チームプロジェクト：スマロー

(Smart Parking Lot)

カテゴリー

- 卒業プロジェクト

担当パート

- バックエンド

主要技術

- イメージプロセッシング

チームプロジェクト：スマロー

(Smart Parking Lot)

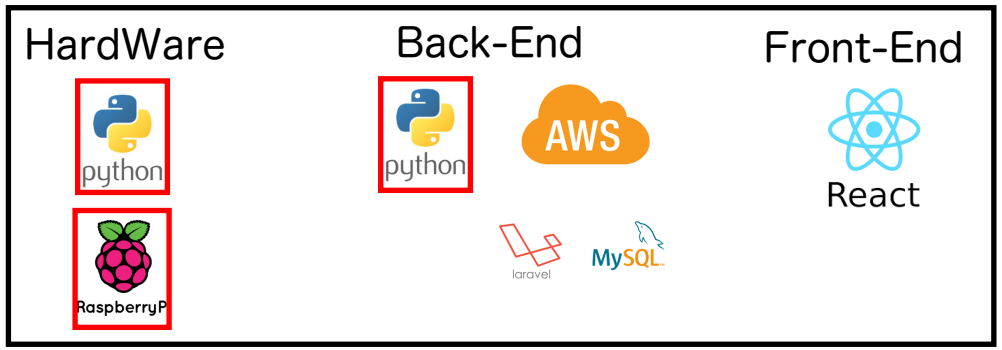
概要

イメージプロセッシングを活用し、
駐車場の利用や管理を便利に低費用で行えるスマート駐車場

主要技術

Object Detection

適用技術



：担当パート

URL：

Code：https://github.com/JeongYeonSeong/Smart_Parking_Lot

プロジェクト紹介

- 屋外駐車場にカメラを設置し、画像撮影
- リアルタイムで画像データを分析
- 駐車場内の車両の位置や駐車スペースのデータを収集
- 駐車場の状況・統計照会
- 駐車経路案内サービス

特徴

- カメラを使い、既存の超音波や赤外線センサーを使った
スマート駐車場に比べ、環境の影響が受けにくい
→屋外設置・管理が容易
- 駐車スペースと経路案内により、駐車時にかかる時間と
場内の混雑度を減らし、HOL(Head Of Line)問題を最小限化

主なサービス

1. User
   - ウェブで駐車場の状況をリアルタイムで照会
   - 駐車経路案内
   - 駐車位置照会

2. Administrator
   - ウェブで駐車場の状況をリアルタイムで照会
   - 期間別、統計照会

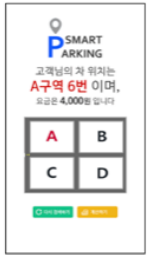
スマロー

サービスの流れ

User

出発前

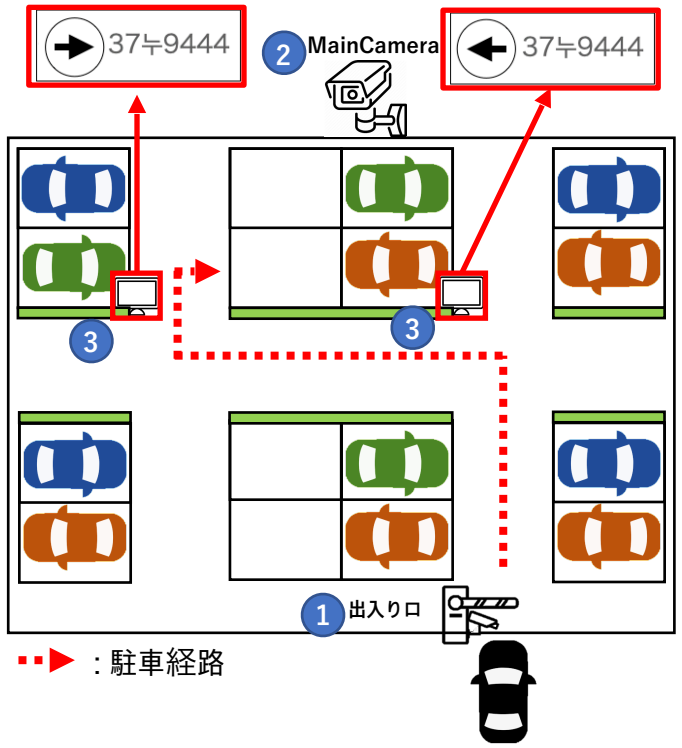
- 駐車場状況照会
  1. 駐車スペースの数
  2. リアルタイム駐車状況



駐車後

- 車両位置照会
  → 駐車位置を忘れた場合
- 料金照会

駐車場内



1 出入り口

- 車両のナンバープレート撮影
- 車両番号認識

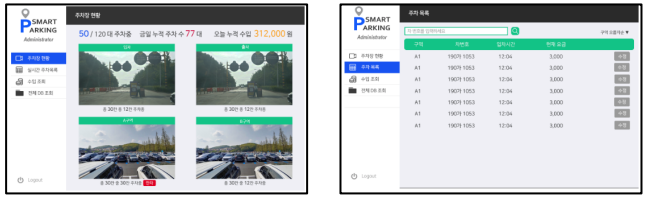
2 MainCamera

- 駐車場全体の画像撮影
- イメージプロセッシング
  -> 車両認識・追跡
- 駐車経路案内

3 経路案内monitor

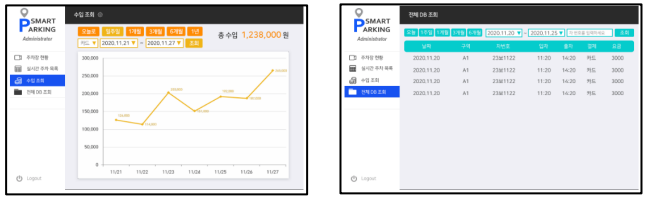
- 車両番号、進行方向案内
  -> Userに経路案内

Administrator



リアルタイムで駐車場の状況確認

- 駐車スペース
- 駐車状況
  -> 車両位置、駐車料金



期間別照会

- 期間別駐車場利用状況
- 車両別駐車場利用状況

スマロー

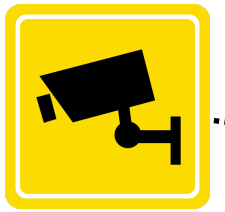
□ 担当パート

システムの構造

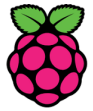
データ収集

Main Camera

画像撮影



出入り口



RaspberryPi

ナンバープレート撮影

ナンバープレート認識

・車両番号

Main Computer

画像データ分析

・車両認識
・車両位置追跡

データ加工

・Navigating

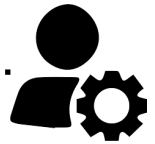
・車両番号
・車両の進行方向

User

User

・駐車場状況照会
・駐車位置確認

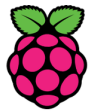
Administrator



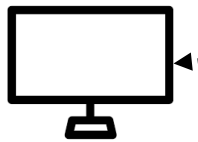
・駐車場状況照会
・期間別統計照会

Server

経路案内Monitor



RaspberryPi



User

・駐車経路案内

スマロー

担当パート

担当パート設計

課題：空車の場所をどうコンピュータに認識させるか？

思考の流れ

- 現状持っているデータ、新たに必要なデータを整理
- 持っているデータをどう活用するか？
- 新たに必要なデータをどう収集するか？

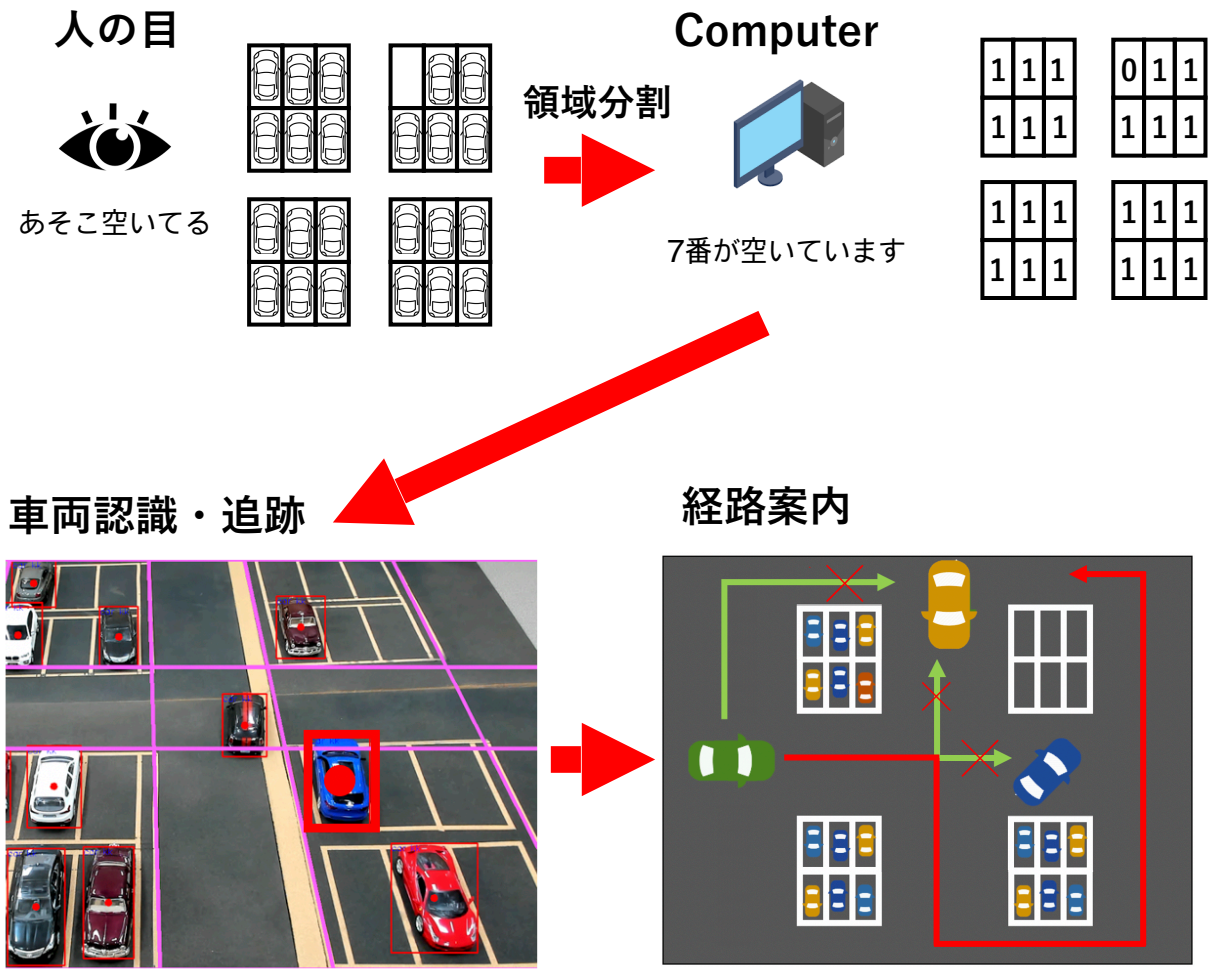
Navigating → 駐車場の情報が必要
[対象車両の位置] => (YOLO)
駐車スペース => 獲得必要
移動可能道路 => 獲得必要
道路の混雑度 => 獲得必要

コンピュータに認識させる

方法：
領域デジタル化 -> 領域分割 - Numbering
YOLO -> 車両の有無 – 座標抽出
データ加工 -> **データ獲得**

設計：
仮想の**領域分割**
-> 車両の座標と比較 - Detecting(**車両認識**)
-> 車両の位置情報更新
-> 駐車場内の情報デジタル化 } Tracking(**車両追跡**)
-> Navigating(**経路案内**)

設計から実装まで1人で担当

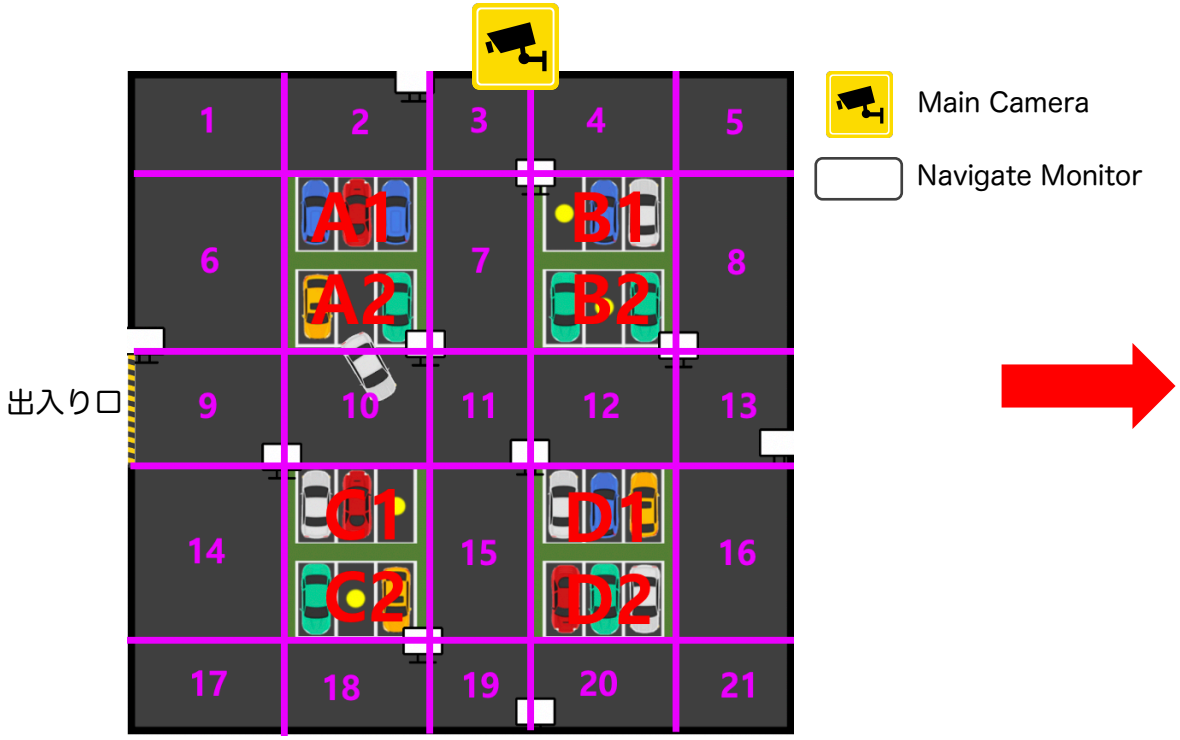


スマロー
- データ取集

担当パート

領域分割1

駐車場の平面図(例)



* 上の平面図は理解しやすくするための例で
実際はカメラが約40度傾いている

1	2	3	4	5
6	A1 A2	7	B1 B2	8
9	10	11	12	13
14	C1 C2	15	D1 D2	16
17	18	19	20	21

図にした「Area Node」テーブル

- 駐車区域
- 出入り口：9
- 最終案内目的地：2、4、10、12、18、20

- 車両位置確認や経路設定の基準を決めるための駐車場領域分割
- これに基づいて車両の現在位置や進行方向を把握し、経路設定
- 最終案内目的地：駐車区域に一番近い領域まで案内し、
利用者が駐車スペースを直接目視で確認後駐車

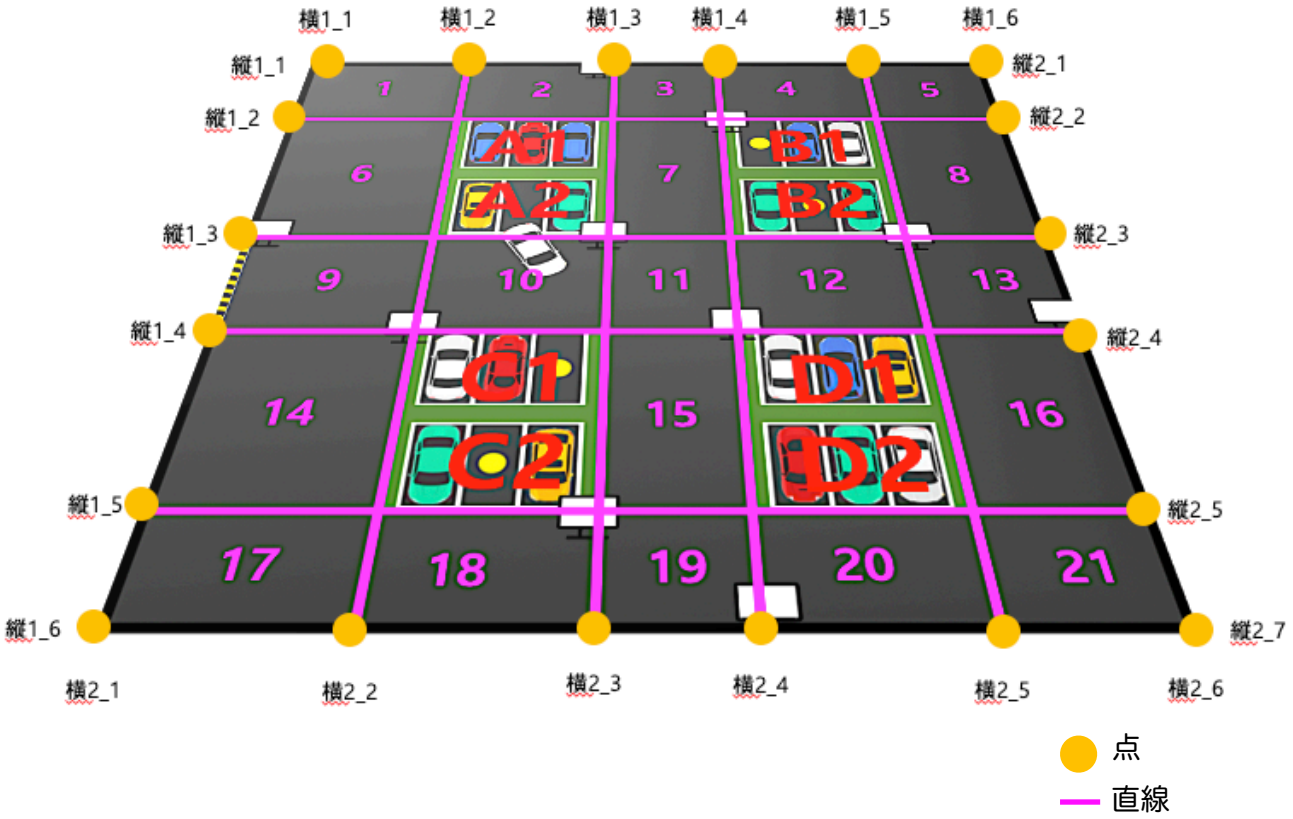
スマロー
- データ収集

担当パート

領域分割2

実装時の領域分割

- 駐車区域の点を設定し、直線を書いてできた四角形の領域抽出
  -> 領域分割の便利性や拡張性のため



設置時の駐車場領域抽出 – (例)

1. 点を向き合っている点と繋ぎ、直線を書く
   ex) (横1_1 – 横2_1)、 (縦1_1 – 縦2_1)

2. 図のように領域に番号を設定
   - 本プロジェクトでは1~21に設定

3. 駐車可能区域設定
   - 本プロジェクトではA、B、C、Dに設定

4. 駐車スペース内の細部分割
   - 本プロジェクトでは区域ごとに６スペースに設定

ex) A区域

1	2	3
4	5	6

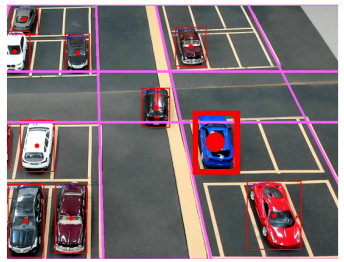
* イメージプロセッシングで領域を自動分割もできるが、
雪や障害物などによって駐車線が認識不可能になる恐れがあるため、上記の方法を適用

担当パート

スマロー
- データ収集

車両認識・追跡

車両認識



- **YOLO(You Only Look Once)**

  YOLO：イメージプロセッシングを通じたリアルタイム物体認識システム

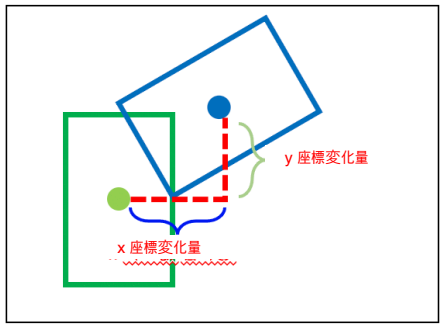
  使用目的：車両認識、車両位置抽出

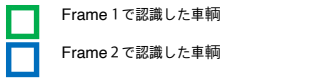
車両位置追跡

- **アルゴリズム**

  - 今のフレームと前のフレームを比較し、認識した車両の座標が
    基準値範囲以内の場合、車両の座標更新

  ex) | frame1のx – frame2のx | < 基準値
  | frame1のy – frame2のy | < 基準値) **条件を満たす場合、車両の座標更新**





lx 座標変化量l
ly 座標変化量l **< 基準値**

=> 車輌の位置座標更新

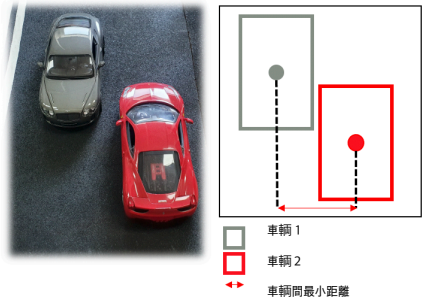
基準値

基準値設定

- 対象を同じ車両と認識する数値
- 基準値が大きすぎる場合、他の車両と座標がすれ違う恐れがある

***車両位置追跡の信頼度を高めるため、下記のような状況を考慮する**

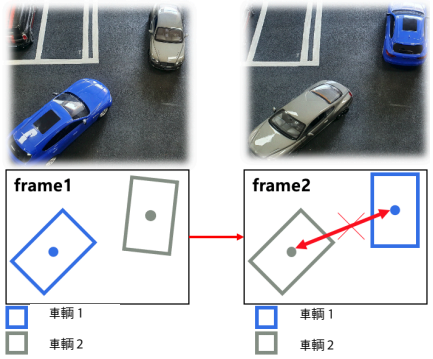
1. 車両がすれ違う場合



問題：各車両の座標が入れ替わる恐れがある
ex) 図1で灰色の車と
赤色の車がすれ違う場合

解決：基準値を各車両の間の最小距離以下に設定

2. 各車両の位置が変わる場合



問題：車両が他の車両があった位置に移動すると、
速度によっては座標が入れ替わる恐れがある
ex) 図2のように２枚のフレームの間で
（実際）車両の位置が入れ替わる場合

解決：駐車場内の平均スピード：約30km/h
TOYOTA プリウス：長さ-約4.5m, 幅-約1.7m
24fps 環境での
平均移動距離：34.7cm/frame(車両の幅の約5/1)
-> 基準値 <= 34.7cmに設定

結論

＊平均移動距離 < 各車両間の最小距離

＊基準値：移動距離以下 -> 車両追跡の信頼度最大化

スマロー
- データ収集

担当パート

経路案内

Navigating

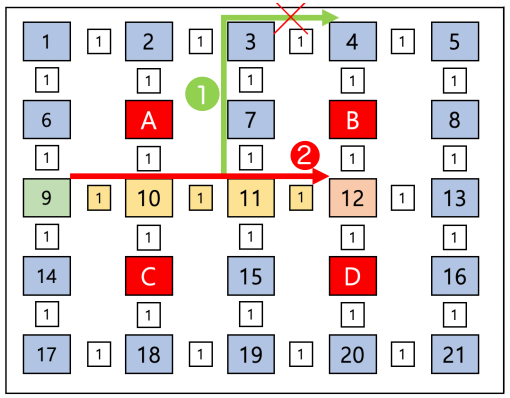
Navigatingの目的

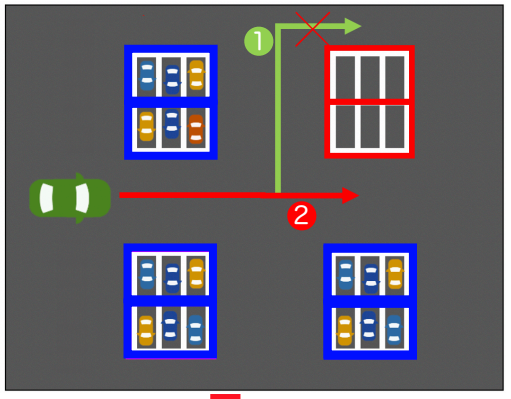
- Userが自ら駐車スペースを探す時に生じるTime LossとHOL問題を最小限化

* HOL(Head Of Line)：車両列の先頭車が車路を占有し、後方車が進める車路がなくなり、混雑度が連鎖的に高くなる問題

・ Daikstra (出発点から到着点までの最短経路を探すアルゴリズム)活用

・ **リアルタイムで経路間の混雑度、駐車可能可否を元に経路設定**





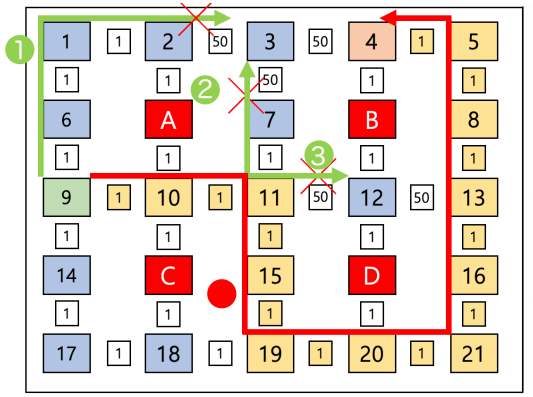
各経路の混雑度

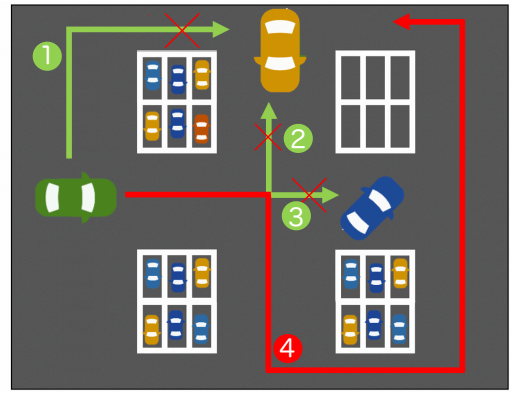
1) 9 –> 10 –> 11 –> 12 : 3
2) 9 –> 10 –> 11 –> 7 –> 3 –> 4 : 5

: 駐車スペース
: 満車

駐車場内の状況	経路案内
駐車区域	出発地
駐車場領域	案内経路
領域間の混雑度	目的地

* 目的地までの経路内に車両がある場合

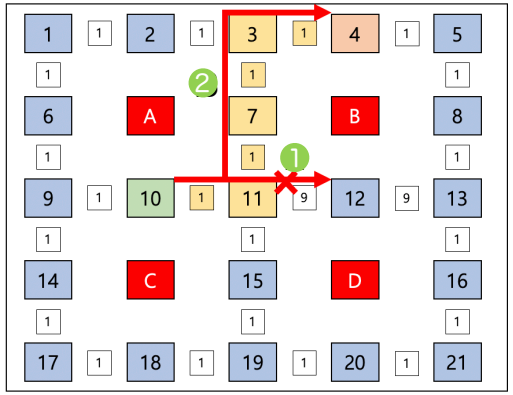


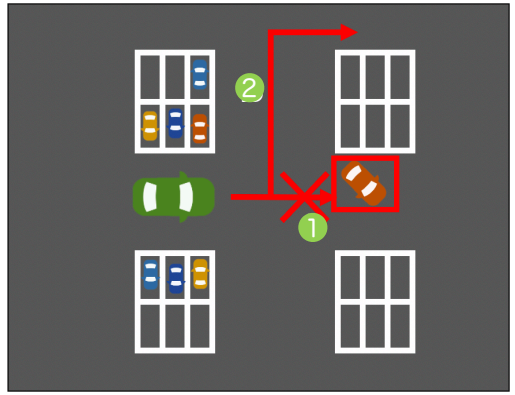


経路内に車両がある場合の混雑度

1) 9 –> 6 –> 1 –> 2 : 53
2) 9 –> 10 –> 11 –> 7 –> 3 –> 4 : 103
3) 9 –> 10 –> 11 –> 12 : 52
4) 9 –> 10 –> 11 –> 15 –> 19 –> 20 –> 21 –> 16 –> 13 –> 8 –> 5 –> 4 : 11

* 駐車経路案内中駐車場の状況が変わる場合





各経路の混雑度

1) 10 –> 11 –> 12 : 10
2) 10 –> 11 –> 7 –> 3 –> 4 : 4 - 経路再設定

1 : 案内中経路内の車両認識
2 : 再設定された経路

スマロー

プロジェクトを終わらせて

* 苦労したこと

- **個人的**
  - AI、イメージプロセッ シング、IoT技術は初めてでしたので実装しなければならないものの具体化が難しく、細部設計が大変でした。
  - 実装中、AIの認識信頼度が低く、データ加工が大変でした。
- **チームメンバーとして**
  - 個人プロジェクトとは違って、チームメンバーみんなの計画を立ててまた一つに集めることが難しかったです。

* 学んだこと

- **個人的**
  - Pythonのnumpy、openCVなどのLibraryを活用したイメージプロセッシングの基礎を勉強し、生かせる方法を学びました。
  - 作曲家としてではなく、ITエンジニアとしての問題を見る視線や姿勢に気づき、設計・実装時の心構えを身につけました。
  - 機器間の有機的な連動のために Socket 通信の基礎と活用方法を勉強し、通信プロトコル設定の重要性を実感しました。
- **チームメンバーとして**
  - 個人ではなく、チームの目標を達成するために必要な責任感について悩みました。

    今後のチーム活動をよりうまくやっていける自信ができました。