Vol.2018-CDS-22 No.13
2018/6/1

スマートフォンカメラによる距離画像生成のための自動パラメータ推定

北野 和彦[1]　小林 亜樹[1]

概要：奥行き情報を加えて商品の様子を伝えたり，物体認識への応用などを目指して，スマートフォン単体での距離推定方式を開発している．本方式では，単眼カメラの撮影画像から距離を推定する DFF(Depth From Foucs) 法を用いるが，一般のレンズではフォーカス位置変化によって引き起こされる像倍率変化が問題となる．そこで，これを補償するため倍率変化率を推定する手法を提案している．本手法では，フォーカス画像列中のエッジ軌跡画像を生成し，Hough 変換によりエッジ移動を追跡することで変化率を推定する．本稿では，処理手順中のパラメータ設定の必要箇所において，自動的にパラメータを決定する手法を提案し，いくつかの画像例について適用して有効性を確認する．

キーワード：Android，Hough 変換，フォーカス位置，像倍率変化，距離画像

KITANO Kazuhiko[1]　KOBAYASHI Aki[1]

1. はじめに

広く普及したスマートフォンは，多数のセンサを搭載しており，外界の状況を取得してデータとして転送できる．カメラはその中でも最も理解されているセンサであり，実用から趣味までの幅広く利用されている．しかし，一部の機種に搭載されたステレオカメラや，別途センサとなる距離センサ等を除くと，得られる画像情報は 2 次元に限られる．

これに対して，3 次元情報のうちでも少なくとも奥行き情報が得られれば様々な応用が考えられる．例えば，家具などの模様をよりリアルに伝えることができるようになる．例えば，寸法を個別に測ることなく大きさを合わせた部品を購入することができるようになる．例えば，料理の盛り付けの様子を伝えることができる．例えば，生産品の状況をより正確に伝えられるようになる．

このように，2 次元の画像情報だけでは伝えきれない情報を取り扱えると様々な応用が考えられる [*1]．奥行き情報を広く利用できるようにするためには，広く普及したスマートフォンによって取得できることが重要であると考えている．そこで筆者らはスマートフォンに搭載されたカメラを用いた，奥行き情報の取得を試みている．

カメラから取得できる撮影画像から奥行き距離を推定する手法としては，ステレオ法が広く知られているが，2 台のカメラを使用し，同じ場所を違った角度から撮影するため 2 台のカメラの間隔をある程度広げなければならなく，小型化が困難であるとともに，利便性に劣る．そこで，単眼カメラから奥行き情報を推定する方法になると，ステレオ法とは異なり「両眼立体視」に基づいた推定ができないため，DFF（Depth From Focus）法を用いる．これは異なるフォーカス位置で撮影された画像列に対して画素毎に合焦判定を行い，合焦したフォーカス位置を合焦フォーカス位置とする．この合焦フォーカス位置から対象物までの奥行き距離に対応付けて推定する手法 [3] である．

しかし，多くのカメラは，インナー（インターナル）フォーカスが採用され，フォーカス位置変化によるレンズ長が変わらない特徴があるが，映る像の大きさが変化する像倍率変化が生じる．この像倍率変化は，DFF 法によって奥行き距離を推定する場合に推定精度の低下につながるため，フォーカス位置変化に伴う像倍率変化の少ないテレセントリック光学レンズ [4][5] を用いた研究が多く見られる．

一方，像倍率変化を補償するために一般の光学系で活用

[1] 工学院大学
Kogakuin University, Japan

[*1] 本稿では，カメラ前数十 cm 程度の距離での奥行き情報取得を主たる目的としている．より精密な測定ができれば，洋服のテクスチャを伝えたりできるようになる．また，より遠方を測定できれば風景や街並みのリアルを伝えられるようになる．

しようとする研究もあり，SURF により特徴点抽出を行い，画素間の対応付けから補償する手法 [7] や位相関数によって推定された画像特徴のシフトに基いて補償する手法 [6] などがある．しかし，これらの研究は，一眼レフカメラを用いており，広く普及したスマートフォンのカメラなどで利用できるかは不明である．

そこで，筆者ら [9] は Android OS 搭載のスマートフォンのカメラを用いて取得した画像を用いて奥行き距離推定を行っており，最終的にはスマートフォン単体での距離推定方式の開発を目指している．

スマートフォンのカメラを使用することで 3 つのメリットが考えられる．1 つ目は，コスト面である．スマートフォンの普及率は高く，Android 5.0（APILeve21）以降の OS が搭載されている端末（一部を除く）では，手動でのフォーカス位置制御が可能であり，別途にカメラを購入する必要がない．2 つ目は，採用性である．スマートフォンの平均使用年数は，4.3 年 [10] であり，買い替える頻度が多くマニュアルフォーカス制御が可能な端末が普及しやすい．3 つ目は，利便性である．一眼レフカメラは，写真の撮影が主な用途であり，距離推定は別途 PC 等を用いた処理が必要になるが，スマートフォンは，端末自体で処理が可能であり，スマートフォン単体での距離推定が可能である．

スマートフォンのカメラを用いて撮影された画像であっても遠近の違いが判別できる程度の距離値推定は可能であるが，像倍率変化による影響により一部の画素において誤った距離が推定される [8] のが現状である．フォーカス位置の変化による像倍率変化率は，フォーカス位置によらず一定で，画像内の位置に対して不変であると仮定すれば，単純な拡大縮小で補償が可能である．筆者らは撮影物体上のエッジの変化を直線近似できるものとみなすことで，エッジ位置のフォーカス位置変化に伴う変化を表現するエッジの変化画像を生成し，この上で Hough 変換 [1][2] による直線検出を行い，検出された直線の傾きから像倍率変化率の推定を行った．このとき，Hough 変換に用いたパラメータを手動で設定したため自動化が課題であった．

そこで，本稿では，処理手順中のパラメータ設定の必要箇所において，自動的にパラメータを決定する手法を提案し，いくつかの画像例について適用して有効性を確認する．

本章の構成は，以下の通りである．2 章に Hough 変換等の自動パラメータ推定手法を含む像倍率変化率を推定手法を示す．3 章にて，複数の画像列に適用する．4 章にて実験結果と考察を示す．5 章にて本稿をまとめる．

2. 像倍率変化率推定手法

2.1 用語定義

- フォーカス位置
  Andorid OS 5.0 (APILevel21) 搭載の端末を制御する API の 1 つであるカメラのフォーカス制御を行うパラメータであり，本稿ではフォーカス位置 d と表記する．フォーカス位置 $d \in [0.0, 10.0]$ である．このとき $d = 0.0$ は，無限遠に相当し，$d = 10.0$ は最短撮影距離に相当する．
- マルチフォーカス画像
  同一シーンについて，適当なステップ幅を以て異なるフォーカス位置毎に撮影した画像列である．これら画像のことを撮影画像と呼ぶ．
- エッジ強度
  撮影画像について，適当なエッジ抽出オペレータにて抽出した画素毎の値を表す．この値で再構成した画像をエッジ画像と呼ぶ．
- 2 値化閾値
  Hough 変換は，入力画像が 2 値画像である必要があるため，画像内を白と黒の 2 段調に変化する処理の際に使用する閾値であり，2 値化閾値あるいは，th_b と記述する．
- 投票数閾値
  Hough 変換から直線を検出する場合，p-θ パラメータ空間内における投票数は，直線上の点の数に依存するため，直線とみなすために必要な最低限の投票数を意味する閾値であり，投票数閾値あるいは，th_v と記述する．
- 距離値
  マルチフォーカス画像より DFF 法によって推定した画素毎の奥行き距離である．この値で再構成した画像を距離画像と呼ぶ．

2.2 概要

スマートフォンのカメラを用いて取得したマルチフォーカス画像から DFF 法の原理を利用して対象物までの距離を推定する．このとき，スマートフォンのカメラもフォーカス位置変化に伴う像倍率変化が生じるレンズであるため，マルチフォーカス画像をそのまま使用するだけでは，実際の遠近に対応した距離値とそれに反した距離値が推定される距離推定精度の低下につながる．筆者らは Hough 変換を用いて像倍率変化率を補償する手法に提案を行った．Hough 変換を適用するための入力画像の 2 値化処理に使用する 2 値化閾値 th_b，直線とみなすための投票数閾値 th_v は，手動で決定をしていたため，像倍率変化率推定の完全自動化までは達成できていない．本稿では，この 2 つの閾値パラメータを自動で設定する手法を提案し，いくつかの撮影画像列において像倍率変化率の推定を行う．

まず，距離推定を全体の手法の流れを図 1 に示す．あるステップ幅で分割したフォーカス位置 d でマルチフォーカス画像を撮影し，エッジ抽出オペレータを用いてエッジの画像列を用意し，入力画像とする．距離を推定する前処理として，像倍率変化率の推定及び像倍率変化を補償する．

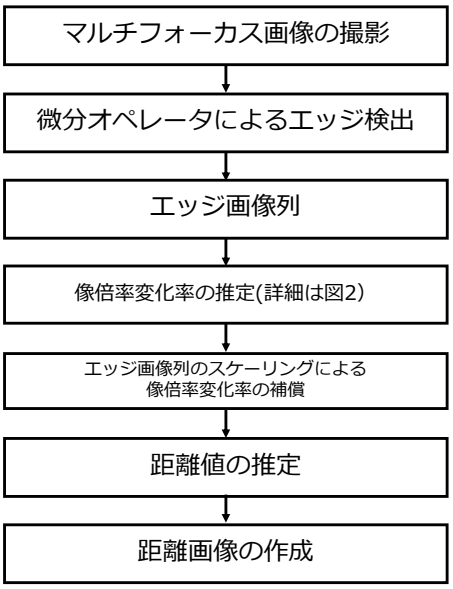


図 1: 距離推定手法全体の流れ

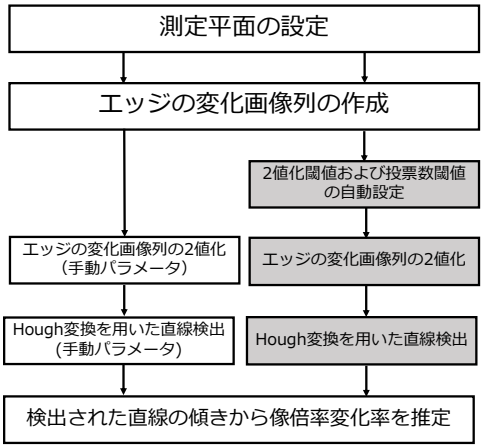


図 2: 像倍率変化率推定手法の流れ

像倍率変化率推定手法の流れは図 2 の通りである．

フォーカス位置変化に伴う像倍率変化によるエッジ位置の変化を測定することでエッジの変化画像を作成する．このエッジ位置変化の分布は直線上であると仮定しているため，エッジの変化画像に対して Hough 変換を行い，直線検出を行う．この直線の傾きを変化率として，複数領域で測定を行ったエッジの変化画像列を用いて像倍率変化率を推定し，その像倍率変化率を用いた，スケーリング処理を入力画像列に適用し補償する．

本稿では，後述するアルゴリズムを用いて 2 値化閾値 th_b および投票数閾値 th_v を決定する．

2.3 入力画像列の生成

AndroidOS 搭載にスマートフォンのカメラにおいて，API で制御できるフォーカス位置は $d = [0.0, 10.0]$ であり，N 段階に分割をしてスマートフォンを固定した状態で N 枚撮影を行う．マルチフォーカス画像は，x 軸，y 軸に加えてフォーカス位置 d 軸の 3 次元の画像で表現され (図 3 参照)，各 d で撮影画像の位置 (x, y) は画素値 $i(x, y, d)$ である．マルチフォーカス画像の画素値 $i(x, y, d)$ を用いて，エッジ抽出オペレータ $\mathcal{E}(\mathbf{I})$ から画素全体に対してエッジ抽出したエッジ画像列 $E(x, y, d)$ を生成する（(1) 式）．

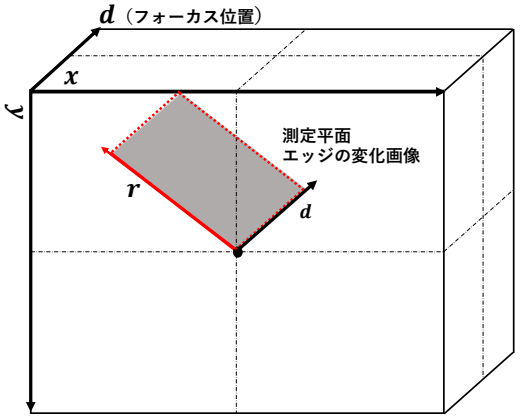


図 3: 画像列の 3 次元表現とエッジ変化画像平面

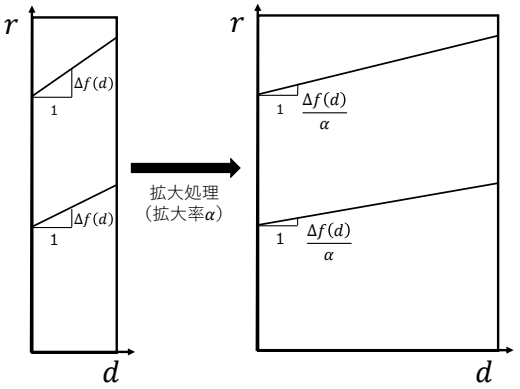


図 4: エッジ変化画像

$$E(x, y, d) = \mathcal{E}(i(x, y, d)) \qquad (1)$$

このエッジ画像列が入力画像列となる．

2.4 エッジの変化画像の作成

フォーカス位置変化に伴うエッジ位置の変化を測定し，エッジの変化画像を作成するにあたって，エッジの変化を測定する平面を設定する．画像内の xy 平面上で画像中心から周辺方向へ向かう線分上（図 3 参照）であり，すべてのフォーカス位置におけるエッジ画像において同一の画素とする．したがって，測定領域はこの線分を d 方向へ拡大した平面上が測定する平面とする．このとき，線分長 r は，入力画像列の短辺方向の長さの半分とする．また，線分の方向については特段の制限を設けないものとする．この測定平面上に展開されたエッジ強度分布をエッジの変化画像とする．

2.5 像倍率変化率の推定および像倍率変化率の補償

エッジの変化画像 $E_c(r, d)$（図 4 参照）は，エッジ画像

列の画像中心から周辺に向かう線分上のエッジ強度分布を縦方向に，フォーカス位置 d の変化に従って横方向に並べた画像である．ここで，エッジの変化画像はその後の処理精度を考慮してアスペクト比がほぼ 1:1 となるような拡大率 α を用いてスケーリングを行った画像である．

Hough 変換は，入力画像が 2 値画像である必要があるため，多値画像であるエッジの変化画像を 2 値化閾値 th_b を用いて，2 値化を行い，2 値化エッジの変化画像 $B(d,r)$ へと変換する（(2) 式）．

$$B(d,r) = \begin{cases} 0 & Ec(d,r) < t_h \\ 1 & Ec(d,r) \geq t_h \end{cases} \tag{2}$$

Hough 変換から得られるパラメータ θ を用いて 2 値化エッジの変化画像で検出された直線の傾きは $1/\tan(\theta)$ であり，この傾きを変化率とする．ここで，処理精度上の問題のために拡大した d 方向の拡大率 θ を考慮すると，推定すべき変化率 $\Delta f(d)$ は，

$$\Delta f(d) = \frac{\alpha}{\tan(\theta)} \tag{3}$$

と示せる．

本手法では，撮影画像の d 変化に伴う像倍率変化率は，d によらず一定であり，画像内の位置に対して不変で単純な拡大縮小で補償が可能であると仮定し，その仮定の下では，任意の d における撮影画像のある座標に対応する，他の d における撮影画像の画素位置の変化は，画像中心で最小値 (0) であり，周辺に向かうにつれて単調増加する．これは，エッジの変化画像上では，（ほぼ）直線状に分布するエッジ強度点列の傾きは，エッジの変化画像上部ほど大きくなる．そこで，このエッジ強度点列が近似直線で表せるとすると，その直線の傾きである変化率 $\Delta f(d)$ は，画像中心から距離 r に比例すると考えられるため，

$$\Delta f(d)_N = \frac{\Delta f(d)}{r} \tag{4}$$

で示すように正規化した $\Delta f(d)_N$ を用いて像倍率変化率を推定することとする．Hough 変換は投票数閾値 th_v よって検出される直線の本数が異なる．このように複数の直線が検出された場合は，各直線の変化率 $\Delta f(d)$ の中央値をとることにより像倍率変化率 R とする．この R を用いて，d 毎のエッジ画像に対して縮小処理を行うことで像倍率変化率を補償する．2 値化閾値 th_b，投票数閾値 th_v の 2 つのパラメータの自動推定法は次節で述べる．

2.6 パラメータの自動設定

2 値化閾値 th_b を自動推定するアルゴリズムを algorithm 2.1，投票数閾値 th_v を自動推定するアルゴリズムを algorithm 2.2 に示す．

Algorithm 2.1 Automatic binarization threshold estimation

Input:
  $num \Leftarrow$ Number of edges change image
  $width \Leftarrow$ Width of edges change image
Output: th_b: binarization threshold
  $hist[num] \Leftarrow$ Histogramr of edge change images
  for $n \Leftarrow 0$ **to** num
    $k \Leftarrow 255$
    $hist_{sum} \Leftarrow 0$
    while $k \geq 0$
      $hist_{sum} \Leftarrow hist_{sum} + hist[num][k]$
      if $hist_{sum} \leq width/10$ **then**
        $k \Leftarrow k - 1$
        continue
      $th[num] \Leftarrow k$
      break
  $th_b \Leftarrow th.max$

Algorithm 2.2 AutomaticVoting number threshold estimation

Input:
  $th_b \Leftarrow$binarization threshold
  $num \Leftarrow$Number of edges change image
  $width \Leftarrow$Width of edges change image
Output: $th_v[5]$: Voting number threshold based on five binarization thresholds
  for $i \Leftarrow 0$ **to** 4
    $th_b \Leftarrow th_v - (i \times 10)$
    $v \Leftarrow width$
    while $v \geq 0$
      for $n \Leftarrow 0$ **to** num
        $lines[n] \Leftarrow$ Acquire detected number from Hough transform for edges change image（th_b,v）
      if $lines \leq 4$ **then**
        $v \Leftarrow v - 10$
        continue
      $th_v[i] \Leftarrow v$
      break

表 1: 撮影に使用した端末

Device	Nexus 5X
OS	Andorid 7.0
CPU 1.8GHz	Qualcomm Snapdragon 808 (MSM8992) ＋ 1.4GHz ヘキサコア
Memory	2GB
画素数	約 1230 万画素
ピクセルサイズ	1.55[μm]
絞り	f/2.0

表 2: 画像処理に使用した PC

OS	ubuntu 14.04 LTS
CPU	Intel Core i3-6100 3.70GHz
Memory	32GB

2.7 距離値推定

距離値推定は，エッジの強度比較手法 [8] を用いる．この手法は，エッジ画像列を入力画像とし，このエッジ画像列 $E(x, y, d)$，座標 (x, y) における距離値 $s(x, y)$ として，(5) 式より算出する．

$$s(x, y) = \min\{\arg\max_{d} E(x, y, d)\} \tag{5}$$

3. 実験

3.1 目的

スマートフォンの実機を用いて取得したマルチフォーカス画像から，パラメータの自動推定法によって決定した，2 つの閾値（2 値化閾値 th_b，投票数閾値 th_v）を用いて，像倍率変化率を推定した場合，どの程度効果があるのかを確認することを目的とする．

3.2 実験条件

フォーカス位置 d の変化に伴う，エッジ位置の変化を測定する領域は，画像の中心から 4 隅に向かう対角線上の線分と，d とで張られた平面とした．本実験では，Android OS を搭載したスマートフォンは，マルチフォーカス画像の取得だけにとどめ，Hough 変換等の処理は，USB ケーブル等を用いて PC に画像列を転送し，PC 内で処理を行う．撮影に使用した端末は，Nexus 5X であり，詳細を表 1 に示す．画像処理に使用した PC の詳細は，表 2 に示す．エッジの変化画像の 2 値化処理や Hough 変換等は，Python 上で OpenCV を使用した．

3.3 撮影アプリケーション

マルチフォーカス画像の撮影は，自作アプリケーションを使用した．このアプリケーションには，以下の 2 つの機能を有している．

表 3: デバイスのサポートレベル

0	INFO_SUPPORTED_HARDWARE_LEVEL_LIMITED
1	INFO_SUPPORTED_HARDWARE_LEVEL_FULL
2	INFO_SUPPORTED_HARDWARE_LEVEL_LEGACY
3	INFO_SUPPORTED_HARDWARE_LEVEL_3

- マニュアルフォーカスモード
- 任意のシャッター間隔による自動撮影

1 つ目のマニュアルフォーカスモードは，フォーカス位置 d を手動で変更できる機能である．このマニュアルフォーカスモードは，Android 5.0 Lolipop(APIlevel21) 以上であることが必須の条件である．すなわち，Android 4.4 KitKat 以下の OS では，動作しない．また，端末側がマニュアルフォーカスに対応した機種である必要がある．これは，API を通じて確認することが可能である．サポートレベル一覧を表 3 に示す．ここで，マニュアルフォーカスモードが可能な機種は，`INFO_SUPPORTED_HARDWARE_LEVEL_FULL` または，`INFO_SUPPORTED_HARDWARE_LEVEL_3` がサポートされている必要がある．Nexus 5X は，`INFO_SUPPORTED_HARDWARE_LEVEL_3` がサポートされているためマニュアルフォーカスが可能である．

2 つ目は，自動撮影である．マルチフォーカス画像を撮影するにあたって，同一のシーンである必要がある．そこで，d を手動で変えながら，撮影してしまうとスマートフォンがずれる可能性がある．そこで，d を変えながら，自動撮影できる機能を追加した．ここで，$d = [0.0, 10.0]$（無限遠から最短撮影距離）までをステップ幅を 0.1 として 101 枚の撮影した．

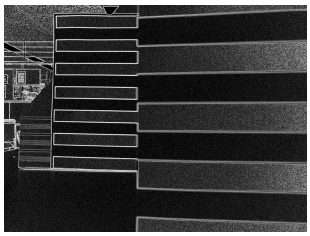
3.4 エッジ位置変化測定領域

フォーカス位置 $d = 0.0$（無限遠の設定）のときの撮影画像を図 7 に示す．この，9 つのマルチフォーカス画像に本手法である，Hough 変換等の自動パラメータ推定手法（2.6 節参照）を用いた像倍率変化率推定手法を適用する．その際に必要となるエッジの変化画像を作成するための測定平面領域を設定する．9 つのシーンの中で，シーン 1 を用いて測定平面を設定し，他の 8 シーンにおいても同様の測定平面を用いる（図 5, 6 参照）．ただし，紙面上では図を見やすくするために画像処理をしてある．以下のエッジ画像やエッジ画像から作成したエッジの変化画像などのすべてが同様である．

このエッジ画像上で，画像中心から，4 隅に向かう対角線上の線分①から④までの 4 本と d とで張られた平面を測定平面とする．

3.5 エッジの変化画像

例として，シーン 1 の線分③の測定平面におけるエッジ

(a) 撮影画像 ($d = 0.0$) **(b)** エッジ画像 ($d = 0.0$)

図 5: シーン 1 における撮影画像およびエッジ画像

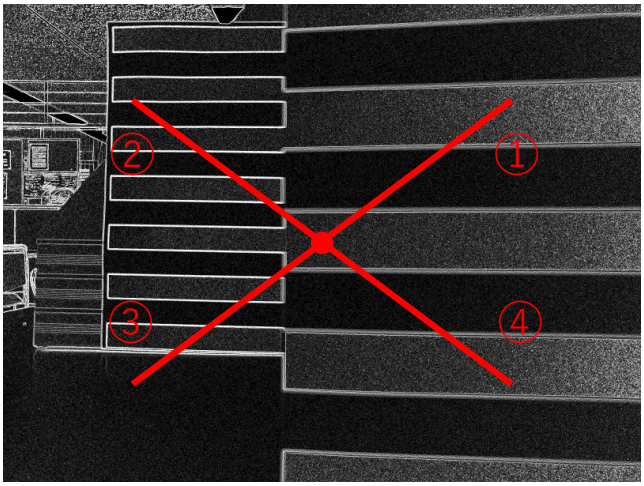


図 6: エッジ画像上の測定領域

の変化画像を図 8 にシーン 8 の線分②の測定平面におけるエッジの変化画像を図 9 に示す．

図 8 において，5 つの高エッジ点分布領域を通過しており，エッジの変化画像においても 5 点のエッジの推移が確認できる．図 9 では，撮影画像のエッジ画像がシーン 1 とは異なり，高エッジ領域が広い範囲に分布しているため，その領域を横切る，シーン 8 の線分②では，多くのエッジ位置の変化をとらえている．これは，エッジの変化画像においてもその傾向があり，エッジの推移が直線上の分布ではなく，エッジ画像上部において，広がった形状が確認できる．

3.6 エッジの変化画像における Hough 変換

前節で作成した，シーン 1 から 9 における 1 セット 4 枚計 36 枚の画像に対して，パラメータ自動推定手法に従い処理を行った．具体的には，Algorithm2.1 によって算出した 2 値化閾値 th_b を基準に 10 刻みの 5 段階で 2 値化した．2 値化されたエッジの変化画像に 1 セット 4 枚の合計で検出本数が 4 本以上となるまで，投票数閾値 th_v を 10 刻みで下げながら Hough 変換を行った．検出された，2 値化されたエッジの変化画像から検出したそれぞれの直線の傾きから変化率を推定して中央値をとり，5 段階の 2 値化閾値 th_b における各変化率から相加平均をとることで像倍率変化率 R を推定した．

これを 9 つのシーンすべての画像列に行った．エッジの変化画像に Hough 変換を行ったときに，ノイズ等の影響により，直線の傾きの正負が混在することが想定される．一般にフォーカス位置変化に伴う像倍率は，近距離ほど大きくなるため，本手法では，エッジの変化画像における直線上分布の傾きが一方向であると仮定し，正の傾き集合と負の傾き集合に分類し，想定と異なる傾き集合は破棄して，算出した．

4. 結果と考察

各シーンにおける像倍率変化率を表 4 に示す．

表 4: 像倍率変化率 R

シーン	像倍率変化率 $R \times 10^{-4}$
シーン 1	4.59
シーン 2	5.09
シーン 3	4.56
シーン 4	5.22
シーン 5	5.49
シーン 6	5.53
シーン 7	3.82
シーン 8	18.31
シーン 9	4.67
正解値	5.36

ここでの正解値は，筆者らが目視でもっとも適当であると判断した，像倍率変化率を求めた結果である．正解値と比較するとシーン 4，シーン 5，シーン 6 では，おおむね正解値と近い値が推定されていることがわかる．また，シーン 1，シーン 2，シーン 3，シーン 9 においては，正解値との差が $\pm 1 \times 10^{-4}$ 以内にとどまった．以上から，パラメータの自動推定法の有効性は示せた．しかし，シーン 7 およびシーン 8 では，正解値と大きくずれていることから必ずしも，すべての画像列に対して有効ではない結果となった．

ここでは，正解値に近い値となったシーン 4, 5, 6 と大きくずれていたシーン 7, 8 について考察する．本手法では，各 2 値化閾値において，全測定平面（本稿では 4 つ）におけるエッジの変化画像に Hough 変換を行い，合計検出本数が 4 本以上になった最初の投票数閾値 th_v で像倍率変化率の推定を行っているため，少ない本数で決定している．そのため，エッジの測定領域を決める線分が，高エッジ領域を横切っていて，またその周囲に複数の高エッジ領域が存在しない線分で測定できているシーン 4, 5, 6 では正解値と近い値となったのではないかと考えられる．

シーン 7 においては，エッジの変化を測定できるような高エッジ領域を線分が横切っていないことから，エッジの変化画像において 4 枚中 2 枚はエッジの推移がほとんど確

(a) シーン 1 **(b)** シーン 2 **(c)** シーン 3

(d) シーン 4 **(e)** シーン 5 **(f)** シーン 6

(g) シーン 7 **(h)** シーン 8 **(i)** シーン 9

図 7: それぞれのシーンでの撮影画像（$d = 0.0$ 無限遠）

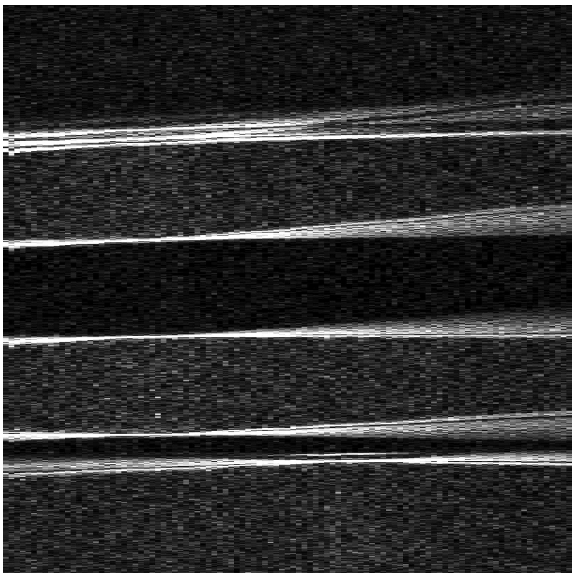


図 8: エッジの変化画像（シーン 1 の線分③）

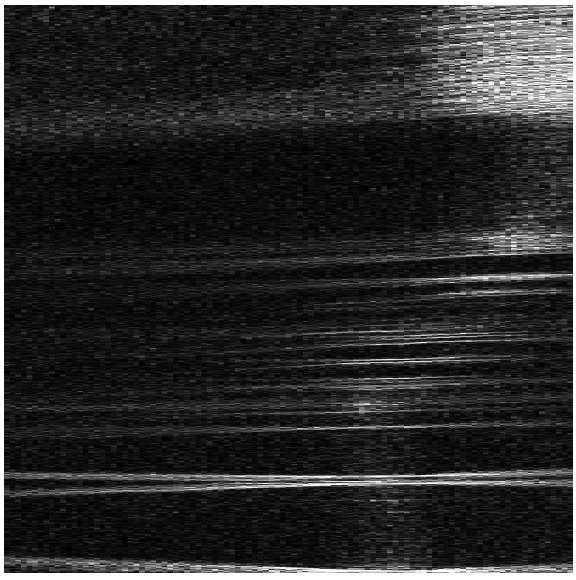


図 9: エッジの変化画像（シーン 8 の線分②）

認できず，ほか 2 枚も若干確認できる程度でしかなかったことが原因であると考えられる．シーン 8 においては，正解との差が 12 $\times 10^{-4}$ 以上ずれていた．考えられる原因として，図 9 より，エッジの推移が直線上に分布している領域は，画像下部にあり，上部では，比較的低い値のエッジが広がった形状となり，直線上の分布ではないことから，検出される直線が下部に集中していた．像倍率変化による画素位置の変化は，画像の中心で最小値となり，画像の隅に行くにつれて単調増加するため，エッジの変化画像において上部の方が，直線の傾きが大きくなることから推定精度が高くなると考えられる．このことから下部に集中していたシーン 8 では，正解値との差が大きくなってしまったのではないかと考えられる．

今後は，エッジ位置の変化を測定する測定平面を設定する線分を，2 つの条件に合うように決定できるアルゴリズムの提案を行う．1 つ目は，高エッジ領域を横切るような線分を決定する．ただし，画像中心付近では，推定精度低下の恐れがあるため，ある程度画像中心から離れた位置の高エッジ領域を横切るような線分とする．2 つ目は，高エッジ領域の周囲（像倍率変化による位置ずれの最大値以内）に複数異なる高エッジ領域が存在しないような線分を決定する．また，エッジの変化画像の上部から検出した直線の傾きのみ使用するといった改善案を検討している．

5. おわりに

本稿では，Android OS 搭載のスマートフォンのカメラを用いてマルチフォーカス画像を撮影し，エッジ画像列によるエッジの変化画像を作成し，Hough 変換に必要な 2 値化閾値 th_b および投票数閾値 th_v を自動推定した上で Hough 変換を行うことで像倍率変化率を推定した．9 つの画像列に対して本手法を適用した結果，3 つのシーンで概ね正解値に近い値が推定され，4 つのシーンにおいて正解値との差が $\pm 1 \times 10^{-4}$ 以内にとどまった．しかし，2 つのシーンにおいて正解値と大きくずれている結果となった．これは，エッジの測定平面を設定する際に決定する線分が，高エッジ領域を横切り，またその高エッジ領域の周囲に複数異なる高エッジ領域が存在していないような線分が正解値に近い値が推定されていた．そのため 2 つの条件にあった線分を決定するアルゴリズムの提案を行う．

参考文献

[1] Hough, P. V. C. , “Method and means for recognizing complex patterns,” U. S. Patent No. 3069654, 1962.
[2] Ballard, Dana H, “Generalizing the Hough transform to detect arbitrary shapes,” Pattern recognition, Vol. 13, No. 2, pp. 111–122, 1981.
[3] T. Darrell and K. Wohn, “Pyramid based depth from focus,” Computer Vision and Pattern Recognition, 1988, Proceedings CVPR ’88, Computer Society Conference, pp. 504–509, 1988.
[4] Watanabe, Masahiro and Nayar, Shree K, “Telecentric optics for computational vision,” European Conference on Computer Vision, pp. 439–451, 1996.
[5] 日浦慎作，松山隆司，“構造化瞳をもつ多重フォーカス距離画像センサ”，Vol. J82-D-II, No. 11, pp.1912–1920, 1999.
[6] S. Pertuz, D. Puig, and M. A. Garcia, “Improving Shape-from-Focus by Compensating for Image Magnification Shift,” 2010 20th International Conference on Pattern Recognition, pp. 802–805, 2010.
[7] R. Senthilnathan, P. Subhasree, R. Sivaramakrishnan, and P. Karthikeyan, “Estimation of sparse depth based on an inspiration from SFF,” 2016 International Conference on Recent Trends in Information Technology (ICRTIT), pp. 1–6, 2016.
[8] Kitano, Kazuhiko and Kobayashi, Aki, “Implementation of DFF Using a Smartphone Camera,” Consumer Electronics (GCCE), 2017 IEEE 6th Global Conference on, pp. 532–534, 2017.
[9] 北野和彦，小林亜樹，“直線検出による像倍率変化の抑制についての一考察 ”，研究報告コンシューマ・デバイス＆システム（CDS），Vol. 2018-CDS-21，No.30，pp. 1–8，Jan 2018.
[10] 「消費動向調査 - 内閣府」 http://www.esri.cao.go.jp/jp/stat/shouhi/menu_shouhi.html(閲覧日 2018 年 4 月 22 日)