# 「ライトフィールドカメラ Lytro における高速なリフォーカスと被写界深度の制御」

2014 年 1 月 27 日

京都産業大学コンピュータ理工学部ネットワークメディア学科
学籍番号：０４５１００
氏名：三宅　健太

# 要約

従来のデジタルカメラはレンズが形成する光の像に忠実なデジタル画像を生成する装置である。撮影時には、被写体にピントを合わせる必要がある。ライトフィールドカメラ Lytro はレンズに入射する光の情報を光線に分解して記録する。その光線の集合であるライトフィールドを処理することで最終画像を生成するため、撮影時にピントを合わせる必要がない。撮影後に Lytro アプリケーションを用いて、任意の位置にピントが合った画像を高速に生成することができる。

本研究の 1 つ目の目的は、Lytro の出力ファイルを用いた高速なリフォーカス処理の手法を推定し、検証することである。Lytro の出力ファイルには、20×20 のデプス画像、ピント位置の異なる複数の JPEG 画像、それぞれの画像情報が格納されている。Lytro はこれらの出力データを用いて高速なリフォーカスを行っていると推定した。推定手法は、デプス画像からピントを合わせる位置の奥行き情報を読み出し、その位置の前後にピントの合った JPEG 画像を 2 枚選択する。そして、その 2 枚の JPEG 画像を加重平均するものである。Lytro アプリケーションと、推定手法を用いたリフォーカス画像の比較を行った。その結果、推定手法と Lytro アプリケーションで、リフォーカス画像に違いを見つけることができなかった。したがって、推定したリフォーカス手法と Lytro アプリケーションの手法は同じであると判断した。

本研究の 2 つ目の目的は、Lytro の出力ファイルを用いて、高速な被写界深度制御機能を実装することである。提案手法では、ピント位置の異なる複数の JPEG 画像を、デプス画像のサイズに合わせて、20×20 のブロックに分割する。そして、デプス画像を 1 画素ずつ読み込み、そのデプス値に最も近い距離にピントが合う画像の該当する部分を出力画像に代入していくものである。この手法を用いて、被写界深度が深い画像を生成することができた。問題点は、提案手法を用いて生成した被写界深度の深い画像はブロック単位で画像を抽出しているために、不自然なブロック境界が発生することがある。

# 目次

# 第１章　序論

デジタルカメラは、レンズが形成する光の像に忠実なデジタル画像を生成する装置である。デジタルカメラはフィルムカメラを代替して広く普及するとともに、スマートフォンのカメラ機能としても利用されている。その結果、だれもが、どこでも、好きなだけ写真撮影を行うことができるようになった。一方、光像に忠実な像を出力する、という意味においては、フィルムカメラと同じ装置である。撮影時には、被写体にピントを合わせる必要がある。ピントがぼけて撮影されてしまった写真は失敗写真であり、後処理による修正が若干は可能であるが、その程度には限りがある。

ライトフィールドカメラ Lytro はピントを合わせる必要がないカメラである。従来のカメラは、撮影時に被写体にピントを合わせる必要がある。ピント合わせとは、被写体がきれいに写るようにレンズと画像センサの位置関係を調節することである。この作業の有無がライトフィールドカメラと従来のカメラの大きな違いである。この作業がなくなるということは、撮影したいシーンに出会ったとき、即座にシャッターを押せるということである。Lytro はピントを合わせる必要がないカメラであるといったが、厳密にいえば撮影後に任意の位置にピントを合わせることができるので、撮影時にピントを合わせる必要がないということである。このように、撮影後にピントを合わせ直すことをリフォーカス処理という。

リフォーカス画像を生成する手法はこれまでにも多く提案されている。代表的な従来手法は、まず撮影時にレンズ口径を絞ることで被写界深度が深い一枚の画像を取得する。次いで、ある特定の距離に存在する被写体以外を、画像処理によってぼかせることで、リフォーカスに類似した結果画像を生成するものである。これに対して、撮影段階に特別な工夫をこらし、中間情報として得られる生データに高度な信号処理を加えることで、これまで不可能であった写真画像を生成する技術が考案された。この技術をコンピュテーショナル・フォトグラフィとよぶ。その代表的な成果の１つが Lytro である。

ライトフィールドカメラの歴史は、スタンフォード大学の Wilburn、Levoy らの研究から始まる。これは図 1.1 に示すように[1]、96 台のカメラを横 12×縦 8 に配列したものである。他には、株式会社ビュープラスから販売されている、図 1.2 の 25 眼カメラアレイシステム ProFUSION25 がある[2]。これは 25 台のカメラが 12㎜ 間隔で並べられている。これら二つは実用的ではない大きさである。それに対して、Lytro はマイクロレンズ方式であり小型で実用的なサイズに作られている。

図 1.1 スタンフォード大学のマルチカメラアレイ[1]

図 1.2 ビュープラス社の ProFUSION25[2]

Lytro は上記のとおり、小型であり、優れた機能を持つカメラである。しかし、従来のカメラで撮影した写真には別の良さが存在する。従来のカメラは、被写体にピントを合わせるだけでなく、レンズの有効口径を調整することによって、被写体がきれいに写る範囲を調整することができる。この範囲のことを被写界深度という。被写界深度を制御することで、視覚効果を利用して遠近感を出すことや、注目箇所を強調することができる。Lytro にはピントを合わせる機構がないが、レンズの口径を絞る機構もない。そこで本研究は、まず、Lytro のリフォーカス機能の方法を推定し、次いで、その方法を利用し被写界深度を制御する手法を研究した。

# 第 2 章　ライトフィールドカメラ Lytro

## 2.1 ライトフィールドの定義

ライトフィールドは三次元空間の光線の集合であり、光線の分布関数によって表現される。これを表す数学モデルをプレノプティック関数と呼び、式(2.1)のように定義する。

$$I = P(X, Y, Z, \theta, \varphi) \qquad \cdots (2.1)$$

ここで、(X,Y,Z)は光線が通過する 3 次元座標、( $\theta$ , $\phi$ )はその位置における光線の向きである。I は光線の強度、明るさを表す。

## 2.2　Lytro によるリフォーカス機能

本研究で用いるライトフィールドカメラ Lytro の概要、内部構造、動作原理などについて説明する。

### 2.2.1　Lytro の概要

Lytro の大きさは、レンズとディスプレイ面が 41㎜×41㎜である。長さは 112㎜である(図 2.1)。内部には多くの部品が含まれている。主レンズが集める光線をマイクロレンズアレイで分解し、画像センサに記録することで、ライトフィールドを取得する。このマイクロレンズアレイを搭載した画像センサが Lytro のハードウエアにおける最も重要な部分である。その拡大写真を図 2.2 に示す。

図 2.1 Lytro 外観[3]

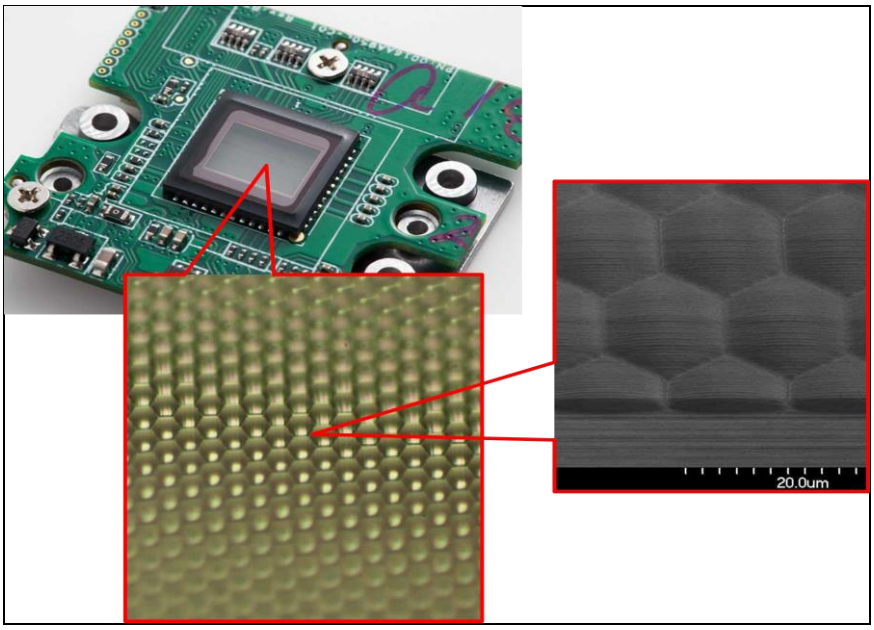


図 2.2 画像センサとマイクロレンズアレイの拡大写真
(写真：豊通エレクトロニクス　ヴァン・パートナーズ
日経エレクトロニクス 2012 年 8 月 20 日号掲載)

Lytro は撮影後にアプリケーションを用いて、高速にリフォーカスを行うことができる。そのアプリケーション画面の一例を図 2.3 に示す。表示される写真の 1 点をクリックすると、その位置にピントが合った画像が生成される。例えば、図 2.3 左のように、手前のトナカイをクリックすると、これにピントが合った画像が生成される。図 2.3 右のように、奥のサンタクロースをクリックすると、これにピントが合った画像が生成される。

図 2.3 Lytro アプリケーションによるリフォーカス画像の例

### 2.2.2　Lytro の動作原理

Lytroはライトフィールドを取得し、そのライトフィールドを処理することで、撮影後に写真画像のピント位置を変更できる。図2.4を用いて、マイクロレンズ方式でライトフィールドを取得する原理を説明する。Aの各点から発した光線はマイクロレンズを通過し、それぞれのマイクロレンズに対応する画素に記録される。マイクロレンズに対応する画素を重ねると、Aにピントを合わせた粗い画像が生成される。Bの位置にある灰色マークに注目すると、この位置を通過する３本の光線は、異なったマイクロレンズを介して撮像素子に記録される。それらの画素値を平均化すると、Bにピントを合わせた写真画像の画素を生成することができる。

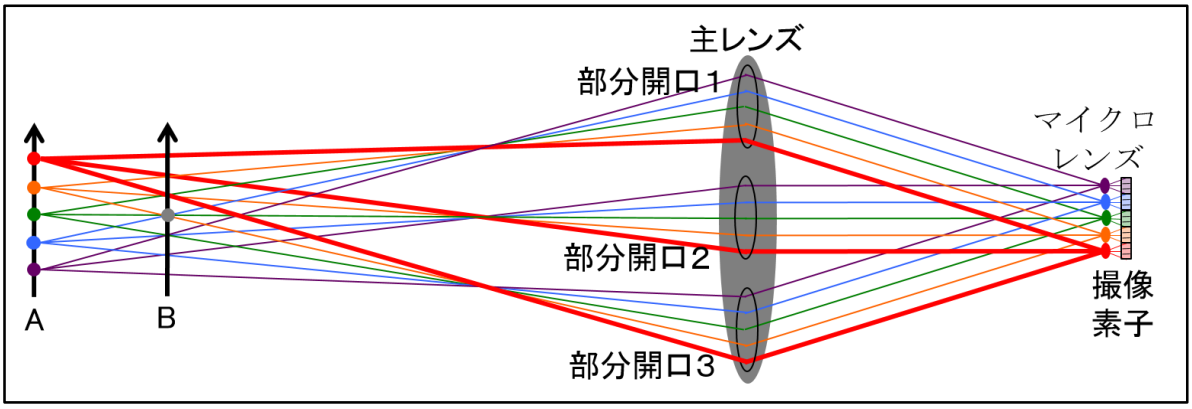


図 2.4　Lytro がライトフィールドを取得する原理

### 2.2.3 ライトフィールドを用いたリフォーカス処理

ライトフィールドを用いたリフォーカス画像の生成手法の一例を図 2.5 に示す。

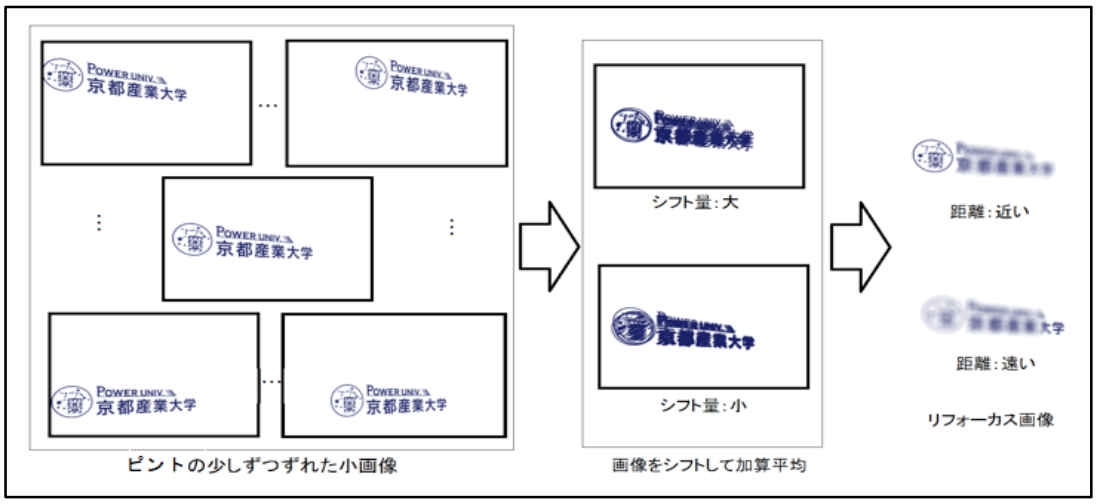


図 2.5 ライトフィールドを用いたリフォーカス画像の生成手法

Lytroは、ライトフィールドを記録したデータを、lfpファイルとして出力する。これを加工すると、図2.5左のように視差が異なる多数の小画像を生成することができる。これらの画像群は、被写体までの距離によって視差が異なっている。これらの画像は視差により少しずつ被写体の位置がずれており、そのずれの大きさは被写体とカメラとの距離によって異なる。これを重ね合わせるために小画像を適量平行移動して加算平均させる。こうすることで、移動量に応じた奥行きに存在する被写体の位置が合うので、その被写体のシャープな像を生成できる。逆にそうでない被写体の像はぼやける。この手法は、画像処理としては複雑なものではないため、比較的高速に実行することができるが、それでも、データ量が大きくなると、リアルタイムで処理することは難しい。

### 2.2.4 Lytro の出力データ

Lytro は 1 回の撮影に対して 2 つのファイルを出力する。1 つのファイルは lfp という拡張子で、Lytro の画像センサが取得する生画像のデータファイルである。もう 1 つのファイルは-stk.lfp という拡張子で、Web でのリフォーカス画像生成に用いる画像情報を保存したファイルである。パソコンの HDD に、図 2.6 のように、ファイル名が同じで拡張子のみが違うファイルが生成される。

第一のファイルは lfp ファイルである。lfp ファイルの内容は、撮影したライトフィールドの生データである。第二のファイルは-stk.lfp ファイルである。-stk.lfp ファイルの内容は Web レンダリングに用いる画像情報である。本研究では、-stk.lfp ファイルを用いた研究を行う。


図 2.6 撮影時に生成される 2 つのファイルの例

-stk.lfp ファイルは、Lytro アプリが lfp ファイルの生データを利用して生成すると考えられる。このように、事前にリフォーカスに必要な情報を生成しておくことにより、高速なリフォーカスを可能としている。-stk.lfp ファイルは、図 2.7 に示す順番で、ファイルに関す

る JSON 記述、20×20 画素のデプス画像、ピント位置の異なる複数の JPEG 画像が格納されている。図 2.7 のセクション 2 のデプス画像、セクション 3 の複数の JPEG 画像は、lfp ファイルに格納された生データを用いて計算していると考えられる。


図 2.7 –stk.lfp ファイルの記述内容と格納順序

続いて-stk.lfp ファイルの内容について詳細に説明する。複数の JPEG 画像は、それぞれの画像情報が-stk.lfp ファイル内に JSON 記述として格納されている。その画像情報は図 2.8 のようなものである。図 2.8 は複数の画像の中の 1 枚に対する情報を抜粋したものである。representation =ファイル形式、width =画像の幅、height =画像の高さ、lambda=画像のピント位置を示している。lambda の値はデプス画像の値と対応している。この lambda の値とデプス値はリフォーカス処理に関係していると考えられる。imageRef は何らかの情報を暗号化しているが、解明できていない。

```
"imageArray" : [
    {
        "representation" : "jpeg",
        "width" : 1080,
        "height" : 1080,
        "lambda" : 2.0610749721527099609,
        "imageRef" : "sha1-3669d312f41b9dfbc4f7a8771bb5c2287461bcd0"
    },
```


図 2.8 Lytro によって生成された JPEG 画像 1 枚分の画像情報

デプス画像は、撮影した写真画像の距離情報を表す 20×20 画素の画像である。図 2.9 の右は、左の写真画像に対応するデプス画像を濃淡で表現したものである。手前に被写体がある部分を黒く、奥に被写体がある部分を白く表現した。

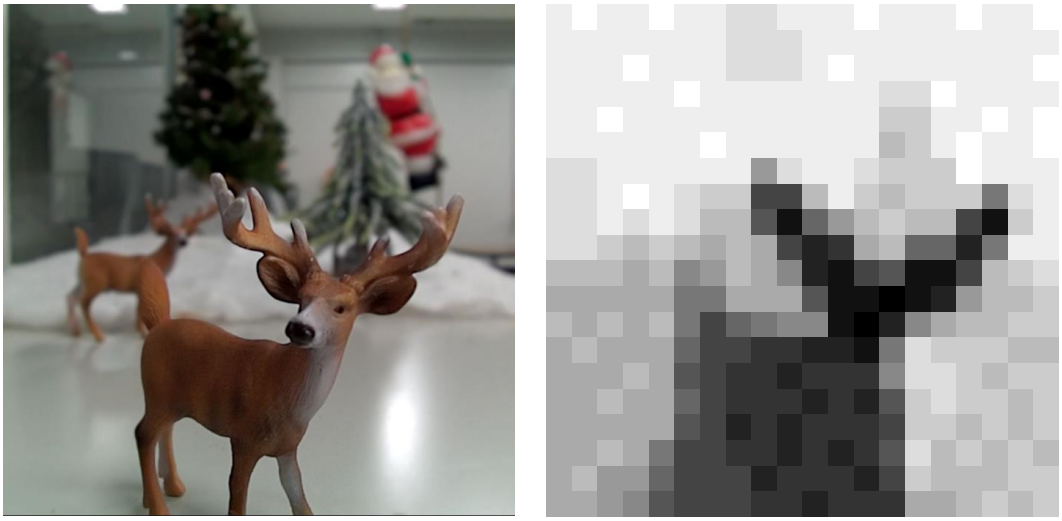

図 2.9 Lytro で撮影した写真画像(左)とデプス画像(右)

複数の JPEG 画像は、Lytro が取得する生データであるライトフィールドを処理することで生成した 1080×1080 画素の画像と推察される。この JPEG 画像は、ほぼ均等な間隔でピント位置が異なる画像であり、画像の枚数は被写体との距離によって変化する。一例を図 2.10 に示す。それぞれの画像のピント位置は、メジャーの目盛 2 付近、人形、ポスターであると確認できる。左上の画像のピント位置は、被写体が存在しないごく近くの位置であると思われる。

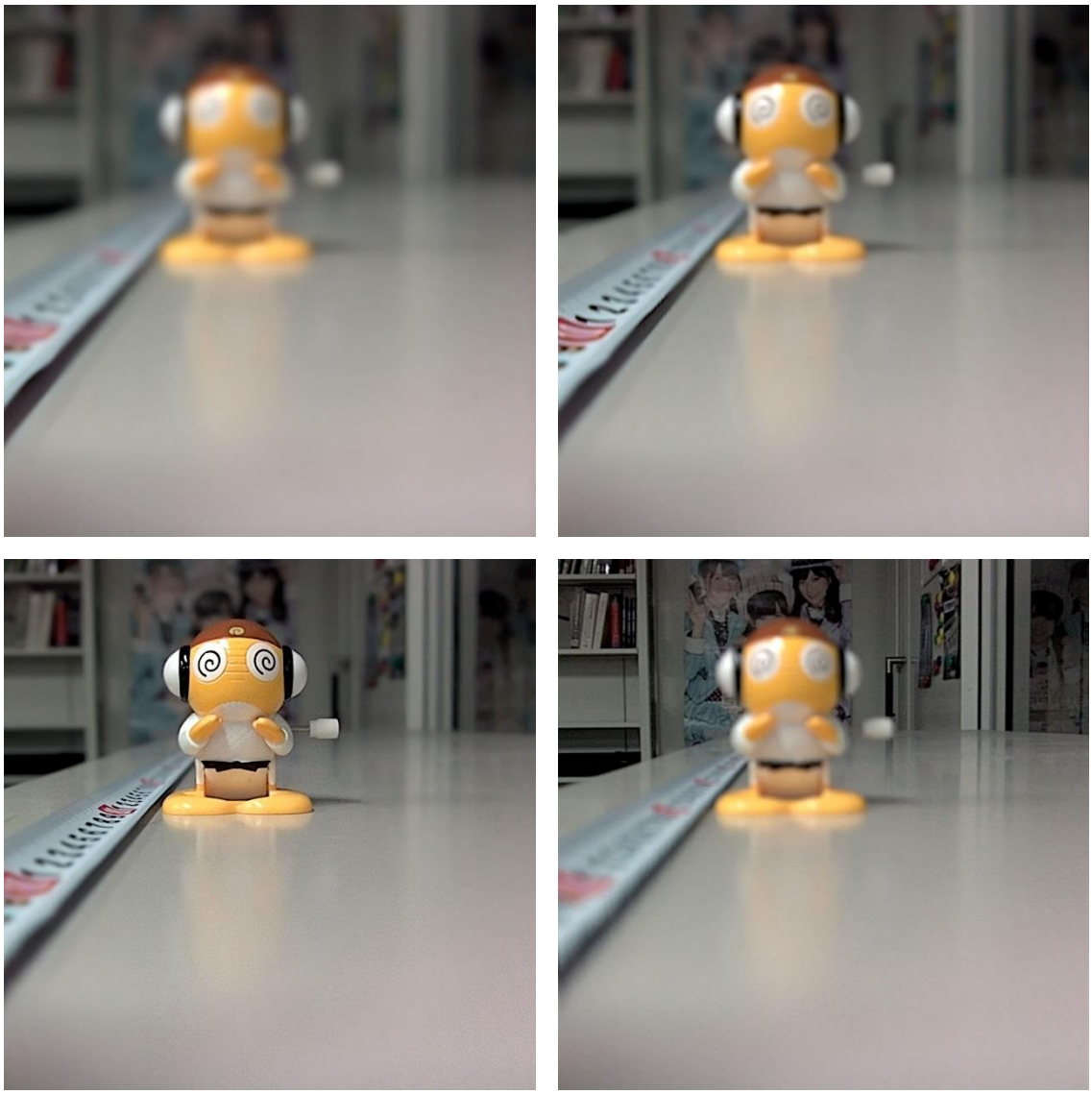

図 2.10 -skt.lfp ファイル内のピント位置の異なった JPEG 画像

# 第 3 章　Lytro の出力ファイルを用いた高速なリフォーカス

## 3.1　高速なリフォーカス処理

Lytro で撮影した画像は、Lytro アプリを用いることで高速にリフォーカスすることが可能である。また、Lytro 社の Web サイトが提供する機能を用いて、同様のことができる。このときにリフォーカス画像を生成する手法は、ライトフィールドから直接リフォーカス画像を計算しているのではない、と考えた。ライトフィールドを直接計算する手法はリアルタイムに処理ができほどに軽くはない。Lytro は別の手法を採用していると考えられる。

### 3.1.1　推定手法の説明

-stk.lfp ファイルには、ピント位置が異なった複数の JPEG 画像と 20×20 画素のデプス画像が格納されている。Lytro アプリが、これらのデータを用いて高速にリフォーカス画像を生成していることは確実である。そこで、このデータを用いてリフォーカスする手法を考えた。Lytro アプリに使われていると推定したレンダリング手法を、図 3.1 を用いて説明する。

推定手法

Step 1

表示されている写真画像をマウスでクリックし、その位置に対応するデプス画像の値を読み出す。

Step 2

その距離に従って、適切な２枚の JPEG 画像を選択する。リフォーカス画像は２枚の JPEG 画像の中間的な画像であるから、各画像に適切な重みを設定する。

Step 3

設定した重みで２枚の JPEG 画像を加重平均する。

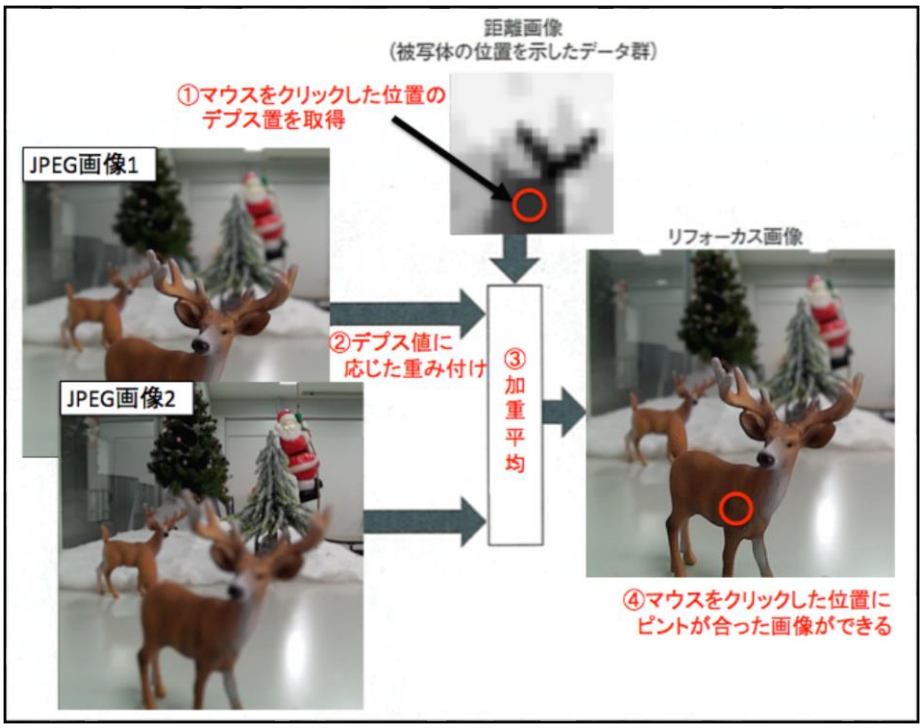


図 3.1 リフォーカス画像の生成手法

上記の Step 2、Step 3 について詳細に説明する。Step 2 は-stk.lfp ファイルに記された画像情報を用いている。各 JPEG 画像の lambda 値を参照し、Step 1 で読み込んだデプス値を間にとるような２つの lambda 値を選択する。例を図 3.2 に示す。

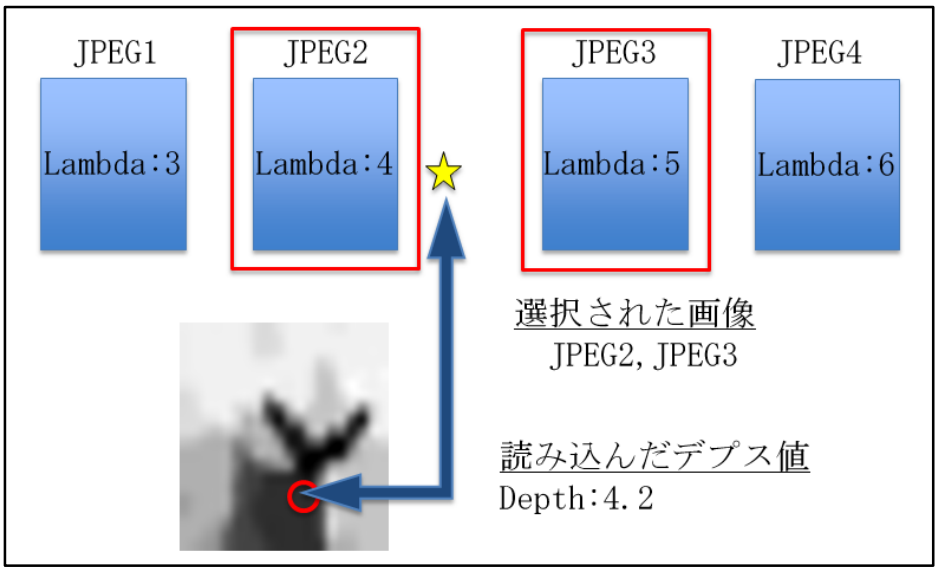


図 3.2 デプス値に対する画像選択の例

例のデプス値は 4.2 である。これは、JPEG2 の lambda=4 と、JPEG3 の lambda=5 の中間であるため、この２枚の画像を選択する。Step 3 で設定する重みは、先ほどの２つの lambda 値とデプス値の関係から設定する。例を図 3.3 記述する。

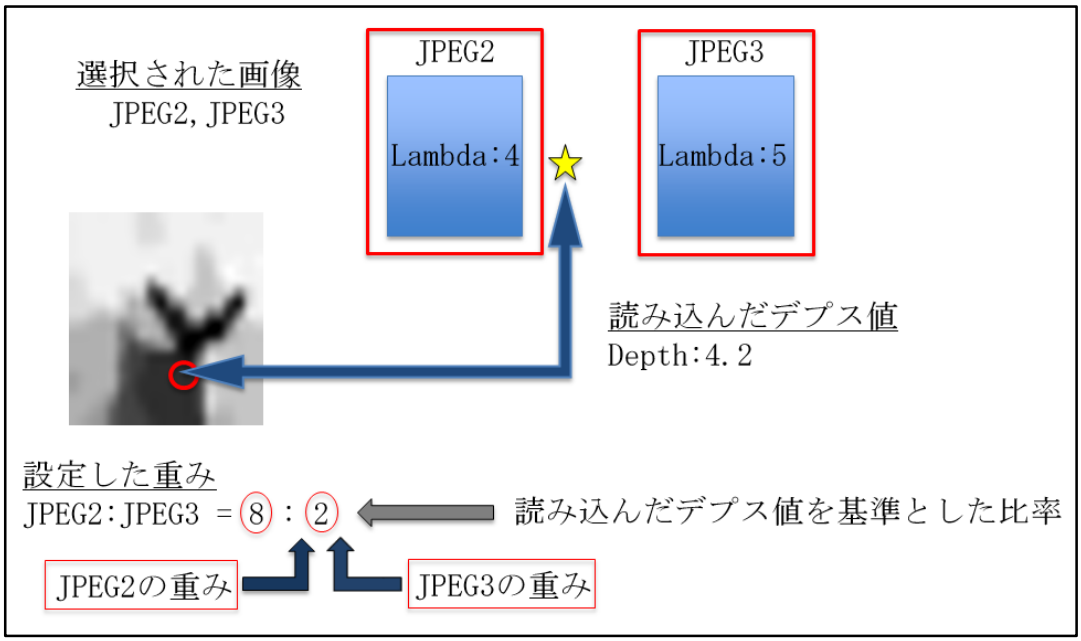


図 3.3 重み設定の例

選択されたデプス値 4.2 は、２つの lambda 値では 4 に近い値である。その関係を比率で計算すると、JPEG2:JPEG3=8:2 となり、例では、JPEG2 の重みは 0.8、JPEG3 の重みは 0.2 と設定する。

## 3.2　高速なリフォーカス処理の実験と検証

本節では、Lytro アプリケーションと推定手法の動作を比較し検証する。検証方法は、リフォーカス結果の画像を視覚的に判断することとする。推定手法を実装したプログラムは、Lytro アプリ同様、マウスクリックでリフォーカス位置を設定できるようにした。しかし、比較実験の際は、スライドバーによって少量ずつデプスの値を変動させた。それによって加重平均に設定する重みを細かく設定し、より近い条件で比較できるようにした。比較した写真を以下の図 3.4、図 3.5 に示す。

図 3.4 Lytro アプリケーションを用いて生成したリフォーカス画像

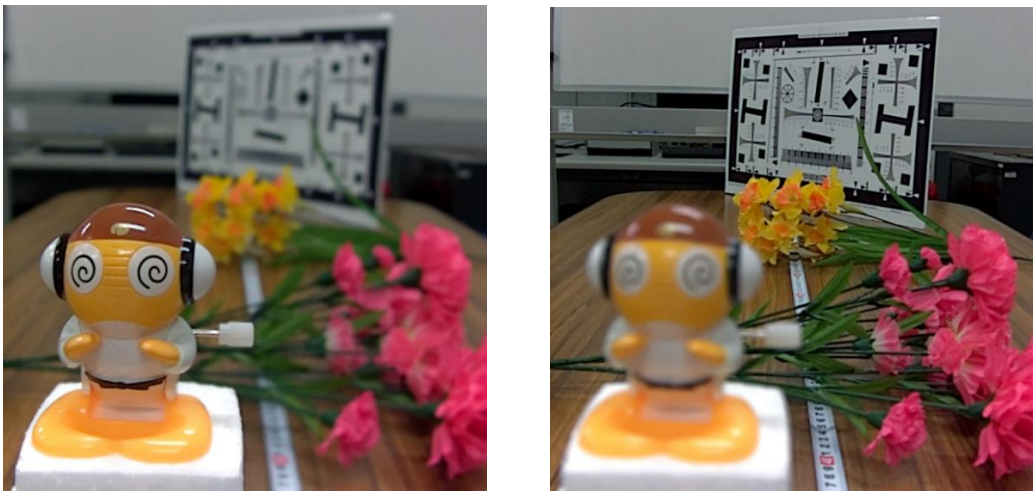

図 3.5 推定手法を用いて生成したリフォーカス画像

図 3.4 は Lytro アプリケーションで生成したリフォーカス画像であり、左の画像は手前の人形にピントを合わせ、右の画像は後ろの解像度チャートにピントを合わせた。図 3.5 は提案手法で生成したリフォーカス画像であり、左の画像は図 3.4 左の画像に可能な限り近づくように調整し、右の画像は図 3.4 右の画像に可能な限り近づくように調整した。このレベルでは図 3.4 と図 3.5 の左の画像同士、および右の画像同士で差を見つけることができない。詳細な比較を行うために、中間位置にあるピンクの花にピント合わせる。その結果画像は図 3.6、図 3.7 である。続いて、その側にあるメジャーの目盛に注目する。メジャーの目盛部分を拡大した画像を図 3.8、図 3.9 に示す。これらを目視で精査することによりリフォーカス精度の検証を行う。図 3.8 と図 3.9 を比較したが、違いを見つけることはできなかった。そのため推定手法によるリフォーカスが正常に動作しており、Lytro アプリケーションと同じ動作をしていると判断した。

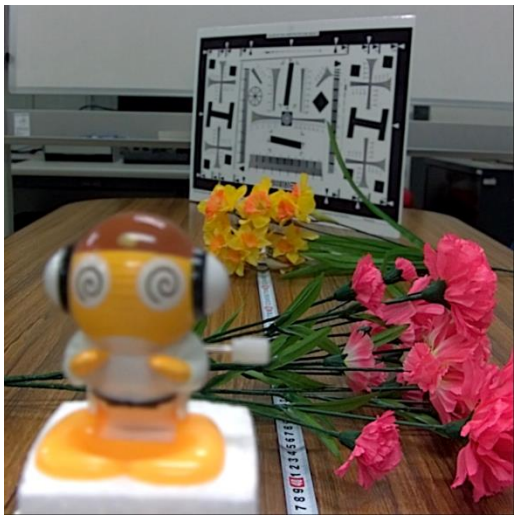

図 3.6 Lytro のリフォーカス処理結果

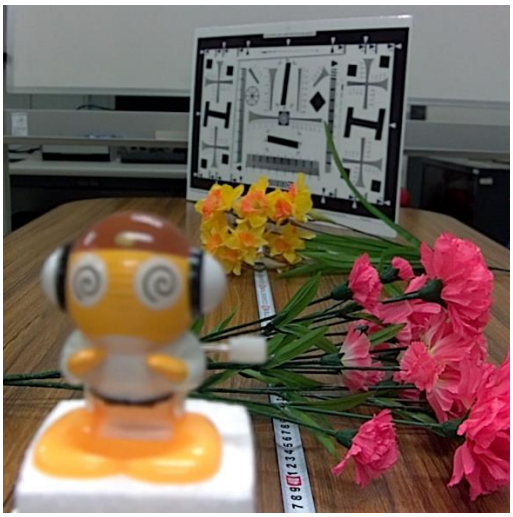

図 3.7 推定手法のリフォーカス処理結果

図 3.8 Lytro の処理結果の拡大図

図 3.9 推定手法の処理結果の拡大図

# 第 4 章　Lytro の出力ファイルを用いた被写界深度制御

## 4.1　被写界深度の定義

被写界深度はピントが合っているとみなすことができる範囲のことである。被写界深度の調節により遠近感を出すことや、注目箇所を強調することができる。被写界深度の浅い写真、深い写真の例を図 4.1 に示す。左の写真は被写界深度の浅い写真である。きれいに写っている範囲が狭く、手前のコスモスが浮き上がっているように見える。右の写真は被写界深度の深い写真である。カメラに近い牛と遠い富士山が共にきれいに写っており、シーン全体の雰囲気をうまく伝えることができる写真となっている。

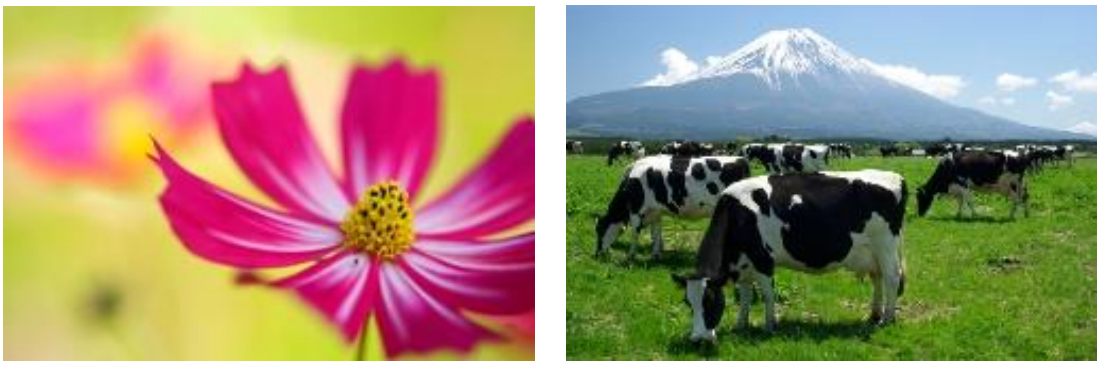

図 4.1　被写界深度の浅い写真(左)、深い写真(右) [4]

被写界深度はレンズの有効口径と焦点距離によって決まる。有効口径が大きいほど、また、焦点距離は短いほど、被写界深度は浅くなる。逆に、有効口径が小さいほど、また、焦点距離が長いほど、被写界深度が深くなる。従来のカメラは、このようにして被写界深度を調節している。その関係を表 4.1 にまとめた。

表 4.1　被写界深度と有効口径、焦点距離の関係

| 有効口径 | 焦点距離 | 被写界深度 |
|---|---|---|
| 大きい | 長い(望遠) | 被写界深度が浅い<br>(ピントの合う範囲が狭い) |
| 小さい | 短い(広角) | 被写界深度が深い<br>(ピントの合う範囲が広い) |

## 4.2　被写界深度の制御手法

Lytro には、被写界深度を調節する機能は搭載されていない。そこで-stk.lfp ファイルを用いた被写界深度の制御手法を提案する。提案手法を、図 4.2 を用いて説明する。

提案手法(被写界深度を深くする場合)

Step 1

デプス画像のサイズに合わせて、全ての JPEG 画像を 20×20 のブロックに分割する。

Step 2

デプス画像の 1 画素目の値を読み込む。そのデプス値と最も近い lambda 値を持つ JPEG 画像を選択する。

Step 3

選択した JPEG 画像の 1 ブロック目の分割画像を出力画像に代入する。Step 2 に戻り全画素分繰り返す。

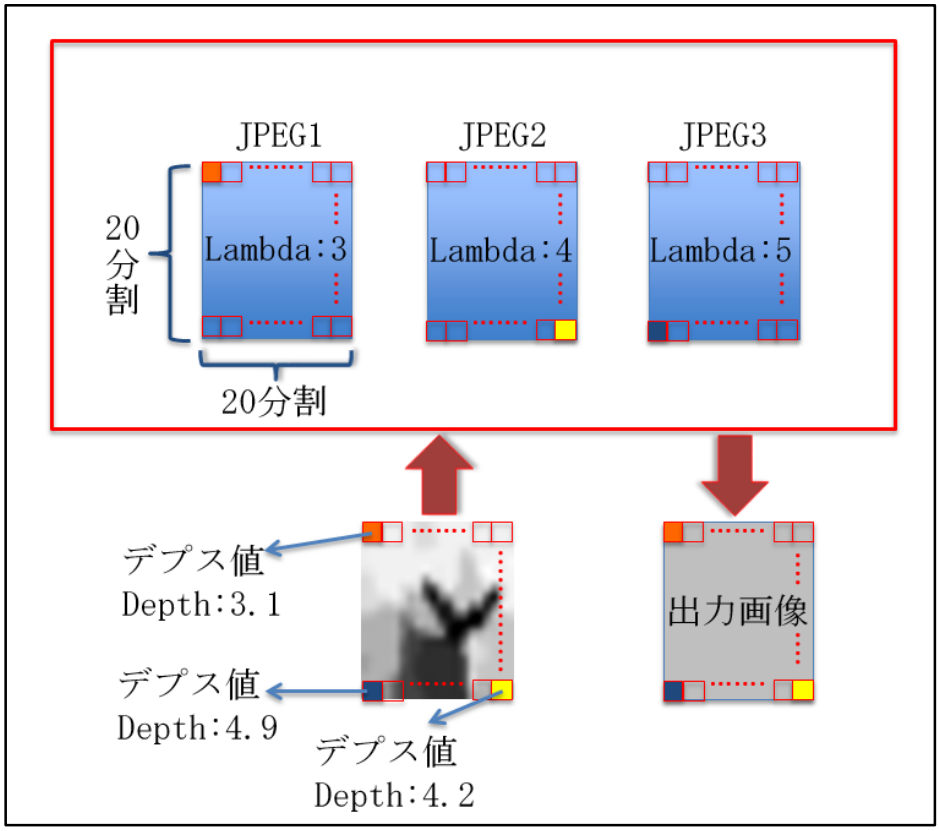


図 4.2 被写界深度の深い画像を生成する手法

図 4.2 の例では、左上のオレンジの画素は、デプス値が 3.1 なので最も近い lambda 値をもつ JPEG1 が選択される。そしてオレンジのブロック内の全画素が出力画像に代入される。同様の計算によって、紺色部分は JPEG3 の対応するブロックから、黄色部分は JPEG2 の対応するブロックが出力画像に代入される。

## 4.3　被写界深度制御の実験と検証

本節では、提案手法を用いて生成した被写界深度の深い画像の精度を検証する。精度の検証方法として、生成した出力結果を目視で確認し、問題点を見つけることとする。元画像と出力結果の例を図 4.3 に示す。

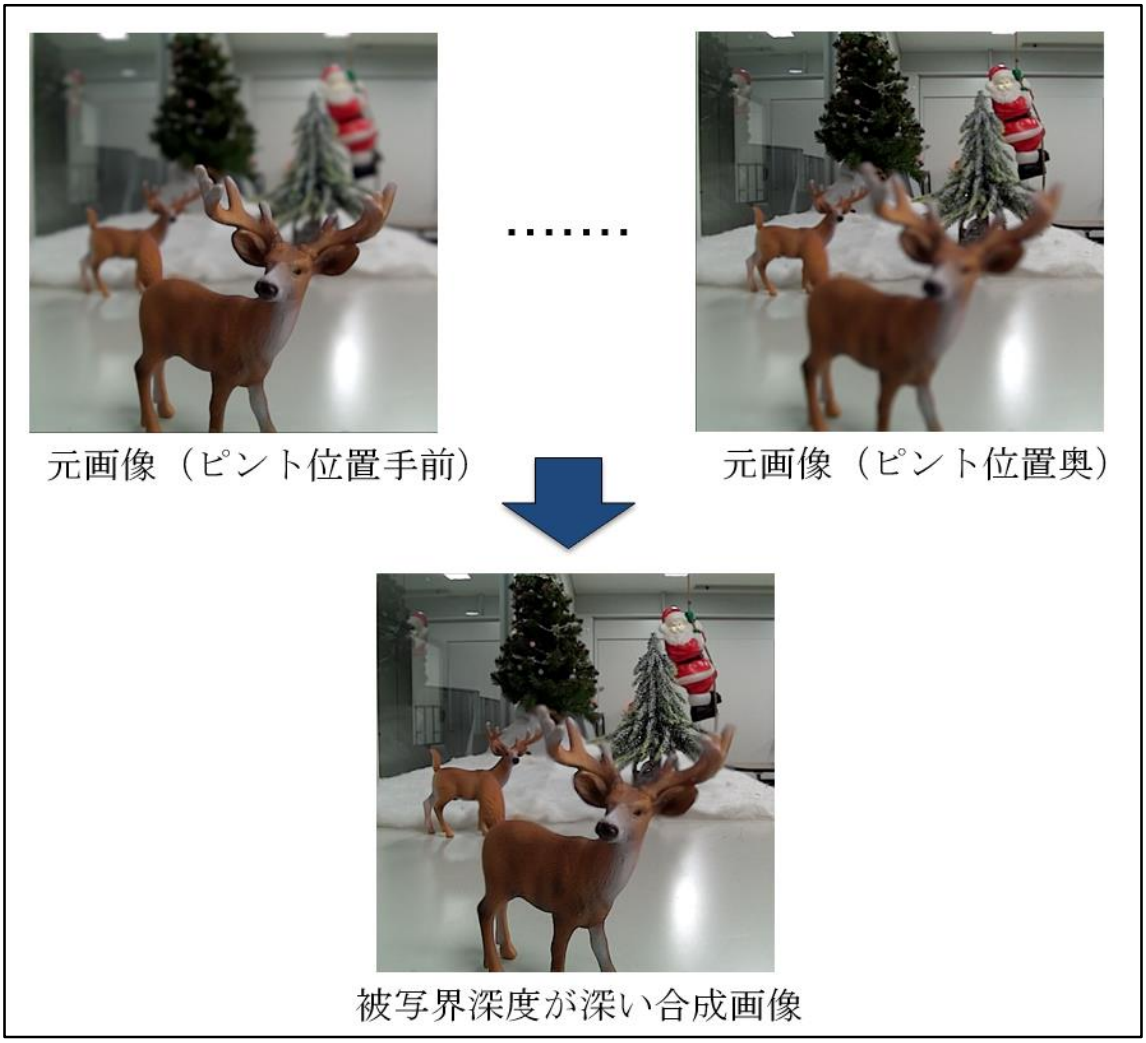


図 4.3 元画像とそれらから生成した被写界深度の深い画像

上記の図 4.3 を見て分かる通り、ピント位置の異なる多数の元画像と生成された画像を見比べると、生成された画像は手前と奥の被写体の両方が鮮明に写っている。これによって、被写界深度の深い画像が生成できたと言える。しかし、出力結果を拡大し確認してみると、少し不自然な部分もあった。不自然な部分の拡大を図 4.4 に示す。図 4.4 を見ると、急激に画質の差が生じている部分がある。この問題点の原因は、JPEG 画像を 20×20 のブロックに分割し、ブロック単位で合成しているためである。一つのブロックに異なる距離にある被写体が写ると、ブロック境界が生じやすい。

図 4.4 生成した被写界深度の深い画像の拡大図

# 第 5 章　結論

## 5.1　成果

Lytro アプリケーションに用いられているリフォーカス手法を推定し、検証した。その結果、推定手法と Lytro アプリケーションでリフォーカス画像に違いを見つけることができなかった。いくつかのテストデータで検証したが、特に気になる問題点は見当たらなかった。したがって、推定したリフォーカス手法と、Lytro アプリケーションの手法は同じであると判断した。

また、被写界深度を深くする手法を提案した。用いたアルゴリズムに大きな問題はないが、被写体中で奥行きが急減に変化する部分では、出力画像に不自然なブロック状の境界が見られた。そのような場所には、平均化するような処理を加えなければいけないと思われる。

## 5.2　課題

課題として、本文中では言及しなかったが、-stk.lfp ファイルに記録されている複数の JPEG 画像の順番と、それに対応する画像情報の順番が異なるため、手作業で順番を直さなければならなかった。そこを自動化することが望ましい。

被写界深度の制御において、深度を深くする処理は概ね実装できたが、浅くする機能を実装できていない。この機能が実現できていないため、写真画像の一部分を強調することができておらず、今後の課題である。

## 謝辞

本研究を進めるにあたって、様々なご指導を頂きました蚊野浩先生に感謝の意を表します。

## 参考文献

[1] http://graphics.stanford.edu/projects/lightfield/tiled-side-view-cessh.jpg

[2] http://www.viewplus.co.jp/product/camera/profusion25.html

[3] Lytro社のホームページ, https://www.lytro.com/

[4] PIXTA Channel, http://pixta.jp/channel/

[5] 蚊野浩,「あらゆるところに賢いカメラ」,日経エレクトロニクス,8b月20日号,P42-47

## 付録

開発した主なプログラム

| | |
|---|---|
| lytroIO.cpp | Lyto の出力ファイルの一つである-stk.lfp ファイルを入力し、その内を解析する。その結果を depth.tif、jpegdata[N]、sec1.txt として保存する。ここで、depth.tif は 20×20 画素のデプス画像であり、jpegdata[N]はピント位置の異なる N 個分の画像データである。sec1.txt は、N 個分の画像情報をテキストに書き出したものである。 |
| lytro_refocus.cpp | LytroIO.cpp が書き出した depth.tif、jpegdata[N]を利用し、リフォーカス画像を生成するプログラム。OpenCV のマウスイベント機能を利用して対話的な操作を実装した。ウインドウ上に表示した撮影画像に対して、ピントを合わせたい位置をクリックすると、その位置にピントが合ったリフォーカス画像を同じウインドウに表示する。 |
| test_slider.cpp | lytro_refocus.cpp と同様のリフォーカス処理をスライドバーによって行う。それによりピント位置を少量ずつ変化させることが可能になり、精度検証に用いた。加えて、被写界深度を深くする処理をスライドバーによる操作で行うことが可能である。 |