情報理工学特別研究報告書

題目

フリマアプリなどに用いる前景抽出処理に関する研究

学生証番号　054812

氏名　矢田部 陸

提出日　令和 6 年 1 月 26 日

指導教員　蚊野　浩

京都産業大学
情報理工学部

要約

フリマアプリに商品を出品する場合,商品が魅力的に映るように，出品物だけを抽出し，それ以外を背景部分として白く塗りつぶすことがある.このような画像処理を前景抽出処理と呼ぶ.凹凸係数を用いた手法やGrabCut，マジックワンドなど,従来の前景抽出処理は画素値の勾配や閾値により自動的に前景を抽出するため,処理の結果が物体の色や影に大きく左右される.物体が背景や影と同色である場合,画素値の差が極端に小さくなるため,画素値の勾配や閾値による検出は困難となる.しかし,ユーザーが抽出する輪郭を手作業で選択する Intelligent Scissors であれば,画素値の差が小さい輪郭であっても抽出することができるため,物体の色や影に左右されずに幅広い物体を柔軟に処理することができた.しかし,形状が複雑な物体や,輪郭が薄い物体の輪郭を正確に選択するには,高度な操作を求められる.そのため，凹凸係数や GrabCut 等で処理を行い,抽出することができなかった輪郭にのみ,Intelligent Scissors を使用するべきだと考えた.

目次

1章 序論

メルカリのようなフリマアプリに自分の商品を出品する場合,その商品ができるだけ魅力的に写っている写真を使うのが良い.例えば,机の上に載せたカバンの写真を出品に使う時,光の写り加減などにより，机の一部や出品物以外のものが写ることは望ましくない.そこで,出品物だけを画像から抽出し,それ以外の領域を白く塗りつぶすような処理を行うことがある.このような処理のことを画像処理では前景抽出処理と呼ぶ[1].ポイントサービス「Ponta（ポンタ)」を運営する株式会社ロイヤリティ マーケティングの調査によると,自社の会員 25,000 人の内,44%の人がフリマアプリを利用していることが確認された.そのため,先ほど述べた前景抽出処理の需要は高まっていると考えられる.そこで,本論文ではこの前景抽出処理について研究を行い,フリマアプリで利用するのに適した処理について研究した.

2 章　様々な前景抽出処理のアルゴリズム

2.1 Intelligent Scissors

Intelligent Scissors[2]は,画像中の物体の輪郭を,ユーザーの操作で対話的に抽出する処理である.ユーザーが物体の抽出したい輪郭の一部をクリックすることで,その位置を始点とし,ドラッグを開始すると,ドラッグした位置までの輪郭を描画する.図 2.1 は,実際に描画した画像である.赤色の円はユーザーがクリックした始点を表している.本研究で用いた Intelligent Scissors のプログラムは,OpenCV にある,Intelligent Scissors のための関数を用いて, 処理の実装を行なった.

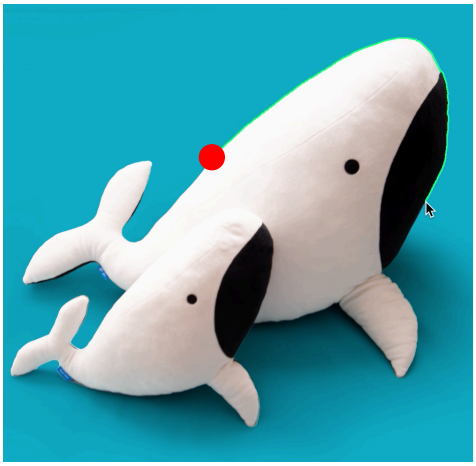

図 2.1 Intelligent Scissors で輪郭を描画した画像

2.2 ローカルコスト

ローカルコストは Intelligent Scissors で輪郭を抽出するために用いる値である.Intelligent Scissors は画像上で選択した 2 点間を通る経路の中で,前景と背景の境界となる可能性が最も高いエッジを検出する処理である.ローカルコストは,各ピクセルが前景と背景の境界となる可能性の低さを表す値である.Intelligent Scissors は,ローカルコストが低い値を取るピクセルを通るようなエッジを描画する.ローカルコストは,「エッジの有無に関する評価値」,「勾配強度に関する評価値」,「勾配方向からのズレに関する評価値」の三つの要素から境界としての評価値を計算する.式(1)のように計算し,それぞれの要素の値は 0～1 の範囲にスケーリングする.評価が高いほど低い値をとり,評価が低いほど高い値を取る.

$$l(\boldsymbol{p},\boldsymbol{q}) = \omega_Z \cdot f_Z(\boldsymbol{q}) + \omega_D \cdot f_D(\boldsymbol{p},\boldsymbol{q}) + \omega_G \cdot f_G(\boldsymbol{q}) \quad \cdots(1)$$

f_Z :エッジの有無に関する評価値
f_G :勾配強度に関する評価値
f_D :勾配方向に関する評価値

ω はそれぞれの要素に対する重みである[2]. $\omega_Z = 0.43$, $\omega_D = 0.43$, $\omega_G = 0.14$ の重みが,幅広い種類の画像に対し,正しく機能するとされている.

次に,それぞれの要素と計算方法について説明する.エッジの有無はラプラシアンフィタを用いて計算する.ラプラシアンフィルタの結果がゼロになる部分は,画素値の動きが正から負,または負から正に変化する部分を表しているため,境界としての評価が高くなる.そのため,ゼロの部分をエッジが存在するとし,式(2)のように計算する.ここで$I_L(\boldsymbol{q})$は画素 $\boldsymbol{q}$ のラプラシアンである.

$$f_Z(\boldsymbol{q}) = \begin{cases} 0; & \text{if } I_L(\boldsymbol{q}) = 0 \\ 1; & \text{if } I_L(\boldsymbol{q}) \neq 0 \end{cases} \quad \cdots(2)$$

勾配強度は各ピクセルにおけるエッジの強さを表しており, 式(3)で示す通り,X,Y 方向の偏微分の二乗和の平方根で求める.X,Y 方向の偏微分は X 方向と Y 方向の画素値の変化を表している.式(4)のように,各ピクセルの勾配強度 G を画像内の最大勾配強度 max(G)で正規化することで, 0〜1 の範囲にスケーリングする.

$$G = \sqrt{I_x^2 + I_y^2}. \quad \cdots(3)$$

$$f_G = \frac{\max(G) - G}{\max(G)} = 1 - \frac{G}{\max(G)} \quad \cdots(4)$$

勾配方向は各ピクセルにおける画素値が最も変化している方向を表している.この方向と垂直な方向は画素値の変化が最も少なくなる.そのため,勾配方向に垂直なエッジは,画素値の変化の境界となる可能性が高い.ローカルコストにおける「勾配方向のズレ」

は,描画する線が勾配方向に垂直であるほど,境界としての評価が高くなり,低いコストをとる.勾配方向は X 方向と Y 方向の偏微分を用いて計算することができ,勾配方向に垂直なベクトルを表す式は式(5)のようになる.

$$\boldsymbol{D'}(\boldsymbol{p}) = [I_y(\boldsymbol{p}), -I_x(\boldsymbol{p})] \quad \cdots(5)$$

式(6)は,Intelligent Scissors によって,線が描画されるピクセル間の方向を計算する式である.L(p,q)は,ピクセル p とピクセル q の間の方向を表している. p から q へのベクトルが D’(p)と逆方向を向いていない場合は,p から q のベクトルが p と q の間の方向を表すベクトルとなり,D’(p)と逆方向であった場合,q から p のベクトルが p と q の間の方向を表すベクトルとなる.

$$\boldsymbol{L}(\boldsymbol{p},\boldsymbol{q}) = \begin{cases} \boldsymbol{q}-\boldsymbol{p}; & \text{if } \boldsymbol{D'}(\boldsymbol{p})\cdot(\boldsymbol{q}-\boldsymbol{p}) \geq 0 \\ \boldsymbol{p}-\boldsymbol{q}; & \text{if } \boldsymbol{D'}(\boldsymbol{p})\cdot(\boldsymbol{q}-\boldsymbol{p}) < 0 \end{cases} \quad \cdots(6)$$

式(7)は勾配方向に垂直なベクトルとピクセル間のベクトルの内積を計算しており,二つのベクトルの角度のズレを表す.値は 0～1 の範囲をとる.垂直な場合は 1 となり,平行な場合は 0 をとる.$d_p(p,q)$は,p の勾配方向に垂直なベクトルからのズレを表し,$d_q(p,q)$は,q の勾配方向に垂直なベクトルからのズレを表している.

$$\begin{aligned} d_{\boldsymbol{p}}(p,q) &= \boldsymbol{D'}(\boldsymbol{p})\cdot\boldsymbol{L}(\boldsymbol{p},\boldsymbol{q}) \\ d_{\boldsymbol{q}}(p,q) &= \boldsymbol{L}(\boldsymbol{p},\boldsymbol{q})\cdot\boldsymbol{D'}(\boldsymbol{q}) \end{aligned} \quad \cdots(7)$$

ベクトルの内積の逆余弦関数は,ベクトル間のなす角を表す.そのため,式(8)は,ピクセル間のベクトルと,p と q の勾配方向に垂直なベクトルからのなす角の合計となる.値は 0～πの範囲をとる.この値をπで除算することで,0～1 の範囲にスケーリングする.そのため,ローカルコストの「勾配方向からのズレ」は pq 間で描画した線と,p と q の勾配方向に垂直なベクトルとの,角度のズレの合計を表している.

$$f_D(\boldsymbol{p},\boldsymbol{q}) = \frac{1}{\pi}\{\cos[d_{\boldsymbol{p}}(p,q)]^{-1} + \cos[d_{\boldsymbol{q}}(p,q)]^{-1}\} \quad \cdots(8)$$

2.3 最少累積コストマップ

最短経路の計算には章 2.2 で計算した,各ピクセルのローカルコストを保存したローカルコストマップを用いる.図 2.2 はローカルコストマップの一例を表しており,○で囲まれている部分はユーザーが選択したピクセルである.

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	(2)	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

図 2.2 ローカルコストマップ

選択したピクセルを始点とし,各ピクセルまでのコストの合計が最小となる経路を求める.各ピクセルまでの最小コストの経路を表したものを最小累積コストマップという.ローカルコストは境界としての評価が高いほど値が小さくなるため,最小コストの経路は,そのピクセルまでの評価が最も高い経路となる.図 2.3,図 2.4,図 2.5 は最小累積コストマップを生成する流れを表している.図 2.3 は始点と隣接するピクセルのコストの合計値を表している.図 2.4,図 2.5 では少しずつ範囲を広げていき,赤い円で囲まれている部分に隣接するピクセルまでの最小コストの経路を求める.範囲を広げていく中でよりコストの合計が低くなる経路を見つけた場合はそちらに変更する.

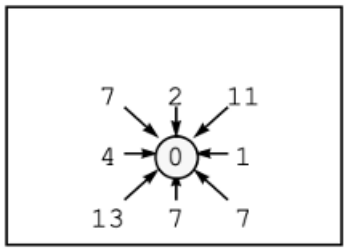

図 2.3 最小累積コストマップの生成手順 1

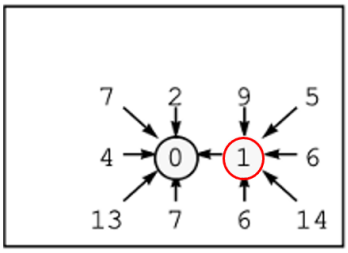

図 2.4 最小累積コストマップの生成手順 2

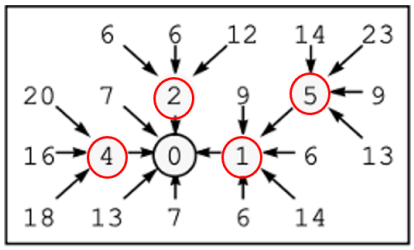

図 2.5 最小累積コストマップの生成手順 3

全てのピクセルにおける, 最小コスト経路を求めることで, 最小累積コストマップが完成する. 図 2.6 は, この最小累積コストマップを用いて, 選択したピクセルまでの経路を抽出する様子を表している. 画像の座標(10, 8)を終点として選択したことで, 最小累積コストマップの同座標に設定されている, コスト合計が 39 の座標までの経路を抽出する.

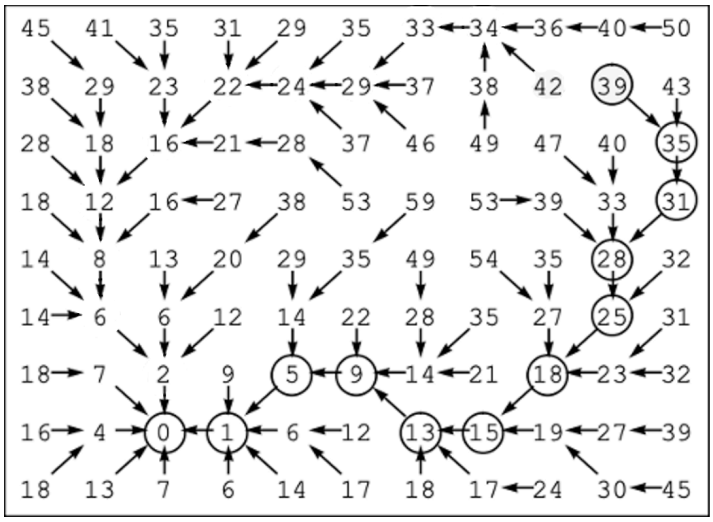

図 2.6 最小経路の抽出

2.4 Intelligent Scissors による前景抽出

Intelligent Scissors はユーザーが手動で物体の輪郭を選択することで, 輪郭を自動的に抽出する前景抽出処理である. ユーザーがクリックで始点を選択し, ローカルコストマップと最小累積コストマップを作成する. マウスをドラッグすることで, カーソル位置にあるピクセルまでの最小コスト経路を最小累積コストマップから読み取り, リアルタイムで描画する. 図 2.7 に対する Intelligent Scissors の処理を例として説明する. 図 2.7 の赤い円の部分をクリックし, カーソルがシートの上部をなぞるように左側

にドラッグすると,図 2.8〜2.10 のように,カーソル位置までの輪郭をリアルタイムで描画する.

図 2.7 元画像

図 2.8 Intelligent Scissors による輪郭の描画 1

図 2.9 Intelligent Scissors による輪郭の描画 2

図 2.10 Intelligent Scissors による輪郭の描画 3

描画する輪郭は最小コストを通る経路であるため,必ずしもマウスの動きに沿った輪郭を抽出するわけではない.例えば,図 2.10 の位置からシートの輪郭を抽出するためにマウスを下側にドラッグすると,図 2.11 の位置では,シートの上から下に線を引く.ここからマウスを少し右にドラッグすると,それまでの輪郭が消えて,より短い経路である図 2.12 を描画する.これは描画する経路が長くなったことで,コストが増加し,コストの低い経路が他に見つかったため,描画される経路が変更された.

図 2.11 Intelligent Scissors による輪郭の描画 4

図 2.12 Intelligent Scissors による輪郭の描画 5

このように,物体の領域全体を囲む輪郭を一度に描画しきることができない場合,途中まで描画した輪郭を，一旦，確定し,その位置を始点として処理を継続する.途中まで描画した位置を始点とするため,図 2.13 のように輪郭の続きを描画することができる.図 2.13 は図 2.11 から処理を継続した画像を表している.

図 2.13 輪郭の続きを描画した画像

図 2.14 の写真に Intelligent Scissors を用いて前景抽出を行った様子を説明する.図 2.15～21 のように,物体の輪郭の一部を抽出した画像を，順次，生成する.図 2.15～21 の画像を輪郭とそれ以外の部分に 2 値化する．図 2.22 は図 2.15 を 2 値化した画像である.

図 2.14 元画像

図 2.15 輪郭が描画した画像 1

図 2.16 輪郭が描画した画像 2

図 2.17 輪郭が描画した画像 3

図 2.18 輪郭が描画した画像 4

図 2.19 輪郭が描画した画像 5

図 2.20 輪郭が描画した画像 6

図 2.21 輪郭が描画した画像 7

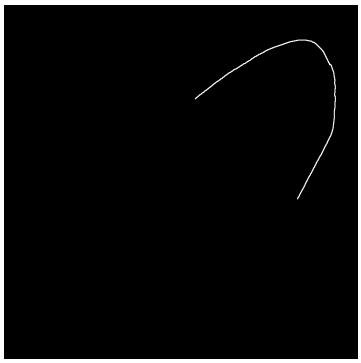

図 2.22 輪郭のみの 2 値画像

それらを繋ぎ合わせることで， 図 2.23 のように, 物体の輪郭のみの画像を生成する. その抽出した輪郭の内側領域を塗りつぶすことで図 2.24 のようなマスク画像を生成し, マスク処理を行うことで, 図 2.25 のような背景と分離した画像を生成する.

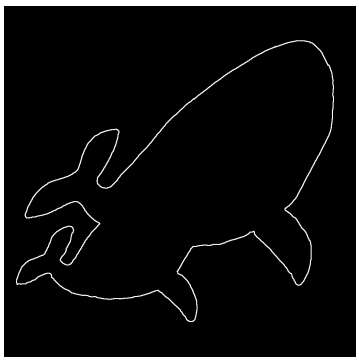

図 2.23 輪郭の 2 値画像

図 2.24 内側領域を塗りつぶした画像

図 2.25 前景抽出画像

2.4 GrabCut

GrabCut[3]は,ユーザーが画像内の対象物をマウスで囲むなどすることで,自動的に物体と背景の分離を行う前景抽出処理である.今回は,画像処理ソフト GIMP に実装されている GrabCut のツールを用いて,評価した.まず,図 2.27 のように,領域抽出をしたい物体の周囲をマウスで囲む.図 2.28 のように,外側の領域を完全な背景領域とし,内側の領域を前景か背景かわからない未知の領域として設定する.次いで, 図 2.29 のように,前景領域を手作業で塗り潰す.

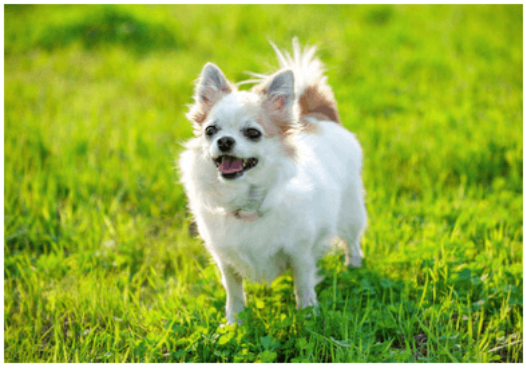

図 2.26 元画像

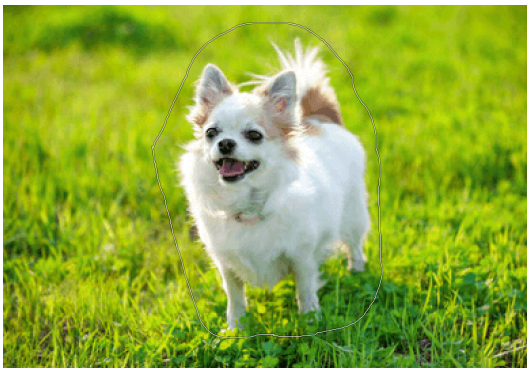

図 2.27 対象物を囲んだ画像

図 2.28 領域を設定した画像

図 2.29 塗り潰した画像

こうすることで,塗り潰した前景領域の画素と,背景領域の画素を基に,前景領域と背景領域における画素値の分布をグラフとして生成する.その画素値の分布から,各ピクセルにおける前景らしさ,背景らしさを表す確率を計算する.各ピクセルの確率と隣接画素間の画素値の差から自動的に,図 2.30 のように,前景領域を抽出する.

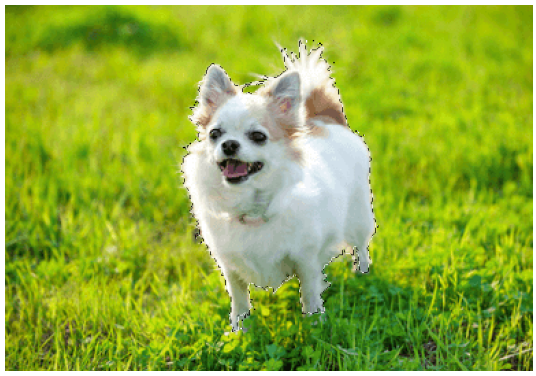

図 2.30 前景抽出した画像

2.5 マジックワンド

マジックワンド[4]は,ユーザーが選択した背景領域の画素値を基に,自動的に物体と背景を分離する前景抽出処理である. 今回は, 画像処理ソフト GIMP に,実装されているマジックワンドのツールを用いて,評価した. マジックワンドは, ユーザーがクリックした背景ピクセルと, それに隣接する画素に基づいて背景領域を判定する.マウスカーソルを下方向にドラッグすることで閾値を大きくすることができ,より明るい領域を選択することができる.マウスカーソルを上方向にドラッグすることで閾値を小さくすることができ,より暗い領域を選択することができる.この二つの操作により,選択範囲を設定し,背景領域を設定することで前景と背景の分離を行う.図 2.32 は,図 2.31 の背景領域をクリックしたことで,クリックしたピクセルの類似色の領域を背景として選択

している. マウスカーソルを上方向にドラッグすると, 図 2.33 のように, より明るい部分が背景領域として選択される. マウスカーソルを下方向にドラッグすると, 図 2.34 のように, より暗い部分が背景領域として選択される.

図 2.31 元画像

図 2.32 クリックした画像

図 2.33 カーソルを上方向にドラッグした場合

図 2.34 カーソルを下方向にドラッグした場合

2.6 凹凸係数

凹凸係数[5]は, 元画像にガウシアンフィルタで平滑化した画像を生成し, その画像で元画像を除算することで求めることができる. その値は, 各ピクセルにおける, 周辺画素の画素値との類似度を表している. 主に, 2 値化する前の陰影削除とエッジ検出に用いる. 図 2.35 のように, 凹凸係数の値は 1 付近に分布する. 1 のピクセルは周辺の画素と全く同じであるため, 平坦な部分となる. 図 2.31 の画像のように, 変化がなだらかな影は, 周辺画素との変化が小さいため, 比較的平坦な部分となる. 逆に, 1 から離れるほど周辺画素からの変化が大きい, 傾斜がきつい部分となる. また, 1 よりも小さい値は周辺画素より暗い部分での傾斜を表し, 1 よりも大きい値は明るい部分での傾斜を表している. そのため, 1 よりも小さい値を閾値として, 2 値化した場合, 周囲より暗く変化している部

分のエッジを抽出し,1 よりも大きい値を閾値として,2 値化した場合,周囲より明るく変化している部分のエッジを抽出する.

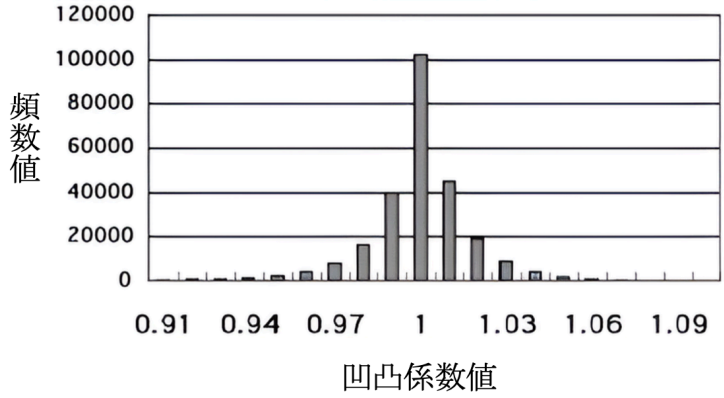

図 2.35 凹凸係数の分布

図 2.37 は,図 2.36 の輪郭部分を表している.図 2.37 のように,輪郭付近のピクセルの凹凸係数を計算した場合,周辺画素に物体領域の画素と,背景領域の画素が含まれる.そのため,輪郭に近づくほど,背景領域のピクセルは周辺画素の平均値が下がるため,1 よりも大きい値を取り,前景領域のピクセルは周辺画素の平均値が上がるため,1 よりも小さい値を取る.

図 2.36 元画像

図 2.37 輪郭部分

図 2.38 は,1 よりも小さい値(0.95)を閾値として, 2 値化した場合である.物体の輪郭のわずかに内側の,凹凸係数の値が小さい部分をエッジとして抽出した.図 2.39 は,1

よりも大きい値(1.05)を閾値として,2 値化した場合である． 物体の輪郭のわずかに外側の,凹凸係数の値が大きい部分をエッジとして抽出した.

図 2.38 0.95 を
閾値とした 2 値画像

図 2.39 1.05 を
閾値とした 2 値画像

図 2.40～41 は,図 2.38～39 で抽出したエッジの内側領域を,前傾抽出した画像である.図 2.41 は,図 2.39 で抽出したエッジが,輪郭よりも外側であったため,抽出した前景領域に背景領域が一部含まれた.そのため,物体の内側領域を抽出する場合は,1 よりも小さい値を閾値として,2 値化するため,凹凸係数に変換した画像は 1 以上の値をクリッピングされるが,前景抽出処理をする上では問題はない.

図 2.40 前傾抽出処理した画像 1

図 2.41 前傾抽出処理した画像 2

次に, 図 2.42 を用いて, 凹凸係数による 2 値化と通常の 2 値化の比較を行う. 図 2.42 のような, 影のある画像を 2 値化する場合, 一つの閾値で背景と前景を分離することは難しい. 図 2.43 は図 2.36 を 2 値化した画像であるが, 影の一部を前景と判断してしまい, 物体の輪郭が一部消えている. 図 2.44, 45 は画素値を凹凸係数に変換した画像と, それを 2 値化した画像である. 図 2.45 は, 影が少し残っているが物体の輪郭を鮮明に抽出している.

図 2.42 元画像

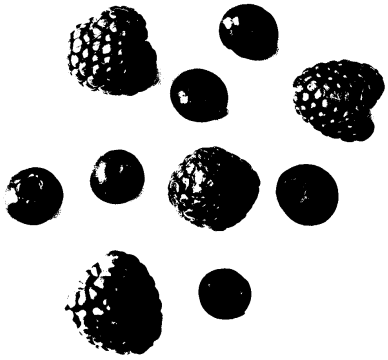

図 2.43 2 値化した画像

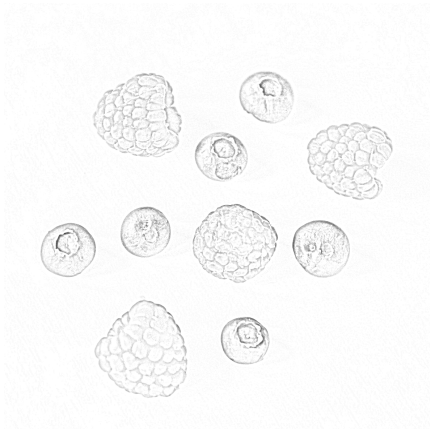

図 2.44 凹凸係数に変換した画像

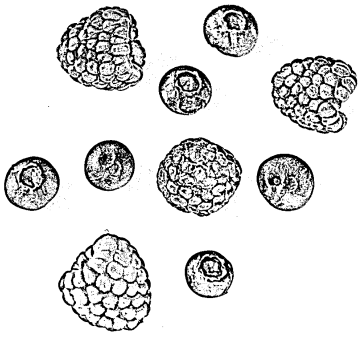

図 2.45 凹凸係数の画像を 2 値化した画像

2.7 モフォロジー処理による輪郭復元とノイズ除去

2 値化した際に, 物体の輪郭が途切れることや, 背景領域にノイズがある場合. 画像上の物体のサイズや, 形状を変化することで, 輪郭を復元する. このような処理をモフォロジー処理という. モフォロジー処理には主に膨張処理と収縮処理の二つがある. 膨張処理は図 2.46〜48 のように, 1 つのピクセルに対して, 上下左右に 1 画素ずつ膨張させる.

収縮処理は図 2.49～51 のように, 1 ピクセルに対し, 上下左右に 1 画素ずつ収縮させる. この二つの処理を組み合わせた処理をクロージング・オープニング処理といい, 2 値画像のノイズ除去や物体の隙間埋めを行うことができる.

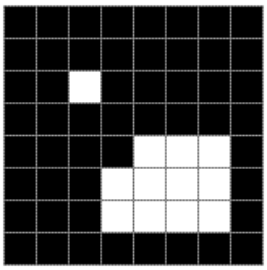
図 2.46 膨張処理 1

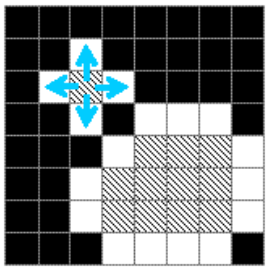
図 2.47 膨張処理 2

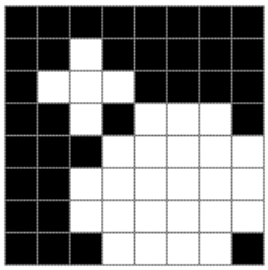
図 2.48 膨張処理 3

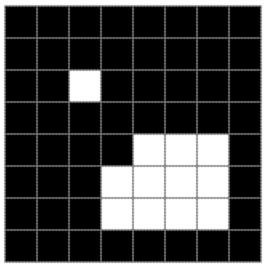
図 2.49 収縮処理 1

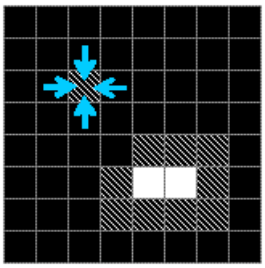
図 2.50 収縮処理 2

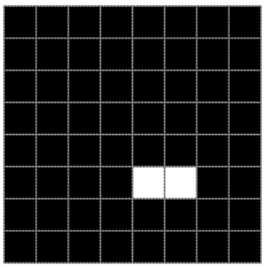
図 2.51 収縮処理 3

クロージング処理は, 2 値画像に対して数回の膨張処理を行った後, 膨張処理と同じ回数の収縮処理を行う処理である. 主に, 物体の隙間埋めに用いられる. 図 52～54 で, 物体の隙間が空いている画像に膨張処理を行い, 隙間を埋める. 次に, 元のサイズになるよう収縮処理を行うことで, 隙間が埋められた図 2.54 を作成する.

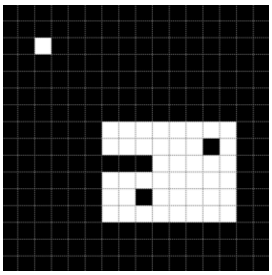
図 2.52 元画像

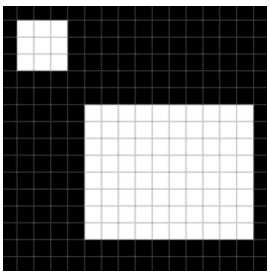
図 2.53 膨張処理

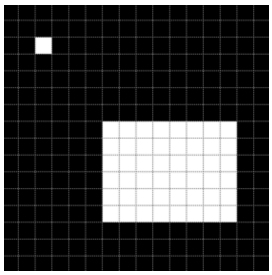
図 2.54 収縮処理

一方, オープニング処理は, 2 値画像に対して数回の収縮処理を行った後, 収縮と同じ回数の膨張を行う処理である. 主に, ノイズ除去に用いられる. 図 2.55～57 で, 背景領域

にノイズがある画像に収縮処理を行い，ノイズを除去する．次に，元のサイズになるよう膨張処理を行うことで，ノイズのみを削除した図 2.57 を作成する．

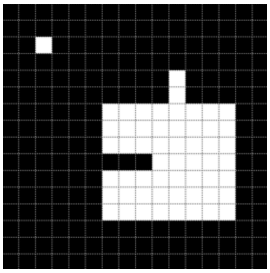

図 2.55 元画像

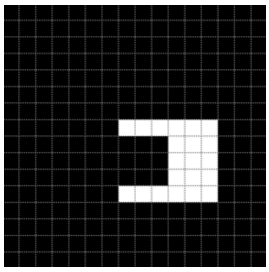

図 2.56 収縮処理

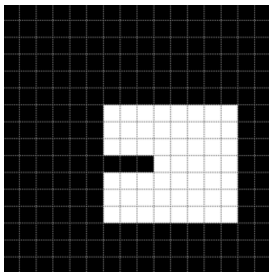

図 2.57 膨張処理

3 章　それぞれの前景抽出処理の性能評価

3.1 凹凸係数による前景抽出処理の性能評価

一般的に，領域抽出は，画像を何らかの方法で 2 値化し，その後，抽出した領域のノイズを除去することで完成する．しかし，背景領域に前景物体の影が映り込んだ画像を，一つの閾値で正確に 2 値化することは難しい．図 3.2 は図 3.1 を 2 値化した画像である．この場合，影を除去するために閾値を低く設定したことで，物体の輪郭が一部途切れている．このように，影のある画像に対し，そのまま 2 値化することで輪郭抽出することは困難である．そのため，2 値化する前に影を薄くする必要がある．凹凸係数は周辺ピクセルとの画素値のズレに基づいて計算されるため，図 3.3 のようになだらかな影を除去することができる．図 3.4 は凹凸係数に変換した画像を 2 値化した画像であり，影の影響を受けず，前景領域の輪郭部分を正しく 2 値化していることがわかる．

図 3.1 元画像

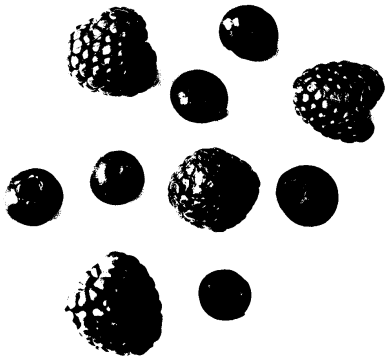

図 3.2 画素値に基づいた 2 値画像

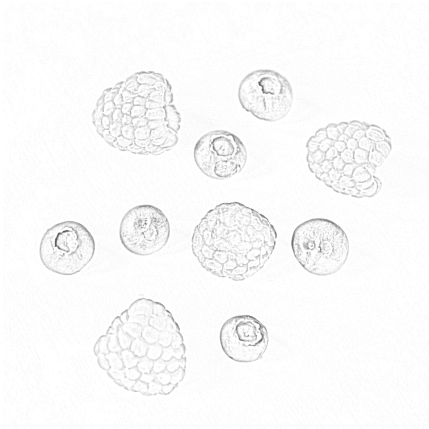

図 3.3 凹凸係数に変換した画像

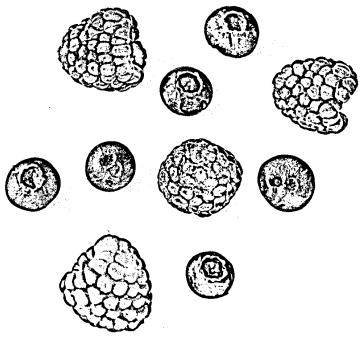

図 3.4 凹凸係数に基づいた 2 値画像

しかし, 輪郭が弱い部分では 2 値化する段階で輪郭が断裂する場合がある. そのため, クロージング処理を行うことで, 輪郭を復元する必要がある. クロージング処理の前処理として, 図 3.4 を白黒反転した図 3.5 を生成する. その後, 白黒反転した画像にクローシング処理を行うことで輪郭の復元した図 3.6 を生成する.

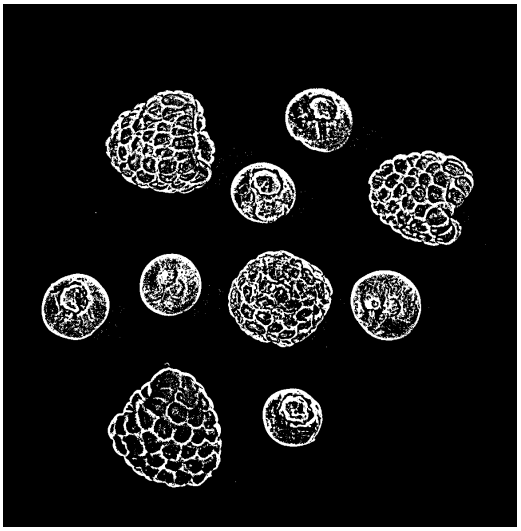

図 3.5 白黒反転した画像

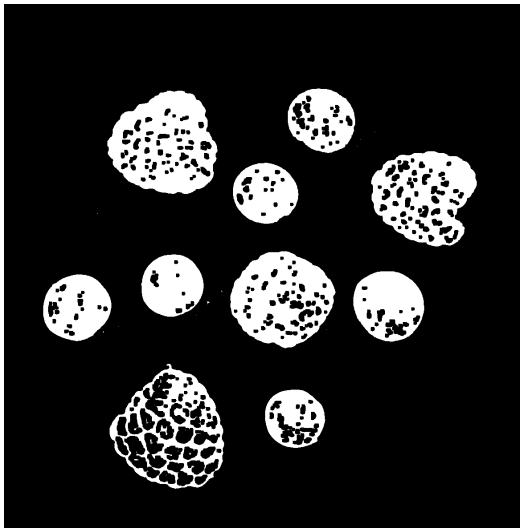

図 3.6 輪郭復元した画像

次に, 図 3.6 のそれぞれの物体における最外輪郭の内側を塗り潰した図 3.7 を生成する. 図 3.7 の赤い円に囲まれている部分のように, 背景領域にノイズが発生する可能性があるため, オープニング処理を行い, ノイズを除去した図 3.8 を生成する. 図 3.8 の画像でマスク処理を行い, 物体と背景を分離した図 3.9 を生成する.

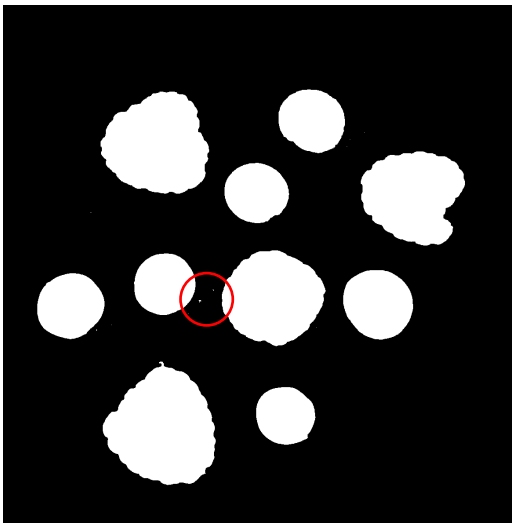

図 3.7 最外輪郭を塗り潰した画像

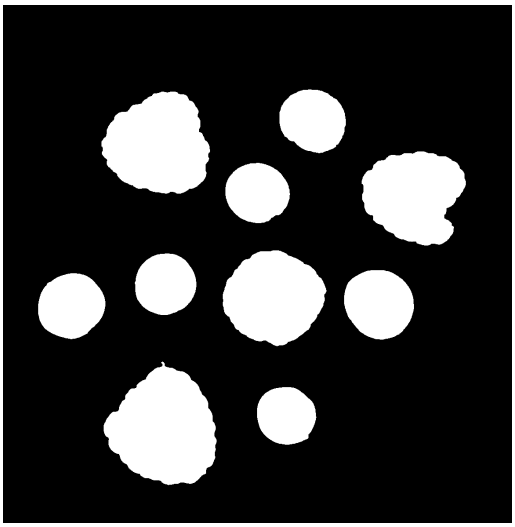

図 3.8 ノイズ除去した画像

図 3.9 マスク処理した画像

フリマアプリで使用される写真は机の上などで撮影されるため, 背景は単純なものとなる. そのため, 前景抽出処理がうまくいかない理由は影による影響が大きい. 今回の研究では, 凹凸係数により除去できる影の範囲を調べることで, 実際にフリマアプリで利用するのに適しているかどうか考察する.

3.2 Intelligent Scissors と従来手法の性能比較

Intelligent Scissors は, ユーザーが物体の輪郭を直接指定できるため, 複雑な形状の物体に対しても柔軟に対応することができる. しかし, 正確な輪郭の指定が必要となるため, 図 3.10 のような, 輪郭が弱い物体や, 図 3.11 のような, 複雑な形状の物体には

適さない． GrabCut は，ユーザーが選択した背景領域と前景領域の画素値を基に前景抽出をするため，図 3.10 のような，物体と背景の色の差が小さいと正確な境界を見つけることができない．マジックワンドはユーザーの操作が単純であるため，処理にかかる時間は少なくなる．しかし，設定した閾値よりも大きい領域と小さい領域に分離するアルゴリズムは 2 値化処理とほとんど同じであるため，影などの影響を強く受ける．これらの特性を踏まえて，Intelligent Scissors を GrabCut およびマジックワンドと比較し，フリマアプリで利用するのに適しているか考察する．

図 3.10 輪郭が弱い物体

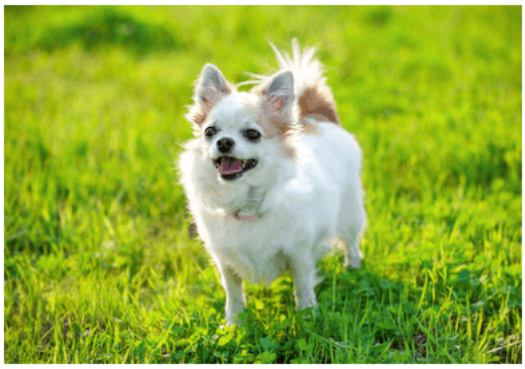

図 3.11 複雑な形状の物体

4 章　フリマアプリに適した前景抽出処理の考察

4.1 凹凸係数の性能評価

凹凸係数は周辺画素との画素値のズレが大きい部分を抽出し, 均一で弱い影を除去する. そこで, 影の濃さがバラバラである図 4.1 を用いて, 除去できる影の範囲を考察する.

図 4.1 影の濃さがバラバラな物体

図 4.2 の凹凸係数画像を見ると, 概ね影を除去できているが, ヘッドホン上部の弱い輪郭が消えかけている. 図 4.3 は, 閾値を低くし, 抽出するエッジの範囲を広げて 2 値化することで, 消えた弱い輪郭を抽出した. しかし, 抽出する範囲を広げたことで影の領域を少し抽出してしまった.

図 4.2 凹凸係数に変換した画像

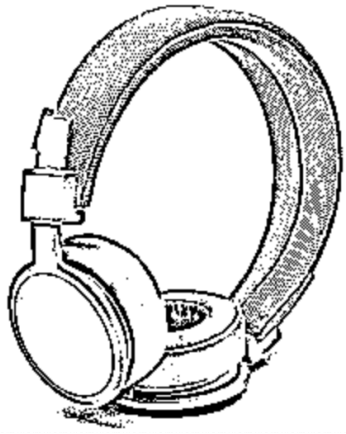

図 4.3 2 値化した画像

このことから, 凹凸係数は影を除去するために, 弱い輪郭も共に消してしまうことがわかった. そこで, 図 4.4 と図 4.5 のように, 影や背景と同じ色をした, 輪郭が弱い物体

を処理することで,陰影除去できる物体の範囲を考察する.

図 4.4 影と近い色の物体

図 4.5 背景と近い色の物体

図 4.4 を凹凸係数に変換した場合,図 4.6 のように,影を除去するはできたが,影と隣接していた輪郭も除去した.そのため,2 値化した画像は図 4.7 のように,輪郭が大きく断裂した画像となった.

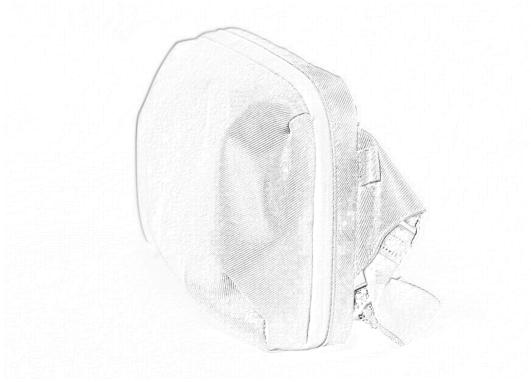

図 4.6 凹凸係数に変換した画像

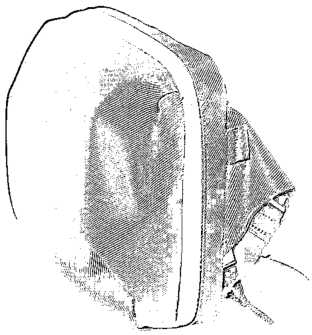

図 4.7 2 値化した画像

図 4.5 を凹凸係数に変換した場合, 前景物体の弱い輪郭を鮮明に抽出することはできず,図 4.8 のように,弱い途切れた輪郭となった.そのため,2 値化した画像は,図 4.9 のように,途切れた輪郭が抽出した.また,弱い輪郭を抽出したため,凹凸係数に変換した際に,薄めた影の輪郭や背景ノイズも抽出した.

図 4.8 凹凸係数に変換した画像

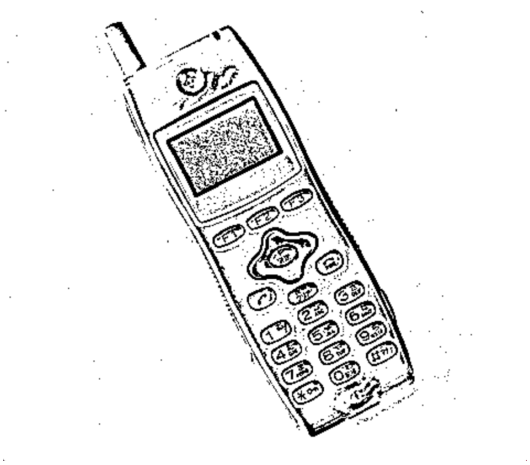

図 4.9 2 値化した画像

凹凸係数への変換で影を除去する際に,物体の薄い輪郭も除去されるため,輪郭が途切れてしまう.そのため,図 4.1 のヘッドホンの上部や,図 4.5 の電話の左上部ように,薄い輪郭が存在する物体での影の除去は困難である.図 4.4 のように,影と近い色をした物体では,影と接している部分の輪郭が薄くなるため,同じように,影を除去すると薄い輪郭が途切れてしまう.そのため,凹凸係数を利用した物体の陰影除去は,物体の輪郭が薄くならないように光の当たり方を調整した上で,背景と影の色が同色でない物体の場合のみ,利用できる.しかし,フリマアプリでは, 使わなくなった衣服やアクセサリーなどの日用品が出品されることが多く,影と同色である黒色はそれらによく使用されている.そのため,利用できる場面は制限される.

4.2 Intelligent Scissors と従来の手法の性能比較

Intelligent Scissors と比較するために,4.1 の実験で使用した画像に.Intelligent Scissors,GrabCut, マジックワンドでそれぞれ処理を行った.図 4.1 の場合,Intelligent Scissors は正しく前景を抽出することができたが,GrabCut ではヘッドホン上部の弱い輪郭を正しく抽出することができず,抽出した前景領域に背景領域が含まれた.マジックワンドでは,影と接した輪郭を正しく抽出できなかった.図 4.5～9 は,図 4.1 を Intelligent Scissors,GrabCut,マジックワンドによって前景抽出した画像である.

図 4.5 Intelligent Scissors で
前景抽出した画像

図 4.6 GrabCut で
前景抽出した画像 1

図 4.7 GrabCut で
前景抽出した画像 2

図 4.8 マジックワンドで
前景抽出した画像 1

図 4.9 マジックワンドで
前景抽出した画像 2

図 4.4 の場合， Intelligent Scissors は影に接した輪郭を正しく抽出した. しかし, 輪郭が非常に薄いため, 輪郭上を正確にクリックする操作に時間を要した. GrabCut とマ

ジックワンドでは影を輪郭として抽出した.図 4.10～12 は,図 4.4 を Intelligent Scissors,GrabCut,マジックワンドによって前景抽出した画像である.

図 4.10 Intelligent Scissors で
前景抽出した画像

図 4.11 GrabCut で
前景抽出した画像

図 4.12 マジックワンドで
前景抽出した画像

図 4.5 の場合, Intelligent Scissors のみが輪郭を正しく抽出した.図 4.2 と同じように,輪郭が非常に弱いため,輪郭上を正確にクリックする操作に時間を要した.GrabCut は部分的に誤った輪郭をとっているが,マジックワンドは背景と物体の色が近いため,抽出した前景領域のほとんどが背景領域となった.図 4.13～15 は,図 4.5 を Intelligent Scissors,GrabCut,マジックワンドによって前景抽出した画像である.

図 4.13 Intelligent Scissors で
前景抽出した画像

図 4.14 GrabCut で
前景抽出した画像

図 4.15 マジックワンドで
前景抽出した画像

GrabCut とマジックワンドの処理速度は早いが,物体の色や光の当たり方などを考慮した限定的な条件でなければ正しく処理することができないため,凹凸係数と同じく,フリマアプリで利用できる場面は制限される.Intelligent Scissors は時間を要するが,描画する輪郭を自身で選択することができるため,ユーザーが正しく操作することができれば,物体の色や影に依存せず,幅広い画像を柔軟に処理することができる.しかし,図 4.16 のような,形状が複雑な物体や,図 4.17 のような,輪郭が弱く見づらい物体を正しく処理するためには高度な操作が求められる.そのため,正確に輪郭を抽出したい場合や,GrabCut などの従来の手法では正しく処理できない物体でのみ Intelligent Scissors を用いるべきである.

図 4.16 複雑な形状の物体

図 4.17 輪郭が薄い物体

5章　結論

フリマアプリで利用することを前提として,複数の前景抽出処理の研究を行なった.結果として,どんな場面でも利用できるような処理を開発することはできなかった.フリマアプリの出品写真は,テーブルや机の上で撮影されることが多いため,通常の写真と比べて,背景にノイズとなる要因はほとんどない.そのため,前景抽出処理が正しく行えない原因は背景のノイズではなく,影や物体の色によって輪郭が薄くなることであった.背景や影と同色の物体や,強い光が当たっている物体の輪郭は非常に薄くなるため,従来の画素値の勾配や閾値による処理では抽出することができなかった.　しかし,ユーザーが抽出する輪郭を手作業で選択する Intelligent　Scissors であれば,薄い輪郭であっても抽出することができるため,物体の色や光のあたり方に左右されずに幅広い物体を柔軟に処理することができた.しかし,複雑な形状や輪郭の薄い物体の輪郭を正しく選択するには,高度な操作を求められる.フリマアプリの利用者のほとんどは一般の方であるため,高度な操作を必要とする処理は適していない.そのため,Intelligent Scissorsのみを使用するのではなく,従来の処理では抽出することが難しい輪郭のみに使用することが最も適していると言える.今後の課題として,Intelligent　Scissors と従来の前景抽出処理とを組み合わせるプログラムの作成が挙げられる.

参考文献

[1] Loyalty Marketing, Inc, SERVICE FOR BUSINESS, 2022年, 株式会社ロイヤリティ マーケティング「2500 人に聞いたフリマアプリの利用実体」, https://biz.loyalty.co.jp/report/019/

[2] Eric N. Mortensen & William A. Barrett, Intelligent Scissors for Image Composition, Brigham Young University, 1995年, https://dl.acm.org/doi/pdf/10.1145/218380.218442,

[3] Alexander Monrdvintsev & Abid K, GrabCutを使った対話的前傾抽出処理, 2018年, http://oyamada/OpenCV/py_tutorials/py_imgproc/py_grabcut/py_grabcut.html,

[4] カンデザWeb, ファジー選択を使った切り抜き・透明化 - GIMP, 2021年, https://citrus-designs.com/gimp-corp-with-fuzzy-selection/

[5] 佐藤弘起, シェーディング画像に良好なしきい値を設定できる変動しきい値式2値化処理法, 2007年, https://www.jstage.jst.go.jp/article/iieej/36/3/36_3_204/_pdf/-char/ja

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　本研究で開発したプログラム

1. open.py

   凹凸係数による前景抽出処理を行うプログラム

2. gpt.py

   Intelligent Scissors による前景抽出処理を行うプログラム