令和2年度卒業論文

画像処理を用いた立石港待機レーンの

混雑量計測システムの開発

令和 3年 1月 14日

学籍番号：163038

氏名：宮地香樹

指導教員：田房　友典

弓削商船高等専門学校情報工学科

1. はじめに
2. 概要
   1. システム全体の構成
   2. 開発環境
3. カメラについて
   1. 使用するカメラ
   2. カメラの設定
4. 画像処理について
   1. 画像処理の流れ
   2. ディレクトリの監視
   3. 特徴点検出
   4. 背景差分・エッジ検出
   5. 車の台数計測
5. 実験
   1. 実験内容
   2. 結果
   3. 誤差
6. Webの表示部
7. おわりに
8. 参考文献
9. はじめに

現在，上島町には他の島につながる橋がなく，陸路での移動は不可能である．そのため，住民は通勤や通学などに船を利用して生活を送っている．その中で車を伴う移動の際に最も利用されているのが，立石港と長崎桟橋をつなぐ生名フェリーである．利用客が多い要因として，航路が短く，乗船時間，待ち時間共に短いことが挙げられる．

一方で，利用客が多いことによる問題も発生している．それは，朝方の通勤ラッシュや，夕方の帰宅ラッシュの時間帯の車の量が多く，待機レーンが埋まってしまうことである．さらに2021年には，岩城橋の開通が予定されている．岩城橋が開通することで，従来利用していた航路より，生名フェリーを利用したほうが，移動時間が短縮され利便性が高まる．これによって生名フェリーの利用者がさらに増加すると予想されている．対策として，より多くの車両を運搬できる新しいフェリーを導入し，立石港の待機レーンを1つ増加させた．

本研究では，上記の対策に加えて，ソフトウェア面での支援を目的とし，待機レーンの混雑状況を住民が把握できる仕組みを作り，さらに混雑を解消させるシステムを開発する．

IPカメラを用いて待機レーンを一定時間おきに撮影する．撮影した画像を用いて画像処理を行い，待機レーンの混雑状況を計測する．画像処理は背景差分とエッジ検出，特徴点検出を用い，言語はPython Open-CVを使用する．画像処理で求めた結果をWeb上で公開する．Web上ではグラフを用いて視覚的にわかりやすく混雑状況を住民に知らせる．使用言語はPHP，HTML，JavaScript，MySQLである．

1. 概要
   1. システム全体の構成

　本システムの全体の構成は図2.1のようになっている．IPカメラを設置し，待機レーンを左の画像のような角度で一定時間おきに撮影する．撮影した画像をサーバ上に転送し，その画像を使用して画像処理を行う．このカメラとサーバはローカルなネットワークで構成されている．画像処理で得た待機レーンのデータをクラウド上のWebサーバに送る．このデータを棒グラフでWeb上に公開し，住民が混雑度を把握することができる．画像の転送をローカルなネットワークで行い，グローバルなネットワークではデータのやり取りのみを行うことで最低限のセキュリティを確保している．

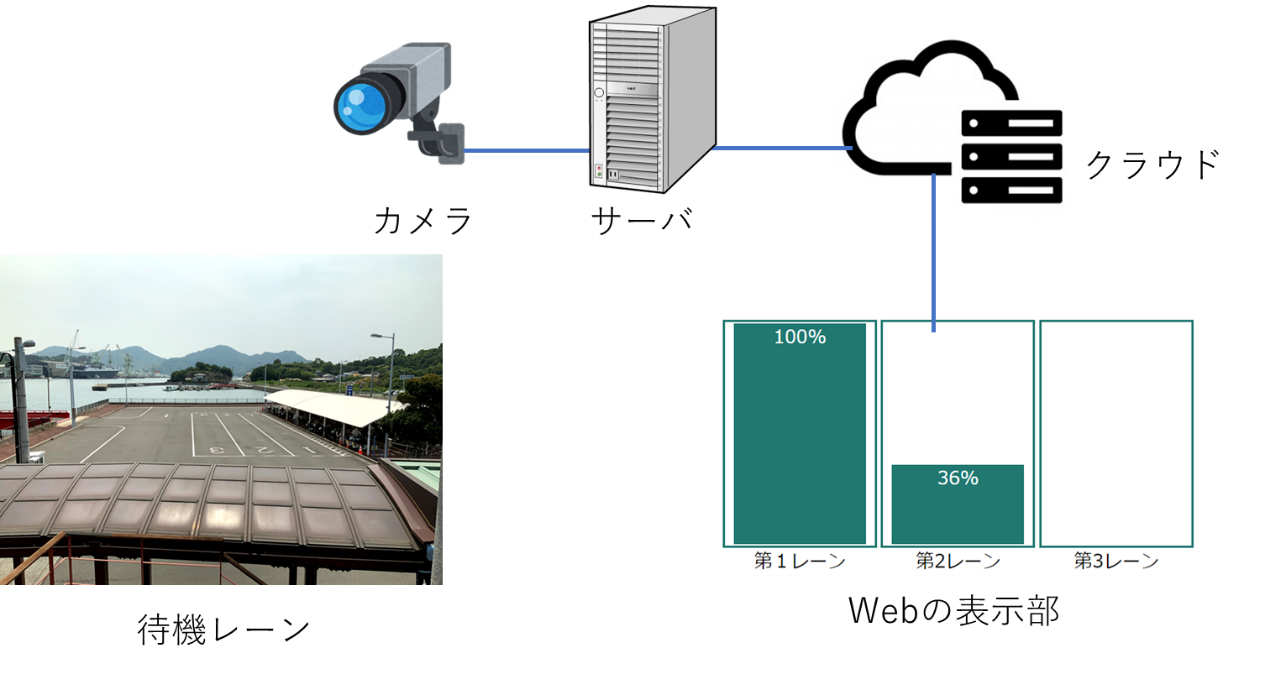


図2.1 システム構成

* 1. 開発環境

　開発に使用した言語と，開発に使用したソフト・環境は以下の通りである．

言語：Python Open-CV，PHP，HTML，JavaScript，MySQL

環境：anaconda，Ubuntu，Visual Studio Code

1. カメラについて
   1. 使用するカメラ

　使用するカメラはPanasonicのi-Pro WV-S1511LNである．このIPカメラには以下のような機能が備わっている．下記の機能を利用して，待機レーンを一定時間おきに撮影し，その画像を転送する．

* スケジュール管理機能
* 一定時間おきの画像撮影（1s/3s/5s/10s/15s/30s/60s…　この中から設定可能）
* 画像の転送
* 認証設定
  1. カメラの設定

　本研究では，実際に待機レーンに設置するものと同じカメラを用意し，カメラの設定方法の確認と簡単な動作実験を行う．これを行う実験に使用している待機レーンの映像は，指導教員の協力のもと別カメラで撮影したものである．理由は，コロナウィルスの関係で学校に登校することが出来ず，現地に赴くことも困難であり，いつ再開するかも不明な状態だったためである．

具体的に行った設定について説明する．まずは認証情報を追加し，パスワードがなければアクセス出来ないようにする．次にスケジュール管理と一定時間おきの画像撮影の設定を行う．スケジュール管理ではカメラが画像を撮影したり，転送したりする時間帯を設定することができる．これはフェリーの稼働時間と合わせて設定する．また撮影間隔については，待機レーンの状態は1秒ごとに変化が起こるようなものではないため，15秒おきに設定する．この撮影した画像をftp転送で指定したサーバへ送る．

1. 画像処理について
   1. 画像処理の流れ

　画像処理の流れを図4.1に示す．ディレクトリの監視を行い，画像が送られてきた場合にのみプログラムが動作する．画像が送られてきたら，車が一台もない待機レーンの画像と，撮影された待機レーンの画像を用いて背景差分を行う．背景差分を行う際に，より精度を上げるために特徴点検出を行い，撮影した画像の位置を修正する．背景差分を行ったあとにエッジ検出を行う．指定した範囲の白画素を計測して，車がそこにあるかどうかを判定する．

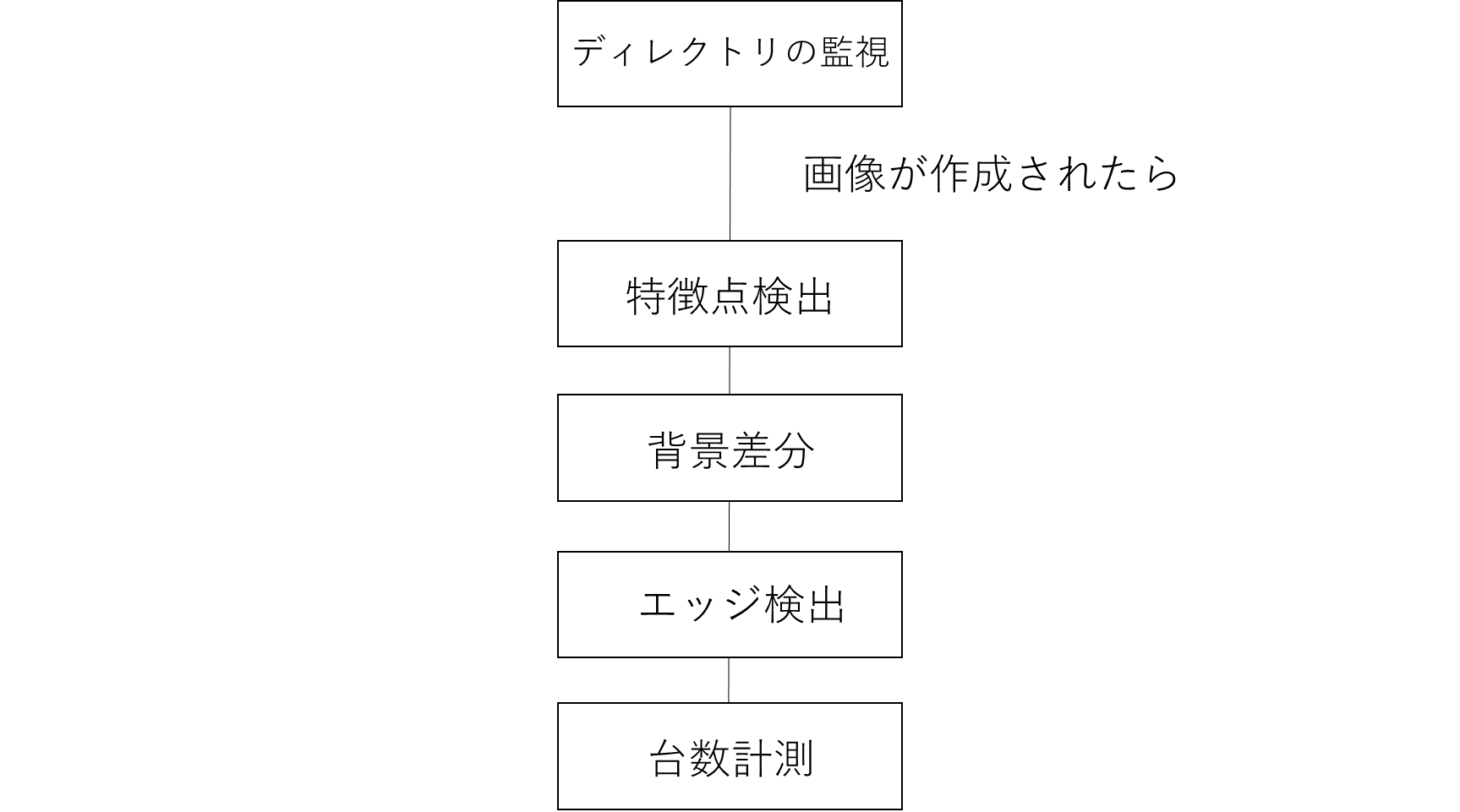


図4.1　画像処理の流れ

* 1. ディレクトリの監視

　ディレクトリの監視にはPythonのwatchdog[1]というライブラリを使用している．このwatchdogは，任意のディレクトリの任意のパターンの名前を持つファイルのイベントを監視し，特定のイベントが発生した場合に，任意の作業を行うようにするモジュールである．これを利用して，ディレクトリに画像が作成された場合にのみ画像処理を行うプログラムを作成する．

* 1. 特徴点検出

　カメラは野外に設置するため，風などの自然現象により多少角度がずれることが予測される．しかし，後述する背景差分を利用するには，できる限り車以外は同じ条件であることが望ましい．そこで図4.3に示すような待機レーンの前側で特徴点検出[2]を行い，撮影した画像の位置を修正する．これによって，多少のずれであれば位置を調整し，正しく背景差分を行うことが可能になる．



図4.3　特徴点検出を行う場所

* 1. 背景差分・エッジ検出

　車の位置を検出するには背景差分とエッジ検出を用いる．背景差分とは，少し前の状態の背景画像と，物体の入り込んだ画像を比較し，2つの画像の差をとることで物体を検出する手法である．特徴点検出によって位置を修正した画像と，車が一台も止まっていない画像の2つを用いて背景差分[3]を行い，車を検出する．実際に背景差分を行った画像が図4.4に示す．この画像にエッジ検出[4][5]を行う． エッジ検出とは，物体の形や材質，色が大きく変わるエッジ部分を検出することである．このエッジ検出を行うことによって，物体の輪郭を求めることができる．エッジ検出を行う理由は，そちらのほうが背景差分のみを行うよりも精度が高くなったためである．なぜそうなるのかについては後述する誤差のところで考察する．

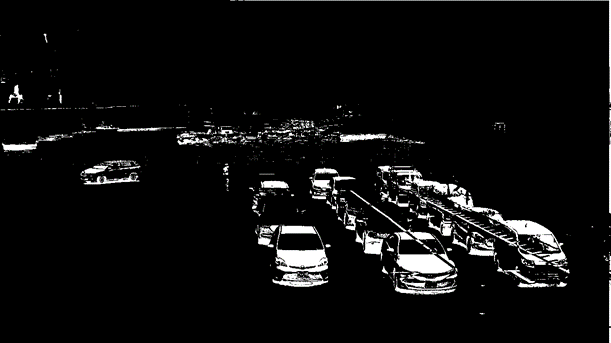


図4.4　背景差分を行った画像

* 1. 車の台数計測

　車の台数の計測は，指定した範囲の白画素数を数える[6]ことによって行う．図4.5のようにエッジ検出を行った画像のうち，車がいると想定される範囲を矩形で囲む．囲まれた範囲の白画素数を調べ，一定以上の白画素数があった場合にはそこに車がいると判定している．車があると判定された回数によって車の台数を求める．

またそれぞれのレーンの車の台数から，レーンの埋まり具合を求める．レーンの埋まり具合については1つのレーンにおよそ6台の車が入ると想定し，求めたレーンごとの車の台数を6で割って求める．これを百分率で表すために，100でかけてレーンの埋まり具合とする．レーンの埋まり具合を求める理由は，車の台数よりも，レーンの埋まり具合のほうが，船に乗れるかどうかにおいて重要だからである．例えば，トラックなどの大型車の場合は1台であっても，2台以上の幅をとることがある．こういった場合に，車の台数ではなくレーンの埋まり具合を求めた方が，誤差が小さくなる．

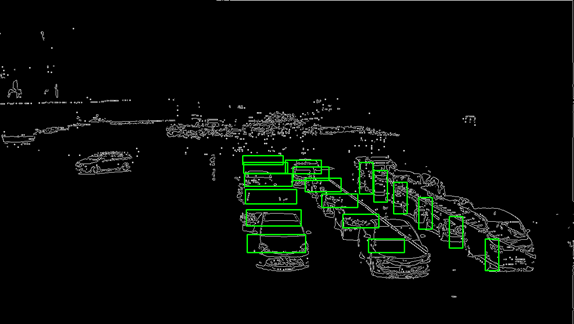


図4.5　車の台数計測を行う際の指定範囲

1. 実験
   1. 実験内容

　作成した画像処理プログラムの動作実験を行う．それぞれ違う状態の待機レーンの画像を51枚用意する．ここで利用する画像は，前述した別カメラで撮影した映像から，車の台数が大きく変化した場合に画像として保存したものである．この画像を，カメラから転送されてくることを想定し，手動で指定のディレクトリへ移動させる．ディレクトリに画像ができたかを判断し，プログラムが正しく動くかを調べる．またプログラムが正しく待機レーンの混雑状況を計測できているかを調べるために，出力結果と実際の待機レーンの状態を比較し，誤差を求める．

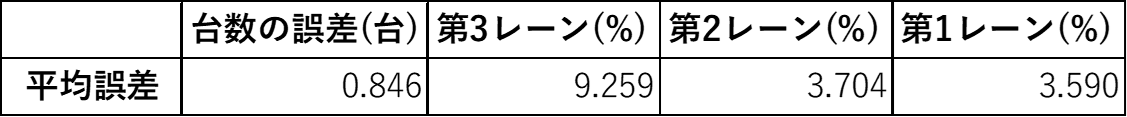
* 1. 結果

　プログラムは正しく動作し，51枚すべてにおいてエラーなく出力結果を返すことができた．そこで，画像ファイルが増えすぎて重くならないようにするために，画像ファイルを同じ名前にして上書きした場合は，プログラムが正しく動作しなかった．これは，watchdogライブラリが，画像が作成された時に動作するよう設定されているからだと考える．上書き保存では，ファイルは作成ではなく変更という形になる．そのため，現在のプログラムでエラーなく動作させるためには，画像の名前はユニークなものである必要がある．

* 1. 誤差

　出力結果と実際の待機レーンの状態を比較する．比較は，台数とレーンの埋まり具合に分ける．実際の待機レーンの埋まり具合は，現在停まっている車の台数を，実際に停まることができた台数で割って求める．例として車が4台停まっており，最終的には6台まで停まることができた場合には66.7%とする．同じく車が4台停まっていた場合でも，最終的に車が5台しか止まらなかった場合には80%として計算している．この結果と画像処理の出力結果の差を絶対値で表したものを誤差とする．その結果，51枚の画像の平均誤差は表5.3のようになった．この表から，第1レーンと第2レーンの誤差は5%未満であることが分かる．しかし，第3レーンの誤差は約10%と，第1・第2レーンに比べて誤差が大きくなってしまった．誤差が大きい理由は，第3レーンがカメラから見て正面に位置しているため，車間距離や車の高さなどの影響を受けやすいからだと考えられる．

表5.3　画像処理結果と実際の待機レーンの誤差



1. Webの表示部

画像処理で求めた結果をWeb上に表示することで，住民が待機レーンの混雑度を把握することが出来る．表示には，図6のように数値だけでなく棒グラフを用いて視覚的にわかりやすく表現している．画像処理の結果はjson形式でWeb上のサーバに送信される．これを読み取り，図のようなグラフを作成している．

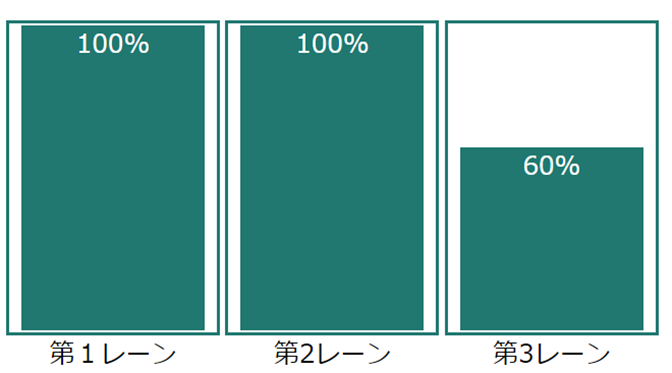


図6　Webで表示される棒グラフ

1. おわりに

　背景差分法とエッジ検出を用いて待機レーンの混雑量を計測することができるシステムを開発した．このデータをWeb上で公開し，住民が待機レーンの混雑具合を把握することができる．計測した混雑量の誤差は，第1・第2レーンで5%未満であり，十分実用が可能な範囲である．しかし，第3レーンの誤差は高くなった．理由は前述の通り，カメラの正面に位置していたからである．これを解決するために，現在はカメラの設置する位置を実験で撮影した場所よりも左側にすることが決まっている．これによって，第3レーンが他のレーンと似た状況になるため，この問題は解決すると考える．

また，このシステムだけでは混雑状況を改善することはできない．理由は，船に乗ろうと思ってから船に乗るまでに時間差があるからである．上島町の住民が船に乗ろうと思ってから港に到着するまでに，5分から10分はかかってしまう．これによって，出発するとき見た混雑状況と，港に着くときに見た混雑状況が大きく変わってしまい，混雑状況の信頼性が失われ，混雑を回避することができない．これを解決するためには，出発しようとしたときの混雑度ではなく，到着するときの混雑度を住民に知らせる必要がある．そこで，今回開発したシステムを利用し，曜日や時間帯ごとの混雑度を記録する．この記録した混雑度を利用し，混雑度を予測することができる仕組みを作る．この仕組みを利用して住民が到着するまでにかかる時間を考慮した予測混雑度を住民に伝えることで，この問題を解決することができる．

1. 参考文献

[1]watchdog:ファイル監視ライブラリ

<https://qh73xebitbucketorg.readthedocs.io/ja/latest/1.Programmings/python/LIB/watchdog/main/>

[2]Pythonによる写真画像の位置合わせ

<https://campkougaku.com/2020/04/15/alignment/>

[3]背景差分で物体抽出(画像比較)

<https://qiita.com/goodboy_max/items/71a3b804d14f961c0d91>

[4]canny法によるエッジ検出

<http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_canny/py_canny.html>

[5]エッジ検出

<https://axa.biopapyrus.jp/ia/opencv/edge.html>

[6]Python，OpenCVで二値画像から白と黒の面積比を算出

https://techtech-sorae.com/pythonopencvで二値画像から白と黒の面積比を算出/