第8章 検出結果の視覚化と表示/白線の補間/レーン逸脱防止警告の表示など

実践6：動画像に対するセグメンテーション処理の実施

土井 伸洋

最後に取り組むのはアプリケーションの構築です．第3章以降の取り組みを通じて，走行画像から白線領域とその種類を抽出する学習済みモデルが完成しました．しかし，ただ推論を行うだけで「この結果を利用して何かに役立てる」という肝心の部分が抜けています．

そこで本章では以下の内容に取り組みます．

- 動画への適用
- 表示の工夫

さらに，次章以降では，

- エッジ・デバイスへのマッピング（Jetson，ラズベリー・パイ4）
- 処理の高速化

を行います．

ドライブ・レコーダの動画へ処理結果を重ねる

● 動画像のリアルタイム処理を目指す

アプリケーションの最終的な目標はリアルタイム処理です．まずは，推論結果を視覚化します．

具体的には，動画像を入力とし，白線検出結果を重畳した動画像を出力として得るものとします．このフェーズはGoogle Colaboratory（Colab）上で実施できます．

実験にあたっては，前章のモデル構築で用いたColabの環境を引き続き利用します．入力データとして構築した学習済みモデル（my_model，MobileNetV2＋U-Netベースのもの）に加え，本誌のウェブ・ページから取得できる動画データCQ-Interface202304_LaneLineDetection_SampleMovie.mp4を準備します．

動画は教師データ作成のためにドライブ・レコーダで撮影したものです．2fpsで撮影してあります．そのため再生すると早送り動画に見えます．容量の関係上，切り出した短時間の動画としてあります．

● コードの解説

コードを**リスト1**に示します．

▶入力する動画の読み出し

リスト1（a）は入力動画像の場所指定と，必要なライブラリの読み込み処理です．入力動画像はGoogle Driveから読み出します．

▶モデルと推論データの準備

リスト1（b）は推論部の定義です．初めに学習済みモデルを`tf.keras.models.load_model`関数で読み込みます．これで学習後の重みを持ったモデルを利用できるようになります．以降の`detect_lanelines`関数の中ではモデルを用いた推論の実施と，画像への結果重畳を行います．

まず，入力画像を学習時と同じサイズのテンソルへ変換します．

（320，320，3）で値域が0～1.0

これをバッチ化してモデルへ入力します．モデルの入出力は画像単位ではなく，画像を複数枚まとめたバッチ単位になることに気をつけてください．

・14～16行目：カラー画像へ変換

モデルより得られた推論結果は4チャネルであり，それぞれの種別（マスク画像で言うところの黒/青/緑/赤）の尤度（ゆうど）を示しています．この結果をカラーのマスク画像へ変換します（**図1**）．

最後にマスク画像のサイズを入力画像を合わせるために，`cv2.resize()`関数で拡大します．

▶レーンを検出する

リスト1（c）は，`detect_lanelines`関数を動画1フレームずつに適用し，書き出すコードです．OpenCVの動画像I/O関数を用いて記述しています．注意点は，推論前後に画素値の並びをRGBとBGRで相互に変換していることです．OpenCVの画像I/O関数は，画素値の並びがBGR順です．対して，学習済みモデルはRGB順の画像が入力されてくる想定です．そのため推論前後で画素値の並び順を変換し，この差を吸収しています．なお，筆者は実装中にBGR並びのままの画像をモデルへ入力するという間違いもしましたが，それでも悪くない推論がなされるあたり，深層学習モデルの汎用性の高さを感じました．

特集のプログラムや動画データは下記URLから入手できます．
https://interface.cqpub.co.jp/2304t1/