スモールスケール上での自動運転(最終発表)

目的

マイクロスケール(RCカースケール)上での
自動運転プラットフォームの開発

プロジェクトに於いて学ぶこと

- 組み込み開発
- リアルタイム制御
  ->クライアント-サーバ間通信構築
  ->PID制御の実装法
- 画像処理手法
  ->OpenCV/NNによる画像処理/認識/推論方法

プロジェクト内容

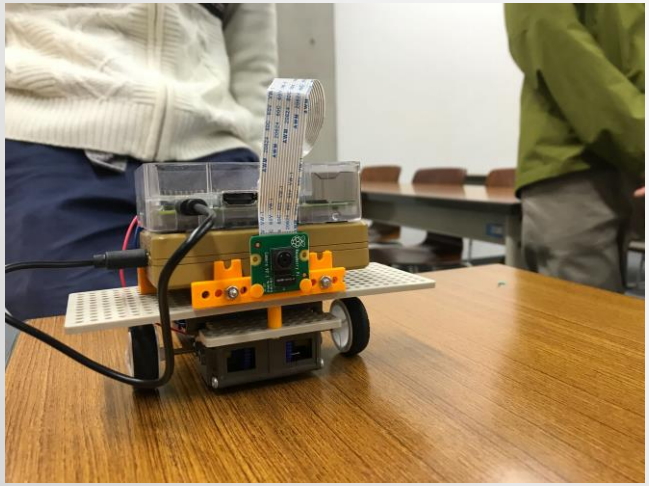
自動運転において実装すること

・路面認識/操舵制御

・標識検知

環境

車両(クライアント)紹介



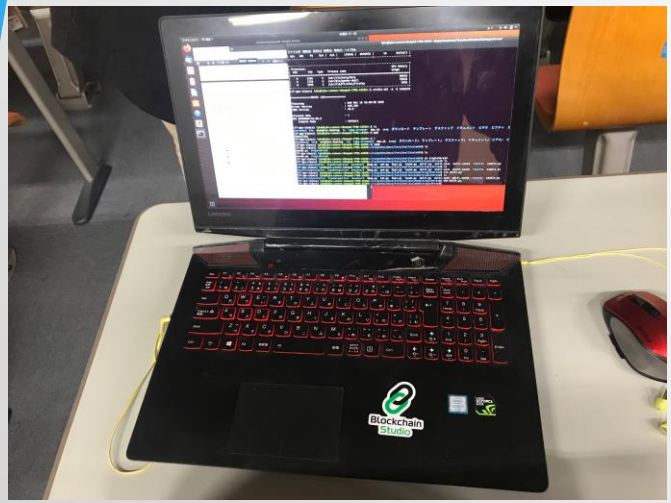
二輪駆動三輪車

駆動系:左右独立モーター制御
奥行:160 mm,
タイヤ間距離:115 mm
カメラ高:60 mm
最高速度:30 cm/s

Raspberry Pi 3B+ 搭載

環境

サーバ側PC紹介



CPU:インテル® Core™ i7-6700HQ プロセッサー

GPU:Nvidia GeForce 960M (CUDA 640コア搭載)

OS:Linux Ubuntu 18.04 LTS

RAM:16GB

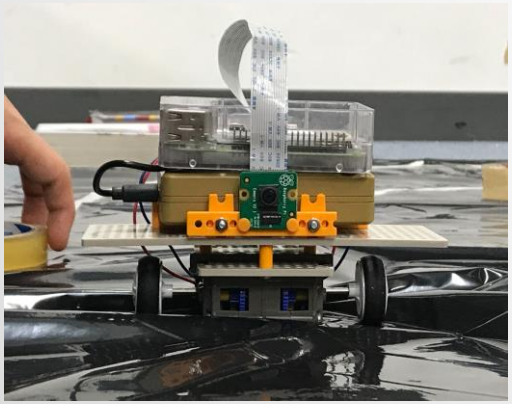
採用理由 ->可搬性に優れる
LAN構築により遅延抑制が期待できる

システム構成

推論情報に
基づくPID制御

カメラ画像送信
(MJPEG-Streamer)

画像認識・
推論
(OpenCV,
Pytorch)



車両
(Rust)

推論情報送信
(MessagePack)

サーバー
(Python)

システム紹介

画像認識

路面検出
(セマンティック
セグメンテーション)

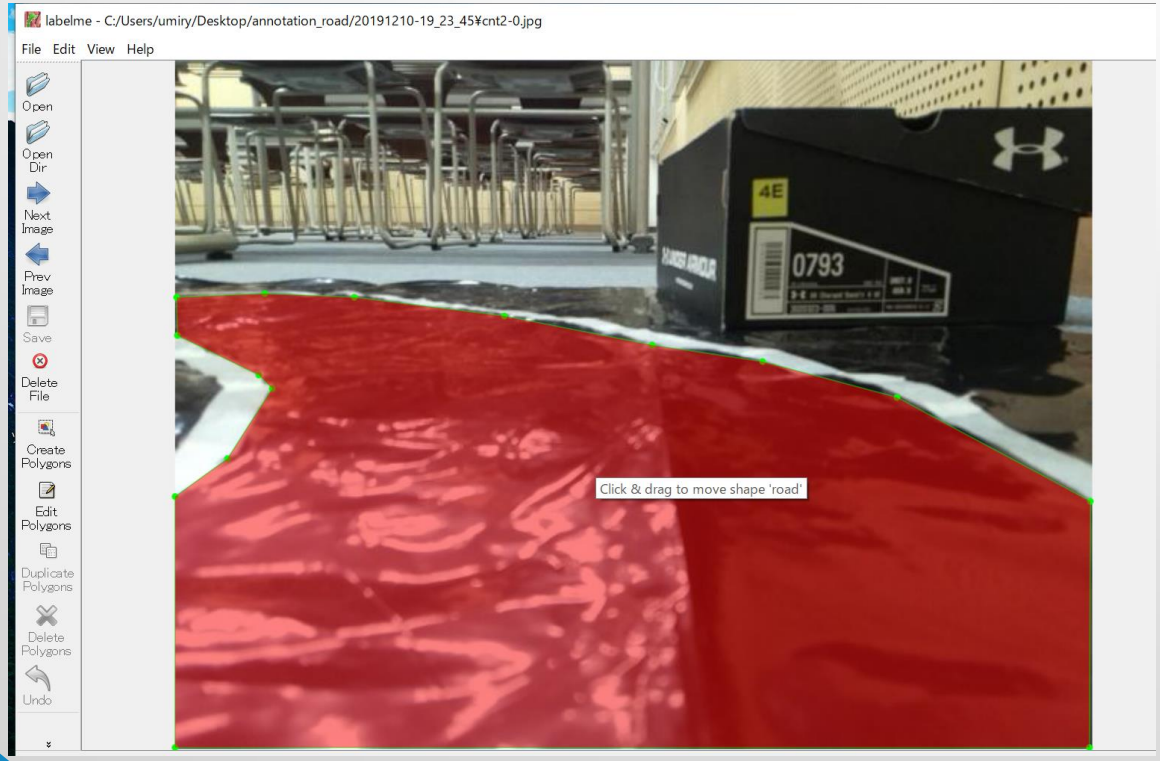
標識検出
(selective search, CNN)

システム紹介

セマンティックセグメンテーションについて

・画像内の各ピクセルについてカテゴリ分類する手法
->今回はESPNetを採用

・自作したデータセットを用いて学習(約8oo枚)

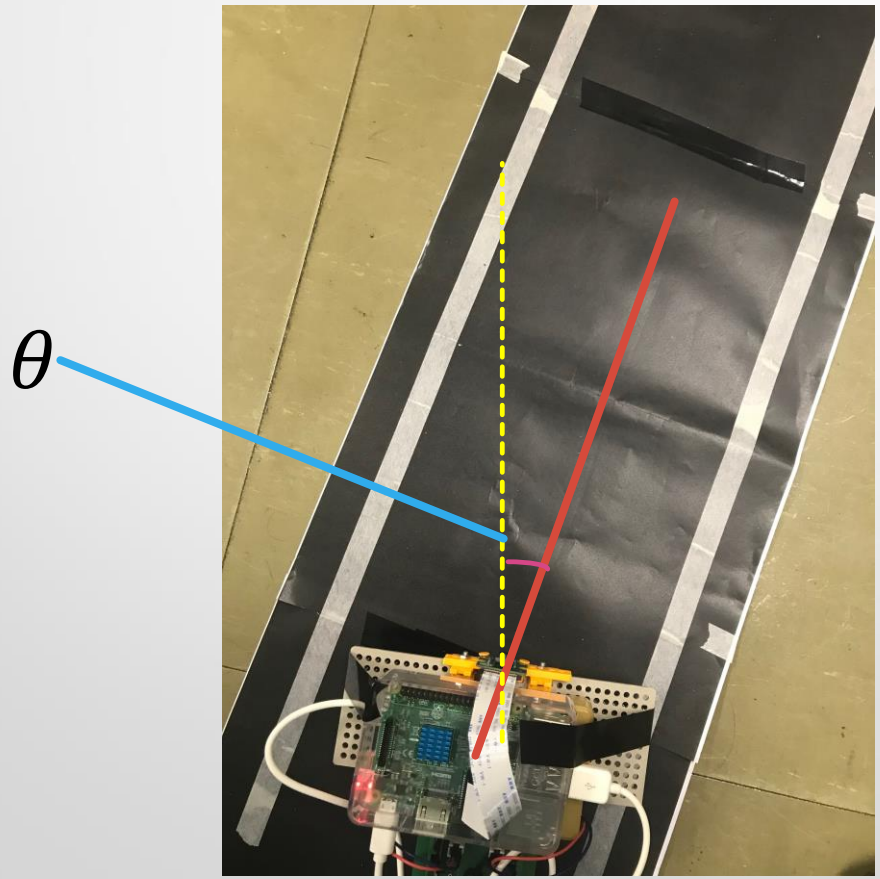


学習用データセットの例

システム紹介

推論(制御用パラメータの算出)

今回は路面の角度を採用

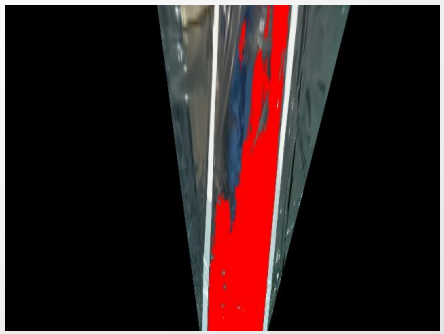


システム紹介

推論

元画像

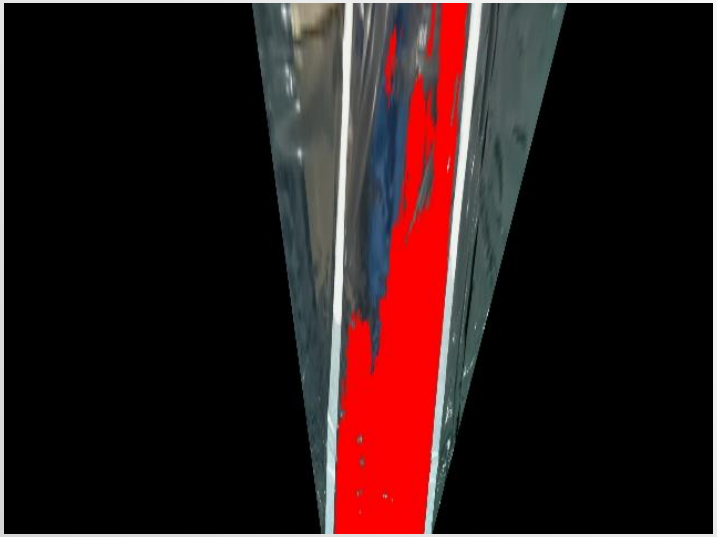
セグメンテーション



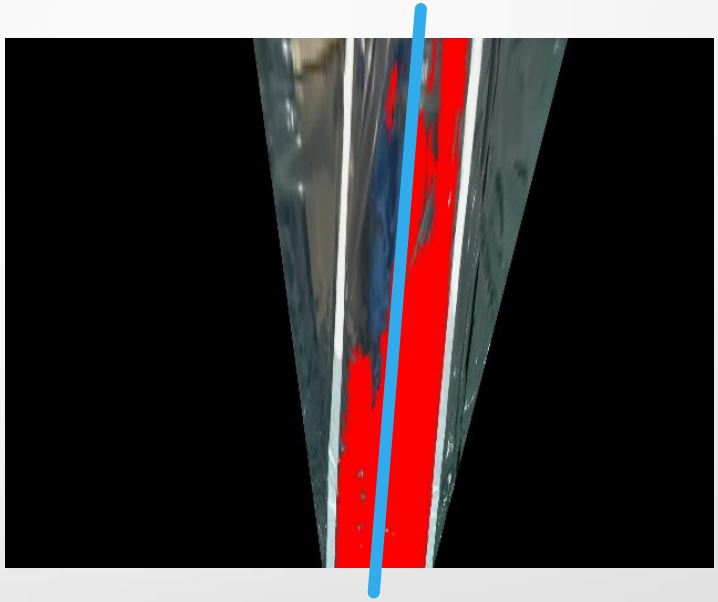
鳥瞰図変換

システム紹介

推論



鳥瞰図変換後



最小二乗法による
一次近似
$\tan^{-1}$での角度算出

システム紹介

推論：標識認識

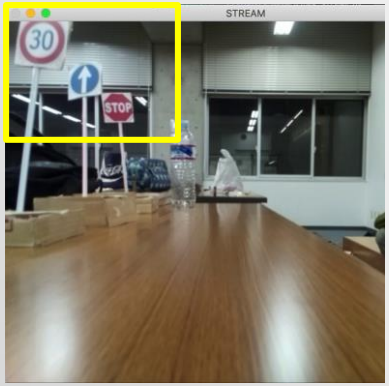
以下の三種類を分類する

速度変更　　運転開始　　止まれ

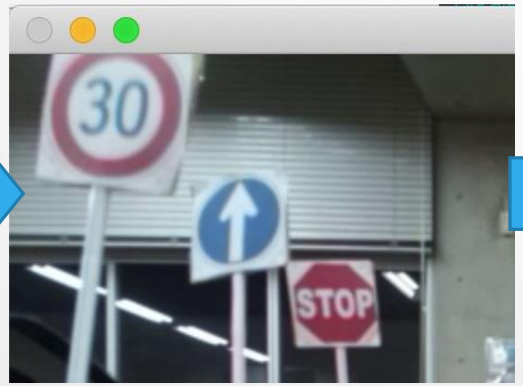
システム紹介

推論：標識認識

認識の流れ



元画像



ROI抽出

SelectiveSearch 適用
(候補領域抽出)

システム紹介

推論：標識認識

認識の流れ

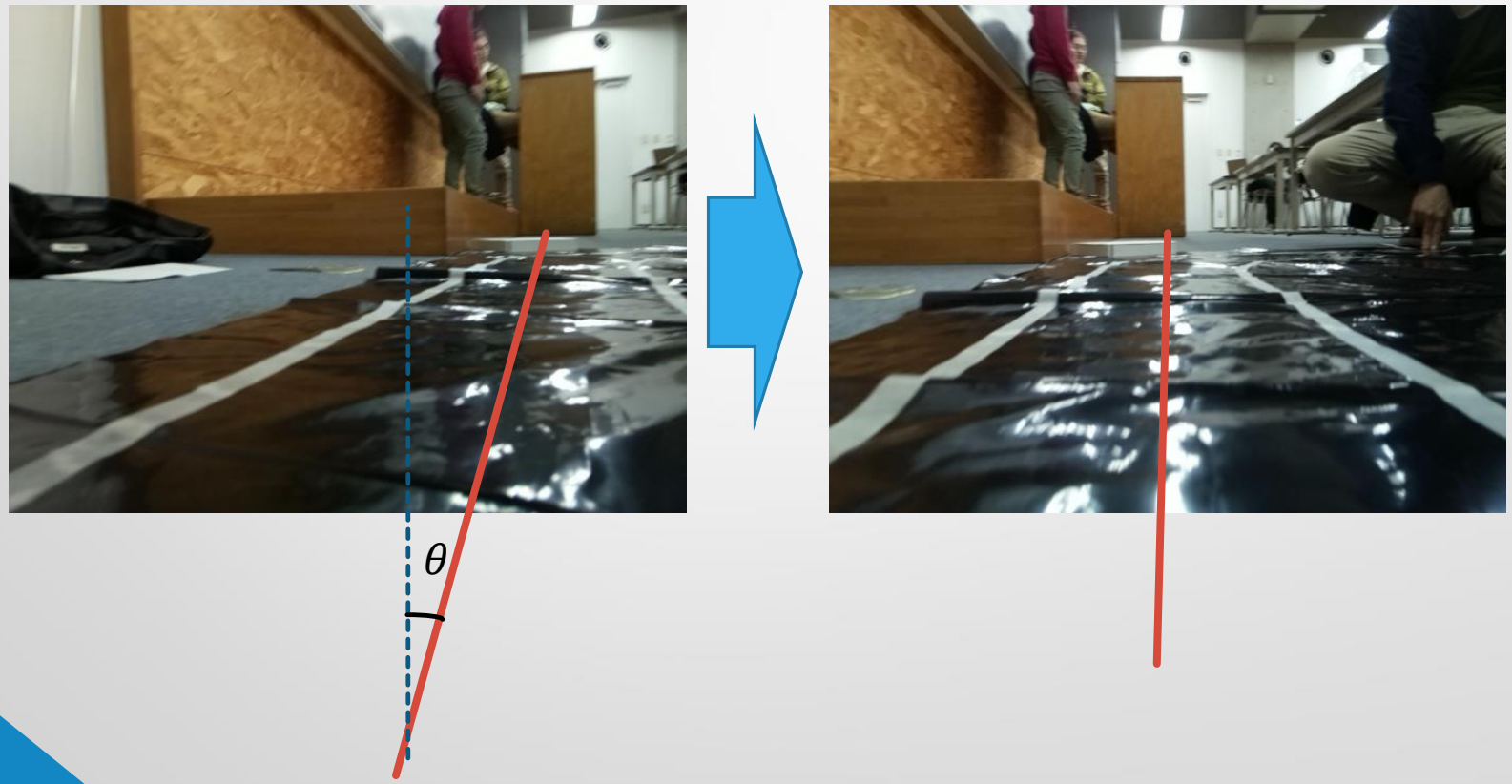
SelectiveSearch 適用
(候補領域抽出)

Vgg19(学習済みモデル)で推論
3class-分類問題を解く

内容

PID制御

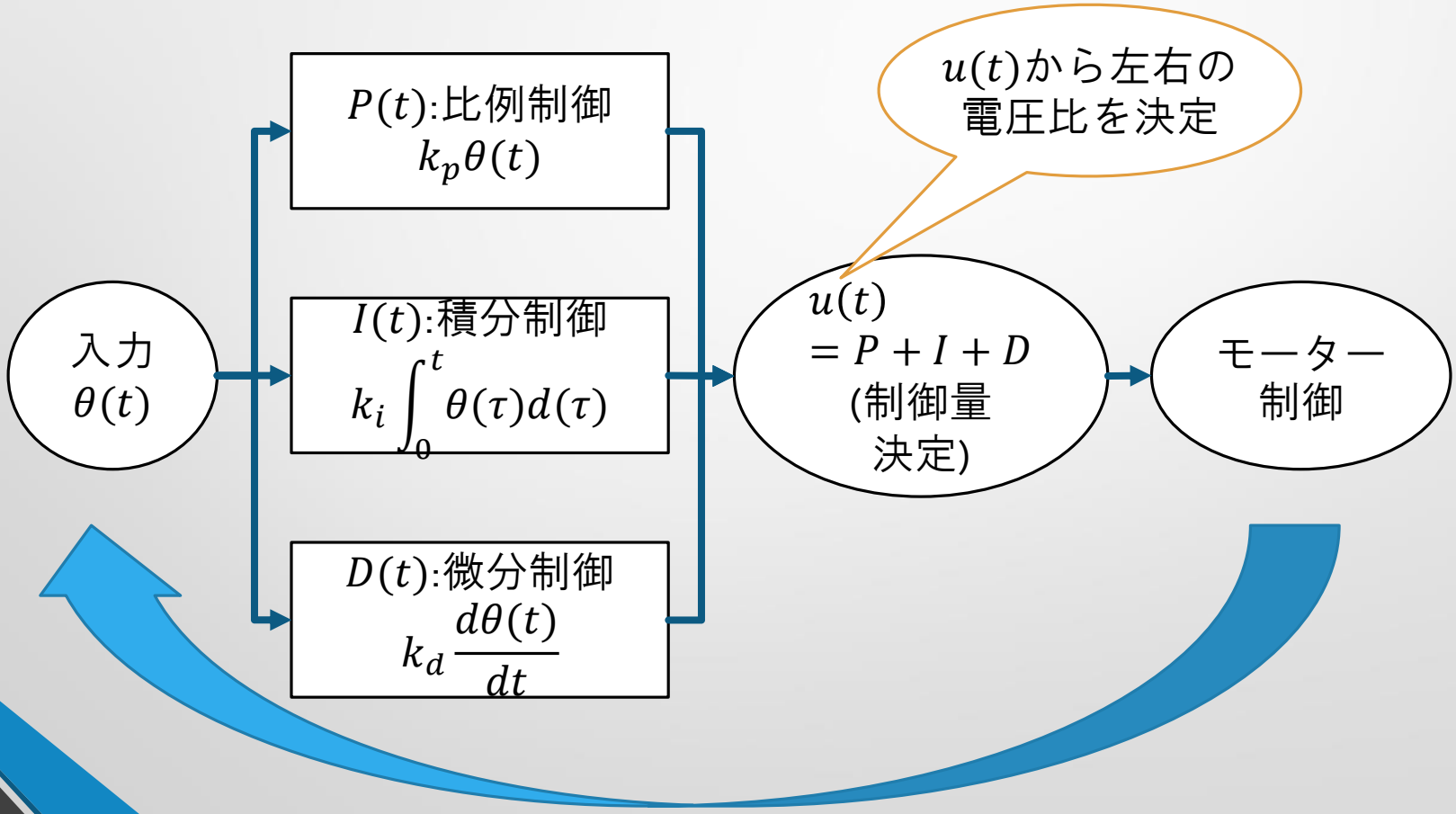
得られるパラメータθ(路面角度)を0にしたい



内容

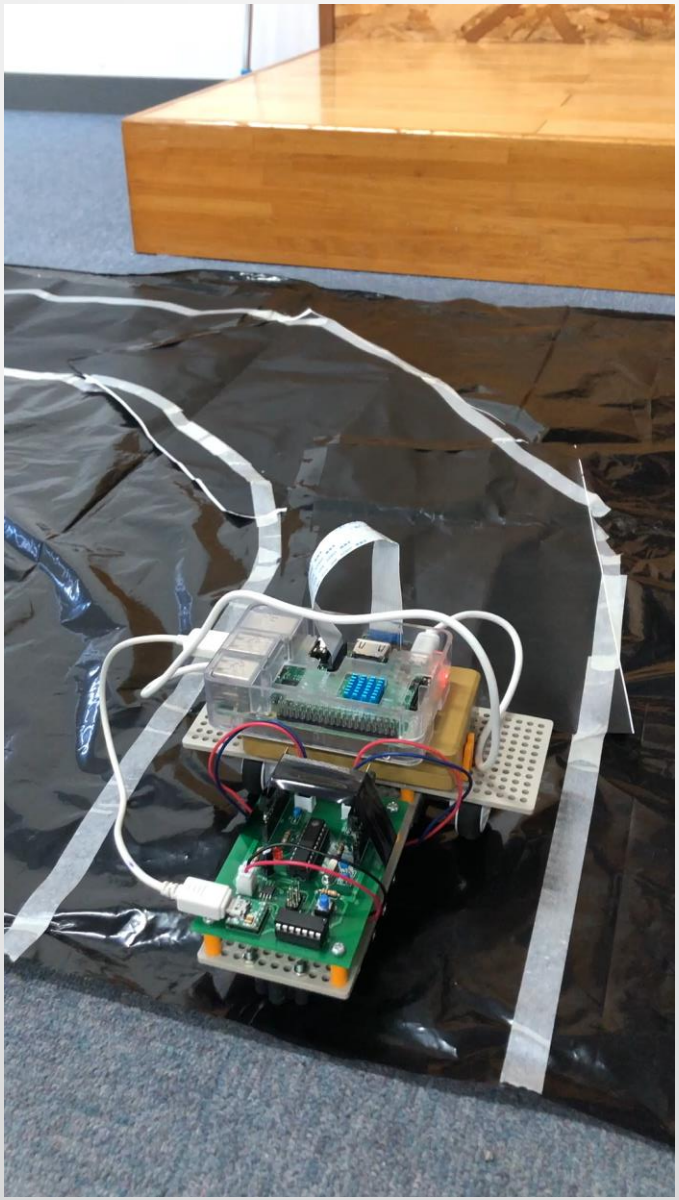
PID制御

時刻tでの制御用パラメータu(t)の算出



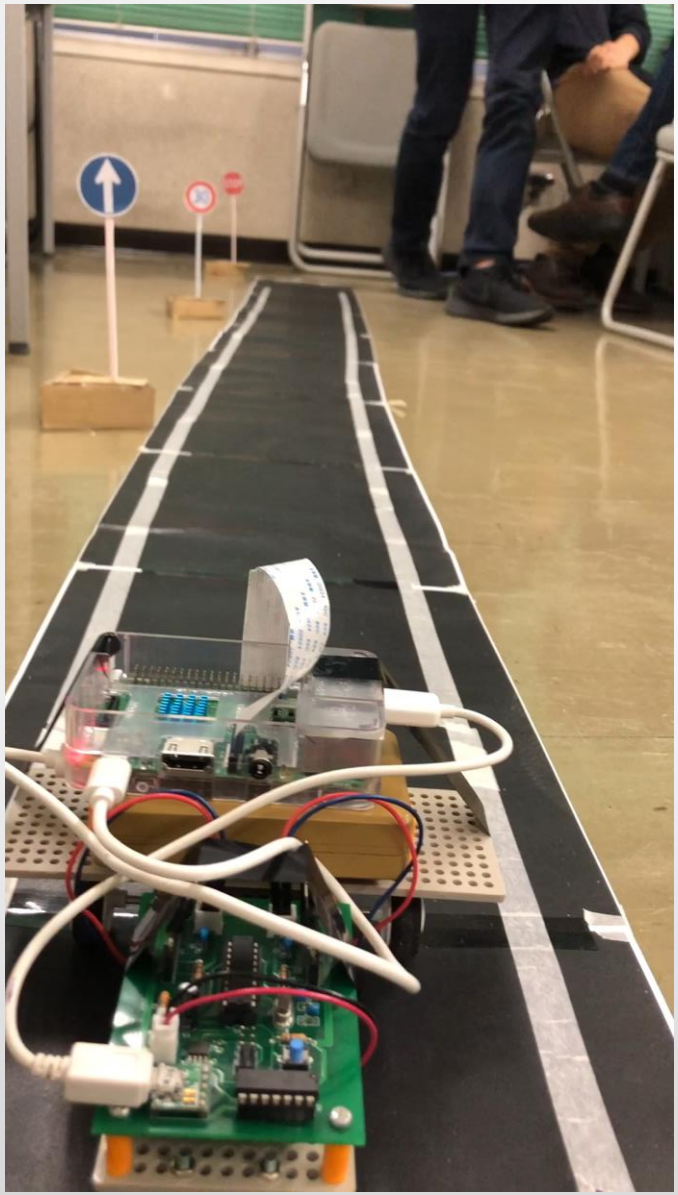
実行結果

カーブに対する応答



実行結果

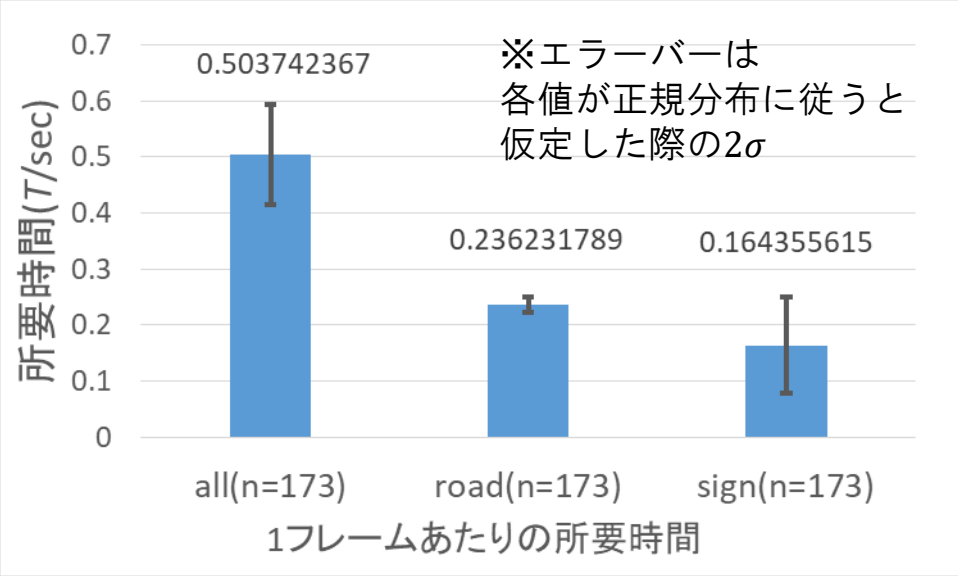
直線に対する応答,
標識認識



システムの評価

処理時間の計測

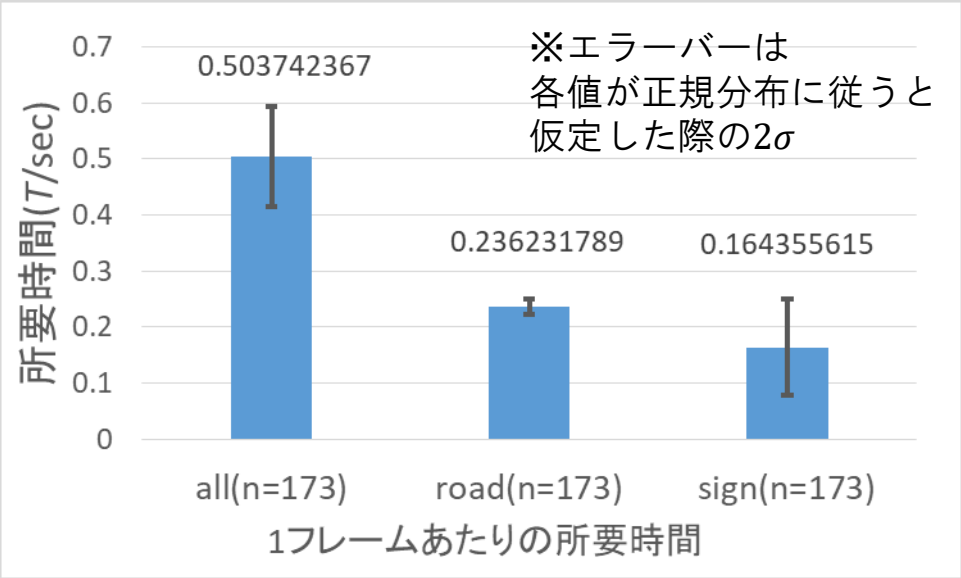
1フレームに対する所要時間



2fps下での自動運転が可能

考察

処理時間の検証



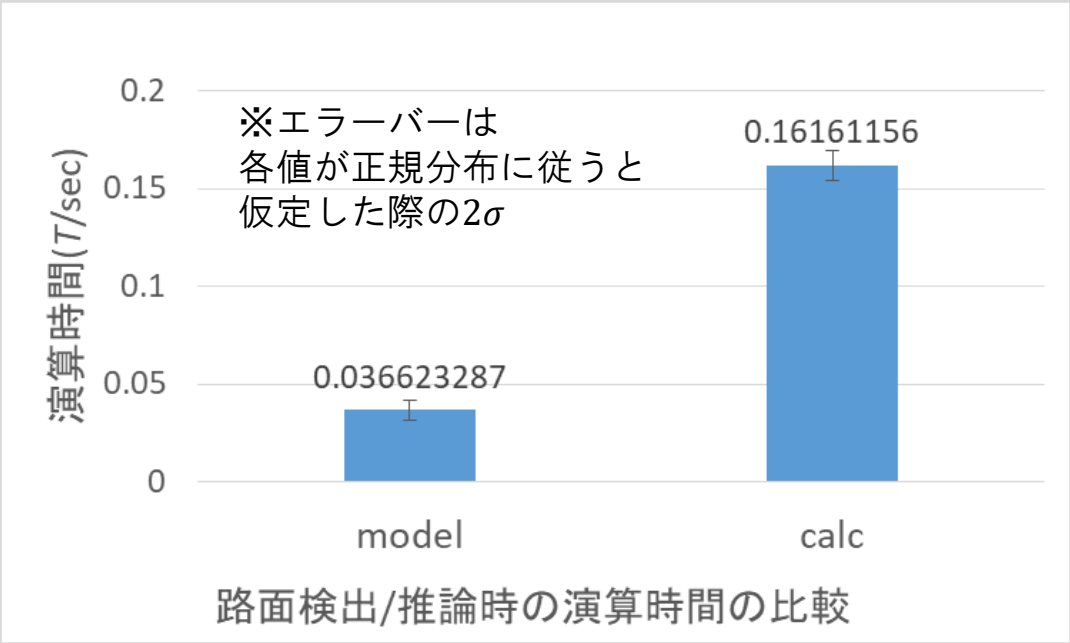
all : 1フレームの全処理の所要時間
road : 路面検出/推定の所要時間
sign : 標識検出の所要時間

・標識検出の分散

->SelectiveSearch後の候補数が影響

考察

処理時間の検証



model : NNモデルでの推論時間
calc: model適用後の演算時間

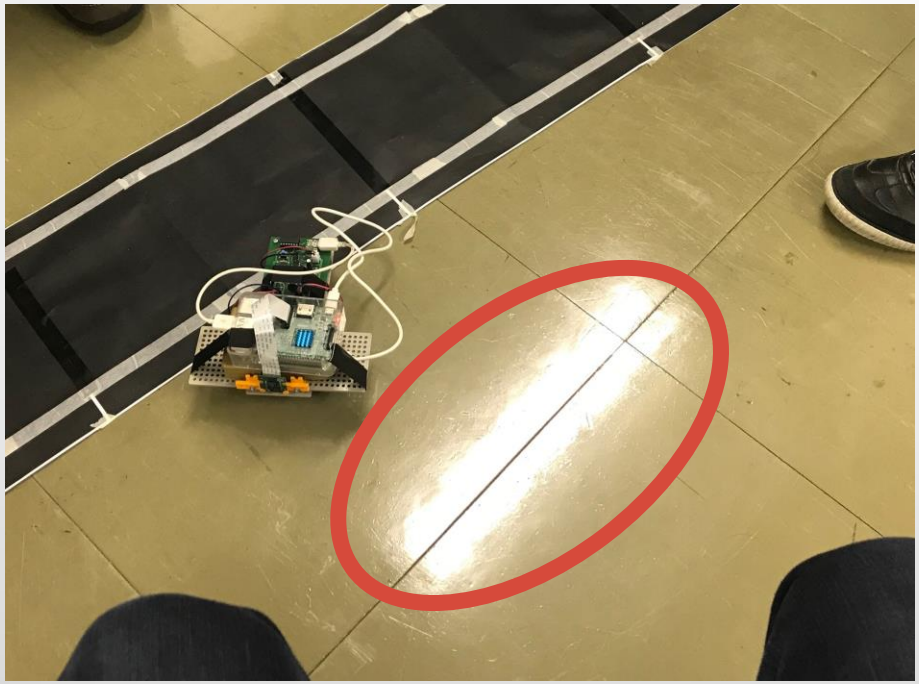
・鳥瞰図変換
・線形近似アルゴリズム

にボトルネック

考察

走行精度について

ノイズ(環境光)による影響が無視できない



考察

走行精度について

Q.精度を向上させるにはどうすればよいか

- 路面推定精度の向上
  ->ノイズを含んだより多くのデータセットでの学習
- 画像処理順序の変更
  ->学習データへの前処理の追加
- 画角の変更、カメラの追加
  ->路面認識範囲の拡大
- PID制御のパラメータ調整
  ->微分制御パラメータの変更

得られた知見

- ハードウェアの(通信)制御の手法
  ->マイクロコントローラ上のアセンブリ実装
- NNを用いたリアルタイム制御の難しさ
  ->時間制約下でのパフォーマンス確保
- 実機と仮想空間上での差異
  ->環境光の厄介さ, マシントラブル