空席情報配信システムの製作

Making of vacant seat information distribution system

高田順正[1,2]，原田理郁[1,2]，谷澤慶香[1,2]，佐藤勝治[2]

TAKADA Junsei[1,2]，HARADA Masafumi[1,2]，YAZAWA Keika[1,2]，and SATO Katsuharu[2]

東北大学　探求型「科学者の卵養成講座」[1], 山形県立山形東高等学校[2]

EGGS, Tohoku University[1]　Yamagatahigashi Senior High School[2]

Corresponding Author's e-mail: ssatokatsuh@yamagatahigashi.jp

(Received: 30 March 2021; Accepted: 19 July 2021; Released: 7 September2021)

［要約］

学習室などに行ったものの空席が無かったというような経験や、コロナ対策として混雑状況の把握が重要視されていることから空席検知に着目した。既存のシステムはコスト面で導入のハードルが高いことがわかったため、既存のものに比べ安価で手間が少なく、導入しやすいシステムを製作することを目標として研究を進めた。研究をはじめた当初は、空席検知の新たな手法として二値化を用いた空席検知について研究していた。目標に適した検知の手法について模索する中で、Raspberry Piとカメラを用いて、撮影した画像を解析しその結果をLINE BOTで配信するシステムを製作するに至った。ローカル環境でこのシステムを実行したところ、多少の誤差が生じたがシステムとして機能することがわかった。今後はシステムを製作し実際に動かしていく過程で得た多くの課題を解決していくとともに、より導入しやすいシステムとして完成させたい。

Sometimes we are not able to find empty seats, such as in the school study room. The COVID-19 epidemic requires us to avoid the crowd. Based on these experiences we take notice of vacant detection systems. Since we found that cost is a hurdle to the introduction of existing vacant seat detecting systems, we set a goal to make a system that is more affordable and easier to be implemented than the existing one. At first, we conducted a study about a new way to detect vacant seats by using the method for binarizing. However, we changed our plan in researching because we learned that using AI can make vacancy detection cheaper and easier, which fits our research goal. We made a trial product that detects vacant seats with a Raspberry Pi and a camera and notifies people of the information through LINE. We ran our program in the local environment. Its output was not always exact, but it works as a system. In the future, we want to resolve some problems such as improving detection accuracy and how to install devices used for vacancy detection and make it simple to implement. Ultimately, we want people to actually use this system.on through LINE.

［キーワード］空席検知，YOLO，機械学習，Raspberry Pi，LINE BOT

Vacant seat detection, YOLO, Machine learning , Raspberry Pi, LINE BOT

１．はじめに

自習室へ行ったのに空席が無く、無駄足になったという経験のある人は多いだろう。また、コロナウイルス感染症の流行により、混雑状況の把握はより重要となっている。これらのことから、現地に行かなくとも空席の有無がわかる仕組みの需要は高い。

そこで、既存の空席検知について調べた。株式会社ファンブライトの提供する座席やフリースペースの可視化をするIoTサービスの場合、1席毎に温度差を検知するセンサーをテーブル下に設置することで座席に人が居るか居ないかを検知する仕組みをとっていることがわかった。 [17]

また、「画像センサーによる教室の空席検知方法」の研究では、各席の真上に画像センサー（デジタルカメラ）やDepthセンサーを設置することで空席検知を行っている。 [18]

その他の既存の空席検知のついても各席ごとにセンサーを設置する方法をとるものが多くみられた。また、企業が提供する既存のサービスでは利用に毎月料金がかかったり、初期費用が高額であったりすることがわかった。

このようなサービスは導入のハードルが高いと感じたため、私たちはできるだけお金がかからない方法で空席情報を配信できるシステムを作りたいと考えた。

©2021 The EGGS program committee of Tohoku University

２．研究内容１

空席検知サービスとして主流なのは AI を活用した画像解析によるものである。そこでこの研究では、新たな手法として二値化を用いた物体検知の方法の提案を行った。処理は以下の順で行われる。

1.人がいない写真といる写真の差分画像の作成
2.差分画像の二値化
3.中央値フィルタによるノイズ除去
4.オブジェクト数のカウント（ラベリング処理）

1 では教室内に人が居る時といない時の写真を撮り、これらの写真をグレースケールにしたあと、そのグレースケール画像の差分をとった。

2 では大津の手法を用いて差分画像を二値化した。大津の二値化とは、横軸に画素値、縦軸に画素の個数をとったヒストグラムが双峰性を示す白黒画像において、その 2 つのピークの間の値を閾値として選ぶ方法である。[15]

3 では二値化処理を施した画像に中央値フィルタをかけ、ノイズを取り除いた。中央値フィルタとは、カーネル（画素の集まり）のすべての画素の中央値をそのカーネルの中心の画素（注目画素）とするものである。このフィルタを用いることで、ごま塩のような細かなノイズを取り除くことができる。カーネルサイズが大きければ大きいほど、フィルタの強度は強まる。[1][16]

4 ではノイズを取り除いた画像にラベリング処理を施し、画像の中の像をカウントした。ラベリング処理とは入力画像や、その中の像（オブジェクト）についての情報を得ることのできる処理である。簡易ラベリング処理と詳細ラベリング処理の二種類がある。簡易ラベリング処理では戻り値が入力画像に対して画像中の像の数（オブジェクト数）とラベリングされた画像であるのに対し、詳細ラベリング処理ではその２つに加えてそれぞれの像の座標や面積も戻り値に含まれる。

2.1 実験１

二値化を空席検知に応用する前に、二値化の基本的な性質と明るさが処理結果に与える影響について知りたいと考え、実験を行った。

2.1.1 実験方法

プログラミングには Python[10]と Open CV[9]を用いた。（これ以下に記載するすべての実験、製作も同様）

Python とはプログラム言語の一種である。また、OpenCV (Open Source Computer Vision Library) は、オープンソースのコンピュータビジョンおよび機械学習ソフトウェアライブラリである。

カメラを固定して静止画像をとり、1 と 2 の処理を行った後白黒を反転させた。3 と 4 の処理は行わなかった。カメラには iPhone8 を用いた。使用した画像を図 1、図３に、処理の結果を図 2、図 4 に示す。

図 1 は部屋内の蛍光灯をつけ、図 3 は蛍光灯を消して撮影した。

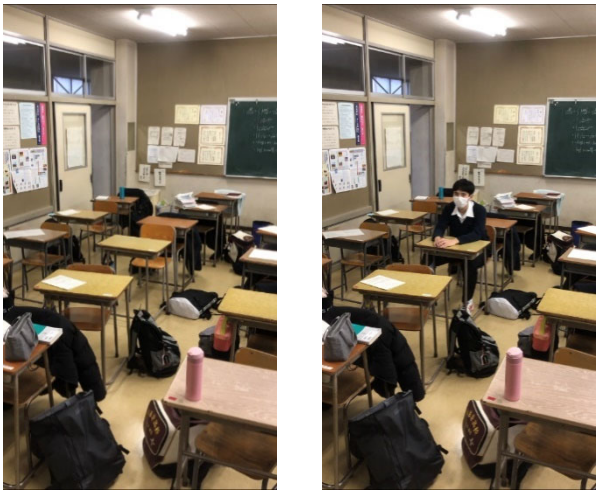

図１ 使用した写真
人が映っていないもの（左）と人が映っているもの（右）

図２ 図１の画像の差分を二値化したもの

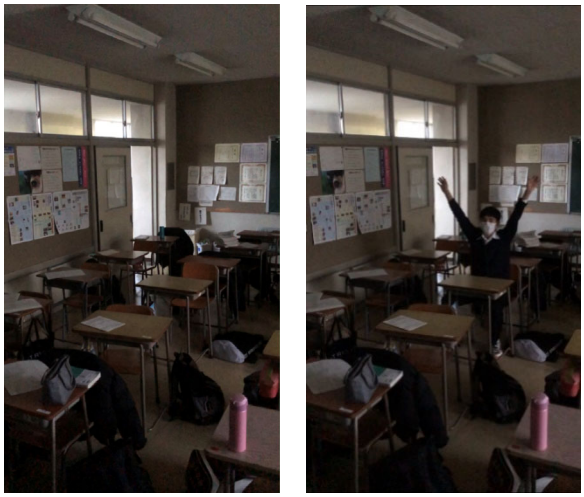

図3　使用した写真
人が映っていないもの（左）と人が映っているもの（右）

図4　図3の画像の差分を二値化したもの

2.1.2　結果と考察

教室内は生徒の荷物や壁に貼られた掲示物など物が多いにも関わらず、人を差分として取り出すことができている。しかしながら、人以外のもの（机など）を差分として取り出してしまってるほか、画像のところどころにノイズがみられる。図2と図4と比べると、図4では胴体部分の差分がうまく取り出せていない。実験では濃色の服を着ているため、部屋の明るさが明るいほうが差分を取り出すのに適していると考えられる。ノイズに関しては中央値フィルタを用いて取り除けると考えられる。

2.2　実験2

二値化を実際に空席検知に応用するため、実験を行った。同時に、大津の二値化と適応的閾値処理の比較も行った。

適応的閾値処理とは閾値を固定せず、それぞれの画素の閾値を算出することで行う方法である。具体的には、注目画素とその周辺の画素の平均値を注目画素の閾値とする。画像内の小領域ごとに異なる閾値を設けられるため、画像内の箇所によって明るさが変わる場合に適している。[1][5][15]

2.2.1　実験方法

図5と図6、図5と図7でそれぞれ差分を取り二値化を行った。その際、二値化の手法とノイズ除去の中央値フィルタのカーネルサイズを変え、3通りの処理を試した。図5と図6での結果を図8、図10、図12に、図5と図7での結果を図9、図11、図13に示した。

また、図8、図9では二値化に大津の手法を用い、ノイズ除去のためにカーネルサイズが7×7ピクセルの中央値フィルタをかけた。図10、図11では二値化に大津の手法を用い、カーネルサイズが15×15ピクセルの中央値フィルタをかけた。図12、図13では二値化に適応的閾値処理を用い、カーネルサイズが7×7ピクセルの中央値フィルタをかけた。

ソースコードはこの論文の最後に記載した。

図5　背景画像　人が映っていないもの

図6　人が映っている画像

図 7　人が映っている画像（図 6 とは人の位置が異なる）

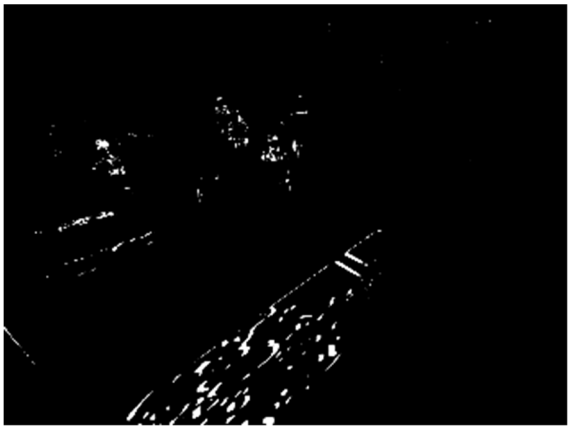

図 10　図 5 と図 6 の差分を大津の二値化で処理した後
カーネルサイズ 15 の中央値フィルタをかけた

2.2.2　結果と考察

処理後の画像を図 8 から図 13 に示す。

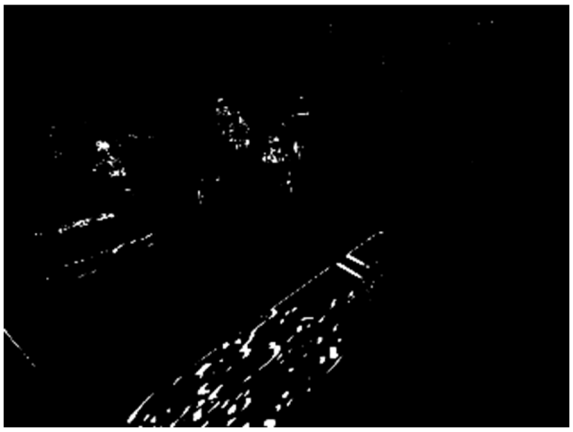

図 8　図 5 と図 6 の差分を大津の二値化で処理した後
カーネルサイズ 7 の中央値フィルタをかけた

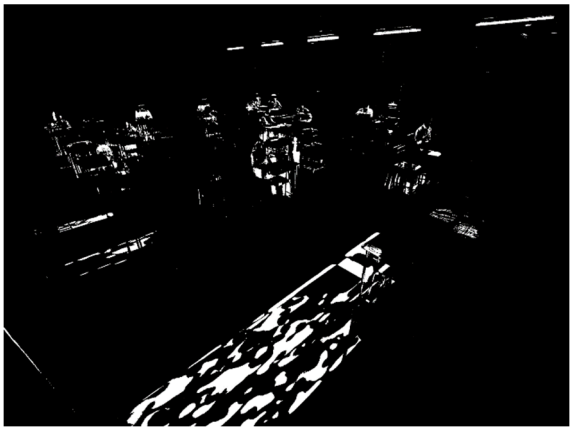

図 11　図 5 と図 7 の差分を大津の二値化で処理した後
カーネルサイズ 15 の中央値フィルタをかけた

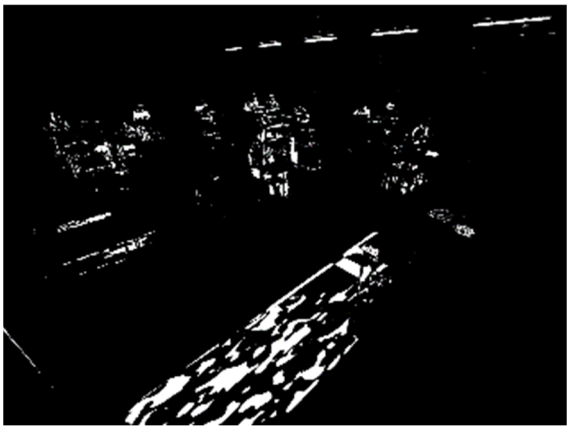

図 9　図 5 と図 7 の差分を大津の二値化で処理した後
カーネルサイズ 7 の中央値フィルタをかけた

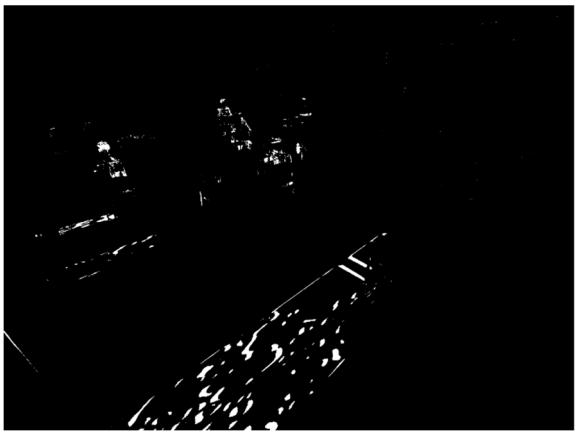

図 12　図 5 と図 6 の差分を適応的閾値処理で処理した後
カーネルサイズ 7 の中央値フィルタをかけた

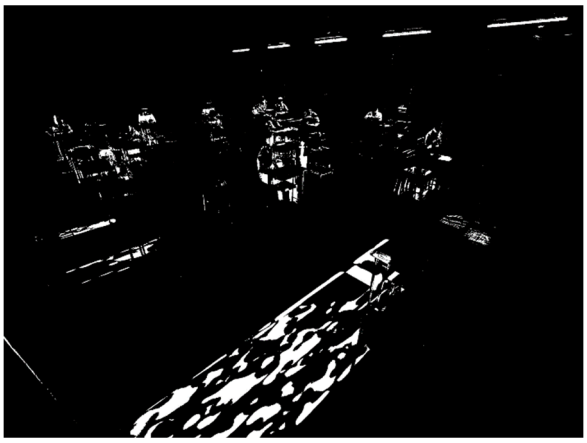

図 13　図 5 と図 7 の差分を適応的閾値処理で処理した後
カーネルサイズ 7 の中央値フィルタをかけた

二値化の種類、中央値フィルタのカーネルサイズに関わらず、図 8 から図 13 に共通して、明らかに人以外のオブジェクトが写っている。図 5 と図 7 の差分画像では窓側の列の蛍光灯が差分画像に出てしまっているため、蛍光灯の点灯などの人為的なミスが疑われる。また、日差しによる影の変化なども差分として取りだしてしまっていることから、二値化による手法は光による影響を受けやすいことが示唆される。加えて、人を一塊のオブジェクトとして取り出せていない。これでは正確な人数をカウントすることは難しい。

2.3　結論

実験から、二値化による空席検知には限界があるということがわかった。

そこで研究の方針を研究 1 での「新たな空席検知の手法、二値化を用いた空席検知の提案」から、「よりシンプルで多くの人が空席情報を受け取ったり配信したりできるようなシステムを製作する」へと変更した。

3．研究内容２

3.1　製作

3.1.1　システムの構成

考案した空席情報配信システム以下の通りである。

1. LINE BOT[6]を通じてリクエストを受信
2.Raspberry Pi[12]で写真を撮影
3.画像に映る人を YOLO[13]を使って検知、カウントする（図 14）
4.情報を Heroku[4]サーバーを経由して LINE BOT で配信

Raspberry Piとは小型のデュアルディスプレイ・デスクトップコンピュータ、YOLO とはリアルタイム物体検知アルゴリズムである。

また、Heroku とは、Ruby、Node.js、Java、Python、Clojure、Scala、Go、PHP で作成されたアプリケーションのデプロイ、実行、管理を行うことができるプラットフォームである。

3 で行う空席検知には YOLOv5[3]を利用した。図 12 は図 5 で人を検出した例だが、二値化を用いるよりも高い精度で人を検出出来ることがわかった。

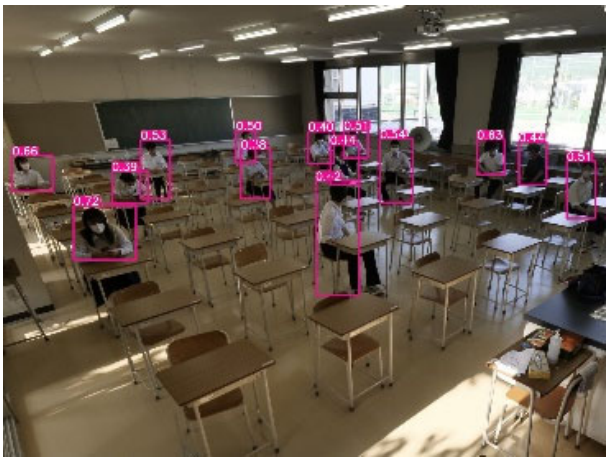

図 14　図 7 を YOLOv5 で人を検出した画像

また、3 の処理を Heroku サーバーで行う案（図 15）と Raspberry Pi で行う案（図 16）の２種類を考えた。

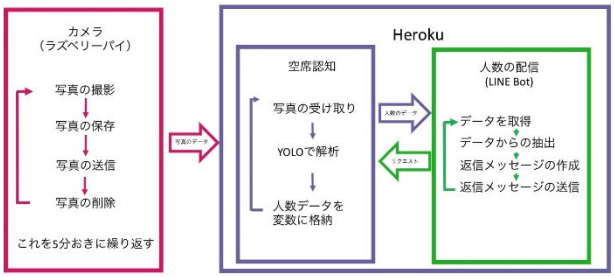

図 15　案１　YOLO を使った人の検知とカウントを Heroku で行う

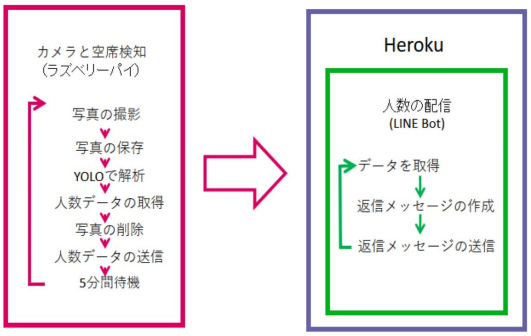

図 16　案２　YOLO を使った人の検知とカウントを Raspberry Pi で行う

Raspberry Pi4 Computer 8GB, Raspberry Pi Camera V2, Entaniya Raspberry Pi Camera 07L92, 電源としてモバイルバッテリーを用いて、これらのシステムを製作した。

3.1.2 結果と考察

考案した２種類の空席情報配信システム（図 15, 図 16）を実際に製作した。その結果 3 万 3000 円程度で製作できることが分かった。

しかし、案１と案２のシステムの両方で問題が起き、完成には至っていない。

案１の問題点

物体検知を Raspberry Pi で行う方法では、Raspberry Pi を YOLO に使う PyTorch[11]のライブラリである torchvison をインストールできていない。

案２の問題点

物体検知を Heroku で行う方法では、ストレージ容量が Heroku サーバーを無料で使う際の上限である 500MB を超えたためプログラムをデプロイできなかった。解決策として、プログラムの大半を占めている PyTorch を CPU のみのものにする事で容量を減らすことができるようだ。[14]

3.2 実験

3.2.1 実験方法

システムを構成するパートが互いに連携して動き、空席情報を配信できるかどうかを確かめるため、物体検知を Heroku 上で行う案（図 15）で実験を行った。

3.2.1 に記載したとおりシステムを Heroku にデプロイできなかったため、ローカル PC 上でサーバーを立てられる Ngrok[8]を用いてローカル環境で模擬的にシステムを動かした。

実験は Raspberry Pi4 Computer 8GB, Raspberry Pi Camera V2, Entaniya Raspberry Pi Camera 07L92, モバイルバッテリー, 三脚を用いて行った。

教室の隅に Raspberry Pi、カメラ、バッテリーを三脚で固定し(図 17)、教室内の人が 5 分おき（空席情報の更新のたび）に席を移動した。

また、[7]を参考にして、「図書室」と LINE BOT にメッセージを送ると空席情報がテキストメッセージで返ってくるように設定した。

図 17 Raspberry Pi、カメラ、バッテリーを固定した三脚

3.2.2 結果と考察

図 18 実際に空席情報が配信された LINE BOT の画面

教室内の空席情報を LINE BOT で配信することができた。5 分ごとに情報が更新されることも確認できたが、人数に誤差が出てしまった。（図 18）

今回の実験では三脚を用いて機材を設置したが、場所を取るためこの方法は適切ではないと考えられる。同様に、機材の電源としてモバイルバッテリーを用いたが、プラグを用いてコンセントから電源を得られるように改善すべきだと感じ

た。

また、この実験はシステムが想定通りに機能するかを確認するにとどまっている。よって、次の実験では人を検知した画像を保存し人の座った位置を記録することで、検知精度を検証したい。

3.3 結論

Raspberry Pi とカメラを用いて空席を検知し、LINE BOT で空席情報を配信するシステムを製作・実験した。その結果、既存の空席検知システムの多くより安価に製作できることが分かった。しかしながら、考案した二種類のどちらにおいても、問題が生じ完成させることはできていない。また、機材の設置方法や、電源を得る方法にも改善の余地がある。

4 今後の展望

4.1 研究内容１について

方針変更後、YOLO を用いた物体検知等を行いプログラミングや画像処理についての知識を得ていくにつれ、二値化による空席検知の改善点が見えてきた。

まず、簡易ラベリング処理ではなく詳細ラベリング処理を行い、オブジェクトの面積を得る事で小さなオブジェクトを取り除いたり、人を一塊のオブジェクトにまとめたりすることができる。その結果、精度を向上させることができると考えられる。

また、人として認識したオブジェクトを枠で囲む処理を施すことで、検知の状況を可視化でき、研究しやすくなる。得た知識をもって再チャレンジしたい。

4.2 研究内容２について

3.2 で挙げた２つの問題を解決し、システムを完成させる。その後、学校の図書室、教室などで実験的に動かしたいと考えている。また、物体検知をする時の変数を調節したり、短時間のうちに複数回検知を行ったりすることで、立って歩いている人を省くような仕組みを作って、検知の精度を向上させる。その際、検知の精度の検証・計測も行いたい。また、プログラムなどに関する知識のない人でも利用しやすいものにする。具体的には、Raspberry Pi の機能を、スマートフォンアプリで代用し、LINE BOT のコードを GitHub[2]にアップロードしておく。こうすることで、スマートフォンにアプリをダウンロードし、Heroku サーバーを用意して GitHub にアップロードされた LINE BOT のコードを Heroku にデプロイするだけでシステムを導入できるようになる。GitHub とはプログラムコードを公開、共有できるサービスである。

将来的には、空席情報のデータベース化を行い、「空席予測システム」として発展させたい。

謝辞

本研究は東北大学 探求型「科学者の卵養成講座（JST グローバルサイエンスキャンパス）」の支援のもとで実施されました。研究を進めるに際し、多大なご協力をいただいた東北大学工学部 高橋宏輔様 高橋佑輔様、山形県立産業技術短期大学 間宮明様 ありがとうございました。

引用及び参考文献

[1] Alexander Mordvintsev, Abid Rahman K (2013). OpenCV-Python Tutorials Documentation. https://opencv-python-tutroals.readthedocs.io/_/downloads/en/latest/pdf/

[2] GitHub. https://github.co.jp/

[3] glenn-jocher (2021). yolov5. https://github.com/ultralytics/yolov5

[4] Heroku. https://jp.heroku.com

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2015). You Only Look Once: Unified, Real-Time Object Detection

[6] LINE BOT. https://developers.line.biz/ja

[7] louis70109 (2020). line-bot-sdk-python. https://github.com/line/line-bot-sdk-python/blob/master/examples/flask-kitchensink/app.py

[8] ngrok. https://ngrok.com

[9] OpenCV. https://opencv.org

[10] Python. https://www.python.org/

[11] PyTorch. https://pytorch.org

[12] Raspberry Pi. https://www.raspberrypi.org

[13] YOLO. https://pjreddie.com/darknet/yolo/

[14] @nori2711 (2019). Heroku で機械学習モデルを利用したアプリのデプロイ時の 500MB 制限回避策. https://qiita.com/nori2711/items/615183fdc6858758d380

[15] 小山田 雄仁(2016a). 画像のしきい値処理. http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

[16] 小山田 雄仁(2016b). 画像の平滑化. http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html

[17] 株式会社ファンブライト(2019). 座席やフリースペースの可視化. https://www.fanbright.jp/wp-content/uploads/2019/08/IoT_Presence_201908.pdf

[18] 嶋 好博、河合 勇輝(2015）画像センサーによる教室の空席検知方法.PC Conference

2.2 で用いたソースコード

二値化に大津の手法を用いたもの

```
#「背景」となる画像の取り込み（グレースケール）
img_src01 = cv2.imread(".JPG", 0)

#「差分」をもった画像の取り込み（グレースケール）
img_src02 = cv2.imread(".JPG", 0)

#「背景差分」計算用オブジェクトの作成
bgObj = cv2.bgsegm.createBackgroundSubtractorMOG()



#差分となっている「前景領域」に対してマスクをかける
fgmask = bgObj.apply(img_src01)
fgmask = bgObj.apply(img_src02)

#「差分」画像のファイル名
bg_diff_path = "./diff5.JPG"
 #「差分」画像の保存
cv2.imwrite(bg_diff_path, fgmask)

#中央値フィルタ
img = cv2.imread("./diff.JPG")
ksize=7
img_mask = cv2.medianBlur(img,ksize)
plt.imshow(img_mask)

bg_obj_path = "./obj.JPG"

cv2.imwrite(bg_obj_path,img_mask)

# 画像の読み込み
img = cv2.imread('obj.JPG')

# グレースケール化
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


# 大津の二値化
gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

# 白黒反転
gray = cv2.bitwise_not(gray)

# ラベリング処理（簡易版）
n, label = cv2.connectedComponents(gray)

# ラベリング結果書き出し準備
color_src = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
height, width = gray.shape[:2]
colors = []

for i in range(1, n + 1):
    colors.append(np.array([random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)]))

# ラベリング結果を表示
# 各オブジェクトをランダム色でペイント
for y in range(0, height):
    for x in range(0, width):
        if label[y, x] > 0:
            color_src[y, x] = colors[label[y, x]]
        else:
            color_src[y, x] = [0, 0, 0]

# オブジェクトの総数を黄文字で表示
cv2.putText(color_src, str(n - 1), (70, 70), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 255))

# 画像の保存
cv2.imwrite('rab1.JPG', color_src)
```

二値化に適応的閾値処理を用いたもの

```
#「背景」となる画像の取り込み（グレースケール）
img_src01 = cv2.imread(".JPG", 0)

#「差分」をもった画像の取り込み（グレースケール）
img_src02 = cv2.imread(".JPG", 0)


#「背景差分」計算用オブジェクトの作成
bgObj = cv2.bgsegm.createBackgroundSubtractorMOG()

#差分となっている「前景領域」に対してマスクをかける
fgmask = bgObj.apply(img_src01)
fgmask = bgObj.apply(img_src02)

#{差分」画像のファイル名
bg_diff_path = "./diff.JPG"

#「差分」画像の保存
cv2.imwrite(bg_diff_path, fgmask)

img = cv2.imread("./diff.JPG")

#中央値フィルタ
ksize=7
img_mask = cv2.medianBlur(img,ksize)
plt.imshow(img_mask)

bg_obj_path = "./obj.JPG"

cv2.imwrite(bg_obj_path,img_mask)

# 画像の読み込み
img = cv2.imread('obj.JPG')

# グレースケール化
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 適応的閾値処理
from matplotlib import pyplot as plt

img = cv2.imread('obj.JPG',0)
img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,¥
        cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,¥
        cv2.THRESH_BINARY,11,2)

images = [img, th1, th2, th3]
```

```
#画像のファイル名
bg_diff_path_ = "./diff.JPG"

#画像の保存
cv2.imwrite(bg_diff_path_, fgmask)

# 白黒反転
gray = cv2.bitwise_not(gray)

# ラベリング処理(簡易版)
n, label = cv2.connectedComponents(gray)

# ラベリング結果書き出し準備
color_src = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
height, width = gray.shape[:2]
colors = []

for i in range(1, n + 1):
    colors.append(np.array([random.randint(0, 255), random.ran
dint(0, 255), random.randint(0, 255)]))

# ラベリング結果を表示
# 各オブジェクトをランダム色でペイント
for y in range(0, height):
    for x in range(0, width):
        if label[y, x] > 0:
            color_src[y, x] = colors[label[y, x]]
        else:
            color_src[y, x] = [0, 0, 0]

# オブジェクトの総数を黄文字で表示
cv2.putText(color_src, str(n - 1), (70, 70), cv2.FONT_HERSH
EY_PLAIN, 1, (0, 255, 255))

# 画像の保存
cv2.imwrite('rab.JPG', color_src)
```