Fachhochschule Nordwestschweiz
Hochschule für Architektur, Bau und Geomatik

SmartGeoAR
Projektbericht

Fachhochschule Nordwestschweiz
Prof. Martin Christen
Hofackerstrasse 30
4132 Muttenz

Version 2.1
06.09.2022

Inhaltsverzeichnis

1 Ausgangslage

1.1 Ziele

Im Rahmen des Forschungsprojektes «SmartGeoAR» sollen die bestehenden Erkenntnisse im Bereich Augmented Reality im Strassenraum um folgende Ziele weiterentwickelt werden:

Ziel 1. Entwicklung einer **vollautomatischen** Echtzeit-Georeferenzierungs API unter Verwendung der Daten der Amtlichen Vermessung und anderen Datenquellen, z.B. swissSURFACE^{3D}. Die Daten sollen alle von swisstopo bezogen werden.
Die Bildsequenz des Tablets/Smartphones direkt in Echtzeit vollautomatisch mit den Daten verglichen und angepasst, was eine Georeferenzierung ermöglicht.

Die Bedienung einer mit dieser API entwickelten App soll auch durch Laien möglich sein und erfordert keine Kenntnisse über Geodaten/Georeferenzierung.

Ziel 2. Es soll keine Augmentierung auf bewegenden Objekten oder temporären Objekten wie z.B. parkierten Fahrzeugen, Personen usw. angezeigt werden.

Ziel 3. Eine Unterstützung für 3D-Anwendungen soll realisiert werden. Dies dient für diverse Anwendungen in der Raumplanung. Das Einblenden projektierter Gebäude, Strassen, etc. soll über einen 3D-Streamingdienst ermöglicht werden. Für das Streaming sollen 3D-Tiles (Cesium) verwendet werden. Zukünftig sollen auch Verkehrssimulationen als Augmented Reality eingeblendet werden können. Zur Kontrolle sollen Gebäude aus dem swissBUILDINGS^{3D} Daten von swisstopo visualisiert werden.

Ziel 4. Unterstützung gängiger OGC Schnittstellen: WMS, WMTS, (TMS), …
Es sollen verschiedener Geo-Portale einbindbar sein, wie z.B. map.geo.admin.ch, map.geo.bs.ch etc. Es soll die REST API von swisstopo (GeoAdmin) verwendet werden, z.B. für die Adresssuche und Darstellung der Strassenachsen.

Ziel 5. Mit der API wird eine Referenz-App entwickelt, welche auf gängigen Tablets / Smartphones (Android & iOS) funktioniert.

Ziel 6. Das Projekt soll komplett Open Source sein und zukünftig auch in organisierten Code-Sprints weiterenwickelt werden. Dazu gehört das Erstellen und Pflegen einer Open Source Community.

Ziel 7. Das Projekt soll an nationalen und internationalen Tagungen präsentiert werden.

1.2 Workpackages

Das Projekt wurde in die folgenden Workpackages unterteilt:

WP1: Vollautomatische Echtzeit-Georeferenzierung
Automatisches Matching soll mittels aktuellen Deep Learning Ansätzen im Bereich **Semantischer Segmentierung** von Strassenbildern realisiert werden. Vollautomatisches Matching mit AV-Daten und auch anderen Datenquellen wie swissSURFACE^{3D} wird somit möglich. Datensatz wird entsprechend gelabled und ein neuronales Netzwerk wird mit unserem GPU Cluster erstellt. (Probleme gibt es jedoch mit den temporären Objekten wie parkierte Autos oder Personen, dies wird im WP2 implementiert)

WP2: Detektion temporärer Objekte
Um das vollautomatische Matching zu verbessern, sollen temporäre Objekte (insbesondere parkierte und fahrende Fahrzeuge) mittels Instance Segmentation ausgeblendet werden. Dafür wird ein neues neuronales Netzwerk trainiert. Auch bewegte Objekte wie Autos, Velos etc. können so ignoriert werden. Als Grundlage dazu soll u.a. das Paper von Vertens J. et al. (2017) dienen. Die verbleibenden Daten (z.B. Trottoir, Strasse, …) werden dann direkt mit den AV-Daten über das neuronale Netzwerk von WP1 abgestimmt.

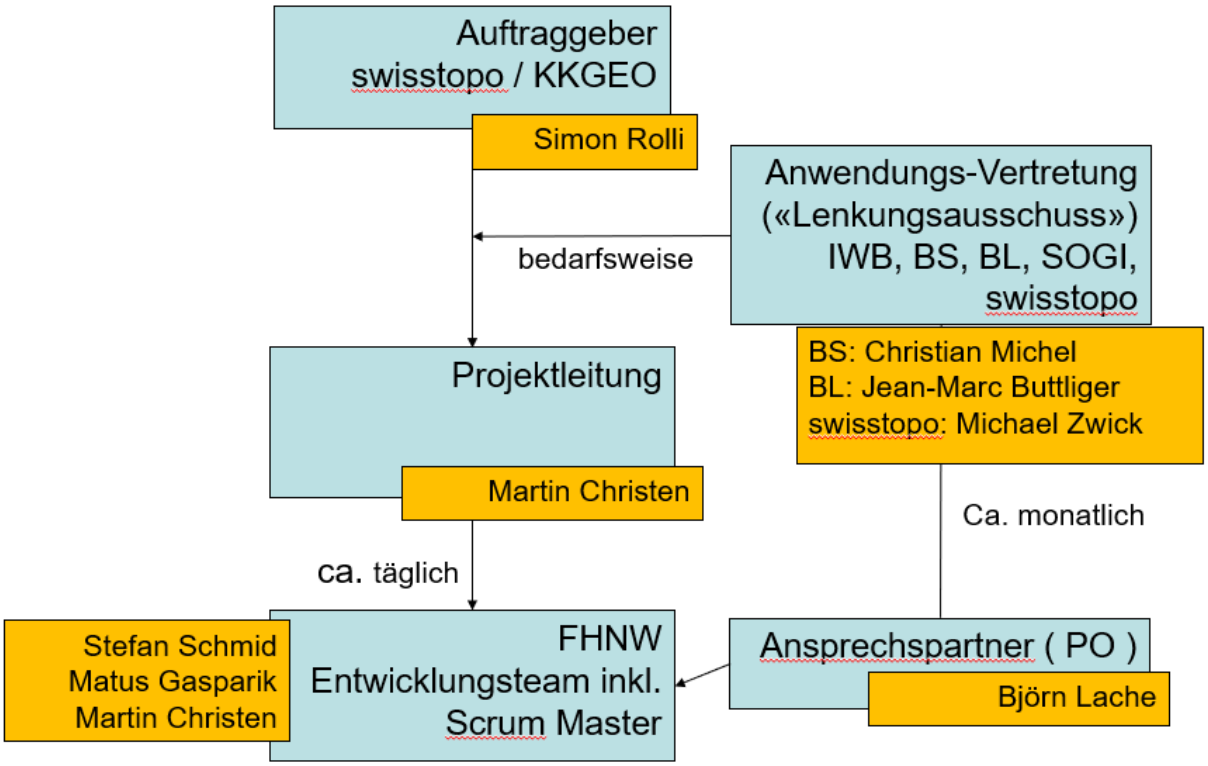
WP3: Unterstützung von 3D Objekten
In diesem Workpackage wird ein Streaming-Service für 3D Objekte implementiert, welche per Augmented Reality verwendet werden können. Als Test-Datensatz kann z.B. swissBUILDINGS3D verwendet werden. Für das Streaming sollen 3D-Tiles (Cesium) verwendet werden können. Auch weitere OGC Streaming-Ansätze (3DPS) werden ermöglicht.

WP4: Schnittstellen und Kartendienste
Zusätzlich zu WFS sollen auch andere Dienste unterstützt werden. Primär sollen Bilddienste (primär WMS, WMTS, TMS) unterstützt werden und auch Tiles von bestehenden Kartendiensten (map.geo.admin.ch, map.geo.bs.ch, …) augmentiert werden.

WP5: Unterstützung von iOS und Android
Es wird eine Referenzapplikation entwickelt, welche die API aus WP1-WP4 unterstützt. Um eine breite Unterstützung dieser Technologie zu erreichen, wird neben iOS auch eine Android-Version implementiert. Dazu wird ARKit und ARCore verwendet. Der Source Code Deep Learning muss nicht speziell portiert werden, da dieser 99% kompatibel für beide Betriebssysteme ist.

WP6: Aufbau Open Source Community **WP7:** Präsentation
Das Projekt wird an 2-3 Tagungen vorgestellt (u.a. FOSS4G) und mindestens ein Journal Paper wird geschrieben. An Community Tagungen (FOSS4G, FOSSGIS) wird ein Code-Sprint organisiert. Bei allen Präsentationen wird auf die Finanzierung NGDI (Bund/Kantone) hingewiesen.

1.3 Rollen im Projekt



2 Stand der Technik und Begriffe

In diesem Abschnitt wird auf den Stand der Technik eingegangen und auch einige Begriffe erklärt. welche später in diesem Dokument relevant sein werden.

2.1 Augmented Reality

Augmented Reality (AR) erweitert die reale Welt mit virtuellen Objekten in Echtzeit. Virtuelle 3D-Objekte werden geometrisch registriert und in der realen Welt dargestellt. Sofern z.B. dem Objekt keine Animation hinterlegt ist, bleiben diese Objekte fix an dem zugewiesenen Ort, unabhängig des Betrachtungswinkels. Dies führt dazu, dass die anwendende Person die Objekte genau gleich wahrnimmt wie ein reales Objekt.
Nebst AR wird häufig auch der Begriff Mixed Reality (MR) für dieses Verfahren verwendet. Dies bedeutet nichts anderes, als dass die virtuelle- und reale Welt miteinander vermischt weden.
Nach der Definition von Azuma (1997) muss ein AR System folgende drei Charaktereigenschaften mit sich bringen:

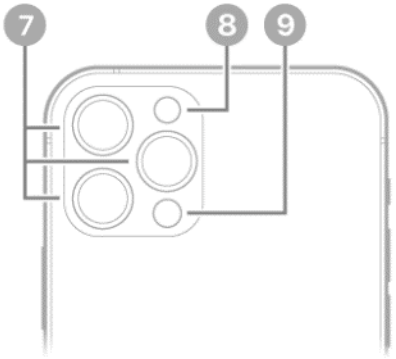
- Es kombiniert Realität und Virtualität
- Es ist interaktiv in Echtzeit
- Die virtuellen Inhalte sind im 3D registriert

Während die Interaktivität in Echtzeit ebenfalls in VR zum Tragen kommt, bilden die zwei weiteren Punkte den Unterschied zwischen AR und VR. Bei der AR Technologie stellt immer die Umgebung die Referenz dar. So sollten für eine möglichst echte Wahrnehmung der 3D-Objekte, diese immer im Massstab 1:1 vorliegen. So können virtuelle und reale Objekte möglichst genau ineinander verschmolzen oder ergänzt werden.
Im Unterschied zur VR Technologie gibt es keine räumlichen Beschränkungen und es ist durchaus möglich AR im Freien anzuwenden.
Eine grosse Herausforderung, um eine möglichst genaue Einbettung der 3D Objekte in die reale Umgebung zu erhalten, ist das Referenzieren des Betrachtungspunktes. Diese Referenzierung steht im Bezug zu einem Raum oder im entsprechenden Koordinatensystem. In diesem Bereich läuft auch dieses Projekt.

2.1.1 Augmented Realiity Hardware

Es gibt inzwischen zahlreiche Augmented Reality Brillen, wie beispielsweise die Microsoft Hololens oder die MagicLeap. Für dieses Projekt wurden nur Smartphones und Tablets verwendet.

Mitten im Projekt, im Oktober 2020 stellte Apple ein neues iPhone vor. Nebst 5G-Technologie, neustem A14 Bionic-Chip und Super Retina XDR Display verfügt das iPhone 12 Pro über ein komplexes Kamera-System. Dieses setzt sich aus einer Ultra Weitwinkel-, einer Tele- und einer Weitwinkelkamera zusammen. Unterstützt werden die Kameras durch den seit neustem verbauten LiDAR-Scanner, der insbesondere bei schwierigeren Lichtverhältnissen hilft, schärfere Fotos zu macht. Dieser Scanner verbessert aber auch Augmented-Reality (AR) und erlaubt schnelles Laden dank MagSafe. (Apple Inc. 2020) Die Anordnung der Kameras, des Blitzes und des LiDAR-Scanners sind in der folgenden Abbildung zu sehen.



7 Kameras
8 Blitz
9 LiDAR-Scanner

Abb. 1: iPhone 12/13 Pro Sensoren

Auf der Hinterseite des iPhone 12 Pro sind oben links drei Kameras, der Blitz und der LiDAR-Scanner verbaut. Dieses System steht der Glasplatte leicht vor. (Apple Inc. 2021a)
Die Art des LiDAR-Scanners ist ein Betriebsgeheimnis, Murtiyoso et al. (2021) gehen davon aus, dass es sich bei dem LiDAR-Scanner um einen des Typs Solid-State LiDAR handelt. Es ist möglich, dass der verbaute LiDAR auf einer Single-Photon-Avalanche-Diode (SPAD) basiert, welche mit einer Laserquelle gekoppelt ist. (Murtiyoso et al. 2021)
Für Augmented Reality sind nebst Kamera und LiDAR-Scanner noch andere Bestandteile von Bedeutung. Insbesondere Sensoren zur Ermittlung der Position im Raum und deren Veränderung spielen eine grosse Rolle. Dafür ist ein Drei-Achs Gyrometer und Beschleunigungsmesser und ein Näherungssensor verbaut. (Apple Inc.)

2.1.2 Augmented Reality APIs

Es gibt inzwischen einige Augmented Reality APIs für Smartphones. Die bedeutendsten sind ARKit von Apple und ARCore von Google.

ARKit ist Apples API für Augmented Reality Anwendungen unter iOS. ARKit wurde von Apple auf der WWDC 2017 angekündigt und wurde später im Jahr als Bestandteil von iOS 11 für neuere iPhones und iPads veröffentlicht. Zu den Features gehören die Kernbausteine “Finden von horizontalen Flächen wie Boden oder Tischplatten“, “Platzieren von virtuellen Objekten in Realgröße” und das “Annähern der Lichtsituation / Beleuchtung virtueller Objekte an die reale Umgebung“.

ARKit ist inzwischen in der Version 5 verfügbar und verfügt unter anderem über sogenannte Location Anchors, welche eine Georeferenzierung vornehmen kann unter Abgleichung der aktuellen Umgebung. Dieses Feature ist jedoch (Stand Februar 2022) nach wie vor nur in den USA verfügbar.

ARCore ist Googles neuer Technologie-Baustein für Augmented Reality Anwendungen und die Antwort auf Apples ARKit. ARCore wurde von Google am 29. August 2017 angekündigt und stand kurz danach für ausgewählte Smartphones (u.a. das Google Pixel und Samsung Galaxy S8) zur Verfügung. Als Augmented Reality Technologiebaustein für das Betriebssystem Android zieht ARCore mit den Funktionen von ARKit gleich und umfasst somit die Kernbausteine “Finden von horizontalen Flächen wie Boden oder Tischplatten”, “Erkennen und Platzieren von Objekten in Realgröße” und das “Annähern der Lichtsituation / Beleuchtung virtueller Objekte an die reale Umgebung”. Dabei nutzt ARCore Visual Inertial Odometry (VIO), welche eingehende Daten der Kamera mit Daten des Gyroskops (Position und Rotation) sowie des Beschleunigungssensors verbindet und somit die eigene Bewegung und Ausrichtung des Smartdevice im Raum sehr genau errechnen kann. Gleichzeitig können dadurch Grössenrelationen zwischen Realität und virtuellen Ergänzungen sauber abgeglichen werden.

2.2 Deep Learning

Schon seit mehr als 100 Jahren träumen Erfinder von einer Maschine, die selbstständig denken kann. In der Anfangszeit der künstlichen Intelligenz (engl. **Artifical Intelligence**) versuchte man, Probleme zu lösen die für den Menschen schwierig, für den Computer jedoch unkompliziert sind. Dies gelang vor allem bei Problemen, die sich gut mit mathematischen Regeln beschreiben liessen. Die Herausforderung besteht heute darin, Probleme mit dem Computer zu lösen, die vom Menschen intuitiv und nahezu automatisch erledigt werden. Daran wird immer noch intensiv geforscht. (Goodfellow u. a., 2018)

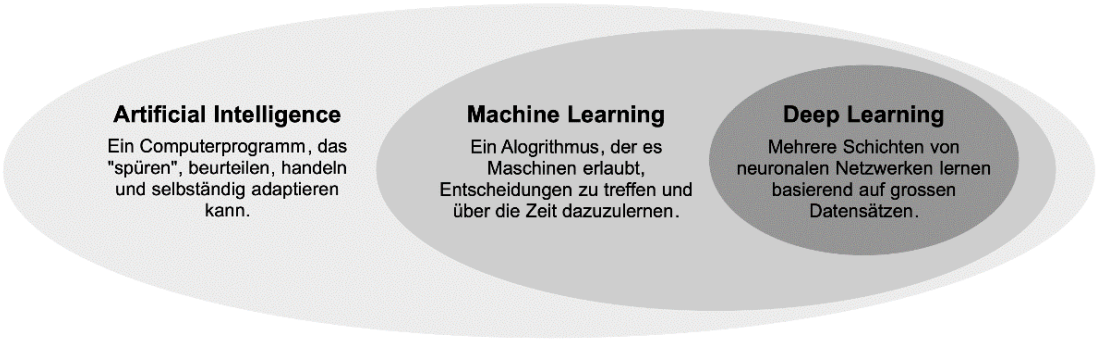


Abb. 2: Deep Learning ist ein Unterbereich von Machine Learning, welches wiederum ein Teilgebiet von künstlicher Intelligenz (engl. Artifical Intelligence) ist.

Maschinelles Lernen ist ein zentraler Bestandteil der künstlichen Intelligenz (Abb. 2). Mit mathematischen Methoden erkennt der Computer in den Daten verschiedene Muster, welche es ihm ermöglichen eigene Regeln zu entwickeln. Die aus den Daten gewonnen Erkenntnisse lassen sich danach auf unbekannte Daten anwenden und können verschiedene Aufgaben automatisieren. Ein solches System wird nicht explizit programmiert, sondern anhand von vielen Trainingsdaten trainiert. (Chollet, 2017)

Deep Learning ist ein Unterbereich des Machine Learning. Die Lernmethode ist der Funktionsweise des menschlichen Gehirns nachempfunden und nutzt künstliche neuronale Netze, um grosse Datenmengen zu analysieren. Auf Basis vorhandener Informationen wird die Fähigkeit antrainiert, selbstständig eigene Prognosen oder Entscheidungen zu treffen. Deep Learning eignet sich besonders für Anwendungen, wo grosse Datenbestände verarbeitet werden, aus welchen sich wiederum Muster und Modelle ableiten lassen. Darum wird Deep Learning häufig für die Gesichts-, Objekt- oder Spracherkennung eingesetzt. Mit Hilfe von Deep Learning wird der Maschine das Lernen beigebracht, sodass diese in der Lage ist, ohne menschliches Eingreifen ihre Fähigkeiten zu verbessern. (Luber, 2017)

Bei einem **Convolutional Neural Network** CNN handelt es sich um eine Sonderform eines neuronalen Netzes, die speziell für die Verarbeitung von Bild- oder Audiodaten entwickelt wurde. Ein CNN besteht aus mehreren Schichten wie Input-, Hidden- und Classification Layer.

Ein Convolutional-Layer verarbeitet den Input und extrahiert einzelne Merkmale wie Linien, Kanten oder bestimmte Formen mithilfe von Filtern. Der Pooling-Layer verwirft die überflüssigen Informationen und reduziert die Auflösung. Dadurch wird die Berechnungsgeschwindigkeit erhöht. Diese Anreihungen werden als Hidden-Layers zusammengefasst. Am Schluss werden im Fully Connected-Layer alle Merkmale der vorherigen Ebenen zusammengesetzt, um die Ergebnisse den Labelklassen zuzuordnen. (Luber & Litzel, 2019)

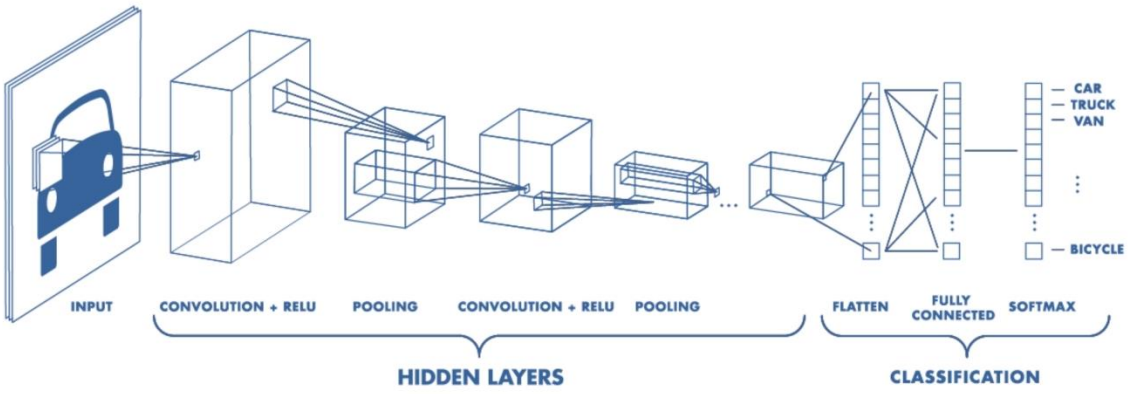


Abb. 3: Typische Architektur eines Convolutional Neural Network (CNN) für die Verarbeitung von Bild- und Audiodaten (Patel & Pingel, 2017)

Um Objekte in Bildern mit einem CNN zu identifizieren, wird ein massiver Datensatz benötigt. Üblicherweise steht jedoch nur eine begrenzte Anzahl von Trainingsdaten zur Verfügung. Für kleinere Datensätze kann die Methode des Transfer Learning angewendet werden, bei der kein komplett neues CNN erstellt und trainiert werden muss, sondern ein vortrainiertes Modell verfeinert werden kann. Dazu werden die Gewichte der Convolutional Layer des CNN eingefroren. Anschliessend wird nur der letzte Layer, der die Vorhersage trifft, trainiert. So bleibt das erlernte Wissen über allgemeine Merkmale wie Kanten, Muster und Verläufe in den Convolutional Layern bestehen. Auf diese Weise können Netzwerke für die eigene Problemlösung angepasst werden. Das Transferlernen funktioniert auch für Netzwerke, bei denen die Kategorien unterschiedlich sind, jedoch bereits mit einem sehr grossen Datensatz trainiert wurden. (Koehrsen, 2018)

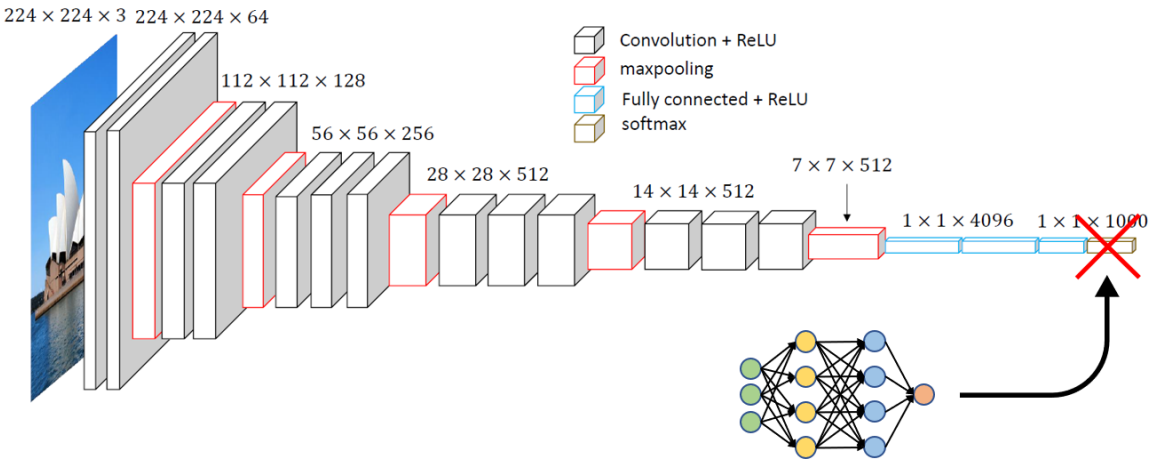


Abb. 4: Beim Transferlernen wird der letzte Layer ersetzt und mit den neuen Daten trainiert, hier am Beispiel einer VGG16 Architektur (Wang u. a., 2017)..

Für maschinelles Lernen werden die Daten in einer speziellen Datenablage numerisch aufbereitet. Dieser Container ist auch als Tensor bekannt und kann Daten in N-Dimensionen aufnehmen. Ein Tensor ist ein mathematisches Objekt aus der linearen Algebra, nicht zu verwechseln mit Vektoren oder Matrizen. (Jean, 2018)

Abb. 5: Darstellung von einem Skalar, Vektor, Matrix und einem Tensor

2.3 Qualitätsmetriken

Die erreichbare Genauigkeit einer Deep Learning Anwendung ist vom konkreten Problem, der Grösse des Datensatzes und der Wahl des verwendeten Modells abhängig.

2.3.1 Intersection over Union (IoU)

In der Bildsegmentierung und in der Computer Vision kommt der Jaccard-Koeffizient oder Intersection over Union (IoU) oft zur Anwendung. Der Wert beschreibt das Verhältnis zwischen dem Überlappungsbereich und dem Bereich der Vereinigung (Abbildung links). Dabei werden die Bounding Boxen aus dem Testdatensatz (Grundwahrheit) mit denen aus der Vorhersage verglichen (Abbildung rechts). Ein IoU-Wert nahe 1 bedeutet, dass sich Boxen mit den Grundwahrheiten sehr gut überschneiden, während bei einem Wert von 0 keine Überlappung vorhanden ist.

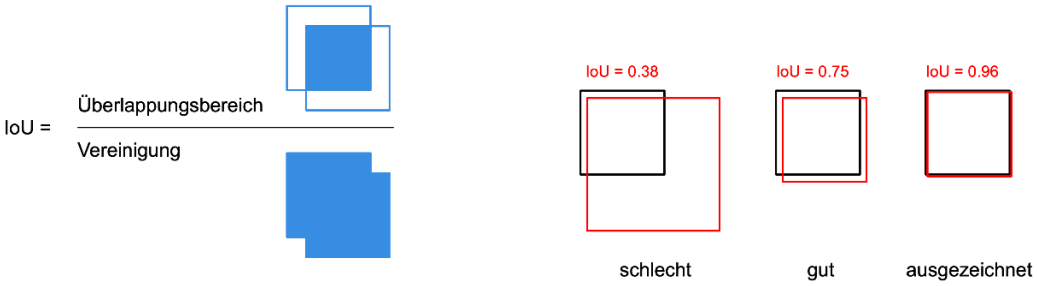


Abb. 6: links: Berechnung der Intersection over Union, rechts: die Bedeutung der IoU-Werte wobei die schwarzen Rechteecke die Grundwahrheit und die roten die Vorhersagen darstellen

2.3.2 Wahrheitsmatrix

Die Grundlage, um ein Klassifikationsproblem zu bewerten, bildet die Wahrheitsmatrix. Dabei werden die Häufigkeiten der richtigen und falschen Klassifikationen ermittelt.

		Realität	
		Ja	**Nein**
Entscheidung des Klassifikators	**Ja**	**Richtig positiv (true positive, TP**): Die Prognose hat ein Objekt richtig erkannt IoU > 0.5 Zahl der richtig positiven Klassifikationen	**Falsch positiv (false positive, FP):** Die Prognose hat ein Objekt erkannt jedoch von einer anderen Klasse IoU < 0.5 Zahl der falsch positiven Klassifikationen
	Nein	**Falsch negativ (false negativ, FN):** Die Prognose hat kein Objekt erkannt, obwohl es vorhanden ist IoU > 0.5 Zahl der falsch negativen Klassifikationen	**Richtig negativ (true negative, TN):** Die Prognose hat erkannt, dass kein Objekt vorhanden ist. Zahl der richtig negativen Klassifikationen

Abb. 7: Die Wahrheitstabelle zeigt alle möglichen Wahrheitswerte einer Klassifizierung

2.3.3 Recall

Recall beschreibt das Verhältnis zwischen TP (True Positive) und der Gesamtzahl aller Objekte im Datensatz einer Klasse. Recall kann auch als True Positive Rate (TPR) bezeichnet werden. Bei einem Recall-Wert nahe 1 werden praktisch alle Objekte im Datensatz erkannt.

$$Recall\ (TPR) = \frac{TP}{TP + FN}$$

2.3.4 Precision

Die Relevanz einer bestimmten Klasse in einer Klassifizierung wird Precision genannt. Diese wird aus dem Verhältnis von TP und der Gesamtzahl der vorhergesagten Positiven berechnet.

$$Precision\ = \frac{TP}{TP + FP}$$

2.3.5 Loss

Um die Prognose nach einem Training zu beurteilen, wird mit einer Loss-Funktion der Loss-Wert berechnet. Dieser gibt die Fehlerrate an, die zeigt wie gut oder schlecht eine Vorhersage eines Modells ist und beschreibt den Abstand zwischen dem erreichten und tatsächlichen Output. Die gängigste Loss-Funktion ist die mittlere quadratische Abweichung.

2.3.6 Epochen

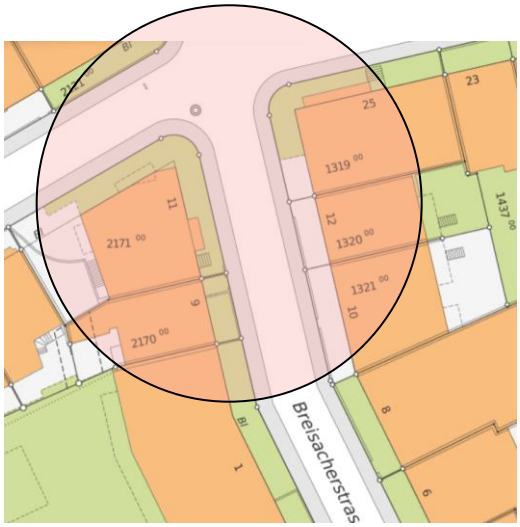
Beim Training neuronaler Netze verarbeitet das System Eingangssignale (Bilder mit Strassenobjekten) und wandelt diese in interne Repräsentationen (Zahlenvektoren) um, mit denen Berechnungen durchgeführt werden können. Über mehrere Durchläufe (Epochen) der Daten durch das Netz lernt dieses dann, wie die Gewichtungen im Netz verteilt sein müssen, sodass am Ende die gewünschten Ergebnisse herauskommen. Was diese gewünschten Ergebnisse sind, weiss das Netz anhand des Trainingssets. Dieses bildet die Wissensgrundlage des deep learnings und beinhaltet eine möglichst grosse Menge von Datensätzen, zu denen die korrekten Resultate (Trottoirs, Strassen, Ampeln, ...) bekannt sind.

3 Realisierung

3.1 Lösungsansatz Gesamtübersicht

Mit der Smartphone Kamera soll ein Bild aufgenommen werden:

Von diesem Bild ist die ungefähre Position über GPS bekannt. Dies ist im Bereich von rund 30 Metern. Diese Position soll mit einem Umkreis mit den AV-Daten abgeglichen werden:



Um dies mit den AV-Daten zu vergleichen, müssen folgende Daten vorhanden sein:

- Erkennung was ist Trottoir, was ist Strasse mittels **Deep-Learning**
- Berechnung der Ebenen mittels **Augmented Reality APIs**
- Erstellen einer Tiefenkarte
- Berechnen der Kamera Position
- Matching mit den AV-Daten
- Georeferenzierung der Kamera-Position und somit der Szene

Dieser Prozess soll als Cloud-Dienst realisiert werden, da die AV-Daten nicht komplett auf dem Smartphone gespeichert werden und auch um Rechenpower auszulagern.

3.2 Anpassung der Work-Packages für Deep Learning

Bei den Untersuchungen bestehender neuronaler Netzwerke musste festgestellt werden, dass die Erkennung von Objekten aus der Fussgängerperspektive nicht funktioniert, sondern nur aus der Fahrzeugperspektive (siehe Abbildung). In den Bildern sieht man, dass das Trottoir nicht zuverlässig erkannt werden kann.

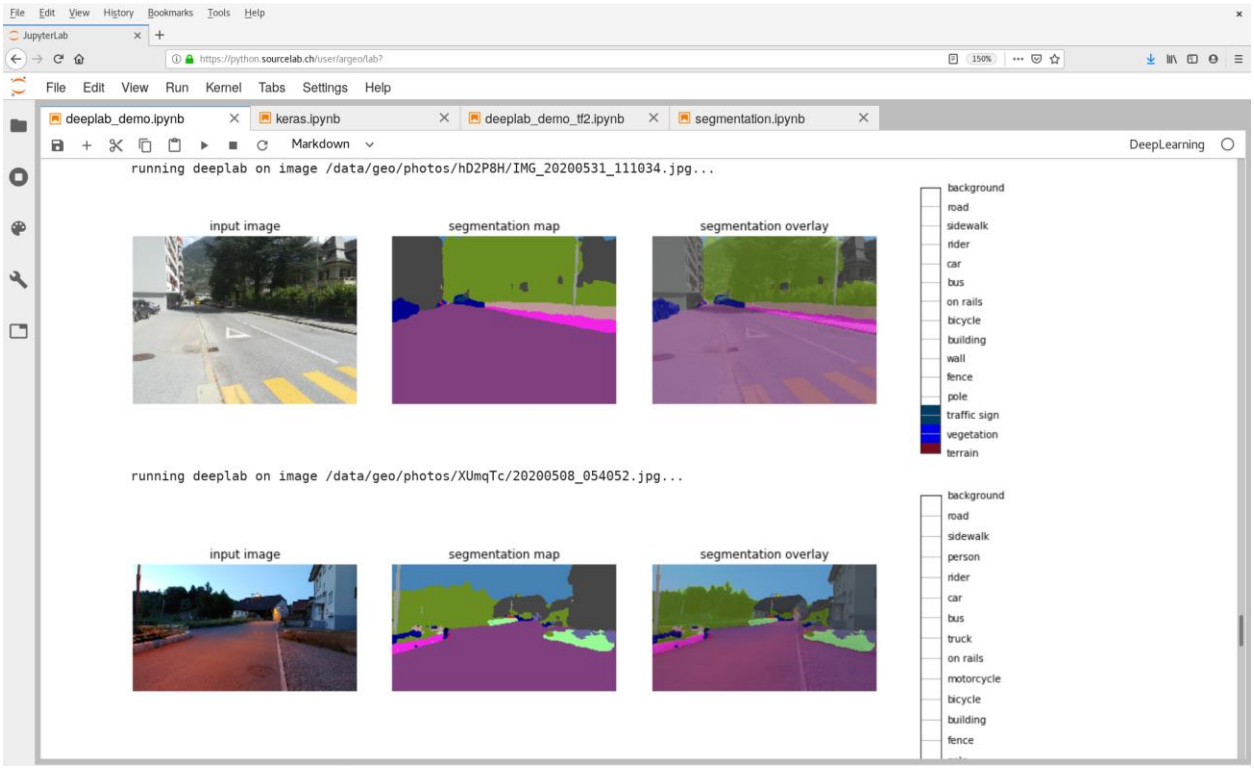


Abb. 8: Detektion von Strassenobjekten mit bestehenden neuronalen Netzen funktioniert nicht gut

Dazu wurden die Workpackages 1 und 2 angepasst:

WP1_neu: Aufbau eines neuen neuronalen Netzwerks
• Entwicklung eines Tools zum Upload von Fotos aus der Fussgängerperspektive • Erfassen von Fotos an möglichst vielen Orten • Definition von Labeln • Labelling mittels segments.ai • Entwicklung eines neuen neuronalen Netzwerks

WP2_neu: Detektion von Objekten im Strassenraum
Um das vollautomatische Matching zu realisieren, sollen Objekte im Strassenraum aus Fussgängerperspektive über einen Webservice automatisch detektiert werden. Dabei werden über 20 verschiedene Kategorien definiert (Trottoir, Vegetation, Fussgängerstreifen, ...). Auch Fahrzeuge und Personen sollen automatisch detektiert werden, um diese ignorieren zu können.

3.3 Aufnahme von Fotos im Strassenraum (Foto-Upload-Tool)

Um Fotos hochzuladen, wurde ein Webservice entwickelt, welcher es ermöglicht Fotos hochzuladen. Die Bilder werden Serverseitig in einer Datenbank gespeichert und EXIF-Informationen (z.B. Orientierung des Bildes, GPS falls vorhanden, ...) werden mitgespeichert.

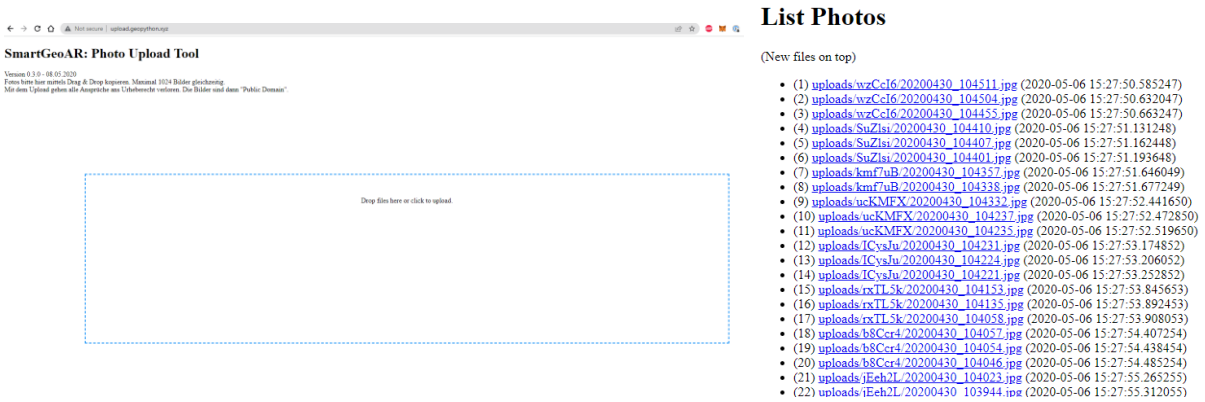


Abb. 9: Foto-Upload-Tool: Einfacher Service zum hochladen von Strassenbildern.

Vom gesamten Projektteam wurden über 2000 Fotos im Strassenraum mit unterschiedlichen Smartphones erfasst.

3.4 Labeling des Fotos

Die Labels für das trainieren des neuronalen Netzwerks wurden an die AV-Daten angelehnt, enthalten jedoch auch weitere wichtige Objekte im Strassenraum. Es wurden total 30 verschiedene Klassen erfasst, welche in der folgenden Tabelle zusammengefasst sind.

static	Alles unbewegliche, welches nicht in anderen Labels vorkommt
curbstone	Bordstein, meistens zwischen Strasse und Trottoir
sidewalk	Das Trottoir, eine der wichtigsten Klassen
building	Alle Gebäude werden unter diesem Label erfasst
vegetation	Alle Vegetation, ausser Bäume.
road	Strassen
car	Autos werden einzeln erfasst. Dazu gehören auch Lieferwagen, aber keine LKW
sky	Der Himmel wird als eigene Klasse erfasst
traffic sign	Sämtliche Verkehrszeichen werden als einzelne Klasse erfasst
pole	Masten werden erfasst
tree	Bäume werden als separate Klasse erfasst, da diese auch zur Positionierung eingesetzt werden können
person	Personen werden erfasst, um diese ausblenden zu können
ground	Alles was nicht Strasse oder Trottoir ist und auch nicht Vegetation
fence	Zaun wird als Klasse erfasst
bicycle	Velos werden einzeln erfasst
dynamic	Bewegliche Objekte, welcher keiner anderen Klasse zugewiesen werden können
drain cover	Kanaldeckel werden in dieser Klasse erfasst
crosswalk	Zebrastreifen
motorcycle	Mottorräder
wall	Jegliche Mauern
traffic island	Verkehrsinseln
truck	LKW
rail track	Gleise, Tram und Zug
traffic light	Ampeln
tram	Tram / S-Bahn
guard rail	Leitplanken
bus	Bus
bridge	Brücken
trailer	Anhänger
tunnel	Tunnel

Die Fotos wurden gelabelt, die Verteilung der Labels ist in der folgenden Abbildung zu sehen.

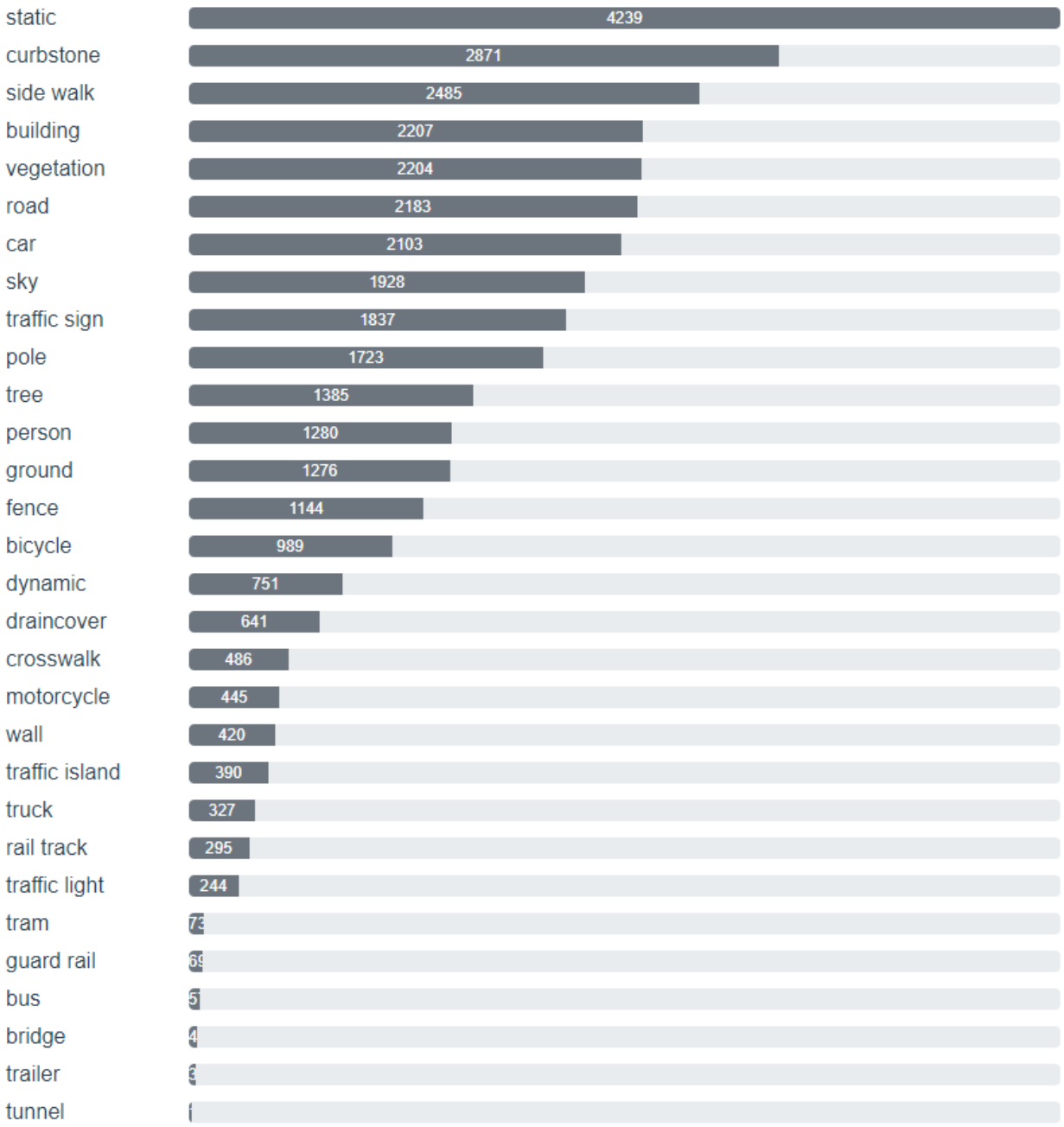
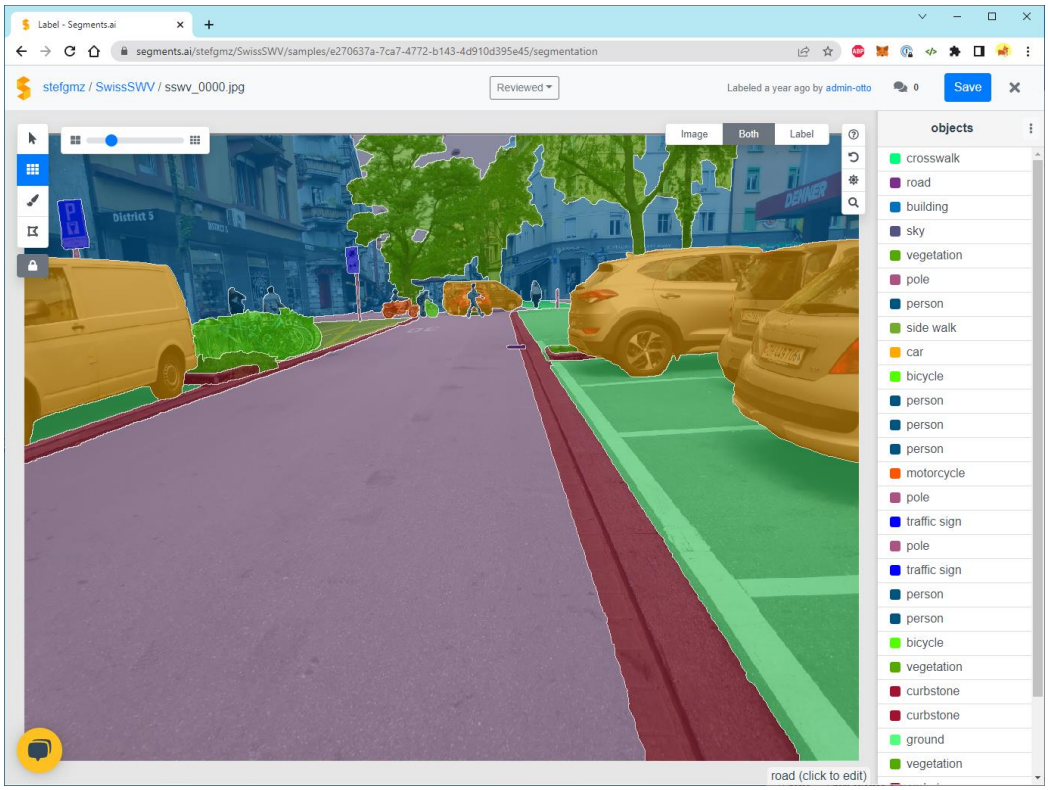


Abb. 10: Verteilung der Erfassten Labels

Für die meisten Labels sind genügend Bildmaterial vorhanden. Einige Labels wie Busse, Brücken, Anhänger und Tunnel waren untervertreten. Dies spielt jedoch keine Rolle, da diese für die Anwendung nicht zentral sind. Es wurden total **2152 Bilder** mit Labeln versehen.

Die Labels wurden über den Webdienst von segments.ai erfasst. Es folgen noch zwei Screenshots der Erfassung:



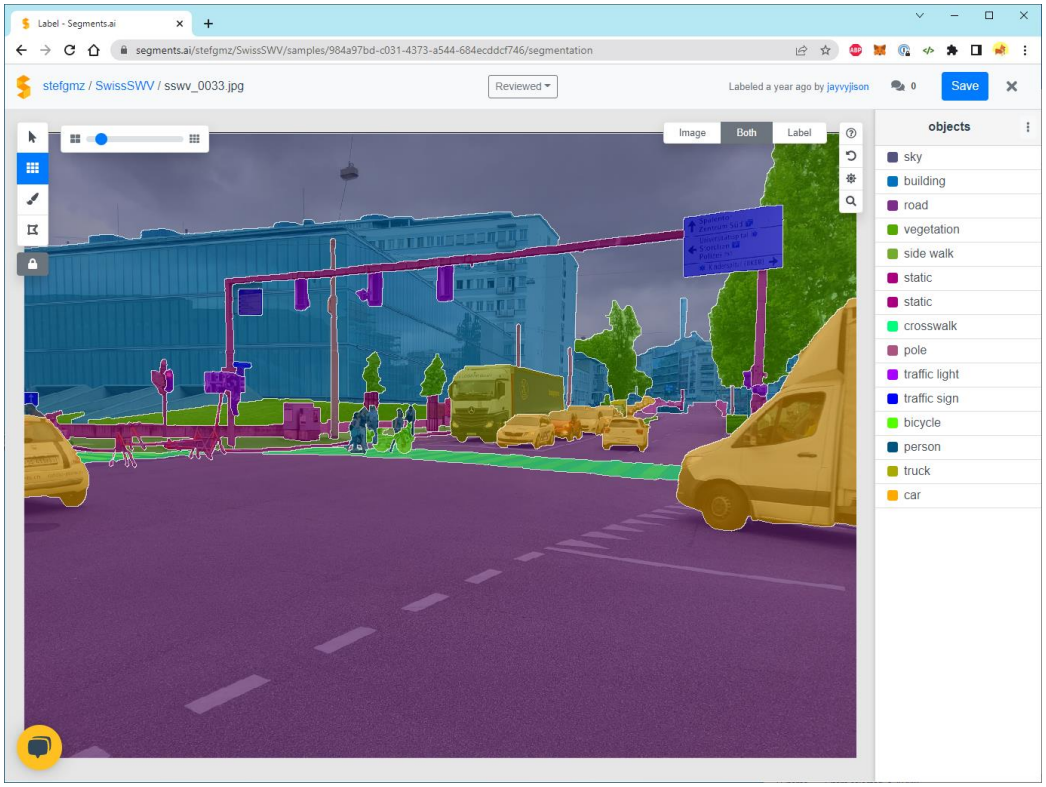


Abb. 11: Beispiele von erfassten Bildern mit segments.ai. Es wurden total 30 Klassen erstellt.

3.5 Erstellen des Neuronalen Netzwerks

3.5.1 Erstellen der Masken aus segments.ai

Nach dem Labeling Prozess konnten aus segments.ai Masken erzeugt werden. Masken sind Bilder, welche die Objekte mit einer bestimmten und bekannten Farbe einfärbt. Über alle Bilder sind dies dieselben Farben. Da wir total 30 Klassen haben, werden auch 30 verschiedene Farben verwendet. Zur Illustration werden zufällige Farben verwendet, in der Realität sind die Farben in RGB die ersten 30 Graustufen, welche fast total schwarz sind.

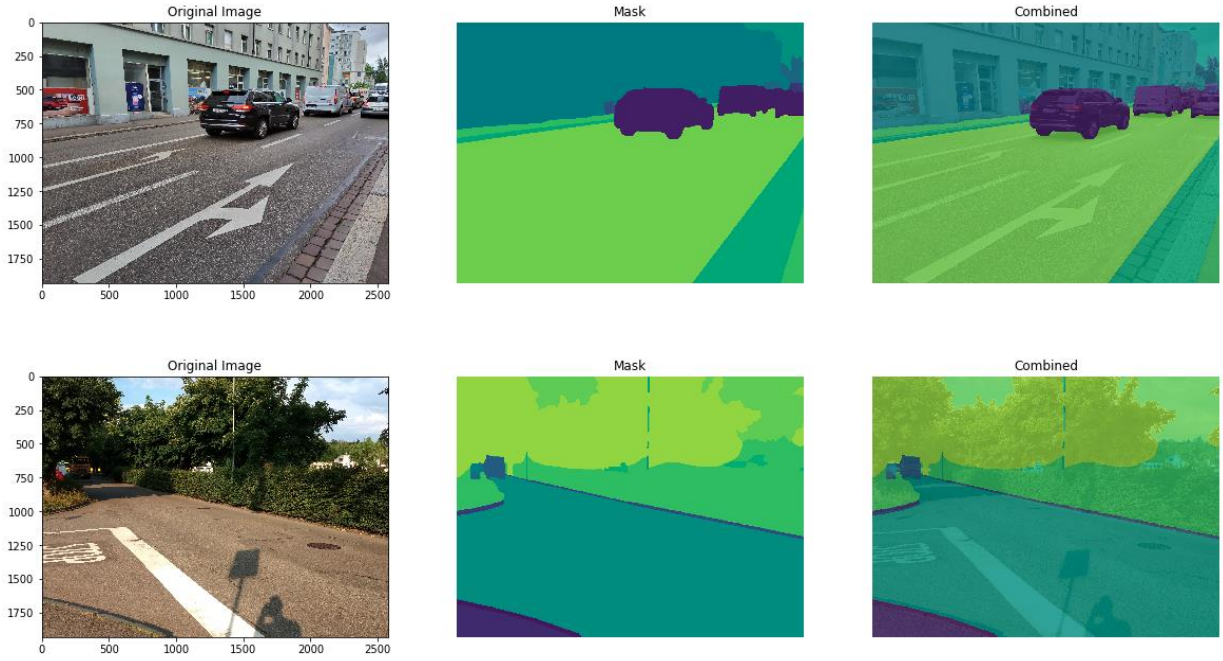


Abb. 12: Bilder, Masken und die kombinierte Ansicht.

3.5.2 Problembilder und Skalierung

Einige Bilder mussten aussortiert werden. Einige wenige Bilder wurden im Hochformat aufgenommen, diese wurden für das neuronale Netzwerk nicht verwendet, da wir entschieden, dass die App vorerst nur im Querformat (“Landscape”) funktionieren soll. Um Hoch- und Querformat zu unterstützen, müssten mehr Bilder im Hochformat vorliegen, dies war aber nicht der Fall.

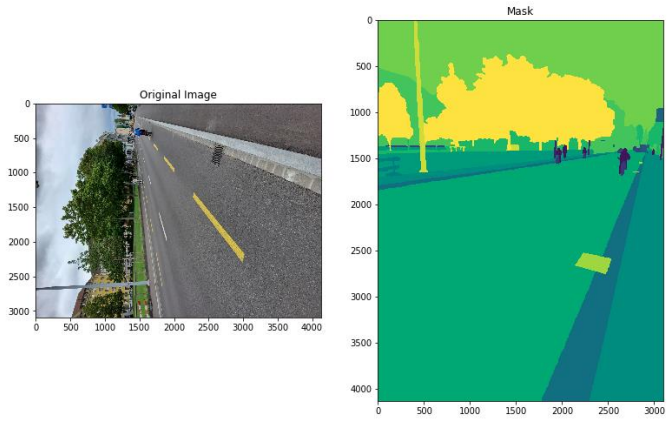


Abb. 13: Bilder im Hochformat werden zurzeit ignoriert

Es wurden total 164 Bilder im Hochformat entfernt. Damit waren 1988 Bilder übrig. Da die Bilder mit verschiedenen Smartphones aufgenommen wurden, lagen diese auch in verschiedenen Auflösungen vor (rot: Hochformat).

Auflösung	Anzahl
4608x2592	564
3968x2976	453
4128x3096	404
4160x3120	289
4032x3024	167
2576x1932	90
5120x3840	67
4032x2272	43
2976x3968	35
3120x4160	26
3840x5120	13
2592x4608	2

Abb. 14: Auflösungen der Bilder, in Rot sind die Hochformat, welche ignoriert wurden

Die Bilder wurden alle einheitlich auf die Auflösungen 2048x1536 und 512x384 skaliert. Mit beiden Auflösungen wurden zwei verschiedene neuronale Netzwerke gerechnet.

3.5.3 Datensatz zur Validierung und zum Testen

Es wurden **1520 Bilder** für das Trainieren des neuronalen Netzwerkes verwendet. Die restlichen Bilder wurden folgendermassen aufgeteilt: **269 für die Validierung** und **199 für das**

Testing. Validierung wird gebraucht, um das beste Modell zu erstellen und Testen wird benötigt um das Modell zu evaluieren.

3.5.4 Trainieren des Modells

Das Modell wurde in 14 Epochen berechnet. Mehr als 14 Epochen gaben keine signifikanten Verbesserungen des Modells. Die Berechnung des Modells erfolgte auf unserem High Performance Compute-Cluster (HPCC) mit 4 von den besten Grafikkarten (Nvidia Tesla V100). Die Bilder sind auf einer Storage angebunden, welche mit 1 GB/s Daten lesen kann. Die Berechnung dauerte rund 12 Stunden pro neuronales Netzwerk.

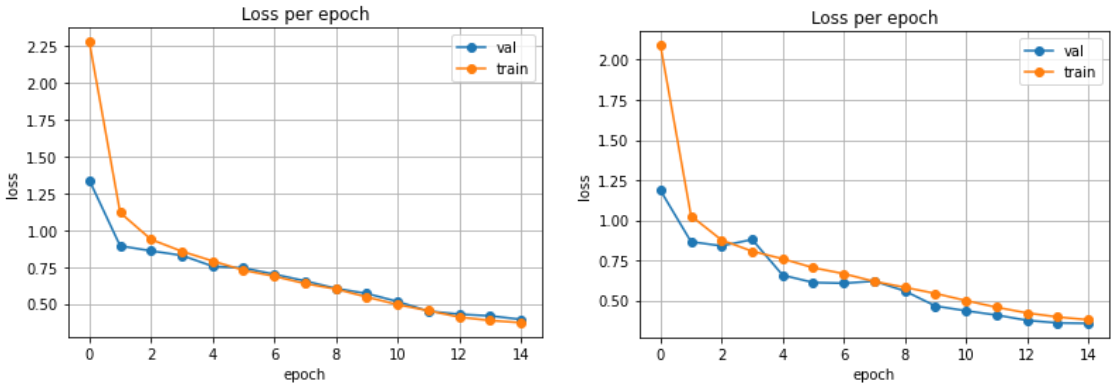


Abb. 15: Loss über 14 Epochen. Links: 512x384 Auflösung und Rechts: 2048x1536

3.5.5 Genauigkeit des Modells

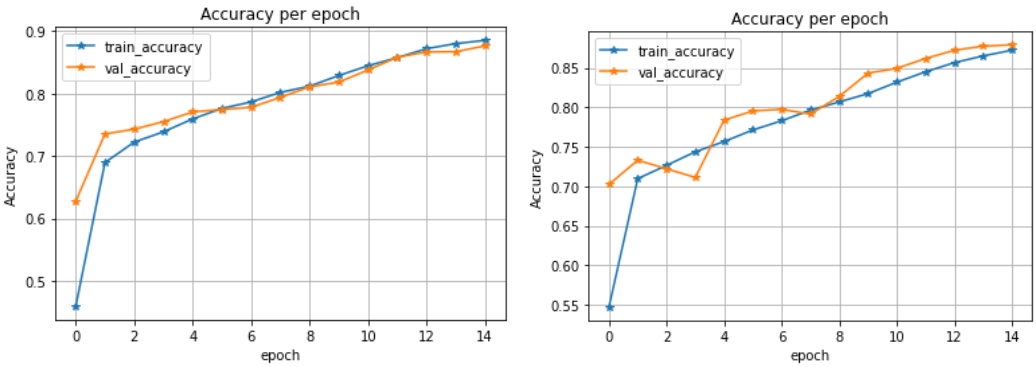


Abb. 16: Genauigkeit über 14 Epochen. Links: 512x384 Auflösung und Rechts: 2048x1536

Nach 14 Epochen wurde eine Genauigkeit von 87% erreicht. Zu beachten ist, dass es durch einige Klassen, welche untervertreten sind (z.B.: Tram, Bus, Tunnel) zu Verlusten kommt. Die wichtigsten Klassen wie Strassen und Trottoirs allein betrachtet erreichen wir Genauigkeiten von über 95%.

Weitere/genauere Zahlen:

Large-Model (2048x1536):

14 Epochen
IoU: 0.55 (über alle Bilder der Validierung)
Pixel Accuracy: 0.87

Min-Model (512x384):

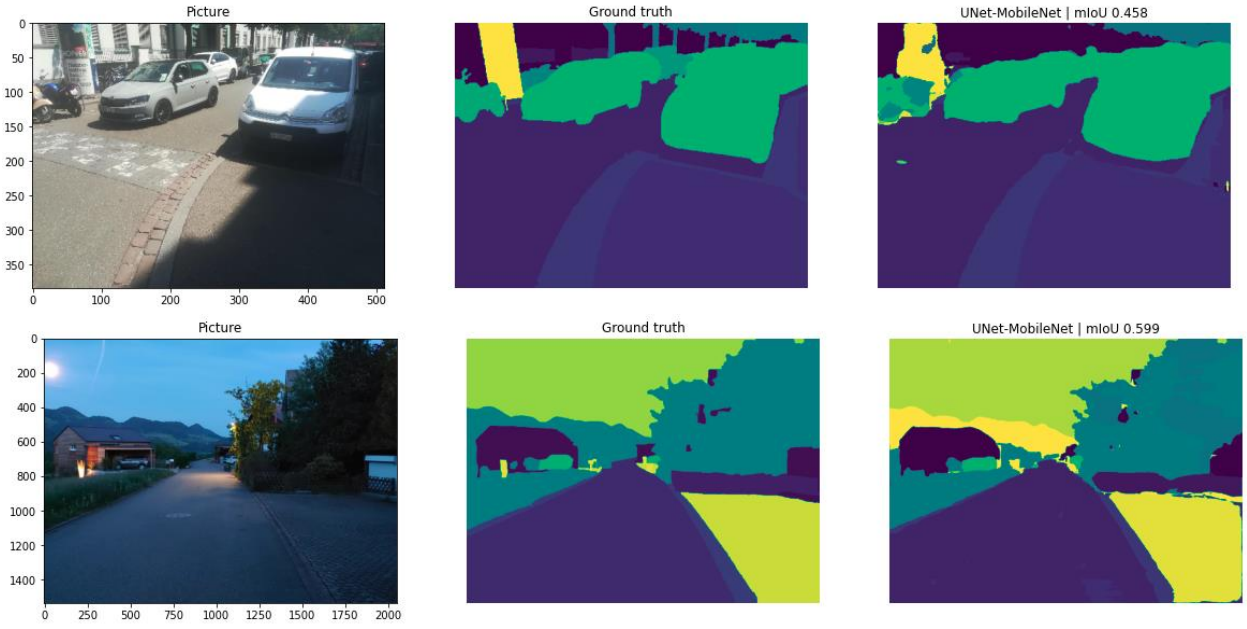
14 Epochen
IoU: 0.48 (über alle Bilder der Validierung)
Pixel Accuracy: 0.88

Eine Auflistung von Wahrheitstabellen sprengt den Rahmen, da dies für jede Klasse einzeln gemacht werden muss. Da wir für viele Klassen einfach zu wenig Daten hatten sind diese auch nicht wirklich nutzbar. Trotzdem ist die mittlere Genauigkeit über alle Klassen von 88% recht hoch. Bei den Impressionen im Kapitel 3.5.6 wird die IoU noch für die einzelnen Bilder angegeben.

Das optimierte Modell ist 26 MB gross. Quantisiert kann es auch direkt auf dem Smartphone angewendet werden, jedoch wurde darauf verzichtet, da die Inferenz bei den ersten Versuchen über 5 Sekunden gedauert hat. Mit Verwendung der GPU auf dem Smartphone könnte dies vermutlich reduziert werden, dieser Ansatz wurde jedoch nicht weiter verfolgt.

3.5.6 Impressionen

Die Bilder des Tests konnten verwendet werden, um die Genauigkeit zu bestimmen. Hier folgen einige Screenshots der Tests, welche visuell schon mal zeigen, wie gut die Detektion funktioniert hat, insbesondere in schwierigen Situationen wie Szenen mit Schatten, Dämmerung, unklare Trottoirs oder Schäden auf Strassen/Trottoirs, siehe Abbildung 17.



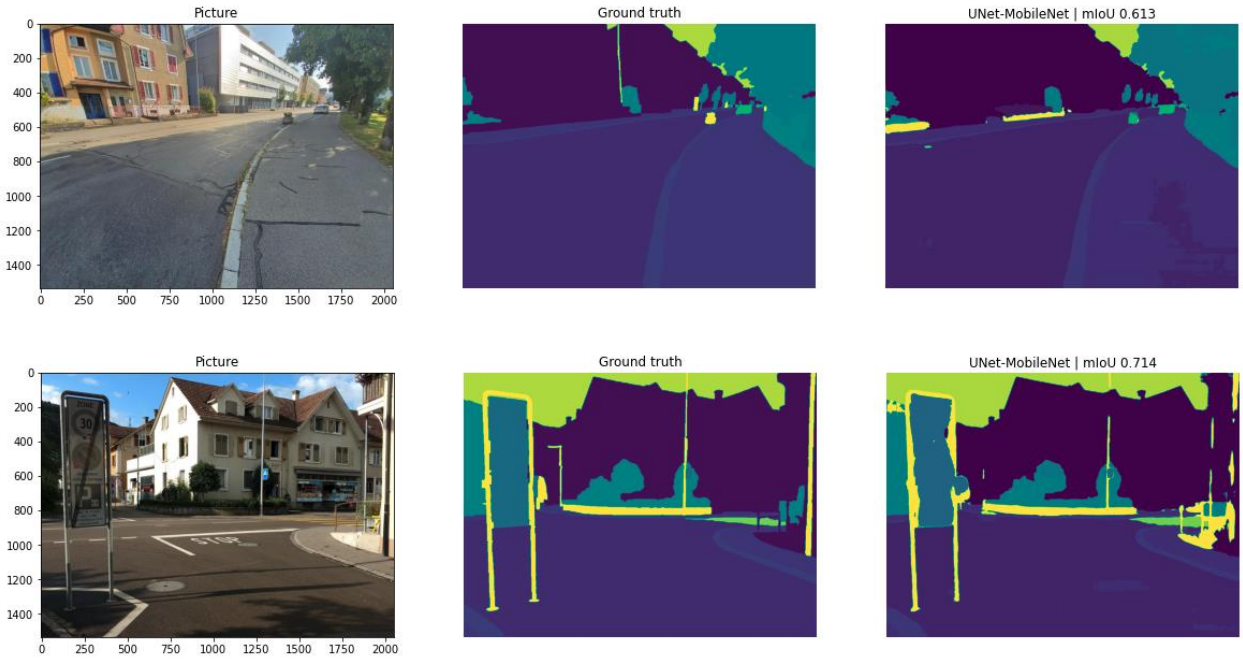


Abb. 17: Links ist jeweils das Foto, in der Mitte ist die Grundwahrheit. Man sieht, dass auch Szenen mit Schatten und andere spezielle Situationen problemlos funktionieren.

3.6 Webdienst für die Strassen-Detektion

Um das Resultat auf einer App nutzen zu können, wurde ein Webdienst entwickelt, welcher ein hochgeladenes Foto automatisch detektiert und als Resultat ein segmentiertes Bild zurückgibt. Der Dienst ist verfügbar unter https://street.argeodata.ch
Zu beachten ist jedoch, dass die App nur im **Querformat** funktioniert. Es gibt auch keine Warnung, dass man in das Querformat wechseln soll, da dies nur eine Demo App für den Webdienst gedacht ist und keinen weiteren Nutzen hat. Das Resultat dieser Segmentierung soll schliesslich von Apps verwendet werden können und nicht dem Benutzer angezeigt.

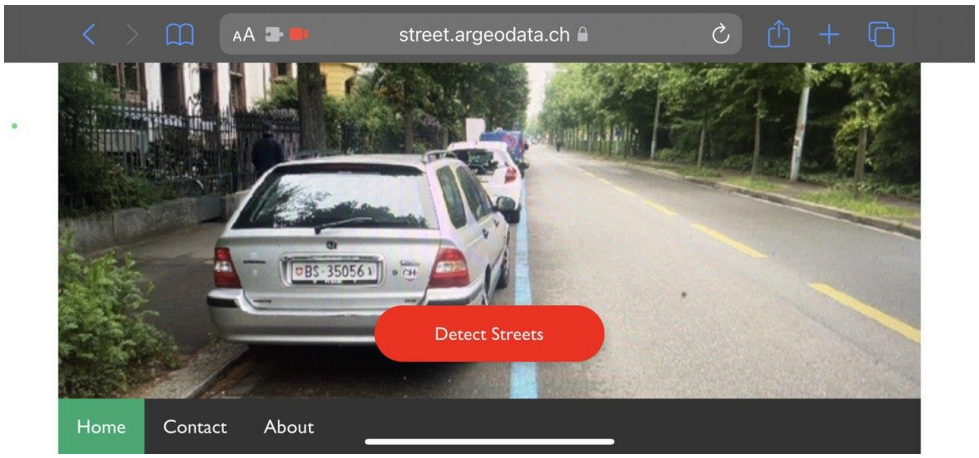


Abb. 18: Web-App zum Testen des Web-Dienstes zur Strassendetektion. Es benötigt die Kamera des Smartphones. Die App sollte nur im Querformat verwendet werden. Funktioniert mit den meisten gängigen Browsern (Safari auf iOS und Chrome auf Android)

3.7 Webdienst für die Tiefenkarte

Ein weiterer Teil der Cloud Lösung besteht aus dem Hochladen der Tiefenkarte zu dem aktuellen Bild. Dies wird benötigt, um die Positionen jedes Pixels in das Welt-Koordinatensystem zurückzurechnen und damit auch Messungen im Bild vorzunehmen.

Sowohl ARKit (Apple) wie auch ARCore (Google) bieten das Erfassen einer Punktwolke an. Erste Untersuchungen zu Beginn des Projektes haben gezeigt, dass die Tiefenkarten resp. die daraus resultierende Punktwolke mit beiden Toolkits nicht ideal ist, wie in Abbildung 19 zu sehen ist.

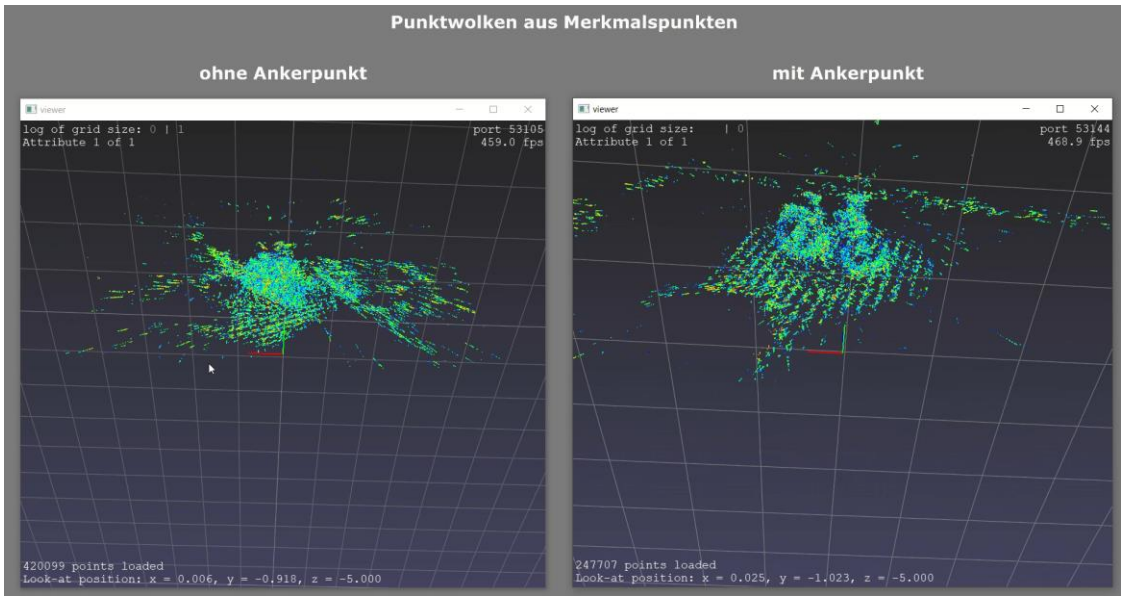


Abb. 19: Erfassung eines Velo mit einem Samsung Galaxy Tab S. Die resultierende Punktwolke ist unbrauchbar. Das Velo ist ein wenig erkennbar, aber die Streuung der Punkte ist zu gross.

Dies verunmöglicht das Vorhaben eine genaue Tiefenkarte mit dem Smartphone zu berechnen. Die Tiefenkarte hätte Abweichungen im Bereich 20-30cm, und das ist nicht ausreichend für unser Vorhaben. Einige Android Modelle wie z.B. Samsung Galaxy haben zusätzlich eine Range Kamera verbaut. Mit diesen funktioniert das Erfassen viel besser. Auch unter iOS wird die Tiefenkarte realistisch, sobald ein LiDAR Sensor vorhanden ist (iPhone Pro 12, iPhone Pro 13, iPad Pro 2020 und später). Erneute Untersuchungen haben gezeigt, dass die Tiefenkarte mit diesen Geräten funktioniert. Für die Genauigkeitsuntersuchung der Tiefenkarte wurde ein Test-Setup aufgebaut, welches im folgenden Bild schematisch dargestellt wird. Dies wurde im Rahmen einer Bachelor Thesis (Müller, 2021) untersucht.

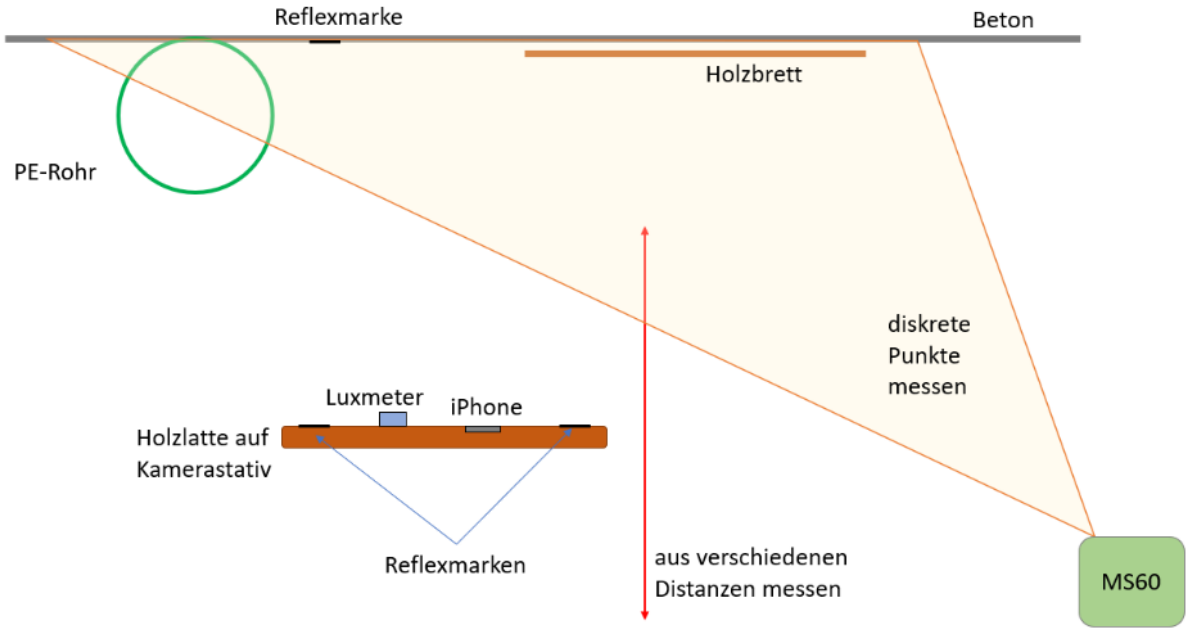


Abb. 20: Setup für die Genauigkeitsuntersuchung der Tiefenkarte des iPhone 12. (Müller, 2021)

Im ersten Schritt wurde bestätigt, dass die Beleuchtung keinen Einfluss auf die Genauigkeit der Tiefenkarte hat. Im zweiten Schritt wurden die Tiefenkarten mit dem eigens entwickelten iPhone Tool erstellt. Das Resultat dieser Genauigkeitsuntersuchung ist in der untenstehenden Tabelle (Müller, 2021) zu finden.

Standardabweichung bzw. Präzision in mm				
Abstand	**Beton**	**Holz**	**Reflex**	**Rohr**
0.5 m	0.71	0.84	0.57	1.75
1 m	0.86	1.13	0.85	3.40
2 m	1.33	1.23	1.49	2.00
3 m	2.80	2.75	1.33	7.08
4 m	3.87	4.26	1.81	6.64
5 m	5.64	4.63	3.39	7.71
6 m	13.94	24.42	13.74	28.03
7 m	56.70	48.83	53.98	15.83

Abb. 21: Für unterschiedliche Materialien und Aufnahmedistanzen wird hier die Standardabweichung in Millimeter (mm) angegeben.

Mit dieser Genauigkeit funktioniert der Webdienst für den Tiefenkarten-Upload.

Das Tiefenkarten Upload Tool ermöglicht das zusätzliche Hochladen von Tiefenkarten zu jedem Bild. Zusätzlich zur interpolierten Tiefenkarte wird auch eine Konfidenz-Map mitgeliefert, welche die Qualität von jedem Pixel angibt. Im Beispiel unten sind schlechte Werte als rot eingefärbt. Dieser werden zur Positionsberechnung nicht verwendet. Als Beispiel wurde ein Bild verwendet, welches deutlich mehrere Ebenen aufweist (Tisch, Wände, Bildschirm). Die Filegrösse ist ca. 5 KB pro Upload (ohne RGB)

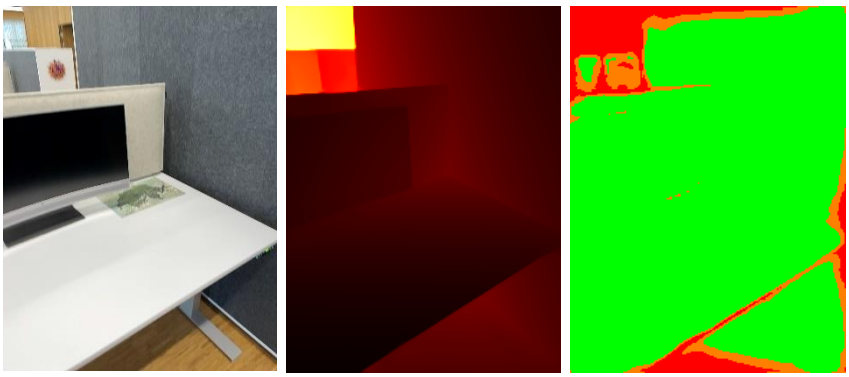


Abb. 22: Beispiel einer Tiefenkarte (Mitte) und der Konfidenzkarte (rechts).

Mit der Tiefenkarte und den verschiedenen Experimenten sehen wir, dass der LiDAR Sensor der aktuellen iPhones/iPads für AR-Applikationen ideal ist. Als visueller Vergleich wurde das Velo mit einem solchen Gerät nochmals aufgenommen und eine Punktwolke erstellt. Das Resultat sieht komplett anders aus und das Velo ist nun deutlich erkennbar. Feine Details wie Speichen sind jedoch nicht sichtbar.

Abb. 23: Erfassung eines Velos mit dem iPhone 13. Die resultierende Punktwolke ist besser als das erste Experiment mit dem Android Tablet.

3.8 Webdienst: Streaming von 3D Objekten und WMS/WMTS

Analog zum Webdienst für die Strassendetektion wurde ein Service entwickelt, welcher 3D Objekte in einer Bounding Box abfragen kann (WP3) resp. einen WMS/WFS Dienst (WP4). Die 3D Objekte wurden von SWISSBUILDINGS3D verwendet. Jedoch ist das gelieferte Format nicht ideal, so wurde manuell nur ein Teil von Basel-Stadt extrahiert. Momentan sind 3D Objekte auch nur in dieser Region vorhanden. Es wäre möglich dies zu erweitern, aber benötigt für alle Regionen momentan einen manuellen Prozess. Es wäre wünschenswert die 3D Objekte von SWISSBUILDINGS3D würden in einerm einfacheren Grafikformat objektweise vorliegen, das würde den Prozess besser automatisierbar machen. Die 3D Objekte werden momentan in der App jedoch noch nicht dargestellt. Die Darstellung von 3D Objekten ist technisch zwar nicht ein Problem, wurde aus Zeitgründen jedoch nicht umgesetzt,

Andere Datenquellen (z.B. OpenStreetMap, kantonale 3D-Stadtmodelle etc.) werden momentan nicht unterstützt, könnten aber zu einem späteren Zeitpunkt auch hinzugefügt werden.

Über den Webdienst können mittels Bounding Box die 3D Daten abgefragt werden und kommen als Wavefront OBJ zurück. Die Koordinaten sind in LV95. Um die Koordinaten klein zu halten, kann auch optional ein Offset berechnet werden, welches von jeder Koordinate abgezogen wird.

Für WMS/WFS wurde die Open Source Bibliothek OWSLib verwendet. https://github.com/geopython/OWSLib

3.9 Weitere Genauigkeits-Untersuchungen mit dem iPhone LiDAR

Ziel dieser Untersuchung im Rahmen der Bachelorthesis von Müller, 2021 war, die relative Messgenauigkeit des iPhone 12 Pro zu ermitteln. Auch kann mit dieser Untersuchung der Einfluss der Objektausdehnung auf die Genauigkeit ermittelt werden. Dafür wurde ein Teil des Campus der FHNW gescannt. Nicht nur die ungefähr 57 m lange Seite wird gescannt, sondern auch die angrenzende ca. 5 m kurze Fassade. Die Messungen wurden mit den AV-Daten verglichen. Der erfasste Bereich ist in der folgenden Abbildung blau markiert.

Abb. 24: AV Grundkarte mit dem Messbereich für die Untersuchung der relativen Genauigkeit blau (Daten: GIS-Fachstelle BL)

	iPhone Pro 13 - Messungen					
	lange Wand			kurze Wand		
	Anz. Pt	worst Pt	Stabw.	Anz. Pt	worst Pt	Stabw.
M1	179’736	0.151 m	0.0474 m	18’724	0.0518 m	0.0102 m
M2	183’463	0.141 m	0.0245 m	20’028	0.0437 m	0.0092 m
M3	167’550	0.134 m	0.0260 m	21’706	0.0736 m	0.0122 m
M4	174’893	0.122 m	0.0276 m	20’715	0.0445 m	0.0121 m
M5	191’352	0.122 m	0.0255 m	23’913	0.0381 m	0.0075 m

Abb. 25: Messungen der Wände

	lange Wand	kurze Wand	Winkel
Präzision	0.228 m	0.034 m	0.358 °
Bias	0.886 m	-0.004 m	0.033 °

Abb. 26: Präzision und Bias

Die Messungen zeigen, dass kurze Strecken eine recht gute Präzision haben, bei längeren Scans gibt es grössere Abweichungen. Zudem konnte gezeigt werden, dass auf 60 Meter ein Drift von rund 20cm auftaucht, dies entspricht den Angaben des Herstellers.

3.10 Entwicklung der AR App

Die Referenz-App wurde für iOS Entwickelt. Aus Zeitgründen wurde darauf verzichtet die Android Version weiterzuentwickeln. Insbesondere auch, da die Genauigkeitsuntersuchungen noch nicht mit den Range-Sensoren gemacht werden konnte und die Qualität bei den ersten Untersuchungen gezeigt, hat dass es noch Mängel gibt.

Ein Prototyp der Android App ist zwar noch verfügbar, jedoch nicht kompatibel mit den finalen Webservices. Die Android App müsste nochmals komplett implementiert werden, da sich die Spezifikation der Webdienste während der Projektlaufzeit verändert hat.

Die Entwicklungsumgebung unter iOS war Swift. Unterstützt werden iPhones Pro und iPads Pro. Die Applikation ist somit komplett native entwickelt.

3.10.1 Funktionsweise im Gesamtüberblick

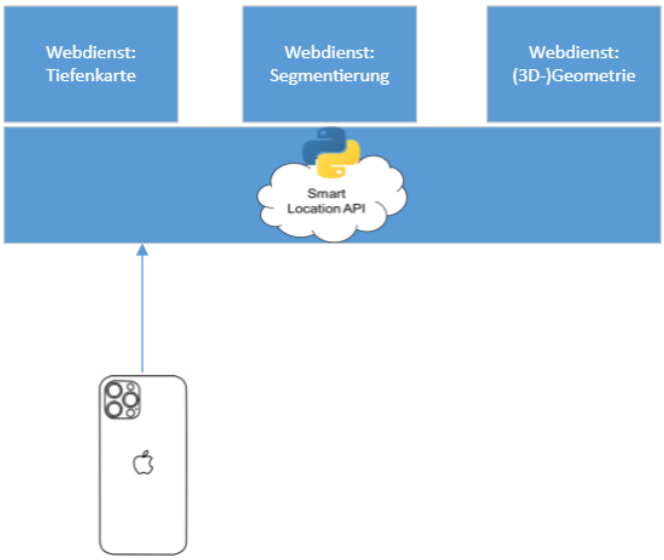


Abb. 27: Alle APIs / Webdienste im Überblick

Das Smartphone oder Tablet sendet ein skaliertes Bild + Tiefenkarte + Konfidenzkarte + Detektierte Ebenen + Sensordaten (Orientierung/Position in lokalen Koordinaten) zum Webdienst “SmartLocationAPI”. Die Grösse des gesendeten Bildes kann auf 2048x1536 und 512x384 eingestellt werden. Je nach Bandbreite kann es sinnvoll sein die kleinere Repräsentation zu nehmen.
Das Bild wird dann an den Segmentierungs-Webdienst weitergegeben. Die Segmentierung wird auf die Ebenen projiziert und mit AV-Daten gematched. Dieser Prozess wiederholt sich für jedes Bild. Der Datentransfer zwischen dem Smartphone/Tablet und dem Webdienst ist

überschaubar bei ca. 500KB-3MB Pro Paket, je nach Qualitätseinstellung, jedoch der Bottleneck ist momentan die Prozessierungsdauer in der Cloud, welche relativ hoch ist. Es kann ca. 1 Paket pro 5 Sekunden prozessiert werden (30% Segmentierung / 70% Matching)

3.10.2 Anleitung SmartLocationAPI

Die SmartLocation API wird ständig weiterentwickelt. Eine Dokumentation gibt es zur Zeit nur in Form des Source-Codes.

3.10.3 Aufbau der App

Die App funktioniert auf **iPhones 12** und **13** und **iPad Pro** mit LiDAR Sensor. Die App startet auf dem Homescreen und auf dem Bildschirm unten ist eine Menüleiste sichtbar.

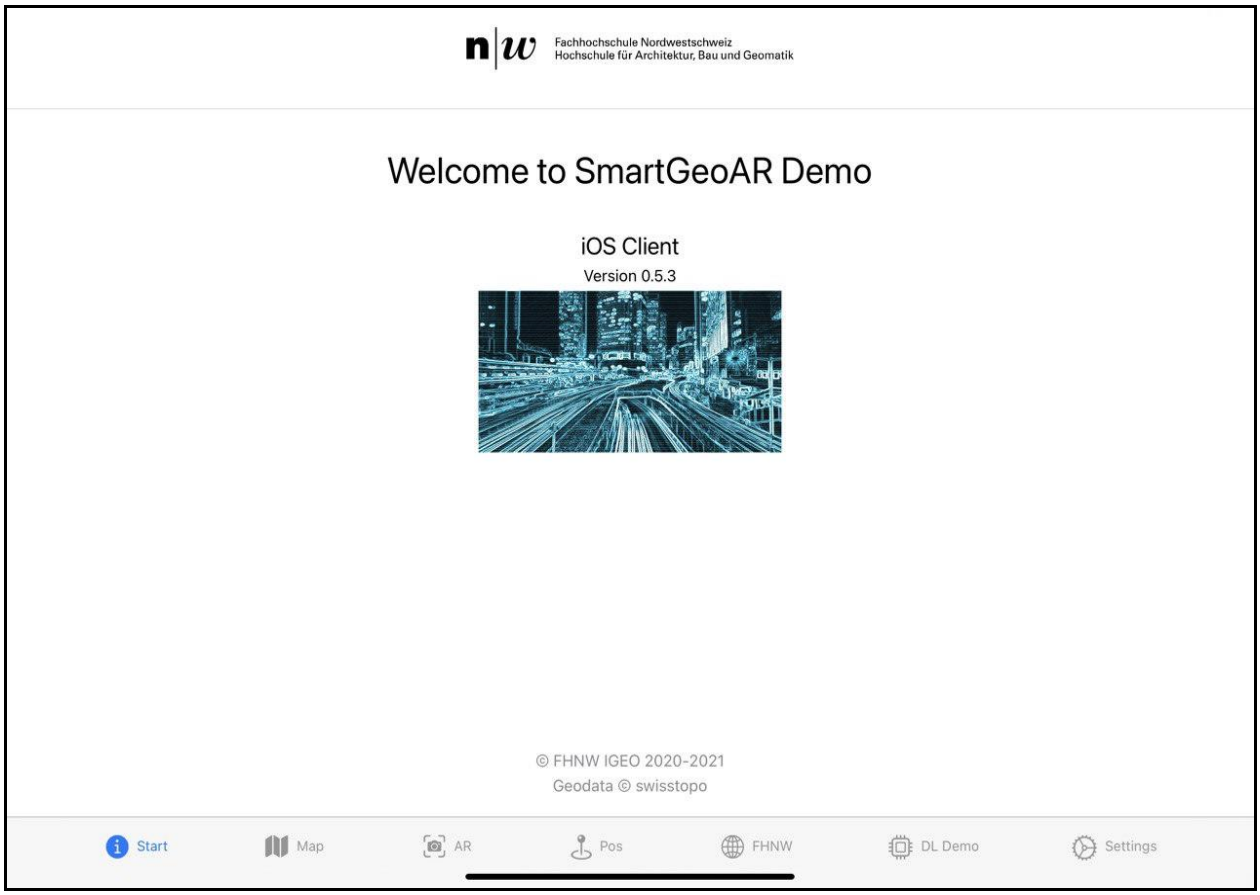


Abb. 28: Der Willkommensscreen auf iPad Pro.

Die Navigationsleiste besteht aus folgenden Elementen:

- **Start**: Der Willkommensscreen mit Copyright Informationen
- **Map**: Kartenansicht mit der aktuellen Position *(Momentan nur die Standard Apple Karte und noch kein allgemeiner WMTS Support)*

- **AR**: Die Augmentierte Ansicht
- **Pos**: “Position Debugger”. Informationen über die aktuelle Position des Gerätes in lokalen Koordinaten/WGS84 und LV95. Auch die Orientierung wird dort zusätzlich angezeigt. Dieser Menüeintrag wurde vor allem für die Entwicklung gemacht und hat keinen weiteren Nutzen.
- **DL-Demo**: Deep Learning Demo. Hier werden Daten vom Segmentierungs-Algorithmus angezeigt. Auch dieser Tab wurde für Entwicklungs-/Demonstrationszwecke hinzugefügt und hat keinen weiteren Nutzen.
- **Settings**: Hier können einige Einstellungen vorgenommen werden wie z.B. Paketgrösse (Bildgrösse, welche gesendet wird).

Abb. 29: Der AR-Screen mit Überlagerung vordefinierter Punkten (Musical Theater Basel)

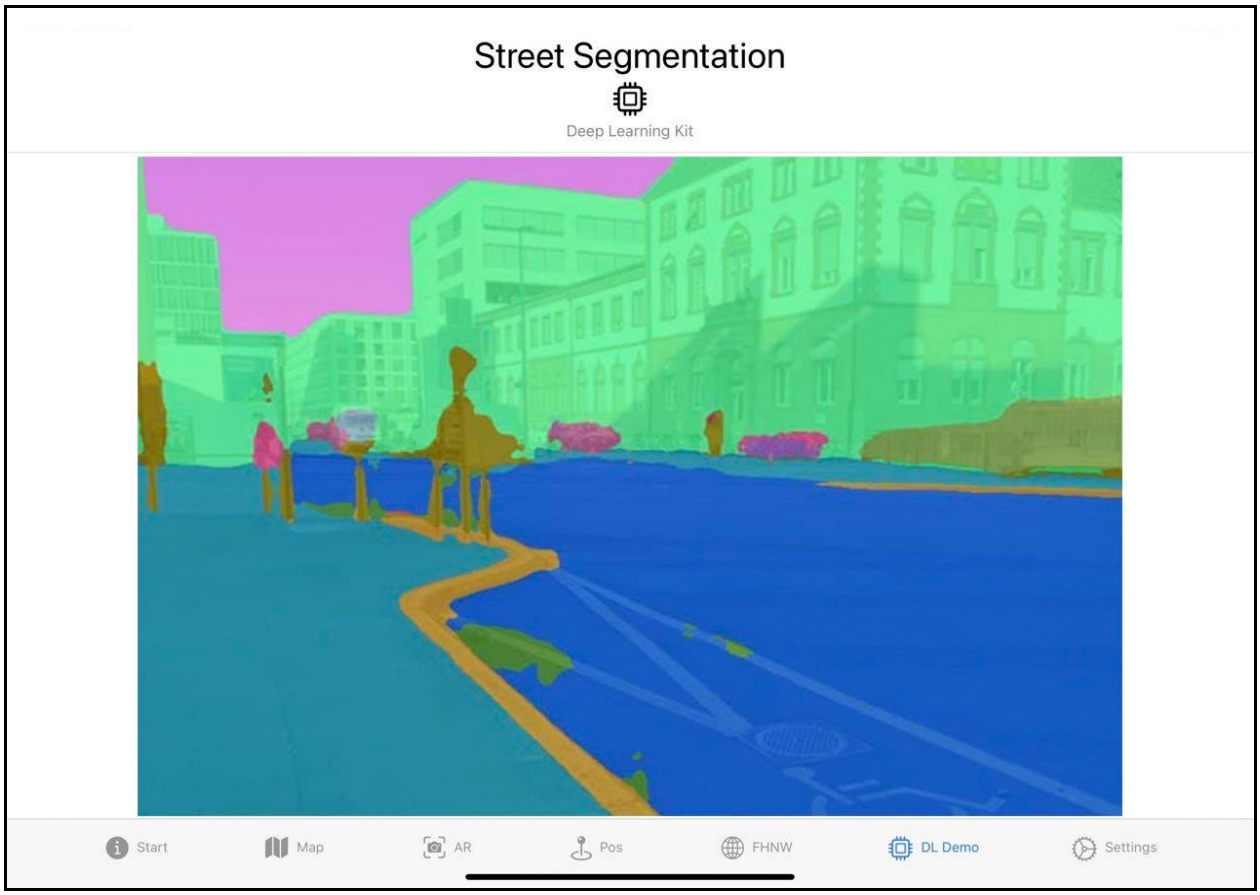


Abb. 30: Die Segmentierungsansicht, kombiniert mit dem Bild.

Bei der Segmentierung sieht man schön, wie die Gebäude praktisch pixelgenau detektiert werden.

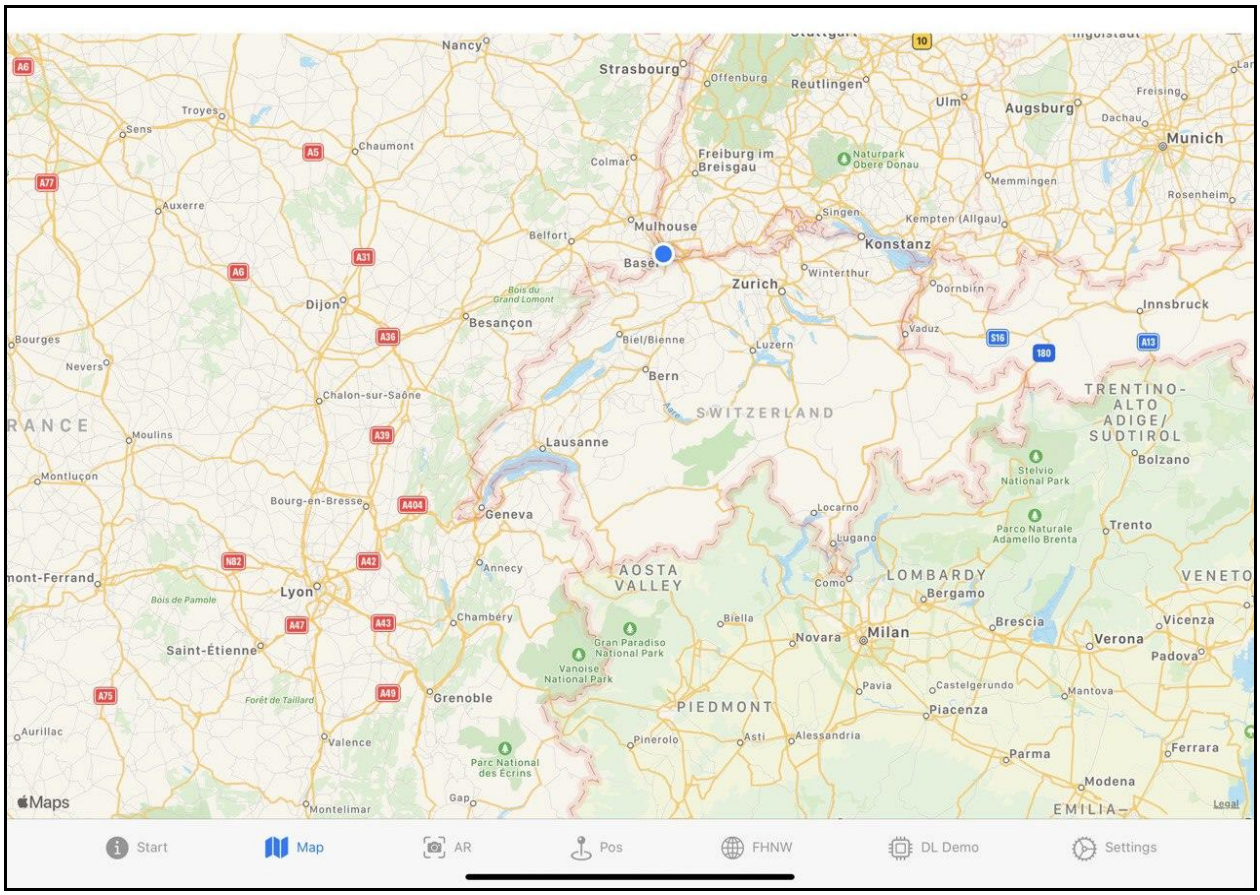


Abb. 31: Die Kartenansicht verwendet im Moment noch die Apple Karte (basierend auf OpenStreetMap). In Zukunft soll diese Karte durch die Pixelkarte von swisstopo ersetzt werden.

3.11 Installation der AR App

Die Installation der App erfolgt direkt über den Source Code mittels XCode. Die App ist nicht im App Store erhältlich und dies ist zur Zeit auch nicht geplant, da die nötige Server-Infrastruktur für eine breitere Anwendung fehlt.

3.11.1 Genauigkeit und Restriktionen der Detektion

Die Detektion funktioniert nur bei Strassenabschnitten mit eindeutigen Merkmalen. Im Abschnitt oben war das beispielsweise die markante “Schwingung” im Trottoir. Momentan wird auch nur der **Trottoir-Rand (Klasse “curbstone”)** für das Matching verwendet. Im folgenden Bild ist eine Detektion beispielsweise nicht möglich, da keine eindeutigen Merkmale für die Positionierung detektiert werden können. Wenn jedoch detektiert werden kann ist die relative Genauigkeit im Zentimeterbereich für Objekte im Abstand bis 2m.

3.12 Fehlende Features

Einige Dinge wurden noch nicht entwickelt:

- Die AV-Daten werden nicht synchronisiert. Es ist ein statischer Zustand auf dem Server, die Daten wurden manuell abgelegt und es befinden sich auch nicht alle Kantone auf dem Server. Die App funktioniert zurzeit dementsprechend nur in Teilen von Basel-Stadt und Basel-Land. Andere Kantone funktionierten vermutlich wurden aber nicht getestet. Copyright Abklärungen/Verwendungszweck diesbezüglich sind auch noch ausstehend.
- Die Anbindung des 3D-Webdienstes an die Smart Location API ist aus Zeitgründen noch nicht fertig programmiert und damit können noch keine 3D-Objekte überlagert werden.
- Der Matching Algorithmus verwendet nur den Trottoir-Rand (curbstone). Andere Klassen werden noch nicht verwendet. Es wäre sinnvoll weitere Klassen schrittweise hinzuzufügen um die Detektion zu verbessern.
- Momentan wird nur der Kartenviewer von Apple verwendet, eine Einbindung der WMTS Karten von Swisstopo ist noch ausstehend. Da dies kein essenzielles Feature ist, wurde es auch nicht hoch priorisiert. Bei der iOS Entwicklung ist es viel einfacher die integrierte Apple-Karte zu verwenden als eine andere, welche ein zusätzliches SDK erfordert. Eine Einbindung einer anderen Karte war nicht prioritär.

4 Fazit und Ausblick

4.1 Publikationen und Vorträge

Publikationen und Vorträge sind für 2022/23 geplant, sobald sich die Lage mit Covid-19 ein wenig entspannt und Konferenzen wieder normal durchgeführt werden. Das Projekt wurde an der Online GeoPython 2020 (September 2020) Konferenz vorgestellt. Im Rahmen dieser Vorträge wird auch versucht eine OpenSource Community um das Projekt aufzubauen.

4.2 Weiterentwicklungen

In Zukunft sind Weiterentwicklungen im Rahmen von Bachelor-Thesen und Master-Thesen geplant, welche insbesondere die Skalierung der Cloudlösung weiterentwickeln sollen. Auch der Einbezug von bestehenden Punktwolken für das Matching bringt eine bessere Stabilität in der Positionsbestimmung.
Eine Weiterführung des Projektes im Rahmen eines Innosuisse-Projektes wäre auch denkbar.

5 Quellen

Apple Inc. (2020): Apple introduces iPhone 12 Pro and iPhone 12 Pro Max with 5G. Press Release. Online verfügbar unter https://www.apple.com/newsroom/2020/10/apple-introduces-iphone-12-pro-and-iphone-12-pro-max-with-5g

Chollet, Francois (2017). *Deep Learning with Python*. Shelter Island, NY: Manning Publications Co.

Goodfellow, Ian; Bengio, Yoshua & Courville, Aaron (2018). *Deep Learning: das umfassende Handbuch: Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze*. Auflage: 1. Auflage, Frechen: mitp.

Jean, Hadrien (2018). *Introduction to Scalars Vectors Matrices and Tensors using Python/Numpy examples and drawings*. hadrienj. https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.1-Scalars-Vectors-Matrices-and-Tensors/

Kling, Andrea (2018). *KI testen? Qualitätskriterien für Machine Learning*. Nagarro. http://www.anecon.com/blog/ki-testen-qualitaetskriterien-fuer-machine-learning/

Koehrsen, Will (2018). *Transfer Learning with Convolutional Neural Networks in PyTorch*. Towards Data Science. https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce

Liu, Yanfeng (2019). *The Confusing Metrics of AP and mAP for Object Detection / Instance Segmentation*. Medium. https://medium.com/@yanfengliux/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef

Luber, Stefan (2017). *Was ist Deep Learning?*. bigdata-insider. https://www.bigdata-insider.de/was-ist-deep-learning-a-603129/

Luber, Stefan & Litzel, Nico (2019). *Was ist ein Convolutional Neural Network?*. bigdata-insider. https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/

Müller, K. (2021), Leistungsuntersuchung 3D-Erfassung offener Gräben mit LiDAR Smartphones

Rosebrock, Adrian (2016). Intersection over Union (IoU) for object detection. PyImageSearch. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Tan, Ren Jie (2019). *Breaking Down Mean Average Precision (mAP)*. Towards Data Science. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

Thomas, Sherin & Passi, Sudhanshu (2019). *PyTorch deep learning hands-on: apply modern AI techniques with CNNs, RNNs, GANs, reinforcement learning, and more*. Birmingham: Packt.