情報理工学特別研究報告書

題目

物体検出技術と深度カメラを用いた
人体密度可視化システムの開発

学生証番号　954020

氏名　西尾漱一郎

提出日　令和 5 年 1 月 26 日

指導教員　蚊野　浩

京都産業大学
情報理工学部

要約

新型コロナウイルスや韓国のハロウィンでの群衆雪崩による圧死事故などによって，人が密集している状態を検出できることの重要性が高まっている．人が密集しているという情報はビジネス面にも活用ができ，人の集まりや流れをリアルタイムに可視化する技術は，さまざまな場面で利用することができる重要な技術である．本研究では，物体検出技術 Yolox と深度カメラ RealSenseD435 を組み合わせ，人体の 3 次元座標を測定し，人体の密度を可視化するシステムを構築した．

RealSense で RGB 画像と深度情報を取得し，RGB 画像に対して Yolox を実行させ，人体の検出を行う．人体が検出されるとバウンディングボックスという矩形に囲まれる．その中心点のピクセル座標と，RealSense の深度情報を使い，人体の 3 次元座標を測定する．得られた 3 次元座標から，カーネル密度推定を用いて密度推定を行い，その結果を可視化した．本研究では，特別研究 II の授業中に研究室メンバー5 人の 3 次元座標を測定した．0.25 秒ごとに１回，３次元座標を計算し，約 3 時間の動作中の３次元座標を測定・分析した．

システム構築の結果，人体の検出，人体の 3 次元座標の測定，人体密度の可視化が可能になった．動作実験の結果，5 人とも研究室のそれぞれの座席位置の座標あたりで最も密度が濃くなった．特別研究時間中，ほとんどの時間は 5 人とも各自の座席で研究や勉強をしていたためである．自分の座席位置から離れた場所で密度が濃くなっている人がいることが確認できた．これは研究の休憩時間などで，他の人の座席位置近くに移動し，雑談していたためである．目標は達成できたが，奥行きの測定精度に課題が生じた．人と人が重なった時，奥側の人の 3 次元座標を正確に測定できなかった．１台のカメラだけでは，画角などから人体の検出や 3 次元座標の計算精度に限界があるため，複数のカメラで異なる角度から観察することによって，より正確に人体を検出し，精度の高い 3 次元座標の計算ができると考えた．

目次

1章 序論

2019 年 12 月頃に新型コロナウイルスが中国の武漢市で確認され，瞬く間に世界中に広がり多くの死者を出した．また 2022 年 10 月のハロウィンにおいて韓国・梨泰院で群衆雪崩による圧死事故が発生した．これらの災害から，人が密集している状態を即座に検出できることの重要性が認識されている．繁華街やレストン，コンサート会場のように人が集まりやすい場所で利用でき，人体の密集度を自動的に検出できる技術を開発することで，これらの災害を未然に防ぐことが可能であるとか考えられるからである．

人が集まっているという情報は，ビジネスに活用することも可能である．例えば展示会場の，あるブースに人が集まっているということからその展示物が人気であることがわかる．人気の展示物がわかれば，それを広報すれば，さらなる集客が狙える．小売店の場合，人が集まりやすい場所を特定できれば，そこに商品を重点的に置くと，商品が消費者の目に留まりやすくなり，商品の販促につながる．このように人の集まりや流れをリアルタイムに可視化する技術は，さまざまに利用することができる重要な技術である．

注目している空間をビデオカメラで撮影し，その映像から人体を検出すれば，見かけの人の集まりや流れを計測することができる．これを可能にする技術に深層ニューラルネットワーク (DNN) がある．人体を検出する DNN の中でも、Yolo は高速に検出ができる技術として知られている．Yolo は 2015 年に発表された画像認識用 DNN であり，その後も改良が進められている．本研究では 2021 年に発表された Yolox を用いて動画像から人体を検出する．

カメラで観察する人体の位置は映像中の位置であり，実空間の位置とは異なっている．すなわち，映像中では離れているように見えていても，人体がカメラに近ければ密集している可能性がある．逆に，映像中では集まっているように見えても，人体がカメラから離れていれば，それほど密集していない可能性がある．ビデオカメラの映像を使って人体の密集度や流れを計測する場合には，カメラの透視投影による遠近効果を補正する必要がある．今回の研究では，人体の存在する空間的な位置を正確に把握するために深度カメラを用いる．最近では，深度カメラとしていくつかの方式のものが選択可能である．本研究では，深度カメラとして実績の高い Intel RealSenseD435 を用いる．Yolox と RealSenseD435 を組み合わせて利用することで，3 次元空間に存在する人体の位置を正確に把握することが可能になる．

本研究では，Yolox と RealSeneseD435 を用いて，複数の人体の 3 次元位置の時系列を処理するプログラムを Python で開発した．以下， 2 章では物体検出と深度カメラについて記述する．3 章はシステムの構築と実験方法について記述する．4 章は実験結果について記述する．5 章は結論について記述する．

2 章 物体検出と深度カメラ

ここでは本研究で使う物体検出と深度カメラについて記述する.

物体検出とは，入力画像や入力動画の中から特定の物体を検出するものである．物体の位置をバウンティングボックスという矩形で取り囲み，その物体がどのクラスにどの程度の確率で属しているのかというクラス確率を計算する．バウンディングボックスとクラス確率という２つの情報で物体の位置と属性を出力する．物体検出は画像分類，顔認証，自動運転など多くの分野で使われている．ディープラーニングを活用した物体検出が一般的であり，R-CNN や Fast R-CNN，Faster R-CNN，Yolo など何種類もの物体検出アルゴリズムが存在し，利用されている[1].

深度カメラとは，対象物までの距離情報を計測するカメラである．通常のカメラが被写体の明るさやカラー情報を撮影するのに対し，深度カメラは被写体までの距離情報を測定する．深度情報の計算手法は，大きく分けて，受動ステレオ方式，ToF 方式，アクティブステレオ方式がある．受動ステレオ方式は 2 台のカメラで構成されており，それらで同一の対象物を撮影する．２台のカメラの位置は異なるので，同じ対象物を写してもその見え方にズレが生じる．そのズレを用いて三角測量の原理で距離を計算する.

ToF は Time of Flight の略である．画像センサの近傍から赤外線を照射し，その赤外線が物体に反射し，カメラに戻ってくるまでの時間を計測する．反射して戻ってくるまでの時間から距離を計算する方式である．アクティブステレオ方式はプロジェクターなどの光源とカメラから構成される．光源から縞模様やランダムドットなどのパターン光を照射し、そのパターンの見え方から深度を計算する．深度カメラは自動運転やロボットの自律移動などに利用されており，コンピュータが我々の 3 次元空間を認識するのになくてはならない技術である.

2．1 物体検出の従来研究

現在の物体検出はCNN(畳み込みニューラルネットワーク)を利用したものが主流である．CNN を用いた物体検出アルゴリズムの先駆けとなったのが R-CNN である.

R-CNN とは Regions with Convolutional Neural Networks のことであり，矩形領域の提案と CNN での特徴検出を組みわせた物体検出アルゴリズムである．R-CNN では，入力された画像に対して，selective search で領域提案を行い，物体が存在すると思われる領域を 2000 個以上抽出する．それらの画像サイズを wrap 処理で 227×227 の正方形のサイズに変更し，CNN で処理することで，4096 次元の特長ベクトルを得る．4096 次元の特長ベクトルをサポートベクトルマシーンで分類する．R-CNN は高精度な物体検出が可能である．一方，2000 個以上ある領域を CNN で処理するため，長い計算時間を

要する．そのため，動画の解析やリアルタイム処理へ応用することが難しい．

2015 年に R-CNN の改良版として Fast R-CNN が開発された．Fast R-CNN は R-CNN と比較して，学習時間を 9 倍，推論時間を 213 倍に高速化させた．Fast R-CNN は R-CNN と同様に，selective search で領域提案を行い，オブジェクトの候補領域を抽出する．またあらかじめ入力画像全体に対して ImageNet で学習済みモデル(VGG 等)の CNN を使って特徴マップを計算する．そして先ほどの候補領域を特徴マップに投影する．特徴マップに投影された候補領域のことを ROI という．その ROI を ROI Pooling という処理で，一定のサイズにリサイズする．こうすることでクラス分類を行うネットワークへの入力が同じ次元数になるため，異なる次元数でクラス分類を行うよりも処理が軽くなり，結果として全体の処理時間も速くなった．

R-CNN と FastR-CNN では，候補領域を selective search で検出していた．この処理時間が遅いことが問題であった．これを改善するために，2015 年に Faster R-CNN が提案された．Faster R-CNN は候補領域の検出も CNN で行い，このネットワークを RPN(Region Proposal Network)と呼んだ．これによりさらに処理時間が短縮された．また Faster-RCNN は end to end 学習であるため，高精度でもある．

2. 2 Yolo, Yolox について

2016 年に Yolo が提案された．Yolo は You Look Only Once の略であり，検出窓をスライドさせる手法ではなく，画像全体を 1 度 CNN に通すだけで物体検出を行う．Yolo の１番の特徴は処理速度が速いことである．Faster R-CNN よりも速く，リアルタイムで物体検出が可能となった．図 2.1 は Yolo の処理の流れを示す図である．Yolo の処理の流れは以下の４つである[2]．

① 入力された正方画像を S×S のグリッドセルに分割する．
② それぞれのグリッドセルごとに複数のバウンティングボックスを計算する（セル内およびセルを囲むような複数のバウンティグボックス）．そのバウンティングボックスに対して、背景か物体かを評価する．0<=評価値<=1 であり，0 に近いほど背景，１に近いほど物体である．
③ それぞれのグリッドセルに対して，セルに写っている物体のクラス確率を求める．
④ その後，②のバウンティングボックスの結果と，③のグリッドセルのクラス分類結果を組み合わせ，最終結果を出力する．

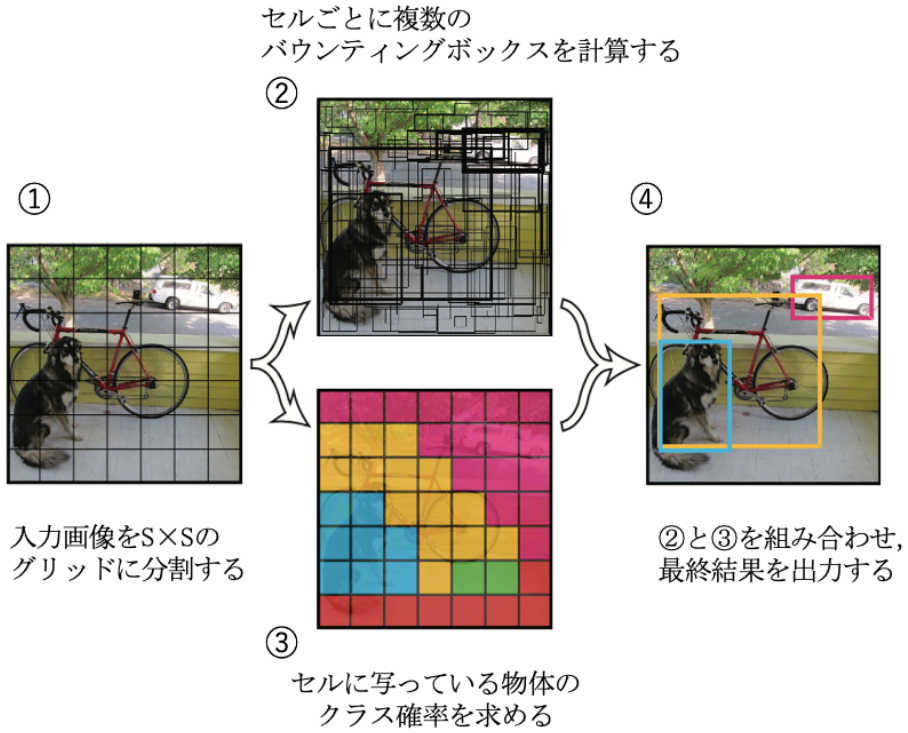

図 2.1 Yolo の処理の流れ

Yolo は，2016 年の発表以来，研究が継続されており，複数の Yolo のバージョン(Yolo シリーズ)が存在する．本研究では 2021 年に開発された Yolox[3]を使い研究を行う．従来のモデルはアンカーボックスという，中心と縦横比があらかじめ定義された矩形をテンプレートとして用意し，その矩形パターンから物体の形状を予測していた．このテンプレートを用いることにより，直接バウンティングボックスの大きさを計算する必要がなく，学習の安定や検出精度が向上していた．しかし，適当な形のアンカーボックスが存在しない場合にうまく検出ができないことや，アンカーの設計に関するパラメーターを用意する必要があり，CNN に入力するデータの次元数が増えてしまうという問題点があった．そこで Yolox は特徴マップから直接バウンティングボックスの幅，高さを予測することで，それまでにあったアンカーボックスの不一致や入力データの次元数が増えるという問題点を解決した[4]．従来の Yolo シリーズよりも処理が速くなり，精度も安定して良くなった．

2. 3　Intel RealSense D435

図 2.2　Intel RealSense D435

本研究で使用した深度カメラ Intel RealSense D435 を説明する（図 2.2）. Intel RealSense D435 は，2018 年に Intel 社が発売した深度カメラである．普通のカメラは対象物の明るさやカラー情報を撮影する．RealSense D435 は対象物までの距離情報を画像として計測する（図 2.3）. RealSense D435 は赤外線のパターン光を照射し，対象物の表面にできるその像を 2 台の赤外線カメラで撮影する．2 台のカメラで観察した像のズレから深度を求めるアクティブステレオカメラ方式である．奥行きはカメラから最大 10m までの距離を測ることができる．またグローバルシャッタを採用しており，歩いている人など，動きがある対象物に対しても正確な深度情報を計測することができる．正確な深度情報を測ることができ，またカメラの大きさも小さいため使いやすく，さまざまな分野で使用されている．利用例には，自立移動ロボットやロボットアームの制御，体を使った学習手段での部位認識や，バスケットボールでのシュートの成功率とシュート位置との関係性の解析などが挙げられる[5]. RealSense は，最新の深度カメラである Azure Kinect などよりも使用実績があり，本体の大きさは横幅 90mm，高さ 25mm，厚さ 25mm と小さく，スペースが限られている場所でも使用可能である．また拡張機能も豊富であり，Github などからいくつもの拡張機能をダウンロードし実行することができる．使用実績の高さ，拡張機能による応用のし易さ，またサイズが他の深度カメラよりも小さいため設置がし易いといった点から，本研究では RealSenseD435 を使用する．

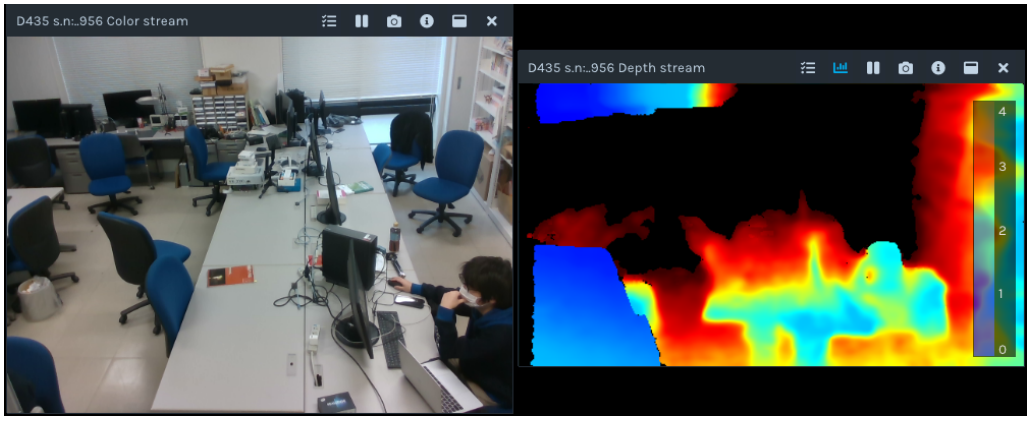

図 2.3 RealSense での撮影の様子．　左:RGB 画像 右:深度画像．

3章　システムの構築と実験方法

本研究では，Yolox と RealSense を使い，人体位置の 3 次元座標を測定し，記録・解析するシステムを構築する.以下の開発環境で,Anaconda で python の仮想環境を作り，研究を行った.

開発環境:

Windows version 10 Pro

Visual studio code

Anaconda 4.12.0

プログラミング言語:

Python 3.6.13

使用機器:

Intel RealSense D435

Windows PC

3. 1　Yolox と RealSense の実装

開発の基になる Yolox のプログラムを GitHub のサイト[6]からダウンロードする. code というボタンを押し，Download ZIP を押すと，YoloX で使うライブラリ一式をダウンロードすることができる. 解凍するとディレクトリが作成されるので，そのディレクトリに cd コマンドで移動する. そのターミナル上で

python tools/demo. py image -n yolox-x -c yolox_x. pth --path assets/dog. jpg --conf 0.25 --nms 0.45 --tsize 640 --save_result --device gpu

と入力するとサンプルプログラムの demo.py が実行される. 正しく動作すれば図 3.1 の結果が表示される. 基本的に Yolox は GPU を利用して実行されるが，GPU がない場合は CPU で実行される. demo.py では事前に用意した画像ファイルを処理しているが，PC にカメラを接続すると，カメラからのリアルタイム画像に対して，物体検出ができる.

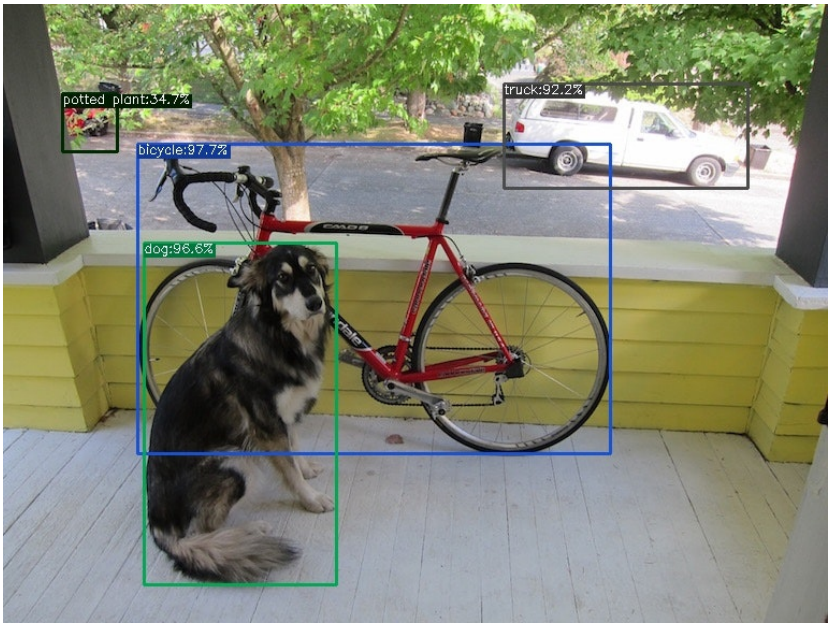

図 3.1 demo.py の結果

次に RealSense を python で動かすためのライブラリを Github の[7]のサイトからダウンロードする．code というボタンを押し，Download zip を押すと RealSense を動かすためのクロスプラットフォームライブラリをダウンロードできる.RealSense のライブラリはオープンソースであるため，自由にソースコードを書き換えて，RealSense の動作を変更することができる．

3．2 3 次元座標の測定

Yolox で人体を検出するとバウンディングに囲まれる．バウンティングボックスの中心点をその人がいる位置であると仮定し，中心点の 3 次元座標を測定する．

図 3.2 は人体の 3 次元座標を取得する流れをまとめた図である．RealSense は RGB 画像と深度情報を出力する．RGB 画像に対して Yolox を実行させ物体検出を行う．検出結果の中で，クラスが person(人)と判定されたバウンディングボックスの中心点のピクセル座標と，深度情報を使い，人体の 3 次元座標を計算する．RealSense のライブラリには，指定したピクセル座標の 3 次元座標を計算する関数が存在する．この関数を使用し，バウンディングボックスの中心点の 3 次元座標を取得した．

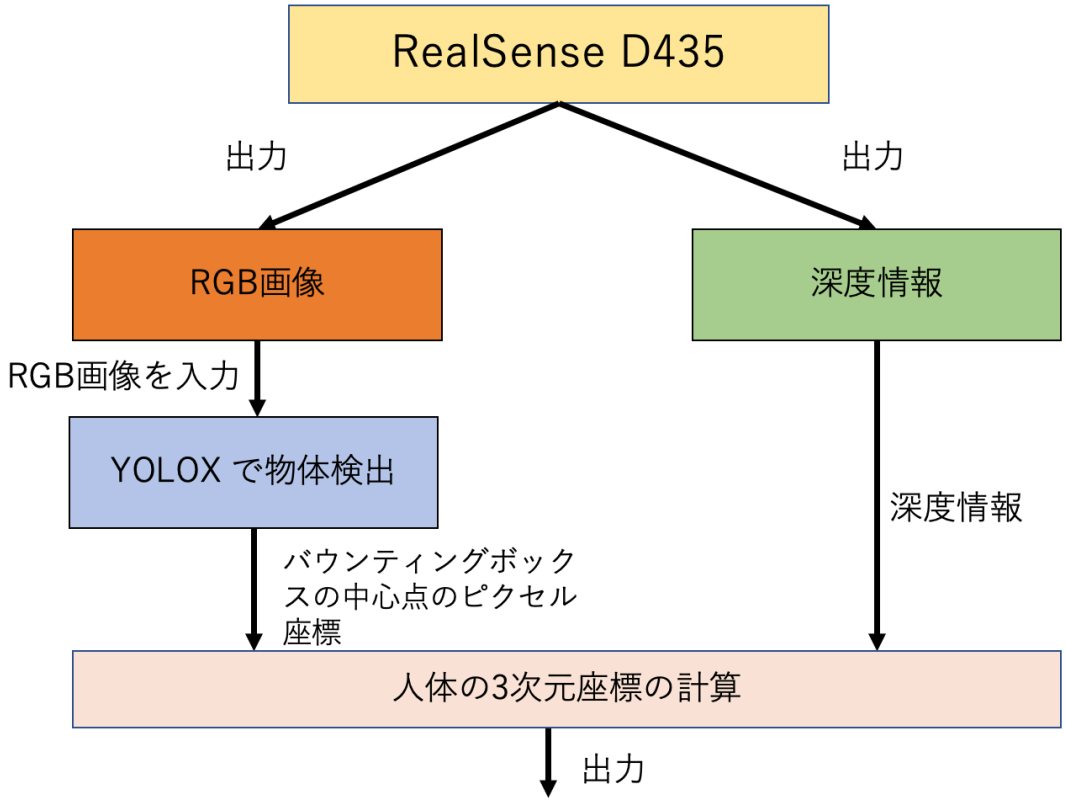

図 3.2　人体の 3 次元座標を取得する流れ

RealSense を中心とする 3 次元座標系は，原点を RealSense 本体の右端にある RGB カメラのレンズの中心とし，レンズに垂直で外側方向を Z 軸にとる．RealSense を正面から見て，原点から左横方向が X 軸の正の方向である．RealSense を正面から見て，原点から上側が Y 軸の正の方向である（図 3.3）．

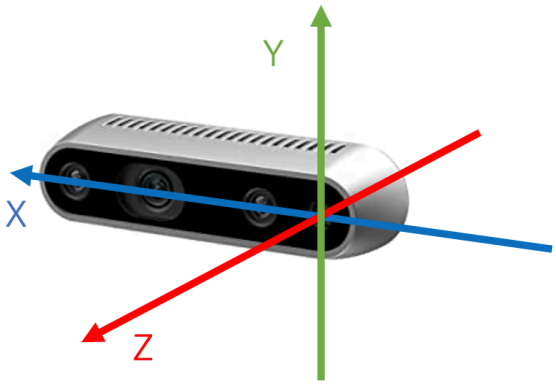

図 3.3 RealSense を中心とする 3 次元座標系

3. 3　物体追跡

本研究では，全体的な人体密度の可視化だけでなく，個人ごとにも人体密度の可視化を行う．個人の人体密度とは，長時間にわたる人体位置の測定データから，存在する時間が長かった位置を密度分布の形で示すものである．そのために，個人ごとの 3 次元座標を測定するとともに，人体の追跡を行う．画像処理による物体追跡は，動画の連続する画像間で同一物体に対して同じ id を割り当て続けることで行う．これにより，動画内に写っている人間の数や車の数を数えることができ，また個人の行動を追跡することができる．python には motpy という物体追跡のライブラリがある．motpy は Yolox で検出したバウンディングボックスの結果を使い，物体追跡を行う．図 3.4 は Yolox に motpy を実装した図である．バウンディングボックスの上に，分類クラス(person)と id が割り振られている．　オレンジ色で塗りつぶした小さい四角形はバウンディングボックスの中心点である．

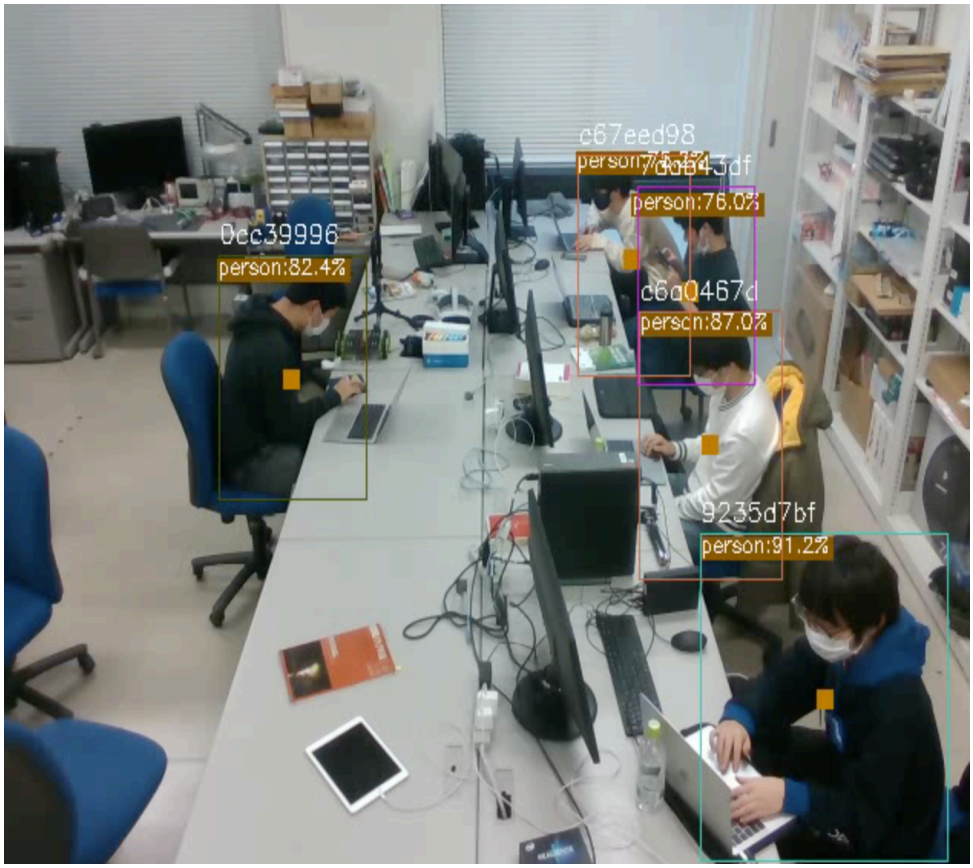

図 3.4 Yolox に motpy を実装した結果

3. 4　システムの構築

3.2 節で説明した 3 次元座標を用いて人体密度の推定を行う．Yolox と RealSense のプログラムはそれぞれ独立したソースコードで，同時に動作させることができない．そこで，RealSense を動かす python プログラム realsense.py を作成した．このファイルは RealSense を動かすソースコード全体をクラスとして定義しているファイルである．クラスとして定義することで，他のファイルからでも RealSense を動かすコードを呼び出すことができる．こうすることで，Yolox を動かすプログラムから RealSense を動かすクラスを呼び出し，Yolox を動かしながら，RealSense も同時に動作させることができた．RealSense の RGB 画像を Yolox で処理し，得られたバウンティングボックスのピクセル座標と RealSense の深度情報を使い，人体の 3 次元座標を取得した．作成したプログラムファイルのファイル名は Yolox_motpy.py である．そのファイルのソースコードの一部を図 3.5 に示す．

全体を while 文でループ化し，このループが終わるまで Yolox による物体検出と RealSense による 3 次元座標の取得を行う．539 行目の cap.read で RealSense からリアルタイムの RGB 画像と深度情報を変数 frame に格納する．540 行目から 542 行目で，変数 RGB_frame と depth_frame にそれぞれ RGB 画像と深度情報を格納し，color_intr に 3 次元座標を計算するためのカメラパラメータを格納する．

544 行目の predictor.inference(RGB_frame)で RGB 画像に対して推論を行い，結果を outputs に返す．その後，545 行目の predictor.visual 関数で結果の可視化を行う．546 行目の mot.track で物体追跡を行う．Mot.track の結果を tracks に格納する．tracks には Yolox で検出したバウンティグボックスの座標とクラス分類結果と物体追跡による id が格納されている．

547 行目で tracks の要素に対して，for 文でループさせ，548 行目の if trc[3]==0 でクラス分類結果が人間であるバウンティグボックスに対してのみ，その下の処理を行う．549 行目で変数 id_str に id を格納する．550 行目でバウンティングボックスの座標値に対して，Three_Dimensional_Coordinates 関数で 3 次元座標の計算を行い，結果を変数 threecor に格納する．551 行目では，rotate_x 関数で 3 次元座標に回転行列をかける．552 行目で df.append で 3 次元座標を pandas のデータフレームに保存する．pandas とは，独自のデータフレームによりデータ解析の作業を python でやり易くしてくれるライブラリである．553 行目で結果を csv ファイルに書き出す．

```
while True:
    ret_val, frame = cap.read()
    RGB_frame = frame[0]
    depth_frame = frame[1]
    color_intr = frame[2] #カメラパラメータ
    if ret_val:
        outputs, img_info = predictor.inference(RGB_frame)
        result_frame = predictor.visual(outputs[0], img_info, cap.depth_frame, color_intr, predictor.confthre)
        tracks = mot.track(outputs, img_info["ratio"], cap.depth_frame, color_intr)
        for trc in tracks:
            if trc[3] == 0:
                id_str = str(trc[0])
                threecor = Three_Dimensional_Coordinates(trc[1], cap.depth_frame, color_intr)
                threecor = rotate_x(threecor)
                df = df.append({"ID" : id_str, "X": threecor[0][0], "Y":threecor[0][1],  "Z":threecor[0][2]}, ignore_index = True)
                df.to_csv("tools/test_result_rere11.csv", encoding="utf-8", header=True, index=False)
                draw_track(result_frame, trc, thickness=1)

        if args.save_result:
            vid_writer.write(result_frame)
            cv2.imshow('frame', result_frame)
            ch = cv2.waitKey(1)
            if ch == 27 or ch == ord("q") or ch == ord("Q"):
                break
    else:
        break
```

図 3.5 作成したソースコードの一部

3. 5　実験環境

構築したシステムを用いて実験を行った．実験場所は蚊野研究室で，特別研究中の研究室メンバー5 人の 3 次元座標を Yolox の 1 回の推論ごとに記録した．1 回の推論時間は平均 0.25s である．PC と type-C ケーブルで繋いだ RealSense を，図 3.6 のように，扉の上のスペースに固定し，特別研究中の 5 人を観察した．図 3.6 で，扉の上の空いたスペースの上に置いてあるものが RealSense である．

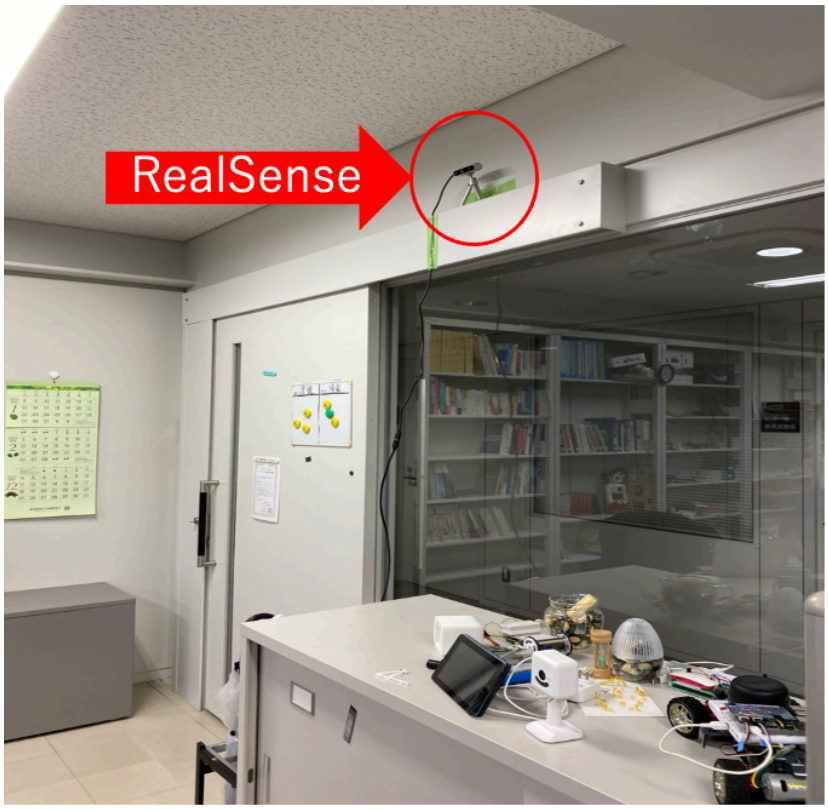

図 3.6　扉の上に置いた RealSense

このカメラから，図 3.7 のように研究室を観察できる．図 3.7 が RGB 画像であり，図 3.8 が深度画像である．

図 3.7　RGB 画像

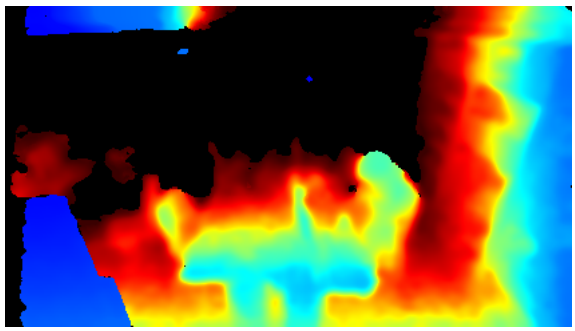

図 3.8 深度画像

カメラは 5 人の座席と身体が見えるように斜め下方向に向けている．そのため RealSense を中心とした座標系において，Z 軸は斜め下向きとなる．この座標系で 3 次元座標を測定し，真上から見下ろした平面での人体密度を表示することにする．そのための前処理として，計測した 3 次元座標を x 軸周りに回転させ，変換後の xz 平面が研究室の床面に平行になるようにする．そのための回転行列の式は式(3.1)である．ここで，θ はカメラの傾き角度である．今回の実験ではカメラを下向きに 30 度傾けたため，

$\theta = 30$として計算する．$(x,\ y,\ z)$が変換前の座標であり，$(x',\ y',\ z')$が変換後の座標である．

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \cdots (3.1)$$

3．6　カーネル密度推定による人体密度の可視化

得られた 3 時間分の 3 次元座標をもとに人体密度の可視化を行う．密度推定とは，ある変数 X について，大量の観察値から，連続的な分布密度を推定することである．大量の数値を数学的な関数で表現することで，解析が容易になる．

密度推定には，パラメトリックな推定とノンパラメトリックな推定がある．パラメトリックな推定とは，データの分布がある特定の分布モデルに従っていると仮定し，密度の推定を行うことである．代表的なパラメトリックな密度推定方法として，データの平均と分散共分散行列を求めることで，正規分布で近似する手法がある．これはデータが正規分布に従っていないと，正確な結果を得ることができない．ノンパラメトリックな推定とは，データがある特定の分布モデルに従っていないと仮定し，密度推定を行うことである．代表的なノンパラメトリックな密度推定方法としてカーネル密度推定がある．本研究では，測定する 3 次元座標データの分布が正規分布などの特定の分布モデルに従っているのかどうか不明瞭なため，データがある特定の分布モデルには従っていないと仮定し，ノンパラメトリックな密度推定方法のカーネル密度推定を用いた．図 3.9 はカーネル密度推定の計算方法を説明したものである．各標本データ（図では$-2.1,\ -1.3,\ -0.4,\ 1.9,\ 5.1,\ 6.2$）にバンド幅と呼ばれる赤色の山を張る．青色の山はその赤色の山の総和であり，その青色の山を密度推定の結果とする密度推定方法である．バンド幅を大きくすると，データ全体の普遍的な密度の結果が得られ，逆にバンド幅を小さくすると，局所的に詳細な密度を得ることができる．

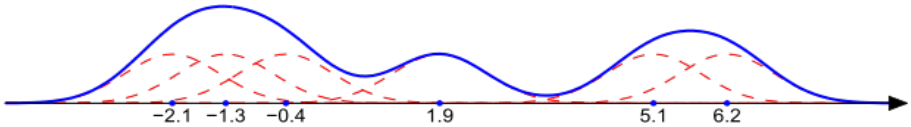

図 3.9 カーネル密度推定のバンド幅と密度推定の結果

4 章　実験結果

この章では第３章で説明した方法で実験を行なった結果を示す.

4. 1　出力ファイル

計測したデータは csv ファイル形式で出力する．個人の id，x 座標，y 座標，z 座標の 4 つの数値を 1 セットとして csv ファイルに保存した.図 4.1 は出力した csv ファイルである．A 列に id，B 列に x 座標，C 列に y 座標，D 列に z 座標が保存されている．Motpy が出力する id は，図 4.1 の A 列二行目の 9235d7bf-b3 のように長い英数文字である.今回の実験の被験者は５人だけなので，図 4.2 のように，わかりやすく A，B，C，D，E の文字に Excel で変換した．密度推定を行う前に誤検出のデータは削除した．また人体がフレームから外れると，外れる前とは違う id が付与されたので，手作業で id の振り直しを行なった.

	A	B	C	D	E
1	ID	X	Y	Z	
2	9235d7bf-b3	0.67323166	-0.6063513	2.05631898	
3	c6a0467d-0a	1.06271935	-2.3369601	4.11133442	
4	0cc39996-80	-0.9170109	-2.1159357	3.27130406	
5	c67eed98-23	0.8874073	-3.9983579	4.65208224	
6	7dab43df-2b	1.07284534	-3.224683	4.05491404	
7	c6805921-92	1.11408699	-1.6988992	3.37467059	
8	9235d7bf-b3	0.6995998	-0.6242622	2.11295082	
9	c6a0467d-0a	1.05501413	-2.3105202	4.08503054	
10	0cc39996-80	-0.9215809	-2.0972799	3.254362	
11	c67eed98-23	0.92831272	-4.1543285	4.91998966	
12	7dab43df-2b	1.07795572	-3.18741	3.93209603	
13	c6805921-92	1.141945	-1.7261258	3.41322246	
14	76671096-ac	1.0737437	-3.6297913	4.73092878	
15	9235d7bf-b3	0.67401731	-0.6062467	2.04829651	
16	c6a0467d-0a	1.06316745	-2.2934917	4.13643088	
17	0cc39996-80	-0.9011929	-2.0581767	3.19379978	

図 4.1 結果の出力ファイル

	A	B	C	D	E
1	ID	X	Y	Z	
2	A	0.67323166	-0.6063513	2.05631898	
3	B	1.06271935	-2.3369601	4.11133442	
4	E	-0.9170109	-2.1159357	3.27130406	
5	D	0.8874073	-3.9983579	4.65208224	
6	C	1.07284534	-3.224683	4.05491404	
7	A	0.6995998	-0.6242622	2.11295082	
8	B	1.05501413	-2.3105202	4.08503054	
9	E	-0.9215809	-2.0972799	3.25436201	
10	D	0.92831272	-4.1543285	4.91998966	
11	C	1.07795572	-3.18741	3.93209603	
12	A	0.67401731	-0.6062467	2.04829651	
13	B	1.06316745	-2.2934917	4.13643088	
14	E	-0.9011929	-2.0581767	3.19379978	
15	D	0.89935184	-4.0601401	4.75497623	
16	C	1.08106899	-3.19727	3.92640337	
17	A	0.68373781	-0.6138871	2.07737167	

図 4.2 id を変更

出力した csv ファイルを使い，カーネル密度推定を行った．python にある jointplot という関数を用い，推定結果の可視化を行った．jointplot 関数とは，seaborn という python のライブラリに実装されている関数であり，直観的にわかりやすくデータを可視化してくれるライブラリである.jointplot 関数はカーネル密度推定の他に,散布図,ヒストグラム，線形回帰などのグラフを表示できる．また id ごとにカーネル密度推定を行うことができるので，個人ごとにカーネル密度推定を行うことも容易である.

4. 2　全体の人体密度分布

３時間分のデータに対してカーネル密度推定を行なった結果を示す．図 4.3，図 4.4 は物体追跡を行なっていない場合の結果である.図 4.3 はバンド幅を小さめに設定し，図 4.4 はバンド幅を広めにとった結果である．X 座標，Z 座標ともに単位はメートルである．色が濃くなっている部分は，人体密度が高い部分である．(X,Z)=(0,0)位置に

RealSense がある．図 4.4 を見ると，(X,Z)=(１.0,2.7)と(X,Z)=(1.0，4.5)あたりの密度が高くなっている．図 4.3 はバンド幅を小さく取っているため，より細かな結果を確認できる．密度が高い位置は(X，Z)=(0.8，2.2)，(1.2，2.8)，(1.0，4.2)，(1.0，4.9)，(-1.0，3.0)の５つである．この５箇所で人体密度が高くなる理由は，これらの座標は RealSense から見た時の５人の研究室での座席位置であり，長時間椅子に座って研究していたためである．図 4.4 では大きく３つだけだった山が，図 4.3 では山が６つぐらいに分かれ，すこし歪な形になった．椅子にもたれかかった状態で研究をするなどして，体の位置がなんらかの動きで変わった結果が反映されているためである．

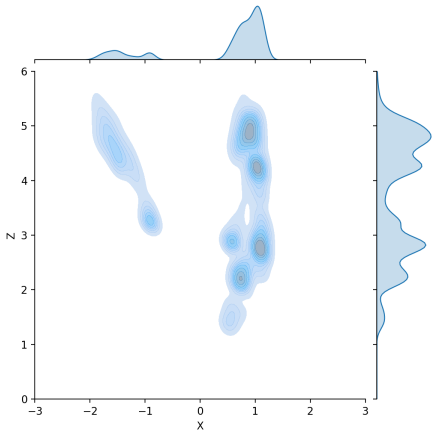

図 4.3 密度推定の結果（バンド幅小）

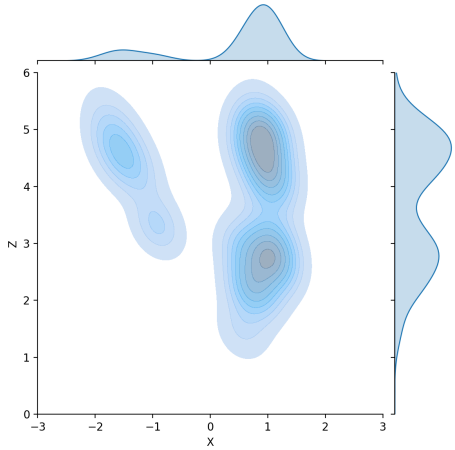

図 4.4 密度推定の結果（バンド幅大）

4．3　個人ごとの人体密度分布

人体の追跡を行い，id ごとにカーネル密度推定を行なった結果を図 4.5 から図 4.9 に示す．これらから，A，B，C，D，E の個人ごとにどの場所に長く留まっていたか，ということがわかる．全ての図でバンド幅は同じである．X 座標，Z 座標ともに単位はメートルである．(X,Z)=(0,0)位置に RealSense がある．

まず，全体的に言えることとして，分布密度が縦長になっている．これは，RealSense を，斜め 30°から観察するように設置したためであると考えている．図 3.7 から分かるように，RealSense は観察する空間を，やや横長の画像に収めている．この空間は，どちらかといえば縦長であるから，これを縦方向に圧縮して観察したために，縦方向の分布の誤差が大きくなり，全体として，分布密度が縦長になったものと考えられる．

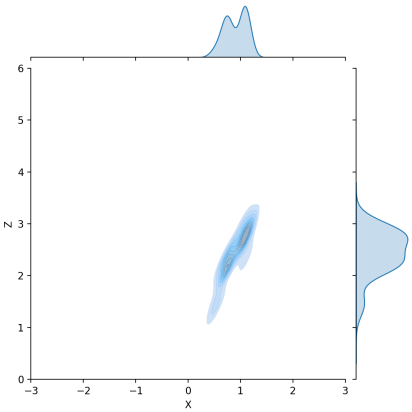

図 4.5 A の結果

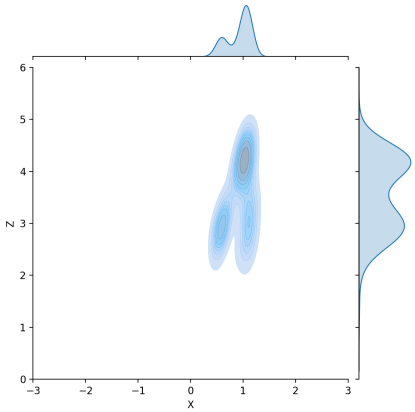

図 4.6 B の結果

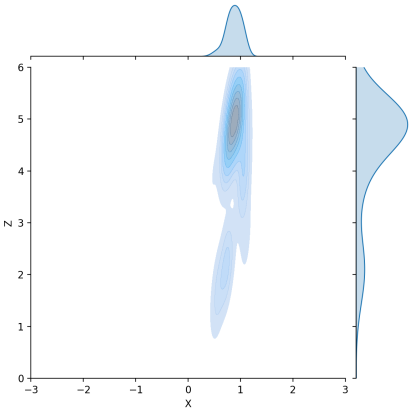

図 4.7 C の結果

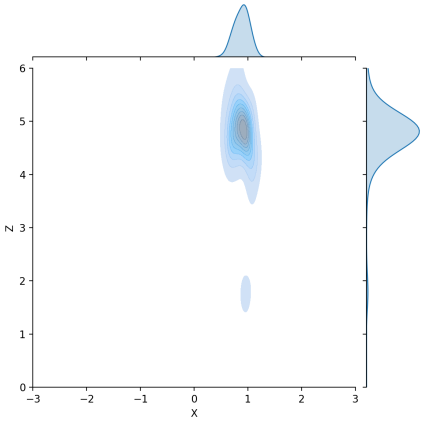

図 4.8 D の結果

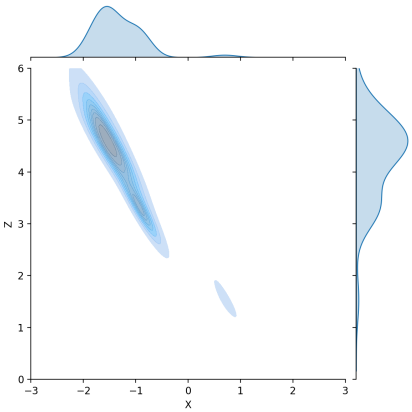

図 4.9 E の結果

図 4.5 の A の結果を確認する. RealSense から見た時, A の研究室内での座席位置は (X, Z)=(1.0, 2.5)あたりである. 図 4.5 の結果から, (X, Z)=(0.7, 2.2), (1.2, 2.7)で密度が高くなった. この理由は, A は特別研究中, 自分の座席に座って研究している時

間が長かったためである．また２ヶ所密度が高い理由は，研究中の体の姿勢の変化によるものである．前にのめり込んだり，後ろにもたれかかったりするなど，姿勢を変化させたためこのような結果になった．

図 4.6 の B の結果を確認する．RealSense から見た時，B の研究室内での座席位置は(X, Z)=(1.0, 3.5)あたりである．図 4.6 の結果から，(X, Z)= (0.7, 2.9)，(1.0, 2.9)，(1.0, 4.1)で密度が高くなった．(X, Z)= (0.7, 2.9)，(1.0, 2,9)での密度が高い理由は，B は A と雑談を行なっていたため，普段の座席位置よりも A の方に近寄り，研究を行なっていたためである．X 座標が 0.7 と 1.0 の２ヶ所の密度が高い理由は姿勢の変化である．(X, Z) = (1.0, 4.1)の密度が高い理由は，普段の座席位置に戻って研究を行なったためである．しかし座席位置は(X, Z)=(1.0, 3.6)で測定結果は(X, Z) = (1.0, 4.1)である.X の位置はまったく同じであるが,Z の値に 0.5m も誤差が生じてしまった．

図 4.7 の C の結果を確認する．RealSense から見た時，C の研究室の座席位置は (X, Z)=(1.0, 4.5)あたりである．図 4.7 の結果から，(X, Z)=(1.0, 4.9)で密度が高くなった．この理由は，C は特別研究中，自分の座席に座って研究している時間が長かったためである．また C の結果は Z 方向に他の人よりも大きく広がった．これは，C は特別研究中に座席を離れ，動いていたためである．中でも(X, Z)=(0.7, 2.0)で密度が高くなっているが，これは A の座席近くに移動し，雑談をしていたためである．結果から C は特別研究中，自分の座席だけでなく，他の場所に移動していたことがわかった．

図 4.8 の D の結果を確認する．RealSense から見た時，D の研究室の座席位置は(X, Z)=(1.0, 5.3)あたりである．図 4.8 の結果から，(X, Z)=(1.0, 4.8)で密度が高くなった．この理由は，D は特別研究中，自分の座席に座って研究している時間が長かったためである.また D は A と少しの時間,雑談を行なっていたため,A の座席位置に近い(X, Z)=(1.0, 1.9)で密度が高くなった．D と C の座席位置は近いが，D のほうがカメラよりも遠く，Z 座標が C よりも大きい値にならなければいけないが，最も密度が高い位置の Z 座標がほぼ同じになってしまった．これはカメラで撮影した時，C と D の体が被っていまい，うまく測定できなかったことが原因であると考える．

図 4.9 の E の結果を確認する．RealSense から見た時，E の研究室内での座席位置は(X, Z)=(-1.0, 3.5)あたりである．図 4.9 の結果から(X, Z)=(-1.0, 3.4)，(-1.5, 4.5)で密度が高くなった．(-1.0, 3.4)の密度が高い理由は，E は特別研究中，自分の座席に座って研究している時間が長かったためである．(-1.5, 4.5)の密度が高い理由は，教授の話を聞くためや,他の研究室メンバーの発表を聞くために体の向きを反対方向に向けていたためである．教授の位置は E が向いている位置と反対方向であるため，話を聞く場合は椅子を動かし逆方向を向かないといけない．その時，すこし斜め後ろ，つま

り負の X 方向と正の Z 方向に移動し，話を聞いていたため，(-1.5, 4.5)の密度が高くなったと考えた．また(X, Z)=(1.0, 1.5)でも密度が高くなった．これは E が A の座席近くに移動し，雑談を行なっていたため，A の座席位置に近い(1.0, 1.5)に少しだけ密度の分布が現れた．

特別研究中ということもあり，5 人とも自分の座席位置の座標で密度が高くなった．また，他の離れた位置で密度が高くなった人もいた．これにより，少しの間雑談を行った結果がうまく反映されていることがわかった．このように，人体の 3 次元座標の測定，また密度の可視化に成功した．しかし全体的に Z 方向に対して，測定の誤差があった．ほとんど動いていないのに，Z 座標が大きく変化してしまったという事例もあり，測定精度に関して，まだまだ改善する必要がある．

5章　結論

物体検出技術と深度カメラを用いた人体密度の可視化システムの構築に関する研究を行なった．その結果，システムを構築し，人体密度を可視化することができた．また物体追跡を行うことで個人ごとの密度の結果も可視化することができ，個人の行動履歴などを確認することができた．

本研究では，システムの構築に成功したが，いくつかの課題点が残っている．第一の課題は奥行き方向の測定精度が不十分なことである．実験結果によると，奥行き方向に最大で 0.5m 程度の誤差が発生した．その理由は，観察した空間に対して，RealSenseを30°の角度で見下ろすように設置したことであると考えている．実用化においては，RealSense の設置位置や設置角度，あるいは設置台数などを，用途に合わせて最適化する必要がある．

2 つ目の課題は人体が重なった時，3 次元座標を計算できない人体が生じるという点である．今回の実験では被験者が５人だけなのでほとんど重なることはなかったが，イベント会場など，人が多い状態で測定を行うと，人体の重なりが頻発する．１つのカメラだけでは，この問題を根本的に解決することは難しい．例えば，異なる角度から複数のカメラで計測すれば，改善できると考えている．

1 年間本研究を行い，物体検出のアルゴリズムや深度カメラについて勉強することができた．課題点が沢山残り，物足りなさを感じているが，これまでの研究で得た知識や技術を使い，これからの研究や情報技術の発展に努めていきたいと考える．

参考文献

[1] 原田達也，「画像認識」，講談社出版，2017 年

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, 「You Only Look Once: Unified, Real-Time Object Detection」2016 年 5 月 9 日
https://arxiv.org/pdf/1506.02640.pdf

[3] Zheng Ge ，Songtao Liu, Feng Wang, Zeming Li, Jian Sun, MegviiTechnology, 「YOLOX: Exceeding YOLO Series in 2021」
https://arxiv.org/pdf/2107.08430v1.pdf

[4] 物体検出手法の一つ YOLOX の概要まとめ | Hakky Handbook
https://book.st-hakky.com/docs/object-detection-yolox/

[5] Use Cases – Intel® RealSense™ Depth and Tracking Cameras
https://www.intelrealsense.com/use-cases/

[6] GitHub - Megvii-BaseDetection/YOLOX: YOLOX is a high-performance anchor-free YOLO
https://github.com/Megvii-BaseDetection/YOLOX

[7] GitHub - IntelRealSense/librealsense: Intel® RealSense™ SDK
https://github.com/IntelRealSense/librealsense

[8] RealSense を cv2.VideoCapture ライクに使う - Qiita
https://qiita.com/yumion/items/6eeb820c1f06839d57a7

謝辞

本論文を作成にあたり，丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　開発したプログラムとその説明

Yolox_motpy.py

Yolox を python で動かすためのファイル． RealSense の RGB 画像で物体検出を行い，得られたバウンティングボックスのピクセル座標と RealSense の深度情報を使い，人体の 3 次元座標の測定を行っている．物体追跡の Motpy も実装し，個人ごとに 3 次元座標を出力する．以下主に使う関数を説明する．

- predictor.inference
  RGB 画像に対して物体検出を行う関数
- predictor.visual
  predictor.inference で得た結果を可視化する関数
- mot.track
  物体追跡を行う関数
- Three_Dimensional_Coordinates
  3 次元座標を計算する関数
- Rotate_x
  Three_Dimensional_Coordinates で得た 3 次元座標に，回転行列をかける関数

realsense.py

RealSense を他ファイルから動かすためのプログラム． cap.start()で RealSense のキャプチャがスタートし，cap.read()で RealSense の RGB 画像，深度情報，カメラパラメータを取得する．作成するに当たり，ネットサイト[8]を参考にさせていただいた． 3 次元座標の計算に必要なカメラパラメータを読み込み、出力できるように改変した．