7R-02

深層学習による深度画像推定を用いたARアプリケーションの開発

吉田　顕†　　田村　仁†

日本工業大学　工学部　創造システム工学科†

1. はじめに

AR では実在する物体の位置を把握して、表示したい 3D オブジェクトや注釈などとの前後関係を考慮し表示する。この際に、3D オブジェクトが実在する物体の後ろにあるように表示したいとき時、重ならないようにするための処理を隠れ処理という。しかし、スマートフォンなどの個人携帯端末にはほとんど深度画像センサが搭載されておらず、RGB カメラが搭載されていることが一般的である。そのためスマートフォンではこの隠れ処理を行うことができない。(図 1)

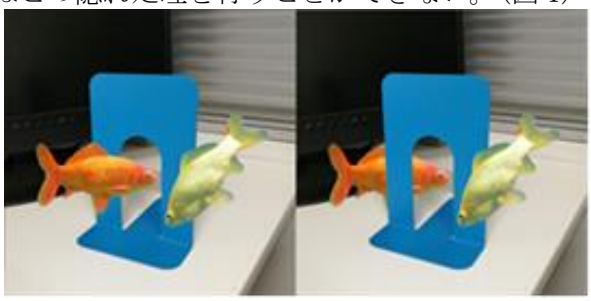

図 1. 隠れ処理を行う前(左)と後(右)の例

そこで深層学習による深度画像推定を用いることで、RGB カメラのみでスマートフォン上で AR の隠れ処理を行うことができるのではないかと考えた。

2. 関連研究

類似システムとして、RGB カメラを固定して使用することを前提として、現実物体と同じ位置に透明化した同じ形状の 3D 物体を作成して表示することで現実物体と 3D 物体の前後関係を考慮して表現することができるシステム[1]がある。このシステムでは、隠れ処理を行うことはできるが RGB カメラを固定して使用しなければならない。また、前後関係を考慮したい物体を事前に一度 3D 物体として作成する必要があるため動く現実物体には対応することができず、限定的な利用となってしまって開発の幅が狭くなっていしまっている

3. ARアプリケーションの処理内容

本研究では、スマートフォン上で隠れ処理を行うことができる AR アプリケーションの開発を目的としている。隠れ処理を行うためには現実の物体の距離を測る必要があり、深度センサなどによる深度画像があると実現できる。しかし一般的なスマートフォンには深度センサは搭載されていないため、深層学習モデルを用いることで距離画像推定を行う。スマートフォン上で深層学習モデルを動作させるためには TensorFlow.js や TensorFlow Lite などが存在する。本研究では深層学習モデルを動作させるために Web フォーマットで推論を行うことができる Tensorflow.js を選択した。TensorFlow.js には python モデルを JavaScript にコンバートする機能がある。そこで、モデルの学習時間の短縮のために深層学習は PC 上で行うことにした。学習に使用する画像合成手法には Pix2Pix を用いた。Pix2Pix は二種類の画像をペアとして対学習を行い、片方の画像を入力するともう片方の画像と似た画像を生成する画像生成アルゴリズムの一つである。この学習には SceneNet RGB-D[2]という photo 画像(RGB 画像)と depth 画像(深度画像)が含まれているデータセットを選んだ。このデータセットのうち、photo 画像と depth 画像をペアとしてそれぞれの画像を 256×256 程度にリサイズし、連結した画像を一万枚程度用意して深度画像推定を行うモデル[3]を作成する。このモデルを JavaScript にコンバートして AR アプリケーションに組み込み、モデルから生成された深度画像を三次元空間認識に用いることで隠れ処理を行うことができる。

4. 実験

4.1 モデルの生成

はじめに Pix2Pix を用いてモデルを作成した。この実験は以下の環境で行った。表 1 に示す。

表 1　環境

OS	Windows 10 Pro
CPU	Intel(R)Core(TM) i7-6850K CPU @ 3.60GHz
RAM	64.0GB
GPU	GeForceGTX1080 8GB

実験では GitHub にある affinelayer の pix2pix-tensorflow[4]リポジトリを使用した。データセットには 1 万枚の画像を使用し、およそ 30 時間かけて 100 回学習を行ってモデルを作成した。

Development of AR application using depth image synthesized by deep learning

†Akira Yoshida, Hitoshi Tamura, Nippon Institute of Technology faculty of innovative system engineering.

作成したモデルを使用して深度画像を生成した。(図２、図３)

図 2. 入力した画像(左)、生成した画像(中)、本来の深度画像(右)

図 3. 入力した画像(左)、生成した画像(中)、本来の深度画像(右)

実験の結果、図 2 のように本来の深度画像に近い画像を生成することができ、画像内の物体奥行きをある程度認識できることができた。しかし、色鮮やかな物体の距離がうまく表示できない部分や、一部の物体にちらつきができて生成できていない部分があった。

また、モデルをスマートフォンに実装するため、CPU のみのモデルのロード時間と画像一枚当たりの計算時間を計測した。結果を表 2 に示す。

表 2　計算時間

回数	ロード時間(s)	一枚当たりの時間(s)
1	15.96	0.1847
2	16.25	0.1830
3	15.94	0.1858
4	16.16	0.1835
5	15.94	0.1855
6	15.92	0.1844
7	16.08	0.1835
8	16.33	0.1827
9	16.24	0.1842
10	16.01	0.1851
平均	16.08	0.1842

実験の結果、この環境下では画像一枚当たりの計算速度は約 5fps だということが分かった。

4.2 モデルの WEB 上での推論

作成したモデルは TensorFlow.js のコンバータを用いることで TensorFlow のモデルを TensorFlow.js の Web フォーマットに変換することができる。今回の実験では TensorFlow の SavedModel でモデルデータを保存した。SavedModel では変数、グラフ、およびグラフのメタデータが保存される。しかし TensorFlow.js ではプロトコルバッファ形式から変換するため、一度プロトコルバッファ形式に変換する必要がある。しかし、機能の問題が多く今回はモデルデータをプロトコルバッファに変換することができなかった。そこで TensowFlow.js は keras モデルの変換もできるため、Pix2Pix を使用した本研究と似たような RGB 画像から depth 画像を生成するモデルのデータを GitHub の gautam678 の Pix2Depth[5]からダウンロードして Web フォーマットに変換してみたところうまく変換することができた。そこで、その変換したモデルデータを使い、実際に Web ブラウザで推論を行った。しかし、HTML 上での Tensorflow.js のロードに失敗して推論を行うことができなかった。

5. 考察

今回作成したモデルは PC 上で約 5fps であったことから、このままスマートフォンに実装するならスマートフォン上では 2fps を切ることが考えられる。実際にはどちらの CPU も使用用途が違うこと、モデルデータの変換をすることで処理が軽くなることを考えるとある程度の速度を確保できると考えられる。モデルがコンバートできなかったことは TensoFlow.js の変換機能をより詳しく調べ、モデルデータに問題があることも考え、改めてモデルの作成から行っていく必要があると考える。

6. おわりに

生成された深度画像は屋内での利用を考えれば十分な精度を実現できた。しかし、今回作成したモデルはコンバートできなかった。今後は TensorFlow.js のコンバート機能をよく調べ、モデルを Web 上で推論できるようコンバートして、深層学習モデルから生成された深度画像を AR の 3 次元空間の認識に用いることで隠れ処理を行う AR アプリケーションの開発を目指す。

参考文献

[1] 鈴木翼・太田高志(2011),「AR における前後関係を考慮した現実と CG の表示」,第 73 回全国大会講演論文集,2011 巻,1 号,pp. 293-294

[2] John McCormac and Ankur Handa and Stefan Leutenegger and Andrew J. Davison. 2016. SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth.
(https://robotvault.bitbucket.io/scenenet-rgbd.html)
(accessed 2018-1-10)

[3] 佐藤颯人ら,「機械学習による RGB 画像からの距離画像の生成」,第 80 回全国大会公園文集,2018 巻,1 号,pp. 229-230

[4](https://github.com/affinelayer/pix2pix-tensorflow)
(accessed 2018-1-10)

[5](https://github.com/gautam678/Pix2Depth)
(accessed 2018-1-10)