnisse erschien es sinnvoll, zusätzlich eine Repräsentierung zu definieren, welche kompakter ausfällt, d.h., welche ein Objekt mit weniger Aufwand bezüglich des Speicherplatzes darstellen kann. Der OKSTRA nutzt beispielsweise die Abstraktion bzw. die Reduzierung der Geometrie von Objekten auf Punkte oder Linien.

Bounding Boxes

Eine naheliegende Abstraktion solcher einzelner Objekt-Punktwolken besteht bspw. darin, die acht Eckpunkte einer minimalen Bounding Box zu berechnen, welche diese Punktmenge enthält.

Bounding Boxes können erzeugt werden, indem die beiden stärksten Eigenwerte der Varianz in der Punktwolke bestimmt werden, die das jeweilige Objekt repräsentiert. Über die zugehörigen Eigenvektoren und deren Kreuzprodukt erhält man drei Richtungen, über die Skalierung der Länge mittels des Betrags der Varianz ein gerades Prisma mit einem Parallelogramm als Grundfläche. Ein einfacherer Algorithmus verwendet die minimalen und maximalen Punktkoordinaten der jeweiligen Objekt-Punktwolke, um eine etwas gröbere, parallel an den Koordinatenachsen orientierte Bounding Box zu generieren.

Eine solche Bounding Box ist jedoch keine geeignete Repräsentierung für ausgedehnte Objekte, welche eine Krümmung aufweisen. Bild 79 veranschaulicht dies für einen Bordstein (Kerbstone): In der Box sind nicht nur die Punkte enthalten, die zum Bordstein gehören, sondern auch viele aus dem angrenzenden Gehweg und der Straße.



Quelle: Fraunhofer IPM

Bild 79: Die Repräsentierung als Bounding Box ist für langgestreckte, gekrümmte Objekte wie einen Bordstein wenig geeignet (Label aus der manuellen Annotierung; Auflösung der Punktwolke auf 10 cm reduziert).

Die Ergebnisse der prototypischen Prozessierung wurden deshalb nicht in der Form von Bounding Boxes abgelegt, sondern als 2D-Polygon im Sinne einer konkaven Hülle berechnet (s.u.).

Einzelpunkte, Linien und Polygone

Für Punktobjekte wie bspw. Verkehrsschilder ist es naheliegend, den Mittelpunkt eines Clusters (Mittelung der Koordinaten) zu berechnen und für die Speicherung eines Objekts zu verwenden.

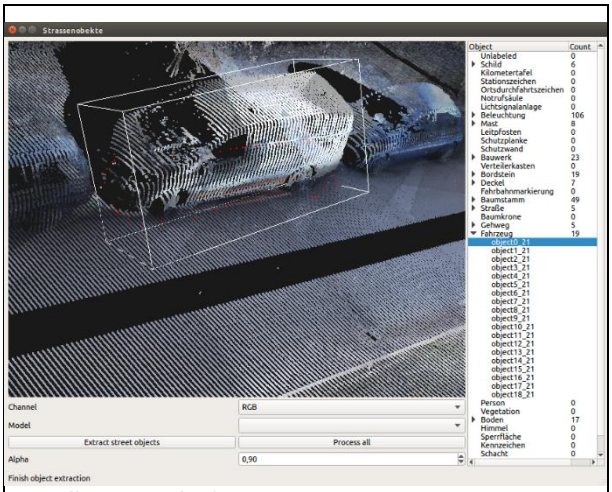
Dieses Vorgehen könnte prinzipiell auch auf Linien- und Polygonobjekte ausgedehnt werden. Hierfür erscheint jedoch die Berechnung einer minimalen Hülle das sinnvollere Vorgehen, aus der in der Folge Linienobjekte oder Polygone um Flächen herum bestimmt werden.

Die Umwandlung von flächenhaften Objekten in Linienobjekte ist nicht eindeutig bestimmt und deshalb nicht trivial zu implementieren. Im Zweifel geht dabei Information über die tatsächliche Form eines Objektes, z.B. auch bei lokalen Deformationen, verloren. Im Forschungsprojekt wurden die Ergebnisse deshalb in ihrer flächenhaften Repräsentierung belassen.

Konkave Hülle

Die konkave Hülle ist geeignet, auch die Form von länglichen, gekrümmten Objekten anzunähern. Hierfür ist ein Parameter notwendig, welcher angibt, wie detailliert die Außenhülle des Objekts gerastert werden soll. Er bestimmt damit die Anzahl der Punkte, welche die Hülle eines Objekts definiert.

Die konkave Hülle kann mithilfe der PCL sowohl in 2D als auch in 3D berechnet werden. Beispiele für eine Rasterung mit einer Auflösung von ca. 10 cm zeigen Bild 80 und Bild 81. Bei Bedarf kann zu dieser Grundfläche z.B. die mittlere oder maximale Höhe des Objekts berechnet und so die Ergebnisse in einer vereinfachten 3D-Repräsentierung als Polygon mit Höhenwert gespeichert werden.



Quelle: Fraunhofer IPM

Bild 80: Die roten Punkte markieren die Projektion der konkaven Hülle des Fahrzeugs auf die zuvor berechnete Straßenebene (Label aus der manuellen Annotierung; Alpha-Parameter 0,9).

Die Auflösung der konvexen Hülle wird über den Alpha-Parameter des Algorithmus zur Berechnung der konkaven Hüllte gesteuert. Um eine sinnvolle Reduktion der Auflösung zu erreichen, muss jedoch zunächst die Auflösung der Punktwolke reduziert werden. Dies wird wiederum durch die Nutzung von Octrees in der Software realisiert.

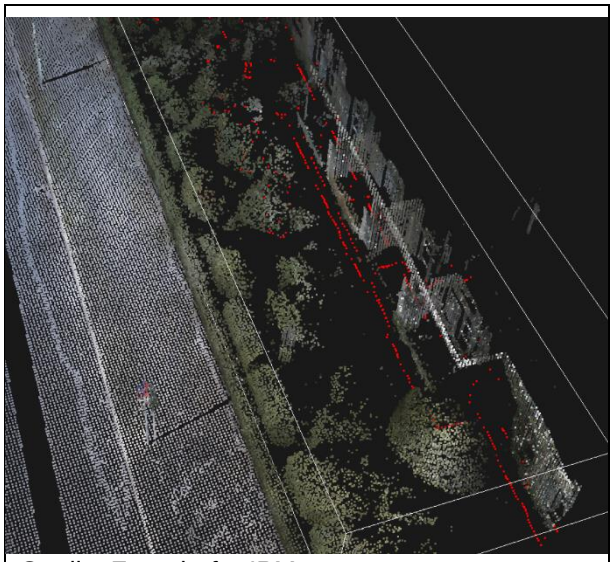
Quelle: Fraunhofer IPM

Bild 81: Bounding Box (weiß) und konkave Hülle (rote Punkte) eines Bordsteins (Label aus der manuellen Annotierung; Alpha-Parameter 0,9; Punktwolke von unten betrachtet).

Bezug zur Straßenebene

Wird zu der oben beschriebenen Vorgehensweise noch eine Straßenebene berechnet, kann die Länge der senkrechten Projektion des höchsten Punktes im Cluster als Höhe des Objekts mit abgelegt werden. Die senkrechte Projektion der Eckpunkte

eines Polygons auf die Straßenebene ergibt einen Polygonzug, der einen ebenen Grundriss des Objektes beschreibt.



Quelle: Fraunhofer IPM

Bild 82: Die roten Punkte markieren die Projektion der konkaven Hülle des Bauwerks auf die zuvor berechnete Straßenebene (Label aus der manuellen Annotierung; Auflösung der Punktwolke auf 10 cm reduziert; Alpha-Parameter: 0,9).

Diese Vorgehensweise musste für steile Straßenanstiege angepasst werden, für welche die senkrechte Projektion zu Abweichungen von den tatsächlichen Verhältnissen führen würde.

In der prototypischen Software zum Forschungsprojekt wurde dieser Fall nicht gesondert behandelt. Die Software berechnet im geladenen Datensatz die Straßenebene aus dem größten Cluster, welches als einzelnes Objekt „Straße“ identifiziert wurde. Bei größeren Datensätzen sollte deshalb zusätzlich berücksichtigt werden, dass eine Ebene gewählt wird, welche sich ausreichend nahe am jeweiligen Objekt befindet.

5.4.9 Georeferenzierung und Verortungsgenauigkeit

Da die Punktwolke georeferenziert vorlag, war eine Verortung der Ergebnisse im Idealfall mit der Genauigkeit des Postprocessing, das für die Punktwolke durchgeführt wurde, erreichbar.

Eine Reduzierung der maximal erreichbaren Genauigkeit ergab sich jedoch, wenn durch die Nutzung von Octrees bzw. durch eine Rasterung der Punktwolke systematische Offsets in der Größenordnung der Kantenlänge des kleinsten betrachteten Volumens entstanden. Dies wurde bereits in Kap. 5.1.3 thematisiert.

Auch die Offsets, welche durch Abweichungen in der Kalibration bei der Rückprojektion entstehen, können zur Fehlplatzierung von Objekten in der Punktwolke bzw. zu False Positives führen. Die korrekte Bestimmung der Ausdehnung von Objekten wird durch diesen Effekt ebenfalls erschwert. Dieser Effekt zeigt sich z.B. durch die Verlängerung des Objekts durch seinen „Schatten“ in Bild 76.

Eine Optimierungsmöglichkeit wäre möglicher Weise ein klassenspezifisches Postprocessing, in dem z.B. Meta-Wissen genutzt wird: So weisen ein Schild und der zugehörige Mast i.d.R. keine solche horizontale Ausdehnung auf.

5.4.10 Klassifizierte Objekte in 3D

Mit den durch das FCN-Netzwerk generierten Segmentierungen, dessen Training in Kap. 5.3 beschrieben wurde, erfolgte die beispielhafte Prozessierung eines 200 m langen Teilstücks einer außerorts befindlichen Straße.

Die Übertragung der Klassenlabel wurde anhand des auf dem Lochkamera-Modell basierenden Algorithmus durchgeführt (vgl. Kap. 5.4.3). Es wurden die Segmentierungen der beiden nach vorne sowie der nach hinten gerichteten Kamera verwendet. Bei der Übertragung der Labelinformation erhielten 96,5 % aller Punkte in der Punktwolke mindestens ein Klassenlabel. Im Durchschnitt erhielt jeder dieser Punkte 6,4 Klassenlabel, mit denen die Mehrheitsentscheidung für das endgültige Label des Punktes getroffen wurde.

Im Folgenden werden einige Ergebnisse der Rückprojektion der Klassenlabel und der Objektdetektierung in 3D anhand der in Bild 83 abgebildeten Kreuzung qualitativ dargestellt.

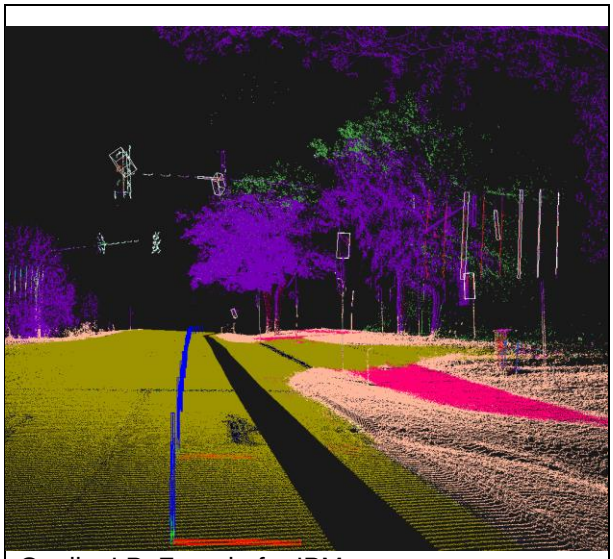
Quelle: LP

Bild 83: Ausschnitt eines CCD-Bildes, dessen semantische Segmentierung durch das FCN-Netzwerk für die Objekterkennung in der Punktwolke verwendet wird.

Aus Bild 84 wird anhand der weißen Bounding Boxes ersichtlich, dass drei der sichtbaren Instanzen der Objektklasse Schild korrekt erkannt wurden. Die „False Positives“ im Bereich der Lichtsignalanlagen sind auf Fehlklassifikationen in den 2D-Segmentierungen zurückzuführen, welche nicht durch die Mehrheitsentscheidung pro Punkt in der Punktwolke „überstimmt“ wurden. Der Clustering-Algorithmus, der die Klassifikation und Extraktion von Objekten in der Punktwolke fortsetzt, arbeitete bei dieser Rückprojektion ohne einen Mindestwert für die Anzahl von Punkten mit demselben Label in der unmittelbaren Nachbarschaft. Eine geeignete Parametrisierung des Clustering kann solche „False Positives“ in gewissem Umfang reduzieren.

Die Darstellung der Schilder in der Punktwolke zeigt, dass die Klassifizierung durch die Labelinformation aus den segmentierten Bilddaten wesentlich erleichtert wird. Ohne diese Information wäre es kaum möglich, anhand der wenigen Scanprofile, die auf einem Schild aufgezeichnet werden, eine zuverlässige Unterscheidung z.B. zwischen einer Lichtsignalanlage, einer Beleuchtungseinrichtung oder sonstigen schmalen, aufrecht stehenden Objekten zu treffen.

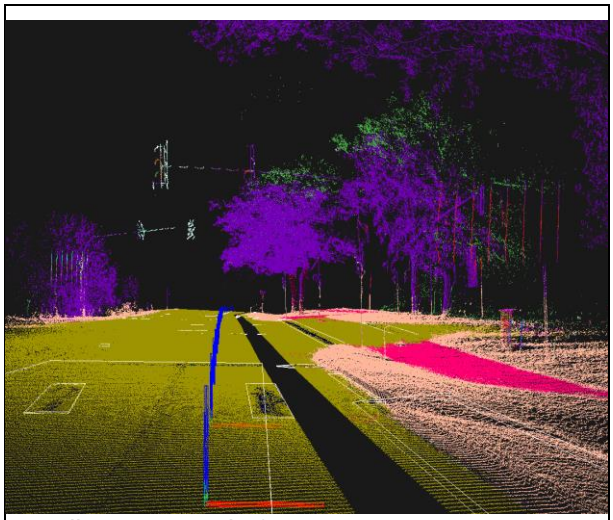
Es zeigt sich im rechten Bildbereich, dass der Clustering-Algorithmus für größere Schilder noch angepasst werden musste, da die einzelnen vertikalen Scanprofile als separate Instanzen von Schildern extrahiert wurden.



Quelle: LP; Fraunhofer IPM

Bild 84: Bounding Boxes um die Instanzen der Objektklasse Schild an derselben Kreuzung (Punktwolke anhand der Label der Punkte eingefärbt).

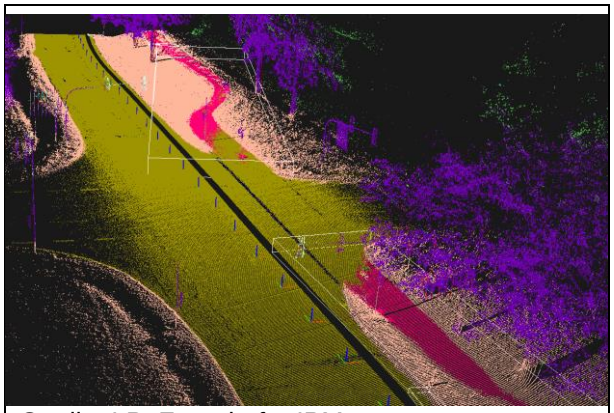
Die in diesem Farbschema schwarz eingefärbten Punkte mit Label Fahrbahnmarkierung sind in Bild 85 mit weißen Bounding Boxes umrissen.



Quelle: LP; Fraunhofer IPM

Bild 85: Bounding Boxes um die Instanzen der Objektklasse Fahrbahnmarkierung (Punktwolke anhand der Label der Punkte eingefärbt).

Hervorzuheben ist, dass die Objektklasse Gehweg relativ zuverlässig von der Straße unterschieden werden konnte, obwohl sie visuell sehr ähnlich erscheinen. Dies ist in Bild 86 sichtbar, wo die mit Label „Straße“ versehenen Punkte bräunlich, die mit „Gehweg“ gelabelten Punkte pink eingefärbt sind. Am direkten Übergang sieht man jedoch auch, dass der Übergang nicht ganz exakt getroffen wurde.



Quelle: LP, Fraunhofer IPM

Bild 86: Bounding Boxes um die Instanzen der Objektklasse Gehweg (Punktwolke anhand der Label der Punkte eingefärbt).

Bild 87 zeigt dieselbe Szene mit der Punktwolke, die durch die RGB-Information der Kamerabilder eingefärbt wurde.

Quelle: LP, Fraunhofer IPM

Bild 87: Bounding Boxes um die Instanzen der Objektklasse Gehweg (Punktwolke anhand der RGB-Information eingefärbt).

5.4.11 Batch-Prozessierung

Da die Punktwolke zusammen mit den Bilddaten eine erhebliche Menge an zu verarbeitenden Daten darstellt, ist es notwendig, die Prozessierung jeweils auf Teilstücken einer befahrenen Strecke durchzuführen. Für das Forschungsprojekt wurde die Länge dieser Einheiten auf 50 m festgelegt. Die Anzahl an 3D-Punkten in einem solchen Abschnitt bewegt sich, je nach Befahrungsgeschwindigkeit, zwischen ca. 3 und 15 Mio. Punkten. In Extremfällen auch mehr, z.B. wenn die Erfassung des Laserscanners bei Stillstand des Messfahrzeugs nicht unterbrochen wird.

Die Prozessierung erfolgte in einem prototypischen Programm, welches mit bestimmten Kommandozeilenparametern gestartet wird. Darüber hinaus war keine Benutzerinteraktion erforderlich. Die Software ist ohne grafisches Benutzerinterface implementiert, was die Auslagerung eines solchen Prozesses auf leistungsfähige Hardware erleichtert, um auch längere Strecken in vertretbarem Zeitaufwand prozessieren zu können.

Die Syntax des Programmaufrufs und die für die Prozessierung erforderliche Settings-Datei pro Befahrungsordner wurden in der Dokumentation zur Auslieferung der prototypischen Software näher beschrieben.

5.4.12 Überführung der Daten in eine Datenbank

Für die Übertragung der georeferenzierten Messdaten in eine Datenbank wurde auf die Bibliothek libpq.dll zurückgegriffen. Diese Bibliothek ermöglichte einen direkten Zugriff auf eine PostgreSQL Datenbank. Der entsprechende Teil des Prototyps wurde in den PhoML -Viewer integriert. Dieser ermöglicht die Interaktion mit den Bilddaten und mit den Tiefenbildern. Durch die georeferenzierte Messfunktionalität diente dieser zur Überprüfung und Rücktransformation der Objekte.

Voraussetzung für die Datenintegration war eine bestehende PostgreSQL-Datenbank mit einem intrigierten PostGIS-Template. Dieses Template enthält die Koordinaten und Geometriedefinitionen für eine georeferenzierte Abbildung und Transformation von georeferenzierten Daten.

Die Datenbanktabellen werden automatisch bei der ersten Verwendung erzeugt. Somit werden Benennungsfehler oder Zugriffsfehler auf die jeweiligen Spalten vermieden. Die erstellten Tabellen können beliebig erweitert werden, da der Zugriff auf die Spalten per Name und nicht per Indizes erfolgt.

Folgende Datenbankobjektarten wurden für den Prototyp festgelegt: Unterschieden wird zwischen PUNKT, LINIE und POLYGON. Es wurden 3D-Geometrien verwendet. Die sind zwar ungeeignet für eine 2D-GIS-Bearbeitung, erfüllen aber die Vorgabe einer 3D-Geometrie. Dies bedeutet, es werden die Stützpunkte des abzubildenden Objektes in allen drei Dimensionen gespeichert.

Die Festlegung der endgültigen Datenbankstruktur steht noch aus.

5.4.13 Erstellung des OKSTRA-konformen XML-Schemas

Die Umsetzung zur Erzeugung eines OKSTRA-konformen Schemas wurde in dem Prototyp Elementweise umgesetzt. Folgende minimal Tags / Knoten wurden extrahiert um ein punktuelles georeferenziertes Objekt in einem Schema abzubilden.

```
<gml:FeatureCollection gml:id="INT_ID"…>
 <gml:featureMember>
  <okstra:allgemeines_Punktobjekt gml:id="IN_ID">
   <okstra:Punktgeometrie>
    <gml:Point gml:id="P.XXXX">
     <gml:pos srsDimension="3">
      EASTING NORTHING HEIGHT
       </gml:pos>
    </gml:Point>
   </okstra:Punktgeometrie>
   <okstra:Punktnummer>11
   </okstra:Punktnummer>
  </okstra:allgemeines_Punktobjekt>
 </gml:featureMember>
</gml:FeatureCollection>
```

Dieses Beispiel zeigt einen einzelnen Punkt mit der der Punktnummer 11. Alle weiten Attribute wurden, wie in Kap.2.3, beschrieben noch erweitert. Die automatisch generierten Objekte wurden einzeln in der jeweiligen Rohdaten-Ordnerstruktur in das XML Schema umgewandelt und gespeichert.
Zur Kontrolle der generierten XML-Daten dient die Software OKSTRA-Werkzeug 64 Bit in der Version 1.3.0.36 von der OKSTRA Webseite.

6 Manuelle Auswertung

Die klassische Auswertung der Objekte erfolgte entlang der automatisch erfassten, prozessierten oder trainierten Strecken. Hierbei wurde unterschieden, ob die Strecke bereits als Trainingsdaten Anwendung fand oder es eine für das neuronale Netz unbekannte Strecke war. Bei bekannten Strecken wurde die direkte Lageabweichung mit bereits herkömmlich erfassten Daten abgeglichen. Hierbei zeigten sich die Abweichungen aus der Rückprojektion und der Auswertung der mehrfach erfassten Objektpunkte durch ein Label.

Fehlereinflüsse bei korrekten Labels (Trainingsdaten) waren:

- Fehler in der Rückprojektion (Lageversatz).
- Mehrfachlabelung eines Objektpunktes kommt zu keinem sinnvollen Ergebnis (keine eindeutige Objekt-ID in Überlappungsbereichen).
- Fehlerhafte Label (durch bewegte Objekte oder fehlerhafte Trainingsdaten).

6.1 Kontrolle der Objektdaten

Die Kontrolle der gefundenen Objektdaten erfolgte über einen händisch digitalisierten Datensatz. Ein Teil dieses Datensatzes war im Bild zu sehen. Hier wurden alle relevanten Objekte in einer SHAPE-Datei georeferenziert erfasst. Die Lage dieser Objekte sollte anschließend mit den automatisch gefundenen Objekten verglichen werden und eine statistische Grundlage über ordnungsgemäß gefundene Objekte bilden.

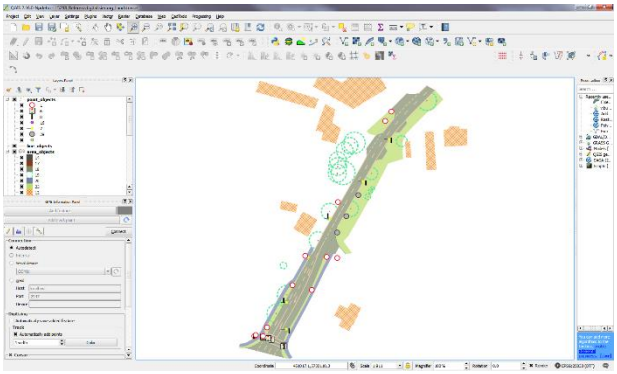


Bild 88: Digitalisierte Referenzstrecke mit allen zu erfassenden Objekten, Visualisierung in QGIS.

Der Vergleich wurde im Weiteren über eine Datenbank durchgeführt.

Bild 89: Referenzdatensatz mit hinterlegtem Luftbild

7 Statusbericht: Prototyp und Klassifizierung

7.1 Portierung des Prototyps auf Windows

Frameworks für Deep Learning werden in der Regel primär für Linux-basierte Betriebssysteme entwickelt, da sie oft aus forschungsnahen Initiativen hervorgegangen sind. Für das Forschungsprojekt war jedoch die Lauffähigkeit unter dem Betriebssystem Windows erforderlich, was eine Portierung der Anwendung erforderlich machte.

Diese Portierung war mit erheblichem Aufwand verbunden. Die wichtigsten Eckdaten und Erkenntnisse werden im Folgenden dargestellt. Eine ausführliche Dokumentation erfolgte im Rahmen der Übergabe von Code und Software.

7.2 Entwicklungsumgebung und Zielplattform

Als Entwicklungsumgebung kam Microsoft Visual Studio 2013 unter Windows 7 Enterprise (64 Bit) zum Einsatz. Das aktuelle Visual Studio 2017 wurde als Entwicklungsumgebung verworfen, da die Herausgeber von Caffe diese Version noch nicht offiziell unterstützten.

Zielplattform für die im Forschungsprojekt erstellte Software war Windows 7 in der 64 Bit-Version. Da die Software für 64 Bit-Systeme kompiliert wurde, ist sie auf einem 32-Bit-Windows nicht lauffähig.

Für das Training des neuronalen Netzes war eine CUDA-fähige Grafikkarte mit einer leistungsstarken GPU erforderlich. Um die Karte nutzen zu können, waren verschiedene Entwicklungsbibliotheken wie CUDA und cudnn von NVIDIA zu installieren.

7.3 Abhängigkeiten: Caffe

Das Caffe-Framework bietet die Möglichkeit, die benötigten Bibliotheken über einen automatischen Build-Vorgang zu erstellen. Um diesen Build-Vorgang anzustoßen, werden im Code-Repository von Caffe entsprechende Skripte mitgeliefert.

Der Quellcode des Windows-Branches von Caffe wurde in der am 11.07.17 heruntergeladenen Version verwendet.

Die Abhängigkeiten von Caffe umfassen:

- CUDA (NVIDIA)
- cuDNN (NVIDIA)
- glog (C++ “Google Logging Module”, https://github.com/google/glog)
- openblas (“An optimized BLAS library”, www.openblas.net, with LAPACK)
- hdf5
- gflags (formerly Google Commandline Flags, https://github.com/gflags/gflags)
- leveldb (Google, “LevelDB is a fast key-value storage library”)
- lmdb (Lightning Memory-Mapped Database))
- protobuf (Protocol Buffers - Google's data interchange format, https://github.com/google/protobuf)
- snappy (compression/decompression)
- Boost (>= 1.55, hier 1.61)
- opencv (>= 2.4, hier 3.1.0)
- zlib (caffezlib.dll, caffezlib1.dll)
- Python (2.7)

Außerdem wurden weitere DLLs mitgeliefert, die zur Ausführung benötigt werden:

- libgcc_s_seh-1.dll
- libfortran-3.dll
- libquadmath-0.dll
- msvcp120.dll
- msvcr120.dll

Zielführend war die Berücksichtigung von vorkompilierten „Binaries“, die von Caffe jedoch nur für bestimmte Kombinationen von Compiler, Nutzung von CPU oder GPU und verschiedene CUDA-Versionen angeboten werden.

7.4 Abhängigkeiten Prototyp

Die Software, die im Rahmen des Forschungsprojektes erstellt wurde, hat eigene Abhängigkeiten. Diese umfassen neben Caffe:

- CUDA (NVIDIA, 8.0.62)
- cuDNN (NVIDIA, 6.0.21)
- Qt

- Boost
- QHull
- VTK (Konfiguration mit Qt) für Visualisierungen
- Point Cloud Library (PCL, mit VTK, Qt, QHull, Visualization-Modul muss mit kompiliert werden, 1.8.0)
- eigen3
- flann
- opencv (Konfiguration mit Qt, 3.2.0)
- zlib
- Perl (z.B. Strawberry Perl, benötigt für pkg-config-Skripte)
- OpenGL 2 für Visualisierungen
- (OpenCV contrib, optional)
- (Google Test, für Debug-Build)

  (gcov, für Debug-Build)
- (gdal, optional für spezielle Ergebnisschnittstellen)

7.5 Hinweise zur Konfiguration der Bibliotheken und Software

Bei der Kompilierung der Bibliotheken unter Windows sind verschiedene, auf den Herstellerseiten teilweise in der Dokumentation kaum aufzufindende, Probleme aufgetreten.

PCL

Für die Kompilierung müssen die Präprozessor-Definitionen sowie die Compiler- und Linker-Flags, wie sie im CMake-Buildsystem vordefiniert sind, angepasst werden. Wird dies unterlassen, ist die Bibliothek zwar kompiliert, es treten jedoch im Prototyp zu einem späteren Zeitpunkt schwer zu interpretierende Laufzeitfehler auf.

- Präprozessor-Define: „PCL_NO_PRECOMPILE“
- Compiler Flag „/bigobj“

Diese Änderungen können im CMake-Buildsystem erfolgen; es ist dann jedoch in der nachfolgend erzeugten Visual Studio-Solution nochmals zu kontrollieren, ob die Änderungen wie gewünscht übernommen wurden. Andernfalls können die Änderungen auch dort vorgenommen werden.

Prototyp

Im Release-Build sind folgende Linker-Flags erforderlich:

- Compiler-Flag „/EHsc“
- Linker Flag: „/NODEFAULTLIB:msvcrtd.lib“

Diese Flags sind zwar im Buildsystem Cmake angelegt, werden jedoch von Visual Studio 2013 nicht zuverlässig interpretiert und müssen deshalb manuell kontrolliert und ggf. gesetzt werden.

8 Vergleich und Schluss-folgerungen

8.1 Manuelle Auswertung der Teststecke

Die Teststrecke wurde in Abstimmung mit den AG ausgewählt und am 30.10.2017 befahren. Es wurden Teilstücke der BAB A3 und A4, der Bundesstraßen B55 und B484 und mehrere Kreisstraßen befahren. Enthalten waren auch 3 Ortsdurchfahrten.

Die Teststrecke hat eine Länge von ca. 46 km. Für die Aufnahme wurde das Messfahrzeug IRIS12 eingesetzt. Dieses ist mit vier CCD Umfeld Kameras und zwei Laserscannern ausgestattet. Die erfassten Rohdaten hatten eine Größe von ca. 200 GB. Im Zuge der Messung wurden rund 38000 Bilder und 2,8 Milliarden Laserscannerpunkte erfasst.

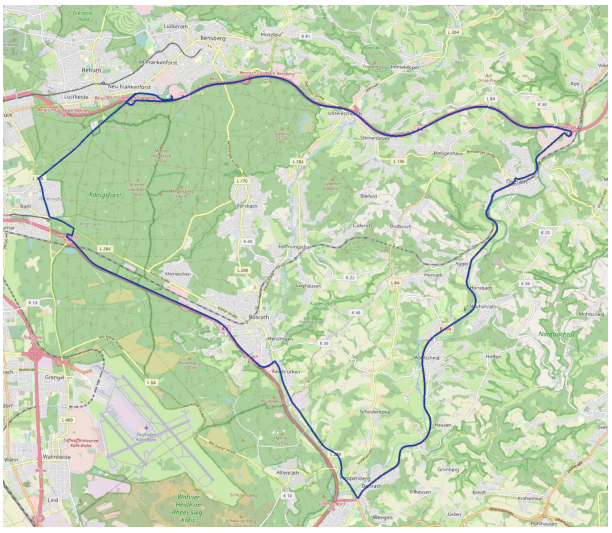


Bild 90: 46km lange Teststrecke im Raum Bergisch Glattbach

Die Aufnahme erfolgte als Rundkurs, ohne Unterbrechung der Messung, sodass eine vollständige Abbildung der Teststrecke möglich war. Die Auswertung der Trajektorien ergaben eine mittlere Standardabweichung der Trajektorie von 4 cm.

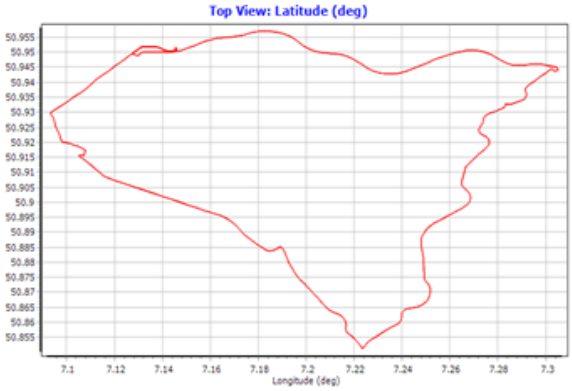


Bild 91: Ausgewertete Trajektorie des Messfahrzeuges IRIS12

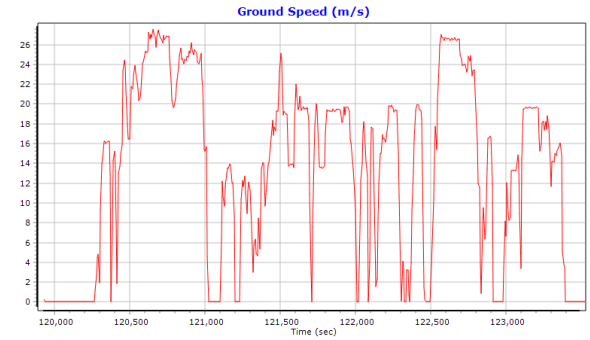


Bild 92: Genauigkeiten der Testfahrt über die Zeit in Meter, rot –> Nord-Positionsfehler RMS [m], Schwarz -> Ost–Positionierungsfehler RMS [m] und grün Höhenfehler RMS [m]

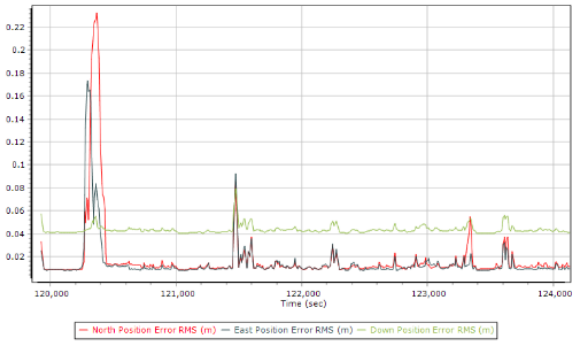


Bild 93: Befahrungsgeschwindigkeit der Teststrecke über die Zeit

Die Digitalisierung erfolgte auf Grundlage der georeferenzierten Orthobilder der Punktwolke. In einem externen Prozess wurden die 3D Punktwolken in ein planares Grauwertbild transformiert und als georeferenziertes Orthophoto gespeichert. Die Pixelauflösung betrug 1cm und

jeder Teilbereich hatte eine Größe von ca. 10 x 20m.

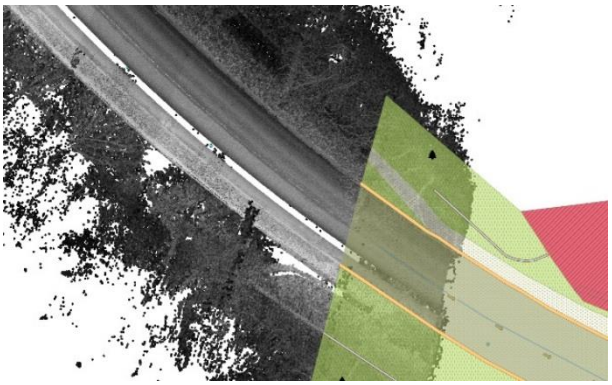


Bild 94: Referenzdatensatz mit georeferenziertem Orthophoto aus Laserscanner Daten

Auf Grundlage dieser Befahrung wurde eine Referenzdigitalisierung erstellt. Die erfassen Objekte sind identisch mit den zu detektierenden Objekten des Neuronalen Netzes. Zur Digitalisierung wurde das Geoinformationssystem QGIS (Version 10) verwendet auf Basis einer Postgres Datenbank.

Die Digitalisierung erfolgte in einem Korridor von 20m entlang der Trajektorie. Das entspricht dem Bereich, in dem das Neuronale Netz aufgrund der Dichteverteilungen in der Punktwolke Objekte detektieren kann. Die erfassten Objekte wurden mit einer Label-ID versehen, welche Rückschlüsse auf die Art des Objektes gibt. Die Referenzobjekte wurden in drei Tabellen, in Bezug auf den jeweiligen Geometrietypen, aufgeteilt (siehe Tabelle 7).

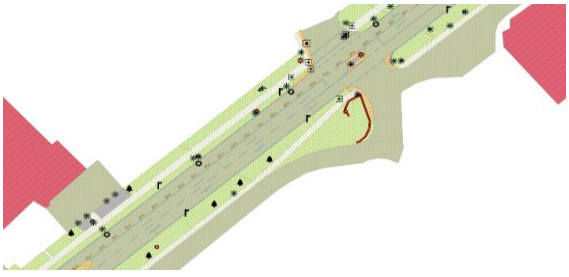


Bild 95: Referenzdigitalisierung Innerorts (Geometrietypen: POINTS, LineSTRING, POLYGON)

Die Darstellung und Editierung erfolgte ebenfalls in QGIS. Das Programm kann verschiedene Datenbanken und Websevices visualisieren.

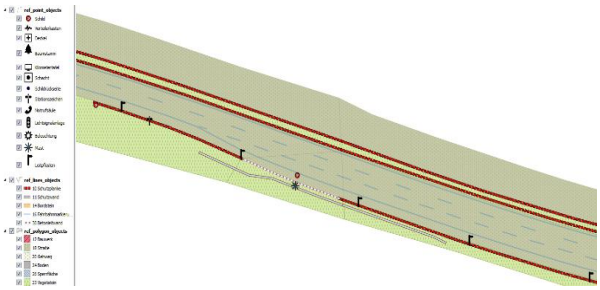


Bild 96: Referenzdigitalisierung BAB A4 (Geometrietypen: POINTS, LineSTRING, POLYGON).

Tabelle 7 Zuordnung der Repräsentation in der Datenbank für die Referenzobjekte

ID	Objekttyp	Geometrie/WKT
255	Unlabeled	-
1	Schild	Point
2	Kilometertafel	Point
3	Stationszeichen	Point
4	Ortsdurchfahrtszeichen	Point
5	Notrufsäule	Point
6	Lichtsignalanlage	Point
7	Beleuchtung	Point
8	Mast	Point
9	Leitpfosten	Point
10	Schutzplanke	MultiLinestring
11	Schutzwand	MultiLinestring
12	Bauwerk	Polygon
13	Verteilerkasten	Punkt
14	Bordstein	MultiLinestring
15	Deckel	Punkt
16	Fahrbahnmarkierung	MultiLinestring
17	Baumstamm	Punkt
18	Straße	Polygon
19	Baumkrone	Polygon
20	Gehweg	Polygon
21	Fahrzeug	-
22	Person	-
23	Vegetation	Polygon
24	Boden	Polygon
25	Himmel	-
26	Sperrfläche	Polygon
27	Kennzeichen	-
28	Schacht	Point
29	Schildrückseite	Point
30	Betonleitwand	MultiLinestring

```
CREATE TABLE ref_polygon_objects
(
  gid serial NOT NULL,
  label_id integer,
  label_name text,
  the_geom geometry,
  height double precision,
  comment text,
  CONSTRAINT ref_polygon_objects_pkey PRIMARY KEY (gid),
  CONSTRAINT enforce_dims_the_geom CHECK (st_ndims(the_geom) = 2),
  CONSTRAINT enforce_geotype_the_geom CHECK (geometrytype(the_geom) = 'MULTIPOLYGON'::text OR the_geom IS NULL),
  CONSTRAINT enforce_srid_the_geom CHECK (st_srid(the_geom) = 25832)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ref_polygon_objects
  OWNER TO postgres;

-- Index:
```

Bild 97: Datenbank Tabelle für die Referenzobjekte für die geometrische Repräsentation - Polygon - SQL

Die digitalisierten Objekte wurden weiterhin mit den Luftbildern DOP20 des WMTS Service des Landes NRW abgeglichen. Der DOP20 Link für die Einbindung in einem Geoinformationssystem lautet:

https://www.wmts.nrw.de/geobasis/wmts_nw_dop20)

Weitere Informationen sind unter folgendem Link erhältlich:

https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/webdienste/geodatendienste/index.html

Das Luftbild wurde zusammen mit den CCD Bildern des Messfahrzeuges zur besseren Orientierung verwendet. Bild 98 zeigt die deckungsgleiche Referenz der Digitalisierung mit den Luftbilderzeugnissen des oben genannten Webservices. Die Genauigkeit der Digitalisierung betrug ca. +/- 10cm. Einzige Einschränkung waren die ersten 1,5km der Testmessung. In diesem Bereich war die Abweichung zwischen Luftbild und Messung des Mobile Mapping Fahrzeuges > 30cm. Dies wurde auf den dortigen sehr schlechten GPS Empfangs zurückgeführt. Nach Erreichen der Autobahnauffahrt zur BAB A4 entsprach die Genauigkeit wieder ca. +/-10cm (siehe Bild 98).

Es konnten nicht alle Objektklassen für den Referenzdatensatz digitalisiert werden. Dies betraf Objektklassen, die nur temporär vorhanden waren (bspw. Personen und Fahrzeuge), und Objektklassen, die nicht in den Daten des Laserscanners sichtbar waren (bspw. Objektklasse Himmel). Die Objektlasse „Schilderrückseite“ wurde mit in die Objektklasse 1 „Schild“ integriert. Diese Objektklasse diente lediglich zur Unterscheidung eines Schildes für das Neuronale Netzwerk.

Bild 98: Luftbild DOP20 überlagert mit der Referenzdigitalisierung, BAB A4 (rot – Schutzplanke, gelbe Fläche – Fahrbahn, grün Fläche – Vegetation – gelbe Linie – Bordstein)

Bild 99: Luftbild DOP20 überlagert mit der Referenzdigitalisierung, BAB A4 mit Schutzwand (rot – Schutzplanke, gelbe Fläche – Fahrbahn, grün Fläche – Vegetation, beige gepunktete Linie – Schutzwand)

Die Referenzdigitalisierung umfasste 2100 Punktobjekte, 3650 Linienobjekte mit einer Gesamtlänge von ca. 4,5 km und 1772 Polygone mit einer Gesamtfläche von ca. 20 Hektar.

8.2 Aufbereitung der Extraktions-ergebnisse zum Import in die Datenbank

Die Ergebnisdaten aus dem Neuronalen Netz wurden in die Ordnerstruktur der Messdaten mit eingegliedert. Alle Messdaten wurden, wie in Kap. 5.1.1 beschrieben, im Ordner „objects“ gespeichert. Zur besseren Prozessierung wurde die Auswertestrecke in Teilbereiche von 50 m aufgeteilt. Der Teilbereich basierte auf den prozessierten Punktwolken des Messfahrzeuges, welche in einem Abstand von 10m exportiert wurden. Grundlage für diese Stationierung bildet hier der im Messfahrzeug verwendete Odometer.

Jeder 50 m Teilbereich enthielt alle validierten und extrahierten Objekte in einen weiteren Ordner, weiterhin eine fortlaufende Objekt ID und eine Label ID welches der gefunden Objektklasse entsprach.

Im jeweiligen Objektordner befanden sich 2 Textdateien. Die erste beschrieb das extrahierte Objekt näher. Hier wurden Label ID und das Umringpolygon für den Import in die Datenbank gespeichert.

ObjektTyp=449545556

ObjektID=0

LabelID=10

ProjektString=P15298_Referenzstrecke

Punktwolke=punktwolke_0_10_object0_10.txt

Postgis='POLYGON((7.2000012400247888
50.951652225647628,7.1999674136539191
50.951665931968321,7.199958075331943
50.951669416733289))'

Bild 100: Ergebnis Datei mit Attributen und Umring Polygon.

Die zweite Datei enthielt die extrahierte Punktwolke des Objektes. Des weiten wurden auch die Farbinformationen aus den Flächensensoren gespeichert. Das Format dieser Datei wurde wie folgt definiert:

Northing [m] Easting [m] Height[m] Rot[8bit] Grün[8bit] Blau[8bit]

Durch diese Struktur konnte das Umringpolygon mit der Punktwolke und der Objekt ID eindeutig zugeordnet werden.

Zum Vergleich mit den Referenzdaten und auch in Hinblick auf die OKSTRA konforme Speicherung der Daten mussten die Umringpolygone noch in den jeweiligen Geometrietyp transformiert werden. Hierzu wurden folgende nachfolgend aufgeführte mathematische Umwandelungen vorgenommen.

Für die Transformation der Punktobjekte wurde ein Schwerpunkt aus den einzelnen Punkten des Umringpolygons berechnet.

$$s = \frac{1}{m}\sum_{i=0}^{m} x_i$$

Diese Repräsentation wurde zusammen mit allen verfügbaren Attributen in die Datenbank übernommen.

Für die Transformation der Linienobjekte wurde ein „2D Straight Skeleton“ Algorithmus verwendet. Dieser Algorithmus wird definiert über eine schrittweise Erosion des Polygons. Die Seiten des Polygons werden hierbei parallel verschoben. Durch die Verbindung der neu entstandenen Polygonpunkte jeder Erosionsstufe wird die Linienstruktur des Ergebnisses gebildet.

Dieses Ergebnis wurde mit einem Douglas-Peucker-Algorithmus gefiltert und in die Datenbank übernommen. Wie im Bild 102 zu sehen ist, wurde der zuvor als Polygon detektierte Bereich, durch eine Linie generalisiert.

Bild 101: Extrahierte Linienobjekte (rot) aus Umring-Polygonen (grün) (Objektklasse -Fahrbahnmarkierung)

Die weiteren Objektklassen mit einer flächenhaften Ausprägung wie „Fahrbahnoberfläche“ oder „Gehweg“ mussten zur Validierung der Daten auch angepasst werden.

Die Umringpolygone, welche aus der Prozesskette des Neuronalen Netztes entstehen, wurden aus den automatisch extrahierten und gruppierten Daten berechnet. Hierbei entstanden zum Teil große Überlappungsbereiche der einzelnen Objekt-klassen, welche sich in den Umringpolygonen wiederspiegeln. Zur Validierung der Daten sollten keine überlappenden Bereiche entstehen, da in diesen Bereichen keine eindeutige Zuordnung erfolgen kann. Im Bereich von extrahierten Vegetationsflächen und Gehwegflächen wurde dies besonders deutlich. Die Vegetation wird z.B. als

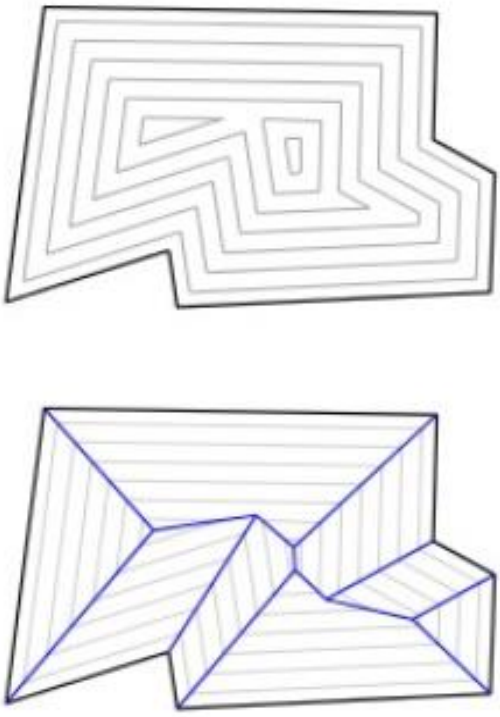


Bild 102: Grafische Darstellung des „2D Straight Skeleton“ Algorithmus, Quelle: Greenwood Mapping Data: 2D Straight Skeleton algorithm in SFCGAL - s C++ wrapper library around CGAL

Punktwolke neben der Straße korrekt detektiert. Im Anschluss folgte ein 1m breiter Gehweg und eine Vegetationsfläche. Bei der Gruppierung der Objekte auf Basis der Punktwolke wurden beide räumlich getrennten Vegetationsflächen durch einzelne Punkte auf dem Gehweg, die als Vegetation extrahiert wurden, zusammengefasst. Dies entstand durch kleine Pflanzen zwischen den Gehwegplatten oder durch eine falsche Extraktion. Durch die Zusammenfassung der beiden Vegetationsflächen entstand somit für den Gehweg ein doppeltes Extraktionsergebnis: Zum einen „Vegetation“ und zum anderen „Gehweg“. Dies obwohl die Repräsentation des Punktwolken-ausschnittes korrekt abgebildet wurde.

Dieser Umstand musste vor dem Import in die Datenbank bereinigt werden. Auf Grundlage, dass der Separierungs- bzw. Clusteralgorithmus sequenziell die Objektklassen aus der Punktwolke extrahierte, konnte mit Hilfe dieser Reihenfolge eine nachträgliche Verschneidung der überlappenden Polygone erfolgen.

Die Extraktionssequenz war wie folgt festgelegt: „Straße, Boden, Vegetation, Gehweg, Bauwerk“.

Die Sperrfläche wurde ausgelassen, da im Referenzdatensatz dieses Polygon auch als Layer über der Straßenfläche digitalisiert wurde. Die Dopplung im Bereich der Sperrflächen war hier beabsichtigt, da mehr als die Hälfte dieser Oberfläche „Straßenbelag“ war und somit nicht eindeutig getrennt werden konnte. Die extrahierte Punktwolke enthielt somit nur die Markierung und wurde durch den Umringalgorithmus als Fläche realisiert.

Die Verschneidung aller benachbarten Flächen erfolgte über die Datenbank. Im Ergebnis lagen eindeutig abgegrenzte Polygone zur Beschreibung der Objektklasse vor.

8.3 Zuordnung der extrahierten Daten zu ASB / OKSTRA

Die Anweisung Straßendatenbank (ASB) regelt bundesweit die Stationierung einer Straße, den Aufbau des Ordnungssystems und die Abbildung der baulichen Anlagen.

Die im Projekt erhobenen Daten waren georeferenziert, folgten allerdings keinem Schema analog der o.g. Straßendatenbank. Die Zuordnung musste im anschließend erfolgen. Hierzu sind generell folgende Voraussetzungen erforderlich:

- Straßenachse als Kleinpunktliste in einer Datenbank
- Festlegung der Netzknoten (VNK, NNK)
- Stationierungsrichtung

Mit diesen Informationen konnte eine geometrische Zuordnung der einzelnen automatisch extrahierten Objekte erfolgen.

Die Zuordnung zum Netzabschnitt wurde über Voronoi-Diagramme gelöst. Hierbei wurden die Kleinpunkte, die den Netzabschnitt geometrisch beschreiben, als Ausgangspunkt für die Berechnung verwendet. Die hierdurch

entstandenen Flächen um die Kleinpunkte wurden mit den Netzabschnittsattributen versehen. Somit konnte die Objektposition geometrisch durch eine Polygon-Punkt Verschneidung dem Netzabschnitt zugeordnet werden. Die Stationierung konnte durch die Erzeugung des Lotes auf die Kleinpunktliste von den Objektteilen berechnet werden.

Die Netzknotenregion nimmt bei der Erstellung der Voronoi-Diagramme eine Sonderrolle ein. Die umgebene Fläche der Voronoi-Zelle muss je nach Lage der benachbarten Streckenabschnitte unterteilt werden. Die Unterteilung wurde so vorgenommen, dass die Zelle gleichmäßig auf Grundlage der Streckenführung zerlegt wurde. Damit war es möglich, die ASB Netzinformationen an die georeferenzierten Daten der Extraktions-ergebnisse zu verknüpfen.

Die Umsetzung des OKSTA-konformen Exports konnte nur über eine vorhandene ASB-konforme Abbildung der Netzknotenabschnitte erfolgen. Die weitere Verknüpfung mit den Metadaten wurde anschließend auf Grundlage der Zuordnung zum Netzknotenabschnitt vorgenommen.

Die OKSTRA-Abbildung sieht nicht vor, Bilddaten und Laserscannerdaten zu integrieren. Somit beschränkte sich die Zuordnung auf eine 2D-Abbildung der extrahierten Objekte. Die Attribuierung blieb erhalten, so dass ein späterer Zugriff auf die Rohdaten möglich war.

8.4 Auswertung und Vergleich der Referenzstrecke mit den automatisch extrahierten Objekten

Der Ausgangspunkt des Vergleichs der automatisch extrahierten Objekte mit den Objekten aus der Referenzdigitalisierung (siehe Kap. 8.1) war der Import in die Datenbank. Das Testgebiet wurde jeweils in einen Abschnitt Autobahn, Landstraße und Ortsdurchfahrt unterteilt. Jeder Geometrietyp (Punk, Linie, Polygon) wurde separat mit dem Referenzdatensatz validiert. Die Referenz-digitalisierung war hierbei die in Kapitel 8.1 genannte Stecke, die manuell über die Messbilder und die vorhandenen Luftbilder erfasst wurde.

8.4.1 Punktobjekte:

Die Auswertung der Punktobjekte erfolgte um jeden Referenzpunkt im jeweiligen Testgebiet anhand eines Buffers. In diesem Buffer wurde die Menge aller extrahierten Punktobjekte gesucht. Im Anschluss erfolgte die Prüfung der jeweiligen Label-Klasse eines gefundenen Objektes. Wurde exakt ein Punkt im Buffer mit identischer Label ID gefunden, so konnte davon ausgegangen werden, dass das korrespondierende Objekt eindeutig zugeordnet werden konnte. Diese Möglichkeit wird als Fall1 definiert (siehe Bild 103: Auswertungsfall 1 Punktobjekte: Punktgeometrie und die Label ID konnten eindeutig zugeordnet werden).

Bild 103: Auswertungsfall 1 Punktobjekte: Punktgeometrie und die Label ID konnten eindeutig zugeordnet werden

Wurden mehrere Objekte mit gleicher oder unterschiedlicher Label ID gefunden, so konnte die Zuordnung nicht eindeutig vorgenommen werden.

Enthielt der Buffer genau ein extrahiertes Element, welches aber eine andere Label ID besaß, war die Objektklassifizierung fehlerhaft. Der Fall 2 wurde entsprechend Bild 104 festgelegt.

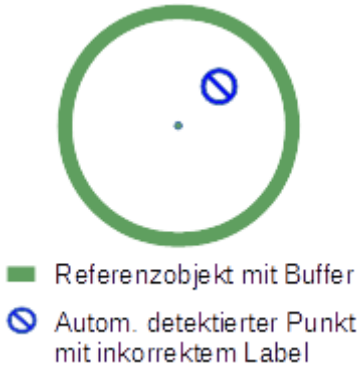


Bild 104: Auswertungsfall 2 Punktobjekte: Punktgeometrie konnte eindeutig zugeordnet werden und Label ID ist nicht korrekt

Enthielt der Buffer mehrere Objekte ohne passender Label ID, wurde das Objekt, welches den kleinsten geometrischen Abstand zum

Referenzobjekt besaß, zugeordnet. Fall3 beschreibt diese nicht eindeutige Zuordnung (siehe Bild 105). Diese Möglichkeit stellte allerdings eine Ausnahme für Objekte, die i.d.R. die gleiche oder eine ähnliche Position besaßen, wie z.B. der Mast und das Schild, dar.

Bild 105: Auswertungsfall 3 Punktobjekte: Punktgeometrie und LABEL ID konnten nicht eindeutig zugeordnet werden

Beinhaltete ein Buffer um ein Referenzobjekt kein automatisch extrahiertes Element, konnte keine Zuordnung erfolgen. Das Objekt galt somit als nicht erkannt. Diese Möglichkeit wurde als Fall 4 definiert (siehe Bild 106).

Bild 106: Auswertungsfall 4 Punktobjekte: Punktobjekt wurde nicht im automatisch generierten Datensatz gefunden

Des Weiteren konnte es durch eine fehlerhafte Extraktion vorkommen, dass ein Objekt erkannt wurde, welches im Referenzdatensatz nicht vorhanden war. Dieser Fall kann durch dynamische Effekte auftreten, z.B. Fahrzeuge und Personen, die mit einer falschen LABEL ID erkannt wurden und im Referenzdatensatz nicht vorhanden waren. Die Möglichkeit wird im Weiteren als Fall 5 bezeichnet (siehe Bild 107).

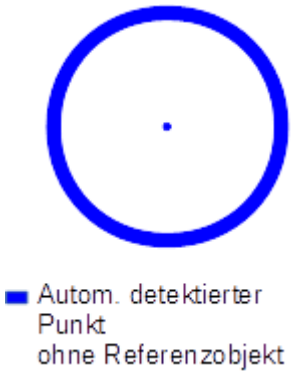


Bild 107: Auswertungsfall 5 Punktobjekte: Automatisch detektiertes Objekt wurde nicht im Referenzdatensatz gefunden

Die Fälle 1 bis 4 bildeten die Gesamtheit der Referenzobjekte und ergaben in Summe 100%. Des Weiteren wurde für jedes korrespondierende Punktobjekt ein mittlerer Abstand berechnet.

8.4.2 Linienobjekte:

Die Auswertung der Linienobjekte erfolgte anhand eines Buffers von 1 m um jede Referenzlinie im jeweiligen Testgebiet. Im ersten Schritt wurde die Menge aller zu extrahierten Linienobjekte gesucht, die sich geometrisch in diesem Bereich befanden. Im zweiten Schritt wurden alle Linienelemente am Umring des Buffers geteilt. Die jetzt im Buffer liegenden Objekte bildeten die Vergleichsmenge. Wurden mehrere Linien im Buffer mit identischer Label ID gefunden, so wurde daraus gefolgert, dass das korrespondierende Objekt eindeutig zugeordnet werden konnte. An die Linienobjekte wurden zwei weitere Bedingungen gestellt: Zum einen durfte der mittlere Abstand zum Referenzobjekt nicht größer sein als die Buffer-Größe und zum anderen musste die aufsummierte Länge der gefundenen Linienobjekte dem Referenzobjekt entsprechen. Diese Möglichkeit wurde als Fall 1 definiert (siehe Bild 108).

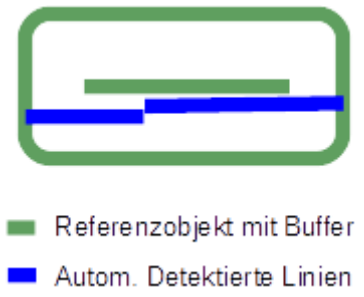


Bild 108: Auswertungsfall 1 Linienobjekte: Referenzlinie und automatische detektiere Linie waren in Lage, Länge und Position ähnlich, die Label ID wurde korrekt zugeordnet

Beinhaltete der Buffer mehrere extrahierte Elemente, welche aber eine andere Label ID besaßen, war die Objektklassifizierung fehlerhaft. Die Bedingungen, dass der mittlere Abstand zum Referenzobjekt nicht größer als die Buffer-Größe sein durfte und die aufsummierte Länge dem Referenzobjekt entsprechen musste, blieben erhalten. Dies entspricht Fall 2 (siehe Bild 109).

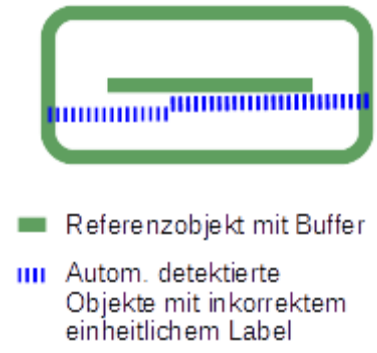


Bild 109: Auswertungsfall 2 Linienobjekte: Referenzlinie und automatische detektiere Linie waren in Lage, Länge und Position ähnlich. Die Label ID wurde falsch zugeordnet

Enthielt der Buffer mehrere Objekte ohne passender Label ID wurde das Objekt zugeordnet, welches den kleinsten geometrischen Abstand zum Referenzobjekt besaß. Zeigte der Buffer der Referenzlinie ein Objekt mit identischer Lable ID und nicht übereinstimmender Länge auf, wurde dieses Objekt dem Referenzobjekt zugeordnet. Diese nicht eindeutige Zuordnung beschrieb den Fall 3, in dem nur Teile mit der Referenz-digitalisierung übereinstimmen (siehe Bild 110).

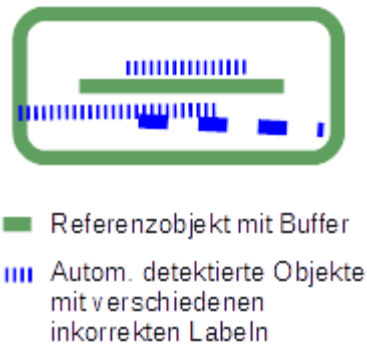


Bild 110: Auswertungsfall 3 Linienobjekte: Referenzlinie und automatische detektiere Linie waren in Lage und Position ähnlich. Die Länge ist des Objektes ist unvollständig

Beinhaltete ein Buffer um ein Referenzobjekt kein automatisch extrahiertes Element, konnte keine Zuordnung erfolgen und das Objekt galt als nicht erkannt. Diese Möglichkeit wurde mit Fall 4 definiert (siehe Bild 111).

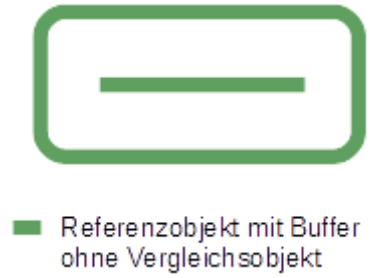


Bild 111: Auswertungsfall 4 Linienobjekte: Für die Referenzlinie konnte kein automatisch detektiertes Objekt zugeordnet werden

Des Weiteren konnte es durch eine fehlerhafte Extraktion vorkommen, dass ein Linienobjekt erkannt wurde, welches im Referenzdatensatz nicht vorhanden war. Der Fall wurde durch dynamische Effekte hervorgerufen, z.B. bei Fahrzeugen, die mit einer falschen LABEL ID erkannt wurden. Herbei waren Objekte extrahiert worden, die nicht im Referenzdatensatz enthalten waren. Diese Möglichkeit wurde im Weiteren als Fall 5 bezeichnet (siehe Bild 112).

Bild 112: Auswertungsfall 5 Linienobjekte: Für das automatisch detektierte Objekt konnte keine Referenzlinie zugeordnet werden

Die Fälle 1 bis 4 bildeten die Gesamtheit der Referenzobjekte und ergaben in Summe 100%. Des Weiteren wurde für jedes korrespondierende Linienobjekt ein mittlerer Abstand berechnet und angegeben.

8.4.3 Flächenobjekte:

Voraussetzung für die Auswertung der Flächenobjekte war die Extraktion der Referenzpolygone im jeweiligen Testgebiet. Jedes Polygon im Testgebiet diente im Weiteren als zu prüfende Schnittmenge mit einem Referenz-Buffer. Der hier eingeführte Buffer diente zur besseren Erfassung der Randbereiche der automatisch detektierten Flächen. Ein Nachteil hierbei war, dass kleine Flächenstücke aus den angrenzenden Flächenobjekten mit in der Schnittmenge vorhanden waren. Voraussetzung für die Auswertung war, wie in Kap. 8.2 beschrieben, dass sich die einzelnen Flächenobjekte nicht überlappen durften. Anderenfalls kam es in den Randbereichen zu keiner eindeutigen Aussage beim Soll-Ist-Vergleich.

Erster Schritt war die Detektion aller in einem Referenzobjekt enthaltenen automatisch extrahierten Flächenobjekten. In der Regel befanden sich mehrere Flächen mit gleicher oder unterschiedlicher Label ID im Überlappungsbereich des Referenzobjektes.

Im zweiten Schritt wurden alle automatisch extrahierten Flächen am Umring des Referenzobjektes geteilt. Die jetzt innerhalb liegenden automatisch detektierten Flächenteile bildeten die Vergleichsmenge.

Die Polygone der Referenzdigitalisierung und der automatischen Extraktion sollten deckungsgleich übereinanderliegen. Die Referenzdigitalisierung der Flächen bestand aus einzelnen Polygonen die überlappungsfrei die reale Oberfläche beschrieben.

Eine Referenzfläche bestand somit immer aus mehreren automatisch detektierten Flächen oder mehrere Flächen der Referenzkalibrierung ergaben eine automatisch generierte Fläche.

Wurden mehrere Flächen mit identischer Label ID im Referenzobjekt gefunden, ergab die aufsummierten Flächenüberdeckung aller Flächenobjekte in der Referenzfläche den späteren Vergleichswert als Flächenmaß.

Wenn die Summer der einzelnen Flächenteile einer LABEL ID 80% erreichte, wurde davon ausgegangen, dass das korrespondierende Objekt eindeutig zugeordnet werden konnte.

Die 80% Flächenüberdeckung ergab sich aus der Digitalisiergenauigkeit und aus einem empirischen Wert, der aus der Dichteverteilung der Punktwolke und der Extraktionsgenauigkeit des Neuronalen Netzes (welches die Punktwolken) klassifiziert wurde.

Im Weiteren wurde die aufsummierte Flächenüberdeckung mit einer 80% Übereinstimmung als Fall1 definiert.

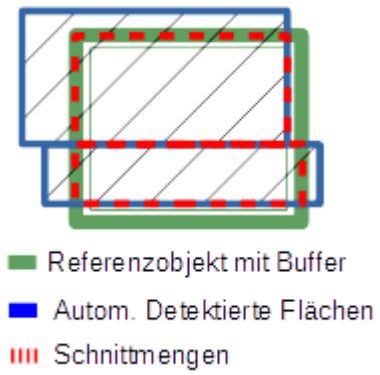


Bild 113: Auswertungsfall 1 Flächenobjekte: Referenzfläche und automatische detektierte Polygonteile sind ähnlich, die Label ID wurde korrekt zugeordnet

Beinhaltete die Referenzfläche mehrere extrahierte Elemente, welche eine andere Label ID als die Referenzfläche besaßen, war die Objektklassifizierung fehlerhaft. War eine Label ID mit einer aufsummierten Fläche von mehr als 80% vorhanden, entsprach dies einer falschen Zuweisung der Label ID. Dies entsprach Fall 2.

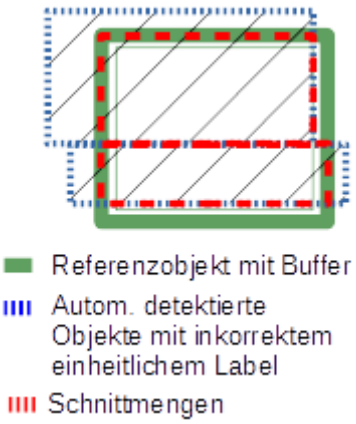


Bild 114: Auswertungsfall 2 Flächenobjekte: Referenzfläche und automatische detektierte Polygonteile waren in Lage, Flächeninhalt und Position ähnlich (die Label ID wurde nicht korrekt zugeordnet)

Enthielt die Referenzfläche mehrere Objekte, welche die Fälle 1 und 3 nicht erfüllten, wurde das Objekt zugeordnet, welches die größte aufsummierte Fläche besaß. Das bedeutete, dass es keine eindeutige Zuordnung zum Referenzobjekt gegeben hat. Dies lag an einer falschen Zuordnung des Labels oder an einem Lageversatz der automatisch extrahierten Fläche, welches die Bedingung von Fall 1 (80% Überdeckung) unterschreitet. Diese nicht eindeutige Zuordnung beschrieb Fall 3.

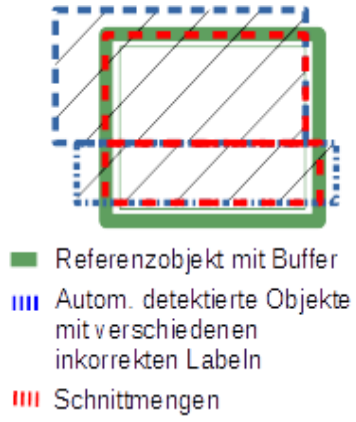


Bild 115: Auswertungsfall 3 Flächenobjekte: Referenzfläche und automatische detektierte Polygonteile waren in Lage identisch (Flächeninhalt oder Label ID konnten nicht korrekt zugeordnet werden)

Beinhaltete eine Referenzfläche kein automatisch extrahiertes Element, konnte keine Zuordnung erfolgen und das Objekt galt als nicht erkannt. Diese Möglichkeit wurde als Fall 4 definiert.

Bild 116: Auswertungsfall 4 Flächenobjekte: Zur Referenzfläche konnten keine automatisch detektierten Polygonteile zugeordnet werden

Des Weiteren konnte es durch eine fehlerhafte Extraktion vorkommen, dass Objekte erkannt wurden, welche im Referenzdatensatz nicht vorhanden waren. Diese Möglichkeit wird als Fall 5 bezeichnet.

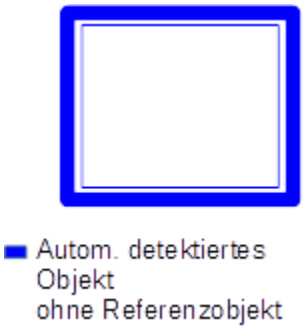


Bild 117: Auswertungsfall 5 Flächenobjekte: Für automatisch detektierte Polygonteile wurde keine Referenzfläche zugeordnet

Die Fälle 1 bis 4 bilden die Gesamtheit der Referenzobjekte und ergaben in Summe 100%.

Alle hier beschrieben Fälle sind in Bild 118 abschließend zusammengefasst.

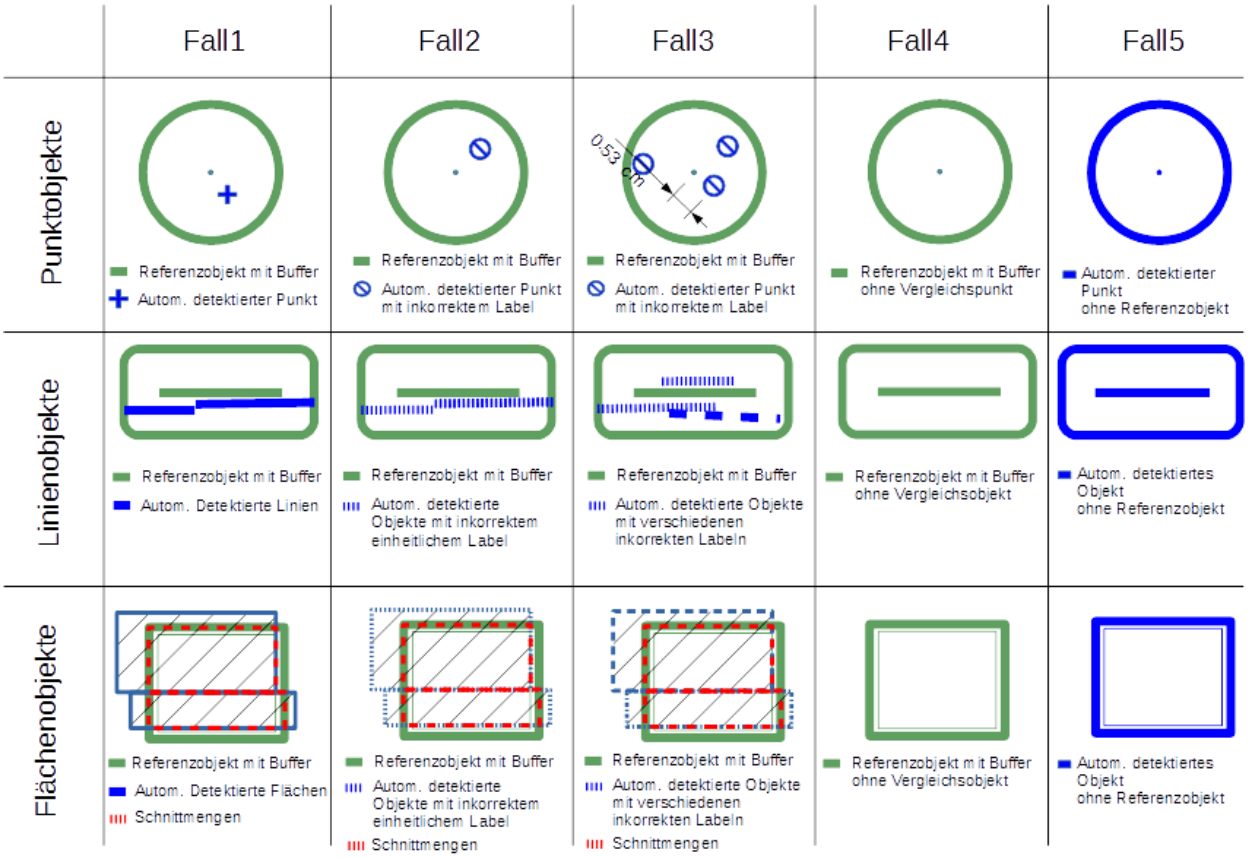


Bild 118: Unterscheidungen zur Eingruppierung des Soll-IST Vergleiches zwischen dem Referenzdatensatz und dem automatisch generierten Datensatz. Hierbei wurde nach Punkt- Linien und Flächenobjekt unterschieden (siehe Kap. 8.3)

Die Suchkriterien wurden dem jeweiligen Geometrietyp angepasst. Hauptkriterium hierbei war die Vergleichbarkeit der einzelnen Objekte im Referenzdatensatz und in den automatisch generierten Daten.

Bei den Punktobjekten wurde ebenfalls die Anzahl der im Suchradius gefunden Objekte berücksichtigt. Der festgelegte Suchradius ergab sich aus heuristischen Vergleichen der Extraktions-genauigkeit einzelner Punktobjekte aus einer Punktwolke. Weiteren Einfluss auf die Extraktions-genauigkeit hatten die bei Laserscannmessungen bedingten Kanteneffekte, die entlang der Aufnahmerichtung zu einer Vergrößerung des Objektes führen und den Verdeckungen, die zu einer virtuellen Trennung eines Objektes führen. Mit diesen Einflüssen ergab sich ein, für die weitere Auswertung, optimaler Buffer von 3m.

Der Buffer erstreckte sich, wie bereits in Kap. 8.4 beschrieben, um jeden Referenzpunkt. Desweitern wurden innerhalb des Buffers die gefundenen Objekte nach der kürzesten Distanz zum Referenzobjekts und nach der Label ID sortiert. Auf Grundlage dieser Vorsortierung fand die Unterscheidung der einzelnen Fälle statt (siehe Bild 118).

Die Suchkriterien für Linienobjekte basierten in erster Linie auf der Längenübereinstimmung zwischen Referenzlinie und automatisch extrahierten Objektstrukturen. Da beide Datensätze aus der gleichen Befahrung extrahiert bzw. digitalisiert wurden, war Ihre georeferenzierte Lage ähnlich. Hingegen unterscheideten sich die einzelnen Liniensegmente und deren Aufteilung maßgeblich. Hierzu wurde angenommen, dass eine Referenzlinie aus mehreren automatisch detektierten Linien oder eine automatisch detektierte Linie aus mehreren Referenzlinien bestand. Mit dieser Annahme wurde um jede Referenzlinie ein Buffer von 1m festgesetzt. Dieser Buffer bildete die Referenzschnittmenge der zu prüfenden Linienobjekte, die durch das neuronale Netz gefunden wurden. In dieser Schnittmenge befanden sich alle lagemäßig benachbarten Objekte. Erster Schritt nach der Erstellung der Schnittmenge war die Gruppierung der Prüfobjekte nach Label ID, der aufsummierten Länge je Label

ID und dem minimalen geometrischen Abstand vom Referenzobjekt. Somit wurde jeder Label ID eine kumulierte Gesamtlänge und ein geometrischer Abstand zugeordnet. Die Zuordnung der einzelnen Extraktionsfälle (siehe Bild 118) erfolgte dann über den Vergleich der Gesamtlänge und den Abstand. Diese beiden Parameter dienten der Zuordnung der in Kap. 8.4.2 vorgenommenen Unterteilungen. Für die Übereinstimmung eines Linienobjektes wurde eine maximale Abweichung von 10% festgelegt. Hierbei stellte es sich heraus, dass der eingeführte Buffer um das Referenzobjekt zu einer Verlängerung des Vergleichsobjektes führte. Somit musste die zuvor eingeführte Buffer-Größe der Vergleichsgröße addiert werden. Daraus ergab sich eine maximale Längenabweichung in Bezug auf die Referenzlänge von

$$L_{ref} - 10\%(L_{ref}) < L_{neuro} < L_{ref} + 10\%(L_{ref}) + 2m.$$

Der Abstand der Vergleichsobjekte war der minimale geometrische Abstand der beiden Geometrien. Bei sich schneidenden Geometrien war der Abstand Null.

Die Polygonobjekte wurden im gleichen Kontext verglichen wie die Linienobjekte. Der Unterschied hierbei war lediglich die verwendete Vergleichsmenge. Statt einer aufsummierten Länge wurde die aussummierte Fläche angesetzt. Der hierbei verwendete Buffer wurde auf 10 cm um das Referenzobjekt festgelegt. Die entstandenen Schnittmengen wurden mit Hilfe der LABEL ID gruppiert. Die Fallzuordnung erfolgte über die aufsummierten Flächen und die Abstände der Flächenschwerpunkte (siehe Bild 118).

Tabelle 8: Festsetzung der Vergleichswerte zwischen Referenzdatensatz und zu prüfendem Datensatz

Objekttyp	Buffergröße	Vergleichskriterien
Punkt	3m	ID, min. Abstand
Linie	1m	ID, Länge: $L_{ref} - 10\%(L_{ref}) < L_{neuro} < L_{ref} + 10\%(L_{ref}) + 2m$, min. Abstand
Polygon	0.1m	ID, Fläche: $F_{ref} - 30\%(F_{ref}) < F_{neuro} < F_{ref} + 30\%(F_{ref})$, min. Abstand

Die folgende Auswertung der einzelnen Fallunterscheidungen pro Objekttyp wurde als SQL Skript umgesetzt. Mit der vorhandenen Datenbank konnte somit jederzeit die Auswertung nachvollzogen werden.

Die Referenzdigitalisierung wurde entlang der gefahrenen Trajektorie in einem Korridor +/- 10m erfasst. Das Neuronale Netz lieferte zum Teil auch Daten außerhalb dieses Bereiches. Um eine Vergleichbarkeit sicherzustellen, wurde eine Überlappungsfläche festgesetzt, welche eine Schnittmenge darstellt. Dies sollte vermeiden, dass Objekte im Randbereich, die nicht in der Referenzdigitalisierung enthalten waren, in die Auswertung mit einbezogen wurden. Diese Überlappungsfläche wurde in folgende drei Abschnitte aufgeteilt:

- BAB
- Landstraße
- Innerortstraße

Die Auswertung erfolgte innerhalb dieses Bereiches. Alle Daten der Referenzstrecke waren dem Neuronalen Netz unbekannt. Alle Trainingsdaten stammten nicht aus der näheren Umgebung der Teststrecke.

Die folgenden vier Tabellen zeigten die Ergebnisse des Soll-Ist-Vergleiches. Die Tabelle 12 stellt hierbei die Zusammenfassung aus den drei vorhergehenden Tabellen dar.

Die Spalten sind bei allen vier nachfolgenden Tabellen identisch und wurden mit den in Bild 118 gezeigten Vergleichsmethoden ausgewertet.

Die ersten beiden Spalte gaben die Label ID und die dazugehörige Beschreibung des Objektes an. Die 3. Spalte definierte den Objekttyp und gab somit Auskunft über die angewendete Vergleichsmethode. Die Auswertefälle 1 bis 4 geben an, ob ein Objekt eindeutig zugeordnet werden konnte, ob Einschränkungen vorlagen oder gar nicht gefunden wurde. Die Prozentangaben beziehen sich auf die Spalte „Referenz“. Die Referenz gibt je nach Objekttyp an, wieviel Punkobjekte bzw. Gesamtlänge der Referenzlinienobjekte oder Gesamtfläche der Referenzpolygone untersucht wurden. Die Spalte F5 gibt Auskunft darüber, wie viele Objekte extrahiert wurden und welche keinem Referenzobjekt zugeordnet werden konnten. Weiterhin wurde der mittlere Abstand zwischen Referenzobjekt und Vergleichsobjekt angegeben.

Abschnitt Autobahn:

Tabelle 9: Vergleich der Referenzdigitalisierung mit den aufbereiten, automatisch gefunden Objekten des Neuronalen Netzes im Abschnitt Autobahn

ID	Objekttyp	Geometrie	F1 [%]	F2 [%]	F3 [%]	F4 [%]	Referenz	F5 [%]	Abstand [m]
1	Schild	Point	**59.46**	**2.7**	**1.35**	**36.49**	74	67.35	0.46
2	Kilometertafel	Point	**80.00**	**20.00**	**0.00**	**0.00**	5	20.00	0.78
3	Stations-zeichen	Point	0.00	14.29	14.29	71.43	7	0.00	0.65
4	Ortsdurch-fahrtszeichen	Point					0	0.00	
5	Notrufsäule	Point	0.00	0.00	0.00	100.00	4	0.00	
6	Licht-signalanlage	Point					0	0.00	
7	Beleuchtung	Point					0	0.00	
8	Mast	Point	0.00	33.33	33.33	33.33	3	92.94	0.46
9	Leitpfosten	Point	13.50	1.23	1.84	83.44	163	18.52	0.86
10	Schutzplanke	Multi-Linestring	**58.00**	**37.33**	**0.00**	**4.00**	24690m	30.44	0.16
11	Schutzwand	Multi-Linestring	6.67	80.00	13.33	0.00	147m	-0.82	0.50
12	Bauwerk	Polygon	**100.00**	**0.00**	**0.00**	**0.00**	6198 m^2	0.70	0.00
13	Verteiler-kasten	Punkt	0.00	50.00	50.00	0.00	2	0.00	0.47
14	Bordstein	Multi-Linestring	0.00	60.00	40.00	0.00	6m	-33.99	0.42
15	Deckel	Punkt	0.00	0.00	0.00	100.00	35	0.00	0.00
16	Fahrbahn-markierung	Multi-Linestring	**90.91**	**4.62**	**0.14**	**4.20**	19390m	7.31	0.09
17	Baumstamm	Punkt	0.00	0.00	0.00	0.00	0	81.82	0.00
18	Straße	Polygon	**64.81**	**18.52**	**0.00**	**16.67**	222079 m^2	25.51	0.10
20	Gehweg	Polygon	0.00	0.00	0.00	0.00	29773 m^2	0.84	0.00
23	Vegetation	Polygon	**53.21**	**36.70**	**0.92**	**9.17**	174322 m^2	17.57	0.02
24	Boden	Polygon	**73.53**	**0.00**	**23.53**	**2.94**	9463 m^2		0.03
26	Sperrfläche	Polygon	40.00	13.33	46.67	0.00	5004 m^2	2.98	0.03
28	Schacht	Point	0.00	0.00	0.00	100.00	30	0.00	0.00
30	Betonleitwand	Multi-Linestring	0.00	25.00	75.00	0.00	255m	72.60	0.14

Abschnitt Landstraße:

Tabelle 10: Vergleich der Referenzdigitalisierung mit den aufbereiten, automatisch gefunden Objekten des Neuronalen Netzes im Abschnitt Landstraße

ID	Objekttyp	Geometrie	F1 [%]	F2 [%]	F3 [%]	F4 [%]	Referenz	F5 [%]	Abstand [m]
1	Schild	Point	23.31	5.56	18.06	52.78	72	49.09	0.75
5	Notrufsäule	Point					0	100.00	
6	Lichtsignalanlage	Point	0.00	33.33	33.33	33.33	6	0.00	0.60
7	Beleuchtung	Point	0.00	40.00	40.00	20.00	5	0.00	1.09
8	Mast	Point	30.00	40.00	5.00	25.00	20	42.86	0.44
9	Leitpfosten	Point	12.64	6.90	6.90	73.56	87	58.14	1.05
10	Schutzplanke	Multi-Linestring	**51.52**	**42.42**	**6.06**	**0.00**	3379m	32.46	0.14
11	Schutzwand	Multi-Linestring	0.00	0.00	0.00	100.00	525m	0.00	0.00
12	Bauwerk	Polygon	**75.00**	**25.00**	**0.00**	**0.00**	6198m	2.74	0.00
14	Bordstein	Multi-Linestring	0.00	75.76	12.12	18.18	2548m	0.00	0.23
15	Deckel	Punkt	0.00	20.00	0.00	80.00	5	0.00	2.74
16	Fahrbahnmarkierung	Multi-Linestring	**75.49**	**10.98**	**0.90**	**12.63**	9434m	15.45	0.15
17	Baumstamm	Punkt	47.37	16.84	1.05	34.74	285	59.25	1.13
18	Straße	Polygon	44.23	51.92	0.00	3.85	222079 m^2	9.99	0.07
20	Gehweg	Polygon	**61.54**	**10.77**	**24.62**	**3.07**	29773 m^2	10.16	0.03
23	Vegetation	Polygon	**59.75**	**34.59**	**3.14**	**2.52**	174322 m^2	19.08	0.07
24	Boden	Polygon	33.33	0.00	66.67	0.00	25458 m^2	0.00	0.13
26	Sperrfläche	Polygon	11.11	16.67	72.22	0.00	5004 m^2	6.24	0.01
28	Schacht	Point	0	0	0	100	5	0.00	0.00

Abschnitt Innerortstraßen:

Tabelle 11: Vergleich der Referenzdigitalisierung mit den aufbereiten, automatisch gefunden Objekten des Neuronalen Netzes im Abschnitt Innerortstraßen

ID	Objekttyp	Geometrie	F1 [%]	F2 [%]	F3 [%]	F4 [%]	Referenz	F5 [%]	Abstand [m]
1	Schild	Point	18.62	35.22	6.88	39.27	247	7.21	0.88
3	Stations-zeichen	Point	0.00	50.00	0.00	50.00	2	0.00	0.14
4	Ortsdurch-fahrtszeichen	Point	0.00	40.00	20.00	40.00	5	0.00	0.39
5	Notrufsäule	Point					0	33.33	
6	Lichtsignal-anlage	Point	1.67	33.33	21.67	43.33	60	0.00	0.59
7	Beleuchtung	Point	0.00	39.64	23.42	36.94	111	0.00	1.79
8	Mast	Point	18.73	30.30	11.85	39.12	363	9.36	0.80
9	Leitpfosten	Point	20.90	11.94	5.97	61.19	67	83.33	1.09
10	Schutzplanke	Multi-Linestring	46.15	38.47	15.38	0.00	1191m	56.73	0.21
11	Schutzwand	Multi-Linestring	0.00	32.44	18.92	48.65	193m	72.99	0.13
12	Bauwerk	Polygon	**62.86**	**30.00**	**5.71**	**1.43**	6198 m^2	19.79	0.06
13	Verteilerkasten	Punkt	0.00	28.13	25.00	46.88	32	0.00	1.07
14	Bordstein	Multi-Linestring	1.13	59.32	7.91	30.51	1146m	56.43	0.23
15	Deckel	Punkt	0.00	13.73	11.76	74.51	255	0.00	2.06
16	Fahrbahn-markierung	Multi-Linestring	**64.97**	**9.98**	**0.31**	**24.80**	11015m	20.29	0.18
17	Baumstamm	Punkt	32.89	16.45	7.89	42.76	152	113.26	0.94
18	Straße	Polygon	**54.22**	**30.72**	**10.84**	**4.22**	222079m^2	9.53	0.08
20	Gehweg	Polygon	**50.59**	**17.06**	**28.24**	**4.11**	29773 m^2	22.30	0.03
23	Vegetation	Polygon	**53.28**	**22.78**	**18.53**	**5.41**	174322 m^2	5.95	0.05
24	Boden	Polygon	23.33	0.00	76.67	0.00	9777 m^2	0.25	0.01
26	Sperrfläche	Polygon	6.25	31.25	56.25	6.25	5004 m^2	15.60	0.00
28	Schacht	Point	3.54	12.39	8.85	75.22	113	0.00	1.61
30	Betonleitwand	Multi-Linestring	0.00	0.00	0.00	100.00	219m	76.87	

Gesamtauswertung:

Tabelle 12: Vergleich der Referenzdigitalisierung mit den aufbereiteten, automatisch gefunden Objekten des Neuronalen Netzes für die Abschnitte BAB, Landstraße und Innerortstraßen

ID	Objekttyp	Geometrie	F1 [%]	F2 [%]	F3 [%]	F4 [%]	Referenz	F5 [%]	Abstand [m]
1	Schild	Point	27.23	5.6	25.95	41.22	393	62.62	0.78
2	Kilometer-tafel	Point	**80.00**	**0.00**	**20.00**	**0.00**	5	16.67	0.78
3	Stations-zeichen	Point	0.00	11.11	22.22	66.67	9	0.00	0.48
4	Ortsdurch-fahrtszeichen	Point	0.00	20.00	40.00	40.00	5	0.00	0.39
5	Notrufsäule	Point	0.00	0.00	0.00	100.00	4	100.00	-
6	Licht-signalanlage	Point	1.52	22.73	33.33	42.42	66	66.67	0.59
7	Beleuchtung	Point	0.00	24.14	39.66	36.21	116	28.57	1.76
8	Mast	Point	19.17	11.66	30.83	38.34	386	65.13	0.77
9	Leitpfosten	Point	14.83	4.10	5.05	76.03	317	44.00	1.00
10	Schutzplanke	Multi-Linestring	**56.12**	**2.04**	**38.78**	**3.06**	29260m	31.75	0.16
11	Schutzwand	Multi-Linestring	1.69	15.25	42.37	40.68	340m	41.06	0.29
12	Bauwerk	Polygon	**65.06**	**28.92**	**4.82**	**1.20**	6198 m^2	23.23	0.05
13	Verteilerkasten	Point	0.00	26.47	29.41	44.12	34	100.00	1.01
14	Bordstein	Multi-Linestring	0.93	9.30	61.86	27.91	1152 m	55.96	0.23
15	Deckel	Point	0.00	10.17	12.20	77.63	295	50.00	2.07
16	Fahrbahn-markierung	Multi-Linestring	**73.50**	**0.40**	**8.92**	**17.17**	39839 m	12.83	0.15
17	Baumstamm	Point	42.33	3.43	16.70	37.53	437	59.48	1.07
18	Straße	Polygon	**56.13**	**30.06**	**5.52**	**8.29**	222079 m^2	45.02	0.09
20	Gehweg	Polygon	**53.62**	**15.32**	**27.23**	**3.83**	29773 m^2	33.30	0.03
23	Vegetation	Polygon	**55.22**	**29.22**	**10.25**	**5.31**	174322 m^2	42.60	0.05
24	Boden	Polygon	48.57	0.00	50.00	1.43	9777 m^2	0.25	0.02
26	Sperrfläche	Polygon	18.37	20.41	59.18	2.04	5004 m^2	24.82	0.02
28	Schacht	Point	2.70	6.76	9.46	81.08	148	0.00	1.61
30	Betonleitwand	Multi-Linestring	0.00	60.00	20.00	20.00	474m	74.57	0.14

Im Abschnitt BAB gab es große Unterschiede in der Übereinstimmung zwischen Referenzdatensatz und Testdatensatz. 8 von 24 Objektklassen konnten zu mehr als 50 % der Referenzobjekte eindeutig zugeordnet werden. Bestes und repräsentativstes Beispiel war das Extraktionsergebnis der Objektklasse „Fahrbahnmarkierung“ mit 91 % Übereinstimmung und 4 % nicht gefundener Linienabschnitte bei einer Gesamtlänge von 19 km. Bei der Objektklasse „Kilometertafel“ waren sehr gute Übereinstimmungen festgestellt worden. Allerding wurden hier nur 5 Referenzobjekte überprüft, welches die statistische Aussagefähigkeit der Übereinstimmung einschränkt. Für alle Objekte, die in der Spalte „Referenz“ den Wert 0 besitzen, war kein Referenzobjekt in diesem Streckenabschnitt vorhanden.

Die Spalte „Abstand“ gibt die jeweilige mittlere geometrische Distanz zum Referenzobjekt an. Auffällig war, dass die Punktobjekte i.d.R. einen größeren Abstand zum Referenzobjekt aufwiesen als die Linien oder Flächenobjekte. Ursache hierfür war die nicht optimale Bestimmung des Repräsentationspunktes in der Datenbank über eine Schwerpunktberechnung. Weitere Optimierungen, bspw. durch die Anwendung eines abstandsgewichteten Meridianfilter, der Ausreißer in der Punktwolke des jeweiligen Objektes ausschließen kann, wurden im Zuge der Auswertung realisiert.

Im angrenzenden Abschnitt „Landstraße“ ergab sich ein anderes Extraktions- bzw. Vergleichsergebnis bezüglich der einzelnen Objektklassen. Der Extraktionswert der Fahrbahnmarkierung (F1) verringerte sich auf 75 %. Die nicht gefundenen Linienobjekte im Vergleichsdatensatz erhöhten sich auf 13 %. Dies war auf eine schlechtere Sichtbarkeit der Objektklasse „Markierung“ im Landstraßenabschnitt zurückzuführen. Hingegen hat sich das Extraktionsergebnis der Objektklasse „Vegetation“ verbessert. Dies wurde auf den kleineren Straßenquerschnitt und einer einfacheren Gestaltung der Nebenflächen zurückgeführt.

Im Abschnitt „Innerortstraßen“ wurde die größte Anzahl an Objekten detektiert. Die Anzahl der Referenzobjekte hat sich bei ähnlicher Vergleichslänge zu den anderen Abschnitten teilweise verdreifacht. Hingegen ist die Extraktionsrate nicht gestiegen. Dies deutete darauf hin, dass Objekte, die eine hohe räumliche Dichte zueinander aufweisen, schlechter detektiert wurden. Bei komplexeren Scenes zeigte sich, dass der Extraktionsalgorithmus noch größere Defizite hinsichtlich der Trennung der Objekte auswies.

8.5 Schlussfolgerungen zur Mustererkennung

8.5.1 Schritt 1: Mustererkennung in 2D mittels eines KNN

Der wesentliche Teil der Muster- bzw. Objekterkennung wurde in der prototypischen Verarbeitungskette des Forschungsprojekts durch den Einsatz eines Künstlichen Neuronalen Netzes (KNN) realisiert, welches 2D-Bilddaten semantisch segmentiert.

Es zeigte sich, dass eine Mustererkennung mittels eines KNN für viele der relevanten Klassen mit einer ausreichenden Genauigkeit erzielt werden konnte. Darunter fallen vor allem Klassen, welche größere Bereiche flächenhaft abdecken, wie z.B. die Straßenoberfläche, Vegetationsbereiche, Bauwerke oder Bäume (Baumstämme, Baumkronen). Auch Objekte, welche in ihrer Erscheinung wenig variieren (bspw. Leitpfosten), sind für das KNN leicht zu identifizieren.

Es konnten auch komplexe, in ihrer Erscheinung stark variierende Objekte, welche mit „herkömmlichen“ Algorithmen nur schwer erkannt werden konnten, detektiert werden. Darunter fielen bspw. Vegetation und Bäume, aber auch Bauwerke und Schilder in ihrer Vielzahl an Ausprägungen. Ebenso ließ sich die Objektklasse „Fahrzeuge“ gut erkennen, die als Hilfsklasse mit in das Training aufgenommen wurde. Die Objektklasse „Fahrzeuge“, tritt häufig auf und verdeckt oft andere relevante Klassen.

Insgesamt hat sich die Nutzung eines KNN als wesentlicher Bestandteil der Objekterkennung bewährt, sodass auch komplexe Objekte im Straßenumfeld automatisch detektiert werden können. Die Erkennungsgenauigkeit fällt jedoch nicht für alle Klassen zufriedenstellend aus. Um die Leistung des KNN zu erhöhen, erscheint primär eine ausreichende und sorgfältig durchgeführte Vergrößerung des manuell annotierten Datensatzes für das Training erforderlich. So stellt die begrenzte Anzahl der im Rahmen des Forschungsprojektes zur Verfügung stehenden Trainingssamples einen limitierenden Faktor für die Erkennungsgenauigkeit dar. Dies gilt insbesondere für kleiner und selten vorkommende Objekte, wie z.B. Kilometertafeln oder Stationszeichen.

Unklar bleibt, warum bspw. die Objektklasse „Lichtsignalanlagen“ nicht zuverlässig erkannt wurde, obwohl sie in ihrer Erscheinung nur gering variiert. Dass eine zuverlässigere Erkennung solcher Objekte technisch möglich ist, zeigten neben vorliegenden Erfahrungen des

Forschungsnehmers auch die Rankings der öffentlichen Vergleichswettbewerbe wie etwa Cityscapes (CORDTS et al. 2016). Dort standen allerdings deutlich mehr Trainingsdaten zur Verfügung.

Zusätzlich zur Erweiterung des Trainingsdatensatzes gab es Möglichkeiten, den Trainingsvorgang in geeigneter Weise zu steuern und zu optimieren, z.B. durch eine höhere Gewichtung der schwer erlernbaren Objektklassen.

8.5.2 Schritt 2: Mustererkennung und Objektextrahierung

Nach der Objekterkennung mittels des KNN und der Übertragung der Klasseninformationen in die 3D-Daten bildete eine semantisch segmentierte Punktwolke die Basis der weiteren Verarbeitung. Diese umfasste insbesondere die Identifikation einzelner Objektinstanzen (Clustering, weitere Schlussfolgerungen hierzu in Kap. 8.6) sowie die Berechnung der Ergebnisse als 2D-Polygon (Kap. 8.7). Letzteres war eine konkave Hülle um die einzelnen Objekte auf eine zuvor im Algorithmus bestimmte Straßenebene zu projizieren.

Bei dieser zweiten Stufe der Mustererkennung konnten manche Fehlklassifizierungen des KNN gefiltert werden. Dies erfolgte durch die aus verschiedenen Perspektiven durchgeführte Klassifizierung einzelner Objekte. Ebenso konnten einige der „überschüssigen“ Objekte für manche Klassen („False Positives“), die durch Mehrdeutigkeiten bei der Rückprojektion der Klassenlabel entstehen, eliminiert werden. Dasselbe galt für die Fehlklassifikation von 3D-Punkten bei der Datenfusionierung aufgrund von Restungenauigkeiten in der Kalibrierung.

Mittels der vorliegend angewendeten Parametrisierung der Algorithmen war es möglich, sowohl die Anzahl an „False Positives“ als auch an „False Negatives“, zu beeinflussen. Eine geringere Rate an „False Positives“ konnte z.B. den für die manuelle Kontrolle der Ergebnisse nötigen Aufwand minimieren. Dies erfolgte dann, wenn es weniger relevant war, dass weniger stark ausgeprägte Objekte gefiltert wurden und dadurch mehr „False Negatives“ zu verzeichnen waren.

Bei der Analyse der Ergebnisse wurde deutlich, dass die infolge unterschiedlicher Befahrungsgeschwindigkeiten mitunter stark variierende lokale Punktdichte problematisch war. Je stärker diese variierte, desto schwieriger war es, die Parameter der Objektextraktion geeignet zu wählen. Insbesondere konnte eine lokal sehr hohe Punktdichte die Rate von „False Positives“ für manche Klassen signifikant erhöhen.

8.5.3 Erkennungsgenauigkeit und Diversität der Trainingsdaten

Es zeigte sich, dass der im Forschungsprojekt für das Training zur Verfügung stehende Satz an manuell annotierten Trainingsdaten für einige Klassen ausreichend groß war. Dies galt insbesondere dann, wenn diese Objekte häufig vorkommen und typischerweise einen größeren Teil der Bildfläche einnehmen, wie z.B. Straße, Vegetation und Baumkronen.

Andere Klassen waren deutlich unterrepräsentiert, was sich auch bei der quantitativen Analyse der Trainingsergebnisse zeigte. Die Erkennungsgenauigkeit korrelierte sehr stark mit dem prozentualen Anteil den die einzelnen Klassen an Pixeln im Trainingsdatensatz einnahmen. Dies war, wie in Kap. 3.1.8 beschrieben, auf das mathematische Prinzip, auf welchem das Training von KNN basiert, zurückzuführen.

Der Vergleich mit der manuellen Auswertung von Objekten entlang der Teststrecke zeigte, dass manche Klassen mit niedriger IoU nahezu gleich gut identifiziert werden konnten wie Klassen mit deutlich höherer IoU.

Insgesamt lässt sich festhalten, dass Klassen, welche mit einem geringen Anteil an Pixeln im Trainingsdatensatz vertreten waren, deutlich schlechter gelernt wurden. Solche nicht ausbalancierten Klassenverteilungen (Imbalanced Data) stellen grundsätzlich eine große Herausforderung für das Training dar.

Diese Problematik kann idealerweise durch eine Vergrößerung des Trainingsdatensatzes gemindert werden, indem Bilder, welche vermehrt „seltene“ Klassen enthalten, hinzugeführt werden. Alternativ steht die Technik der „Data Augmentation“ zur Verfügung. Dies erfordert jedoch eine intensive Analyse der Trainingsdaten, eine Anpassung der Trainingsinfrastruktur sowie eine sorgfältige Parametrisierung.

Auch die Diversität der Trainingsdaten spielt bei der Generalisierungsleistung des KNN eine Rolle. Ein Beispiel ist die Klassifizierung von Gebäuden: Sind im Trainingsdatensatz keine Fachwerkhäuser vorhanden, wird die Ausgabe des KNN weniger sicher und damit stärker rauschbehaftet sein, wenn neue Daten prozessiert werden, welche ein solches „unbekanntes“ Gebäude enthalten. Entscheidend ist, wie viele der für die Gebäude-Klasse gelernten

Merkmale auch bei solchen Häusern noch in ausreichender Ausprägung vorliegen.

8.5.4 Weitere Schlussfolgerungen zur Mustererkennung

Insbesondere die Erkennung seltener, kleiner und schmaler Objekte stellte für die Objekterkennung eine Herausforderung dar. Dies war nicht nur im vorliegenden Forschungsprojekt der Fall, sondern stellt ein allgemeines Problem der Muster- und Objekterkennung dar. Jede Anwendung wird ihre eigenen Anforderungen und Schwerpunkte definieren müssen, damit die Algorithmen entsprechend angepasst und erweitert werden können.

Sollten künftig innerhalb einer Objektklasse weitere Unterklassen detektiert werden, stehen auch hier wieder verschiedene Methoden zur Verfügung: Der vorhandene Trainingsdatensatz kann entsprechend überarbeitet werden. Alternativ können für diese Unterklassen separate KNN trainiert werden, welche anhand der „groben“ semantischen Segmentierung eine verfeinerte Erkennung (z.B. einzelne, spezielle Typen von Verkehrsschildern) erlauben. Dies kann wiederum in Form einer semantischen Segmentierung oder als Klassifikation des relevanten Bildausschnitts erfolgen.

Ein ähnliches Vorgehen kann eingeschlagen werden, wenn eine Detektierung spezieller Attribute von Objekten erforderlich ist, wie z.B. Befestigungsarten. Dabei ist darauf zu achten, dass die benötigten Merkmale evtl. nur in Bilddaten mit höherer Auflösung oder anderer Brennweite ausreichend detailliert erfasst werden können.

Weiterhin wäre es denkbar, für die semantische Segmentierung einzelner Klassen jeweils ein eigenes KNN zu trainieren. Auch damit könnte der Eigenschaft des Trainings entgegengewirkt werden, dass kleine oder selten vorkommende Klassen in der Regel schlechter gelernt werden.

Für manche Klassen würde auch die Verwendung weiterer Heuristiken bzw. Regelwerke sinnvoll erscheinen, etwa dort, wo bestimmte Objekte nur gemeinsam vorkommen. Ein Beispiel stellen Pfosten oder Masten dar, welche i.d.R. nur in Verbindung mit einem Schild, einer Lichtsignalanlage oder Ähnlichem, nicht jedoch isoliert, auftreten. So können „False Positives“ reduziert werden.

In jedem Fall ist jedoch der Aufwand zu berücksichtigen, den ein zusätzliches Training darstellt; genauso wie die Erhöhung der Prozessierungsdauer durch die Erhöhung der Anzahl der einzelnen Klassifizierungsschritte.

8.6 Schlussfolgerungen zur Objekt-Extrahierung

Bei der Klassifizierung der Bilddaten und der Rückprojektion der Klasseninformation in die Punktwolke kam es zu lokal fehlerhaft klassifizierten Bereichen in der Punktwolke. Dies war z.B. dann der Fall, wenn die Klassifizierung der Bilddaten nicht eindeutig ausfiel. Außerdem führte die „Transparenz“ von Objekten in der Punktwolke zusammen mit Abweichungen bei der Kalibrierung oder lokalen Störungen der Genauigkeit der Positionierungsdaten zu einer stellenweise ungenauen Rückprojektion der Klassen-informationen (vgl. Kap. 5.4.4).

Solche Fälle führten zu „False Positives“, also Bereiche, die fälschlicherweise einer bestimmten Klasse zugeordnet wurden. Seltener traten die Fälle „False Negatives“ dort auf, wo die entsprechenden Bereiche der eigentlich richtigen Klasse „abgezogen“ wurden.

Des Öfteren konnten „False Positives“ Fälle durch eine geeignete Parametrisierung bei der Extrahierung von Objektinstanzen entfernt werden. Strikte Grenzwerte, z.B. für eine minimale Anzahl an räumlich zusammenhängenden Punkten pro Objektinstanz, führten auch dazu, dass korrekte Instanzen als zu klein betrachtet und gefiltert wurden.

Eine geeignete Parametrisierung hängt grundsätzlich davon ab, welche Anforderungen an eine produktive Anwendung gestellt werden. Sind z.B. „False Positives“ Fälle problematisch, ist eine strengere Parametrisierung notwendig. Sollen wiederum keine Objekte übersehen werden, ist eine weniger strikte Filterung besser. Da die Parametrisierung pro Klasse erfolgen muss, sind ausreichend viele Experimente zur Bestimmung der idealen Parameter notwendig.

Die im Forschungsprojekt erzielten Ergebnisse wurden anhand einer stichprobenartigen Analyse von „False Positives“ und „False Negatives“ Fälle in 24 Abschnitte von je 50 m Länge als Kompromisslösung festgelegt. Die dabei pro Klasse gewählten Parameter für die minimale und maximale Punktanzahl pro Objektinstanz sowie das zugehörige Distanzkriterium der Software wurden dokumentiert.

8.7 Schlussfolgerungen zur Berechnung der konkaven Hülle

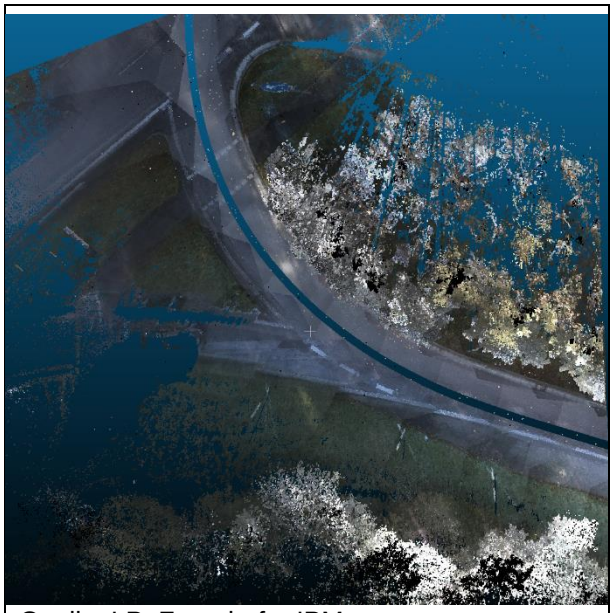
Ähnliche Überlegungen wie zur Objekt-Extrahierung ließen sich zur Parametrisierung des Algorithmus anstellen, der die konkave Hülle zu den auf die Straßenebene projizierten Objektpunktwolken berechnete. Hier konnte der Alpha-Parameter entsprechend der Anforderungen einer konkreten Anwendung optimiert werden.

Ein niedrig angesetzter Wert führte zu feingliedrigen Polygonen, bei denen stark ausgeprägte konkave Eigenschaften erlaubt waren. Die resultierenden Polygone hatten eine entsprechend hohe Anzahl an Eckpunkten. Unter idealen Umständen wurde der tatsächliche Umriss des Objekts so besser angenähert als mit einem größer gewählten Wert.

Ein größerer Wert für Alpha reduziert die erlaubte Größe einer konkaven Ausprägung. Dies führte dazu, dass weniger Punkte für das resultierende Polygon ausgewählt wurden, wodurch es gleichmäßiger und weniger „ausgefranst“ erschien. Gleichzeitig fiel bei höherem Alpha die Fläche des Polygons größer aus, da Lücken und/oder konkave Ausprägungen zunehmend geschlossen wurden. So kam es in Bereichen, in denen Objekte eng miteinander verschränkt waren, zu Überschneidungen zwischen den Polygonen verschiedener Klassen.

Der Wert für Alpha sollte demnach möglichst gering gewählt werden. Eine größere Wahl führte je nach Anwendung zu gleichmäßigeren Polygonen. In der finalen Prozessierung der Daten wurde für Alpha der Wert 5 festgelegt, welcher einen guten Kompromiss zwischen Detailierung und Gleichmäßigkeit darstellte. Bei Bedarf könnte dieser Wert auch klassenspezifisch festgelegt werden.

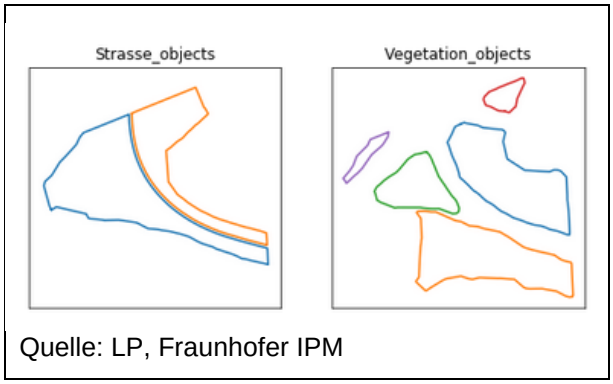
Zu beachten ist, dass der ConcaveHull-Algorithmus der PCL keine Lücken innerhalb von Polygonen erlaubt. Eine entsprechende Szene zeigt Bild 119.



Quelle: LP, Fraunhofer IPM

Bild 119: Straßenführung um eine Grünfläche herum.

Die zu dieser Szene erzeugten 2D-Polygone zeigen, dass die Fläche der Straße die Grünfläche einschließt, dort jedoche keine entsprechende Aussparung aufweist (s. Bild 120).



Quelle: LP, Fraunhofer IPM

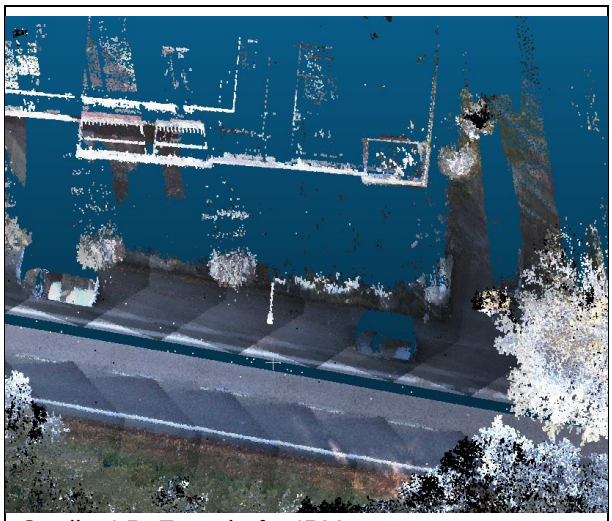
Bild 120: Die 2D-Polygone zur oben dargestellten Szene zeigen, dass die konkaven Hüllen der Straße-Objektinstanzen im Bereich der Vegetation keine Aussparung aufweisen.

Sollte die o.g. Anforderung ermöglicht werden, muss der Algorithmus erweitert oder eine andere Bibliothek gewählt werden, welche diese Funktionalität zur Verfügung stellt.

8.8 Einfluss von Verdeckungen durch stationäre Objekte

Entlang einer Befahrungsstrecke im öffentlichen Raum gibt es eine Vielzahl von Situationen, bei denen Straßenbestandsobjekte durch andere Objekte verdeckt werden. Dies können stationäre oder sich bewegende Objekte sein.

Problematisch sind insbesondere größere Objekte, die räumlich nahe an der Befahrungsachse liegen, z.B. Plakatwände, Büsche, Bäume, Einfriedungen, oder große Schilder. Immer wieder befinden sich auch kleinere Objekte nahe an der Befahrungsachse, z.B. Unterstände an Haltestellen. Auch die bereits erwähnten Fahrzeuge führen aufgrund ihrer Größe häufig zu Verdeckungen, gleich ob im parkenden oder bewegten Zustand.



Quelle: LP, Fraunhofer IPM

Bild 121: Verdeckungen durch parkende Fahrzeuge, höhere Sträucher und Gebäudefronten: Die Lücken in der Punktwolke sind als transparente Bereiche sichtbar.

Verdeckungen waren im Forschungsprojekt vor allem in der vom Laserscanner generierten Punktwolke relevant, da während der Datenaufnahme jedes Objekt in der Regel nur einmal vom Scanner aus einer bestimmten Richtung detektiert wurde. Abschattungen konnten in gewissem Umfang durch eine Messanordnung mit mehreren, in einem größeren Winkel zueinander angeordneten Laserscannern, reduziert werden. Bei großen oder räumlich dicht positionierten verdeckenden Objekten, wie bspw. parkende PKW, war durch eine solche Modifizierung am Messsystem nur eine geringfügige Verbesserung zu erreichen. Meist entsprach die zu erwartende Verbesserung durch einen solchen Aufbau jedoch nicht der dazu aufzuwendenden Investition in zusätzliche Laserscanner und die damit verbundene wachsende Komplexität des Messsystems.

Die Verdeckung von Objekten war in den Bilddaten weniger relevant, da mehrere Kameras zum Einsatz kamen. Wenn während der Fahrt in festen Abständen Bilder in verschiedene Richtungen aufgenommen wurden, war, ähnlich wie bei einer Messanordnung mit mehreren Scannern wahrscheinlich, dass ein verdecktes Objekt aus einem anderen Aufnahmewinkel sichtbar wurde. Für die Kamerabilder galt: Je größer oder dichter angeordnet die verdeckenden Objekte waren, desto geringer fiel der Vorteil durch die Verwendung von Bildern aus mehreren Blickwinkeln aus.

Für die Zusammenführung der Daten von Kameras und Laserscannern galt, dass Lücken in der Punktwolke, die durch Abschattungen entstehen, nicht ohne weiteres „nachträglich“ gefüllt werden konnten. Dies war auch dann der Fall, wenn die dort für den Laserscanner unsichtbaren Objekte auf einem Kamerabild aus einem anderen Aufnahmewinkel dargestellt waren.

Eine möglichst hohe Montage von Laserscanner und Kameras war ein effektiver Weg, Verdeckungen zu reduzieren. Durch den „Blick von oben“ konnte der abgeschattete Bereich minimiert werden. Gänzlich ließen und lassen sich Verdeckungen im öffentlichen Raum jedoch nicht vermeiden.

8.9 Einfluss von sich bewegenden Objekten

Während der Einfluss der Verdeckungen durch stationäre Objekte beschränkt ist, hatten sich bewegende Objekte einen größeren Einfluss auf das Ergebnis.

8.9.1 Einfluss sich bewegender Objekte auf die Punktwolke

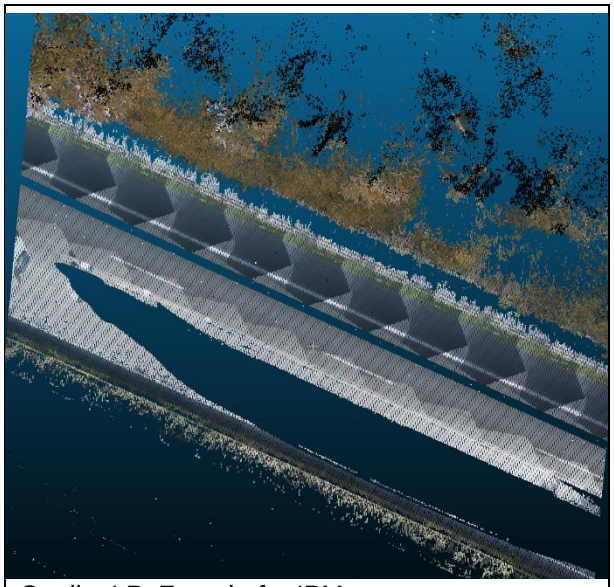
Im Zuge der Erfassung sich bewegende Objekte mittels Laserscanner erscheinen sich bewegende Objekte entweder vergrößert oder verkleinert, je nachdem, ob diese sich mit oder entgegen der Fahrtrichtung bewegen.

Objekte, deren Bewegung entgegengesetzt zu der des Messfahrzeugs erfolgt, erscheinen in der Punktwolke verkürzt, da sie sich um eine entsprechend ihrer Geschwindigkeit verkürzte Zeitdauer im Erfassungsbereich des Scanners befinden. Wenn ein Objekt sich in derselben Richtung wie das Messfahrzeug bewegt, können mehrere Effekte auftreten, die jedoch alle zu einer künstlichen Vergrößerung des Objektes führen.

Im ungünstigsten Fall befindet sich ein Objekt während einer längeren Zeitspanne im Erfassungsbereich des Laserscanners. Dies kann bspw. auf mehrspurigen Straßen der Fall sein, wenn sich ein Fahrzeug neben dem Messfahrzeug befindet und sich dabei nur geringfügig schneller oder langsamer bewegt. Wird ein Fahrzeug vom

Messfahrzeug überholt, erscheint dieses in der Punktwolke entsprechend verlängert. Dies ergibt sich aus dem Zeitraum, in dem das Fahrzeug sich in der Erfassungsrichtung des Laserscanners befunden hat.

Wird das Messfahrzeug von einem anderen Fahrzeug überholt, so wird dieses zusätzlich „gespiegelt“, da die Front des Fahrzeugs als erstes in den Erfassungsbereich des Laserscanners eintritt.

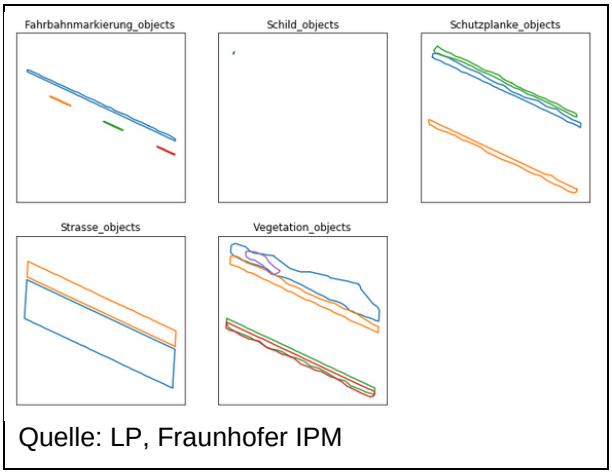


Quelle: LP, Fraunhofer IPM

Bild 122: Darstellung eines sich bewegenden Fahrzeuges neben dem Messfahrzeug

Bei der Übertragung der Klasseninformation aus den semantisch segmentierten Bilddaten waren solche sich bewegenden Objekte problematisch, da sie größere Bereiche der Umgebung verdeckten. Weiterhin konnten die dazu gehörigen Punkte nur dann korrekt klassifiziert werden, wenn das Objekt auch in den Kameradaten sichtbar war.

Lücken in der Punktwolke konnten bei der Berechnung der konkaven Hülle einzelner Objektinstanzen mit einer geeigneten Parametrisierung bis zu einer bestimmten Größe geschlossen werden. Für die in Bild 122 dargestellte Szene zeigen die auf die Straßenebene projizierten 2D-Polygone für die Straßenoberfläche die Lücke in der Punktwolke nicht mehr (siehe Bild 123).



Quelle: LP, Fraunhofer IPM

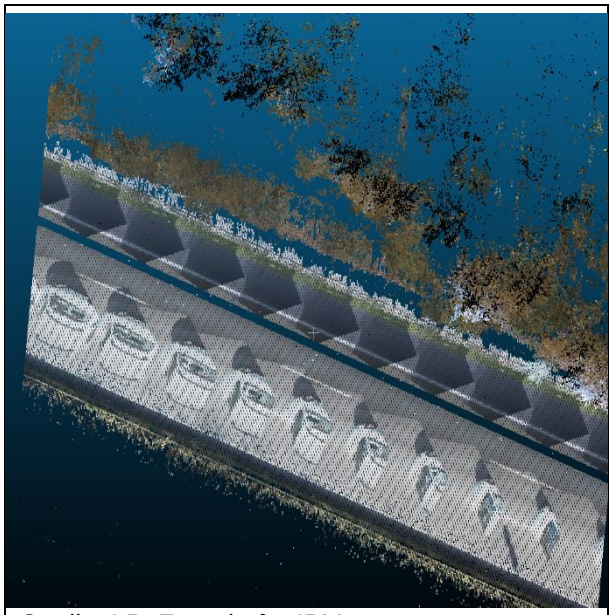
Bild 123: Klassifizierte 2D-Polygone zum oben dargestellten Abschnitt, in dem das Messfahrzeug überholt wird: Die Lücke in der Punktwolke wurde bei der Berechnung der konkaven Hülle korrekt geschlossen

Es ist allerdings nicht immer möglich, größere Lücken zu schließen, vor allem, wenn die Szene komplexer ist als die hier dargestellte mehrspurige Straße.

8.9.2 Einfluss sich bewegender Objekte auf die Klassifizierung

Sich bewegende Objekte sind nicht nur im Aufnahmebereich des Laserscanners problematisch, sondern auch, wenn sie sich über längere Zeit im Blickfeld einer Kamera befinden. Es reduzierte sich die Sicherheit, mit der ein solcher Bereich in der Punktwolke der korrekten Klasse zugeordnet werden konnte.

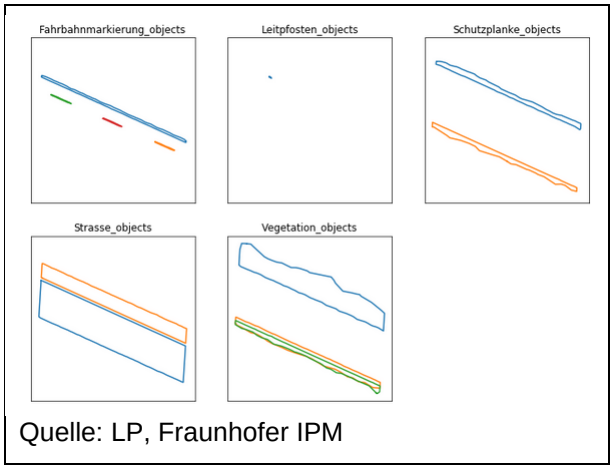
Bei einer Befahrung mit dem Messsystem trat dies bei der vorderen und der hinteren Kamera auf, wo eine Überlagerung von klassifizierten Bilddaten und Punktwolke dann nicht mehr optimal möglich war. Bei den Seitenkameras war ein ähnlicher Effekt zu erkennen, wenn das Fahrzeug den Aufnahmebereich des Laserscanners verlassen hat (vgl. Bild 124). Die Informationen aus Laserscanner und Kamera waren in diesen Bereichen widersprüchlich.



Quelle: LP, Fraunhofer IPM

Bild 124: Befand sich ein Fahrzeug beim Überholen nicht mehr im Aufnahmebereich des Laserscanners, war die Information bei der Fusionierung von Punktwolke und Kamerabildern nicht mehr eindeutig.

Die Problematik bei der Klassifizierung der Punktwolke konnte teilweise reduziert werden, da die Einfärbung und die Klassifizierung unterschiedlich vorgenommen wurden. Bei den hier gezeigten Ausschnitten aus der RGB-Punktwolke wurde die Farbinformation pro Punkt nach der Rückprojektion der verschiedenen Bilder durch Mittelung berechnet. Im Gegensatz dazu wurde die Klasseninformation pro Punkt durch eine Mehrheitsentscheidung festgelegt. Da der betroffene Abschnitt der Straße in der Regel auf Bildern der anderen Kameras ohne dieses Fahrzeug zu sehen war, konnte ein Punkt auf der Asphaltoberfläche dennoch korrekt klassifiziert werden. Obwohl die RGB-Punktwolke in Bild 124 auf der linken Fahrspur viele Fahrzeuge zeigt, wurde die gesamte Fläche korrekt als Straße klassifiziert, während keine Polygone für Fahrzeuge ausgegeben wurden, wie in Bild 125 ersichtlich ist.



Quelle: LP, Fraunhofer IPM

Bild 125: Klassifizierte 2D-Polygone zum oben dargestellten Abschnitt, in dem Fahrzeuge in der RGB-Punktwolke auf die Straße projiziert wurden: durch die Mehrheitsentscheidung bei der Klassifizierung wurde der Bereich korrekt als Straßenoberfläche erkannt.

Es trat auch den Fall auf, dass sich z.B. im dichten Verkehr Fahrzeuge dauerhaft dicht vor und hinter dem Messfahrzeug befanden. Da auf den Kamerabildern die Straße in weiten Bereichen verdeckt war, konnte keine korrekte Klassifizierung der Straßenoberfläche stattfinden, da in der Mehrheitsentscheidung pro 3D-Punkt das Klassenlabel-Fahrzeug überwog.

Eine Reduzierung des Einflusses von sich bewegenden Objekten in den Bilddaten konnte auch dadurch erreicht werden, wenn bei der Befahrung auf einen möglichst großen Abstand zum vorausfahrenden Fahrzeug geachtet und auf der Rückseite des Messfahrzeugs nachfolgende Fahrzeuge durch ein Schild auf eine Abstandshaltung hingewiesen wurden.

8.10 Erforderliche Auflösungen

Eine hohe Auflösung der Punktwolken- und der Kameradaten liefert mehr Daten und somit mehr Informationen. Anhand der Erfahrungen im vorliegenden Forschungsprojekt ließen sich die Anforderungen an die Auflösungen anhand des Zusammenspiels der eingesetzten Sensoren und Algorithmen näher definieren.

8.10.1 Auflösung der Punktwolken

Von der Auflösung der Punktwolke hing die Genauigkeit ab, mit der Objekte im dreidimensionalen Raum beschrieben und in der Folge auch in 2D auf eine Fläche projiziert wurden. Die Auflösung einer Punktwolke, die mit einem bestimmten Laserscanner erreicht werden kann,

hängt dabei immer von mehreren Faktoren ab (vgl. Kap. 2.1.6 und 2.1.7).

Zunächst war die Samplingrate zu berücksichtigen, welche die Anzahl der pro Sekunde gemessenen 3D-Punkte spezifizierte. Dieser Wert legte die Höhe der Auflösung innerhalb eines Scanprofils fest. Bei dem in diesem Projekt eingesetzten Clearance Profile Scanner (CPS) des Fraunhofer IPM betrug die Samplingrate bis zu 2 Millionen Punkte pro Sekunde. Relevant war ebenfalls die Profil- bzw. Scanfrequenz. Diese bestimmt, wie oft der Aufnahmebereich des Scanners pro Sekunde vom Laserstrahl abgetastet wird.

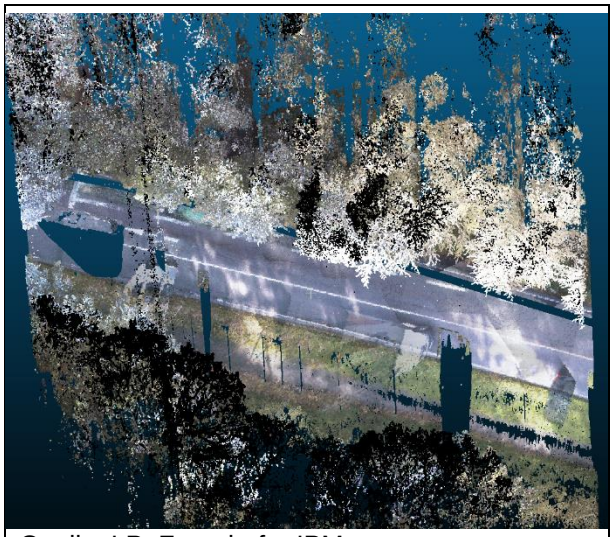
Bei einer Profilrate von bis zu 200 Hz, wie dies mit dem CPS-Scanner erfasst wird, konnten Objekte in 5 m Entfernung mit einer Punktdichte von bis zu 6,2 mm innerhalb eines Profils aufgelöst werden. Zusammen mit der Fahrgeschwindigkeit des Messfahrzeugs ergab sich aus der Scanfrequenz außerdem der Abstand zwischen zwei aufeinanderfolgenden Scanprofilen. Bei 50 km/h betrug der Profilabstand ca. 7 cm. Soll eine Befahrung mit straßentypischen Geschwindigkeiten durchgeführt werden, ist die Scanfrequenz die kritischere Größe bezüglich der Auflösung der Punktwolke.

8.10.2 Punktdichte und Befahrungsgeschwindigkeit

Da Laserscanner kontinuierlich Daten aufzeichnen, war, wie oben ausgeführt, die Punktdichte in der Punktwolke direkt von der Befahrungsgeschwindigkeit abhängig. Daraus ergab sich zum einen ein abschnittsweise erhöhter Speicherbedarf, wenn sich das Messfahrzeug sehr langsam bewegte oder z.B. an Kreuzungen zum Stillstand kam.

Außerdem konnte eine lokal stark erhöhte Punktdichte zu „False Positives“ bei der Extraktion der einzelnen Objektinstanzen führen, da hier ein Schwellenwert für eine minimale Anzahl an Punkten derselben Objektklasse in einer räumlichen Nachbarschaft angewandt wurde. War die Punktdichte stark überhöht, wurden potentiell räumlich sehr kleine Ansammlungen von Punkten zu eigenständigen Objekten.

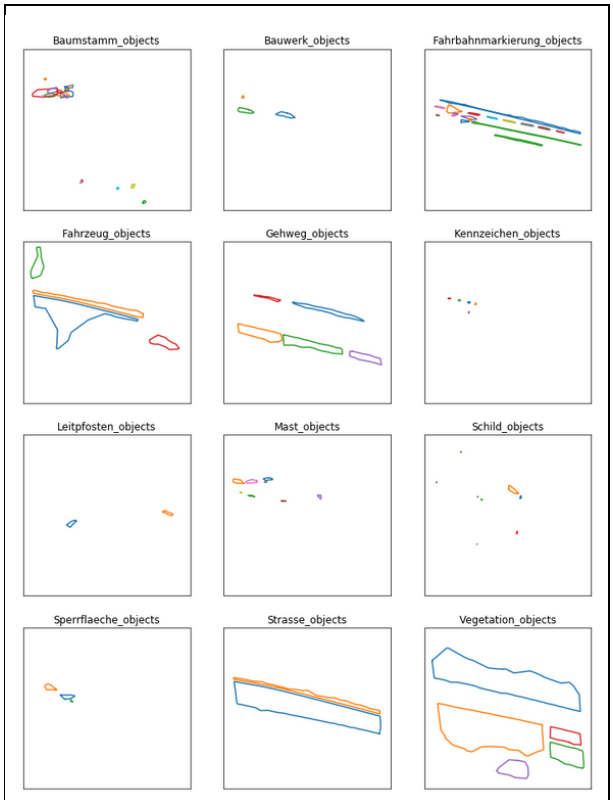
Ein Beispiel für dieses Phänomen zeigt Bild 126 an einer Stelle, an der das Messfahrzeug kurzzeitig sehr langsam fuhr.



Quelle: LP, Fraunhofer IPM

Bild 126: Punktwolke mit sehr hoher Punktdichte aufgrund von geringer Befahrungsgeschwindigkeit

Die für diese Szene resultierenden Polygone in 2D zeigten eine erhöhte Anzahl an „False Positives“-Fälle, z.B. für die Objektklassen Fahrzeug, Mast und Schild (Bild 127).



Quelle: LP, Fraunhofer IPM

Bild 127: Klassifizierte 2D-Polygone zum oben dargestellten Abschnitt mit sehr hoher Punktdichte

Bei sehr hohen Befahrungsgeschwindigkeiten konnte es zum umgekehrten Effekt kommen. Einzelne Objekte wurden gefiltert, da sie nicht die

erforderliche Mindestanzahl an Punkten erreichten („False Negatives“). Um beide Effekte zu minimieren, müsste der Algorithmus für die Extrahierung der einzelnen Objektinstanzen um eine Betrachtung der mittleren Punktdichte im aktuellen Befahrungsabschnitt erweitert werden. Dies um den Schwellwert für die minimale Punktzahl eines einzelnen Objektes entsprechend anzupassen.

8.10.3 Punktdichte und Prozessierungszeit

Bezüglich der Berechnungszeit des Algorithmus aus der PCL, der zur Extraktion einzelner Objekte zum Einsatz kam, spielte die lokale Punktdichte der Punktwolke eine zentrale Rolle. Es war zielführend, für die Prozessierung Streckenabschnitte mit fester Länge auszuwählen (vorliegend Fall 50 m). Die Anzahl an Punkten in einem Abschnitt wird unmittelbar durch die Befahrungsgeschwindigkeit beeinflusst. Sie variierte dabei sehr stark in einer Größenordnung von ca. 2 bis 15 Mio. Punkte pro 50 m-Abschnitt.

Wurden nach der Übertragung der Klassenlabel in die Punktwolke sehr viele Punkte mit demselben Klassenlabel ausgewiesen, stieg die Prozessierungsdauer überproportional an. Die PCL gab für den EuclideanClusterExtraction-Algorithmus, der an dieser Stelle verwendet wurde, keine explizite Komplexitäts- bzw. Laufzeitanalyse an. Durchgeführte Experimente zeigten, dass ab einer Anzahl von ca. 200.000 Punkten die Prozessierungsdauer für das Clustering der entsprechenden Klasse mit bis zu 10 Minuten nicht mehr akzeptabel war.

Der Arbeitsspeicher, welcher der CPU zur Verfügung stand, war hierbei nicht voll ausgelastet. Möglicherweise konnte eine Analyse der Belegung der einzelnen Cache-Bereiche des Prozessors Anhaltspunkte liefern; in jedem Fall erschien jedoch die Nutzung des Algorithmus zunächst nur eingeschränkt. Als Arbeitsprozess wurde ein Downsampling der Punktwolke mittels eines Octrees auf ein 3D-Raster mit 5 cm Seitenlänge angewendet. Der Aufwand für die Berechnung des Octree-Rasters war deutlich geringer als die bei der Objektextraktion eingesparte Prozessierungszeit.

Die resultierenden Objektpunktwolken lagen somit in reduzierter Auflösung vor. Dies war für die meisten Anwendungsfälle hinreichend und trug zusätzlich zur Reduzierung des benötigten Speicherplatzes für die Ergebnisse bei. Sollte beim Downsampling die RGB-Information erhalten bleiben, konnte an dieser Stelle eine Mittelung der Farbinformation im entsprechenden Volumenelement stattfinden bzw. die Farbe des räumlich nächsten Punktes übernommen werden. Beides hätte jedoch wiederum eine Verlängerung der Prozessierungsdauer zur Folge.

Die Genauigkeit der Ergebnisse in Form der 2D-Polygone wurde durch das Downsampling nur gering beeinflusst, da auch der ConcaveHull-Algorithmus aus der PCL an dieser Stelle durch die Auswahl einzelner Punkte für die Hülle eine implizite Reduzierung der Auflösung bewirkte.

Der Zeitaufwand für die semantische Segmentierung nahm in der Prozessierung insgesamt den geringeren Teil in Anspruch. Bei einer Auslösung der Kameras im Abstand von 5 m und dem Einsatz von 5 Kameras wurden pro Kilometer 1.000 Bilder aufgezeichnet. Die Bilder hatten eine Auflösung von 5 Megapixeln. Für die semantische Segmentierung wurden sie mit dem Faktor 0,6 skaliert. Die Berechnung der Segmentierung durch das KNN benötigte etwa 1,2 s auf einer Titan X-GPU, was pro Kilometer 1.200 s bzw. 20 Minuten Laufzeit ergab.

Die nachfolgende Prozessierung im 3D-Raum hing stark von der Befahrungsgeschwindigkeit innerhalb einer Prozessierungseinheit und der daraus resultierenden lokalen Punktdichte ab. Mit der Parametrisierung der finalen Prozessierung (insbesondere dem Downsampling von Objektpunktwolken ab 200.000 Punkten) betrugt die Prozessierungszeit pro Prozessierungseinheit von 50 m zwischen 5 Minuten bei geringer Punktdichte außerorts und ca. 20 Minuten bei hoher Punktdichte innerorts. Der zeitaufwändigste Teil der Prozessierung, die Extrahierung der Einzelobjekte, wurde parallel auf 12 CPU-Kernen ausgeführt. Pro Kilometer erforderte dies zwischen 100 und 400 Minuten Prozessierungszeit.

Die Laufzeit der Prozessierungsschritte zusammen betrug damit von der Eingabe der Messdaten bis zur Ausgabe der 2D-Polygone und der Punktwolken pro Einzelobjekt ca. 120 bis 420 Minuten pro Kilometer. Die Prozessierungszeiten bezogen sich auf die Verarbeitung der Daten auf einer einzelnen modernen Deep Learning-Workstation aus der Preisklasse unterhalb von 10.000 EUR.

Die Prozessierung konnte an verschiedenen Stellen hinsichtlich der Laufzeit optimiert werden. Dies stellte jedoch, je nach dem erforderlichen Grad an Beschleunigung, einen erheblichen Implementierungsaufwand dar.

8.10.4 Auflösung der Kamerabilder

Die Kamerabilder sollten eine möglichst hohe Auflösung erzielen. Dies war, wie es sich im Zuge der Projektbearbeitung zeigte, nicht immer zielführend. Prinzipiell enthalten höher aufgelöste Bilder zwar mehr Informationen; für die Klassifizierung der für das Forschungsprojekt relevanten Objektklassen waren aber kleine Details in der Regel weniger entscheidend. Wichtiger war es, „offensichtliche“ Merkmale zu lernen, die auch weniger davon abhingen, in welcher Entfernung ein Objekt aufgenommen wurde.

Ebenso war zu berücksichtigen, dass eine höhere Auflösung nicht immer eine bessere Bildqualität bedeutete, sondern dadurch bspw. ein höheres Rauschen der Bilddaten hingenommen werden musste.

Um eine möglichst robuste Klassifizierung der Kamerabilder zu erlauben, war es wesentlich, dass die Bilddaten auch unter schwierigen Belichtungsverhältnissen möglichst einheitlich aufgenommen wurden. Wichtiger als eine sehr hohe Auflösung der eingesetzten Kameras war demnach:

- Eine stabile Montage, um eine stabile Kalibration zu gewährleisten.
- Hochgenaue Auslösezeitpunkte, um eine möglichst exakte Fusionierung mit der Punktwolke zu ermöglichen.
- Wenig Rauschen bei ungenügender Ausleuchtung.
- Wenig Lichtreflexe in den Bildern auch bei ungünstigen Einfallswinkeln.
- Wenig Blooming bei überbelichteten Bildbereichen.

Bei der Klassifizierung von Messdaten, die aus verschiedenen Kamerasystemen stammen, war darauf zu achten, dass die Unterschiede auf ein Minimum reduziert wurden, z.B. hinsichtlich der Brennweite der Objektive, der Montagewinkel auf dem Fahrzeug und der Farb- und Beleuchtungseinstellungen. Alternativ musste das KNN mit manuell annotierten Daten von allen Kamerasystemen trainiert werden, was einen erheblichen Mehraufwand für das Labeling mit sich brachte.

Die verwendeten Kameras lieferten Bilder mit einer Auflösung von 5 Megapixeln. Es zeigte sich, dass dies für die Klassifizierung ausreichend war. Zu beachten war weiterhin, dass bei der semantischen Segmentierung von Bilddaten vom KNN zur Klassifizierung eines Pixels ein sogenanntes Receptive Field um das Pixel herum in Betracht gezogen wurde. In vorliegender Netzarchitektur betrug die Größe des Receptive Field 404 Pixel (vgl. Kap. 5.3.2). Wurden bei gleichbleibender Brennweite Bilder mit einer höheren Auflösung verwendet, war zu berücksichtigen, dass sich die Menge an Information, welche im Receptive Field jeweils zur Verfügung stand, ebenfalls ändert.

Weiterhin stellte beim Training tiefer Neuronaler Netze der Speicher der Grafikkarte einen limitierenden Faktor dar. Die Kamerabilder wurden für das Training des KNN aus diesem Grund mit einem Faktor von 0,6 skaliert. Dies lag darin begründet, dass bei Nutzung der maximalen Bildauflösung der Arbeitsspeicher der Grafikkarte mit 12 G nicht mehr ausreichend war. Sollten Bilder mit höherer Auflösung für das Training genutzt werden, war die Infrastruktur für das Training aufwändig anzupassen bzw. zu erweitern.

Grundsätzlich können vom KNN nach dem Training Bilder in einer höheren Auflösung semantisch segmentiert werden. Es war jedoch zu beachten, dass die typische Menge an Merkmalen, welche dem Netzwerk im Receptive Field pro Pixel zur Verfügung standen, mit der Auflösung variierte, was die Klassifizierungsleistung bei höherer Auflösung für Objekte negativ beeinflussen konnte. Dies war dann der Fall, wenn sich die Objekte nahe an der Kamera befanden.

Eine Auflösung von deutlich weniger als 5 Megapixel pro Kamera ist demnach nicht zu empfehlen, da bei Bedarf Algorithmen implementiert werden müssen, welche die Bildinformation auch außerhalb des KNN nutzen und dann von einer ausreichend hohen Auflösung profitieren können. Auch für eine visuelle Kontrolle einzelner Objekte auf den Kamerabildern war es vorteilhaft, wenn viele Details in den Bildern zu erkennen waren.

8.11 Nutzung der PCL (Point Cloud Library)

In den Algorithmen der prototypischen Software wurden, wo es möglich und sinnvoll erschien, Standard-Implementierungen aus der PCL verwendet. Darunter fielen z.B.:

- Datenstrukturen für die Verarbeitung von Punktwolken.
- Generierung von Bounding Boxes.

- Nutzung der Octree- und k-d-tree-Datenstrukturen mit zugehörigen Funktionen.
- Berechnung der Punkte der konkaven Hülle einzelner Objekte in 2D.

Grundsätzlich stellte die PCL für verschiedenste Anwendungen im Bereich der Punktwolken-verarbeitung und -visualisierung, teilweise auch für die Objekterkennung, nützliche Funktionalität zur Verfügung. In solchen Fällen war es zielführender, diese Bibliotheksfunktionen zu verwenden anstatt diese selbst zu implementieren.

Weiterhin hat sich im Forschungsprojekt gezeigt, dass die Laufzeit des EuclideanClusterExtraction-Algorithmus der PCL deutlich überproportional anstieg, wenn Punktwolken von der Größe als Input verwendet wurden, wie sie in vorliegendem Forschungsprojekt vorlagen. Dies bedeutet mit teilweise wesentlich mehr als 200.000 Punkten für einzelne Klassen in jedem Prozessierungsabschnitt von 50 m Länge (vgl. Kap. 8.10.3). Hier wurde die Nutzbarkeit durch die Implementierung eines auflösungsabhängigen Downsamplings hergestellt.

Insgesamt war die Dokumentation zur PCL nicht zufriedenstellend. Dies galt bspw. für den ConcaveHull-Algorithmus, dessen Implementierung auf die externe QHull-Bibliothek zurückzuführen war. Die Implementierung und die Parametrisierung war in der Dokumentation der PCL nur unzureichend beschrieben.

Dennoch boten die PCL viele umfangreiche und von einem großen Nutzerkreis getestete Funktionen, welche die Implementierung von Software zur Verarbeitung, Klassifizierung und Visualisierung von Punktwolken wesentlich erleichterte.

9 Datenübergabe und Software

Es wurden folgende Daten und Software übergeben:

Daten:

- Trainingsdaten des neuronalen Netzes
- Rohdaten Befahrung Teststrecke
- Trajektorie Teststecke
- Referenzdigitalisierung der Teststrecke (Datenbank)
- Prozessierte Daten der Teststrecke (inkl. Bild-Viewer)
- Extrahierte Daten des Neuronalen Netzes (Rohdaten)
- Konvertierte Daten des neuronalen Netztes (Datenbank)

Software:

- Programm zur automatischen Objekterkennung
- Programm zur Konvertierung der Daten in die Datenbank

Jede Software wird mit einer Kurzanleitung ausgeliefert, die die Funktionsweise, die Abhängigkeiten anderer Bibliotheken und die Handhabung des Programms beschreibt

10 Analyse des Messfahrzeuges „MESUV“ als Vergleichsfahrzeug

Die Bundesanstalt für Straßenwesen (BASt) betreibt das Messfahrzeug „MESUV“ (Messsystem zur Erfassung von Straßen- und Verkehrsdaten), welches mit Kameras und einem Laserscanner ausgestattet ist. Derzeit können diese Daten nicht mit dem im Rahmen dieses Forschungsprojektes entwickelten Algorithmus zur automatischen Objekterkennung analysiert werden. Im Rahmen einer Projekterweiterung sollte das Datenformat des Fahrzeugs „MESUV“ so aufgearbeitet werden, dass dieses mit den entwickelten Algorithmen im vorliegenden Forschungsprojekt weiterverarbeitet werden kann. Die Erweiterung gliederte sich in zwei Meilensteine. Der Meilenstein 1 enthielt eine Analyse der Hardware des Messfahrzeuges „MESUV“. Nach erfolgreicher Analyse sollte im Meilenstein 2 eine Implementierung erfolgen. Bei nicht erfolgreicher Analyse der Hard- und Softwarekomponenten des Messfahrzeuges „MESUV“ sollte das Projekt abgeschlossen werden.

10.1 Analyse der Hardware des Messfahrtzeuges „MESUV“

Das Messfahrzeug „MESUV“ wird seitens der BASt für Untersuchungen zum Einfluss von Straßengeometrieparametern auf die Verkehrssicherheit eingesetzt.

Bild 128: MESUV (Messsystem zur Erfassung von Straßen- und Verkehrsdaten) Messfahrzeug der BASt

Das Fahrzeug ist mit einem kombinierten GPS- und Inertialnavigationssystem, fünf Frontkameras, einem Laserscanner am Heck und einem kamerabasierten „Mobile Eye“-System ausgestattet. Die technische Ausrüstung ermöglicht die Nachmessung der Trassierung von Straßen, das Abmessen von Elementen des Straßenraums, die Erfassung der Querneigung der Straße, der Fahrstreifenbreite so wie der gefahrenen Geschwindigkeit und des Abstandes vorausfahrender Fahrzeuge.

Im Rahmen einer Testmessung wurde das Fahrzeug dem Forschungsnehmer in Dresden am 08.09.2017 vorgestellt. Hierbei wurden drei Testfahrten durchgeführt. Es handelte sich um die Aufnahmen von speziell eingerichteten Kalibrierungsfeldern in Dresden, welche LP für die Realisierung der Einmessung der eigenen Messfahrzeuge verwendet.

Des Weiteren wurde eine Bestandsaufnahme der verwendeten Hardware durchgeführt. Hierbei wurden folgende Komponenten identifiziert. Da die Hardware fest am Fahrzeug verbaut ist, waren keine Seriennummern erkennbar.

Trägerfahrzeug: VW Multivan T5

Stromversorgung: 230V über Sinus-Wechselrichter

Kamerasystem Umgebung:

Blickrichtung: in Fahrtrichtung

Kameras: 5 Kameras

- Allied Vision Guppy Pro F-146 (F-201C) Typ nicht sicher erkennbar
- Sensorformat ½ "
- Shuttertyp Global shutter/Prog Scan
- Auflösung 1388 (H) x 1038 (V) Bildpunkte mit 4,65 µm Pixelgröße
- Pixel Bit depth 8 oder 12 bit
- Bildwiederholfrequenz: 17 (fps)
- Hersteller: Allied Vision
- Objektivanschluss: C-mount
- Objektive: Tamron 219 HB mit 8 mm Brennweite (Information AG)
    - Sichtbereich: bei einem ½ ” Sensor - 43.6 ° horizontal x 33.1 ° vertikal

- Schnittstelle:FireWire A

Ausrichtung: die fünf Kameras decken in der horizontalen Achse einen +/- ~90 ° Bereich vor dem Fahrzeug ab.

Die technischen Parameter und die Ausrichtung der Kameras sind für eine Erfassung des in Fahrtrichtung liegenden Straßenraumes geeignet. Die Kameras sind so angeordnet, dass diese ein 180° Panorama abbilden. Für die Auswertung im vorliegenden Forschungsprojekt sind drei Kameras mit Blickrichtung in Fahrtrichtung hinreichend. Hingegen würde eine Kamera, die den rückwärtigen Bereich erfasst, verdeckte Bereiche im Seitenraum noch besser erfassen. Grundsätzlich ist festzuhalten, dass die verwendete Kameratechnik für das aktuelle FE Projekt einsetzbar ist.

Laserscanner:

Hersteller: SICK

Typ: LMS151-10100 (Typ ist nicht ersichtlich / Typschild nicht sichtbar / Bezeichnung aus C# Code)

Aufnahmewinkel Vertikal:270 °

Winkelauflösung: 0.25 ° - 0.5 °

Messbereich: 0.5 m ... 50 m

Messbereich bei

10 % Remission: 18 m ... 30 m

90 % Remission: 20 m ... 50 m

Scan-Frequenz: 25 Hz / 50 Hz

Betriebstemperatur: – 40 °C ... + 60 °C

Schnittstellen: Ethernet, Serielle Schnittstelle und CAN Bussystem

Gewicht: 1.1 kg

Positionierungssystem: IXblue LANDIS GPS/INS Positionierungssystem

Das System besteht aus einer GPS Antenne die auf dem Dach montiert ist, einem Odometer (Peiseler) und einer Zentraleinheit mit integriertem Mems-Kreiselsystem. Das System wird laut Datenblatt standardmäßig mit einem RTK (Real Time Kinematik) Modul betrieben. Dieses ermöglicht in Echtzeit Korrektursignale einer Basisstation (z.B. von Serviceprovidern wie SAPOS oder ASCOS) zu verwenden um die Positionsgenauigkeit zu verbessern.

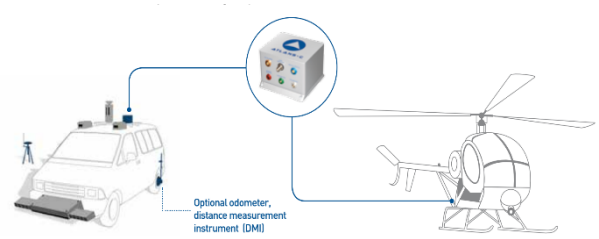


Bild 129: GPS/INS Positionierungssystem Atlans-c der Firma IXblue Verwendungsschema [Quelle IXblue]

Gewicht: 2.6 kg

Größe: 160 mm x 160 mm x 113 mm

Embedded GNSS: L1/L2, GPS, GLONASS, SBAS, RTK, TERRASTAR ready

Stromverbrauch: < 22 W

Spannung: 12 bis 33 VDC

Erreichbare Genauigkeiten:

Positionierungssystem [Quelle IXblue]

LAND PERFORMANCE

	GNSS RTK*	GNSS PPK**	GNSS RTK* with 60 seconds dropout duration	GNSS PPK* with 60 seconds dropout duration
Heading accuracy [1][2]	0.02°	0.02°	0.02°	0.02°
Roll and pitch accuracy [2]	0.008°	0.005°	0.015°	0.015°
Horizontal accuracy (X,Y) [2]	0.035 m	0.02 m	0.35 m ***	0.15 m ***
Vertical accuracy (Z) [2]	0.05 m	0.05 m	0.3 m ***	0.1 m ***
Range	Heading: 0° to 360° Roll: -180° to +180° Pitch: -90° to +90°			
Setup time	5 min stationary +15 min in motion (typical)			

[1] Secant latitude = 1 / cosine latitude
[2] RMS values
Actual results are dependant upon satellite configuration, atmospheric conditions and other environmental effects.
* RTK: real-time kinematic
** PPK: post-processed kinematic using ATLANS Post-Processing Software
*** Values with typical vehicle dynamics and environment
The performance results require using a distance measurement unit.

Bild 130: Angaben zur Genauigkeit IXblue Atlans-c

Da die Postprocessing-Software in der Datenaufbereitung nicht genutzt wird, ist eine absolute Genauigkeit ohne Korrekturdatenverlust von +/- 4 bis 35 cm anzunehmen.

Kamerasystem: MOBILEYE C2-170

Das System dient zur Spurüberwachung und zur Abstands- bzw. Geschwindigkeitsmessung zum vorrausfahrenden Fahrzeug. Der im Sensor wird über den Can-Bus ausgelesen, sodass die Daten des Sensors in der Aufnahmesoftware verfügbar sind.

Bild 131: Kamerasystem: MOBILEYE C2-170

Kommunikation: CAN-BUS

Sensor-Sichtfeld: 40 ° x 30 ° (B x H)

Sensor-Typ: CMOS Aptina MT9V024 (1/3 ”) RCC

Auflösung Sensor: 752 Hx 480 V - aktive: 640 H x 480 V

Pixelgröße: 6.0 µm x 6.0 µm

Typische Detektordistanz: 70 m

Echtzeit Bildverarbeitung: 15 Bildern pro Sekunde

Eingangsspannung: 12-28 VA

Mit dem Sensor können Verfolgungsfahrten durchgeführt und Relativgeschwindigkeiten zu vorrausfahrenden Fahrzeugen ermittelt werden.

Folgende Daten werden in der Erfassungs-Software vom Sensor ermittelt:

Markierungslinien:

- DistanceToLeftLaneMark
- DistanceToRightLaneMark
- RoadSlope
- RoadCurvature;
- TwoLanesConfidence
- RightLaneConfidence
- LeftLaneConfidence

Hindernisse/Fahrzeuge (Obstacle):

- IsValid
- IsCIPV
- ID
- Distance
- Angle
- Width
- RangeRate

Die Daten werden aus dem Sensor über den CAN-Bus ausgelesen und weiterverarbeitet. Der Sensor hat im Weiteren keinen Einfluss auf die automatische Objektextrahierung, da dieser hauptsächlich für die Fahrbahnbreiten und die Verfolgungsfahrten des Messfahrzeuges Anwendung findet

Wegstreckensensor: Peiseler MT 1000E KFZ

Dieser Sensor dient zur hochgenauen Erfassung des gefahrenen Weges und zur Unterstützung des GPS/IMU gestützten Positionierungssystems. Der verwendete Sensor hat die Bezeichnung Peiseler MT 1000E KFZ. Dieser Sensor erzeugt 1.000 Pulse je Radumdrehung. Durch einen Kalibrierungsfaktor kann von den Pulsen pro Radumdrehung auf einen abgefahrenen Weg geschlossen werden.

10.2 Analyse der Software

Von dem Messfahrzeug „MESUV“ wurden drei Software Module übergeben. Die Software besteht aus jeweils einer Installationsdatei und einer Kopie einer bereits installierten Version des jeweiligen Programms. Die Software wurde größtenteils in der Programmiersprache C# mit dem .NET Framework 3.5 geschrieben und als „Click-Once“ Anwendung übersetzt. Jedes Softwaremodul besteht aus einer Anwendung (ausführbare Datei) und mehreren dynamischen Bibliotheken, die als Deploy Datei gelinkt sind. Click-Once erlaubt die Verteilung und automatische Aktualisierung von Windows-Anwendungen über Webserver und Netzwerk-laufwerke. Die Bibliotheken wurden in den folgenden Anwendungen mehrfach benutzt, sodass die Softwarebibliotheken zum Teil identisch waren.

Übergebene Software:

- BAStKalib 1_0_0_14
- BAStRecord
- BAStView
- Quellcode einer Bibliothek „RoadSegmentEstimator.dll“
- verschiedene SQL Skripte

Da zu keiner der oben genannten Software-anwendungen ein Manual oder eine Bedienungsanleitung existiert, gestaltete sich die Analyse der Software sehr schwierig. Einziger Quellcode der vom AG übergeben wurde ist der C# Quellcode der Bibliothek RoadSegment-Estimator.dll. Die verwendete Dateiversion 1.0.0.0, zum einen im Quellcode und zum anderen in der bereits kompilierten Bibliothek in der übergebenen

Software, deuten darauf hin, dass diese den gleichen Entwicklungsstand haben.

Aufgrund der fehlenden Dokumentation der Programme wurde die Software mittels dekompilieren analysiert. Dekompilierung ist eine Reverse-Engineering-Technik. Sie wird zum nachträglichen erzeugen von Quellcode benutzt, der auf einem ausführbaren Programm oder Bibliothek basiert. Der hierbei gewonnene Quellcode hat Ähnlichkeit mit dem ursprünglichen Quellcode, der zur Erzeugung des ausführbaren Programms oder Bibliothek verwendet wurde.

Alle dekompilierten Programme wurden dem AG übergeben. Diese somit entstandenen Quelltexte waren nur zum Teil kompilierbar und ließen sich nicht wieder in eine lauffähige Bibliothek oder ein Programm übersetzen. Der hier entstandene Quellcode diente ausschließlich der Analyse der übergebenen Software. 95 % des Quellcodes konnte dekompiliert werden und könnte zur Neuentwicklung einzelner Programmteile verwendet werden.

Nachfolgend werden die einzelnen Software-Typen beschrieben und analysiert:

10.2.1 Software BAStKalib:

Die Software dient zur Ermittlung der Kalibrierungsparameter des Fahrzeuges. Die hier ermittelten Werten werden für die Datenauswertung in der Software „BAStView“ benötigt.

Die Software greift auf die Hardwarekomponenten des Messfahrzeuges zu. Aus Bild 5 ist zu entnehmen, dass alle Softwarebibliotheken die verwendet werden auch bei der Software „BAStRecord“ genutzt wurden.

Die beim Messfahrzeug „MESUV“ verwendeten Kalibrierungsparameter konnten identifiziert werden. Zur Berechnung der geometrischen Transformationen, die für die einzelnen Geräte-Koordinatensysteme ins Fahrzeug-Koordinatensystem notwendig sind, wurden die unten folgenden 3x4 Matrizen verwendet.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Diese Matrizen enthalten Brennweite, Rotation, Translation und die Verschiebung der optischen Achse.

Die folgenden Daten sind teilwiese aus der Settings Datei „C:\Users\<Nutzer>\AppData\Local\TU Chemnitz\BASt\1.0.0.0\Settings.xml“ entnommen. Diese Datei setzt die Kalibrierungsparameter für die Auswerteprogramme fest.

Die Parameter für die Kalibrierung der fünf Umgebungskameras wurden mit Hilfe der OPENCV Bibliothek erstellt. Die Werte der inneren und äußeren Kalibrierung wurden vorliegend in eine 3x4 Matrix zusammengefasst. Für eine Kamera, die zum Fahrzeugkoordinatensystem transformiert ist, würde sich folgende Matrix P ergeben:

$$P = \begin{bmatrix} fx' & 0 & cx' \\ 0 & fy' & cy' \\ 0 & 0 & 1 \end{bmatrix} * [R * t]$$

mit:

fx‘; fy‘: Brennweite

cx‘; cy‘: Verschiebung der optischen Achse

Hieraus ergibt sich ein dreidimensionaler Punkt im Kamerakoordinatensystem. Der Vektor t mit Tx, Ty und Tz beschreiben die Translation und R die Rotation zum Koordinatenursprung. Der Koordinatenursprung wird in der Mitte des Fahrzeuges, in der Nähe des Schwerpunktes, angenommen.

Die Verzeichnungsparameter sind nicht dokumentiert. Nach der Analyse der eingebundenen Bibliotheken wurde angenommen, dass es sich um ein einfaches übliches Kalibrierungsmodel mit fünf Parametern handelte. Diese fünf Parameter werden wie folgt bezeichnet:

$(\mathrm{k1}, \mathrm{k2}, \mathrm{p1}, \mathrm{p2}, \mathrm{k3})$.

$$x_{distorted} = x(1 + k1r^2 + k2r^4 + k3r^6)$$

$$y_{distorted} = y(1 + k1r^2 + k2r^4 + k3r^6)$$

$$x_{distorted} = x + [2p1xy + p2(r^2 + 2x^2)]$$

$$y_{distorted} = y + [p1(r2 + 2y^2) + 2p2xy]$$

Die Verzeichnungsparameter wurden im Nachgang auf die erhaltenden Bildkoordinaten angepasst.

Somit war jeder Offset der fünf Umgebungskameras vom Fahrzeugnullpunkt zum lokalen Kamera-koordinatensystem mathematisch beschrieben. Die Kalibrierungsparameter der Umgebungskameras sind in Anhang 1 dargelegt.

Aufgrund der kombinierten Speicherung von der Matrix P, in der die innere und äußere Orientierung enthalten ist, konnten die einzelnen Parameter nicht mehr eineindeutig bestimmt werden.

Für die Kamera MOBILEYE C2-170 konnte keine Kalibrierung ermittelt werden. Hier wurde auf die vom Hersteller vorgegeben Werte zurückgegriffen.

IMU Korrekturwinkel:

Die Korrekturwinkel der IMU beschreiben das Missalignment zwischen Kreiselsensor und Fahrzeugplattform.

Rollwinkel = 0.001129221 rad

Nickwinkel = -0.01168131 rad

Gierwinkel = 0.01013284 rad

Laserscanner Boresight Alignment:

Die Beschreibung der Transformation vom Fahrzeug Koordinatensystem zum Laserscanner ist mit einer Translation [m] und 2 Rotationen [Grad] angegeben. Die Ausführung des Geräte-koordinatensystems ist unbekannt und musste geschätzt werden.

$Tx_{Scanner}$ = -0.968 m

$Ty_{Scanner}$ = 0 m

$Tz_{Scanner}$ = 1.176 m

rotxScanner = 3 °

rotyScanner = -0.2 °

rotzScanner = nicht angegeben

10.2.2 Software BAStRecord

Die Software „BAStRecord“ in der Version 1.0.0.0 ist die zentrale Aufnahmesoftware, welche im MESUV Fahrzeug installiert ist. Diese Software liest alle Daten der einzelnen Messsensoren aus, verknüpft diese Online (serialisiert die Daten) zu einem Datenstrom, der dann final als Rohdatendatei mit der Endung „*.bst“ gespeichert wird. Das Programm ist Modular aufgebaut. Jeder Sensor ist mit der entsprechenden Auslesesoftware in einer Programmbibliothek gekapselt. Folgende Abhängigkeiten von einzelnen Software-bibliotheken sind analysiert worden (siehe in Anhang 1 Bild 135).

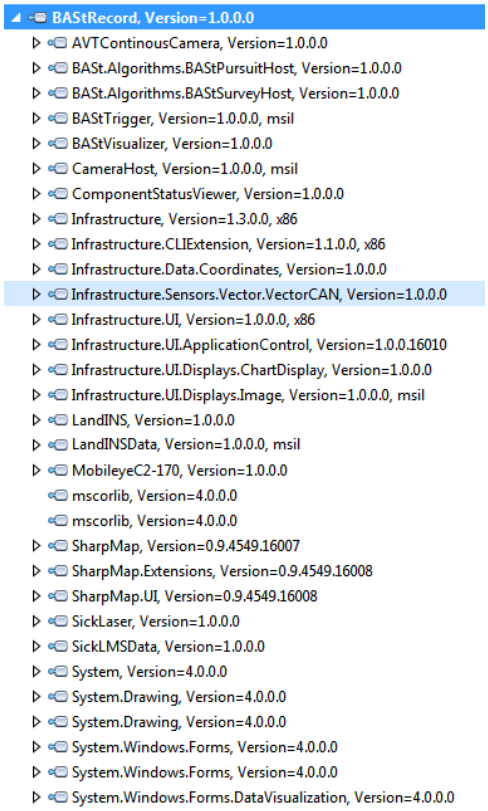


Bild 132: Software BastRecord - erste Stufe der benötigten Software Bibliotheken inkl. deren Versionsnummern

Die Software wurde mit einem eigenen Installer übergeben und installiert sich als „Click-Once“ Anwendung. Die Software stellt verschiedene grafische Anzeigen zur Verfügung, um das Fahrzeug während der Fahrt zu überwachen (siehe Bild 136).

Die analysierten Programmteile wurden als Visual Studio Projekt dem AG übergeben.

10.2.3 Software BAStView

Die Software „BAStView“ wurde in zwei verschieden Versionen übergeben. Beide Oberflächenprogramme unterscheiden sich lediglich durch zwei zusätzliche Funktionen. Dies war der Export von Bildern und die Funktion zur Reparatur von Messdaten „Stream“ (siehe Bild 137). Beide Versionen wurden Programmseitig mit der Versionsnummer 1.0.0.0 gekennzeichnet, welches die Unterscheidung der beiden Versionen erschwert. Die verschiedenen Versionen sind derzeit nur über die verschiedenen Installationsroutinen oder über das Erstellungs-datum des ausführenden Programms zu unterscheiden(Anhang 1, Bild 138).

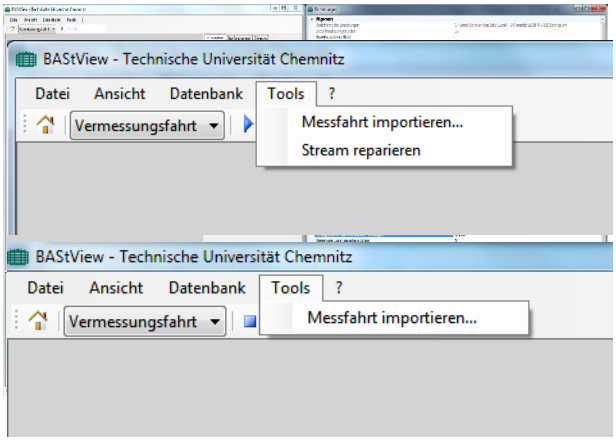


Bild 133: Unterschied in der Benutzeroberfläche der beiden übergebenden Softwareversionen mit der Versionsnummer 1.0.0.0

Die Software „BAStView“ setzt eine PostgreSQL Datenbank voraus. Aufgrund der fehlenden Dokumentation wurde einen PostgreSQL 10.4 Datenbank verwendet mit einer POSTGis v2.4 Datenbankerweiterung, die für die Speicherung von georeferenzierten Daten und Koordinaten verwendet wird. Das Programm setzt eine spezielle Datenbankstruktur voraus. Die Datenbankstruktur wurde als SQL Skript mit Übergeben. Darüber hinaus waren noch 2 weitere Skripte erforderlich, die die notwendigen Datenbanktabellen erzeugt und Funktionen neu definiert. Diese Funktionen waren in älteren Versionen vorhanden, aber in der POSTGis Version 2.4 nicht mehr.

Notwendige Skripte zur Vorbereitung der notwendigen Datenbank für das Programm „BAStView“:

- createDataBaseBASt.sql
- createAllBAStViewTables.sql
- fehlende_postgis_functions_erstellen.sql

Nach dem erfolgreichen einrichten der Datenbank konnte über die Software BAStView eine Datenbankverbindung hergestellt werden. Dies war auch Grundvorrausetzung um die Rohdaten aus einer „bst Datei“ einzulesen.

10.2.4 Ausgabe *.bst Formatbeschreibung

Die mit dem Messfahrzeug „MESUV“ erfassten Daten können in einem Rohdatencontainer über die Software „BAStRecord“ gespeichert werden. Diese Datei dient als Austauschdatei für Rohdaten. Es wird hierbei unterschieden zwischen einer Vermessungsfahrt und einer Verfolgungsfahrt.

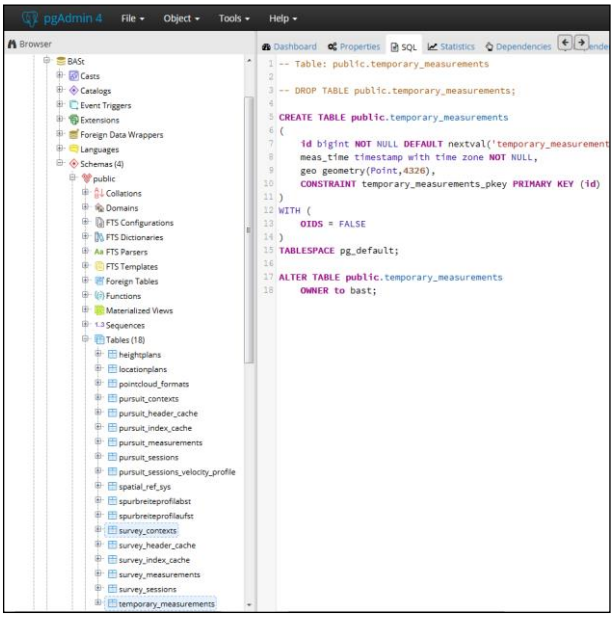


Bild 134: Datenbank PostgreSQL mit Database und den 18 vordefinierten Tabellen

Die Datei wurde über eine spezielle Serialisierung (Abbildung von strukturierten Daten auf eine sequenzielle Darstellungsform) der einzelnen Sensorrohdaten erreicht. Alle Sensordaten wurden hierzu einzeln in einer bestimmten Objektklasse gespeichert und im Anschluss in einer bestimmten Reihenfolge in die Ausgabe-Datei geschrieben. Die Objektkassen bestimmten hierbei die Reihenfolge der einzelnen Messdaten. Die Reihenfolge der einzelnen Objektklassen legte die Aufnahme fest. Jeder Sensor wurde sequenziell mit einem bestimmten Header in diese Datei geschrieben. Die Reihenfolge der Sequenz und die Abfolge der einzelnen Messwerte konnte aus dem dekompilierten Quellcode entnommen werden. Aufgrund der sehr verschachtelten Klassen-hierarchie und der verwendeten vorgefertigten Standardbibliothek von C# passten nicht alle Angaben exakt mit dem verglichenen Binary-Aufbau der Austauschdatei zusammen. Die aufgetretenen Unterschiede waren auf unterschiedliche Programversionen zurück-zuführen.

Für die Extraktion der einzelnen Messdaten aus dem Rohdatencontainer empfahl es sich nach bestimmten Schlüsselwörtern im Header zu parsen.

- Infrastructure.Data.Image
- Infrastructure.Data.GNSS.NMEA
- Infrastructure.Data.LMDscandata
- Infrastructure.Data.Mobileye

Alle hier aufgeführten Klassen/Schlüsselwörter waren im dekompilierten Code und auch in der *.bst Datei wiederzufinden. Diese Klassen waren in der oben aufgeführten Reihenfolge in der Rohdatendatei abgelegt. Abschließend konnte nicht geklärt werden, wie viele Scannerdaten zu einer Szenerie-Aufnahme (5 Kamera-Bilder) gespeichert wurden. Des Weiteren enthielt die Struktur „Infrastructure.Data.LMDscandata“ keinen Zeitstempel der einzelnen Laserscannerpunkte, welches einer nachträglichen Georeferenzierung der Punktwolke entgegensteht.

Alle Headerinformationen der einzelnen Klassen waren als JSON ähnliches Format in der Rohdatendatei gespeichert. Somit war es möglich, mit einem Hex-Editor (spezielles Programm zum darstellen hexadezimaler Dateiinhalte) die Datei zu durchsuchen (Anhang 1, **Fehler! Verweisquelle konnte nicht gefunden werden.**).

10.3 Zusammenfassung und weitere Schritte Messfahrzeug „MESUV“

Die drei übergebenen Softwareteile wurden im Rahmen der Erweiterung des Forschungsprojektes analysiert. Da weder eine Bedienungsanleitung noch eine Schnittstellenbeschreibung seitens des AG übergeben wurde, gestaltete sich die Analyse schwierig. Alle ermittelten Daten konnten nur auf Grundlage des dekompilierten Quellcodes extrahiert werden. Durch den sehr komplexen Aufbau des Quellcodes konnten Hardwarekomponenten und die Softwarefunktionsweise nur teilweise nachvollzogen werden. Die Kalibrierungsdaten der einzelnen Sensoren konnten aus einer XML-Datei extrahiert werden. Die Boresightvektoren konnten für das Front-Kamerasystem und für den Laserscanner ermittelt werden. Die Rotationen der einzelnen Sensorkoordinatensysteme waren aufgrund der fehlenden Dokumentation der Drehreihenfolgen nicht ohne Annahmen bestimmbar. Die verwendete Hardware konnte vollständig dokumentiert werden. Auflösung und Ausrichtung waren hinsichtlich des Meilensteines 2 ausreichend. Das verwendete Positionierungssystem entspricht den Genauigkeitsanforderungen (5 bis 10 cm) zur Positionierung der Einzelsensoren um messtechnisch eine gute Abbildung der erfassten Objekte realisieren zu können.

Im vorliegenden Forschungsprojekt wurde ein CPS Laserscanner mit einer Rotationgeschwindigkeit von 200 Hz und mit einer Auflösung von 5.000 Messungen im Querprofil eingesetzt. Bei einer Fahrzeuggeschwindigkeit von 80 km/h ergibt sich einen Profilabstand in Fahrtrichtung von 11 cm. Mit dieser Abtastdichte/Auflösung lassen sich schmale Objekte wie Leitpfosten und Masten im Straßenraum erfassen.

Der Sick Laserscanner, welcher am Messfahrzeug „MESUV“ montiert ist, verfügt über eine Rotationsgeschwindigkeit von 50 Hz mit max. 1.080 Messungen pro Umdrehung, entspricht nicht den verfügbaren bzw. erforderlichen Sensoren. Bei einer Fahrzeuggeschwindigkeit von 80 km/h entspricht dies einem Profilabstand in Fahrtrichtung von 45 cm. Dies ist zur Erfassung von Objekten im Straßenraum nicht ausreichend. Die verwendeten Algorithmen setzten eine Punktwolkendichte voraus, welche mindestens ein Objekt durch mehre zusammenhängenden Laserscannerpunkte abgebildet.

Der verwendete SICK-Laserscanner kann aufgrund der nicht ausreichenden Messgeschwindigkeit und damit nicht ausreichenden Punktdichte nicht für die automatische Objekterkennung in Punktwolken verwendet werden. Ein Profilabstand von 45 cm in Fahrtrichtung kann, wie bereits erläutert, vertikale Objekte nicht mit der erforderlichen Eindeutigkeit abbilden.

Durch die nicht vollständig ermittelbaren Kalibrierungswerte und der zuvor beschriebenen nicht ausreichenden Auflösung der resultierenden Scannerpunktwolke des Messfahrzeuges „MESUV“ konnten die resultierenden Messwerte nicht im Rahmen der automatischen Objekterkennung Anwendung finden. Da das Fahrzeug „MESUV“ nicht den Dichteansprüchen an die Laserpunktwolken des zum Einsatz kommenden automatischen Extraktionsprogramms entspricht, wurde ein Abschluss des Projektes empfohlen. Die Extraktionsergebnisse wurden aufgrund der nicht ausreichenden Punktdichte falsch in die Punktwolke projiziert. Hierdurch entstanden grobe Fehler in der Auswertung, welche nicht programmtechnisch abgefangen werden konnten.

11 Zusammenfassung

Als Grundlage des Forschungsprojektes „Analyse von Straßenbestandsobjekten aus Laserpunktwolken durch Mustererkennung bzw. Objekterkennung einschließlich der Georeferenzierung“ wurden Messdaten des Mobile Mapping Fahrzeuges IRIS5 und IRIS12 der Firma LEHMANN+PARTNER GmbH (LP) verwendet. Die georeferenzierten Punktwolken entstanden mit Hilfe eines CPS Laserscanners des Projektpartners IPM Fraunhofer, Freiburg.

Die ausschließliche Analyse bzw. Extraktion von Objekten auf Grundlage der reinen Punktwolke erwies sich am Anfang des Projektes als nicht zielführend. Größter limitierender Faktor war die mit der Entfernung zunehmende Dichteverringerung in den Punktwolkendaten. Dies führte zum Teil zu unterrepräsentativen Abbildungen von kleinen oder schmalen Objekten. Um dieses Defizit auszugleichen, wurden die georeferenzierten Bilddaten der Mobile Mapping Fahrzeuge als Analysehilfe verwendet. Die Bilddaten haben um ein vielfaches höheres Auslösungsvermögen im Vergleich zur Punkwolke. Zur Analyse der Daten kamen verschieden Neuronale Netze zum Einsatz, die zunächst die Bildinformationen analysieren. Für die Detektion der zuvor definierten Straßenobjekte wurden sogenannte Trainingsbilder gelabelt. Dies bedeutet, dass jeder Bereich in einer aufgenommenen Scene eindeutig einem Objekt zugeordnet werden muss. Diese Annotierung erwies sich im Nachhinein als besonders schwierig, da jeder Bearbeiter die verschiedenen Objektklassen anders generalisierte. Dies hatte zur Folge, dass kleinere Objekte, zum Teil im Hintergrund einer Scene, unterschiedlich gelabelt wurden. Das hatte wiederum Einfluss auf das verwendete Neuronale Netz. Hierzu wurden vom Projektpartner umfangreiche Tests durchgeführt, welche Netzarchitektur sich für eine Objekterkennung im Straßenraum am besten eignet. Nach der Trainingsphase des Neuronalen Netzes konnten dann Objekte einer Scene detektiert werden. Durch die georeferenzierten Bilddaten wurden alle automatisch gefundenen Objektinformationen in die Punktwolke übertragen. Hierbei konnte eine weitaus größere Diversität der Extraktionsergebnisse erzielt werden, als mit der Analyse der reinen Punktwolke. Die finale Lösung der automatischen Extraktion bestand in der Projektion der einzelnen Objekte vom Neuronalen Netz in die Punktwolke. Dadurch, dass jedes Objekt mehrfach in jeder Bildscene erfasst wurde, besaß jeder Laserscannerpunkt mehrere automatisch generierte Objektlable. Mit Hilfe von Clusteranalysen und Mehrheitsentscheidungen konnte die Ausgangspunktwolke in einzelne Objekte vollautomatisch zerlegt werden.

Für die Verwendung in einem GIS oder für die OKSTRA-konforme Speicherung mussten die Daten weiter aufbereitet werden. Hierzu wurden die einzelnen Objektklassen einer repräsentativen Klasse zugeordnet, sodass eine eindeutige Darstellung in einem Geoinformationssystem erfolgen konnte.

Zur Kontrolle und Validierung der Extraktionsergebnisse wurde eine für den Auswerteprozess unbekannte Teststecke mit dem Fahrzeug IRIS12 aufgenommen. Die Digitalisierung jedes Referenzobjektes erfolgte klassisch und wurde in einer Datenbank hinterlegt. Die automatischen georeferenzierten Extraktionsergebnisse wurden ebenfalls in eine Datenbank importiert. Der Vergleich der Objekte erfolgte aufgrund Lage, Ausprägung und Objekttyp. Hierbei stellte es sich heraus, dass insgesamt 66% der Objekte komplett oder teilweise extrahiert werden konnten. Davon konnten 26% der Objekte mit korrektem Objekttyp und korrekter Lage, 14% mit korrekter Lage aber falschem Objekttyp und 26% unvollständig detektiert werden. 34% der Objekte waren in dem automatisch detektierten Datensatz im Vergleich zum Referenzdatensatz nicht gefunden worden. Es stellte sich heraus, dass bei den Flächenobjekten nur 4% der Referenzobjekte nicht gefunden wurden. Die korrekt extrahierten Flächenobjekte konnten mit einer Erfassungsrate von 50%, über alle Testgebiete, erfasst werden. Bestes Extraktionsergebnis lieferte der Objekttyp Markierungslinien mit 90% Übereinstimmung mit dem Referenzdatensatz auf dem Testabschnitt der Autobahn.

Die Punktobjekte konnten in Summe nur in 47% der Fälle mit dem Referenzdatensatz verifiziert werden. Die Zuordnungsdefizite kamen hauptsächlich aus den Label-Zuordnungsfehlern, die sich zum Teil aus den Kalibrierungsdefiziten der Messsensoren und zum Teil aus der Abbildungsgeometrie schmaler und kleiner Objekte in der Punktwolke ergaben. Hier wurde viel Verbesserungspotential für künftige Forschungsarbeiten detektiert.

Die Forschungsarbeit zeigte, dass es möglich ist, unter Zuhilfenahme von georeferenzierten Bildern der gleichen Scene, vollautomatisch Objekte aus einer Punktwolke zu extrahieren. Im Ergebnis liegen georeferenzierte Objekte vor, die in einem Geoinformationssystem abgebildet werden können.

12 Literatur

ADAM, A.; IOANNIDIS, C. Automatic road sign detecion and classification based on support vector machines and HOG descriptos. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2014, 2. Jg., Nr. 5, S. 1.

BREMER, M.; WICHMANN, V.; RUTZINGER, M. Eigenvalue and graph-based object extraction from mobile laser scanning point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2013, 5. Jg., S. W2.

CIGNONI, Paolo; CORSINI, Massimiliano; RANZUGLIA, Guido. Meshlab: an open-source 3d mesh processing system. *Ercim news,* 2008, 73. Jg., Nr. 45-46, S. 6.

CYBENKO, George. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems,* 1989, 2. Jg., Nr. 4, S. 303-314.

CORDTS, M.; OMRAN, M.; RAMOS, S.; REHFELD, T.; ENZWEILER, M.; BENENSON, R.; FRANKE, U.; ROTH, S.; SCHIELE, B. The cityscapes dataset for semantic urban scene understanding. *arXiv preprint arXiv:1604.01685*, 2016.

DE LA ESCALERA, Arturo, et al. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 1997, 44. Jg., Nr. 6, S. 848-859.

Deumlich, F. und Staiger, R. (2002): Instrumentenkunde der Vermessung. *Herbert Wichmann Verlag, Heidelberg*.

El-Sheimy, N. and Schwarz, K. P., (1999): Navigating Urban Areas by VISAT – A Mobile Mapping; *System Integrating GPS/INS/Digital Cameras for GIS Application, Navigation, Vol. 45, No. 4,pp. 275-286.*

El-Sheimy, N., Schwarz K.P., and Gravel, M. (1995), “Mobile 3-D Positioning Using GPS/INS/Video Cameras “, *The Mobile Mapping Symposium, Columbus, OH,USA, pp. 236-249, May 24-26.*

EVERINGHAM, M.; ESLAMI, S. M. A.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 2015, 111. Jg., Nr. 1, S. 98-136.

FILIPE, Silvio; ALEXANDRE, Luis A. A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset. In: *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*. IEEE, 2014. S. 476-483.

GIERING, Michael; VENUGOPALAN, Vivek; REDDY, Kishore. Multi-modal sensor registration for vehicle perception via deep neural networks. In: *High Performance Extreme Computing Conference (HPEC), 2015 IEEE*. IEEE, 2015. S. 1-6.

GIRSHICK, Ross, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014. S. 580-587.

GROSS, Hermann; THOENNESSEN, Ulrich. Extraction of lines from laser point clouds. *In: Symposium of ISPRS Commission III: Photogrammetric Computer Vision PCV06. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.* 2006. S. 86-91.

GUO, Y.; BENNAMOUN, M.; SOHEL, F.; LU, M.; WAN, J. 3D object recognition in cluttered scenes with local surface features: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2014, 36. Jg., Nr. 11, S. 2270-2287.

GUPTA, Saurabh, et al. Learning rich features from RGB-D images for object detection and segmentation. In: *Computer Vision–ECCV 2014*. Springer International Publishing, 2014. S. 345-360.

HARIHARAN, B.; ARBELÁEZ, P.; GIRSHICK, R.; MALIK, J. Simultaneous detection and segmentation. In: *Computer vision–ECCV 2014*. Springer International Publishing, 2014. S. 297-312.

Harris, C G & J M Pike, (1988); 3D Positional Integration from Image Sequences, *Proceedings third Alvey Vision; Conference (AVC87), pp. 233-236,1987; reproduced in Image and Vision Computing, vol 6, no 2, pp. 87-90*,

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015. S. 1026-1034.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. CoRR abs/1512.03385 (2015).

HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask R-CNN. arXiv:1703.06870. 2017.

HIRSCHMÜLLER, Heiko. Accurate and efficient stereo processing by semi-global matching and mutual information. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).* IEEE, 2005. S. 807-814.

Huber D. (2011): The ASTM E57 File Format for 3D Imaging Data Exchange; *Proceedings of the SPIE Vol. 7864A, Electronics Imaging Science and Technology Conference (IS&T), 3D Imaging Metrology, January, 2011*.

HOUBEN, S.; STALLKAMP, J.; SALMEN, J.; SCHLIPSING, M.; IGEL, C. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013. S. 1-8.

Kern, F. (2002). Automatisierte Modellierung von Bauwerksgeometrien aus 3D-Laserscanner-Daten. *PhD thesis, Technische Universität Braunschweig. Geodätische Schriftenreihe der Technischen Universität Braunschweig Nr.19.*

Kraus, K. (2004). Photogrammetrie Band 1: *Geometrische Informationen aus Photographien und Laserscanneraufnahmen. Walter de Gruyter, Berlin - New York.*

KINGMA, Diederik; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KLETTE, SCHLUNS, KOSCHAN; Computer Vision: Three-Dimensional Data from Images; Springer-Verlag, 1998.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012. S. 1097-1105.

Luhmann, T. (2002). Photogrammetrie und Laserscanning, *Anwendung für As-Built-Dokumentation und Facility Management. Herbert Wichmann Verlag, Heidelberg*.

LIN, G.; SHEN, C.; VAN DEN HENGEL, A.; REID, I. Efficient piecewise training of deep structured models for semantic segmentation. *In CVPR,* 2016, to appear.

LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016. S. 3431-3440.

MOUTARDE, F.; BARGETON, A.; HERBIN, A.; CHANUSSOT, L. Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system. *In: Intelligent Vehicles Symposium,* 2007 IEEE. IEEE, 2007. S. 1122-1126.

PU, S.; RUTZINGER, M.; VOSSELMAN, G.; OUDE ELBERINK, S. Recognizing basic structures from mobile laser scanning data for road inventory studies. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2011, 66. Jg., Nr. 6, S. S28-S39.

QI, C.R.; SU, H.; MO, K.; GUIBAS, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593, 2016.

Riegl (2005). Internetseite. http://www.riegl.com/products/terrestrial-scanning/produktdetail/product/scanner/4/; general informations about terrestrial laserscanners (letzter Zugriff: Juli 2016).

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. *In: Medical Image Computing and Computer-Assisted Intervention–MICCAI* 2015. Springer International Publishing, 2015. S. 234-241.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A.; FEI-FEI, L. Imagenet large scale visual recognition challenge*. International Journal of Computer Vision,* 2015, 115. Jg., Nr. 3, S. 211-252.

Scheller, S.; Westfeld, P.; Ebersbach, D. (2007): Calibration of a mobile mapping camera system with photogrammetric methods. *5th International Symposium on Mobile Mapping Technology (MMT 2007), Padua, Italy*

Schwarz, K.P., Cannon, M.E., and Wong, R.V.C. (1989), “A Comparison of GPS Kinematic Models for the Determination of Position and Velocity along a Trajectory”, *Manuscripta Geodaetica, 1989, 14(2), pp. 345-353.*

SERMANET, Pierre; LECUN, Yann. Traffic sign recognition with multi-scale convolutional networks. In: *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011. S. 2809-2813.

STALLKAMP, J.; SCHLIPSING, M.; SALMEN, J.; IGEL, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 2012, 32. Jg., S. 323-332.

STEDER, B.; RUSU, R.; KONOLIGE, K.; BURGARD, W. Point feature extraction on 3D range scans taking into account object boundaries. In: *Robotics and automation (icra), 2011 ieee international conference on*. IEEE, 2011. S. 2601-2608.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

SUNG, A DDA Octree Traversal Algorithm for Ray Tracing, Eurographics'91, North Holland-Elsevier, ISBN 0444 89096 3, p. 73-85.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A.. Going Deeper with Convolutions. arXiv:1409.4842, 2014.

Ulrich, A., Stucknika, N. und Riegl, J. (2005). High-resolution laser scanner with waveforme digitisation for subsequent full waveforme analysis. From Conference Volume 5791 Gary W. Kamerman; Laser Radar Technology and Applications X; Orlando, Florida, USA | March 28, 2005

WEINMANN, M.; JUTZI, B.; HINZ, S.; MALLET, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing,* 2015, 105. Jg., S. 286-304.

WU, Zifeng; SHEN, Chunhua; VAN DER HENGEL, Anton. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. CoRR abs/1611.10080 (2016).

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. *In ICLR,* 2016, to appear.

ZEILER, Matthew D.; FERGUS, Rob. Visualizing and understanding convolutional networks. In: *Computer vision–ECCV 2014*. Springer International Publishing, 2014. S. 818-833.

ZHOU, Liang; VOSSELMAN, George. Mapping curbstones in airborne and mobile laser scanning data. *International Journal of Applied Earth Observation and Geoinformation*, 2012, 18. Jg., S. 293-304.

ZHAO, Hengshuang; SHI, Jianping; QI Xiaojuan; WANG Xiaogang, JIA, Jiaya. Pyramid Scene Parsing Network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

ZHOU, B.; ZHAO, H.; PUIG, X.; FIDLER, S.; BARRIUSO, A.; TORRALBA, A.. Scene Parsing through ADE20K Dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

13 Anhang 1

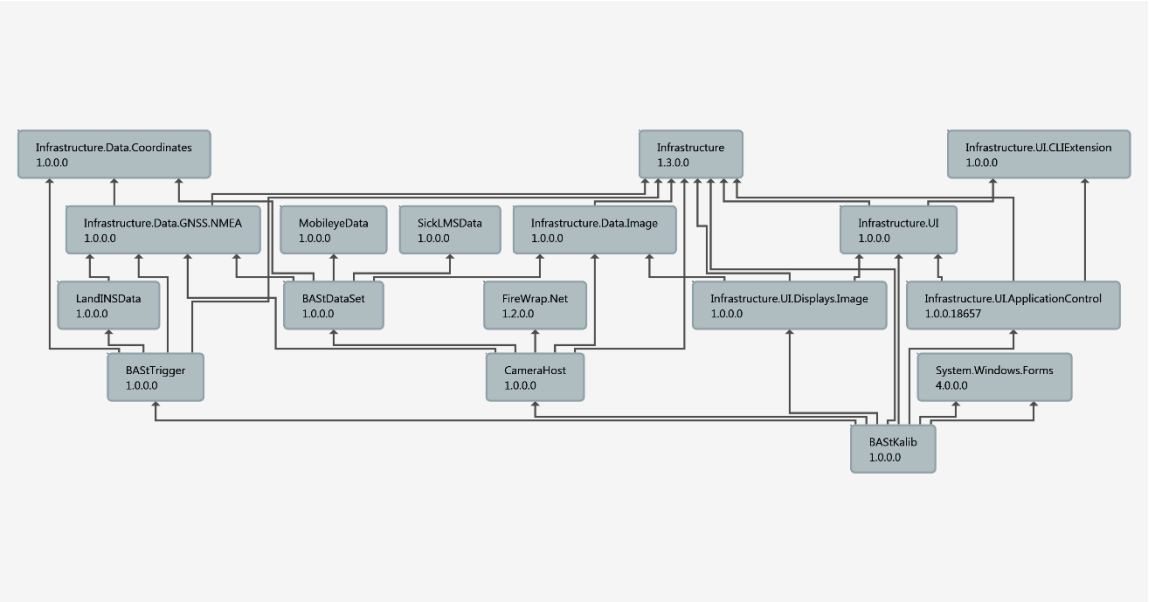


Bild 135 Abhängigkeiten der Software BastRecord von den einzelnen Softwarebibliotheken

Die Kalibrierungsparameter der Umgebungskameras konnten wie folgt ermittelt werden.

Kamera1 (links):

$$P = \begin{bmatrix} -1091.716 & 202.755 & 32.019 & 3346.911 \\ -121.396 & -263.108 & 1044.986 & -1006.007 \\ -0.304 & -0.492 & 0.011 & 1 \end{bmatrix}$$

mit den Verzeichnungsparametern:

k1 = -0.18423

k2 = 0.05384

p1 = 0.00859

p2 = 0.0023

k3 = 0

Kamera2 (Mitte links):

$$P = \begin{bmatrix} -502.493 & 486.932 & 20.353 & 1578.866 \\ -145.752 & -86.444 & 673.567 & -564.213 \\ -0.331 & -0.168 & 0.018 & 1 \end{bmatrix}$$

mit den Verzeichnungsparametern:

k1 = -0.24653

k2 = 0.30689

p1 = 0.00361

p2 = -0.00203

k3 = 0

Kamera3 (Mitte):

$$P = \begin{bmatrix} -210.587 & 589.629 & 9.610 & 628.747 \\ -154.321 & -2.829 & 598.034 & -365.763 \\ -0.332 & -0.0007 & 0.013 & 1 \end{bmatrix}$$

mit den Verzeichnungsparametern:

k1 = -0.18956

k2 = -0.04771

p1 = 0.00283

p2 = -0.00021

k3 = 0

Kamera4 (Mitte rechts):

$$P = \begin{bmatrix} 150.519 & 726.313 & -9.367 & -494.264 \\ -153.102 & 118.190 & 702.244 & -480.867 \\ -0.330 & 0.207 & 0.008 & 1 \end{bmatrix}$$

mit den Verzeichnungsparametern:

k1 = -0.24464

k2 = 0.28297

p1 = 0.00522

p2 = 0.00138

k3 = 0

Kamera5 (rechts):

$$P = \begin{bmatrix} 675.417 & 811.216 & -24.904 & -1955.307 \\ -102.796 & 262.911 & 1013.178 & -1074.934 \\ -0.294 & 0.484 & 0.018 & 1 \end{bmatrix}$$

mit den Verzeichnungsparametern:

k1 = -0.24464

k2 = 0.28297

p1 = 0.00522

p2 = 0.00138

k3 = 0

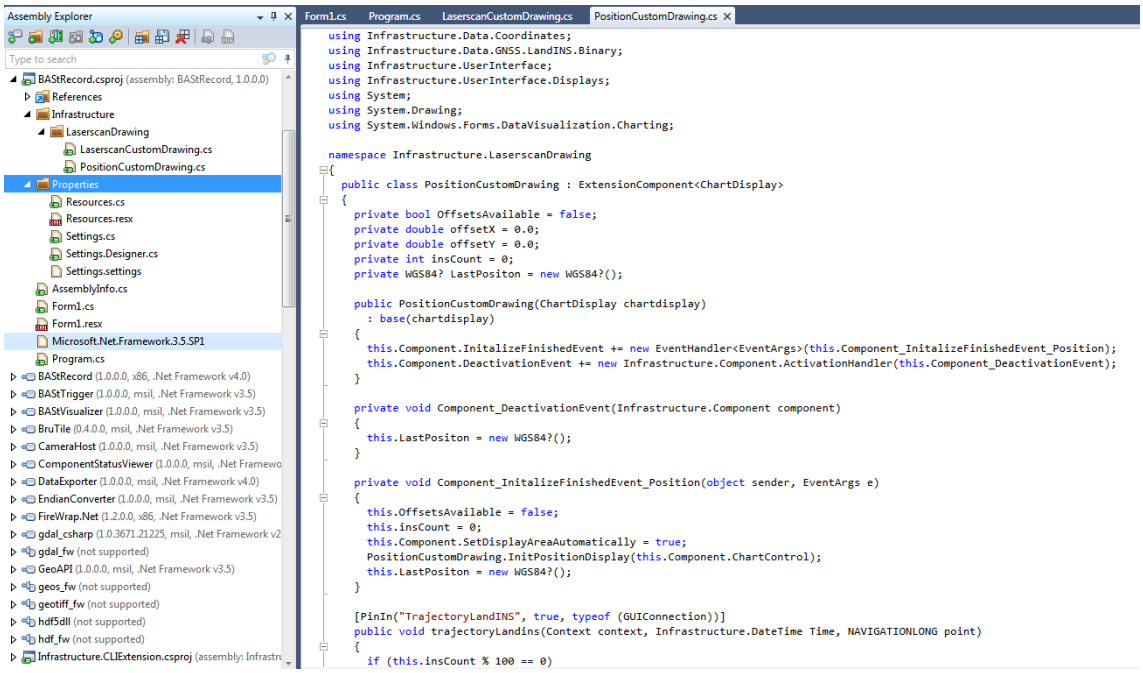


Bild 136 Dekompilierte Software BastRecord mit den beiden Klassen LaserscannCustomDrawing und PositionCustomDrawing

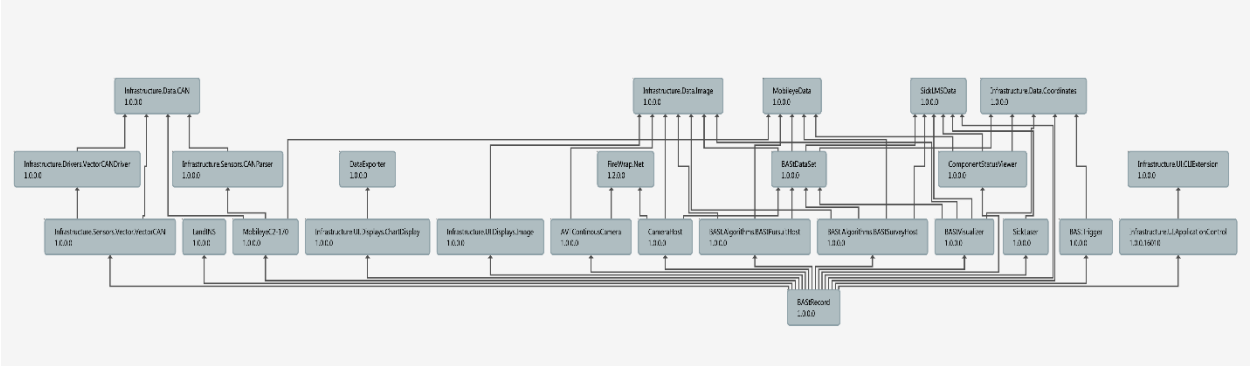


Bild 137 Abhängigkeiten der Software BastKalib von den einzelnen Softwarebibliotheken

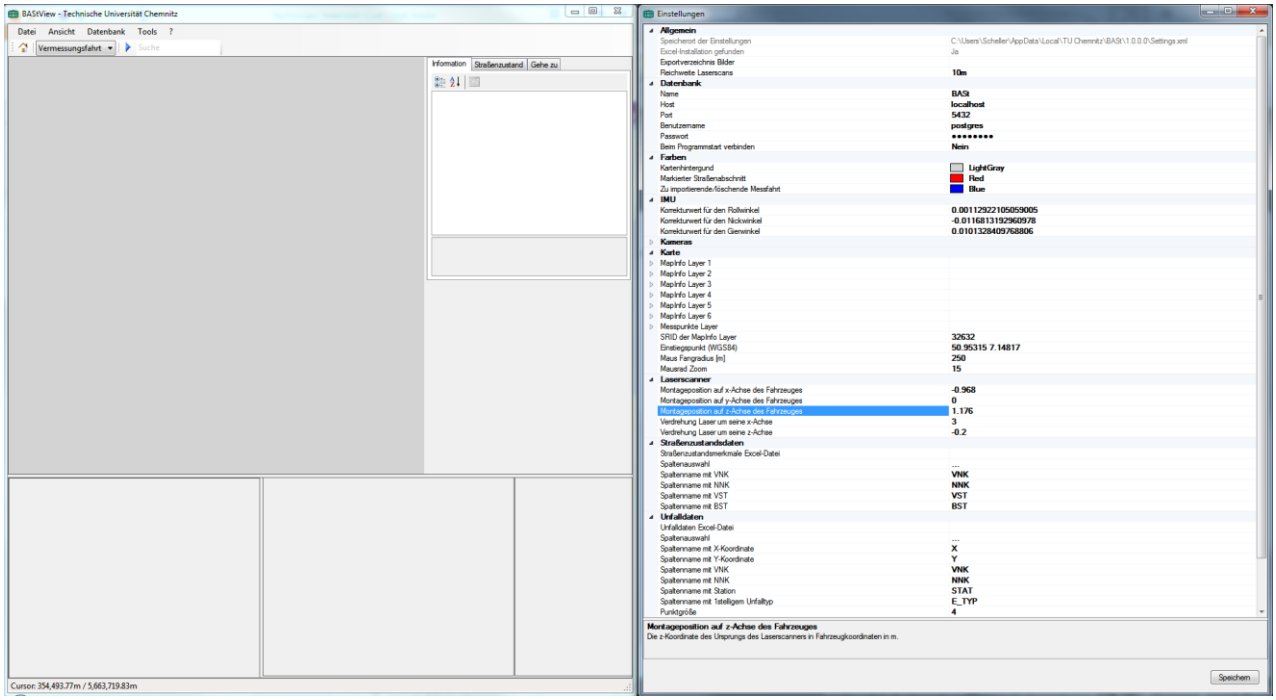


Bild 138 Software „BAStView“ mit dem Einstellungsdialog - Fenster rechts

```
00:00569E80  69 65 6C 64 20 3C 49 6D 61 67 65 46 72 6F 6E 74   ield <ImageFront
00:00569E90  52 69 67 68 74 3E 6B 5F 5F 42 61 63 6B 69 6E 67   Right>k__Backing
00:00569EA0  46 69 65 6C 64 1B 3C 49 6D 61 67 65 52 69 67 68   Field.<ImageRigh
00:00569EB0  74 3E 6B 5F 5F 42 61 63 6B 69 6E 67 46 69 65 6C   t>k__BackingFiel
00:00569EC0  64 1A 3C 4C 61 73 65 72 44 61 74 61 3E 6B 5F 5F   d.<LaserData>k__
00:00569ED0  42 61 63 6B 69 6E 67 46 69 65 6C 64 1C 3C 47 72   BackingField.<Gr
00:00569EE0  6F 75 6E 64 54 72 75 74 68 3E 6B 5F 5F 42 61 63   oundTruth>k__Bac
00:00569EF0  6B 69 6E 67 46 69 65 6C 64 1B 3C 44 65 74 65 63   kingField.<Detec
00:00569F00  74 69 6F 6E 73 3E 6B 5F 5F 42 61 63 6B 69 6E 67   tions>k__Backing
00:00569F10  46 69 65 6C 64 04 04 04 04 04 03 04 03 21 49 6E   Field........!In
00:00569F20  66 72 61 73 74 72 75 63 74 75 72 65 2E 44 61 74   frastructure.Dat
00:00569F30  61 2E 49 6D 61 67 69 6E 67 2E 49 6D 61 67 65 03   a.Imaging.Image.
00:00569F40  00 00 00 21 49 6E 66 72 61 73 74 72 75 63 74 75   ...!Infrastructu
00:00569F50  72 65 2E 44 61 74 61 2E 49 6D 61 67 69 6E 67 2E   re.Data.Imaging.
00:00569F60  49 6D 61 67 65 03 00 00 00 21 49 6E 66 72 61 73   Image....!Infras
00:00569F70  74 72 75 63 74 75 72 65 2E 44 61 74 61 2E 49 6D   tructure.Data.Im
00:00569F80  61 67 69 6E 67 2E 49 6D 61 67 65 03 00 00 00 21   aging.Image....!
00:00569F90  49 6E 66 72 61 73 74 72 75 63 74 75 72 65 2E 44   Infrastructure.D
00:00569FA0  61 74 61 2E 49 6D 61 67 69 6E 67 2E 49 6D 61 67   ata.Imaging.Imag
00:00569FB0  65 03 00 00 00 21 49 6E 66 72 61 73 74 72 75 63   e....!Infrastruc
00:00569FC0  74 75 72 65 2E 44 61 74 61 2E 49 6D 61 67 69 6E   ture.Data.Imagin
00:00569FD0  67 2E 49 6D 61 67 65 03 00 00 00 80 03 53 79 73   g.Image......Sys
00:00569FE0  74 65 6D 2E 43 6F 6C 6C 65 63 74 69 6F 6E 73 2E   tem.Collections.
00:00569FF0  47 65 6E 65 72 69 63 2E 4C 69 73 74 60 31 5B 5B   Generic.List`1[[
00:0056A000  53 79 73 74 65 6D 2E 43 6F 6C 6C 65 63 74 69 6F   System.Collectio
00:0056A010  6E 73 2E 47 65 6E 65 72 69 63 2E 4B 65 79 56 61   ns.Generic.KeyVa
00:0056A020  6C 75 65 50 61 69 72 60 32 5B 5B 49 6E 66 72 61   luePair`2[[Infra
00:0056A030  73 74 72 75 63 74 75 72 65 2E 44 61 74 61 2E 4C   structure.Data.L
00:0056A040  4D 44 73 63 61 6E 64 61 74 61 2C 20 53 69 63 6B   MDscandata, Sick
00:0056A050  4C 4D 53 44 61 74 61 2C 20 56 65 72 73 69 6F 6E   LMSData, Version
00:0056A060  3D 31 2E 30 2E 30 2C 20 43 75 6C 74 75 72 65 3D   =1.0.0.0, Cultur
00:0056A070  65 3D 6E 65 75 74 72 61 6C 2C 20 50 75 62 6C 69   e=neutral, Publi
00:0056A080  63 4B 65 79 54 6F 6B 65 6E 3D 6E 75 6C 6C 5D 2C   cKeyToken=null],
00:0056A090  5B 49 6E 66 72 61 73 74 72 75 63 74 75 72 65 2E   [Infrastructure.
00:0056A0A0  44 61 74 61 2E 47 4E 53 53 2E 4E 4D 45 41 2E 49   Data.GNSS.NMEA.I
00:0056A0B0  4E 53 50 56 41 53 2C 20 49 6E 66 72 61 73 74 72   NSPVAS, Infrastr
00:0056A0C0  75 63 74 75 72 65 2E 44 61 74 61 2E 47 4E 53 53   ucture.Data.GNSS
00:0056A0D0  2E 4E 4D 45 41 2C 20 56 65 72 73 69 6F 6E 3D 31   .NMEA, Version=1
00:0056A0E0  2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75 72 65 3D   .0.0.0, Culture=
00:0056A0F0  6E 65 75 74 72 61 6C 2C 20 50 75 62 6C 69 63 4B   neutral, PublicK
00:0056A100  65 79 54 6F 6B 65 6E 3D 6E 75 6C 6C 5D 5D 2C 20   eyToken=null]],
00:0056A110  6D 73 63 6F 72 6C 69 62 2C 20 56 65 72 73 69 6F   mscorlib, Versio
00:0056A120  6E 3D 34 2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75   n=4.0.0.0, Cultu
00:0056A130  72 65 3D 6E 65 75 74 72 61 6C 2C 20 50 75 62 6C   re=neutral, Publ
00:0056A140  69 63 4B 65 79 54 6F 6B 65 6E 3D 62 37 37 61 35   icKeyToken=b77a5
00:0056A150  63 35 36 31 39 33 34 65 30 38 39 5D 5D 25 49 6E   c561934e089]]%In
00:0056A160  66 72 61 73 74 72 75 63 74 75 72 65 2E 44 61 74   frastructure.Dat
00:0056A170  61 2E 47 4E 53 53 2E 4E 4D 45 41 2E 49 4E 53 50   a.GNSS.NMEA.INSP
00:0056A180  56 41 53 04 00 00 00 85 03 53 79 73 74 65 6D 2E   VAS......System.
00:0056A190  43 6F 6C 6C 65 63 74 69 6F 6E 73 2E 47 65 6E 65   Collections.Gene
00:0056A1A0  72 69 63 2E 4C 69 73 74 60 31 5B 5B 53 79 73 74   ric.List`1[[Syst
00:0056A1B0  65 6D 2E 43 6F 6C 6C 65 63 74 69 6F 6E 73 2E 47   em.Collections.G
00:0056A1C0  65 6E 65 72 69 63 2E 4B 65 79 56 61 6C 75 65 50   eneric.KeyValueP
00:0056A1D0  61 69 72 60 32 5B 5B 49 6E 66 72 61 73 74 72 75   air`2[[Infrastru
00:0056A1E0  63 74 75 72 65 2E 44 61 74 61 2E 4D 6F 62 69 6C   cture.Data.Mobil
00:0056A1F0  65 79 65 2E 43 32 5F 31 37 30 2C 20 4D 6F 62 69   eye.C2_170, Mobi
00:0056A200  6C 65 79 65 44 61 74 61 2C 20 56 65 72 73 69 6F   leyeData, Versio
00:0056A210  6E 3D 31 2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75   n=1.0.0.0, Cultu
00:0056A220  72 65 3D 6E 65 75 74 72 61 6C 2C 20 50 75 62 6C   re=neutral, Publ
00:0056A230  69 63 4B 65 79 54 6F 6B 65 6E 3D 6E 75 6C 6C 5D   icKeyToken=null]
00:0056A240  2C 5B 49 6E 66 72 61 73 74 72 75 63 74 75 72 65   ,[Infrastructure

astructure.Data.GNSS.NMEA.INSPVAS,
Infrastructure.Data.GNSS.NMEA, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=null]], mscorlib,
Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]
```

Bild 139 *.bst Datei im Hex-Editor mit JSON- ähnlichen -Headerinformationen