

2025年度 卒業論文

論文フォーマット

日本語タイトル

Thesis Format
English Title

成蹊大学 理工学部 システムデザイン学科
流体力学研究室

M222221 今井 直幸
M222222 今井 直太郎

指導教授：小川 隆申

提出日 2025年 12月 9日

記号一覧

- v : 速度
- p : 圧力
- ν : 動粘性係数
- μ : 粘性係数

目次

| | |
|--------------|---|
| 記号一覧 | i |
| 第1章 序論 | 1 |
| 1.1 研究背景 | 1 |
| 1.2 既往の研究 | 1 |
| 1.3 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 結果 | 3 |
| 第4章 結論 | 4 |
| 参考文献 | 5 |
| 謝辞 | 6 |
| 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 | 5 |
| B.1 実験結果 | 5 |

第1章 序論

1.1 研究背景

研究背景である。

1.2 既往の研究

既往の研究である。^[1]

1.3 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。

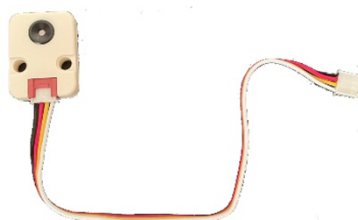


図 2.1: Melexis 社 MLX90640

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°C ～ $+300^{\circ}\text{C}$ 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 | $110^{\circ}\times 75^{\circ}$ |
| 測定範囲 | -40°C ～ $+300^{\circ}\text{C}$ |
| 分解能 | $\pm 1.5^{\circ}\text{C}$ |
| 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。

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

```
1 import matplotlib.pyplot as plt      # pip install matplotlib
2 import numpy as np                  # pip install numpy
3 import os
4 import glob
5 import cv2
6
7 # データの保存先を作成
8 input_dir_path = "./input"
9 os.makedirs(input_dir_path, exist_ok=True)
10 output_dir_path = "./output"
11 os.makedirs(output_dir_path, exist_ok=True)
12 os.makedirs(output_dir_path + "/thermalmap", exist_ok=True)
13
14 def CountContours(impath):
15     cimg=cv2.imread(impath)
16     bigimg=cv2.resize(cimg,None,fx=20,fy=20, interpolation=cv2.INTER_NEAREST)
17     grayimg=cv2.cvtColor(cimg,cv2.COLOR_BGR2GRAY)
18     blurimg=cv2.blur(grayimg,(3,3)) #平滑化
19     blurimg=cv2.resize(blurimg,None,fx=20,fy=20, interpolation=cv2.INTER_NEAREST)
20     ret,dst = cv2.threshold(blurimg,0,255,cv2.THRESH_BINARY + cv2.THRESH_OTSU)
21     contours,hierarchy=cv2.findContours(dst,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE) #輪郭抽出
22     img_result = cv2.drawContours(bigimg, contours, -1, (0, 0, 255), 2)
23     print("輪郭数:"+str(len(contours)))
24     cv2.imshow("IMAGE", img_result) #輪郭を描きこんだ画像を表示
25     cv2.waitKey()
26
27
28
29 for f in glob.glob(input_dir_path + "/*.csv"): #赤外線画像データ読み込み
30     csv_file_name = os.path.split(f)[1]
31     date_str = csv_file_name[:-4]
32     thermal_data = np.loadtxt(input_dir_path + '/' + csv_file_name, delimiter=',')
33     fig, ax = plt.subplots(figsize=thermal_data.shape[:,-1],dpi=1, tight_layout=True)
34     ax.imshow(thermal_data, cmap="gist_gray") #グレースケール:gist_gray カラー:jet
35     ax.axis("off")
36     fig.savefig(output_dir_path + "/thermalmap/" + date_str + ".png",dpi=1)
37     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)
```

```

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=
85                   es_cb)
86
87 # テストデータを予測
88 predict_prob=model.predict(X_test)
89 predict_classes=np.argmax(predict_prob,axis=1)
90
91 # 混合行列の作成
92 mg_df = pd.DataFrame({'predict': predict_classes,
93                       'class': np.argmax(y_test, axis=1)})
94 pd.crosstab(mg_df['class'], mg_df['predict'])
95
96 # 評価指数を表示
97 score = model.evaluate(X_test,y_test,verbose=1)
98 print('Test Loss:',score[0])
99 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。