AI を用いた物体の位置と状態を特定する手法の検討

Investigation of method for specifying position and state of objects using AI technology

技術開発部 生産・加工科　鈴木健司　近野裕太　柿崎正貴　清野若菜　山田昌幸

応募企業　株式会社ランプハウス

製品検査等に応用するため、ライン上を移動する物体を想定し、物体認識と位置特定を行うための手法を検討した。AI 技術のひとつである YOLO を用いて物体認識と状態識別を行うことができた。

Key words: AI、ディープラーニング、YOLO、物体検出

１．緒言

応募企業の株式会社ランプハウスは、生産設備における自動化・省力化機器や各種検査機器の設計・製作を行っている。生産工程において、ある作業の自動化や検査を行うためには、製品の識別と状態を把握する必要がある。これまでは、画像処理やパターン認識など一定のアルゴリズムにしたがって物体の認識や状態の把握を行うことが多かった。しかし、近年では多品種少量生産の生産工程が増えてきており、品種によってアルゴリズムのパラメータを変えて対応することは、検証などにも時間がかかり困難である。

そこで、本研究では、リアルタイム物体検出システムである YOLO[1-2]を用いて物体認識と状態識別を行う手法を検討した。さらに、認識した物体の位置を特定するためのプログラムを作成した。これにより、学習データを集めるだけで、プログラムを変更せずに多品種に対応できる可能性を示すことができた。

２．実験と結果

手法の検討を行うため、図１に示す人形を使って学習及び学習結果の評価を行った。人形は座った状態（sitdown）と立った状態（standup）の２つの状態とした。これにより、２つの状態を別物として学習させることで、物体の認識と状態の識別が同時に可能となる。学習には表１に示す計算機を用いた。

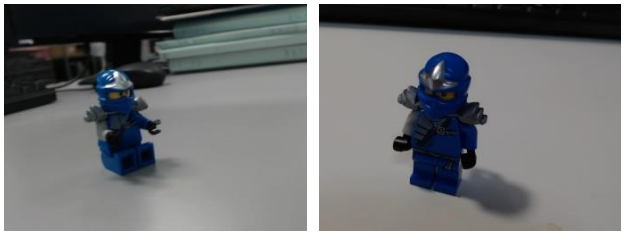

図１　学習対象の人形の sitdown 状態（左図）と standup 状態（右図）

表１　学習に用いた計算機

OS	Ubuntu16.04LTS
メモリ	768 [GB]
CPU	Intel Xeon E5-2699 v4 x2
GPU	NVIDIA Tesla V100 x8
GPGPU プラットフォーム	CUDA V9.0
学習モデル	YOLO-v3
実装	Darknet

２．１．YOLO を用いた学習

YOLO は、リアルタイム物体検出システムであり、Darknet[3]や Keras/TensorFlow、PyTorch など様々なディープラーニングフレームワーク上で実装されたものが提供されている。

本研究では、Darknet を github サイト[4]からダウンロードして学習に利用した。Darknet を用いて学習を行うため、以下のコマンドによりソースをダウンロードし、コンパイルした。

```
$ git  clone https://github.com/pjreddie/dark ¥
  net.git
$ cd darknet
$ make
```

GPU や cuDNN を利用して学習を高速化するため、また様々な画像を処理できるようにするため、Makefile 中のパラメータは GPU = 1、CUDNN = 1、OPENCV = 1 に設定した。この後に、最後の make コマンドを実行すると実行ファイルが作成される。また、学習の実行には以下のファイルを設定する必要がある。

```
datasets.data
train.txt
test.txt
```

事業課題名「AI を用いた物体の位置と状態を特定するための手法の検討」
[令和元年度][新製品・新技術開発促進事業]

class.txt
yolov3-voc.cfg
train.txt と test.txt にはそれぞれ、学習用データとテスト用データのファイルパスが記述されている。datasets.data では、分類するカテゴリ数、train.txt と test.txt、class.txt のファイルパス、学習済みモデルの保存場所などを指定する。yolov3-voc.cfg には、ニューラルネットワークの構造や各種パラメータが定義されており、入力データのサイズに応じて width と height の値を、式（１）に従い filters の値をそれぞれ変更させる必要がある。

$$filters = mask\_num * (classes + 5) \quad (1)$$

mask_num は標準のものから変更しないので 3 を、classes は分類するカテゴリ数なので、今回は sitdown と standup の２つの状態であるので 2 を入力する。したがって、filters = 21 となる。また、学習データのサイズは今回、幅 640、高さ 480 ピクセルとなるので、width = 640、height = 480 とした。学習は以下のコマンドで実行した。

```
$ ./darknet detector train datasets.data ¥
  yolov3-voc.cfg
```

２．２．学習データの準備

学習データは、Web カメラに写っている映像を一定時間ごとに画像に保存する Python プログラムにより収集した。画像は JPEG 形式で、画像サイズは幅 640、高さ 480 ピクセルに設定した。画像枚数は、sitdown と standup のそれぞれについて約 100 枚ずつ準備した。

次に、得られた画像中でどの領域が認識するべき物体であるかを矩形の領域（Bounding Box）で学習モデルに示す作業（Annotation）が必要になる。Annotation には、BBox-Label-Tool[5]を使用した。BBox-Label-Tool は Annotation を GUI 操作によって実施できる補助ツールであり、その作業は図２に示すように、画像中の対象物を矩形で囲う作業になる。これにより、画像と同じファイル名で、Bounding Box の座標値を示すテキストファイルが保存され、学習する対象が画像中のどこにあるかを示すことができる。しかし、この座標値は、Bounding Box の左上と右下の座標値の組になっているが、YOLO で扱う形式は、Bounding Box の中心座標と幅、高さになるので、変換が必要である。変換は、Python によるプログラムで行った。

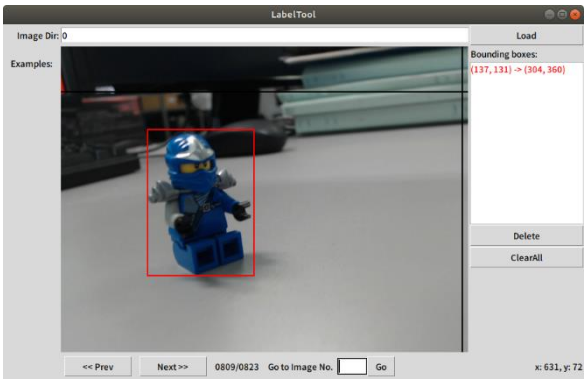

図２　Annotation 作業の様子

２．３．YOLO による物体認識と状態識別の結果

学習が終了すると yolov3-voc_40000.weights のファイルが作成される。これは、学習済みのモデルで以下のコマンドにより、img.jpg の画像に対して、物体認識を実行することができる。

```
$ ./darknet detector test datasets.data ¥
  yolov3-voc.cfg yolov3-voc_40000.weights ¥
  img.jpg
```

図３は、学習に使用していない画像による識別結果である。sitdown と standup の２つの状態について正しく識別されていることが確認できた。さらに、未学習の画像 20 枚を識別したところ、すべての状態を正しく識別できた。

また、学習済みモデルを用いて、Web カメラの映像をリアルタイムで物体認識することも可能である。実行コマンドは以下のとおりである。

```
$ ./darknet detector demo datasets.data ¥
  yolov3-voc.cfg yolov3-voc_40000.weights
```

上記コマンドで、Web カメラの映像をリアルタイムで物体認識した様子を図４に示す。角度によっては正しく認識しないこともあるが、おおよそ正しく判断することができた。しかし、背景が変わったり、ある光の当たり方になったりすると全く正しく識別しなくなることがあった。そのため、実際に利用する際は、使用場所や明るさの環境に応じた学習をする必要があると考えられる。

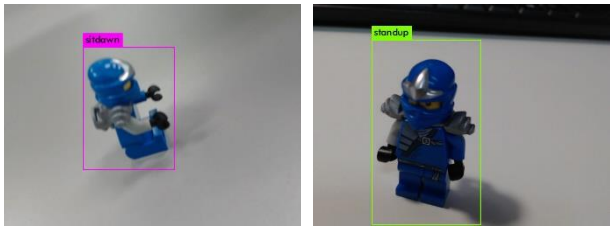

図３　未学習データによる識別結果

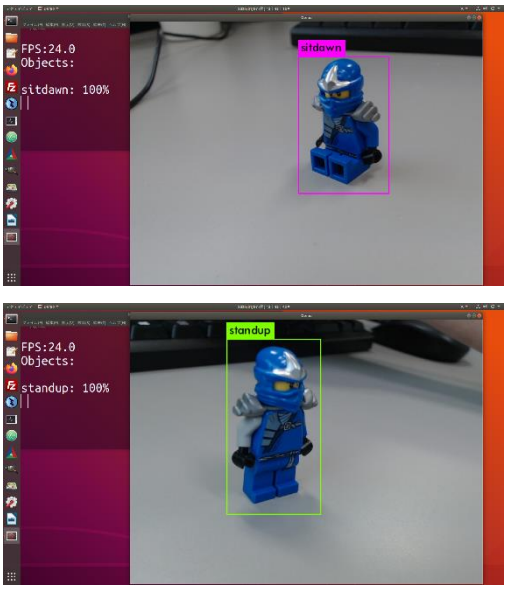

図４　Web カメラによるリアルタイム物体認識

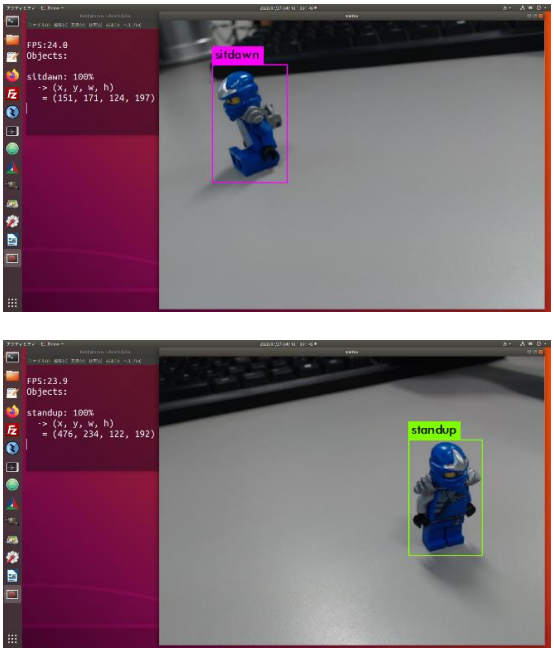

図５　Bounding Box の座標値出力の結果

２．４．位置特定のためのプログラム

リアルタイムの物体認識の結果を使って、認識対象に何かしらの動作をする場合には、その位置を特定しなければならない。darknet の実行ファイルの現状では、Bounding Box の座標値は表示されないので、ソースコードの修正が必要になる。darknet の実行ファイルがある場所の src ディレクトリの中に、image.c のファイル名でソースコードがある。その中に draw_detections 関数が記述されているので、その中の print 文の下に以下のコードを挿入する。

```
printf(" -> (x, y, w, h)¥n");
printf(" = (%d, %d, %d, %d)¥n",
      (int)(dets[i].bbox.x*im.w),
      (int)(dets[i].bbox.y*im.h),
      (int)(dets[i].bbox.w*im.w),
      (int)(dets[i].bbox.h*im.h));
```

座標値は Bounding Box の中心の座標値(x, y)、幅 w、高さ h で示される。修正後、darknet の実行ファイルがあるディレクトリで make コマンドを実行し、リビルドを行うと Bounding Box の座標値を得る実行ファイルが作成される。図５に Web カメラのリアルタイム物体認識の結果を示す。ターミナル画面に、Bounding Box の座標値が表示されていることがわかる。

３．結言

本研究では、可動式の人形を使って、２つの異なる状態を作り、それらを学習対象とし、YOLO による物体認識を行った。これにより、同一の物体であっても別のものとして認識することで、異なる状態を識別することができることを確認できた。

また、学習済みモデルを用いて、認識した物体の座標値を出力するプログラムを用いることで、物体の画面上での位置を特定することができた。

これらの結果より、状態の異なる物体を識別し、さらには、画面上での識別対象の位置を特定することで、生産工程における自動化への応用の可能性を示すことができた。

一方、背景や明るさの違いなどで全く識別できなかったこともあったので、使用環境に合わせて学習を調整することが今後の課題である。また、識別対象の画像上の座標値と実座標値をどのように対応させるかも検討していく必要がある。

参考文献

１）Joseph Redmon. “YOLO: Real-Time Object Detection”. https://pjreddie.com/darknet/yolo/. (参照 2020-01-31).

２）J. Redmon, S. Divvala, R. Girshick and A. Farhadi. "You only look once: Unified, real-time object detection." arXiv preprint arXiv:1506.02640,2015.

３）Joseph Redmon. " Darknet: Open Source Neural Networks in C”. https://pjreddie.com/darknet/ .(参照 2020-01-31).

４）GitHub. “pjreddie/darknet”.　Convolutional Neural Networks. https://github.com/pjreddie/darknet. (参照 2020-01-31).

5）GitHub.“puzzledqs/BBox-Label-Tool”.
https://github.com/puzzledqs/BBox-Label-Tool.
(参照 2020-01-31).