北海道科学大学研究紀要
第52号（2024年）
研究報告

Bulletin of Hokkaido University
of Science, No.52(2024)
Research Reports

３ＤＣＧを利用した物体検出

Object Detection Using 3DCG

小松 隆行*

Takayuki Komatsu

概要

本稿では，物体検出において使用される画像データを，新たに制作した 3DCG モデルを使って自動生成する．自動生成したデータを使い，基礎的な物体検出の学習と検証，およびテストに使用する．自動生成するデータはすべて異なるようにでき，かつその数を任意に指定することもでき，大量のデータを容易に生成できる．さらに，物体検出の学習で必要とされる手作業での画像データ毎のラベル付け（アノーテーション）を自動化するプログラムを開発し，手作業でのラベル付け作業を不要とし効率化を図った．物体検出の対象は，手話の母音のポーズ，スマートフォンを持った手のポーズである．これらのために，人間の手の 3DCG モデルを新たに制作した．このモデルは，ボーンを組み込んでポーズを変えることができ，手話で表現する各母音のポーズをとらせたり，スマートフォンの 3DCG モデルを持つポーズをとらせたりすることができ，それらのポーズから大量の画像データを自動生成する．物体検出に使用するライブラリは，広く使われている YOLOv5（You Only Look Once ver.5）[(1)]である．学習後の YOLOv5 モデルが，自動生成したデータ以外の実写の画像データに対しての汎化性能が獲得できているかどうかの検証も試みる．

1. はじめに

近年，3DCG 技術の発展は目覚ましく，様々な分野において広く活用されている．AI 技術の基盤を支える GPU も 3DCG などのグラフィック処理の分野から発展してきたものである．3DCG 技術の活用は，エンターテインメント系[(2)]のみならず，それ以外のエンタープライズ系の目的での活用も活発になってきている．国土交通省が推進する日本全国の都市の景観をデジタルツイン化するプロジェクト[(3)]や，自動車やドローンの自動運転技術の開発におけるシミュレータにおいても，3DCG 技術は非常に重要なものである．

一方で，AI 技術の進展は目覚ましく，人間の能力を凌駕する生成系 AI も開発され，社会に急速に広がっている．AI 技術の一つである物体検出技術の進歩もめざましく，多くの分野で実用化されている．物体検出モデルのための学習データは，検出対象を特定するラベル付けという手作業が必要だが，3DCG 技術でそれを自動化する研究や，学習データを自動生成する研究[(4)][(5)]が始まっている．

本報告では，カメラ画像から手話ポーズや ATM での振り込め詐欺防止のためのスマートフォンを持つ手を認識するために，物体検出ライブラリ YOLOv5[(1)]の学習，検証，テストにおいて，3DCG モデルからのデータ自動生成とその利用の基礎的な方法を提案し，その有効性を調べ検証することである．

2. 物体検出ライブラリ YOLOv5 と実行環境について

YOLOv5 は，物体検出の手法の一つであり，性能が高いことで知られている．画像データや動画データから，目的の対象物を矩形領域（画像内の相対座標）で特定し，それが何であるかをラベル（カテゴリー名）で示し，同時にその確信度を 0.0 から 1.0 の範囲の数値で示すものである．その数値が高いほど，確信度が高い．YOLO は，いくつかのヴァージョンがあるが，本報告では YOLOv5 を用いる．また，すべての処理は，MacBook Pro（14 インチ，2021），Apple M1 Max，メモリ 64 GB，MacOS Monterey で行った．プログラム開発は，Anaconda 環境の Jupyter Notebook で，Python でプログラムを開発し行った．

＊北海道科学大学未来デザイン学部メディアデザイン学科

3. 手話の母音ポーズの検出

まず，YOLOv5 での手話の母音のポーズの物体検出を試みる．手順は，①ボーンを組み込んだ右手の 3DCG モデルを制作，②ボーンを操作し母音を表現するポーズを作る（母音毎に 1 ポーズ），③モデルの垂直な中心軸を中心に回転するアニメーションを設定，④モデルを回転させアニメーションレンダリングで静止画像のシーケンスを生成，⑤開発したアプリで生成画像毎に YOLOv5 入力用のラベルファイル（txt 形式）を自動生成，⑥画像ファイルと対応するラベルファイルのペアを 1 個の入力データとし全データを学習用データと検証用データに分ける（約 8:2 の割合），⑦学習用データと検証用データを入力として YOLOv5 を学習させる，⑧学習結果（各種ログや検証データでの検出結果）を確認，⑨未知のデータ（学習と検証で未使用の生成データ，実写の画像データ）での検出テスト，である．

3.1 手話の学習データの生成

3.1.1 3DCG モデル制作とデータ生成

まず，手順①から④を説明する．制作し使用する 3DCG モデルは，著者の右手を参照し制作したものである．今回は，3DCG モデル制作と学習画像データ生成のために，DCC ツールとして Blender [6] を用いた．質感も，実際の色に似た色になるように HSV を設定した．図 1 に，手話の母音「あ」での例を示す．モデルにはボーン（アーマチュア）が組み込まれており，これを回転させて手のポーズを変えることができる．図 1 の状態から垂直軸方向の中心軸を中心に，左右±30 度の状態のキーフレームの間を，長さ 30 フレームで回転するモーションを設定し，アニメーションレンダリングして PNG 形式の静止画像シーケンスを出力させた．1 ポーズにつき，30 個が出力される．生成した画像データの抜粋を図 2 に示す．

図 1　Blender で表示した 3DCG モデル

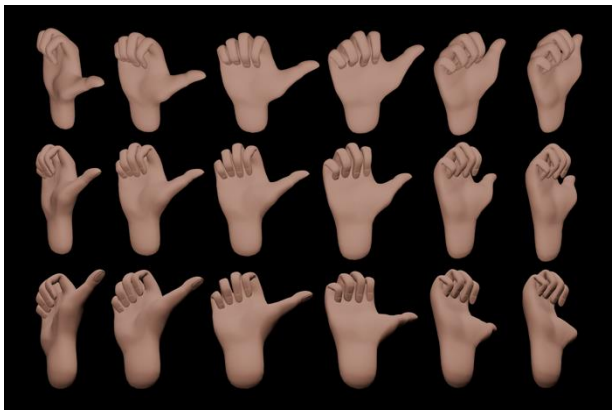

図 2　生成した静止画像データの例（抜粋）

3.1.2 YOLOv5 入力用ラベルデータの自動生成

次に，手順⑤を説明する．YOLOv5 の学習では，多数の静止画像データファイルとそれらに対応するラベルファイルの 2 種類が必要となる．ラベル付けはアノーテーションとも呼ばれ，一般的にはラベル付けのためのアプリを使って，手動で 1 枚ずつ作成してゆく．図 3 に示したような，ラベル付けソフト labelImg [7] などを使用する．作業は単純であるが，静止画像ファイルの数が莫大であるため，膨大な単純作業が必要だが，本報告ではアプリで自動化する．

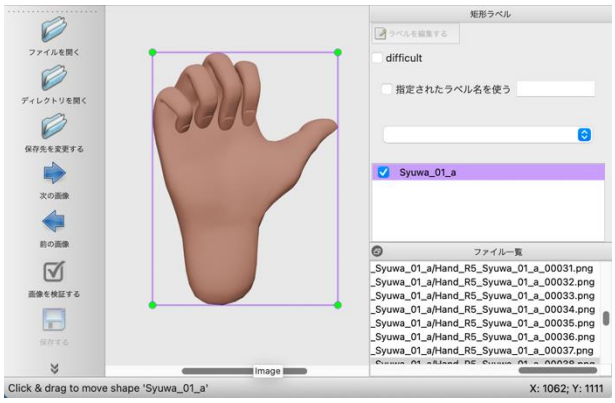

図 3 labelImg によるラベル付けの例

YOLOv5 への入力データは，静止画像データのファイルとこれらの各静止画像ファイルに対応するラベルデータのファイルのペアである．ラベルデータの内容の例を図 4 に示す．内容は，ラベル付されたカテゴリー番号，ラベル付された矩形域の中心の X 座標，その Y 座標，その X 軸方向領域幅，その Y 軸方向の高さである．データは，複数行あってもよい．本報告では，画像データ毎に一つのカテゴリーのデータのみなので，1 行のみのデータが格納される．カテゴリー番号は，整数値であり，画像内で設定されたすべての矩形域について，それぞれのカテゴリー番号が指定される．各座標値と幅と高さの値は，

画像の幅と高さのピクセル数で正規化されている．本報告では，新たに開発したPythonプログラムで，各画像データファイルに対応したラベルファイルを，すべて自動生成することで効率化を図る．Pythonアプリでは，ラベル生成のために画像処理ライブラリOpenCVを使用した．

Hand_R5_Syuwa_01_a_00030.txt

```
0 0.555729 0.481019 0.391667 0.828704
```

図4　YOLOv5のラベルファイルの例

1枚の静止画像データを入力として，そのラベルファイルを自動生成するイメージを図5に示す．黒背景の3DCGモデルにおいて，矩形領域を赤い線で囲むように検出し，そこから矩形域の中心（白い線の交点）を算出し，YOLOv5用のラベルファイルフォーマットに合わせて，対応するラベルファイルをテキスト形式で出力する．今回は，「あ」，「い」，「う」，「え」，「お」の各母音のポーズに，正面からのカメラの上下方向の角度を3パターン設定し，その角度パターン毎に30枚の画像を生成させた．したがって，画像データとして母音毎に90枚ずつ，計450枚生成した．よって，ラベルデータファイルも450個自動生成した．これらは全て，labelImgで開き，適切にラベル付けされているかどうかを確認した．

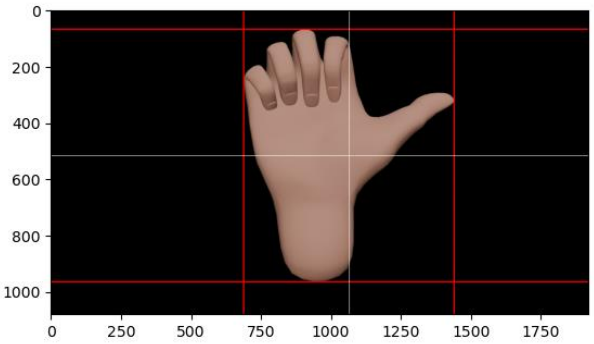

図5　ラベルデータの自動生成のイメージ

3.2 自動生成データを使ったYOLOv5の学習と検証

ここでは，手順⑥から⑧を説明する．3.1節での準備作業で，静止画像ファイルとラベルデータファイルがすべて用意できた後に，これらのデータを学習用データと検証用のデータに分ける．分割比は，データ数の比で約8:2である．データサイエンスにおける機械学習では，学習後のモデルが未知のデータに対してどれだけ性能が良いかを問われる．すなわち，汎化性能の高さがモデルの性能の高さと言われている．学習に使用できるデータ数には限りがあるため，それらを学習のみに使用するデータと，未知のデータの意味で性能を検証する検証データに分け学習アルゴリズムで学習させ，未知のデータに対して最良のモデル（のパラメータ値）を学習済みモデルとして使用する．今回は，画像ファイルとラベルファイルがペアで用意されているので，そのペアを学習用データと検証用データに分けて取り扱う．学習は，前述のMacintoshで，バッチサイズ5，エポック数300で学習を行った．図6に学習のログ（終了部分の抜粋）を示す．また，機械学習の学習ログ分析ライブラリであるTensorBoardでの学習と検証の結果のグラフ類を図7から図15に示す．

```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
298/299         0G   0.005569   0.003187   0.004974         10        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         90         90      0.997          1      0.995      0.995

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
299/299         0G   0.005357   0.003189   0.006144         10        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         90         90      0.997          1      0.995      0.995

300 epochs completed in 6.026 hours.
Optimizer stripped from runs/train/exp26/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp26/weights/best.pt, 14.4MB

Validating runs/train/exp26/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50
                 all         90         90      0.997          1      0.995      0.995
          Syuwa_01_a         90         18      0.997          1      0.995      0.995
          Syuwa_02_i         90         18      0.997          1      0.995      0.995
          Syuwa_03_u         90         18      0.997          1      0.995      0.995
          Syuwa_04_e         90         18      0.997          1      0.995      0.995
          Syuwa_05_o         90         18      0.998          1      0.995      0.995
```

図6　YOLOv5の学習ログ：学習終了時の例

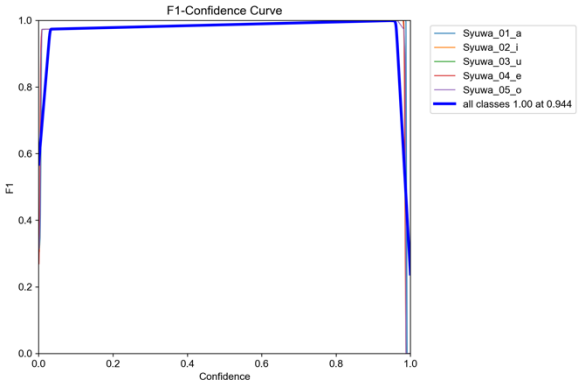

図7　F1値の曲線

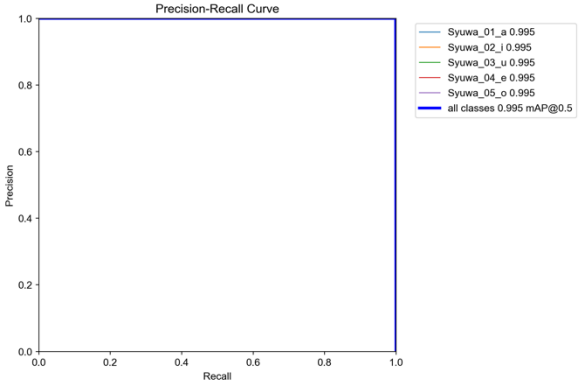

図8　PR曲線（適合率-再現率の曲線）

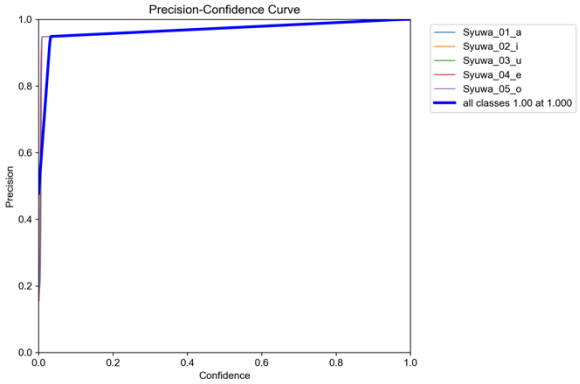

図 9　適合率(Precision) 曲線

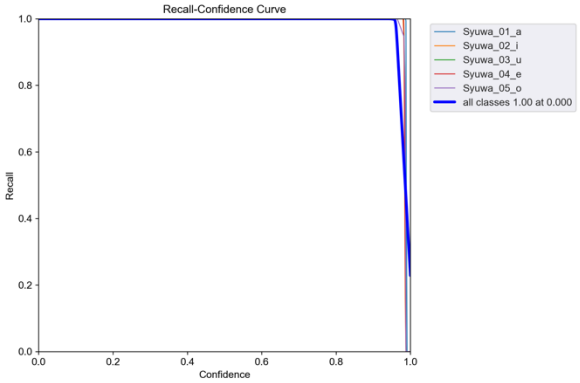

図 10　再現率(Recall)曲線

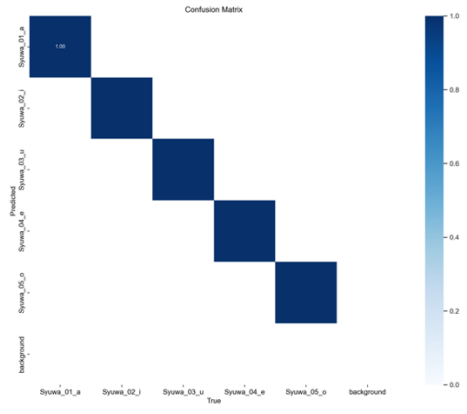

図 11　混同行列

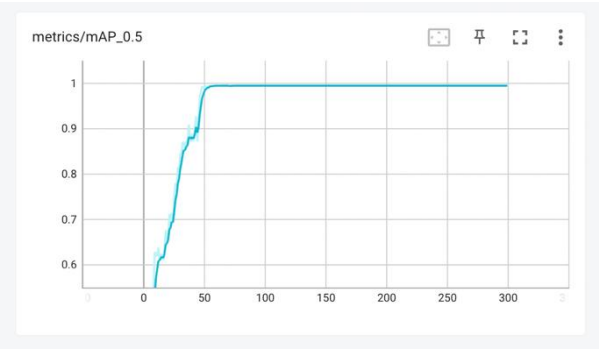

図 12　mAP0.5 の変化曲線

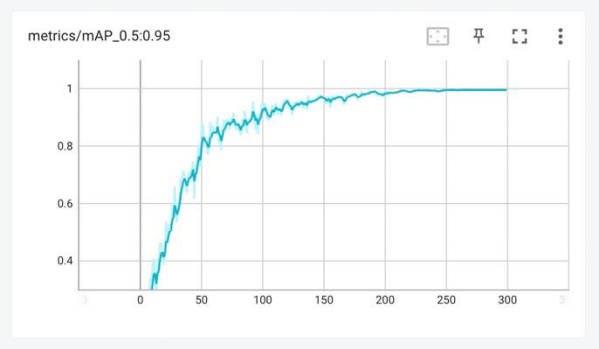

図 13　mAP0.5:0.95 の変化曲線

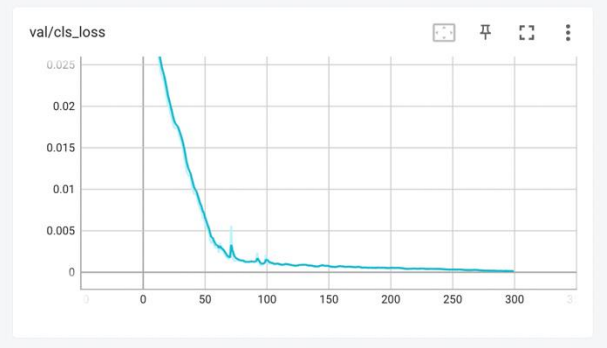

図 14　検証でのクラス損失曲線

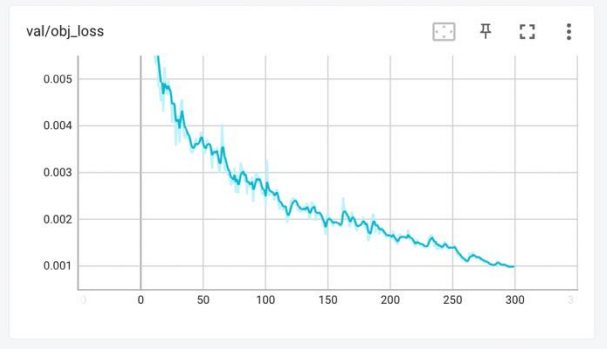

図 15　検証でのオブジェクトネス損失曲線

これらが示すように，非常に良い学習結果となっている．物体検出の評価指標の IoU の閾値が 0.50 で計算された平均精度 mAP0.5 の変化（図 12），IoU の閾値を 0.5 から 0.95 まで 0.05 きざみで mAP を求め，それらを平均した mAP0.5:0.95 の変化（図 13）では，少しばらつきはあるものの値が安定して増加し，エポック数がそれぞれ 50，200 くらいで収束していることが分かる．

検証用の画像データでの検証結果を，図 16（正面からの画像データ）と図 17（少し斜め方向からの画像データ）に示す．全母音の画像で，良い検出結果となっている．学習データは，ファイル 1 枚に一つのポーズであったが，複数カテゴリーのポーズをコラージュした 1 枚の画像にして検証している．

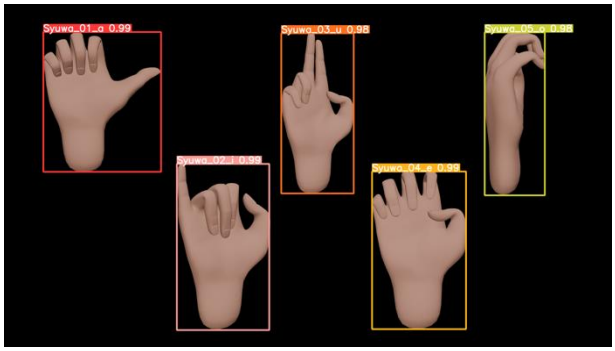

図 16　検証データでの検出結果の例

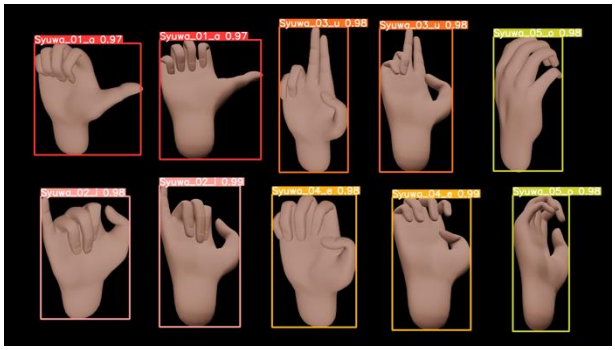

図 17　検証データでの検出結果の例

3.3 未知のデータを使ったテスト

最後に手順⑨を説明する．汎化性能の確認として，学習にも検証にも使用していない実写の画像を使ってテストを行ってみる．図 18 は，著者が右手で各母音のポーズを作り，それを撮影した実写画像をコラージュして 1 枚にし，前節での学習済み YOLOv5 モデルへ入力した検出結果の一例である．「あ」「い」「う」「え」は高い確信度で検出されているが，「お」は確信度がやや低く，他の認識候補の表示も出ている．左下ポーズは，「え」のつもりで作ったものであるが，ポーズの作り方が正しいポーズとは少し異なっていたため，「あ」のポーズと判断されている．この結果を見て改めて作ったポーズが下段中央のものだが，これは正常に「え」と判断されている．筆者は，初めて手話ポーズを試すような手話の素人であり慣れていないが，5 個のポーズが正確に検出されていることが分かる．このテストは，背景も写っている画像で試したが，矩形域さえも検出ができなかった．そのため，学習データと同じように，黒背景にした画像にしてテストに使用したところ，うまく検出できた．なお，これらの画像は，自然光で撮影され，手の部分を選択して切り抜いた以外の画像処理，例えば HSV の加工処理やフィルターによる加工処理はいっさい行っていない．

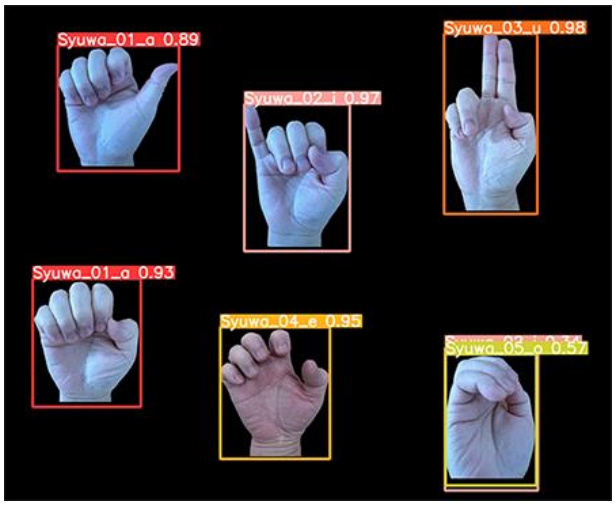

図 18　実写画像（未知のデータ）のテスト結果例

4. スマートフォンを持つ手の検出

3 章と同様に，スマートフォンを持つ手の検出を試みる．同じ手の 3DCG モデルを使い，スマートフォンを持つ典型的なポーズを作る．簡易なスマートフォンの 3DCG モデルも作成し，持ったポーズを使い YOLOv5 用のデータを同様に自動生成し，学習，検証，テストを試みる．

4.1 スマートフォンを持つ手の学習データの生成

4.1.1 3DCG モデル制作とデータ生成

まず手順①から④を説明する．新たに，簡易的なスマートフォンの 3DCG モデルを作る．スマートフォンの質感は，一般的なグレイ色にした．付属品は，背面の簡易なカメラレンズパーツと，単純な直方体による周囲の簡易な電源ボタンや音量調整ボタンパーツのみである．画面の質感は，背面と色味が少し異なるグレイ色である．スマートフォンを持つ手のポーズを図 19 のように作った．スマートフォンの上下方向の角度を変えたポーズ毎に，同様に回転モーションをさせ，アニメーションレンダリングで，PNG 形式の静止画像シーケンスを出力させる．

図 19　Blender で表示した 3DCG モデル

図 20 に書き出された画像シーケンスの一部を示す．垂直軸方向の中心軸を中心に，0 度から 360 度の状態のキーフレームの間を，長さ 30 フレームで回転するモーションを設定しアニメーションレンダリングして，PNG 形式の静止画像シーケンスを 1 ポーズにつき 30 個出力する．スマートフォンの方向を変えた基本の 5 ポーズ毎に 30 枚ずつ，スマートフォンを持つ手，何も持たない手のみ，スマートフォンのみの画像をそれぞれ 150 枚ずつ，合計 450 枚の画像を自動生成した．

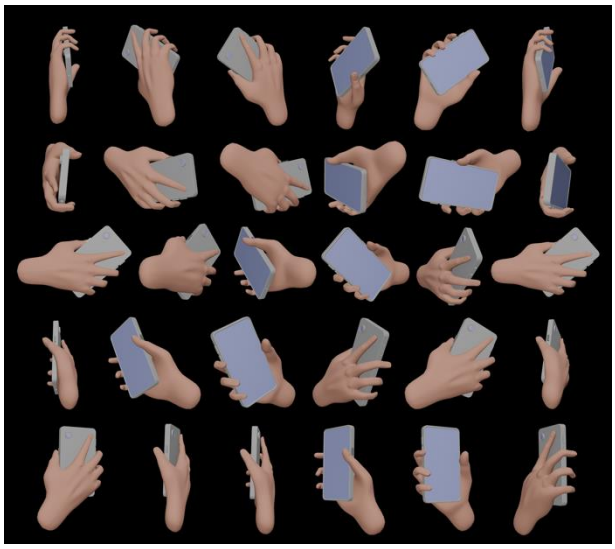

図 20　生成した静止画像データの例（抜粋）

4.1.2 YOLOv5 入力用ラベルデータの自動生成

次に，手順⑤を行う．前節で生成した 450 枚の画像データ毎にラベルデータファイルを前章と同様に自動生成した．ラベルデータも 450 枚になる．図 21 は，Python での自動ラベル付けのイメージである．これら全てのデータは，labelImg で開き，適切にラベル付けされているかどうかを確認した．

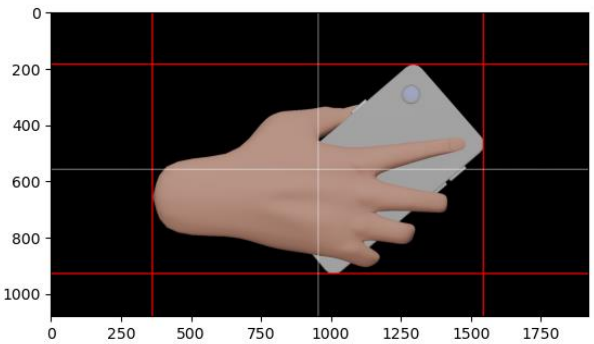

図 21　Python での入力ラベル自動生成のイメージ

4.2 自動生成データを使った YOLOv5 の学習と検証

ここでは，手順⑥から⑧の学習と検証を行う．バッチサイズ 5，エポック数 300 で行った．この学習のログ（終了部分の抜粋）を図 22 に示す．また，TensorBoard での学習と検証の結果のグラフ類を図 23 から図 31 に示す．良好な学習結果となっている．mAP0.5 の変化（図 28），mAP0.5:0.95 の変化（図 29）は，学習初期に値のばらつきが目立っているが，エポック数がそれぞれ 150，200 くらいで収束している．オブジェクトネス損失の減少（図 31）が，学習終了時に十分に収束していないので，学習回数を増加させることも必要であると考えられる．

```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
297/299          0G   0.006757    0.00431   0.001204         11        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         75         75      0.997          1      0.995      0.993

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
298/299          0G   0.006585   0.004207   0.001769          6        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         75         75      0.997          1      0.995      0.993

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
299/299          0G   0.006711   0.004751   0.001963         12        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         75         75      0.997          1      0.995      0.993

300 epochs completed in 6.344 hours.
Optimizer stripped from runs/train/exp33/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp33/weights/best.pt, 14.4MB

Validating runs/train/exp33/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50
                 all         75         75      0.997          1      0.995      0.994
       00_Hand_Phone         75         25      0.995          1      0.995      0.995
             01_Hand         75         25      0.996          1      0.995      0.995
            02_Phone         75         25      0.998          1      0.995      0.992
Results saved to runs/train/exp33
```

図 22　YOLOv5 の学習ログ：学習終了時の例

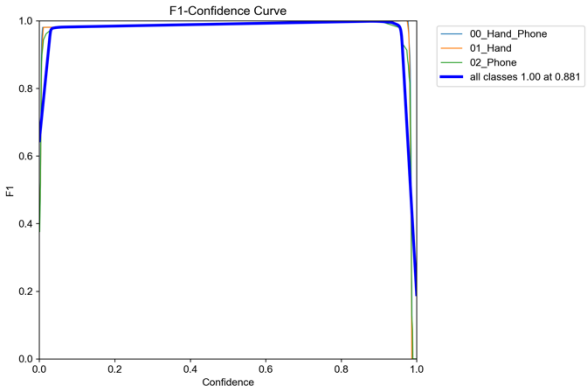

図 23　F1 値曲線

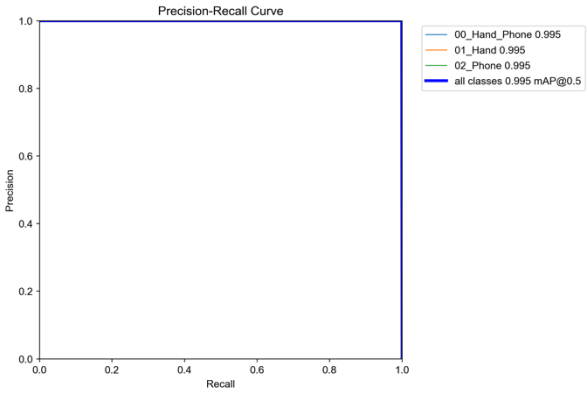

図 24　PR 曲線（適合率-再現率の曲線）

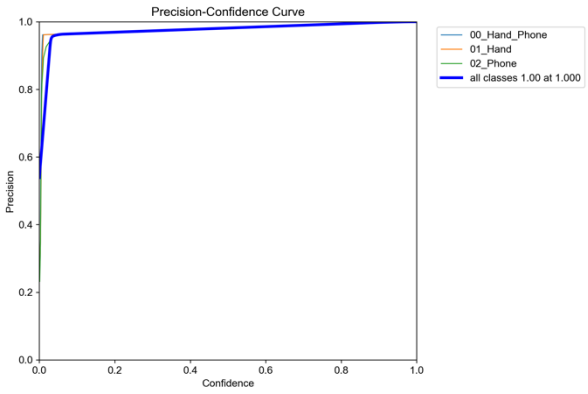

図 25　適合率(Precision) 曲線

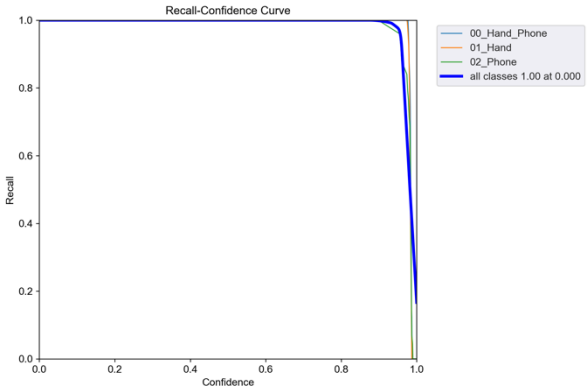

図 26　再現率(Recall)曲線

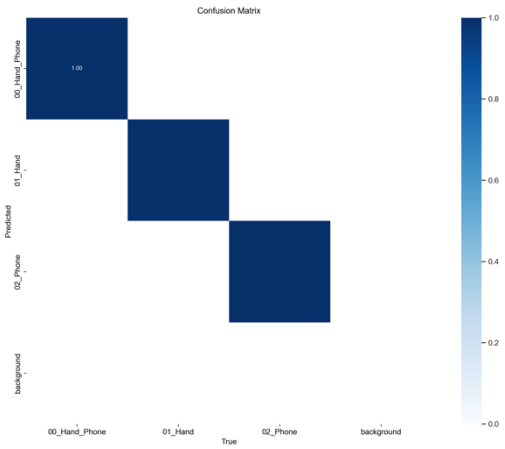

図 27　混同行列

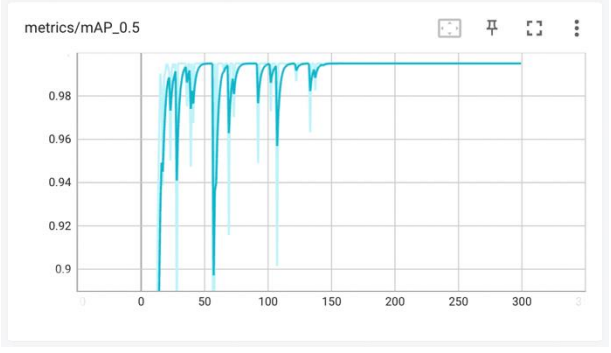

図 28　mAP0.5 の変化曲線

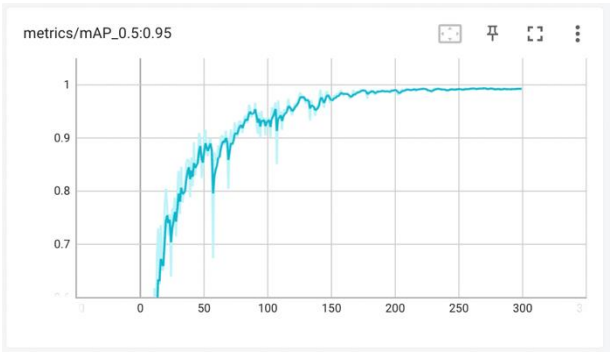

図 29　mAP0.5:0.95 の変化曲線

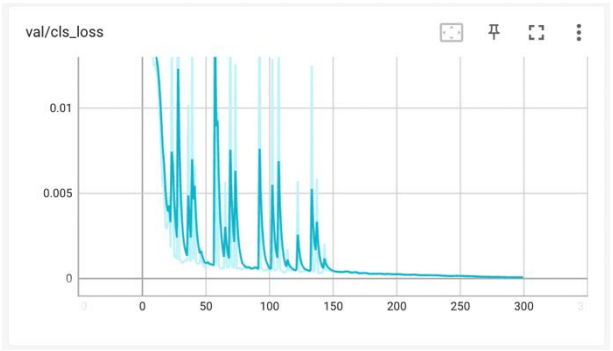

図 30　検証でのクラス損失曲線

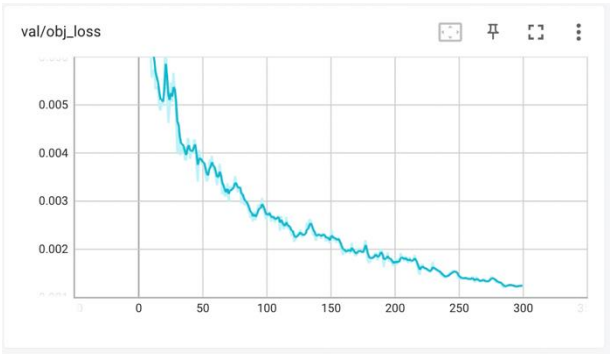

図 31　検証でのオブジェクトネス損失曲線

検証データの検出結果例を図 32 に示す．3 つのカテゴリーとも良好な検出結果であることが分かる．

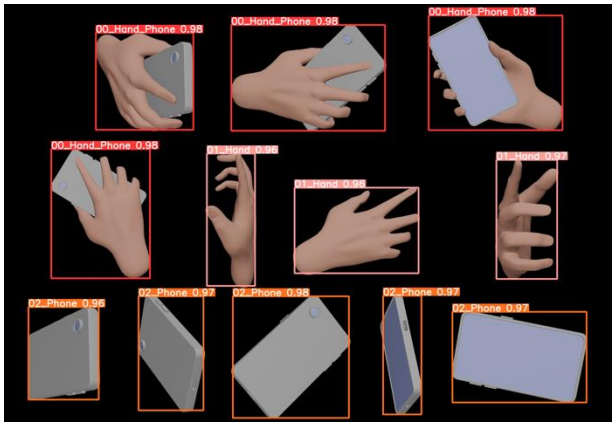

図 32　検証データでの検出結果の例

4.3 未知のデータを使ったテスト

最後に手順⑨を説明する．汎化性能の確認として，学習にも検証にも使用していない，実写の画像を使ってテストを行ってみる．図 33 は，自作したスマートフォンの 3DCG モデルとはカメラ形状が異なるものを実際に持ち，著者がとったポーズを自ら撮影した画像データを，前節の学習済みの YOLOv5 モデルで検出した結果である．スマートフォンを持つ手を良好に検出しており，汎化性能を獲得していると考えられる．特に，上段最右端の画像データは，モデルにはなかった左手で持ったポーズの実写画像であるが，高い確信度で検出されている．最下段の画像のようにスマートフォンのみの検出も良好である．しかし，2 個のスマートフォンが重なっている最右端の画像データは，検出はできているが，一つの検出領域内に 2 個が入っているという点では正しくはない．中段の手だけの画像データは，誤認識されてしまっている．学習データを自動生成する際に，手だけの画像はスマートフォンの 3DCG モデルを非表示にしただけの画像であったため，手の部分の画像は「スマートフォンを持つ手」のラベルで学習した画像とほぼ同一ということが影響したのかもしれない．また，ここでも背景を黒色で塗りつぶさない画像では，前章同様に検出がされなかった．そのため，黒背景の画像を作ってテストした．

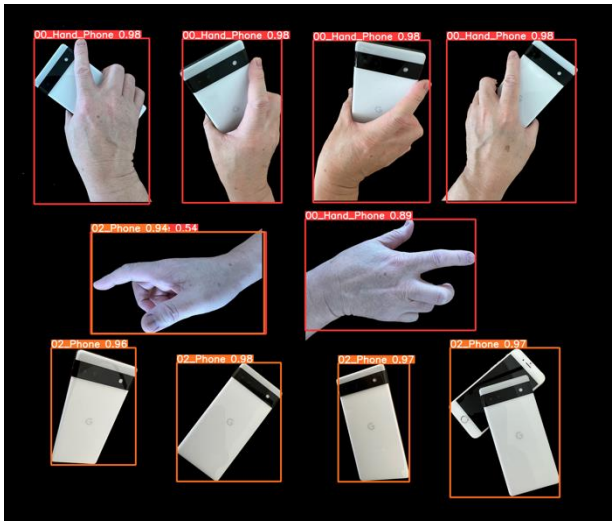

図 33　実写画像（未知のデータ）のテスト結果例

5. まとめ

3DCG モデルを使った YOLOv5 用のデータの自動生成と，それらを使った物体検出のための学習，検証，テストを行ない，その有効性を確認できた．実写画像を使った検出も確認し，十分でないが汎化性能の獲得も確認できた．しかし，まだ十分な学習，検証，テストができていないことや，分類カテゴリー（手話のポーズの種類，スマートフォンの種類やその持ち方の種類など）の数が少ないことから，それらを含めて多種多様の大量のデータを用意したシミュレーションが今後は必要と考えられる．黒背景での画像ではなく，多種多様な背景での検出ができるように，自動生成画像を用意して学習させ検証することも必要である．学習における適切なバッチサイズの検討も必要であろう．YOLOv5 より新しいヴァージョンの YOLO を用いた検証と比較も課題であると考えられる．

参考文献

(1) Github ultralytics/YOLOv5, 2024 年 3 月 3 日, https://github.com/ultralytics/YOLOv5.
(2) Unreal Engine, 2024 年 3 月 3 日, https://www.unrealengine.com/ja/.
(3) PLATEAU, 2024 年 3 月 3 日, https://www.mlit.go.jp/plateau/.
(4) H. Kubo, K. Kobayashi and Y. Aoyagi, "Automatic Annotated Farm Image Generation from 3D Computer Graphics for Machine Learning," 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS), Ise, Japan, 2022.
(5) Rasmussen, Ingeborg, et al. "Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine." Applied Sciences 12.17 (2022): 8534.
(6) Blender, 2024 年 3 月 3 日, https://www.blender.org/.
(7) Github HumanSignal/labelImg, 2024 年 3 月 3 日, https://github.com/HumanSignal/labelImg.