,

2025年度　卒業論文

画像処理およびセンサを用いた遠隔在庫把握システムの構築

Development of a Remote Inventory Monitoring System Using Image Processing and Sensors

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

目次

第1章　序論

1.1　研究背景

近年，様々な技術の発達に伴い，人々の生活はより快適になっている．その技術として，デジタル技術が挙げられる．これには，家電製品や自動車，センサ，カメラなどの様々な「モノ」がインターネットに接続し，相互にデータをやり取りするIoT（Internet of Things）[?]，感知器や測定器などを用いて対象の定量的な情報を取得するセンシング[?]，コンピューターに人間の知的活動を模倣させるAI（Artificial Intelligence）などが該当し[?]，農業，製造業，医療，物流と幅広い分野で活用が進んでいる[?]．　具体的な活用事例として，スマートストアが挙げられる．これは，デジタル技術を取り入れ，様々な作業を自動化，最適化した小売店舗のことである．スマートストアでは，決済処理や商品管理，混雑状況の把握，入退店管理などにデジタル技術が活用されており，効率的で正確な店舗運営の実現に寄与している[3]．　このように，デジタル技術の普及により，データの収集および活用が容易となり，業務の効率化やシステムの高度化が進んでいる．そして，大学においてもこのような技術の需要は高まっている．　大学では，キャンパス内に設置された購買施設の１つとして，図**??**に示す無人店舗が存在する．この施設は，学生をはじめとする多くの人々に利用されており，学内コミュニティの活性化に寄与している．その一方，無人店舗の運営においていくつかの課題が存在し，特に重要な課題として，在庫状況を手軽に確認する手段が十分に整備されていない点が挙げられる．これにより，目当ての商品を購入するために無人店舗を訪れたにもかかわらず，売り切れや欠品によって商品を入手できず，時間を浪費してしまう可能性がある．また，無人店舗から離れた場所では，販売されている商品を確認できないことから，無人店舗が利用選択肢として選ばれにくくなり，結果として無人店舗および利用者の双方に悪影響を及ぼすことが懸念される．そこで，無人店舗の在庫状況を遠隔から把握可能なシステムが導入されれば，無人店舗の利便性向上による利用者の増加や，さらなる学生コミュニティの活性化が期待できる．

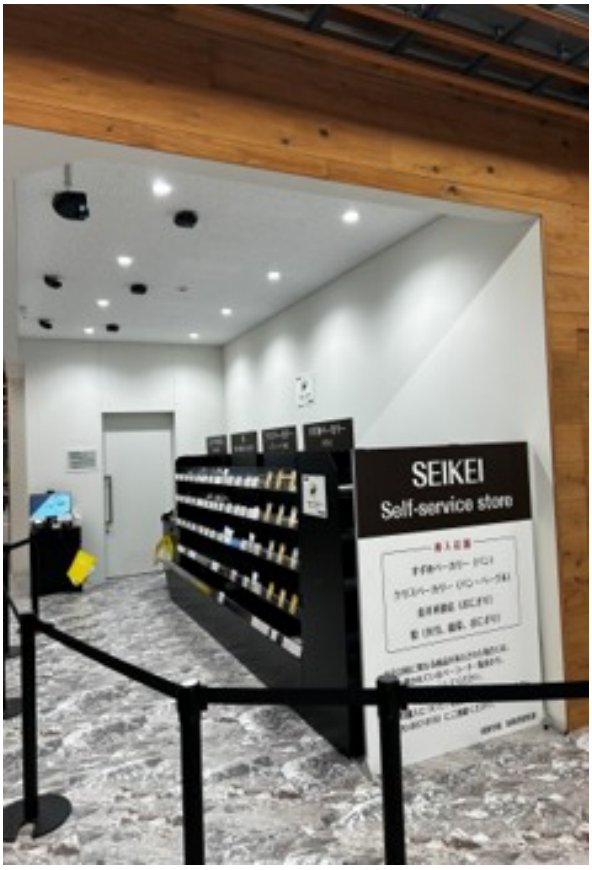

図 1.1: 大学内の無人店舗

1.2 研究目的

本研究では，大学キャンパス内に設置された無人店舗に超音波センサやデプスカメラを設置し，在庫状況に関するデータを収集，管理する方法および収集したデータを利用者に共有する方法を検討する．さらに，これらの検討結果に基づき，利用者が遠隔から在庫状況を把握可能なシステムの構築を目的とする．

第２章　画像処理による在庫検出

無人店舗の在庫状況に関するデータの収集を実現するため，複数の技術的アプローチについて検討を行い，最適な在庫検出手法を明確にする．本章では，画像処理技術を用いた在庫検出について，OpenCV を用いた検出方法，輪郭抽出，座標による商品識別，検出した結果と考察を説明する．

2.1　画像取得に用いた機器

画像処理を行うにあたり，無人店舗の商品棚の画像を取得するため，スマートフォンのカメラ機能を利用した．本章では，画像処理技術を用いた在庫検出の実現性について検討することを主な目的としているため，コスト削減や画像取得の容易さの観点からこの選択をした．実際の画像取得に用いたスマートフォンを図**??**に，取得した商品棚の画像を図**??**，図**??**に示す．

図 2.1: 画像取得用スマートフォン

2.1.1　サーマルカメラの精度

MLX90640 の性能を**??**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。

図 2.2: おにぎりを中心とする商品棚

図 2.3: 様々な色の商品が並ぶ棚

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検出

3.1　使用する超音波センサの説明

次に、超音波センサを用いてセンサと商品間の距離を計測した。使用したセンサはRainbow E-Technology 社の HR-SC04 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

3.1.1　超音波センサの精度

HC-SR04 の性能を**??**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　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。

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

ここで、音速は約343m/sである。

3.3　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在庫検出

4.1　使用するデプスカメラの説明

本研究ではデプスカメラを用い、センサと商品間の距離を計測した。使用したセンサはである。このセンサ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 4 Model Bを使用した。

4.1.1　デプスカメラの精度

D435の性能をに示す。測定範囲は0.1m〜10mであり、混雑状況のモニタリングの際に観測される距離は測定可能であると考えられる。

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

デプスカメラを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、カメラから赤外線を発信し、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。

第5章　結論

結論である。

第6章　結論

結論である。

参考文献

[1]

[2]

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．??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()
```

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

??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
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from PIL import Image
import os
import re
import matplotlib.pyplot as plt

#画像を読み込む
def list_pictures(directory, ext='jpg|jpeg|bmp|png|ppm'):
    return [os.path.join(root, f)
            for root, _, files in os.walk(directory) for f in files
            if re.match(r'([\w]+.(?:' + ext + '))', f.lower())]

X = []
Y = []
#0人の画像
for picture in list_pictures('./train_picture/zero/'):
    img = img_to_array(load_img(picture, target_size=(24,32)))
    X.append(img)
    Y.append(0)
#1人の画像
for picture in list_pictures('./train_picture/one/'):
    img = img_to_array(load_img(picture, target_size=(24,32)))
    X.append(img)
    Y.append(1)
#2人の画像
for picture in list_pictures('./train_picture/two/'):
    img = img_to_array(load_img(picture, target_size=(24,32)))
    X.append(img)
    Y.append(2)
#3人の画像
for picture in list_pictures('./train_picture/three/'):
    img = img_to_array(load_img(picture, target_size=(24,32)))
    X.append(img)
    Y.append(3)
# arrayに変換
X = np.asarray(X)
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.