

資料 3

カイワレダイコンの育成に 必要な情報を得る画像処理

本作「シミュレーションソフト」を
説明する前に、本作に必要な植物ラ
イブラリの製作過程を説明します。

カイワレダイコンの育成に必要な情報得る画像処理

※カイワレダイコン(以下「植物」という。)

<目的>

コンピュータによって植物の育成に必要な情報を随時、
得られるプログラムを作成する。以下3つを必要とする。

- ・植物の伸長(小数点第一位まで)
- ・植物の成長過程での時期(発芽期、成長期、収穫期、過剰成長期、測定不可)
- ・植物育成 LED ライトの入り・切り(成長期・収穫期：ON)

以上の項目以外にも、 シミュレーションソフトのために必要なプログラムも作成する。

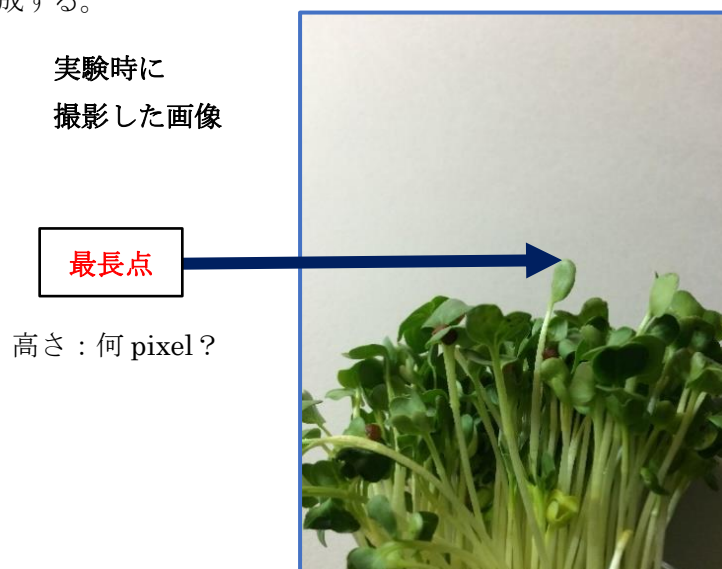
- ・元の画像を 1/4 リサイズし、罫線のある画像を生成

<実験方法>

植物の情報を得るには伸長が重要となる。

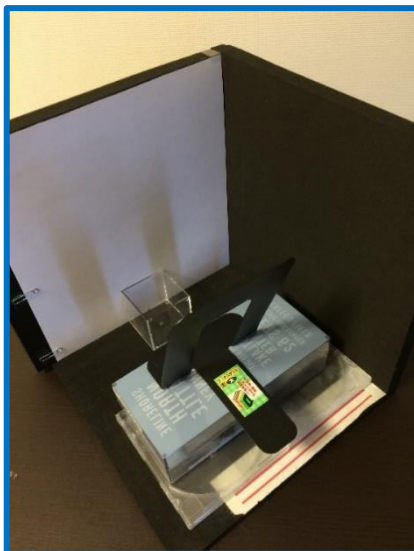
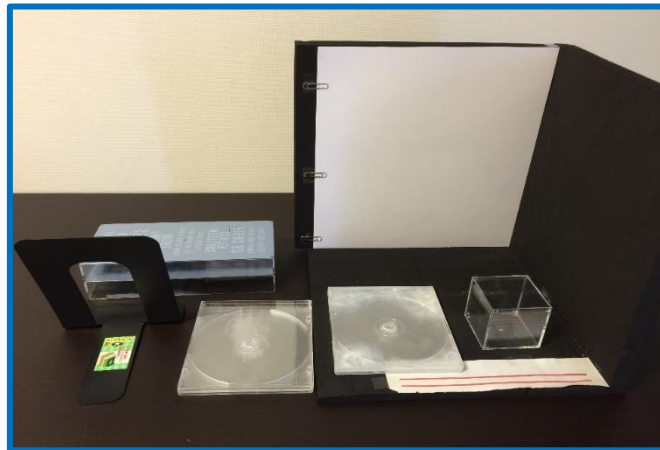
植物の伸長を得るにはいくつかの方法がある。赤外線センサーなどの電子機器を利用して伸長を得る方法・測量技術を用いて精密な伸長を得る方法などがあるが費用や技術が限られているためできない。費用や技術力を考慮して以下の実験方法とする。

植物を対象に横から随時、撮影する。画像の位置が変わらないよう、植物の育成する位置とカメラの位置を固定し撮影。その画像から伸長を得る。実験1、植物を測る上で2cm、3cm…13cmの境界線(高さ:y 軸)の値(単位:pixel)が画像のどの位置なのかを求めるプログラムを作成し、求めたデータを実験2で活用する。実験2、随時、カイワレダイコンの個々の成長にばらつきがあるためその中で一番長く成長した点(以下「最長点」という。)の値(単位:pixel)を求めるプログラムを作成する。そして、最長点の値を元に、上記「目的4つ」のプログラムを作成する。



＜実験環境の構築＞

コーナンや100円ショップなどから購入し5000円以内で用具を購入
家にあるものも活用する。



<実験 1>

植物を測る上で 2 cm、3 cm…13 cm の境界線(高さ:y 軸)の値(単位:pixel)が
画像のどの位置なのかを求めるプログラムを作成し、求めたデータを実験 2 で活用

設備環境が保証されないため実験結果データが正しく出るか不安定ではあるのでカメラ等
をしっかり固定し、植物とカメラの距離に気を付けながら撮影する(画像 1)。

A4 の紙に 1 cm の間隔で赤色の線 16 本を作成する(画像 2)。その用紙を育成する位置に置
き、画像内に 12 本の赤色の線が映るよう撮影する(画像 3)。

また、下から見て、最初の赤色の線を 2 cm の境界線とし最後の線を 13 cm とする。

画像 1



画像 2



画像 3



13 cm 境界線

画像の幅 : 中央・高さ : 0~3263(pixel)か
ら各境界線の幅をピクセル単位で求める

画像をプログラムで扱いやすくするため
に「(1)元の画像.JPG=>(2)二値化画像-モ
ノクロ.bmp(修正あり)=>(3)二値化画像-
24 ビット.bmp(この画像を使用)」に変
換。

2 cm 境界線

実験 1 で作成するプログラムの情報

(C 言語)

コンパイラ :

Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland

実行ファイル :

- ・「./実験 1 /実験 1 .exe」 実行ファイル

- ・ (3) 二値化画像-24 ビット.bmp

- ・ 幅 : 中央・高さ : 0~3263(pixel)の RGB 色数値.txt (色数値 : 「0 0 0」 か 「ff ff ff」)

※プログラムで扱いやすくするためにモノクロビットにしてから 2 4 ビットマップに変換

1~3 の画像を参照する場合「./実験 1 /実験 1 で使用した画像/」のディレクトリにある

※コードレビューについて :

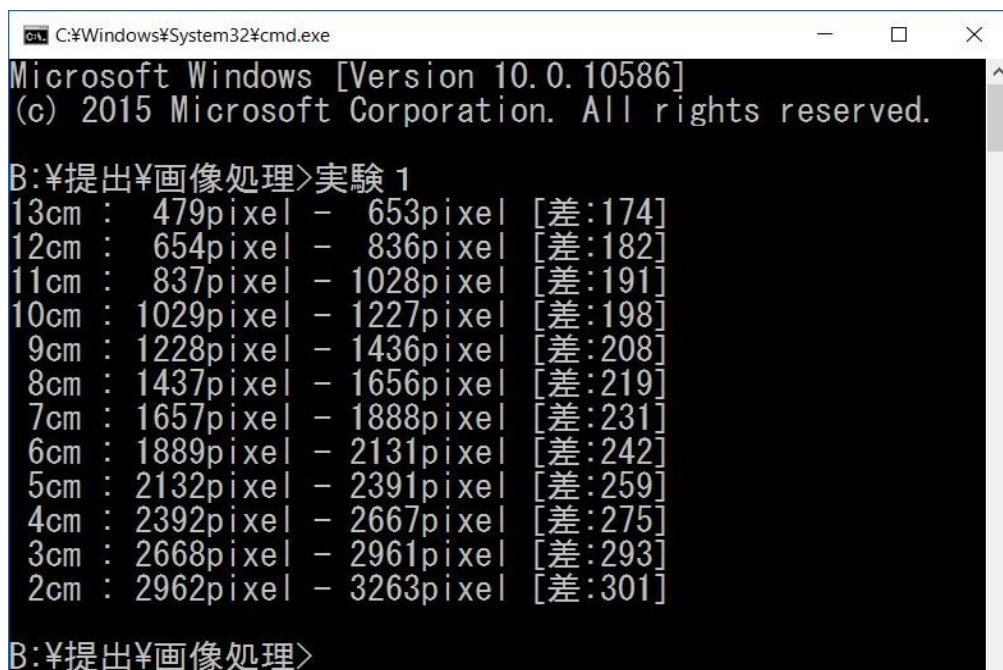
「./提出/src/実験 1 /」フォルダ内に「実験 1 .c」があります。

また、詳細なコメントはしていません。

実行方法

1. コマンドプロンプトを起動
2. カレントディレクトリを「./実験 1 /」に設定
3. コマンド入力「>実験 1」
4. 最初の行「13cm : 479pixel - 653pixel [差:174]」

「479」は 1 3 c m の境界線の値・「653」は 1 2 c m の境界線を超過する境界線の値



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

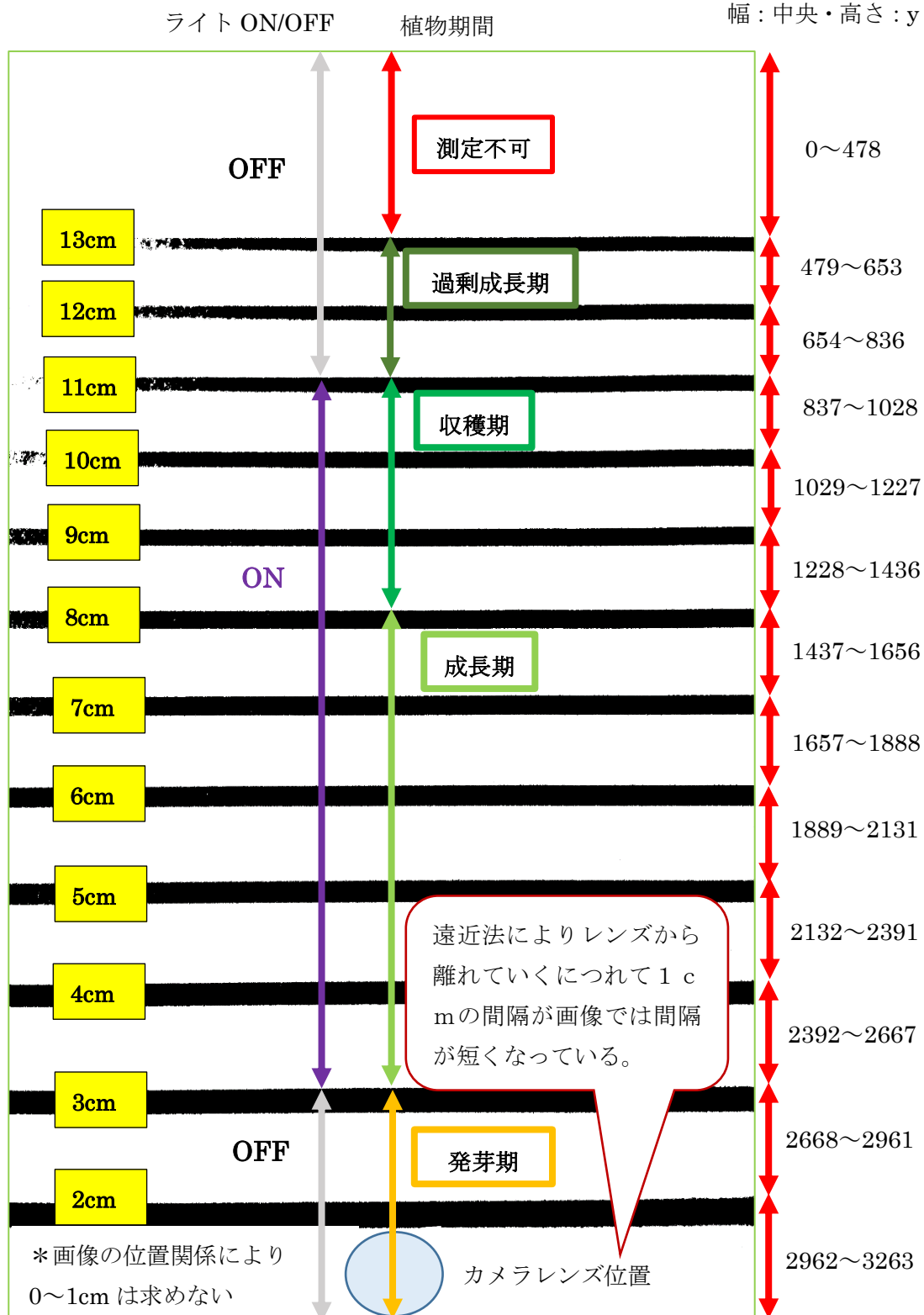
B:\提出\画像処理>実験 1
13cm : 479pixel - 653pixel [差:174]
12cm : 654pixel - 836pixel [差:182]
11cm : 837pixel - 1028pixel [差:191]
10cm : 1029pixel - 1227pixel [差:198]
9cm : 1228pixel - 1436pixel [差:208]
8cm : 1437pixel - 1656pixel [差:219]
7cm : 1657pixel - 1888pixel [差:231]
6cm : 1889pixel - 2131pixel [差:242]
5cm : 2132pixel - 2391pixel [差:259]
4cm : 2392pixel - 2667pixel [差:275]
3cm : 2668pixel - 2961pixel [差:293]
2cm : 2962pixel - 3263pixel [差:301]

B:\提出\画像処理>
```

撮影した画像(プログラムで扱いやすいよう 2 値化した画像)と
プログラムの出力結果、その他関連することをまとめた図

単位 : pixel

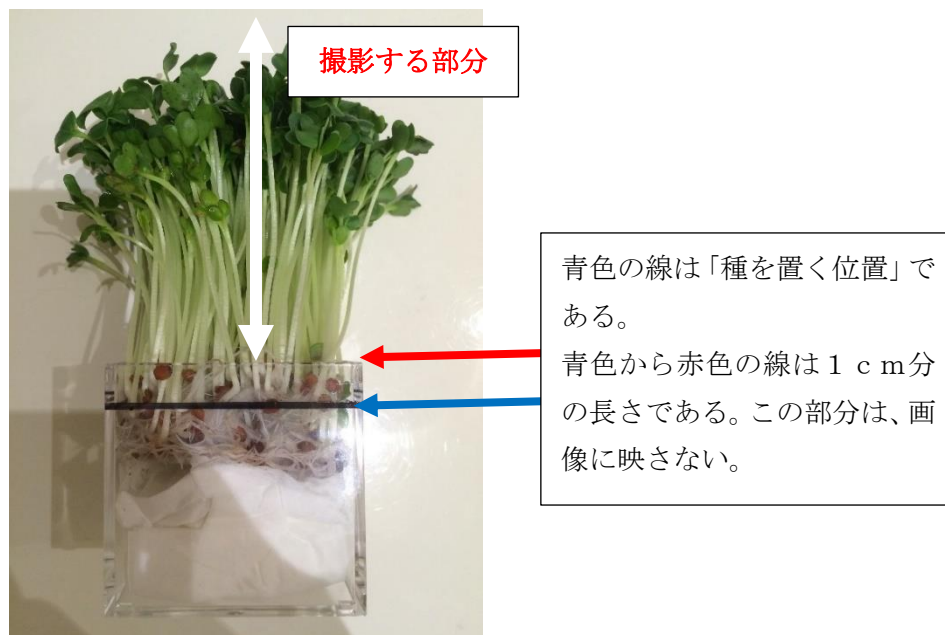
幅 : 中央・高さ : y



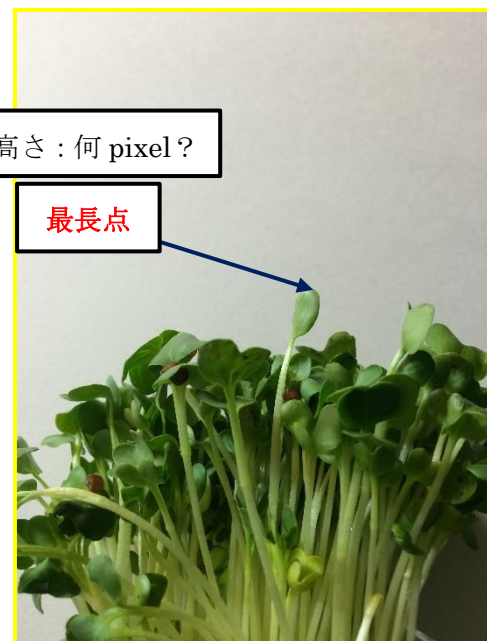
<実験 2>

植物を対象に撮影した画像から最長点の値(単位:pixel)を
求められるプログラムを作成

背景画像と植物育成時の画像を差分する方法と、植物育成時の画像をエッジ検出する方法がある。「差分する方法」では、2枚の画像が必要になり設備環境が整っていない中で行うには不向きと考え、「エッジ検出する方法」を模索する。



今後、この画像を使用する



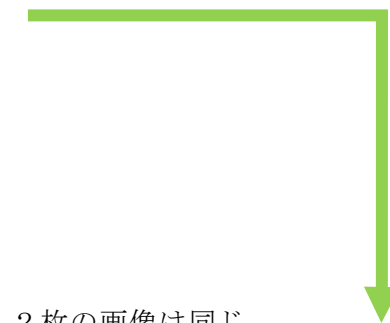
1. 植物画像からエッジ検出をし、最長点を求める

※カラー画像をエッジ検出する前に画像をグレースケールにする必要がある。

1. 元の画像をグレースケールに変換



2. エッジ検出



2枚の画像は同じ
大きさである。

幅 (x 軸) : 2 4 4 8 pixel

高さ (y 軸) : 3 2 6 4 pixel

y の値で伸長を求める

3. 最長点を求める

2 値画像なので、(白 :
2 5 5) の位置を探索
し、画像の左上の端か
らピクセルごとに右に
移動し最長点を求める
for (y ...) for (x ...)



コード：

```
Mat edge, gray = imread(argv[1], IMREAD_GRAYSCALE);
Canny(gray, edge, 40, 200);
for (int y = 0; y < edge.rows; y++) {
    for (int x = 0; x < edge.cols; x++) {
        int intensity=edge.at<unsigned char>(y, x);
        if (intensity)return y;
    }
}
return -1;|
```

出力結果：

(最長点：y 軸) **1 4 2 6** (pixel)

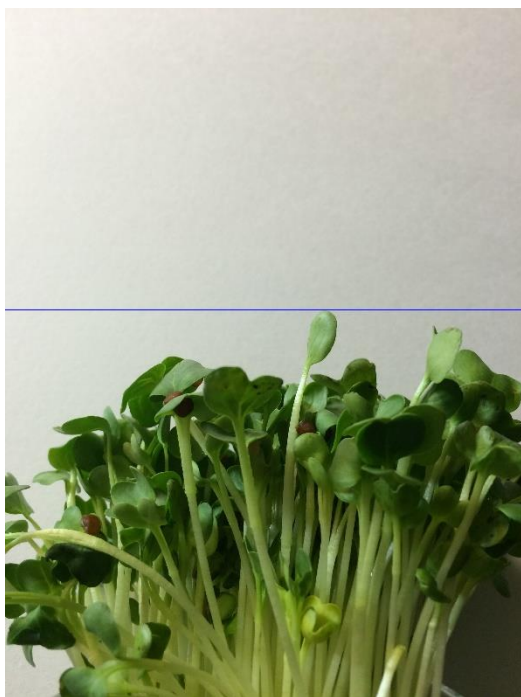
この出力結果が正しく最長点を求められているか確認する。

以下、確認するため最長点の位置を青 (255,0,0) の線を元の画像に印字して画像生成する

コード：

```
int pixel = y;
Mat gridpicture = imread(argv[1]);
if (pixel >= 479 && pixel <= 2668)//実験1でのデータを使用
    for (int i = 0; i < 6; i++, pixel--)
        for (int j = 0; j < gridpicture.cols; j++)
            gridpicture.at<Vec3b>(pixel, j) = Vec3b(255, 0, 0);
imwrite("./出力画像//(3)最長点位置確認.JPG", gridpicture);//画像出力
```

出力結果：(画像)



まとめ

以上の結果から最長点の位置が正確であると言える。

よって、「植物画像からエッジ検出をし、最長点を求める」プログラムを今後、使用する。

最長点の値を元に...

2. リサイズし野線ある画像を生成
3. 伸長を求める
4. カイワレダイコンの各期間
5. 成長促進ライトの入り切りの各プログラムを作成

2. 元の画像を 1/4 リサイズし、罫線のある画像を生成

実験 1 で求めたデータと最長点の値を使用する。

※Iot を想定したシミュレーションソフトを製作しますので、ネットワークを使用する。よって、通信の安定性を高めるため、元の画像を 1/4 にリサイズする。またリサイズした画像で罫線のある画像を生成する。

コード：

```
int resizepixel = (int)floor((double)y / 4); //最長点を1/4
Mat origin = imread(argv[1]), resizepicture;
resize(origin, resizepicture, cv::Size(), 0.25, 0.25, INTER_AREA); //リサイズ
/*リサイズ画像で罫線ある画像を作成*/
int lkup_pic[12] = { 119, 163, 209, 257, 307, 359, 414, 472, 533, 598, 667, 740 }; //実験 1 の各データを1/4にする
int lkup_num[12] = { 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1 };
Vec3b r = Vec3b(0, 0, 255); //赤
Vec3b b = Vec3b(255, 0, 0); //青
Vec3b gr = Vec3b(0, 255, 255); //黄
for (int i = 0; i < 12; i++){
    for (int j = 1; j <= lkup_num[i]; j++, lkup_pic[i]--){
        if (i == 2 || i == 5)
            for (int k = 0; k < resizepicture.cols; k++)resizepicture.at<Vec3b>(lkup_pic[i], k) = gr;
        else for (int k = 0; k < resizepicture.cols; k++)resizepicture.at<Vec3b>(lkup_pic[i], k) = r;
    }
}
if (resizepixel >= lkup_pic[0] && resizepixel <= lkup_pic[10])
    for (int i = 0; i < 2; i++, resizepixel--)
        for (int j = 0; j < resizepicture.cols; j++)resizepicture.at<Vec3b>(resizepixel, j) = b;
imwrite("./出力画像/(5)罫線あり.JPG", resizepicture);
```

出力結果：

元画像から 1/4 リサイズした画像



罫線のある画像



3. 伸長を求める

以下の表は処理に必要なデータ表である。

プログラム1の出力結果

1 4 2 6

(最長点)を参考に伸長を求める。

例外処理：以下、3つの条件

- ・⑩番目の境界線「479」と同じ値ならば、「13.0cm」である
- ・⑩番目の境界線「479」より小さければ、「-2.0(測定不可)」である
- ・①番目の境界線「2668」より大きければ、「-1.0(発芽期)」である

これらの条件にどれも当てはまらなければ処理を続行。

「1426」は当てはまらないので続行。

最長点が⑨番目から①番目の境界線の各数値以下の場合だったときの添字を探索する。

(以下、探索添字とする) 「1426」の探索添字は⑤番目

「num[N]-num[N-1]」に「探索添字」を代入し「差」を求める。

「1426」の場合、「209」

「差」は1cm分のピクセル数である。また、1cmを「差」で割った値と「num[N]-最長点」を掛けた値、そして、「探索添字」を足した値を浮動小数(以下、仮伸長とする)で求める。

「1426」の仮伸長は、「5.052631578947368」

「仮伸長」を小数点第二位で切り捨て、3(発芽期の3cm)を足した値が伸長である。

よって「1426」の伸長は「8.0」となる。

使用した植物の実際の伸長は「8.2」だった。

つまり、ある程度の近い値が求められていることが分かる。

以上、方法をプログラムにする。

N	1～13cm の実数	境界線(num) 単位:pixel	num[N]- num[N+1]	処理に必要な データ表
⑩	13cm	479	※	
⑨	12cm	654	175	※実験1のデータを使用
⑧	11cm	837	183	
⑦	10cm	1029	192	
⑥	9cm	1228	199	
⑤	8cm	1437	209	
④	7cm	1657	220	
③	6cm	1889	232	
②	5cm	2132	243	
①	4cm	2392	260	
①	3cm	2668	276	
*	2cm	2962	※	N: 添字 ⑨: 探索時の開始点 ①⑩: 例外処理に使用する行(データ) *※: 伸長を求めるのに使わない行(データ)
*	1cm	(3264)	※	

コード：

```
int i;
int[] pixel_num = new int[11]{2668,2392,2132,1889,1657,1437,1228,1029,837,654,479};
if (pixel == pixel_num[10]) return 13.0; //伸長
else if (pixel < pixel_num[10]) return -2.0; //測定不可
if (pixel > pixel_num[0]) return -1.0; //発芽期
for (i = 9; i >= 0; i--) if (pixel <= pixel_num[i]) break;
double result, cmp;
cmp = (double)i + (double)1 / (double)(pixel_num[i] - pixel_num[i + 1]) * (double)(pixel_num[i] - pixel);
result = (double)Math.Truncate(cmp * 10.0) / 10.0 + 3;
return result;
```

出力結果：「1426」=>「8.0」

4. カイワレダイコンの各期間を求める

0cm～3cm 未満を発芽期(－1)・3cm 以上～8cm 未満を成長期・8cm 以上～11cm 未満を収穫期・11cm 以上～13cm 以下を過剰成長期・13cm 超過で測定不可(－2)とする。

コード：

```
double lenght = result;
if (lenght == -2) printf("測定不可\n\n");
else if (lenght == -1) printf("発芽期\n\n");
else if (lenght < 8) printf("成長期\n\n");
else if (lenght < 11) printf("収穫期\n\n");
else printf("過剰成長期\n\n");
```

出力結果：「1426」=>「成長期」

5. 植物育成 LED の入り切り求める

実験1のデータを使用

条件：最長点が「837を超過かつ2668以下」であれば植物育成 LED：ON
でなければ OFF

コード：

```
if (pixel > 837 && pixel <= 2668) printf("植物育成LED：ON\n");
else printf("植物育成LED：OFF\n");
```

出力結果：「1426」=>「植物育成 LED：ON」

< 1～5のプログラムをまとめて実行できる「植物画像処理.exe」作成 >

(C++/64 ビットアプリケーション)

コンパイラ：Visual Studio 2015 community

ライブラリ：Opencv3.1 使用(64 ビット)

実行ファイル・フォルダ：

- ・「./実験 2 /植物画像処理/植物画像処理.exe」実行ファイル
- ・「./実験 2 /植物画像処理/出力画像」内に 5 つの画像ファイルを生成

著作権が関わるファイル：

「VC++ランタイムライブラリ」

msvcp140.dll・concert140.dll・vcruntime140.dll

「Opencv ライブラリ(bin)」

opencv_world310.dll

以上、ファイルの著作権は資料 1 (本書)の「著作権が関わるライブラリ・ファイルの使用について」

記載しています。

※コードレビューについて

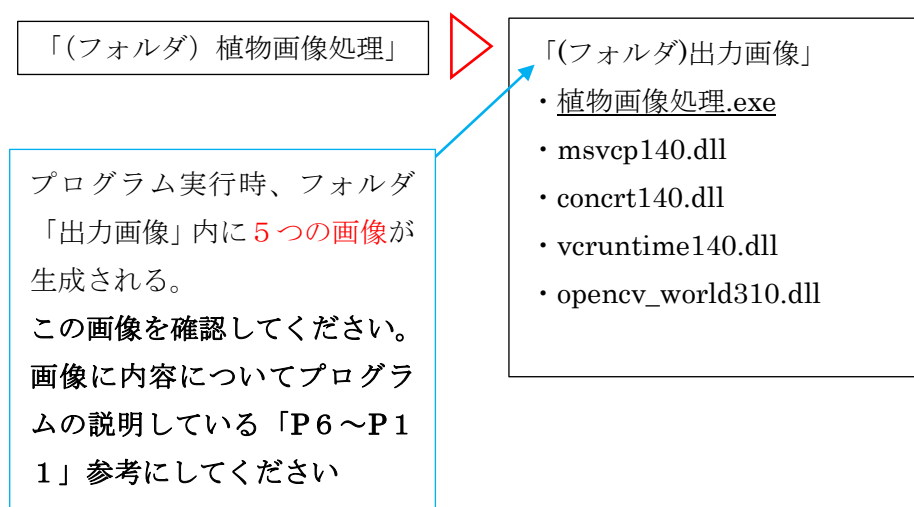
「./実験 2 /植物画像処理」内に「植物画像処理.sln」があります。

「Visual Studio 2015 community」で開いてください。

コードを見ることができると思いますが、OpenCV3.1 を利用しているので

エラー非表示・ビルド等をする際には資料 1 (本書)の「コードレビューについて」=>「①」を参考に設定して下さい。

ファイル・フォルダ構成：



実行方法

フォルダ「./テスト用画像」内にある複数枚の画像を使用します。

その中の一つの画像ファイルのディレクトリを引数とする。

複数枚あるので違う画像で数回、実行できる。

コマンド入力：

1. コマンドプロンプト起動
2. カレントディレクトリを「./実験 2 /」に設定
3. コマンド入力「>植物画像処理 引数1 (./テスト用画像/----.JPG)」
4. テスト用画像から抜き出した植物の情報が画面上に出力・
元の画像から生成した5つの画像が「(フォルダ)出力画像」に出力される。

出力内容と画像

```
C:\Windows\System32\cmd.exe
B:\植物画像処理>植物画像処理 B:\テスト用画像\image2. JPG

「最長点をピクセル単位で特定する」ライブラリ化(dllファイル:C++)
(1)「./出力画像」にグレースケール画像を作成。
(2)「./出力画像」にエッジ画像を作成
最長点をピクセル単位で表示
高さ：1426
(幅：1506)(色値：255)

「最長点の位置が正しいか確認」
(3)「./出力画像」に最長点の位置確認画像を作成。

「リサイズ」ライブラリ化(dllファイル:C++)
(4)「./出力画像」にリサイズ画像を作成

「リサイズ画像で野線ある画像を作成」ライブラリ化(dllファイル:C++)
(5)「./出力画像」に野線ありの画像を作成

「伸長を求める」ライブラリ化(dllファイル:C#)
長さ：8.000000cm

「カイワレダイコンの各期間」(直接記述)
収穫期

「植物育成LEDの入り切り」(直接記述)
植物育成LED：ON
B:\植物画像処理>
```



＜植物専用のライブラリの作成＞

シミュレーションソフト(C#)で動かすため、「.NET 対応」である C++/CLI または C#でライブラリを作成。

プログラム 1. 植物画像からエッジ検出をし、最長点を求める

プログラム 2. 元の画像を 1/4 リサイズし、罫線のある画像を生成

プログラム 3. 伸長を求める

プログラム 4. カイワレダイコンの各期間を求める

プログラム 5. 植物育成 LED の入り切り求める

※コードレビューについて：

「./MyLibrary /plant_library/...」内に各ライブラリのフォルダがあります。

「Visual Studio 2015 community」で開いてください。

OpenCV3.1 を利用している自作ライブラリはコードを見ることができると思いますが、エラー非表示・ビルド等をする際には資料 1 (本書)の「コードレビューについて」=>「①」を参考に設定して下さい。

植物ライブラリ

プログラム 1：

(C++/CLI・64 ビット dll)

コンパイラ：Visual Studio 2015 Community

ライブラリ：Opencv3.1 使用(64 ビット)

dll ファイル・フォルダ名：

plant_pixel.dll

(フォルダ名)「plant_pixel」// Visual Studio 2015 Community

プログラム 2：

プログラム 2 では、リサイズし、罫線のある画像を出力するプログラムだった。本ソフトのために「リサイズ」「罫線のある画像を出力」に 2 つに分けて、ライブラリを作成。

(C++/CLI・64 ビット dll)

コンパイラ：Visual Studio 2015 Community

ライブラリ：Opencv3.1 使用(64 ビット)

dll ファイル・フォルダ名：

plant_resizepicture.dll・plant_gridpicture.dll

(フォルダ名)「plant_resizepicture」// Visual Studio 2015 Community

(フォルダ名)「plant_gridpicture」// Visual Studio 2015 Community

プログラム 3 :

このライブラリは OpenCV を使用しないため、C#で作成する

(C#・64 ビット dll)

コンパイラ : Visual Studio 2015 Community

dll ファイル・フォルダ名 :

plant_lenght.dll

(フォルダ名)「plant_lenght」// Visual Studio 2015 Community

※プログラム 4・5 については「シミュレーションソフト」に直接記述する。

上記より、4つの植物ライブラリを「シミュレーションソフト」で使用する。