YOLO を用いた視覚障碍者支援デバイスの開発

関西大学　環境都市工学部　建築学科

建築環境工学第 I 研究室

建 18-35　清水　俊作

指導教官　豊田　政弘

目次

1　序論

1.1　研究背景

現在、視覚障碍者の方々が外出した場合、車や自転車、段差、用水路などの障害を回避し、安全に歩行するために白杖や盲導犬の利用、点字ブロック、音響装置付信号機の設置などの工夫がされている。これらは視覚障碍者が周囲の状況を把握するために必要不可欠なものになっている。

しかし、白杖や盲導犬を利用するには訓練が必要であり、点字ブロックは自転車などの障害物が置かれてしまうといった問題点がある。また、白杖や盲導犬の利用の際には、片手がふさがった状態である。そこで、視覚障碍者自身が音の再生装置を装着し、「音」によって常時感覚的に障害物との距離や位置、名前を把握することが出来れば、両手の空いた状態で活動することができ、それらの問題点が解決できると考えられる。

本研究では、上記実現のために YOLO v3 と音声合成を用いた視覚障碍者支援のためのデバイスを開発し、その有効性を検証する。

1.2　YOLO について

YOLO とは「You Only Look Once」の略称であり、日本語にすると「一目見るだけで良い」という意味である。YOLO は入力された画像や動画から物体の位置と種類を検出する AI モデルである。応用例としては、店舗における来店人数の確認や、レストランにおける満席率の調査、交通量調査、危険領域への立ち入り検出などに使用可能である。

本研究では、画像認識モデルとして YOLO v3 を用いる。

写真 1　検知した画面

写真 1 のように物体をリアルタイムで検出することが可能である。物体認識は事前に学習された学習モデルを用いて物体認識を行うため、学習されていない物体は認識されない。

本研究では、YOLO v3 のサンプルとして提供されている学習データを利用することし、それ以上の学習の更新はせず、Python 用のサンプルコードを修正することで、視覚障碍者のためのデバイスを開発する。

1.3　既往研究

以下に、視覚障碍者向け補助装置に関する既往研究を紹介する。

・高知大学での研究[1]

写真２の Xtion を腰に付け、壁や物への距離を測定し、両手首に装着した振動装置で着用者に警告するシステムである。また、白杖に右、左、中央の３つのボタンを取り付け、ボタンを押すと割りあてられた領域の探索を行い、障害物の有無と障害物までの歩数を音声で通知する。

写真２　Xtion

・萩原の研究[2]

PC に写真３の RealSense、写真４の骨伝導ヘッドフォンを接続し、障害物の検知をして使用者への通知を行う。音は、左、中央、右の３方向と各方向を上下に分割した 2×3 の６パターンで通知する。また、RealSense との互換性とプログラム作成の簡易性を考慮して、デバイスのためのシステム作成には python を使用している。

写真３　RealSense

写真４　骨伝導ヘッドフォン

・水﨑の研究[3]

［2］の既往研究で確認されたデバイスのサイズの大きさや、有線接続の問題を解決するために写真 5 の FRAMEALTO を採用して障害物を検知し、使用者へ通知する。BOSE 社の FRAMEALTO は、耳をふさがずに音が聞けるオープンイヤーオーディオサングラスで、Bluetooth が搭載されており完全ワイヤレス接続ができる。また、物体の上下の位置を表すのに、ラとミの 2 つの音ではなく、音階を使用したため、上下の位置もわかりやすくした。以下にある写真 6 は最も近くにある障害物を RealSense より取り込み、深度を検出するところである。

写真 5　FRAMEALTO

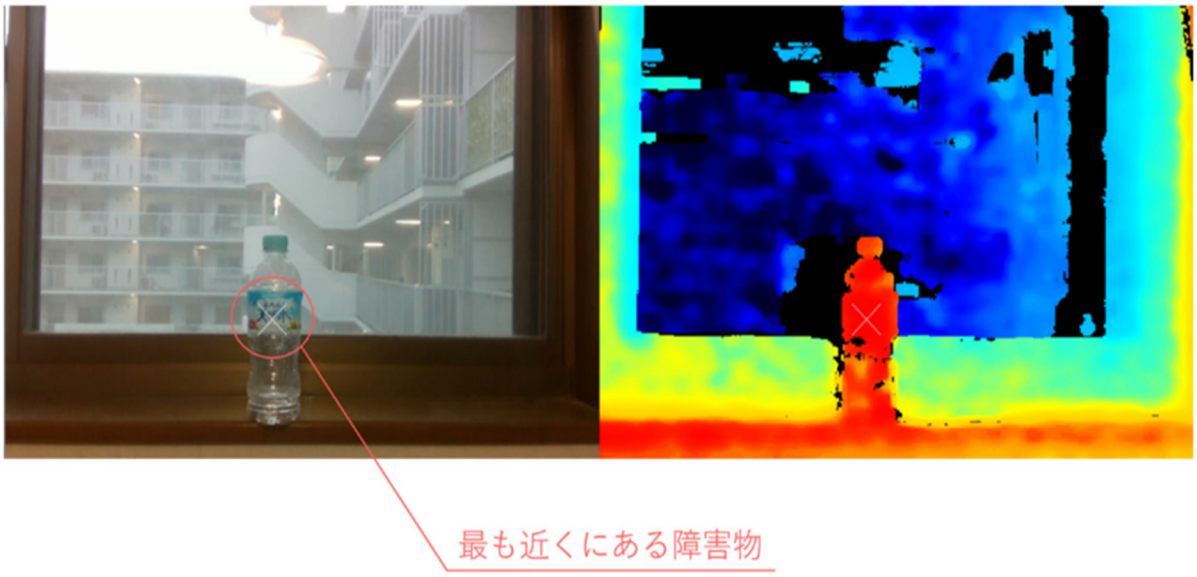

写真 6　RealSense により取り込んだデータ

1.4　研究目的

YOLO を使って物体を認識し、音声合成した「音」によって視覚障碍者が、より直感的に、 また正確に物体を認識し、行動できるようになるためのデバイスを開発することを目的とする。

1.5　既往研究の問題点と改善

1. 5 .1　問題点

既往研究[2][3]では一番近い物体の位置を音程に高低差のある機械音で知らせるが、検出された物体の名称や大きさが分からないので、視覚障害者の日常生活には不便な点がある。

1.5.2　提案

物体を認識できる画像認識システムを利用することを考える。現在、物体認識システムとして多くのモデルが存在しているが、YOLO v3 の特徴としては高精度な検出を可能としているため、このモデルを採用する。また、音声合成により、物体の名称を読み上げて提示する。これらにより、視覚障碍者の方でも身の回りの対象物が把握できる可能性がある。

2　研究方法

2.1　開発環境

本研究では、システムの処理装置としてノート PC を用いる。PC に FRAMEALTO を接続し、使用者への通知を行う。障害物の検知を行うカメラに関しては PC に搭載のカメラ、もしくは RealSense を使う。物体を認識するシステムには YOLO v3 を使う。また、YOLO v3 の環境構築には簡易性を考慮して、Python を使用する。以下に各機器の接続と使用機器を示す。

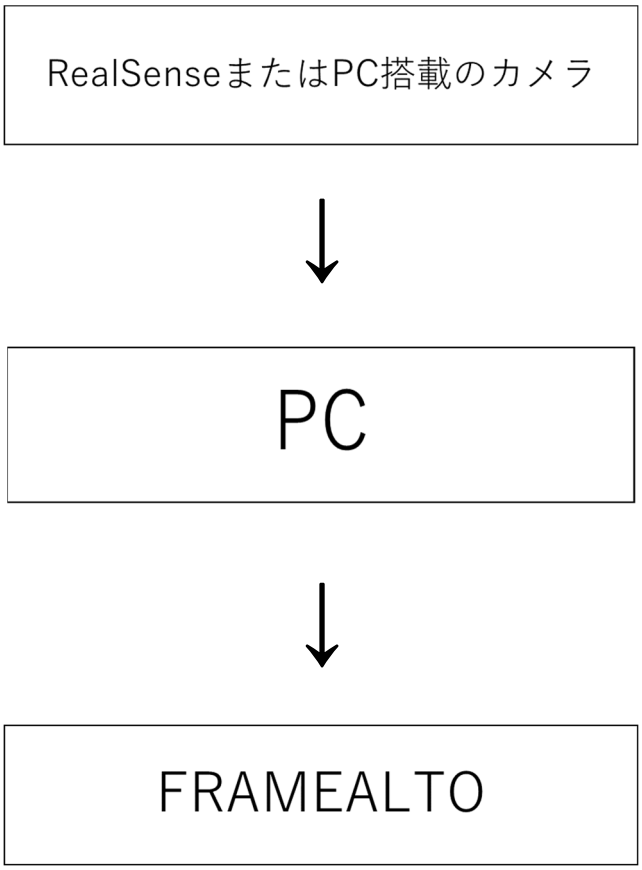

写真 7　RealSense

写真 7 が Intel 社の RealSense モデルD435 である。RGB センサと赤外線を用いた深度センサが 2 つ搭載されており、障害物との距離を測ることができる。尚、深度センサの最高解像度を以下に示す。

仕様
・深度センサ解像度／1280×720pixel
・ＲＧＢセンサ解像度／1920×1080pixel
・角度横／85°　　縦／58°
・距離計測範囲　 0.2～10ｍ
・サイズ　 90×25×25ｍｍ

写真 8　FRAMEALTO

写真 8 が BOSE 社が開発した FRAMEALTO である。これは耳をふさがずに音が聞けるオープンイヤーオーディオサングラスなので、使用者の聴覚の妨げにならない。それに加えて、Bluetooth が搭載されており完全ワイヤレス接続ができる。

2.2　提案システム

2.2.1　物体の検知

すでに学習済みの物体認識システムのモデルを導入したＰＣに RealSense を取り付け、（または PC 搭載のカメラで）障害物を読み取り、位置データと物体の大きさを取得する。

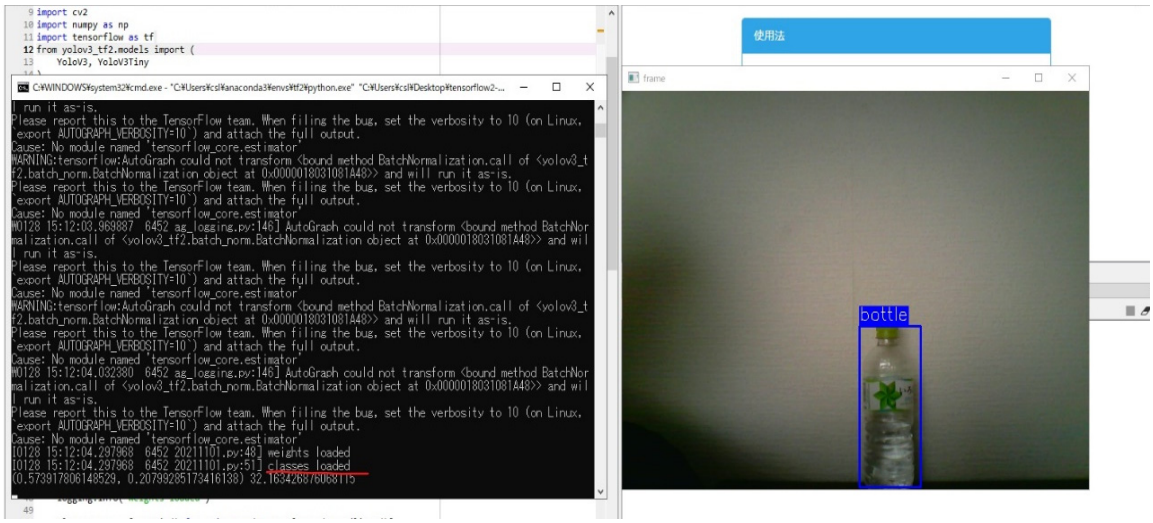

写真９　カメラにより取り込んだ位置データ

classes loaded の以下にある()内の数字が物体の位置、()の外の数字が物体の大きさをそれぞれ表している。()内の最初の数字が横軸、後の数字が縦軸を示している。位置については、検知された物体の名称が書かれた青枠の中心がその物体の位置と認識する。例として、Frame ウィンドウ画面真ん中で物体を検知すれば、(0.5, 0.5) になる。大きさについては、青枠の大きさを 0～100 の間で表している。実験を行っていく中で、画面全てを占めて表示する物体が表れなかった為、Frame ウィンドウ画面の 7 割程度を占める物体を最高数値である 100 に設定した。

以下にある図 1 は、物体の位置を縦横共に 5 分割して表している。実験の際、実験者はこの図 1 を使用し、物体の位置を認識する。

0.8以上~1.0					
0.6以上~0.8未満					
0.4以上~0.6未満					
0.2以上~0.4未満					
0~0.2未満					
縦軸/横軸	0 ~0.2未満	0.2以上 ~0.4未満	0.4以上 ~0.6未満	0.6以上 ~0.8未満	0.8以上 ~1.0未満

図１　物体の位置を表す図

2.2.2　使用者への通知

すでに学習済みの画像認識システムを使用し、物体を検知して名称が認識されると、音声合成により、女性の声で物体の名称を喋るように設定した。

検知した物体の左右の位置を知らせる方法については、パニングと呼ばれる、スピーカー間の音量差によって左右方向に音像を定位させる方法を使用する。Frame の画面に映る物体が左端にあれば、FRAMEALTO の左のみから聞こえるようにし、物体が中央に映っていれば、どちらからも聞こえるようにし、物体が右端にあれば、右のみから音が聞こえるようにした。

次に、検知した物体の上下の位置を知らせる方法については、喋るスピードの速さで表現した。Frame の画面に映る物体が上方向にあれば、速いテンポで喋らせ、物体が下方向にあれば、遅いテンポで喋らせるようにした。

最後に、検知した物体の大きさを知らせる方法については、喋る音の大きさで表現する。Frame の画面に映る物体が大きければ、大きい音で喋らせ、物体が小さければ、小さい声で喋らせるようにした。

図 2 に Python によるプログラムの一部を示す。

```
if np.sum(boxes[0][0].numpy()) != 0.0:
                box=boxes[0][0].numpy()
               # voice.Speak(class_names[int(classes[0][0].numpy())])
                fs = wincl.Dispatch("SAPI.SpFileStream")
                fs.Open("rokuon.wav", 3)
                voice.AudioOutputStream = fs
                voice.Volume = (((abs(box[3]-box[1]))+(abs(box[2]-box[0])))*60)
                voice.Rate = (-((box[1]+box[3])/2*10-5))　#-5~5 の範囲　5 は上
                voice.Speak(class_names[int(classes[0][0].numpy())])
                                #voice.Voice = oldv
                fs.Close()
                sound=AudioSegment.from_wav('rokuon.wav')
                panned=sound.pan(((box[0]+box[2])/2-0.5)*2) #-1 は左、0 はどっちも、
 1 は右
                play(panned)
```

図 2　python のコード（通知の部分）

音量と速さを調節した音声合成により作成された女性の声を rokuon.wav というファイル名で一度保存し、Frame に検知した物体の左右の中心を求める計算をした後、保存したフ

ァイルをパニングして再生し、使用者に通知する。

以下にある図 3 は、検知された物体の位置によって、被験者がどのように聞こえるかを表した図である。

+5~+3以上					
+3未満~+1以上					
+1未満~-1以上					
-1未満~-3以上					
-3未満~-5					
喋る速さ/パニングの割合	-1.0 ~-0.6未満	-0.6以上 ~-0.2未満	-0.2以上 ~+0.2未満	+0.2以上 ~+0.6未満	+0.6以上 ~+1.0

図 3　各領域に対応する音源

3　評価実験

3.1　実験 1　物体を認識したときの位置の精度実験

物体がどこに位置しているかの正確性を評価する。物体を検知したときに発生する音を FRAMEALTO から聞き取り、自分の耳に伝わってくる音によってどの程度正確に物体の位置を判断できるかを調べる。

実験方法

被験者の視野を下の図 4 の縦 5 マス横 5 マスのマス目に置き換え、最も対応していると思われるマスにチェックを入れてもらう。実験者は様々な大きさの物体を用意し、一種類ずつ配置する。

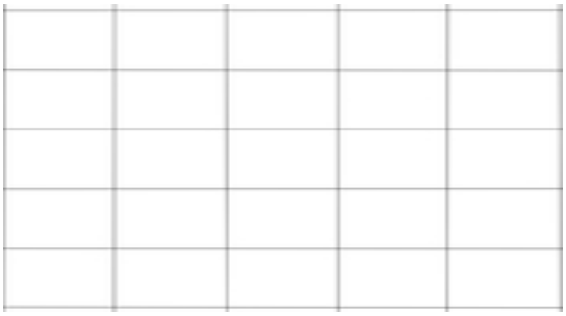

図 4　縦 5 横 5 のチェックシート

以上の工程を 5 人に行う。1 人 20 回行い、合計 100 データを得る。

3.2　結果 1

図 5 に実験結果を示す。1 番上の行は試行回数、2 番目の行には縦方向のズレ、3 番目の行には横方向のズレを示している。 2 番目の行に 1 が入っていれば、縦方向に 1 マスずれていたということになる。

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	1	0	1	0	1	1	0	1	1	0	1	0	1	2	0	0	0	1	0.55
横	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	1	1	1	0	1	0.6

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	1	1	1	0	1	1	0	0	0	0	1	1	0	2	2	1	2	1	0	1	0.8
横	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0.3

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	1	0	1	1	1	1	1	2	2	2	2	1	1	0	1	0	1	1	1	1
横	1	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1	1	0	1	0	0.4

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	2	1	1	1	0	2	1	2	3	1	0	0	1	0	2	0	0	1	0	1	0.95
横	0	1	1	0	2	0	0	1	1	1	0	1	0	0	1	1	1	2	0	0	0.65

21歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	1	1	0	2	1	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0.5
横	0	2	1	2	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0.4

図 5　実験結果 1

縦平均	0.76
横平均	0.47

図 6　実験結果 1

3.3　考察 1

図 6 に示す実験結果の平均値より、横方向、縦方向共にズレが 1 マス以下であることから、物体の位置はほとんど把握できたことが分かる。

縦方向のズレは 1 マス以下ではあるが、横方向に比べると少しズレが大きいため、喋るスピードのふり幅をもう少し大きくし、被験者に上下方向の位置をわかりやすくすることが必要である。

被験者からは、試行回数を重ねていくごとに、喋るスピードの標準がどれかわからなくなってきたという意見を多く得た。今後の実験では 5 回ごとに喋るスピードの標準を聞かせるなど、多少の訓練を実施すればもう少し上下位置の認識がうまくできたのではないかと予想される。

3.4　実験 2　物体を認識したときの大きさの精度実験

物体がどの程度の大きさなのかの正確性を評価する。物体を検知したときに発生する音の大きさを FRAMEALTO から聞き取り、自分の耳に伝わってくる音によってどの程度正確に物体の大きさを判断できるかを調べる。

物体の大きさは、写真 9 のように（）の後ろに来るので実験者はそれを読み取る。数値が 0〜20 未満は 1、20 以上〜40 未満は 2、40 以上〜60 未満は 3、60 以上〜80 未満は 4、80 以上〜100 は 5 とする。

実験方法

実験 1 と一緒に実験を進め、位置のチェックした後に、1 が最も小さい、2 が小さい、3 が普通、4 が大きい、5 が最も大きいと割り当て、別の用紙に 1 〜 5 段階で回答してもらう。

3.5　結果 2

図 7 に実験結果を示す。1 番上の行は試行回数、2 番目の行には大きさのズレを表している。2 番目の行に 1 が入っていれば、大きさが 1 段階ずれていることになる。

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
大きさ	1	0	1	1	1	1	1	1	0	0	0	1	0	2	2	1	0	0	0	0	0.65

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
大きさ	0	0	0	2	1	1	1	0	1	1	0	1	1	1	2	2	2	2	2	1	1.05

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
大きさ	0	0	2	1	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	2	0.5

22歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
大きさ	0	0	1	1	0	0	1	0	0	1	2	1	0	0	1	1	1	1	2	2	0.75

21歳男性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
大きさ	0	1	1	2	0	0	0	1	1	2	1	0	1	1	1	0	1	1	0	0	0.7

図 7　実験結果 2

大きさ平均	0.73

図 8　実験結果 2

3.6　考察 2

図 8 に示す実験結果の平均値より、大きさのズレが 1 段階以下であるため、物体の大きさがほとんど認識できたことが分かる。

問題点としては、連続で 20 回のデータを取ったため、一度間違えてしまうと標準の音の大きさが分からなってしまうことである。実験結果をよく見てみると、どこかで気づかなければ、間違え続けてしまうことがデータから読み取れる。そのため、考察 1 でも書いたように、5 回ごとに標準の音の大きさを聞いてもらうような訓練が必要であると思われる。

4　総括

4.1　まとめ

YOLO を用いた視覚障害者支援デバイスを視覚障害者の方々が街中で実用化するには、障害物の認識の精度をより向上させることが求められており、改良すべき点がまだ多くあると考える。

本研究では、過去に行われていた研究のシステムを参考にしながら、新しいシステムである YOLO v3 を取り入れることで、日常生活においてより便利に使用することができるようになったと思われる。

実験を通して、物体の上下の位置と大きさは、図 2 にある voice.Volume と voice.Rate の式にある数値を変えていけば、認識精度の向上が期待できる。

実験後には、FRAMEALTO の出す音にもっと慣れていけば、物体の認識の精度もさらに向上するだろうと感じた。

4.2　今後の展開

本研究では、物体の名称、位置、大きさの認識のみを考えていたため、歩行などをするにはまだまだ改善するべき点が多くある。例としては、段差は物体として検知しないため、非常に危険になると考えられる。

また、物体検知の精度も上げていく必要がある。本研究で使用したモデルはすでに学習済みのモデルなので、詳しい認識ができるような、使用者個人の暮らす環境に合った独自の学習モデルを作ることも検討すべきである。

参考文献

[1] 篠原克麻、森雄一郎、“視覚障害者のための白杖型歩行支援デバイスの開発”、高知大学卒業論文、2016 年、URL(http://trick.is.kochi-u.ac.jp/Vol08/article01.html)

[2] 萩原洋平、“RealSense と音響信号を用いた視覚障害者支援デバイスの開発”、関西大学卒業論文、 2019 年

[3] 水﨑景太、“RealSense と音響信号を用いた視覚障害者支援デバイスのシステム改善と性能評価”、関西大学卒業論文、 2020 年

[4] 高校数学からはじめるディープラーニング補足情報
URL(https://dlbb1.blogspot.com/)

Python コード

```
import pyrealsense2 as rs
import matplotlib.pyplot as plt
import time
from absl import app, flags, logging
from absl.flags import FLAGS
import cv2
import numpy as np
import tensorflow as tf
from yolov3_tf2.models import (
   YoloV3, YoloV3Tiny
)
from yolov3_tf2.dataset import transform_images
from yolov3_tf2.utils import draw_outputs

#import subprocess
import win32com.client as wincl
from pydub import AudioSegment
from pydub.playback import play

voice = wincl.Dispatch("SAPI.SpVoice")

flags.DEFINE_string('classes', './data/coco.names', 'path to classes file')
flags.DEFINE_string('weights', './checkpoints/yolov3.tf',
                    'path to weights file')
flags.DEFINE_boolean('tiny', False, 'yolov3 or yolov3-tiny')
flags.DEFINE_string('image', './data/girl.png', 'path to input image')
flags.DEFINE_string('output', './output.jpg', 'path to output image')
flags.DEFINE_boolean('webcam', False, 'image or webcam')

# ストリーム(Color/Depth)の設定
config = rs.config()

config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
```

```
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

port_num = 8002

def main(_argv):
    if FLAGS.tiny:
        yolo = YoloV3Tiny()
    else:
        yolo = YoloV3()

    yolo.load_weights(FLAGS.weights)
    logging.info('weights loaded')

    class_names = [c.strip() for c in open(FLAGS.classes).readlines()]
    logging.info('classes loaded')

    if not FLAGS.webcam:

        img = tf.image.decode_image(open(FLAGS.image, 'rb').read(), channels=3)
        img = tf.expand_dims(img, 0)
        img = transform_images(img, 416)

        t1 = time.time()
        boxes, scores, classes, nums = yolo(img)
        t2 = time.time()
        logging.info('time: {}'.format(t2 - t1))

        logging.info('detections:')
        for i in range(nums[0]):
            logging.info('¥t{}, {}, {}'.format(class_names[int(classes[0][i])],
                                               scores[0][i].numpy(),
                                               boxes[0][i].numpy()))

        img = cv2.imread(FLAGS.image)
        img = draw_outputs(img, (boxes, scores, classes, nums), class_names)
```

```
    cv2.imwrite(FLAGS.output, img)
    logging.info('output saved to: {}'.format(FLAGS.output))

else:
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame1 = cap.read()

        img = cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB)
        img = tf.expand_dims(img, 0)
        img = transform_images(img, 416)
        boxes, scores, classes, nums = yolo(img)

        frame2 = draw_outputs(frame1, (boxes, scores, classes, nums), class_names)
        cv2.imshow('frame', frame2)

        if np.sum(boxes[0][0].numpy()) != 0.0:
            box=boxes[0][0].numpy()
            # voice.Speak(class_names[int(classes[0][0].numpy())])
            fs = wincl.Dispatch("SAPI.SpFileStream")
            fs.Open("rokuon.wav", 3)
            voice.AudioOutputStream = fs
            voice.Volume = (((abs(box[3]-box[1]))+(abs(box[2]-box[0])))*60)
            voice.Rate = (-((box[1]+box[3])/2*10-5)) #-5~5 の範囲　5 は上
            voice.Speak(class_names[int(classes[0][0].numpy())])
                    #voice.Voice = oldv
            fs.Close()
            sound=AudioSegment.from_wav('rokuon.wav')
            panned=sound.pan(((box[0]+box[2])/2-0.5)*2) #-1 は左、0 はどっちも、1 は右
            play(panned)
            print((abs((box[2]+box[0])/2)),(abs(1-((box[3]+box[1])/2))))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

```
if __name__ == '__main__':
    try:
        app.run(main)
    except SystemExit:
        pass
```