コンピュータ理工学特別研究報告書

題目　ライトフィールドカメラ Lytro を用いた
任意視点画像の生成

学生証番号　０４４９３０

氏　名　菱沼　宏亮

提出日　平成 26 年 1 月 28 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

従来のデジタルカメラはレンズが形成する光像をデジタル画像として記録する装置である。その画素値はさまざまな方向から入射する光線の積分になっており、光線の方向に関する情報が失われている。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ことができ、その光線の集合、すなわちライトフィールドを処理することで、さまざまな画像を生成することができる。任意視点画像も、そのような画像の一つである。本研究の目的は、ライトフィールドカメラ Lytro の出力ファイルを用いて任意視点画像を生成することである。

Lytro はマイクロレンズアレイと画像センサを組合せることでライトフィールドを取得する。Lytro が任意視点画像を生成可能である原理は、全てのマイクロレンズに対して同じ位置にある画素が、同じ方向の光線を記録しているという性質である。この性質によって、マイクロレンズに対して同じ位置の画素を抽出し、マイクロレンズの配列にしたがった再配列した画像は、主レンズの部分領域からシーンを観察した画像になる。マイクロレンズは 50 個程度の画素をカバーしており、上記の手順で生成した 50 個程度の画像は、主レンズの異なる部分領域からシーンを観察した画像になる。これらの画像の集合が任意視点画像の基になる。このように生成した画像集合は、マイクロレンズに対する画素の位置によって、色と明るさが異なっていた。それを修正するために、ColorChecker の灰色部分における RGB 値がそろうように信号値を補正した。その結果、全体的な色のバランスと明るさが大幅に改善された。

本研究の機能を Lytro アプリと比較した。本研究の結果画像には部分的に筋状のにじみや青みが残っている。また、任意視点画像の表示機能において、比較対象は滑らかな表示を実現しているのに対し、本研究による表示は滑らかさに欠けていた。

目次

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おり、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。

米国 Lytro 社が開発したライトフィールドカメラ(以下 Lytro)は、撮影時にピントを合わせる必要がない。[2]Lytro(図 1.1)は、撮影後に専用のソフトウェアを使用し、任意の位置にピントを合わせる事ができるカメラである。Lytro は、従来のカメラと異なり、3 次元空間を飛び交う光線を記録できるようになった。光像よりも多くの情報を含む光線を利用することで、従来のデジタルカメラでは不可能であった画像の生成が可能になる。Lytro は任意の位置にピントを合わせることができるとともに、任意視点画像も生成することができる。Lytro における任意視点画像は、主レンズ口径内の任意の位置からシーンを撮影した画像である。視点が変化すると、それに応じてシーンの見え方が異なる。

初期のライトフィールドカメラとして、スタンフォード大学の Marc Levoy らの研究がある[1]。これは、図 1.2 のように縦 8 個、横 12 個に配列された 96 台のカメラから構成されたものである。また、株式会社ビュープラスが開発した ProFUSION 25(図 1.3)がある。これは、25 個のカメラを並べることで複数の撮像系を有している。これらの方式をカメラアレイと呼ぶ。一方 Lytro は、第 2 章で述べるように、内部の画像センサの直前にマイクロレンズを配置したマイクロレンズアレイ方式である。

図 1.1 ライトフィールドカメラ Lytro

図 1.2 スタンフォード大学のカメラアレイ方式によるライトフィールドカメラ

図 1.3 ビュープラス社の ProFUSION 25

Lytro は、取得したライトフィールドに対するソフトウェア的な処理によってピント位置を自由に変えることができるだけでなく、任意視点画像を生成することができる。これも従来のカメラでは不可能なことである。それをどのように処理しているのか、ということについて興味を持った。そこで本研究では、任意視点画像を生成する手法を推定し、Lytro アプリケーションと比較することで推定した手法の正当性を検証した。

2 章 ライトフィールドカメラ Lytro

2.1 Lytro の概要

Lytro は、ライトフィールドを取得し、それに対する画像処理によって最終画像を生成する装置である。ここで、Lytro の構造を説明する。Lytro は 41mm×41mm×112mm の四角柱形状であり、比較的、コンパクトである。図 2.1 は Lytro の分解図である。Lytro 内部に、ライトフィールドを取得するための CMOS イメージ・センサ(以下画像センサ)と、その直前にマイクロレンズアレイ(図 2.2)が搭載されている。マイクロレンズアレイを構成する個々のレンズをマイクロレンズとよび、その直径は 14μm である。

図 2.3 は、画像センサ、マイクロレンズアレイ、カラーフィルタを断面から見た図である。画像センサの保護ガラスは、マイクロレンズアレイの機能を兼ねており、上面は平坦であり、底面がハニカム状のマイクロレンズアレイとなっている。また、画像センサの画素表面にカラーフィルタが設置され、カラー情報を記録することができる。

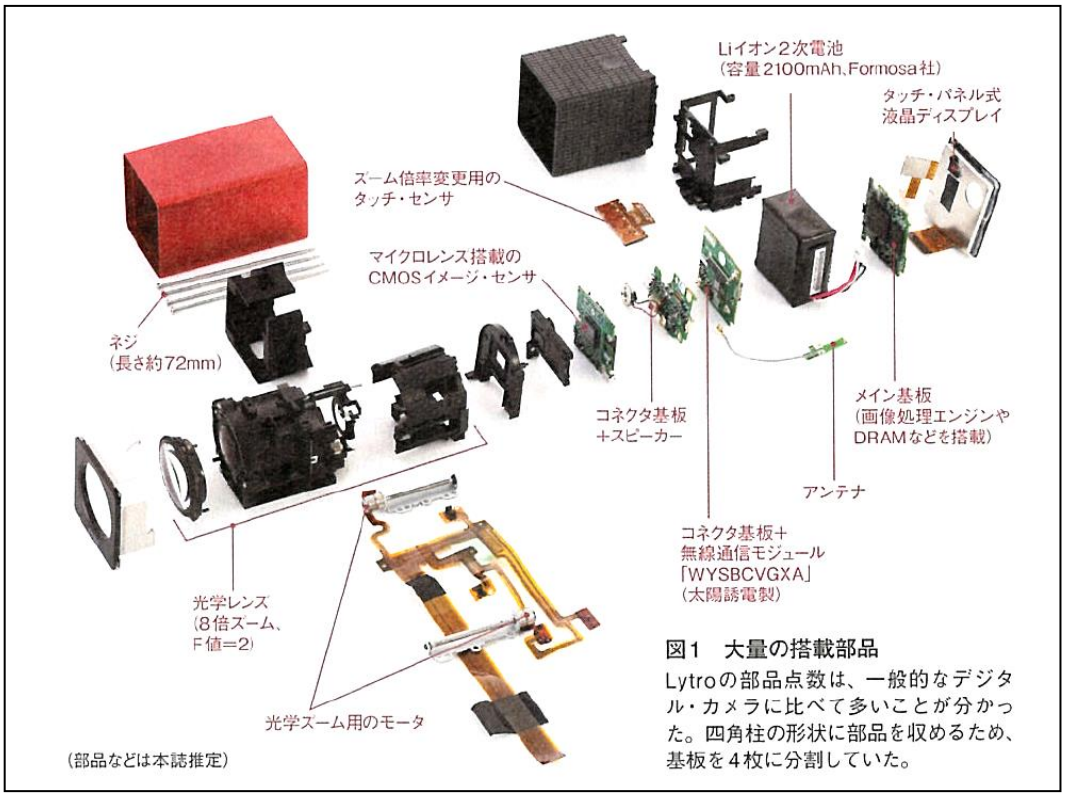

図 2.1 Lytro の分解図

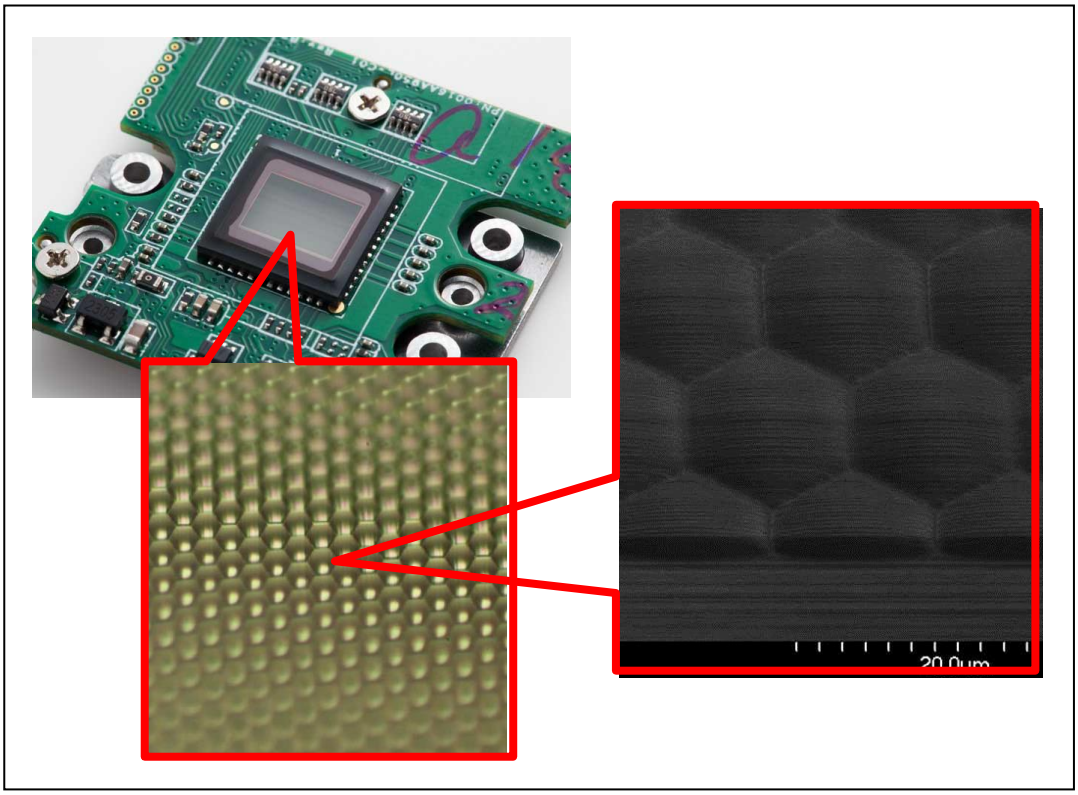

図 2.2 CMOS イメージ・センサとマイクロレンズアレイ、その一部の拡大図

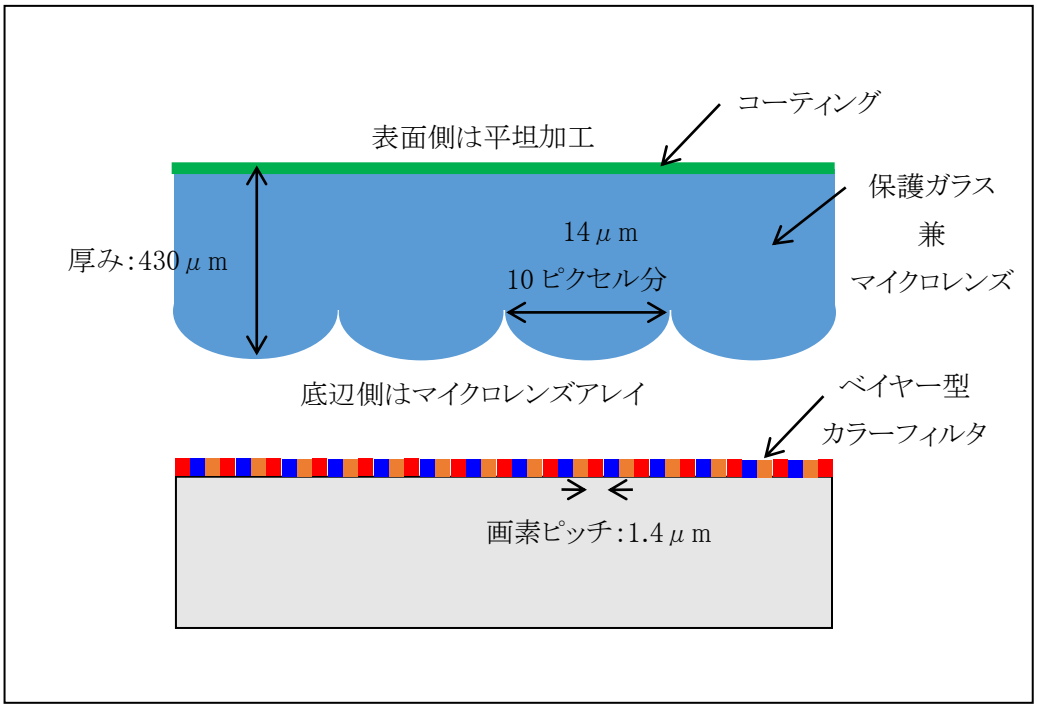

図 2.3 画像センサ部の断面構造

2.2 ライトフィールドの定義

ライトフィールドは、3 次元空間における光線の分布を表す。これを表す数学モデルをプレノプティック関数とよび、式(1)のように表現する。

$$I = P(X,Y,Z,\theta,\varphi) \cdots (1)$$

ここで、X,Y,Z は光線が通過する 3 次元座標、θ , ϕ はその位置における光線の方向を示す角度である。I は光線の強度を表す。光線のカラー情報を表す場合には、I に代えて RGB 値を与える。

続いて、ライトフィールドの取得手段について説明する。ライトフィールドは 2 つの方法によって取得することができる。第 1 はカメラアレイによる方法である。ピンホールカメラは、空間の 1 点を通過する光線集合を水平・垂直の角度方向にサンプリングする装置である。これを平面上でアレイ的に並べる事で、ライトフィールドを取得できる。第 2 はマイクロレンズを用いる方法である。その詳細は後述するが、主レンズが集める光線をマイクロレンズで分解する事でライトフィールドを取得できる。

2.3 ライトフィールドの取得

Lytro が用いる画像センサは、3280×3280 画素である。この直前を 330×380 個のマイクロレンズアレイがハニカム構造で覆っている。図 2.4 に、画像センサとマイクロレンズアレイの関係、およびその一部を拡大したものを示す。マイクロレンズアレイを構成する個々のレンズをマイクロレンズとよぶ。

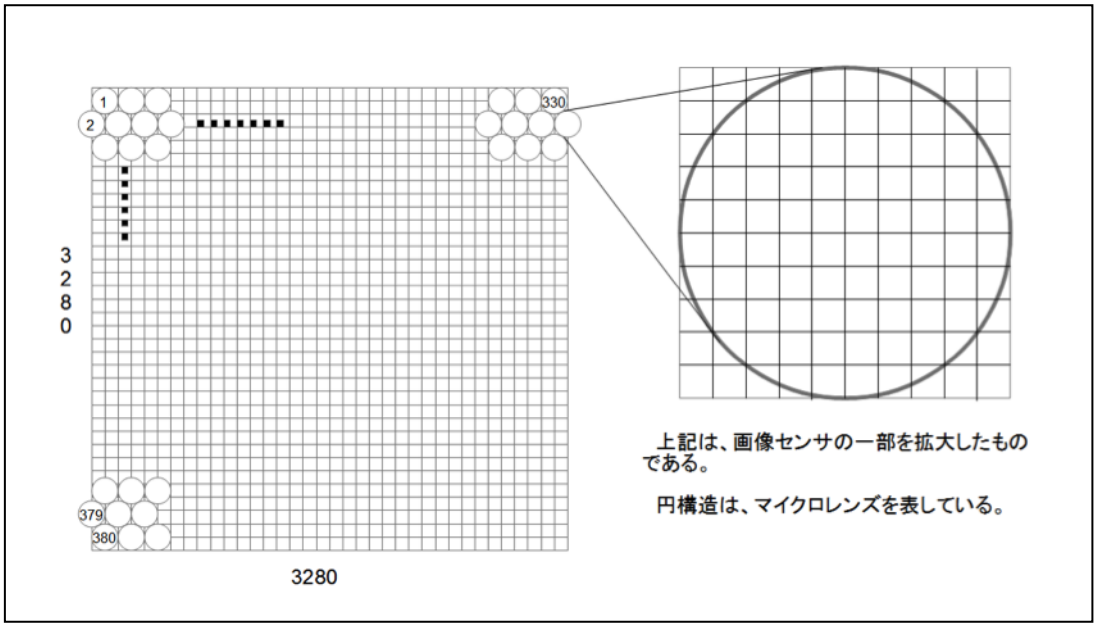

図 2.4 画像センサとマイクロレンズの関係

2.4 Lytro の出力ファイル

Lytro で撮影した画像を Mac に保存すると、lfp と-stk.lfp という拡張子の 2 つのファイルが生成される。lfp ファイルは、ライトフィールドの生データを保存したファイルで、5 つのセクションに別れている。第一セクションは、16 バイトのヘッダである。データ解析には不要となる部分である。第二セクションは、ファイルの概要に関する JSON 記述が書かれている。第三セクションは、生データが格納されている領域である。生データは各画素 12 ビット×3280×3280 の 16MB である。Lytro では、3 バイトに 2 画素のデータを格納している。画素値は原色ベイヤー配列の色フィルタで分解された RGB のいずれかの値である。第四セクションは、生データに関する JSON 記述で、画素値の範囲、色変換係数、撮影パラメータなどが書かれている。第五セクションは、画像センサと製品番号の JSON 記述が書かれている。

-stk.lfp ファイルには、Web レンダリングに用いる画像情報が保存されている。ファイルの概要に関する JSON 記述や、20×20 のデプス画像などが格納されている。

3 章 任意視点画像の生成

3.1 任意視点画像生成の原理

Lytro がライトフィールドを記録する原理を、図 3.1 を用いて説明する。A にある被写体から発した光線は、主レンズで屈折しマイクロレンズ位置で焦点を結ぶ。そして、光線はマイクロレンズを通過し、各画像センサの画素に記録される。図 3.1 の主レンズ上の部分開口 1 に注目する。A の各点から発した光線の中で部分開口 1 を通過したものは、ここで屈折し、各色のマイクロレンズを通過し、画像センサの一番下の画素に記録される。従って、各色の一番下の画素を集めることで部分開口 1 から撮影した画像を生成することができる。同様に部分開口 2 に注目する。A の各点から発した光線の中で部分開口 2 を通過したものは、ここで屈折し、各色のマイクロレンズを通過し、画像センサの真ん中の画素に記録される。マイクロレンズが N 個の要素を覆うとき、上記のようにマイクロレンズに対して同じ位置にある画素を集めてできる N 個の画像は、N 個の部分開口から撮影した N 個の画像となる。

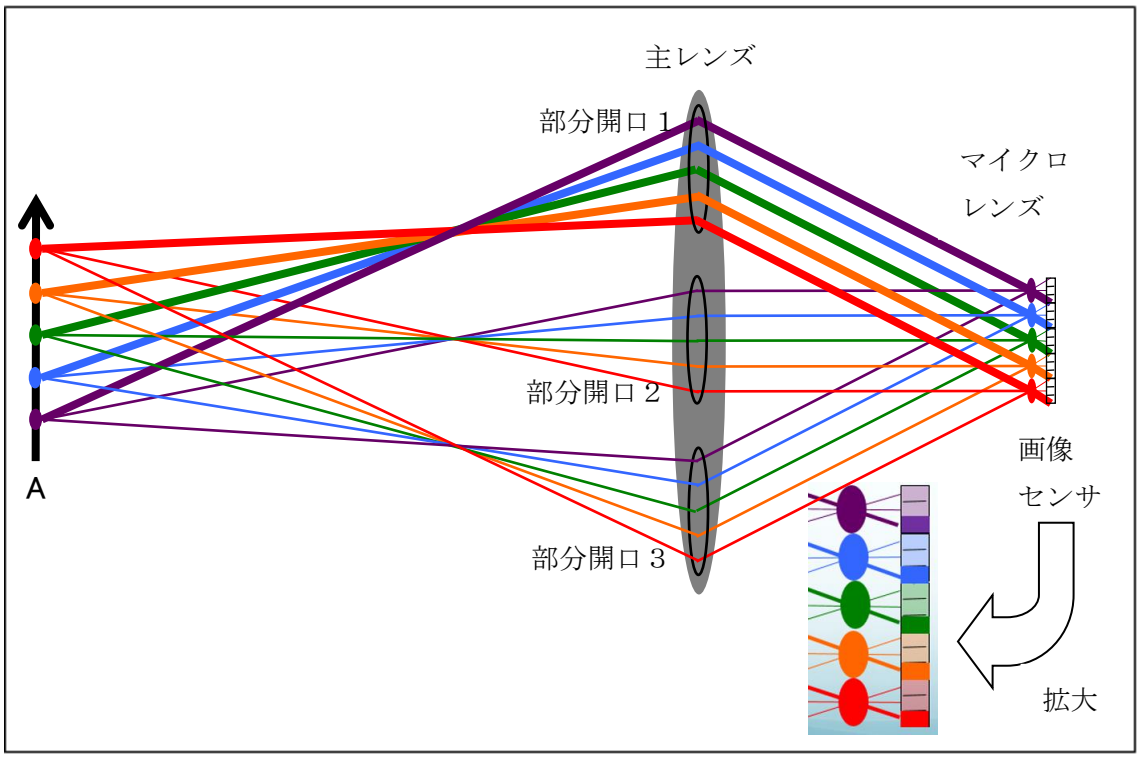

図 3.1 ライトフィールドを記録する原理[3]

3.2 任意視点画像の生成手順

任意視点画像の生成を行う手順を図 3.2 に示す。まず Lytro の出力ファイルである lfp ファイルを読み込む。読み込んだ lfp ファイルをベイヤー型カラーフィルタで RGB カラー画像へデモザイクする。最後にデモザイクした RGB カラー画像を、マイクロレンズに対して同じ位置にある画素を抽出して並べ直すことで視差画像群へ分解を行う。分解した視差画像群をユーザの指示に従って、適切な任意視点画像を生成する。

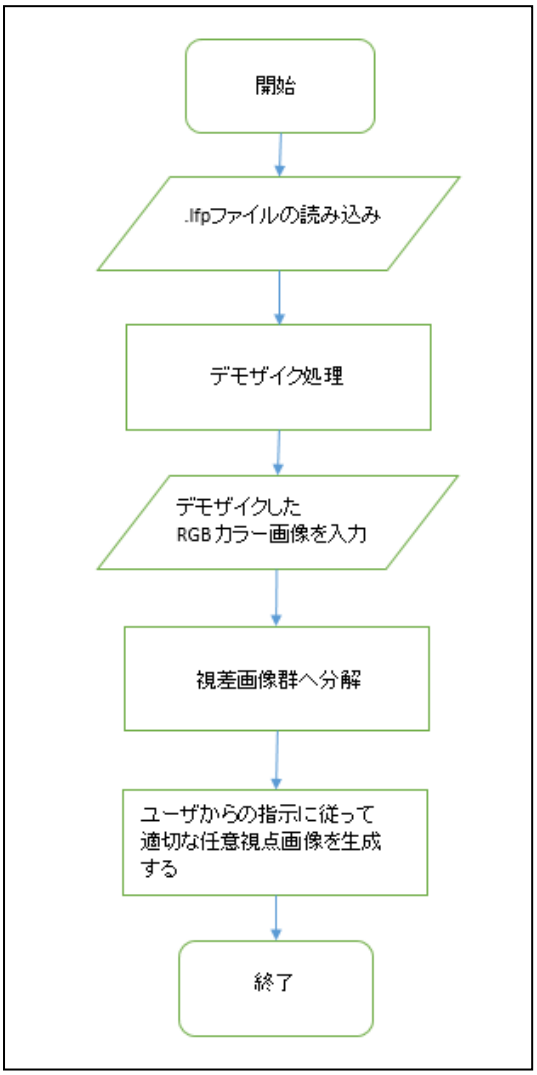

図 3.2 任意視点画像の生成手順

3.2.1 lfp ファイルの読み込み

Lytro で撮影すると図 3.3 のような 2 つの出力ファイルが生成される。この内、lfp ファイルを解析することで任意視点画像を生成する。

lfp ファイルには、画像センサが取得する生データが保存されている。生データは画素あ

たり 12 ビット×3280×3280 の 16MB である。画素値は、原色ベイヤー配列の色フィルタで分解された RGB いずれかの値である。

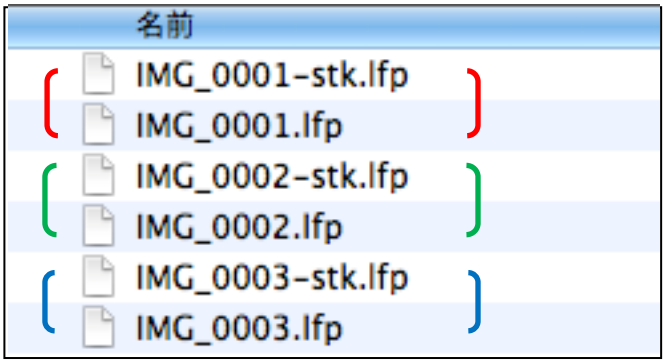

図 3.2 Lytro の出力ファイル

3.2.2 デモザイク処理

単板の画像センサを用いるデジタルカメラの画像処理において、カラーフィルタで色分解した一枚の濃淡画像を RGB カラー画像に変換する処理をデモザイクとよぶ。lfp ファイルは、ライトフィールドを記録した生データであるが、画像センサの直前にカラーフィルタが設置されているため、色分解された濃淡画像にもなっている。そこで、この濃淡画像をデモザイクすることで、カラー画像に変換する。図 3.3 に、Lytro が用いる RGB ベイヤー型カラーフィルタの配列を示す。

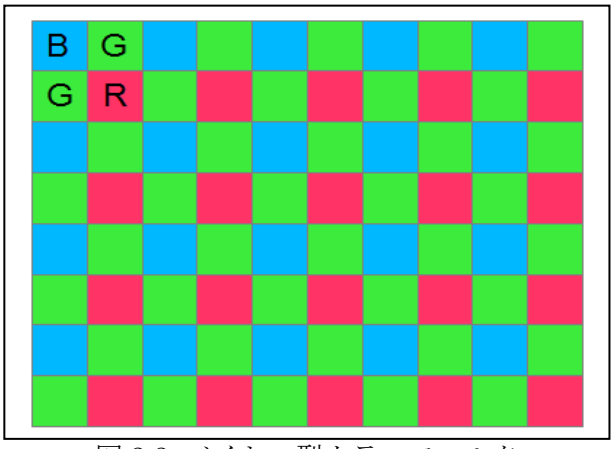

図 3.3 ベイヤー型カラーフィルタ

3.2.3 視差画像群への分解

3.1 で述べたように、Lytro はマイクロレンズに対して同じ位置にある画素を抽出して並び直すことで、主レンズの部分領域からシーンを観察した画像を生成することができる。図 3.5 でその処理手順を説明する。図 3.4 は、3280×3280 画素の画像センサとそれをカバ

ーするマイクロレンズアレイの一部を拡大したものである。異なるマイクロレンズの同じ位置にある画素に注目する。図 3.4 上に示すように、黄色の画素を取り出し、順を変えることなく 330×380 に配列することで 1 枚の画像を生成する。この画像は、主レンズの一部からシーンを撮影した画像に相当し、本論文では視差画像とよぶ。1 個のマイクロレンズの直径は約 10 画素分あるため、この図では 100 枚近くの視差画像を生成できるように見える。実際には、マイクロレンズ境界付近の画素は光線を検出できないため、マイクロレンズが正しくカバーする画素は 50 個程度である。したがって、約 50 枚の視差画像を生成することができる。生成された視差画像は 330×380 画素であるが、縦・横を揃えるため、660×660 画素にバイキュービック変換を行う。このように生成した画像群を視差画像群とよぶ。

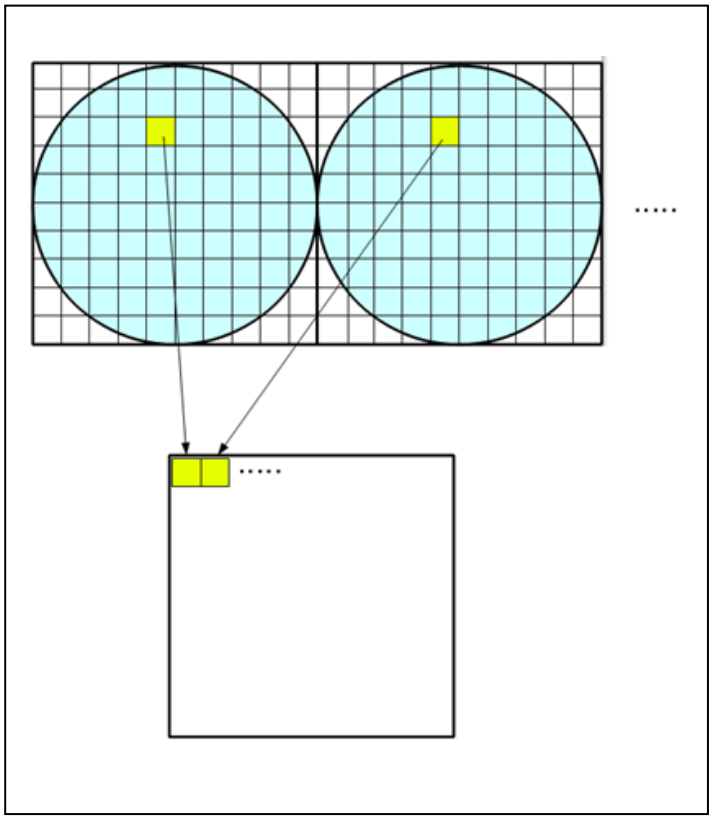

図 3.4 視差画像の生成法

3.3 生成の各段階で得る画像

Lytro の出力ファイルの一つである lfp ファイルを読み込み、3.2 節で述べた手順を、順次、実行するプログラムを開発した。そのプログラムが生成する画像について説明する。まず、lfp ファイルを解析し、画像センサの生データを 3280×3280 画素のカラーフィルタアレイ画像として tif フォーマットで出力する。図 3.5 左は生成されたカラーフィルタアレイ画像の例である。右はその拡大図となっている。拡大した図の円構造はマイクロレンズの構造に対応する。カラーフィルタアレイ画像は、画素ごとに 1 つの色しか持たないため

濃淡画像として出力される。そこで、ベイヤー型のカラーフィルタの構造を用いたデモザイク処理によってRGB画像に変換する。図3.7はデモザイクしたRGB画像である。

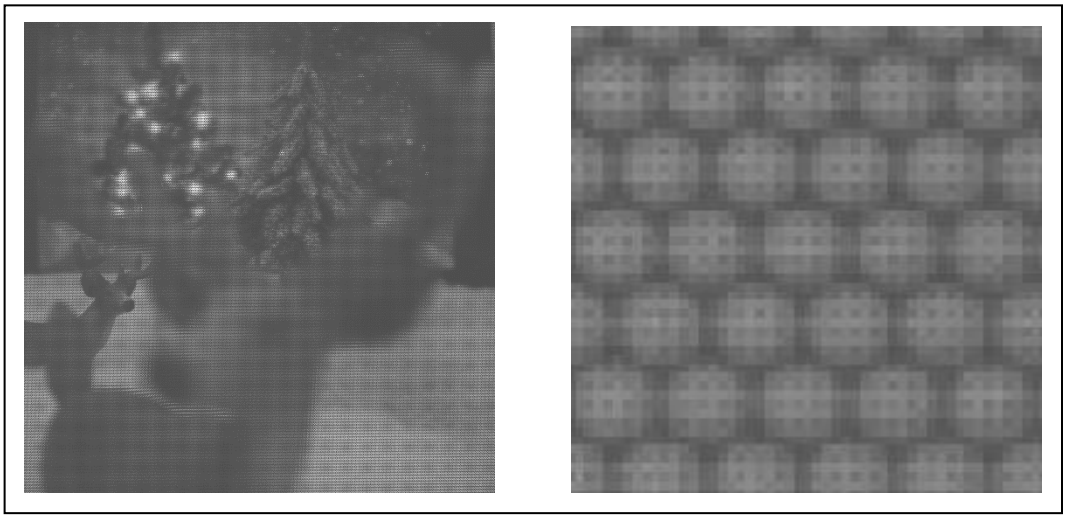

図3.5 濃淡画像として表示したカラーフィルタアレイ画像

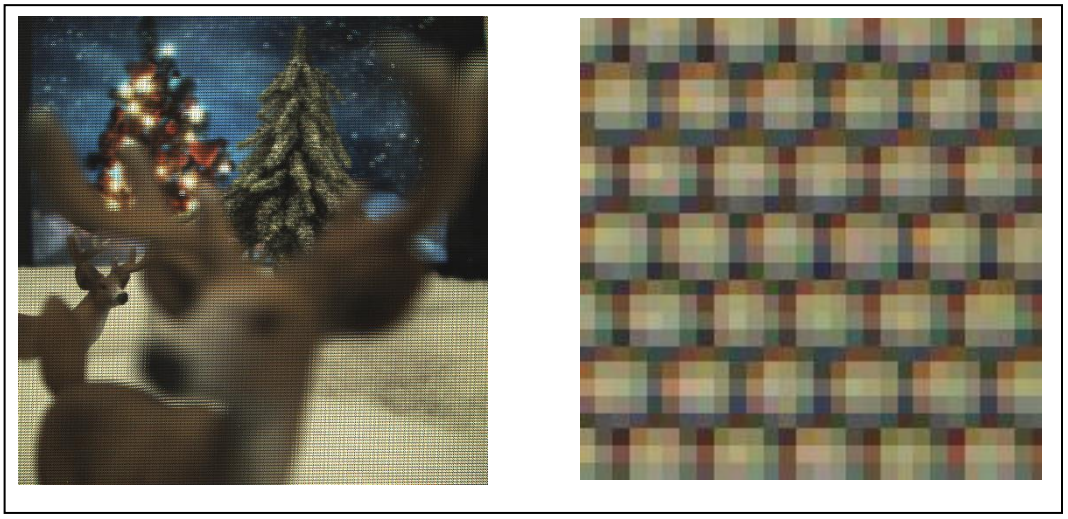

図3.6 カラーフィルタアレイ画像をデモザイクしたRGB画像

続いて、生成された図3.6を入力とし、マイクロレンズに対して同じ位置にある画素を抽出・配列し視差画像群を生成する。この時、マイクロレンズアレイと画像センサの画素は完全には整列していないため、マイクロレンズに対して同じ位置の画素値を求めるためにバイリニア補間を用いた。図3.7にマイクロレンズに対する画素位置と、その位置の画素で生成される視差画像の例を示す。

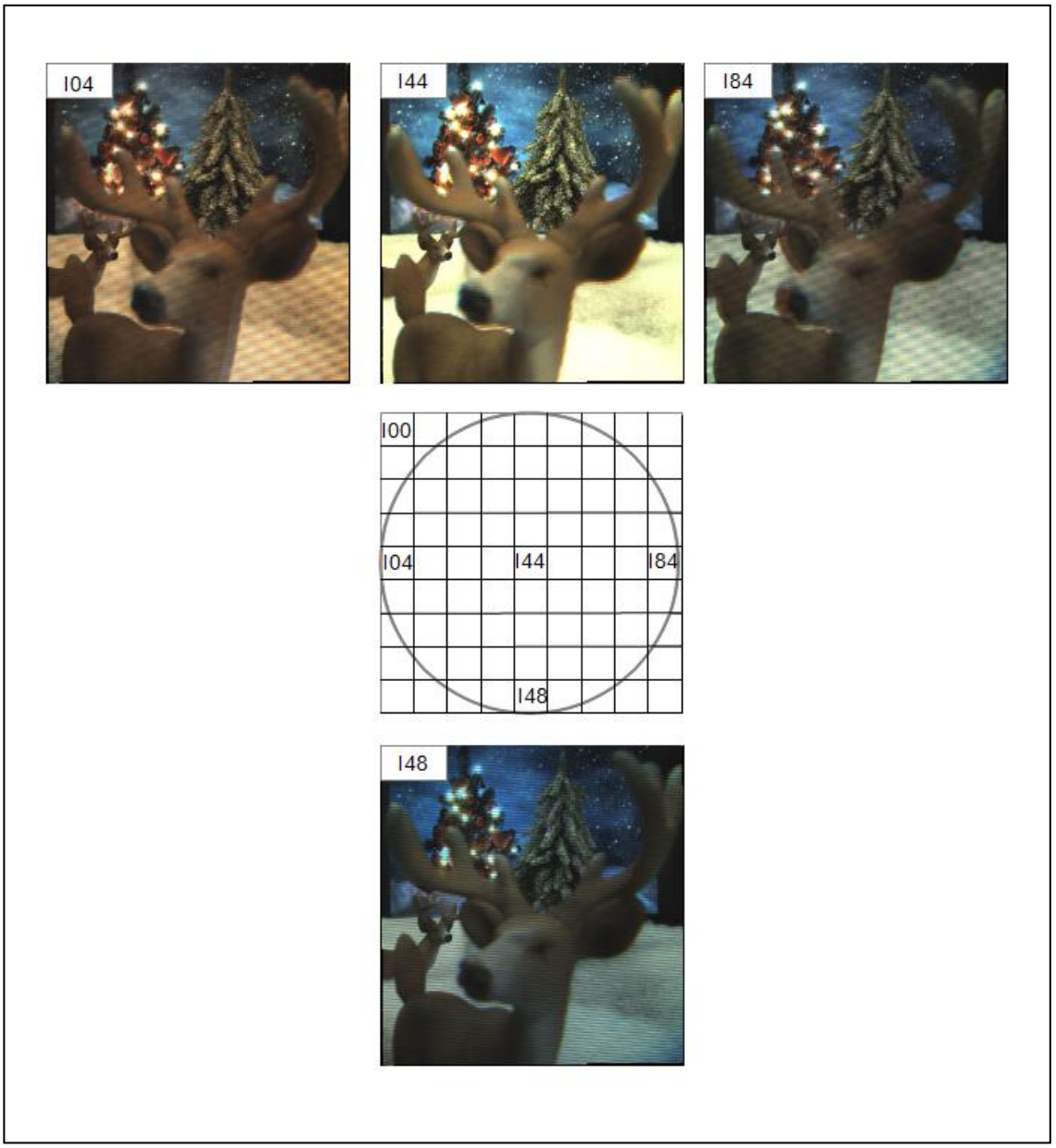

図 3.7 マイクロレンズの各位置に対する視差画像

図 3.7 中央に示す円構造と格子は、一つのマイクロレンズとそれがカバーする画素を示している。3.2 節で述べた処理手法を用いて視差画像を生成すると図 3.8 の結果が得られた。画像からは、視差の有無は判別しづらいが、I44 と I48 に注目する。I48 は、手前のトナカイの頭と木の間に雪が映っていないが、I44 では映っていることがわかる。従って視差を持つ画像の生成に成功したことが分かる。

3.2.3 節で述べたように、生成される視差画像には光線を正しく検出するものと検出しないものがある。図 3.9 がその比較画像になる。左図は、図 3.8 において I00 に位置する画素を取り出し、並べ直した視差画像である。右図は、I44 に位置する画素を取り出し、並べ直

した視差画像である。I00 は光線を正しく検出していないため、画像にずれが見られる。また、全体的に筋のようなものが見えている。

図 3.8 光線を正しく検出できない視差画像と検出できた視差画像

任意視点画像の元になる視差画像群を生成する手法について説明し、具体的な画像を例示した。図 3.8 に示すように、マイクロレンズの中央に位置する画素から構成される視差画像は、比較的鮮明で色のバランスも良い、それに対して、マイクロレンズの周辺に位置する画素で構成される視差画像は不鮮明で暗く、色のバランスも崩れている。したがって、これらの問題を修正する必要がある。また、画素の中には光線を正しく検出できていないものも存在する。任意視点画像の生成において、これらの画素を利用しないように配慮する必要もある。

4 章 視差画像群の色収差補正

4.1 視差画像群に生じる色収差

3 章で説明したように、視差画像を構成する画像間で明るさと色にばらつきが存在する。これらは、マイクロレンズに入射する光線の方向によって、光量と色が変化することが原因である。本論文では、これらの問題をマイクロレンズの色収差とよぶことにする。

これらの問題を修正するために、X-rite 社の ColorChecker(図 4.1)を用いて、色信号を補正する方法を検討した。ColorChecker は縦 4 色、横 6 色の 24 色のカラーパッチからなり、本研究では一番下の White～Black の 6 色を用いる。説明を分かりやすくするために、各色に図 4.1 に示す名前をつけた。

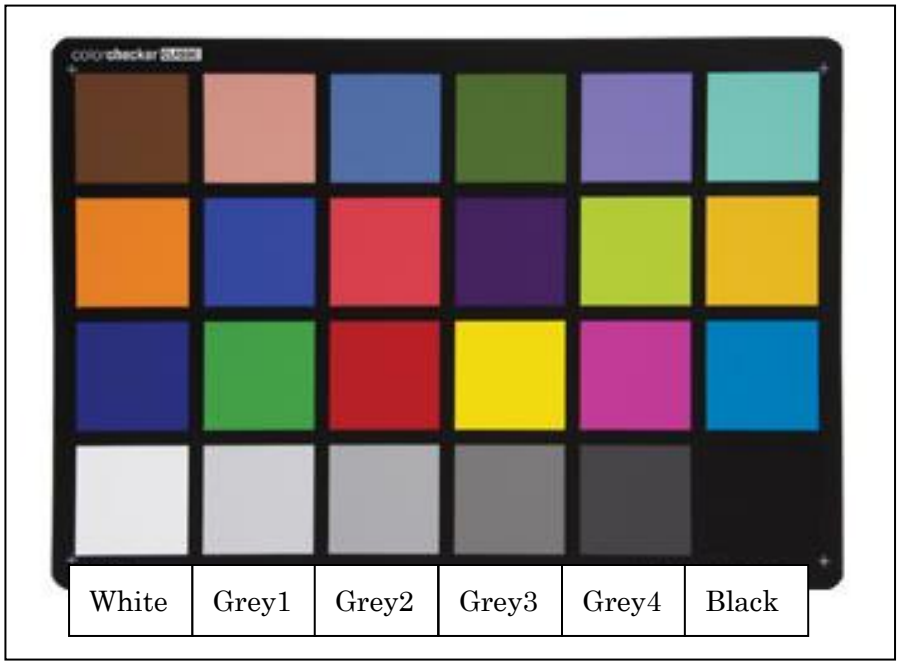

図 4.1 ColorChecker

4.2 色収差の補正

色収差を補正する手順は以下になる。

Step 1

Lytro で ColorChecker を撮影する。

Step 2

3.2 節の生成手順に従って視差画像を生成する。

Step 3

生成した視差画像の White～Black までの RGB 値を調べる。

Step 4

1 枚の基準画像を決め、残りの視差画像の RGB 値を基準画像の RGB 値に揃える。

RGB 値を取得するために、フリーソフトである GIMP2 を使用した。GIMP2 とは PhotoShop なみの高機能な画像編集ソフトである。

Lytro で ColorChecker を撮影し、その lfp ファイルを視差画像に分解した後、それらの RGB 値を調べた。図 4.2 は図 3.7 における I44 に位置する画素を取り出して、配列した視差画像である。図 4.2 を見ると全体的に黄色成分が多いと考えられる。続いてこの図の、RGB 値を調べる。

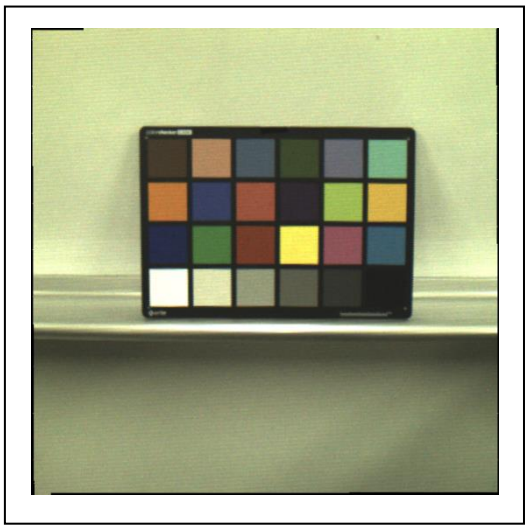

図 4.2 I44 における画素を集めて並べ直した小画像

表 4.1 I44 における RGB 値

I44	White	Grey1	Grey2	Grey3	Grey4	Black	Gr1~Gr4計
R	255	228	149	90	52	32	519
G	255	232	154	92	53	33	531
B	245	178	120	75	46	30	419

表 4.1 は I44 の白～黒までの RGB 値である。表を見ると、Blue 成分が Red,Green 成分と比べ少ないことが分かる。最も右の欄に Gr1～Gr4 計とあるが、これは Grey1～Grey4 の RGB 値の合計である。これらの数値を用いて、補正量を決定した。White と Black を使用しない理由として、この 2 色は飽和しやすい特徴を持っているためである。各成分の Gr1～Gr4 の合計を Rsum,Gsum,Bsum とする。式(2)、(3)に示すように、Gsum に対する Rsum の割合を G_Rratio、Gsum に対する Bsum の割合を G_Bratio とする。

$$G\_Rratio = Gsum / Rsum \cdots (2)$$

$$G\_Bratio = Gsum / Bsum \cdots (3)$$

表 4.2 は、上記の式を用いて計算された結果である。R 成分を 1.023 倍する事で、R 成分と G 成分のバランスが補正される。同様に B 成分を 1.267 倍することで、B 成分と G 成分の色バランスが補正される。

表 4.2 I44 の G に対する R と B の割合

I44	
G_Rratio	1.023
G_Bratio	1.267

同様に他の視差画像の補正を行う。図 4.3 は、マイクロレンズの I84 に位置する画素を取り出し、配列してできた視差画像である。表 4.3 は、同様の手順に基づいて求めた RGB 値となる。表 4.3 と表 4.1 を比べると、全体的に RGB 値が少ないことが分かる。G 成分に R 成分、B 成分を揃えるだけでは、完全に補正できるとはいえない。

そこで以下の式(4)を用いて明るさの補正を行う。

$$Gratio = \text{I44 の } Gsum / \text{I44 以外の } Gsum \cdots (4)$$

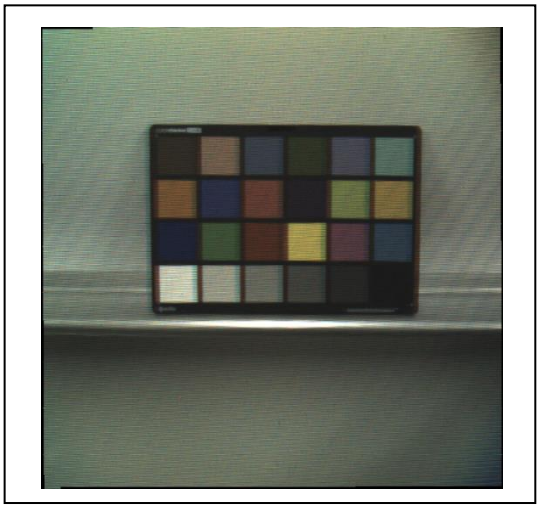

図 4.3 I84 における画素を集めて並べ直した小画像

表 4.3 I84 における RGB 値

I84	White	Grey1	Grey2	Grey3	Grey4	Black	Gr1~Gr4計
R	196	147	99	61	37	27	344
G	198	149	101	62	38	27	350
B	175	130	89	57	37	26	313

表 4.4 I84 の G に対する R と B の割合

I84	
G_Rratio	1.017
G_Bratio	1.118

表 4.4 は、式(2),(3)を用いて求めた割合である。R 成分に 1.017 倍、B 成分に 1.118 倍することで RGB 値のバランスが取れる。さらに、式(4)を用いて I44 の Gsum と I84 の Gsum の値を揃える。式(4)を使い求めた値は、1.517 である。I84 の各成分を、さらに 1.517 倍することで RGB 値を揃える。

4.3 色収差を補正した視差画像群の生成

4.2 節の補正手法に基づいて生成した視差画像の例を図 4.4 に示す。図 4.4 左は I44、右は I84 の色を補正した結果である。両者を比べると完璧とはいえないが、明るさと色の補正ができていることがわかる。まず、ColorChecker の灰色部分は正しく補正されている。しかし、I84 では、ColorChecker 周囲の背景に青みがかっている。さらに、画像の四隅を確認すると I44 は良好に補正されているが、I84 ではさらに青みが深く、十分な補正できていないことが分かる。

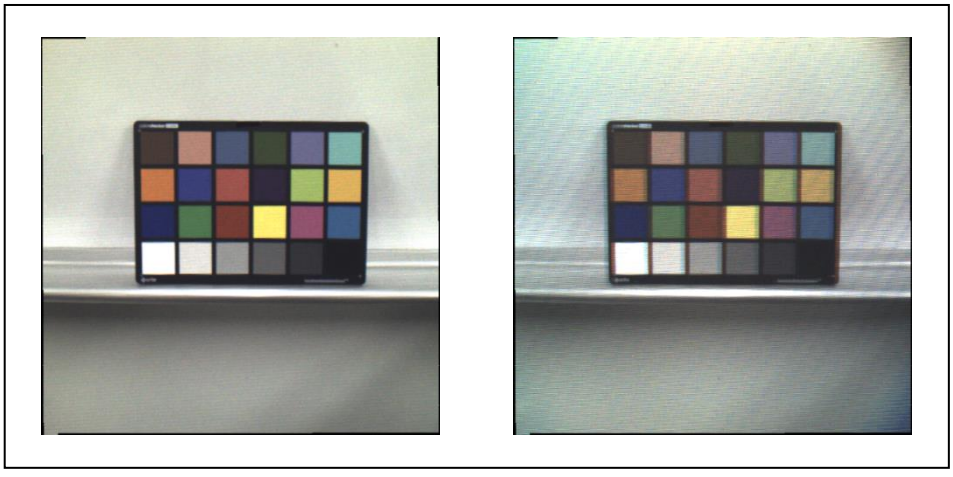

図 4.4 補正後の任意視点画像

ColorChecker を用いた補正を行い、3 章で扱った画像に対して適用した。図 4.5 は適用後の画像であり、左が I44 を補正した結果、右は I84 を補正した結果である。I84 を補正した結果について、図 4.4 と比べると全体的な青みの強さはやや弱いように見える。しかし、右下に青みが残ってしまった。これは、この部分ではマイクロレンズが光線を正しく記録できていないためと考えられる。図 4.6 は、I44 とその 8 近傍に対応する画素を取り出し配列した視差画像を補正したものである。これらは正しく補正されている。一方、マイクロレンズに対して周辺に位置する視差画像の補正には不十分な点が残る。従って色収差の補正はほぼ成功したといえる。

図 4.5 補正したトナカイの画像（I44 と I84 に対応した視差画像）

図 4.6 補正したトナカイの画像（I44 とその近傍に対応した視差画像の補正結果）

5 章 Lytro アプリケーションとの比較検証と考察

本論文で提案した機能と同様の機能が Lytro アプリケーションとして提供されている。表 5.1 に、Lytro アプリケーションが生成した画像と比較した結果を示す。

任意視点画像は、ユーザの指示によって表示される任意の視点からの画像である。任意視点画像は、その元になる視差画像群から生成する。本研究のアプリケーションは、あらかじめ作成した 50 枚程度の視差画像群からユーザのマウス指示によって決まる 1 枚の画像を選択し表示するものである。Lytro アプリケーションも、類似の処理を行っていると考えられる。

視差画像群を生成するまでの時間は、推定手法は約 20 秒、Lytro アプリケーションは 2~3 分である。Lytro アプリケーションは、一度、視差画像群を生成するとそれをファイルとして保存する。それに対し本研究で開発したプログラムは、Lytro アプリケーションより早く視差画像を生成することができるが、その結果を保存しないので、2 回目以降も同様の時間が必要である。

画像の質について検証する。色について考えると推定手法は筋のようなものや青みが残っているのに対し、Lytro アプリケーションは全体的にきれいな結果が得られた。しかし気になる点も見つかった。図 5.1 左側は推定手法を用いて生成した視差画像の一部を拡大した画像で、右側は Lytro アプリケーションの一部を拡大した画像である。右上の Lytro アプリケーションに注目する。トナカイの角の上部に本来無いはずの筋が見える。さらに、下 2 枚を比較すると、Lytro アプリケーションでは角の輪郭がぼやけていることが分かる。

続いて、画像の動きについて検証する。推定手法は Lytro アプリケーションと比べ、動きが、若干かくかくとする。この問題点は、視差画像間の中間画像を生成することで、改善できると考えられる。中間画像は、生成された視差画像の 2 枚を加重平均することで生成することが可能である。

最後に、視点を変えたことによる、画像の見かけの変化量について比較する。本研究、Lytro アプリケーションを比較しても変化量は同じという結果が得られた。

表 5.1 Lytro アプリケーションとの比較

比較する項目	本研究	Lytro アプリケーション
視差画像群を生成するまでの時間	約 20 秒	2～3 分
画像の動き	若干かくかく	なめらか
色の違い	部分的に汚い	全体的にきれい
変化量	同じ	

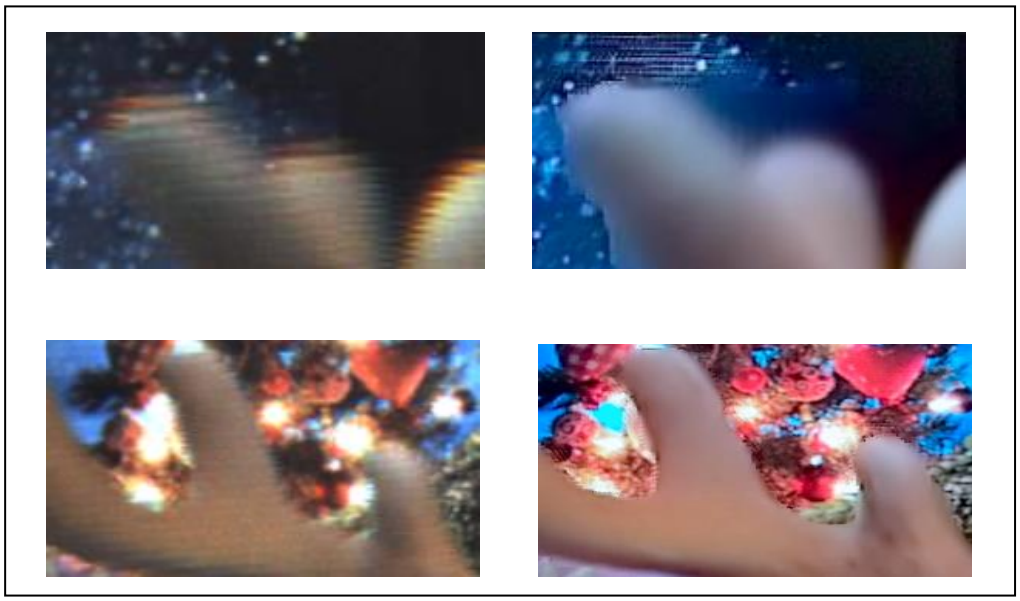

図 5.1 任意視点画像生成後の一部拡大図

6 章　結論

6.1 成果

Lytro が記録したライトフィールドである出力ファイルを用いて、任意視点画像を生成した。

任意視点画像の元になる視差画像群を解析すると、画像ごとに色と明るさが異なっていた。そこで X-rite 社の ColorChecker を使用し、灰色部分における RGB 値が揃うように信号値を補正した。その結果、全体的な色のバランスを補正することはできた。ただし、マイクロレンズに対して、周辺部にある画像を用いて生成した視差画像に対しては、色を完全に補正することはできなかった。

明るさと色バランスを補正した視差画像群を入力として、ユーザがマウスで指定する視点からシーンを観察した任意視点画像を生成するプログラムを作成した。これは、マイクロレンズが正しく認識する約 50 枚の任意視点画像をマウスドラッグすることで 3D 化にしている。

Lytro アプリケーションと比較すると、まず画質が違うことが分かった。さらに画像の動きでも推定手法が劣っている結果となったが、任意視点画像の生成方法については正しいと判断した。画質や動き方はプログラムの問題であると考え、推定手法の正当性という面では本研究は成功したといえる。さらに詳しく検証すると Lytro アプリケーションにも問題点が見つかった。撮影した画像に本来無いはずの筋や、視差の動きが大きい輪郭部分にぼけがあった。Lytro にも問題点があり、まだまだ改善の余地があるという結論に至った。

6.2 課題

課題として、色バランス修正後の筋のようなもの、部分的に色が残っているので修正する必要がある。3D 化を行った際に、画像をよりなめらかにするために、中間画像を生成する必要もある。

謝辞

本研究を進めるにあたり、担当教授の蚊野浩先生に感謝致します。丁寧かつ熱心なご指導本当にありがとうございました。

参考文献

[1]Lytro 社の Web サイト, https://www.lytro.com/
[2]Marc Levoy の Web サイト, http://graphics.stanford.edu/~levoy/
[3]蚊野浩、日経エレクトロニクス、42-47、2012-8-20

付録

【研究環境】

Microsoft Visual Studio 2010

【コンピュータ本体】

1.6GHz 以上のプロセッサを搭載したパーソナルコンピュータ

【オペレーティングシステム】

Windows 7

Windows Vista Service Pack 2 以降

Windows XP Service Pack 3 以降

Windows Server 2008 R2

Windows Server 2008 Service Pack 2 以降

Windows Server 2003 R2

Windows Server 2003 Service Pack 2 以降

詳細は、

www.microsoft.com/visualstudio/2010/sysreq/

【メモリ】

1GB(x86)または 2GB(x64)以上の実装メモリが必要

【ハードディスク】

4GB 以上の空き容量が必要

5,400RPM 以上のハードディスクドライブが必要

【ディスク装置】

DVD-ROM ドライブ

【その他】

DirectX 9 互換のグラフィックシステム(解像度：1,024×768 以上)

【同梱ソフトウェア】

Microsoft Visual Studio 2010 Professional

本論文で作成したプログラムを説明する。

【プログラム名】

demosaic.cpp

【内容】

IMG_0025.lfp を読み込み～RGB カラー画像へデモザイクまでを行うプログラムである。5 つのセクションで作られ、第 2 セクションでは濃淡画像のカラーフィルタアレイ画像と、デモザイクを行うプログラムである。IMG_0025.lfp には、論文で使用したトナカイの画像データが入っている。IMG_0008.lfp という ColorChecker の画像も同フォルダに入っている。プログラムを実行することで、testRaw.tif と winterRGB.tif ファイルという名前で濃淡画像と、RGB カラー画像が生成される。

【プログラム名】

arbitraryViewpoint.cpp

【内容】

このプログラムでは、demosaic.cpp で生成された winterRGB.tif を入力とし、任意視点画像を生成するプログラムである。まず、main 関数で 81 枚の視差画像群を生成する。視差画像群の生成する際、makeSpecfiedViewpointImage 関数でバイリニア補間を行う。その後、色収差補正を行う bgr_split 関数を経由し、任意視点画像を生成するため、onMouse 関数で処理を行っている。生成後、660×660 のウィンドウ上でマウスドラッグすると 3D に見える画像が出力される。

【ファイル名】

比較.xlsx

【内容】

このファイルは色収差補正の際に調べた RGB 値の画素値、それに基づく割合を記載したファイルである。