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 2025年 12月 18日

記号一覧

目次

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。

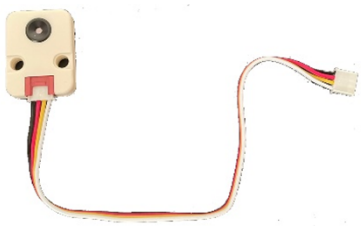

図 2.1: Melexis 社　MLX90640

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　結果

結果である。

第4章　結論

結論である。

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　結論

結論である。

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す．A.1 に輪郭抽出に用いた一連のプログラムを示す。

ソースコード A.1: findContours

```
import matplotlib.pyplot as plt     # pip install matplotlib
import numpy as np                  # pip install numpy
import os
import glob
import cv2

# データの保存先を作成
input_dir_path = "./input"
os.makedirs(input_dir_path, exist_ok=True)
output_dir_path = "./output"
os.makedirs(output_dir_path, exist_ok=True)
os.makedirs(output_dir_path + "/thermalmap", exist_ok=True)

def CountContours(impath):
    cvimg=cv2.imread(impath)
    bigimg=cv2.resize(cvimg,None,fx=20,fy=20, interpolation=cv2.INTER_NEAREST)
    grayimg=cv2.cvtColor(cvimg,cv2.COLOR_BGR2GRAY)
    blurimg=cv2.blur(grayimg,(3,3)) #平滑化
    blurimg=cv2.resize(blurimg,None,fx=20,fy=20, interpolation=cv2.INTER_NEAREST)
    ret,dst = cv2.threshold(blurimg,0,255,cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    contours,hierarchy=cv2.findContours(dst,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)    #輪郭
        抽出
    img_result = cv2.drawContours(bigimg, contours, -1, (0, 0, 255), 2)
    print("輪郭数："+str(len(contours)))
    cv2.imshow("IMAGE", img_result) #輪郭を描きこんだ画像を表示
    cv2.waitKey()



for f in glob.glob(input_dir_path + "/*.csv"):  #赤外線画像データ読み込み
    csv_file_name = os.path.split(f)[1]
    date_str = csv_file_name[:-4]
    thermal_data = np.loadtxt(input_dir_path + '/' + csv_file_name, delimiter=',')
    fig, ax = plt.subplots(figsize=thermal_data.shape[::-1],dpi=1, tight_layout=True)
    ax.imshow(thermal_data, cmap="gist_gray")    #グレースケール：gist_gray カラー:jet
    ax.axis("off")
    fig.savefig(output_dir_path + "/thermalmap/" + date_str + ".png",dpi=1)
    plt.close()
```

```
38      impath=output_dir_path + "/thermalmap/" + date_str + ".png"
39      CountContours(impath)
```

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

```
1  import tensorflow as tf
2  from tensorflow import keras
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.layers import Activation, Dense, Dropout, Flatten
5  from tensorflow.keras.utils import to_categorical
6  from tensorflow.keras.preprocessing.image import array_to_img, img_to_array, load_img
7  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)
```

```
# 画素値を0から1の範囲に変換
X = X.astype('float32')
X = X / 255.0
# クラスの形式を変換
Y = to_categorical(Y)
# 学習用データとテストデータに分ける
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=111)

# モデルの構築
model = Sequential()
model.add(keras.layers.Conv2D(16, (3, 3), padding='same',input_shape=X_train.shape[1:])) # 畳み込み層
model.add(Activation('relu')) # 活性化関数
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2))) # プーリング層

model.add(keras.layers.Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(keras.layers.Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten()) # 二次元データを一次元データに変換
model.add(Dense(512)) # 全結合層
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))       # クラスは4個
model.add(Activation('softmax')) # ソフトマックス関数

# モデルをコンパイル
model.compile(loss="categorical_crossentropy", # 損失関数 交差エントロピー誤差
              optimizer='adam', # 最適化アルゴリズム ADAM
              metrics=["accuracy","Precision","Recall","F1Score"]) # 評価指数 正答率、適合率、再現率、F1Score
es_cb=keras.callbacks.EarlyStopping(monitor='val_loss',mode='auto')# EarlyStopping

# 学習を実行
history=model.fit(X_train, y_train, epochs=200,validation_data = (X_test, y_test),callbacks=es_cb)

# テストデータを予測
predict_prob=model.predict(X_test)
predict_classes=np.argmax(predict_prob,axis=1)

# 混合行列の作成
mg_df = pd.DataFrame({'predict': predict_classes,
                      'class': np.argmax(y_test, axis=1)})
pd.crosstab(mg_df['class'], mg_df['predict'])

# 評価指数を表示
score = model.evaluate(X_test,y_test,verbose=1)
print('Test␣loss:',score[0])
print('Test␣Accuracy:', score[1])
```

```
print('Test Precision:', score[2])
print('Test Recall:', score[3])
print('Test F1Score:', score[4])

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.