NARO 農研機構

MATLAB EXPO 2023
2023年5月31日

画像認識AIとLiDARを用いた農業機械周辺における人検出

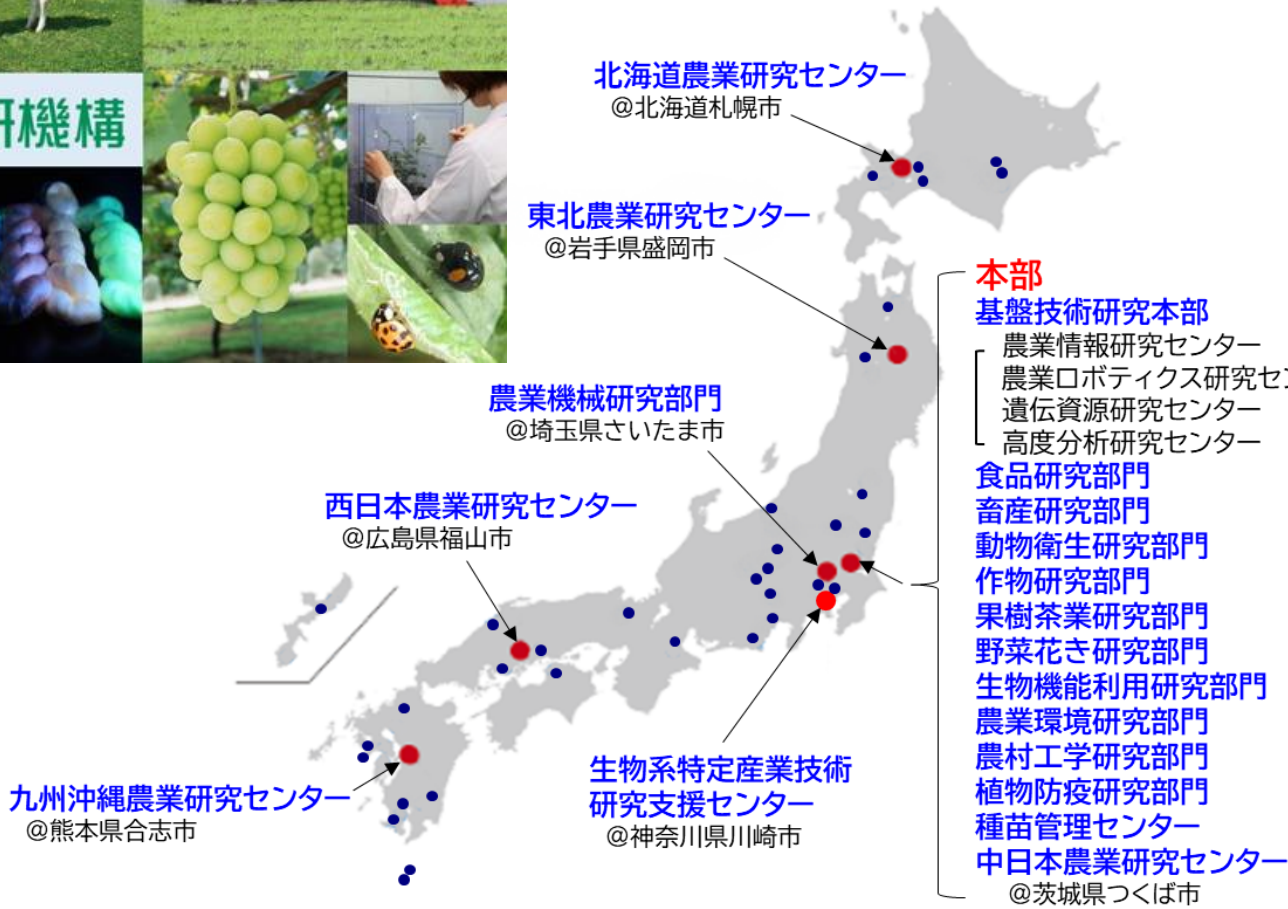
国立研究開発法人 農業・食品産業技術総合研究機構
農業機械研究部門 システム安全工学研究領域
梅野　覚

NARO

農研機構の紹介

農研機構とは

NARO 130th Anniversary



我が国の農業と食品産業の発展のため、
基礎から応用まで幅広い分野で研究開発を行う機関

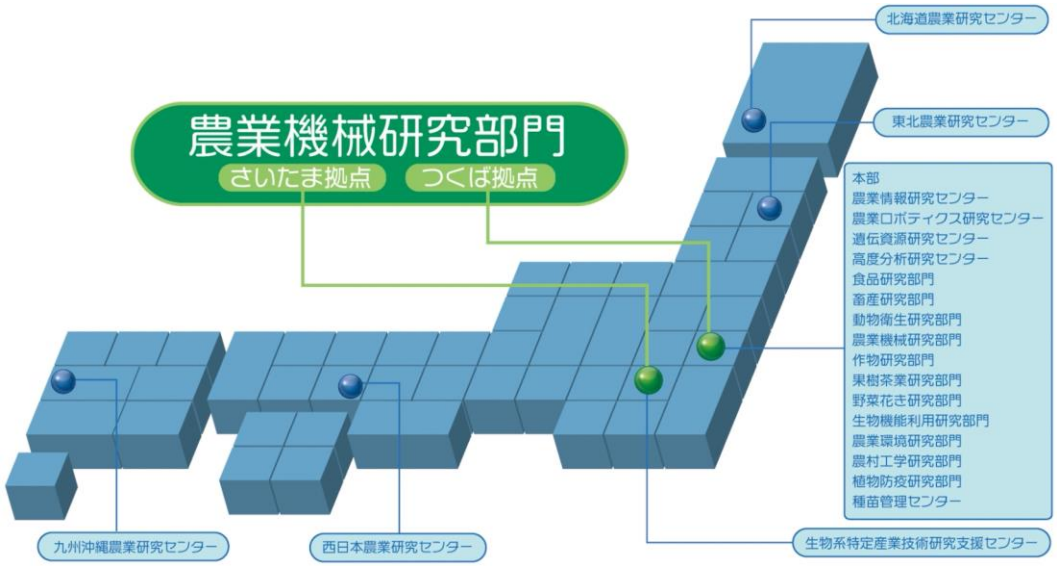
農研機構とは

出典：広報誌NARO No.29
https://www.naro.go.jp/publicity_report/publication/laboratory/naro/quarterly-newsletter/157966.html

- 日本の農業と食品産業の発展のための研究開発を実施
- 農林水産省の試験研究機関を統合して2001年に設立
  ※起源は1893年(明治26年) 農事試験場
- 予算額 742億円(2021年度決算)
- 職員数 3,279名(2022年4月)〔うち研究職員 1,747名(53.3%)〕

農研機構で働く
人材を募集中！

農業機械研究部門とは



自動運転田植機　ロボットトラクタ

農作業事故体験VR

実施する研究と農業機械研究部門の役割

高能率・安全スマート農業の構築と国際標準化の推進

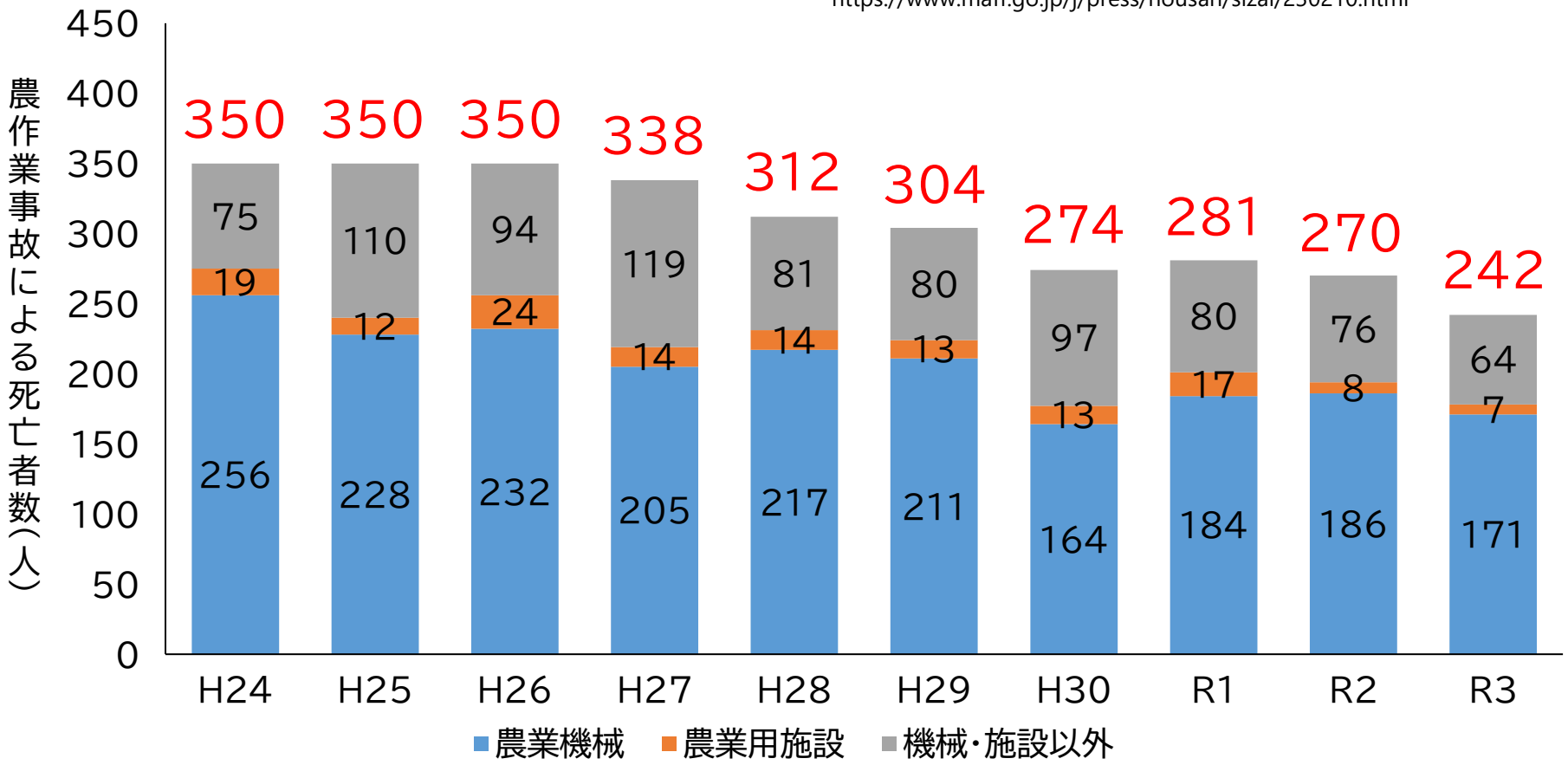
- 知能化農機の開発と国際標準化の推進
- 小型電動ロボットを核とする無人化農業の実現
- AIと人の融合による事故ゼロに向けた農作業安全システムの構築

安全性検査

農作業事故の発生状況
本課題の目標

農作業死亡事故の発生状況

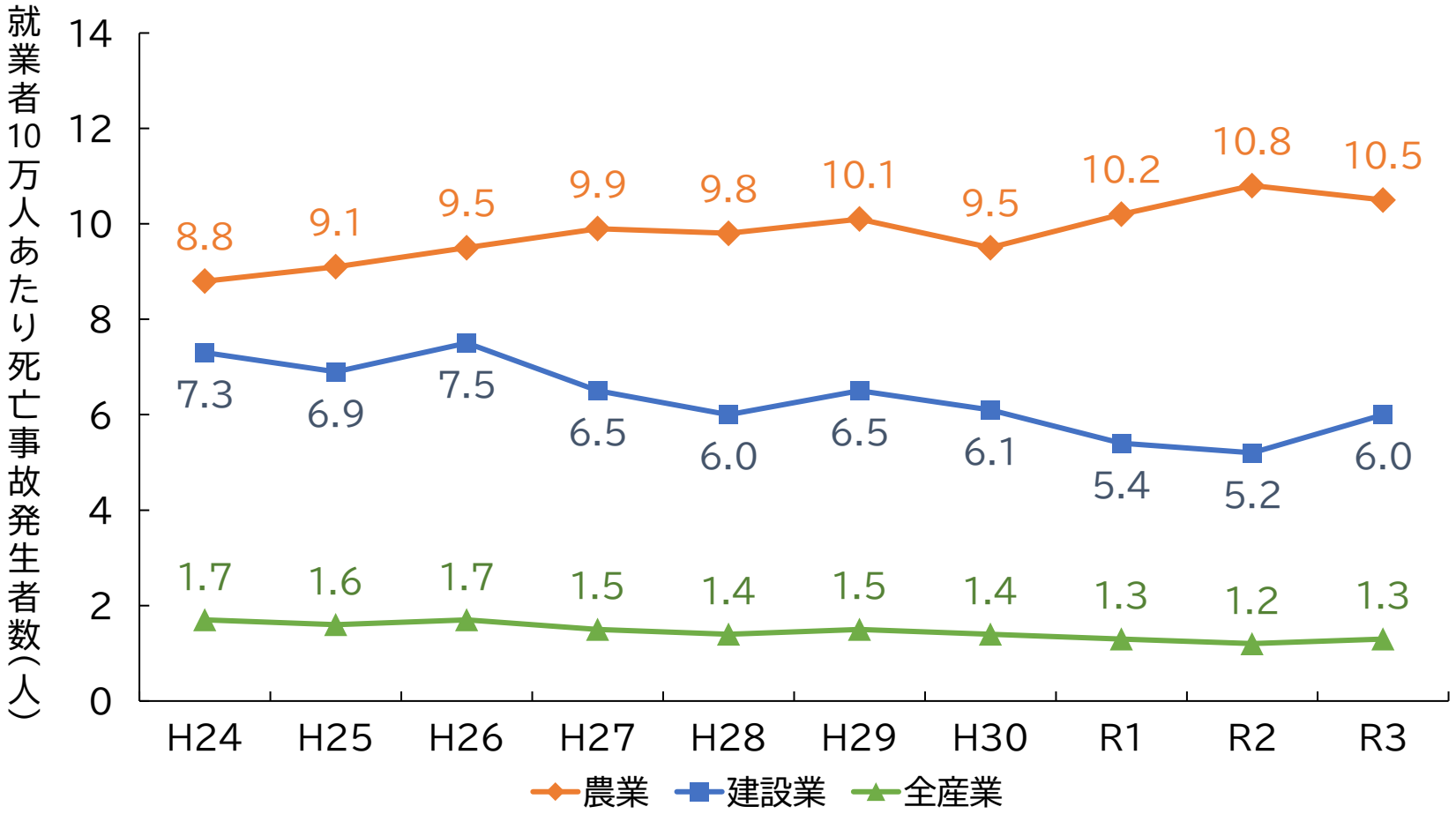
出典：農林水産省、令和3年に発生した農作業死亡事故の概要
https://www.maff.go.jp/j/press/nousan/sizai/230210.html



<u>毎年約240～350人</u>の農作業死亡事故が発生
農業機械に関する事故が**<u>約6～7割</u>**を占める

農作業死亡事故の発生状況

出典：農林水産省、(参考)10万人あたり死亡事故発生者数の推移
https://www.maff.go.jp/j/press/nousan/sizai/230210.html

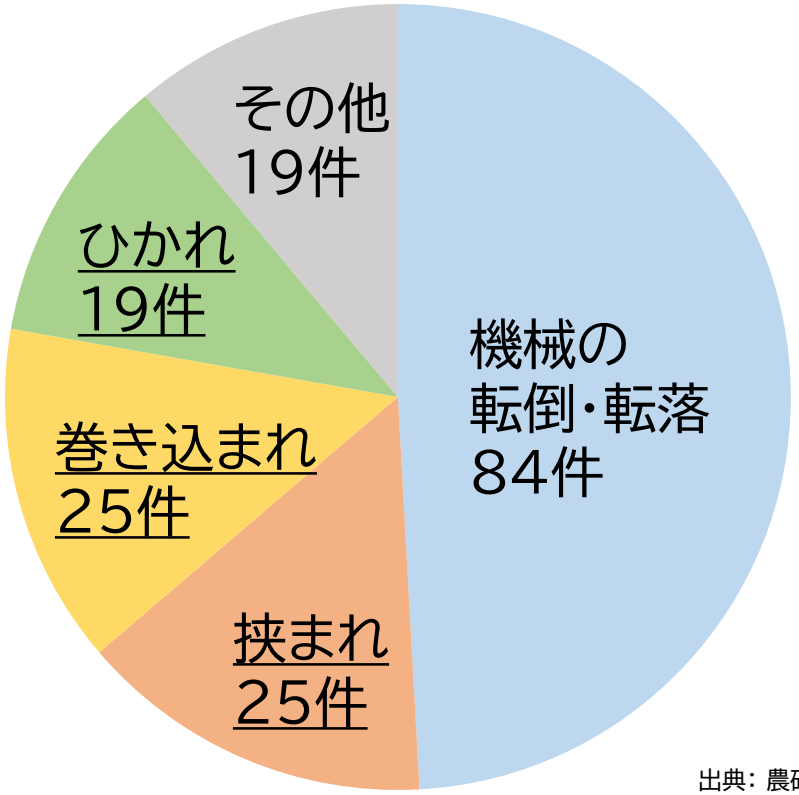


就業者10万人あたり死亡事故発生者数は**建設業の約1.8倍、全産業の約8倍**

農業は危険な産業

農業機械に関する死亡事故内訳

出典：農林水産省、令和3年に発生した農作業死亡事故の概要
https://www.maff.go.jp/j/press/nousan/sizai/230210.html



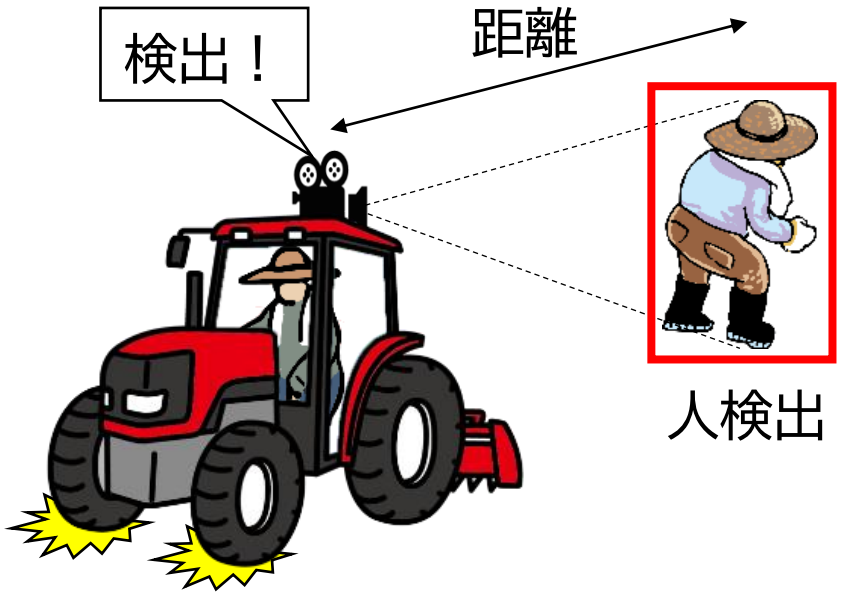
安全キャブ　安全フレーム

出典：農研機構、農作業安全情報センター
https://www.naro.affrc.go.jp/org/brain/anzenweb/index.html

農業機械との接触する事故（挟まれ、巻き込まれ、ひかれ）事故は多い
（機械の転倒・転落事故：安全キャブ・フレーム装備、シートベルト装着により身を守ることが可能）

接触事故の一因：　運転者が周囲の作業者に気付きにくい

目標



（例）事故リスクに応じて速度を変更

最終目標

農業機械がセンサにより作業者の情報（人検出、距離等）を取得し、
それらの情報に基づいて事故リスクを判断し、
安全行動（危険度通知、機械制御等）を行うシステムを開発する

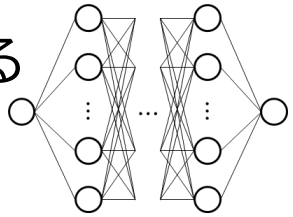
課題目標

作業機周辺への人の接近を精度80%以上で検出する装置を
開発する

全体概要

カメラ画像データ

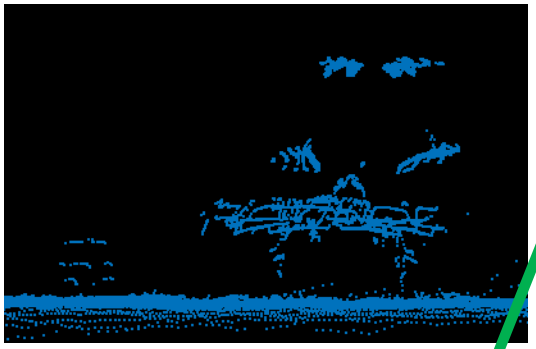
AIによる
人検出



学習
&
検証

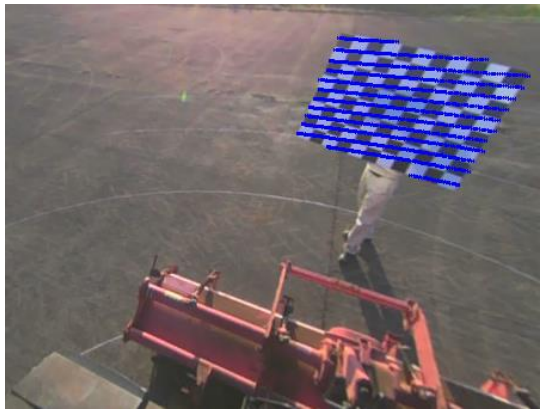
取得

取得



3次元点群データ

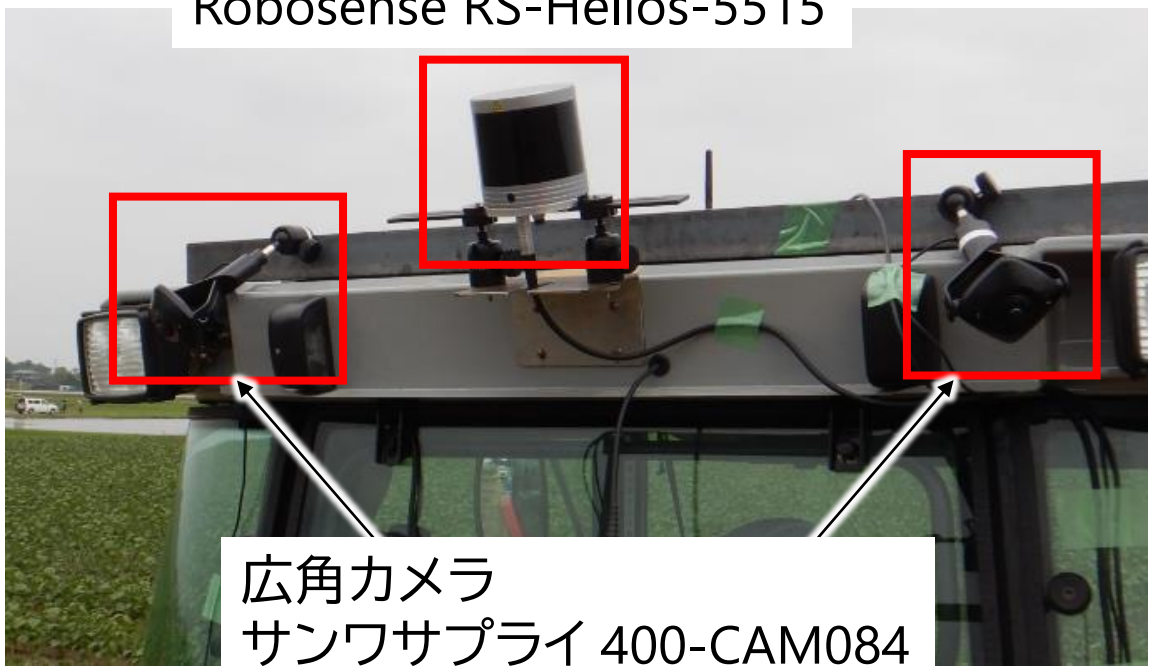
カメラ画像＆
3次元点群データの
キャリブレーション



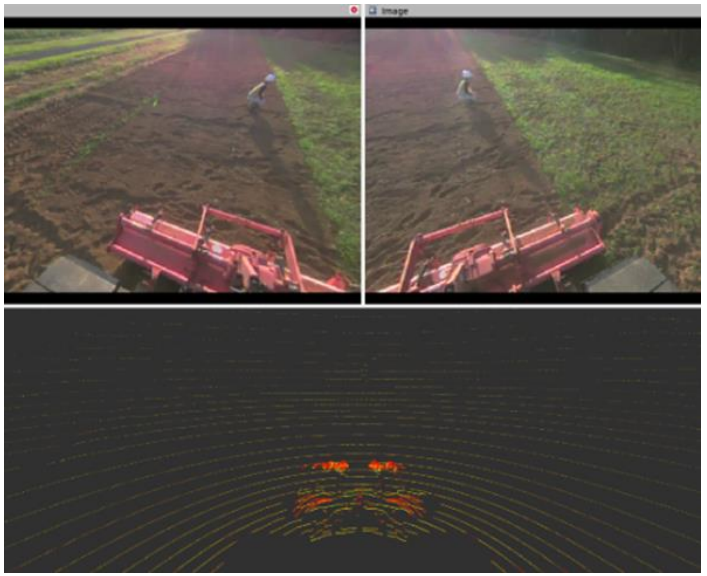
距離検出

現時点ではオフライン上で
人＆距離検出を検証

カメラ＆3DLiDARの設置、データ取得



乗用型トラクタ 62.5kW

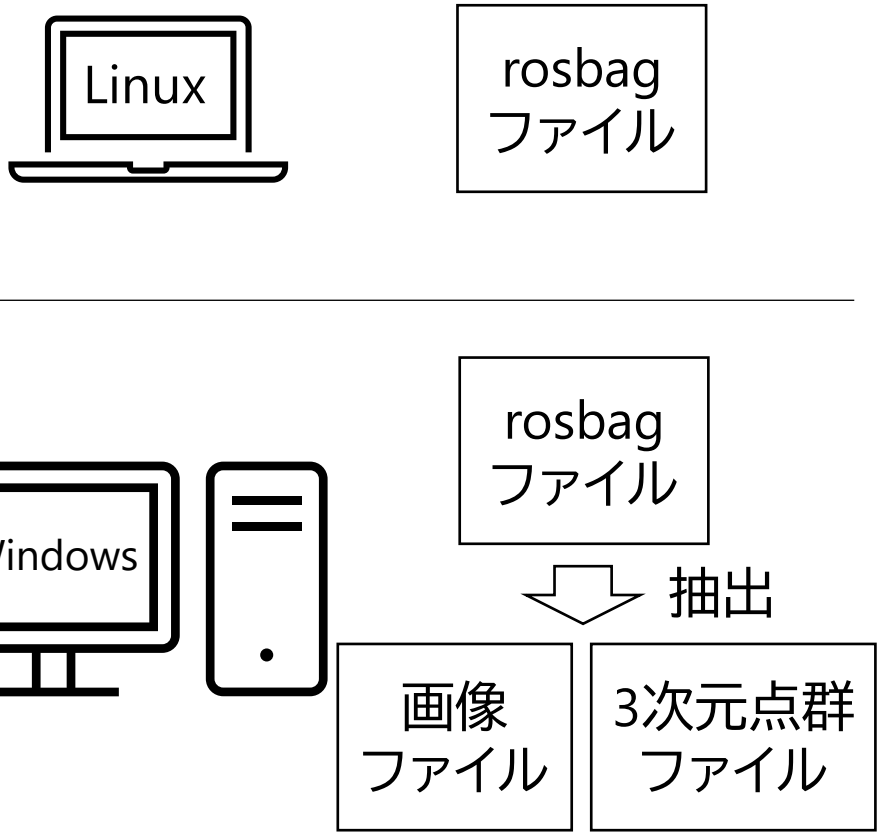


rvizによる可視化

ROSにより
- 両カメラの動画
- 3DLiDARの3次元点群

のrosbagデータ取得＆保存

rosbagデータからの抽出



抽出をROS Toolboxにて実施

参考URL：
Work with rosbag Logfiles
https://jp.mathworks.com/help/ros/ug/work-with-rosbag-logfiles.html

（プログラム一部抜粋）

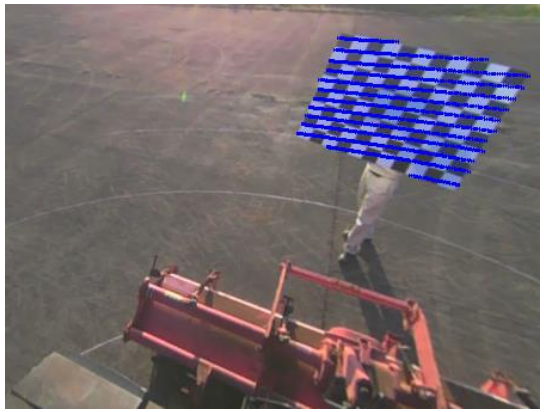
```
for kk = 1:filenum
    clearvars -except kk filenum filepath filename_v
    bagfile = fullfile(filepath, filename_v{kk});
    bagMsgs = rosbagreader(bagfile);

    % cameraデータ取得
    camera_Msgs1 = select(bagMsgs,Topic='/usb_cam1/image_raw2');
    camera_Msgs1 = readMessages(camera_Msgs1,'DataFormat','struct');
    camera_Msgs2 = select(bagMsgs,Topic='/usb_cam2/image_raw2');
    camera_Msgs2 = readMessages(camera_Msgs2,'DataFormat','struct');

    % lidarデータ取得
    lidar_Msgs = select(bagMsgs,Topic='/rslidar_points2');
    lidar_Msgs = readMessages(lidar_Msgs,'DataFormat','struct');

    targetdirname = erase(filename_v{kk},".bag");
    targetdir = fullfile(filepath, targetdirname);
    camera1dir = fullfile(targetdir, "camera1");
    camera2dir = fullfile(targetdir, "camera2");
    lidardir = fullfile(targetdir, "lidar");

    mkdir2(targetdir);   %フォルダ作成
    mkdir2(camera1dir);
    mkdir2(camera2dir);
    mkdir2(lidardir);

    imagemaxnum = min([length(camera_Msgs1);length(camera_Msgs2);length(lidar_Msgs)]);
    n = 1;     %10/n(Hz)で画像と3dデータ取得
    k = 1;

    for i = 1:imagemaxnum
        if i == (k - 1) * n + 1     %10/n(Hz)で画像と3dデータ取得
            camera_image1 = rosReadImage(camera_Msgs1{i});
            camera_image2 = rosReadImage(camera_Msgs2{i});

            filename = sprintf('%05d',k);
            fullname1 = append(camera1dir,"\",filename,".png");
            fullname2 = append(camera2dir,"\",filename,".png");
            imwrite(camera_image1,fullname1);
            imwrite(camera_image2,fullname2);

            lidar_points = rosReadXYZ(lidar_Msgs{i});
            lidar_intensitys = rosReadField(lidar_Msgs{i},'intensity');
            ptCloud = pointCloud(lidar_points,'Intensity',lidar_intensitys);
            ptfullname = append(lidardir,"\",filename);
            pcwrite(ptCloud,ptfullname);
```

画像認識AIによる
人検出の検証

画像認識AIによる人検出

取得

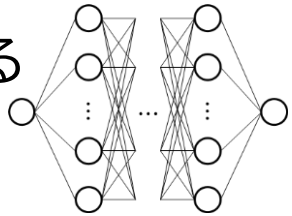
カメラ画像＆
3次元点群データの
キャリブレーション



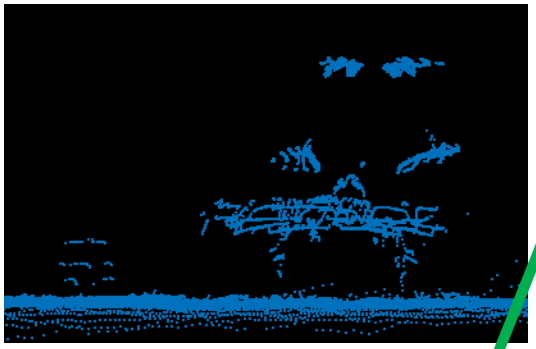
カメラ画像データ

取得

AIによる
人検出



学習
&
検証



3次元点群データ

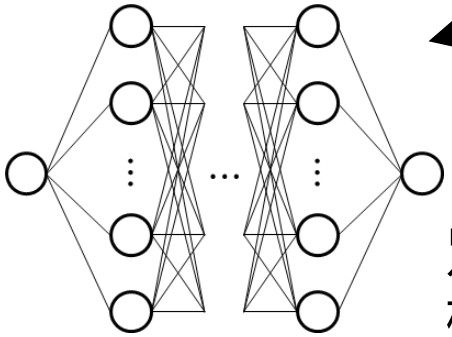
距離検出

画像認識AIによる人検出の疑問点

周囲の作業者： 様々な服装、様々な姿勢

ある畑で学習

人検出AI



別の畑で
検出できる？

- 既往のAIで十分？
（学習は必要？）
- 学習時に必要なことは何？
（姿勢？服の色？）

様々な服で立位・中腰・しゃがみ姿勢を学習させたAI
既往のAI
を比較

学習用＆検証用静止画群の取得方法

■ 学習用静止画群（農機研畑a、被験者1名、服6色）

赤 青 緑 黄 黒 白

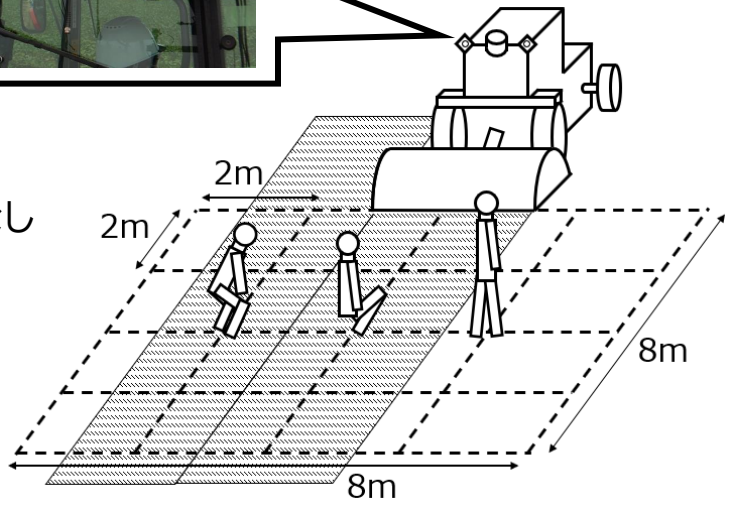
6個のAI作成
AI（赤）～AI（白）
学習に服の色の
影響があるか検証

■ 検証用静止画群（農機研畑b、被験者6名）

A B C D E F

学習用静止画群の取得

ロータリ周辺で
左右のカメラ
による動画取得



平たん
作物なし
畑a

被験者
男性1名
(身長:171cm)

服:6色
赤、青、緑
黄、黒、白

動画から各格子点での静止画群を抽出
服の色:赤青緑黄黒白で分割(各348枚)

ex. 服の色が赤

ex. 服の色が黒

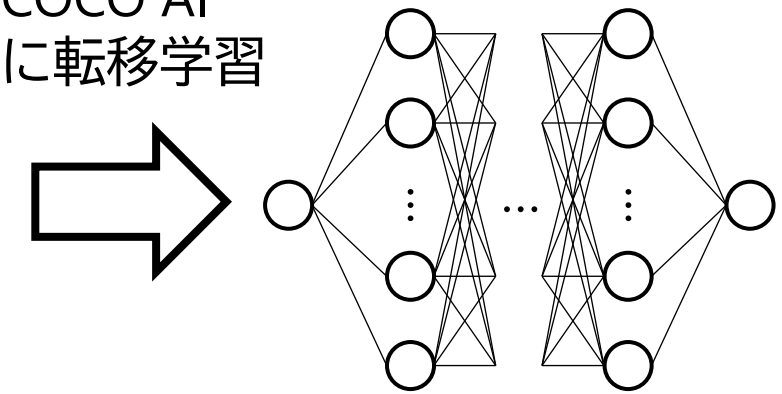
学習用静止画群の変換と転移学習方法

■ 人に対して長方形座標付記(Annotation)

■ 色相、彩度、明度、拡大、左右反転のランダム変換(Augmentation)

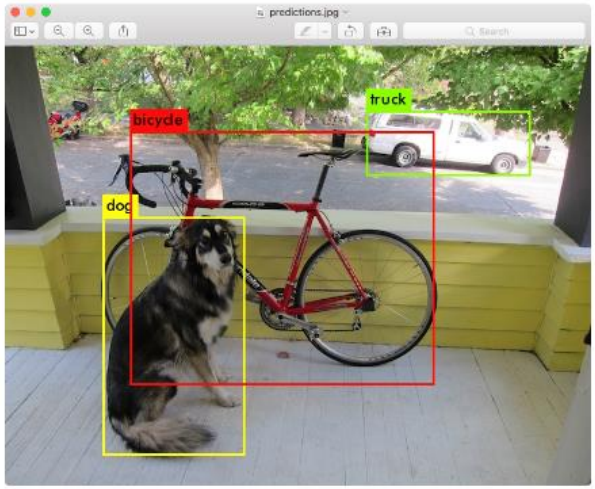
- 各服の学習用静止画群で6個のCOCO AIに対し転移学習
（以下、3姿勢AI）
- 学習回数は100エポック
（348枚を1セットとして反復100回学習）
- 転移学習には
MATLAB® (MathWorks®)
を使用

COCO AI
に転移学習



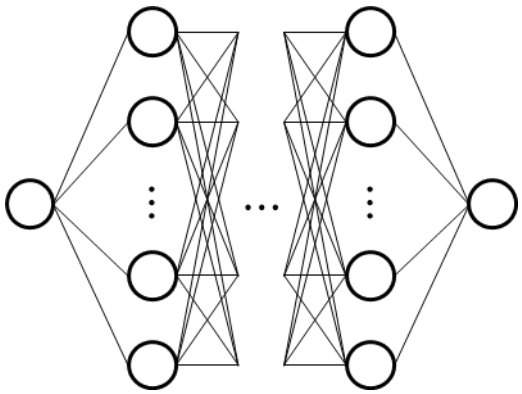
COCO AIについて

物体検出方法



YOLOv3

(https://pjreddie.com/darknet/yolo/より引用)



ネットワーク構造

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53

(https://arxiv.org/abs/1804.02767
より引用)

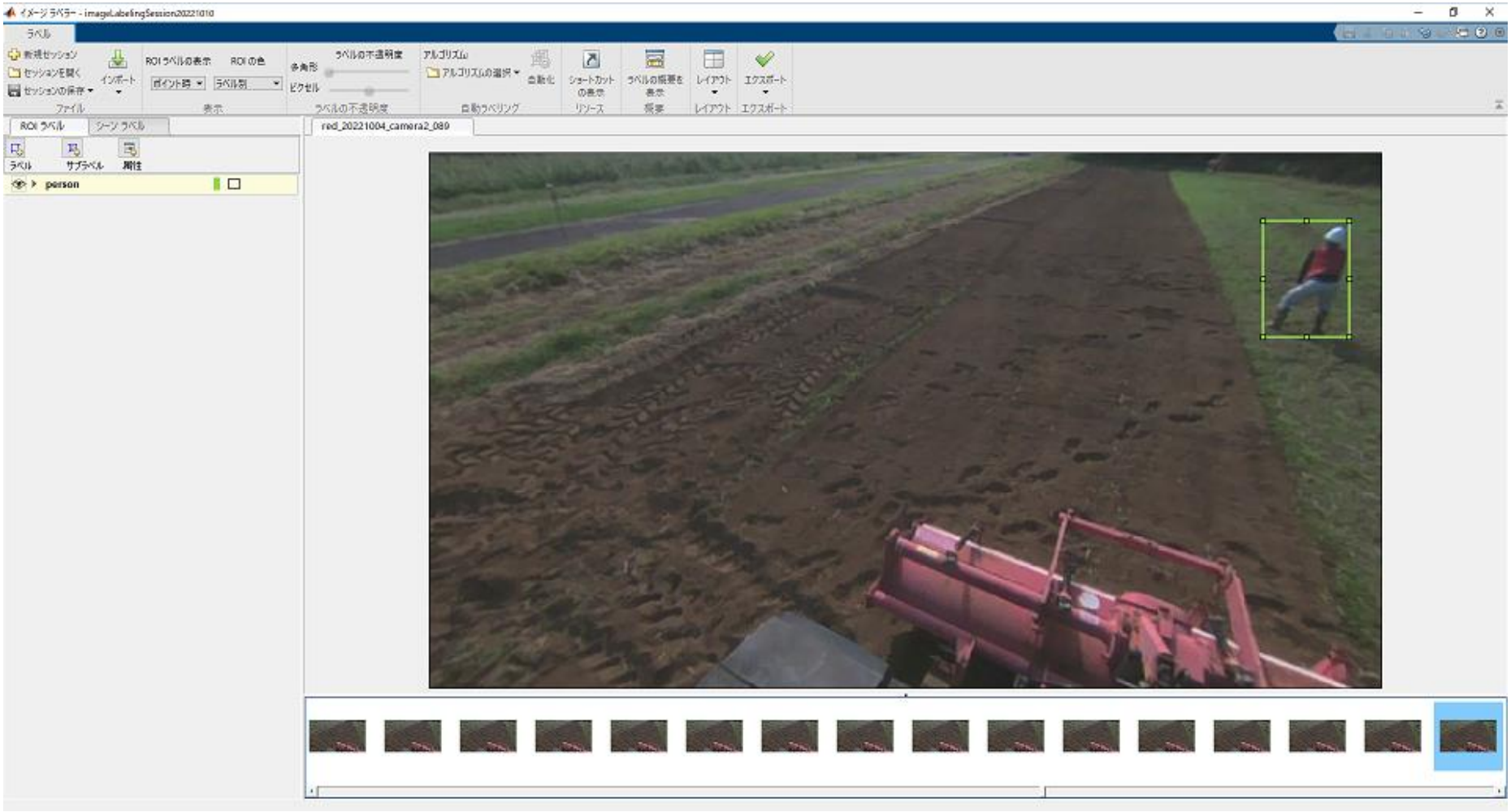
データセット

Common Objects in Context
(オープンデータセット)
約25万人の画像を含む

(https://cocodataset.org/#exploreより引用)

- 転移学習の基となるAIとして使用
- 既往のAI

Annotation



参考URL：
イメージラベラー
https://jp.mathworks.com/help/vision/ref/imagelabeler-app.html

転移学習プログラムの構築(一部抜粋)

農研機構 NARO

設定

```
augmentedTrainingData = transform(trainingData, @augmentData);
preprocessedtrainingData = transform(augmentedTrainingData, @(data)preprocess(yolov3Detector, data));

numepoch = 100;           % epoch数
miniBatchSize = 12;        % ミニバッチサイズ
numiteration = numepoch * ceil(height(trainingdataT)/miniBatchSize);  %iteration数
learningRate = 0.001;      % 学習率
warmupPeriod = round(numiteration/2);     % ウォームアップ回数
l2Regularization = 0.0005;  % L2正則化係数
penaltyThreshold = 0.5;     % ペナルティ閾値
velocity = [];             % 勾配速度

mbqTrain = minibatchqueue(preprocessedtrainingData, 2,...
        "MiniBatchSize", miniBatchSize,...
        "MiniBatchFcn", @(images, boxes, labels) createBatchData(images, boxes, labels, classNames), ...
        "MiniBatchFormat", ["SSCB", ""],...
        "DispatchInBackground", true,...
        "OutputCast", ["", "double"]);
```

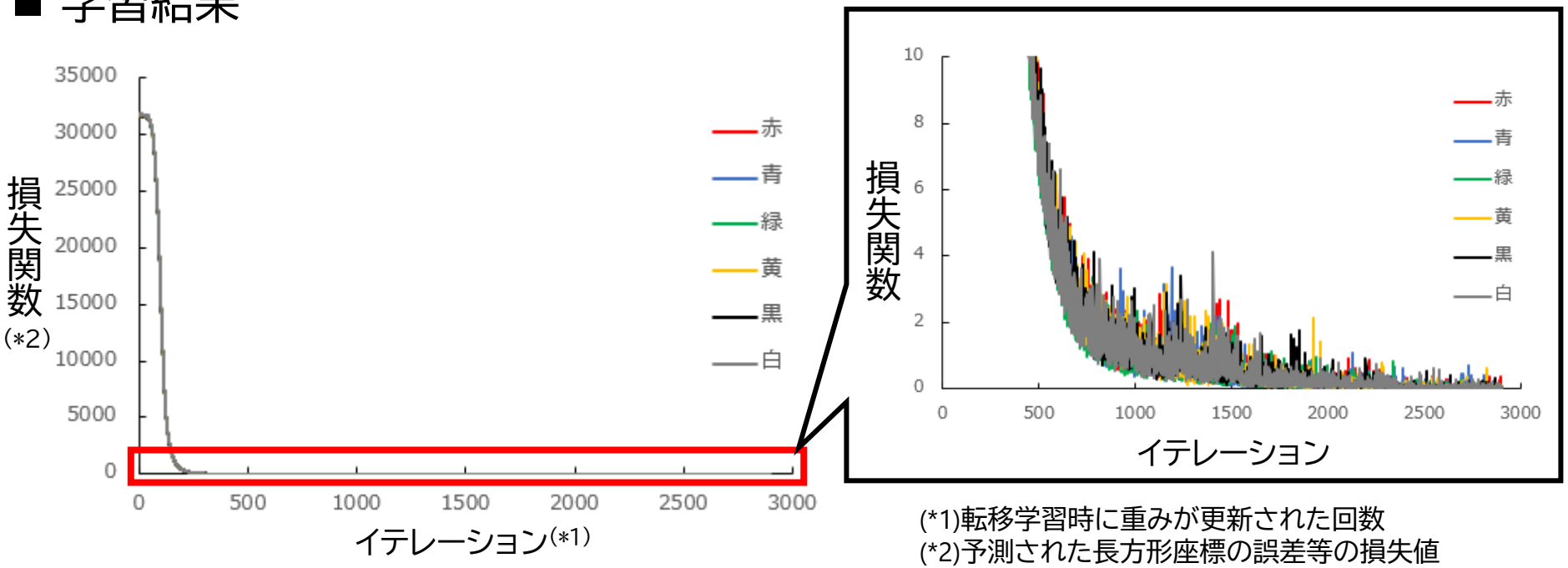
学習

```
iteration = 0;

trainresults = struct('epoch_v', zeros(numiteration, 1), ...
    'iteration_v', zeros(numiteration, 1), ...
    'currentLR_v', zeros(numiteration, 1), ...
    'TotalLoss_v', zeros(numiteration, 1), ...
    'BoxLoss_v', zeros(numiteration, 1), ...
    'ObjectLoss_v', zeros(numiteration, 1), ...
    'ClassLoss_v', zeros(numiteration, 1));

for epoch = 1:numepoch
    reset(mbqTrain);
    shuffle(mbqTrain);

    while(hasdata(mbqTrain))
        iteration = iteration + 1;

        [XTrain, YTrain] = next(mbqTrain);
        [gradients, state, lossInfo] = dlfeval(@modelGradients, yolov3Detector, XTrain, YTrain, penaltyThreshold);
        gradients = dlupdate(@(g,w) g + l2Regularization*w, gradients, yolov3Detector.Learnables);
        currentLR = piecewiseLearningRateWithWarmup(iteration, epoch, learningRate, warmupPeriod, numepoch);
        [yolov3Detector.Learnables, velocity] = sgdmupdate(yolov3Detector.Learnables, gradients, velocity, currentLR);
        yolov3Detector.State = state;

        trainresults.epoch_v(iteration) = epoch;
        trainresults.iteration_v(iteration) = iteration;
        trainresults.currentLR_v(iteration) = currentLR;
        trainresults.TotalLoss_v(iteration) = double(gather(extractdata(lossInfo.totalLoss)));
        trainresults.BoxLoss_v(iteration) = double(gather(extractdata(lossInfo.boxLoss)));
        trainresults.ObjectLoss_v(iteration) = double(gather(extractdata(lossInfo.objLoss)));
        trainresults.ClassLoss_v(iteration) = double(gather(extractdata(lossInfo.clsLoss)));
    end
end
```

参考URL:
YOLO v3 深層学習を使用したオブジェクトの検出
https://jp.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html

転移学習結果

■ 学習結果



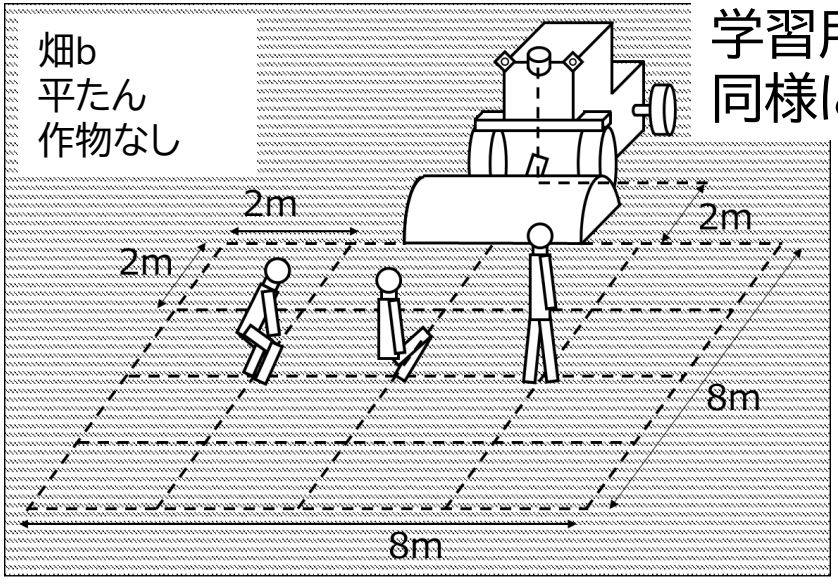
(*1)転移学習時に重みが更新された回数
(*2)予測された長方形座標の誤差等の損失値
引用：https://arxiv.org/pdf/1804.02767.pdf

■ 最終学習結果

	3姿勢AI(赤)	3姿勢AI(青)	3姿勢AI(緑)	3姿勢AI(黄)	3姿勢AI(黒)	3姿勢AI(白)
損失関数	0.041	0.056	0.020	0.056	0.026	0.024

損失関数が十分に小さな値となった(収束した)ため、
全てのAIで十分な学習が行えたと判断

検証用静止画群



学習用静止画群と
同様に動画取得

被験者6名　身長: 166～182cm

被験者	服	被り物
A	上下:黒	黒帽子
B	上下:青	無
C	上下:灰	麦わら帽子
D	上下:赤	橙帽子
E	上:青 下:薄緑	橙ヘルメット
F	上下:薄緑	無

各格子点での静止画群を抽出
被験者:A~Fで分割(各348枚)
姿勢:立位、中腰、しゃがみで
分割(各696枚)

ex. 被験者C

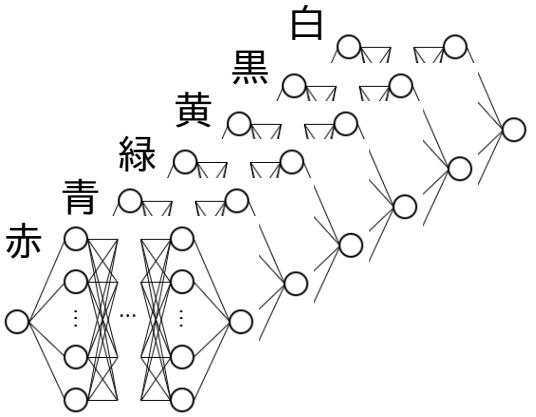
ex. しゃがみ姿勢

AIの検証方法

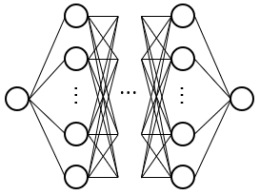
■ 人に対して長方形座標付記
（Annotation）

7個の
AIを検証

転移学習した
6個の3姿勢AI



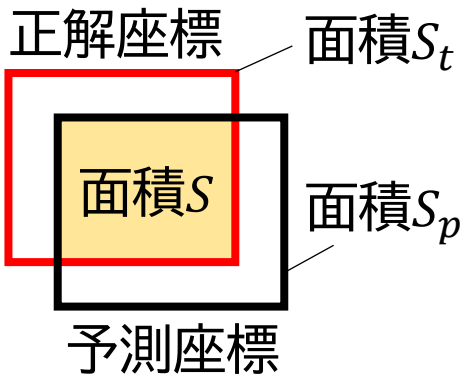
既往のAI
(COCO AI)



■ 検出精度評価方法

		検出器	
		検出	不検出
真実	存在	真陽性	偽陰性
	存在せず	偽陽性	真陰性

安全性を考慮すると
実際に存在した人の数に対し
真陽性の割合が高い検出器
が人検出器として望ましい



再現率を採用 $\frac{\text{AIが検出した人数}}{\text{検証時の画像に存在した全人数}} = \frac{\text{真陽性}}{\text{真陽性+偽陰性}}$

$$\mathrm{IOU} = \frac{S}{S_t + S_P - S}$$

IOUが0.3以上でAIが検出したと判断

検出の様子（対象例：被験者C）

■ 3姿勢AI（赤）

■ COCO AI

検出評価結果

■ 各被験者での検証

再現率(%)		検証静止画群(被験者)					
		A	B	C	D	E	F
3姿勢AI	赤[a]	99.7	99.1	97.4	99.7	100.0	100.0
	青[a]	99.4	98.6	98.3	98.0	100.0	99.4
	緑[a]	100.0	98.9	97.7	98.9	100.0	100.0
	黄[a]	100.0	99.1	99.4	98.9	100.0	99.7
	黒[a]	98.3	98.9	98.9	98.9	99.4	99.4
	白[a]	97.1	99.4	98.9	99.4	100.0	100.0
COCO AI[b]		63.2	59.5	42.5	53.4	87.4	69.0

注) 異なる小文字間では有意差5%あり(Steel-Dwass法)

- 3姿勢AI間では有意差なし
- 3姿勢AIとCOCO AIでは5%で有意差あり

⇩

- 服の色での転移学習による差はない
- 3姿勢AIはCOCO AIよりも精度が高い

■ 各姿勢での検証

再現率(%)	検証用静止画群(姿勢)		
	立位	中腰	しゃがみ
3姿勢AI	**99.4**	**100.0**	**98.1**
COCO AI	79.6	68.8	39.1

注) 3姿勢AIは、6個のAIの平均値

- 3姿勢AIはCOCO AIと比較して3姿勢ともに精度が高い(特にしゃがみ姿勢)

⇩

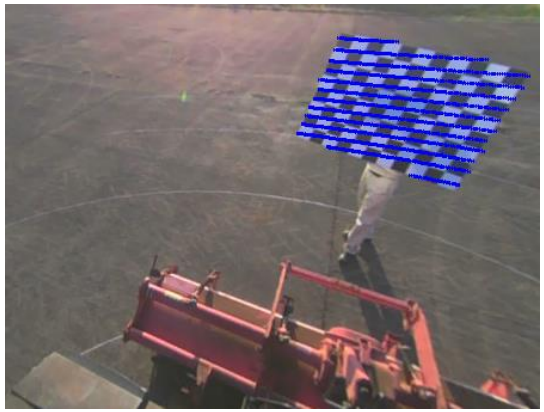
- Common Objects in Context内に俯瞰した3姿勢静止画群が少ないためCOCO AIの検出精度が落ちた可能性あり
- 3姿勢の転移学習により既往のAIよりも3姿勢の精度が高いAIが生成可能

カメラと3DLiDARによる距離検出の検証

カメラと3DLiDARによる距離検出

取得

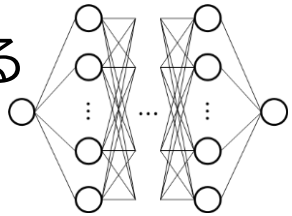
カメラ画像＆
3次元点群データの
キャリブレーション



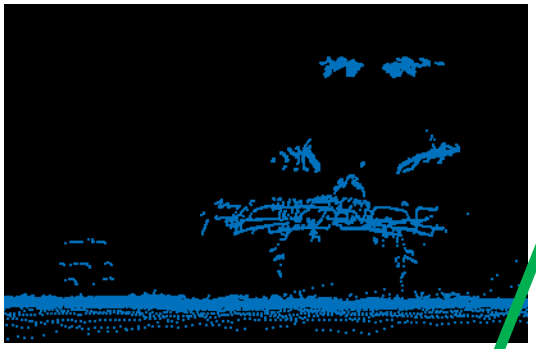
取得

カメラ画像データ

AIによる
人検出



学習
&
検証



3次元点群データ

距離検出

カメラと3DLiDARのキャリブレーション

チェッカーボード
正方形1辺：105mm
1,155mm×840mm

- 各カメラの歪み補正
- 各カメラと3DLiDARとの補正

参考URL：
LiDAR とカメラのキャリブレーション
https://jp.mathworks.com/help/lidar/ug/lidar-and-camera-calibration.html

カメラと3DLiDARのキャリブレーション

■ 各カメラの歪み補正

パラメータ取得
- 焦点距離
- 主点
- 半径方向歪み

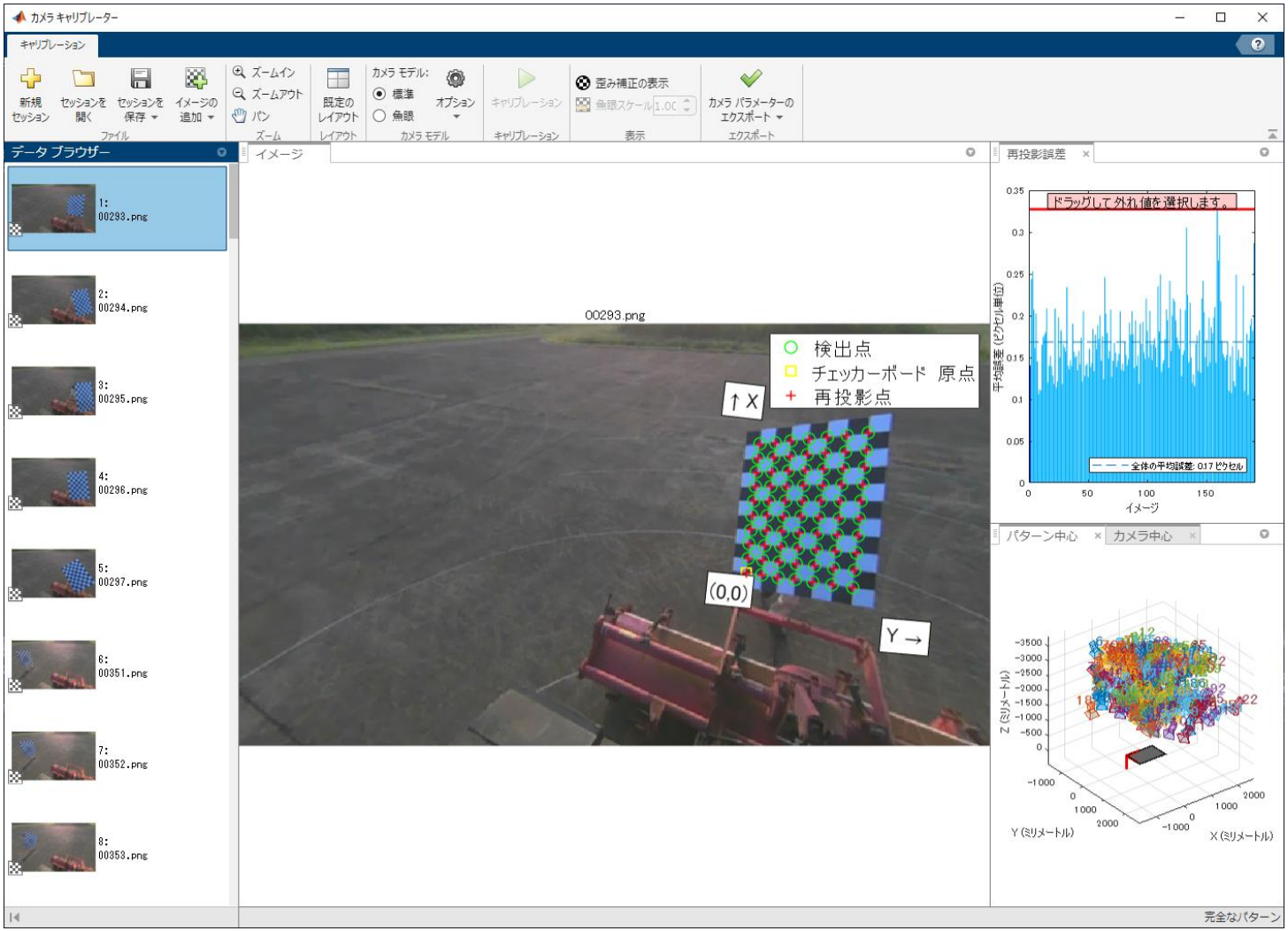
参考URL：
カメラのキャリブレーションとは
https://jp.mathworks.com/help/vision/ug/camera-calibration.html

■ 各カメラと3DLiDARとの補正

パラメータ取得
- 3次元点群の変換行列

参考URL：
LiDAR カメラ キャリブレーションとは
https://jp.mathworks.com/help/lidar/ug/lidar-camera-calibration.html

カメラキャリブレーター



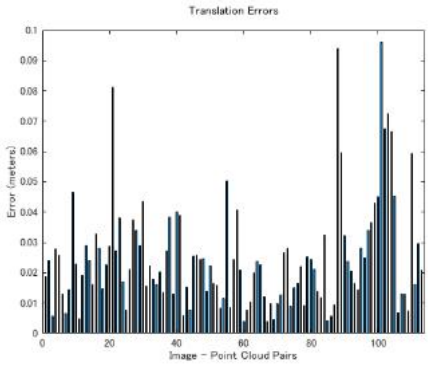
参考URL：
カメラキャリブレーター
https://jp.mathworks.com/help/vision/ref/cameracalibrator-app.html

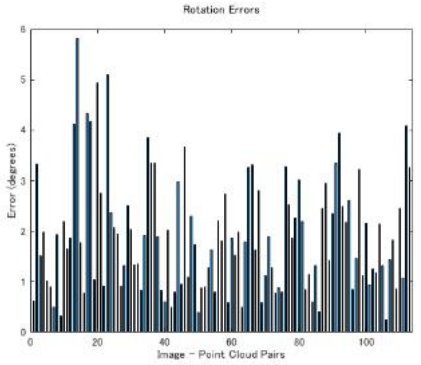
カメラと3DLiDARとの補正プログラム構築（一部抜粋）

```
%% Estimate 3D checkerboard points from images
minCornerMetric = 0.150000;
[imageCorners3d, planeDimension, dataUsed] = estimateCheckerboardCorners3d(imageFilePaths, ...
    intrinsics, squareSize, 'Padding', padding, 'MinCornerMetric', minCornerMetric);

% Filter images and point clouds that are not used
imageFilePaths2 = imageFilePaths(dataUsed);
ptcFilePaths2 = ptcFilePaths(dataUsed);
```

```
%% Detect plane segment from point clouds
minDistance = 0.500000;
roi = [-10 0 -10 10 -5 5];
dimTol = 0.050000;
removeGround = true;
rng('default');
detectionResults = struct();
k = 1;

for i = 1:numel(ptcFilePaths2)
    [detectionResults(i).lidarCheckerboardPlane, detectionResults(i).ptCloudUsed] = detectRectangularPlanePoints(ptcFilePaths2{i}, ...
        planeDimension, 'RemoveGround', removeGround, 'ROI', roi, 'DimensionTolerance', dimTol, 'MinDistance', minDistance);
    if detectionResults(i).ptCloudUsed
        lidarCheckerboardPlanes(k,1) = detectionResults(i).lidarCheckerboardPlane;
        k = k + 1;
    end
end
```

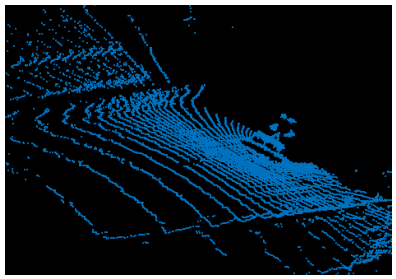
```
% Filter images and point clouds that are not used
imageFilePaths3 = imageFilePaths2([detectionResults.ptCloudUsed]);
% Filter image corners that are not used
imageCorners3d2 = imageCorners3d(:, :, [detectionResults.ptCloudUsed]);
% Adjust dataUsed
rmInd = 0;

if ~any(rmInd) || isempty(rmInd)
    imageFilePaths4 = imageFilePaths3;
    imageCorners3d3 = imageCorners3d2;
    lidarCheckerboardPlanes2 = lidarCheckerboardPlanes;
else
    imageFilePaths4 = removerow1(imageFilePaths3,rmInd);
    imageCorners3d3 = removerow2(imageCorners3d2,rmInd);
    lidarCheckerboardPlanes2 = removerow3(lidarCheckerboardPlanes,rmInd);
end
```



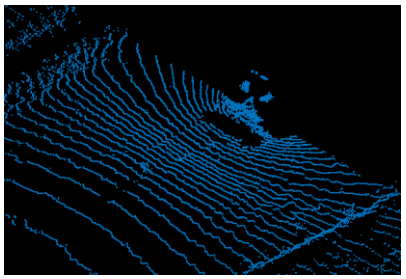


参考URL：
estimateLidarCameraTransform
https://jp.mathworks.com/help/lidar/ref/estimatelidarcameratransform.html

距離取得方法詳細

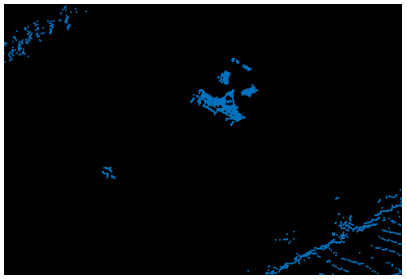


3次元点群データ

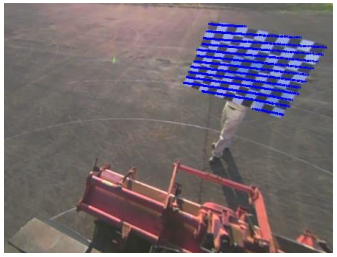


3次元変換

参考URL：pcfitplane
https://jp.mathworks.com/help/vision/ref/pcfitplane.html



地面点群推定・除去

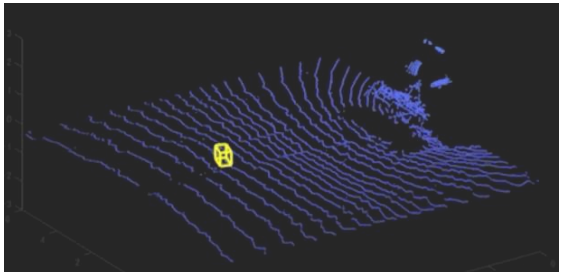


キャリブレーション
パラメータ

画像データ

AIによる検出

画像と3次元点群の合致



該当部分の直方体推定

作業者－3DLiDAR間
水平距離検出

参考URL：Detect Vehicles in Lidar Using Image Labels
https://jp.mathworks.com/help/lidar/ug/detect-vehicles-in-lidar-using-image-labels.html

距離取得の流れ

左カメラ
画像データ
取得

3DLiDAR
3次元点群
データ取得

右カメラ
画像データ
取得

AIによる
人検出

3次元変換及び
地面点群除去

AIによる
人検出

左CC、LCC
パラメータ

右CC、LCC
パラメータ

L_L=8.19m

距離L_L取得

距離L_R取得

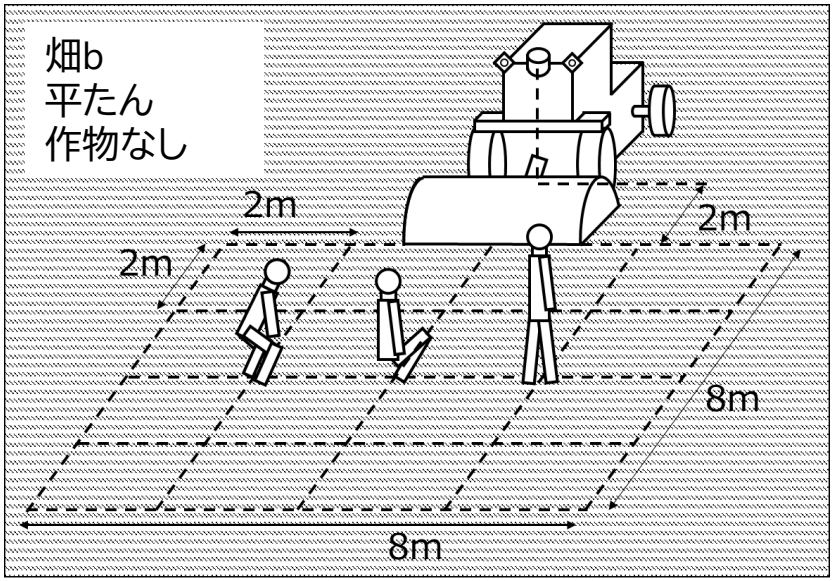
L_R=8.20m

L_LとL_Rを比較
短い方を選択

L=8.19m

CC：カメラキャリブレーション
LCC：LiDARカメラキャリブレーション

距離検証試験（ほ場b、被験者6名）

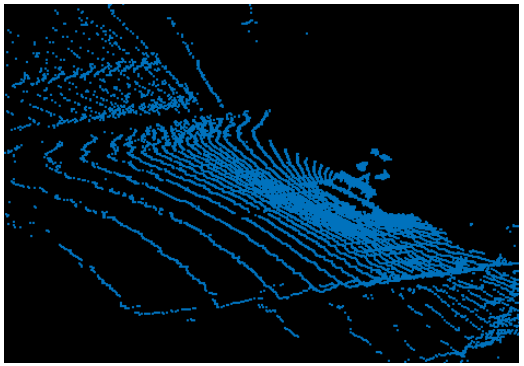


検証用静止画群の取得時に
静止画群と同時に3次元点群を取得
被験者A~Fに対して距離が取得可能か検証

静止画例

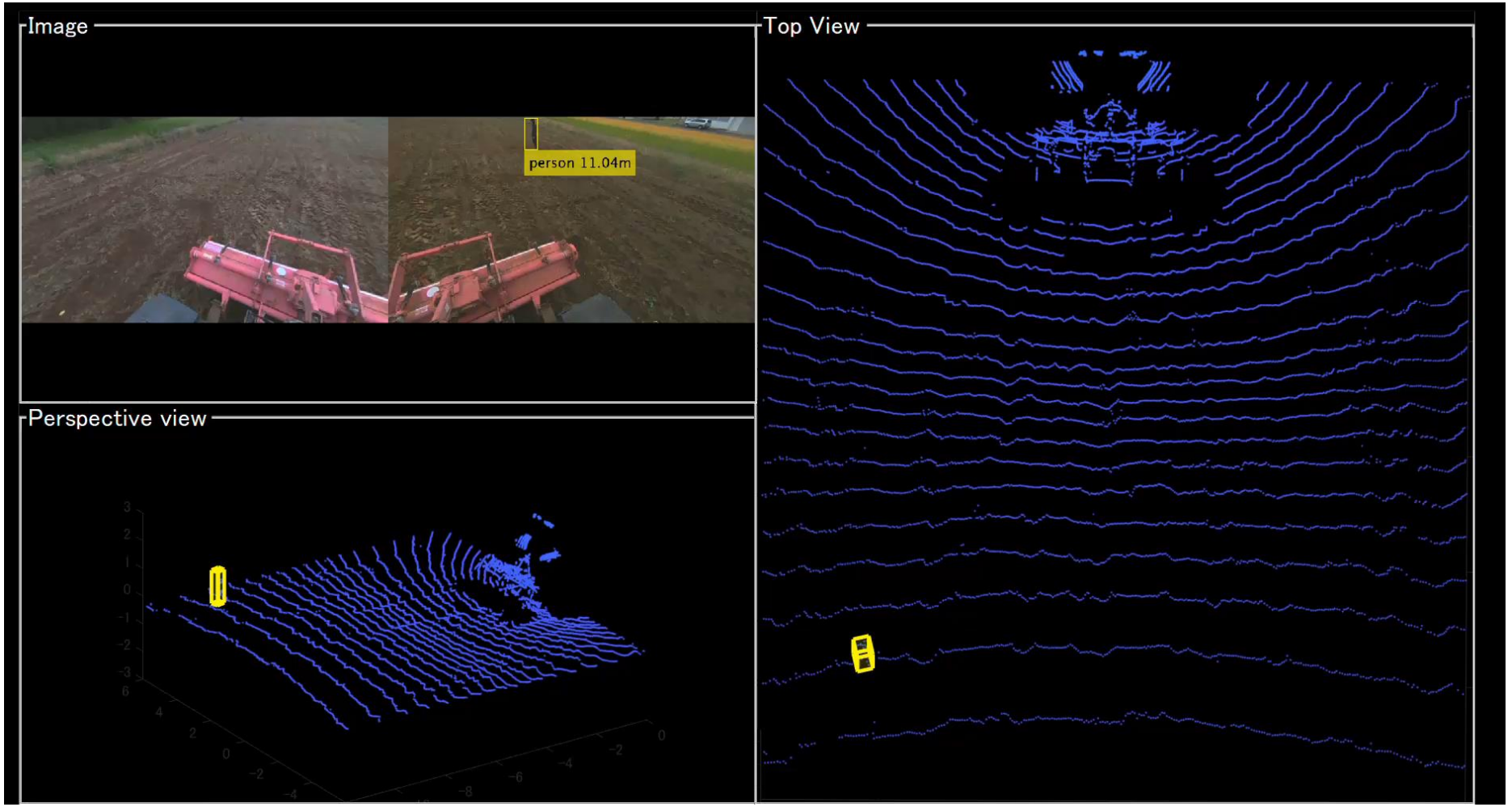
被験者6名　身長: 166～182cm

被験者	服	被り物
A	上下:黒	黒帽子
B	上下:青	無
C	上下:灰	麦わら帽子
D	上下:赤	橙帽子
E	上:青 下:薄緑	橙ヘルメット
F	上下:薄緑	無



3次元点群例

距離検出の様子（対象例：被験者A）



参考URL：
Detect Vehicles in Lidar Using Image Labels
https://jp.mathworks.com/help/lidar/ug/detect-vehicles-in-lidar-using-image-labels.html

距離検出結果

■ 距離精度評価方法

二乗平均平方根誤差率

$$\text{距離精度} = 1 - \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(\frac{\widehat{L_k} - L_k}{L_k}\right)^2}$$

n：検出数、$\hat{L}$：距離推定値、L：距離真値

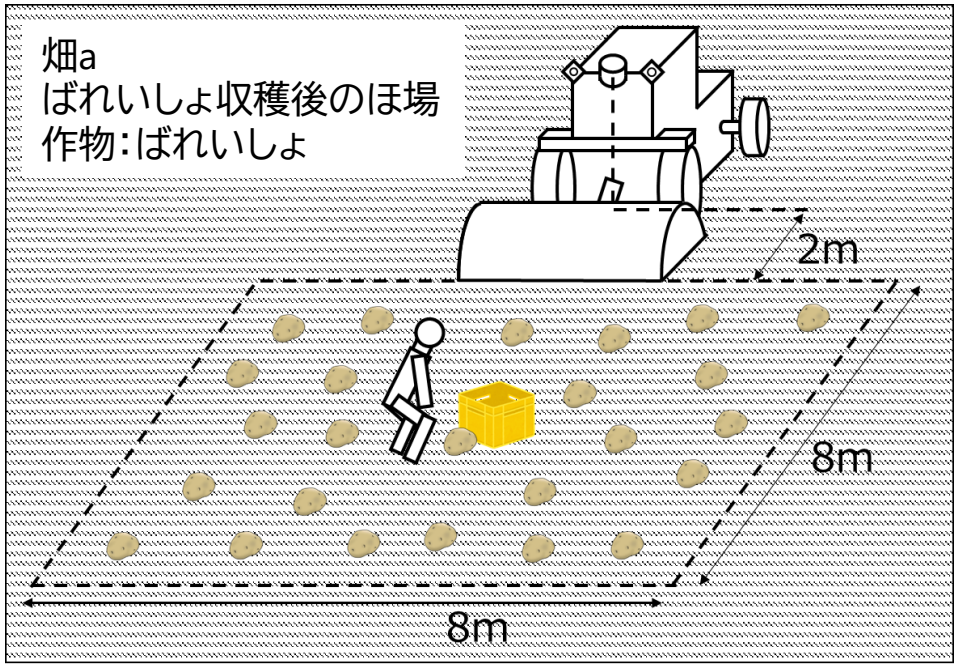
■ 結果

距離精度(%)		検証用静止画群と3次元点群					
		A	B	C	D	E	F
3姿勢AI	赤	96.3	96.8	96.0	95.8	96.6	96.5
	青	96.2	96.6	95.9	95.9	96.4	96.3
	緑	96.2	96.5	96.1	95.8	96.3	96.0
	黄	96.1	96.6	95.8	95.5	96.2	96.2
	黒	96.3	96.7	95.7	94.5	96.7	96.3
	白	94.3	95.2	95.8	95.5	96.1	94.7
平均		95.9	96.4	95.9	95.5	96.4	96.0
標準偏差		0.769	0.621	0.128	0.513	0.237	0.671

- 3姿勢AI間では有意差なし (Steel-Dwass法)
- 距離精度は約96%となり安定して高い精度となった。

作物拾い上げ時の検証

作物拾い上げ検証試験（畑a、被験者12名）



ばれいしょ拾い上げ試験を実施
静止画と同時に3次元点群を取得
（1Hz、120セット／1被験者）

被験者A~Lに対して検出可能か検証
3姿勢AI(赤)とCOCO AIで比較

$$検出率＝\frac{距離を検出した人数}{検証時の画像に存在した全人数}$$

で評価

被験者12名　身長: 150～178cm

被験者	頭	上	下	コンテナ
A	黒	黒	黒	黄
B	白	青	青	黄
C	黒	薄橙	薄橙	青
D	黒	紺	紺	青
E	黒	緑	緑	橙
F	薄橙	薄青	薄青	黄
G	薄橙	黒	黒	青
H	黒	濃緑	濃緑	橙
I	白	白	白	青
J	黒	濃緑	濃緑	黄
K	黒	薄青	薄青	青
L	黒	紺	薄青	橙

検出結果(検出例:被験者I)

3姿勢AI(赤):検出率99%
COCO AI:検出率82%

両AIともコンテナ、影等による誤検出はなし
考察:3姿勢を学習することが重要

まとめ

■ 本研究のまとめ

- ロータリ装着のトラクタ後部に接近する人を検出するためのシステムを構築した。
- 農作業環境にてトラクタ後部に接近する人を学習及び検証するための静止画群取得方法を考案した。
- 立位、中腰、しゃがみ姿勢を転移学習することにより、既存のAIと比較して画像認識AIの検出精度が安定して高くなった。
- 広角カメラと3DLiDARによる人と機体との距離を取得するためのプログラムを構築した。距離検証の結果、距離精度は安定して高い結果となった。

■ 残された課題

- 本研究は平たんな作物なしの畑のみの転移学習、検証であったため、他に想定される場面での検討が必要である。
- 本研究はオフラインでの検証にとどまっている。
本研究を活用して安全装置を開発する場合、人検出から装置が作動するまでの時間を考慮する必要がある。

MATLAB®の利点

■ 関数が豊富

- 各種Toolboxの導入により
  数多くの関数を使用可能
  → プログラミングが簡略化可能
- 関数の中身を確認可能
  → 独自の関数を生成可能

■ ヘルプセンターが充実

- 関数、カテゴリごとの
  ドキュメンテーション参照可能
- 例が豊富、即座に確認可能
  → 理解を早めることが可能
  解決の時間短縮

例：YOLO v3 深層学習を使用したオブジェクトの検出
https://jp.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html

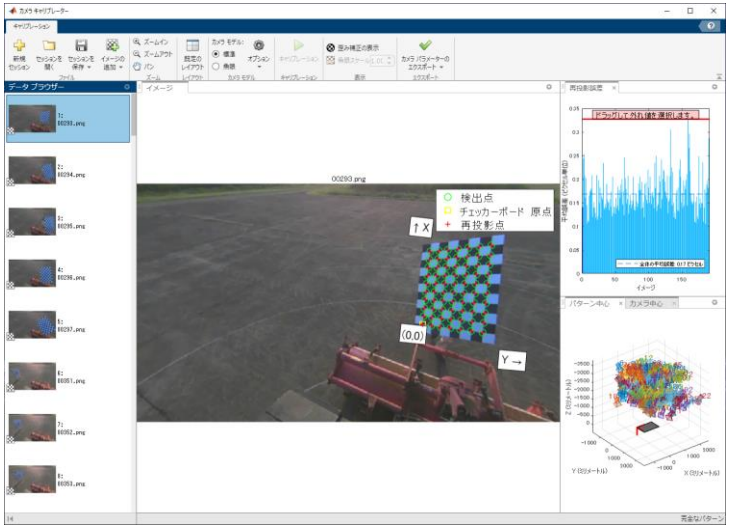
例をすぐにスクリプトで
見ることが可能

MATLAB®の利点

■ 充実したGUI

- カメラキャリブレーター、
  Lidarビューワー、
  LiDAR カメラ キャリブレーター、
  イメージラベラー等
  直感的に操作しやすいGUIが充実

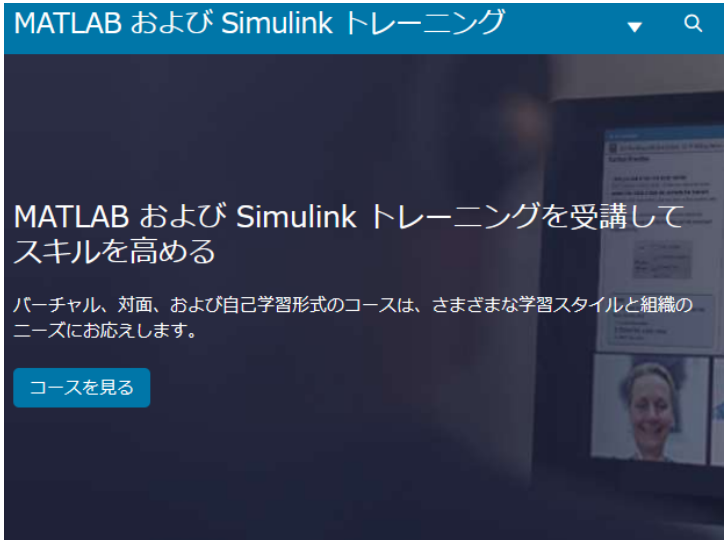
例:カメラキャリブレーター
https://jp.mathworks.com/help/vision/ref/cameracalibrator-app.html



■ MATLAB および Simulink トレーニング
によるスキル向上

- 受講者のニーズに合わせた
  トレーニングコースが存在
- クラスルーム形式、対面形式、
  自己学習形式(オンラインコース)

  → 理解を早めることが可能



https://jp.mathworks.com/learn/training.html?s_tid=gn_trg_ov