1　課題内容の概要

このプロジェクト課題において我々 9 班は、「マイコンを用いたスモールスケールでの自動運転」という題目で課題に取り組んだ。概要は以下の通りである。
走行可能領域を示す 2 本の白線と道路標識を設置したサーキットの上を、マイコンとカメラを搭載した電動 RC カーに走らせる。このとき車は自動でカーブや標識を認識して、車輪に対し両輪停止や速度差付与といった適切な制御を加える。

2　担当部分の説明と考察

2.1　概要

標識認識を行うにあたって以下のような手順を踏んだ。

1. 標識が存在する画像の上半分のみを切り取る。その後さらに画像を横に三分割し、標識が存在する可能性が高い左端から標識の探索を開始する。
2. 標識が存在すると思われる候補領域を抜き出す。(Region Proposal)
3. 抜き出した各々の領域を 64 × 64 にリサイズした後に識別機 (VGG19) を施して、設定した閾値を超えなかった場合は標識なしと判定する。

2 と 3 についてそれぞれ説明した後でまとめて考察する。なおページの都合上、実際のプログラムは掲載しない。

2.2　Region Proposal

厳密な定義はないが、基本的には SuperPixel や Segmentation[*1] などの手法を用いて画像を一定の基準の下で小領域に区分けする。その後、小領域同士の結合やノイズの除去などを行って候補領域を作成する。
今回は既に存在するライブラリ[*2] を使用して Region Proposal を行った。
このライブラリでは、まず Felzenszwalb’s efficient graph based segmentation[*3]によって画像を小領域に分割し、その後 LocalBinaryPatterns [*4]や RGB 情報などの特徴量をもとに領域同士を結合している。

[*1] https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_segmentations.html#sphx-glr-auto-examples-segmentation-plot-segmentations-py
[*2] https://github.com/AlpacaDB/selectivesearch
Copyright (c) 2015-2016 AlpacaDB
Copyright (c) 2016 Oussama ENNAFII
[*3] http://cs.brown.edu/people/pfelzens/papers/seg-ijcv.pdf
[*4] https://en.wikipedia.org/wiki/Local_binary_patterns

2.3 VGG19

VGG19[*5]は CNN[*6]を用いたニューラルネットワークのアーキテクチャの一つである。
本来は 1000 クラス分類に使用されるが、以下の 3 クラスを分類できるようにした。
学習に使用したデータは GTSRB[*7]である。それぞれの画像のデータ数を均一にするためにデータ拡張を行い、一つの種類につき 2400 データずつ用意した。

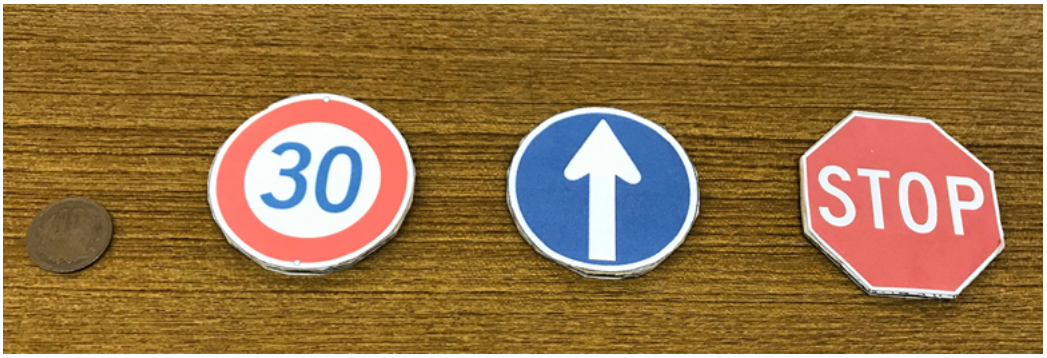

図 1　今回使用した速度変更、走行開始、走行停止の 3 種類の標識

[*5] https://arxiv.org/pdf/1409.1556.pdf
[*6] https://en.wikipedia.org/wiki/Convolutional_neural_network
[*7] https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

2.4　考察

今回作成したプログラムと物体検出に広く用いられる SSD[*8]を比較する。

図 2　実験に使用した画像の例

そして上記のような三種類のうち一種類の標識のみが映った 300 個の学習データとテストデータ 100 枚に対して実験を行い、**IoU** と **F 値**と**一枚当たりの処理時間**を測る。
IoU は Region Proposal の精度を測るため、F 値は識別機の精度を測るために測定する。
なお、IoU は一つの検出対象に対して最も正解確率が高いバウンディングボックスを選択して計算する。そもそも検出対象をバウンディングボックスで囲まなかった場合や、対象に対して誤ったバウンディングボックスしか存在しない場合は計算しない。
また、F 値は確率が 7 割を超える物体を対象にして計算する。

表 1　自作プログラムと SSD の比較

	IoU	F 値	処理時間
自作	0.49	0.72	0.31
SSD	0.83	0.78	0.08

IoU に対して言及する。
自作したものの IoU が SSD の 60 %ほどの性能になっている。結果を見たところ、最も正解率が高いバウンディングボックスはアノテーションで作成したバウンディングボックスよりも大きくなる傾向があった。そのため IoU が小さくなってしまったと思われる。
F 値に対して言及する。
IoU と比べると SSD にやや近い値を取ったように見えるが、True Positive(標識を検出) や False Positive(誤った領域を検出) や False Negative(標識があるのに検出しない) の値を確認してみたところ、自作したものは FN の場合が多く、SSD は FP が多かった。

[*8] https://arxiv.org/pdf/1512.02325.pdf

処理時間に対して言及する。
SSD の方が自作したものより性能が良い。また、自作したものは画像によって処理時間にかなりの差があったが、SSD は差はあまり無かった。
自作したもので採用している Region Proposal のアルゴリズムは画像に存在する色相、彩度、明度の複雑さによって小領域への切り分け方や領域の結合にかかる時間が変わる一方、SSD は end-to-end なニューラルネットワークなので処理時間にほとんど差が出なかったのではないかと思われる。

3　担当部分についてのまとめ

担当部分について総括する。

- 実装したもの
  自動運転における標識認識プログラム。
- プログラムの詳細
  Region Proposal による候補領域検出と VGG19 による分類。
- プログラムに対する考察
  SSD と比較すると、SSD の方が性能が良い。

今回、標識検出のシステムを作成するにあたって既存の物体検出ライブラリ (YOLO、SSD など) を使用せず、上述してきたような手順を取った。それは汎用性のある標識検出システム作成しようと考えたときに、大量の学習データとデータの作成に対する時間が必要だが、このプロジェクトではそれが不可能だと判断したからである。
考察の項にある通り、完成したものと既存のライブラリを比較した結果、後者の方が性能が良かった。しかし自作したものはライブラリとは違い、事前学習のために大量のデータと加工を必要とせず、公的に配布されているデータ使用して識別機を訓練するだけで汎用性とある程度の精度を持つ物体検出システムが構築できる。その点に関して、今回作成したプログラムには利点があると思われる。