

Bachelorarbeit

Fast and Curious

Entwicklung eines Low-Cost Verkehrsflussmesssystems

Josef Böckle
Brauentinweg 12
6713 Ludesch
josef.boeckle@ntb.ch

Daniel Lüchinger
Oberauweg 5d
9453 Eichberg
daniel.luechinger@ntb.ch

4. August 2017

Inhaltsverzeichnis

1 Danksagung	3
2 Abstract	4
3 Einleitung	6
3.1 Aufgabenstellung	6
3.2 Wertschöpfung	6
4 Fast and Curious	7
4.1 Anforderungen	7
4.1.1 Robustheit	7
4.1.2 Laufzeit	7
4.1.3 Verarbeitungszeit	7
4.1.4 Kosten	7
4.2 Umwelteinflüsse	7
5 Hardware	8
5.1 Komponenten	8
5.1.1 Rechnereinheit	8
5.1.2 Messsensor	9
5.1.3 WIFI-Adapter	10
5.1.4 Spannungsversorgung	10
5.1.5 Gehäuse	11
5.2 Gesamtsystem	12
6 Software	14
6.1 Verwendete Tools und Libraries	14
6.1.1 Mjpg-Streamer	14
6.1.2 Avconv	14
6.1.3 Apache	14
6.1.4 OpenCV	15
6.2 Architektur	15
6.2.1 Aufnahme	16
6.2.2 Vorverarbeitung	16
6.2.3 Nachverarbeitung	19
6.2.4 Steuerung und Überwachung	23
6.3 Github	25
6.3.1 Ordnerstruktur	25
7 Schwierigkeiten	27
7.1 Beaglebone Green Wireless	27
7.2 Radarsensor	27
7.3 OpenCV	28
7.4 WLAN	28
7.5 Videoaufnahme	29
8 Bedienungsanleitung	30
8.1 Installation	30
8.2 Updates	30
8.3 Aufstellung des Gerätes	31

8.4 Systemstart	31
8.5 Monitoring	32
8.6 Datenerwerb	32
9 Testversuche	33
9.1 Prototypen	33
9.2 Video mittels Laptop	33
9.3 Video mittels Beaglebone	34
9.4 Verarbeitung	34
9.4.1 Schattenentfernung	34
9.4.2 GrabCut	36
9.4.3 K-Means Clustering	36
9.4.4 Kategorisierung	37
9.4.5 Geschwindigkeit	38
10 Resultate	39
10.1 Satteins	39
10.1.1 Topologie	39
10.1.2 Erfassung der Daten	40
10.2 Einzelnes Gerät	41
10.2.1 Zählen	41
10.2.2 Kategorisieren	41
10.2.3 Geschwindigkeitserkennung	41
10.3 System	44
11 Ausblick	46
11.1 Erweiterung des Gerätes	46
11.2 Features erweitern	46
11.2.1 Fahrzeuggänge & -breite	46
11.2.2 Kategorisieren	46
11.2.3 Farbe	46
11.2.4 Farbfläche	46
11.2.5 Sonderanbringungen	46
11.2.6 Nummerntafeln	47
11.2.7 Geschwindigkeit	47
11.3 Optimale Aufstellung	47
11.4 Vernetzung	47
11.5 Echtzeitdarstellung	47
11.6 Spannungsversorgung	48
12 Verzeichnisse	49
12.1 Literatur	49
12.2 Quellen	50
12.3 Tabellen	50
12.4 Abbildungen	51
12.5 Abkürzungen	52
12.6 Glossar	53
13 Eidestattliche Erklärung	54
14 Anhang	55

1 Danksagung

An dieser Stelle möchten wir Herrn Thomas Vogt und der Adlos AG danken, weil wir für sie diese Bachelorarbeit durchführen durften. Herr Vogt stand uns bei Fragen und Anliegen jederzeit zur Seite und hat uns einige Male mit zusätzlicher Hardware ausgeholfen, wenn wir etwas benötigten.

Weiterhin möchten wir unserem Koreferenten Herrn Prof. Dr. Tindaro Pittorino danken. Besonders bei der Auswahl des Mikroprozessors und bei sämtlichen Fragen bezüglich Elektrotechnik konnten wir jederzeit mit seiner Kooperation rechnen. Ausserdem durften wir durch ihn unsere Bachelorarbeit bereits am Technologietag der NTB vor zahlreichen Interessenten präsentieren.

Ebenso geht ein besonderer Dank an Herrn Prof. Dr. Klaus Frick, unserem Referenten für diese Bachelorarbeit. Er unterstützte uns mit all seinen Mitteln, stand uns jederzeit, auch am Wochenende, zur Verfügung, besorgte die notwendige Hardware um die Geräte herstellen zu können und half uns die Geräte in Satteins anzubringen.

Zuletzt möchten wir uns noch bei der Gemeinde Satteins bedanken, da wir dort unsere Geräte aufstellen und austesten durften. Dadurch war es uns möglich diese in einer realen Umgebung zu testen und jederzeit Anpassungen durchführen zu können, wenn dies notwendig war.

2 Abstract

Das Ziel von "Fast and Curious" war es, ein System zu entwickeln, welches aus einzelnen Geräten besteht, die vorbeifahrende Verkehrsteilnehmer zählen, kategorisieren und deren grobe Geschwindigkeit einschätzen können. Durch den Zusammenschluss der Geräte, sollte es möglich sein eine statistische Aussage über das gesamte Fahrverhalten der aufgezeichneten Verkehrsteilnehmer treffen zu können. Dabei mussten bestimmte Rahmenbedingungen, wie niedrige Kosten und lange Laufzeit eingehalten werden. Das gesamte System sollte bei jeder Witterung einsatzfähig sein und am Straßenrand einer maximal zweispurigen Strasse platziert werden. Die einzelnen Geräte werden an Straßenlaternen auf einer Höhe von sechs Meter befestigt.

Ein einzelnes Gerät besteht aus einer Rechnereinheit mit einem Quad-core Prozessor, welcher in der Lage ist, mehrere Prozesse parallel durchzuführen. Dies ist notwendig um eine beinahe Echtzeitauswertung gewährleisten zu können. Um die Verkehrsteilnehmer einzeln zu identifizieren, wird eine Kamera mit hoher Bildrate, jedoch geringer Auflösung, verwendet. Die Geräte werden mit einer externen Stromversorgung ausgestattet, welche die Einsatzfähigkeit über mehrere Tage hinweg gewährleistet. Ebenso ist es dadurch möglich sie an jeder beliebigen Straßenlaterne anzubringen und Messungen durchführen zu können. Damit sich die Geräte in der richtigen Position befinden, wird mithilfe eines WIFI-Adapters eine Verbindung zu einem Endgerät herstellt. Dabei muss vom Benutzer ein Hotspot errichtet werden, auf welchen sich das Gerät verbindet. Mithilfe einer Website kann im Anschluss das Gerät gesteuert und überwacht werden. Um das Gerät parallel zur Strasse ausrichten zu können, wird auf der Website ein Live-Stream der Kameraposition angezeigt. Dadurch kann die genaue Ausrichtung justiert und das Gerät im Anschluss gestartet werden.

Im Betriebsmodus beschäftigt sich der erste Kern permanent mit der Aufnahme von Videos, welche nach 15 Minuten abgespeichert werden. Ein weiterer Kern analysiert die aufgenommenen Videos, indem jeweils zwei aufeinanderfolgende Frames verglichen werden. Somit können Bewegungen erkannt und Bilder mit Bewegungen temporär gespeichert werden. Nachdem das Video abgearbeitet wurde, wird es vom temporären Speicher entfernt. Der dritte Kern verwendet die Bilder, auf welchen Bewegungen erkannt wurden und verknüpft diese logisch miteinander um einen Verkehrsteilnehmer identifizieren zu können. Daraus werden zu jedem Verkehrsteilnehmer Eigenschaften extrahiert und in einem Feature Vektor gespeichert. Zu den extrahierten Eigenschaften gehört unter anderem ein Zeitstempel, die Fahrtrichtung, ein Zähler für die Anzahl der Fahrzeuge pro Fahrtrichtung, eine Kategorisierung, sowie eine Geschwindigkeitsteilung. Dieser Feature Vektor wird später verwendet, um den Verkehrsfluss in einem begrenzten Gebiet rekonstruieren und daraus eine statistische Aussage treffen zu können. Der letzte Kern beschäftigt sich mit der Steuerung und Überwachung der anderen Kerne. Dies ist notwendig, um garantieren zu können, dass den Prozessen jederzeit genügend Ressourcen zur Verfügung stehen. Außerdem ermöglicht ein kleiner zeitlicher Puffer, dass nebenbei noch kleinere Aufgaben durchgeführt werden können. Dies sind beispielsweise das Monitoring der Rechnerreinheit sowie das Handeln des Webservers, damit unverzüglich auf Eingaben des Benutzers reagiert werden kann.

Nachdem das Gerät entwickelt und den ersten Tests standgehalten hatte, wurde ein erster Feldversuch mit mehreren Geräten in der Gemeinde Satteins im Vorarlberg durchgeführt. Dabei wurden sechs Geräte zur selben Zeit an verkehrsrelevanten Stellen angebracht und über eine Dauer von mehreren Tagen in Betrieb genommen. Aus den gewonnenen Daten geht hervor, dass zum einen die Anzahl Verkehrsteilnehmer an jedem Gerät gezählt und kategorisiert, zum anderen der Verkehrsfluss in der Gemeinde Satteins statistisch rekonstruiert und dargestellt wurde. Anhand von Testbildern, welche parallel aufgenommen wurden, konnte der nachgestellte Verkehrsfluss überprüft werden.

Mit den Geräten wäre es möglich eine Echtzeitauswertung zu realisieren, welche jederzeit online abgerufen werden kann. Dazu müssten diese ihre Daten aber entweder auf einen Online-Server transferieren oder mittels LoRa-WAN an einen Host übertragen. LoRa-WAN ist ein Long-Range Wirelessprotokoll, welches über mehrere Kilometer hinweg Daten versenden kann. Die zu übertragenden Daten dürfen dabei nicht zu gross sein, da LoRa-WAN nur eine geringe Übertragungsgeschwindigkeit bietet. Ebenfalls besteht die Möglichkeit, die Geräte mit zusätzlichen Sensoren zu erweitern, welche weitere verkehrsrelevante Daten wie Lautstärkemessungen, Wettersituation oder Schadstoffmessungen aufnehmen könnten.

3 Einleitung

3.1 Aufgabenstellung

Die Aufgabe des Projektes war es, ein Gerät zu entwickeln, welches autonom vorbeifahrende Verkehrsteilnehmer erkennt und im Anschluss diverse Operationen auf diese ausführt, um somit eine genaue Identifikation dieser Verkehrsteilnehmer zu erhalten. Das Gerät sollte zwei verschiedene Zähler beinhalten, welche sowohl links- als auch rechtsfahrende Fahrzeuge unabhängig voneinander zählen kann. Zudem sollte das Gerät die aufgenommenen Fahrzeuge je nach Grösse in verschiedene Kategorien unterteilen und ebenfalls eine grobe Abschätzung über die momentane Geschwindigkeit treffen können. Ein zusätzlicher Zeitstempel, welcher bei der Vorbeifahrt der einzelnen Fahrzeuge aufgenommen wurde, trägt alle gefundenen Indikatoren in eine Tabelle ein, damit diese für spätere Auswertungen verwendet werden können. Eine wichtige Voraussetzung war es, dass ein Gerät etwa eine Woche ohne Unterbrechung im Betrieb bleiben kann und zudem die eingesetzte Speicherkarte genügend Kapazität aufweist, damit diese im besagten Zeitraum nicht überfüllt wird. Ausserdem wurde eine möglichst einfache Montage der einzelnen Geräte angestrebt, wobei dennoch ein gewisses Monitoring möglich ist. Im besten Fall soll die Ausrichtung mit einem Mobilphone durchgeführt werden können und somit die momentane Ausrichtung ohne grossen Zeitaufwand kontrolliert und korrigiert werden können.

Um eine aussagekräftige Verkehrsverfolgung zu erhalten, werden die einzelnen Geräte zu einem kompletten System gekoppelt. So sollen die Fahrzeuge an mehreren Orten von den verschiedenen Geräten wiedererkannt und damit der Bewegungsablauf der einzelnen Verkehrsteilnehmer rekonstruiert werden. Durch strategisch günstige Positionen der Geräte und geeignete Algorithmen soll es möglich sein, dass nur ein Bruchteil aller aufgenommenen Fahrzeuge für die Identifikation eines bestimmten Teilnehmers in Betracht kommen kann. Dadurch wird eine genauere und schnellere Auswertung erreicht. Es muss nicht zwingend jeder Verkehrsteilnehmer erkannt werden, da es sich dabei lediglich um eine statistische Aussage handelt. Diese Entscheidung wurde so festgelegt, da nicht alle Verkehrsteilnehmer zu 100% verfolgt werden können. Probleme entstehen beispielsweise, wenn diese im Ort wohnen oder für längere Zeit dort verweilen.

Damit eine Verkehrsverfolgung gelingen kann, müssen neben den bereits erwähnten Indikatoren noch weitere Eigenschaften gefunden werden, um die Fahrzeuge genauer einteilen zu können. Welche zusätzlichen Indikatoren damit gemeint sind, sollte sich im Verlauf dieser Bachelorarbeit zeigen. Letztendlich soll die Auswertung der aufgenommenen Resultate noch visualisiert werden. Bestenfalls könnte dies mithilfe von Open Source Tools wie beispielsweise Open Street Map geschehen.

3.2 Wertschöpfung

Das System soll Verkehrsplaner zukünftig bei Entscheidungen unterstützen, damit diese die Fahrtwege von einzelnen Teilnehmern genauer verstehen und nachvollziehen können. Durch das Aufstellen von Hindernissen, wie zum Beispiel Baustellen, oder Geschwindigkeitsbegrenzungen ist eine Veränderung des Verkehrsflusses erkennbar. Nun soll durch den optimalen Einsatz von Verkehrssignalen der Verkehr so beeinflusst werden, dass er an ungewollten Gebieten, wie Wohnsiedlungen, vorbeigelenkt wird. Manuelle Verkehrszählungen können aufgrund der Konzentration derzählenden Personen nur begrenzte Zeit durchgeführt werden. Durch die Geräte hat man die Möglichkeit, das Geschehen im Straßenverkehr automatisch über einen längeren Zeitraum zu analysieren.

Ebenso soll dieses System zusätzlich von Gemeinden genutzt werden, damit diese gezielt in ihren Orten diverse Veränderungen durchführen können. Zu diesen Veränderungen können Verkehrsberuhigungen, Geschwindigkeitsanpassungen oder Umleitungen zählen. Um mit diesen Veränderungen langfristig Erfolg zu haben, ist es von Vorteil die Geschwindigkeit, die genaue Anzahl und die Art der Fahrzeuge zu kennen.

4 Fast and Curious

4.1 Anforderungen

An das Gerät und das gesamte System werden diverse Anforderungen gestellt, welche im Folgenden aufgezählt werden.

4.1.1 Robustheit

Das System muss eine sehr robuste Auswertung liefern, so dass es möglich ist, bei jeder Wettersituation dieselben Resultate zu erzielen. Im Allgemeinen bedeutet es, dass das System die Verkehrsteilnehmer eindeutig identifizieren können muss, egal ob die Verkehrsteilnehmer voller Schmutz, im Regen oder unter Sonneneinstrahlung sind. Das Gerät wird dazu in ein wasserdichtes Gehäuse gehüllt.

4.1.2 Laufzeit

Das Gerät arbeitet eigenständig und ist mit einer externen Spannungsquelle versehen, da nicht an jeder Strassenlaterne ein Anschluss vorhanden ist, um die Geräte dort zu laden. Die Laufzeit der Geräte sollte mehrere Tage betragen, da nur so gewährleistet ist, dass aussagekräftige Ergebnisse im Bezug auf den Verkehrsfluss erzielt werden können.

4.1.3 Verarbeitungszeit

Die Verarbeitung der gesammelten Daten sollte beinahe in Echtzeit geschehen, um eventuell die Ergebnisse online darstellen zu können, wenn das System zu einem späteren Zeitpunkt weiter ausgereift wird. Aufgrund dessen wird ein sehr hoher Anspruch an die verwendete Hardware gesetzt. Ebenso muss die Software so angepasst sein, dass auch sie die Daten in Echtzeit verarbeiten kann.

4.1.4 Kosten

Man möchte jeder Gemeinde die Möglichkeit geben mit diesem System zu arbeiten. Damit es sich jede auch wirklich leisten kann, müssen die einzelnen Geräte sehr preisgünstig hergestellt werden. Die Vorgabe für ein einzelnes Gerät lautet deshalb, einen Preis von 200 CHF nicht zu überschreiten. Folglich ist es dann möglich für 1'000 CHF fünf solcher Geräte anzuschaffen. Das System wird dabei nicht durch die Qualität der einzelnen Bauteile definiert, sondern durch den Preis und das Zusammenspiel zwischen günstigen Bauteilen und gut durchdachter Software. Dadurch soll versucht werden mit den preisgünstigen Bauteilen eine gute Qualität des gesamten Gerätes zu erreichen.

4.2 Umwelteinflüsse

Die Geräte müssen den unterschiedlichsten Umwelteinflüssen trotzen. Zu denen gehören unter anderem stark auftretende Winde und starke Regengüsse. Folglich muss das Gehäuse des Gerätes wasserdicht sein und die Software mit dem hin und her Schwanken der Strassenlaterne zureckkommen. Außerdem müssen die Geräte direkter Sonneneinstrahlung standhalten können. Sowohl der Prozessor, als auch andere elektronische Bauteile reagieren meist empfindlich auf Wärme, weshalb das Gehäuse gut durchlüftet werden muss. Wird das Gerät direkter Sonneneinstrahlung ausgesetzt, kann es notwendig sein dieses mittels aktivem Kühler zu temperieren.

5 Hardware

5.1 Komponenten

Bei den Komponenten musste darauf geachtet werden, dass diese den äusseren Einflüssen, sowie den spezifischen softwaretechnischen Daten genügen. Die einzelnen Elemente der Geräte bestehen aus Rechnereinheit, Messsensor, WIFI-Adapter, Gehäuse und einer Spannungsversorgung.

5.1.1 Rechnereinheit

An die Rechnereinheit wurden sehr grosse Anforderungen gestellt, da diese sowohl bei -10°C , als auch bei über 40°C die Auswertung korrekt durchführen muss. Ebenso muss die Rechnereinheit sehr viel Auswertungen direkt durchführen, damit so wenig wie möglich gespeichert wird. Der Rechnerkern des Gerätes besteht aus einem Allwinner H3, welcher einen Quad-core Cortex-A7 mit bis zu 1.2 GHz Taktrate besitzt. Dieser Rechnerkern ist mit am besten für das Gerät geeignet, da auf diesem mehrere Prozesse gleichzeitig ablaufen können. Dies ist nötig um die gewünschte Verarbeitungsgeschwindigkeit zu erreichen. Im Zuge dessen wurde ein vorgefertigtes Board im Gerät eingesetzt. Das eingesetzte Board ist ein NanoPi NEO von FriendlyARM [L1]. Der NanoPi NEO hat den oben genannten Quad-core Prozessor, mit dem die Aufgabe problemlos gelöst werden kann.

Dieses Board bietet ebenso den Vorteil, dass sich alles auf einer Fläche von 40x40 mm befindet. So kann das gesamte Gerät sehr kompakt gestaltet werden. Das Board wird mit einer externen Speicherplatte verwendet, auf welcher sich das Betriebssystem befindet. Im Falle von "Fast and Curious" ist es Ubuntu 16.04 (Long Time Supported).

Nachteil am NanoPi NEO sind zum einen die eher schlechte Dokumentation sowie das Vorhandensein von lediglich einem einzigen USB Anschluss. Aus diesem Grund musste beim Gerät ein USB Hub verwendet werden, um den Messsensor sowie den USB-WIFI-Adapter anzuschliessen. Nachfolgendes Bild (Abbildung 1) zeigt das Layout des NanoPi NEO.

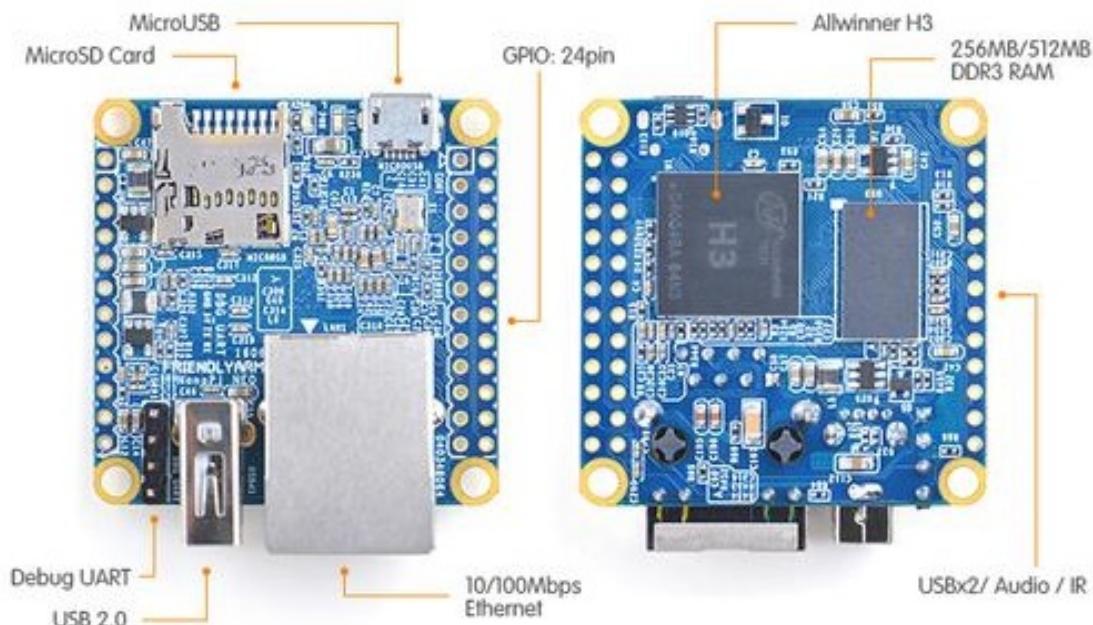


Abbildung 1: Layout des NanoPi NEO. [Q1]

5.1.2 Messsensor

Der Messsensor des Gerätes liefert die Daten der Verkehrsteilnehmer, mit denen dann die weitere Auswertung geschehen kann. Es müssen aus den Daten des Messsensors Kenngrößen extrahiert werden, welche die Verkehrsteilnehmer eindeutig identifizieren lässt. Es wurde auf eine Kamera zurückgegriffen, welche im Stande ist zwei Fahrspuren zu überdecken und mindestens drei Bilder pro vorbeifahrendem Verkehrsteilnehmer aufzunehmen.

Mithilfe nachfolgender Berechnung wurde die geeignete Kamera, sowie das passende Objektiv dazu gefunden. Die Skizze (Abbildung 2) zeigt die schematische Darstellung eines Aufstellpunktes des Gerätes. In Berechnung (1) wird der Öffnungswinkel des Objektivs bei einer Aufnahmgeschwindigkeit von 30 FPS berechnet.

$$\begin{aligned} l_{1,30} &= \frac{22.22 \text{ m/s}}{30 \text{ FPS}} = 0.75 \text{ m} \\ l_{4,30} &= 4 \cdot l_{1,30} = 3 \text{ m} \\ \alpha_{30} &= 2 \cdot \arctan(l_{4,30}) \approx 150^\circ \end{aligned} \quad (1)$$

In der nachstehenden Berechnung (2) wurde der selbe Winkel, wie in Berechnung (1), jedoch bei einer Aufnahmgeschwindigkeit von 60 FPS berechnet.

$$\begin{aligned} l_{1,60} &= \frac{22.22 \text{ m/s}}{60 \text{ FPS}} = 0.37 \text{ m} \\ l_{4,60} &= 4 \cdot l_{1,60} \approx 1.5 \text{ m} \\ \alpha_{60} &= 2 \cdot \arctan(l_{4,60}) \approx 120^\circ \end{aligned} \quad (2)$$

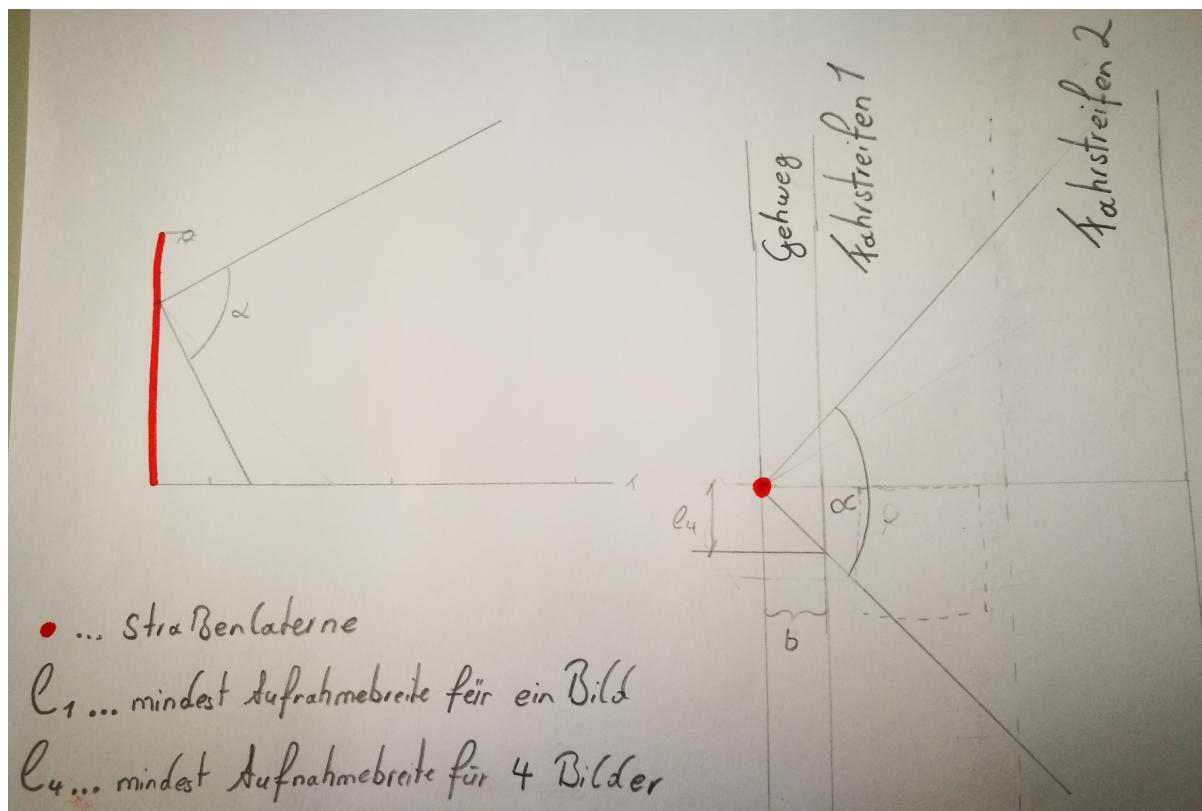


Abbildung 2: Skizze zur Berechnung der Kameradaten.

Durch diese Berechnung wurde eine Kamera eingesetzt, welche mit einer Framerate von 30 FPS Bilder aufnimmt.

Im produzierten Prototypen wurde eine "ELP 1080p Full HD H.264" Kamera mit 3.6 mm Objektiv verwendet. Diese Kamera hat den Vorteil, dass die gewünschten Treiber auf dem Betriebssystem schon installiert sind und die Kamera sehr preisgünstig zu erwerben ist. Diese Kamera besitzt bei einer FPS Anzahl von 30 eine Auflösung von 640x480 Pixel und ein 3.6 mm Objektiv. Jedoch nimmt die Kamera erst unter 25 FPS die Bilder ohne Fehler auf. Diese Anzahl an Pixel reicht aus, um die nötigen Daten der Verkehrsteilnehmer zu extrahieren. [L2]

5.1.3 WIFI-Adapter

Um ein Gerät steuern und überwachen zu können, wird eine kabellose Verbindung benötigt. Dies ist am einfachsten über eine WIFI-Verbindung zu erreichen. Da das NanoPi NEO keine internen Funksender eingebaut hat, wird ein WIFI-Adapter benötigt. Dazu kann jeder WIFI-Adapter verwendet werden, welcher das Linux System unterstützt, da somit bereits ein Treiber vorhanden ist. Werden für die Geräte unterschiedliche WIFI-Adapter eingesetzt, so muss nach der Anleitung für die Installation der Geräte vorgegangen und die Namen der Adapter geändert werden.

Die einzige Anforderung an den WIFI-Adapter ist, dass dieser über USB angeschlossen werden kann. Die Geräte, welche produziert wurden, haben unterschiedliche WIFI-Adapter eingebaut und es funktioniert mit allen WIFI-Adaptoren problemlos. Die Verbindung geschieht über einen Hotspot vom Endgerät zu NanoPi NEO.

5.1.4 Spannungsversorgung

Die Spannungsversorgung der Geräte wurde mittels Autobatterie und Spannungswandler realisiert. Das gesamte Gerät benötigt bei Volllast ca. 400 mA Strom. Die verwendeten Autobatterien haben eine Kapazität von 40 Ah und eine Spannung von 12 V.

$$\frac{40000 \text{ mAh} \cdot 12 \text{ V}}{400 \text{ mAh} \cdot 5 \text{ V}} = 240 \text{ h} = 10 \text{ d} \quad (3)$$

Wie aus der Berechnung (3) entnommen werden kann, beläuft sich die Betriebszeit mit einer Akku Ladung theoretisch auf etwa zehn Tage. Wie erwartet ist die tatsächliche Laufzeit der Geräte kleiner als die Berechnete, da aufgrund von Temperaturschwankungen die Leistung der Batterie nachlässt. Ebenso dürfen die Batterien bis zu einer minimalen Entladungsspannung von 1.75 V [L3] entladen werden. Die Geräte hingegen benötigen 5 V um die Daten sammeln zu können. Die tatsächliche Laufzeit beträgt etwa vier Tage.

Im Falle einer Fertigung des Gerätes muss eine neue Spannungsquelle gefunden werden, da Autobatterien beinahe immer bleihaltig sind. Es könnten später Lithium-Ionen-Akkus verwendet werden, welche kompakter und leistungsfähiger sind, jedoch waren diese für die Prototypen zu teuer und nicht rentabel. Um die Geräte längere Zeit betreiben zu können, wäre es möglich, die Akkumulatoren während des Tages mithilfe von Photovoltaikzellen aufzuladen.

5.1.5 Gehäuse

Das Gehäuse der Prototypen besteht aus einer Kunststoffbox, in welcher sich früher Lebensmittel befanden. In diesem Gehäuse wurden alle Komponenten, welche oben genannt wurden, mittels Klettverschluss angebracht, um die einzelnen Komponenten ohne separate Gehäuse weiter zu verwenden. Dies ermöglichte eine leichtere Bedienung bei der Programmierung der Software. Am Gehäuse wurden zwei Gewindestangen angebracht, um die Ausrichtung des Gerätes einzustellen zu können. Weiterhin wurde im Deckel der Kunststoffboxen Schlitze geschnitten, die die Abwärme der Komponenten besser nach aussen befördern. Die Gehäuse werden an den Strassenlaternen mit grossen Kabelbindern befestigt.

Wichtig: In den Gehäusen befindet sich keine Batterie.

Aus Sicherheitsgründen wurde ein zweipoliges Kabel von den Geräten nach unten gezogen und an den Polen der Batterie befestigt. Das folgende Bild (Abbildung 3) dient dabei zur Illustration des leeren Gehäuses.



Abbildung 3: Foto eines leeren Gehäuses.

5.2 Gesamtsystem

Das Gesamtsystem wurde, wie bereits erwähnt, so günstig wie möglich aufgebaut. Wenn die einzelnen Komponenten bei den preiswertesten Lieferanten bezogen werden, dann kann das gesamte Gerät unter 150 CHF (exkl. Gehäuse) hergestellt werden. Die nachfolgende Tabelle (Tabelle 1) zeigt eine Auflistung der Kosten eines Gerätes. Zu den Kleinteilen zählen beispielsweise Schrauben oder Kabel.

Komponente	Preis
NanoPi NEO	10 CHF
Full HD Kamera	50 CHF
Autobatterie	50 CHF
WIFI-Adapter	10 CHF
USB-Hub	5 CHF
Spannungswandler	10 CHF
Kleinteile	15 CHF
Gesamt	150 CHF

Tabelle 1: Kostenaufstellung.

Auf den folgenden Bildern wird das Gesamtsystem dargestellt. Dabei ist im ersten Bild (Abbildung 4) das System in Grossansicht zu sehen, während im zweiten Bild (Abbildung 5) das System im Einsatz, hängend an einer Strassenlaterne, zu betrachten ist.

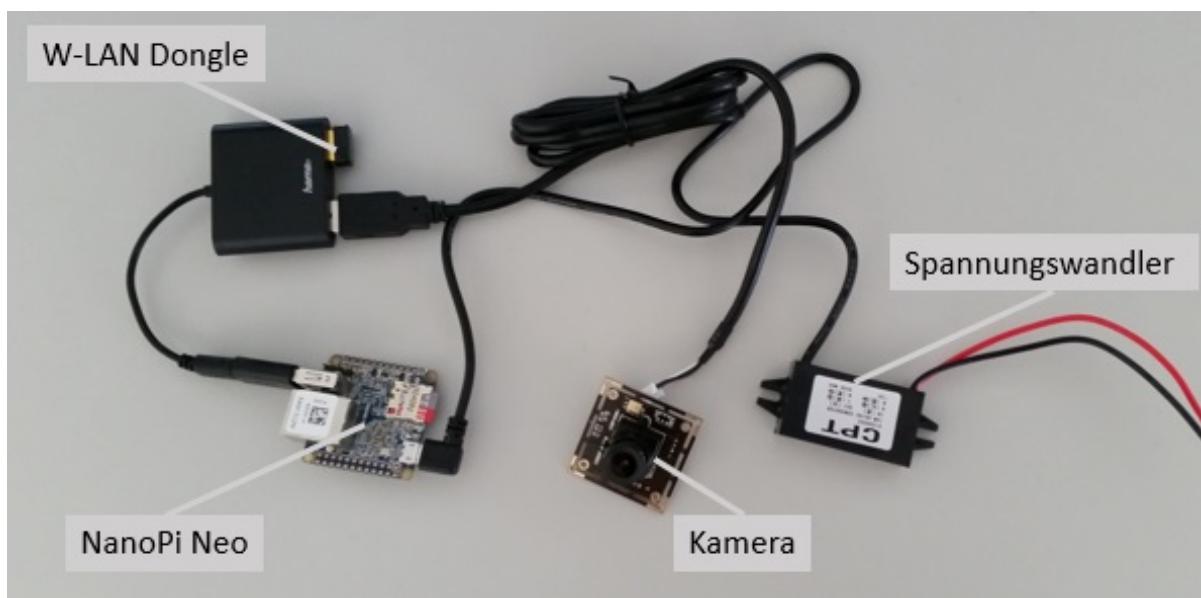


Abbildung 4: Gesamtsystem.



Abbildung 5: Einsatzfähiges Gesamtsystem.

6 Software

6.1 Verwendete Tools und Libraries

6.1.1 Mjpg-Streamer

Beim Mjpg-Streamer handelt es sich um ein Werkzeug für die Kommandozeile, welches ermöglicht Videos von einer Videoquelle aufzunehmen und diese entweder in einer Datei abzuspeichern oder es auf einem Webserver zu streamen. Beim Abspeichern eines Videos konnte etwa eine Bildrate von 15 Bilder pro Sekunde erreicht werden, bevor einige Bilder verloren gingen.

Grundvoraussetzung für den Mjpg-Streamer ist eine Kamera, welche korrekt eingerichtet wurde. Bei den meisten Modellen muss deshalb der Treiber für “Video 4 Linux 2” zuerst installiert werden. Die Installation vom Mjpg-Streamer funktioniert nicht über die offiziellen Paketquellen. Aus diesem Grund müssen zuerst einige andere Pakete der offiziellen Paketquellen installiert werden, bevor der aktuelle Quelltext der eigentlichen Software heruntergeladen und installiert werden kann. Die Verwendung erfolgt relativ simpel, indem zuerst das gewünschte Plugin und danach dessen Parameter übergeben wird. Dies wird sowohl für den Eingang, als auch den Ausgang des Videos benötigt. Einige Parameter für den Eingang der Kamera sind neben dem Plugin auch die Auflösung, der Gerätename, die Bildwiederholrate und die Bildqualität. Bei den Parametern für die Ausgabe handelt es sich um den Speicherort, das Bildintervall oder auch den Port, falls es sich beim Ausgang um einen Webserver handelt. [L4]

6.1.2 Avconv

Avconv ist ein sehr schneller Audio- und Videokonverter, welcher dazu verwendet werden kann, um Videos mit Bild und Ton aufzunehmen und diese in einer Datei abzuspeichern. Auch dieses Tool kann von der Kommandozeile aus gesteuert werden und liefert viele Möglichkeiten um Konvertierungen von Bild- und Audiomaterial durchzuführen. Es kann ebenfalls dazu genutzt werden, Videomaterial in andere Datentypen zu konvertieren oder aber Videomaterial schneller beziehungsweise langsamer laufen zu lassen. Avconv bietet beispielsweise die Möglichkeit eine der beiden Spuren auszuschalten, wenn diese nicht benötigt werden. Es können viele Bearbeitungsmöglichkeiten, wie das Schneiden eines Videos in eine gewünschte Länge oder das Verbinden von zwei einzelnen Videos zu einem längeren Video, durchgeführt werden. Ebenso können mehrere Eingänge und Ausgänge angegeben werden, womit es die Möglichkeit bietet, eine eigene Bild- und Audiospur zu einem Video zu migrieren und es gleichzeitig in mehrere Videoformate abzuspeichern. Anders als beim Mjpg-Streamer kann Avconv direkt von den offiziellen Paketquellen heruntergeladen und installiert werden. Die Verwendung dieses Tools ist komplizierter als beim Mjpg-Streamer, da es viel mehr Parameter beinhaltet. Bei der Videoaufnahme können beispielsweise Parameter wie Auflösung, Eingangsvideogerät, Aufnahmezeit, Bildwiederholungsrate, Bitrate oder Bildqualität angegeben werden. [L5]

6.1.3 Apache

Unter Apache versteht man bei Linux den “Apache HTTP Server“. Dies ist ein sehr bekannter Webserver, welcher meist mit Skriptsprachen, wie PHP, oder Datenbanken, wie MySQL, verwendet wird, um Webseiten zu hosten. Die Installation des “Apache HTTP Servers“ unter Linux erfolgt lediglich über einen einfachen Befehl und wird dann von den offiziellen Paketquellen heruntergeladen und installiert. Nach der Installation befindet sich der HTTP Server ohne Konfiguration im Autostart und kann nach einem Neustarten der Linux Plattform direkt verwendet werden. Um eigene Internetseiten auf dem Apache zugänglich zu machen, muss die gewünschte Seite als HTML oder PHP in den richtigen Ordner geschoben werden und kann daraufhin über die IP erreicht werden. [L6]

6.1.4 OpenCV

OpenCV ist die Abkürzung für "Open Computer Vision" und ist eine Bibliothek, bei der der Code offen zugänglich und nutzbar ist. Diese Softwarebibliothek ist für Bild- und Videoverarbeitung sowie "Machine Learning" und kann in sämtlichen kommerziellen Produkten eingesetzt werden. Die Bibliothek besteht aus mehr als 2'500 optimierten Algorithmen und beinhaltet sämtliche gängige Methoden für Bildverarbeitung und "Machine Learning". Zu den Algorithmen gehören beispielsweise das Detektieren und Erkennen von Objekten, Klassifizieren von menschlichen Aktionen, Aufnehmen von Kamerabewegungen, Erkennen von 3D-Objekten und das Produzieren von 3D-Objekten aus einem normalen Kamerabild. Ebenso gehören dazu noch das Zusammenführen von Bildern um hochauflösende Bilder ganzer Szenen zu erschaffen, um ähnliche Bilder in einem Set von Bildern zu finden, rote Augen von Bildern zu entfernen, Verfolgen von Augen oder Gesichtern, "Augmented Reality" und vieles mehr. OpenCV bietet Schnittstellen für die Programmiersprachen "C++", "C", "Python", "Java" und "MATLAB" und unterstützt die gängigen Plattformen Windows, Android, Mac Os und Linux. Es wird stetig weiterentwickelt und kann ohne Probleme in ein Projekt eingebunden werden. Die aktuelle Version ist OpenCV 3.2 und besteht aus Hauptmodulen und Extramodulen. Die normale Installation beinhaltet nur die Hauptmodule, welche jedoch für den normalen Gebrauch ausreichen. Um die Extramodule verwenden zu können, müssen diese zusätzlich heruntergeladen und in den OpenCV Ordner verschoben werden. Für die Installation von OpenCV werden einige offizielle Softwarepakete benötigt, welche man auch vorher herunterladen und installieren muss. Die Installation der neusten Software von OpenCV selber funktioniert nicht über die offiziellen Paketquellen, weshalb die Software manuell heruntergeladen werden muss. [L7]

6.2 Architektur

Beim NanoPi Neo handelt es sich, wie bereits erwähnt, um einen Prozessor mit vier Kernen. Dadurch hat man die Möglichkeit vier Prozesse parallel ablaufen zu lassen. Aus diesem Grund wurde in der untenstehenden Abbildung (Abbildung 6) der Kern symbolisch in vier Abschnitte geteilt, welche die einzelnen Prozesse symbolisieren sollen. Nachfolgend werden die Aufgaben der einzelnen Kerne gezeigt, die schlussendlich das gleichzeitige Aufnehmen und Verarbeiten der Verkehrsteilnehmer ermöglichen.

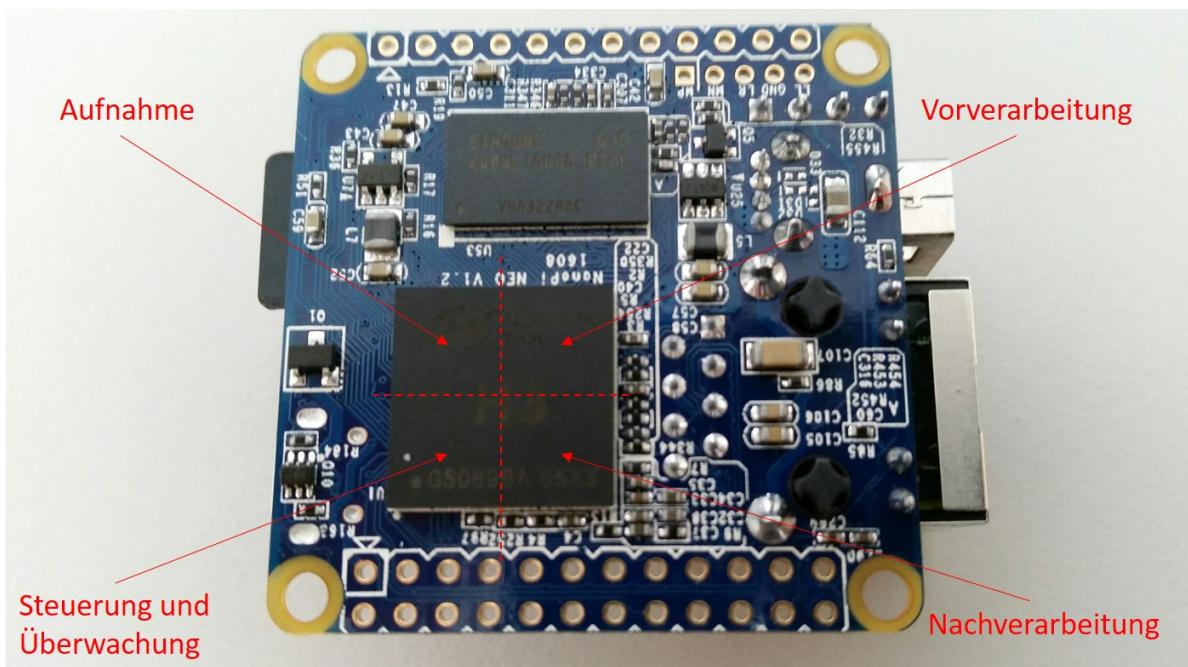


Abbildung 6: Aufteilung der vier Kerne.

6.2.1 Aufnahme

Der erste Prozess beinhaltet die Aufnahme von Videos mit der Kamera. Die Videos werden mit 25 Bildern pro Sekunde und einer Auflösung von 640x480 Pixeln aufgenommen und auf der Speicherkarte abgelegt. Dabei muss ein Video immer eine Länge von 15 Minuten erzielen, bevor es abgespeichert wird. Die Zeitspanne von 15 Minuten wurde gewählt, damit nach Abschluss dieser Zeit der nächste Prozess sich schon um die Verarbeitung kümmern kann. Währenddessen kann sich der Aufnahmeprozess bereits mit dem nächsten Video beschäftigen. So erreicht man die parallelen Abläufe. Eine geringere Aufnahmezeit würde einen größeren Fehler verursachen, da jedes abspeichern und neu-starten eines Videos ein bis zwei Sekunden dauert, in welcher die Verkehrsteilnehmer nicht erkannt werden können. Die Videos sind im "AVI" Format abgespeichert und beinhalten im Durchschnitt etwa 22'500 Einzelbilder, welche dann vom nachfolgenden Prozess analysiert werden müssen. Mithilfe von Avconv werden diese Videos aufgenommen und mittels Zeitstempel, welcher Datum und Uhrzeit des Aufnahmestarts beinhaltet, im zugehörigen Ordner abgespeichert. Der Zeitstempel wird benötigt um später den Zeitpunkt eines vorbeifahrenden Verkehrsteilnehmers zu errechnen und um zu wissen, welche Videos der Reihe nach weiterverarbeitet werden müssen. Der Prozess der Videoaufnahme ist spezifisch auf den ersten der vier Prozessoren zugeteilt. Während der Aufnahmephase befindet sich dieser Kern stetig bei einer Auslastung von 95 - 100%.

Das unten dargestellte Flussdiagramm (Abbildung 7) dient zur Veranschaulichung des Ablaufes der Videoaufnahme.



Abbildung 7: Flussdiagramm der Videoaufnahme.

6.2.2 Vorverarbeitung

Sobald sich zwei Videos im dafür vorgesehenen Ordner befinden, kann davon ausgegangen werden, dass die Aufnahme des ersten Videos abgeschlossen ist. Zu diesem Zeitpunkt beginnt die Vorverarbeitung damit, die Bilder zu analysieren und diese, sobald eine Bewegung auf dem Bild erkannt wurde, in einen separaten Ordner abzuspeichern, welcher extra für diese Frames genutzt wird. Da die Videoaufnahme innerhalb von 15 Minuten jeweils 22'500 Bilder erzeugt, muss die Vorverarbeitung etwa gleich viel Bilder in der selben Zeit auswerten können. Aus diesem Grund wurde das Programm für diesen Prozess soweit optimiert, dass nur sehr wenig einzelne Schritte durchgeführt werden müssen. Diese Analyse wird mithilfe der Bildverarbeitungsbibliothek OpenCV durchgeführt. Dabei kann ein Video angegeben werden, woraufhin von diesem das nächste nicht genutzte Frame in einer temporären Variablen gespeichert wird. Mit diesen Variablen können im Anschluss weitere Berechnungen durchgeführt werden. Die Vorverarbeitung nimmt jeweils zwei aufeinanderfolgende Frames und subtrahiert bei diesen Bildern jedes einzelne Pixel an derselben Stelle voneinander ab. Falls es keine Bewegung beim spezifischen Pixel gab, so ergibt die Subtraktion der Pixel an dieser Stelle den Wert Null. Der Pixel erscheint auf diesem Bild dann schwarz. Falls sich jedoch Bewegungen abgespielt haben, ergibt diese Berechnung einen höheren Wert und es erscheint auf dem Bild heller bzw. farbig. Dies wird für jedes der 640x480 Pixel in einem Bild durchgeführt. Daraufhin werden im nächsten Schritt die erhaltenen Differenzen addiert. Folglich erhält man eine Zahl, welche die Gesamtsumme aller Differenzen dieser 307'200 einzelnen Pixel repräsentiert. Diese Zahl ist schlussendlich ausschlaggebend für eine allfällige Bewegung zwischen diesen beiden Bildern. Die nachfolgende Abbildung (Abbildung 27) zeigt zwei aufeinanderfolgende Frames.



(a) Frame1

(b) Frame2

Abbildung 8: Zwei aufeinanderfolgende Frames.

Anschliessend wurde die oben erwähnte Berechnung mit diesen beiden Frames durchgeführt, wodurch Bewegungen zwischen den Bildern heller bis farbig erscheinen. Das Ergebnis der Subtraktion ist im folgenden Bild (Abbildung 9) ersichtlich.



Abbildung 9: Differenzbild der aufeinanderfolgenden Frames.

Damit ein allfälliges Rauschen oder ein vorbeilaufender Fußgänger nicht erfasst wird, wurde ein gewisser Schwellwert angegeben. Sobald die Summe der Pixel den Schwellwert überschreitet, wird das Bild temporär abgespeichert. Damit die Bilder im nächsten Schritt logisch verknüpft werden können, muss im Speichername eine Indexierung angegeben werden. Diese Indexierung ist die Zahl des Frames, welches vom Video stammt und liegt aus diesem Grund zwischen 0 und 22'499. Falls keine Speicherung stattgefunden hat wird das nächste Bild genommen und der Vorgang beginnt erneut. Mit dieser Methode werden etwa 25 Bilder pro Sekunde verarbeitet. Da jedoch für das Abspeichern der Bilder mehr Zeit benötigt wird als für das Verwerfen, können bei höherem Verkehrsaufkommen nur etwa 20 Einzelbilder verarbeitet werden. Nachdem ein komplettes Video abgearbeitet wurde, wird es aus Platzgründen und weil es für den weiteren Verlauf der Auswertung nicht mehr benötigt wird gelöscht. Dies ist die letzte Aufgabe der Vorbearbeitung. Von den 22'500 Einzelbildern, die von diesem Prozess bearbeitet wurden, sind zu diesem Zeitpunkt, je nach Verkehrsaufkommen, noch etwa 500 - 1'500

Bilder im Zwischenspeicher. Sie werden dann vom nächsten Prozess, der Nachverarbeitung, weiterbearbeitet. Da die Vorverarbeitung sehr viel Leistung erbringen muss, wurden ihm die Prozessorkerne zwei und vier zugeteilt. Die Steuerung und Überwachung, welche fix auf dem vierten Prozessorkern liegt, benötigt nur eine geringe Auslastung. Aufgrund dessen kann der Prozess der Vorverarbeitung die zwei Kerne beinahe voll auslasten, um somit die vielen Einzelbilder möglichst effizient verarbeiten zu können.

Auf dem unten dargestellten Bild (Abbildung 10) ist das Flussdiagramm des geschilderten Algorithmus visualisiert.

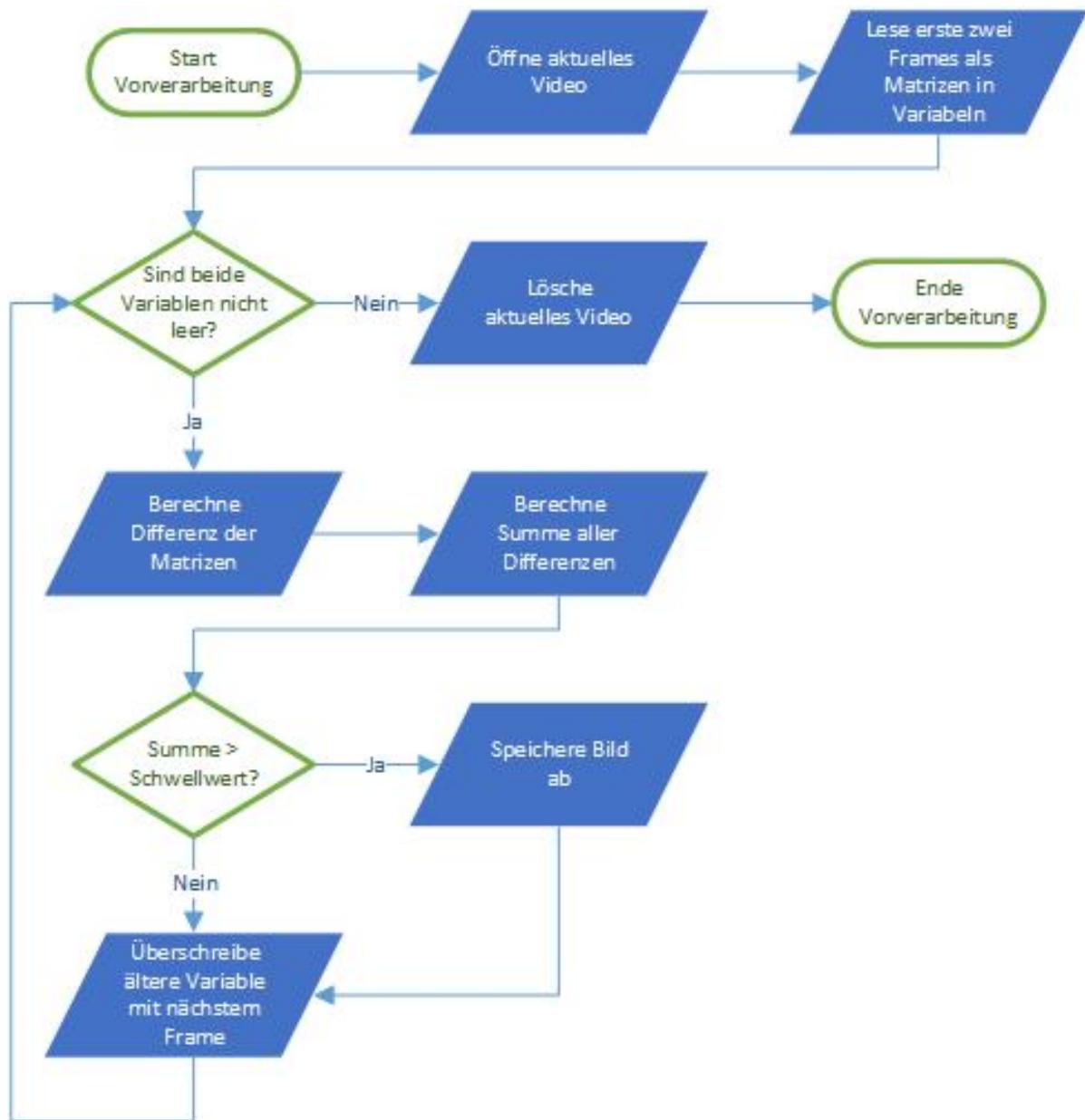


Abbildung 10: Flussdiagramm der Vorverarbeitung.

6.2.3 Nachverarbeitung

Nachdem die Vorverarbeitung ein Video analysiert und die relevanten Frames abgespeichert hat, kann die Nachverarbeitung beginnen. Auch hier wird der Prozess erst gestartet, wenn sich zwei unterschiedliche Ordner mit Frames im zugehörigen Ordner befinden. Somit soll gewährleistet sein, dass die Vorverarbeitung mit dem Video fertig ist, bevor die Nachverarbeitung mit dem Prozess beginnt. Das Endprodukt der Nachbearbeitung ist ein Feature Vektor. Dies ist eine Tabelle mit je einem Eintrag pro Verkehrsteilnehmer, welche mithilfe eines Tabellenkalkulationsprogramm geöffnet und bearbeitet werden kann. Dabei werden möglichst viele Indikatoren pro Verkehrsteilnehmer aufgenommen und abgespeichert. Je mehr Features pro Fahrzeug gefunden werden können, desto besser kann dieses später wiedererkannt werden und die eigentliche Verkehrsverfolgung durchgeführt werden. Nachfolgende Tabelle (Tabelle 2) zeigt ein Beispiel eines Feature Vektors.

I	Timestamp	RP	D	DC	CropN	BlurN	PicN	RX	RY	RW	RH
1	18.06.17 22:55:12	5	R	1	crop001.tif	blur001.tif	pic001.tif	95	141	377	190
2	18.06.17 22:57:19	8	R	2	crop002.tif	blur002.tif	pic002.tif	146	118	387	196
3	18.06.17 22:57:37	7	L	1	crop003.tif	blur003.tif	pic003.tif	138	145	373	251
4	18.06.17 22:58:10	7	R	3	crop004.tif	blur004.tif	pic004.tif	152	112	345	165
5	18.06.17 22:59:02	6	L	2	crop005.tif	blur005.tif	pic005.tif	115	126	415	296
6	18.06.17 22:59:23	5	R	4	crop006.tif	blur006.tif	pic006.tif	145	126	364	194
7	18.06.17 22:59:56	5	R	5	crop007.tif	blur007.tif	pic007.tif	113	100	453	311
8	18.06.17 22:59:59	9	R	6	crop008.tif	blur008.tif	pic008.tif	165	138	358	182
9	18.06.17 23:00:09	9	L	3	crop009.tif	blur009.tif	pic009.tif	148	160	367	212
10	18.06.17 23:00:18	7	R	7	crop010.tif	blur010.tif	pic010.tif	147	127	371	198
11	18.06.17 23:00:37	4	R	8	crop011.tif	blur011.tif	pic011.tif	155	128	341	189

Tabelle 2: Feature Vektor.

Der Feature Vektor zeigt den Index anhand einer fortlaufenden Nummer und den Zeitstempel, an welchem sich der Verkehrsteilnehmer in etwa in der Mitte des Kamerabildes befand. Ebenso enthält er die Information wie viele aufeinanderfolgende Bilder zu diesem Ergebnis geführt haben, in welche Richtung der entsprechende Verkehrsteilnehmer unterwegs war und einen Zähler der sowohl für links- als auch rechtsfahrende Fahrzeuge stetig erhöht wird. Daneben werden im Moment noch drei Bilder abgespeichert, welche für weitere Berechnungen von Features benutzt werden können. Die letzten vier Spalten zeigen die Position des Verkehrsteilnehmers im Originalbild.

Die Nachverarbeitung kümmert sich in erster Linie darum, den Feature Vektor zu öffnen, falls dieser vorhanden ist, um daraus relevante Informationen zu entnehmen. Ist der Feature Vektor nicht vorhanden, so wird ein neuer generiert. Zu den relevanten Informationen gehört ein Index. Dieser ist notwendig, damit im Feature Vektor kein Index doppelt vorkommt und somit bereits vorhandene Bilder überschrieben werden. Zudem wird die Anzahl an Verkehrsteilnehmer, welche bis zu diesem Zeitpunkt nach links bzw. rechts gefahren sind, in einer Variablen gespeichert. Diese beiden Zähler können weiter erhöht werden, wenn neue Verkehrsteilnehmer erkannt werden. Dies ist erforderlich, da sich dieses Programm nach Abarbeitung der Frames beendet und somit alle bestehenden Informationen verliert. Anschliessend wird die Anzahl der relevanten Bilder ermittelt, welche zu einem Verkehrsteilnehmer gehören. Dies passiert, indem die Frames genommen und deren Indexierungen betrachtet werden. Falls die Indexierungen der Frames fortlaufend sind, so muss es sich zwangsläufig um denselben Verkehrsteilnehmer handeln. Falls der Abstand zwischen zwei Frames einen gewissen Wert überschreitet, hat die Kamera für kurze Zeit keine Bewegung erkannt. Folglich muss es sich dann um einen neuen Verkehrsteilnehmer handeln. Sobald die Zahl der relevanten Bilder bekannt ist, werden diese logisch miteinander verknüpft.

Aus diesen etwa drei bis zehn Bildern, je nach Geschwindigkeit des Fahrzeugs, wird nun mit der Bildverarbeitungsbibliothek OpenCV ein “Blob Detection“ mit anschliessendem “Blob Tracking“ durchgeführt. Zuerst werden Bewegungen zwischen zwei aufeinanderfolgenden Bildern ermittelt und dann um die Pixel in der unmittelbaren Umgebung eine konvexe Hülle, die auch Blob genannt wird, gebildet. Auf den fortlaufenden Bildern wird sich diese konvexe Hülle immer weiter zum Rand bewegen. Im Programm selbst wurde festgelegt, welche Voraussetzungen erfüllt werden müssen, damit dieser Blob als Verkehrsteilnehmer gezählt wird. Dazu gehören die Fläche der Hülle, das Verhältnis zwischen Länge und Breite, die minimale Breite und Länge und die minimale Länge der Diagonale. Falls es sich beim Blob um einen Verkehrsteilnehmer handelt, muss noch ermittelt werden, zu welchem Zeitpunkt dieser die Mitte passiert und somit die Bilder generiert werden müssen. Dies wird mithilfe einer imaginären Linie durchgeführt, welche sich in der Mitte des Bildes befindet. Da die Position des Blobs von Anfang an bekannt ist, kann auch ermittelt werden, ob der Verkehrsteilnehmer von links nach rechts oder von rechts nach links gefahren ist.

In nachfolgender Abbildung (Abbildung 11) ist ein von rechts nach links fahrendes Fahrzeug dargestellt, welches von der Kamera erfasst wurde. Dabei wurde das Fahrzeug auf drei Frames als Ganzes abgebildet. Da dieses Auto alle festgelegten Parameter erfüllt, wird es als Verkehrsteilnehmer erkannt und eine konvexe Hülle gebildet. Die konvexe Hülle wird immer um zwei nachfolgende Frames gebildet, weshalb diese bei sämtlichen Bildern länger als der entsprechende Verkehrsteilnehmer ist. Im ersten und zweiten Frame befindet sich die Mitte der konvexen Hülle und damit auch die Mitte des gelben Quadrats jeweils auf der rechten Seite der Linie. Zu diesem Zeitpunkt wurde noch keine Zählung durchgeführt. Erst beim dritten Frame hat die Mitte des Blobs die rote Linie überquert. Dadurch wurde die Funktion zum Abspeichern eines Fahrzeugs und generieren einer neuen Zeile im Feature Vektor ausgelöst.

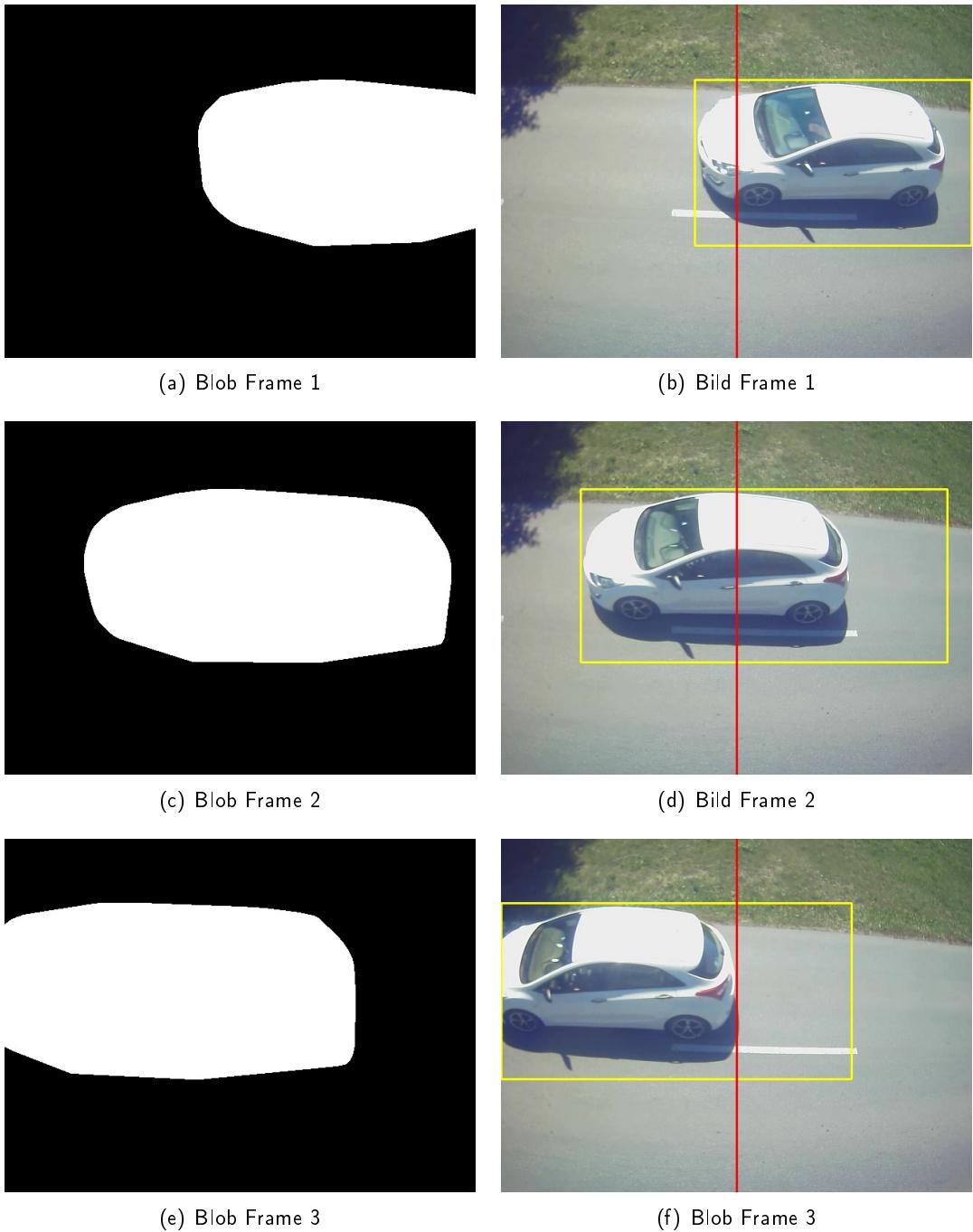


Abbildung 11: Vehicle Tracking mithilfe von Blobs.

Die drei vorher erwähnten Bilder werden in den entsprechenden Ordnern im Speicher abgelegt und alle relevanten Informationen an den Filehandler übergeben. Dieser öffnet die Datei des Feature Vektors und schreibt alle Informationen inklusive berechnetem Zeitstempel in eine neue Zeile. Nach diesem Vorgang ist der Verkehrsteilnehmer erfolgreich aufgenommen und abgeschlossen. Der nächste Teilnehmer kann nun ermittelt werden. Wenn sich keine Bilder mehr im Frames-Ordner befinden, beginnt die letzte Aufgabe dieses Programms. Der bereits bearbeitete Ordner wird gelöscht, da auch dieser für die weitere Auswertung nicht mehr benötigt wird. Der Nachbearbeitung wurde fix der dritte Prozessorkern zugeteilt. Bei hohem Verkehrsaufkommen benötigt diese auch die vollen 100% dieses Kerns.

Diese Software basiert auf dem Grundgerüst von C. Dahms "OpenCV_3_Car_Counting_Cpp"-Projekt [L8] und wurde an die Aufgaben von "Fast and Curious" angepasst, verändert und erweitert.

Der oben beschriebene Algorithmus ist auf nachfolgender Abbildung (Abbildung 12) aufgezeigt.

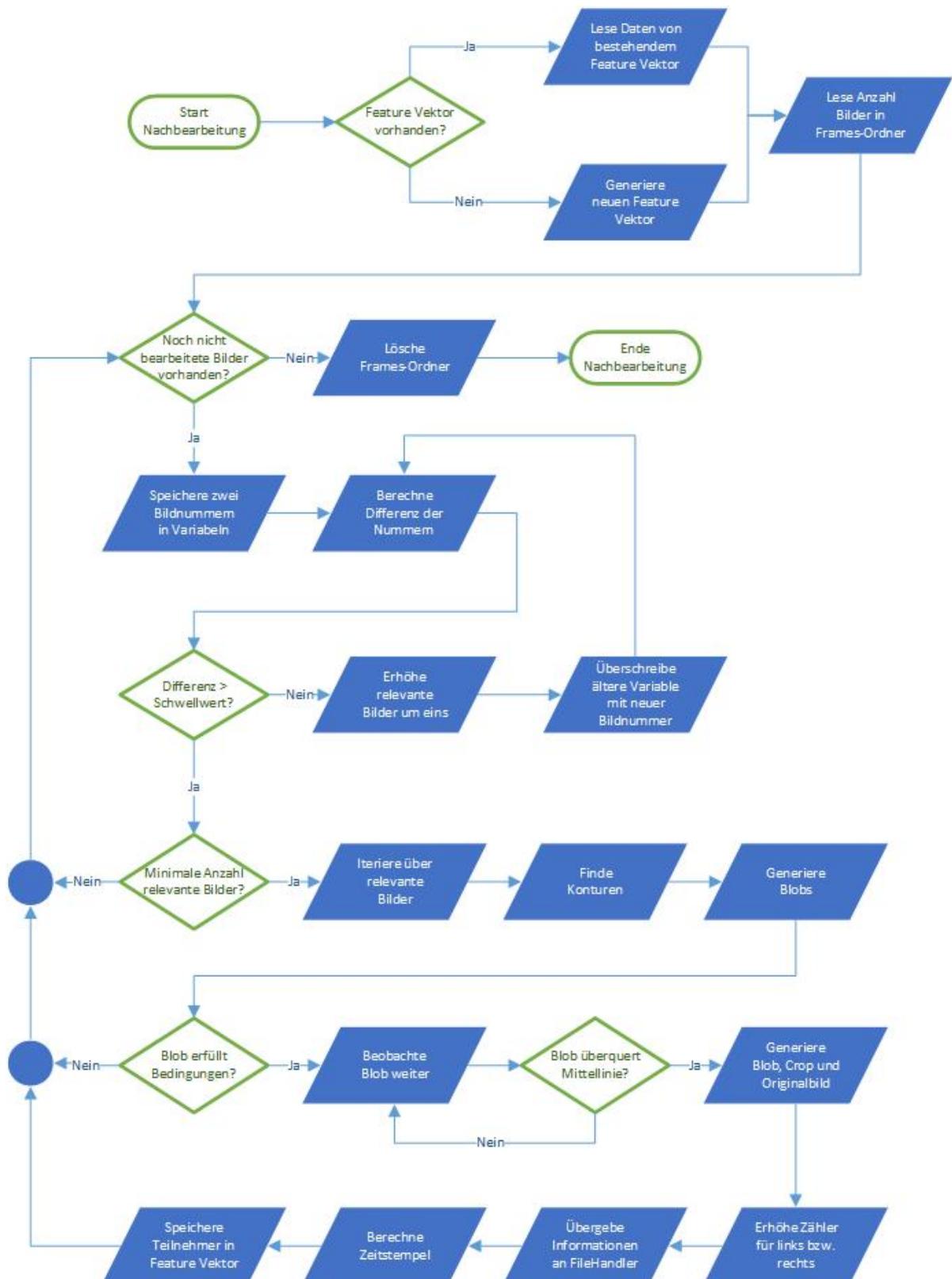


Abbildung 12: Flussdiagramm der Nachverarbeitung.

6.2.4 Steuerung und Überwachung

Vorhin wurde bereits erwähnt, dass sich die Prozesse selber beenden, sobald sie ihre Arbeit erledigt haben. Folglich wird ein Prozess benötigt, der sich hauptsächlich um die gesamte Koordination kümmert. Videos werden nur zwischen morgens um 05:00 Uhr und abends um 21:00 Uhr aufgenommen. Dies ist notwendig um sicherstellen zu können, dass alle Videos bis zum nächsten Morgen bearbeitet wurden. Dadurch soll ein Speicherplatzproblem verhindert werden, falls die Vorverarbeitung aufgrund eines hohen Verkehrsaufkommens die Videos langsamer prozessiert als diese aufgenommen werden.

Die Steuerung und Überwachung ist so konzipiert, dass sie ständig eine Schleife von Befehlen durchläuft. Die erste Aufgabe besteht darin den aktuellen Zeitpunkt zu kontrollieren. Falls die Nachphase noch nicht aktiv ist wird im nächsten Schritt geprüft, ob der Prozess zur Videoaufnahme noch läuft. Dadurch kann sofort ein neuer Prozess zur Aufnahme gestartet werden, falls kein Video mehr aufgenommen wird.

Als nächstes wird die Anzahl der Videos im dazugehörigen Ordner kontrolliert und anschliessend auch hier der Prozess zur Vorverarbeitung analysiert. Wenn die Vorverarbeitung nicht läuft, muss diese neu gestartet werden. Dazu wird jedoch der Pfad zum Video benötigt, welches als nächstes bearbeitet werden muss. Aus diesem Grund muss es möglich sein den Pfad anhand des Erstellungsdatums der Videos herauszulesen. Danach kann die Vorverarbeitung mit dem dazugehörigen Pfad des Videos gestartet werden.

Nach dem gleichen Konzept geht es auch beim dritten Prozess, der Nachbearbeitung, weiter. Der einzige Unterschied zwischen dem zweiten und dritten Prozess besteht darin, dass hierfür zum Starten der Pfad zum Ordner mit den Frames benötigt wird.

Da nebenbei noch ein Webserver läuft, der Informationen über das System bereitstellt, hat die Steuerung und Überwachung noch kleinere Aufgaben. Dies bedeutet, auf Eingaben der Website zu reagieren und Ordner zu erstellen oder zu verschieben, wenn dies vom Benutzer gewünscht wird. Zudem ist eine Temperaturüberwachung eingebaut. Einmal pro Minute wird die Kerntemperatur ausgelesen und in ein File geschrieben. Diesem Prozess ist der vierte Prozesskern zugeteilt. Da die Auslastung des Scripts jedoch nicht sonderlich hoch ist, teilt sich dieser Prozess den Kern mit der Vorverarbeitung. [L9]

Die Software zur Steuerung und Überwachung des Gerätes ist im unten dargestellten Bild (Abbildung 13) visualisiert.

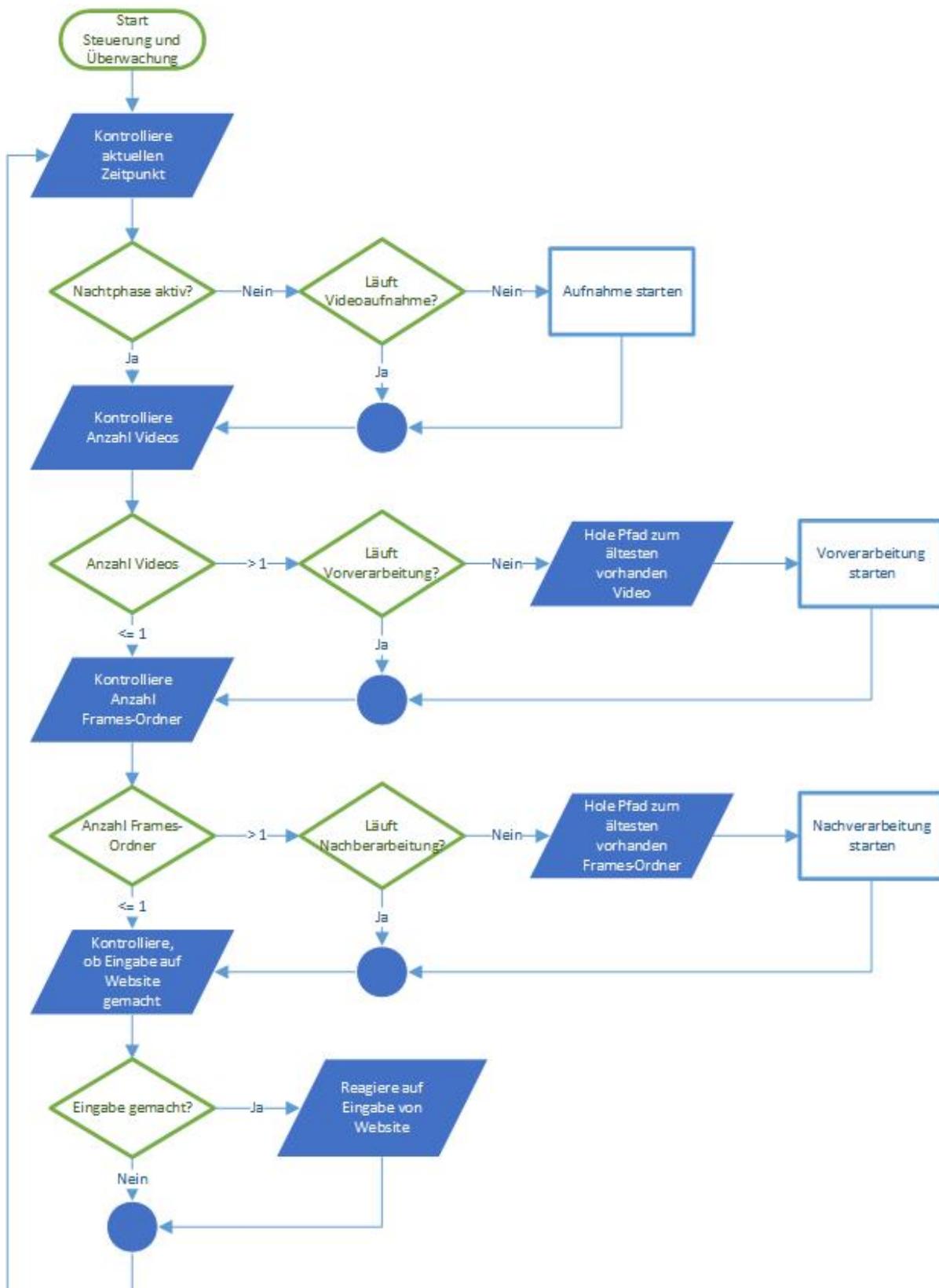


Abbildung 13: Flussdiagramm der Steuerung und Überwachung.

6.3 Github

Zur Verwaltung der Software wurde Github verwendet. Damit konnten Anpassungen oder Änderungen des Quellcodes einfach von der Kommandozeile, im Webbrower oder mithilfe eines Programms unter Windows hochgeladen und anschliessend auf anderen Geräten verbreitet werden. Ebenfalls war es jederzeit möglich die Software auf vorherige Versionen des Quellcodes zurückzusetzen, wenn dies notwendig war. Durch “SourceTree“, einem Programm unter Windows, konnten zudem sämtliche vorgängige Commits rasch und einfach nachgelesen und nachvollzogen werden, da dort zu jedem Commit die spezifischen Änderungen markiert wurden. Ein weiterer Vorteil von Github besteht darin, dass der Quellcode von überall aus erreichbar ist und somit keine verschiedenen Versionen von Codes vorhanden waren. Im späteren Verlauf der Bachelorarbeit war es möglich den Quellcode der Geräte, die bereits an den Strassenlaternen angebracht wurden, stetig zu verbessern. Ohne die Geräte zu entfernen und neu ausrichten zu müssen, konnten neue Versionen der Software per WLAN heruntergeladen und installiert werden. Der Quellcode konnte also jederzeit getestet, angepasst und innerhalb weniger Minuten auf den Geräten aktualisiert werden. Die Installationszeit für Updates wurde sehr kurz gehalten, da es mittels Script durchgeführt wurde, welches sämtliche Programme unter Linux überprüft und neue Software nur installiert, falls diese auf dem Board noch nicht vorhanden ist.

6.3.1 Ordnerstruktur

Sämtlicher Quellcode der Geräte ist auf dem USB-Stick im Anhang der Arbeit beigelegt. Die nachfolgende Tabelle (Tabelle 3) zeigt die Ordnerstruktur des Quellcodes.

MakeVideo			
	makeVideo.sh		
CheckMotion			
	CheckMotion.cpp		
VehicleCount			
Blob.cpp	FileHandler.cpp		VehicleCount.cpp
Blob.h	FileHandler.h		
Handler			
handler.sh	settings.sh		temperature.sh
WLAN			
WLANSettings.sh			
Temp			
deleteFrames.sh	index.php		startVideo.sh
generateCrops.sh	rc.local		temperature.php
generateFeatureVec.sh	startStream.sh		
initialization.sh			

Tabelle 3: Ordnerstruktur.

Das Skript "makeVideo.sh" dient dazu, die Videoaufnahme zu starten. Nach Aufruf des Scripts, wird ein Zeitstempel mit aktuellem Datum und Uhrzeit herausgelesen und daraufhin ein 15-Minütiges Video gestartet, welches im Videos-Ordner gespeichert wird.

Die C++ Datei CheckMotion ist für die Vorverarbeitung zuständig. Nachdem die Datei kompiliert wurde, kann diese ausgeführt werden. Dieses Programm benötigt jedoch den Namen des Videos und den Pfad für die generierten Frames als Argument.

Sämtliche Dateien im VehicleCount werden benötigt, um die Nachbearbeitung durchzuführen. Auch diese Dateien müssen vorgängig kompiliert werden, damit sie ausführbar sind. Dabei ist VehicleCount die Hauptklasse, Filehandler kümmert sich um das Datei-Handling und Blob um die eigentliche "Blob-Detection". Als Argument benötigt die Hauptklasse den korrekten Pfad des Ordners.

Die drei Dateien im Ordner "Handler" gehören zur Steuerung und Überwachung. Dabei beinhaltet die Datei "handler.sh" die Hauptschleife. "settings.sh" wird benötigt, um den Live-Stream zu starten, damit das Gerät ausgerichtet werden kann und "temperature.sh" liest alle 60 Sekunden die Kerntemperatur aus, welche danach in eine Datei geschrieben wird.

Die Datei "WLANSettings.sh" wird benötigt, um die gewünschten WIFI Einstellungen durchzuführen. Dort kann die SSID und das Passwort des Netzwerks eingetragen werden, auf welches sich das Gerät verbindet.

Alle Dokumente, die sich im Ordner "Temp" befinden, müssen nach herunterladen vom Git Repository noch in den korrekten Ordner verschoben werden. Die PHP Dateien beinhalten die Webseite von "Fast and Curious". Die SH Skripte werden für die Eingaben durch den Benutzer auf der Webseite benötigt. Die letzte Datei, "rc.local" ist notwendig, um diverse Programme bereits beim Autostart zu aktivieren, damit kein manuelles anmelden auf dem NanoPi und starten der Skripte nötig ist.

Damit eine Neuinstallation oder ein Update ohne grosse Probleme durchgeführt werden kann, ist die Datei "initialization.sh" vorhanden. Nachdem das Git Repository heruntergeladen wurde, kann dieses Skript ausgeführt werden. Nach dem Starten wird zuerst nach Upgrades und Updates vom sämtlichen Programmen auf dem Board gesucht. Daraufhin werden die notwendigen Ordner erstellt, welche für den Betrieb des Gerätes benötigt werden. Im Anschluss werden OpenCV, Avconv, Apache2 und Mjpg-Streamer installiert, falls diese sich noch nicht auf dem Gerät befinden. Anschliessend werden alle Dateien im Temp Ordner in die richtigen Ordner auf dem Gerät verschoben und dieser danach gelöscht. Die erfolgreiche Lösung des Ordners dient als Indikator, ob alle Dateien korrekt verschoben wurden. Zu guter Letzt werden die C++ Dateien kompiliert und sämtliche Skripte ausführbar gemacht. Erst nachdem all diese Aufgaben abgeschlossen sind, erscheint eine finale Meldung, welche den Benutzer informiert, ob alles erfolgreich abgelaufen ist und das Gerät nach einem Neustart verwendet werden kann.

7 Schwierigkeiten

7.1 Beaglebone Green Wireless

Die Erfahrung mit dem Umgang von Entwicklungsboards war am Anfang der Bachelorarbeit praktisch nicht vorhanden. Dies spiegelte sich auch beim Umgang mit dem Beaglebone Green Wireless wieder. Es handelte sich dabei um ein Entwicklungsboard, welches ohne Bildschirmausgabe verwendet werden musste. Es bestand lediglich die Möglichkeit, die Kommandozeile zu verwenden und dort Eingaben zu machen. Da sich "Fast and Curious" um die Analyse von Fahrzeugen mithilfe von Kameras beschäftigte, war es schwierig, eine Bildverarbeitung ohne visuelle Ausgaben durchzuführen. Damit dies dennoch möglich war, wurde ein HDMI Cape bestellt, um die Kameraausgabe auf dem Bildschirm betrachten zu können. Leider funktioniert das HDMI Cape nur für das Beaglebone Green, nicht aber für das Beaglebone Green Wireless. Aus diesem Grund konnte auch damit keine visuelle Ausgabe erreicht werden. Alternativ wurde auch das Beaglebone Black getestet, da es mit einem HDMI-Ausgang ausgestattet ist. Der Unterschied zwischen diesen beiden Boards war jedoch so gross, dass die Zeit lieber für das eigentliche Board, das Beaglebone Green Wireless, verwendet wurde.

Bis die erste Software auf dem Beaglebone Green Wireless lief, verging einige Zeit. Die erste funktionsfähige Software zum Ansteuern der Kamera wurde mithilfe von Visual Studio 2015 auf dem Computer unter Windows programmiert und getestet. Anschliessend wurde es auf einer virtuellen Maschine unter Eclipse und Linux optimiert und erneut getestet. Nach einigen weiteren Optimierungen konnte es auf dem Beaglebone Green Wireless zum Laufen gebracht werden. Die erste funktionsfähige Software nahm mithilfe der USB Kamera und OpenCV ein Bild auf und legte es im internen Speicher ab. Das Ergebnis konnte danach auf die externe Speicherkarte verschoben und auf dem Laptop zum ersten Mal betrachtet werden.

7.2 Radarsensor

Anfangs wurde die Möglichkeit eines Radarsensors zur Bestimmung der Geschwindigkeit in Betracht gezogen. Jedoch ergab die Auswertung der Daten des Radarsensors einige Schwierigkeiten. Zum einen waren die analogen Eingänge des Beaglebones nur über ihr eigenes BoneScript-Programm erreichbar. Man musste also zuerst über das Programm die Eingänge ansprechen und konnte erst danach die Eingänge über C++ auslesen. Zum anderen sind die Beschreibungen des Herstellers sehr ungenau, da beispielsweise die angegebene Reichweite von ca. zehn Metern nie erreicht wurde. Es wurde lediglich eine Reichweite von knapp sechs Metern erreicht. Dies stellte ein Problem dar, da die Geräte auf einer Höhe von sechs Metern an den Strassenlaternen befestigt werden. Zwangsläufig muss die Reichweite des Sensors dann erheblich grösser sein. Ein weiterer Nachteil des Beaglebones mit dem verwendeten Radarsensor waren die unterschiedlichen Spannungen. Der Radarsensor lieferte eine Ausgangsspannung von 0-5 V, wobei das Beaglebone nur analoge Eingänge mit 3.3 V besitzt. Deshalb musste zusätzlich ein Spannungsteiler eingelötet werden.

Mit der Verwendung des NanoPi NEO fiel schlussendlich der gesamte Radarsensor aus dem Konzept, weil dieser keine internen analogen Eingänge besitzt. Es musste also eine neue Methode gefunden werden, um die Geschwindigkeit der Verkehrsteilnehmer zu bestimmen.

7.3 OpenCV

Eine weitere Hürde war die Bibliothek OpenCV erfolgreich auf dem Beaglebone zu installieren. Mithilfe der offiziellen Paketquellen kann lediglich die Version 2.4 von OpenCV installiert werden. Da jedoch für sämtliche Features, welche für "Fast and Curious" benutzt wurden, die Version 3.2 notwendig war, musste die Software anderweitig installiert werden. Es kam soweit, dass ebenfalls geprüft wurde, ob es möglich wäre den Sourcecode von "Fast and Curious" soweit zu optimieren, dass er auch mit der Version 2.4 laufen konnte. Jedoch wurde diese Idee relativ schnell wieder verworfen, da die wichtigsten Methoden von OpenCV nur unter der Version 3.2 installiert waren. Zudem war es ein Problem die riesige Library von OpenCV auf dem knappen, internen Speicher des Beaglebones, welcher nur vier Gigabyte gross ist, zu installieren. Aus diesem Grund wurden viele Anläufe und Versuche benötigt, bis ein funktionsfähiges OpenCV, mit der Version 3.2, auf dem Entwicklungsboard installiert war. OpenCV besteht, wie schon im Kapitel Software erwähnt, aus Haupt- und Extramodulen. Die Hauptmodule konnten nach längerem testen und dem Löschen von nicht benötigter Software auf dem Beaglebone installiert werden, jedoch wurden die Extramodule zu diesem Zeitpunkt als nicht notwendig erachtet und aufgrund des Speicherplatzes weggelassen. Im späteren Verlauf der Bachelorarbeit wurde aber festgestellt, dass einige Zusatzmodule zum Auswerten der Verkehrsteilnehmer benötigt wurden. Aufgrund dessen mussten diese dann ebenfalls noch zum Laufen gebracht werden.

Schlussendlich konnte OpenCV mit der Version 3.2 und den benötigten Zusatzmodulen auf dem Beaglebone installiert werden. Danach wurde ein Git Repository, speziell für OpenCV, mit allen notwendigen Modulen und Einstellungen erstellt und seither nur dieses verwendet. Das Repository konnte selbst beim Wechsel vom Beaglebone zum NanoPi ohne Probleme weiterverwendet werden. Der Speicherplatz war auf dem NanoPi kein Problem mehr, da dort kein interner Speicher vorhanden ist und die Installation auf einer SD Karte mit mehr Speicherkapazität durchgeführt werden konnte.

7.4 WLAN

Das WLAN des Beaglebone Green Wireless liess sich nach leichten Schwierigkeiten sehr rasch mit dem Eduroam Netzwerk der NTB verbinden. Laut den Angaben des Herstellers kann das Beaglebone Green Wireless das integrierte WLAN in einen SoftAP-Mode umwandeln, so dass man mit einem externen Gerät Zugriff darauf erhält. Nach langer Internetrecherche und einigen Testphasen am Board selbst, ist es jedoch nicht gelungen das integrierte WLAN in diesen Mode zu versetzen. Es blieb keine andere Möglichkeit, als die schon vorhandene Webseite des Herstellers umzufunktionieren und diese selbst zu verwenden. So wurde aus dem Startfenster zum Verbinden mit einem herkömmlichen WLAN, die Startseite des Gerätes. Dies war nur eine Notlösung, welche aber sehr gut funktionierte.

Nach dem Wechsel auf den NanoPi NEO, musste das ganze Prozedere von vorne durchgeführt werden. Trotz einer guten Anleitung eines Mitstudenten konnten auch hier die WLAN-Adapter am NanoPi nicht in den AP-Mode versetzt werden. Somit war es vom NanoPi nicht möglich einen Hotspot zu errichten. Trotz gleichem Image, selbem Board und auch gleichem WIFI-Adapter funktionierte es nicht und es musste schnell eine andere Lösung gefunden werden. Da jeder ein Smartphone mit Hotspotfunktion hatte, wurde von diesen aus ein Hotspot eröffnet, auf welchen sich der NanoPi verbinden konnte. Erst danach war es möglich das Gerät zu verwenden. Eine weitere Hürde war es, mit den unterschiedlichen WIFI-Adaptoren zurecht zu kommen. Aufgrund der Verfügbarkeit der WIFI-Adapter wurden schlussendlich drei verschiedene Arten genutzt, wodurch es umso schwieriger war, das System mit allen Adaptoren stabil zum Laufen zu bekommen.

7.5 Videoaufnahme

Die Aufnahme der Videos erwies sich ebenfalls als schwierig, wodurch einige Schritte und Möglichkeiten getestet wurden, bis die Lösung mit "Avconv" gefunden wurde. Die erste Idee war die direkte Verarbeitung des aufgenommenen Videomaterials. Dafür wurde mithilfe von OpenCV ein Frame nach dem anderen aufgenommen und verarbeitet. Aufgrund der Prozessorgeschwindigkeit des Beaglebones wurde diese Idee jedoch schnell wieder verworfen, da mit dieser Methode nur etwa ein Bild pro Sekunde erreicht wurde. Die nächste Möglichkeit bestand darin, ebenfalls mit OpenCV die Videos aufzunehmen und ohne Verarbeitung direkt zu speichern. Somit hätte man die Verarbeitung extern auf dem Computer durchführen müssen. Jedoch wurden selbst mit dieser Variante nur etwa zehn Bilder pro Sekunde erzielt, was für den eigentlichen Verwendungszweck immer noch viel zu wenig war. Da es mit OpenCV nicht möglich ist, über zehn FPS zu kommen, wurden andere Programme unter Linux gesucht, welche für eine Videoaufnahme verwendet werden konnten. Daraufhin wurde das Programm "Streamer" getestet. Leider konnten auch mit diesem Programm die gewünschten Resultate nicht erreicht werden. Sobald eine Bildrate von mehr als 15 angegeben wurde, gingen einzelne Bilder verloren. Aus diesem Grund war das Video schlussendlich nicht zu gebrauchen. Das Programm zeigte aber, dass eine Videoaufnahme ohne OpenCV vielversprechend sein könnte. Deshalb wurden anschliessend noch weitere Programme ausgetestet. [L10]

Im Anschluss wurde das Programm "Mjpg-Streamer" genauer analysiert. Für eine Videoaufnahme mit anschliessender Speicherung war das Programm nicht gut genug, da auch hier einzelne Bilder verloren gingen. Entwickelt wurde dieses Programm jedoch um zu Streamen. Da es in diesem Bereich sehr viel Potential zeigte, konnte es bei "Fast and Curious" zum streamen des Videos auf der Webseite eingesetzt werden. [L4]

Als ein sehr vielversprechendes Programm zeigte sich "ffmpeg". Damit ist es möglich Einstellungen bis ins kleinste Detail vorzunehmen. Mit diesem Programm wurden etwa 20 FPS erreicht, jedoch auch nicht mehr, egal welche Parameter eingestellt wurden. Ob diese 20 Bilder pro Sekunde ausreichen, wurde in vielen Testaufnahmen mit diesem Programm ausprobiert. Da diese Anzahl an Bildern pro Sekunden eher knapp waren, wurde dennoch nach einem weiteren Programm gesucht. [L11]

Schliesslich wurde das Programm "Avconv" gefunden. Dabei handelte es sich um ein ähnliches Programm wie bei "ffmpeg", jedoch mit mehr Einstellungen. Es verfügte zudem über eine sehr umfangreiche Dokumentation und hilfreiche Ausgaben während der Aufnahme. Mithilfe dieser Ausgaben konnte das Video besser analysiert und die Aufnahme angepasst werden. Durch diese Anpassungen wurden dann zum ersten Mal Videos mit 25 FPS ohne den Verlust einzelner Bilder erreicht, weshalb die Entscheidung schlussendlich auf dieses Programm fiel. Mit anderen Einstellungen wurde getestet, ob noch höhere Bildraten zu erreichen sind. Jedoch fanden bei 30 Bildern pro Sekunde auch mit "Avconv" die ersten Bildverluste statt, weshalb der jetzige Aufnahmeprozess mit einer Bildrate von 25 FPS arbeitet. [L5]

8 Bedienungsanleitung

8.1 Installation

Um ein neues Gerät zu erstellen, müssen zuerst sämtliche Hardwareteile besorgt und korrekt zusammengebaut werden.

Beim Zusammenbau der einzelnen Geräte muss lediglich darauf geachtet werden, dass der Spannungswandler polgerecht an der Autobatterie angeschlossen wird. Werden die Drähte vertauscht so kann ein Kurzschluss entstehen und die gesamte Hardware zerstören. Der restliche Zusammenbau ist sehr einfach. Den USB-Hub am NanoPi NEO anschliessen und an diesen dann sowohl Kamera als auch den WIFI-Adapter anbringen. Danach wird lediglich der Strom korrekt angeschlossen. Das Gerät ist danach einsatzbereit.

Achtung: Auf richtige Polung achten! Kurzschlussgefahr!

Nachdem das Gerät zusammengebaut wurde, muss im nächsten Schritt die Speicherkarte mit dem Ubuntu Image beschrieben werden. Das verwendete Image der Bachelorarbeit wurde dem USB-Stick dieser Bachelorarbeit beigelegt. Dies kann mit einem dazu geeigneten Programm, beispielsweise "Win32 Disk Imager", durchgeführt werden. Im Anschluss muss die beschriebene Karte in den Speicherkartenslot des NanoPi geschoben und dieser dann anschliessend gestartet werden. Die Rechnereinheit sollte zur Installation via Netzwerkkabel verbunden werden um die IP-Adresse beim Router nachschauen zu können. Zudem ist zu diesem Zeitpunkt das WLAN Netzwerk noch nicht konfiguriert. Via "Putty" kann über die IP-Adresse auf den NanoPi zugegriffen werden. Für das bereitgestellte Image von Ubuntu lautet der Benutzername "root" und das Passwort "fa". Es empfiehlt sich das Standardpasswort zu diesem Zeitpunkt zu ändern, um die Daten auf der Speicherkarte zu schützen. Die benötigten Dateien und sonstige Software können von Github bezogen werden. Sobald die Dateien heruntergeladen wurden muss das Bash-Skript "initialization.sh" ausgeführt werden, worauf alles Notwendige installiert wird. Dieser Vorgang dauert etwa drei Stunden, da für OpenCV viel Dateien kompiliert werden müssen.

Damit das WLAN Netzwerk später funktioniert, muss der Name in "/etc/network/interfaces" angepasst werden. Dies wurde nicht im Installationsskript durchgeführt, da für die Bachelorarbeit drei verschiedene WLAN Sticks verwendet wurden. Sobald alles fertig installiert und der Name des Netzwerks angepasst ist, kann das Gerät neu gestartet werden. Somit ist die Installation des Gerätes "Fast and Curious" abgeschlossen.

8.2 Updates

Updates sind möglichst einfach durchzuführen, da die Software auf einem beliebigen Gerät angepasst und die Änderungen dann anschliessend auf Github hochgeladen werden kann.

Damit die durchgeführten Änderungen auf die anderen Geräte verbreitet werden können, muss man via "Putty" auf das Gerät zugreifen. Dies kann direkt vor Ort an einem Gerät stattfinden, wenn mit dem Mobilphone ein Hotspot errichtet wird. Dazu benötigt der Hotspot jedoch die im Skript "WLAN-Settings.sh" eingegebenen Parameter für SSID und Passwort. Im Anschluss sollte sich das Gerät mit dem Hotspot verbinden und die IP-Adresse angezeigt werden. Wenn der Zugriff auf das Gerät erfolgreich war, muss der Ordner "BA" gelöscht und neu von Github heruntergeladen werden. Im Anschluss daran kann das Initialisierungsskript ausgeführt werden, wodurch alles neu installiert, verschoben und kompiliert wird. Nach einem Neustart befindet sich das Gerät auf dem neusten Stand.

8.3 Aufstellung des Gerätes

Um gewährleisten zu können, dass das Gerät an einer neuen Position optimal funktioniert, sollte eine Strassenlaterne benutzt werden, welche genug hoch ist, um das Gerät auf einer Höhe von sechs Metern platzieren zu können. Da die Rechnereinheit bei erhöhter Auslastung eine hohe Kerntemperatur aufweist, wäre es von Vorteil, wenn die Strassenlaterne keiner direkter Sonneneinstrahlung ausgesetzt ist. Die Geschwindigkeitsbegrenzung der Strasse sollte 80 km/h nicht überschreiten, damit die Verkehrsteilnehmer gut erkannt werden. Die Ausrichtung des Gerätes erfolgt in etwa Richtung der Strasse. Die genaue Ausrichtung erfolgt beim Systemstart.

8.4 Systemstart

Nach dem Aufstellen kann das Gerät in Betrieb genommen werden, jedoch muss noch eine genauere Ausrichtung durchgeführt werden, damit die Strasse optimal im Sichtfeld der Kamera ist. Diese Ausrichtung wird erreicht, indem das Mobilphone, wie bei einem Update als Hotspot eingesetzt wird. Sobald die Stromversorgung zum Gerät angeschlossen wurde, wird es versuchen, sich mit dem Hotspot zu verbinden. Nach erfolgreicher Verbindung des Gerätes kann die IP-Adresse, welche im Anschluss im Browser eingegeben wird, ausgelesen werden, um auf die Webseite von "Fast and Curious" zu kommen. Auf dieser Webseite kann nun die genaue Ausrichtung des Gerätes erfolgen, da dort der Live-Stream der Kamera zu sehen ist. Das nachfolgende Bild (Abbildung 14) zeigt die Webseite des Gerätes.

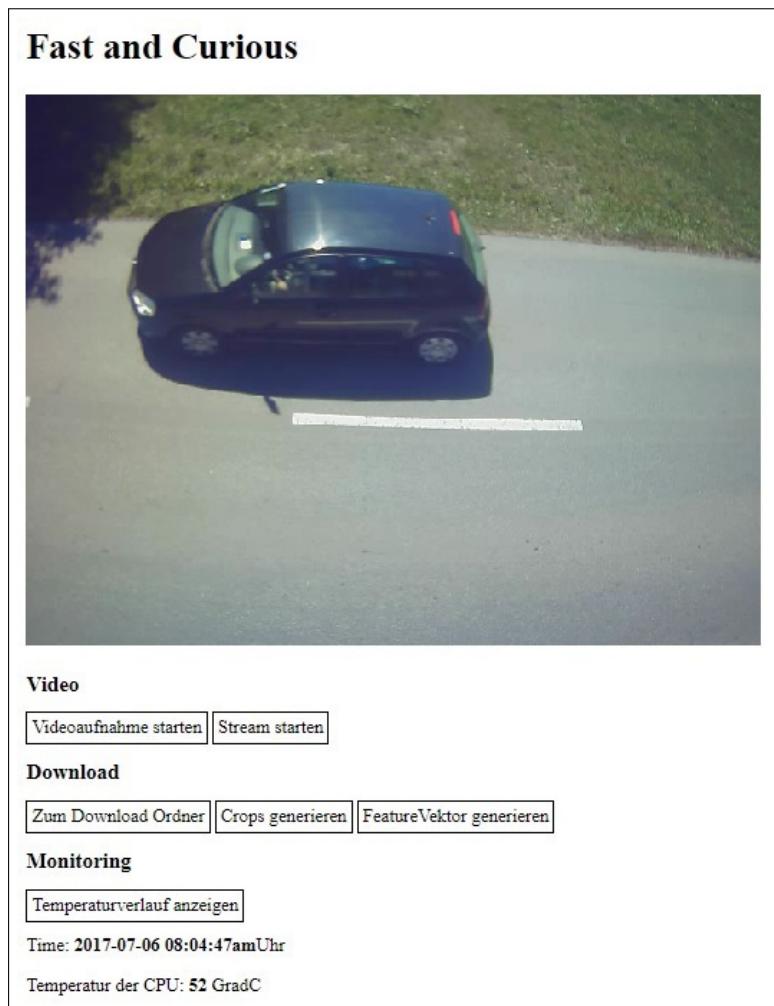


Abbildung 14: Webseite von "Fast and Curious".

Wenn die Feinjustierung des Gerätes abgeschlossen ist, kann die eigentliche Aufnahme der Verkehrsteilnehmer durch den Knopf "Videoaufnahme starten" gestartet werden. Zu diesem Zeitpunkt wird der Live-Stream einfrieren, da die Kamera vom Streaming Programm "Mjpg-Streamer" an die Videoaufnahme "Avconv" übergeben wird.

8.5 Monitoring

Das Monitoring soll dazu dienen die Funktionstüchtigkeit des Gerätes zu überprüfen. Zudem kann damit die Ausrichtung überprüft und die Kerntemperatur des NanoPi analysiert werden. Eine blaue LED auf dem NanoPi signalisiert, dass das Gerät mit Spannung versorgt wird und somit noch eingeschaltet ist. Um weitere Informationen des aktuellen Gerätes zu erhalten, muss eine Verbindung via Hotspot errichtet werden. Im Anschluss kann auf die Website zugegriffen und dort weitere Daten begutachtet werden. Die Ausrichtung des Gerätes kann man überprüfen, indem der Knopf "Stream starten" betätigt wird. In diesem Moment wird die Videoaufnahme unterbrochen und das aktuelle Video beendet. Die Verantwortung der Kamera geht vom Aufnahmeprogramm "Avconv" an das Streaming Programm "Mjpg-Streamer" über, wodurch der Live-Stream auf der Homepage erscheint. Nachdem die Ausrichtung der Kamera überprüft und, je nach Notwendigkeit, angepasst wurde, kann die Videoaufnahme wieder durch den zugehörigen Knopf gestartet werden. Ein Einblick in den Temperaturverlauf ist mithilfe des Knopfes "Temperaturverlauf anzeigen" möglich. Da die Kerntemperatur einmal pro Minute ausgelesen und in ein File geschrieben wird, kann diese jederzeit dem Diagramm entnommen werden.

Auf dem untenstehenden Bild (Abbildung 15) ist ein Beispiel eines solchen Verlaufs, wie er auch auf der Webseite angezeigt wird. Diese Daten stammen von einem NanoPi, welcher während den Testaufnahmen in Satteins eingesetzt wurde.

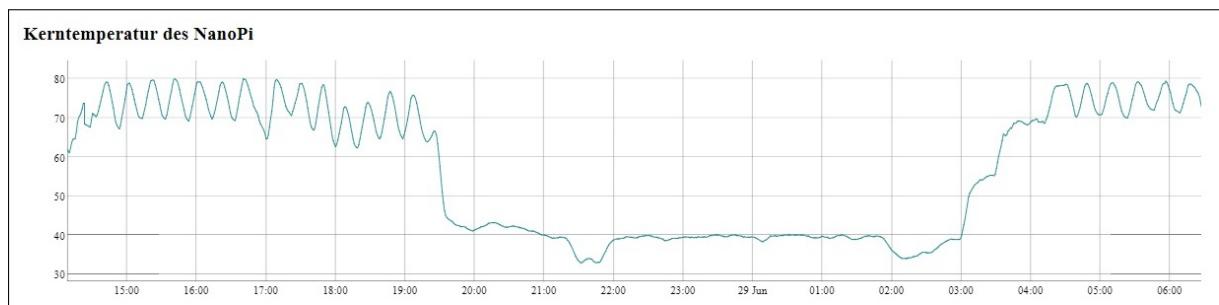


Abbildung 15: Temperaturverlauf eines NanoPi.

8.6 Datenerwerb

Die aufgenommenen Daten können jederzeit, auch während dem Betrieb des Gerätes, heruntergeladen werden. Damit dies ohne grossen Aufwand passiert, muss wieder eine Verbindung zum Gerät mithilfe eines Hotspots erstellt werden. Im Anschluss kann der Download Ordner über die Webseite erreicht werden. Damit darin die neusten Daten vorhanden sind, müssen diese vorgängig generiert werden. Dies geschieht, indem die Crops und der Feature Vektor über die zugehörigen Knöpfe auf der Webseite gezippt und zum Download Ordner übertragen werden. Beim Feature Vektor handelt es sich um eine Tabelle, wie sie im Abschnitt "Software" (Tabelle 2) zu sehen ist. Die Crops beinhalten ein Originalbild, ein Differenzbild und ein kleineres Bild mit dem bewegten Blob des Verkehrsteilnehmers. Es kann ebenfalls eine Tabelle mit sämtlichen aufgezeichneten Temperaturdaten heruntergeladen werden, damit diese Daten extern ausgewertet werden können.

9 Testversuche

9.1 Prototypen

Während der Arbeit wurden zwei verschiedene Prototypen realisiert, bevor dann das Endprodukt entstanden ist. Nachfolgende Abbildung (Abbildung 16) zeigen diese beiden eingesetzten Varianten. Dabei handelt es sich beim linken Bild um den ersten Prototypen, der beim Video mittels Laptop zum Einsatz kam. Beim rechten Bild um denjenigen, der im späteren Verlauf für die Videos mithilfe des Beaglebones eingesetzt wurde.

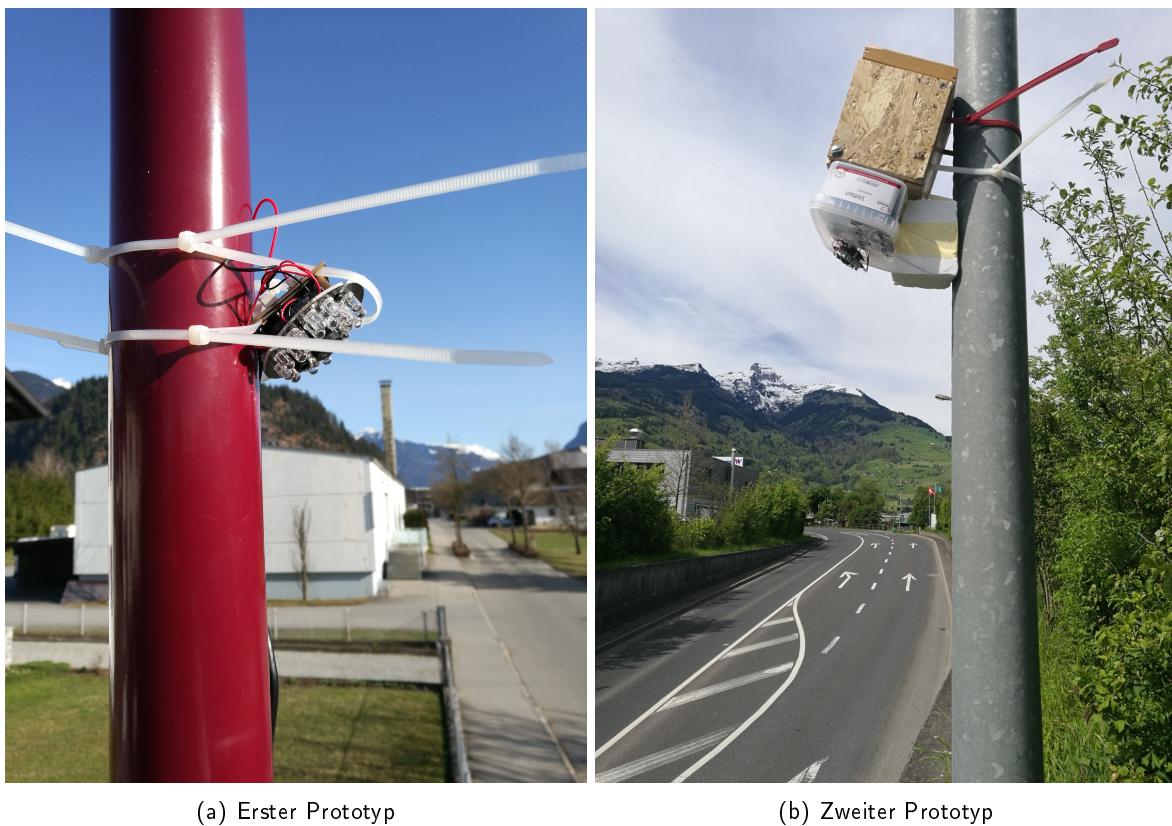


Abbildung 16: Eingesetzte Prototypen.

9.2 Video mittels Laptop

Die ersten Videos für die Bachelorarbeit wurden mithilfe der Bildverarbeitungsbibliothek OpenCV auf einem Laptop aufgenommen. Zu diesem Zeitpunkt war die Aufgabe von OpenCV lediglich, die Videos Frame für Frame aufzunehmen und sie mit einem Videowriter aneinander zu reihen. Da der Prozessor des Laptops sehr leistungsstark war, konnten qualitativ ausgezeichnete Videos aufgenommen werden. Es ist lediglich aufgefallen, dass beim Abspielen des Videomaterials auf einem Computer die Geschwindigkeit erhöht war, wodurch ein 30-minütiges Video in 15 Minuten zu Ende war. Dies wurde zu diesem Zeitpunkt auf eine falsche Einstellung am Videowriter zurückgeführt. Da aber lediglich die einzelnen Frames relevant waren, konnte die Abspielgeschwindigkeit des Videos vernachlässigt werden. Durch das aufgenommene Videomaterial konnte schon in einer frühen Phase der Arbeit die Qualität der Kamera überprüft und Material für weitere Verarbeitungsschritte aufgenommen werden. Diese Aufnahmen entstanden an zwei Tagen in Ludesch und auf dem Parkplatz des NTBs. Obwohl die Kamera an beiden Orten nicht optimal positioniert wurde, konnten die Aufnahmen dennoch für viele Testprogramme und Szenarien verwendet werden. In diesem Moment wurde die Option der Aufnahme via Laptop gewählt, da das Beaglebone noch nicht einsatzbereit war, jedoch parallel zur Hardware auch an der Software entwickelt werden sollte.

9.3 Video mittels Beaglebone

Nachdem alle notwendigen Programme installiert und die zugehörigen Einstellungen auf dem Beaglebone abgeschlossen waren, konnten erste Videoaufnahmen mithilfe dieses Entwicklungsboards aufgenommen werden. Zu diesem Zeitpunkt war das Gerät bereits standalone und benötigte aus diesem Grund keine externe Stromzufuhr mehr. Der Testversuch wurde an einer Strassenlaterne in Buchs, auf einer Höhe von etwa fünf Metern, durchgeführt. Die präzise Ausrichtung des Gerätes konnte durch das Mobilphone erfolgen, da das Programm alle zehn Sekunden ein Bild abgespeichert und es auf einer Internetseite angezeigt hatte. Dort konnten dann zum ersten Mal Bilder mit optimaler Positionierung der Kamera aufgenommen werden. Die Aufgabe des Programms war es, die Fahrzeuge aufzunehmen, mithilfe von OpenCV zu identifizieren und bei erfolgreicher Identifikation eines Verkehrsteilnehmers zwei Bilder in einem Ordner abzuspeichern. Zudem sollte das komplette Video ebenfalls abgespeichert werden, damit dies für weitere Auswertungen verwendet werden konnte. Die Software selbst funktionierte sehr gut. Leider musste aber festgestellt werden, dass durch die lange Prozedurdauer der Hauptschleife nur eine Bildrate von etwa einem FPS erzielt werden konnte. Da die Videos dennoch mit 30 FPS aneinander gereiht wurden, war ein 30-minütiges Video in etwa zwei Minuten vorbei. Aus diesem Grund konnte das Videomaterial praktisch gar nicht verwendet werden.

Diese Testaufnahmen zeigten, dass die Durchführung mittels Prozessor mit nur einem Kern praktisch unmöglich für diese Arbeit war. Eine Möglichkeit, welche mit dem Beaglebone dennoch bestand, war es, nur Videos aufzunehmen und diese danach sofort abzuspeichern. Da der Speicher des Beaglebones trotz Speicherkarte sehr schnell voll gewesen wäre, hätte die Speicherkarte des Gerätes täglich gewechselt und die komplette Verarbeitung der Videos extern durchgeführt werden müssen. Da dies sehr umständlich gewesen wäre, mussten Alternativen gesucht werden. Aus diesem Grund folgte dann der Umstieg vom Beaglebone zum NanoPi, da dieser Prozessor mit vier Kernen ausgestattet war.

9.4 Verarbeitung

Nachfolgend werden die Methoden beschrieben, welche für die Verarbeitung der generierten Bilder verwendet werden könnten. Je nach Methode könnte man diese Funktionen direkt auf den NanoPi implementieren. Somit wäre es möglich die Nachverarbeitung zu erweitern und bei sämtlichen Verkehrsteilnehmern diese Funktionen ebenfalls durchzuführen und auszuwerten. Das Resultat daraus wäre ein besserer Feature Vektor, was die Verkehrsverfolgung erleichtern würde. Wichtig dabei ist es jedoch zu beachten, dass die Verarbeitungszeit auf dem NanoPi nicht allzu sehr erhöht wird, da ansonsten die Prozessierung der Verkehrsteilnehmer zu viel Zeit in Anspruch nehmen würde.

9.4.1 Schattenentfernung

Als die ersten Testaufnahmen durchgeführt wurden, musste festgestellt werden, dass Schatten zu einem grossen Problem führen konnte. Dies trat vor allem dann auf, wenn die Sonne den Schatten des Verkehrsteilnehmers in Richtung Kamera projizierte. Dadurch wurde der Bildausschnitt des Verkehrsteilnehmers grösser als er tatsächlich war, was weitere Auswertungen verfälschte. Aufgrund dessen wurde nach Möglichkeiten gesucht, um diesen Schatten zu entfernen, weshalb schlussendlich eine Funktion dafür geschrieben wurde. Falls ein Schatten vorhanden war, konnte dies am Differenzbild erkannt werden. Dort traten jeweils, waagrecht betrachtet, zwei helle Streifen und dazwischen ein dunkler Abschnitt auf. Auffällig daran ist, dass die Breite der hellen Steifen fast gleich gross war, was auf die Geschwindigkeit des Fahrzeugs zurückzuführen ist. Der helle Streifen entstand, wenn nur auf einem der beiden Bilder, welche für das Differenzbild genutzt wurden, ein Schatten vorhanden war. Der dunkle Abschnitt zwischen den hellen Stellen entstand, wenn auf beiden Eingangsbildern bereits ein Schatten vorlag.

Die nachfolgende Abbildung (Abbildung 17) zeigt die geschilderte Situation eines Differenzbildes mit Schatten.



Abbildung 17: Differenzbild zur Demonstration des Schattens.

Beim Algorithmus der Funktion wird jeweils eine horizontale Linie betrachtet und dabei die Anzahl an hellen Pixeln addiert. Dies wird für beide hellen Teile durchgeführt und schlussendlich deren Anzahl verglichen. Falls die Unterschiede der beiden Zähler weniger als drei Pixel betragen, wird eine Variable um eins erhöht und dasselbe auf der nächsten Linie durchgeführt. Falls der Unterschied beider Zähler grösser als drei Pixel ist, wird die besagte Variable wieder auf null gesetzt. Sobald die Variable 20 erreicht, bedeutet dies, dass 20 Linien am Stück gefunden wurden, an denen die hellen Teile fast gleich gross waren. Dies zeigt, dass es sich hier um den Schatten und nicht mehr um den Verkehrsteilnehmer handeln muss. Deshalb kann ab diesem Bereich der untere Teil des Bildes abgeschnitten werden.

In der untenstehenden Abbildung (Abbildung 18) wurde der Algorithmus auf das obere Bild (Abbildung 17) angewandt.



Abbildung 18: Differenzbild mit entferntem Schatten.

Die Abmasse des neuen Differenzbildes können nun auf den normalen Bildausschnitt des Verkehrsteilnehmers übertragen werden, damit dieser dann ohne Schatten weiterverarbeitet werden kann.

9.4.2 GrabCut

Bei "GrabCut" handelt es sich um einen Algorithmus, welcher den Hintergrund eines Bildes entfernt und diesen durch eine beliebige Farbe ersetzt. Dadurch kann danach lediglich der Vordergrund des Bildes weiterverwendet werden. Dieser Algorithmus ist bereits in OpenCV implementiert und hatte das Ziel, mit möglichst wenig Eingabeparameter durch den Benutzer eine Extraktion des Vordergrunds zu erreichen. Damit "GrabCut" korrekt funktioniert, benötigt es einer Deklaration, was als Hintergrund gewertet werden soll. Dies geschieht, indem ein Rahmen in das Bild gelegt wird. Alles was außerhalb dieses Rahmens ist, zählt zum Hintergrund. OpenCV schneidet im Anschluss den Rahmen und sämtliche, darin enthaltene Ähnlichkeiten innerhalb des Rahmens aus, bis eine zu grosse Veränderung zwischen den umliegenden Pixeln gefunden wurde.

Die folgende Abbildung (Abbildung 19) dient dabei zur Veranschaulichung von "GrabCut".

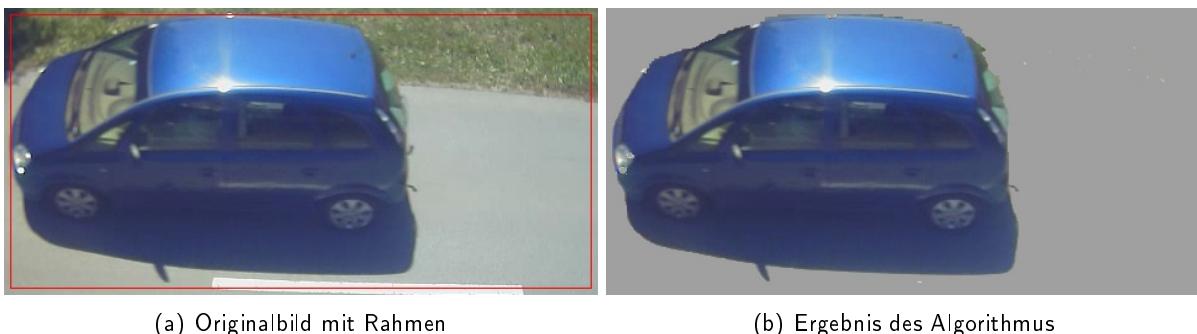


Abbildung 19: Beispiel des "GrabCut"-Algorithmus.

Auf dem linken Bild sind das Originalbild und der definierte Rahmen zu sehen, auf dem rechten Bild das Ergebnis dazu. Dem Ergebnis wurde ein grauer Hintergrund hinzugefügt, jedoch kann diese Farbe auch frei gewählt werden. Je nachdem welches Bild verwendet wird, kann "GrabCut" schlechte bis hervorragende Resultate liefern. Dabei kommt es vor allem darauf an, wie starke Farbveränderungen im Hintergrund vorhanden sind. Bei diesem Algorithmus handelt es sich um eine komplexe Methode, welche selbst auf dem Computer einige Sekunden an Verarbeitungszeit benötigt. [L12]

9.4.3 K-Means Clustering

"K-Means Clustering" ist ebenfalls ein Algorithmus, der bereits in OpenCV implementiert ist. Dieser reduziert die Anzahl an Farben innerhalb eines Bildes auf eine beliebige Anzahl, was für die Bestimmung der dominanten Farbe verwendet werden könnte. Somit könnte hiermit die Farbe des Fahrzeugs ermittelt und danach dem Feature Vektor ergänzt werden. Mithilfe eines Integer kann bestimmt werden, wie viele Cluster das Bild schlussendlich haben soll. Dies bedeutet, wie viele verschiedene Farben nach Abschluss des Algorithmus noch vorhanden sind. Die dominanten Farben werden anfangs zufällig gewählt und danach durch mehrere Iterationen verschoben, bis der quadratischen Abstand zu einem der Farbpunkte minimal wird. Aus diesem Grund benötigt dieser Algorithmus einige Zeit, bis sich die Farbpunkte nicht mehr ändern und deshalb das Resultat errechnet wurde.

Zur Illustration des geschilderten Algorithmus wurde das nachfolgende Beispiel (Abbildung 20) hinzugefügt. Es zeigt das Originalbild und das Ergebnis des Clustering Algorithmus. Damit dafür die dominante Farbe ermittelt werden konnte, wurde das Bild vorgängig mit “GrabCut“ bearbeitet.

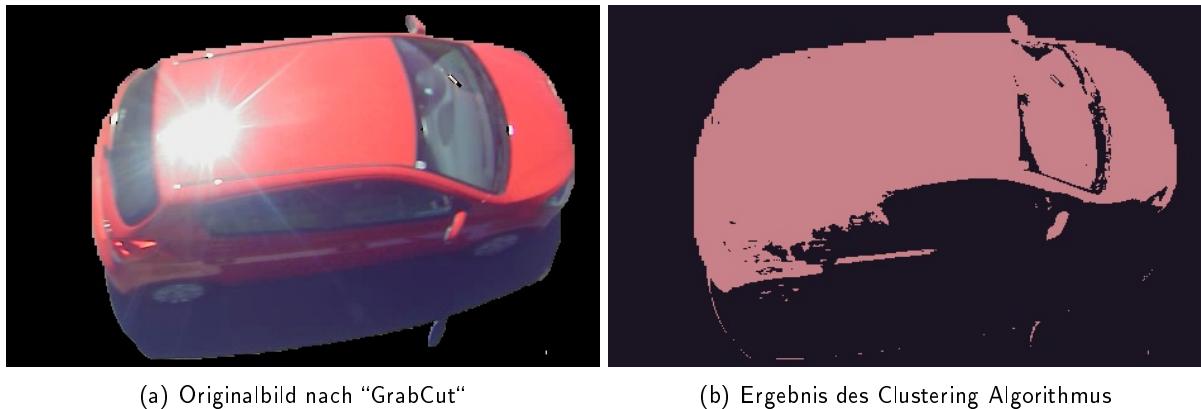


Abbildung 20: Beispiel K-Means Clustering.

Die Anzahl der eingegebenen Cluster im Beispiel betrug zwei. Obwohl die Farbe des Ergebnisses heller als beim Originalbild erscheint, könnte damit die Farbe “rot“ extrapoliert und dem Feature Vektor ergänzt werden. [L12], [L13]

9.4.4 Kategorisierung

Die Kategorisierung erfolgt durch die Anzahl an Pixeln, welche ein Fahrzeug beinhalten. Ein Schwellwert hierbei trennt die einzelnen Kategorien untereinander ab. Dazu muss jedoch zusätzlich, je nach Fahrbahn, ein Faktor eingerechnet werden, da weiter entfernte Objekte kleiner sind und somit weniger Pixel enthalten. Um die besagte Anzahl an Pixeln zu errechnen, wird vorrangig wieder der “GrabCut“-Algorithmus verwendet um zu erkennen, welche Bildpunkte zum Fahrzeug gehören. Nach dem Algorithmus besteht das Bild nur noch aus weissen und schwarzen Pixeln. Dabei sind die schwarzen Pixel jene, die zum Hintergrund gehören, während die weissen Pixel zum bewegten Fahrzeug zählen. Die genaue Anzahl an Pixeln kann über die einfache, bereits in OpenCV implementierte Methode “CountNonZero“ berechnet werden. Diese Methode zählt die Anzahl aller Pixel des Bildes, die nicht schwarz sind.

Das nachfolgende Beispiel (Abbildung 21) dient dabei zur Verdeutlichung des geschilderten Vorgangs.

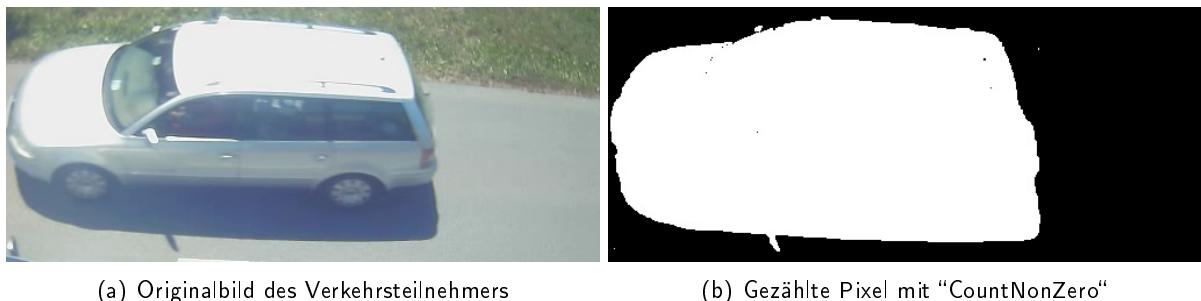


Abbildung 21: Beispiel zur Kategorisierung der Verkehrsteilnehmer.

Die weissen Pixel können addiert und das Ergebnis nach der Kategorisierung durch den Schwellwert ebenfalls dem Feature Vektor hinzugefügt werden. [L12], [L14]

9.4.5 Geschwindigkeit

Vom Prinzip her funktioniert die Errechnung der Geschwindigkeit nach einem ähnlichen Muster wie die Kategorisierung. Auch hier wird der "Grabcut"-Algorithmus verwendet, um den Verkehrsteilnehmer vom Hintergrund zu trennen. Anschliessend wird in der Mitte des Bildes, sowohl links als auch rechts, die Anzahl an schwarzen Pixeln bis zum Rand ermittelt und das grössere Ergebnis genommen. Dieses Ergebnis zeigt die Distanz in Pixeln, welche innerhalb zwei aufeinanderfolgenden Frames vom Fahrzeug zurückgelegt wurde. Der verwendete Bildausschnitt ist dabei stets auf einer Seite länger, da dort die Position des Fahrzeugs vom vorherigen Frame noch involviert ist. Aus diesem Grund kann direkt dieser Abstand also Berechnungskriterium gewählt werden. Eine grobe Einschätzung der Geschwindigkeit ist somit relativ einfach zu bewältigen. Wie beim Kategorisieren muss auch hier ein Faktor aufgrund der Fahrbahn mit eingerechnet werden.

Zur besseren Verständlichkeit wurde das folgende Bild (Abbildung 22) hinzugefügt.



(a) Originalbild des Verkehrsteilnehmers

(b) Ergebnis nach “GrabCut“ und “CountNonZero“

Abbildung 22: Beispiel zur Geschwindigkeitsermittlung eines Verkehrsteilnehmers.

Die rote Linie zeigt den Ort der errechneten schwarzen Pixel bis zum Rand. Daraus kann die Geschwindigkeit berechnet und dem Feature Vektor ergänzt werden. [L12], [L14]

10 Resultate

Sämtliche Beispiele, Diagramme und Daten, welche in diesem Abschnitt erwähnt werden, wurden in Satteins aufgenommen.

10.1 Satteins

Das Testgebiet von "Fast and Curious" war in der Gemeinde Satteins. Diese Gemeinde liegt östlich von Feldkirch im Bundesland Vorarlberg in Österreich. Da Herr Prof. Dr. Klaus Frick in dieser Ortschaft in der Gemeindefevertretung tätig ist, wurde eine Bewilligung zur Aufstellung der Geräte ausgehändigt. Dort konnten die Geräte installiert und für mehrere Tage in Betrieb gehalten werden. In den folgenden Punkten wird das Gebiet beschrieben und anschliessend darauf eingegangen, wie die Daten erfasst wurden.

10.1.1 Topologie

Im nachfolgenden Bild (Abbildung 23) ist die Topologie von Satteins dargestellt. In dieser Darstellung sind die Aufstellpunkte der sechs Geräte mit einem schwarzen Punkt schematisch markiert. Jedes dieser Geräte wurde an den Ein- und Ausfahrtspunkten, sowie an strategisch interessanten Verkehrspunkten platziert. Dadurch sollte eine möglichst effiziente Auswertung durchgeführt werden können. Somit ist es mit diesen sechs Geräten möglich, den grössten Teil des Verkehrsflusses in der besagten Ortschaft quantitativ rekonstruieren zu können.



Abbildung 23: Topologie von Satteins. [Q2]

Mit der Hilfe des folgenden Bildes (Abbildung 24) wurden die wichtigsten Abzweigungen, welche die Verkehrsteilnehmer nehmen können, identifiziert und in der Berechnung des Verkehrsflusses beachtet. Durch den Graphen konnten auch die wichtigsten Punkte ermittelt werden, wo die Geräte aufgestellt werden sollten, um eine möglichst effiziente Verfolgung garantieren zu können. Bei diesem Verkehrsnetz beinhaltet dies, die vier wichtigen Zufahrtsstrassen zum Gebiet und ebenfalls zwei relevanten Straßen innerhalb. Daraus lässt sich der Verkehrsfluss im Graphen darstellen und im Anschluss auf die Karte von Satteins zurück projizieren.

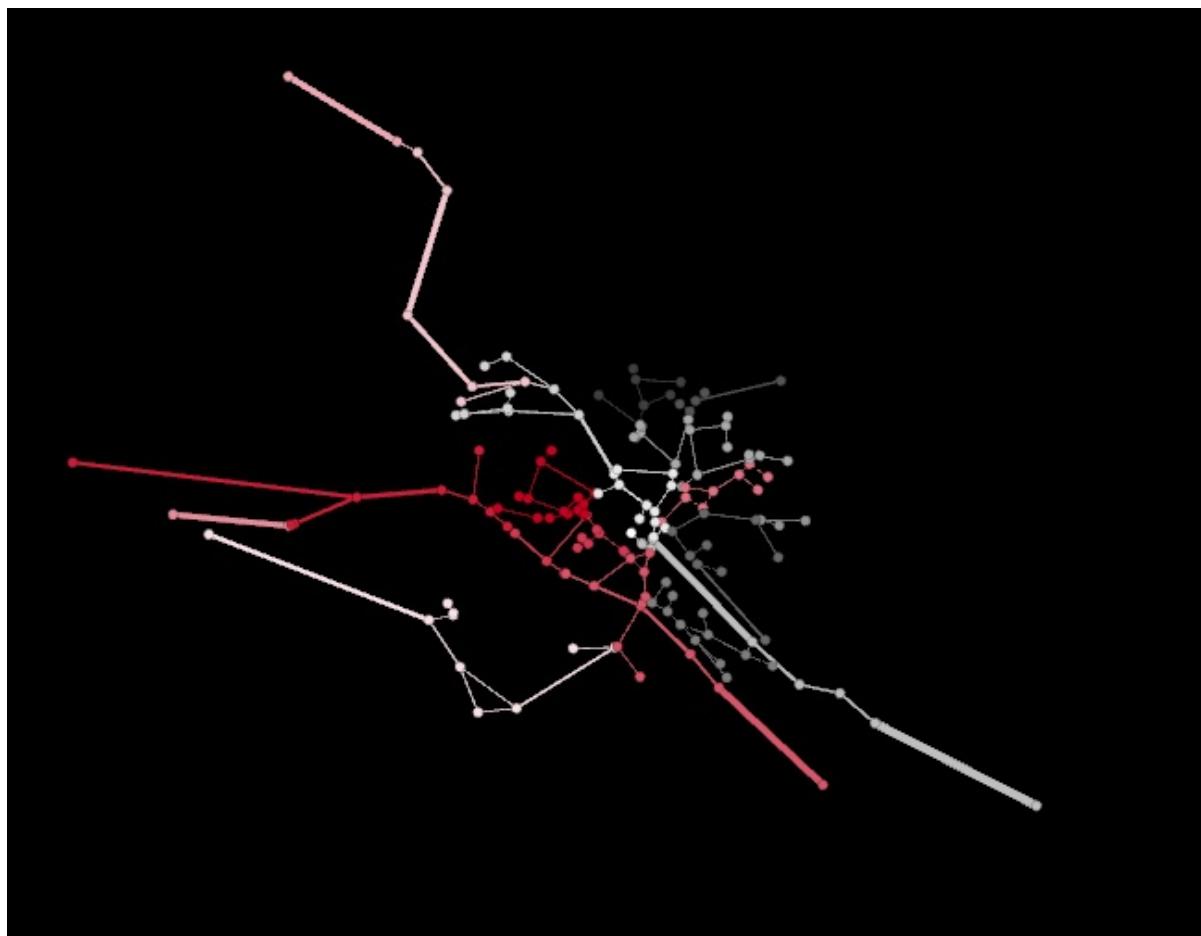


Abbildung 24: Graph des Verkehrsnetzes von Satteins.

10.1.2 Erfassung der Daten

Alle Daten, welche die Geräte aufgezeichnet haben, wurden auf den internen SD-Karten abgespeichert und jeden zweiten Tag mittels Smartphone-Hotspot auf den Laptop übertragen. Dabei wurde bei jedem einzelnen Gerät der Feature Vektor, sowie der Temperaturverlauf heruntergeladen. Die Feature Vektoren der Geräte enthielten mehrere tausend Einträge mit vorbeigefahrenen Verkehrsteilnehmern.

10.2 Einzelnes Gerät

In diesem Abschnitt werden die Ergebnisse erwähnt, welche durch ein einzelnes Gerät entstehen. Jedes dieser Geräte speichert die gesammelten Daten in einem Feature Vektor ab. Die folgende Abbildung (Abbildung 25) zeigt zehn Einträge eines solchen Feature Vektors.

Index	Timestamp	RtPi	Direction	DirCount	CropName	BlurName	PicName	Roi X	Roi Y	Roi Width	Roi Height
1	28.06.2017 13:12:06	8	L	1	crop001.tif	blur001.tif	pic001.tif	123	211	469	212
2	28.06.2017 13:12:30	8	L	2	crop002.tif	blur002.tif	pic002.tif	0	214	617	201
3	28.06.2017 13:12:33	11	L	3	crop003.tif	blur003.tif	pic003.tif	85	223	454	222
4	28.06.2017 13:12:47	5	L	4	crop004.tif	blur004.tif	pic004.tif	44	199	508	217
5	28.06.2017 13:12:49	10	L	5	crop005.tif	blur005.tif	pic005.tif	51	216	505	198
6	28.06.2017 13:13:18	9	L	6	crop006.tif	blur006.tif	pic006.tif	155	212	446	208
7	28.06.2017 13:13:36	10	R	1	crop007.tif	blur007.tif	pic007.tif	0	0	565	239
8	28.06.2017 13:13:50	11	R	2	crop008.tif	blur008.tif	pic008.tif	45	0	595	242
9	28.06.2017 13:14:00	11	L	7	crop009.tif	blur009.tif	pic009.tif	0	164	625	316
10	28.06.2017 13:16:29	5	R	3	crop010.tif	blur010.tif	pic010.tif	0	0	488	264

Abbildung 25: Feature Vektor eines einzelnen Gerätes.

10.2.1 Zählen

Durch den Einsatz eines einzelnen Gerätes kann an den Aufstellpunkten die Anzahl der vorbeifahrenden Verkehrsteilnehmer gezählt werden. An jedem dieser Punkte erhält der Bediener die Anzahl der Verkehrsteilnehmer, welche in das Gebiet hinein- und welche aus dem Gebiet heraus fahren. Ebenso enthält jedes dieser gezählten Verkehrsteilnehmer eine fortlaufende Nummer, einen Zeitstempel und die Information über dessen Fahrtrichtung.

10.2.2 Kategorisieren

Jedes Einzelgerät ist ebenso im Stande die vorbeifahrenden Verkehrsteilnehmer in vordefinierte Kategorien einzuteilen. Dabei wird die Anzahl an Pixeln, welche sich bewegen, gezählt. Dadurch kann eine quantitative Kategorisierung in drei Kategorien geschehen. Wie in Abbildung 28 in Spalte drei zu sehen ist, werden die Kategorien in "NV", "1" und "2" eingeteilt. "NV" sind Verkehrsteilnehmer, welche kleiner als ein Auto sind. Die "1" gibt an, dass es sich um ein Auto handelt. Die Nummer "2" deutet darauf hin, dass es sich um grössere Verkehrsteilnehmer handelt.

10.2.3 Geschwindigkeitserkennung

Die Geschwindigkeit der Verkehrsteilnehmer dient als Indikator, wie schnell tatsächlich an den Geräten vorbei gefahren wird. Ein Grossteil der Fahrzeuglenker reduzieren ihre Geschwindigkeit, falls sie merken, dass diese aufgezeichnet wird. Da sich "Fast and Curious" auf einer Höhe von sechs Metern befindet, werden diese Geräte von den meisten Verkehrsteilnehmern nicht erkannt, weshalb diese ohne die Geschwindigkeit zu Drosseln vorbei fahren. Aufgrund dessen kann die tatsächliche Geschwindigkeit besser errechnet werden.

Die Geschwindigkeit kann aus zwei aufeinanderfolgenden Frames extrahiert werden. Dabei wird der Positionsunterschied der Fahrzeuge auf den Frames verwendet. Um die Geschwindigkeit berechnen zu können, wurde eine Referenzmessung durchgeführt, bei welcher zwischen zwei aufeinanderfolgenden Frames, eine Verschiebung von 58 Pixeln pro km/h gemessen wurde. Nun konnte durch eine einfache Schlussrechnung die Geschwindigkeit der Verkehrsteilnehmer berechnet werden. Dabei wird der Positionsunterschied, in Pixeln, durch die Referenzmessung geteilt. Das Ergebnis dabei ist die Geschwindigkeit des Fahrzeugs. Bei höheren Fahrzeugen, wie einem LKW oder einem Traktor, wurde die Verschiebung von 58 Pixeln pro km/h auf 25 Pixel pro km/h gesenkt, da sich die Kanten dieser Fahrzeuge näher an der Kamera befinden. Auf Abbildung 28 in Spalte sechs, sind die berechneten

Geschwindigkeiten dargestellt. Mit einem frei wählbaren Farbcode wurde das Ergebnis markiert. Für eine besser Veranschaulichung wurde zudem ein Diagramm (Abbildung 26) hinzugefügt. Darauf ist ersichtlich, dass sich der grösste Teil der Verkehrsteilnehmer beinahe an die maximale Höchstgeschwindigkeit von 50 km/h im Dorf hielt. Dennoch wurden auch Geschwindigkeiten weit über der maximal tolerierten Höchstgeschwindigkeit gemessen.

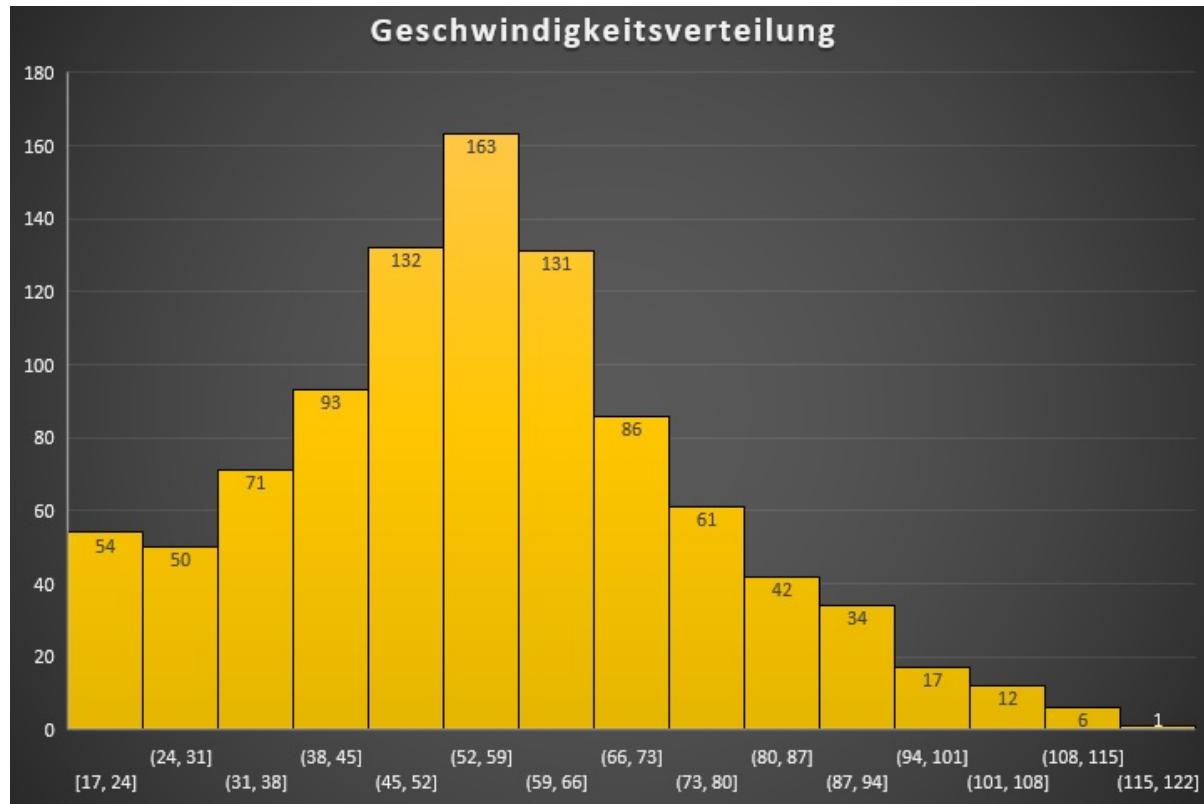
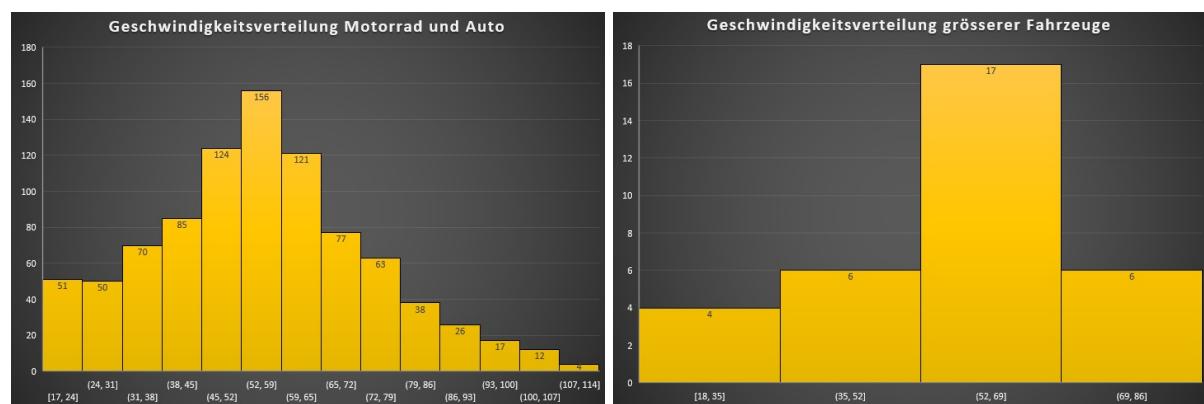


Abbildung 26: Diagramm der Geschwindigkeitsverteilung auf einer 50er Strasse am Ende eines Dorfes.



(a) Geschwindigkeitsverteilung Auto und Motorrad

(b) Geschwindigkeitsverteilung grösserer Fahrzeuge

Abbildung 27: Geschwindigkeitsverteilung, aufgetrennt in die Kategorien kleine und grosse Fahrzeuge.

cut00044.tif	36094		1	2800	2800	48
cut00045.tif	90155		1	3500	3500	60
cut00046.tif	58421		1	3075	3075	53
cut00047.tif	61716		1	3400	3400	58
cut00048.tif	57993		1	2925	2925	50
cut00049.tif	65579		1	3550	3550	61
cut00050.tif	60188		1	3275	3275	56
cut00051.tif	32139		1	2450	2450	42
cut00052.tif	12763		1	4500	4500	77
cut00053.tif	46702		1	1850	1850	32
cut00054.tif	29136	#NV		6700	6700	#NV
cut00055.tif	197318		2	3175	3175	55
cut00056.tif	43491		1	3250	3250	56
cut00057.tif	44819		1	3600	3600	62
cut00058.tif	43802		1	1925	1925	33
cut00059.tif	26387		1	4450	4450	76
cut00060.tif	49010		1	3425	3425	59
cut00061.tif	38608		1	4250	4250	73
cut00062.tif	63568		1	3075	3075	53
cut00063.tif	38341		1	2500	2500	43
cut00064.tif	34262		1	2450	2450	42

Abbildung 28: Tabelle des nachverarbeiteten Feature Vektors.

10.3 System

Ein System besteht aus mehreren Einzelgeräten, mit welchen, wie vorhin beschrieben, das Zählen, die Kategorisierung und die Geschwindigkeitsmessung durchgeführt wird. Es werden mehrere Geräte benötigt, um den Verkehrsfluss im definiert begrenzten Gebiet darstellen zu können. Dabei konnte der Verkehrsfluss mithilfe des aufgenommenen Zeitstempels und der Fahrtrichtung statistisch rekonstruiert werden. Hierbei konnte das vorhin erstellte Netzwerk des begrenzten Gebietes, als Darstellungsgrundlage verwendet werden. Zunächst wurden die nächstgelegenen Geräte, welche auf direktem Weg erreichbar sind, identifiziert und von diesen die Daten des Feature Vektors extrahiert. Diese Daten wurden dann nach dem Zeitstempel sortiert und mit einem Index versehen. Nachdem die Daten vorbereitet waren, wurde die voraussichtliche Durchfahrtszeit anhand der Abstände der Geräte anhand einer Testmessung bestimmt. Wenn die Verkehrsteilnehmer die Geschwindigkeitsbegrenzungen einhalten, dann brauchen diese die Zeiten welche unten in Tabelle 4 aufgelistet sind. Die höchste Wahrscheinlichkeit, welchen Weg der Verkehrsteilnehmer nahm, konnte aufgrund dessen berechnet und anschliessend eingezeichnet werden. Der geschilderte Vorgang konnte mit jedem Gerät durchgeführt und diese untereinander verglichen werden. Die möglichen Wege wurden dem Netzwerk ergänzt und somit konnte die folgende Auswertung (Abbildung 29) durchgeführt werden.

Fahrtweg	Zeit in Sekunden
1 nach 2	60
1 nach 3	50
2 nach 3	60
2 nach 4	100
2 nach 5	85
2 nach 6	70
4 nach 5	80
4 nach 6	150
5 nach 6	140

Tabelle 4: Fahrzeiten zwischen den Stationen.

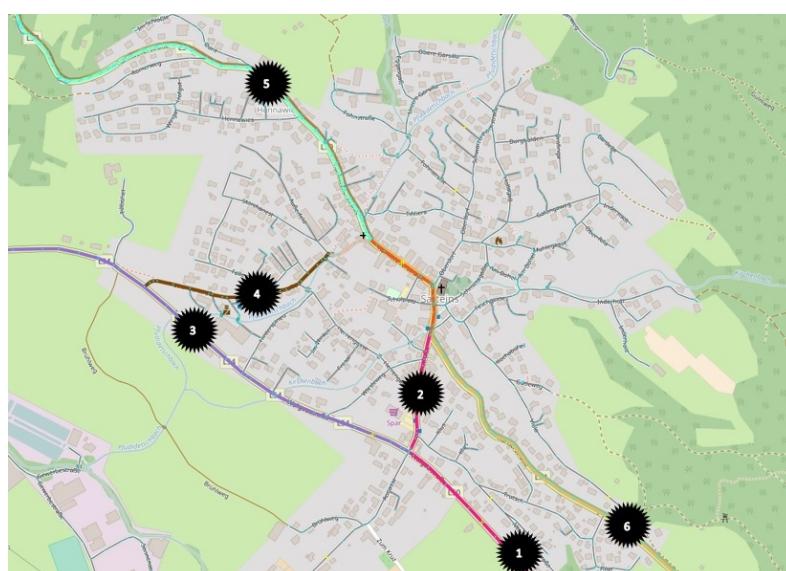


Abbildung 29: Darstellung des Verkehrsflusses in Satteins.

In Tabelle 5 ist der Verkehrsfluss von jedem Gerät zu seinem direkten Nachfolger tabellarisch dargestellt.

von/nach	1	2	3	4	5	6	Out
1		952	849				1874
2	926		679	105	834	575	
3	996	852					1719
4		149			105	84	520
5		723		159	521	224	1668
6		313		78	326		720

Tabelle 5: Tabellarische Darstellung des Verkehrsflusses.

In der nachfolgenden Tabelle (Tabelle 6) ist zu jeder Stunde, der vorbeifahrende Verkehr jedes Gerätes aufgetragen. Dabei wurden die vorbeifahrenden Fahrzeuge in folgende Fahrtrichtungen eingeteilt. "Raus" bedeutet, dass sich das Fahrzeug vom Ortszentrum weg bewegt hat. "Rein" heisst, dass es in Richtung Ortszentrum gefahrten ist.

Station	1		2		3		4		5		6	
Uhrzeit	Rein	Raus										
4-5	28	28	24	28	59	17	3	1	20	12	22	24
5-6	102	105	56	71	171	51	72	20	117	84	20	67
6-7	173	187	84	64	218	104	57	30	190	117	29	85
7-8	159	98	63	103	132	86	39	16	104	80	33	50
8-9	88	92	160	124	95	112	21	28	79	84	23	25
9-10	133	155	137	87	102	98	36	36	79	118	28	48
10-11	110	119	233	111	115	152	19	47	84	125	40	40
11-12	107	151	64	114	163	183	25	48	93	108	62	74
12-13	178	161	168	147	102	164	31	38	106	104	33	59
13-14	161	117	132	134	132	112	34	58	117	126	64	61
14-15	120	131	165	88	89	101	12	23	54	69	46	44
15-16	142	165	226	173	105	126	37	50	103	182	72	42
16-17	178	156	156	163	75	145	25	50	116	216	93	36
17-18	72	80	182	86	126	78	16	35	75	105	39	20
18-19	58	69	156	84	76	127	33	24	47	80	32	27
19-20	58	60	73	40	83	63	4	16	46	58	21	18
Total	1867	1874	2079	1617	1843	1719	464	520	1430	1668	657	720

Tabelle 6: Stündlicher Verkehr bei jeder Station.

11 Ausblick

11.1 Erweiterung des Gerätes

Da die Geräte an mehreren Standorten innerhalb einer Gemeinde platziert werden, könnte man diese Positionen nutzen, um zusätzliche Informationen zu erhalten. Dafür müsste das Gerät jedoch, je nach Information, mit zusätzlicher Hardware ausgestattet werden. Eine zusätzliche Information wäre beispielsweise der Geräuschpegel der Fahrzeuge. Ebenfalls wäre es plausibel eine Schadstoffmessung an den vorbeifahrenden Verkehrsteilnehmern durchzuführen. An den einzelnen Geräten wäre es zudem möglich Wetterstationen zu installieren.

11.2 Features erweitern

Je besser der Feature Vektor ist, desto einfacher und genauer kann die anschliessende Verkehrsverfolgung durchgeführt werden. Aus diesem Grund wäre es vorteilhaft, wenn aus den vorhandenen Bildern mehr Features zu erkennen sind. Nachfolgend sind deshalb einige Möglichkeiten aufgelistet, wie der Feature Vektor um zusätzliche Spalten erweitert werden könnte.

11.2.1 Fahrzeuglänge & -breite

Ein mögliches Feature, das realisiert werden könnte, wäre die Identifikation der Fahrzeuglänge und -breite. Im besten Falle könnte man dies direkt am vorhandenen Bild ableiten. Dazu müsste aber ein Referenzmass für beide Fahrbahnen in Sichtweite der Kamera aufgetragen werden. Das Herauslesen von Längenmassen ist jedoch auch über andere Sensoren realisierbar.

11.2.2 Kategorisieren

Ebenfalls plausibel wäre beispielsweise das Kategorisieren der Verkehrsteilnehmer in mehrere Unterklassen. Mögliche Unterteilungen sind hierbei unter anderem Fahrrad, Motorrad, PKW, Lieferwagen, Lastwagen oder Traktor. Die Unterscheidung könnte man anhand des Verhältnisses zwischen Länge und Breite und anhand der Pixelanzahl der Blobs erhalten. Die im Abschnitt Testversuche geschilderte Kategorisierung könnte dabei auf mehrere Klassen ausgeweitet werden.

11.2.3 Farbe

Die Farbe des Fahrzeugs ist ein weiteres wichtiges Merkmal, womit es erneut identifiziert werden kann. Damit dieses Feature gegenüber verschiedenen Belichtungen dennoch genügend resistent wäre, müsste dabei die erkannte Farbe in eine von etwa zehn Möglichkeiten eingeteilt werden. Die Tageszeit spielt dabei eine grosse Rolle, da eine Farberkennung unmöglich ist, wenn es draussen dunkel ist. Die Unterteilung in eine der zehn Farben wäre mithilfe des "K-Means Clustering"-Algorithmus plausibel.

11.2.4 Farbfläche

Je grösser ein Auto ist, desto mehr Farbflächen sind vorhanden. Die Anzahl der Pixel dieser Farbflächen könnte, neben der Farbe an sich, ebenfalls dem Feature Vektor ergänzt werden. Damit eine Unterscheidung zwischen Fensterscheibe und Farbfläche stattfinden kann, müsste vorgängig ein Clustering auf eine einheitliche Farbe durchgeführt werden.

11.2.5 Sonderanbringungen

Auf einigen Fahrzeugen befinden sich Sonderanbringungen wie Beschriftungen oder Bilder. Wenn diese Sonderanbringungen erkannt und bestenfalls abgespeichert werden könnten, so wäre dadurch eine zusätzliche Identifizierung dieses Verkehrsteilnehmers denkbar. Realisierbar wäre dies, wenn ein Algorithmus eine Sonderanbringung erkennt und davon ein zusätzliches Bild abspeichert.

11.2.6 Nummerntafeln

Mithilfe der Nummerntafeln sind die meisten Verkehrsteilnehmer ohne grossen Aufwand verfolgbar, da es nur wenige Verkehrsteilnehmer gibt, die keine Nummerntafel besitzen. Es könnte jedoch zum Problem werden, da die Ausrichtung der Kamera an der falschen Position ist und deshalb das Kennzeichen nicht richtig erkannt wird. Ebenso darf eine Nummerntafel aus rechtlichen Gründen eigentlich nicht ohne Weiteres aufgenommen werden, da damit ein Verkehrsteilnehmer eindeutig identifiziert werden kann.

11.2.7 Geschwindigkeit

Für eine Verkehrsverfolgung ist die Geschwindigkeit kaum relevant, da sie sich während der Fahrt kontinuierlich ändern kann. Jedoch wäre diese Information für die Gemeinden selbst nützlich. Mit der Geschwindigkeit der Verkehrsteilnehmer könnte ein Durchschnitt ermittelt werden, der als Indikator dient, wie schnell tatsächlich am Gerät vorbeigefahren wurde. Die geschilderte Geschwindigkeitserkennung im Kapitel Testversuche wäre dabei eine Variante, die verwendet werden könnte. Um jedoch genauere Daten zu erhalten, müssten bestenfalls andere Sensoren, wie Ultraschall, Radar oder Lichtschranken, verwendet werden.

11.3 Optimale Aufstellung

Um eine optimale Verkehrsverfolgung durchführen zu können, müssen die Geräte an möglichst sinnvollen Stellen innerhalb der Ortschaft platziert werden. Diese Positionen könnten mithilfe eines Algorithmus berechnet werden, welcher als Eingabe ein Strassennetz benötigt und die optimalen Standorte als Ausgabe liefert. Dafür wären zwei verschiedene Varianten denkbar. Einerseits könnte man die Anzahl an Geräten angeben und daraus dann die Aufstellungen errechnen. Andererseits könnte man, neben den Aufstellungen, auch die Anzahl der Geräte zusätzlich errechnen. Das Problem dabei wäre aber, dass man, je nach eingegebenem Strassennetz, selbst in kleineren Ortschaften eine riesige Menge an Geräten benötigen könnte.

11.4 Vernetzung

Im Moment funktioniert die Auswertung, indem die Daten vor Ort abgeholt und danach auf einem Rechner verarbeitet werden. Es wäre denkbar, dass die Geräte über ein Lora-WAN Netzwerk miteinander verbunden werden. Dadurch könnten die vorhandenen Feature Vektoren direkt übertragen werden, weshalb kein manuelles Abholen der Daten mehr notwendig ist. Das Monitoring der Geräte wäre so komplett online möglich. Zudem ist eine Überprüfung der Temperaturen und Ausrichtung der Kamera jederzeit möglich. Lediglich eine Feinjustierung der Kamera und das Auswechseln des Akkus müsste weiterhin vor Ort stattfinden.

11.5 Echtzeitdarstellung

Ausserdem ausführbar wäre eine Echtzeitdarstellung, womit ein Online-Monitoring realisierbar ist. Dabei wäre ein 24 Stunden Betrieb optimal. Jedoch ginge dies nur durch eine tiefere FPS-Zahl der Aufnahme, damit eine Nachtabbildung nicht mehr notwendig wäre, um die Bilder fertig zu prozessieren. Die aufgenommenen Daten müssten direkt auf einen Server hochgeladen und auf einer Webseite bereitgestellt werden. Die Webseite könnte, in der gleichen Art wie "Google Maps", die verwendeten Geräte innerhalb der Ortschaft mit einigen zusätzlichen Informationen anzeigen. Plausible Informationen sind hierbei beispielsweise die Anzahl der Verkehrsteilnehmer, Anzahl der links- bzw. rechtsfahrenden Fahrzeuge, Geschwindigkeit des letzten Verkehrsteilnehmers oder die momentane Kerntemperatur des Gerätes.

11.6 Spannungsversorgung

Da sich die Batterien im Moment unterhalb der Strassenlaterne befinden, könnten Unbefugte an sie gelangen und sie zerstören oder entfernen. Aus diesem Grund wäre es sinnvoll die Spannungsversorgung auf eine andere Art zu ermöglichen. Damit man die Akkumulatoren nicht ständig aufladen müsste, wäre es sinnvoll auf Photovoltaikzellen oder auf eine gemeinsame Nutzung des Stromes der Strassenlaterne zurückzugreifen.

12 Verzeichnisse

12.1 Literatur

- [L1] FriendlyARM NanoPi Team. (2014). The Feature for the NanoPi-NEO Board, Adresse: http://www.nanopi.org/NanoPi-NEO_Feature.html (besucht am 14.07.2017).
- [L2] o.V. (2017). Full HD USB Camera Module 1080P USB2.0 OV2710 Color Sensor Support MJPEG with 3.6MM Lens, Adresse: <http://www.elpcctv.com/full-hd-usb-camera-module-1080p-usb20-ov2710-color-sensor-support-mjpeg-with-36mm-lens-p-203.html> (besucht am 14.07.2017).
- [L3] Xocolatl, Siwibegewp und Gariwago. (23.04.2017). Tiefentladung, Adresse: <https://de.wikipedia.org/wiki/Tiefentladung> (besucht am 04.08.2017).
- [L4] H. Schwietering. (6.05.2017). MJPG-Streamer, Adresse: <https://wiki.ubuntuusers.de/MJPG-Streamer/> (besucht am 02.07.2017).
- [L5] L. Cons, K. Berry und O. Bachmann. (2.07.2017). Avconv Documentation, Adresse: <https://libav.org/avconv.html> (besucht am 02.07.2017).
- [L6] o.V. (8.03.2017). Apache HTTP Server, Adresse: https://wiki.archlinux.org/index.php/Apache_HTTP_Server (besucht am 02.07.2017).
- [L7] OpenCV team. (14.06.2017). OpenCV About, Adresse: <http://opencv.org/about.html> (besucht am 02.07.2017).
- [L8] C. Dahms. (20.02.2016). OpenCV_3_Car_Counting_Cpp, Adresse: https://github.com/MicrocontrollersAndMore/OpenCV_3_Car_Counting_Cpp (besucht am 04.07.2017).
- [L9] kegelkugel. (6.10.2016). Bash-Skripting-Guide für Anfänger, Adresse: [https://wiki.ubuntuusers.de/Shell/Bash-Skripting-Guide_für_Anfänger/](https://wiki.ubuntuusers.de/Shell/Bash-Skripting-Guide_f%C3%BCr_Anf%C3%A4nger/) (besucht am 04.07.2017).
- [L10] D. Kirkland. (2010). streamer - record audio and/or video, Adresse: <http://manpages.ubuntu.com/manpages/precise/man1/streamer.1.html#contenttoc0> (besucht am 07.07.2017).
- [L11] The FFmpeg developers. (6.07.2017). ffmpeg Documentation, Adresse: <https://ffmpeg.org/ffmpeg.html> (besucht am 07.07.2017).
- [L12] o.V. (23.12.2016). Interactive Foreground Extraction using GrabCut Algorithm, Adresse: http://docs.opencv.org/3.2.0/d8/d83/tutorial_py_grabcut.html (besucht am 04.07.2017).
- [L13] —, (23.12.2016). K-Means Clustering in OpenCV, Adresse: http://docs.opencv.org/3.2.0/d1/d5c/tutorial_py_kmeans_opencv.html (besucht am 07.07.2017).
- [L14] —, (23.12.2016). Operations on arrays, Adresse: http://docs.opencv.org/3.2.0/d2/de8/group__core__array.html (besucht am 10.07.2017).
- [L15] Magic links bot, Dopexxx und DePiep. (5.07.2017). Blob detection, Adresse: https://en.wikipedia.org/wiki/Blob_detection (besucht am 26.07.2017).
- [L16] Niederrheiner81, Aka und KnightMove. (13.06.2017). Cluster (Datenanalyse), Adresse: [https://de.wikipedia.org/wiki/Cluster_\(Datenanalyse\)](https://de.wikipedia.org/wiki/Cluster_(Datenanalyse)) (besucht am 26.07.2017).
- [L17] Vienna University. (27.11.2016). Eduroam, Adresse: <https://www.aco.net/eduroam.html> (besucht am 04.08.2017).
- [L18] Singsangsung, MyContribution und L. Cogito. (30.11.2016). Merkmalsvektor, Adresse: <https://de.wikipedia.org/wiki/Merkmalsvektor> (besucht am 26.07.2017).
- [L19] Schnabeltassentier, F. Stember und Ohrnwurzler. (16.05.2016). Einzelbild (Film), Adresse: [https://de.wikipedia.org/wiki/Einzelbild_\(Film\)](https://de.wikipedia.org/wiki/Einzelbild_(Film)) (besucht am 26.07.2017).

- [L20] Saure, Stobaios und G. Hügler. (19.05.2017). Bildfrequenz, Adresse: <https://de.wikipedia.org/wiki/Bildfrequenz> (besucht am 26.07.2017).
- [L21] Aka, E. Rashid und Jbergner. (5.07.2017). Git, Adresse: <https://de.wikipedia.org/wiki/Git> (besucht am 26.07.2017).
- [L22] Invisigoth67, G. Hügler und DaDoKa. (25.06.2017). Monitoring, Adresse: <https://de.wikipedia.org/wiki/Monitoring> (besucht am 26.07.2017).
- [L23] J. Grote, Messerjokke79 und Maelcum. (11.07.2017). MySQL, Adresse: <https://de.wikipedia.org/wiki/MySQL> (besucht am 26.07.2017).
- [L24] Ogrk, Distelfinck und Minihaa. (3.07.2017). Open Source, Adresse: https://de.wikipedia.org/wiki/Open_Source (besucht am 26.07.2017).
- [L25] Zinnmann, Gr1 und Wiki-vr.mp. (24.05.2017). OpenStreetMap, Adresse: <https://de.wikipedia.org/wiki/OpenStreetMap> (besucht am 26.07.2017).
- [L26] H. Gräßner, Messerjokke79 und Trustable. (13.06.2017). Plug-in, Adresse: <https://de.wikipedia.org/wiki/Plug-in> (besucht am 26.07.2017).
- [L27] Fomafix, Trustable und Chotaire. (8.07.2017). PuTTY, Adresse: <https://de.wikipedia.org/wiki/PuTTY> (besucht am 26.07.2017).
- [L28] ClaudiaSbg, Tzeh und Trudels. (21.06.2017). Repository, Adresse: <https://de.wikipedia.org/wiki/Repository> (besucht am 26.07.2017).
- [L29] o.V. (26.07.2017). SourceTree, Adresse: <https://www.sourcetreeapp.com/> (besucht am 26.07.2017).

12.2 Quellen

- [Q1] "NanoPi NEO Layout", Adresse: <https://1.f.ix.de/scale/geometry/680x453/q75/imgs/71/1/8/5/3/6/8/6/NanoPi-NEO-layout-bd228443bd85b402.jpg> (besucht am 14.07.2017).
- [Q2] "OpenStreetMap", Adresse: <https://www.openstreetmap.org/search?query=satteins#map=14/47.2276/9.6738> (besucht am 27.07.2017).

12.3 Tabellen

1	Kostenauflistung	12
2	Feature Vektor	19
3	Ordnerstruktur	25
4	Fahrzeiten zwischen den Stationen	44
5	Tabellarische Darstellung des Verkehrsflusses	45
6	Stündlicher Verkehr bei jeder Station.	45

12.4 Abbildungen

1	Layout des NanoPi NEO. [Q1]	8
2	Skizze zur Berechnung der Kameradaten.	9
3	Foto eines leeren Gehäuses.	11
4	Gesamtsystem.	12
5	Einsatzfähiges Gesamtsystem.	13
6	Aufteilung der vier Kerne.	15
7	Flussdiagramm der Videoaufnahme.	16
8	Zwei aufeinanderfolgende Frames.	17
9	Differenzbild der aufeinanderfolgenden Frames.	17
10	Flussdiagramm der Vorverarbeitung.	18
11	Vehicle Tracking mithilfe von Blobs.	21
12	Flussdiagramm der Nachverarbeitung.	22
13	Flussdiagramm der Steuerung und Überwachung.	24
14	Webseite von "Fast and Curious".	31
15	Temperaturverlauf eines NanoPi.	32
16	Eingesetzte Prototypen.	33
17	Differenzbild zur Demonstration des Schattens.	35
18	Differenzbild mit entferntem Schatten.	35
19	Beispiel des "GrabCut"-Algorithmus.	36
20	Beispiel K-Means Clustering.	37
21	Beispiel zur Kategorisierung der Verkehrsteilnehmer.	37
22	Beispiel zur Geschwindigkeitsermittlung eines Verkehrsteilnehmers.	38
23	Topologie von Satteins. [Q2]	39
24	Graph des Verkehrsnetzes von Satteins.	40
25	Feature Vektor eines einzelnen Gerätes.	41
26	Diagramm der Geschwindigkeitsverteilung auf einer 50er Strasse am Ende eines Dorfes.	42
27	Geschwindigkeitsverteilung, aufgetrennt in die Kategorien kleine und grosse Fahrzeuge.	42

28 Tabelle des nachverarbeiteten Feature Vektors.	43
29 Darstellung des Verkehrsflusses in Satteins.	44

12.5 Abkürzungen

AVI	Audio Video Interleave
BASH	Bourne-again shell
BLOB	Binary Large OBject
FPS	Frames per Second
HDMI	High Definition Multimedia Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LoRa-WAN	Long Range Wide Area Network
OpenCV	Open Computer Vision
PHP	Hypertext Preprocessor
SD	Secure Digital
SSID	Service Set Identifier
USB	Universal Serial Bus
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network

12.6 Glossar

Blob detection	In der Bildverarbeitung benutze Methode, um ein Objekt mit konstanten Eigenschaften zu detektieren. [L15].
Blob tracking	Benutzte Methode in der Bildverarbeitung, um ein detektiertes Objekt über mehrere Bilder hinweg zu verfolgen.
Cluster	Ein Cluster definiert eine Gruppe von Objekten, welche ähnliche Eigenschaften aufweisen. [L16].
Eduroam	Eduroam steht für “education roaming”. Mithilfe von Eduroam kann mit einem Benutzer einer teilnehmenden Organisation auf jedes WLAN-Netzwerk anderer teilnehmender Organisationen beigetreten werden. [L17].
Feature Vektor	Ein Feature Vektor umfasst die parametrisierbaren Eigenschaften eines Musters in vektorieller Weise. Verschiedene, für das Muster charakteristische Merkmale bilden die verschiedenen Einträge dieses Vektors. [L18].
Frame	Ein Frame bezeichnet eine Einzelbild, welches aus einem Video extrahiert wurde. [L19].
Framerate	Die Framerate bezeichnet die Anzahl der Einzelbilder pro Zeiteinheit, welche aufgenommen und wiedergegeben werden. [L20].
Git	Freie Software zur verteilten Versionsverwaltung von Dateien. [L21].
Monitoring	Als Monitoring bezeichnet man das Messen, Überwachen und Beobachten eines Prozesses. [L22].
MySQL	MySQL ist ein Open Source Tool für relationalen Datenbanken. Es ist für mehrere Betriebssysteme verfügbar und beinhaltet zudem eine Online-Schnittstelle. [L23].
Open Source Tool	Open Source Tool definiert öffentlichen Quelltext von Software, welche kostenlos eingesehen, genutzt oder geändert werden kann. [L24].
Open Street Map	Open Street Map ist ein freies Projekt, welches dazu dient, frei nutzbare Geodaten für jeden frei zu Verfügung zu stellen. [L25].
Plugin	Es handelt sich dabei um optionale Software-Module, welche die Grundfunktionen der Software erweitern. [L26].
PuTTY	PuTTY ist eine freie Software die verwendet wird, um über verschiedene Schnittstellen auf andere Geräte und Objekte zuzugreifen. [L27].
Repository	Ein Repository ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten. [L28].
SoftAP-Mode	Software aktivierter Verbindungspunkt, ohne dabei in einem Netzwerk eingebunden zu sein.
SourceTree	SourceTree ist ein Programm für Windows, mit welchem es ermöglicht wird, einfachen Zugriff auf ein Git-Repository herzustellen. [L29].

13 Eidestattliche Erklärung

Hiermit versichern wir, dass wir die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Quellen verfasst haben.

Ort, Datum

Josef Böckle

Daniel Lüchinger

14 Anhang

USB-Stick:

- Verwendetes Ubuntu Image
- Neuste Software des Quellcodes
- Dokumentation der Software mit Hilfe von Doxygen