2025年度　卒業論文

IoTを用いた無人店舗における遠隔在庫把握システムの構築

Development of a Remote Inventory Monitoring System for Unmanned Retail Stores Using IoT

成蹊大学 理工学部 理工学科 機械システム専攻
流体力学研究室

S226109　前澤 柊飛
S226003　秋元 大生

指導教授：小川 隆申

提出日 2026年 1月 23日

記号一覧

目次

記号一覧 **i**

第1章　序論 **1**

1.1　研究背景 …… 1

1.2　研究目的 …… 1

第2章　解析手法 **2**

2.1　サーマルカメラ …… 2

第3章　超音波センサによる在庫数の把握 **3**

3.1　超音波センサによる在庫数把握の方法 …… 3

3.2　研究に使用した装置 …… 4

3.3　超音波センサの配置と距離の検出方法 …… 4

3.4　超音波センサの試行の結果と考察 …… 4

第4章　デプスカメラによるデータ収集 **5**

4.1　使用するデプスカメラの説明 …… 5

4.2　デプスカメラの配置と距離の検出方法 …… 6

第5章　結論 **7**

第6章　結論 **8**

第7章　総合的考察 **9**

7.1　本研究の限界と今後の課題 …… 9

参考文献 **10**

謝辞 **11**

付録A　プログラム **1**

A.1　ソースコード …… 1

付録B　原稿非掲載データ **4**

B.1　実験結果 …… 4

第1章　序論

1.1　研究背景

研究背景である。

1.2　研究目的

研究目的である。（図1.1）

図1.1: 学生食堂　行列の様子

第2章　解析手法

2.1　サーマルカメラ

本研究では、混雑状況の測定に図 2.1 のサーマルカメラで撮影された赤外線画像を用いる。このサーマルカメラは MLX90640 赤外線アレイモジュールを搭載した M5Stack 用赤外線画像ユニットであり、小型のマイコンボードである M5Stack に接続することにより撮影を行うことができる。

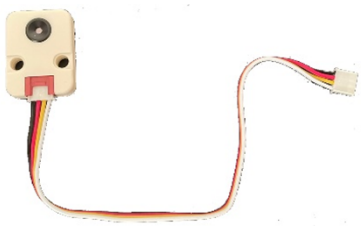

図 2.1: Melexis 社　MLX90640

2.1.1　サーマルカメラの精度

MLX90640 の性能を表 2.1 に示す。測定範囲は-40℃～+300℃であり、混雑状況のモニタリングの際に観測される温度は測定可能であると考えられる。

表 2.1: MLX90640 の精度

動作電圧	3～3.6V
消費電流	23mA
視野	110°×75°
測定範囲	-40℃～+300℃
分解能	±1.5℃
リフレッシュレート	0.5Hz～64Hz
解像度	32×24 ピクセル

第3章　超音波センサによる在庫数の把握

本章では、本研究で行った超音波センサによる在庫数の把握の手段について述べる。

3.1　超音波センサによる在庫数把握の方法

本研究では、無人店舗内の商品棚における在庫数の把握を超音波センサを用いることによって行った。商品棚の奥側に超音波センサを設置し、センサから発信された超音波が商品に反射して戻ってくるまでの時間を距離に計算し直すことで行う。距離は以下の式で求められる。

$$\text{距離} = \frac{\text{音速} \times \text{時間}}{2} \tag{3.1}$$

ここで、音速は約 343m/s である。この距離の長短で在庫の状況を判断する。使用したセンサは図 3.1 に示す Rainbow E-Technology 社の HC-SR04 である。このセンサは単体では動作しないため、マイコンボードと組み合わせて使用する必要がある。マイコンボードには Raspberry Pi Pico 2 WH を使用した。

図 3.1: Rainbow E-Technology 社 HC-SR04

3.2　研究に使用した装置

3.2.1　超音波センサの性能

HC-SR04 の性能を表 3.1 に示す。測定可能距離は 0.02m〜4.5m であり、商品棚での在庫検出は可能であると考えられる。

表 3.1: HC-SR04 の精度

動作電圧	3〜5.5V
測定可能距離	0.02〜4.5m
測定方式	超音波
動作温度	-10〜70℃

3.2.2　マイコンボードを含めた装置全体

前述の通り、この超音波センサは単体では動作しないためマイコンボードと組み合わせる必要がある。

3.3　超音波センサの配置と距離の検出方法

超音波センサを 11 号館 1 階の無人店舗の商品棚の奥側のスペースに配置し、商品との距離を計測する。距離の検出方法は、超音波がセンサから発信され商品に反射して戻ってくるまでの時間を計測することで行う。距離は以下の式で求められる。

$$\text{距離} = \frac{\text{音速} \times \text{時間}}{2} \tag{3.2}$$

ここで、音速は約 343m/s である。

3.4　超音波センサの試行の結果と考察

今回は、何センチのとき在庫あり、何センチのとき在庫なしと設定した。だが、商品と棚の手前部分にスペースがある場合、正確に距離を計測できない。そのため、本研究では超音波センサ単体での在庫把握は困難であると判断した。

第4章　デプスカメラによるデータ収集

本章では、本研究で行ったデプスカメラによる在庫数の把握について述べる。

4.1　使用するデプスカメラの説明

本研究では、超音波センサだけでなくデプスカメラも用いて無人店舗内の在庫数の把握を行った。使用したデプスカメラはDFRobot社のSEN0581とArducam社のB0410である。

図4.1: DFRobot社 SEN0581

図4.2: Arducam社 B0410

4.1.1　デプスカメラの精度

これらのデプスカメラの性能をに示す。

表 4.1: D435 の精度

動作電圧	5V
消費電流	150mA
視野	87°×58°
測定範囲	0.1m〜10m
深度分解能	1mm

4.2　デプスカメラの配置と距離の検出方法

第5章　結論

サーマルカメラを用いた混雑状況のモニタリング手法を開発するために、輪郭抽出及び機械学習を用いて赤外線画像内の人数を数えるシステムを開発した。モデルケースとして想定する成蹊大学の第一学生食堂での撮影状況を考慮し、研究室内での赤外線画像の撮影を行った。輪郭抽出による赤外線画像内の人数把握により撮影範囲内の混雑状況を推定することを試みたが、ノイズや人以外の熱源等の輪郭を抽出してしまい、正確に人数を把握することができない。従って、別の手法として機械学習を用いることにより、画像の特徴からの人数把握を試みた。機械学習では画像処理に適した CNN を使用し、0〜3 人を撮影した計 130 枚の赤外線画像を用いてモデルを作成し、学習を行った。作成したモデルを用いて赤外線画像内の人数予測を行った。

機械学習による人数予測の正解率は 94.87%となった。また、適合率、再現率、F1 スコアの評価指数もそれぞれ90%を超えたため、モデルの正確性が高いと考えられる。しかし、今回行った人数把握実験で使用した赤外線画像は研究室内で撮影したものであり、3 人以上の場合や人が重なっている場合は考慮されていない。また、食堂への設置は長期間を想定しているが、季節の変化等による温度変化の影響ついても考慮されていない。そのため、実際に食堂へ測定システムを設置し、長期的に撮影を行うことで多人数や人が重なった場合等の多様なデータを用いてモデルの精度や対応力を向上させる必要がある。

第6章　結論

結論である。

第7章　総合的考察

本章では、第3章で実施した超音波センサによる在庫把握と、第4章で実施したデプスカメラによる在庫把握、

7.1　本研究の限界と今後の課題

本節では、本研究における限界を明示するとともに、今後の課題について述べる。

7.1.1　本研究の課題

7.1.1.1　超音波センサに関する課題

まず、超音波センサでの在庫数把握において、商品が購入されたあと、在庫が一部の小売店では、に

参考文献

[1] 西海 大愛・川合 康央：『ライブカメラを用いた人流可視化手法の提案』, エンタテインメントコンピューティングシンポジウム, pp341-344, 2023.

謝辞

謝辞である。

付録A　プログラム

A.1　ソースコード

以下はソースコードである。超音波センサでの在庫数把握において、マイコンボードで実行したプログラムがA.1である。

ソースコード A.1: findContours

```
from machine import Pin
import machine
import utime

hasshin = Pin(14, Pin.OUT) #14番を出力（発 信）と定義
jyushin = Pin(15, Pin.IN) #15番を入力（受 信）と定義

def read_distance():
    hasshin.low() #9行から13行までのコードで超音波を一瞬だけ発信
    utime.sleep_us(2)
    hasshin.high()
    utime.sleep_us(10)
    hasshin.low()
    while jyushin.value() == 0: #超音波が受信されていないとき
        start = utime.ticks_us() #現時点での稼働時間をマイクロ秒単位で返す
    while jyushin.value() == 1: #超音波が受信されているとき
        goal = utime.ticks_us() #現時点での稼働時間をマイクロ秒単位で返す
    #超音波は一瞬なの で、jyushin.value()==0とな り、無限ループにはならない
    passed = goal - start
    distance = float((passed * 340 * 0.0001) / 2) #cmで出力
    print(distance)

while True:
   read_distance()
   utime.sleep(1)
```

超音波センサA.2に機械学習に用いた一連のプログラムを示す。

ソースコード A.2: MachineLearning

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Dropout, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import array_to_img, img_to_array, load_img
from tensorflow.keras.optimizers import Adagrad
```

```
8  from tensorflow.keras.optimizers import Adam
9  from sklearn.model_selection import train_test_split
10 import pandas as pd
11 import numpy as np
12 from PIL import Image
13 import os
14 import re
15 import matplotlib.pyplot as plt
16
17 #画像を読み込む
18 def list_pictures(directory, ext='jpg|jpeg|bmp|png|ppm'):
19     return [os.path.join(root, f)
20             for root, _, files in os.walk(directory) for f in files
21             if re.match(r'([\w]+.(?:' + ext + '))', f.lower())]
22
23 X = []
24 Y = []
25 #0人の画像
26 for picture in list_pictures('./train_picture/zero/'):
27     img = img_to_array(load_img(picture, target_size=(24,32)))
28     X.append(img)
29     Y.append(0)
30 #1人の画像
31 for picture in list_pictures('./train_picture/one/'):
32     img = img_to_array(load_img(picture, target_size=(24,32)))
33     X.append(img)
34     Y.append(1)
35 #2人の画像
36 for picture in list_pictures('./train_picture/two/'):
37     img = img_to_array(load_img(picture, target_size=(24,32)))
38     X.append(img)
39     Y.append(2)
40 #3人の画像
41 for picture in list_pictures('./train_picture/three/'):
42     img = img_to_array(load_img(picture, target_size=(24,32)))
43     X.append(img)
44     Y.append(3)
45 # arrayに変換
46 X = np.asarray(X)
47 Y = np.asarray(Y)
48 # 画素値を0から1の範囲に変換
49 X = X.astype('float32')
50 X = X / 255.0
51 # クラスの形式を変換
52 Y = to_categorical(Y)
53 # 学習用データとテストデータに分ける
54 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=111)
55
56 # モデルの構築
57 model = Sequential()
58 model.add(keras.layers.Conv2D(16, (3, 3), padding='same',input_shape=X_train.shape[1:])) # 畳み
       込み層
59 model.add(Activation('relu')) # 活性化関数
60 model.add(keras.layers.MaxPooling2D(pool_size=(2, 2))) # プーリング層
```

```
61
62  model.add(keras.layers.Conv2D(32, (3, 3), padding='same'))
63  model.add(Activation('relu'))
64  model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
65
66  model.add(keras.layers.Conv2D(64, (3, 3), padding='same'))
67  model.add(Activation('relu'))
68  model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
69
70  model.add(Flatten()) # 二次元データを一次元データに変換
71  model.add(Dense(512)) # 全結合層
72  model.add(Activation('relu'))
73  model.add(Dropout(0.5))
74  model.add(Dense(4))       # クラスは4個
75  model.add(Activation('softmax')) # ソフトマックス関数
76
77  # モデルをコンパイル
78  model.compile(loss="categorical_crossentropy", # 損失関数 交差エントロピー誤差
79                optimizer='adam', # 最適化アルゴリズム ADAM
80                metrics=["accuracy","Precision","Recall","F1Score"]) # 評価指数 正答率、適合率、再
                      現率、F1Score
81  es_cb=keras.callbacks.EarlyStopping(monitor='val_loss',mode='auto')# EarlyStopping
82
83  # 学習を実行
84  history=model.fit(X_train, y_train, epochs=200,validation_data = (X_test, y_test),callbacks=
        es_cb)
85
86  # テストデータを予測
87  predict_prob=model.predict(X_test)
88  predict_classes=np.argmax(predict_prob,axis=1)
89
90  # 混合行列の作成
91  mg_df = pd.DataFrame({'predict': predict_classes,
92                        'class': np.argmax(y_test, axis=1)})
93  pd.crosstab(mg_df['class'], mg_df['predict'])
94
95  # 評価指数を表示
96  score = model.evaluate(X_test,y_test,verbose=1)
97  print('Test␣loss:',score[0])
98  print('Test␣Accuracy:', score[1])
99  print('Test␣Precision:', score[2])
100 print('Test␣Recall:', score[3])
101 print('Test␣F1Score:', score[4])
102
103 pd.crosstab(mg_df['class'], mg_df['predict'])
```

,

付録B　原稿非掲載データ

B.1　実験結果

例えばここに本文中に載せられなかった実験結果などを載せる.