Ruhr-Universität Bochum

Bachelorarbeit

Pose Estimation in Gebäuden anhand von Convolutional Neural Networks und simulierten 3D-Daten

Schriftliche Prüfungsarbeit
für die Bachelor-Prüfung des Studiengangs Angewandte Informatik an der Ruhr-Universität Bochum

vorgelegt von
Abdullah Sahin

am
Lehrstuhl für Informatik im Bauwesen
Prof. Dr.-Ing Markus König

Abgabedatum:	9. September 2019
Matrikelnummer:	108016202304
1. Prüfer:	Prof. Dr.-Ing. Markus König
2. Prüfer:	Patrick Herbers, M. Sc.

Abstract

.....

Erklärung

Ich erkläre, dass das Thema dieser Arbeit nicht identisch ist mit dem Thema einer von mir bereits für eine andere Prüfung eingereichte Arbeit.

Ich erkläre weiterhin, dass ich die Arbeit nicht bereits an einer anderen Hochschule zur Erlangung eines akademischen Grades eingereicht habe.

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen der Entlehnung kenntlich gemacht. Dies gilt sinngemäß auch für gelieferte Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Statement

I hereby declare that except where the specific reference is make to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Datum Unterschrift

Inhaltsverzeichnis

1 Einleitung

...

2 Stand der Forschung und Grundlagen

Das vorliegende Kapitel versucht einen Überblick des Forschungsstandes in den unterschiedlichen Aspekten der Arbeit zu verschaffen. Anschließend vermittelt das Kapitel notwendige Grundkenntnisse.

2.1 Einführung

// Evt. Einblick über indoor Lokalizierungstechnicken

Pose Estimation wird in dieser Arbeit als eine Methode der visuellen Lokalisierung (*Visual-Based Localization, kurz VBL*) betrachtet. VBL beschäftigt sich mit der Bestimmung der Pose (*Position + Orientierung*) eines visuellen Abfragematerials (*z.B. ein RGB-Bild*) in einer zuvor bekannten Szene [47]. Ein naheliegendes Themengebiet der Robotik ist die visuelle Ortswiedererkennung (*Visual Place Recognition, kurz VPR*) [36]. Die visuelle Ortswiedererkennung fokussiert sich auf das Feststellen eines bereits besuchten Ortes und definiert sich aus einer Mapping-, Datenverarbeitungs- und einem Orientierungsmodul. Allgemein lässt sich das Prozess eines VPRs folgend beschreiben. Eine interne Karte bekannter Orte wird durch das Mappingmodul verwaltet. Die Daten werden vom Datenverarbeitungsmodul vorbereitet und anschließend an das Orientierungsmodul übergeben. Daraufhin bestimmt das Orientierungsmodul die Pose und entscheidet mit der immer aktuell gehaltenen Karte, ob ein Ort bereits besucht wurde. Im Vergleich zur VPR versucht die visuelle Lokalisierung eine Pose zu bestimmen und benötigt daher neben den zwei Modulen kein Mappingmodul.

Die rein visuellen Methoden des VBLs unterteilen sich in indirekte und direkte Methoden [36]. Die indirekten Methoden behandeln das Lokalisierungsproblem als eine Bildersuche in einer Datenbank, ähnlich wie das *Content Based Image Retrieval* [33] Problem. Dabei wird das Abfragebild über eine Ähnlichkeitsfunktion mit den Vergleichsbildern aus der Datenbank abgeglichen [68, 5, 49]. Diese Art von Methoden benötigen eine sehr große Bildergalerie (*Datenbank*) und liefern Ergebnisse bei Fund eines korrespondierenden Bildes [36]. Hingen versuchen die direkten Methoden die Pose über eine Referenzumgebung zu bestimmen und benötigen meist daher keine große Bildergalerie [47]. Es gibt drei Arten der direkten Methoden: 1) Abgleichen von Features zu Punktwolken (*z.B. [35]*) 2) Pose Regression mit Tiefenbilder (*z.B. [53]*) 3) Pose Regression nur mit Bildern (*z.B. [28]*)

Die erste Art von Methoden versucht die Pose zu bestimmen, indem die 2D-3D Korrespondenz über das Abgleichen von Features des Abfragebildes gegen die Deskriptoren der 3D-Punkte hergestellt werden [24, 35, 56]. Diese Vorgehensweise hat Ähnlichkeiten zu den indirekten Methoden und benötigt statt einer Bildergalerie eine repräsentative 3D-Punktwolke der Szene [47]. Die zweite Art von Methoden bestimmt anhand von Tiefenbilder die Pose z.B. über Regression Forests [53], Randomize Ferns [16], Coarse-to-Fine Registierung [52] oder Neuronale Netze [39]. Diese Forschungsprojekte liefern

mit 3D-Bildern gewünschte Resultate. Die Ergebnisse sind abhängig von 3D-Kameras, diese sind jedoch nicht verbreitet.

Convolutional Neural Networks (*CNN*) werden erfolgreich im Bereich des Maschinelles Sehens, wie z.B. bei der Klassifizierung von Bildern [30, 54, 20] sowie bei der Objekterkennung [15, 50, 14] eingesetzt. Ein verbreiteter Ansatz beim Entwurf von CNNs ist das Modifizieren der vorhandenen Netzwerkarchitekturen, die z.B. für die Bildklassifizierung angesichts der Wettbewerbe von ImageNet Large Scale Visual Recognition Challenge (*ILSVRC*) [51] konstruiert wurden. Dieser Ansatz konnte beispielsweise erfolgreich in der Objekterkennung [14], Objektsegmentierung [29, 38], semantische Segmentierung [44, 19] und Tiefenbestimmung [34] verfolgt werden. Seit Kurzem werden CNNs auch in den Anwendungsgebieten der Lokalisierung verwendet. Zum Beispiel verwenden Parisotto et al. [45] CNNs in Bezug auf das Simultaneous-Localization-and-Mapping (*SLAM*) Problem. Melekhov et al. [41] schätzen anhand CNNs die relative Pose zweier Kameras. Costante et al. [9] und Wang et al. [64] setzen es im Bereich der visuellen Odometrie ein.

Geleitet von den *state-of-the-art* Lokalisierungsergebnissen der CNNs stellen Kendall et al. [28] den ersten Ansatz zu direkten Posebestimmung nur mit RGB-Bildern vor. PoseNet ist die Modifikation der GoogLeNet [58] Architektur und zweckentfremdet es von der Bildklassifizierung zu einem Pose-Regressor. Trainiert mit einem Datensatz, bestehend aus Paaren von Farbbild und Pose, kann es die sechs Freiheitsgrade der Kamerapose in unbekannten Szenen mittels eines Bildes bestimmen. Dieser Ansatz benötigt weder eine durchsuchbare Bildgalerie noch eine Punktwolke oder Tiefenbilder der Szene. Im Vergleich zu den metrischen Ansätzen wie SLAM oder visuelle Odometrie liefert es eine weniger akkurate Pose. Es bietet jedoch eine hohe Toleranz gegenüber Skalierungs- und Erscheinungsänderungen des Anfragebildes an [47].

Es gibt mehrere Ansätze, die die Genauigkeit von PoseNet übertreffen. Einen Fortschritt erhalten die Autoren von PoseNet durch die hier [27] vorgestellte Anpassung ihres Modells an einem Bayessian Neural Network [11, 37]. Dieselben Autoren erweitern PoseNet mit einer neuen Kostenfunktion unter Berücksichtigung von geometrischen Eigenschaften [26]. Walch et al. [63] und Clark et al. [8] setzen Long-Short-Term-Memory (*LSTM*) [21] Einheiten ein, um Wissen aus der Korrelation von Bildsequenzen zu gewinnen. Wu et al. [65] und Naseer and Burgard [43] augmentieren den Trainingsdatensatz. Wu et al. [65] stocken den vorhandenen Datensatz auf, indem sie die Bilder künstlich rotieren. Naseer and Burgard [43] erweitern zuerst über ein weiteres CNN den Datensatz um Tiefenbildern. Anschließend simulieren die Autoren RGB-Bilder aus verschiedenen Viewpoints. Im Vergleich zu PoseNet verwenden Müller et al. [42] und Melekhov et al. [40] eine andere Architektur. Das Modell von Müller et al. [42] basiert auf die SqueezeNet [23] Architektur. Melekhov et al. [40] stellen HourglassNet, basierend auf einem symmetrischen Encoder-Decoder Architektur, vor. Brahmbhatt et al. [7] und Valada et al. [60, 61] binden zusätzliche Informationen wie z.B. visuelle Odometrie, GPS oder IMU ein.

Jedes dieser Ansätze benötigen annotierte Traininsdaten. Für die Erstellung solcher

Daten wurden beispielsweise mit entsprechender Hardware ausgerüstete Trolleys [22], 3D-Kameras [25] oder SfM-Methoden [28] eingesetzt.

Simulierte 3D-Daten werden in der Literatur oft eingesetzt, um das manuelle Erzeugen und Annotieren von Daten umzugehen. Pishchulin et al. [48], Peng et al. [46], Su et al. [55] und Varol et al. [62] erzeugen ihren Trainingsdaten, indem sie virtuelle Objekte auf reale Hintergrundbildern platzieren. Pishchulin et al. [48] generieren Daten zwecks Personenerkennung und Bestimmung derer körperlicher Pose. Zuvor werden auf den vorhandenen Bildern die körperliche Pose der Personen bestimmt und daran deren 3D Modelle rekonstruiert. Anschließend werden die 3D-Modelle in ihrer Pose variiert auf reale Hintergrundbildern platziert. Die Autoren konnten vergleichbare Ergebnisse zu den vorhandenen Ansätzen mit realen Daten ermitteln. Peng et al. [46] erstellen Daten, um Objekte auf realen Bildern zu detektieren. Von jeder Objektklasse werden 3D-Modelle auf einem Hintergrundbild aus einer Sammlung gelegt. Die Autoren stellen fest, dass das Feintunen mit synthetischen Daten eines Netzwerkes dann zu Abnahme der Akkuratesse führt, wenn das Netzwerk *nur* für die Detektierung eines Objektes bestimmt ist. Hingegen konnten sie eine Steigung der Ergebnisse beim Trainieren mit simulierten Daten auf vortraniertem Netzwerk mit einer größeren Klassifierungskatalog ermitteln. Su et al. [55] generieren einen großen Datensatz mit 3D-Modellen, um den Viewpoint von Objekten auf realen Bildern zu bestimmen. Bei dieser Datengenerierung wird jedes virtuelle Objekt auf zufällige Hintergrundbildern positioniert und mit unterschiedlichen Konfigurationen (*z.B. Beleuchtung*) gerendert. Die Autoren konnten mit der Datenaugmentierung *state-of-the-art* Viewport-Estimation Methoden zur *PASCAL 3D+*[66] Benchmark übertreffen. Varol et al. [62] erstellen künstliche Personen auf Bildern, um beispielsweise den menschlichen Körper in seine Glieder zu segmentieren. Dabei rendern sie zufällige virtuelle Personen mit zufälliger Pose auf beliebige Hintergrundbildern und konnten zeigen, dass die Akkuratesse einiger CNNs durch das Trainieren mit den erzeugten Daten steigt. Fanello et al. [13] rendert künstliche Infrarotbilder von Händen sowie Gesichtern zwecks Tiefenerkennung und Segmentierung der Hand in den einzelnen Fingern sowie des Gesichtes in Bereiche aus einem RGB-Bild. Die Autoren konnten konventionelle Methoden über Helligkeitsabfall übertreffen und vergleichbare Ergebnisse zu den Ansätzen mit einer herkömmlichen 3D-Kamera erzielen. Dosovitskiy et al. [12] erlernen mit synthetischen Daten den optischen Fluss von Bildsequenzen. Hierbei werden auf Hintergrundbildern aus einer Sammlung mehrmals bewegte virtuelle Stühle platziert. Die Autoren konnten mit syntethischen Daten *state-of-the-art* Ansätze über reale Daten übertreffen.

Motiviert von der Datengenerierung über 3D-simulierten Daten stellt Ha et al. [18] einen Ansatz zur bildbasierte Lokalisierung in Gebäuden vor. Dieser Forschungsansatz generiert synthetische Daten aus einem Building-Information-Modeling (*BIM*). Bei den Daten werden die durch das vortrainierte VGG Netzwerk [54] extrahierte Features als wesentlich erachtet und in einer Datenbank gepflegt. Ein reales Aufnahmebild im Gebäude lässt sich durch den Vergleich der Features lokalisieren. Acharya et al. [3, 4] erzeugen ebenso Trainingsdaten aus einem BIM, jedoch verwenden sie zur Lokalisierung keine

Datenbank bedürftiges Verfahren, sondern bestimmen die Pose direkt über PoseNet. Die Daten werden entlang einer ca. $30m$ langem Flugbahn aus der Simulation eines ca. $230m^2$ Korridors gesammelt. Hierbei werden sich in der Realitätstreue vom a) karikaturistisch zu b) fotorealistisch hin über zu c) fotorealistisch-texturiert unterscheidende Daten erzeugt. Traniert mit den unterschiedlichen synthetischen Daten, getestet auf die realen Daten, erzielen die Forscher eine Akkuratesse von a) 6,25m b) 5,99m c) 3,06m in der Position und a) 37,16° b) 11,33° c) 12,25° in der Orientierung. Die besten Ergebnisse konnten die Autoren trainiert mit den d) Gradienten- und e) Kantenbilder der karikaturistischen Daten, getestet auf die Gradientenbilder der realen Aufnahmen, erzielen. Die Autoren erhalten eine Akkuratesse von d) 2,63m e) 1,88m in der Position und d) 6,99° e) 7,73° in der Orientierung.

// Anbindung zu meinem Beitrag

Im weiteren Verlauf des Kapitels werden einige grundlegende Themen erläutert. Zuerst werden künstliche neuronale Netze definiert. Danach wird ein elementares Wissen an Convolutional Neural Networks vermittelt und anschließend bekannte CNN Modelle näher erläutert.

2.2 Künstliche neuronale Netzwerke

Künstliche neuronale Netze sind ein Forschungsgebiet der *künstlichen Intelligenz* und imitieren die Beschaffenheit natürlicher neuronale Netze, um komplexe Probleme zu lösen. Inspiriert von ihren biologischen Vorbildern [1], vernetzen künstliche neuronale Netzwerke (*KNN*) künstliche Neuronen miteinander [1]. Dabei kann die Verbindung unidirektional (*feedforward*) oder bidirektional (*feedback*) sein.

Bei einem feedforward Netzwerk werden die Daten im Netz immer vorwärts übertragen, hingegen kann ein feedback Netzwerk, auch bekannt als *Recurrent Neural Networks*, Daten rückwärts, sowie in einer Schleife zum selben Neuron, übergeben [17]. Da feedback Netzwerke keinen Einsatz in dieser Arbeit haben, ist im weiteren Verlauf dieser Arbeit bei einem Netzwerk immer ein feedforward Ansatz gemeint. In diesem Kapitel werden als Nächstes ein künstliches Neuron definiert und anschließend ein feedforward Netzwerk beschrieben.

2.2.1 Künstliches Neuron

Ein einzelnes Neuron erhält einen Inputsignal auf mehreren Kanälen und löst erst ein Signal (*output*) aus, falls die gewichtete Summe des Inputs einen gewissen Schwellwert erreicht [1]. Abbildung 1 stellt eine beispielhafte Visualisierung eines künstlichen Neurons dar.

[1] das Nervensystem eines Lebewesen, z.B. des Menschen

Ein künstliches Neuron mit der Inputgröße M ist mathematisch die nicht-lineare Funktion $y : \mathbb{R}^M \mapsto \mathbb{R}$ mit dem Parameter x als Input, w als Gewichtsvektor, b als ein Bias, ϕ als eine nicht-lineare *Aktivierungsfunktion* [1]:

$$y(x) = \phi\left(\sum_{m=1}^{M} w_i x_i + b\right) = \phi(W^T x + b) \tag{1}$$

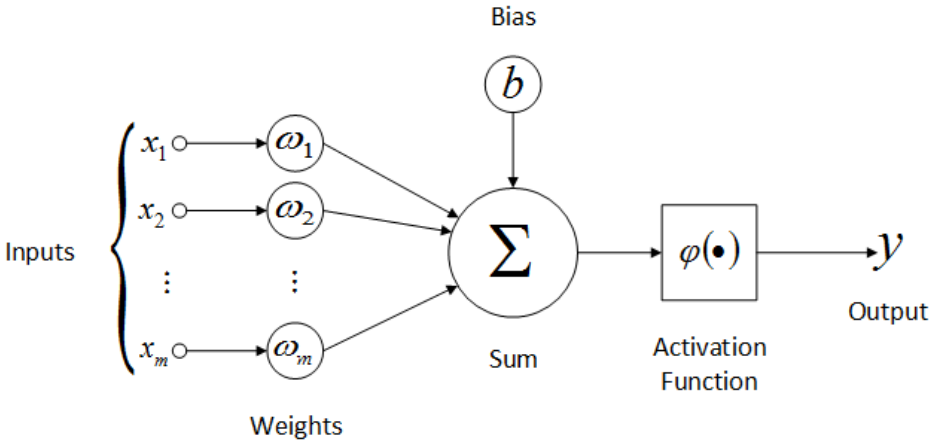

Abbildung 1: Visualisierung eines künstlichen Neurons definiert nach der Gleichung 1. Dieser Neuron summiert das Produkt des Inputvektors x mit den jeweiligen Gewichten w und addiert einen Bias b. Durch die Summe erzeugt die Aktivierungsfunktion ϕ das Output y des Neurons. Entnommen aus [10].

2.2.2 Feedforward Neural Networks

Künstliche Neuronen können zu einer Schicht (*layer*) zusammengeführt werden. Die Verbindung solcher Schichten bildet ein neuronales Netzwerk. Bei einem feedforward (*fully-connected*) Netzwerk übergibt jedes Neuron aus der Schicht l seinen Output y_l an jedem Neuron der Schicht $l + 1$ weiter. Ebenso sind Neuronen aus der gleichen Schicht untereinander nicht verbunden [17]. Eine Schicht l eines feedforward Netzwerkes operiert somit auf das Ouput y_{l-1} und stellt die nicht-lineare Funktion $f_l : \mathbb{R}^{M_{l-1}} \mapsto \mathbb{R}^{M_l}$ dar [6]:

$$y_l = f_l(y_{l-1}) = \phi(W_l^T x_{l-1} + b_l) \tag{2}$$

Die erste Schicht eines Netzwerk wird als Input-, die letzte Schicht als Ouput-Layer bezeichnet. Alle Schichten dazwischen sind Hidden-Layer [17]. Der Ouput-Layer liefert zugleich auch das Ergebnis eines Netzwerks, daher haben die Neuronen des Ouput-Layers grundsätzlich keine Aktivierungsfunktion [1].

Die Tiefe (*depth*) eines Netzwerks ist gegeben durch die Anzahl der Layer [2] und die Breite (*width*) eines Layers wird durch die Anzahl der Neuronen bestimmt [17]. Abilldung 2 illustriert ein feedforward neuronales Netz als ein azyklischer Graph.

Ziel eines KNNs ist es eine Funktion f^* zu approximieren, dass einen Input x auf einen Output y abbildet. Durch das Output y kann das Input x klassifiziert oder ein Wert regressiert werden. Sei $y = f(x;\theta)$ solch eine Funktion, dann besetzt ein KNN die Werte des θ Parameters mit eines der besten Approximierung. Der Parameter θ stellt hierbei die Gewichte dar, welche erlernt werden sollen [17]. Das Lernen ist die strategische Anpassung der Gewichte über Input-Output Paare (*Trainingsdaten*) und findet grundsätzlich durch ein *Backpropagation*-Verfahren statt [17].

Die Funktion $y = f(x;\theta)$ bildet sich aus den Funktionen der Schichten (Gleichung 2) im Netzwerk und kann bei einer Tiefe L repräsentiert werden als die folgende Funktion $f : \mathbb{R}^{M_0} \mapsto \mathbb{R}^{M_L}$ [17, 6]:

$$y = f(x;\theta) = f_L(...f_2(f_1(x))) = (f_L \circ ... \circ f_2 \circ f_1)(x) \tag{3}$$

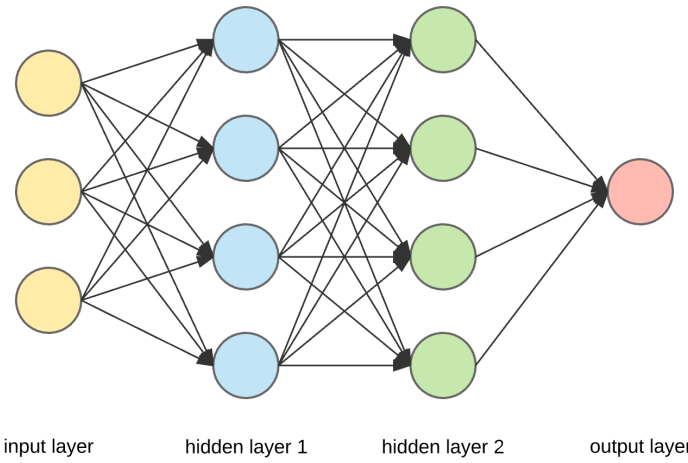

Abbildung 2: Ein feedforward neuronales Netz mit der Tiefe 3 bestehend aus einem Input-Layer der Breite 3, aus zwei Hidden-Layer der Breite 4 und einem Output-Layer der Breite 1. Mit der Gleichung 3 lässt sich dieses Netzwerk als die Funktion $f : \mathbb{R}^3 \mapsto \mathbb{R}^1$ mit $f(x) = f_3(f_2(f_1(x)))$ darstellen.

[2] der Input-Layer ist ausgeschloßen

2.3 Convolutional Neural Networks

Einfache neuronale Netze, wie sie in Abschnitt 2.2 beschrieben werden, arbeitet auf einem Inputvektor $x \in \mathbb{R}^M$. Hingegen arbeiten CNNs auf einem drei-dimensionalen Inputvolumen $x \in \mathbb{R}^{width} \times \mathbb{R}^{heigth} \times \mathbb{R}^{depth}$. CNNs werden hauptsächlich im Kontext von Bildern eingesetzt, dabei stellt z.B. ein 32 × 32 RGB-Bild ein Volumen von 32 × 32 × 3 dar [2].

Angefangen mit der LeNet-5 [32] Architektur, setzt sich typischerweise ein Convolutional Neural Network aus einer Sequenz von unterschiedlichen Layer-Arten zusammen [58, 2]. Im weiteren Verlauf dieses Kapitels werden die Arten der Layer beschrieben. Tabelle 1 gibt eine Übersicht der Layer und ihrer Parameter an.

Tabelle 1: Übersicht der Paramter und Hyperparamter der Layer eines Convolutional Neural Networks. Die Parameter werden während der Trainingsphase optimiert und Hyperparameter werden vorab fest definiert [67].

Art des Layers	**Parameter**	**Hyperparameter**
Convolutional Layer	Filter	Anzahl der Filter Filtergröße Stride Padding Aktivierungsfunktion
Pooling Layer	*keine*	Pooling Methode Filtergröße Stride Padding
Fully-Connected Layer	Gewichte	Anzahl der Gewichte Aktivierungsfunktion

2.3.1 Convolution Layer

Der Convolutional Layer ist der Hauptbestandteil eines CNNs, welches die Kombination eines Convolution Operationen und einer Aktivierungsfunktion ist [67]. Abbildung 3 illustriert eine Convolution Operation. Abbildung 4 stellt eine Aktivierungsfunktion dar.

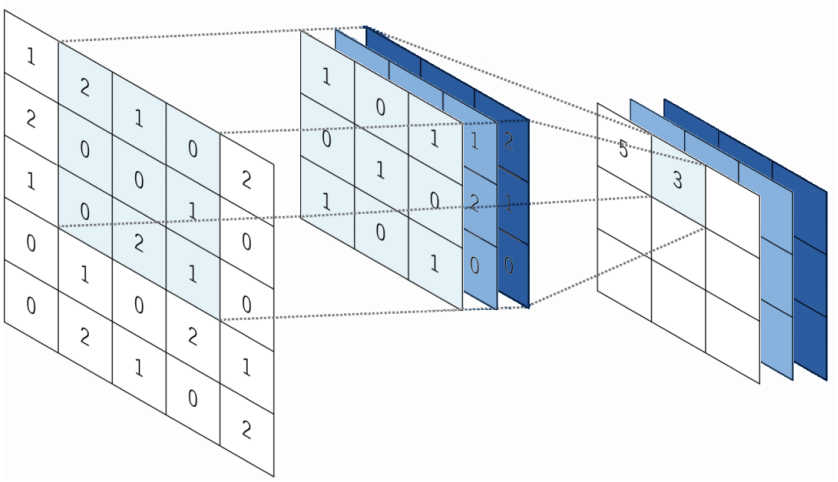

Abbildung 3: Ein Beispiel für die Convolution Operation mit 3 Filtern der Größe 3 × 3, einem Stride von 1 und keinem Padding. Ein Filter bewegt sich entlang des gesamten Inputs mit der Schrittweite *Stride* und bildet die Summe der elementweise multiplizierten Werte. Die Summe wird dann im Output an die korrespondierende Position geschrieben. Dieser Vorgang wiederholt sich für jedes Filter. [67].

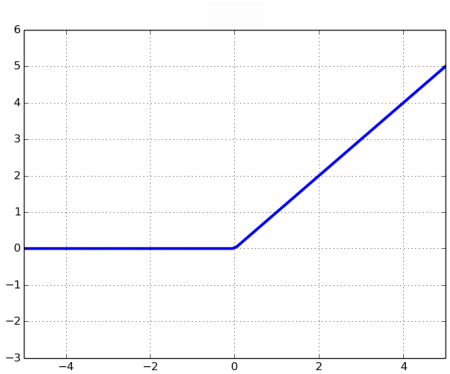

Abbildung 4: Ein Beispiel für eine Aktivierungsfunktion. Die ReLU (*rectified liniear unit*) Aktivierungsfunktion wird typischerweise in CNNs eingesetzt und ist mathematisch definiert als: $f(x) = max(0, x)$ [17]

2.3.2 Pooling Layer

Auf einen Convolutional Layer folgt i.d.R. ein Pooling Layer. Pooling Layer reduzieren die Größe eines Inputvolumens und verringert somit die Anzahl der erlernbaren Parameter. Die Pooling Operation wird auf jeder Schicht der Eingabe ausgeführt [2]. Die meist verbreitete *Pooling Methode* ist die Max-Pooling. Beim Max-Pooling iteriert ein Filter einer bestimmten Größe mit einer Schrittweite, gegeben durch dem Stride, über das Inputvolumen und extrahiert das Maximum im aktuellen Filterbereich. Das Maximum wird für die weitere Berechnung beibehalten und die restlichen Werte verworfen [2]. In dieser Arbeit wird neben der Max-Pooling Operation auch die Average-Pooling Operation eingesetzt. Das Average-Pooling behält im aktuellen Filterbereich den Durchschnittswert, statt das Maximum [2].

Im Vergleich zu einem Convolutional Layer wird ein Pooling Layer nur aus Hyperparameter definiert und bleibt daher statisch [67]. Abbildung 5 zeigt eine beispielhafte Ausführung einer Max-Pooling Operation. Tabelle 1 listet die Hyperparameter eines Pooling Layers auf.

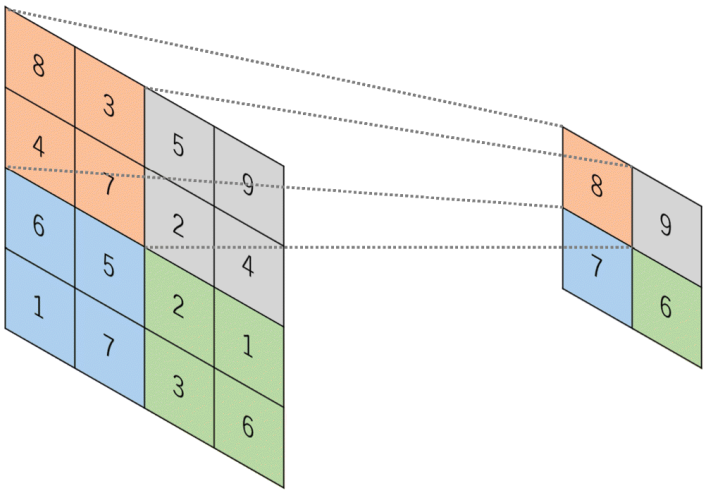

Abbildung 5: Ein Beispiel für eine Max-Pooling Operation mit einer Filtergröße von 2 × 2, einem Stride von 2 mit keinem Padding. In diesem Beispiel wird der Input in 2 × 2 Bereiche unterteilt und der Maximum jedes Bereiches als Output berechnet. Es werden die markantesten Werte einer Nachbarschaft behalten und der Rest verworfen. Diese Operation führt zu einer Reduzierung der Inputgröße um den Faktor 2. Entnommen aus [67].

2.3.3 Fully-Connected Layer

Nach einer Periode von Convolution und Pooling Layer folgt meist eine fully-connected (*FC*) Layer, auch bekannt als *Dense Layer*. Dieser Layer folgt den gleichen Konzept der feedforward Neural Networks, wie in Abschnitt 2.2.2 beschrieben. Das Output dieses Layers wird häufig einer weiteren Aktivierungsfunktion übergeben [67] und prinzipiell wird dadurch das Output des CNNs bestimmt.

2.4 Bekannte CNN Modelle

Es existiert eine Menge von bekannten CNN Modellen mit ausgezeichneten Ergebnissen in internationalen Wettbewerben, z.B. Wettbewerbe wie die ILSVRC [51]. Dieses Kapitel behandelt nur die Architektur des GoogLeNet Modells und dessen Modifikation PoseNet.

2.4.1 GoogLeNet

GoogLeNet, konstruiert von Szegedy et al. [58] im Jahr 2014, ist der Sieger des ILSVRC14 Wettbewerbes. GoogLeNet ist eine besondere Inkarnation des sogenannten Inception Moduls, welche zeitgleich mit GoogLeNet vorgestellt wird. Ein Inception Modul beinhaltet mehrere Convolutional sowie einen Pooling Layer und verarbeitet das Inputvolumen parallel auf 4 Zweigen. Vor rechenintensiven Convolutional Layer werden zusätzliche 1×1 dimensionsreduzierende Layer geschaltet. Abschließend werden die Zweige, welche ein Volumen mit der gleiche Breite und Höhe produzieren, zu einem Outputvolumen (in die Tiefe) zusammengeführt. Durch die parallele Berechnung der Zweige und das Einschalten von dimensionsreduzierenden 1×1 Layer vor rechenintensiven Operationen ermöglicht ein Inception Modul neuronale Netze bei zunehmender Tiefe und Breite konstante Rechenkapazität in Anspruch zu nehmen. Inzwischen wurde das Inception Modul [59] und die gesamte Architektur rundherum [57] weiterentwickelt. Abbildung 6 stellt das Inception (*v1*) Modul dar.

GoogLeNet ist ein 22 Layer tiefes neuronales Netz [3], bestehend aus 9 Inception Modul (je von Tiefe 2). Das Ungewöhnliche an GoogLeNet ist, das es 3 Output-Zweigen hat. Jedes der Ouput-Zweige produziert ein Klassifizierungsergebnis. Output-Zweig 2 & 3 sind Hilfszweige, welche nur in der Trainingsphase einen Einfluss haben und in die Evaluationsphase nicht einfließen. Abbildung 7 illustriert die GoogLeNet Architektur in Detail.

[3] inkl. Output Layer

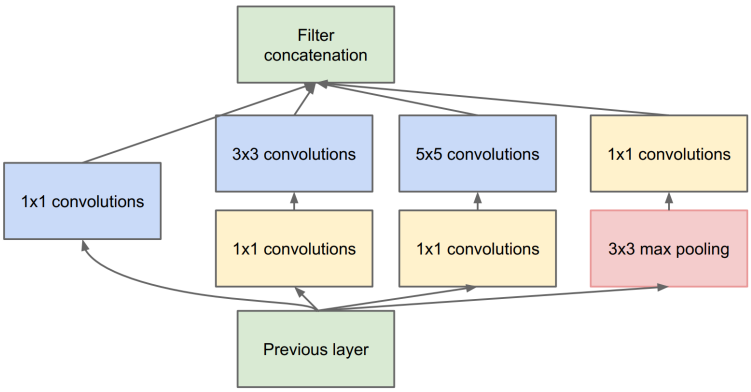

Abbildung 6: Darstelltung des Inception (*v1*) Moduls. Das Inputvolumen wird auf 4 Zweige unabhängig verarbeitet. Die Ergebnisse der unabhängigen Operationen haben die gleiche Breite und Höhe. Das Output des Inception Moduls wird durch das Konkatenieren (in der Tiefe) der einzelnen Ergebnisse bestimmt. Die gelben 1×1 Convolution Layer dienen zur Dimensionsreduktion. Entnommen aus [58].

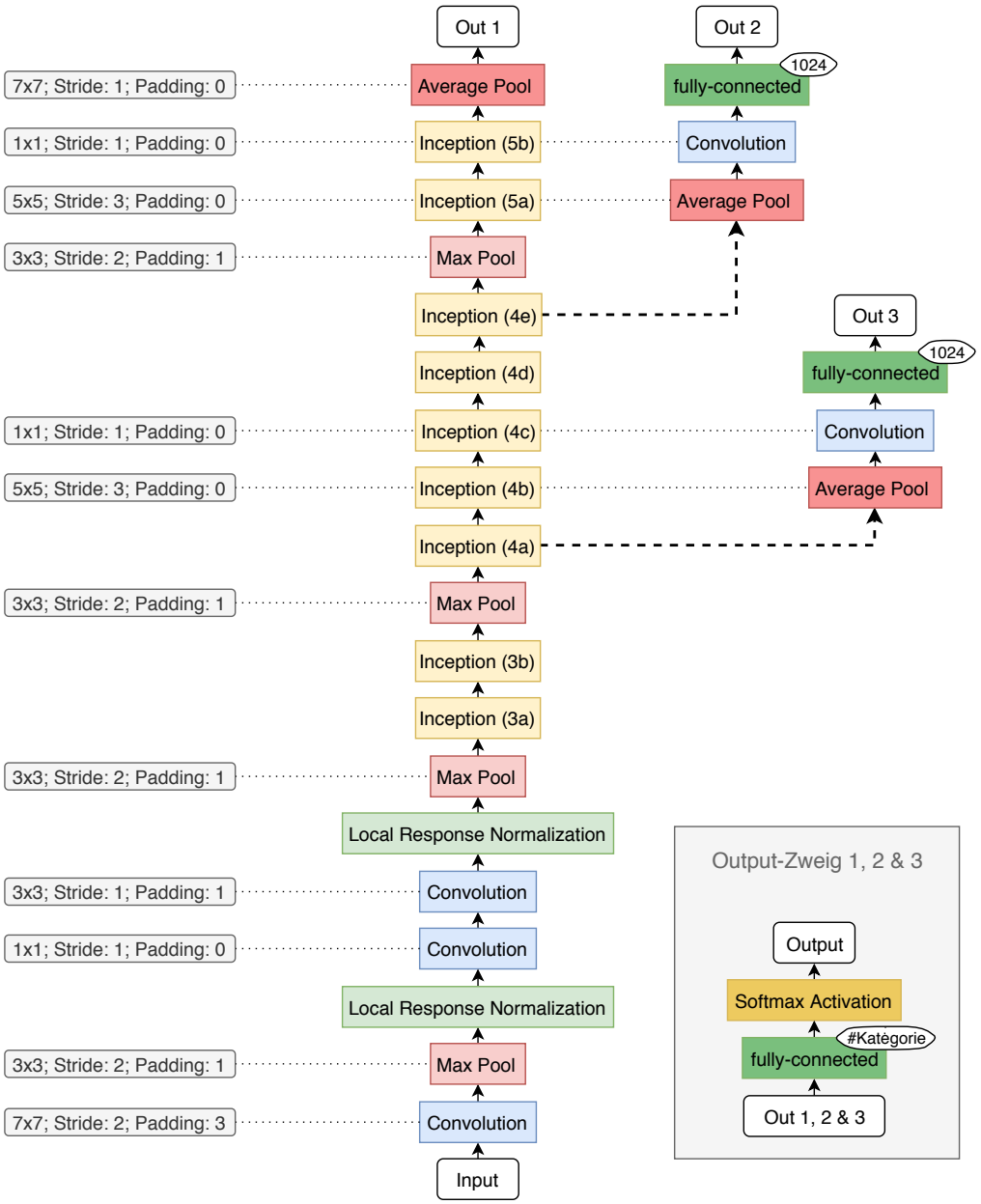

Abbildung 7: Der architektonische Aufbau des GoogLeNet Models. Das Inception-Block wird in Abbildung 6 detaillierter dargestellt. Es gibt 3 Ouput-Zweige, die mit einer identisch aufgebauten Output-Knoten enden. Die Werte der Output-Zweige 2 & 3 haben nur während der Trainingsphase einen Einfluss und werden in der Evaluationsphase verworfen. In der Trainingsphase wird vor dem Output-Knoten 1 ein Dropout von 40% und vor den Output-Knoten 2 & 3 ein Dropout von 70% angewandt. Jedes Convolution Layer hat die ReLU (Abbildung 4) als Aktivierungsfunktion. Die *Local-Response-Normalization* [31] ist ein bekanntes Normierungsverfahren und kommt prinzipiell in Verbindung mit der ReLU Aktivierungsfunktion vor. Abbildung basiert auf [58].

2.4.2 PoseNet

PoseNet, vorgestellt von Kendall et al. [28] im Jahr 2015, basiert auf die im Kapitel 2.4.1 eingeführte GoogLeNet Architektur. Statt eine Wahrscheinlichkeitsverteilung der Klassifierungsergebnisse, liefert PoseNet die 6 Freiheitsgrade einer Pose $y = [\boldsymbol{p}; \boldsymbol{q}]$, bestehend aus einem Positionsvektor $\boldsymbol{p} \in \mathbb{R}^3$ und eine Quaternion der Orientierung $\boldsymbol{q} \in \mathbb{R}^4$.

PoseNet modifiziert die GoogLeNet Architektur an den Output-Knoten, dargestellt in Abbildung 7, wie folgt [28]:

- Jedes der Softmax-Klassifikatoren werden ersetzt durch Regressoren. Dabei wird der Softmax-Activation Layer entfernt und der FC-Layer so modifiziert, dass es einen 7-dimensionalen Vektor[4] ausgibt.
- Vor dem finalen Regressor des Output-Zweiges 1 wird ein weiteres FC-Layer der Breite 2048 eingefügt.

Abbildung 8 veranschaulicht die Modifikation von GoogLeNet durch PoseNet.

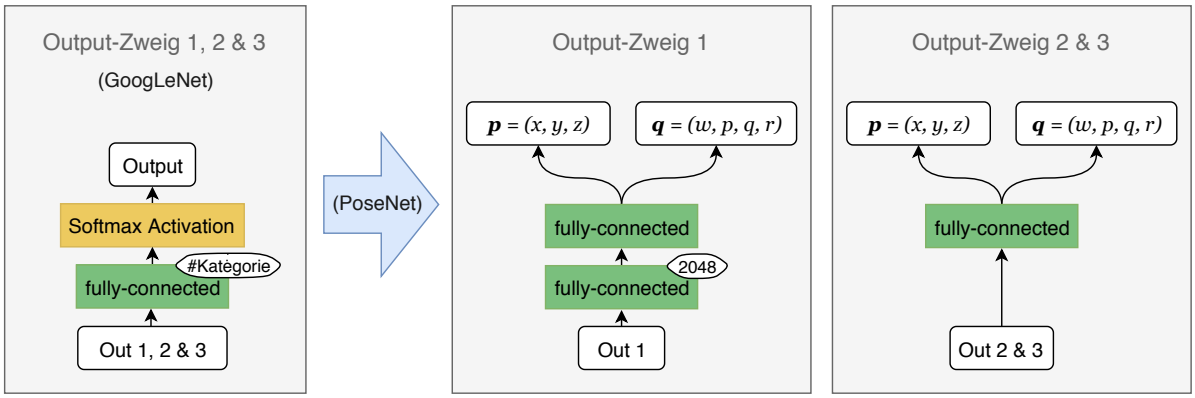

Abbildung 8: Veranschaulichung der Modifikation von den Output-Zweigen der GoogLeNet Architektur durch PoseNet. Die Softmax Aktivierungsfunktion sowie das fully-connected Layer mit Anzahl der Klassifizierungskategorien als Breite wurde von allen Output-Modulen entfernt. Jeder Output-Zweig hat einen FC-Regressionsschicht erhalten, welches die 6 Freiheitsgrade einer Pose bestimmt. Der Output-Zweig 1 wird zusätzlich mit einer weiteren FC-Layer der Breite 2048 vor dem Regressionsschicht erweitert [28].

[4](3) für die Position und (4) für die Orientierung

Die Autoren Kendall et al. [28] stellen zusätzlich eine neue Kostenfunktion, mit den Parametern $\hat{y} = [\hat{\boldsymbol{p}}; \hat{\boldsymbol{q}}]$ als Soll-Wert und $y = [\boldsymbol{p}; \boldsymbol{q}]$ als Ist-Wert, vor:

$$loss(I) = \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_2 + \beta \left\| \hat{\boldsymbol{p}} - \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \right\|_2 \tag{4}$$

Der Hyperparameter β soll eine Balance der Kosten zwischen dem Positions- und Orientierungsdiskrepanz darstellen und wird in Gebäuden im Wertebereich zwischen 120 bis 750, sowie außerhalb des Gebäudes zwischen 250 bis 2000, empfohlen Kendall et al. [28].

3 Training des CNNs

cnns werden erst modelliert und anschließend traniert. posenet caffe implementierung Trainingsdaten sind die synthetischen Daten und Testdaten sind die realen Daten

3.1 Erhebung der realen Daten

Intel RealSense T265 D435, ROS

3.2 Generierung der synthetischen Daten

Blender, eigenes Addons erstellt ein Kamerakonstrukt, welches ein NURBs-Pfad entlag aufnahmen erstellt.

3.3 Verarbeitung der Daten

3.4 Trainingsparameter

Loss function beta learningrate weight sdecay

4 Ergebnisse

4.1 Versuch 1

4.2 Versuch 2

4.3 Versuch 3

5 Diskussion

6 Fazit

Literatur

[1] CS231n convolutional neural networks for visual recognition, . URL `http://cs231n.github.io/neural-networks-1/`.

[2] CS231n convolutional neural networks for visual recognition, . URL `http://cs231n.github.io/convolutional-networks/#pool`.

[3] D. Acharya, K. Khoshelham, and S. Winter. BIM-PoseNet: Indoor camera localisation using a 3d indoor model and deep learning from synthetic images. 150:245–258, . doi: 10.1016/j.isprsjprs.2019.02.020.

[4] D. Acharya, S. Roy, K. Khoshelham, and S. Winter. MODELLING UNCERTAINTY OF SINGLE IMAGE INDOOR LOCALISATION USING a 3d MODEL AND DEEP LEARNING. .

[5] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. pages 2911–2918. doi: 10.1109/CVPR.2012.6248018.

[6] C. Bauckhage, C. Ojeda, J. Schucker, R. Sifa, and S. Wrobel. Informed machine learning through functional composition. page 6.

[7] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-aware learning of maps for camera localization. pages 2616–2625. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Brahmbhatt_Geometry-Aware_Learning_of_CVPR_2018_paper.html`.

[8] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. VidLoc: A deep spatio-temporal model for 6-DoF video-clip relocalization. URL `http://arxiv.org/abs/1702.06521`.

[9] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia. Exploring representation learning with CNNs for frame-to-frame ego-motion estimation. 1(1):18–25. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2015.2505717. URL `http://ieeexplore.ieee.org/document/7347378/`.

[10] R. De Oliveira, R. C. Fernandes Araújo, F. Barros, A. Paranhos Segundo, R. Zampolo, W. Fonseca, V. Dmitriev, and F. Brasil. A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges. 16:628–645. doi: 10.1590/2179-10742017v16i3854.

[11] J. S. Denker and Y. LeCun. Transforming neural-net output levels to probability distributions. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 853–859. Morgan-Kaufmann. URL `http://papers.nips.cc/paper/419-transforming-neural-net-output-levels-to-probability-distributions.pdf`.

[12] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.316. URL https://ieeexplore.ieee.org/document/7410673/.

[13] S. R. Fanello, T. Paek, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, and S. B. Kang. Learning to be a depth camera for close-range human capture and interaction. 33(4):1–11. ISSN 07300301. doi: 10.1145/2601097.2601223. URL http://dl.acm.org/citation.cfm?doid=2601097.2601223.

[14] R. Girshick. Fast r-CNN. URL http://arxiv.org/abs/1504.08083.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. URL http://arxiv.org/abs/1311.2524.

[16] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi. Real-time RGB-d camera relocalization via randomized ferns for keyframe encoding. 21(5):571–583. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2360403. URL http://ieeexplore.ieee.org/document/6912003/.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[18] I. Ha, H. Kim, S. Park, and H. Kim. Image-based indoor localization using BIM and features of CNN. doi: 10.22260/ISARC2018/0107. URL http://www.iaarc.org/publications/2018_proceedings_of_the_35th_isarc/image_based_indoor_localization_using_bim_and_features_of_cnn.html.

[19] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. FuseNet: Incorporating depth into semantic segmentation via fusion-based CNN architecture. In S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, editors, *Computer Vision – ACCV 2016*, volume 10111, pages 213–228. Springer International Publishing. ISBN 978-3-319-54180-8 978-3-319-54181-5. doi: 10.1007/978-3-319-54181-5_14. URL http://link.springer.com/10.1007/978-3-319-54181-5_14.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. URL http://arxiv.org/abs/1512.03385.

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. 9(8):1735–1780. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[22] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, and E. Steinbach. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. In *2012 19th IEEE International Conference on Image Processing*, pages 1773–1776. IEEE. ISBN 978-1-4673-2533-2 978-1-4673-2534-9 978-1-4673-2532-5. doi: 10.1109/ICIP.2012.6467224. URL http://ieeexplore.ieee.org/document/6467224/.

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. URL http://arxiv.org/abs/1602.07360.

[24] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206587. URL https://ieeexplore.ieee.org/document/5206587/.

[25] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. *KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera*. ISBN 978-1-4503-0716-1. URL https://www.microsoft.com/en-us/research/publication/kinectfusion-real-time-3d-reconstruction-and-interaction-using-a-moving-depth-camera/.

[26] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564. IEEE, . ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.694. URL http://ieeexplore.ieee.org/document/8100177/.

[27] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. . URL http://arxiv.org/abs/1509.05909.

[28] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. pages 2938–2946. URL https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Kendall_PoseNet_A_Convolutional_ICCV_2015_paper.html.

[29] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. URL http://arxiv.org/abs/1511.07386.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., . URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., . URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha. Gradient-based learning applied to document recognition. page 46.

[33] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. 2(1):1–19. ISSN 15516857. doi: 10.1145/1126004.1126005. URL http://portal.acm.org/citation.cfm?doid=1126004.1126005.

[34] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. pages 1119–1127, . URL http://openaccess.thecvf.com/content_cvpr_2015/html/Li_Depth_and_Surface_2015_CVPR_paper.html.

[35] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, volume 7572, pages 15–29. Springer Berlin Heidelberg, . ISBN 978-3-642-33717-8 978-3-642-33718-5. doi: 10.1007/978-3-642-33718-5_2. URL http://link.springer.com/10.1007/978-3-642-33718-5_2.

[36] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. 32(1):1–19. ISSN 1552-3098. doi: 10.1109/TRO.2015.2496823.

[37] D. J. C. MacKay. A practical bayesian framework for backprop networks. 4:448–472.

[38] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. 9905:580–596. doi: 10.1007/978-3-319-46448-0_35. URL http://arxiv.org/abs/1608.02755.

[39] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. S. Torr. Random forests versus neural networks - what's best for camera localization? URL http://arxiv.org/abs/1609.05797.

[40] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Image-based localization using hourglass networks. pages 879–886, . URL http://openaccess.thecvf.com/content_ICCV_2017_workshops/w17/html/Melekhov_Image-Based_Localization_Using_ICCV_2017_paper.html.

[41] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Relative camera pose estimation using convolutional neural networks. In J. Blanc-Talon, R. Penne, W. Philips, D. Popescu, and P. Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, volume 10617, pages 675–687. Springer International Publishing, . ISBN 978-3-319-70352-7 978-3-319-70353-4. doi: 10.1007/978-3-319-70353-4_57. URL http://link.springer.com/10.1007/978-3-319-70353-4_57.

[42] M. S. Müller, S. Urban, and B. Jutzi. SQUEEZEPOSENET: IMAGE BASED POSE REGRESSION WITH SMALL CONVOLUTIONAL NEURAL NETWORKS FOR REAL TIME UAS NAVIGATION. IV-2/W3:49–57. ISSN 2194-9050. doi: 10.5194/isprs-annals-IV-2-W3-49-2017. URL https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-2-W3/49/2017/.

[43] T. Naseer and W. Burgard. Deep regression for monocular camera-based 6-DoF global localization in outdoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1525–1530. IEEE. ISBN 978-1-5386-2682-5. doi: 10.1109/IROS.2017.8205957. URL http://ieeexplore.ieee.org/document/8205957/.

[44] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. pages 1520–1528. URL https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.html.

[45] E. Parisotto, D. S. Chaplot, J. Zhang, and R. Salakhutdinov. Global pose estimation with an attention-based recurrent network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 350–35009. IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00061. URL https://ieeexplore.ieee.org/document/8575522/.

[46] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. URL http://arxiv.org/abs/1412.7122.

[47] N. Piasco, D. Sidibé, C. Demonceaux, and V. Gouet-Brunet. A survey on visual-based localization: On the benefit of heterogeneous data. 74:90 – 109. doi: 10.1016/j.patcog.2017.09.013. URL https://hal.archives-ouvertes.fr/hal-01744680.

[48] L. Pishchulin, A. Jain, M. Andriluka, T. Thormählen, and B. Schiele. Articulated people detection and pose estimation: Reshaping the future. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3178–3185. doi: 10.1109/CVPR.2012.6248052.

[49] F. Radenović, G. Tolias, and O. Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In B. Leibe, J. Matas, N. Sebe, and

M. Welling, editors, *Computer Vision – ECCV 2016*, volume 9905, pages 3–20. Springer International Publishing. ISBN 978-3-319-46447-3 978-3-319-46448-0. doi: 10.1007/978-3-319-46448-0_1. URL http://link.springer.com/10.1007/978-3-319-46448-0_1.

[50] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-CNN: Towards real-time object detection with region proposal networks. URL http://arxiv.org/abs/1506.01497.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. URL http://arxiv.org/abs/1409.0575.

[52] D. R. d. Santos, M. A. Basso, K. Khoshelham, E. d. Oliveira, N. L. Pavan, and G. Vosselman. Mapping indoor spaces by adaptive coarse-to-fine registration of RGB-d data. 13(2):262–266. ISSN 1545-598X. doi: 10.1109/LGRS.2015.2508880.

[53] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-d images. pages 2930–2937. URL http://openaccess.thecvf.com/content_cvpr_2013/html/Shotton_Scene_Coordinate_Regression_2013_CVPR_paper.html.

[54] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. URL http://arxiv.org/abs/1409.1556.

[55] H. Su, C. R. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3d model views. URL http://arxiv.org/abs/1505.05641.

[56] L. Svarm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. 39(7):1455–1461. ISSN 0162-8828, 2160-9292. doi: 10.1109/TPAMI.2016.2598331. URL http://ieeexplore.ieee.org/document/7534854/.

[57] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-ResNet and the impact of residual connections on learning. . URL http://arxiv.org/abs/1602.07261.

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. pages 1–9, . URL https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html.

[59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. pages 2818–2826, . URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html.

[60] A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6939–6946. IEEE, . ISBN 978-1-5386-3081-5. doi: 10.1109/ICRA.2018.8462979. URL https://ieeexplore.ieee.org/document/8462979/.

[61] A. Valada, N. Radwan, and W. Burgard. Incorporating semantic and geometric priors in deep pose regression. page 4, .

[62] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4627–4635. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.492. URL http://ieeexplore.ieee.org/document/8099975/.

[63] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using LSTMs for structured feature correlation. URL http://arxiv.org/abs/1611.07890.

[64] S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989236. URL http://ieeexplore.ieee.org/document/7989236/.

[65] J. Wu, L. Ma, and X. Hu. Delving deeper into convolutional neural networks for camera relocalization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. IEEE. ISBN 978-1-5090-4633-1. doi: 10.1109/ICRA.2017.7989663. URL http://ieeexplore.ieee.org/document/7989663/.

[66] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond PASCAL: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. IEEE. ISBN 978-1-4799-4985-4. doi: 10.1109/WACV.2014.6836101. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6836101.

[67] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. 9(4):611–629. ISSN 1869-4101. doi: 10.1007/s13244-018-0639-9. URL https://doi.org/10.1007/s13244-018-0639-9.

[68] W. Zhang and J. Kosecka. Image based localization in urban environments. page 9.