

令和 4 年度
卒業研究論文

ステレオカメラを用いた農作物と雑草の判別

旭川工業高等専門学校

システム制御情報工学科 5 年

学籍番号：183123

氏名：篠原 啓希

指導教員：戸村 豊明

目次

第 1 章 序論	1
1.1 研究背景	2
1.2 研究目的	2
第 2 章 関連技術	3
2.1 OpenCV	3
2.2 ステレオカメラ	3
第 3 章 雑草と農作物の判別システムの構成	4
3.1 概要	4
3.2 農作物の畝に中心線を引く方法	5
3.2.1 緑色部分の抽出	5
3.2.2 ノイズ除去	6
3.2.3 農作物の領域の中心線の取得	6
3.3 カメラと農作物の距離の計算・推定	8
3.3.1 カメラキャリブレーション	8
3.3.2 入力画像の補正	11
3.3.3 カメラ農作物間距離の推定	12
第 4 章 雑草と農作物の判別システムの評価	13
4.1 実験方法	23
4.2 実験結果	23
4.3 実験に対する考察	34
第 5 章 結論	35
参考文献	36
謝辞	37
付録 プログラムソース	

第1章 序論

1.1 研究背景

近年、一般物体認識は様々な手法が開発されたことにより、デジタルカメラやスマートフォンに搭載されている顔認識機能[1]や自動運転に用いられる人認識[2]などが実用化されるなど、急激な発展を遂げている。

しかし、雑草と農作物の判別については実用例が少なく、判別の手法が確立されていないのが現状である。

雑草と農作物の判別が、どのような分野に応用できるか例を挙げると、図 1.1 に示すような除草作業時のカルチベーターの自動化や精度向上につながる事が考えられる。図 1.2 に示すようにカルチベーターは、一つのビームに刃を複数取り付け、トラクターに装着し走行することで畝間の雑草を刈り取るものであり、使用することにより除草剤の使用量を削減できる。このシステムにより、間違えて農作物を刈ることを減少させ、精度の高い除草が可能になる。



図 1.1 除草中のカルチベーター



図 1.2 カルチベーター

1.2 研究目的

本研究では図 1.3 のようなステレオカメラを用い、OpenCV による画像処理やカメラキャリブレーションを施して、農作物の畝に中心線を引く方法を使い雑草と農作物の判別をするとともに、カメラと農作物の距離を三角測量などの理論から求める。



図 1.3 ステレオカメラ

第2章 関連技術

2.1 OpenCV

OpenCV (Open Source Computer Vision Library)[3]は 1998 年に Intel が開発し、現在でも活発に開発を行っている無償のオープンソース画像処理ライブラリ集である。OpenCV には、画像の変換処理やテンプレートマッチ、物体認識、映像解析などのアルゴリズムが多数用意されている。

さらに、静止画像だけでなく、ビデオカメラからの入力映像にも対応している。近年では、高速なマイクロプロセッサや USB2.0 などのインターフェース技術により高速なビデオキャプチャが可能になり、標準的な PC と安価な USB ウェブカメラを利用することにより、リアルタイムでのコンピュータビジョンが簡単にできるようになっている。

本研究では画像の各画素の青、緑、赤それぞれの要素の比較や変更、モルフォロジー変換処理や画像サイズの変更といった画像処理だけでなく、カメラキャリブレーションやその結果に応じた画像補正、入力映像の対応に OpenCV が使われる。

2.2 ステレオカメラ

本研究では、図 1.3 のような Web カメラ 2 台を雲台に取り付けたステレオカメラを用い、トラクターに乗せることを想定する。距離を測定するため焦点距離が互いに等しく、光軸が平行で、かつ各々の画像が平面上に存在するように配置する。

使用する Web カメラは図 2.1 に示すロジクール製の「c920n HD pro webcam」で、仕様を以下に示す。

- 最大解像度:1080p/30 fps – 720p/30 fps
- カメラ画素数 (メガピクセル) :3
- フォーカスタイプ:オートフォーカス
- レンズタイプ:ガラス
- 対角視野 (dFoV) :78°
- USB プロトコル:USB2.0



図 2.1 c920n HD pro webcam

第3章 雑草と農作物の判別システムの構成

3.1 概要

本研究では、農作物の畝の画像を入力とし、農作物の領域を取得して雑草と農作物の判別を行い、農作物の領域の特徴点を抽出し視差を求めることでカメラと農作物の距離を推定する。手順を図 3.1 に示す。

図 3.2、図 3.3 は造花を作物に見立てた、左右それぞれのカメラから取得した画像の例である。図 3.1 における(1)にあたる。

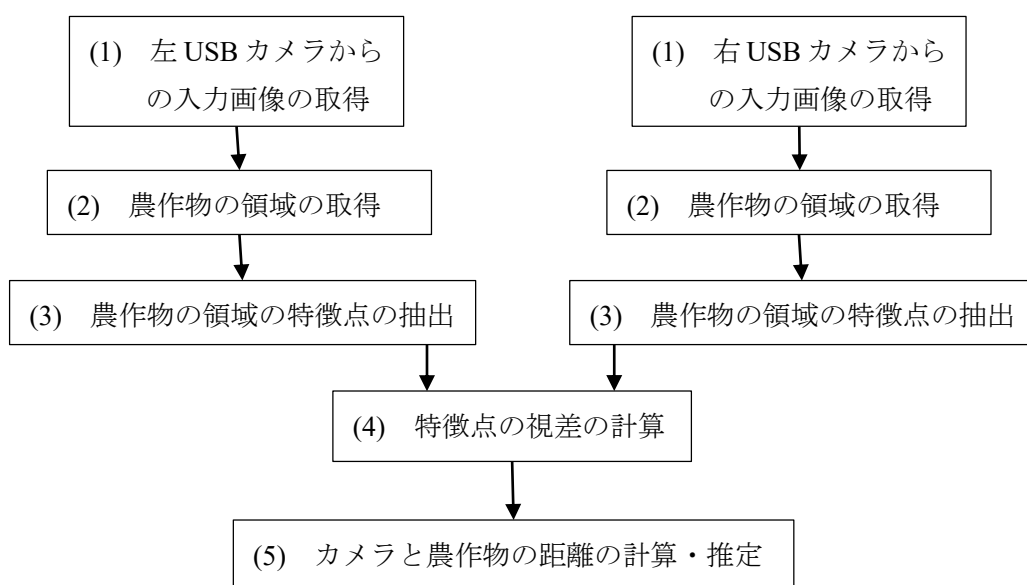


図 3.1 農作物の判別・距離推定手順

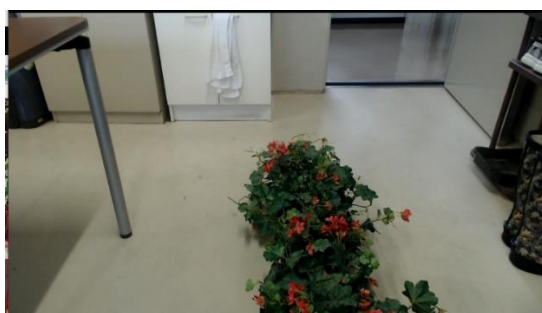


図 3.2 左カメラの画像例



図 3.3 右カメラの画像例

3.2 農作物の畝に中心線を引く方法

本節では、図 3.1 における(2)から(4)の手順として、農作物の領域の中心線を特徴点として抽出し、視差を計算する。抽出手順を図 3.4 に示す。

3.2.1 緑色部分の抽出

図 3.1 における(2)の手順として、農作物の領域の抽出のために入力画像に対して緑色部分の抽出を行う。本研究では各画素の青、緑、赤それぞれの要素を比較し、緑要素が他の要素よりも大きい画素を白、その他の画素を黒とした二値画像を生成することで抽出する。

図 3.5、図 3.6 はそれぞれ抽出する前の画像と抽出し生成した二値画像の例である。

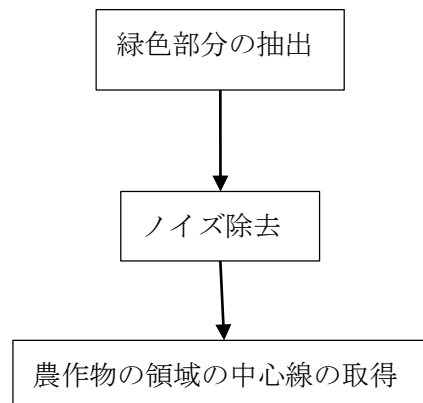


図 3.4 中心線の抽出手順

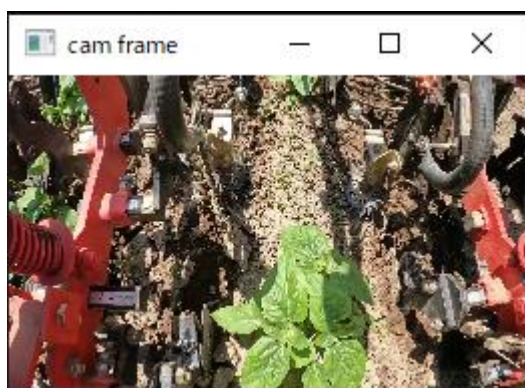


図 3.5 入力画像の例



図 3.6 二値画像の例

3.2.2 ノイズ除去

前節での処理ではノイズのほか、小さい雑草も緑色部分として抽出されるため、モルフォロジー変換[4][5]によるノイズ除去を行う。本研究では、収縮と膨張の順で繰り返すオープニング処理ののち、膨張と収縮の順で繰り返すクロージング処理を行う。

図 3.7 は、二値画像の図 3.6 のノイズを除去した画像の例である。

3.2.3 農作物の領域の中心線の取得

図 3.1 における(3)の手順として、前節で得られた領域の中心線を取得する。画像の各行において白い部分の右端と左端の画素の座標を足して 2 で割った座標の画素を赤で表示し、これをすべての行で繰り返して中心線を得る。

この処理を図 3.7 に行い、中心線の座標を赤で示した画像を図 3.8 に示す。

図 3.8 の段階では、ノイズや作物の隙間の影響で中心線が各行に複数ある場合があり、視差が定まらないため、本研究では作物の中心線が画像の中央付近にあると仮定し、画像の中心の列に一番近い座標の中心線を作物の中心線とする。図 3.9、3.10 では左右のカメラからの入力画像の例、図 3.11、3.12 では図 3.9、3.10 に対して中心線を定めた画像を示す。

図 3.1 における(4)の手順として、各列で得た左カメラの画像の中心線の座標から右カメラの中心線の座標を引いたものを視差とする。



図 3.7 ノイズ除去後の画像の例

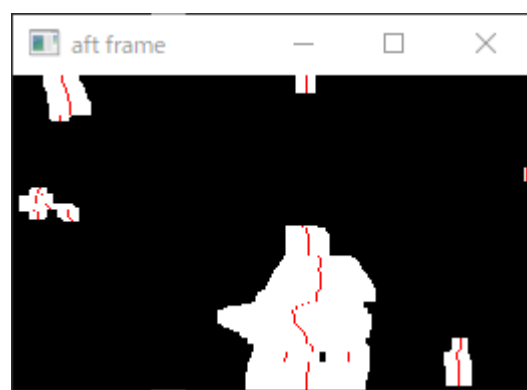


図 3.8 中心線を示した画像の例



図 3.9 左カメラの入力画像例

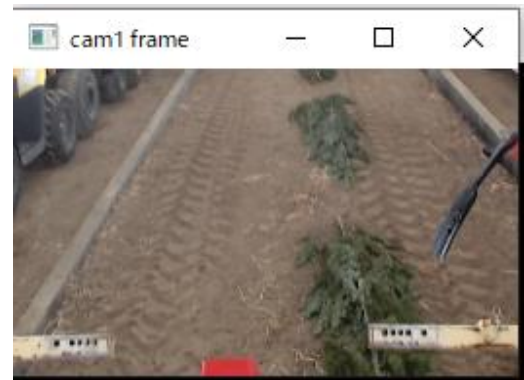


図 3.10 右カメラの入力画像例



図 3.11 左カメラの処理後の画像例



図 3.12 右カメラの処理後の画像例

3.3 カメラと農作物の距離の計算・推定

本節では図 3.1 における(5)の手順として、カメラと農作物の距離の計算・推定を行う。

3.3.1 カメラキャリブレーション

本研究では、二つのカメラを用いて、三角測量により距離を求める。これは焦点距離が互いに等しく、光軸が平行で、かつ各々の画像が平面上に存在するように配置する。

本研究ではロジクール製の「c920n HD pro webcam」という同機種の Web カメラを二つ使用する。

図 3.13 に示すように、両カメラ間の実距離 B は既知である。本実験では 279[mm]としている。

カメラの焦点距離を取得するには、カメラキャリブレーション[6]という機能を利用する。カメラキャリブレーションとは、撮影画像からカメラの位置姿勢や特性を推定することである。カメラをキャリブレーションするためには、カメラをモデル化し画像の写り方に影響するパラメータを求める必要がある。

パラメータには、カメラの 3 次元空間における位置姿勢を表す外部パラメータと、3 次元空間を 2 次元画像平面へ投影するためのパラメータやレンズやカメラそのものの特性を表す内部パラメータ行列 A がある。この 2 つのパラメータを世界座標の点 P と点に対応する画像座標の点 p の関係式 3.1 を用いて導出する。カメラの内部パラメータ行列 A から導かれるカメラ特性は図 3.14 のようになっている。ここでカメラの内部パラメータ A は式 3.2 のように表す。なお、 c_x と c_y は主点の座標である。カメラのレンズの特徴は撮影すること微妙に違うため、各パラメータは定まらないので、カメラキャリブレーションは実験時に必ず行う。

$$sp = A[R|T]P \quad (3.1)$$

$$A = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

ここで s では画像のスケール係数、 A は内部パラメータ行列(3×3 行列)、 $[R|T]$ は外部パラメータ行列(3×4 の回転並進行列)を表している。 R は回転行列を表し、 T は平行移動ベクトルを表す。OpenCV では、キャリブレーションの一つの手法である Zhang の手法が実装されている。Zhang の手法は、幾何特性が既知の平面パターンを多方向から 3 回以上撮影し得られた画像中の特徴点をもとにカメラパラメータを推定する。本実験では 20 回撮影する。平面パターンは図 3.15 のようなチェックパターンを使用し、横線と縦線の交点の座標値を式 3.1 に適用し、パラメータの推定を行う。この結果より内部パラメータが求められ、焦点距離 f が得られる。

同様にして、2つのカメラでキャリブレーションを行うことにより、二つのカメラ座標系間の回転並進行列が求まる。

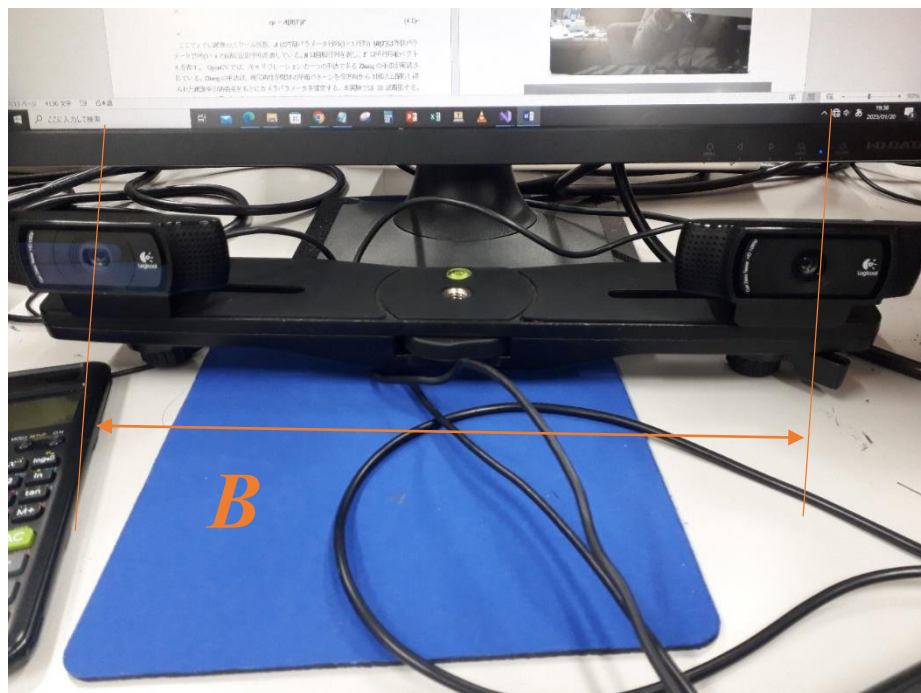


図 3.13 カメラ間距離

```
焦点距離:467.311177[Pixel]  
主点c_x:351.436836[Pixel]  
主点c_y:172.431728[Pixel]
```

図 3.14 内部パラメータの例

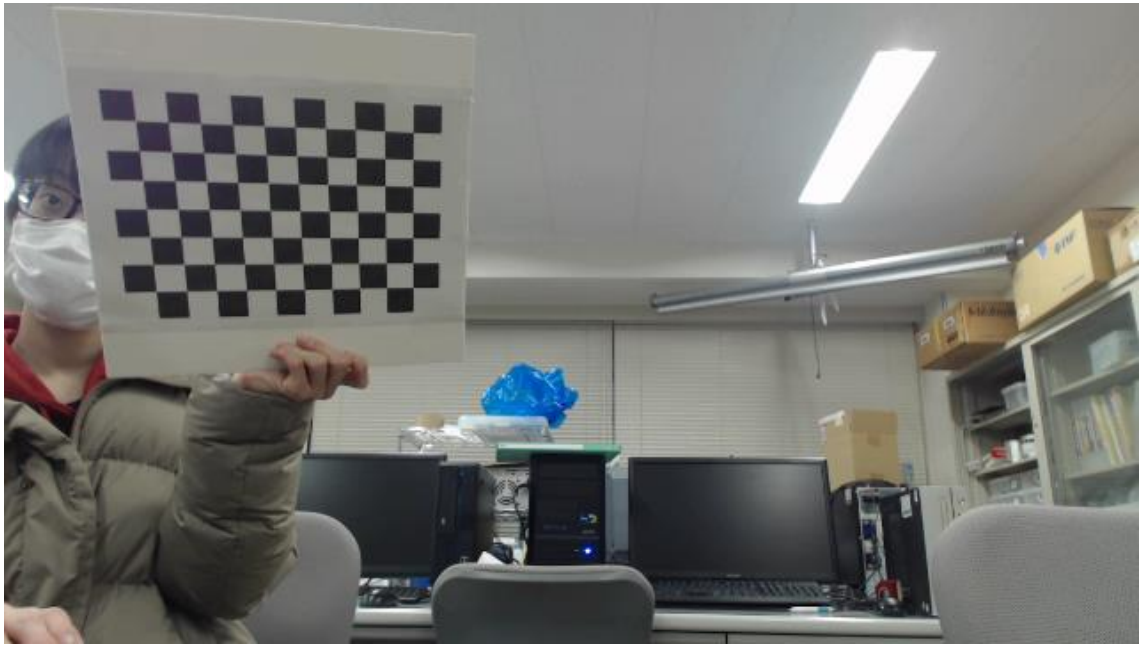


図 3.15 チェックパターンの例

3.3.2 入力画像の補正

前節で求めた各パラメータをもとに、入力画像の補正を行う。左右の入力画像の例とそれらを補正した画像の例を図 3.16～3.19 に示す。

補正した入力画像をもとに 3.2.1～3.2.3 で述べた処理を行う。



図 3.16 左カメラの入力画像例



図 3.17 右カメラの入力画像例



図 3.18 左カメラの補正後の画像例



図 3.19 右カメラの補正後の画像例

3.3.3 カメラ農作物間距離の推定

3次元座標の計算は本来、2つのカメラ画像を画像処理により平行化し、三角測量をする必要があるが、本研究では図 3.13 のようにカメラを平行に配置することにより、両画像間での点の対応付けはできているためより簡単に実距離の計算ができる[7]。三角測量を行うことにより導かれる実距離 x 、 y 、 z は式 3.3、式 3.4、式 3.5 より求まる。

$$x = \frac{Bx_l}{d} \quad (3.3)$$

$$y = \frac{By_l}{d} \quad (3.4)$$

$$z = \frac{Bf}{d} \quad (3.5)$$

ここで2画面での見え方の差であり視差と呼ばれ、 $d=x_l-x_r$ で表される。図 3.20 に示すように物体の奥行 z は視差 d に反比例するので、カメラからの奥行きが遠い物体は視差が小さく、奥行きが小さい物体は視差が大きくなる。

式 3.3、式 3.4、式 3.5 に 3.2.3 節で得られた中心線の座標を代入することで、3次元座標を求めることができる。本実験では奥行きである z の値を、カメラと農作物の距離として計算する。

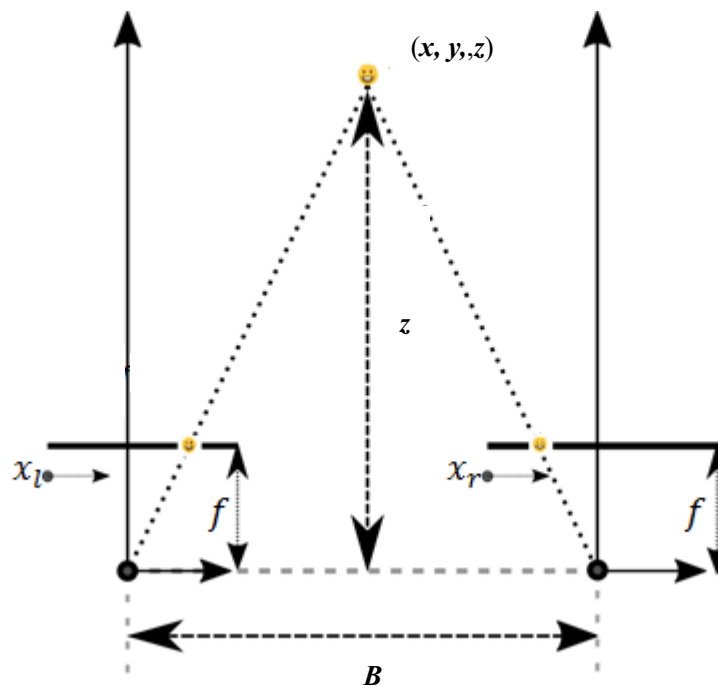


図 3.20 三角測量

第4章 雑草と農作物の判別システムの評価

本研究では、画像を入力として、農作物と雑草の判別を行い、中心線の座標からカメラとの距離を求める。

本研究での研究環境は以下のとおりである。

- OS:windows10 64ビット
- プロセッサ: Intel(R) Core(TM) i7-3770 CPU 3.40GHz
- 実装 RAM: 16.0 GB (15.9 GB 使用可能)
- Web カメラ:Logicool c920n HD pro webcam

4.1 実験方法

本研究の実験方法として、左右の映像を入力として、その画像を前章の処理を行うことで距離を求める。本研究では、造花を作物に見立てた映像に対し処理をし、得られたカメラとの距離が適切かどうかを検証するものとする。また、モルフォロジー変換のパラメータの変化による違いも考える。

入力画像の元サイズは 1920×1080 と大きいので、 640×360 にリサイズして処理する。また、カメラキャリブレーションで用いる画像のサイズは、入力画像の元サイズである 1920×1080 と縦横比を合わせるために 640×360 とする。

使用する造花の寸法を以下に示す。

- 幅:約 26cm
- 奥行き:約 135cm
- 高さ:約 27cm

4.2 実験結果

本研究では、造花の全景が見える 1 フレーム分の画像と計算結果を抜き出し評価する。本研究では奥行きのデータを csv ファイルに書き込むことを試みたが、ファイルに何も書き込まれなかったため、実行結果を、[縦の画素値]距離(m)のように表示し、スクリーンショットで保存する。縦の画素値が大きいほど手前側になる。この実験ではモルフォロジー変換のパラメータをオープニングでの収縮と膨張の回数を 4、クロージングでの膨張と収縮の回数を 12 としたものと、オープニングでの収縮と膨張の回数を 12、クロージングでの膨張と収縮の回数を 4 としたものの 2 つで行った。画像は 640×360 で保存している。

以下に、前者の条件での結果を示す。図 4.1~4.11 は奥行きのデータ、図 4.12、4.15 は左右の入力画像、図 4.13、4.16 は左右の補正後入力画像、図 4.14、4.17 は左右の中心線の処理後の画像を示す。

```
選択Microsoft Visual Studio デバッグ コンソール
151572,78828110992,119408[49] 3.165003m
[50] 3.165003m
[51] 3.165003m
[52] 3.165003m
[53] 3.165003m
[54] 3.165003m
[55] 3.165003m
[56] 3.165003m
[57] 3.165003m
[58] 3.165003m
[59] 3.165003m
[60] 3.165003m
[61] 3.165003m
[62] 3.165003m
[63] 3.165003m
[64] 3.165003m
[65] 3.165003m
[66] 3.165003m
[67] 3.165003m
[68] 3.165003m
[69] 3.165003m
[70] 3.165003m
[71] 3.165003m
[72] 3.165003m
[73] 3.165003m
[74] 3.165003m
[75] 3.165003m
[76] 3.165003m
[77] 3.165003m
[78] 3.165003m
```

図 4.1 実験結果 1-1

```
選択Microsoft Visual Studio デバッグ コンソール
[79] 3.165003m
[80] 3.165003m
[81] 3.165003m
[82] 3.165003m
[83] 3.165003m
[84] 3.165003m
[85] 3.165003m
[86] 3.165003m
[87] 3.165003m
[88] 3.165003m
[89] 3.165003m
[90] 3.165003m
[91] 3.165003m
[92] 3.165003m
[93] 3.165003m
[94] 3.165003m
[95] 3.165003m
[96] 3.165003m
[97] 3.165003m
[98] 3.165003m
[99] 3.165003m
[100] 3.165003m
[101] 3.165003m
[102] 3.165003m
[103] 3.165003m
[104] 3.165003m
[105] 3.165003m
[106] 3.165003m
[107] 3.165003m
[108] 3.165003m
```

図 4.2 実験結果 1-2


```
選択Microsoft Visual Studio デバッグ コンソール
[109] 3.165003m
[110] 3.165003m
[111] 3.165003m
[112] 3.165003m
[113] 3.165003m
[114] 3.165003m
[115] 3.165003m
[116] 3.165003m
[117] 3.165003m
[118] 3.165003m
[119] 3.165003m
[120] 3.165003m
[121] 3.165003m
[122] 3.165003m
[123] 3.165003m
[124] 3.165003m
[125] 3.165003m
[126] 3.165003m
[127] 3.165003m
[128] 3.165003m
[129] 3.165003m
[130] 3.165003m
[131] 3.165003m
[132] 3.165003m
[133] 3.165003m
[134] 3.165003m
[135] 3.165003m
[136] 3.165003m
[137] 3.165003m
[138] 3.165003m
```

図 4.3 実験結果 1-3

```
選択Microsoft Visual Studio デバッグ コンソール
[139] 3.165003m
[140] 3.165003m
[141] 3.165003m
[142] 3.165003m
[143] 3.165003m
[144] 3.165003m
[145] 3.165003m
[146] 3.165003m
[147] 3.165003m
[148] 3.165003m
[149] 3.165003m
[150] 3.165003m
[151] 3.165003m
[152] 0.531041m
[153] 0.529265m
[154] 0.515473m
[155] 0.515473m
[156] 0.515473m
[157] 0.515473m
[158] 0.515473m
[159] 0.515473m
[160] 0.515473m
[161] 0.515473m
[162] 0.515473m
[163] 0.515473m
[164] 0.515473m
[165] 1.376088m
[166] 1.451836m
[167] 1.451836m
[168] 1.438638m
```

図 4.4 実験結果 1-4

```
選択Microsoft Visual Studio デバッグ コンソール
[169] 1.438638m
[170] 1.388159m
[171] 1.376088m
[172] 1.376088m
[173] 1.376088m
[174] 1.376088m
[175] 1.376088m
[176] 1.376088m
[177] 1.376088m
[178] 1.376088m
[179] 1.376088m
[180] 1.364226m
[181] 1.352565m
[182] 1.352565m
[183] 1.352565m
[184] 1.364226m
[185] 1.536409m
[186] 1.739013m
[187] 1.739013m
[188] 1.739013m
[189] 1.739013m
[190] 4.057696m
[191] 4.057696m
[192] 4.057696m
[193] 4.057696m
[194] 4.057696m
[195] 4.057696m
[196] 4.057696m
[197] 4.057696m
[198] 4.057696m
```

図 4.5 実験結果 1-5

```
選択Microsoft Visual Studio デバッグ コンソール
[199] 4.057696m
[200] 4.057696m
[201] 4.057696m
[202] 4.057696m
[203] 4.057696m
[204] 4.057696m
[205] 4.057696m
[206] 4.057696m
[207] 4.057696m
[208] 4.057696m
[209] 4.057696m
[210] 4.057696m
[211] 4.057696m
[212] 4.057696m
[213] 4.395838m
[214] 4.395838m
[215] 4.654416m
[216] 4.654416m
[217] 4.654416m
[218] 4.654416m
[219] 1.352565m
[220] 1.352565m
[221] 1.478973m
[222] 1.507144m
[223] 1.507144m
[224] 2.003167m
[225] 2.003167m
[226] 2.003167m
[227] 2.260717m
[228] 2.260717m
```

図 4.6 実験結果 1-6

```
選択 Microsoft Visual Studio デバッグ コンソール
[229] 2.293481m
[230] 2.327208m
[231] 2.293481m
[232] 2.197919m
[233] 2.197919m
[234] 2.228875m
[235] 2.228875m
[236] 2.228875m
[237] 2.260717m
[238] 2.260717m
[239] 2.228875m
[240] 2.055197m
[241] 2.055197m
[242] 2.055197m
[243] 2.055197m
[244] 2.055197m
[245] 2.055197m
[246] 3.165003m
[247] 1.318751m
[248] 1.318751m
[249] 1.307853m
[250] 1.286587m
[251] 1.286587m
[252] 1.276211m
[253] 1.266001m
[254] 1.255954m
[255] 1.255954m
[256] 1.246064m
[257] 1.155111m
[258] 31.650032m
```

図 4.7 実験結果 1-7

```
選択 Microsoft Visual Studio デバッグ コンソール
[259] 26.375027m
[260] 19.781270m
[261] 3.680236m
[262] 3.680236m
[263] 3.680236m
[264] 3.680236m
[265] 3.102944m
[266] 3.102944m
[267] 5.275005m
[268] 5.275005m
[269] 5.275005m
[270] 4.057696m
[271] 2.472659m
[272] 2.472659m
[273] 2.472659m
[274] 2.472659m
[275] 2.472659m
[276] 2.472659m
[277] 2.472659m
[278] 2.472659m
[279] 2.472659m
[280] 2.472659m
[281] 2.472659m
[282] 2.472659m
[283] 2.472659m
[284] 2.472659m
[285] 2.472659m
[286] 2.228875m
[287] 52.750053m
[288] 52.750053m
```

図 4.8 実験結果 1-8

```
選択 Microsoft Visual Studio デバッグ コンソール
[289] 52.750053m
[290] 52.750053m
[291] 52.750053m
[292] 52.750053m
[293] 52.750053m
[294] 52.750053m
[295] 52.750053m
[296] 52.750053m
[297] 52.750053m
[298] 52.750053m
[299] 52.750053m
[300] 52.750053m
[301] 52.750053m
[302] 52.750053m
[303] 52.750053m
[304] 52.750053m
[305] 52.750053m
[306] 1.701615m
[307] 1.701615m
[308] 1.701615m
[309] 1.701615m
[310] 1.861767m
[311] 1.861767m
[312] 2.055197m
[313] 2.055197m
[314] 2.055197m
[315] 2.055197m
[316] 2.138516m
[317] 2.138516m
[318] 26.375027m
```

図 4.9 実験結果 1-9

```
選択 Microsoft Visual Studio デバッグ コンソール
[319] 26.375027m
[320] 26.375027m
[321] 26.375027m
[322] 26.375027m
[323] 26.375027m
[324] 3.680236m
[325] 3.680236m
[326] 3.680236m
[327] 3.680236m
[328] 3.680236m
[329] 3.680236m
[330] 3.680236m
[331] 4.057696m
[332] 4.057696m
[333] 4.057696m
[334] 1.163604m
[335] 1.163604m
[336] 1.163604m
[337] 1.163604m
[338] 3.680236m
[339] 3.680236m
[340] 3.680236m
[341] 3.680236m
[342] 3.680236m
[343] 3.680236m
[344] 3.680236m
[345] 5.456902m
[346] 5.456902m
[347] 5.456902m
[348] 5.456902m
```

図 4.10 実験結果 1-10

```
選択Microsoft Visual Studio デバッグ コンソール
[330] 3.680236m
[331] 4.057696m
[332] 4.057696m
[333] 4.057696m
[334] 1.163604m
[335] 1.163604m
[336] 1.163604m
[337] 1.163604m
[338] 3.680236m
[339] 3.680236m
[340] 3.680236m
[341] 3.680236m
[342] 3.680236m
[343] 3.680236m
[344] 3.680236m
[345] 5.456902m
[346] 5.456902m
[347] 5.456902m
[348] 5.456902m
[349] 5.456902m
[350] 5.456902m
[351] 1.027598m
[352] 1.041119m
[353] 1.048014m
[354] 5.861117m
[355] 5.651791m
[356] 5.651791m
[357] 5.651791m
[358] 5.651791m
[359] 5.651791m
```

図 4.11 実験結果 1-11



図 4.12 左カメラの入力画像 1



図 4.13 左カメラの補正後の入力画像 1

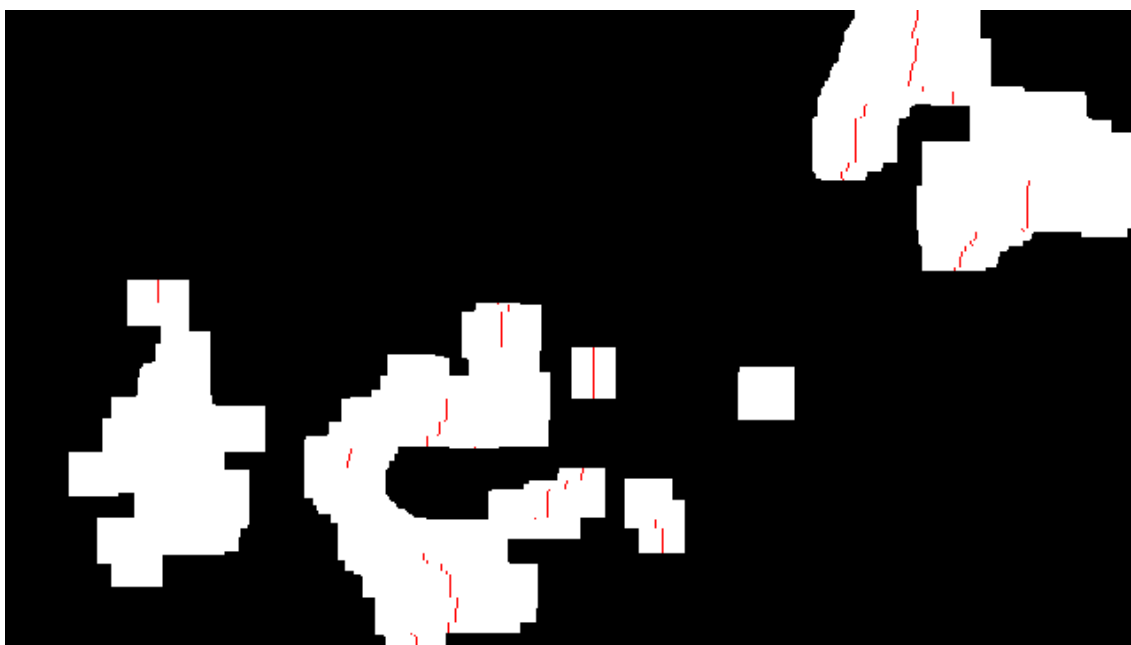


図 4.14 左カメラの中心線の処理後画像 1



図 4.15 右カメラの入力画像 1



図 4.16 右カメラの補正後の入力画像 1

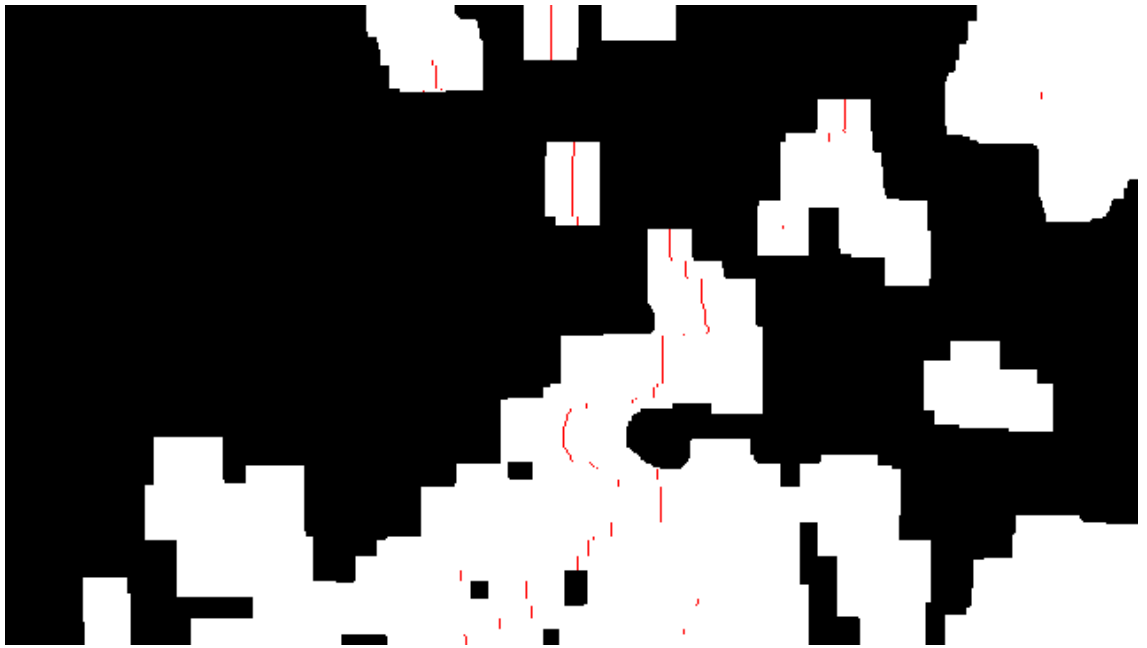


図 4.17 右カメラの中心線の処理後画像 1

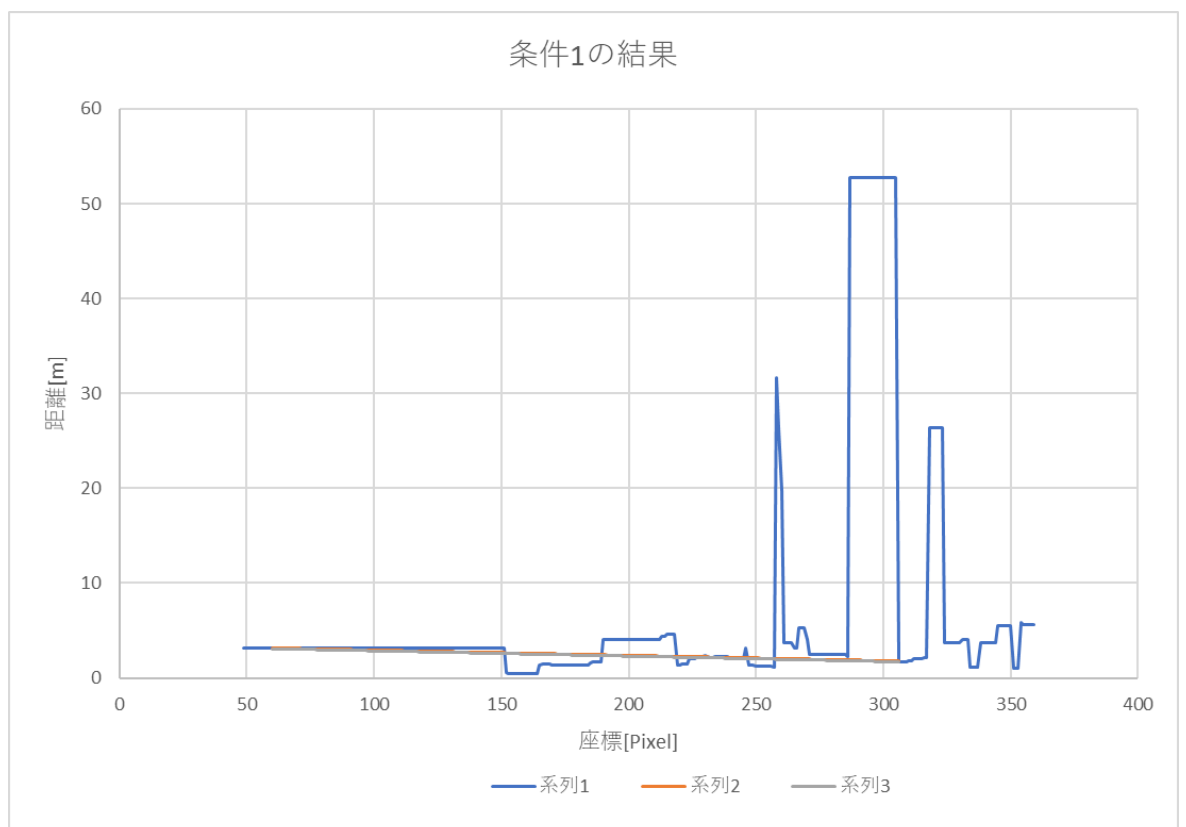


図 4.18 条件 1 の結果

図 4.13、4.16 から造花の一番奥の縦の座標値が 60 付近、一番手前の縦の座標値が 300 付近と考えられるので、図 4.1、4.9 から、60 ピクセル付近での距離から 300 ピクセル付近での距離を引いた値が増加の奥行きと等しくなると考えられる。60 ピクセルでの距離は 3.165003[m]、300 ピクセルでの距離は大きすぎるので、306 ピクセルでの距離は 1.7016115[m]と分かり、60 ピクセルでの距離から 306 ピクセルでの距離を引いた値は 1.463388[m]と計算できる。これを造花の奥行きと比較すると、造花の奥行きは 1.35[m]なので、その差を増加の奥行きで割ることで算出される誤差は 15.30738[%]と求められる。

図 4.18 では、系列 1 に得られたデータ、系列 2 に 60 ピクセルでの距離が正しいと仮定して 60 ピクセルでの距離から 300 ピクセルの距離を 1.35[m]とした 60 から 306 ピクセルまでの直線、系列 3 に 306 ピクセルでの距離が正しいと仮定して 60 ピクセルでの距離から 300 ピクセルの距離を 1.35[m]とした 60 から 306 ピクセルまでの直線を示した。誤差の範囲を $\pm 30\%$ として系列 2 と 3 の平均値と得られたデータの差が誤差以内だった場合は 247 個に対し 89 個と、精度は 36.0%となった。最大誤差 $1.35 \times 0.3 = 0.405$ から各値の誤差の絶対値を引いたデータを図 4.19 に示す。0 以上であれば範囲内となる。

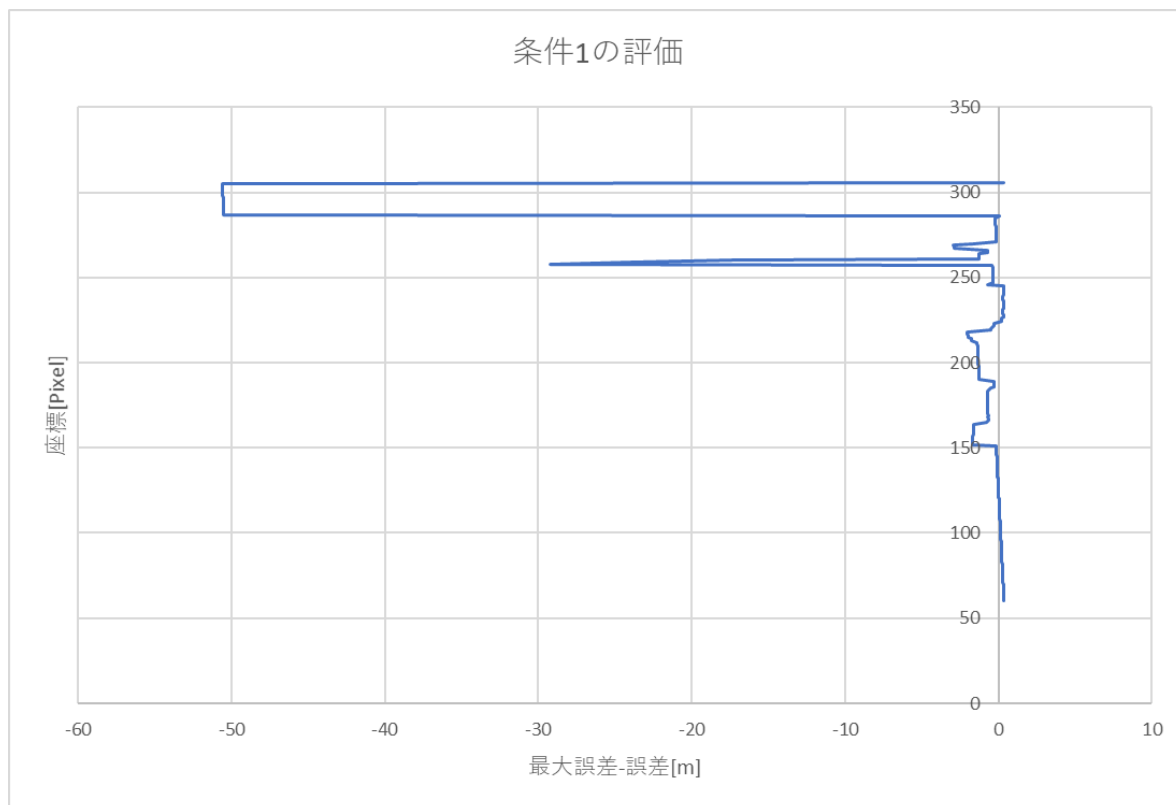
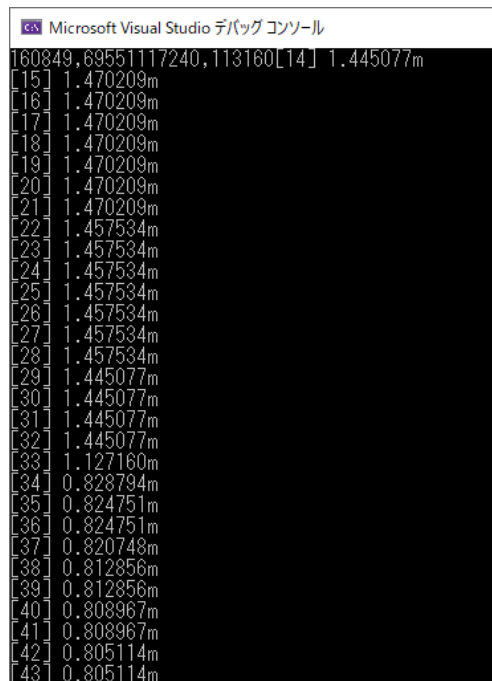


図 4.19 条件 1 の評価

以下に後者での条件の結果を示す。図 4.20~4.31 は奥行きのデータ、図 4.32、4.35 は左右の入力画像、図 4.33、4.36 は左右の補正後入力画像、図 4.34、4.37 は左右の中心線の処理後の画像を示す。

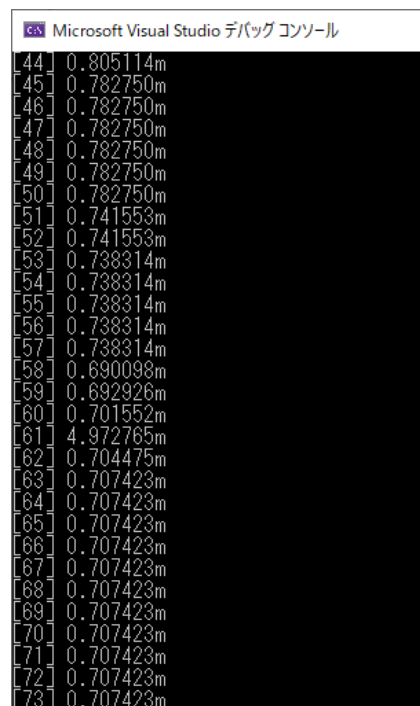


```

Microsoft Visual Studio デバッグ コンソール
160849.69551117240,113160[14] 1.445077m
[15] 1.470209m
[16] 1.470209m
[17] 1.470209m
[18] 1.470209m
[19] 1.470209m
[20] 1.470209m
[21] 1.470209m
[22] 1.457534m
[23] 1.457534m
[24] 1.457534m
[25] 1.457534m
[26] 1.457534m
[27] 1.457534m
[28] 1.457534m
[29] 1.445077m
[30] 1.445077m
[31] 1.445077m
[32] 1.445077m
[33] 1.127160m
[34] 0.828794m
[35] 0.824751m
[36] 0.824751m
[37] 0.820748m
[38] 0.812856m
[39] 0.812856m
[40] 0.808967m
[41] 0.808967m
[42] 0.805114m
[43] 0.805114m

```

図 4.20 実験結果 2-1



```

Microsoft Visual Studio デバッグ コンソール
[44] 0.805114m
[45] 0.782750m
[46] 0.782750m
[47] 0.782750m
[48] 0.782750m
[49] 0.782750m
[50] 0.782750m
[51] 0.741553m
[52] 0.741553m
[53] 0.738314m
[54] 0.738314m
[55] 0.738314m
[56] 0.738314m
[57] 0.738314m
[58] 0.690098m
[59] 0.692926m
[60] 0.701552m
[61] 4.972765m
[62] 0.704475m
[63] 0.707423m
[64] 0.707423m
[65] 0.707423m
[66] 0.707423m
[67] 0.707423m
[68] 0.707423m
[69] 0.707423m
[70] 0.707423m
[71] 0.707423m
[72] 0.707423m
[73] 0.707423m

```

図 4.21 実験結果 2-2

```
Microsoft Visual Studio デバッグ コンソール
[74] 0.707423m
[75] 0.707423m
[76] 0.707423m
[77] 0.707423m
[78] 0.707423m
[79] 0.707423m
[80] 0.710395m
[81] 0.710395m
[82] 0.710395m
[83] 0.692926m
[84] 0.692926m
[85] 0.692926m
[86] 0.692926m
[87] 0.692926m
[88] 0.692926m
[89] 0.713392m
[90] 0.713392m
[91] 0.713392m
[92] 0.713392m
[93] 0.713392m
[94] 0.722538m
[95] 0.722538m
[96] 0.722538m
[97] 0.722538m
[98] 0.722538m
[99] 0.722538m
[100] 0.722538m
[101] 0.722538m
[102] 0.722538m
[103] 0.722538m
```

図 4.22 実験結果 2-3

```
Microsoft Visual Studio デバッグ コンソール
[104] 0.722538m
[105] 0.722538m
[106] 0.722538m
[107] 0.722538m
[108] 0.722538m
[109] 0.722538m
[110] 0.722538m
[111] 0.722538m
[112] 0.722538m
[113] 0.722538m
[114] 0.722538m
[115] 0.722538m
[116] 0.722538m
[117] 0.722538m
[118] 0.722538m
[119] 0.722538m
[120] 0.722538m
[121] 0.722538m
[122] 0.722538m
[123] 0.722538m
[124] 0.722538m
[125] 0.722538m
[126] 0.722538m
[127] 0.722538m
[128] 0.722538m
[129] 0.722538m
[130] 0.722538m
[131] 0.722538m
[132] 0.722538m
[133] 1.798660m
```

図 4.23 実験結果 2-4

```
Microsoft Visual Studio デバッグ コンソール
[134] 1.798660m
[135] 1.818000m
[136] 1.818000m
[137] 1.818000m
[138] 1.818000m
[139] 1.818000m
[140] 1.818000m
[141] 1.595038m
[142] 1.595038m
[143] 1.595038m
[144] 1.595038m
[145] 1.595038m
[146] 1.595038m
[147] 1.595038m
[148] 1.595038m
[149] 1.595038m
[150] 1.595038m
[151] 1.595038m
[152] 1.595038m
[153] 1.595038m
[154] 1.595038m
[155] 1.595038m
[156] 1.595038m
[157] 1.595038m
[158] 7.351043m
[159] 7.351043m
[160] 7.351043m
[161] 7.351043m
[162] 7.351043m
[163] 7.351043m
```

図 4.24 実験結果 2-5

```
Microsoft Visual Studio デバッグ コンソール
[164] 7.351043m
[165] 7.351043m
[166] 7.351043m
[167] 7.351043m
[168] 7.351043m
[169] 7.351043m
[170] 7.351043m
[171] 7.351043m
[172] 7.351043m
[173] 7.351043m
[174] 7.351043m
[175] 7.351043m
[176] 7.351043m
[177] 7.351043m
[178] 7.351043m
[179] 7.351043m
[180] 7.351043m
[181] 7.351043m
[182] 7.351043m
[183] 7.351043m
[184] 7.351043m
[185] 7.351043m
[186] 7.351043m
[187] 7.351043m
[188] 7.351043m
[189] 7.351043m
[190] 7.351043m
[191] 7.351043m
[192] 7.351043m
[193] 7.351043m
```

図 4.25 実験結果 2-6

```
Microsoft Visual Studio デバッグ コンソール
[194] 7.351043m
[195] 7.351043m
[196] 7.351043m
[197] 7.351043m
[198] 7.351043m
[199] 7.351043m
[200] 7.351043m
[201] 7.351043m
[202] 7.351043m
[203] 7.351043m
[204] 7.351043m
[205] 7.351043m
[206] 7.351043m
[207] 7.351043m
[208] 7.351043m
[209] 7.351043m
[210] 7.351043m
[211] 7.351043m
[212] 7.351043m
[213] 7.351043m
[214] 7.351043m
[215] 7.351043m
[216] 7.351043m
[217] 7.351043m
[218] 7.351043m
[219] 7.351043m
[220] 7.351043m
[221] 7.351043m
[222] 7.351043m
[223] 7.351043m
```

図 4.26 実験結果 2-7

```
Microsoft Visual Studio デバッグ コンソール
[224] 7.351043m
[225] 16.907400m
[226] 16.907400m
[227] 16.907400m
[228] 16.907400m
[229] 16.907400m
[230] 16.907400m
[231] 0.573132m
[232] 0.575082m
[233] 0.575082m
[234] 1.943379m
[235] 1.943379m
[236] 1.943379m
[237] 1.943379m
[238] 1.943379m
[239] 1.943379m
[240] 1.943379m
[241] 1.943379m
[242] 1.943379m
[243] 1.965977m
[244] 1.965977m
[245] 1.965977m
[246] 1.965977m
[247] 1.965977m
[248] 4.830686m
[249] 4.830686m
[250] 4.830686m
[251] 4.830686m
[252] 4.830686m
[253] 4.830686m
```

図 4.27 実験結果 2-8

```
Microsoft Visual Studio デバッグ コンソール

[254] 4.830686m
[255] 4.830686m
[256] 4.830686m
[257] 4.449316m
[258] 4.449316m
[259] 4.449316m
[260] 4.449316m
[261] 4.449316m
[262] 4.449316m
[263] 4.449316m
[264] 4.449316m
[265] 4.449316m
[266] 4.449316m
[267] 4.449316m
[268] 4.449316m
[269] 4.449316m
[270] 1.625712m
[271] 1.625712m
[272] 1.565500m
[273] 1.565500m
[274] 1.565500m
[275] 1.565500m
[276] 1.483105m
[277] 1.470209m
[278] 1.457534m
[279] 1.457534m
[280] 1.457534m
[281] 1.457534m
[282] 1.457534m
[283] 1.457534m
```

図 4.28 実験結果 2-9

```
Microsoft Visual Studio デバッグ コンソール

[284] 1.457534m
[285] 1.457534m
[286] 1.457534m
[287] 1.457534m
[288] 1.457534m
[289] 1.056712m
[290] 1.056712m
[291] 1.056712m
[292] 1.056712m
[293] 1.056712m
[294] 1.056712m
[295] 1.056712m
[296] 1.056712m
[297] 1.056712m
[298] 1.056712m
[299] 1.056712m
[300] 1.056712m
[301] 1.056712m
[302] 1.050149m
[303] 1.050149m
[304] 1.050149m
[305] 1.050149m
[306] 1.050149m
[307] 1.050149m
[308] 1.050149m
[309] 1.050149m
[310] 1.050149m
[311] 1.050149m
[312] 1.050149m
[313] 1.050149m
```

図 4.29 実験結果 2-10

```
Microsoft Visual Studio デバッグ コンソール
[314] 1.050149m
[315] 1.050149m
[316] 1.050149m
[317] 1.050149m
[318] 1.050149m
[319] 1.050149m
[320] 1.050149m
[321] 1.050149m
[322] 1.050149m
[323] 1.050149m
[324] 1.050149m
[325] 1.050149m
[326] 1.050149m
[327] 1.050149m
[328] 1.056712m
[329] 1.056712m
[330] 1.056712m
[331] 1.056712m
[332] 1.056712m
[333] 1.056712m
[334] 1.056712m
[335] 1.056712m
[336] 1.056712m
[337] 1.056712m
[338] 1.012419m
[339] 1.012419m
[340] 1.012419m
[341] 1.012419m
[342] 1.012419m
[343] 1.300569m
```

図 4.30 実験結果 2-11

```
Microsoft Visual Studio デバッグ コンソール
[330] 1.056712m
[331] 1.056712m
[332] 1.056712m
[333] 1.056712m
[334] 1.056712m
[335] 1.056712m
[336] 1.056712m
[337] 1.056712m
[338] 1.012419m
[339] 1.012419m
[340] 1.012419m
[341] 1.012419m
[342] 1.012419m
[343] 1.300569m
[344] 1.300569m
[345] 1.300569m
[346] 1.300569m
[347] 1.300569m
[348] 1.300569m
[349] 1.300569m
[350] 1.300569m
[351] 1.300569m
[352] 1.300569m
[353] 1.300569m
[354] 1.300569m
[355] 1.300569m
[356] 1.300569m
[357] 1.300569m
[358] 1.300569m
[359] 1.300569m
```

図 4.31 実験結果 2-12



図 4.32 左カメラの入力画像 2



図 4.33 左カメラの補正後の入力画像 2

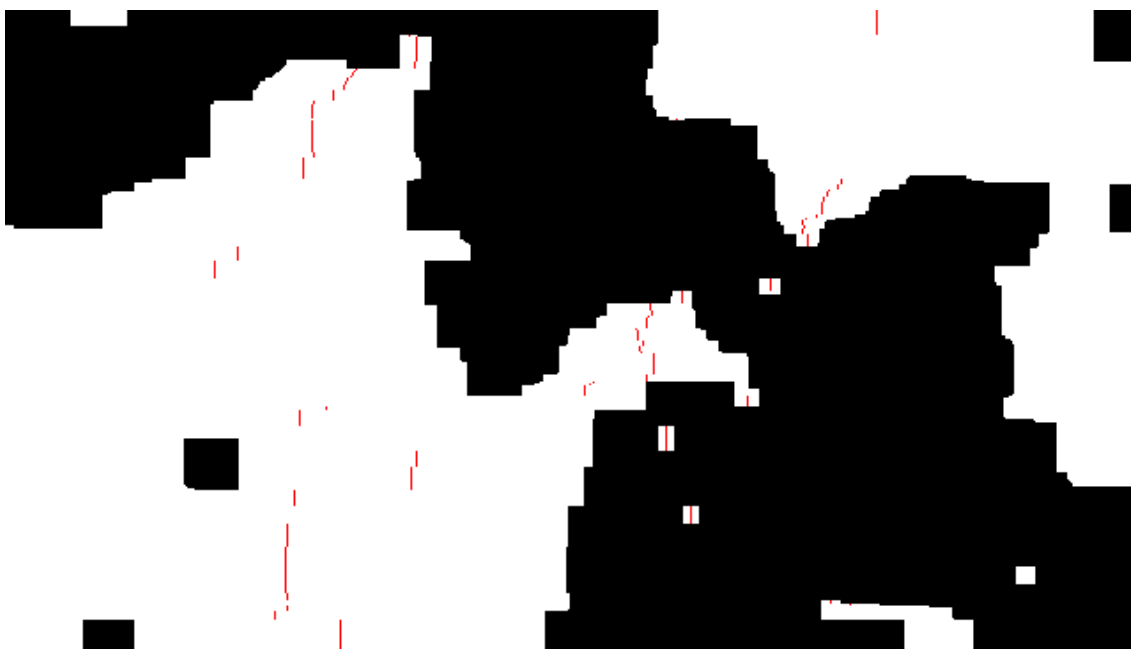


図 4.34 左カメラの中心線の処理後画像 2



図 4.35 右カメラの入力画像 2



図 4.36 右カメラの補正後の入力画像 2



図 4.37 右カメラの中心線の処理後画像 2

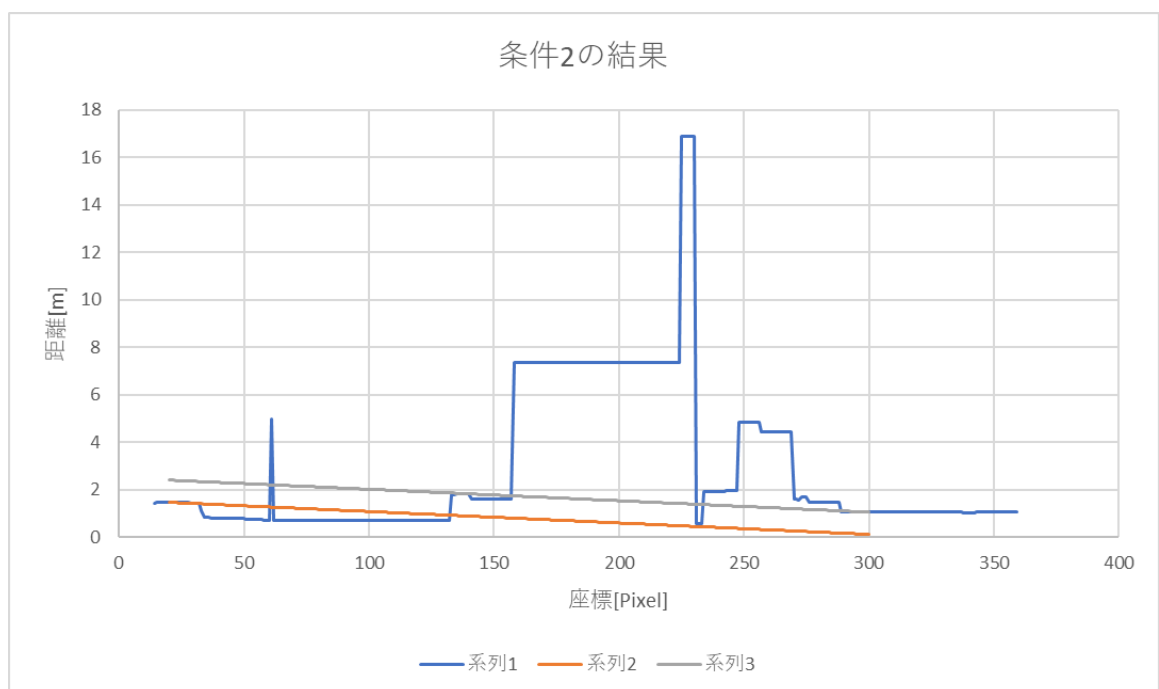


図 4.38 条件 2 の結果

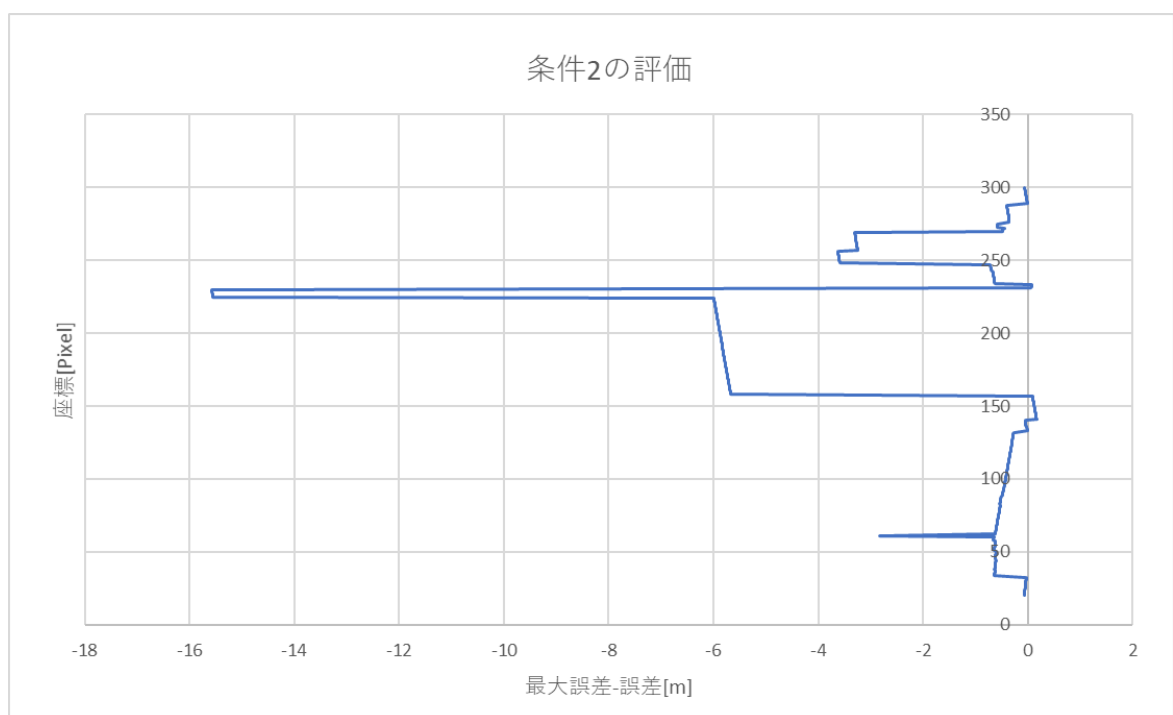


図 4.39 条件 2 の評価

図 4.33、4.36 から造花の一番奥の縦の座標値が 20 付近、一番手前の縦の座標値が 300 付近と考えられるので、図 4.15、4.24 から、20 ピクセル付近での距離から 300 ピクセル付近での距離を引いた値が増加の奥行きと等しくなると考えられる。20 ピクセルでの距離は 1.470209[m]、300 ピクセルでの距離は 1.056712[m]と分かり、60 ピクセルでの距離から 300 ピクセルでの距離を引いた値は 0.413497[m]と計算できる。これを造花の奥行きと比較すると、造花の奥行きは 1.35[m]なので、その差を増加の奥行きで割ることで算出される誤差は 69.37056[%]と求められる。

図 4.38 では、系列 1 に得られたデータ、系列 2 に 20 ピクセルでの距離が正しいと仮定して 20 ピクセルでの距離から 300 ピクセルの距離を 1.35[m]とした 20 から 300 ピクセルまでの直線、系列 3 に 300 ピクセルでの距離が正しいと仮定して 20 ピクセルでの距離から 300 ピクセルの距離を 1.35[m]とした 20 から 300 ピクセルまでの直線を示した。誤差の範囲を $\pm 30\%$ として系列 2 と 3 の平均値と得られたデータの差が誤差以内だった場合は 281 個に対し 20 個と、精度は 7.12%となった。最大誤差 $1.35 \times 0.3 = 0.405$ から各値の誤差の絶対値を引いたデータを図 4.39 に示す。0 以上であれば範囲内となる。

4.3 実験に対する考察

白い判別部分に各行に一つだけ中心線を引くことはできているが、白い床や黒いコートなどでは、緑の画素値が他の画素値より少しでも多い場合に検出されるほか、赤い花がノイズになるなど、造花の中心線の精度としてはいい結果が得られなかった。後者の条件のほうが白い領域が広がり、中心線の誤った検出が増えた。図 4.17、4.38 から、前者の条件では、造花の手前に大きい外れ値が集中し、後者の条件では、造花の中盤から手前にかけて大きい外れ値が多いが前者の条件よりはまんべんなく分布しているため、この差が精度の差につながったと考えられる。図 4.18、4.39 の 0 以上の値の分布からも同じ傾向がみられ、とくに後者ではいい精度が出ていないことがわかる。二値化のプロセスをもう少し工夫し、閾値を超えた場合黒い画素に変換するなど、ある程度の白い領域や黒い領域を検出しないようにすべきだった。また、モルフォロジー変換はノイズを除去するためのものだが、何度も行うと画像の原形を留めることができなくなることから、今回は変換回数が多く、いい結果が得られなかった可能性も考えられるため、適切な回数を見つけるか、ほかのノイズ除去方法を使用することも検討することも必要だと考えた。

第5章 結論

本研究では、雑草と農作物の判別をするシステムにステレオカメラを用い、農作物の畝に中心線を引く方法を使い、カメラと農作物の距離を三角測量などの理論から求めたが、農作物と雑草の判別は、ノイズが少ない場合中心線を正確に得やすかったが、ノイズが多い場合や大きい場合にはうまく得られなかったため、こういったものがノイズになりやすいかを考慮すべきだと感じた。また、実験のたびにカメラキャリブレーションを行うため、手間がかかるほか、キャリブレーションを行うたびにその精度が変わり、補正画像が変わってしまうことが今後の課題だと感じた。

参考文献

- [1] NEC 顔認識とは <https://jpn.nec.com/bimetrics/face/about.html>
- [2] 志磨健、的野春樹、掛川晋司、門司竜彦 ”自動車向け画像認識技術とステレオカメラによる距離計測” 光学 41 巻 5 号 289 ページ 2012
- [3] OpenCV <http://opencv.jp>
- [4] マクセルフロンティア 2 値画像の膨張収縮①
<https://www.frontier.maxell.co.jp/blog/posts/21.html>
- [5] マクセルフロンティア 2 値画像の膨張・収縮②
<https://www.frontier.maxell.co.jp/blog/posts/22.html>
- [6] 関西学院大学工学部知能・機械工学過程バーチャルリアリティ学研究室(井村研)
カメラキャリブレーション <https://imura-lab.org/lecture/eec//camera-calibration/>
- [7] CG-ARTS 協会 ”デジタル画像処理 “ CG-ARTS 協会 258、259 ページ

謝辞

初めに、指導教官としてご指導いただきました戸村先生に、心より感謝申し上げます。また、同じ研究室の薄田君、ナンダ君、細木君、宮元さんにも、合わせて感謝申し上げます。自身の力不足などで、研究が進まない時もありましたが皆さんの多大なご協力のおかげで無事に進めることができました。本当にありがとうございました。

付録 プログラムソース

```
#define _CRT_SECURE_NO_WARNINGS
#include <opencv2/opencv.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <iostream>
#include <string>
#include <sstream>
#ifdef _DEBUG
// Debug モード
#pragma comment(lib, "opencv_world460d.lib")
#else
// Release モード
#pragma comment(lib, "opencv_world460.lib")
#endif

void bin(cv::Mat input_image, cv::Mat aft_zasso_image, int s, int k) {
    int x, y, r, g, b, max, min, bxc;
    for (y = 0; y < input_image.rows; y++) {
        for (x = 0; x < input_image.cols; x++) {
            //画像読み込み
            r = input_image.ptr<cv::Vec3b>(y)[x][2];
            g = input_image.ptr<cv::Vec3b>(y)[x][1];
            b = input_image.ptr<cv::Vec3b>(y)[x][0];
            //最大値
            max = (r > g) ? r : g;
            max = (b > max) ? b : max;
            //左端-1 は黒仮定
            if (x == 0) {
                bxc = 0;
            }
            //緑化(緑を白,他黒化)
            if (g == max) {
                aft_zasso_image.ptr<cv::Vec3b>(y)[x][2] = 255;
```



```

        aft_zasso_image.ptr<cv::Vec3b>(y)[x][1] = 255;
        aft_zasso_image.ptr<cv::Vec3b>(y)[x][0] = 255; s++;
    }
    else {
        aft_zasso_image.ptr<cv::Vec3b>(y)[x][2] = 0;
        aft_zasso_image.ptr<cv::Vec3b>(y)[x][1] = 0;
        aft_zasso_image.ptr<cv::Vec3b>(y)[x][0] = 0; k++;
    }
}
}
printf("%d,%d", s, k); //k>,oc?co? s>,oc
return;
}

void mol(cv::Mat aft_zasso_image, int mo, int mc, int s, int k) {
    //if (s > k) {
        cv::morphologyEx(aft_zasso_image, aft_zasso_image, cv::MORPH_OPEN, cv::Mat(),
            cv::Point(-1, -1),
            mo,
            cv::BORDER_CONSTANT,
            cv::morphologyDefaultBorderValue());
        cv::morphologyEx(aft_zasso_image, aft_zasso_image, cv::MORPH_CLOSE, cv::Mat(),
            cv::Point(-1, -1),
            mc,
            cv::BORDER_CONSTANT,
            cv::morphologyDefaultBorderValue());/**/
    /* } else {
        cv::morphologyEx(aft_zasso_image, aft_zasso_image, cv::MORPH_CLOSE, cv::Mat(),
            cv::Point(-1, -1),
            mc,
            cv::BORDER_CONSTANT,
            cv::morphologyDefaultBorderValue());
        cv::morphologyEx(aft_zasso_image, aft_zasso_image, cv::MORPH_OPEN, cv::Mat(),
            cv::Point(-1, -1),
            mo,
            cv::BORDER_CONSTANT,
            cv::morphologyDefaultBorderValue());
    }*/
}

```

```

    return;
}

void cenlin(cv::Mat input_image, cv::Mat aft_zasso_image, int s, int k) {
    int x, y, r, g, b, max, min, bxc;
    int bxn = 0; int w = 0;
    int axn = 0; int bef = 0;
    int aef = 0;
    for (y = 0; y < aft_zasso_image.rows; y++) {
        int bef = 0; int w = 0;
        for (x = 0; x < aft_zasso_image.cols; x++) {
            //画像読み込み
            r = aft_zasso_image.ptr<cv::Vec3b>(y)[x][2];
            g = aft_zasso_image.ptr<cv::Vec3b>(y)[x][1];
            b = aft_zasso_image.ptr<cv::Vec3b>(y)[x][0];
            //最大値
            max = (r > g) ? r : g;
            max = (b > max) ? b : max;
            //左端-1 は黒仮定
            if (x == 0) {
                bxc = 0;
            }
            //緑化(緑を白,他黒化)
            if (g == 255) { //c=1(siro)
                w = 1;
                aft_zasso_image.ptr<cv::Vec3b>(y)[x][2] = 255;
                aft_zasso_image.ptr<cv::Vec3b>(y)[x][1] = 255;
                aft_zasso_image.ptr<cv::Vec3b>(y)[x][0] = 255;
                //bxc が黒だと白部分開始
                if (bxc == 0) {
                    bxc = 1;
                    bxn = x;
                }
                //右端が白のとき
                if (x == input_image.cols - 1) {
                    //最後なので白部分終了
                    axn = x;
                }
            }
        }
    }
}

```

```

//白部の中心線
/*aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][2] = 255;
aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][1] = 0;
aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][0] = 0; */
aef = (bxn + axn) / 2;
if (abs((aft_zasso_image.cols / 2) - bef) > abs((aft_zasso_image.cols / 2) -
aef)) {
    bef = aef;
}

//printf("[%d] %d,%d,%d\n", y, bxn, axn, abs((aft_zasso_image.cols / 2) -
aef));

}
}
else { //c=0(kuro)
    aft_zasso_image.ptr<cv::Vec3b>(y)[x][2] = 0;
    aft_zasso_image.ptr<cv::Vec3b>(y)[x][1] = 0;
    aft_zasso_image.ptr<cv::Vec3b>(y)[x][0] = 0;
    //bxc が白だと白部分終了
    if (bxc == 1) {
        bxc = 0;
        axn = x;
        //白部の中心線
        /*aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][2] = 255;
        aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][1] = 0;
        aft_zasso_image.ptr<cv::Vec3b>(y)[(bxn + axn) / 2][0] = 0; */
        aef = (bxn + axn) / 2;
        if (abs((aft_zasso_image.cols / 2) - bef) > abs((aft_zasso_image.cols / 2) -
aef)) {
            bef = aef;
        }

        //printf("[%d] %d,%d,%d\n", y, bxn, axn, abs((aft_zasso_image.cols / 2) -
aef));

    }
}
}

```

```

        bxc = 0; //axn = 0; (koreha,attemonakutemoOK)
    }
    if (x == (input_image.cols - 1) && w == 1) {

        aft_zasso_image.ptr<cv::Vec3b>(y)[bef][2] = 255;
        aft_zasso_image.ptr<cv::Vec3b>(y)[bef][1] = 0;
        aft_zasso_image.ptr<cv::Vec3b>(y)[bef][0] = 0; //printf("[%d] %d!¥n", y, bef);
    }
}

}return;
}

void dd(cv::Mat aft_zasso_image0, cv::Mat aft_zasso_image1, int s, int k,int bb,double f,int i) {
    int x, y, r, r1, b, g, g1, b1, max, min, bxc; FILE* fz; int c = 0;
    fz = fopen("z.csv", "w");
    if (fz==NULL) {
        printf("xx");
    }
    int bxn = 0; int bw = 0; int aw = 0;
    int axn = 0; int bef = 0;
    int d = 0;if(i==255) fprintf(fz, "画素,距離[m]¥n");
    //fprintf(fz, "");
    for (y = 0; y < aft_zasso_image0.rows; y++) {
        int aw = 0; int bw = 0;
        for (x = 0; x < aft_zasso_image0.cols; x++) {
            //画像読み込み
            r = aft_zasso_image0.ptr<cv::Vec3b>(y)[x][2];
            g = aft_zasso_image0.ptr<cv::Vec3b>(y)[x][1];
            b = aft_zasso_image0.ptr<cv::Vec3b>(y)[x][0];
            //最大値
            max = (r > g) ? r : g;
            max = (b > max) ? b : max;
            //左端-1 は黒仮定
            if (x == 0) {
                bxc = 0;
            }
            //緑化(緑を白,他黒化)

```

```

if (r == 255 && g==0) { //c=1(siro)
    bw = 1;
    bxn = x;

}

r1 = aft_zasso_image1.ptr<cv::Vec3b>(y)[x][2];
g1 = aft_zasso_image1.ptr<cv::Vec3b>(y)[x][1];
b1 = aft_zasso_image1.ptr<cv::Vec3b>(y)[x][0];
//最大値
max = (r1 > g1) ? r1 : g1;
max = (b1 > max) ? b1 : max;
//左端-1 は黒仮定
if (x == 0) {
    bxc = 0;
}
//緑化(緑を白,他黒化)
if (r1 == 255 && g1 == 0) { //c=1(siro)
    aw = 1;
    axn = x;

}

if (aw==1&&bw==1) {
    if (bxn - axn > 0 && bxn - axn != 0) {
        d = bxn - axn;

    }
    bxc = 0; //axn = 0; (koreha,attemonakutemoOK)if (c % 30 == 0)
cv::waitKey(13000);
}if (i == 255) {

    c++;

    fprintf(fz, "%d,%lf¥n",y, (double)( bb * f / d) / 1000);
    //fprintf(fz, "");
    //printf("[%d] %d¥n", y, d);    if (c % 30 == 0)cv::waitKey(15000);

```

```

        }
    }
    if(d>0&&d!=0)if(i==255) printf("[%d] %lfm¥n", y, (double)( bb * f / d)/1000);
}fclose(fz);
return;
}

```

// Defining the dimensions of checkerboard

```
int CHECKERBOARD[2]{ 10,7 };
```

```
int main()
```

```
{
```

```
    // Creating vector to store vectors of 3D points for each checkerboard image
```

```
    std::vector<std::vector<cv::Point3f> > objpoints;
```

```
    // Creating vector to store vectors of 2D points for each checkerboard image
```

```
    std::vector<std::vector<cv::Point2f> > imgpoints0,imgpoints1;
```

```
    // Defining the world coordinates for 3D pointsstatic
```

```
    std::vector<cv::Point3f> objp,objp1;
```

```
    cv::Mat cameraMatrix0, cameraMatrix1, distCoeffs0, distCoeffs1, R, R0, R1, P1, P0, T, E, F, Q;
```

```
    cv::Mat mapx0, mapy0, mapx1, mapy1, err;//_3dImage,(2000, 2000, CV_16SC2)(2000, 2000,
CV_16UC1)(2000, 2000, CV_16SC2)(2000, 2000, CV_16UC1)
```

```
    cv::Mat dis;
```

```
    for (int i{ 0 }; i < CHECKERBOARD[1]; i++)
```

```
    {
```

```
        for (int j{ 0 }; j < CHECKERBOARD[0]; j++)
```

```
            objp.push_back(cv::Point3f(j, i, 0));
```

```
    }
```

```
    for (int i{ 0 }; i < CHECKERBOARD[1]; i++)
```

```
    {
```

```
        for (int j{ 0 }; j < CHECKERBOARD[0]; j++)
```

```

        objp1.push_back(cv::Point3f(j, i, 0));
    }

    int capt = 0; double b;
    int numdis, brocksize;
    printf("カメラ間距離[mm]¥n");
    scanf("%lf",&b);
    /*printf("視差の検索範囲。各ピクセル アルゴリズムは、0 (デフォルトの最小視差) から
    numDisparities までの最適な視差を見つけます。次に、最小視差を変更することで検索範囲
    をシフトできます。¥n16 倍されます!");
    scanf("%d", &numdis);
    printf("アルゴリズムによって比較されたブロックの線形サイズ。サイズは奇数にする必
    要があります (ブロックは現在のピクセルを中心に行っているため)。ブロック サイズが大き
    いほど、視差マップの精度は低くなりますが、滑らかになります。ブロック サイズが小さ
    いほど、より詳細な視差マップが得られますが、アルゴリズムが間違っただ対応を検出する可
    能性が高くなります。");
    scanf("%d", &brocksize);*/
    cv::VideoCapture camera0,camera1;// 入力画像 static

    cv::Mat input_image0, input_image1, i1, i0,dis2;
    cv::Mat i000, i111, i00, i11,disparity, _3dImage;

    camera0.open(0, cv::CAP_ANY);// 動画ファイルを開く (0)だと USB カメラが開くよ
    if (!camera0.isOpened()) {
        return -1;// 動画ファイルが開けなかったときは終了する
    }camera1.open(1, cv::CAP_ANY);// 動画ファイルを開く (0)だと USB カメラが開くよ
    if (!camera1.isOpened()) {
        return -1;// 動画ファイルが開けなかったときは終了する
    }
    camera0.set(cv::CAP_PROP_FRAME_WIDTH,640);
    camera0.set(cv::CAP_PROP_FRAME_HEIGHT, 360);
    camera1.set(cv::CAP_PROP_FRAME_WIDTH, 640);
    camera1.set(cv::CAP_PROP_FRAME_HEIGHT, 360);
    while (capt < 20) {cv::waitKey(1) != 'q' i++; k = 0; s = 0;
        camera0.read(input_image0);
        camera1.read(input_image1);
        std::string c="a"+std::to_string(capt)+".png";

```

```

std::string cc = "b" + std::to_string(capt) + ".png";
cv::namedWindow("cam0 frame");
cv::namedWindow("cam1 frame");
while (cv::waitKey(1) != 'q') {
    camera0.read(input_image0); //camera >> input_image; でも OK
    cv::imshow("cam0 frame", input_image0);
    camera1.read(input_image1); //camera >> input_image; でも OK
    cv::imshow("cam1 frame", input_image1);
}
//c << "b" << capt << ".png";
//printf("%sn", c.str().c_str());
printf("time=%d", capt);
//cv::resize(input_image0, input_image0, cv::Size(), static_cast<double>(640) / input_image0.cols,
static_cast<double>(360) / input_image0.rows);
//cv::resize(input_image1, input_image1, cv::Size(), static_cast<double>(640) / input_image1.cols,
static_cast<double>(360) / input_image1.rows);
cv::imwrite(c, input_image0);
cv::imwrite(cc, input_image1);
capt++;
}
cv::destroyAllWindows();
// Extracting path of individual image stored in a given directory
std::vector<cv::String> images0, images1;
// Path of the folder containing checkerboard images
std::string path0 = "a*.png";
std::string path1 = "b*.png";

cv::glob(path0, images0);
cv::glob(path1, images1);

cv::Mat frame0, frame1, gray0, gray1;
// vector to store the pixel coordinates of detected checker board corners
std::vector<cv::Point2f> corner_pts0, corner_pts1;
bool success0, success1;

// Looping over all the images in the directory
for (int i{ 0 }; i < images0.size(); i++)

```



```

{
    frame0 = cv::imread(images0[i]);
    cv::cvtColor(frame0, gray0, cv::COLOR_BGR2GRAY);
    frame1 = cv::imread(images1[i]);
    cv::cvtColor(frame1, gray1, cv::COLOR_BGR2GRAY);
    // Finding checker board corners
    // If desired number of corners are found in the image then success = true
    success0 = cv::findChessboardCorners(gray0, cv::Size(CHECKERBOARD[0],
CHECKERBOARD[1]), corner_pts0, cv::CALIB_CB_ADAPTIVE_THRESH |
cv::CALIB_CB_FAST_CHECK | cv::CALIB_CB_NORMALIZE_IMAGE);
    success1 = cv::findChessboardCorners(gray1, cv::Size(CHECKERBOARD[0],
CHECKERBOARD[1]), corner_pts1, cv::CALIB_CB_ADAPTIVE_THRESH |
cv::CALIB_CB_FAST_CHECK | cv::CALIB_CB_NORMALIZE_IMAGE);
    /*
    * If desired number of corner are detected,
    * we refine the pixel coordinates and display
    * them on the images of checker board
    */
    if (success0 && success1)
    {
        cv::TermCriteria criteria(cv::TermCriteria::EPS | cv::TermCriteria::MAX_ITER, 30,
0.001);

        // refining pixel coordinates for given 2d points.
        cv::cornerSubPix(gray0, corner_pts0, cv::Size(11, 11), cv::Size(-1, -1), criteria);
        cv::cornerSubPix(gray1, corner_pts1, cv::Size(11, 11), cv::Size(-1, -1), criteria);
        // Displaying the detected corner points on the checker board
        cv::drawChessboardCorners(frame0, cv::Size(CHECKERBOARD[0],
CHECKERBOARD[1]), corner_pts0, success0);
        cv::drawChessboardCorners(frame1, cv::Size(CHECKERBOARD[0],
CHECKERBOARD[1]), corner_pts1, success1);
        objpoints.push_back(objp);
        imgpoints0.push_back(corner_pts0);
        imgpoints1.push_back(corner_pts1);
    }

    //cv::imshow("Image", frame);
    cv::waitKey(0);
}

```

```

    }

    cv::destroyAllWindows();

    int mo=0, mc = 0,s = 0, k = 0, s1 = 0, k1 = 0,i=0;

    /*
    * Performing camera calibration by
    * passing the value of known 3D points (objpoints)
    * and corresponding pixel coordinates of the
    * detected corners (imgpoints)
    */
    //cv::calibrateCamera(objpoints, imgpoints0, cv::Size(gray0.rows, gray0.cols), cameraMatrix0,
    distCoeffs0, R, T);
    // cv::calibrateCamera(objpoints, imgpoints1, cv::Size(gray1.rows, gray1.cols), cameraMatrix1,
    distCoeffs1, R, T);
    /*for (size_t i = 0; i < imgpoints.size(); ++i) {
        std::cout << imgpoints[i] << " ";
    }*/
    cv::TermCriteria criteria(cv::TermCriteria::EPS | cv::TermCriteria::MAX_ITER, 30, 0.001);int
    axp,ayp, width, height, fourcc; // 作成する動画ファイルの設定
    cv::stereoCalibrate(objpoints,imgpoints0,imgpoints1, cameraMatrix0,distCoeffs0,
    cameraMatrix1,distCoeffs1, cv::Size(input_image0.cols, input_image0.rows), R,T,E,F,0, criteria);
    std::cout << "cameraMatrix : " << cameraMatrix0 << std::endl;
    std::cout << "cameraMatrix : " << cameraMatrix1 << std::endl;
    std::cout << "distCoeffs : " << distCoeffs0 << std::endl;
    std::cout << "distCoeffs : " << distCoeffs1 << std::endl;
    std::cout << "Rotation vector : " << R << std::endl;
    std::cout << "Translation vector : " << T << std::endl;
    printf("%lf", cameraMatrix0.at<double>(0, 0));
    printf("%lf", cameraMatrix1.at<double>(0, 0));
    //

    std::string filepathl = "left_video.mp4";
    std::string filepathr = "right_video.mp4";
    camera0.open(filepathl);// 00 動画ファイルを開く (0)だと USB カメラが開くよ

```

```

if (!camera0.isOpened()) {
    return -1; // 動画ファイルが開けなかったときは終了する
} camera1.open(filepath); // 00 動画ファイルを開く (0)だと USB カメラが開くよ
if (!camera1.isOpened()) {
    return -1; // 動画ファイルが開けなかったときは終了する
}
cv::VideoWriter writer; // 動画ファイルを書き出すためのオブジェクトを宣言する

fourcc = cv::VideoWriter::fourcc('m', 'p', '4', 'v'); // ビデオフォーマットの指定( ISO MPEG-
4 / .mp4)
double fps;
//camera.set(cv::CAP_PROP_FPS, 17);
width = (int)camera0.get(cv::CAP_PROP_FRAME_WIDTH); // フレーム横幅を取得
height = (int)camera0.get(cv::CAP_PROP_FRAME_HEIGHT);
fps = camera0.get(cv::CAP_PROP_FPS); // フレームレートを取得

printf("クロージング回数¥n"); //1?
scanf("%d", &mc);
printf("オープニング回数¥n"); //3?
scanf("%d", &mo);
printf("サイズ変更? yes->1 を入力¥n");
scanf("%d", &i);
if (i == 1) {
    printf("横ピクセル設定¥n"); //640
    scanf("%d", &axp);
    printf("縦ピクセル設定¥n"); //360
    scanf("%d", &ayp);
}
else { axp = width; ayp = height; }
//cv::resize(i0, i0, cv::Size(), static_cast<double>(640) / i0.cols, static_cast<double>(340) /
i0.rows);
//cv::resize(i1, i1, cv::Size(), static_cast<double>(640) / i1.cols, static_cast<double>(340) /
i1.rows);
cv::stereoRectify(cameraMatrix0, distCoeffs0, cameraMatrix1, distCoeffs1,
cv::Size(input_image0.cols, input_image0.rows) ,R, T,R0,R1,P0,P1,Q, 1,0, cv::Size(axp,

```

```

    ayp));//nput_image nput_image
        cv::initUndistortRectifyMap(cameraMatrix0,          distCoeffs0,R0,P0,          cv::Size(afxp,
    ayp),CV_32FC1,mapx0,mapy0);//nput_image nput_image
        cv::initUndistortRectifyMap(cameraMatrix1, distCoeffs1, R1, P1, cv::Size(afxp,ayp), CV_32FC1,
    mapx1, mapy1);//nput_image nput_imagei nput_image0.rows, input_image0.cols
    printf("%0lf", P1.at<double>(0, 0));

    /**/std::cout << "cameraMatrix : " << R0 << std::endl;
    std::cout << "cameraMatrix : " << R1 << std::endl;
    std::cout << "Rotation vector : " << P0 << std::endl;
    std::cout << "Rotation vector : " << P1 << std::endl;

    printf(" 焦点距離 :%0lf[Pixel] ¥n 主点 c_x:%0lf[Pixel] ¥n 主点 c_y:%0lf[Pixel]",
    P1.at<double>(0, 0), P1.at<double>(0, 2), P1.at<double>(1, 2));
    cv::waitKey(7000);

    writer.open("CloneVideo.mp4", fourcc, fps, cv::Size(afxp * width / width, ayp * height / height));
    cv::Mat aft_zasso_image0(cv::Size(afxp * width / width, ayp * height / height), CV_8UC3,
    cv::Scalar(255, 255, 255));
    cv::Mat aft_zasso_image1(cv::Size(afxp * width / width, ayp * height / height), CV_8UC3,
    cv::Scalar(255, 255, 255));
    /**/* /* */
    i = 0;
    while (1) {
        i++;
        camera0 >> i0; //camera.read(input_image); でも OK
        camera1 >> i1;
        //printf("%0d,%0d", input_image.cols, input_image.rows);      ピクセルチェック用
        if (i0.empty() == true|| i1.empty() == true) {
            break;// 画像が読み込めなかったとき、無限ループを抜ける
        }
        //i0 = cv::imread("a.png");
        //i1 = cv::imread("b.png");
        cv::resize(i0, i0, cv::Size(), static_cast<double>(afxp) / i0.cols, static_cast<double>(ayp) /
    i0.rows);
        cv::resize(i1, i1, cv::Size(), static_cast<double>(afxp) / i1.cols, static_cast<double>(ayp) /
    i1.rows);

```

```

cv::remap(i0, i00, mapx0, mapy0, cv::INTER_LINEAR);
cv::remap(i1, i11, mapx1, mapy1, cv::INTER_LINEAR);

/* cv::namedWindow("cam0 frame");
   cv::namedWindow("cam1 frame");
   while(cv::waitKey(1) != 'q'){
cv::imshow("cam0 frame", i00);
cv::imshow("cam1 frame", i11);

}

/*

   cv::cvtColor(i00, i000, cv::COLOR_BGR2GRAY);
   cv::cvtColor(i11, i111, cv::COLOR_BGR2GRAY);

   cv::Ptr<cv::StereoBM> sbm = cv::StereoBM::create( 16 *numdis, brocksize);
   sbm->compute(i000, i111, dis);
   //std::cout << "cameraMatrix : " << input_image0.type() << std::endl;
   printf("%hd", dis.at<short>(0, 0));
   printf("%d", (int)dis.at<short>(i0.rows-1, i0.cols-1));
// cv::normalize(dis, dis2, 0, 255, cv::NORM_MINMAX, CV_8U);
//printf("%d", dis.at<cv::Vec2b>(0, 0));
//cv::namedWindow("aft frame");cv::imshow("aft frame", dis2);
//writer << dis2;
//cv::reprojectImageTo3D(disparity, _3dImage, Q,false);
   */ //while (cv::waitKey(1) != 'q') {
       cv::imshow("c0 frame", i0);
       cv::imshow("c1 frame", i1);
       cv::imshow("ca0 frame", i00);
       cv::imshow("ca1 frame", i11);
//}

   // cv::imshow("cam1 frame", dis2);}
   bin(i00, aft_zasso_image0, s, k);
   bin(i11, aft_zasso_image1, s1, k1);
   mol(aft_zasso_image0, mo, mc, s, k);
   mol(aft_zasso_image1, mo, mc, s1, k1);

```

```

        cenlin(i00, aft_zasso_image0, s, k);
        cenlin(i11, aft_zasso_image1, s1, k1);
//   while (cv::waitKey(1) != 'q') {
        cv::imshow("ce0 frame", aft_zasso_image0);
        cv::imshow("ce1 frame", aft_zasso_image1);
// }
        dd( aft_zasso_image0, aft_zasso_image1, s, k, b, P1.at<double>(0, 0), i);
/*cv::resize(input_image0, input_image0, cv::Size(), static_cast<double>(640) /
input_image0.cols, static_cast<double>(480) / input_image0.rows);
        cv::resize(dis2, dis2, cv::Size(), static_cast<double>(640) / input_image1.cols,
static_cast<double>(480) / input_image1.rows);*/
        int x, y;
        /*y=2cv::waitKey(1000);*///nput_image nput_image
        /*for (y = 0; y < i0.rows; y++) {
            for (x = 0; x < i0.cols; x++) {
                //画像読み込み
                if ((int)dis.at<short>(y, x) != -16 && (int)dis.at<short>(y, x) != 0) {
                    printf("[%d][%d]", x, y);
                    printf("%d,", (16 * ((int)b * (input_image0.cols - x)) / (int)dis.at<short>(y,
x))); //
                    printf("%d,", (16 * ((int)b * (input_image0.rows - y)) / (int)dis.at<short>(y,
x))); //
                    printf("%d ", (16 * (int)b * ((int)P1.at<double>(0, 0))) / ((int)dis.at<short>(y,
x))); //
                    printf("\n");
                }
            }
        }
    }*/// //
    if (cv::waitKey(1) == 'q') break; printf("%f 秒\n", i / fps); printf("%f\n", fps);
    if(i==255){
        cv::imwrite("la.png", i0);
        cv::imwrite("ra.png", i1);
        cv::imwrite("lb.png", i00);
        cv::imwrite("rb.png", i11);

        cv::imwrite("lc.png", aft_zasso_image0);
        cv::imwrite("rc.png", aft_zasso_image1);

```

```
    }  
    }cv::destroyAllWindows();  
    return 0;  
}
```