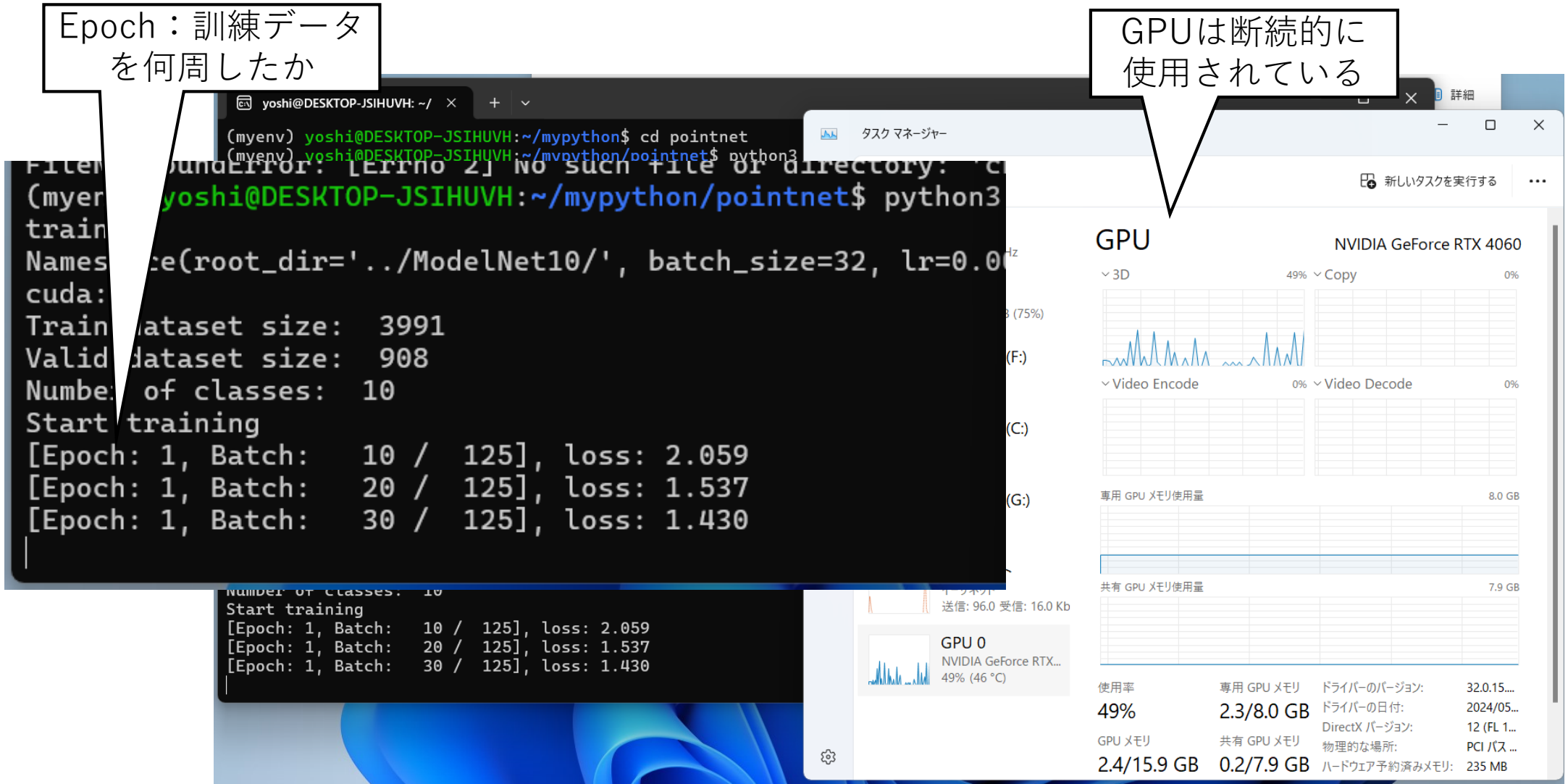
三次元形状認識に関する基礎実験

PointNetの動作確認：トレーニング

トレーニング用データ(ModelNet10)を用いて、3Dモデルを学習させた。



PointNetの動作確認：分類結果

テスト用のoffファイルを分類し、PointNetが判断した分類と
正しい分類の回数を表示した

RealSenseの動作確認

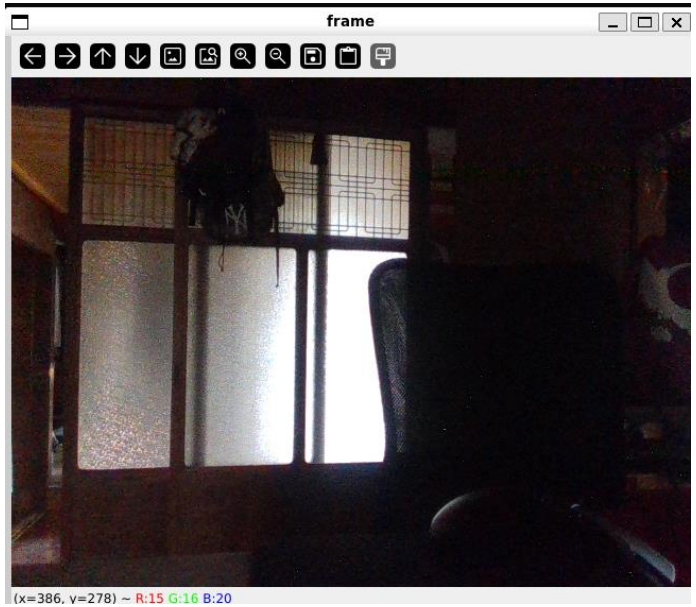
- USBIPDコマンドでUbuntuにRealSenseカメラを認識させると、Video0～Video5までの6つの映像入力が確認できた。
- cv2.Videocapture() 関数にカメラ番号を入れるとモードが切り替わるが、距離情報を取得できなかった。

→仮想マシン（WSL上のUbuntu）ではなく、直にUbuntuをインストールしたPCを使用することにした。

各映像入力に含まれる情報

Video0：深度センサ
Video2：赤外線センサ
Video4：RGBカメラ映像

cv 2 .Videocapture(2)の出力画像



cv 2 .Videocapture(4)の出力画像

PointNetで任意の3Dモデルを判別

目標：3Dモデルを読み込ませると「これは○○%の確率で××です」と出力する

デモプログラムではフォルダにあるすべての3Dモデルを分類し、結果との混同行列を表示していた。

1つの3Dモデルファイルのみを分類し、結果を表示させたい。

まずPointNetのメインプログラムの一層上に判定したいモデルを置くフォルダ”off”を作成し、読み込むフォルダを”../ModelNet10”から”../off”に変更した。

```
20 if __name__ == '__main__':
21     args = parse_args()
22
23     path = Path("../ModelNet10")
24
```

```
23 if __name__ == '__main__':
24     args = parse_args()
25
26     path = Path("../off")
27
```

このフォルダに3Dモデル(.off形式)を入れてプログラムを実行すると、1行1列の行列が出力された。

PointNetで任意の3Dモデルを判別

判定結果を出力するために行ったこと

PointNetのメインプログラムに書かれている判定結果（配列）の出力部分

```
50            outputs, __, __ = pointnet(inputs.transpose(1,2))
```

判定結果から最も確率の高い（大きい）数値を取り出す部分

```
51            _, preds = torch.max(outputs.data, 1)
```

```
53 #カテゴリーを表示?
54            print(preds)
```

←”preds”を表示してみると…

```
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/')
tensor([2])
```

テンソル形式の2が出力された。

PointNetで任意の3Dモデルを判別

preds[0]で要素内の数値を指定して表示した場合tensor形式だが、
preds.item()を使用することで要素を整数として取り出すことができた。

```
56          print("要素")
57          result = preds[0]
58          print(result)


59          print("要素内の数値")
60          nresult = preds.item()
61          print(nresult)
```

```
nu20@nu20-HP-EliteDesk-800-G4-SFF:~/pytest/pointnet$
1mtest2.py
Namespace(batch_size=32, epochs=15, lr=0.001, root_di
del_path='./checkpoints/')
tensor([2])
要素
tensor(2)
要素内の数値
2
```

数値が取り出せたので、カテゴリの配列を用意して分類した物の名前を表示するようにした。

```
7 #0704結果表示のための配列定義
8 element = ['bathtub', 'bed', 'chair', 'desk', 'dresser', 'monitor',
  'night_stand', 'sofa', 'table', 'toilet']
```

結果の数値と名称を関連付ける

PointNetで任意の3Dモデルを判別

判定が正しい確率を表示するには

出力にはそれぞれの要素に割り当てられた数値が含まれており、
一番大きな数値をもつ要素が判別結果になる。

```
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/')
tensor([2])
生の出力
tensor([[-12.6743, -11.7575,  -0.1377,  -8.3899,  -3.9567,  -8.6637,  -8.2141,
         -13.0650, -10.5477,  -2.2179]])
```

ソフトマックス関数

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

判別結果の正確さを求めるために、入力した数値の割合を保ちながら
合計を1に調整できるソフトマックス関数を使用した。

```
52              probabilities = F.softmax(outputs, dim=1)
```

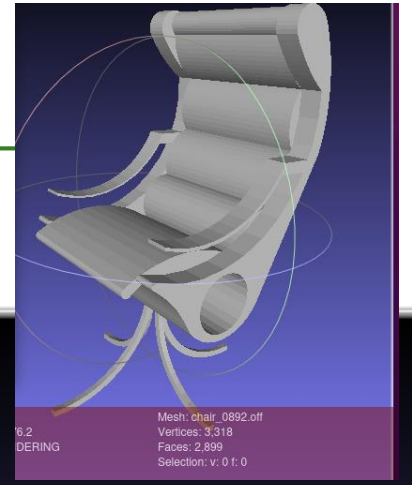
```
62 #精度を表示
63             predicted_probability = probabilities[0, preds.item()].item()
64             print(f'Probability: {predicted_probability}')
```

```
nu20@nu20-HP-EliteDesk-800-G4-SFF
1mtest2.py
Namespace(batch_size=32, epochs=1
del_path='./checkpoints/')
tensor([2])
要素
tensor(2)
要素内の数値
2
Probability: 0.7736487984657288
```

PointNetで任意の3Dモデルを判別

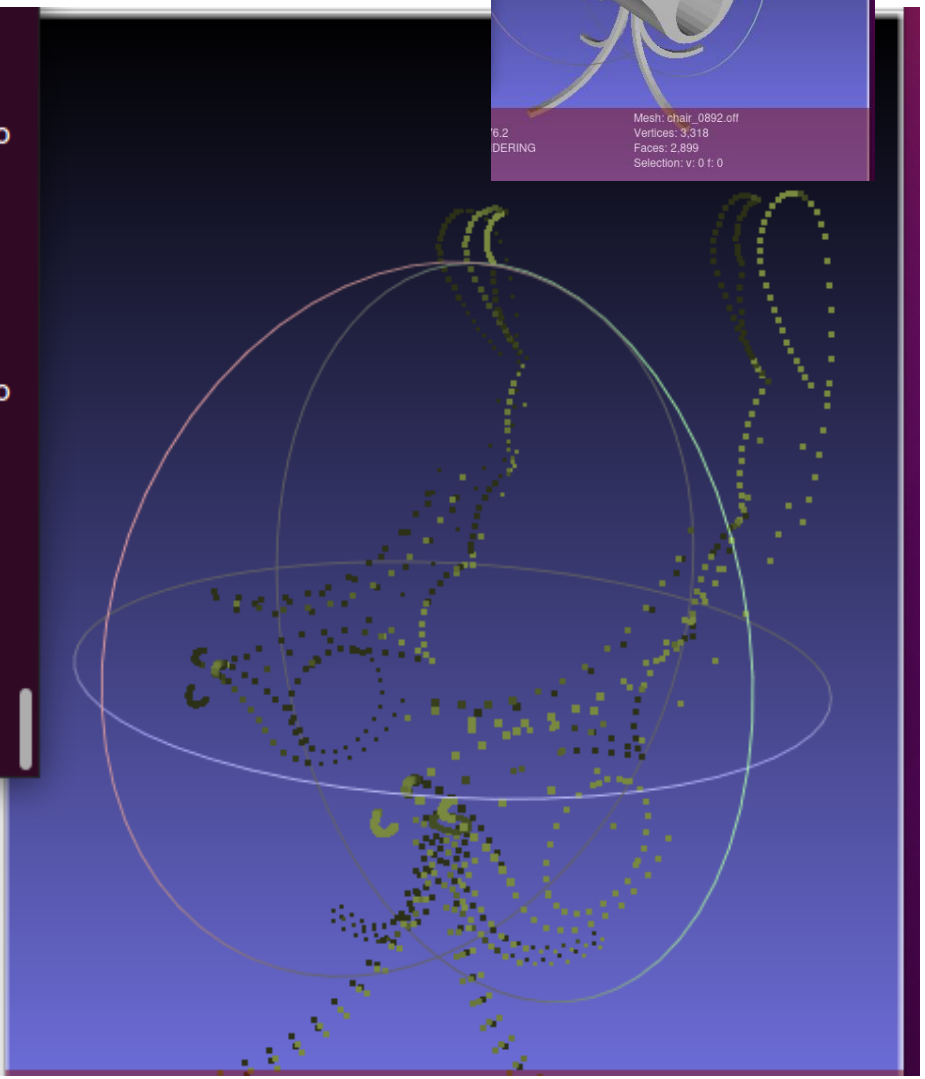
プログラム全体の実行結果

入力した3Dモデル(.off形式)




```
nu20@nu20-HP-EliteDesk-800-G4-SFF:~/pytest$ cd pointnet
nu20@nu20-HP-EliteDesk-800-G4-SFF:~/pytest/pointnet$ python3 1mtest.py
1mtest.py
Namespace(batch_size=32, epochs=15, lr=0.001, root_dir='../ModelNet10/', save_mo
del_path='./checkpoints/')
Batch [   1 /    1]
tensor([2])
Probability: 0.66900068521495
nu20@nu20-HP-EliteDesk-800-G4                                   1mtest2.py
1mtest2.py
Namespace(batch_size=32, epochs=15                  001, root_dir='../ModelNet10/', save_mo
del_path='./checkpoints/')
tensor([2])
要素
tensor(2)
要素内の数値
2
Probability: 0.7736487984657288
名称： chair
Probability[%]： 77.36487984657288
nu20@nu20-HP-EliteDesk-800-G4-SFF:~/pytest/pointnet$
```

名称：chair
Probability[%]：77.36…



疑問点

- 判定結果の正確さが限りなく100%になることが多い
  組込みシステムで画像分類を行ったときも高い確率が出ることがほとんどだったので正常かもしれない。
  ModelNet10に付属しているテスト用ファイルしか入力していないことが影響している可能性がある。

次にやること

- ModelNet10のテストモデル以外の3Dモデル(.off形式)を分類できるか試す
- 読み取り可能なファイル形式を調べる
- RealSenseから点群形式の3Dモデルを作成し、保存する
- RealSenseで作成した点群の保存形式を検討する
- 点群の精度を上げる取り込み方を検討する(複数カメラからの合成など)
- 作成した点群をPointNetで分類してみる
- できそうなら点群作成と分類のプログラムを連携させる
- VoteNetを導入して物体検出