CNN basiertes Tracking für eine mobile Augmented Reality Anwendung

Daniel Eggert | daniel.eggert@haw-hamburg.de | Hochschule für angewandte Wissenschaften Hamburg

Ziele

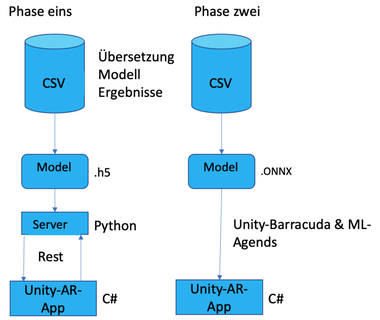
Augmented Reality ist das Verschmelzen der realen mit einer virtuellen Welt. So sollen für ein Forschungs- und Entwicklungsprojekt im CSTI[6] zwei-dimensionale Schaltpläne erkannt und per 2D-Annotation mit virtuellen Informationen angereichert werden. Hierbei gibt es verschiedene Möglichkeiten eine Erkennung zu realisieren. Mit der Möglichkeit die Objekterkennung, als Bestandteil des Trackings, durch maschinelles Lernen zu verwirklichen, beschäftigt sich die vorliegende Arbeit. Der Fokus liegt dabei auf dem Training eines CNNs, welches mit den zu erkennenden Schaltplänen trainiert wird sowie der Integration dieses Modells in Unity.

Motivation

Der Grund die Objekterkennung mit CNNs zu realisieren und nicht etwa mit der Methode des "feature point mapping" liegt darin, dass neuronale Netze auch mehrere Objekte gleichzeitig in einem Bild erkennen können. Des weiteren müssen für das Feature Point Mapping die Objekte reich an Textur und Featurepoints sein, dies ist nicht immer der Fall, gerade für solche Fälle eignen sich CNNs. Außerdem lässt sich ein Netz, welches Objekte erkennt erweitern, sodass auch die Position eines Objektes im Bild getrackt werden kann, wie es das Beispiel des PoseCNN's zeigt.[1]

Auch wenn die Positionserkennung für das Projekt nicht benötigt wird, so ist es doch ein wichtiger Bestandteil der meisten AR-Anwendungen und stellt eine Herausforderung dar, welche mit CNNs gelöst werden kann.[2]

Projektübersicht



Übersicht der Projektphasen

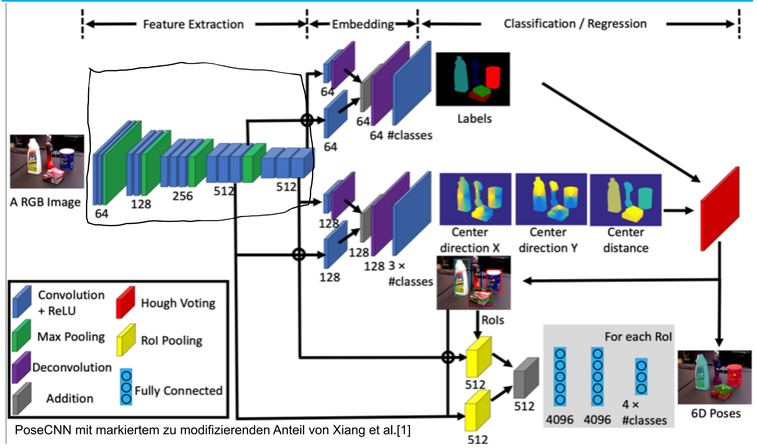
Phase eins(aktuelle Phase):

- Erkennung über eine Rest-Schnittstelle
- Verbessern des Modells
- Testen in einer Unity Simulation

Phase zwei(noch nicht erfolgt):

- Modell in ein Unity kompatibles Modell umwandeln
- I-Phone & I-Pad Performance Test

Vorgehensweise der Modellentwicklung

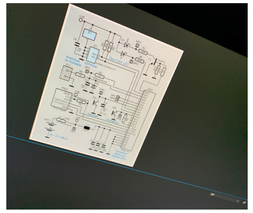


PoseCNN mit markiertem zu modifizierenden Anteil von Xiang et al.[1]

- Der markierte Anteil wird an den Projektfall angepasst bis eine hohe accuracy und val_accuracy erreicht wird
- Unterscheidung von mindestens drei verschiedenen Schaltplänen
- Generierung eines eigenen Trainingdatensatzes

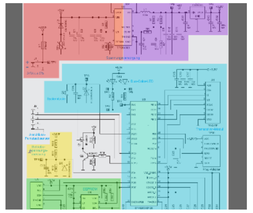
Die AR-App

Tracking



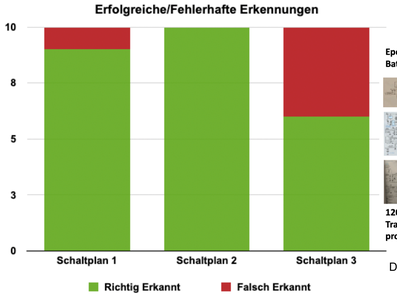
- Aufnahme der Umgebung
- Senden einer Momentaufnahme an den Python Server(Flask)
- Übergang zu der Annotation bei erfolgreicher Erkennung

Annotation

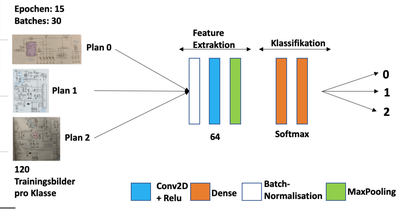


- Hervorhebung der interessanten Stellen
- Bei Berührung der markierten Stellen werden die zusätzlichen Informationen zu diesen angezeigt

(Zwischen)Ergebnisse



Test mit 10 Bildern jeder Klasse



Das aktuelle Modell in ähnlicher Darstellung wie von Xiang et al [1]

- Die Übernahme der kompletten Feature Extraktion des PoseCNN führte bei dem Anwendungsfall zu keinem Trainingsfortschritt, deswegen wurde das Modell zunächst drastisch verkleinert
- Das aktuelle Modell besitzt eine accuracy von 0.97 und eine val_accuracy von 0.34 bei einem loss von 0.04 und val_loss von 1.4
- Aktuelles Problem Overfitting, weitere Schritte dagegen sind notwendig

Ausblick

Sobald das Modell soweit verbessert wurde, sodass es die Schaltpläne zuverlässig erkennt, soll es anschließend wie in der Projektübersicht beschrieben in Unity selbst integriert werden. Denn auch wenn sich das Tracking mit einem Server realisieren lässt, ist es gerade für mobile AR-Anwendungen wichtig, dass sich diese jederzeit und an jedem Ort ausführen lassen. Deswegen wird zu diesem Zeitpunkt auch die Performance der Anwendung auf einem Testgerät(I-Phone& I-Pad) gemessen.

Weitere mögliche Schritte wären das Modell so zu erweitern, dass es auch die Position oder Rotation eines Objektes erkennt und somit real-time Tracking ermöglicht.

Literatur

(1) Yu Xiang u.a."Posecnn: A convolutional neuralnetwork for 6d object pose estimation in clutteredscenes". In:arXiv preprint arXiv:1711.00199(2017).

(2) Shu Zhang, Cheng Song und Rafael Radkowski."Setforge-Synthetic RGB-D Training Data Generation to Support CNN-Based Pose Estimation forAugmented Reality". In:2019 IEEE InternationalSymposium on Mixed and Augmented Reality Ad-junct (ISMAR-Adjunct). IEEE. 2019, S. 237–242.

(3) Matthieu Deru und Alassane Ndiaye.Deep Learning mit Tensorflow, Keras und Tensorflow.js. RheinwerkVerlag, 2020.

(4) Schaltpläne aus den ELV-Magazinen 2/2019, 6/2019 und 1/2020

(5) Dekorationsbilder von https://pixabay.com/

(6) Creative Space for Technical Innovations (CSTI) https://csti.haw-hamburg.de