Technische Universität München

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Computergestützte Modellierung und Simulation

Datenanalyse in der Cloud – Entwicklung eines Prototyps zur Automatisierung von Baudokumentation und Baufortschrittskontrolle mit den Prinzipien des Internet of Things

Masterthesis

für den Master of Science Studiengang Bauingenieurwesen

Autor:	Michael Holland
Matrikelnummer:	[redacted]
1. Betreuer:	Prof. Dr.-Ing. André Borrmann
2. Betreuer:	M.Sc. Felix Eickeler

Ausgabedatum:	01. Januar 2019
Abgabedatum:	01. Juli 2019

Abstract

The goal of the thesis is the development of a tool for supporting the documentation and progress control of construction sites. Currently, this is conducted mainly by manual means and there is a demand for automated instruments. Furthermore, the construction industry needs to follow the trend of digitalization in order to increase its productivity in the future. The prototype which was developed in the scope of this thesis consists of a camera system in the form of Rasperry Pis, a cloud infrastructure and a web application as user interface. It enables the determination of the position of cameras and the tracking of marked objects on a construction site by using picture and sensory data. For this, a neuronal network analyzes the created picture data. The prototype was tested in a representative test. The results of the test show that the cloud infrastructure and the classification by the neural network work reliably and therefore the implemented functionalities could be performed successfully. For an industrially mature application, the gained sensor data should be optimized.

Zusammenfassung

Das Ziel der Arbeit ist die Entwicklung eines Tools zur Unterstützung der Baudokumentation und Fortschrittskontrolle. Aktuell wird dies überwiegend manuell durchgeführt und es gibt einen Bedarf nach automatisierten Hilfsmitteln dafür. Außerdem muss die Baubranche dem Trend der Digitalisierung folgen, um in den nächsten Jahren ihre Produktivität steigern zu können. Der Prototyp, der im Rahmen dieser Arbeit entwickelt wurde, besteht aus einem Kamerasystem in Form von mehreren Raspberry Pis, einer in der Cloud entwickelten Infrastruktur und einer Webanwendung als Nutzerschnittstelle. Er ermöglicht die Bestimmung der Position von Kameras und das Tracking von markierten Gegenständen auf einer Baustelle mithilfe von Bild- und Sensordaten. Ein neuronales Netzwerk analysiert dafür die erzeugten Bilddaten. Der Prototyp wurde durch einen Versuchsaufbau getestet. Die Ergebnisse des Versuchs zeigen, dass die Cloud-Infrastruktur und die Klassifizierung der markierten Gegenstände durch das neuronale Netzwerk zuverlässig funktionieren und damit die entwickelten Funktionalitäten erfolgreich durchgeführt werden können. Für die industriereife Anwendung sollten aber noch die gewonnenen Sensordaten optimiert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BIM	Building Information Modelling
BLOB	Binary Large Objects
C2D	Cloud to Device
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CSI	Camera Serial Interface
CSS	Cascading Style Sheets
D2C	Device to Cloud
Exif	Exchangeable Image File Format
GNR	Gierwinkel, Nickwinkel, Rollwinkel
GPIO	General Purpose Input and Output
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
IMU	Internal Measurement Unit
IoT	Internet of Things
JS	JavaScript
JSON	JavaScript Object Notation
LSVRC	Large scale visual recognition challenge
mAP	Mean Average Precision
NN	Neural Networks
ORM	Object-relational mapper
R-CNN	Regional Convolutional Neural Network

RFID	Radio Frequency Identification
RPN	Region Proposal Network
RTC	Real-time Clock
SAS	Shared Access Signature
SBC	Single Board Computer
SDK	Software Development Kit
SOA	Service-oriented Architecture
SQL	Structured Query Language
SQLIA	Structured Query Language Injection Attack
WSGI	Web Server Gateway Interface

1 Einführung

In der Entwicklung der Bauindustrie in den letzten Jahren ging es viel um die Schlagwörter „Building Information Modeling“, „Baustelle 4.0“, oder „Smart Building Site“. Die Themen, die hinter diesen Schlagwörtern stecken, fallen alle unter den Bereich der Digitalisierung. Dies bedeutet den Wandel hin zu transparenten, vernetzen und modernen Arbeitsweisen und Organisationsformen. In dieser Arbeit wird ein Aspekt davon, die Automatisierung von Baudokumentation, konzeptioniert und anhand eines Prototyps getestet. In der Einleitung wird die Motivation der Arbeit und der Kontext der Entwicklung von Technologien für die Baubranche beschrieben. Das Ziel der Arbeit, die Entwicklung des Prototyps, wird hervorgehoben. Am Ende dieses Kapitels wird der Aufbau der ganzen Arbeit dargelegt.

1.1 Motivation

Statistiken belegen, dass die Baubranche Veränderungen durchlaufen muss, denn seit 2010 konnte die Bauindustrie in Deutschland einen Produktivitätszuwachs von lediglich 2,8% verzeichnen. Zum Vergleich: das entspricht bei allen anderen Geschäftsfeldern der durchschnittlichen Produktivitätssteigerung pro Jahr (PwC 2018).

Die Gründe hierfür sind vielfältig. Produktions- und Planungsprozesse der klassischen Produkterzeugung können nicht direkt auf den Bausektor übertragen werden, denn jedes Bauwerk ist ein Unikat. Kalkulationen, Termin- und Ablaufplanungen, Produktionsplanungen, Baulogistik, Geräte und Personalplanung müssen für jedes Bauwerk weitgehend neu bemessen werden (Zimmermann 2010). Eine Produktionssteigerung durch Standardisierung, die einen großen Einfluss auf die Produktivität in den verarbeitenden Gewerben hatte, ist dementsprechend nur in Teilbereichen möglich.

Die Ursache für mangelnde Produktivität nur auf einen einzigen Faktor zu reduzieren wäre folglich eine nicht zulässige Vereinfachung. Dennoch legt die Korrelation zwischen Investitionen in digitale Technologien und Produktivitätssteigerung den Schluss nahe, dass die Handlungsstrategien der letzten Jahrzehnte ihren Teil zur Stagnation der Wertschöpfung in der Bauindustrie beigetragen haben. Dies bestätigt eine durch McKinsey durchgeführte Studie, in deren Rahmen der Digitalisierungsgrad, ein kombinierter Index, der sich aus mehreren Indikatoren zusammensetzt und den Einsatz

von digitalen Technologien bewertet, branchenübergreifend mit der jährlichen Produktivitätssteigerung verglichen wurde (McKinsey Global Institute 2017). Im europäischen Vergleich weist die Baubranche sowohl den geringsten Zuwachs an Produktivität als auch den niedrigsten Digitalisierungsgrad auf.

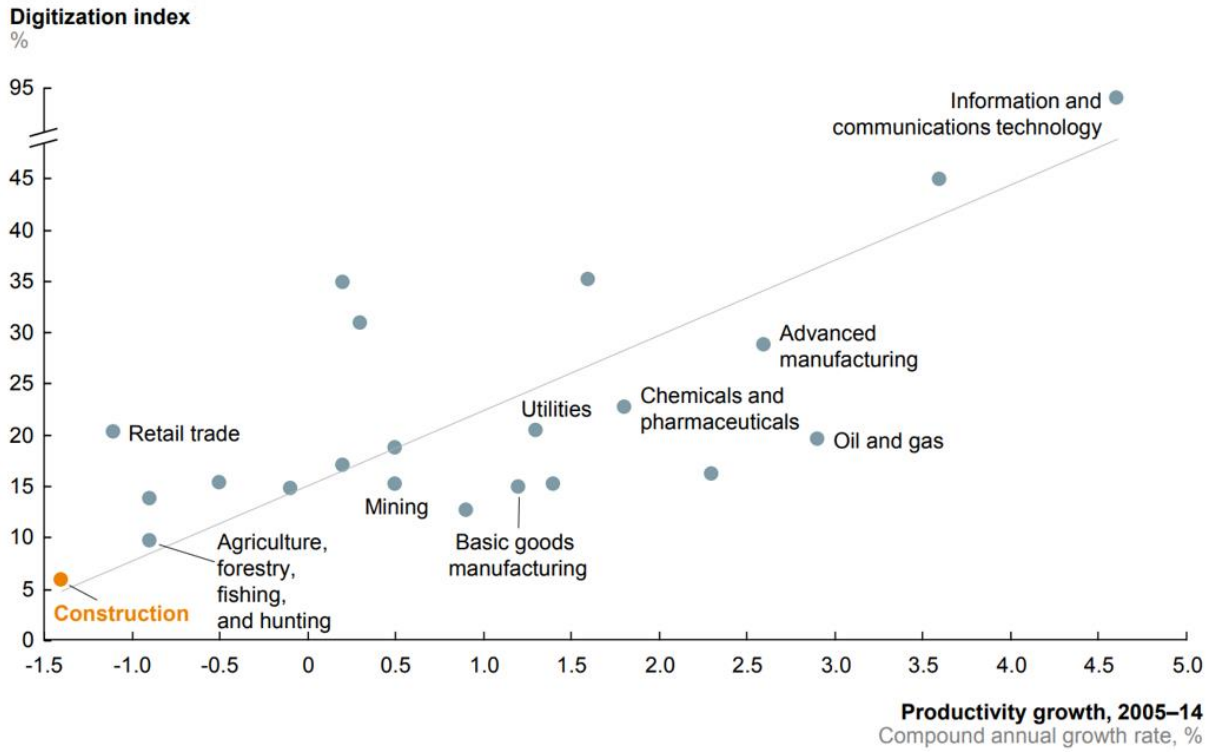


Abbildung 1.1 Zusammenhang zwischen Digitalisierungsindex und Produktivitätszuwachs für verschiedene Geschäftsfelder (McKinsey Global Institute 2017)

1.2 Bestehende Technologielösungen

Die beschriebene Problematik ist von der Industrie nicht unbemerkt geblieben. Seit mehreren Jahren bemühen sich Softwarehersteller, Vereine und Arbeitsgruppen eine gemeinsame Leitlinie bezüglich der Integration der „neuen“ Technologien zu finden und projektunterstützende Software möglichst sinnvoll in das System „Bau“ zu integrieren (VDI 2552; buildingSMART e.V.). Durch diese Bemühungen sind bereits unterschiedliche Tool-Sets zur Unterstützung der Bauüberwachung bei Dokumentation und Protokollierung auf den Markt vorgedrungen.

Das Spektrum der angebotenen Lösungen ist vielfältig. Der gängige Ansatz beinhaltet die Integration der Mängel- und Bilderverwaltung in die Projektmanagement Software. Hier können Bilder hochgeladen werden und in vorgefertigten Templates als Dokumentation ausgegeben und verschickt werden (Mängelmanagement: Software für das

Bauwesen | think project!; Vernetzt und transparent arbeiten mit BIM4You Office; Baudokumentation smart & easy - Bautagebuch - Software & App). Die Bilder werden entweder direkt auf der Baustelle mit einem mobilen Endgerät aufgenommen und auf die Plattform geladen, oder während der Baubegehung aufgenommen und bei der Nachbereitung in das System eingepflegt. Einen anderen Ansatz verfolgt die Firma Matterport: zu dokumentierende Bereiche der Baustelle werden mit Hilfe von 3D-Scannern erfasst und als Punktwolke archiviert. Im Post-Processing kann diese dann zu 3D-Objekten trianguliert, als Lageplan dargestellt, oder nach Nutzerwünschen für Dokumentationszwecke visualisiert werden. Im Unterschied zur bloßen Fotodokumentation können mit dieser Herangehensweise auch geometrische Informationen aus den Aufnahmen extrahiert werden (Matterport).

Alle Ansätze verbindet jedoch die Tatsache, dass sich geschultes Personal auf der Baustelle befinden muss, um aussagekräftige Bilder zu erzeugen und diese den anderen Projektbeteiligten zur Verfügung zu stellen. Dies erfordert Planung und Zeit. Wenn neben einfacher Fotodokumentation geometrische Parameter erfasst werden sollen, muss der Nutzer zusätzliche Investitionen hinsichtlich entsprechender Gerätschaften und Post-Processing tätigen.

1.3 Ziel der Arbeit

Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, der die oben adressierte Problematik angeht. Hierfür wurde unter den Aspekten der Wirtschaftlichkeit und der Zuverlässigkeit ein auf Baustellen integrierbares Konzept entworfen und umgesetzt, das es möglich machen soll, Aspekte der Baudokumentation zu automatisieren und nicht die physische Anwesenheit der Projektbeteiligten auf der Baustelle voraussetzt.

Die in den letzten Jahren stark vorangetriebene Entwicklung des Konzepts „Internet of Things“ (IoT) hat bereits robuste Lösungen zur Anbindung von leistungsschwachen Endgeräten an Cloud-Lösungen hervorgebracht. Daher wurde der Prototyp mit Hilfe der Paradigmen des IoT entworfen, um dessen bewährte Strukturen auf bauspezifische Projekte zu übertragen.

Der Prototyp lässt sich dabei in drei Teilbereiche unterteilen. Zum einen sind dies Kameramodule, die an Raspberry Pis (vgl. Device Hardware) befestigt sind und zusammen mit inertialen Messeinheiten (engl. Internal Measurement Unit; IMU) auf der Baustelle verteilt werden können und Bild- und Sensordaten liefern. Die Module können

dabei stationär oder mobil, beispielsweise an Kranauslegern oder Baumaschinen, befestigt werden. Im Rahmen dieser Arbeit wurden die notwendigen Routinen zur Bildaufnahme, Kommunikation mit der Cloud und Orientierungsbestimmung mithilfe des IMU erarbeitet.

Der zweite Teilbereich beschäftigt sich mit der Übertragung und Organisation der erzeugten Daten. Hierfür wurde eine Cloud-Lösung ausgewählt und eine Datenbank entworfen. Die in der Arbeit vorgestellte Implementierung zeigt, wie Cloudservices genutzt werden können, um die von den Endgeräten übermittelten Daten zuverlässig zu verarbeiten und in persistente Speicher zu überführen.

Der Nutzen des Systems wird schlussendlich im dritten Teilbereich, einer Webanwendung, realisiert. Die Webanwendung stellt die Nutzeroberfläche des Dokumentationstools dar. Hier wird die Kameraverwaltung vorgenommen und übertragene Daten visualisiert. Das Anwendungsbackend der Weboberfläche verfügt dabei über die Möglichkeit, Objekte auf den Bildern in der Datenbank zu erkennen und in Kombination mit den Daten des IMU Rückschlüsse auf Inhalt und Position sowohl der aufnehmenden Kamera als auch der erkannten Objekte zu ziehen.

Dabei wurden zwei Anwendungsfälle ins Auge gefasst: Zum einen soll das System Objekte verorten können. Somit hat der Nutzer die Möglichkeit, Fortschritte bei der Bauabwicklung festzustellen. Denkbar sind Szenarien, bei denen der Baufortschritt maßgeblich von geometrischen Veränderungen des Gebäudes abhängt, wie es beispielsweise beim klassischen Hochbau während des Rohbaus der Fall ist. Gebäudedimensionen können so einfach bestimmt werden. Außerdem soll es dem Nutzer ermöglicht werden, bestimmte Objekte über die Kameraverwaltung „tracken“ zu können. Dies ist sinnvoll, wenn der Nutzer Interesse daran hat, bestimmte Bilddaten nach Themen zu filtern, oder die Bewegungen eines Objekts auf der Baustelle nachzuvollziehen. So können beispielsweise Bewegungsprofile von Baumaschinen, Personen oder Materialien erstellt werden.

Die zu erfassenden Objekte werden durch individuelle Trackingmarker repräsentiert. Jeder Marker kann datenbankseitig einem bestimmten Objekt auf der Baustelle zugeordnet werden. Die Trackingmarker werden auf den von den Endgeräten bereitgestellten Bilddaten durch ein neuronales Netzwerk erkannt. Um im Anschluss die Position der erkannten Marker bestimmen zu können, wurden die mathematischen Zusammenhänge zwischen dreidimensionalen Objekt und dessen Abbildung auf einer Bildebene

anhand des „Lochkameramodells“ erarbeitet und in Form eines Least-Squares-Optimierungsproblems gelöst.

Der zweite Anwendungsfall ist die autonome Bestimmung des Standortes der installierten Kameras über optische Referenzpunkte. Dadurch können auch Kameras, die auf beweglichen Objekten installiert sind, Informationen zur aktuellen Situation auf der Baustelle geben, da ihre Aufnahmen einem eindeutigen Aufnahmestandort zugeordnet sind. Somit hat der Nutzer Zugriff auf eine Vielzahl von eindeutig positionierten Bilddaten, die für Dokumentationszwecke eingesetzt werden können. Zudem müssen Kameras, die sich in unwegsamem Gelände, großer Höhe, oder unzugänglichen Bereichen befinden, nicht extra eingemessen werden, um im Nachgang als Ausgangspunkt für Trackingalgorithmen genutzt zu werden.

Hierfür wurden abermals optische Marker eingesetzt. Im Gegensatz zur Objektverfolgung sind diesen aber eindeutige 3D-Echtweltkoordinaten zugeordnet. Analog zum oben beschriebenen „Tracking“, werden die Referenzpunkte mithilfe des neuronalen Netzwerkes erkannt. Die Positionsbestimmung der Kamera wurde abermals mit der Lösung eines Optimierungsproblems bewältigt.

Im Kontext dieser Arbeit wird der Begriff „Thing Site“ stellvertretend für den Prototypen und all seine Komponenten gebraucht. Der Begriff setzt sich aus „Thing“, der Geräteebene des IoT, und „Site“, der englischen Übersetzung für Baustelle, Construction Site, zusammen.

1.4 Aufbau der Arbeit

In Kapitel 2 werden zunächst die Grundlagen des IoT vermittelt und der Prototyp in dessen Kontext eingeordnet. Dabei wird insbesondere herausgearbeitet, welche Stellung Cloud-Lösungen im Zusammenhang mit IoT einnehmen und welche Technologien für den Prototyp eingesetzt werden können.

In Kapitel 3 wird auf die technischen Grundlagen eingegangen, die für die Umsetzung relevant waren. Dazu zählen die Auswahl eines geeigneten Cloud-Providers und die Vorstellung von dessen Kernkomponenten, ein Diskurs über Datenhaltung, die Hintergründe zur Objekterkennung und schließlich die eingesetzte Hardware.

Kapitel 4 stellt die mathematischen Problemstellungen vor, die für die Umsetzung bewältigt werden mussten. Hier wird insbesondere darauf eingegangen, wie dreidimen-

sionale Objekte auf Kamerabildern abgebildet werden und im Umkehrschluss Annahmen auf die Position von erkannten Objekten oder der Kamera getroffen werden können. Zudem wird gezeigt, wie aus den Sensordaten des IMU die Orientierung der Kamera bestimmt werden kann.

Die Fusion der herausgearbeiteten theoretischen Grundlagen in die praktische Umsetzung folgt in Kapitel 5. Dabei werden drei Ebenen beschrieben. Für die Ebene „Device“, d.h. Endgeräte, die auf der Baustelle verteilt werden, werden Routinen und Konzepte für Datenübertragung, Datenhaltung und Orientierungsermittlung erläutert. Die Ebene „Network“ beschäftigt sich mit der Organisation und Speicherung der übertragenen Daten. Im Rahmen der Ebene „Application“ wird auf die Weboberfläche und deren Funktionen eingegangen.

In Kapitel 6 wird schließlich die Validierung des Prototyps anhand eines Versuchsaufbaus durchgeführt. Hierfür wurden zwei Testszenarien ausgewählt und sowohl die Funktionalitäten der Anwendung demonstriert, als auch räumliche Toleranzen in Bezug auf Positionierungsalgorithmen bestimmt.

Die Bewertung der erhobenen Daten und des Konzepts erfolgt abschließend in Kapitel 7. Anhand dieser werden Chancen und Möglichkeiten für zukünftige Projekte aufgezeigt.

2 Stand der Technik

Dieses Kapitel beschreibt das generelle Konzept IoT und seine technologischen Grundlagen. Das zu erarbeitende Konzept wird in dessen Kosmos eingeordnet. Dies schafft die Basis für die in Kapitel 3 erfolgte Auswahl der IoT-Lösung.

2.1 Das Internet der Dinge

Erstmalig wurde der Begriff „Internet of Things“ im Jahre 1999 im Zusammenhang mit RFID-Tags (Radio Frequency Identification) und Supply-Chain-Management verwendet. Ziel davon war es, die innerbetriebliche Koordination von Fertigungsprozessen zu implementieren (Kevin Ashton 2016). Fast 20 Jahre später, begünstigt durch neue Telekommunikationsstandards, wirtschaftlichere Preise in der Halbleiterindustrie, weltweit verfügbare Internetdienste und optimierte kabellose Sensornetzwerke fällt darunter ein ganzer Industriezweig mit unterschiedlichsten Anwendungsmöglichkeiten (Ray 2016) und ca. 27 Milliarden IoT-fähigen Einheiten (IoT: number of connected devices worldwide 2012-2025 | Statista). Jede dieser Einheiten ist eindeutig identifizierbar und kommuniziert über ein Netzwerk mit der physischen oder digitalen Welt.

Ein für den privaten Nutzer gebräuchliches und häufig rezitiertes Beispiel stellt hier das „Smart Home“ dar. Sensordaten und Geräte werden zentral verwaltet und gesteuert. Dadurch sollen Komfort, Energieeffizienz und Sicherheit erhöht werden. Gewerbliche Einsatzgebiete finden sich beispielsweise in den Bereichen der Logistik, Automation, Transportwesen.

Beim „Internet der Dinge“ handelt es sich also vielmehr um ein Konzept mit verschiedenen Auslegungen und Interpretationen, als um einen scharf definierten Begriff (Atzori et al. 2010). Das gemeinsame Paradigma ist jedoch: IoT-Systeme bestehen aus physischen und virtuellen Objekten, die Informationen über öffentliche oder private Netzwerke teilen, um ein gemeinsames Ziel zu erreichen. „Dinge“ können dabei jede Form von Objekt darstellen, die über das Internet adressierbar, erkennbar, lokalisierbar oder kontrollierbar sind (Yelizavet und Florentino 2019).

2.2 Drei-Schichten-Architektur

Je nach Definition lässt sich das IoT in unterschiedlich viele Wirkungsebenen untergliedern (Muntjir et al. 2017). Für unseren Anwendungsfall wurde eine vereinfachte

Darstellung in „Device“, „Network“ und „Application“ vollzogen. Mit dieser Form der Abgrenzung können die meisten IoT Systeme abgebildet werden (Guth et al. 2018). Eine Einordung des „Thing Site“ begünstigt die Auswahl eines geeigneten Frameworks.

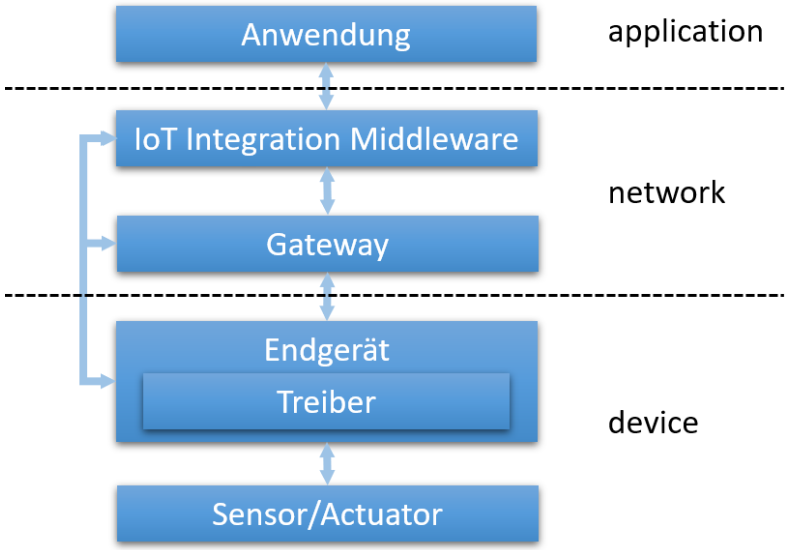


Abbildung 2.1 Wirkungsebenen des IoT und enthaltene Komponenten (eigene Grafik, angelehnt an Guth et al. 2018)

2.2.1 Device

Die Sensor- und Aktuatorenebene und stellt das hierarchisch niedrigste Niveau in jedem IoT System dar. Ihre Aufgabe ist es, Daten mithilfe von Sensoren, RFID-tags, Barcodes o.Ä. zu sammeln, zu generieren oder bereitzustellen und gemessene Informationen in digitale Signale umzuwandeln (ICITST et al. 2015). Mit Hilfe von Aktuatoren manipuliert das IoT System, induziert durch vom Gerät gesendete Signale, seine physische Umgebung.

Ein weiterer Teil des Layer „Device“ ist die Verbindung des Endgerätes mit den beschriebenen Sensoren. Dadurch können Messwerte aufbereitet, zwischengespeichert oder weitergeleitet werden.

Für den Prototypen von „Thing Site“ werden am Raspberry Pi im Rahmen der Masterarbeit vorerst vier Typen von Sensoren verbaut. Dazu gehören ein Bildsensor, der den Bautenstand dokumentiert, ein Accelerometer, ein Magnetometer und ein Gyroskop.

2.2.2 Network

Das „Network“ ist verantwortlich für die Verarbeitung und Übertragung der unter „Device“ erzeugten Daten. Dies umfasst sowohl die Kommunikation der einzelnen Geräte

untereinander als auch den Transfer von Informationen auf geteilte Speicherressourcen. Die Übertragung erfolgt in der Regel kabellos über die gängigen Übertragungsstandards FTTx, 3G/4G, Wifi, Bluetooth, Zigbee, UMB, Infrarot, LoRaWAN o.Ä.
„Smart Gateways“ sind verantwortlich dafür, welche Daten an die nächsthöhere Instanz weitergegeben werden. Durch diese Vorselektion wird die Speicherung sowie der Transfer von nicht benötigten Daten vermieden.
Ein westlicher Bestandteil dieser Layer ist die „IoT Integration Middleware“. Middleware empfängt Daten von verbundenen Geräten, wandelt diese in ein für das Anwendungsbackend verwertbare Format um und speichert diese nach Bedarf in persistente Speicher (Tiburski et al. 2015). Zudem regelt Middleware die Verwaltung und Steuerung von Endgeräten. Durch Middleware werden die Daten erst für die Zielanwendung effizient nutzbar, somit ist sie ein zentraler Bestandteil jeder IoT-Lösung (ICITST et al. 2015).
„Thing Site“ sieht vorerst keine direkte Kommunikation von unterschiedlichen Devices vor. Die Übertragungsrichtung ist im ersten Schritt ausschließlich vertikal, das heißt von Pi zur Speicherressource. Dort werden die Daten verarbeitet und in einer für das Frontend zugänglichen Datenbank gespeichert.
Gemäß der oben beschriebenen Klassifizierung nimmt der Raspberry Pi in „Thing Site” durch seine Funktionalitäten eine Sonderstellung ein. Er ist dabei nicht nur für die Erzeugung der Daten verantwortlich, sondern beheimatet auch Routinen zur Datenanalyse. Zudem nimmt er mithilfe der IoT-Integration Middleware eigenständig Kontakt zur Anwendung auf, nicht wie üblicherweise ein Smart Gateway.
Die Gründe hierfür liegen auf der Hand. Im Gegensatz zu gängigen IoT-Geräten besitzt der Rasperry Pi deutlich mehr Leistung und Möglichkeiten der Datenübertragung. Um die Funktionsweise des angestrebten IoT-Systems zu demonstrieren, ist es für „Thing Site” vorerst irrelevant, ob die erzeugten Daten vor ihrem Upload auf dem Endgerät oder auf einer zusätzlichen Instanz, dem Smart Gateway, gefiltert werden, zumal der Pi alle physischen Voraussetzungen hierfür mitbringt. Er vereint somit die Funktionalitäten des Gateways und des Endgeräts.

2.2.3 Application

Die Ebene „Application“ greift auf die aufbereiteten Daten zurück, um den eigentlichen Nutzen des IoT-Systems zu generieren und das ausgeschriebene Ziel zu realisieren.

Sie stellt das Frontend der Vision dar und bietet neben Informationen auch Steuerungs- und Regelungsmöglichkeiten, um das System zu kontrollieren und nach den Wünschen des Nutzers zu beeinflussen (ICITST et al. 2015).

Im Falle von „Thing Site“ wird diese Ebene durch eine browserbasierte Webanwendung repräsentiert. Der Nutzer erhält hier die Möglichkeit, visualisierte Daten in Form von Bildern und Standortinformationen anzeigen zu lassen, sowie gegebenenfalls Steuerungsmaßnahmen einzuleiten.

2.3 Cloud of Things

Obwohl es vorstellbar und durchaus machbar wäre, die oben vorgestellte Architektur von Grund auf neu zu implementieren und dabei die einzelnen Programmteile nach den Wünschen des Nutzers zu formen, ist dies im Zeitalter von Cloud Computing nicht unbedingt sinnvoll. Eine Vielzahl von Anbietern stellen IoT-Lösungen bereit. Diese liefern mit nahezu unbegrenzten Ressourcen ausgestattete Umgebungen, die ganz nach Bedarf und Anwendungsfall skaliert werden können. Sie sind darauf spezialisiert, große Datenmengen zu organisieren, zu verarbeiten oder weiterzuleiten. Ein entscheidender Vorteil hierbei ist, dass zum Projektbeginn beinahe keine Initialkosten für Erstinvestitionen anfallen und man sich im Projektverlauf keine Sorgen um zur Verfügung stehende Ressourcen machen muss, da die meisten Modelle nach dem „Pay-per-Use“-Prinzip betrieben werden (2015 3rd International Conference on Future Internet of Things and Cloud 2015 - 2015). Zudem wird durch die angekaufte Infrastruktur eine gewisse Robustheit in Bezug auf Stabilität und Sicherheit gewährleistet. Oft werden dem Kunden auch verschiedene vorgefertigte Tools zur Überwachung und Quantifizierung der eingehenden Daten mitgeliefert. Mit diesem Toolset lässt sich folglich mit verhältnismäßig wenig Entwicklungsaufwand eine mächtige IoT-Lösung realisieren.

Provider stellen dabei unterschiedlichste Software-Lösungen zur Verfügung, die sich im Wesentlichen durch Ihre Freiheitsgrade in Bezug auf implementierbare Kommunikationswege zwischen Endgeräte unterscheiden (Xu et al. 2014). Anbieter interpretieren das IoT entsprechend der Interessen ihrer Stakeholder, dem Bedarf eines Geschäftsfeldes und der daraus resultierenden, individuellen Vision (Farahzadia et al. 2018).

2.3.1 „Product specific IoT Middleware“

„Product specific“ IoT-Plattformen bieten dem Entwickler den geringsten Freiheitsgrad. Sowohl die Art als auch die Anzahl der anbindbaren Endgeräte werden vom Provider limitiert. Die Analyse und Visualisierung der Daten erfolgt nach vom Hersteller definierten „Use Cases“. Das heißt, es werden typischerweise Funktionalitäten der Cloud über APIs (Application Programming Interfaces) bzw. fertige Anwendungen zur Verfügung gestellt. Sowohl Protokolle als auch Speicherressourcen sind in diesem Fall alternativlos. Dem Cloud Provider werden somit alle sicherheits- und privatsphärerelevanten Themen anvertraut (Ngu et al. 2016).

Typische Beispiele dafür sind „Google Fit“ oder „Apple HealthKit“. Hier können vom Cloud Provider zertifizierte IoT-Geräte mithilfe einer API abonniert werden und deren ausgelesene Daten z.B. in der „Google Fit App“ dargestellt werden.

2.3.2 „Actor-based IoT Middleware“

Bei „Actor-based“ Plattformen steht die Abstraktion von wiederverwendbaren Softwarekomponenten im Vordergrund. Diese als „Actors“ definierten Programmteile können IoT-Geräte, Rechenoperationen oder von der Plattform bereitgestellte Services sein. Ein „Actor“ wird durch seine In- und Outputs definiert, die wiederum in das vom Provider zur Verfügung gestellte Interface eingepasst werden. Im Kontrast zu anderen Cloud-Konzepten können diese nicht nur in der Cloud oder einer Applikation implementiert werden, sondern in allen zugänglichen Systemkomponenten. Daten werden folglich dort verarbeitet, wo es für das Gesamtsystem am vorteilhaftesten ist. Zudem wird durch die propagierte Architektur die Kommunikation von Device zu Device bzw. Device zu Application standardisiert, was dem Entwickler zusätzliche Flexibilität bei der Umsetzung seiner Vision ermöglicht, ohne das Framework verlassen zu müssen. Somit zeichnen sich „Actor-based“ Architekturen im Vergleich zu anderen Lösungen durch maximale Skalierbarkeit und geringe Latenz aus (Ngu et al. 2016).

2.3.3 „Service-based IoT Middleware“

„Service-based“ oder SOA-Lösungen (Service-oriented Architecture) sind hoch performante Architekturen, die von Devices erzeugte Daten in Form von Services in einer Cloud verwalten. Komplexe Aufgaben werden auf von der Cloud bereitgestellte, generische „Services“ heruntergebrochen und in Form einer „Service Chain“ abgearbeitet.

Die Koordination und Auswahl geeigneter Services wird in Workflows definiert. Services sind beispielsweise virtuelle Gerätemanager, Ereignisverarbeitungs-Engines, oder Speicherressourcen. Die Herausforderungen dieser Architektur für den Anwender sind die Kenntnis von Schnittstellen und der zur Verfügung stehenden Services, deren Konzeptionierung sowie deren effizienter und wirtschaftlicher Einsatz.

Die damit einhergehende Modularität begünstigst Wiederverwendbarkeit, flexiblen Austausch und eine einfache Delegation von Services. Dadurch wird eine an den Markt und technologische Änderungen schnell anpassbare IoT Architektur gewährleistet (Atzori et al. 2010). Device zu Device Kommunikation im klassischen Sinne ist bei SOA nicht vorgesehen.

SOA-Lösungen bieten für „Thing Site“ das nötige Maß an Freiheitsgraden in Bezug auf die Entwicklung der Layer „Application“ und Geräteregistrierung. Die zur Verfügung gestellten Services und Workflows liefern dabei eine geeignete Grundlage für die Verwaltung von IoT-Devices, Nachrichtenmanagement und Speicherzuweisung, ohne dabei den Rahmen dieser Masterarbeit bezüglich Entwicklungsaufwand zu übersteigen. Die Auswahl der IoT-Cloud Lösung wird für die vorliegende Arbeit deshalb auf SOA beschränkt.

3 Grundlagen

Für die Umsetzung von „Thing Site“ mussten mehrere Problemstellungen der Ebenen „Device“, „Network“ und „Application“ bewältigt werden. Daraus ergaben sich Ansprüche an eine Cloud-Lösung, anhand derer ein Anbieter für die Umsetzung ausgewählt wurde. Die verwendete Cloud-Lösung und die benötigten Grundlagen, auf denen die in Kapitel Implementierung vollzogene Umsetzung begründet ist, werden anschließend beschrieben.

3.1 Auswahl der Cloud Plattform

Derzeit gibt es eine Vielzahl an Anbietern der Kategorie Service-oriented. Im Folgenden wird eine kurze Gegenüberstellung der bekanntesten Anbieter vollzogen und darauf eine Auswahl für „Thing Site“ getroffen. Auch wenn es bei den Anbietern unterschiedliche Herangehensweisen und Organisationsstrukturen gibt und die angestrebte Implementierung sich nicht bei allen Konkurrenten eins-zu-eins umsetzen lassen wird, können trotzdem Eckpunkte definiert werden, die die Plattformen aufweisen müssen. Zudem ist es nicht immer möglich die Plattformen bezüglich der oben eingeführten Kategorisierung abzugrenzen, da viele der Anbieter Charakteristika mehrerer Typen aufweisen und somit Hybride darstellen.

3.1.1 Anforderungen

Der zu erarbeitende Prototyp soll Daten in Form von Text und Bildern an die Cloud übermitteln. Dort werden die Daten gespeichert und daraufhin von Services in eine Datenbank eingepflegt. Die Visualisierung erfolgt mithilfe einer Weboberfläche. Die Devices sollen zudem über das Webinterface kontrolliert werden können. Die Kommunikation zwischen Cloud und IoT-Geräten muss also bidirektional unterstützt werden und wegen wechselnd guter Anbindung an das Telekommunikationsnetz möglichst ressourcenschonende Protokolle unterstützen.

3.1.2 Auswahl

Dementsprechend wurden die Anbieter anhand der folgenden Kategorien bewertet:

- Verfügbarkeit: welche Verfügbarkeit besteht für das System? Dies ist bedingt relevant für ein fertiges Industrieprodukt, bei negativer Bewertung jedoch ein

Ausschlusskriterium für diese Arbeit. Unterschieden wird zwischen „Open source“, „Trial“-Version und Zugriff über eine Studentenlizenz. Fast alle Anbieter stellen eine öffentliche Testlizenz zur Verfügung; untersucht wurde jedoch, ob mit dieser alle benötigten Features zugänglich sind.

- Bidirektionale Kommunikation: ist ein beidseitiger Austausch von Daten zwischen Cloud und Device innerhalb des Software Development Kits (SDK) des Providers möglich?
- Protokoll: welche Protokolle werden für die Kommunikation zwischen Device und Cloud unterstützt?
- Serviceseitige Programmiersprachen: welche Programmiersprachen werden auf Cloud Seite zur Programmierung der Services unterstützt?
- Clientseitige Programmiersprachen: welche Programmiersprachen werden für Device und Application unterstützt?
- Speicher: können eingehende Bilddaten unorganisiert gespeichert werden?
- Datenbank: bietet der Provider einen Datenbankservice oder die Möglichkeit, diesen zu hosten?
- Web-Application: kann die Web-Application über den Provider gehostet werden?

In der folgenden Tabelle 1 ist die Auswertung der zuvor definierten Parameter einiger Cloud-Lösungen zu finden, die einen kleinen Querschnitt des verfügbaren Angebots darstellen.

Tabelle 1 Bewertung von IoT-Cloud-Providern. Eigene Tabelle (Azure/azure-iot-sdks; AWS IoT-SDKs - AWS IoT 2019; Cloud IoT Core | Cloud IoT Core | Google Cloud 2019; CloudMQTT - Add-ons - Heroku Elements; IBM Watson IoT Platform - Überblick - Deutschland; DOC: Architecture overview)

Provider	Verfügbarkeit	Bidirektional	Protokoll	Service Sprache	Client Sprache	Speicher	Datenbank	Web-App
Azure	trial	ja	MQTT, AMQP, HTTP	C#, JavaScript, F#, Python, TypeScript	C, C#, C++, Java, Python, Node.js,	Ja	Ja	Ja
AWS	student	ja	MQTT, HTTP	Java, JavaScript, C#, PHP, Python, Ruby, GO, C++	C, C#, C++, Java, JavaScript, Python	Ja	Ja	Ja
Google Cloud IoT	trial	ja	MQTT, HTTP	Java, C#, PHP, Python, Ruby, GO, C++	C, Java, Node.js, Python	Ja	Ja	Ja
Heroku	trial	-	MQTT	Ruby, Java, PHP, Python, Node, Go, Scala, Clojure	-	Ja	Ja	Ja
IBM Watson IoT	trial	ja	MQTT, HTTP	Python, Node.js. Java, C#	Python, C++, C, C#, Java, Node.js	Ja	Ja	Ja
Kaa Platform	trial	-	MQTT, CoAP, HTTP	C,C++, Java	-	Ja	Ja	Ja

Fast alle der ausgewählten Provider weisen die für „Thing Site“ geforderten Kriterien auf. Die endgültige Auswahl des Providers wurde daher anhand von subjektiven Kriterien getätigt. Ausschlaggebend waren vor allem das vom Anbieter zur Verfügung gestellte Testkontingent, die Programmiersprachen und eine öffentlich zugängliche Dokumentation. Die Umsetzung erfolgte daher mit „Azure“ des Anbieters Microsoft.

3.2 Azure

Microsoft stellt dem Nutzer mit Azure eine Vielzahl von Services, Speicherressourcen, Analyse- und Visualisierungstools sowie jegliche Infrastruktur für deren Kommunika-

tion bereit. Dementsprechend gibt es viele mögliche Ansätze, „Thing Site“ zu realisieren. Im Folgenden werden die verwendeten Komponenten vorgestellt und deren Auswahl begründet sowie, wenn vorhanden, mögliche Alternativen benannt.

3.2.1 IoT Hub

Der Azure-Service „IoT Hub“ ist der zentrale Bestandteil von IoT-Cloudlösungen. Der Dienst wurde entwickelt, um eine große Masse von leistungsschwachen Geräten in die Cloud-Infrastruktur zu integrieren und bei großem Datendurchsatz ein hohes Maß an Zuverlässigkeit und Stabilität zu garantieren. Er delegiert an die Cloud gesendete Daten an den jeweiligen Pipelinestartpunkt, verwaltet Device-Instanzen und regelt jegliche Kommunikation zwischen Devices, Cloud, und Application.

Geräteverwaltung und SAS-Tokens

In IoT Hub werden die Geräteidentitäten verwaltet, d.h. definiert, welche Geräte Nachrichten und Dateien an die Cloud senden oder von ihr empfangen können und auf welche Speicherressourcen diese zugreifen dürfen. Zusätzlich können zur Verschlüsselung symmetrische Schlüssel (SAS-Tokens; Shared Access Signatures) eingesetzt werden. Jedem SAS-Token ist eine Rolle zugeordnet, die definiert, welche Dienste der Token-Halter abrufen darf. Die Geräteidentitäten und zugehörigen Tokens können entweder über eine HTTPS-Rest Schnittstelle oder ein von Azure generiertes Userinterface verwaltet werden (Shared Access Signatures) .

Client Endpoints

Für jede Identität stellt IoT Hub eine Reihe von Endpunkten zur Verfügung, die mit Hilfe der Azure Client SDK unter Angabe der in IoT Hub erstellten individuellen Verbindungszeichenfolge abgefragt werden können. Die wichtigsten darunter sind:

- *Send device-to-cloud messages* - Übertragen von Gerätenachrichten an IoT Hub
- *Receive cloud-to-device messages* - Abfragen und Empfangen in IoT Hub eingereihter Nachrichten für Gerät
- *Initiate file uploads* - Einleiten von Dateiuploads. Das Endgerät empfängt die im IoT Hub eingebundene SAS-URI einer Speicherressource (IoT Hub-Endpunkte).

Routing

Eingehende Nachrichten können abhängig von Metadaten und Status bewertet und durch End Points nachfolgenden Services zur Verfügung gestellt werden. So können Dateien in eine andere Pipeline geschickt werden, beispielsweise als Textnachrichten. Dieser Prozess wird als Routing bezeichnet (IoT Hub-Nachrichtenrouting).

Eventhubs

Event Hubs (auch event ingestors) sind die Startpunkte der jeweiligen Pipelines. Ein „event ingestor“ fungiert als Mittelsmann zwischen Erzeuger (z.B. eingehende Gerätenachrichten) und Konsumenten von Ereignissen (z.B. Speicheroperationen, Organisation in Datenbanken, Analyse Tools, Event Grids). Er wird verwendet, um eine organisatorische Trennung zwischen Erzeugung und Verarbeitung zu erreichen. Sowohl Konsumenten als auch Erzeuger sind eindeutig definiert. Jedes Ereignis wird mindestens einmal an jeden Konsumenten übermittelt (Azure Event Hubs).

3.2.2 Verarbeitung

Um die im Event Hub in Form von Events eingehenden Nachrichten zu verarbeiten, stehen eine Reihe von Möglichkeiten zur Verfügung. Im Folgenden werden zwei Services näher beschrieben, für die eine Implementierung vollzogen wurde.

Stream Analytics

Stream Analytics ist ein Ereignisverarbeitungstool. Es ist ein Konsument von in Eventhubs organisierten Daten und Ereignissen. Diese können mithilfe des Tools gefiltert, aufbereitet und an den nächsten Service weitergegeben werden. Die Vorteile von Stream Analytics sind Leistungsfähigkeit und einfache Bedienung.

Jeder Stream Analytics Job besteht aus drei Kernkomponenten:

Als „Quelle“ wird der Eventhub-Endpunkt bezeichnet, aus dem die Events gelesen werden sollen. Die „Transformation“, besteht aus der Deserialisierung der enthaltenen Zeichenfolge, der Selektion der relevanten Parameter, sowie der Analyse und Modifikation der Daten. Diese werden daraufhin an die „Senke“, einen beliebigen nachgeschalteten Service, übermittelt.

Klassische Anwendungsfälle für Stream Analytics Jobs sind Szenarien, bei denen große Datenmengen in Datenbanken organisiert oder auf Dashboards visualisiert werden sollen. Somit erfüllt der Service alle technischen Voraussetzungen für die Implementation in „Thing Site“ (Azure Stream Analytics).

Functions

Als Alternative können auch „serverlose Azure Functions“ für die Interaktion mit Datenbanken eingesetzt werden. Diese ermöglichen es dem Nutzer ereignisbasiert, ausgelöst über sog. „Trigger“, Code ausführen zu lassen. Trigger können in diesem Zusammenhang z.B. http-Anforderungen, Webhooks, Timer oder Zustandsänderungen von Azure-Services sein. Die Abstraktion „serverlos“ bedeutet in diesem Zusammenhang lediglich, dass sich der Nutzer, wie bei allen Cloud-Services, nicht die Umgebung verwalten muss, in der der Code ausgeführt wird.

„Functions“ sind der Azure Service mit den größten Freiheitsgraden. Dieser erlaubt die Integration von externen Assemblies und präsentiert sich als vollwertige Laufzeitumgebung (Azure Functions – Tutorials).

Die beiden beschriebenen Services wurden für die Implementierung von „Thing Site” gegeneinander abgewägt. Die Verarbeitung von Events in Datenbanken ist eine Standardaufgabe für „Stream Analytics Jobs“. Die Integration ist dementsprechend einfach und zuverlässig. Sie verkörpern den klassischen service-oriented Lösungsweg. Dem gegenüber steht der wesentlich günstigere und flexiblere Ansatz der „Azure Functions“.

Der Vollständigkeit halber wurde die Implementation in beiden Services vollzogen. Für Nutzer, die sich nicht zu stark an die Cloud Lösung binden wollen, ist jedoch zweiteres zu empfehlen. Der praktische Teil dieser Arbeit wurde mit beiden Varianten getestet.

3.3 Datenverwaltung

Die Speicherung der Daten erfolgt in einer von Azure gehosteten SQL (Structured Query Language) Datenbank. Entscheidende Vorteile einer in der Cloud gehosteten Datenbank sind die bedarfsmäßig skalierbaren Ressourcen sowie die einfache Integration in die Azure Infrastruktur. Durch wenige Modifikationen könnte diese aber auch bei einem beliebigen Anbieter oder lokal betrieben werden. Welche Modifikationen hierfür nötig sind, wird an späterer Stelle erläutert. werden. Welche Modifikationen hierfür nötig sind, wird an späterer Stelle erläutert.

Wenn Binärdaten mithilfe des Azure Clients übertragen werden sollen, muss vor dem Upload ein Container in der Cloud erstellt werden. In diesem Container werden die eingehenden Daten als Binary Large Objects (BLOB) gespeichert. Container können unbegrenzt viele BLOBs beinhalten und haben kein Ablaufdatum, somit eignen sie sich gut für das dauerhafte Speichern von Daten (roygara).

Um später effizient auf die Bilddaten zugreifen zu können, sollten diese organisiert werden. Dem Entwickler steht hierfür eine Reihe von Optionen zur Verfügung.

Zum einen können die Daten aus dem vorher definierten Container auf Ressourcen außerhalb der Cloud übertragen werden und nur eine Referenz in der Datenbank hinterlegt werden. Dies könnte sinnvoll sein, wenn die zur Verfügung stehenden Datentarife als nicht wirtschaftlich betrachtet werden, oder die Daten ohnehin an anderer Stelle gespeichert werden müssen. Da diese Variante eine zusätzliche Schnittstelle (mit allen dazugehörigen Sicherheitsrisiken und Wartungsarbeiten) schafft und außerdem der geschlossene Charakter von „Thing Site“ verloren geht, werden für die Umsetzung „inhouse“ Ansätze bevorzugt.

Für den Use Case wird deshalb die Hinterlegung von Azure BLOB Referenzen mit der direkten Speicherung von Binärdaten in der SQL Datenbank verglichen und dementsprechend die in der Umsetzung gewählte Strategie begründet.

3.3.1 Indirekte Speicherung

Bei der indirekten Organisation werden lediglich die Metadaten, wie Speicheradressen oder URIs einer Datenbankspalte, hinterlegt. Das eigentliche Dokument liegt auf einem Fileserver oder Cloud Storage. Dieser Herangehensweise bietet eine Reihe von Vorteilen, die im Folgenden erläutert werden.

1. Flexibilität

Sollte es nötig werden, den Speicherort der Bilddaten zu wechseln, müssen lediglich die URIs der Datenbank aktualisiert werden. Komplexere Tasks bei Migrationen können so meist vermieden werden. Zudem können Dateien einfach über ihre in der Datenbank hinterlegten URI heruntergeladen, als Referenz in anderen Systemteilen hinterlegt oder in HTML-Code eingebunden werden.

2. Leistung

Bei den in der Datenbank organisierten Dateien handelt es sich um hochauflösende Bilder mit dementsprechend großem Speicherbedarf. Um Leistungsengpässe zu vermeiden, propagieren Experten die indirekte Speicherung von Binärdaten über 1 MB (Gray 2006). Datenbanken von Cloud Services bieten zwar theoretisch die benötigten Ressourcen an, diese stehen aber nicht im Verhältnis zum wirtschaftlichen Nutzen.

3. Preis

Azure bietet die Möglichkeit, die Kosten für Datenbanken und BLOB-Speicher online überschlägig zu ermitteln. Diese sind abhängig vom Anwendungsfall und werden durch die Faktoren CPU, Speicher und Anzahl der Schreiboperationen bestimmt.

In der Tabelle 2 ist die preisliche Gegenüberstellung der beiden untersuchten Varianten dargestellt. Die Datentarife wurden anhand von Azure-Designempfehlungen getroffen.

Tabelle 2 Kostenübersicht von direkter und indirekter Speicherung in Datenbanken (Eigene Darstellung Preisrechner | Microsoft Azure; stevestein; What the heck is a DTU? - SQLPerformance.com 2017)

Variante	direkte Speicherung [USD/(Gigabyte*Monat)]	Kombination Datenbank-blob Speicher [USD/(Gigabyte*Monat)]
1 TB	0,28	0,060
2 TB	0,28	0,020
3 TB	0,28	0,016

3.3.2 Direkte Speicherung

Obwohl in der Vergangenheit davon abgeraten wurde, große Dateien in Form von Binärdaten direkt in Datenbanken zu organisieren, werden Probleme in Bezug auf die verminderte Leistungsfähigkeit im Zeitalter von Cloud Computing zunehmend zurückgedrängt (Elmasri und Navathe 2011, S. 595). Wer seine Dateien in Datenbanken speichert, profitiert unmittelbar von deren Vorteilen. Zu diesen Vorteilen gehören:

1. Integrität und Konsistenz

Die Dateien sind zu jedem Zeitpunkt mit den in der Datenbank hinterlegten Metadaten synchronisiert. Der Entwickler muss sich also keine Gedanken über „verwaiste“ Dateien auf dem Dateiserver ohne korrespondierenden Datenbankeintrag bzw. Datenbankeinträge ohne gültige Speicheradresse machen. Zudem entfällt die ansonsten notwendige Koordination der Einträge bei Lösch- oder Änderungsoperationen.

Auch im Zusammenhang mit Wiederherstellungen kommen die Stärken der direkten Speicherung zum Tragen: wie jeder andere Datentyp auch, können Binärdateien über Rollbacks oder Backups wiederhergestellt werden. Es ist folglich nur eine Routine für den kompletten Datensatz nötig.

2. Sicherheit

Im Gegensatz zur indirekten Speicherung wird keine zusätzliche Schnittstelle an die Weboberfläche preisgegeben. Die Daten werden im Kontext von SQL-Befehlen statt über Web-, Dateilinks oder dafür entwickelte Funktionen abgefragt. Ein Schutz gegen unerlaubten Zugriff wird in der Datenbank konfiguriert.

3.3.3 Auswahl

Mit der URI basierten, indirekten Speicherung wird dem Nutzer von Azure eine preiswerte, performante Organisationsstruktur zur Verfügung gestellt. Auch wenn für eine Anwendung mit Industriestandard die Vorteile einer direkten Speicherung entscheidend sein können, profitiert der Prototyp „Thing Site“, besonders im Hinblick auf die Webanwendung, von der Flexibilität der losgelösten Speicherung. Für die Implementierung wird deshalb dieser Ansatz verfolgt.

3.4 Device Hardware

Der Raspberry Pi als Single Board Computer (SBC, Einplatinencomputer) wird im Prototyping häufig als Ausgangspunkt für die Konzeptionierung von IoT-Devices verwendet. Das System verbindet die Vorteile von traditionellen Prototyping-Microcontrollern mit den grundlegenden Funktionalitäten eines Heimcomputers. Die geringen Abmessungen sowie der relativ niedrige Preis machen ihn attraktiv für den Einsatz in großen Stückzahlen und realen Testbedingungen (Maksimovic et al. 2014).

Der Pi kann mit den gängigen Betriebssystemen betrieben werden, d.h. der Code kann während der Entwicklung on-board geschrieben werden und verfügt modellabhängig

über alle gängigen Schnittstellen zur Netzwerkintegration und zum Datenaustausch (USB, (W-)LAN, HDMI, Audi-Jacks, Bluetooth u.v.). Der für „Thing Site“ verwendete Raspberry Pi 3B+ verfügt zudem über 29 GPIO-Pins (General Purpose Input and Output) und 15 Pin CSI (Camera Serial Inferace) für die einfache Integration des Systems mit anderen Komponenten und Schaltkreisen.

3.4.1 IMU und Kamera

Zudem wurde die CSI Kamera „Raspberry Pi Camera Module V1“, „Raspberry Pi Camera Module V1“, sowie der Kombisensor „MPU-9250“ mit jeweils 3-achsigen Gyroskop, Magnetometer und Accelerometer verbaut (vgl. Tabelle 3).

Tabelle 3 Übersicht der verbauten Sensorik (PS-MPU-9250A-01-v1.1; Camera Module)

Accelerometer	
Auflösung	0,00060 bis 0,00479 $[m/s^2]$
Initiale Variabilität des Skalenfaktors	3 %
Nichtlinearität des Skalenfaktors	0,5 %
Gyroskop	
Einsatzbereich	±250, ±500, ±1000 und ±2000 $[°/s]$
Auflösung	0,00763 bis 0,06098 $[°/s]$
Initialisierungsdrift	±5 $[°/s]$
Initiale Variabilität des Skalenfaktors	3 %
Nichtlinearität des Skalenfaktors	0,1 %
Magnetometer	
Einsatzbereich	±4800 µT
Auflösung	0,6 µT/LSB (14 bit) oder 15 µT/LSB (16 bit)
Initiale Variabilität des Skalenfaktors	5%
Camera Module v1 (visible light)	
Auflösung	2592 × 1944 Pixel
fokale Länge	3,60 mm ± 0,01
horizontales Sichtfeld	53,50 ± 0,13 °
vertikales Sichtfeld	41,41 ± 0,11 °
Camera Module v2 (infrared)	
Auflösung	3280 × 2464 Pixel
fokale Länge	3,04 [mm]
horizontales Sichtfeld	62,2 °
vertikales Sichtfeld	48,8 °

Um die Sichtachse des Kameramoduls korrekt zum Referenzkoordinatensystem des IMU ausrichten zu können, wurde eine Halterung entworfen, in der das Kameramodul und der IMU untergebracht werden können. Das physische Modell wurde mit einem 3D-Druck umgesetzt (vgl. Abbildung 3.1).

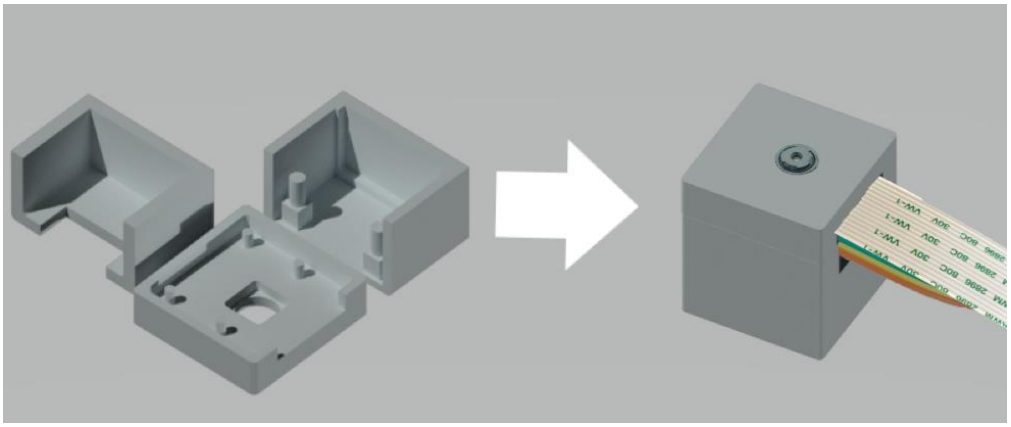


Abbildung 3.1 Kamera- und Sensorgehäuse (Eigene Grafik)

3.4.2 Echtzeituhr

Da der Raspberry Pi über keine physische on-board Real-time Clock (RTC; Echtzeituhr) verfügt, wurde zudem der Chip „DS1307 I2C Real Time Clock“ verbaut. Somit kann nach Systemabstürzen oder Neustarts auch ohne eine aktive Internetverbindung die aktuelle Zeit abgerufen werden (TinyRTC Datasheet).

Die meisten externen RTC-Module werden für Arduino Boards konzeptioniert und besitzen deshalb eine I²C-Ausgangsspannung (Inter Integrated Circuit) von 5 Volt. Damit das Modul stattdessen die vom Raspberry Pi benötigten 3,3 Volt anlegt, werden die Pullup-Widerstände (Spannungserhöhende Widerstände) R2 und R3 ausgelötet. Durch das Entfernen der Diode D1 und der Widerstände R4 und R5, sowie das Überbrücken des Widerstands R6 kann zusätzlich die Ladesschaltung der integrierten Knopfzelle deaktiviert werden (Abbildung 3.2). Das Modul kann somit durch Batterien anstatt Akkus betrieben werden.

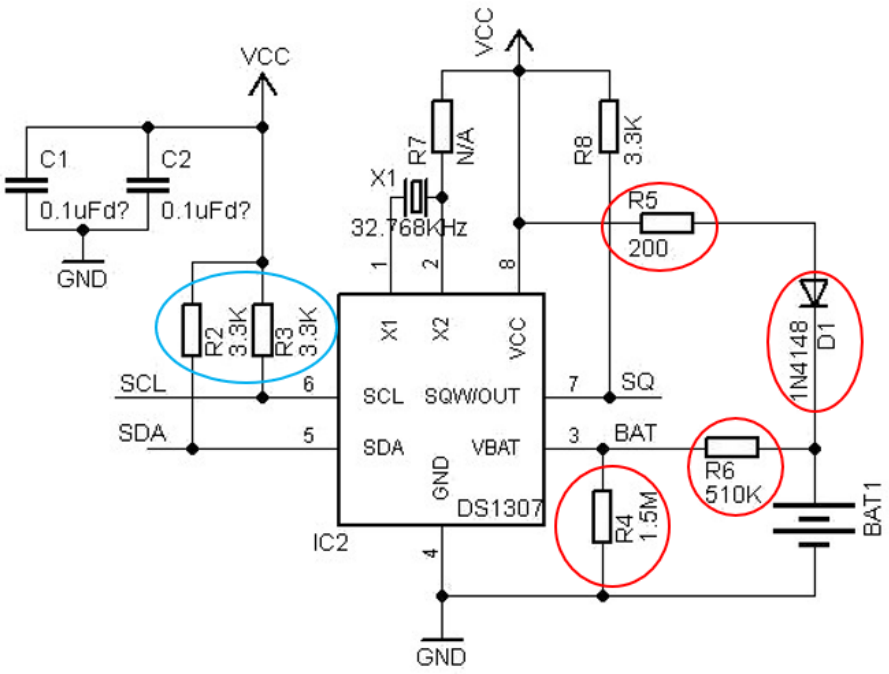


Abbildung 3.2 Schaltplan des Tiny-RTC Moduls. Pullup Widerstände (blau) und Ladeschaltung (rot) (eigene Grafik, angelehnt an Grasböck)

3.5 Neuronale Netzwerke

Mit „Thing Site“ soll es möglich sein, Objekte auf Kamerabildern zu erkennen und deren relative Position auf den Bildern zu erfassen. Dies ist nötig, um Referenzpunkte auf Baustellen zu erkennen, beziehungsweise Objekte tracken zu können.

Für diese Anforderungen war der Einsatz eines neuronalen Netzwerks zweckmäßig. Es erlaubt dem Nutzer nach entsprechender Kalibrierung, benutzerdefinierte Objekte zu erkennen und deren approximierte „Bounding Box“, also den minimalen Umschließungsrahmen, in dem alle Punkte des Objekts liegen, zu bestimmen.

3.5.1 CNN

Bei Convolutional Neural Networks (CNN) handelt es sich um einen Typ von neuronalen Netzwerken, der in Bereichen der Bildanalyse gute Ergebnisse liefert (Krizhevsky et al. 2017). Neural Networks (NN) bestehen aus In- und Outputs, sowie (theoretisch) beliebig vielen dazwischenliegenden „hidden“ (deutsch: versteckte) Layers.

Im Falle von Bildanalysen besteht der Input aus einer Matrix, deren Elemente die Farbwerte der einzelnen Pixel des zu verarbeitenden Bildes repräsentieren und somit als

Input-Neuronen fungieren. Bei Bildklassifizierungsproblemen gibt der Output des letzten Layers in der Regel die Wahrscheinlichkeit des Vorhandenseins von vorher definierten Klassen aus.

Die dazwischenliegenden Layer lassen sich in „image processing“, „convolutional“, „pooling“ und „fully connected“ unterteilen (Cireşan et al. 2011).

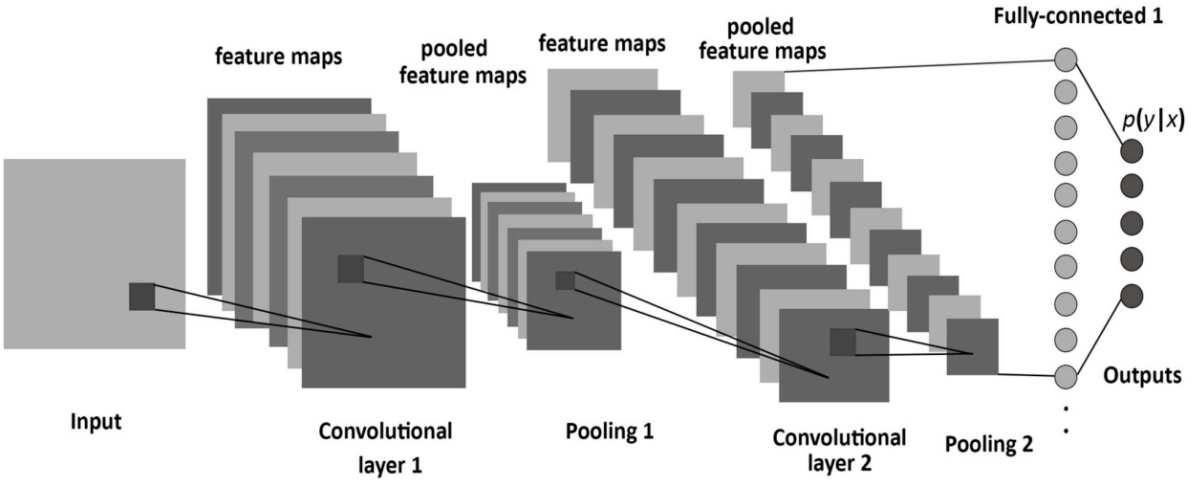


Abbildung 3.3 Aufbau eines CNN (Stefania Trapani 2018)

Convolutional Layer

Im convolutional Layer wird eine Faltungsoperation eines Bildausschnitts des Input-Layers mit einer Filter-Matrix (engl. Kernel) ausgeführt. Die Faltungsmatrix wird schrittweise über die Eingangsmatrix bewegt. Die Schrittweite sowie die Größe der Faltungsmatrix bestimmen die Größe der Output-Matrix (feature map). Die Faltungsoperation ist das Skalarprodukt des Eingangssignals mit der Kernel-Matrix und definiert pro „Schritt“ ein Neuron des Output-Layers.

Die Elemente der Kernel-Matrix werden als „weights“ bezeichnet und werden während des Trainings, also der Kalibrierung, iterativ bestimmt (Abbildung 3.4)(Cireşan et al. 2011). Abstrakt ausgedrückt ist der Prozess der „Faltung“ die Schärfung der durch die Kernel-Matrix definierten Merkmale (Jahr et al. 2018). Dabei steigt der Grad der Komplexität der repräsentierten Features mit der Tiefe des neuronalen Netzwerks. Man spricht auch von „lowlevel“–Features, also einfache Primitive, die in den ersten Layers abgebildet werden, die graduell zu „highlevel“-Features, welche in tieferen Layers des Netzwerks zu finden sind, übergehen (Zeiler und Fergus 2013).

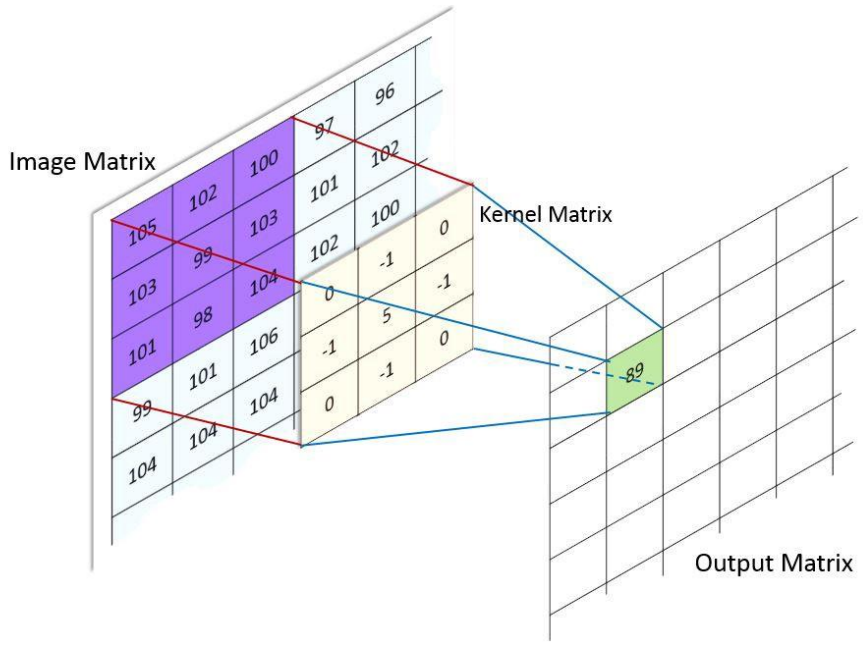


Abbildung 3.4 Faltungsoperation durch Kernel Matrix (Image Convolution 2017)

Pooling Layer

Um die Anzahl der Rechenoperationen und die somit benötigte Leistung für das CNN möglichst gering zu halten, empfiehlt sich der periodische Einsatz von „pooling“-Layers. Diese reduzieren die räumliche Ausdehnung eines Inputs, indem sie die Eingangsmatrix in Unterregionen, üblicherweise 2x2, oder 3x3, aufteilen und diese zum Beispiel mit MAX-, SUM- oder AVG-Operationen auf einen Wert reduzieren (Convolutional Neural Networks for Visual Recognition 2019).

Fully Connected Layer

Im Gegensatz zu convolutional Layers sind fully connected Layers nicht nur mit einer kleinen Teilmenge des vorangehenden Layers vernetzt, sondern mit jedem Neuron der hierarchisch vorangehenden Layer. Der Output dieses Layers ist ein n - dimensionaler Vektor, wobei n der Anzahl der trainierten Klassen entspricht. Beim Einsatz einer softmax Funktion entspricht jeder Wert des Vektors der Auftretenswahrscheinlichkeit der korrespondieren trainierten Klasse (Multi-Class Neural Networks: Softmax | Machine Learning Crash Course | Google Developers 2019)

Faster R-CNN

State-of-the-art Architekturen wie das Faster R-CNN (Regional Convolutional Neural Network), ein Derivat des CNN, sind zudem in der Lage mehr als eine Klasse pro Bild vorherzusagen und mithilfe von Region Proposal Networks (RPN) deren Bounding Boxen (Umschließungsrahmen) nahezu in Echtzeit zu approximieren. Das RPN ist eine

zusätzliche Netzwerkkomponente, welche, auf Basis der durch Convolutional Layers erzeugten Feature Maps, Bounding Boxen generiert. Diese weisen eine hohe Wahrscheinlichkeit auf, eine der trainierten Klassen zu enthalten. Die Feature Maps werden anschließend auf die Größe der vorgeschlagenen Bounding Boxen reduziert und von einem Detection Network hinsichtlich der Auftretenswahrscheinlichkeit der trainierten Klassen bewertet. Das Detection Network wiederum besteht aus mehreren fully connected Layers (Ren et al. 2015).

3.5.2 Transfer Learning

Für die Kalibrierung eines CNN werden zunächst die in „Convolutional Layer“ erwähnten Gewichte auf allen Layers zufällig initialisiert. Die Trainingsdaten werden dem noch unkalibrierten Netzwerk als Input geliefert und der daraus resultierende Outputvektor bestimmt. Dieser Prozess wird als „forward propagation“ bezeichnet.

Da während der Kalibrierung die „forward propagation“ mit Trainingsdaten durchgeführt wird, kann die Abweichung vom gewünschten Zielwert als Fehler bestimmt werden. Durch „backward propagation“ werden die Gewichte der Neuronen, proportional zu ihrem Einfluss auf den Gesamtfehler, so angepasst, dass die Abweichung vom Zielwert minimiert wird. Nachdem das Netzwerk mit ausreichend vielen Trainingsdaten kalibriert wurde, soll es Bilder mit ähnlichen Feature-Sets, welche nicht im eigentlichen Training verwendet wurden, kategorisieren können (Liu et al. 2015).

Die meisten modernen CNN Architekturen bestehen aus mehreren Dutzend Layers und dementsprechend vielen zu kalibrierenden Parametern. Um zuverlässige Ergebnisse zu erhalten, muss das Netzwerk mit einer sehr großen Menge an Bildern über einen langen Zeitraum trainiert werden. Im Einzelfall hängt die Anzahl der benötigten Trainingsdaten stark von der eingesetzten Architektur und deren Einsatzzweck ab. ResNet, eine CNN-Architektur zur Bildklassifizierung und Gewinner der ILSVRC 2015 (Large Scale Visual Recognition Challenge 2015) wurde beispielsweise mit rund 1,3 Millionen Bildern trainiert (Ren et al. 2015).

Das Konzept des „Transfer-Learning“ erlaubt es dem Nutzer auch mit einem wesentlich kleineren Datensatz gute Ergebnisse zu erzielen. Dafür werden die Netzwerkparameter von Trainings auf wesentlich größere Datensätze verwendet, womit dem NN neu hinzugefügte Klassen mit weniger Lernaufwand zuverlässig erkannt werden können (Razavian). Die Überlegung dabei ist, dass „low level“ Features sich bei ähnlichen Tasks nur unwesentlich unterscheiden und die eigentliche Manifestierung des Objekts

in den „highlevel“ Features stattfindet. Die frühen Schichten und deren Gewichte werden erhalten, wohingegen die „fully connected“-Layers neu initialisiert werden (Shin et al. 2016).

4 Mathematische Hintergründe

Im Folgenden werden die mathematischen Grundlagen, die für die Umsetzung von „Thing Site“ herangezogen wurden, hergeleitet und erläutert. Im Einzelnen wurde sich mit den folgenden Fragestellungen befasst: Approximation der Kameraposition, Approximation der räumlichen Position von erkannten Objekten und Approximation der Kameraorientierung.

4.1 Kamera-Modell

Eine Kamera kann mit dem projektiven „Lochkamera-Modell“ beschrieben werden. Dieses beschreibt die Transformation von 3D-Punkten in Szenen bzw. Weltkoordinaten und deren korrespondiere Bildkoordinaten. Objekte werden durch das projektive Zentrum, ein fiktives, unendlich kleines Loch, auf die Bildebene projiziert. Diese Projektion wird durch mehrere innere (intrinsische) und äußere (extrinsische) Parameter beeinflusst (vgl. Abbildung 4.1) (Derek Hoiem 2011).

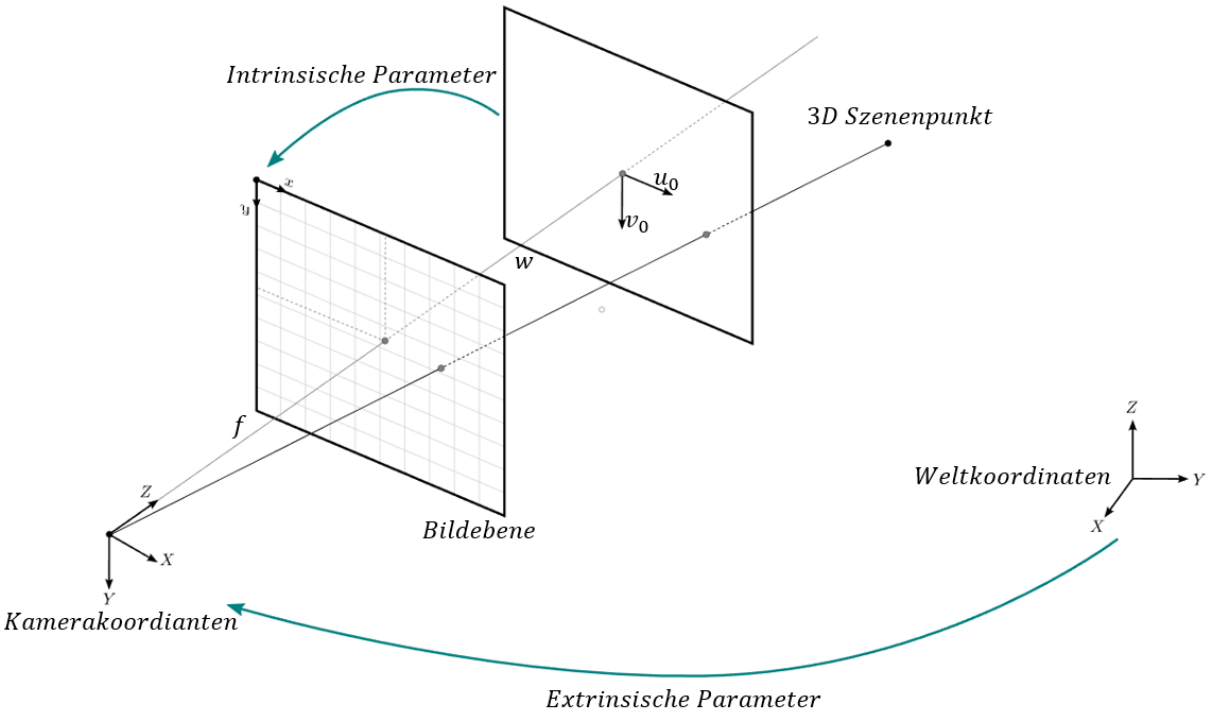


Abbildung 4.1 Zusammenhänge von intrinsischen und extrinsischen Kameraparametern (eigene Grafik, angelehnt an cameras — openMVG library 2017)

4.1.1 Intrinsische Parameter

Intrinsische Parameter, auch Kameraparameter genannt, sind kameraspezifische Eigenschaften, die sich direkt auf die Darstellung der Szene auf der Bildebene auswirken. Die Parameter

f : Fokale Länge, also Abstand der Bildebene zum fiktiven Projektionszentrum,

k_u, k_v: Skalierungsfaktor für anisotrope Pixel,

sowie

u_0, v_0: Ursprung des Bildkoordinatensystems,

bilden die intrinsische Matrix

$$\boldsymbol{K} = \begin{bmatrix} f * k_u & 0 & u_0 \\ 0 & f * k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Peter Egor Bondarev de With 2017).

4.1.2 Extrinsische Parameter

Extrinsische Parameter beziehen sich auf die Transformation des Kamerakoordinatensystems. Die Rotationmatrix $\boldsymbol{R}(\alpha, \beta, \gamma)$ für euklidische Winkel beziehungsweise $\boldsymbol{R}(\varphi, \theta, \psi)$ für Kardanwinkel (Rill und Schaeffer 2010, S. 5–8, Kapitel 2) mit

θ : Nickwinkel, φ : Rollwinkel, ψ : Gierwinkel

sowie für die Implementierung verwendete GNR-Konvektion (Gierwinkel, Nickwinkel, Rollwinkel) mit

$$\boldsymbol{R}_{GNR} = \boldsymbol{R}_z(\psi) * \boldsymbol{R}_y(\theta) * \boldsymbol{R}_x(\varphi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

und

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{R}_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix},$$

$$\boldsymbol{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi) & -sin(\varphi) \\ 0 & sin(\varphi) & cos(\varphi) \end{bmatrix}$$

mit

$$\boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$: Translation der Kamera in Kamerakoordianten

bilden zusammen die extrinsische Matrix

$$\boldsymbol{E} = [\boldsymbol{R}^{-1}\ \boldsymbol{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}.$$

Der Gierwinkel beschreibt dabei die erste Drehung um die z-Achse des Echtweltkoordinatensystems. Die zweite Drehung erfolgt um den Nickwinkel der neuen, aus der ersten Rotation entstandenen y-Achse. Die letzte Drehung erfolgt um den Rollwinkel um die aus der zweiten Drehung entstandene x-Achse (Abbildung 4.2). Diese Abfolge der Rotationstransformationen entspricht dem Luft- und Raumfahrtstandard. Die Rotationsmatrix $\boldsymbol{R}$ beschreibt Transformationen von körperfesten in raumfeste Koordinatensysteme. Die Inverse $\boldsymbol{R}^{-1}$ Transformationen von raumfesten in körperfeste Koordinaten.

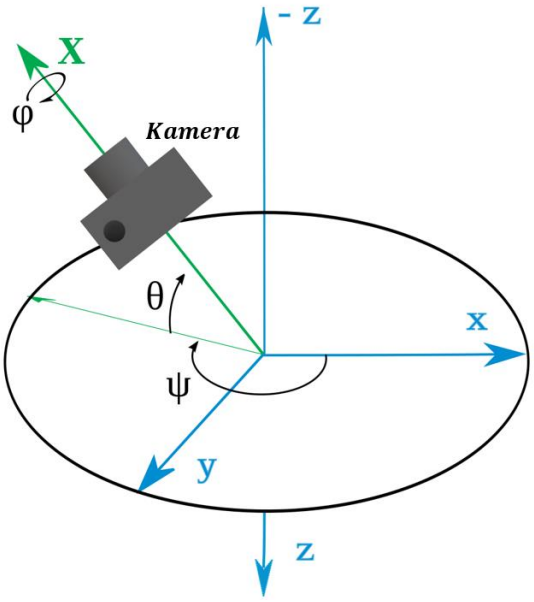


Abbildung 4.2 GNR-Konvention für Luft und Raumfahrt (eigene Grafik, angelehnt an GNR-Konvention 2018)

4.1.3 Beziehung

Die Abbildung des 3D Szenenpunktes

$$\boldsymbol{p}' = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

lässt sich somit auf der Bildebene durch einen Punkt mit dem Skalierungsfaktor w

$$\boldsymbol{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} w$$

durch die Beziehung

$$\boldsymbol{x} = \boldsymbol{K} * \boldsymbol{E} * \boldsymbol{p}$$

als

$$w * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f * k_u & 0 & u_0 \\ 0 & f * k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

ausdrücken (Peter Egor Bondarev de With 2017).

4.2 Kamerakalibrierung

Das oben eingeführte Modell beschreibt, wie dreidimensionale Punkte auf einer 2D-Bildebene abgebildet werden. Da im Falle der Kamera die Lichtstrahlen eine Linse passieren müssen und dabei abgelenkt werden, sind u und v fehlerbehaftet.

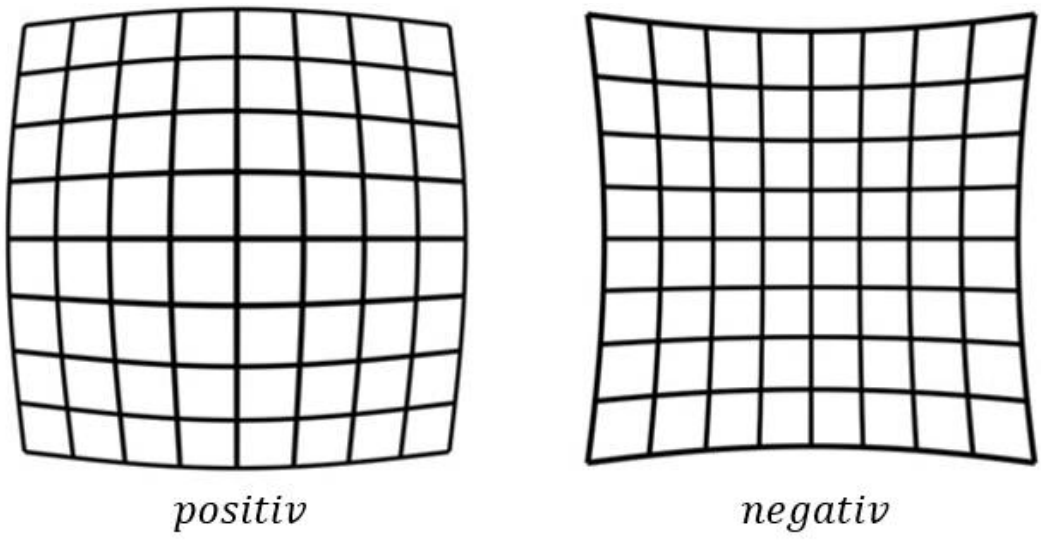


Abbildung 4.3 positive (links) und negative (rechts) radiale Verzerrung (Lens Distortion: What Every Photographer Should Know 2019)

Zur Kompensierung der durch die Linse induzierten Verzerrung modellieren wir die Ablenkung des einfallenden Lichts mithilfe der radialen Verzerrungskoeffizienten k_1 , k_2 und k_3 nach dem Brown-Conrady Modell und erhalten somit die entzerrten Punkte $\bar{u}$ und $\bar{v}$ durch die Beziehung

$$\bar{u}_{radial} = u(1 + k_1\, r^2 + k_2 r^4 + k_3 r^6)$$

$$\bar{v}_{radial} = v(1 + k_1\, r^2 + k_2 r^4 + k_3 r^6),$$

wobei gilt

$$r = \sqrt{u^2 + v^2}.$$

Falls, beispielsweise durch ungenaue Fertigung, der Sensor und die Linse nicht parallel zueinander angeordnet sind, kommt es zudem zu tangentialer Verzerrung. Diese wird mit den tangentialen Verzerrungskoeffizienten p_1 und p_2 durch die Beziehung

$$\bar{u}_{tangential} = u + (2p_1uv + p_2(r^2 + 2u^2))$$

$$\bar{v}_{tangential} = v + (p_1(r^2 + 2v^2) + 2p_2uv)$$

beschrieben. Somit ergibt sich der entzerrte Punkt zu

$$\begin{bmatrix}\bar{u}\\ \bar{v}\end{bmatrix} = (1 + k_1\, r^2 + k_2r^4 + k_3r^6)\begin{bmatrix}u\\ v\end{bmatrix} + \begin{bmatrix}2p_1uv + p_2(r^2 + 2u^2)\\ (p_1(r^2 + 2v^2) + 2p_2uv)\end{bmatrix}$$

(Duane C. Brown 1966).

4.3 Approximation der Kameraposition

Die in 4.1 eingeführte Beziehung

$$\boldsymbol{x} = \boldsymbol{K} * [\boldsymbol{R^{-1}}\ \ \boldsymbol{t}] * \boldsymbol{p}$$

lässt sich durch Umformung als

$$\boldsymbol{K^{-1}}\begin{bmatrix}u\\ v\\ 1\end{bmatrix} w = \boldsymbol{R^{-1}p} + \boldsymbol{t}$$

mit

$$\boldsymbol{K^{-1}}\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = \tilde{\boldsymbol{y}} \quad , \qquad \boldsymbol{R^{-1}p} = \tilde{\boldsymbol{x}}$$

$$\boldsymbol{w\tilde{y}} = \tilde{\boldsymbol{x}} + \boldsymbol{t}$$

ausdrücken. Die Komponente

$$w\tilde{\boldsymbol{y}}$$

spannt den Unterraum

$$\boldsymbol{V_{\tilde{y}}} = \{w\tilde{\boldsymbol{y}} \quad : \quad w \in \mathbb{R}\}$$

auf.

Die Projektion eines beliebigen Vektors $\dot{\boldsymbol{v}}$ auf den Unterraum $\boldsymbol{V_{\tilde{y}}}$ kann durch

$$\dot{\boldsymbol{v}}' = \frac{1}{\|\tilde{\boldsymbol{y}}\|^2}\tilde{\boldsymbol{y}}\langle\dot{\boldsymbol{v}}, \tilde{\boldsymbol{y}}\rangle$$

$$= \frac{\tilde{\boldsymbol{y}} * \tilde{\boldsymbol{y}}^T}{\|\tilde{\boldsymbol{y}}\|^2} * \dot{\boldsymbol{v}}$$

$$= \frac{1}{\tilde{y}_1^2 + \tilde{y}_2^2 + \tilde{y}_3^2}\begin{bmatrix}\tilde{y}_1\tilde{y}_1 & \tilde{y}_1\tilde{y}_2 & \tilde{y}_1\tilde{y}_3\\ \tilde{y}_2\tilde{y}_1 & \tilde{y}_2\tilde{y}_2 & \tilde{y}_2\tilde{y}_3\\ \tilde{y}_3\tilde{y}_1 & \tilde{y}_3\tilde{y}_2 & \tilde{y}_3\tilde{y}_3\end{bmatrix} * \dot{\boldsymbol{v}}$$

ausgedrückt werden.

$$\boldsymbol{P}_{\tilde{\boldsymbol{y}}} = \frac{1}{\tilde{y}_1^2 + \tilde{y}_2^2 + \tilde{y}_3^2} * \begin{bmatrix}\tilde{y}_1\tilde{y}_1 & \tilde{y}_1\tilde{y}_2 & \tilde{y}_1\tilde{y}_3\\ \tilde{y}_2\tilde{y}_1 & \tilde{y}_2\tilde{y}_2 & \tilde{y}_2\tilde{y}_3\\ \tilde{y}_3\tilde{y}_1 & \tilde{y}_3\tilde{y}_2 & \tilde{y}_3\tilde{y}_3\end{bmatrix}$$

ist somit Projektionsoperator auf $\boldsymbol{V}_{\tilde{\boldsymbol{y}}}$ (Abbildung 4.4)

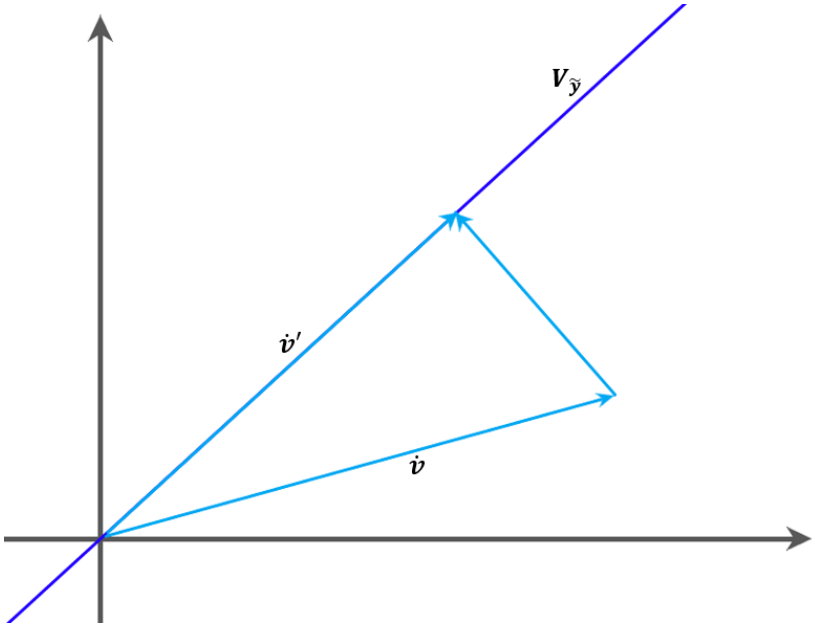


Abbildung 4.4 Projektion eines Vektors $\dot{\boldsymbol{v}}$ auf den Untervektorraum $\boldsymbol{V}_{\tilde{\boldsymbol{y}}}$ (Eigene Grafik)

4.4 Bestimmung der Translation

Um die Translation $\boldsymbol{t}$ und somit den Kamerastandpunkt zu bestimmen, betrachten wir folgenden Fall: zwei Punkte im Echtweltkoordinatensystem erzeugen zwei korrespondiere Punkte auf der Bildebene. Es gilt

$$w_1\tilde{\boldsymbol{y}}_1 = \tilde{\boldsymbol{x}}_1 + \boldsymbol{t}$$

$$w_2\tilde{\boldsymbol{y}}_2 = \tilde{\boldsymbol{x}}_2 + \boldsymbol{t}\ ,$$

wobei $\tilde{\boldsymbol{x}}_1 + \boldsymbol{t}$, bzw. $\tilde{\boldsymbol{x}}_2 + \boldsymbol{t}$ in den Untervektorräumen $\boldsymbol{V}_{\tilde{\boldsymbol{y}}_1}$, $\boldsymbol{V}_{\tilde{\boldsymbol{y}}_2}$ liegen (Abbildung 4.5).

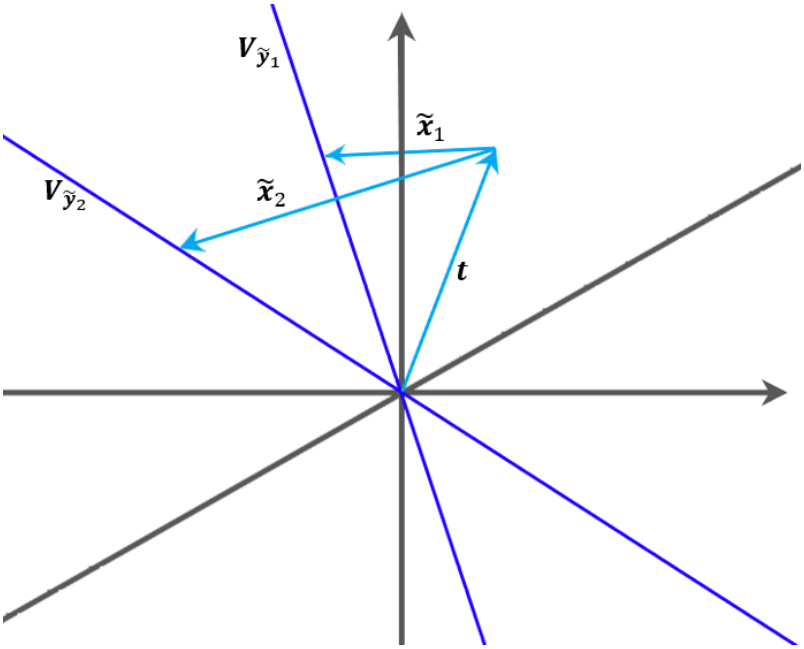


Abbildung 4.5 $\tilde{x}_1 + t$ und $\tilde{x}_2 + t$ mit den korrespondieren Untervektorräumen $V_{\tilde{y}_1}$ und $V_{\tilde{y}_2}$ (Eigene Grafik)

Es folgt für die Bildpunkte mit dem in 4.3 definierten Projektionsoperator $P_{\tilde{y}}$

$$P_{\tilde{y}_1}(\tilde{x}_1 + t) = \tilde{x}_1 + t$$

$$P_{\tilde{y}_2}(\tilde{x}_2 + t) = \tilde{x}_2 + t$$

bzw.

$$P_{\tilde{y}_n}(\tilde{x}_n + t) = \tilde{x}_n + t$$

$$(P_{\tilde{y}_1} - I)\tilde{x}_n = (I - P_{\tilde{y}_n})t$$

mit

$$x'_n = (P_{\tilde{y}_n} - I)\tilde{x}_n$$

erhält man das Gleichungssystem

$$\begin{bmatrix} x'_1 \\ \vdots \\ x'_n \end{bmatrix} = \begin{bmatrix} I - P_{\tilde{y}_1} \\ \vdots \\ I - P_{\tilde{y}_n} \end{bmatrix} t.$$

Zur Approximation der Translation ist die Lösung des Gleichungssystems durch die Methode der kleinsten Quadrate zweckmäßig.

Ausgehend von obigen Gleichungssystem dargestellt in der Form

$$A * t = q$$

mit

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} - \boldsymbol{P}_{\tilde{\boldsymbol{y}}_1} \\ \vdots \\ \boldsymbol{I} - \boldsymbol{P}_{\tilde{\boldsymbol{y}}_n} \end{bmatrix} \quad , \quad \boldsymbol{q} = \begin{bmatrix} \boldsymbol{x}'_1 \\ \vdots \\ \boldsymbol{x}'_n \end{bmatrix},$$

kann das Optimierungsproblem

$$\boldsymbol{t}_{opt} = \arg\min_{\boldsymbol{t}} \left\| \begin{bmatrix} \boldsymbol{x}'_1 \\ \vdots \\ \boldsymbol{x}'_n \end{bmatrix} - \begin{bmatrix} \boldsymbol{I} - \boldsymbol{P}_{\tilde{\boldsymbol{y}}_1} \\ \vdots \\ \boldsymbol{I} - \boldsymbol{P}_{\tilde{\boldsymbol{y}}_n} \end{bmatrix} \boldsymbol{t} \right\|$$

mithilfe der Moore-Penrose Pseudoinverse $\boldsymbol{A}^+$ gelöst werden:

$$\boldsymbol{t}_{opt} = \boldsymbol{A}^+ * \boldsymbol{q}.$$

Die Moore-Penrose Pseudoinverse ist die Verallgemeinerung der Inversen für quadratische Matrizen und wird häufig zum Lösen linearer Gleichungssysteme genutzt. Dabei liefert sie für Gleichungssysteme der Form

$$\boldsymbol{A} * \boldsymbol{x} = \boldsymbol{b}$$

die im Bezug auf die Methode der kleinsten Quadrate beste Lösung mit

$$\boldsymbol{x} = \boldsymbol{A}^+ * \boldsymbol{b}$$

(MoorePenrose).

Somit erhalten wir die Translation in raumfesten Koordinaten $\boldsymbol{t}_{WKS}$ zu

$$\boldsymbol{t}_{WKS} = \boldsymbol{R} * \boldsymbol{t}_{opt}$$

4.5 Approximation der räumlichen Position von erkannten Objekten

Um die Position eines Objektes $\boldsymbol{p}'$ zu bestimmen, das auf mindestens zwei Bildern simultan aufgenommen wurde, betrachten wir abermals die in Kamera Modell eingeführte Beziehung

$$\boldsymbol{x} = \boldsymbol{K} * \boldsymbol{E} * \boldsymbol{p}$$

$$w * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f * k_u & 0 & u_0 \\ 0 & f * k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Zunächst überführen wir die bekannte Translation $\boldsymbol{t}_{WKS}$, also die jeweiligen Kamerastandorte, in Kamerakoordinaten

$$\boldsymbol{t} = \boldsymbol{R}^{-1} * \boldsymbol{t}_{WKS}.$$

Mit

$$\boldsymbol{M} = \boldsymbol{K} * \boldsymbol{E} = \begin{bmatrix} m_{11} & \cdots & m_{14} \\ \vdots & \ddots & \vdots \\ m_{31} & \cdots & m_{34} \end{bmatrix}$$

ergibt sich

$$wu = m_{11}x + m_{12}y + m_{13}z + m_{14}$$

$$wv = m_{21}x + m_{22}y + m_{23}z + m_{24}.$$

Ersetzt man nun w durch

$$w = m_{31}x + m_{32}y + m_{33}z + m_{34}$$

erhält man

$$um_{31}x + um_{32}y + um_{33}z + um_{34} = m_{11}x + m_{12}y + m_{13}z + m_{14}$$

$$u\boldsymbol{M}_{3,1:3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + um_{34} = \boldsymbol{M}_{1,1:3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + m_{14}$$

und

$$vm_{31}x + vm_{32}y + vm_{33}z + vm_{34} = m_{21}x + m_{22}y + m_{23}z + m_{24}$$

$$v\boldsymbol{M}_{3,1:3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + vm_{34} = \boldsymbol{M}_{2,1:3}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + m_{24}.$$

Aufgelöst nach $\boldsymbol{l} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ folgt

$$\left[\boldsymbol{M}_{1,1:3} - u\boldsymbol{M}_{3,1:3}\right]\boldsymbol{l} = um_{34} - m_{14}$$

und

$$\left[\boldsymbol{M}_{2,1:3} - v\boldsymbol{M}_{3,1:3}\right]\boldsymbol{l} = vm_{34} - m_{24}.$$

Und somit

$$\begin{bmatrix} \boldsymbol{M}_{1,1:3} - u\boldsymbol{M}_{3,1:3} \\ \boldsymbol{M}_{2,1:3} - v\boldsymbol{M}_{3,1:3} \end{bmatrix}\boldsymbol{l} = \begin{bmatrix} um_{34} - m_{14} \\ vm_{34} - m_{24} \end{bmatrix}.$$

Mit

$$\widetilde{\boldsymbol{M}} = \begin{bmatrix} \boldsymbol{M}_{1,1:3} - u\boldsymbol{M}_{3,1:3} \\ \boldsymbol{M}_{2,1:3} - v\boldsymbol{M}_{3,1:3} \end{bmatrix} \quad \text{und} \quad \widetilde{\boldsymbol{p}} = \begin{bmatrix} um_{34} - m_{14} \\ vm_{34} - m_{24} \end{bmatrix}$$

lässt sich für n Punkte die Beziehung zu

$$\begin{bmatrix} \widetilde{\boldsymbol{M}}_1 \\ \widetilde{\boldsymbol{M}}_2 \\ \vdots \\ \widetilde{\boldsymbol{M}}_n \end{bmatrix} \boldsymbol{l} = \begin{bmatrix} \widetilde{\boldsymbol{p}}_1 \\ \widetilde{\boldsymbol{p}}_2 \\ \vdots \\ \widetilde{\boldsymbol{p}}_n \end{bmatrix}$$

erweitern.

Mit

$$\boldsymbol{Q} = \begin{bmatrix} \widetilde{\boldsymbol{M}}_1 \\ \widetilde{\boldsymbol{M}}_2 \\ \vdots \\ \widetilde{\boldsymbol{M}}_n \end{bmatrix}, \qquad \boldsymbol{j} = \begin{bmatrix} \widetilde{\boldsymbol{p}}_1 \\ \widetilde{\boldsymbol{p}}_2 \\ \vdots \\ \widetilde{\boldsymbol{p}}_n \end{bmatrix}$$

kann die abermals die Moore-Penrose Pseudoinverse $\boldsymbol{Q}^+$ zum Lösen des Gleichungssystems verwendet werden:

$$\boldsymbol{l}_{opt} = \boldsymbol{Q}^+ * \boldsymbol{j}.$$

Die Untersuchung von Bier und Luchowski kam zu vergleichbaren Ergebnissen (Bier und Luchowski 2009).

4.6 Bestimmung der Gier-, Nick- und Rollwinkel

Bei den im Prototyp verbauten Sensoren handelt es sich um ein dreiachsiges Gyroskop, ein Magnetometer, sowie einen Accelerometer. Aus der Kombination der Sensordaten soll die Orientierung der Kamera im Raum und somit die in Kamera Modell eingeführten Gier-, Nick- und Rollwinkel (θ, φ, ψ) möglichst genau bestimmt werden.

4.6.1 Nick- und Rollwinkel mittels Accelerometer

Bei dem Beschleunigungssensor handelt es sich um drei Messeinheiten, die jeweils parallel zu der x-, y- und z-Achse des körperfesten Koordinatensystems des Chips ausgerichtet sind. Die Einheiten messen dabei die richtungsbezogene Kraft auf ein gedämpftes Feder-Masse-System (Tränkler und Reindl 2014, S. 545).

Bei einem Stillstand des Systems wirkt lediglich die senkrecht in z-Richtung angreifende Erdanziehungskraft g. Somit gilt für die am Sensor angreifenden Teilkräfte $\boldsymbol{f}_g$

$$\|\boldsymbol{f}_g\|_2 = \left\| \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \right\|_2 = g \approx 9{,}81\ [m/s^2].$$

Für das unbewegte System, mit normierten Sensordaten und ausgedrückt in vielfachen von $\boldsymbol{g_e} = \begin{bmatrix}0\\0\\1\end{bmatrix}$, ergibt sich mit der in Extrinsische Parameter eingeführten Konvention für $\boldsymbol{R}_{GNR}$

$$\boldsymbol{f}_s = \frac{\boldsymbol{f}_g}{\|\boldsymbol{f}_g\|} = \boldsymbol{R}_{GNR}^T \boldsymbol{g_e}$$

$$\boldsymbol{f}_s = \begin{bmatrix} f_{sx} \\ f_{sy} \\ f_{sz} \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ sin(\varphi)cos(\theta) \\ cos(\varphi)cos(\theta) \end{bmatrix}.$$

Somit lassen sich Nick- und Rollwinkel z.B. zu

$$\varphi = arctan\left(\frac{f_{sy}}{f_{sz}}\right)$$

$$\theta = arctan\left(-\frac{f_{sx}}{f_{sy}\, sin(\varphi) + f_{sz} cos(\varphi)}\right)$$

bestimmen.

4.6.2 Gierwinkel mittels Magnetometer

Der Gierwinkel kann mithilfe des Magnetometers bestimmt werden. Dieses misst die magnetischen Flussdichten in x-, y-, und z-Richtung mithilfe von drei orthogonal angeordneten, parallel zu den körperfesten Koordinatenachsen verlaufenden Messeinheiten

$$\mathbf{h}_\mathrm{m} = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix}$$

(Tränkler und Reindl 2014).

Unter der Annahme, dass die XY-Ebene des körperfesten Koordinatensystems parallel zur Oberfläche des Referenzellipsoiden der Erde angeordnet ist, bestimmt sich die Abweichung zur Flussrichtung des Erdmagnetfeldes ∂ zu

$$\partial_{\theta=0,\varphi=0} = arctan\left(\frac{-h_y}{h_x}\right).$$

Für geneigte Systeme kann obige Gleichung mit Hilfe der in Nick und Rollwinkel mittels Accelerometer bestimmten Roll- und Nickwinkel modifiziert werden. Nach Ozyagcilar ergibt sich somit

$$\psi = arctan\left(\frac{-h_{ty}}{h_{\mathrm{tx}}}\right)$$

$$= arctan\left(\frac{h_z\, sin(\varphi) - h_y cos(\varphi)}{h_x\, cos(\theta) + h_y\, sin(\theta)\, sin(\varphi) + h_z sin(\theta) cos(\varphi)}\right)$$

(Talat Ozyagcilar).

Die Feldlinien des erdeigenen Magnetfeldes verlaufen auf der Erdoberfläche mehr oder weniger regelmäßig vom magnetischen Süd- zum Nordpol. Der Winkel, der zwischen Feldlinie und geographischem Nordpol eingeschlossen wird, wird als magnetische Deklination bezeichnet. Der Gierwinkel, also die tatsächliche Ausrichtung zum geographischen Norden, errechnet sich mit Hilfe des ortsgebundenen Korrekturwerts für die Deklination d zu

$$\psi = arctan\left(\frac{-h_{ty}}{\mathrm{h_{tx}}}\right) + d.$$

Das Magnetfeld der Erde ist dabei kein statisches System, sondern dauerhaft internen und externen Prozessen, wie Konvektionsströmen im Erdinneren oder veränderlicher Sonnenaktivität, ausgesetzt. Dementsprechend variabel ist auch die magnetische Deklination. Heute beträgt sie in München in etwa 3 Grad und 21 Minuten, 1965 waren es noch -1 Grad 59 Minuten (Kusserow 2013, S. 101).

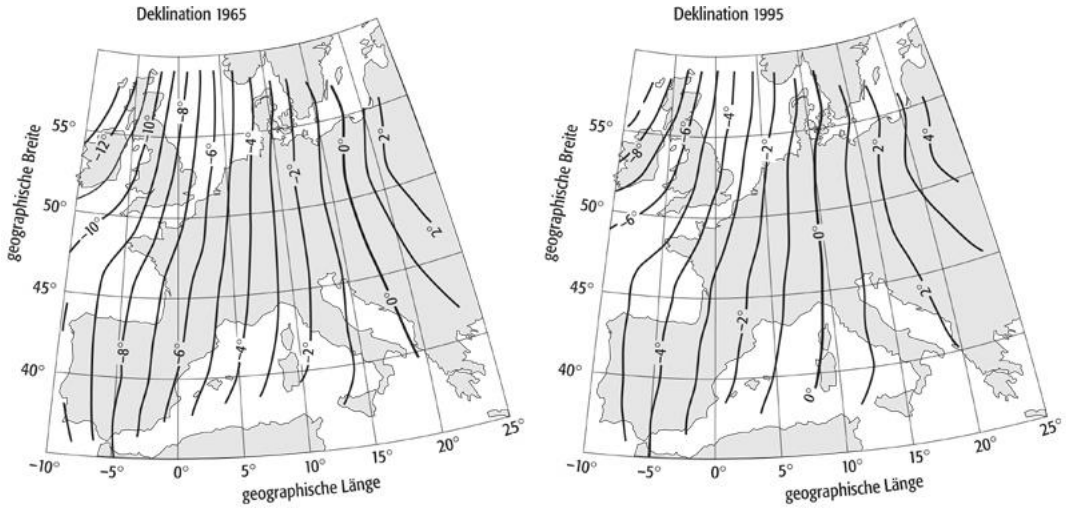


Abbildung 4.6 Magnetische Deklination in Zentraleuropa 1965 (links) und 1995 (rechts) (spektrumverlag 2014)

4.6.3 Gier-, Nick- und Rollwinkel mittels Gyroskop

Das Gyroskop misst die Rotationsgeschwindigkeiten

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

um die körperfesten Koordinatenachsen des Sensors. Die Drehraten beschreiben dabei eine simultane Drehung um alle Koordinatenachsen. Um die Drehraten in den sequentiellen Kontext zu überführen, bedarf es einer Transformierung des Bezugssystems. Hierfür lösen wir die Differentialmatrix

$$\boldsymbol{R}_{GNR} \frac{dR}{dt} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

und erhalten somit nach Titterton und Weston die Änderung der Eulerwinkel als Funktion der vom Sensor erfassten Drehraten zu

$$\dot{\varphi} = \left(\omega_y \, sin(\varphi) + \omega_z \, cos(\varphi)\right) tan(\theta) + \omega_x$$

$$\dot{\theta} = \omega_y \, cos(\varphi) - \omega_z \sin(\varphi)$$

$$\dot{\psi} = \frac{\omega_y \, sin(\varphi) + \omega_z \, cos(\varphi)}{\cos(\theta)}$$

(Titterton und Weston 1997, 40-42 Kap. 3).

Die Integration der Eulergeschwindigkeiten liefert die gesuchten Kardanwinkel

$$\varphi = \int \dot{\varphi} dt$$

$$\theta = \int \dot{\theta} dt$$

$$\psi = \int \dot{\psi} dt.$$

5 Implementierung

Dem in Drei Schichten Architektur vorgestellten Hintergrund folgend, kann die Umsetzung von „Thing Site“ in drei Teilbereiche unterteilt werden. Auf der Ebene „Device“ befindet sich der Raspberry Pi und die dazugehörigen Sensoren, die Ebene „Network“ wird durch die Cloud Lösung „Azure“ verkörpert und schlussendlich der eigentliche primäre Nutzen der Anwendung mithilfe der Ebene „Application“ durch eine Webanwendung realisiert. Die Umsetzung der Ebenen wird in den folgenden Abschnitten genauer beschrieben._Drei_Schichten_Architektur

5.1 Device

Die Aufgabe des Raspberry Pis ist es, Bilder in einem definierbaren Zeitintervall aufzunehmen und diese, zusammen mit der zum Zeitpunkt der Aufnahme ermittelten Orientierung, an die Cloud zu übermitteln. Gewisse Parameter der Programmroutine sollen über Fernzugriff steuerbar sein. Das System soll zudem robust gegenüber äußeren Einflüssen sein: Beim Einsatz auf der Baustelle ist von einer eingeschränkten Netzabdeckung sowie von gelegentlichen Stromausfällen auszugehen.

Im Folgenden werden die Routinen zur Orientierungsbestimmung, Bildaufnahme, Bildablage und Upload beschrieben. Des Weiteren wird darauf eingegangen, wie Funktionsaufrufe der Cloud verarbeitet werden können.

5.1.1 Libraries

Python

- **watchdog** – Der Nutzer kann damit Dateisystemereignisse überwachen und entsprechende Eventhandler konfigurieren. Dateisystemereignisse beinhalten das Ändern, Erstellen, Bewegen und Löschen von Dateien und Verzeichnissen (watchdog).
- **iothub-client** – Geräteseitige Komponente der Python Microsoft Azure IoT SDK. Sie erlaubt es dem Nutzer, Verbindung zu einem Azure IoT-Hub aufzunehmen. Dies beinhaltet unter anderem:
    - Authentifizierung
    - Senden von device-to-cloud Nachrichten
    - Empfangen von cloud-to-device Nachrichten

  - Dateiupload in BLOB-Speicher
  - Verbindungsstatus und Fehler-Reporting (Azure/azure-iot-sdks)
- **Piexif** – Die Library stellt Werkzeuge zur Manipulation der Exif (Exchangeable Image File Format) Daten von Bildern zur Verfügung (Piexif 2018).
- **picamera** – Bietet ein Interface für die Interaktion mit dem Raspberry Pi Kameraboard. Es beinhaltet Funktionalitäten zum Öffnen, Speichern und Darstellen von Bildern, Videos und Streams (Camera Module).
- **Pillow** – Erleichtert den Umgang mit Bilddaten. Nutzer können damit Bilddaten öffnen, speichern und manipulieren (Pillow 2019).
- **RTIMULib2** – Die Library liest die dreiachsigen Sensordaten des Gyroskops, Accelerometers und Magnetometers des IMU aus. Zusätzlich sind verschiedene Funktionalitäten für die Aufbereitung der Sensordaten, wie High- und Lowpassfilter oder Sensorfusionsalgorithmen konfigurierbar. Des Weiteren beinhaltet die Library verschiedene Routinen zur Sensorkalibrierung während und vor dem Betrieb (RTIMULib2)
  - Accelerometer: Der Sensor wird für alle Achsen in den stationären Zustand gebracht. Anhand der minimalen und maximalen Messwerte werden die Achsen entsprechend gewichtet.
  - Magnetometer: Der Sensor wird über alle Achsen bewegt. Die vom Magnetometer erfassten Werte sollten, graphisch dargestellt, idealerweise auf eine Sphäre abgebildet werden. Unregelmäßigkeiten im Erdmagnetfeld, hervorgerufen durch nahegelegene elektronische Geräte, ferromagnetische Metalle und alle Stoffe, die das Feld in irgendeiner Weise verändern, verzerren deren Darstellung. Die Kalibrierung passt die Messungen bestmöglich auf die Sphäre (RTIMULib2 Ellipsoid fit)

5.1.2 Orientierung

In Gier-, Nick- und Rollwinkel mittels Gyroskop wurde erörtert, wie mittels der Sensordaten des IMU die Orientierung der Kamera bestimmt werden kann. Die Gier-, Nick- und Rollwinkel, die für die eindeutige Repräsentation der Orientierung des Geräts im Raum notwendig sind, können entweder mithilfe des Gyroskops oder mit einer Kombination aus Magnetometer- und Accelerometerdaten bestimmt werden. Dabei haben die Sensoren Charakteristika, die sie in verschiedenen Zuständen mehr oder weniger zuverlässig machen.

Das Accelerometer liefert, für sich betrachtet, nur zuverlässige Daten im statischen Zustand des Systems. Jegliche Vibrationen oder hohe Dynamik führen zu Rauschen, beziehungsweise zu einer Verzerrung der Messwerte.

Das Gyroskop hingegen liefert verlässliche Messwerte bezüglich der Drehraten des Systems. Die Sensoren sind dabei unempfindlich gegenüber extern angreifenden Kräften und liefern auch bei hoher Dynamik akkurate Werte. Um die Drehwinkel mit Hilfe des Gyroskops zu bestimmen, müssen dessen Drehraten integriert werden. Messfehler und Rauschen pflanzen sich somit fort. Das Phänomen wird als „driften“ bezeichnet.

Zudem ist die Ausgangslage des Sensors bei der Initialisierung nicht bekannt. Die ermittelten Drehwinkel des Gyroskops beschreiben lediglich die Rotation des Systems bezüglich des Initialisierungszeitpunkts.

Um die Langzeitstabilität des Accelerometers beziehungsweise des Magnetometers mit den Vorzügen des Gyroskops bei hohen Dynamiken zu kombinieren, können die Sensordaten fusioniert werden. Für die Implementierung von „Thing Site“ wurden die Sensordaten mithilfe eines Filters der „RTIMULib“ aufbereitet.

Der Algorithmus schätzt durch lineare Extrapolation die momentane Position mithilfe der Gyroskopdaten der vorangegangenen Messung und dem dazwischenliegenden Zeitinterval. Die mit den Daten des Magnetometers und Accelerometers bestimmten Nick-, Gier- und Rollwinkel liefern eine Referenz gegenüber dem Schätzwert. Für jeden Zeitschritt werden dabei sowohl Referenz als auch Schätzwert bestimmt. Der eigentliche Output wird anteilsmäßig zwischen Referenz und Schätzwert berechnet und durch den slerp-Wert fixiert.

Bei dem für die Implementierung verwendeten slerp-Wert von 0,02 wird der Schätzwert folglich um 1/50 der Differenz zwischen Schätzwert und Referenz korrigiert. Durch einen geringen Anteil von Accelerometer und Magnetometer im Verhältnis zum Output wird versucht, den Drift des auf kurze Zeit zuverlässigeren Gyroskopes zu eliminieren. Zudem wurde zur Rauschunterdrückung ein Tiefpassfilter mit Grenzwert von 41 Hz für Gyroskop und Accelerometer konfiguriert.

5.1.3 Bildaufnahme

Um zu garantieren, dass alle Bilder, die von der Kamera aufgenommen wurden, auch an die Cloud übertragen werden und, bei unerwarteten Programm- und Systemabstürzen, bis dato nicht übermittelte Dateien nicht einfach verloren gehen, werden diese vor

Upload auf dem Dateisystem abgelegt und erst nach erfolgreich abgeschlossener Übertragung gelöscht.

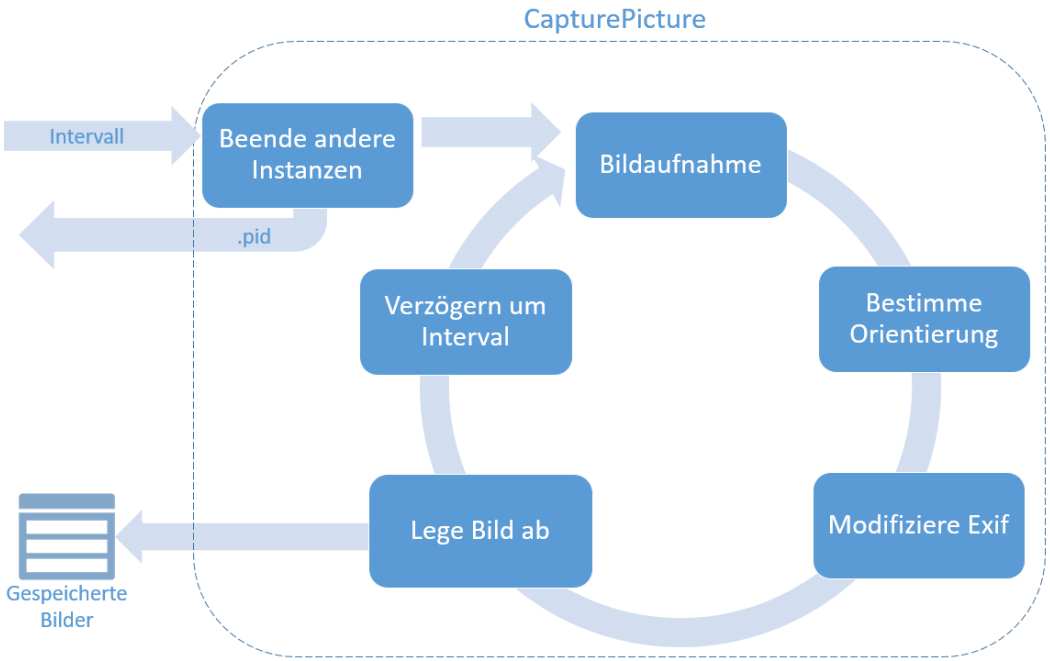


Abbildung 5.1 Aufbau der Routine CapturePicture (Eigene Grafik)

Mit dem Skript „CapturePicture“ wird die Bildablage organisiert. Nach Programmstart wird zunächst mithilfe einer .pid (Process Identifier) Datei überprüft, ob das Skript bereits in einer anderen Instanz läuft. Wenn die Prüfung positiv ausfällt, wird der bereits laufende Prozess beendet und die Prozess-ID des neu gestarteten Skripts hinterlegt. Dazu an späterer Stelle mehr.

Im Anschluss werden Bilder in dem der Funktion übermittelten Intervall aufgenommen und simultan die Orientierung des IMU ausgelesen. Die Orientierung wird in den Exif Daten der Bilder gespeichert und anschließend auf dem Dateisystem mit Datumsstempel abgelegt (Tesic 2005). Somit ist jede Bilddatei eindeutig identifizierbar und einer Orientierung zugeordnet.

Der Raspberry Pi gleicht die Systemzeit abhängig vom installierten Betriebssystem während des Betriebs mit einem zentralen Zeitserver ab. Eine Voraussetzung hierfür ist eine aktive Internetverbindung. Nach Systemabstürzen oder Neustarts wird bis zum Verbindungsaufbau mit dem Zeitserver die „Unix Epoch Time“, der Beginn der Unix Zeitrechnung mit dem 1. Januar 1970, als aktuelle Systemzeit angenommen (How Linux Keeps Track of Time 2001). Da Positionierungs- und Trackingalgorithmen auf

zeitlich synchron aufgenommene Bilder angewiesen sind, wurde mithilfe des RTC-Moduls eine Referenz geschaffen, die auch bei wechselnd guter Internetverbindung zuverlässige Zeitstempel generiert (vgl. Echtzeituhr).

Um die Anzahl an simultan aufgenommenen Bildern zu erhöhen, wurden zudem die bildgebenden Intervalle und das erste Auslösen auf entweder eine Sekunde oder auf ein Vielfaches von fünf Sekunden beschränkt.

5.1.4 „Device to Cloud“ Nachrichten

Die Upload-Routine regelt den Datentransfer von Raspberry Pi zur Cloud. Sie kann durch zwei Ereignisse ausgelöst werden. Nach dem Start des Skripts wird der Ordner, in dem die Bilder nach ihrer Aufnahme abgelegt werden, auf noch nicht verarbeitete Dateien überprüft. Ist dies der Fall, wird der Upload eingeleitet. Zudem abonniert das Skript mithilfe der Library Watchdog Dateisystemereignisse. Wenn dem Ordner durch „CapturePicture“ neue Dateien hinzugefügt werden, übergibt der Eventhandler den Dateipfad und initiiert die Upload-Routine.

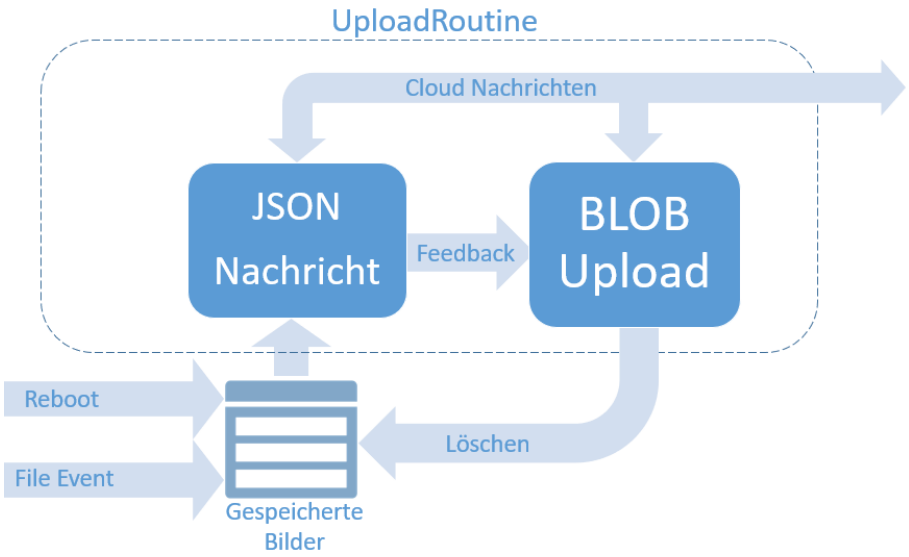


Abbildung 5.2 Aufbau der Routine UploadRoutine (Eigene Grafik)

Der Upload als solches wird dabei durch das von Azure zur Verfügung gestellte „Azure IoT Hub Device SDK“ geregelt. Unter Angabe des zu verwendenden Protokolls und der Connection Strings können so Dateien und Nachrichten an die Cloud übermittelt werden. Die Endpoints und Connection Strings wurden vorher cloudseitig definiert. Sensible Informationen zur Authentifizierung müssen dabei nicht im Code verankert werden.

Es ist zu beachten, dass Azure SDKs zwar in diversen Programmiersprachen veröffentlicht werden, jedoch einzelne Features anbieterübergreifend nur teilweise oder gar nicht unterstützt. Der komplette Umfang an Funktionalitäten ist meistens nur für .NET Plattformen erhältlich. So musste die Implementierung der Gerätenachrichten mittels http und nicht wie ursprünglich angedacht mit mqtt vollzogen werden.

Die Client SDK unterscheidet beim Upload zwischen „blob“ oder „message“. Davon abhängig wird die POST-Anforderung entweder an einen Endpunkt zum Anfordern einer SAS-URI zur Dateiverarbeitung oder zur Nachrichtenverwaltung gesendet. Für die Implementierung wurden beide Funktionalitäten verwendet.

Zunächst sendet das Skript eine JSON (JavaScript Object Notation) -kodierte Nachricht mit datenbankrelevanten Informationen an den „message“-Endpoint (vgl. Abbildung 5.3). Diese enthält Informationen zur Orientierung des Geräts während der Aufnahme, den Zeitstempel, die Geräte-ID, sowie die cloudseitige URI des Bildes. Mit Ausnahme der BLOB-URI wird die Nachricht aus den Informationen der Metadaten des Bildes generiert.

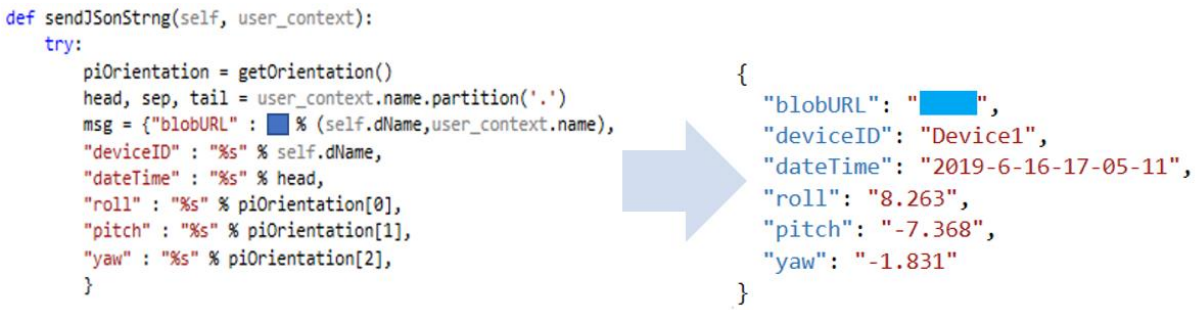


Abbildung 5.3 Ausschnitt der Funktion "sendJSonStrng" und mögliches resultierendes JSON Objekt (Eigene Grafik)

Nach erfolgreicher bzw. fehlerhafter Zustellung verarbeitet die Routine eine „file notification message“. Bei negativer Beurteilung wiederholt sich die IoT-Nachricht, bei positivem Feedback sendet das Skript das zur Nachricht gehörende Bild an einen BLOB-Speicher. Nach erfolgreicher Zustellung des Bildes wird es vom Dateisystem entfernt.

Durch die Entkoppelung des Uploads eines Bildes und dessen Metadaten wird eine potentielle Fehlerquelle geschaffen. Sollte zwischen der erfolgreichen Zustellung der IoT-Nachricht mit den datenbankrelevanten Informationen und dem Dateiupload ein Systemneustart durchgeführt werden, resultiert dies in einem Datenbankeintrag ohne korrespondierenden BLOB. Beim erneuten Zustellen der vom Inhalt her identischen

Nachricht nach Systemstart muss also sichergestellt sein, dass die Datenbank keine Doppeleinträge akzeptiert.

Grundsätzlich wäre es möglich gewesen, den Upload auf die reine Bilddatei zu beschränken, die Metadaten des Bildes mit Hilfe einer „Azure Function“ zu extrahieren und daraufhin die relevanten Informationen an die Datenbank zu übergeben. Die einfache Integration des Event Hubs an nachfolgende Azure Services zur Nachrichtenverwaltung, wie z.B. „Stream Analytics“ oder „Service Bus“ Tools, macht die direkte Nutzung des „message“-Endpoints attraktiv. Zudem soll die entwickelte Infrastruktur die Übermittlung von Sensordaten losgelöst von der Bildübertragung ermöglichen, um zukünftige Applikationen zu unterstützen, die keine Koppelung von Telemetriedaten an Bilder vorsehen.

5.1.5 „Cloud to Device“ Nachrichten

Um das Verhalten des Pis beeinflussen zu können, wurde eine Routine zur Auswertung von „Cloud to Device“-Nachrichten (C2D; Nachrichten der Cloud für ein spezifisches Endgerät) implementiert. Mithilfe des „Azure IoT Hub Client SDK“ wird periodisch der Endpoint für C2D-Nachrichten abgefragt. Falls Nachrichten übermittelt werden, wird der decodierte Inhalt auf vorher definierte Funktionsaufrufe geprüft. Nach erfolgreicher Zustellung der Nachricht wird eine Zustellungsbenachrichtigung an die Cloud übermittelt.

Als grundlegende Funktionen wurden „Reboot“ und „Interval“ implementiert. „Reboot“ startet das Gerät neu, „Interval“ verändert den Zeitabstand, in dem Bilder aufgenommen werden, bzw. schaltet die Bildaufnahme ab. Dabei wird eine neue Instanz der „CapturePicture“ Routine gestartet. Um die Mehrfachausführung zu verhindern, werden, wie in Bildaufnahme erläutert, bereits laufende Instanzen der Routine beendet.

Der übermittelte Wert des neuen Bildintervalls wird ebenfalls in einer Datei hinterlegt. So kann nach einem Neustart die gleiche Frequenz aufgegriffen werden. Der Wert wird vom Anwendungsbackend zusätzlich in der Datenbank gespeichert und könnte auch über eine Datenbankabfrage, beziehungsweise indirekt über eine „Device to Cloud“-Nachricht (D2C) mit Antwortauswertung bereitgestellt werden. Nachdem die Datenbank aus Sicherheitsgründen keine direkte Verbindung mit den Endgeräten haben sollte und die Abfrage nur bei intakter Internetverbindung ausgeführt werden kann, wurde eine lokale Speicherung der Parameter bevorzugt.

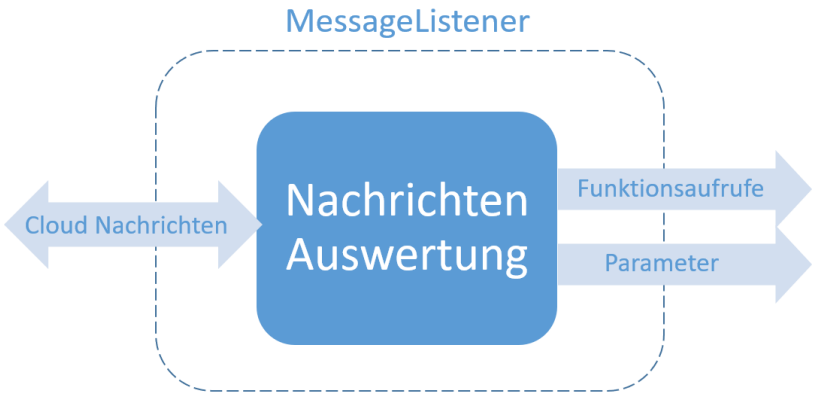


Abbildung 5.4 Aufbau der Routine MessageListener (Eigene Grafik)

5.2 Network

Azure stellt das Bindeglied zwischen Devices und Anwendung dar. Die Cloud regelt die Geräteverwaltung, liefert die Infrastruktur zur Speicherung und Organisation von eingehenden Daten und fungiert zusätzlich als Nachrichtenbroker zwischen IoT-Geräten und Backendanwendung.

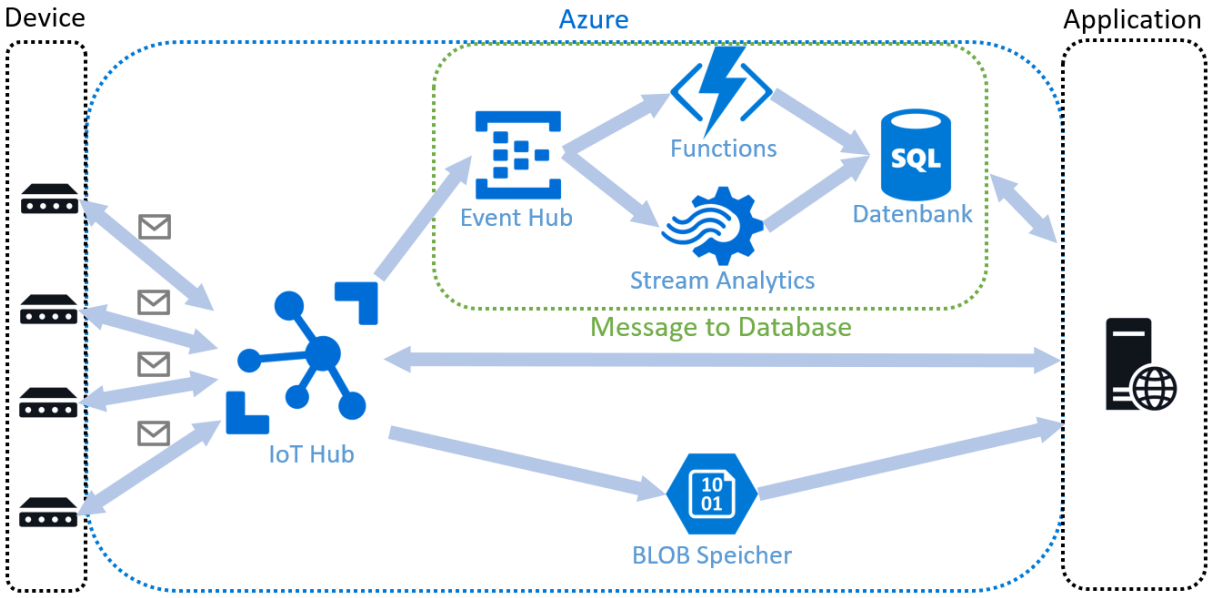


Abbildung 5.5 Interaktion zwischen "Device", "Azure" und "Application" (Eigene Grafik)

5.2.1 Libraries

.NET

- **Belgrade** – eine .NET library für die Abstraktion und Vereinfachung von SQL-Befehlen. Sie regelt das Öffnen, Schließen und Errorhandling von Datenbankverbindungen und überführt SQL-Befehle in einen objektorientierten Ansatz (Belgrade SqlClient 2019).

5.2.2 Konfiguration des „IoT-Hub“

Zunächst müssen dem IoT-Hub im Azure Portal Geräteinstanzen hinzugefügt werden. Dadurch werden alle benötigten Connection Strings und Zugriffsberechtigungen für die jeweiligen Devices generiert.

Wie in „Device to Cloud“ Nachrichten beschrieben, sendet die geräteseitige Anwendung Nachrichten an den „messaging“-Endpoint des IoT Hubs. In den Optionen für „Nachrichtenrouting“ können die Nachrichten selektiert und an unterschiedliche nachgeschaltete Endpunkte weitergeleitet werden. Da unsere Anwendung vorerst nur eine „Route“ für eingehende Nachrichten vom Typ „message“ besitzt, werden alle eingehenden Nachrichten an den in den IoT Hub integrierten Event Hub weitergeleitet.

Da neben Nachrichten auch Dateien von den Devices an die Cloud übermittelt werden sollen, muss im IoT Hub zusätzlich ein Endpunkt einer zuvor generierten Speicherressource integriert werden. Alle Dateien, die der Geräteclient an den „blob“-Endpunkt sendet, werden in dieser Speicherressource abgelegt.

5.2.3 Pipeline „Message to Database“

Die Webanwendung benötigt die von den Geräten übermittelten Daten in aufbereiteter Form. Wie in Datenverwaltung erörtert, ist es für den Anwendungsfall dieser Arbeit zweckmäßig, auf einen URI-basierten Ansatz zur Organisation von Bilddaten zurückzugreifen. Die Pipeline „Message to Database“ demonstriert, wie Gerätenachrichten mit Hilfe von Azure-Services zuverlässig in Datenbankeinträge überführt werden können.

Eventhub kompatible Services

Der Ausgangspunkt der Pipeline ist der Eventhub. Als Konsumenten für eingehende „events“, wie z.B. von Endgeräten gesendete Nachrichten, wurde die Implementierung für „Stream Analytics“-Services und mithilfe von „Azure Functions“ vollzogen.

Stream Analytics

„Stream Analytics“-Services stellen eine zuverlässige, einfach zu implementierende Möglichkeit zur Eventverarbeitung dar. Als Quelle, also Input, für den Service wird der für die Nachrichtenübermittlung verwendete Event Hub angegeben. Ein Teil des Services ist die integrierte Deserialisierung des Nachrichteninhalts. Dieser kann innerhalb

der Transformationsabfrage mithilfe einer deklarativen SQL-ähnlichen Sprache gefiltert und für den Output vorbereitet werden. Beim Output handelt es sich um einen nachgeschalteten Service, oder, wie in diesem Fall, eine Tabelle der SQL Datenbank. Bei der Definition der Senke kann zwischen externen oder Azure-internen Datenbanken unterschieden werden. Eine Anbindung externer Ressourcen ist also an dieser Stelle möglich.

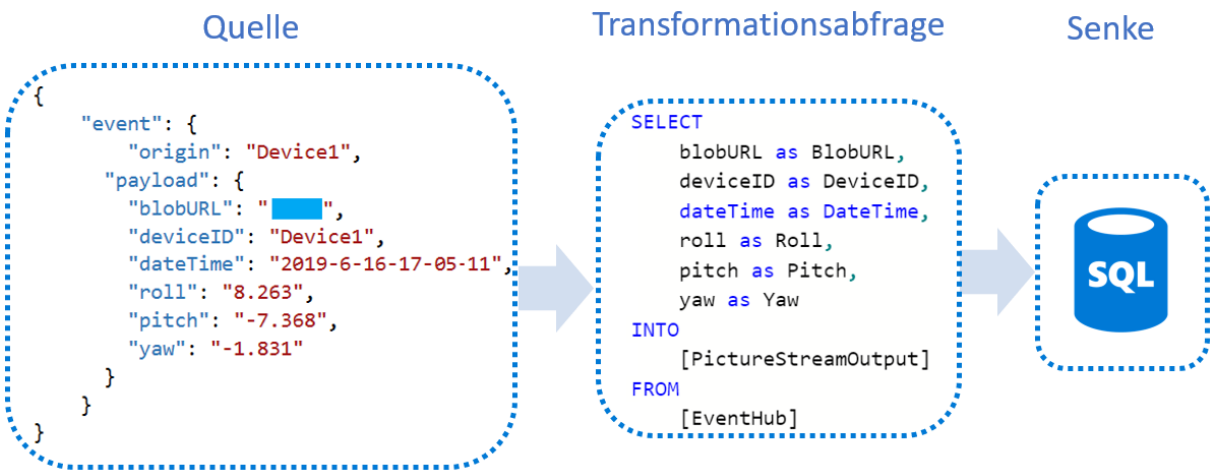


Abbildung 5.6 Aufbau eines Stream Analytics Jobs (Eigene Grafik)

Wie Abbildung 5.6 zu entnehmen ist, wird die in „Device to Cloud“ Nachrichten beschriebene, geräteseitig serialisierte Nachricht, aus dem Event Hub gelesen und mithilfe der Transformationsabfrage in eine Datenbank eingefügt.

Die einfache Umsetzung sowie die Zustellungs- und Verarbeitungsgarantien machen „Stream Analytics“ zur ersten Wahl, falls finanzielle Aspekte eine untergeordnete Rolle spielen. Die im Rahmen der Entwicklung entstandenen Kosten ließen jedoch Zweifel aufkommen, ob mit dieser Herangehensweise ein wirtschaftliches Produkt entwickelt werden kann.

Azure Functions

Aus diesem Grund wurde mit „Functions“ eine kostengünstigere Alternative entwickelt. Diese Funktionen können durch im Event Hub stattfindende Ereignisse ausgelöst werden. In dem hier dargestellten Anwendungsfall ist ein Ereignis der Eingang von Gerätenachrichten.

Zunächst wird die Trigger Function erstellt und Übergabeparameter, Verbindungseinstellungen und Abhängigkeiten definiert. Azure bietet hierfür eine eigene cloudbasierte Entwicklungsumgebung. In dieser kann der Inhalt der Funktion implementiert werden.

```
public static async void Run(string myEventHubMessage, TraceWriter log)
{
    if (String.IsNullOrWhiteSpace(myEventHubMessage))
        return;
    try
    {
        string ConnString = ConfigurationManager.ConnectionStrings["azure-db-connection"].ConnectionString;
        var cmd = new Command(ConnString);
        var sqlCmd = new SqlCommand("ImportEvents");
        sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;
        sqlCmd.Parameters.AddWithValue("event", myEventHubMessage);
        await cmd.ExecuteNonQuery(sqlCmd);
    }
```

Abbildung 5.7 Ausschnitt der Azure Function zur Interaktion mit Datenbanken (Eigene Grafik)

Die Funktion leitet die beim Event Hub eingehenden Nachrichten mithilfe der „Belgrade SqlClient“ Library an die Datenbank weiter.

Die Library führt eine gespeicherte Datenbankprozedur „ImportEvents“ mit der jeweiligen beim Event Hub eingehenden Gerätenachricht „myEventHubMessage“ aus. Die Function ist damit nur der Überbringer; die Decodierung und Einfügeoperation wird datenbankseitig geregelt.

Falls die Funktion nicht erfolgreich ausgeführt werden kann, da beispielsweise die Datenbank offline oder nicht erreichbar ist, geht die Nachricht verloren. Im Gegensatz zur Implementierung mit „Stream Analytics“ musste ein Errorhandling implementiert werden, dass die Zustellung aller Nachrichten garantiert.

Durch Modifikation der Verbindungszeichenfolge, gespeichert in der Variablen „ConnString“, können auch hier externe Datenbanken integriert werden (azure servicefabric).

Datenbank Prozedur

Die Implementierung der Prozedur ist ein thematischer Vorgriff auf das Kapitel Datenbank, jedoch stellt sie den Abschluss der Pipeline „Message to Database“ dar und wird

deshalb bereits an dieser Stelle erläutert. Details bezüglich der verwendeten Tabellen und deren Deklarierung sind an entsprechender Stelle zu finden.

```
CREATE PROCEDURE dbo.ImportEvents   @event NVARCHAR(MAX)
AS  BEGIN
    MERGE INTO dbo.event AS ExistingEvent
    USING (SELECT *
       FROM OPENJSON(@event)
       WITH ([blobURL] nvarchar(100), [deviceID] nvarchar(100),
       [dateTime] datetime2(7), [roll] nvarchar(100),
       [pitch] nvarchar(100), [yaw] nvarchar(100)))
       AS NewEvent
       ON (ExistingEvent.BlobURL = NewEvent.blobURL)
    WHEN NOT MATCHED THEN
        INSERT (BlobURL, DeviceId, DateTime, Roll, Pitch, Yaw)
        VALUES(NewEvent.blobURL, NewEvent.deviceID, NewEvent.dateTime, NewEvent.roll, NewEvent.pitch, NewEvent.yaw);
END
```

Abbildung 5.8 Datenbankprozedur ImportEvents (Eigene Grafik)

Die von der Azure Function aufgerufene Datenbankprozedur erhält als Input das codierte Event. Mithilfe der OPENJSON Funktion wird dieses decodiert (azure servicefabric). Damit die in „Device to Cloud“ Nachrichten erwähnten Doppeleinträge ausgeschlossen werden, ist es zweckmäßig, die neuen Einträge vor dem Einfügen mit den Einträgen der Datenbank zu vergleichen. Falls das Bild noch nicht in der Datenbank existiert, wird es dem Datensatz hinzugefügt.

5.2.4 Datenbank

Das Anwendungsbackend benötigt für die Visualisierung und Verarbeitung der gespeicherten Bilddaten eine geeignete Infrastruktur. Diese wurde mit dem Cloud-internen „Azure SQL-Datenbank“-Service realisiert. Wie bei der Implementierung der anderen Services gezeigt wurde, können aber auch externe Datenbanken problemlos in das System integriert werden. Der Service war jedoch die pragmatischste Lösung.

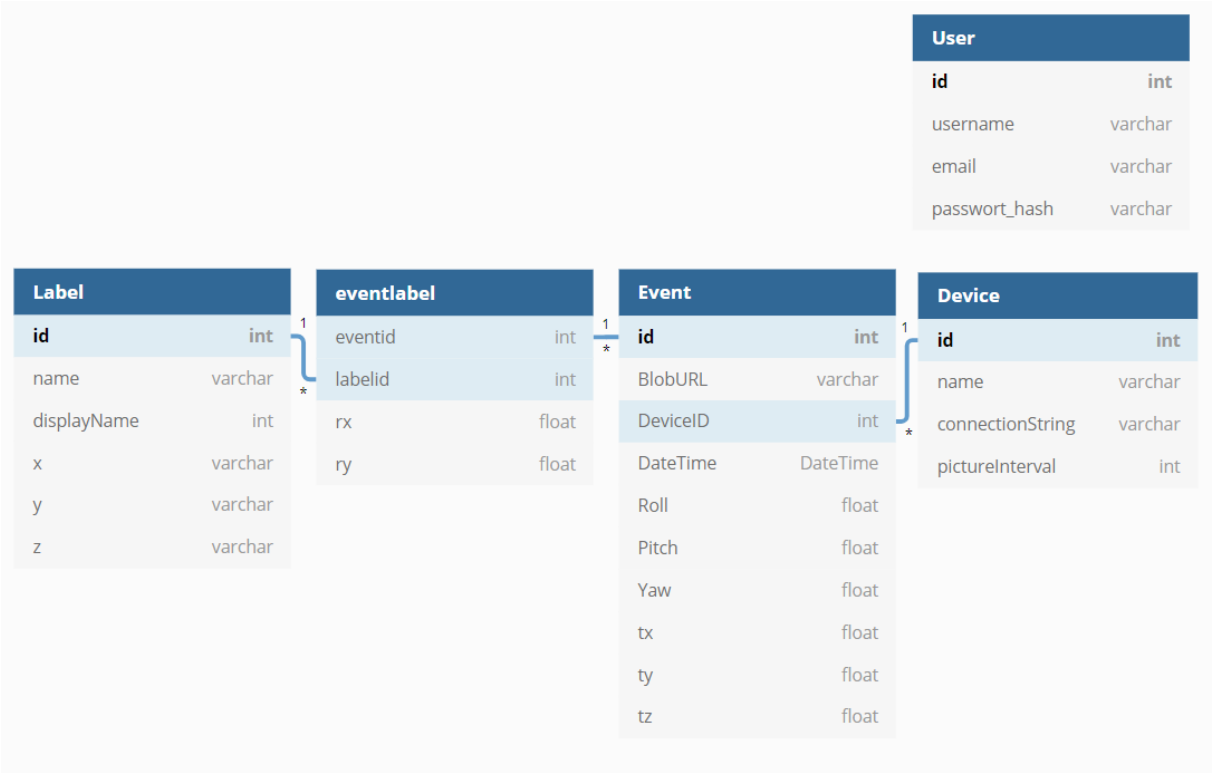


Abbildung 5.9 Aufbau der Datenbank (Eigene Grafik)

Event

Parameter, die durch die Prozedur „ImportEvent“ dem Datensatz hinzugefügt werden, befinden sich ausschließlich in der Tabelle „Event“. Diese hält neben den Werten für „BlobURL“, „DeviceID“ und „DateTime“, die relative Orientierung „Roll“, „Pitch“ und „Yaw“. „Tx“, „Ty“ und „Tz“ speichern die approximierte Position des Bildes. Die Werte werden vom Anwendungsbackend berechnet.

Jedes Event ist einem registrierten Gerät zugeordnet. Registrierte Geräte sind in der Tabelle „Device“ gespeichert.

Device

Die Tabelle enthält alle für das Backend relevanten Informationen. „ConnectionString“ enthält die Verbindungszeichenfolge des C2D Endpoints für die Übermittlung von Nachrichten an das jeweilige Gerät. „pictureInterval“ speichert Session-übergreifend das konfigurierte Bildgebungsinterval der Kamera.

Label

„Label“ speichert alle Daten eines eingemessenen Fixpunktes in den 3D-Szenenkoordinaten „x“, „y“ und „z“ sowie einen Trivialnamen des Punktes.

Eventlabel

Die Verbindungstabelle „eventlabel“ repräsentiert die Verbindung zwischen „Events“ und „Labels“. Sie speichert welche Labels auf den jeweiligen Bildern erkannt werden. Die relative Position der Labels auf dem Bild wird in den Spalten „rx“ und „ry“ erfasst.

User

Zudem wurde ein Usermanagement zur Account- und Rechteverwaltung implementiert. Da es sich hierbei um eine Standardaufgabe aus dem Bereich der Webprogrammierung handelt, wird im Nachfolgenden nicht näher darauf eingegangen.

5.3 Application

Auf der letzten Ebene von „Thing Site” wird der Nutzen für den Anwender realisiert. Dieser besteht aus einer Webanwendung zur Unterstützung der Bauüberwachung.

Der folgende Abschnitt beschreibt, wie aus den datenbankseitig hinterlegten Bilddaten Referenzpunkte extrahiert werden können, die für anschließende Positionierungs- und Trackingalgorithmen benötigt werden. Hierfür wird zunächst das Training des Neuronalen Netzwerks und die Kalibrierung der Kameras beschrieben. Im Anschluss wird die Funktionsweise der Algorithmen und der Aufbau der Weboberfläche dargelegt.

Zur Aufbereitung und Interpretation der Daten wurde ein neuronales Netzwerk eingesetzt.

Die Klassifizierung durch das NN findet für den Prototyp auf dem Anwendungsserver statt, obwohl eine in Azure integrierte Lösung auf den ersten Blick eleganter erscheint. Die für die Klassifizierung benötigte GPU-Zeit (Graphic Processing Unit) konnte jedoch leider nicht vom Provider im Rahmen des Testkontingents bereitgestellt werden.

5.3.1 Libraries

Python

- **Flask** – Flask ist ein Python-basiertes Webframework zur einfachen Gestaltung von Webanwendungen. Das Framework regelt die Kommunikation zwischen Webserver und Webanwendung über WSGI (Web Server Gateway Interface) (Flask).

  Flask baut auf folgenden Abhängigkeiten auf:

  - **Jinja2** – Template-Engine zur Verarbeitung von HTML (Hypertext Markup Language) Code. Die Engine erlaubt es dem Nutzer, HTML-Code mit Python Statements zu erweitern und somit Code dynamisch zu steuern. Des Weiteren integriert sie die Prinzipien der Vererbung und ermöglicht somit modularen Code (Jinja2 2018).
  - **Werkzeug** – Dieses Toolset für WSGI-Anwendungen regelt den Austausch von Informationen zwischen Webanwendung und Server. Teil des Toolsets sind routing Funktionalitäten, Abstraktion von „request“ und „response“ Parametern in Objekten, ein lokaler Entwicklungsserver und ein Debugger (Werkzeug).
- **Matplotlib** – Dient der Visualisierung von Datensätzen mit Hilfe von Grafiken (Matplotlib: Python plotting 2019).
- **WTForms** – Toolset zur Validierung und Visualisierung von Webforms. Beinhaltet eine Reihe von Methoden und Klassen für die effiziente Bewältigung von Standardaufgaben aus der Webprogrammierung (Passworthashing, Validators, Errorhandling). Die vom User über die einzelnen Fields der Forms bereitgestellten Eingaben können somit bewertet und selektiert und mit Hilfe von Validators auf ihre syntaktische Korrektheit geprüft werden. Potenziell bedrohliche Eingaben können so ausgeschlossen werden (WTForms 2018).
- **Imageio** – Ermöglicht das Öffnen, Lesen, Schreiben und Speichern von Bilddaten (imageio).
- **SQLAlchemy** – Mit der Web Application von „Thing Site“ wird es Nutzern und bis zu einem gewissen Grad auch Dritten ermöglicht, mit der Datenbank zu interagieren. Vom Nutzer zu bedienende Textfelder werden mit der Datenbank abgeglichen und basierend auf dieser Abfrage Informationen ausgegeben. Bei einer SQLIA (Structured Query Language Injection Attack) wird versucht durch Eingabe von SQL Operatoren und Anweisungen in die offenen Schnittstellen (Log-In Forms, Suchfelder etc.) zusätzliche Daten auszugeben oder unerwünschtes Verhalten der Datenbank hervorzurufen (Essential SQLAlchemy).
  SQLAlchemy ermöglicht es dem Benutzer SQL-Befehle nicht über die Manipulation von Textbausteinen zu kontrollieren, sondern verfolgt einen objekt- und funktionsorientierten Ansatz (Object-relational mapper, ORM). Diese zusätzlich eingeführte Abstraktion zwischen Datenbank und User-Input verhindert den Missbrauch von Schnittstellen für manipulative Zwecke und erlauben es den

Entwicklern, Datenbanken im Python Syntax zu bedienen (SQLAlchemy - The Database Toolkit 2019).

- **Bootstrap** – Frontend-Frameworks wie Bootstrap unterstützen den Nutzer beim Aufbau von Websites. Enthalten sind Cascading Style Sheets (CSS), HTML und JavaScript (JS) Komponenten, mit denen funktionale, gestylte Anwendungen erstellt werden können. Zudem sind die meisten zur Verfügung gestellten Bootstrap-spezifischen HTML-Elemente responsiv, d.h. sie passen ihre Darstellung und Handhabung an das Endgerät des Clients an (Otto et al. 2019).
- **OpenCV** – OpenCV ist eine Library mit einem breiten Spektrum an Funktionalitäten zur Bewältigung von gängigen Problemstellungen aus dem Bereich Computer Vision und Machine Learning. Darin enthalten sind Routinen zur Ermittlung der Kameraparameter (OpenCV).
- **TensorFlow** – Ein Framework für numerisches Rechnen zur datenstromorientierten Programmierung. TensorFlow erlaubt es dem Nutzer, sogenannte Datenstromgraphen zu entwickeln. Ein Graph besteht aus einer Serie von Knoten. Jeder Knoten repräsentiert eine Rechenoperation. Der In- und Output jedes Knoten ist durch ein mehrdimensionales Array, auch *Tensor*, definiert. TensorFlow wird hauptsächlich im Feld des maschinellen Lernens für die Implementierung, Kalibrierung und Analyse von Neuronalen Netzwerken eingesetzt (TensorFlow).

JavaScript

- **jQuery** – Vereinfacht den Umgang mit JavaScript. Das Ziel ist es, HTML Code übersichtlich zu gestalten und plattformübergreifend zu standardisieren. Die Library fungiert dabei als zusätzliche Abstraktionsebene über dem JS-Code. jQuery wird für die Manipulation von HTML und CSS sowie Eventhandling, Animationen und Einbinden von asynchronen Funktionen clientseitig eingesetzt (jQuery).

5.3.2 CNN

Die Tracking- und Positionierungsalgorithmen des Anwendungsbackends benötigen die relative Position von Referenzpunkten auf dem Kamerabild, um daraus 3D-Echtweltkoordinaten rekonstruieren zu können. Hierfür wurde ein CNN auf benutzerdefi-

nierte Referenzpunkte trainiert. Das CNN bewertet nach Upload von Bildern die Auftretenswahrscheinlichkeit der trainierten Objekte und gibt deren Position anhand von Bounding Boxen aus.

Architektur

Pro Klasse stand eine relativ geringe Anzahl von Trainingsdaten zur Verfügung. Vorrangegangene Untersuchungen zeigten aber, dass auch mit verhältnismäßig wenig Daten gute Ergebnisse erzielt werden können (Shin et al. 2016). Dieser Annahme folgend wurden mithilfe der Tensorflow Object Detection API, ein von TensorFlow bereitgestelltes Framework zur Bewältigung von Aufgabenstellungen aus dem Bereich der Objekterkennung, zwei CNN zunächst auf einen Datensatz von 300 von Hand gelabelten Bildern trainiert (Jonathan Huang et al.). Zudem standen 80 Testbildern zu Verfügung.

Bei der Auswahl der Netzwerkarchitektur wurden die Untersuchungen von Huang et al. herangezogen. Diese verglichen verschiedene, auf den COCO-Datensatz trainierte CNNs hinsichtlich ihrer Genauigkeit der Klassifizierung und der Approximierung von Bounding Boxen im Verhältnis zu benötigter GPU-Zeit und Speicher (Huang et al. 2016; azure servicefabric; COCO - Common Objects in Context 2018). Da für die Klassifizierung der Bilddaten der vorliegenden Arbeit diese Ressourcen keine limitierenden Faktoren darstellten und für den Prototypen zudem keine Klassifizierung in Echtzeit angestrebt wird, wurde die Implementierung zunächst mit der performantesten Architektur „Faster RCNN“ mit dem Feature Extractor „Inception Resnet (v2)“ vollzogen.

Da die Genauigkeit der Klassifizierung und der Approximierung der Bounding Boxen maßgeblich von der Qualität und Quantität der Trainingsdaten abhängt, und zudem die Annahme als nicht sicher anzusehen ist, dass die Ergebnisse der Untersuchung für einen wesentlich kleineren Datensatz reproduzierbar sind, wurde zusätzlich „Faster RCNN“ mit dem wesentlich parameterärmeren Feature Extractor „Inception v2“ auf den gleichen Datensatz trainiert.

Training

Für das Training wurden die von TensorFlow bereitgestellten Checkpoints für das „Faster RCNN Inception ResNet v(2)“ sowie „Faster RCNN Inception v2“, welche ursprünglich auf den COCO-Datensatz trainiert wurden, verwendet. Dem Ansatz von Shin et al. folgend, wurden alle Variablen des Netzwerks während des Trainingsvorgangs als trainierbar markiert (Shin et al. 2016). Die letzten Classification-Layer des

Pretrained Network werden neu initialisiert, um eine vom COCO-Training abweichende Anzahl von Klassen zu ermöglichen.

Der neue Datensatz umfasst zwei Klassen. Jede der Klassen repräsentiert einen Trackingmarker. Diese sind in Abbildung 5.10 dargestellt.

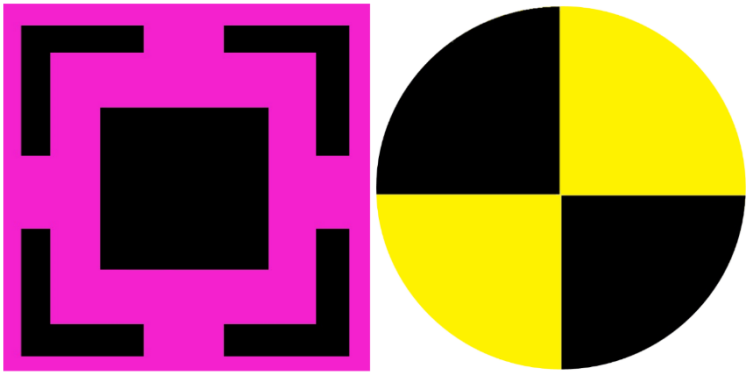


Abbildung 5.10 Trackingmarker Klasse 1 (links) und Klasse 2 (rechts) (Eigene Grafik)

Für jede Klasse standen zunächst ca. 250 Trainingsbilder und 60 Testbilder zur Verfügung. Nach ca. 100.000 Trainingsiterationen erreichten beide Architekturen ein konstantes Niveau bezüglich der mean Average Precision (Maß für Klassifizierungsgüte; vgl. (Hui 2019)). Der Training Loss pro Iteration konvergierte gegen null. Daraufhin folgte ein Vergleich der beiden Architekturen. Tabelle 4 und Tabelle 5 zeigen die jeweilige Konfusionsmatrix des Testdatensatzes.

Die vertikalen Spalten mit den jeweiligen Klassen geben den vom NN vorhergesagten Wert an, die horizontalen Reihen bilden den eigentlichen Zielwert ab. In den Feldern der Reihe „keine“ wird die Anzahl von Klassifizierungen gelistet, die durchgeführt wurden, obwohl keine Klassen auf dem Bild waren. Die Spalte „keine“ zählt ausgelassene Klassifizierungen.

Tabelle 4 Konfusionsmatrix des ResNet Inception (v2) Testdataset (Eigene Darstellung)

		NN - Vorhersage		
	ResNet Inception (v2)	Klasse 1	Klasse 2	keine
Referenz	Klasse 1	60	0	0
	Klasse 2	0	52	1
	keine	4	11	

Tabelle 5 Konfusionsmatrix des Inception V2 Testdataset (Eigene Darstellung)

		NN - Vorhersage		
	Inception v2	Klasse 1	Klasse 2	keine
Referenz	Klasse 1	60	0	0
	Klasse 2	0	53	0
	keine	2	6	

Keines der NN klassifizierte also Klasse 1 für Klasse 2 bzw. umgekehrt. Die Anzahl der fälschlichen Klassifizierungen bei „ResNet Inception (v2)“ ist jedoch doppelt so hoch wie bei „Inception v2“.

Da der genutzte Datensatz bis dato nur aus Bildern erzeugt durch die Kamera „Camera Module v1“ bestand, wurden bedingt durch die deutlich abweichende Bildgebung der „Camera Module v2“, der Trainingsdatensatz auf ca. 400 Bilder pro Klasse und der Testdatensatz auf ca. 100 Bilder pro Klasse erweitert. Nachdem der Feature Extractor „Inception v2“ bis dato bessere Werte bezüglich der erreichten Klassifizierungsgüte lieferte und die benötigte Trainingszeit in etwa 20% der von „ResNet v(2)“ entsprach, wurde das Training nur mit diesem NN fortgesetzt.

Tabelle 6 Konfusionsmatrix des Inception V2 Testdataset 400.000 Iterationen (Eigene Darstellung)

		NN - Vorhersage		
	Inception v2	Klasse 1	Klasse 2	keine
Referenz	Klasse 1	108	0	0
	Klasse 2	0	95	2
	keine	3	6	

Das Training stabilisierte sich nach etwa 200.000 zusätzlichen Iterationen. Der prozentuale Anteil der fälschlich getroffenen Klassifizierungen halbierte sich (vgl. Tabelle 6). Um die Ergebnisse zu bestätigen, wurde ein unabhängiger Validierungsdatensatz erstellt und abermals ausgewertet.

Tabelle 7 Konfusionsmatrix des Inception V2 Validationdataset 400.000 Iterationen (Eigene Darstellung)

		NN - Vorhersage		
	Inception v2	Klasse 1	Klasse 2	keine
Referenz	Klasse 1	90	0	3
	Klasse 2	0	80	6
	keine	1	2	

Die zuvor erzielten Ergebnisse, festgehalten in Tabelle 6, sind weitestgehend konsistent mit den Ergebnissen des Validierungsdatensatz (Tabelle 7). Lediglich der Anteil der nicht klassifizierten Bilder stieg an. Dies liegt mit großer Wahrscheinlichkeit an subjektiv relativ schwer zu klassifizierenden Bildern, die dem Datensatz bewusst hinzugefügt wurden.

5.3.3 Kameraparameter

Wie in Kamerakalibrierung erläutert, sind die von der Kamera aufgenommenen Bilder fehlerbehaftet. Für die Bestimmung der Korrekturwerte für radiale und tangentiale Verzerrung wurde die Library „OpenCV“ eingesetzt. Hierfür wird ein Objekt mit bekannten Abmessungen, in unserem Fall ein Schachbrettmuster, aus verschiedenen Blickwinkeln mit der Kamera des Raspberry Pi aufgenommen. Die Library extrahiert die Begegnungspunkte der schwarzen und weißen Kästchen. Da die Abstände zwischen den Kästchen bekannt sind, kann die theoretische Lage der korrespondieren Bildpunkte mit der in Kamera Modell eingeführten Beziehung

$$\boldsymbol{x} = \boldsymbol{K} * \boldsymbol{E} * \boldsymbol{p}$$

beschrieben werden. Durch eine Ausgleichsrechnung wird das Gleichungssystem hinsichtlich des Projektionsfehlers optimiert und somit die Korrekturwerte für radiale und tangentiale Verzerrung approximiert. Zudem werden die Parameter der Kameramatrix berechnet (rob2-08-camera-calibration).

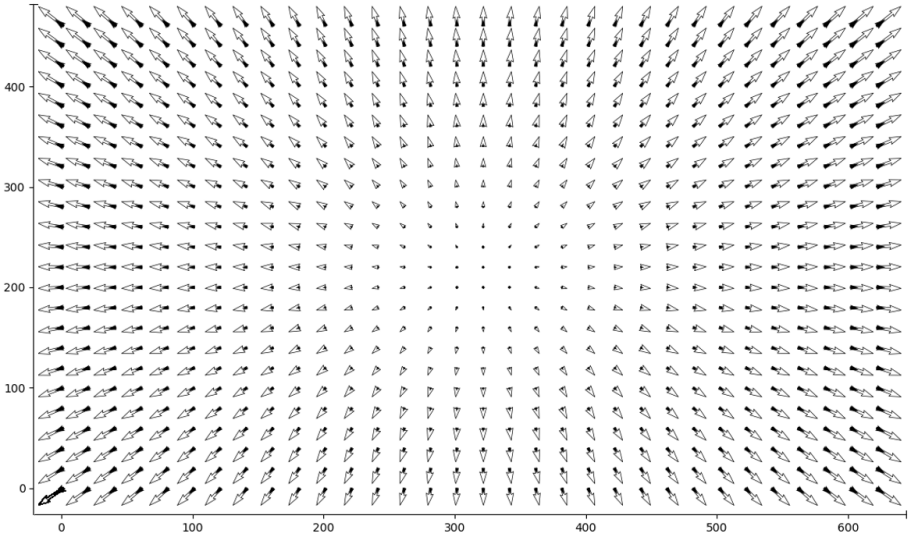


Abbildung 5.11 Vektoren für entzerrte Pixel. Kameraparameter Cam v1 (Eigene Grafik)

Die Kameras wurden mit jeweils 45 Einzelbildern kalibriert. Die Ergebnisse der Kalibrierung sind in Tabelle 8 zu finden.

Tabelle 8 Verzerrungskoeffizienten der eingesetzten Kameras (Eigene Darstellung)

k_1	k_2	k_3	p_1	p_2	RMSE	Modell
0.236	-1.408	0.234	-0.004	-0.001	0.024	Cam v1
0.211	-0.565	0.463	0.001	-0.002	0.025	Cam v2 - 1
0.204	-0.540	0.415	0.003	0.000	0.024	Cam v2 - 2
0.208	-0.501	0.377	0.000	-0.002	0.026	Cam v2 - 3

Die Parameter der Kameramatrizen werden in Tabelle 9 gezeigt.

Tabelle 9 Kameraparameter der eingesetzten Kameras (Eigene Darstellung)

$f * k_u$	$f * k_v$	u_o	v_o	Modell
602.7	603.1	316.2	213.9	Cam v1
498.0	499.2	313.1	256.0	Cam v2 - 1
499.2	500.0	317.5	256.3	Cam v2 - 2
496.8	497.5	322.4	240.6	Cam v2 - 3

5.3.4 Klassifizierung und Positionierung der Kamera

Mit den Kameraparametern und dem kalibrierten neuronalen Netzwerk kann nun der Positionierungsalgorithmus entwickelt werden.

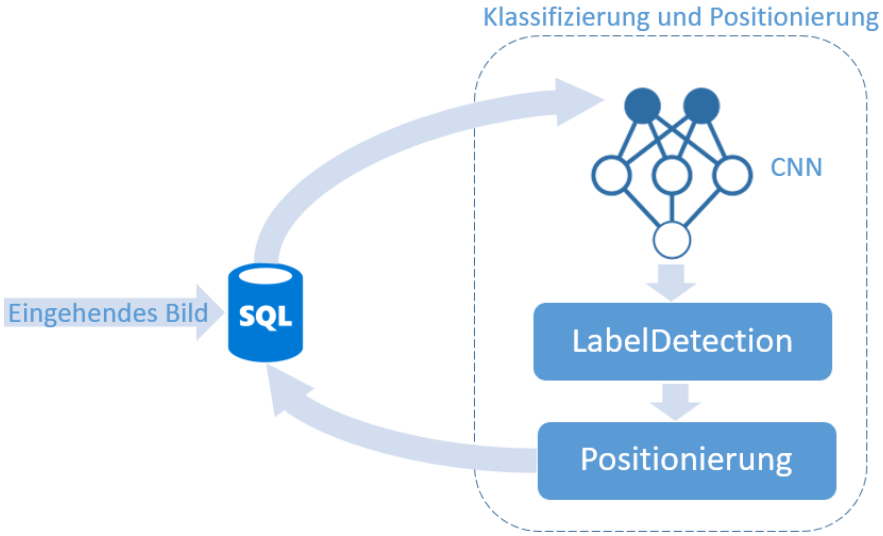


Abbildung 5.12 Routine zur Klassifizierung und Positionierung von eingehenden Bilddaten (Eigene Grafik)

Nachdem der Datenbank ein durch ein Endgerät übermitteltes Bild hinzugefügt wird, wird die Klassifizierungsroutine ausgelöst. Hierfür wird das neu eingegangene Bild mithilfe des NN klassifiziert. Wenn dabei Referenzpunkte mit einer Confidence über 99% erkannt werden, wird der „eventlabel“-Tabelle eine entsprechende Assoziation hinzugefügt. Das Zentrum der jeweiligen Bounding Box wird vereinfacht als Mittelpunkt des Referenzpunktes interpretiert und dessen Bildkoordinaten ebenfalls in einem Datenbankeintrag hinterlegt.

Wenn auf dem Bild mindestens zwei Referenzpunkten mit datenbankseitig gespeicherten 3D-Echtweltkoordinaten erkannt werden, kann durch die in Bestimmung der Translation hergeleitete Beziehung die Position der Kamera approximiert werden.

Die Matrix

$$\boldsymbol{K} = \begin{bmatrix} f * k_u & 0 & u_0 \\ 0 & f * k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

wird mit den in Kameraparameter ermittelten Parametern k_v, k_u, u_0, v_0 konstruiert. Die in Tabelle 9 dargestellten Werte beziehen sich dabei auf Pixelkoordinaten der für die Kalibrierung verwendeten Bildgröße 640 auf 480. Bei abweichenden Bilddimensionen müssen die Werte skaliert werden.

Die Rotationsmatrix $\boldsymbol{R}$ und der 3D-Szenenpunkt $\boldsymbol{p'_n}$ werden der Herleitung entsprechend mit den Feldwerten „Roll“, „Pitch“ und „Yaw“ der Tabelle „Event“, sowie mit „x“, „y“, „z“ des korrespondierenden, durch das CNN erkannten Marker der Tabelle „Label“ konstruiert. Der Bildpunkt $\boldsymbol{x}$ wird aus den Werten der Verbindungstabelle „eventlabel“ konstruiert.

Nach der Evaluation durch die Positionsberechnung können die ermittelten Kamerakoordinaten in den Feldern „tx“, „ty“ und „tz“ des entsprechenden „Events“ gespeichert werden.

5.3.5 Tracking von Referenzpunkten

Die Funktion „Tracking“ wurde implementiert, um die Position von Objekten ohne fixierten Standort bestimmen zu können. Damit die Koordinaten eines Objekts zu einem bestimmten Zeitpunkt errechnet werden können, muss dieser simultan von mindestens zwei unterschiedlichen Kameras aufgenommen worden sein.

Das Feature selektiert Objekte, die für den Positionierungsalgorithmus in Frage kommen, über Datenbankabfragen:

1. Aufnahme eines Objektes auf mindestens zwei Kameras zur gleichen Zeit
2. Standorte der aufnehmenden Kameras bekannt
3. Datenbankseitig kein Standort des Objekts hinterlegt

Sind die oben genannten Kriterien erfüllt, wird die Position des Objekts mithilfe der aus Approximation der räumlichen Position von erkannten Objekten eingeführten Beziehung ermittelt.

Der Kamerastandort $\boldsymbol{p_n}$ setzt sich aus den in Klassifizierung und Positionierung der Kamera ermittelten Kamerakoordinaten zusammen. Der Vektor

$$\begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix}$$

repräsentiert den abbildenden Punkt des zu trackenden Markers auf der Bildebene und wird mithilfe der Werten aus der Verbindungstabelle „rx“ und „ry“ konstruiert. Die Rotationsmatrix $\boldsymbol{R}$ wird wiederum aus den Feldern der „Event“-Tabelle erzeugt.

Die errechnete „Trackingroute“ wird nicht in der Datenbank hinterlegt, sondern auf Anfordern der Webanwendung initialisiert.

5.3.6 Web-Oberfläche

Um zu verdeutlichen, wie durch Cloud-Lösungen ein Mehrwert für Bauprojekte generiert werden kann, wurde eine Web-App für Projektmanagementaufgaben entworfen. Die drei implementierten Funktionen kommunizieren über die in der Cloud geschaffene Infrastruktur mit den IoT-Geräten und analysieren die von der Datenbank bereitgestellten Informationen.

„Manage“

Für die Verwaltung von IoT-Geräten wurde die Funktion „Manage“ implementiert. Die Weboberfläche bietet dem Nutzer ein grafisches Interface für alle registrierten Geräte.

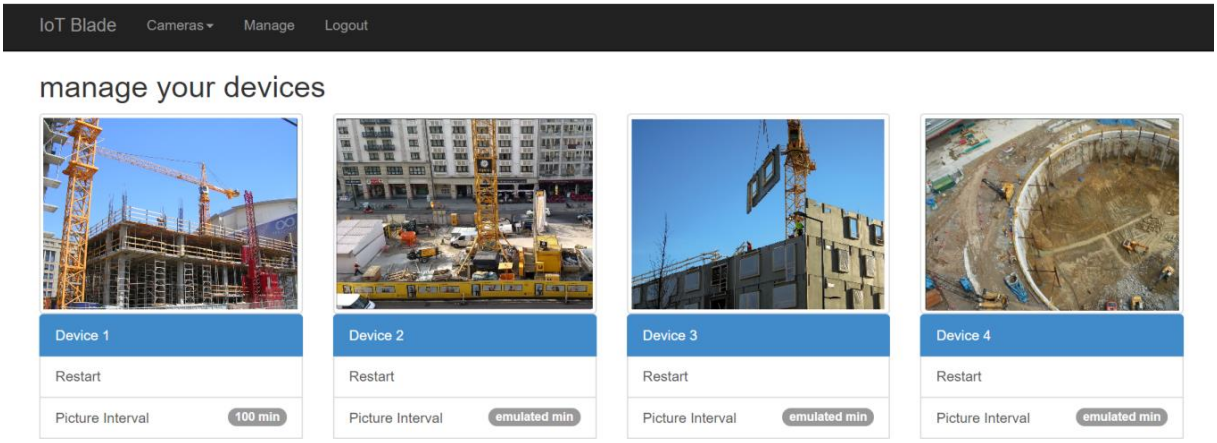


Abbildung 5.13 Funktion "Manage" der Weboberfläche (Eigene Grafik)

Wie in „Cloud to Device“ Nachrichten erläutert, fragen Devices periodisch den messaging Endpoint für eingehende Nachrichten ab. An diesen Endpoint sendet die Web-Applikation, getriggert durch User-Inputs, Funktionsaufrufe. Neben der Funktion

„Restart“ kann auch das Zeitintervall, in dem Bilder von der Kamera aufgenommen werden, über die Schaltfläche „Picture Interval“ angepasst werden.

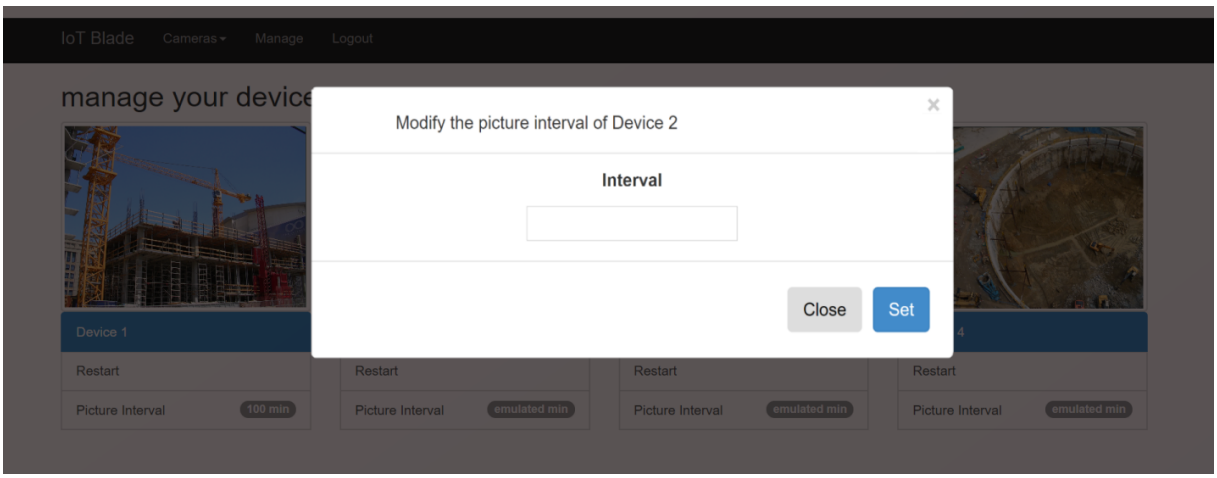


Abbildung 5.14 Dialogfenster zum Anpassen des Bildintervalls (Eigene Grafik)

Nachdem der Nutzer die Änderungen am Device Status im Dialogfenster bestätigt, wird eine IoT-Nachricht mit den relevanten Änderungen an den entsprechenden Endpoint geschickt. Sobald der IoT Hub den Empfang der Nachricht bestätigt, wird der korrespondierende Datenbankeintrag geändert. Somit ist sichergestellt, dass das Gerät und die Datenbank synchron sind.

„Overview und Report“

Mit dem Feature „Overview und Report“ lassen sich die Bilddaten visualisieren, nach bestimmten Kriterien filtern und als Report ausgeben.

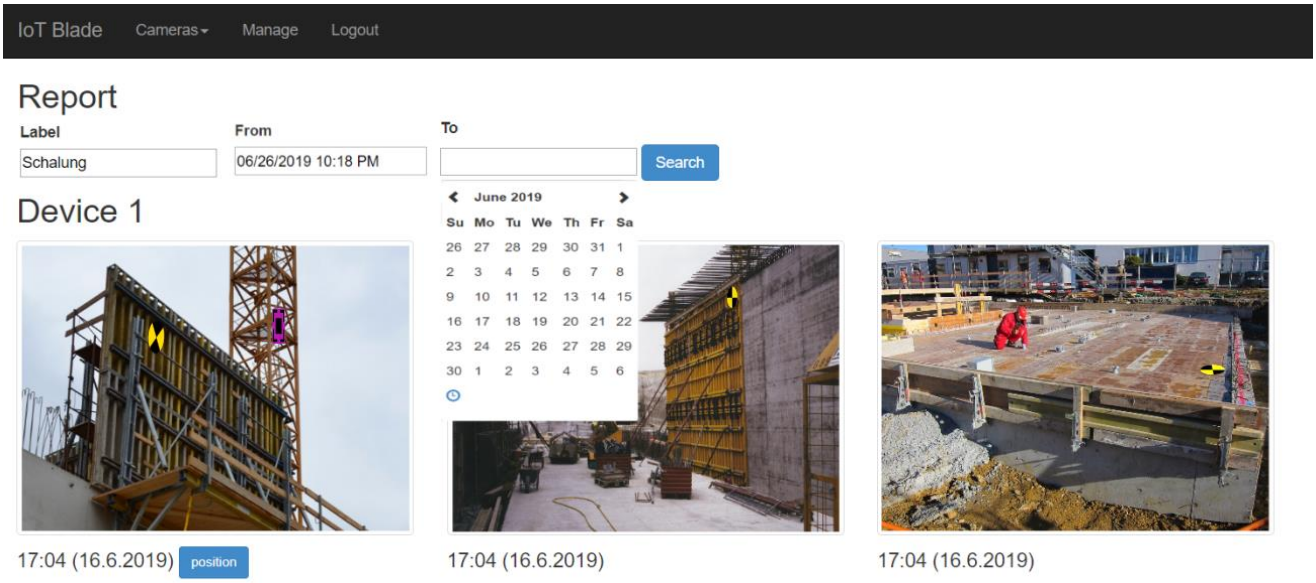


Abbildung 5.15 Funktion "Report" der Weboberfläche (Eigene Grafik)

Organisationsmöglichkeiten sind die Trennung nach Kamera, das Einschränken des Aufnahmezeitpunkts und das Beschränken auf Bilder, die mit einem bestimmten Label assoziiert wurden.

Zudem kann für alle Bilder, für die ein Kamerastandort bestimmt wurde, ein Plot erzeugt werden. Der Plot beinhaltet die Position der Kamera sowie die Standorte aller Referenzpunkte, die in dessen Blickfeld zu sehen sind. Die Punkte werden zusammen mit dem Kamerastandort und Blickfeld der Kamera auf den Projektplan gemappt und als Grafik gespeichert.

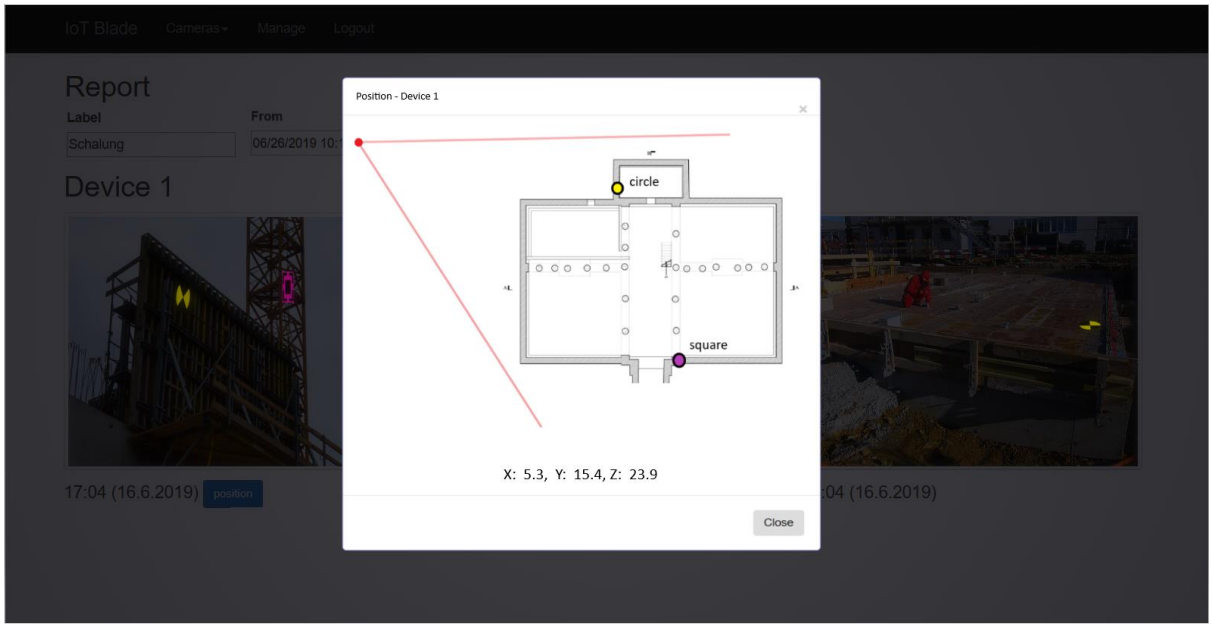


Abbildung 5.16 Durch die Weboberfläche erzeugter Plot. Abgebildet sind Kamera (rot) und Trackingmarker (violett, gelb) (Eigene Grafik)

„Tracking“

Mit dem Feature „Tracking“ wird dem Nutzer eine graphische Oberfläche zur Objektverfolgung geboten.

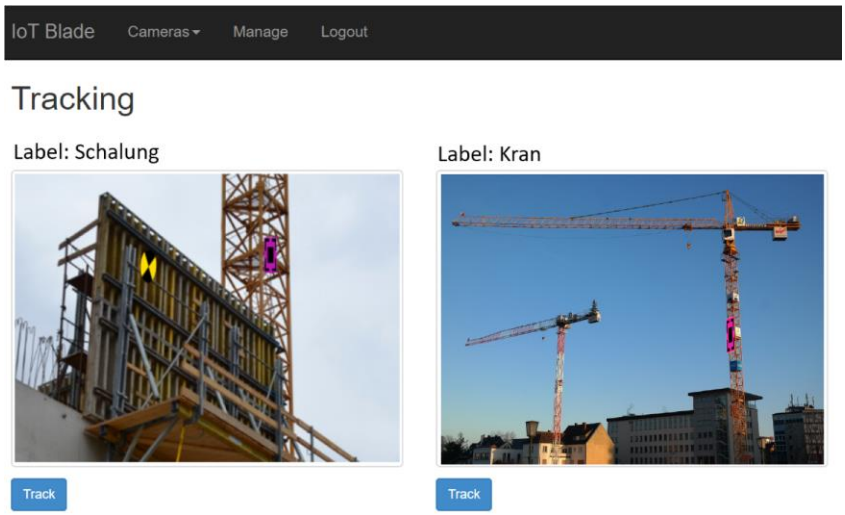


Abbildung 5.17 Funktion "Tracking" der Weboberfläche (Eigene Grafik)

Jeder Marker, der für den Tracking Algorithmus in Frage kommt, wird in Form eines Vorschaubildes angezeigt. Fordert der Nutzer die Trackingroute des jeweiligen Punktes über den Button „Track“ an, wird mit dem in Tracking von Referenzpunkten beschriebenen Algorithmus die chronologische Bewegung des Punktes bestimmt. Die an den Datumsstempel gekoppelten Positionen werden abermals auf den Projektplan gemappt und als Grafik visualisiert. Die Vektoren der Bewegung und zeitliche Abfolge der jeweiligen Punkte werden ebenfalls auf dem Plan abgebildet.

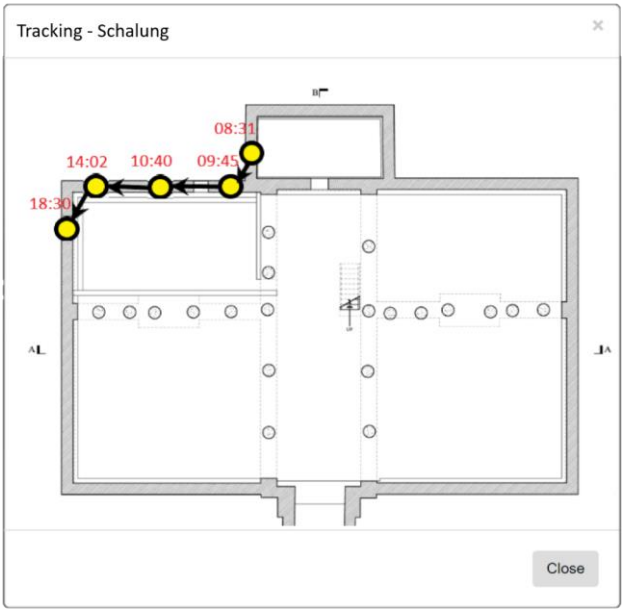


Abbildung 5.18 Durch die Weboberfläche erzeugter Plot. Chronologische Bewegung des Trackingmarkers "Schalung" (Eigene Grafik)

6 Validierung

Um die Funktionalität der erarbeiteten Features zu überprüfen, wurde ein exemplarischer Versuchsstand aufgebaut. Hierfür wurden drei identische Raspberry Pi Module an eingemessenen Standorten angebracht. Im Sichtfeld der Kameramodule wurden acht Punkte mithilfe eines Schnurgerüsts fixiert und ebenfalls eingemessen. Im Anschluss wurde die Referenz in mehreren Versuchsreihen aufgenommen. Die Draufsicht des Versuchsaufbau ist in Abbildung 6.1 dargestellt.

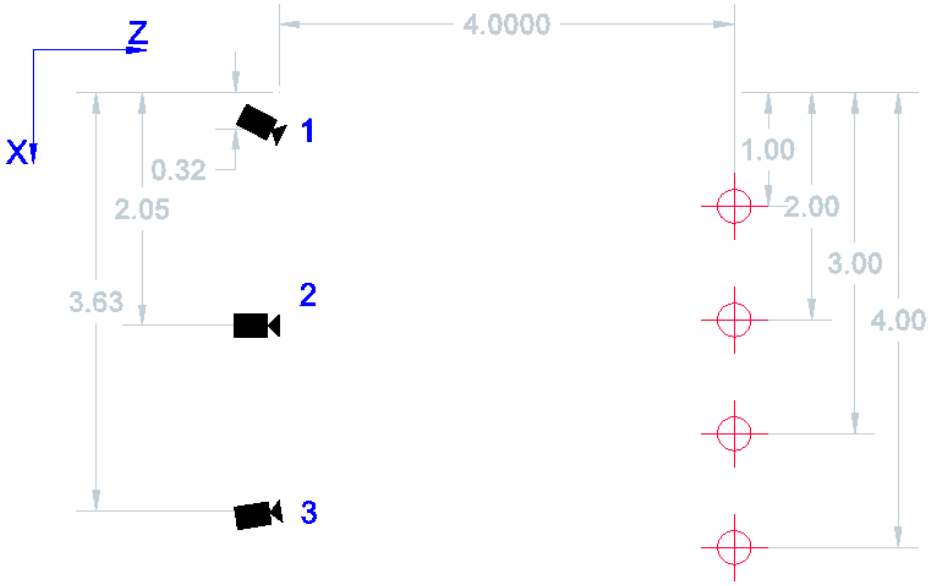


Abbildung 6.1 Abmessungen des Versuchsstandes in Draufsicht (Eigene Grafik)

Die frontale Ansicht der Referenzpunkte aus Kameraperspektive ist in Abbildung 6.2 dargestellt.

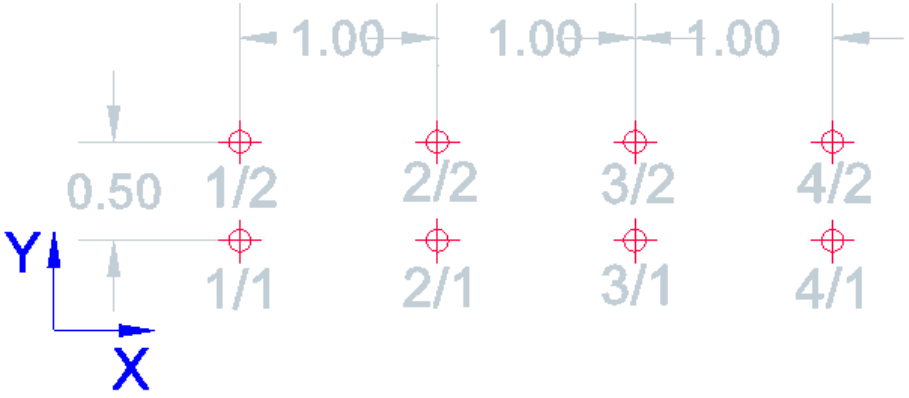


Abbildung 6.2 Abmessungen des Versuchsstandes aus Kameraperspektive (Eigene Grafik)

6.1.1 Positionierung der Kameras

In der ersten Versuchsreihe wurden für jeden Kamerastandort die Punktepaare 1/1 – 4/1, 2/1 – 3/1, 1/2 – 4/2, sowie 2/2 – 3/2 nacheinander aufgenommen. Anhand der Referenzpunkte kann durch den Positionierungsalgorithmus der Standort der jeweiligen Kamera berechnet werden. In der Tabelle 10 ist die räumliche, betragsmäßige Abweichung vom eingemessenen Soll für jeden der Kamerastandorte dargestellt. Die Positionierung wurde einmal mit denen in Kameraparameter beschriebenen Korrekturwerten, sowie einmal ohne Entzerren der Bilder durchgeführt.

Tabelle 10 Räumliche Abweichung der vom Positionierungsalgorithmus berechneten Werte in Metern (Eigene Darstellung)

Standort	1		2		3		
Referenz	verzerrt	entzerrt	verzerrt	entzerrt	verzerrt	entzerrt	Mittel
1/1 – 4/1	0,11	0,09	0,07	0,09	0,36	0,34	0,18
2/1 – 3/1	0,13	0,01	0,08	0,12	0,38	0,29	0,18
1/2 – 4/2	0,11	0,20	0,11	0,20	0,33	0,32	0,21
2/2 – 3/2	0,11	0,05	0,11	0,16	0,29	0,31	0,17
Mittel	0,11		0,10		0,34		0,18
		0,10		0,14		0,32	0,19

Für die Standorte eins und zwei lassen sich relativ akkurate Werte bezüglich der Kameraposition ermitteln. Standort drei fällt dabei deutlich ab.

Erstaunlicherweise resultierten die Messungen der beiden inneren Punktepaare (2/1 – 3/1, 2/2 – 3/2) nicht in schlechteren Ergebnissen im Vergleich zu den beiden äußeren Messungen (1/1 – 4/1, 1/2 – 4/2), wie sich auf Grund der trigonometrischen Abhängigkeiten vermuten lassen würde. Möglicherweise halten sich hier die Abweichungen durch am Bildrand vermehrt auftretende Verzerrungen und die erhöhte Sensibilität des Systems bei nah aneinander gelegenen Referenzpunkten die Waage.

Die entzerrten Bildpunkte hatten durch die geringen Verzerrungskoeffizienten keinen großen Einfluss auf das Resultat und konnten daher nur etwa gleichwertige Ergebnisse im Vergleich zu den ursprünglichen Bilddaten liefern. Es ist bei diesem funktional orientierten Versuchsaufbau jedoch schwierig, eine genaue Aussage zu Ursache und Wirkung von Fehlerquellen zu treffen, bzw. die Resultate als besonders signifikant anzusehen.

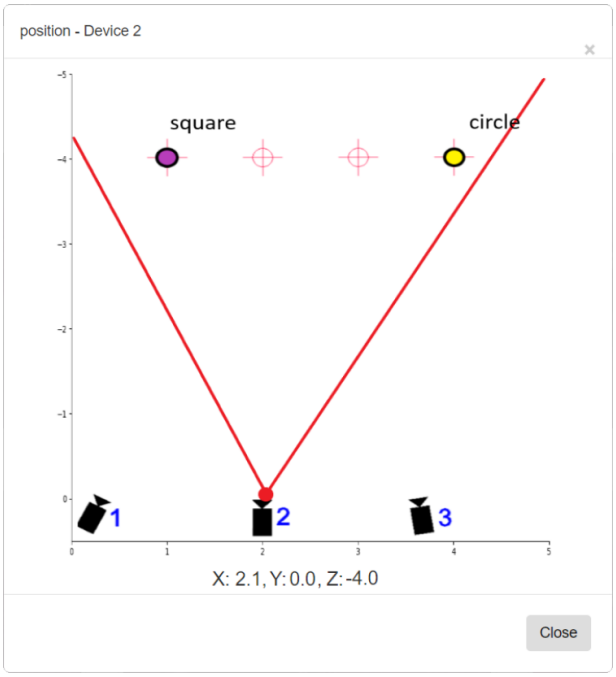


Abbildung 6.3 Von der Webapplikation erzeugter Plot für Kamerastandort 2 (Eigene Grafik)

Die deutlich fehlerbehaftete Positionierung des dritten Kamerastandortes konnte bereits während der Messung vermutet werden, da auch nach mehrmaliger Kalibrierung an dieser Position kein stabiler Zustand des Magnetometers hergestellt werden konnte. Dies kann aus verschiedenen Ursachen hervorgehen: inneren Einflüssen, wie einem Defekt des IMU, äußeren Einflüssen, wie lokalen Schwankungen des Magnetfeldes, oder ungenau durchgeführter Kalibrierung.

Abbildung 6.4 und Abbildung 6.5 zeigen den zeitlichen Verlauf der räumlichen Abweichung von der Referenz zur errechneten Position relativ zum Initialisierungszeitpunkt, sowie die Abweichung der zum jeweiligen Zeitpunkt gemessenen Gier,- Nick und Rollwinkel über eine Zeitspanne von 40 Sekunden zweier Messungen.

Die Meßwerte verdeutlichen, dass die Qualität der Positionierung maßgeblich mit der Stabilität der Sensordaten korreliert. Speziell im Falle des Magnetometers konnte diese Stabilität, abhängig von Standort und verwendetem IMU, nicht immer gewährleistet werden.

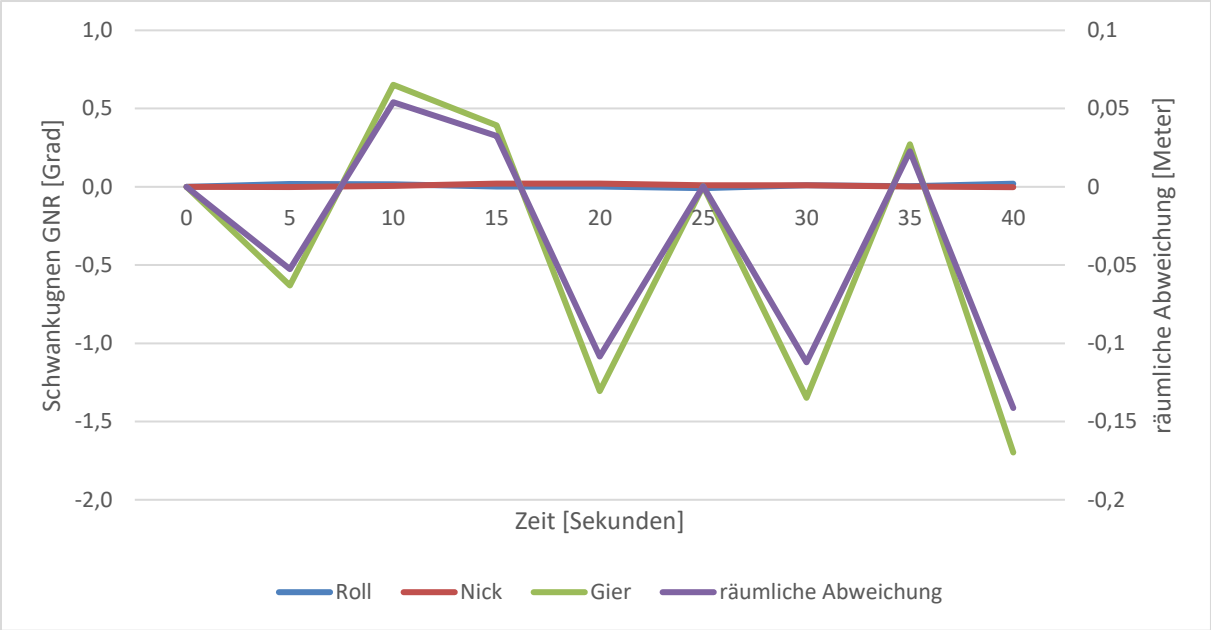


Abbildung 6.4 Räumliche Abweichung von "verrauschter" Messung über einen Zeitraum von 40 Sekunden. (Eigene Grafik)

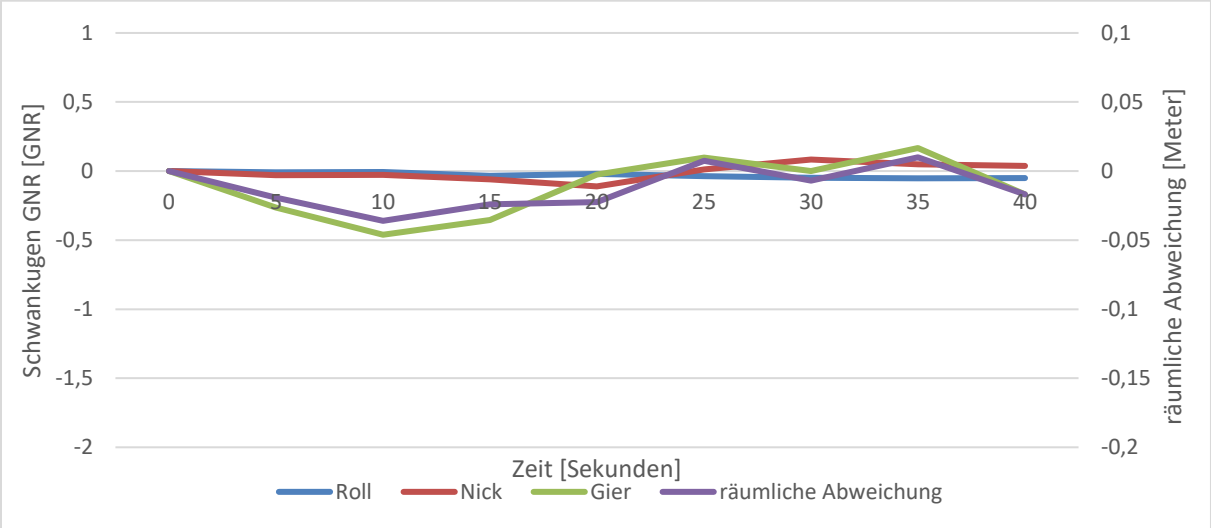


Abbildung 6.5 Räumliche Abweichung von "stabiler" Messung über einen Zeitraum von 40 Sekunden. (Eigene Grafik)

6.1.2 Tracking

Für die zweite Versuchsreihe wurden die Positionen der acht Begegnungspunkte des Schnurgerüsts mithilfe des Trackingalgorithmus bestimmt. Hierfür wurden alle Punkte von zwei bzw. drei Kameras aufgenommen und daraus, analog zu Positionierung der Kameras, die Validierung sowohl im entzerrten wie auch verzerrten Zustand ermittelt. Die Abweichung zur Referenz ist in der Tabelle 11 festgehalten.

Tabelle 11 Räumliche Abweichung der vom Trackingalgorithmus berechneten Werte in Metern (Eigene Darstellung)

Kamerastandorte	Label	1/1	2/1	3/1	4/1	1/2	2/2	3/2	4/2	Mittel
1,2	verzerrt	0,11	0,12	0,13	0,22	0,12	0,11	0,15	0,21	0,15
	entzerrt	0,11	0,12	0,15	0,24	0,11	0,12	0,17	0,24	0,16
1,2,3	verzerrt	0,52	0,45	0,44	0,51	0,47	0,42	0,45	0,51	0,47
	entzerrt	0,51	0,45	0,44	0,51	0,46	0,43	0,44	0,52	0,47
Mittel		0,31	0,29	0,29	0,37	0,29	0,27	0,30	0,37	0,31

Im Wesentlichen lassen sich die Beobachtungen, die im Rahmen der Positionierung getroffen wurden, auf das Tracking übertragen. Die entzerrten Bilder führen auch hier im Rahmen des Versuchsaufbaus zu keiner genaueren Positionierung.

Durch das Hinzuziehen einer dritten Kamera sollte gezeigt werden, wie sich die Ergebnisse durch einen zusätzlichen Aufnahmestandpunkt verbessern lassen. Jedoch hatte die dritte Kamera, induziert durch die vermutlich verfälschten IMU-Daten, einen deutlich negativen Einfluss auf die Ergebnisse. Positiv hingegen verlief die Klassifizierung der Trackinglabels durch das NN. Obwohl die Kameras sehr sensibel auf die wechselnd starken und schwachen Lichtverhältnisse reagierten, wurden die Referenzpunkte stets ausreichend gut approximiert, wie man auf Abbildung 6.6 erkennen kann.

Abbildung 6.6 Trackingmarker wird hinter dem Schnurgerüst vorbeibewegt (rechts). Resultierende Grafik der Webapplikation (links) (Eigene Grafik)

7 Bewertung der Ergebnisse und Ausblick

Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, der Projektbeteiligte von Bauvorhaben durch eine ferngesteuerte Bauüberwachung unterstützen soll. Dabei wurde das Vorhaben in den Kosmos des IoT eingeordnet und bestehende Cloud-Infrastruktur von Microsoft Azure genutzt, um eine zuverlässige und sichere Übertragung von Daten zu gewährleisten.

Den Paradigmen des IoT folgend konnte die Implementierung des Prototyps in drei Teilbereiche unterteilt werden: Device, Network und Application.

Die Endgeräte werden auf der Baustelle verteilt und zeichnen Bild- und Sensordaten auf. Auf der Ebene Network werden die erzeugten Daten mit der vom Provider bereitgestellten IoT-Integration Middleware übertragen und in einer Datenbank organisiert. Schließlich folgt die Ebene Application, die eine für den Nutzer zugängliche Webanwendung darstellt.

Die Webanwendung unterstützt die Bereitstellung von Bilddaten und verfügt über die Funktionalitäten Positionierung und Tracking. Der gewählte Ansatz sieht dabei vor, dass installierte Kameras Bilder aufnehmen und diese mit Informationen zu ihrer räumlichen Orientierung während der Aufnahme, abgeleitet durch die Sensordaten eines IMU, an die Datenbank übertragen. Auf Basis dieser gesammelten Daten werden Positionierungs- und Trackingalgorithmen anwendungsseitig durchgeführt und die Ergebnisse auf einem Projektplan verzeichnet.

Das Feature Positionierung verortet dabei die bildgebende Kamera anhand von sich in ihrem Blickfeld befindenden Referenzpunkten und den Daten des IMU.

Mit Tracking können Objekte mithilfe von Markern verfolgt werden, um beispielsweise die räumliche Veränderung von Bauteilen und Fertigungsmaschinen zu erfassen. Zur Identifizierung und Ermittlung der Position der Marker auf den Kamerabildern wurde ein neuronales Netzwerk eingesetzt.

Außerdem wurde für den Prototyp eine grafische Oberfläche für die Kameraverwaltung implementiert.

Mithilfe eines Versuchsstandes konnten die angestrebten Ziele validiert werden. Sowohl die Positionierungs- als auch Trackingalgorithmen erreichten bei richtigem Versuchsaufbau Toleranzen von ca. 2,5 – 5 Zentimetern pro Meter Abstand zwischen Kamera und Referenz.

Das neuronale Netzwerk lieferte während des Versuchs zufriedenstellende Ergebnisse. Die Trackingmarker konnten bei allen analysierten Versuchsreihen zuverlässig erkannt werden.

Es konnte somit demonstriert werden, wie mithilfe von Orientierung und Objekterkennung Tracking- und Positionierungsalgorithmen entwickelt werden können, die einen Mehrwert für Bauprojekte generieren. Dabei wird auf die für Dokumentationszwecke ohnehin benötigten Bilddaten zurückgegriffen und lediglich ein IMU dem Bildgebungssystem hinzugefügt. Zusätzliche Hardware, wie signalemittierende Tracking-Devices und korrespondierende Empfänger, wird nicht benötigt.

Während der Versuchsdurchführung und dessen Auswertung offenbarten sich jedoch auch Schwächen des Systems:

Da nur das Magnetometer des IMU in der Lage ist, eine Referenz zum magnetischen Norden zu liefern, wird folglich der für die Lagebestimmung benötigte Gierwinkel hauptsächlich von diesem beeinflusst. Im Gegensatz zum Gyroskop und zum Accelerometer konnten durch das Magnetometer während des Versuchsaufbaus nicht immer stabile und vor allem mit den anderen IMU-Einheiten konsistente Messwerte geliefert werden. Schwankungen von mehreren Grad machten einige Versuchsreihen unbrauchbar. Die beobachteten Schwankungen traten dabei sporadisch, ohne erkennbaren Zusammenhang zu äußeren Einflüssen auf und lassen sich vermutlich auf Inhomogenitäten im Magnetfeld zurückführen. Die beobachteten Effekte sind dabei charakteristisch für Sensorik mit dem gewählten Wirkungsprinzip.

Nach einigen Iterationen konnten die Abweichungen zwischen den einzelnen IMUs durch mehrmalige Kalibrierung der Sensoren zwar weitestgehend eliminiert werden, jedoch bleibt die Frage bestehen, ob dies eine geeignete Messmethode auf Baustellen darstellt. Denn Baustellen unterliegen oft starken, wechselnden, elektromagnetischen und ferromagnetischen Einflüssen. Darüber hinaus fordert der Kalibrierungsvorgang an sich schon ein hohes Maß an Aufmerksamkeit.

Des Weiteren ist auch die Identifizierung der Marker mithilfe des neuronalen Netzwerkes durchaus kritisch zu sehen. Bei dem durchgeführten Versuchsaufbau konnten mit

dieser Methode zwar gute Ergebnisse erzielt werden, jedoch befanden sich die Trackinglabels verhältnismäßig nah an der Kamera und wurden zudem frontal aufgenommen. Letztendlich resultiert die maximale Distanz zwischen Kamera und Trackinglabel zwar aus der gewählten Netzwerkarchitektur, der Größe der Labels und den Kameraparametern, allerdings stellten sich mit den im Versuchsaufbau verwendeten Labels der Größe 20 cm x 20 cm eine Wahrnehmungsgrenze von ca. 10 Metern ein. Dies entspricht gerade einmal der Höhe eines Einfamilienhauses.

Zudem ist der Umgang mit neuronalen Netzwerken als eher unflexibel und ressourcenintensiv anzusehen. Eine effiziente Klassifizierung der Bilder, sowie das Training des Netzwerkes kann nur mit entsprechender Rechenleistung durchgeführt werden. Für jeden Marker, der dem Netzwerk hinzugefügt werden soll, muss zusätzlich Aufwand hinsichtlich der Generierung von Trainingsdaten und dem eigentlichen Training betrieben werden.

Trotz dieser Schwächen stellt das erarbeitete Konzept eine wertvolle Methode zur Ergänzung von bauüberwachenden Prozessen dar. Es bestehen noch Potentiale für Weiterentwicklung: Für zukünftige Studien wäre es interessant zu untersuchen, wie sich die erarbeiteten Technologien in bereits bestehende Systeme zur Baudokumentation beziehungsweise Bauüberwachung integrieren lassen. Hierfür könnten Konzepte entwickelt werden, die es ermöglichen aus den Trackingdaten Ereignisse oder Mängel zu generieren, um diese, gekoppelt an geometrische Modelle, an die Projektbeteiligten zu kommunizieren. Des Weiteren könnten die Positions- und Orientierungsdaten der Kamera genutzt werden, um aus einer Überlappung und einem Abgleich der Bilder mit den CAD-Plänen des Bauprojekts, ergänzend zu den aus den Trackingdaten gewonnenen Informationen, eine optimierte automatisierte Baufortschrittskontrolle ermöglichen.

Literaturverzeichnis

2015 3rd International Conference on Future Internet of Things and Cloud (2015 - 2015). 2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud). Rome, Italy, 24.08.2015 - 26.08.2015: IEEE.

Atzori, Luigi; Iera, Antonio; Morabito, Giacomo (2010): The Internet of Things: A survey. In: *Computer Networks* 54 (15), S. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010.

AWS IoT-SDKs - AWS IoT (2019). Online verfügbar unter https://docs.aws.amazon.com/de_de/iot/latest/developerguide/iot-sdks.html, zuletzt aktualisiert am 27.06.2019, zuletzt geprüft am 27.06.2019.

Azure Event Hubs: Was ist Azure Event Hubs? – ein Big Data-Erfassungsdienst. Online verfügbar unter https://docs.microsoft.com/de-de/azure/event-hubs/event-hubs-about, zuletzt geprüft am 28.06.2019.

Azure Functions – Tutorials: Dokumentation für Azure Functions – Tutorials. Online verfügbar unter https://docs.microsoft.com/de-de/azure/azure-functions/, zuletzt geprüft am 28.06.2019.

azure servicefabric. Online verfügbar unter https://github.com/azure-cat-emea/servicefabricjsonsqldb, zuletzt geprüft am 27.06.2019.

Azure Stream Analytics: IoT-Datenströme in Echtzeit mit Azure Stream Analytics. Online verfügbar unter https://docs.microsoft.com/de-de/azure/stream-analytics/stream-analytics-get-started-with-azure-stream-analytics-to-process-data-from-iot-devices, zuletzt geprüft am 28.06.2019.

Azure/azure-iot-sdks. Online verfügbar unter https://github.com/Azure/azure-iot-sdks, zuletzt geprüft am 27.06.2019.

Baudokumentation smart & easy - Bautagebuch - Software & App. Online verfügbar unter https://www.weka-bausoftware.de/baudokumentation, zuletzt geprüft am 27.06.2019.

Belgrade SqlClient (2019). Online verfügbar unter http://jocapc.github.io/Belgrade-SqlClient/, zuletzt aktualisiert am 06.01.2019, zuletzt geprüft am 27.06.2019.

Bier, Agnieszka; Luchowski, Leszek (2009): Error Analysis of Stereo Calibration and Reconstruction. In: André Gagalowicz und Wilfried Philips (Hg.): Computer Vision/Computer Graphics CollaborationTechniques, Bd. 5496. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), S. 230–241.

VDI 2552: Building Information Modeling (BIM).

buildingSMART e.V.: buildingSMART Germany. Hg. v. buildingSMART e.V. Online verfügbar unter www.buildingsmart.de.

Camera Module. Online verfügbar unter https://www.raspberrypi.org/documentation/hardware/camera/, zuletzt geprüft am 27.06.2019.

cameras — openMVG library (2017). Online verfügbar unter https://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/, zuletzt aktualisiert am 08.10.2017, zuletzt geprüft am 27.06.2019.

Cireşan, Dan C.; Meier, Ueli; Masci, Jonathan; Gambardella, Luca M.; Schmidhuber, Jürgen (2011): Flexible, high performance convolutional neural networks for image classification: AAAI Press.

Cloud IoT Core | Cloud IoT Core | Google Cloud (2019). Online verfügbar unter https://cloud.google.com/iot-core/, zuletzt aktualisiert am 22.05.2019, zuletzt geprüft am 27.06.2019.

CloudMQTT - Add-ons - Heroku Elements. Online verfügbar unter https://elements.heroku.com/addons/cloudmqtt, zuletzt geprüft am 27.06.2019.

COCO - Common Objects in Context (2018). Online verfügbar unter http://cocodataset.org/#home, zuletzt aktualisiert am 06.11.2018, zuletzt geprüft am 28.06.2019.

Convolutional Neural Networks for Visual Recognition (2019). Online verfügbar unter http://cs231n.github.io/convolutional-networks/#pool, zuletzt aktualisiert am 21.05.2019, zuletzt geprüft am 27.06.2019.

Derek Hoiem (2011): Projective Geometry and Camera Models. Computer Vision. University of Illinois, 20.01.2011. Online verfügbar unter https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2002%20-%20Projective%20Geometry%20and%20Camera%20Models%20-%20Vision_Spring2011.pdf.

DOC: Architecture overview. Online verfügbar unter https://docs.kaaiot.io/DOC/docs/current/Architecture-overview/, zuletzt geprüft am 27.06.2019.

Duane C. Brown (1966): Decentering Distorsion of Lenses. the prism effect ecountered in metric cameras can be overcome through analytic calibration. Florida: D. Brown Associates, Inc.

Elmasri, Ramez; Navathe, Sham (2011): Fundamentals of database systems. 6. ed. Boston: Addison-Wesley.

Farahzadia, Amirhossein; Shams, Pooyan; Rezazadeh, Javad; Farahbakhsh, Reza (2018): Middleware Technologies for Cloud of Things - a survey. In: *Digital Communications and Networks* 4 (3), S. 176–188. DOI: 10.1016/j.dcan.2017.04.005.

Flask. Online verfügbar unter http://flask.pocoo.org/, zuletzt geprüft am 27.06.2019.

GNR-Konvention (2018). Online verfügbar unter https://upload.wikimedia.org/wikipedia/commons/6/67/Plane.svg, zuletzt aktualisiert am 31.10.2018, zuletzt geprüft am 28.06.2019.

Grasböck, Robert: djtulan - Hackerspace, Retrocomputing - Arduino. Online verfügbar unter http://djtulan.com/tag/Arduino/, zuletzt geprüft am 27.06.2019.

Gray, Jim (2006): To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem (MSR-TR-2006-45). Online verfügbar unter https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/.

Guth, Jasmin; Breitenbücher, Uwe; Falkenthal, Michael; Fremantle, Paul; Kopp, Oliver; Leymann, Frank; Reinfurt, Lukas (2018): A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences. In: Beniamino Di Martino, Kuan-Ching Li, Laurence T. Yang und Antonio Esposito (Hg.): Internet of Everything, Bd. 54. Singapore: Springer Singapore (Internet of Things), S. 81–101.

How Linux Keeps Track of Time (2001). Online verfügbar unter https://www.tldp.org/HOWTO/Clock-2.html, zuletzt aktualisiert am 18.07.2001, zuletzt geprüft am 27.06.2019.

Huang, Jonathan; Rathod, Vivek; Sun, Chen; Zhu, Menglong; Korattikara, Anoop; Fathi, Alireza et al. (2016): Speed/accuracy trade-offs for modern convolutional object detectors. Online verfügbar unter http://arxiv.org/pdf/1611.10012v3.

Hui, Jonathan (2019): mAP (mean Average Precision) for Object Detection - Jonathan Hui - Medium. Online verfügbar unter https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, zuletzt geprüft am 28.06.2019.

IBM Watson IoT Platform - Überblick - Deutschland. Online verfügbar unter https://www.ibm.com/de-de/marketplace/internet-of-things-cloud, zuletzt geprüft am 27.06.2019.

ICITST; International Conference for Internet Technology and Secured Transactions; Chaos-Information Hiding and Security; C-IHS; Cloud Applications and Security; CAS; International Workshop on RFID Security and Cryptography; RISC (2015): 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST). Unter Mitarbeit von George Ghinea, Mohamed Jamal Zemerly und Charles A. Shoniregun. 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST). London, United Kingdom, 12/14/2015 - 12/16/2015. Piscataway, NJ: IEEE.

Image Convolution (2017). Online verfügbar unter http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html, zuletzt aktualisiert am 22.03.2017, zuletzt geprüft am 28.06.2019.

imageio. Online verfügbar unter https://pypi.org/project/imageio/, zuletzt geprüft am 28.06.2019.

IoT Hub-Endpunkte: Informationen zu Azure IoT Hub-Endpunkten. Online verfügbar unter https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-devguide-endpoints, zuletzt geprüft am 28.06.2019.

IoT Hub-Nachrichtenrouting: Grundlegendes zum Azure IoT Hub-Nachrichtenrouting. Online verfügbar unter https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-devguide-messages-d2c, zuletzt geprüft am 28.06.2019.

IoT: number of connected devices worldwide 2012-2025 | Statista. Online verfügbar unter https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, zuletzt geprüft am 27.06.2019.

Jahr, Katrin; Braun, Alexander; Borrmann, André (2018): Formwork detection in UAV pictures of construction sites. In: Jan Karlshoj und Raimar Scherer (Hg.): EWork and EBusiness in Architecture, Engineering and Construction. Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2018), September 12-14, 2018, Copenhagen, Denmark. Milton: Chapman and Hall/CRC, S. 265–271.

Jinja2 (2018). Online verfügbar unter http://jinja.pocoo.org/docs/2.10/, zuletzt aktualisiert am 25.05.2018, zuletzt geprüft am 28.06.2019.

Jonathan Huang; Vivek Rathod; Derek Chow; Chen Sun; Menglong Zhu; Matthew Tang et al.: tensorflow Object Detection API. Online verfügbar unter https://github.com/tensorflow/models/tree/master/research/object_detection, zuletzt geprüft am 28.06.2019.

jQuery: jQuery. Online verfügbar unter https://jquery.com/, zuletzt geprüft am 28.06.2019.

Kevin Ashton (2016): Beginning the Internet of Things - Kevin Ashton - Medium. Online verfügbar unter https://medium.com/@kevin_ashton/beginning-the-internet-of-things-6d5ab6178801, zuletzt geprüft am 27.06.2019.

Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017): ImageNet classification with deep convolutional neural networks. In: *Commun. ACM* 60 (6), S. 84–90. DOI: 10.1145/3065386.

Kusserow, Ulrich von (2013): Magnetischer Kosmos. To B or not to B. Berlin: Springer Spektrum. Online verfügbar unter http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=676019.

Lens Distortion: What Every Photographer Should Know (2019). Online verfügbar unter https://clickitupanotch.com/lens-distortion/, zuletzt aktualisiert am 27.06.2019, zuletzt geprüft am 27.06.2019.

Liu, Tianyi; Fang, Shuangsang; Zhao, Yuehui; Wang, Peng; Zhang, Jun (2015): Implementation of Training Convolutional Neural Networks. Online verfügbar unter http://arxiv.org/pdf/1506.01195v2.

Maksimovic, Mirjana; Vujovic, Vladimir; Davidović, Nikola; Milosevic, Vladimir; Perisic, Branko (2014): Raspberry Pi as Internet of Things hardware: Performances and Constraints.

Mängelmanagement: Software für das Bauwesen | think project! Online verfügbar unter https://www.thinkproject.com/de/loesungen/maengelmanagement-software/, zuletzt geprüft am 27.06.2019.

Matplotlib: Python plotting (2019). Online verfügbar unter https://matplotlib.org/, zuletzt aktualisiert am 19.05.2019, zuletzt geprüft am 28.06.2019.

Matterport, Inc.: Interaktives 3D für Web und Virtuelle Realität. Online verfügbar unter https://matterport.com/de/, zuletzt geprüft am 27.06.2019.

McKinsey Global Institute (2017): reinventing construction. a route to higher productivity. Hg. v. McKinsey&Company. Online verfügbar unter https://www.mckinsey.com/~/media/McKinsey/Industries/Capital%20Projects%20and%20Infrastructure/Our%20Insights/Reinventing%20construction%20through%20a%20productivity%20revolution/MGI-Reinventing-Construction-Executive-summary.ashx.

MoorePenrose. Online verfügbar unter http://www.math.uni-bonn.de/people/woermann/MoorePenrose.pdf, zuletzt geprüft am 27.06.2019.

Multi-Class Neural Networks: Softmax | Machine Learning Crash Course | Google Developers (2019). Online verfügbar unter https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax, zuletzt aktualisiert am 05.03.2019, zuletzt geprüft am 27.06.2019.

Muntjir, Mohd; Rahul, Mohd; Alhumiany, Hesham (2017): An Analysis of Internet of Things(IoT): Novel Architectures, Modern Applications,Security Aspects and Future Scope with Latest Case Studies. In: *Building Services Engineering Research and Technology* 6.

Ngu, Anne H. H.; Gutierrez, Mario; Metsis, Vangelis; Nepal, Surya; Sheng, Michael Z. (2016): IoT Middleware: A Survey on Issues and Enabling technologies. In: *IEEE Internet Things J.*, S. 1. DOI: 10.1109/JIOT.2016.2615180.

OpenCV. Online verfügbar unter https://opencv.org/, zuletzt geprüft am 28.06.2019.

Otto, Mark; Thornton, Jacob; contributors, and Bootstrap (2019): Bootstrap. Online verfügbar unter https://getbootstrap.com/, zuletzt aktualisiert am 05.06.2019, zuletzt geprüft am 28.06.2019.

Peter Egor Bondarev de With (2017): Advanced Topics Multimedia Video. 3D Geometry, Camera Model & Projection Matrix. TU Eindhoven, 2017.

Piexif (2018). Online verfügbar unter https://piexif.readthedocs.io/en/latest/, zuletzt aktualisiert am 15.11.2018, zuletzt geprüft am 27.06.2019.

Pillow (2019). Online verfügbar unter https://pillow.readthedocs.io/en/stable/, zuletzt aktualisiert am 02.04.2019, zuletzt geprüft am 27.06.2019.

Preisrechner | Microsoft Azure. Online verfügbar unter https://azure.microsoft.com/de-de/pricing/calculator/, zuletzt geprüft am 27.06.2019.

PS-MPU-9250A-01-v1.1. Online verfügbar unter https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf, zuletzt geprüft am 27.06.2019.

PwC (2018): Baubranche aktuell. Wachstum 2020 - Digitalisierung und BIM. Hg. v. PwC. Online verfügbar unter https://www.pwc.de/de/industrielle-produktion/baubranche-aktuell-wachstum-2020-maerz-2018.pdf, zuletzt geprüft am 27.06.2018.

Ray, Partha Pratim (2016): A survey of IoT cloud platforms. In: *Future Computing and Informatics Journal* 1 (1-2), S. 35–46. DOI: 10.1016/j.fcij.2017.02.001.

Ren, Shaoqing; He, Kaiming; Girshick, Ross; Sun, Jian (2015): Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Online verfügbar unter http://arxiv.org/pdf/1506.01497v3.

Rill, Georg; Schaeffer, Thomas (2010): Grundlagen und Methodik der Mehrkörpersimulation. Mit Anwendungsbeispielen. 1. Aufl. Wiesbaden: Vieweg + Teubner (Studium).

roygara: Skalierbarkeits- und Leistungsziele für Azure Storage – Speicherkonten. Online verfügbar unter https://docs.microsoft.com/de-de/azure/storage/common/storage-scalability-targets, zuletzt geprüft am 28.06.2019.

RTIMULib2. Online verfügbar unter https://github.com/richardstechnotes/RTIMULib2, zuletzt geprüft am 27.06.2019.

RTIMULib2 Ellipsoid fit. Online verfügbar unter https://github.com/richardstechnotes/RTIMULib2/blob/master/RTEllipsoidFit/ellipsoid_fit_license.txt, zuletzt geprüft am 27.06.2019.

Shared Access Signatures: Verwenden von Shared Access Signatures (SAS) in Azure Storage. Online verfügbar unter https://docs.microsoft.com/de-de/azure/storage/common/storage-dotnet-shared-access-signature-part-1, zuletzt geprüft am 28.06.2019.

Shin, Hoo-Chang; Roth, Holger R.; Gao, Mingchen; Le Lu; Xu, Ziyue; Nogues, Isabella et al. (2016): Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. In: *IEEE transactions on medical imaging* 35 (5), S. 1285–1298. DOI: 10.1109/TMI.2016.2528162}.

spektrumverlag (2014): Deklination. Online verfügbar unter https://www.spektrum.de/lexikon/geographie/deklination/1566, zuletzt aktualisiert am 04.12.2014, zuletzt geprüft am 27.06.2019.

SQLAlchemy - The Database Toolkit (2019). Online verfügbar unter https://www.sqlalchemy.org/, zuletzt aktualisiert am 17.06.2019, zuletzt geprüft am 28.06.2019.

Stefania Trapani (2018): A Correction Method for Pedestrian Detection using a Convolutional Neural Network. Online verfügbar unter https://webthesis.biblio.polito.it/7545/1/tesi.pdf.

stevestein: Azure SQL-Datenbank-Dienstebenen – DTU-basiertes Kaufmodell. Online verfügbar unter https://docs.microsoft.com/de-de/azure/sql-database/sql-database-service-tiers-dtu, zuletzt geprüft am 27.06.2019.

Talat Ozyagcilar: Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors. Unter Mitarbeit von Freescale Semiconductor. Online verfügbar unter https://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf, zuletzt geprüft am 27.06.2019.

TensorFlow. Online verfügbar unter https://www.tensorflow.org/, zuletzt geprüft am 28.06.2019.

Tesic, J. (2005): Metadata Practices for Consumer Photos. In: *IEEE Multimedia* 12 (3), S. 86–92. DOI: 10.1109/MMUL.2005.50.

Tiburski, Ramão Tiago; Amaral, Leonardo Albernaz; Matos, Everton De; Hessel, Fabiano (2015): The importance of a standard securit y archit ecture for SOA-based iot middleware. In: *IEEE Commun. Mag.* 53 (12), S. 20–26. DOI: 10.1109/MCOM.2015.7355580.

TinyRTC Datasheet. Online verfügbar unter https://www.e-gizmo.net/oc/kits%20documents/TinyRTC%20I2C%20module/TinyRTC%20i2c%20module%20%20Techincal%20Manual%20rev1.pdf, zuletzt geprüft am 27.06.2019.

Titterton, David H.; Weston, John L. (1997): Strapdown inertial navigation technology. Stevenage, Herts.: Peregrinus (IEE radar, sonar, navigation and avionics series, 5).

Tränkler, Hans-Rolf; Reindl, Leonhard M. (2014): Sensortechnik. Handbuch für Praxis und Wissenschaft. 2., völlig neu bearb. Aufl. Berlin: Springer Vieweg (VDI-Buch). Online verfügbar unter http://dx.doi.org/10.1007/978-3-642-29942-1.

Vernetzt und transparent arbeiten mit BIM4You Office. Online verfügbar unter https://www.bib-gmbh.de/software/bim4you-office/, zuletzt geprüft am 27.06.2019.

watchdog. Online verfügbar unter https://pypi.org/project/watchdog/, zuletzt geprüft am 27.06.2019.

Werkzeug. Online verfügbar unter https://pypi.org/project/Werkzeug/, zuletzt geprüft am 28.06.2019.

What the heck is a DTU? - SQLPerformance.com (2017). Online verfügbar unter https://sqlperformance.com/2017/03/azure/what-the-heck-is-a-dtu, zuletzt geprüft am 27.06.2019.

WTForms (2018). Online verfügbar unter https://wtforms.readthedocs.io/en/stable/, zuletzt aktualisiert am 07.06.2018, zuletzt geprüft am 28.06.2019.

Xu, Li Da; He, Wu; Li, Shancang (2014): Internet of Things in Industries: A Survey. In: *IEEE Trans. Ind. Inf.* 10 (4), S. 2233–2243. DOI: 10.1109/TII.2014.2300753.

Yelizavet, Domínguez Pérez Dannika; Florentino, Orocio Méndez (2019): Internet of Things Platform (IoT) - Comparison of Layered Architectures. In: *IJCTT* 67 (1), S. 4–10. DOI: 10.14445/22312803/IJCTT-V67I1P102.

Zeiler, Matthew D.; Fergus, Rob (2013): Visualizing and Understanding Convolutional Networks. Online verfügbar unter http://arxiv.org/pdf/1311.2901v3.

Zimmermann, Josef (Hg.) (2010): Unternehmeringenieur in der Bauwirtschaft. Seminarband WS 2009/2010. Technische Universität München. München: TUM.

Anhang A

Auf den beigefügten Speichermedien befindet sich folgender Inhalt:

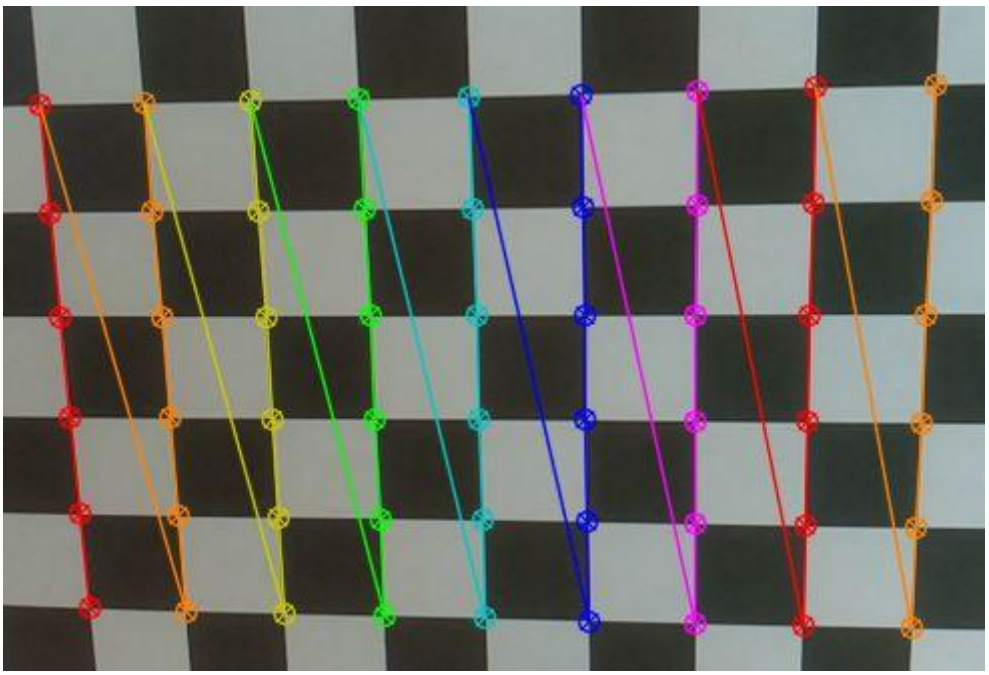
- Der schriftliche Teil der Arbeit als PDF
- Der Quellcode der Webapplikation
- Der Quellcode der Azure Services
- Der Quellcode der Raspberry Pi Routinen
- Die Daten, die zum Training des neuronalen Netzwerkes eingesetzt wurden
- Die Checkpoints des neuronalen Netzwerkes
- Die Bilder des Versuchsaufbaus
- Die CAD Files des Camera Casings
- Das Datenträgerabbild des Raspberry Pi

Anhang B

Bilder

B.1 Versuchsaufbau

B.2 Kamerakalibrierung mit OpenCV



Erklärung

Hiermit erkläre ich, dass ich die vorliegende Master-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 10. Juli 2019

Vorname Nachname

Michael Holland

[redacted]

[redacted]

[redacted]