

物体検知を応用した農作業における 間引き対象発見自動化機構への展開

嘉村勇輝

目次

- 1・提案
- 2・システム構成図
- 3・本葉がある芽の検知について
- 4・葉面積算出について
- 5・間引かない芽(残す芽)の発見について
- 6・まとめ

1・提案

2・システム構成図

3・本葉がある芽の検知について

4・葉面積算出について

5・間引かない芽(残す芽)の発見について

6・まとめ

提案

農作業の1つである間引きは
1つの作物を育てる過程で2,3回程行う必要がある。



間引きの作業が自動化できれば
効率化が図れる

間引き対象発見システムを実装

ロボットに搭載し自動化,若しくは
農業素人でも簡単に間引きが行えるようにする

1・提案

2・システム構成図

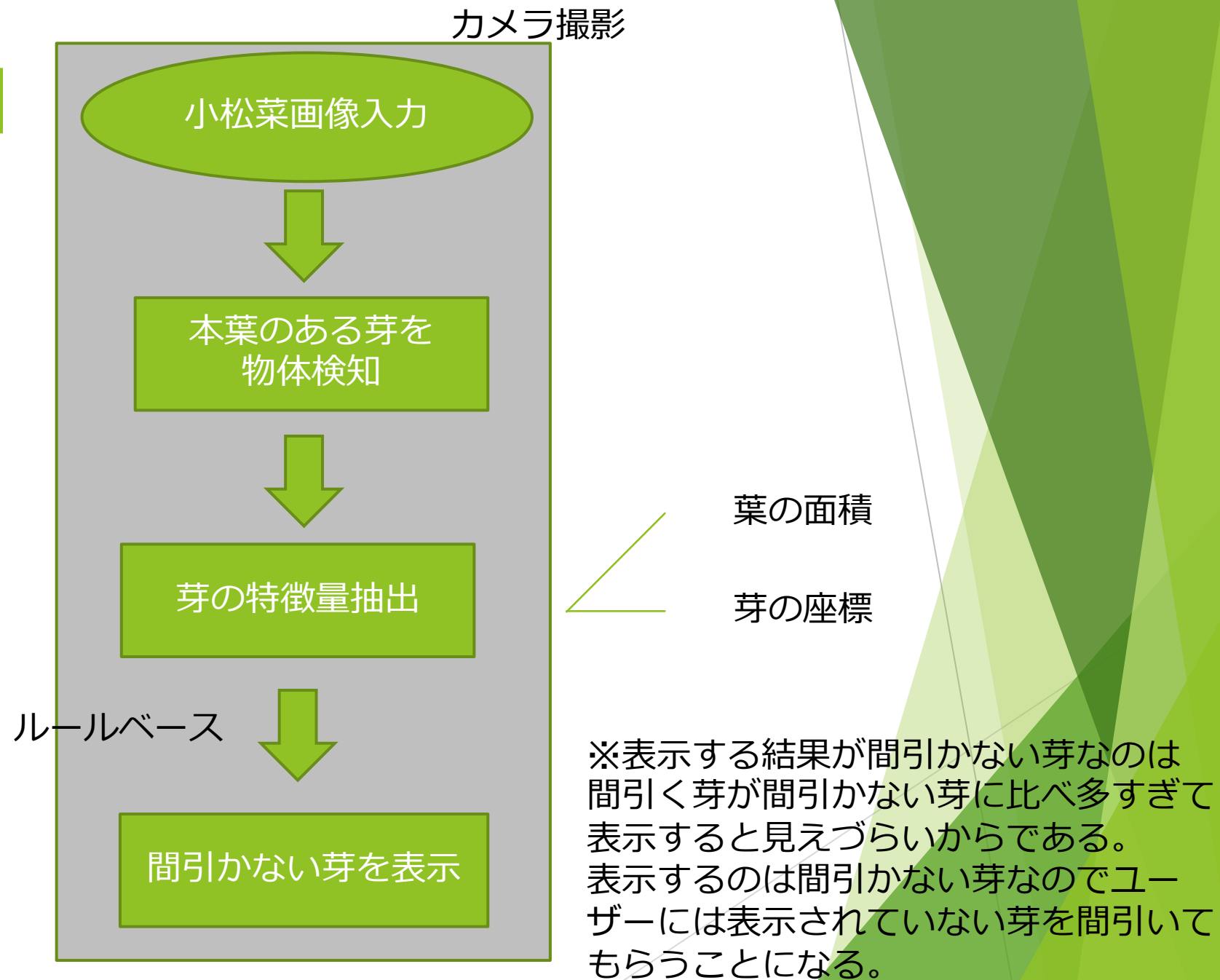
3・本葉がある芽の検知について

4・葉面積算出について

5・間引かない芽(残す芽)の発見について

6・まとめ

システム構成図



1・提案

2・システム構成図

3・本葉がある芽の検知について

4・葉面積算出について

5・間引かない芽(残す芽)の発見について

6・まとめ

子葉と本葉の物体検知

訓練データ例



本葉

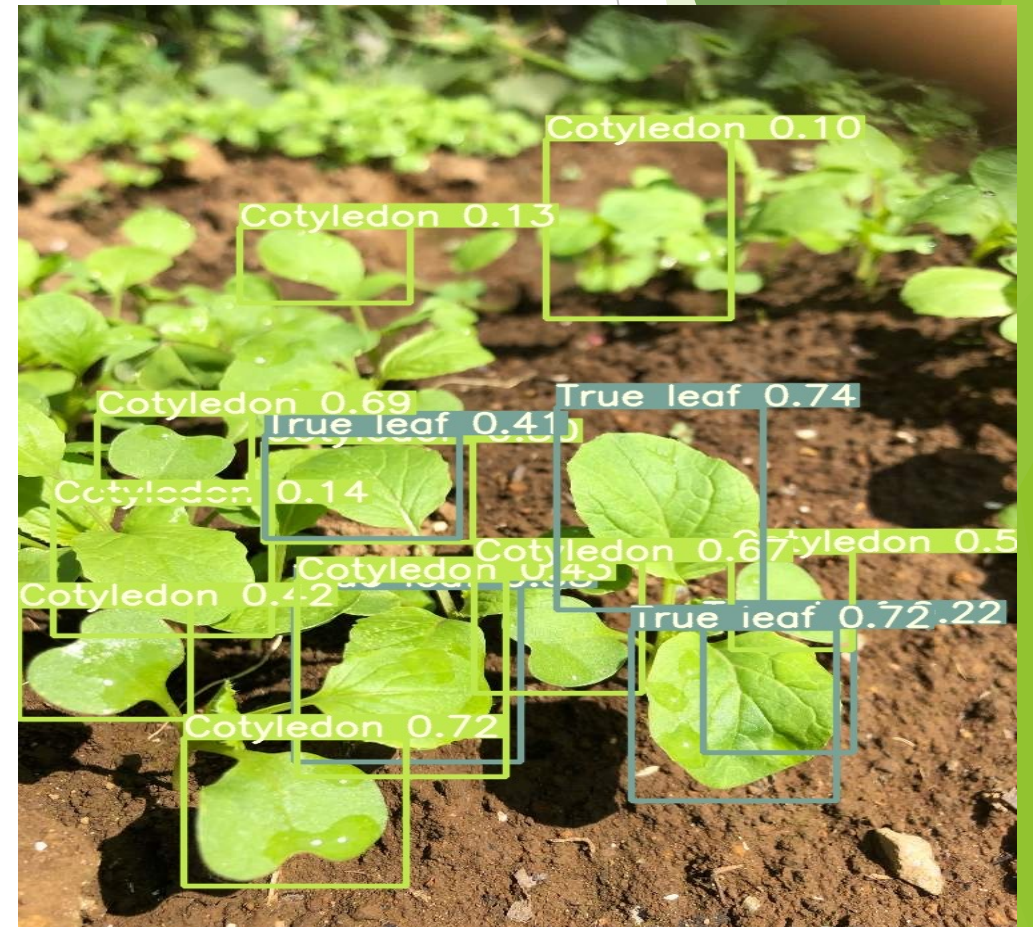


子葉

一枚の画像に対して一枚の子葉・本葉と
わかりやすいように訓練データを作成

精度：71%

再現度：100%



本葉ありの芽・物体検知~part1~

Yolov5が子葉と本葉を認識し検知できることを把握したので次は本葉がある芽を検知する方針に決定した。

精度:85%

再現率:100%

訓練データ例



子葉と本葉の検知時と同じやり方で学習、精度・再現率は高いものの思ったように検知していないことが判明



うまく検知できない原因

訓練データ例



テストデータ例



原因：

- ・ 背景の土や芽の重なりなど学習を行えていない部分がある
- ・ テストデータに実際に使用する画像ではなく
訓練データと同じような検知しやすい画像を利用していたため

本葉ありの芽・物体検知~Part2~

訓練画像



テスト画像



精度 : 38%

再現率: 83%



画像内にある芽を全てアノテーションし学習

- ・ 精度と再現率に信憑性が出てきた
- ・ 学習枚数70枚程だったので学習枚数を増やせば精度は向上するだろう



子葉と本葉の物体検知



本葉あり芽物体検知円①



本葉あり芽物体検知②

1・提案

2・システム構成図

3・本葉がある芽の検知について

4・葉面積算出について

5・間引かない芽(残す芽)の発見について

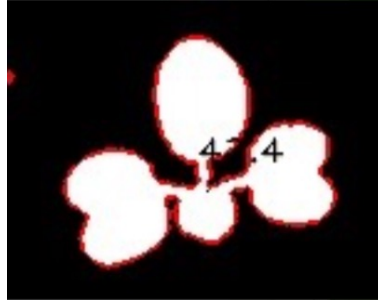
6・まとめ

葉の面積算出～Part1～

Opencv利用

2値化

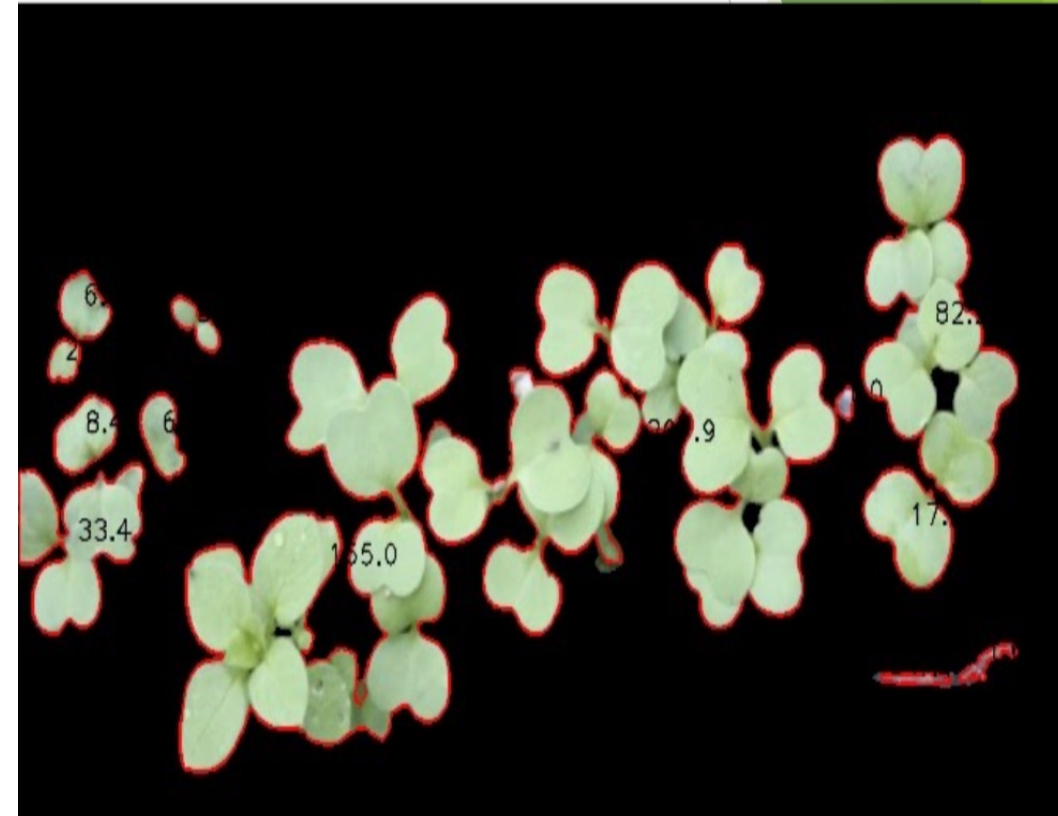
輪郭抽出・面積算出



出力画像

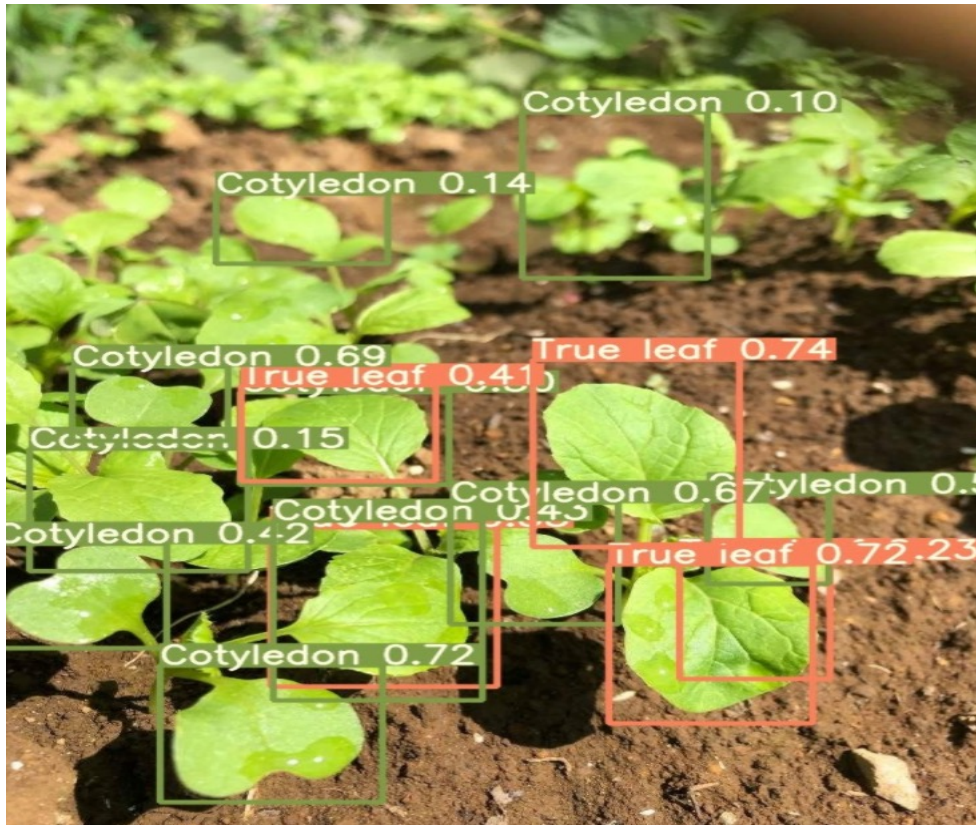
芽ごとに輪郭抽出して欲しいのに区別できず芽が重なった状態で面積を算出している

入力画像



葉の面積算出~Part2~

輪郭抽出し面積を算出する方法は断念し
物体検知時に出力されるバンディングBOXのサイズを
葉の面積の基準にすることにした



座標	面積
(0.620667, 0.241)	0.035954602
(0.307333, 0.28)	0.014104
(0.144, 0.6085)	0.031269381
(0.760667, 0.7365)	0.020942713
(0.35, 0.525)	0.022607963999999998
(0.389333, 0.722)	0.042160062
(0.344, 0.519)	0.021802704000000003
(0.086, 0.709)	0.020984
(0.383333, 0.726)	0.0439599299999999994
(0.774, 0.6425)	0.0127399649999999999
(0.54, 0.671)	0.023856
(0.155333, 0.4845)	0.013571999999999999999
(0.276667, 0.87)	0.03562672
(0.716667, 0.765)	0.037944
(0.642, 0.5425)	0.045260073

1・提案

2・システム構成図

3・本葉がある芽の検知について

4・葉面積算出について

5・間引かない芽(残す芽)の発見について

6・まとめ

間引かない芽の発見

手法

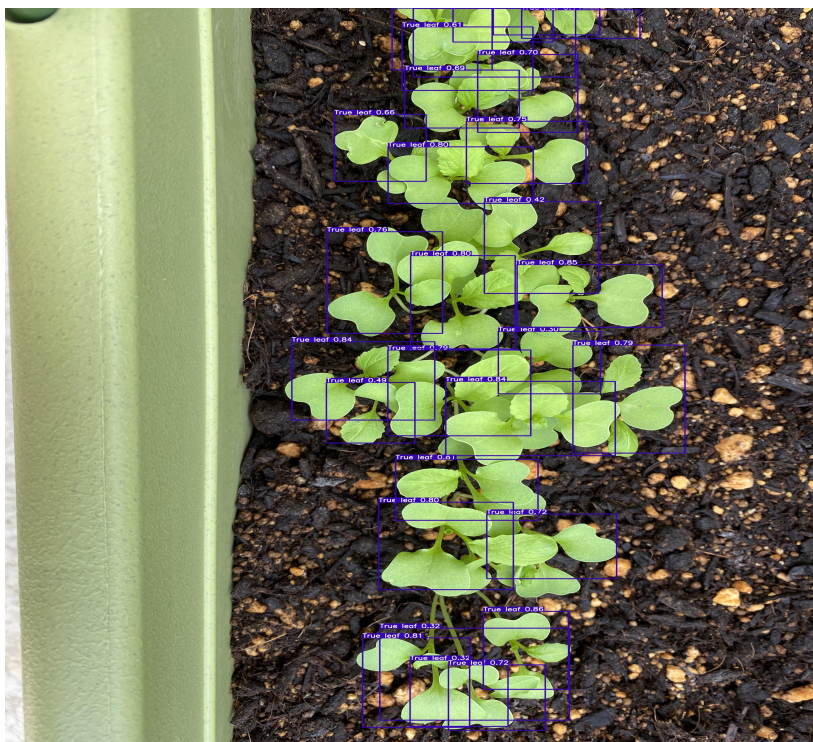
- ① YOLOv5で物体検知、各芽の座標を取得
- ② 芽が密集している範囲を算出(データ加工)
- ③ ②で求めた範囲内でクラスタリング、クラスター内で面積が一番大きい芽を表示

クラスタリングの採用理由

→間引きは間引かない芽(残す芽)を一定の間隔で開けて
密集しないようにする必要があるため、クラスタリングによってグループ分けし密集しないような位置の芽を発見できると考えたから

① YOLOv5で物体検知・ 各芽の座標を取得

YOLOv5を用いて本葉がある芽を物体検知し、
その後その芽の位置を特定する。



本葉のある芽を検知

	X	Y	w	h	Area
0	0.552885	0.219952	0.235577	0.127404	0.030013
1	0.836538	0.651442	0.182692	0.192308	0.035133
2	0.597356	0.453125	0.204327	0.209135	0.042732
3	0.782452	0.455529	0.237981	0.146635	0.034896
4	0.569712	0.643029	0.250000	0.141827	0.035457
5	0.581731	0.090144	0.163462	0.108173	0.017682
6	0.543269	0.903846	0.192308	0.168269	0.032359
7	0.606971	0.814904	0.247596	0.100962	0.024998
8	0.497596	0.478365	0.216346	0.115385	0.024963
9	0.693510	0.194712	0.209135	0.110577	0.023126
10	0.373798	0.611779	0.228365	0.132212	0.030193

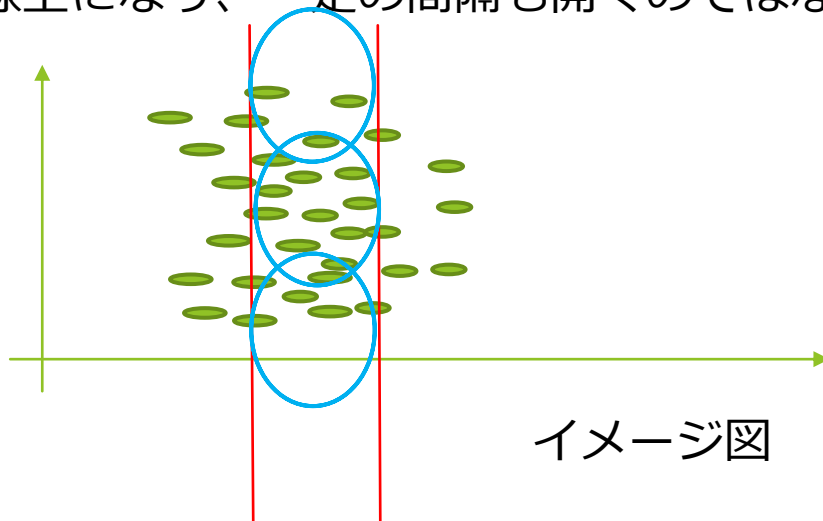
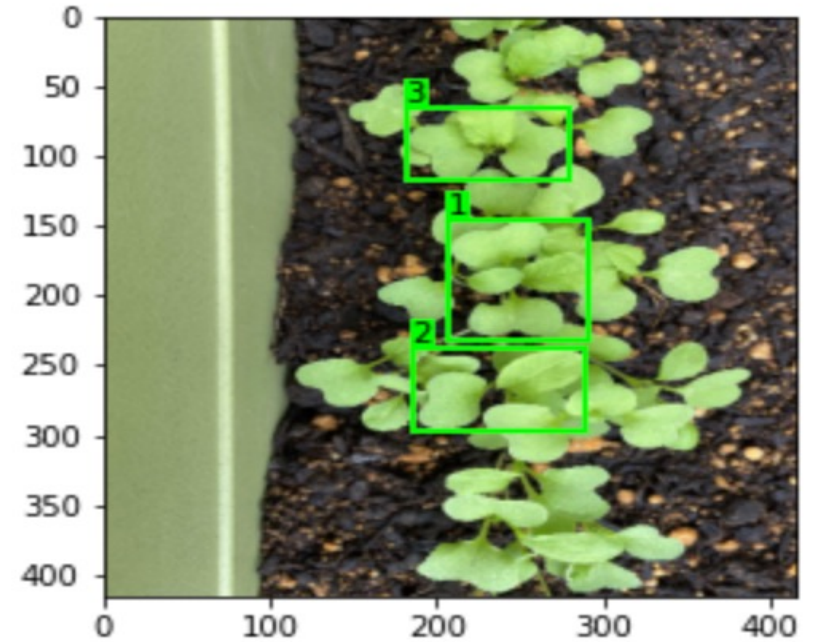
座標取得

②芽が密集している範囲を算出

①で取得した座標をデータ加工せずに
③の手法を行うと算出はしてくれるものの間引かない芽の対象が密集しすぎたりしてしまうことがある。(右写真)
またクラスタリングがうまくいかないと間引かない芽の対象が離れすぎてしまって一直線上にならないこともあった。

改善策

芽が密集している場所(右写真の場合、密集しているx座標の範囲)を取得し、その範囲内の芽をクラスタリングすれば間引かない芽の対象が一直線上になり、一定の間隔も開くのではないかな???



②芽が密集している範囲を算出

芽が密集しているところを探すために
X座標の分布図を可視化し、最頻値を算出

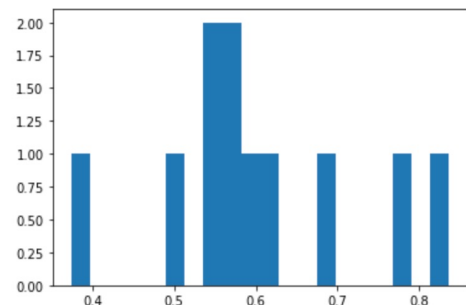
```
import numpy as np
import matplotlib.pyplot as plt

x = df['X']

n, bins, patches = plt.hist(x, bins=20)

# ヒストグラムの表示
plt.show()

mode_index = n.argmax()
# 最も度数の多い階級
print('最も度数の多い階級:(' + str(bins[mode_index]) + ', ' + str(bins[mode_index+1]) + ')')
# 最頻値
print('最頻値:' + str((bins[mode_index] + bins[mode_index+1])/2))
mode = (bins[mode_index] + bins[mode_index+1])/2
```



最も度数の多い階級: (0.535757, 0.558894)
最頻値: 0.5473255



最頻値付近に位置する芽の座標を取得

	X	Y	w	h	Area
0	0.552885	0.219952	0.235577	0.127404	0.030013
1	0.597356	0.453125	0.204327	0.209135	0.042732
2	0.569712	0.643029	0.250000	0.141827	0.035457
3	0.581731	0.090144	0.163462	0.108173	0.017682
4	0.543269	0.903846	0.192308	0.168269	0.032359
5	0.606971	0.814904	0.247596	0.100962	0.024998
6	0.497596	0.478365	0.216346	0.115385	0.024963

③範囲内でクラスタリング・ クラスター内で面積が一番大きい芽を表示

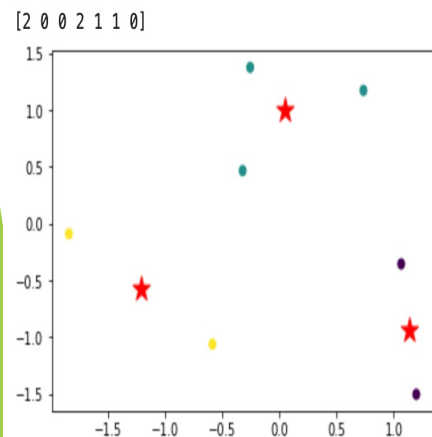
②で求めた座標をクラスタリング

```
# データの整形
X=df[["X","Y"]]

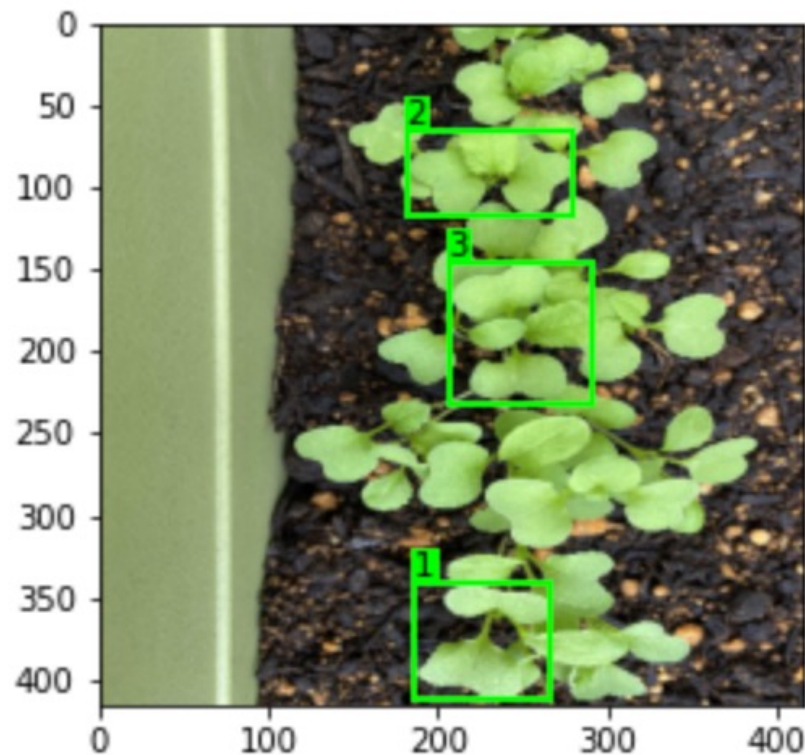
sc=preprocessing.StandardScaler()
sc.fit(X)
X_norm=sc.transform(X)

# クラスタリング
cls = KMeans(n_clusters=3)
pred_cls = cls.fit_predict(X)
print(pred_cls)
result = cls.fit(X_norm)

# 結果を出力
plt.scatter(X_norm[:,0],X_norm[:,1], c=result.labels_)
plt.scatter(result.cluster_centers_[0],result.cluster_centers_[1],s=250, marker='*',c='red')
plt.show()
```



クラスター内で一番面積が大きい芽を表示



- 1・提案
- 2・システム構成図
- 3・本葉がある芽の検知について
- 4・葉面積算出について
- 5・間引かない芽(残す芽)の発見について

6・まとめ

まとめ

今回は葉の面積が大きければ芽の成長は早いと断定し、実装まで行ったが実際には芽の高さ、色合いなど考慮しなければならない点がたくさんある。なので次このようなプロジェクトをする時は上記の点を考慮した手法で行っていく。

また間引かない芽の発見を一応発見することには成功したが今回はルールベース手法を用いて発見したので精度はあまり良くないと考えている。深層学習を取り入れた手法にすれば精度向上も見込めるのでチャレンジしてみたい。