三次元形状認識に関する基礎実験 第4報

・3DLidarカメラの試用
・3D認識ネットワークの生成
・角度の検出

2025.01

土井研究室　卒研資料に追記

研究背景/目的

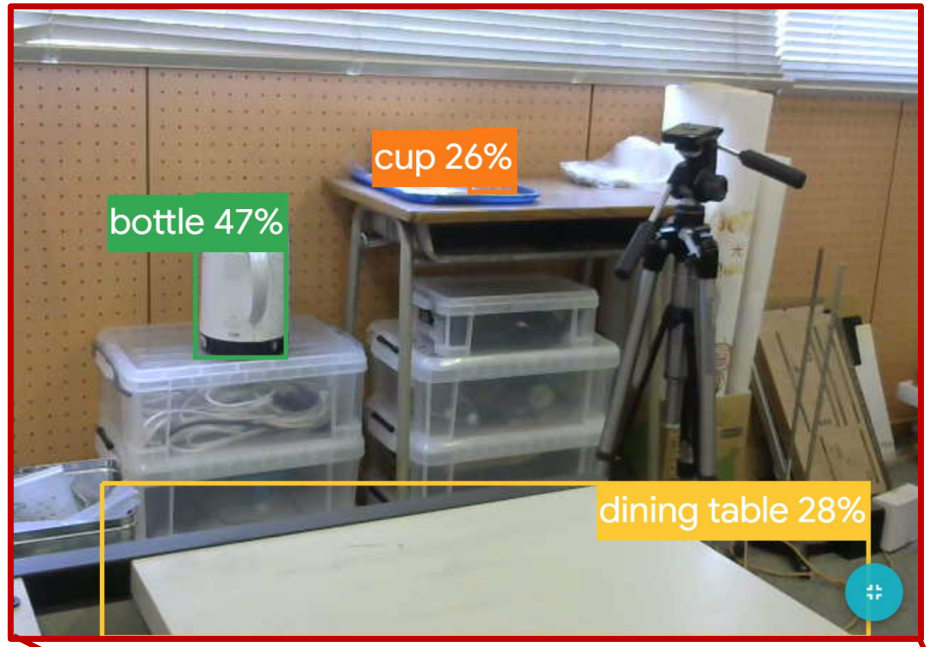
近年、自動運転や農業用ロボットの需要の高まりと共にAIによる画像認識技術の向上が必要とされている。

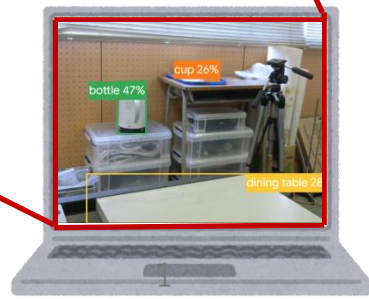
- 二次元画像の認識は普及しており、MediaPipeなどで簡単に試すことができる。
- 三次元情報の中の物体を認識する手法はあまり使われていない。

深度カメラで得た三次元情報に対して物体認識を試みる

MediaPipe：Googleで開発を主導しているオープンソースの機械学習ライブラリ。PCとWEBカメラがあれば画像認識を体験できる。

MediaPipeによる物体検出





応用例：果実収穫ロボット

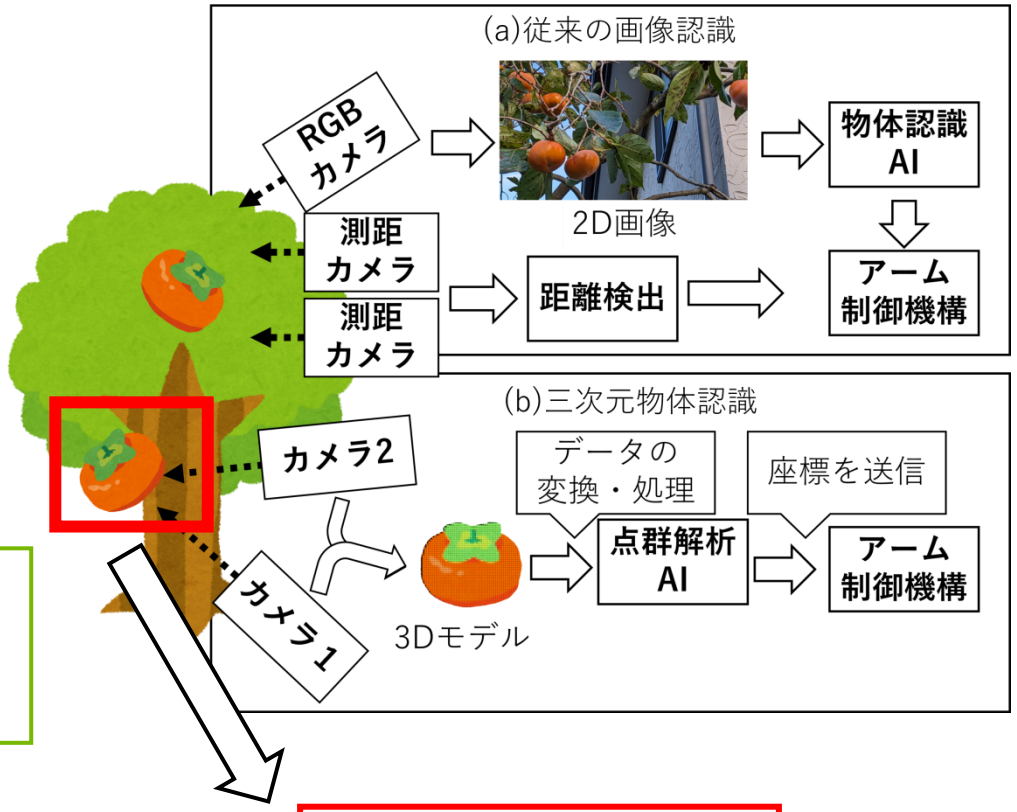
■　従来の収穫ロボット

RGBカメラからの映像で物体検出を行う

■　三次元物体検出を搭載した収穫ロボット

深度カメラを用いて対象物を三次元情報としてとらえ、点群解析AIで果実の情報を取得する。

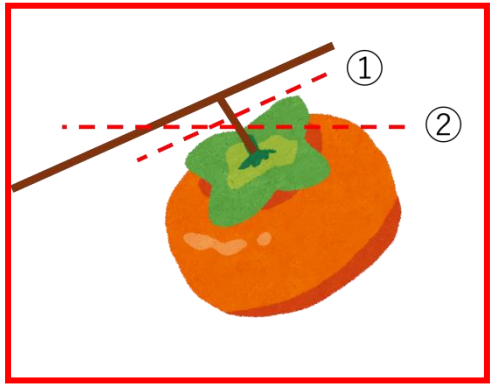
→点群の持つ情報量が多いため、果実の状態に関する情報をより多く得られる。



■ 収穫時、柿の軸を切断することを考える

柿が傾いている場合、右図①のように傾きに合わせて鋏を接近させる必要がある。

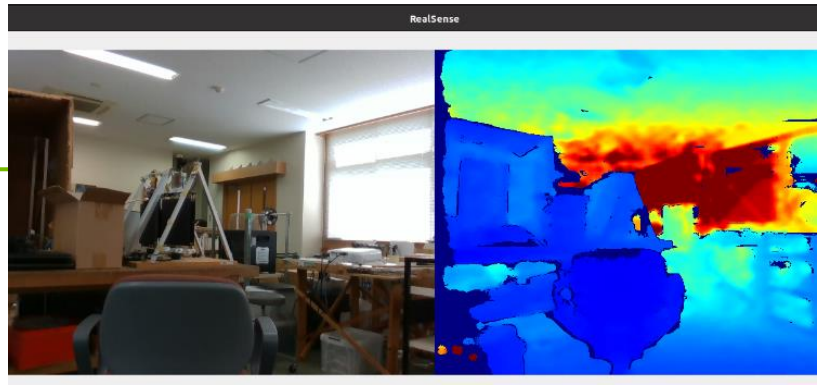
判断時に三次元情報を利用できるのではないか。



三次元情報の取得

■ RealSense

三角測量法を利用した深度カメラIntel RealSense D435iを用いて距離を取得した。



距離を色分けした深度マップ

■ 点群データの取得

RealSenseをpyrealsense2ライブラリで動作させ、撮影した物体の点群データを取得した。

必要なファイル形式に変換するプログラムを作成した。

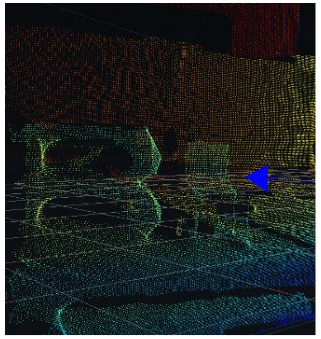
(a) (b)

RealSenseで取得した点群

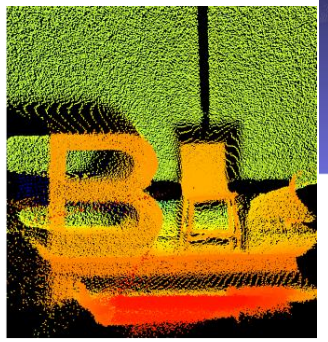
点群：三次元座標で構成された点の集合

■ 3D LiDARカメラの試用

レーザーの反射を利用して距離を測定するLiDARカメラ CS20 を試用した。RealSenseと比較すると対象物の縁がよりはっきりと取得できた。

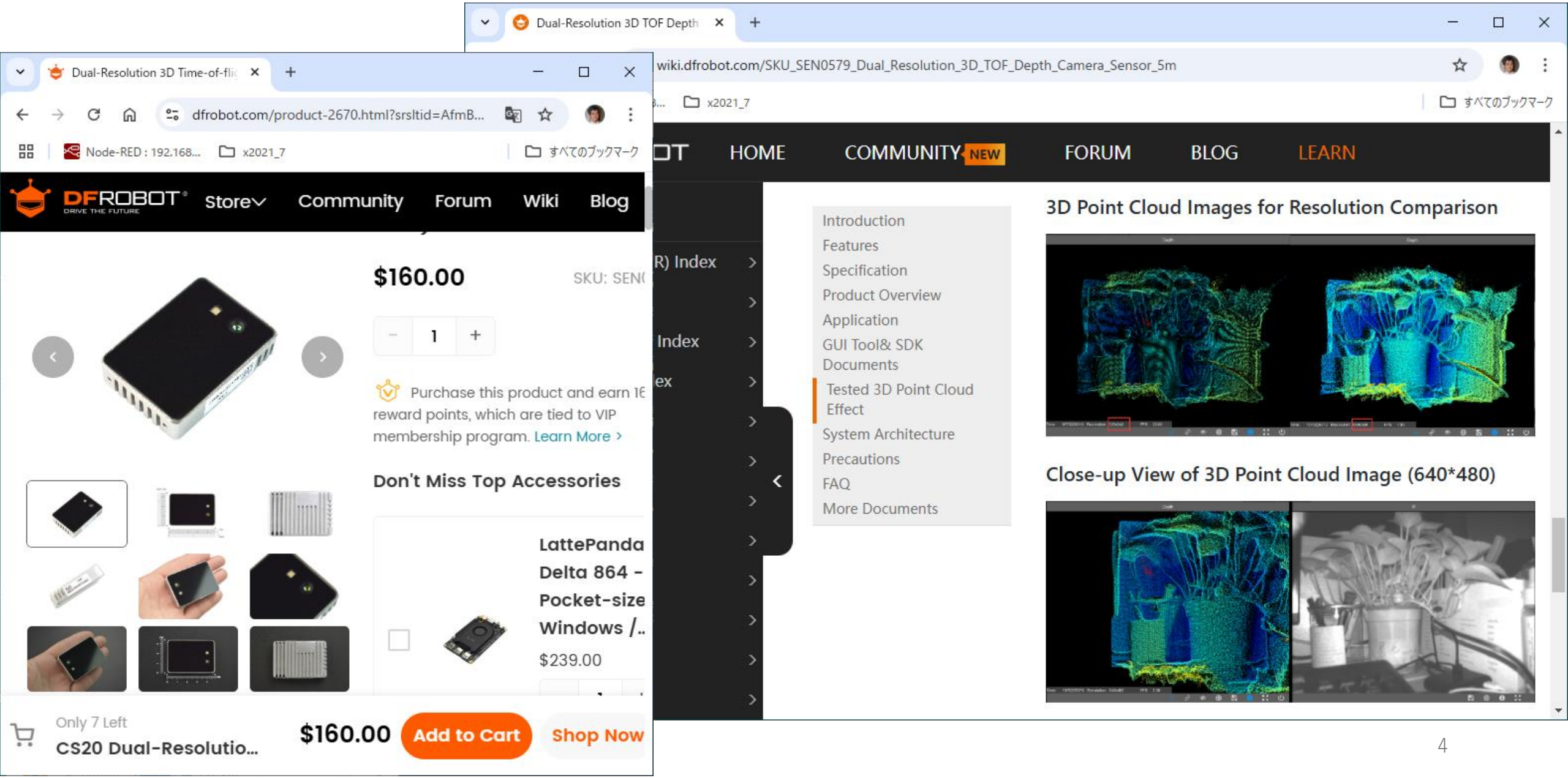


RealSense



LiDAR

CS20　デュアル解像度3D TOFソリッドステートLiDAR
解像度640 x 480のToFカメラ



左　Realsense D435

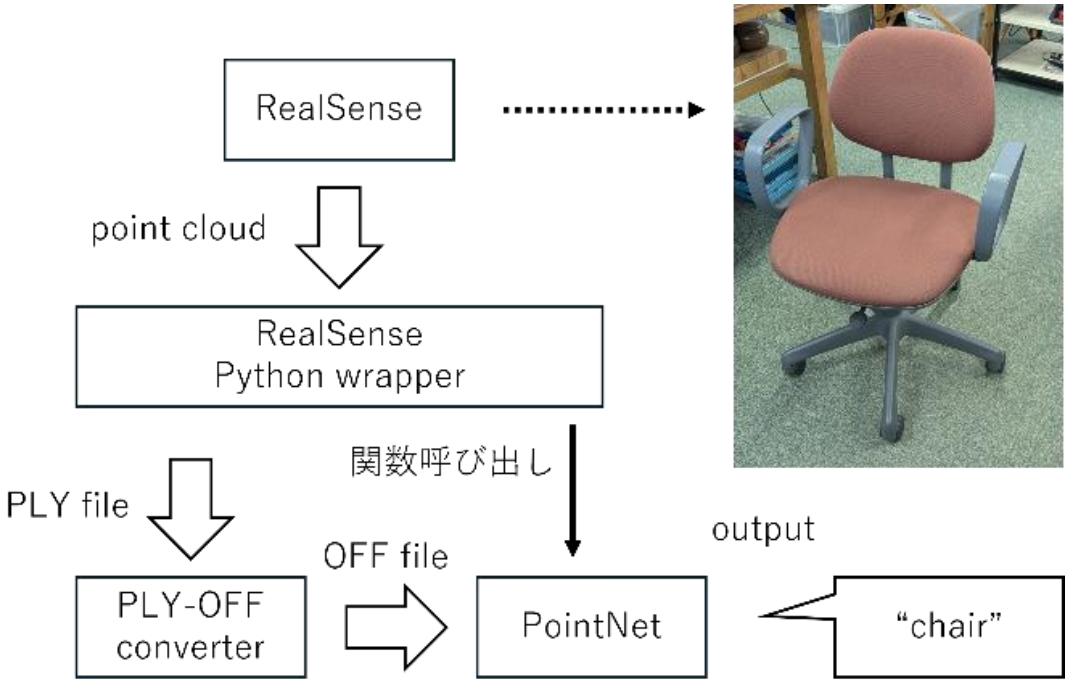
右　CS20

PointNetによる点群分類

PointNetの基本的な動作は２報をご覧ください。

■ リアルタイム分類

RealSenseの駆動プログラムと、以前作成したPointNetによる分類プログラムを統合し、カメラから得た三次元データをリアルタイムで分類するシステムを作成した。



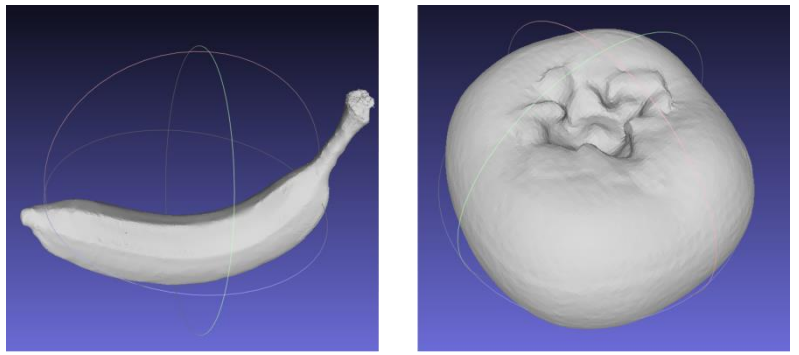
リアルタイム分類システムの動作

分類の様子

データセットの作成

既存のデータセットに含まれない物体を分類するため、独自のデータセットを作成し、PointNetによる学習を行った。

設定クラス：バナナ, 椅子, 柿, ソファー



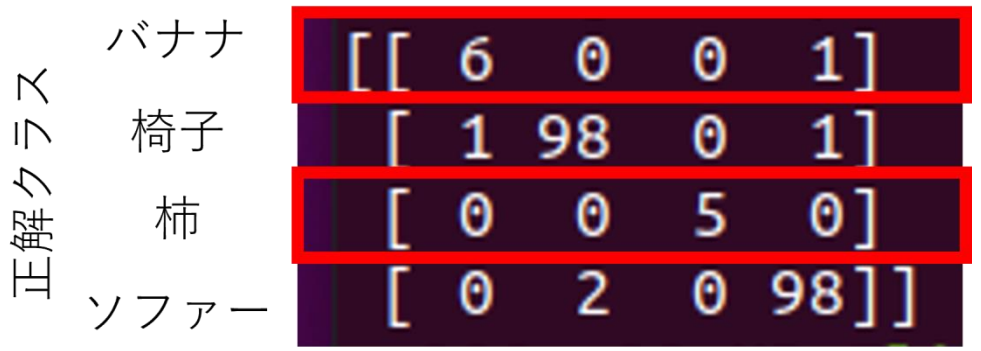
(a)バナナの３Dモデル　(b)柿の3Dモデル

データ収集元:Sketchfab

学習結果を利用してテストデータを分類すると、7個中6個のバナナと5個すべての柿が正しく分類された。

データセット内容

クラス	学習用データ	テスト用データ
バナナ	18	7
椅子	889	100
柿	10	5
ソファー	680	100



分類結果

角度推定

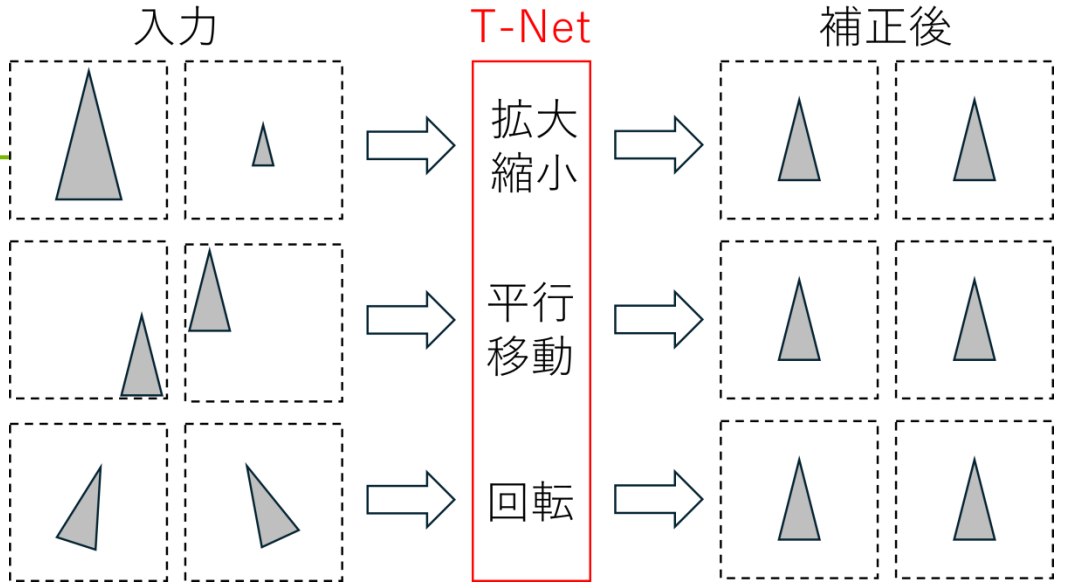
■ T-Netの概要

T-Netとは、PointNetに入力される点群の角度やサイズを統一するために使用される、入力補正に特化した小型のニューラルネットワークである。

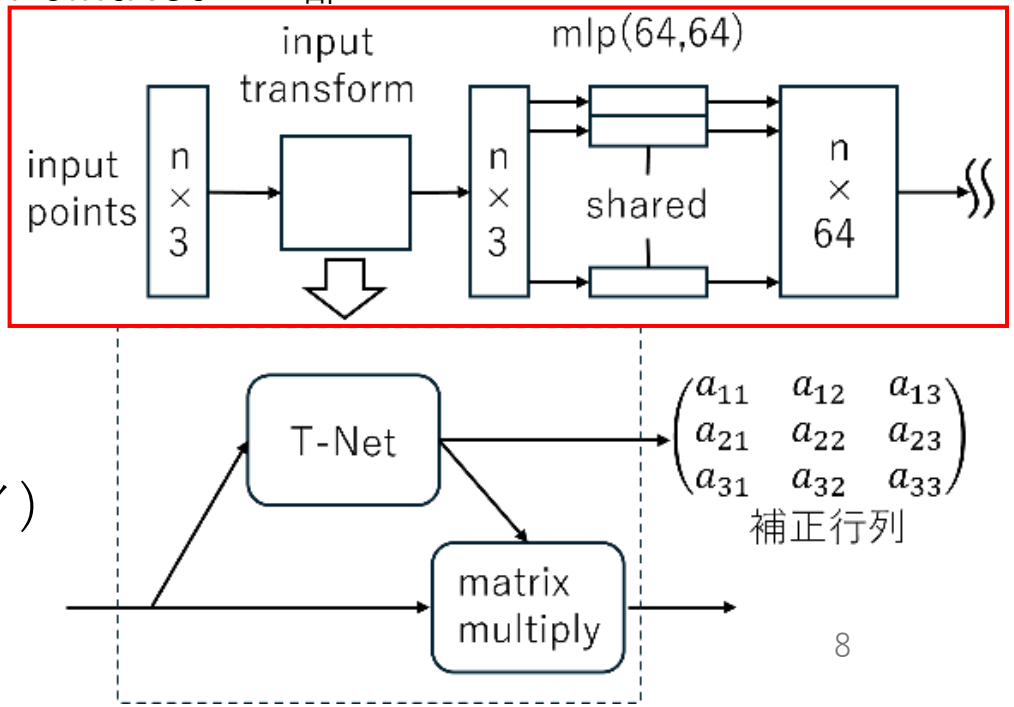
3行3列の補正行列を出力し、入力点群と掛け合わせて補正を行う。

補正行列を分析することで対象物の角度が得られるのではないか。

分類プログラムから補正行列を取り出し、SciPy(サイパイ)ライブラリを用いて回転角に変換する処理を追加した。



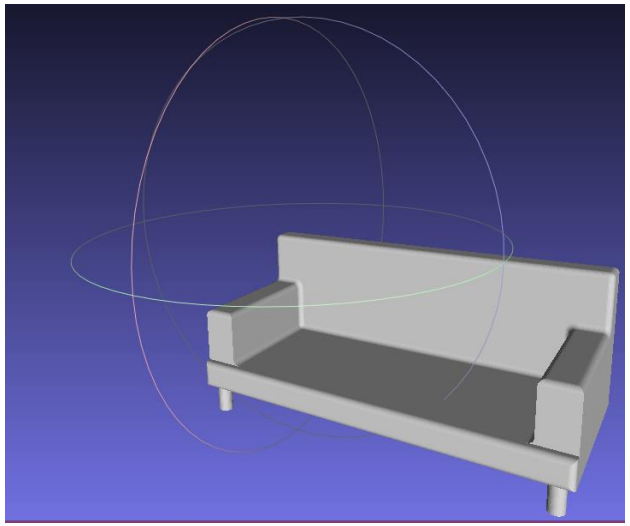
PointNetの一部



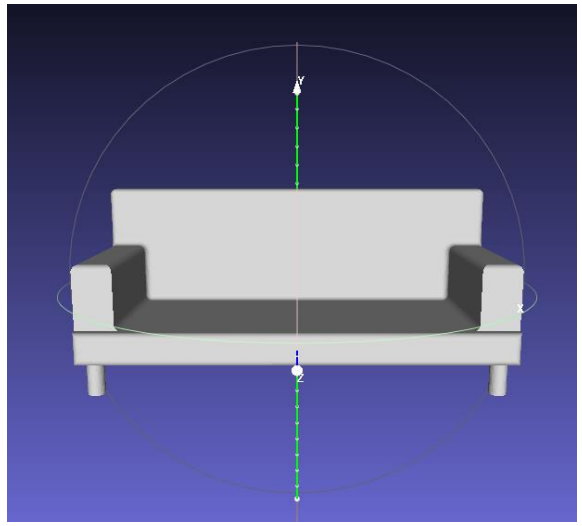
角度推定

■ 回転角度の抽出

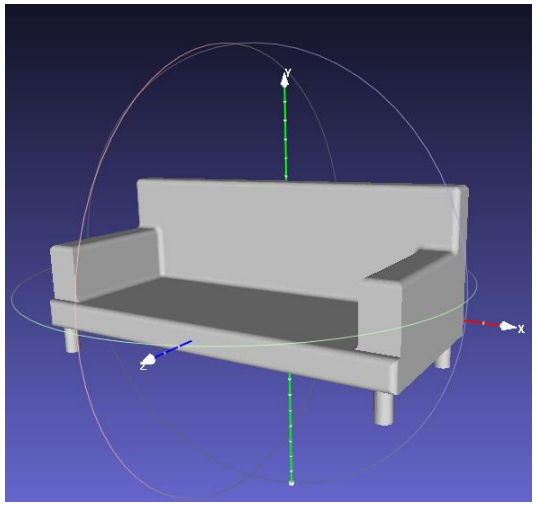
補正行列に含まれる回転成分のみを取り出す手法として、特異値分解(SVD)が挙げられる。



補正行列を掛ける



拡大縮小・平行移動
成分を掛ける



回転成分を掛ける

回転成分の取り出しが可能

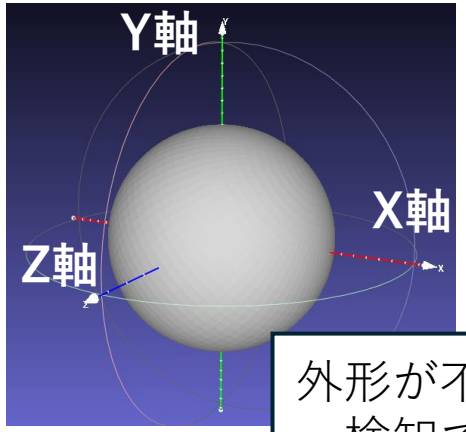
回転角度抽出の概要

角度推定

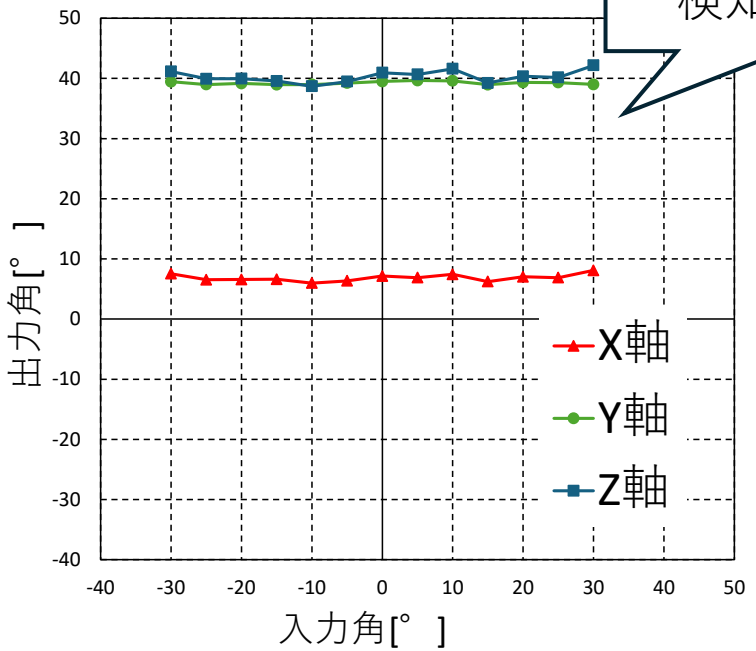
■ 基本図形の角度推定

球

X軸回転

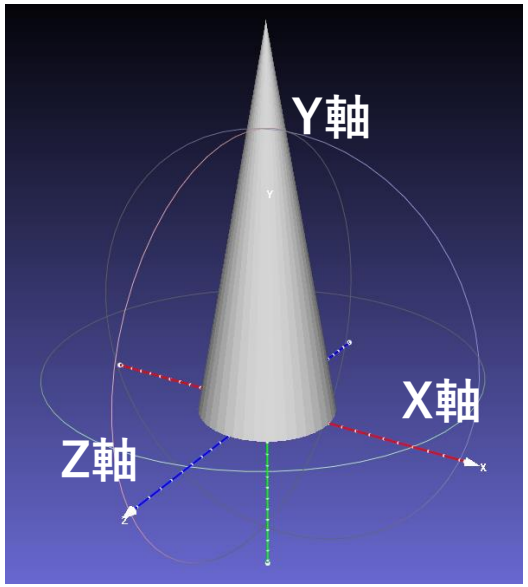


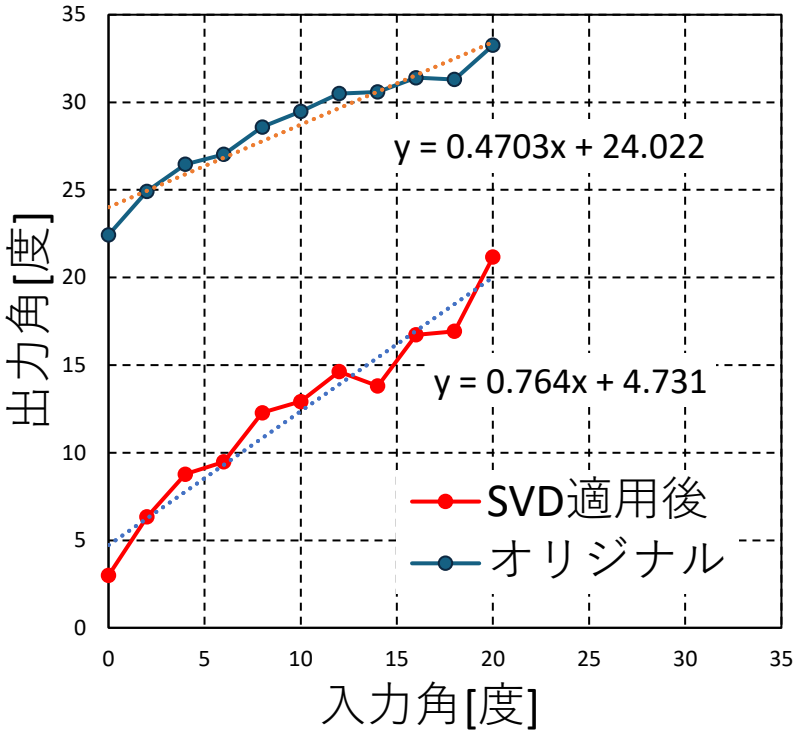
外形が不変なので
検知できない



円錐

X軸回転



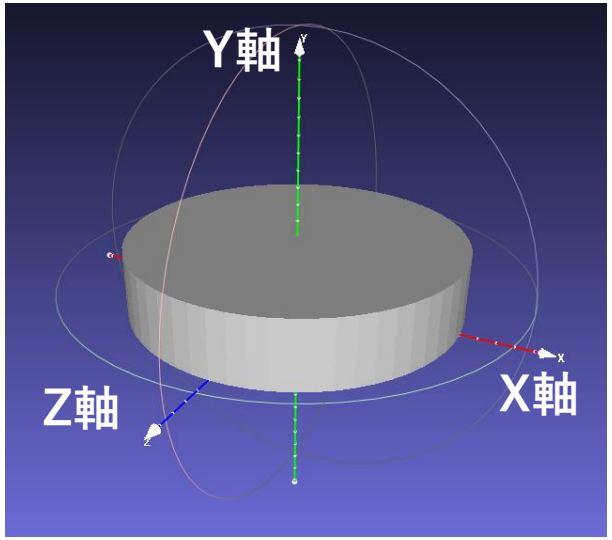


X軸周りに回転させると、X軸の出力角度が増加した。
また特異値分解(SVD)を適用すると近似直線の傾きが1に近づいた。

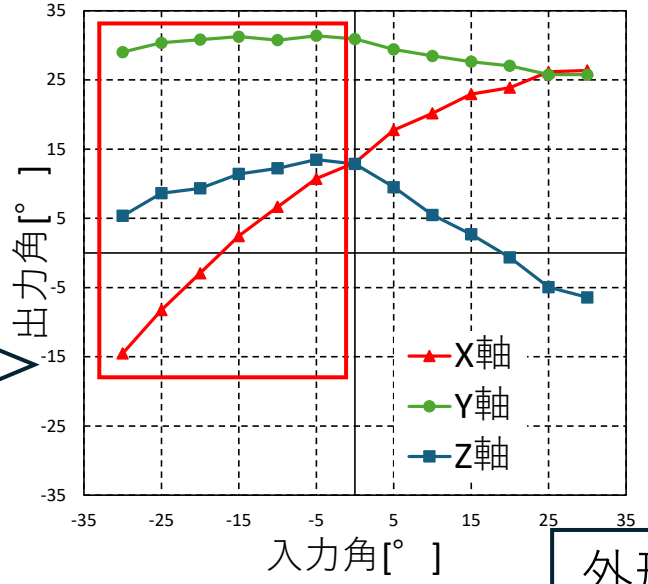
角度推定

■ 基本図形の角度推定

円柱

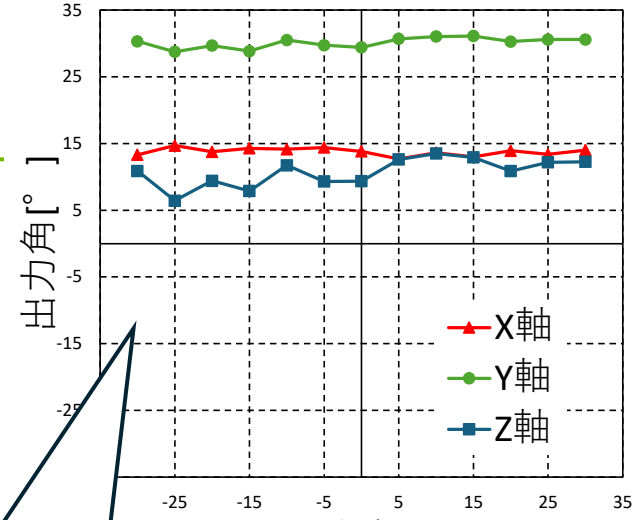


X軸周りの出力角が増加



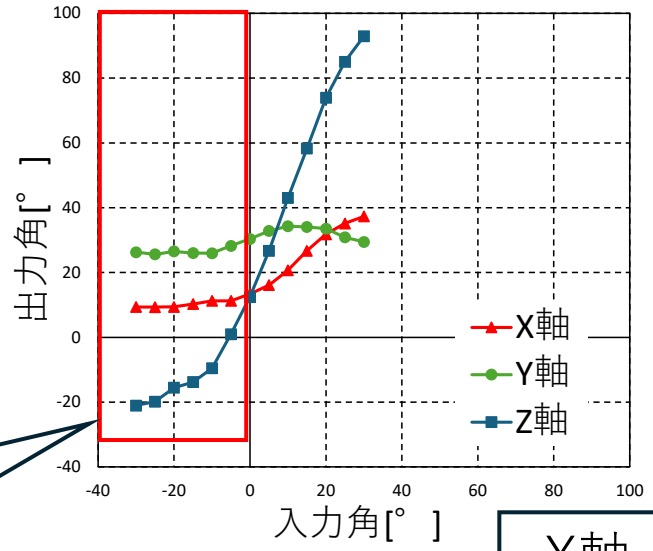
X軸回転

外形が不変なので検知できない



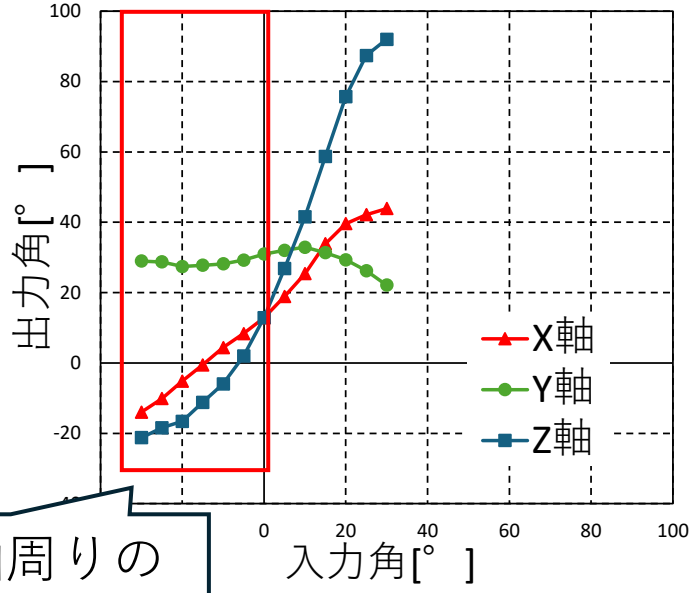
Y軸回転

Z軸周りの出力角が増加



Z軸回転

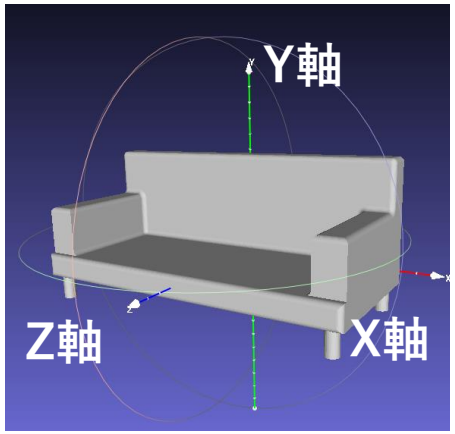
X軸、Z軸周りの出力角が共に増加



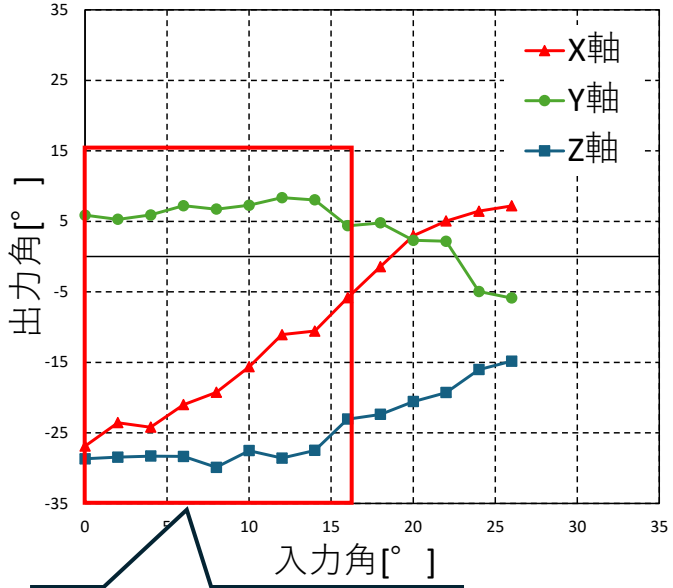
X, Z軸同時回転

角度推定

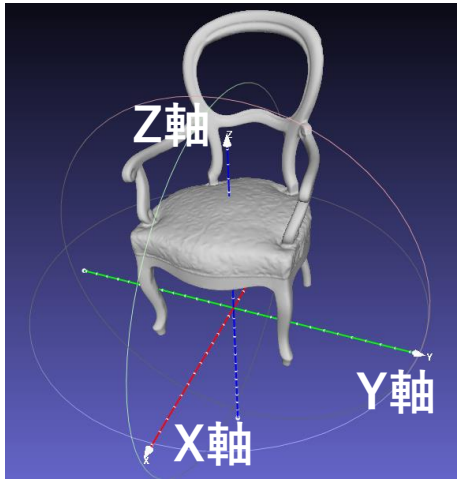
■ 家具の角度推定



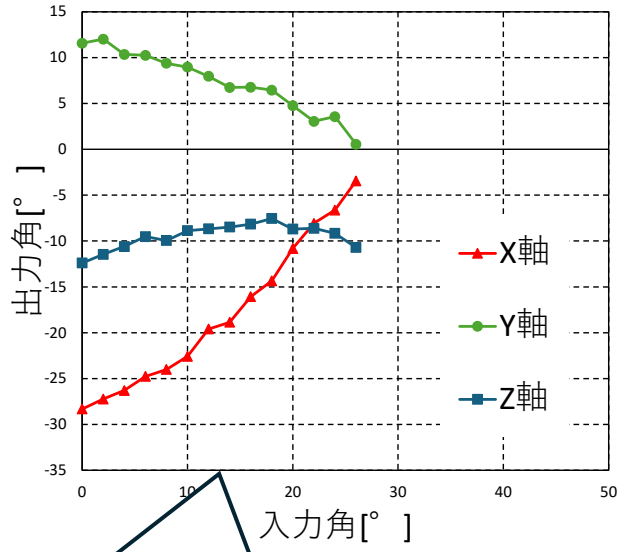
X軸回転



X軸周りの出力角が増加



X軸回転



X軸周りの出力角が増加したが、Y軸周りが減少

家具の3Dモデルに対しても角度推定を行った。
ある角度から、角度を変えていない軸の出力が変化する現象もあり、
さらなる検証が必要である。

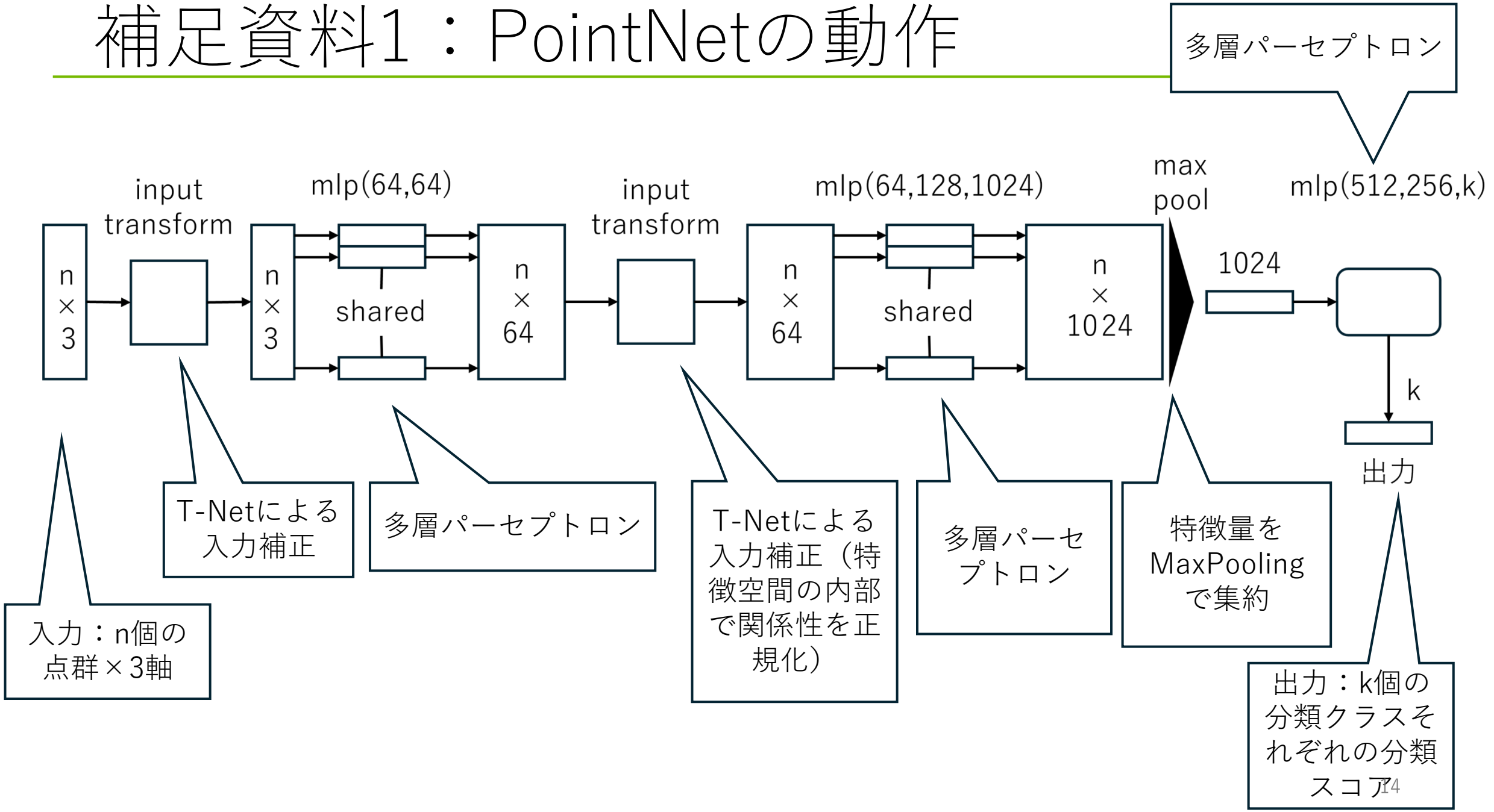
まとめ

- 深度カメラから三次元情報を取得し、それに対する物体認識を実現させ、リアルタイム分類システムを作成
- ニューラルネットワークが物体を認識するときに得られる補正情報から、対象物の角度推定を行った

今後の課題

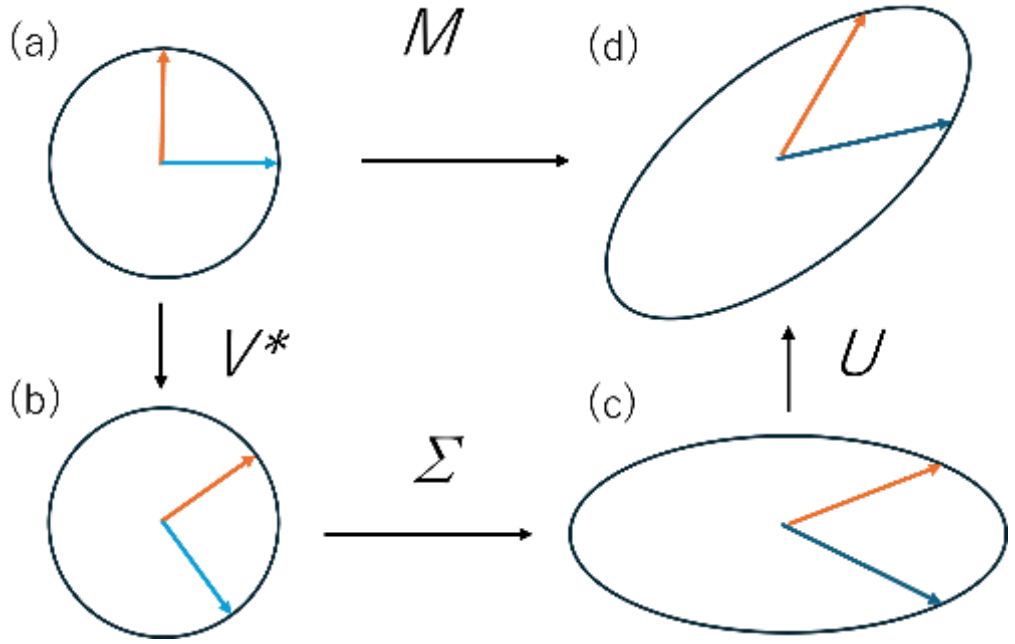
- より小さい物体の点群取得方法を検討
- 取得対象物の切り出し
- 深度カメラで取得したデータの角度推定

補足資料1：PointNetの動作



補足資料2：特異値分解

Mは一般の行列を表しており，これを直交行列V*, Uと対角行列Σに分解する。図におけるV*, Uは回転，Σは伸縮に相当し，(a)を(d)に変形する行列Mを分解することで，回転を行うV*, Uを取り出すことができる。



補足資料3：分類部分のプログラム

```
Open                                    ~/pytest/pointnet
    pointnet = model.PointNet()
#train.pyによる学習で作成されたsave_n.pthを参照
    pointnet.load_state_dict(torch.load('checkpoints/save_14.pth'))
    pointnet.eval();

    with torch.no_grad():
        for i, data in enumerate(valid_loader):
            inputs, labels = data['pointcloud'].float(), data['category']
            outputs, __, __ = pointnet(inputs.transpose(1,2))
            _, preds = torch.max(outputs.data, 1)
            end = time.perf_counter()#計測終了
            probabilities = F.softmax(outputs, dim=1)
#カテゴリーを表示?
            #print(preds)
            #outputsをそのまま表示
            #print("生の出力")
            #print(outputs)
#ファイルを表示
            print("current file:")
            files = glob.glob("../off/test/test/*")
            for file in files:
                print(file)
#0704追加
            #print("要素")
            result = preds[0]
            #print(result)
            print("要素内の数値")
            nresult = preds.item()
            print(nresult)
#精度を表示
            predicted_probability = probabilities[0, preds.item()].item()
            print(f'Probability: {predicted_probability}')
            #要素名を表示
            name = element[nresult]
            pprob = predicted_probability * 100
            print("名称：", name)
            print("Probability[%]：", "%.2f" % pprob)
            print('{:.5f}'.format((end-start)/60), "[s]")
```

Python Tab Width: 8 Ln 50, Col 55 INS

```
nu20@nu20-HP-EliteDesk-800-G4-SFF: ~/pytest/pointnet
nu20@nu20-HP-EliteDesk-800-G4-SFF:~/pytest/pointnet$ python3 1mtest2.py
1mtest2.py
Namespace(batch_size=32, epochs=15, lr=0.001, root_dir='../ModelNet10/', save_mo
del_path='./checkpoints/')
matrix from T-net:
 tensor([[[ 2.8270,  2.5590,  0.5120],
         [ 0.3130,  4.1360,  1.3800],
         [-2.1920, -6.5920,  7.2990]]])
行列式: [101.04817]
angle(yxz): [-56.807   8.703 -21.456]
回転行列:
[[ 0.942  0.192  0.274]
 [-0.308  0.817  0.488]
 [-0.13  -0.544  0.829]]
抽出後の行列式: 0.9999998
ex_angle(yxz): [-33.278   7.458 -18.116]
current file:
../off/test/test/chair_0900.off
要素内の数値
2
Probability: 0.9998206496238708
名称： chair
Probability[%]： 99.98
0.03945 [s]
```

補足資料4：角度推定のプログラム

```
    #1024追加
    if torch.numel(matrix) == 9:
        print("matrix from T-net:\n", my_round2(matrix))
        matb = np.linalg.det(matrix)
        print("行列式:", matb)
        # 回転行列をNumpyに変換してオイラー角を取得
        rotation_matrix = matrix[0].detach().cpu().numpy()
        # 回転行列からオイラー角を取得
        rotation = R.from_matrix(rotation_matrix)
        euler_angles = rotation.as_euler('xyz', degrees=True)
        print("angle(yxz):", my_round2(euler_angles))
        #1210追加：回転行列のみを抽出
        normalized_matrix = normalize_to_rotation_matrix(rotation_matrix)
        print("回転行列:")
        print(my_round2(normalized_matrix))
        print("抽出後の行列式:", np.linalg.det(normalized_matrix))
        #抽出した行列から角度を計算
        ex_rotation = R.from_matrix(normalized_matrix)
        ex_euler_angles = ex_rotation.as_euler('xyz', degrees=True)
        print("ex_angle(yxz):", my_round2(ex_euler_angles))
    return matrix
```

```
    #SVDを用いて直交成分を取得
    U, _, Vt = np.linalg.svd(copy_matrix)
    rotation_matrix = np.dot(U, Vt)
    #サイズが１の次元を削除
    rot_matrix = np.squeeze(rotation_matrix)
    #行列式を確認
    det = np.linalg.det(rot_matrix)
    if det < 0:
        U[:, -1] *= -1
        rot_matrix = np.dot(U, Vt)

    return rot_matrix

#四捨五入関数
def my_round2(x, decimals=3):
    return np.sign(x) * np.floor(np.abs(x) * 10**c

class Tnet(nn.Module):
```