月面の分光画像解析プログラムの開発

宮城教育大学生涯教育総合課程自然環境専攻
D1920　　千葉紀子

目次

第1章　はじめに
　１－１．本研究のテーマ

第2章　システムの概要
　２－１．プログラムの実行に必要なシステム環境

第3章　プログラムの特徴
　３－１．プログラミング言語について
　３－２．プログラム使用のメリット

第4章　画像解析方法
　４－１．解析処理の手順
　　４－１－１．ダークフレーム処理
　　４－１－２．フラットフィールド補正
　　４－１－３．大気差補正
　　４－１－４．輝度値補正
　　４－１－５．幾何補正
　　４－１－６．モザイク処理
　　４－１－７．測光補正
　　４－１－８．標準反射率への変換
　４－２．対象画像

第5章　プログラムの設計
　５－１．フローチャート
　５－２．プログラム

第6章　結果及び評価
　６－１．使用カメラによるデータ格納の違い
　６－２．補正画像

第7章　まとめ
　引用文献
　謝辞
　付録１　画像解析プログラム
　付録２　解析プログラム実行マニュアル

はじめに

１－１．　本研究のテーマ

これまで天体望遠鏡で撮像された月面画像の解析には“ ENVI ”（アダムネット社製）、“ Photoshop ”（Adobe 社製）等の既製の画像処理ソフトウェアを用いて、画像の処理や補正を施してきた。しかしこれらソフトウェアを用いた場合でも、解析に使用する月面画像を仕上げるには多大な時間を要する。解析画像枚数は数 100 枚であり、各画像に複数の処理や補正を実施するため、画像ファイルオープン、演算、ファイル保存の作業が繰り返され、処理枚数に比例して処理時間も増加する。本研究はこの問題を解決するため効率的に画像処理が実施できるように、複数の画像を多プロセスにわたり一括処理できるソフトウェアの開発を行った。画像枚数の増加に対しての処理時間の増加が少ないことも利点の一つである。また、画像解析に必要なデータを必要に応じて出力できるようにプログラムを書き換えられる点もこれまでの画像処理では困難な問題の解消につながる。

さらに、“ ENVI ”等高額ソフトウェアでは、ユーザーが限定されるなどの制限があり、ソフトウェアが稼動するプラットフォームも限られている。しかし、プラットフォームに関わらない共通言語を利用することにより、どのようなシステム環境でも運用可能となる。

そこで本研究では、天体望遠鏡に取り付けた CCD カメラで撮像した画像の補正処理用のプログラムを C 言語を用いて開発した。

第2章　システムの概要

2-1　プログラム実行のシステム環境

今回プログラム開発は UNIX 上で実施される。UNIX は、複数のユーザーが同時に使用可能で、各ユーザーが同時に複数の作業を実行できる環境を提供するオペレーティングシステム（OS）である【1】。

UNIX 上ではコマンドを打ち込み OS に命令する。コマンドは豊富に揃っているが、ユーザーが独自のコマンドを作成しユーザーの環境に合う OS を作り出すことができる。これらのコマンドは、ホスト（procsv サーバ）にログインした状態で起動可能だが、その仕組みは次のようになっている。まず、ユーザーが入力したコマンドはシェルと呼ばれるプログラムに渡り、このシェルがコマンド起動するように UNIX オペレーティングシステムの中核であるカーネルに対し指示を出す。カーネルは指示されたコマンドを起動し、実行する仕組みになっている。

今回、ホストへのログオンは、PC から X-window システム（FUJITSU PC-X（FUJITSU 社製））を使用して行った。PC-X とは Windows 環境下で UNIX の X window システムを実現するアプリケーションである。また今回使用する UNIX は Solaris（Sun Microsystems 社製）という UNIX である。

UNIX の特徴としては以下のようなことが挙げられる【2】。

- 高度な科学技術計算
- 大量のデータ処理
- インターネットを含むコンピュータネットワークの高度な利用
- Web サーバやデータベースサーバなどのサーバソフトウェアの安定稼動

本学のファイルサーバ（“filesv”）システムは、ハードディスクや MO などをソフトウェア的に接続しているため、ネットワークを介することでネットワークリソースにローカルデバイスと同じ様にアクセス可能である。filesv の各自のディレクトリを 1 つのドライブ（=Z ドライブ）として扱うことができる。

本研究では、処理に用いる画像はすべて FFFTP（曽田純氏作成）の FTP クライアントソフトで procsv サーバの Z ドライブに保存し、各端末から X window システムを介しサーバにアクセスし、画像処理の演算、つまりプログラムの実行を UNIX で実施した。

第3章　プログラムの特徴

3-1　プログラミング言語について

本研究で開発したプログラムは C 言語で記述した。C 言語の特徴としては次のようなことがあげられる。【3】

- プログラムにあたって、覚えることが非常に少ない。また、例外も少ない。
- 構造化された制御構造のため、作業の手順化、明確化が容易に可能。
- 言語構造自体がシンプルなので、比較的小さいメモリや外部記憶装置しかないハードウェアでも開発が効率よく行える。
- ビットの操作など、きめの細かい処理が可能。
- 構造体や共用体などの複雑なデータを扱えるので、プログラム自身を簡潔に記述できる。
- 機械語への翻訳がコンパイラによるため、一括して実行形式に変換でき、実行速度が速い。
- CPU や OS に依存しないため、移植性が高い。

開発者本人だけでなく、ソフトウェアのユーザーが見ても理解可能なプログラムを開発することを前提としているため、構造化された言語である C 言語を用いた。また、このプログラムで扱う画像はすべて 16 ビットのデータとして格納されており、ピクセルごとの操作をする際にはビットの操作が可能な C 言語が適している。

3-2　画像処理プログラム開発のメリット

今回開発したプログラムを使用する最大のメリットは効率化にある。コンピュータは繰り返しの計算を得意とする。画像処理においてピクセルごとの細かな作業を行う上でこれまでは人間の目で確認し一つ一つ作業を進めてきたが、これは非常に時間を要する。莫大な枚数の画像１枚１枚に同じ処理や補正を施す作業は人間の手で行うのは非効率的であり、このような繰り返しの作業をコンピュータに行わせることが効率化につながる。

もう一つの利点としては、プログラムは書き換えが可能であるという点である。コンピュータに命令したい作業の変更が可能である。解析では、画像をファイルに書き込むだけでなく、あらゆる計算過程を経て出力されたデータを検討する必要があるわけだが、それぞれの計算結果を必要に応じて数値として出力させる作業等はこれまでの既存のソフトでは柔軟性に欠け困難であった。本研究で開発されるプログラムでは、作業の効率化だけではなく、出力されたデータに疑問を持った際、どういった計算の結果出力された値かを確認できることも目的としている。そのため、ユーザーがプログラム自体の内容を変更し、必要なデータを必要に応じて出力可能であるという点は大きなメリットと言える。また、より優れた画像処理プログラム案が提案された場合においても、本研究で開発したプログラムを書き換え改良することも可能であるため、精度の高い解析につなげることが容易である。

第4章　画像解析方法

4-1　解析処理の手順

天体望遠鏡で撮像した惑星、衛星表層の画像を解析するに至るまでにはいくつかの画像処理や補正が必要である。画像は撮像された段階ではカメラの特性により生じるノイズや光量ムラを含んでいる。また、月面の大気による輝度値の揺らぎや、撮像中においては、月と地球上の撮像地点との位置関係のズレが生じる。月面の物理量としての反射率は上記の影響を取り除くための処理や補正をした後に得られる。また、月面全体を再現するためには月面をいくつかの地域に分割した画像が必要となる。これら多数の画像をつなぎ合わせる処理も必要である。

天体望遠鏡で撮像した月面分光画像の処理や補正では以下の以下のとおりである。

① ダークフレーム処理
② フラットフィールド補正
③ 大気差補正
④ 輝度値補正
⑤ モザイク処理
⑥ 測光補正
⑦ 標準反射率への変換

4-1-1　ダークフレーム処理

ライトフレームで撮像した画像にはダークノイズ（暗電流）が含まれている。CCD カメラの特性から、熱により蓄えられた電子が光により蓄えられている電子に加算され天体自体の持つ正確な輝度値が得られない。このノイズを取り除く手段として、ライトフレーム画像からダークフレーム画像を減算するダークフレーム処理が行われる。ライトフレーム画像とダークフレーム画像をそれぞれ読み込み、１ピクセルずつ減算後、ダークフレーム処理済みの画像としてファイルに出力する。

第 3 章でも記述したが、この処理は可視カメラで撮像した画像は撮像時点で行われているため、赤外カメラで撮像された画像にのみ施す処理である。演算式は以下のようになる。

ダーク処理済画像（x.y） = ライトフレーム画像（x.y）－　ダークフレーム画像（x,y）
（式 1-1）

ここで、
x：画像の幅（sample）
y：画像の高さ（line）
である。

4-1-2　フラットフィールド補正

CCD 各ピクセルの感度ムラや光量ムラを補正するには、ライトフレーム画像をフラットフィールド画像で除算する。フラットフィールドとは、均一な光量の写野を撮像し、CCD 各ピクセルの感度のばらつきや光学系のひずみ、周辺減光など、撮像時のばらつきを取り除くためのものである。演算式は以下のとおりである。

$$E(x,y) = K * \frac{G_L(x,y)}{G_F(x,y)} \qquad \text{（式 2-1）}$$

ここで、

$E(x,y)$:フラｈットフィールド補正済画像
K:ダークフレーム処理済フラットフィールド画像全ピクセル平均
$G_L(x,y)$:ダークフレーム処理済ライトフレーム画像
$G_F(x,y)$:ダークフレーム処理済フラットフィールド画像

である。

フラットフィールド補正では、各画像中に輝度値が 0 となるピクセルがあった場合、除算や乗算の障害となるため、予め値を持たないピクセルを補正しておく必要がある。これをピクセル補間という。ピクセル補間の手順としては、輝度値が 0 となるピクセルの周囲 8 ピクセルの平均輝度値を入力する。このときの演算式は（式 2-2）のとおりである。また、補間されるピクセルとその周囲のピクセルの位置関係は図 2-2 のとおりである。

$$\begin{aligned} A(x,y) = (&A(x-1,y-1) + A(x,y-1) + A(x+1,y-1) + A(x-1,y) + A(x+1,y) \\ &+ A(x-1,y+1) + A(x,y+1) + A(x+1,y+1))/8 + 0.5 \end{aligned}$$

（式 2-2）

A(x-2,y-1)	*A(x,y-1)*	*A(x+1,y-1)*
A(x-1,y)	*A(x,y)*	*A(x+1,y)*
A(x-1,y+1)	*A(x,y+1)*	*A(x+1,y+1)*

図 2-2　ピクセル補間における補完ピクセルの位置関係

4-1-3　大気差補正

月面画像内には月以外のバックグラウンドも含まれる。バックグラウンドは真宇宙であるため輝度値は 0 となるのが理想的であるが実際には月面大気による光の散乱や屈折の影響を受けているため、フラットフィールド済の画像中で数 10～数 100 の輝度値を持つ。この影響を大気差という。大気差を減算しバックグラウンド値を 0 にする補正が大気差補正である。

大気は月面部分にも存在するため、月面部分からも大気差を減算する必要がある。大気差補正ではバックグラウンド中で最も大きい輝度値を示す月面付近のバックグラウンドを基準値とし、画像の全ピクセルから減算する方法をとる。しかし、プログラムで画像を読み込み月面付近のバックグラウンドを判別することは困難である。そこで、画像を 10*10 ピクセルのブロックごとに分割して読み込み、隣り合ったピクセル輝度値の変化率がある一定の範囲内に納まっている場合はそのピクセルをバックグラウンドと判断し、ブロック内のすべてのピクセルがこの条件をクリアした場合、そのブロックをバックグラウンドと判断することにした。また、そのブロックにおける輝度値の平均変化率を算出し、平均変化率の最大値を基準値として保存する。その後、画像の全ピクセルからこの基準値を減算する。基準値は月面付近の大気の影響がもっとも大きく生じる部分から算出されるようになっているため、補正後のバックグラウンドの輝度値はほとんどのピクセルで 0 となる。なお、本来輝度値は負にならないため、減算後マイナス値を示すピクセルについては 0 となるようにプログラムされている。また、バックグラウンド判別に用いる一定の範囲は各カメラの画像からのデータを経験値とし入力した。

4-1-4　輝度値補正

隣り合う画像の重複する地域の輝度値は理想的には同等になるべきだが、時間経過に伴う月の高度変化からの光量変化や、大気差補正の誤差により、重複する地域であっても階調差が生じる。画像の階調が異なると、画像をつなぎ合わせた一枚の月面画像が不自然となる。また、輝度値が決め手となる分光観測では、階調差があると正確なデータを得ることはできない。基準となる画像と重複する地域を持つ画像の輝度比を求め、変換定数として乗算することで輝度値が一致する。これを輝度値補正という。

基準となる地域は予め画像を見たうえで決定しておき、その地域と重複する地域を補正側の画像から検索する。プログラムを用いて特定の地域を決定しておくことも可能だが、補正側の画像に重複する地域が無い場合を考慮に入れ、効率化を最優先し、この作業に限っては人間の目でどの地域を基準とするかの判断を加える。

4-1-5　モザイク処理

幾何補正まで行った画像をつなぎ合わせる処理をモザイク処理という。月面の各部分をモザイクし一枚の月面全体画像を再現する。このとき、各画像に共通して画像の中央部分80%のみを採用する。画像の周辺部は歪みを含んでいるため、モザイクの結果、不自然にならないようにする必要があるからだ。

月面全体の画像を再現する場合、月面をいくつかの部分に分けて撮像したものを張り合わせる方法がある。これは、カメラの感度特性や月面の詳細な輝度値情報を得ることを考慮に入れ、月面全体を視野に入れた画像ではなく、数箇所にわけて撮像した画像を張り合わせる方法が適しているためである。

輝度値補正後の画像で、基準となる画像の特定の地域と重複する地域の座標は、月面全体画像上では同一座標で表示されるため、座標の変換を行う。基準画像と座標の変換を施した画像を同一のファイルに出力すると、2 枚の画像が張り合わされ、モザイク処理が行える。

4-1-6　測光補正

天体望遠鏡で月面の反射率を測定する場合、同じ物質でも太陽や観測者の方向によって輝度に差が生じて見えてしまう。分光観測を行う場合、同じ地質の地域は同じ輝度値を示さなければならない。物質表面からの反射光強度が光の入射方向や観測方向によって変化する角度依存性の補正を測光補正という。

月面のある地点を観測するとき、太陽の方向と天頂のなす角（ｉ）を入射角、観測者（地球）の方向と天頂のなす角（ｅ）を観測角、太陽－月面－地球のなす角（ α）を位相角と呼ぶ。満月の中央付近では、入射角（ｉ）と観測角（ｅ）は 0 °となるが、月の縁部分では入射角と観測角はそれぞれ 90 °になる。そのため、月面における入射角、観測角、位相角を統一して、同じ位相で反射率を議論するために行う補正である。

本研究で処理する月面画像は、Clementane 探査機の画像処理の標準的手法となっている Brown 大学の方法を用いて、i=30 °,e=0 °, α=30 °とした場合の輝度に変換する。このとき、画像の各ピクセル輝度に乗算する係数 factor(i, ｅ, α)は次の式で与えられる。

$$factor(i,e,\alpha) = \frac{Fn(30) * \cos 30 / (\cos 0 + \cos 30)}{Fn(\alpha) * \cos i / (\cos e + \cos i)}$$

Fn(α)は方向性輝度値補正係数で、位相角の違いよる観測輝度の変化を表す係数である。Brown 大学では Fa(α)は次の式で表している。

Fn(α) = a + a1* α + a2* α + a3* α + a4* α

ａ= 0.988

ａ1 = －2.101E－4

ａ2 = 2.527E－4

ａ3 = －1.530E－6

ａ4 = 3.367E－9

補正係数を画像にしたもの（フィルター画像）を作成し、月面画像に乗算することで補正が行われる。補正係数に含まれる変数の入射角（ｉ）、観測角（ｅ）、位相角（α）は次の手順で求める。【松下真人　2002 ※4】

ある月面経緯度を（ θ_m, ϕ_m ）とすると、その法線ベクトル $\hat{R}$ は

（$\cos\phi_m\cos\theta_m$, $\cos\phi_m\sin\theta_m$, $\sin\phi_m$）

太陽が真上にある月面経緯度を（ θ_s, ϕ_s ）とすると、太陽の方向ベクトル $\hat{S}$ は

（$\cos\phi_s\cos\theta_s$, $\cos\phi_s\sin\theta_s$, $\sin\phi_s$）

画像の見かけの月面中心の経緯度（ θ_0, ϕ_0 ）とすると、地球の方向ベクトル $\hat{E}$ は

（$\cos\phi_0\cos\theta_0$, $\cos\phi_0\sin\theta_0$, $\sin\phi_0$）

ここで、それぞれの内積は、

$$\hat{R}\cdot\hat{S} = \cos i = \cos\phi_m\cos\theta_m\cos\phi_s\cos\theta_s + \cos\phi_m\sin\theta_m\cos\phi_s\sin\theta_s + \sin\phi_m\sin\phi_s$$

$$= \cos\phi_m\cos\theta_s\,(\cos\theta_m\cos\theta_s + \sin\theta_m\sin\theta_s) + \sin\phi_m\sin\phi_s$$

同様に、

$$\hat{R}\cdot\hat{E} = \cos e = \cos\phi_m\cos\theta_0\,(\cos\theta_m\cos\theta_0 + \sin\theta_m\sin\theta_0) + \sin\phi_m\sin\phi_0$$

$$\hat{S}\cdot\hat{E} = \cos\alpha = \cos\phi_s\cos\theta_0\,(\cos\theta_s\cos\theta_0 + \sin\theta_s\sin\theta_0) + \sin\phi_s\sin\phi_0$$

よって、

$$i = \cos^{-1}[\cos\phi_m\cos\theta_s\,(\cos\theta_m\cos\theta_s + \sin\theta_m\sin\theta_s) + \sin\phi_m\sin\phi_s] \quad （式 1）$$

$$e = \cos^{-1}[\cos\phi_m\cos\theta_0\,(\cos\theta_m\cos\theta_0 + \sin\theta_m\sin\theta_0) + \sin\phi_m\sin\phi_0] \quad （式 2）$$

$$\alpha = \cos^{-1}[\cos\phi_s\cos\theta_0\,(\cos\theta_s\cos\theta_0 + \sin\theta_s\sin\theta_0) + \sin\phi_s\sin\phi_0] \quad （式 3）$$

このとき、（ θ_0, ϕ_0 ）、（ θ_s, ϕ_s ）は天文年鑑等からデータを調べ、（ θ_m, ϕ_m ）は月面画像から読み取る。

月の北の方向は画像の上向きと一致していないため、画像の z 方向を月の北方向に合致させるために、みかけの中心を（Y_0, Z_0）として、（y, z）を反時計回りに t [rad]回転させた座標を（p, q）とすると、

$$p = y\cos t - z\sin t + Y_0$$
$$q = y\sin t + z\cos t + Z_0 \quad \text{（式 6-9）}$$

このときの y, z, Y_0 ,Z_0 は以下のとおりである。

月を球と仮定し平面に投影した場合、見かけの月面中心を原点としたときの球座標（r, θ, φ)、月の東西方向を y、と南北方向を z とする y-z 平面における関係は次のとおりである。

緯度 φのとき z 座標は、

$$z = -r\sin\phi \quad \text{（式 6-10）}$$

赤道上の経度 θのとき y 座標は、

$$y = r\sin\theta \quad \text{（式 6-11）}$$

だが、緯度が変わると y 座標も変わり、

$$y = r\sin\theta\cos\phi \quad \text{（式 6-12）}$$

となる。よって、

$$\phi = \sin^{-1}(-z/r)$$
$$\theta = \sin^{-1}(y/r\cos\phi) \quad \text{（式 6-13）}$$

となる。ここで、

$$\cos^2\phi + \sin^2\phi = 1 \quad \text{（式 6-14）}$$

（式 6-12）より、

$$\sin^2\phi = (z/r)^2 \quad \text{（式 6-15）}$$

よって、

$$\cos^2\phi = 1 - (z/r)^2 \quad \text{（式 6-16）}$$

$$-(\pi/2) \leqq \phi \leqq (\pi/2) \quad \text{より} \quad \cos\phi \geqq 0$$

したがって、

$$\cos\phi = +\sqrt{1-(z/r)^2} \quad \text{（式 6-17）}$$

という条件から、

$$\theta = \sin^{-1}\left[y/r \cdot \sqrt{1-(z/r)^2}\right]$$
$$= \sin^{-1}\left[y/(r^2-z^2)\right] \quad \text{（式 6-18）}$$

（式 6-9）より、

$$\phi = \sin^{-1}(-q/r)$$
$$\theta = \sin^{-1}\left[y/(r^2-q^2)\right] \quad \text{（式 6-19）}$$

となる。

ここで（ θ, φ）は、画像の中心を原点とした球座標の経緯度で、みかけの中心の経緯度（ θ_0, ϕ_0）は秤動のため月の真の中心からずれており、このずれを平行移動させ補正する。平行移動後の月面経緯度を（ θ_m, ϕ_m）とすると、

$$\phi_m = \phi + \phi_0 = \sin^{-1}(-q / r) + \phi_0$$
$$\theta_m = \theta + \theta_0 = \sin^{-1}[y / (r^2 - q^2)] + \theta_0 \qquad \text{（式 6-20）}$$

となり、月面画像から真の月面経緯度が求められる。

（ θ_m, ϕ_m）、（ θ_0, ϕ_0）、（ θ_s, ϕ_s）を（式 6-6）、（式 6-7）、（式 6-8）に代入し、入射角 *i*、観測角 *e*、位相角 *α*が算出できる。

本研究の測光補正においては、プログラムを実行する前段階で、みかけの中心の月面経緯度（ θ_0, ϕ_0）、太陽が真上にある地点の月面経緯度（ θ_s, ϕ_s）、月面の回転角 t のデータを予め入力する必要がある。経緯度はそれぞれ、N を＋、S を－、E を＋、W を－として計算し、プログラム内では角度の単位はすべて radian で計算する。

4-1-7　標準反射率への変換

測光補正を終えた画像の輝度を、Apollo16 号が採取した岩石資料 62231 の反射率の実測値から、標準反射率に変換する必要がある。62231 の採取地点（較正サイト）は 15.1 ° E, 9.0 ° S である。 i = 30, e = 0, α = 30 のとき、62231 の反射率は 1050nm で 20.796％である【※5】。測光補正後の画像から較正サイトの輝度値を求め、上記の反射率となるような比率を算出し、画像の全ピクセルに乗算して標準反射率への変換を行う。

4-2　対象画像

本研究で開発する解析プログラムは、赤外カメラ（IR）または可視カメラ（VIS）で撮像された画像を対象とするものである。赤外カメラと可視カメラの画像サイズが異なるため、プログラム実行前に予め入力が必要となる。また、赤外カメラと可視カメラでは施される処理が異なる。赤外カメラで撮像した画像には 4-1-1 ダークフレーム処理が必要なのに対し、可視カメラで撮像した画像は、撮像時にダークフレーム処理が済んでいるため実際にプログラムで施す処理・補正は 4-1-2 フラットフィールド補正からになる。

本研究のプログラムを実行し月面全体を再現するために用いた画像は、2002 年 11 月 21 日に宮城教育大学の屋上で撮像されたものである。撮像に使用した機材は以下のとおり。

・近赤外 CCD カメラ
　機種名：SU320-1.7（SENSOR　UNLIMITED 社製）
　CCD チップ：InGaAs sensor
　ピクセル数：320 画素×240 画素
　ピクセルサイズ：28[μm]

・シュミットカセグレン天体望遠鏡
　機種名：Meade　LX200GPS-300　（Meade 社製）
　口径：305[mm]
　焦点距離：3048[mm]
　F 値（＝焦点距離／口径）：10

・バンドパスフィルター　（日本真空光学株式会社製）
　品名：BP-1050
　中心波長：1053.0[nm]
　最大透過率：76.4%
　半値幅：52.0[nm]

チャンネルは SW06（exposure time = 8.135[msec]）を使用している。今回は 23 種類のライトフレーム画像を用いて月面全体を再現するが、それぞれの画像について 10 枚重ね合わせ、平均を取ったものを 1 枚として使用しているため、合計 230 枚の画像を用いることになる。ダークフレーム画像、フラットフィールド画像ともに 10 枚ずつの重ね合わせを行っているため、画像は合計で 250 枚である。画像は 1 枚 1 枚特徴のあるピクセルを持っており、例えば、輝度が飽和状態になっているピクセルや輝度が不十分であるピクセルなどが含まれる。こういったピクセルを含んだ画像を使用することは、処理や補正の段階で行われる演算後、本来あるべき輝度値とかけ離れた値となってしまい、解析に用いる画像としては不適切である。よって、1 種類につき 10 枚の画像を重ね合わせ、枚数で平均することで平均的な画像を得ることが重ね合わせの目的である。

第5章　プログラムの設計

5-1　フローチャート

プログラムは各処理・補正ごとにシステム化されている。これらのサブシステムの集合が一つの大きなプログラムとなる。プログラム設計の手順としては、まず全体のアルゴリズムを設計し、次の段階としてその中に含まれる一つ一つのサブシステムを作成していく。プログラムを設計していく上で重要なことは、アルゴリズムを図表化することである。最初からプログラムを書いていくことは、実際にプログラム実行の段階でエラーが発生する確率が高く、作業の効率化の妨げとなる。また、プログラムを作成した者はどういった順序でコンピュータへ作業の命令が出されているか把握できるが、作成者以外の者も内容を理解できるようにしておく必要がある。そのため、プログラムのアルゴリズムを図表化したもの、フローチャートを作成することが重要なのである。

本研究で開発するプログラムも全体の流れを示すフローチャートがあり、それぞれのサブシステムが含まれている。全体のフローチャートを図 5-1-1 に示す。また、それぞれのサブシステムを図化したものを図 5-1-2～図 5-1-6 に示す。

図 5-1-1 から、処理や補正の流れをつかむことが出来る。実際にはプログラムは 3 部に構成されており、①～③、④・⑤、⑥・⑦、と分かれている。①～③はコンピュータが自動的に行い、④・⑤、⑥・⑦の段階でそれぞれユーザーの判断が入る。そのため、③、⑤の終了時点で一旦画像の出力が行われ、出力された画像は④、⑥で読み込みが行われる。

図 5-1-5、図 5-1-6 ではそれぞれ複数のサブシステムが一つのフローチャートで表示されている。これは、④・⑤、⑥・⑦が互いに作業を共有している点が多いため、個別に作成したフローチャートに比べ、一連の流れとして扱ったフローチャートのほうが作業の手順を理解しやすいと判断したためである。

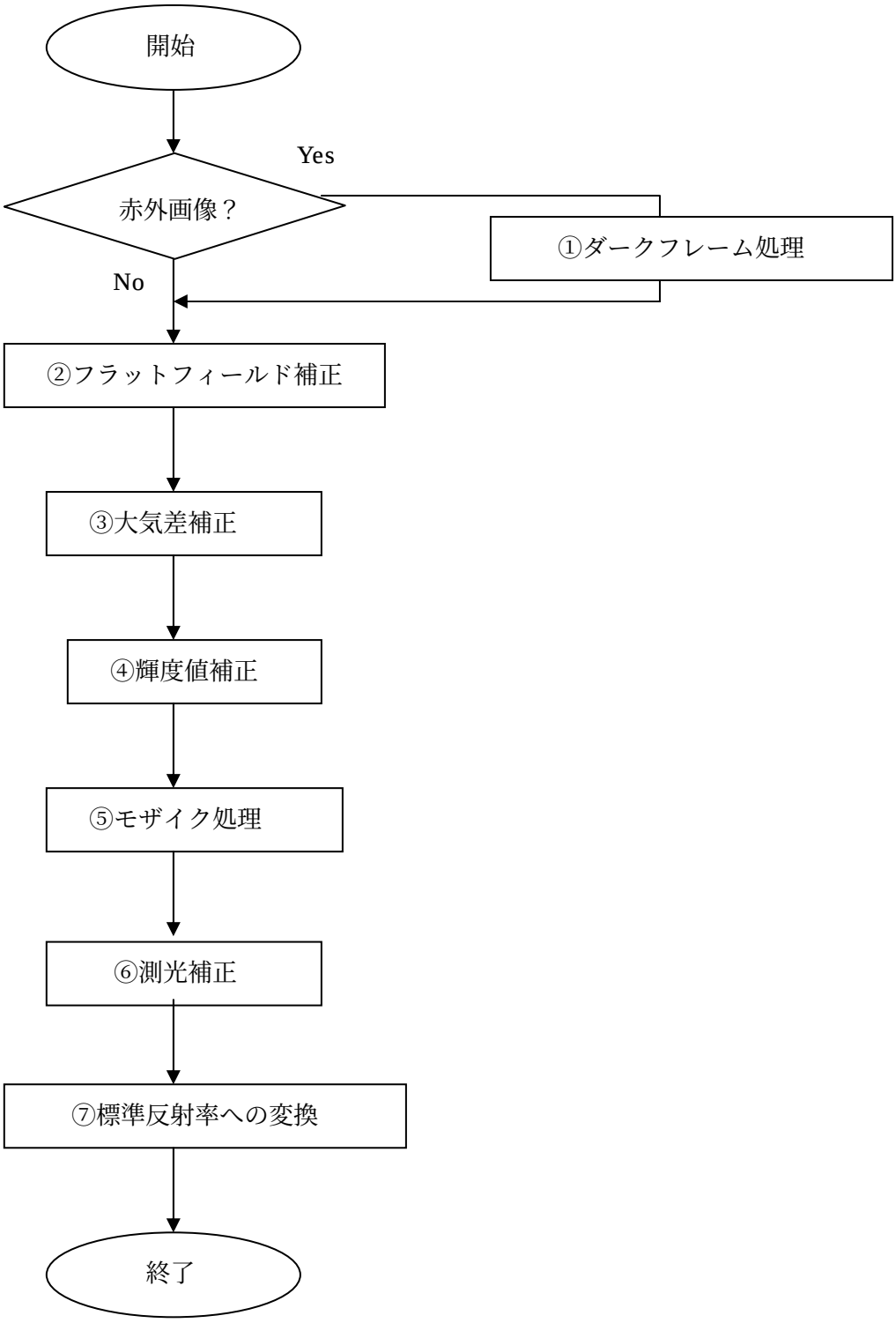


図 5-1-1　全体のフローチャート

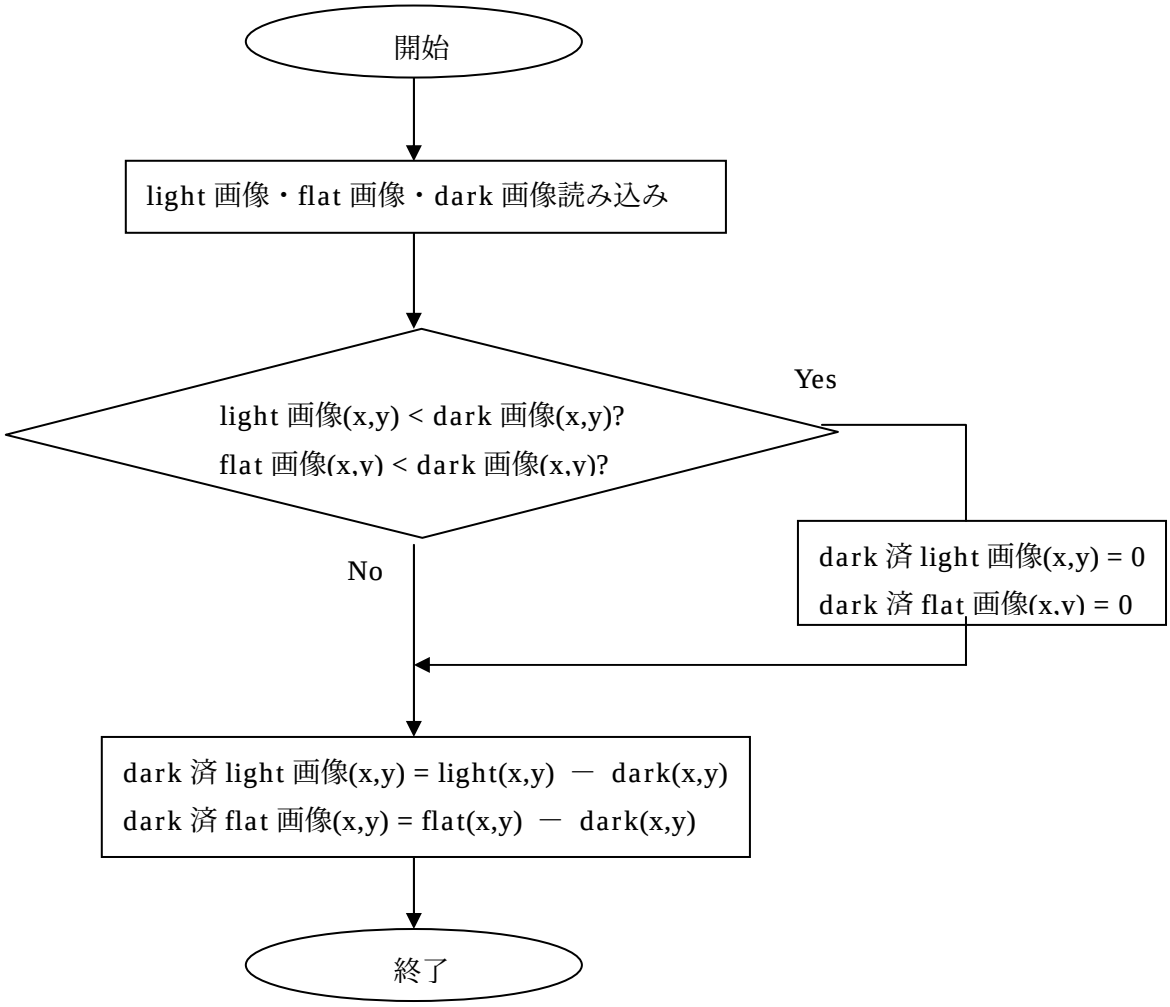


図 5-1-2　①ダークフレーム処理のフローチャート

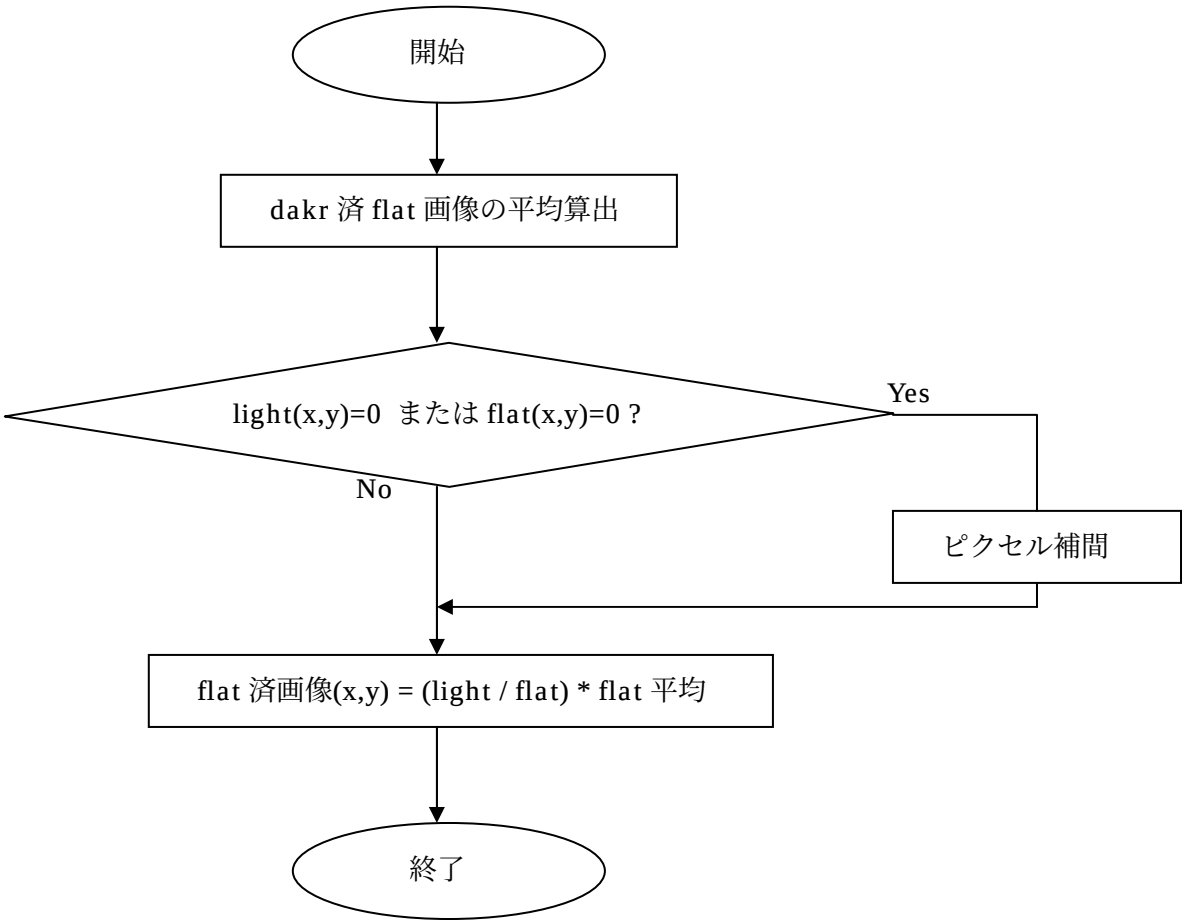
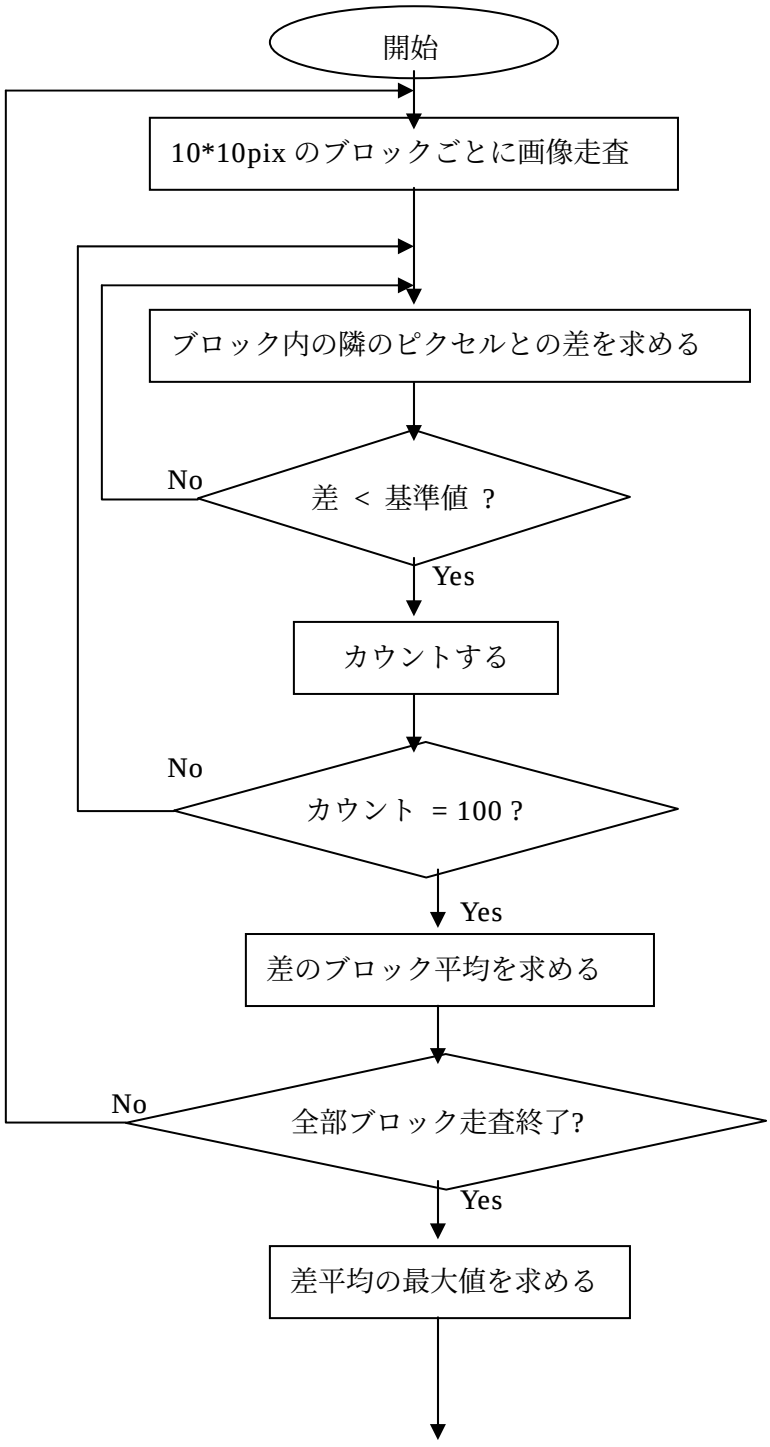


図 5-1-3　②フラットフィールド補正のフローチャート


開始
10*10pix のブロックごとに画像走査
ブロック内の隣のピクセルとの差を求める
No
差 < 基準値 ?
Yes
カウントする
No
カウント = 100 ?
Yes
差のブロック平均を求める
No
全部ブロック走査終了?
Yes
差平均の最大値を求める

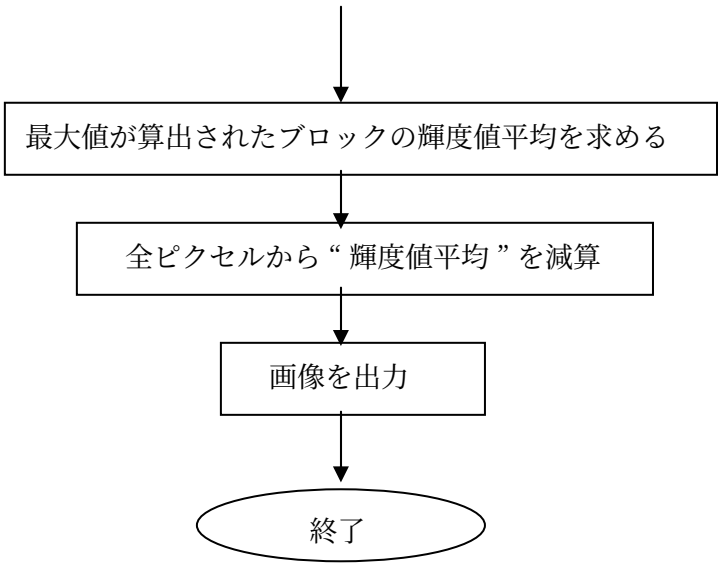
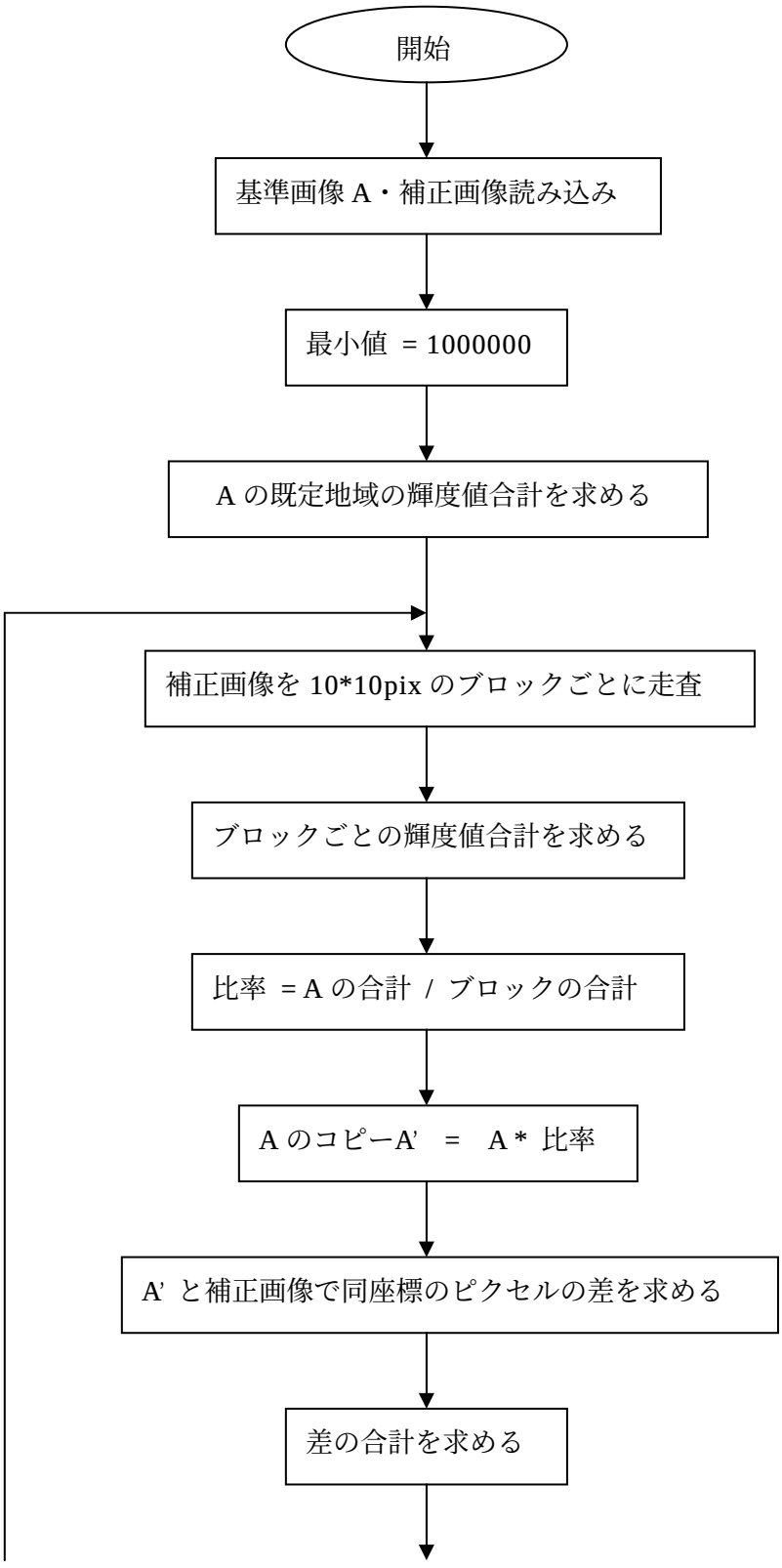


図 5-1-4　大気差補正のフローチャート


開始
基準画像 A・補正画像読み込み
最小値 = 1000000
A の既定地域の輝度値合計を求める
補正画像を 10*10pix のブロックごとに走査
ブロックごとの輝度値合計を求める
比率 = A の合計 / ブロックの合計
A のコピーA' = A * 比率
A' と補正画像で同座標のピクセルの差を求める
差の合計を求める

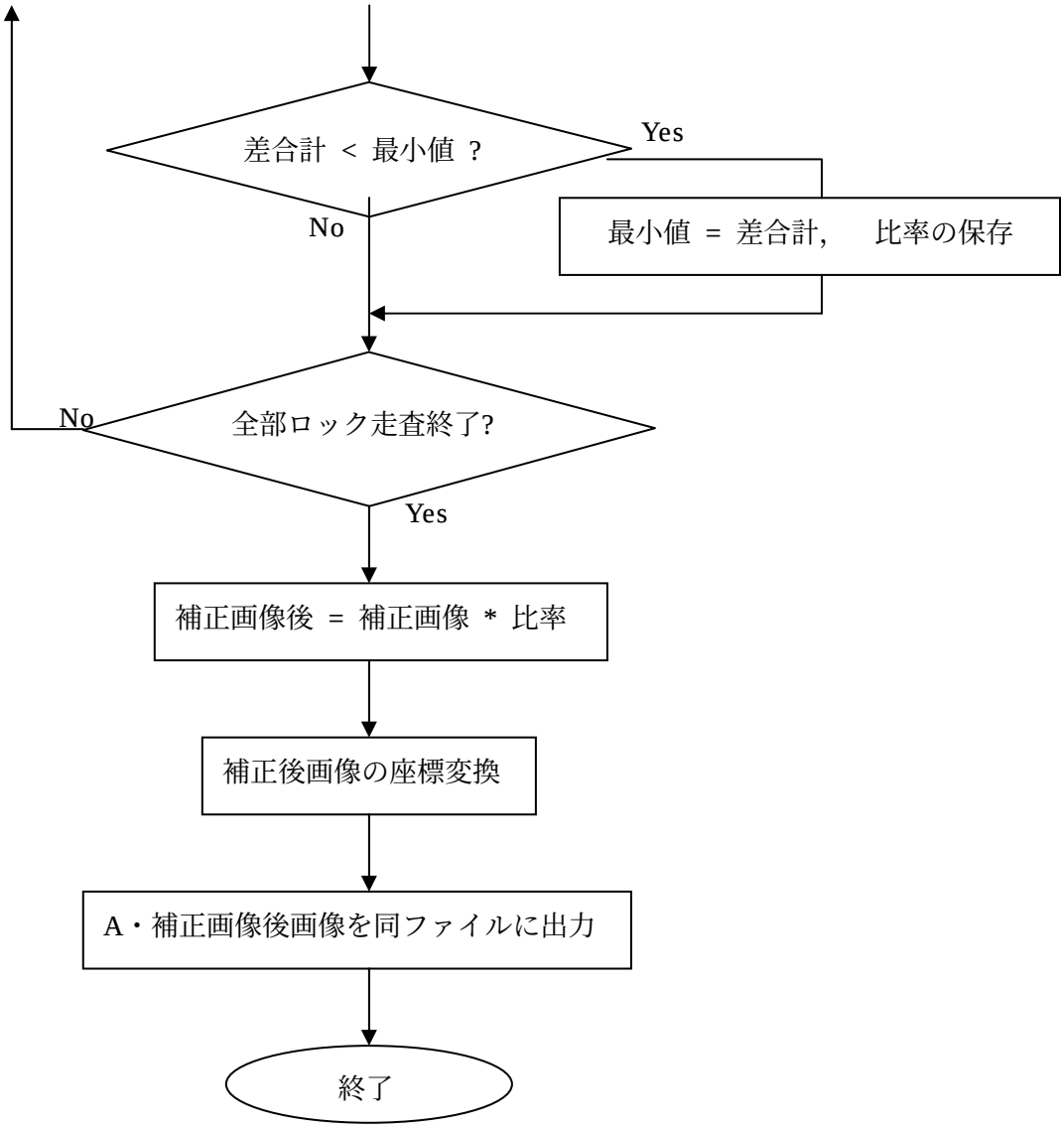


図 5-2-5　④輝度値補正・⑤モザイク処理のフローチャート

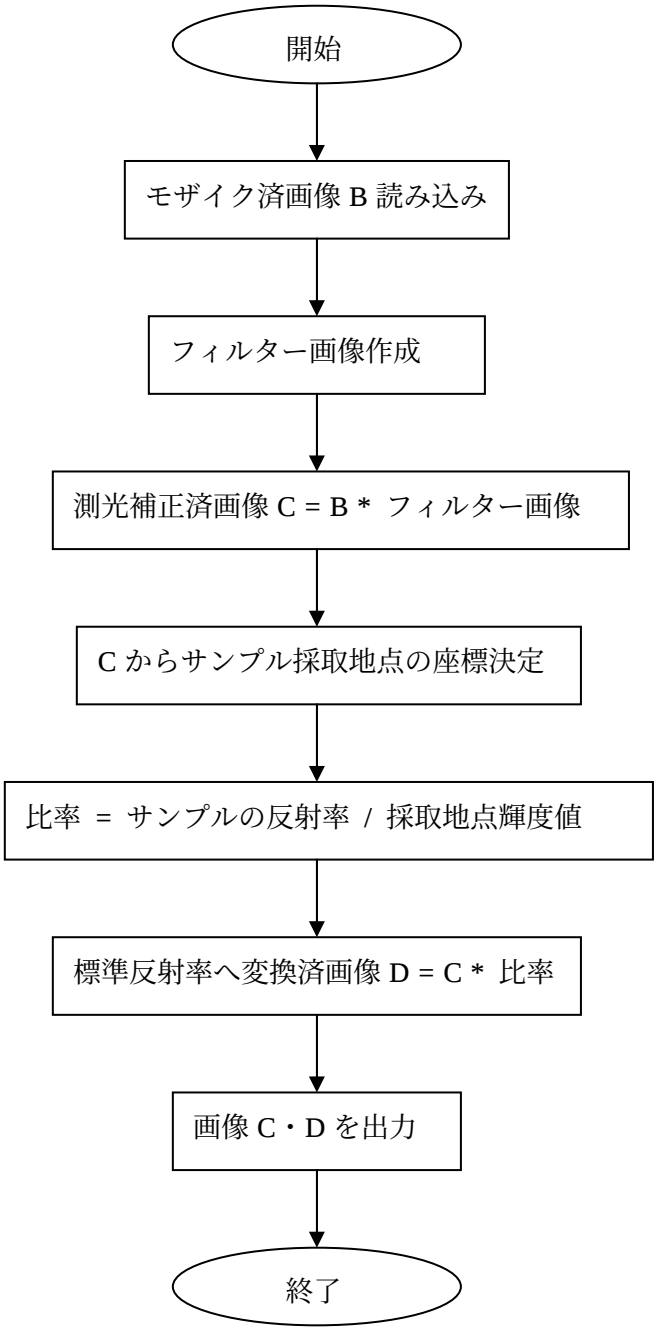


図 5-2-6　⑥測光補正・⑦標準反射率への変換のフローチャート

5-2　プログラム

5-1 でフローチャートに表した作業をプログラムで書くと以下のようになる。リスト 5-2-1 は①～③の作業、リスト 5-2-2 は④・⑤の作業、リスト 5-2-3 は⑥・⑦の作業のプログラムである。

リスト5-2-1　①ダークフレーム処理　②フラットフィールド補正　③大気差補正　のプログラム

```
main( int argc, char *argv[ ] )
{
  int  i,  j,  k,  a,  b,  pre_total,  pre_total2,  disparity,  count,
total_part;
  float   total, total2;

  unsigned short  image_in[Z_SIZE][Y_SIZE];
  unsigned short  image_in_flat[Z_SIZE][Y_SIZE];
  unsigned short  image_in_dark[Z_SIZE][Y_SIZE];
  unsigned short  image_all[num][Z_SIZE][Y_SIZE];
  unsigned short  image_all_flat[num][Z_SIZE][Y_SIZE];
  unsigned short  image_all_dark[num][Z_SIZE][Y_SIZE];
  unsigned long   image_out[Z_SIZE][Y_SIZE];
  unsigned long   image_out_flat[Z_SIZE][Y_SIZE];
  unsigned long   image_out_dark[Z_SIZE][Y_SIZE];
  unsigned short  image_ave[Z_SIZE][Y_SIZE];
  unsigned short  image_ave_flat[Z_SIZE][Y_SIZE];
  unsigned short  image_ave_dark[Z_SIZE][Y_SIZE];
  unsigned short  image_ld[Z_SIZE][Y_SIZE];
  unsigned short  image_fd[Z_SIZE][Y_SIZE];
  unsigned short  image_ldf[Z_SIZE][Y_SIZE];
  unsigned short  image_pre_ldf[Z_SIZE][Y_SIZE];
  unsigned short  image_ldfa[Z_SIZE][Y_SIZE];

  char source_in[80];
  char source_in_flat[80];
  char source_in_dark[80];
  char    source_ldfa[80];
  char   source_ld[80];

/*----------- ① -darkflame --------------*/

  for(k = 0; k < num; k++)
```

```
  {
    strcpy(source_in, *++argv);
    strcpy(source_in_flat, *++argv);
    strcpy(source_in_dark, *++argv);
    image_read(image_in, Y_SIZE, Z_SIZE, source_in);
    image_read(image_in_flat, Y_SIZE, Z_SIZE, source_in_flat);
    image_read(image_in_dark, Y_SIZE, Z_SIZE, source_in_dark);

    for(j = 0; j < Z_SIZE; j++)
      {
        for(i = 0; i < Y_SIZE; i++)
          {
            image_all[k][j][i] = image_in[j][i];
            image_all_flat[k][j][i] = image_in_flat[j][i];
            image_all_dark[k][j][i] = image_in_dark[j][i];
          }
      }
  }
strcpy(source_ld, *++argv);
strcpy(source_ldfa, *++argv);

for(j = 0; j < Z_SIZE; j++)
  {
    for(i = 0; i < Y_SIZE; i++)
      {
        for(k = 0; k < num; k++)
          {
            image_out[j][i] += image_all[k][j][i];
            image_out_flat[j][i] += image_all_flat[k][j][i];
            image_out_dark[j][i] += image_all_dark[k][j][i];
          }
        image_ave[j][i] = image_out[j][i] / num + 0.5;
        image_ave_flat[j][i] = image_out_flat[j][i] / num + 0.5;
        image_ave_dark[j][i] = image_out_dark[j][i] / num + 0.5;
      }
  }
```

```
  for( j = 0; j < Z_SIZE; j++ )
    {
      for( i = 0; i < Y_SIZE; i++ )
        {
          if(image_ave[j][i] >= image_ave_dark[j][i])
            image_ld[j][i] = image_ave[j][i] - image_ave_dark[j][i];
          else
            image_ld[j][i] = 0;
        }
    }

  image_write(image_ld, Y_SIZE, Z_SIZE, source_ld);

  for( j = 0; j < Z_SIZE; j++ )
    {
      for( i = 0; i < Y_SIZE; i++ )
        {
          if(image_ave_flat[j][i] >= image_ave_dark[j][i])
            image_fd[j][i]       =       image_ave_flat[j][i]       -
image_ave_dark[j][i];
          else
            image_fd[j][i] = 0;
        }
    }

  /*------------- ② flatfield correction -------------*/

  for( j = 1; j < Z_SIZE-1; j++ )
    {
      for( i = 1; i < Y_SIZE-1; i++ )
        {
          if( image_ld[j][i] == 0 || image_fd[j][i] == 0 )
          {
            image_ld[j][i]=(image_ld[j-1][i-1]+image_ld[j-1][i]+image
            _ld[j-1][i+1]+image_ld[j][i-1]+image_ld[j][i+1]+image_ld[
```

```
          j+1][i-1]+image_ld[j+1][i]+image_ld[j+1][i+1]) / 8 + 0.5;

          image_fd[j][i]=(image_fd[j-1][i-1]+image_fd[j-1][i]+image
          _fd[j-1][i+1]+image_fd[j][i-1]+image_fd[j][i+1]+image_fd[
          j+1][i-1]+image_fd[j+1][i]+image_fd[j+1][i+1]) / 8 + 0.5;
          }
       }
    }

  pre_total = 0;
  total = 0;

  for(j = 0; j < Z_SIZE; j++)
    {
      for(i = 0; i < Y_SIZE; i++)
        {
          pre_total += image_fd[j][i];
        }
    }

  total = ((float)pre_total / (Y_SIZE * Z_SIZE)) + 0.5;



  for(j = 0; j < Z_SIZE; j++)
    {
      for(i = 0; i < Y_SIZE; i++)
        {
          image_pre_ldf[j][i]    =    ((float)image_ld[j][i]        /
image_fd[j][i]) * total + 0.5;
        }
    }

  for( j = 0; j < Z_SIZE; j++ )
    {
      for( i = 0; i < Y_SIZE; i++ )
        {
```

```
        if( image_pre_ldf[j][i] == 65535 )
          image_ldf[j][i] = 0;
        else
          image_ldf[j][i] = image_pre_ldf[j][i];
       }
    }

/*----------- atmopheric correction ------------*/

  pre_total2 = 0;

   for( a = 1; a < Z_SIZE/10; a++ )
    {
    for( b = 1; b < Y_SIZE/10; b++ )
      {
         total_part = 0;
         total2 = 0;
         count = 0;
      for( j = (a-1)*10; j < (a*10); j++ )
        {
        for( i = (b-1)*10; i < (b*10); i++ )
          {
            disparity = abs(image_ldf[j][i] - image_ldf[j][i+1]);

            if( disparity < lv )
            {
              total_part += image_ldf[j][i];
              count++ ;
            }
          }
        }

      if(count == 100)
        {
          total2 = ((float)total_part / 100);
            if( pre_total2 < total2 )
```

```
            pre_total2 = total2 + 0.5;
        }
       }
     }

    for( j = 0; j < Z_SIZE; j++ )
     {
      for( i = 0; i < Y_SIZE; i++ )
       {
        if( image_ldf[j][i] < pre_total2 )
           image_ldfa[j][i] = 0;
        else
          image_ldfa[j][i] = image_ldf[j][i] - pre_total2;
       }
     }

    image_write( image_ldfa, Y_SIZE, Z_SIZE, source_ldfa );

}
```

リスト 5-2-2　④輝度値補正　⑤モザイク処理　のプログラム

```
main(int argc, char *argv[ ])
{
  int            i1, i2, j1, j2, a, b, I2, J2, minimum, sum1, sum2, total2,
ii, jj, dis;
  float          pre_ratio, ratio;
  unsigned short   image1[Z_SIZE1][Y_SIZE1];
  unsigned short   image2[Z_SIZE][Y_SIZE];
  unsigned short   image3[Z_SIZE][Y_SIZE];
  unsigned short   image4[Z_SIZE][Y_SIZE];
  unsigned short   image5[Z_SIZE1][Y_SIZE1];

  char   source1[800];
  char   source2[80];
  char   source5[800];

  strcpy(source1, *++argv);
  strcpy(source2, *++argv);
  strcpy(source5, *++argv);

  image_read(image1, Y_SIZE1, Z_SIZE1, source1);
  image_read(image2, Y_SIZE, Z_SIZE, source2);

  sum1 = 0;
  minimum = 1000000;

/*-----------------------------------------------------------------*/

  for( j1 = J1; j1 < J1+10; j1++ )
    {
      for( i1 = I1; i1 < I1+10; i1++ )
        {
          sum1 += image1[j1][i1];
        }
```

```
   }

/*-------------------------------------------------------------*/

  for( a = (Z_SIZE/10)*1; a < (Z_SIZE/10)*9; a++ )
    {
      printf("a=%d¥n", a);

      for( b = (Y_SIZE/10)*1; b < (Y_SIZE/10)*9; b++ )
        {
          sum2 = 0;
          total2 = 0;

          for( j2 = a; j2 < a+10; j2++ )
            {
              for( i2 = b; i2 < b+10; i2++ )
                {
                  if( image2[j2][i2] > 10)
                    total2 += image2[j2][i2];
                }
            }

          if( total2 > 300 )
            pre_ratio = (float)sum1/total2;

          jj=0;
          for( j2 = a; j2 < a+10; j2++ )
            {
           ii=0;
              for( i2 = b; i2 < b+10; i2++ )
                {
                  dis = 0;
                  image3[j2][i2] = (image2[j2][i2] * pre_ratio) + 0.5;
                  dis = image3[j2][i2] - image1[J1+jj][I1+ii];
                  sum2 += dis * dis;
                  ii++;
```

```
            }
          jj++;
        }

      if( sum2 < minimum )
        {
          I2 = b;
          J2 = a;
          ratio = pre_ratio;
          minimum = sum2 ;
        }
    }
  }

 for( j2 = (Z_SIZE/10)*1; j2 < (Z_SIZE/10)*9; j2++ )
  {
    for( i2 = (Y_SIZE/10)*1; i2 < (Y_SIZE/10)*9; i2++ )
      {
        image4[j2][i2] = image2[j2][i2] * ratio;
      }
  }
/*-----------------------------------------------------------------*/

 /*  for( j2 = (Z_SIZE/10)*1; j2 < (Z_SIZE/10)*9; j2++ )
  {
    for( i2 = (Y_SIZE/10)*1; i2 < (Y_SIZE/10)*9; i2++ )
      {
        image5[(J1-J2)+j2][(I1-I2)+i2] = image4[j2][i2];
      }
  }*/

 for( j1 = 0; j1 < Z_SIZE1; j1++ )
  {
    for( i1 = 0; i1 < Y_SIZE1; i1++ )
      {
        image5[j1][i1] = image1[j1][i1];
```

```
        }
    }

  for( j2 = /*(Z_SIZE/10)*1*/J2; j2 < (Z_SIZE/10)*9; j2++ )
    {
      for( i2 = (Y_SIZE/10)*1; i2 < (Y_SIZE/10)*9; i2++ )
        {
          image5[(J1-J2)+j2][(I1-I2)+i2] = image4[j2][i2];
        }
    }
    printf("I2=%d     J2=%d       sum1=%d     sum2=%d       minimum=%d
ratio=%f¥n", I2, J2, sum1, sum2, minimum, ratio);

  for(j=0; j<Z_SIZE1; j++)
    {
      for(i=0; i<Y_SIZE1; i++)
        {

          if((image4[j][i-1]==0&&image4[j][i+1]==0)||(image4[j-1][i]=
          =0&&image4[j+1][i]==0))
            image5[j][i] = 0;
          else
            image5[j][i] = image4[j][i];
        }
    }

    image_write( image5, Y_SIZE1, Z_SIZE1, source5 );

    printf("test06¥n");

}
```

リスト 5-2-3　⑥測光補正　⑦標準反射率への変換　のプログラム

```
main(int argc, char *argv[ ])
{
  int              i, j, Ii1, Ii2, Jj1, Jj2, Ycen, Zcen, rY1, rY2, rZ1,
rZ2, Radius;
  unsigned short     image5_1[Z_SIZE2][Y_SIZE2];
  unsigned short     filter[Z_SIZE2][Y_SIZE2];
  unsigned short     filtered[Z_SIZE2][Y_SIZE2];
  unsigned short     reflectance[Z_SIZE2][Y_SIZE2];

  char   source5_1[80];
  char   source_f[80];
  char   source_fed[80];
  char   source_ref[80];
  strcpy  (source5_1, *++argv);
  strcpy  (source_f, *++argv);
  strcpy  (source_fed, *++argv);
  strcpy  (source_ref, *++argv);

  image_read(image5_1, Y_SIZE2, Z_SIZE2, source5_1);

  for(j=0; j<Z_SIZE2; j++)
    {
      for(i=0; i<Y_SIZE2; i++)
        {
          if(image5_1[j][i] > 20)
            Jj1 = j;
        }
    }

  for(j=Z_SIZE2-1; j>0; j--)
    {
      for(i=0; i<Y_SIZE2; i++)
        {
          if(image5_1[j][i] > 20)
```

```
        Jj2 = j;
     }
  }

for(i=0; i<Y_SIZE2; i++)
  {
   for(j=0; j<Z_SIZE2; j++)
     {
       if(image5_1[j][i] > 20)
         Ii1 = i;
     }
  }

for(i=Y_SIZE2-1; i>0; i--)
  {
  for(j=0; j<Z_SIZE2; j++)
    {
     if(image5_1[j][i]>20)
       Ii2 = i;
    }
  }

Ycen = (Ii1 - Ii2)/2 + 0.5 + Ii2;
Zcen = (Jj1 - Jj2)/2 + 0.5 + Jj2;

rY1 = Ii1 - Ycen;
rY2 = Ycen - Ii2;
rZ1 = Jj1 - Zcen;
rZ2 = Zcen - Jj2;

Radius = (rY1 + rY2 + rZ1 + rZ2) / 4 + 0.5;

photo( image5_1, filter, filtered, reflectance, Ycen, Zcen);

image_write( filter, Y_SIZE2, Z_SIZE2, source_f );
image_write( filtered, Y_SIZE2, Z_SIZE2, source_fed );
```

```
  image_write( reflectance, Y_SIZE2, Z_SIZE2, source_ref );

}
/*-----------------------------------------------------------------*/
  photo (raw1, raw2, raw3, raw4, Ycen, Zcen, Radius)

  unsigned short  raw1[Z_SIZE2][Y_SIZE2];
  unsigned short  raw2[Z_SIZE2][Y_SIZE2];
  unsigned short  raw3[Z_SIZE2][Y_SIZE2];
  unsigned short  raw4[Z_SIZE2][Y_SIZE2];
    {
      int    i, j, y, z, theta, PP, QQ;
      float   radius, cp0,sp0,cps,sps,ct0,st0,cts,sts, factor ;
      float   p, q, theta_m, phi_m,cpm,spm,ctm,stm ;
      float   inc_angle, emt_angle, phase_angle ;
      float   cia, cea, b, c, d, e ;
      float   Fnpa, Fn30 ;
      float   factor2,factor0,factor1,pa ;
      float   P, Q, ew, ns, ratio_ref;

      cp0 = cos (Phi0/pi);
      sp0 = sin (Phi0/pi);
      cps = cos (Phi_s/pi);
      sps = sin (Phi_s/pi);
      ct0 = cos (Theta0/pi);
      st0 = sin (Theta0/pi);
      cts = cos (Theta_s/pi);
      sts = sin (Theta_s/pi);

      ns = NS/pi;
      ew = EW/pi;

      for (j=0;j<Z_SIZE2;j++)
        {
          for (i=0;i<Y_SIZE2;i++)
            {
```

```
            y = i - Ycen;
            z = j - Zcen;

            radius =  pow (y,2) + pow(z,2);

            if (radius > Radius*Radius )
              raw2[j][i] = 0;
            else
              {
                p=y*cos (t/pi) - z*sin (t/pi) ;
                q=y*sin (t/pi) + z*cos (t/pi) ;
                phi_m   = asin (-q/Radius) + Phi0/pi ;
                theta_m = asin (p/sqrt(Radius*Radius-q*q)) + Theta0/pi;

                if((phi_m < ns*0.99) && (phi_m > ns*1.01))
                  {

                    if((theta_m > ew*0.99) && (theta_m < ew*1.01))
                      {
                        printf("theta_m=%f    phi_m=%f¥n",  theta_m  ,
phi_m);
                        P = p + Ycen;
                        Q = q + Zcen;
                        printf("P=%f  Q=%f¥n", P, Q );
                      }
                  }

                cpm = cos (phi_m);
                spm = sin (phi_m);
                ctm = cos (theta_m);
                stm = sin (theta_m);
                inc_angle = acos (cpm*cps*(ctm*cts+stm*sts)+spm*sps) ;
                emt_angle =acos (cpm*cp0*(ctm*ct0+stm*st0)+spm*sp0) ;
                phase_angle=acos(cps*cp0*(cts*ct0+sts*st0)+sps*sp0) ;
                cia = cos (inc_angle) ;
                cea = cos (emt_angle) ;
```

```
          b = a1*(phase_angle*pi) ;
          c = a2*pow((phase_angle*pi),2) ;
          d = a3*pow((phase_angle*pi),3) ;
          e = a4*pow((phase_angle*pi),4) ;
          pa = phase_angle*pi ;
          c = a2*pow(pa,2) ;
          q = a3*pow(pa,3) ;
          e = a4*pow(pa,4) ;
          Fnpa = a + b + c + d + e ;
          Fn30=a+a1*30+a2*pow(30,2)+a3*pow(30,3)+a4*pow(30,4) ;
        factor0=(Fn30*cos(30./pi)/(cos(0./pi)+cos(30./pi)))  ;
          factor1  = (Fnpa*cia/(cea+cia)) ;
          factor   =  factor0/factor1 ;
          if (inc_angle >= 1.570796)  factor   =  0. ;
          /*  factor2 = inc_angle*1000.  ;          */
          /* factor2 = emt_angle*1000.  ;          */
          /* factor2 = phase_angle*1000.  ;        */
          factor2 = factor*1000. + 0.5  ;
          raw2[j][i] = factor2;
          factor2 = factor * raw1[j][i] + 0.5  ;
          raw3[j][i] = factor2;

        }
    }
}

PP = P + 0.5;
QQ = Q + 0.5;
ratio_ref = 0;

ratio_ref = (float)(Reflectance / raw3[QQ][PP]);

for( j = 0; j < Z_SIZE2; j++ )
  {
    for( i = 0; i < Y_SIZE2; i++ )
      {
```

```
      raw4[j][i] = (raw3[j][i] * ratio_ref) * 1000 + 0.5;
    }
  }
```

第6章　結果及び考察

6-1　使用カメラによるデータ格納の違い

本研究で開発するプログラムの対象画像は赤外カメラと可視カメラであることは第 4 章でも述べたが、実際に画像処理を行った結果、可視カメラで撮像された画像ファイルの読み込みにおいて問題が生じた。どちらのカメラで撮像された画像も 1 ピクセル 2 バイト（16 ビット）として扱われる。プログラムは UNIX 上で実行され、読み込まれたデータはメモリに格納される。このとき本来、UNIX の OS 上では 2 バイトのデータは上位の 1 バイトを先に格納し、次に下位の 1 バイトを格納するのだが、可視カメラで撮像した画像の場合、上位と下位のデータの格納順が逆転し、読み込みの際、メモリの 1 バイト目に下位の 1 バイトのデータが格納され、メモリの 2 バイト目に上位の 1 バイトが格納される。出力する際は、格納された通りの順で出力される。この結果、格納された順にデータを読み取り数値として出力した場合、本来持つ輝度値とは異なる数値が表示される。当然、一度プログラムで読み込み出力した画像も誤った画像となる。この問題を解消するために、読み込んだデータの上下のバイトを交換し再度読み取る方法などを試みたが、データの格納についての問題は解消されていない。赤外カメラで撮像された画像ではこの問題は生じないことから、撮像時に使用するソフトによるものと考えられる。

6-2 補正画像

実際にプログラムで処理や補正を行った月面画像を図 6-2-1～図 6-2-9 に示す。

図 6-2-1 はライトフレーム画像を 10 枚重ね平均したものである。画像全体に白くもやがかかったように見えるが、これがダークノイズ（暗電流）と呼ばれるものである。この画像をダークフレーム処理したものが図 6-2-2 になる。ノイズが取り除かれ鮮明な画像を見ることができ、月面の地形などもはっきりと写し出されている。

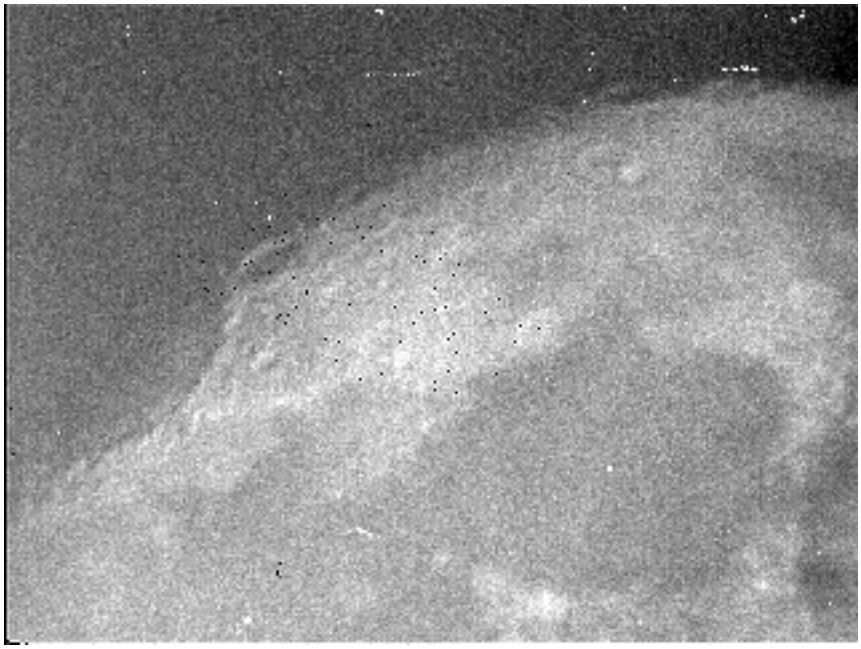


図 6-2-1　ライトフレーム画像（ダークフレーム処理前）

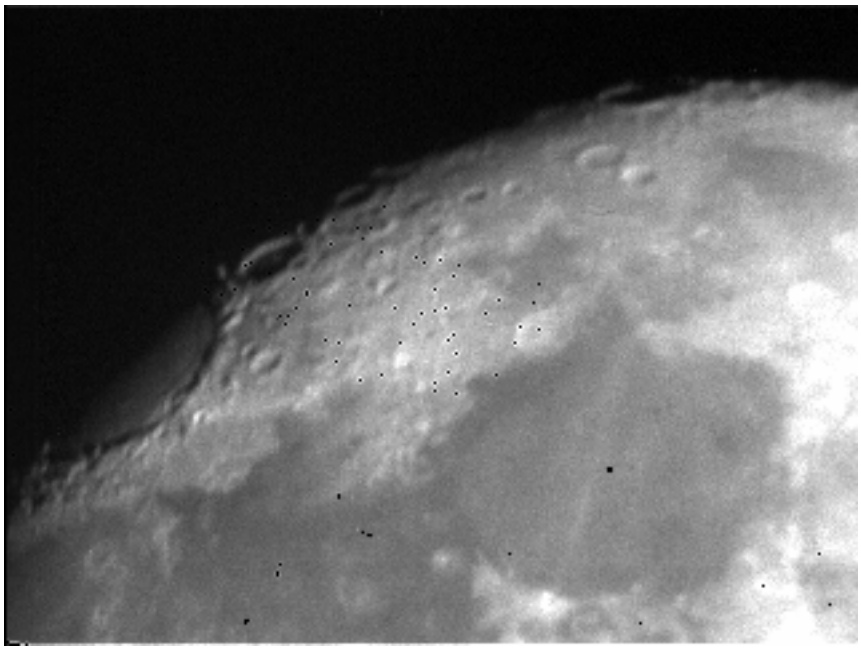


図 6-2-2　ダークフレーム処理済画像

図 6-2-2 ではダークノイズは取り除かれたが、画像のところどころに黒い点のようなピクセルが見られる。これは感度の無いピクセルによるもので、輝度値は 0 である。このようなピクセルをデッドピクセルと呼ぶ。デッドピクセルは以降の処理や補正での演算処理で障害となるため、ピクセル補間を第 4 章 4-1-2 の（式 2-2）に従って実施した。デッドピクセルを補間した画像を図 6-2-3 に示す。

また、光量ムラを取り除くフラットフィールド補正後の画像を図 6-2-4 に示す。

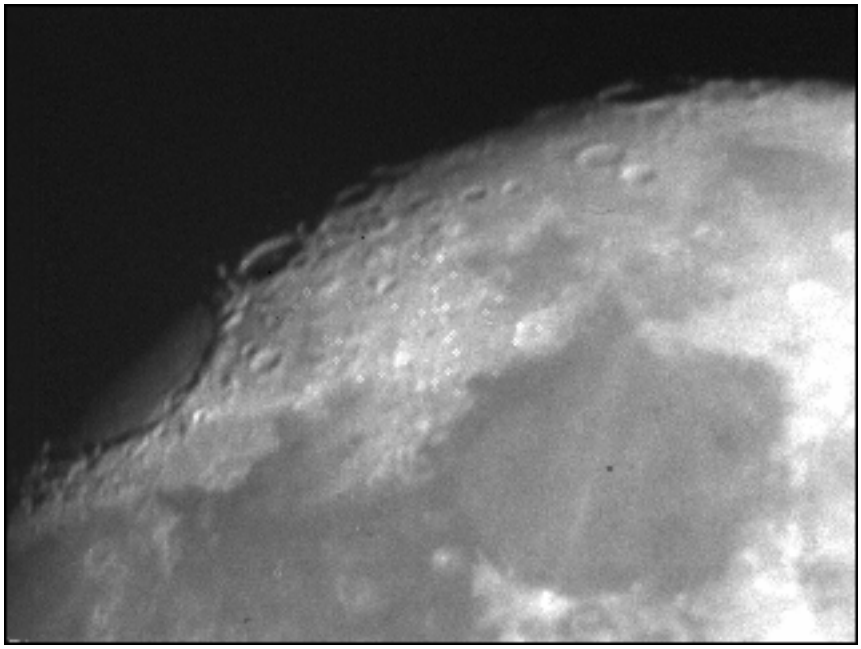


図 6-2-3　ピクセル補間後画像

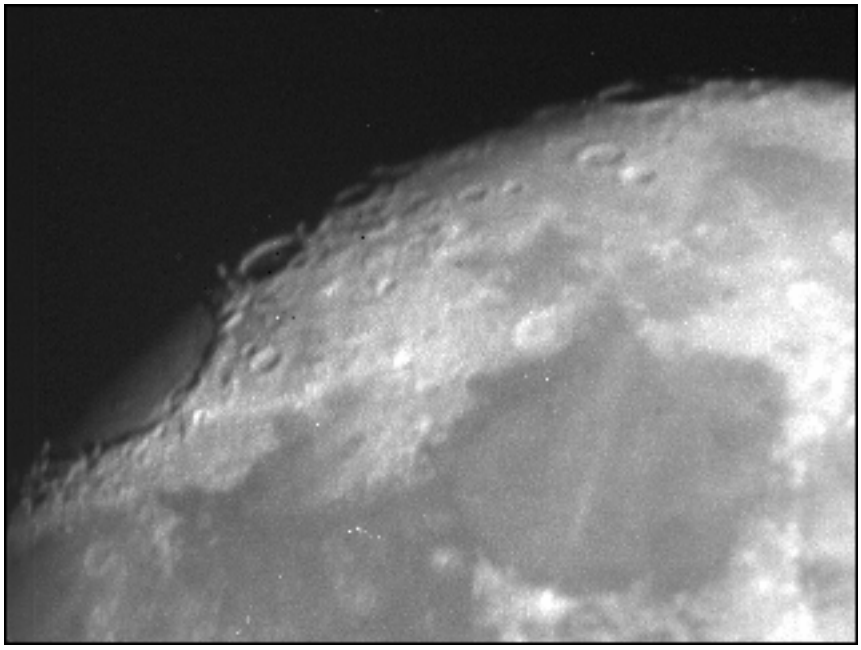


図 6-2-4　フラットフィールド補正済画像

大気差を減算した結果の画像を図 6-2-5 に、演算処理の結果を表 6-2-1 に示す。画像から著しい輝度値変化は見られないが、数値を見ると sampleNo.85～No.92 までのピクセルの輝度値が 0 となっていることがわかる。これらのピクセルは画像上でバックグラウンドにあたる部分であり、フラットフィールド補正後の段階では大気による影響を受け、数 10～数 100 の輝度値を持っていたが、大気差補正により本来の輝度値 0 となった。尚、sample とは画像の幅を表しており、line は画像の高さを表している。lineNo.90 における、sampleNo.85～104 の範囲は、大気部分（バックグラウンド）と月面部分を含む範囲である。

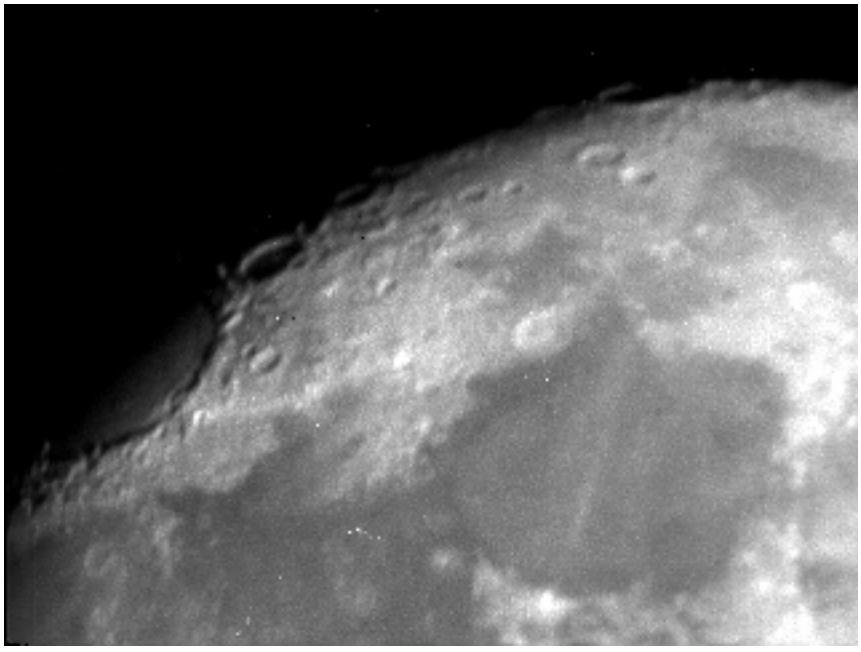


図 6-2-5　大気差補正済画像

表 6-2-1　大気差補正における演算処理結果

sample	line	-pre_total	light(/flat)	pre_total
85	90	0	59	67
86	90	0	62	67
87	90	0	62	67
88	90	0	67	67
89	90	0	63	67
90	90	0	59	67
91	90	0	63	67
92	90	0	61	67
93	90	10	77	67
94	90	40	107	67
95	90	79	146	67
96	90	113	180	67
97	90	147	214	67
98	90	160	227	67
99	90	152	219	67
100	90	135	202	67
101	90	127	194	67
102	90	114	181	67
103	90	83	150	67
104	90	71	138	67

sample：画像の幅
line：画像の高さ
-pre_total：大気差補正後の輝度値
light(/flat)：フラットフィールド後の輝度値
pre_total：大気差

輝度値補正、座標の変換後の画像をモザイクし月面全体を再現したものを図 6-2-6 に示す。画像から分かるように、月面際がところどころ欠けている。各画像の中央部分 80%をモザイクに用いているため、全体をカバーすることができなかった。しかし、月の大気際付近は歪みを含んでいるため、解析には適さないことから、完全な満月を再現する必要はない。また、モザイクでのつなぎ目が見て取れる。これは、輝度値補正の段階で、基準画像との比率（ratio）を求めた結果、ratio = 1 とならない画像が存在するために起きた結果であると考えられる。この原因としては、バックグラウンドを含む画像は大気差補正において画像全体から 40～60 の輝度値減算が行われたが、バックグラウンドを含まない画像では大気差補正が行われていない。そのため、バックグラウンドを含まない画像は、大気差補正を行った画像に比較して、輝度が 40～60 多く入っており、比率に影響したと考えられる。このような階調差を生じさせない方法として、大気差補正で減算された数値の平均を求め、バックグラウンドを含まない画像から減算する手法があげられる。

一般的な画像のモザイク処理では、画像境界を目立たなくするため、画像境界を平均した値を入力する作業が行われる。しかし、見た目の完成度よりも、画像間の比率や演算後の輝度値が把握したいため、画像境界を消す作業を今回は実施していない。

図 6-2-6　モザイク済画像
輝度値補正後の画像（図 6-2-5）を 23 枚モザイクして作成した月表側全面の画像
中心波長 1050[nm]の狭帯域フィルター使用。
2002 年 11 月 21 日、齋藤正晴氏による撮像。

入射角（i)、観測角（e)、位相角（α）をそれぞれ i=30°, e=0°, α=30°として統一した場合の方向性輝度値補正係数のフィルター画像を図 6-2-7 に示す。 この観測日（2002 年 11 月 21 日）における月齢、月面画像のみかけの中心（θ_0, ϕ_0)、太陽が真上にある月面経緯度（θ_s, ϕ_s）は次のとおりである【7】。

月齢　　‥‥‥‥‥‥‥‥‥‥‥‥17 日

（θ_0, ϕ_0）‥‥‥‥‥‥‥‥‥‥‥‥8° W,　3° S

（θ_s, ϕ_s）‥‥‥‥‥‥‥‥‥‥‥‥14.3° E,　0.36° S

フィルター画像を乗算し測光補正を施した画像を図 6-2-8 に示す。

図 6-2-7　測光補正フィルター画像

図 6-2-8　測光補正済画像

測光補正済みの画像から、Apollo16 号が採取したサンプル 62231 の較正サイトの輝度値を決定し、輝度値が 62231 の反射率 20.796%となるような比率を算出し、画像の全ピクセルに乗算して標準反射率に変換した画像を作成する。

このとき、62231 の採取地点（較正サイト）は 15.1°E, 9.0°S で、画像上での座標（x,y）は(597, 470)であった。このピクセルの輝度値と 62231 の反射率から求めた比率(ratio_ref)は ratio_ref = 0.000731。この比率を画像の全ピクセルに乗算して標準反射率に変換した画像を図 6-2-9 に示す。図 6-2-9 は、比率を乗算したのち、画像全体に*100 という計算を施したものである。プログラムで画像のデータ型は 16 ビットで宣言されているが、比率を乗算した値は小数点以下に有効数字があるため、浮動小数点を出力の際データ型は 32 ビットとなってしまう。画像の入力、出力はすべて 16 ビットで宣言されている場合、32 ビットのデータを出力しようとするとエラーが生じてしまう。そこで、比率を乗算した値にさらに 100 を乗算することで、16 ビットのデータ型で出力できるレベルに持っていく必要があるのだ。標準反射率への変換は、画像解析において画像そのものではなく、数値を必要とするものであるため、図 6-2-9 のようにあえて画像として出力する必要はない。したがって、解析に標準反射率へ変換した値を用いる際は、乗算した 100 を除算した数値を使用すればよいのである。

0% 40%

図 6-2-9　1050[nm]の反射率マップ

第7章　まとめ

7-1　本研究の評価と今後の課題

作業の効率化を目的の一つとして開発された今回のプログラムによって、実際の作業時間を想定すると、これまでの月面画像の処理は約 1 ヶ月を要していたのに対し、本研究のプログラムを使用した場合、プログラム実行前に必要なデータ入力の時間を含めおおよそ数 10 分で解析に用いる画像の処理を完了することが出来ると考えられ、目的を十分に果たしていると言える。もう一つの目的である、画像処理プログラムの使用一般化の点は、UNIX 上でプログラムを実行可能であることから、複数のユーザーが同時に本研究で開発した画像処理プログラムを使用することで解決される。また、ユーザーが自由にプログラムを書き換えることが出来るため、処理の過程における演算結果のデータを必要に応じて出力できるため、解析に不可欠な数値の議論をより深めることにつながる。

プログラムの開発は、画像処理をより快適に行うことを当初の目的としていたが、画像解析で重要な撮像用カメラの特性を知るために必要なデータ解析の効率化などにも効果的であると考えられる。

今後の課題としては、現在検討中の問題である可視カメラで撮像した画像の処理にあたってのデータ格納の違いという問題の解決である。本研究では、可視カメラ、赤外カメラどちらのカメラを用いた場合でも画像処理可能なプログラムを開発することが本来のテーマであったため、この問題の解決は不可欠である。C 言語が得意とする 1 バイトごとのデータ操作によるデータの並べ替えを試みたが、未だ解決されていない。よって、C 言語のプログラムでの操作ではなく、UNIX コマンドによるデータ格納の操作を考える必要がある。

作業の効率化が可視、赤外カメラのどちらの画像でも図れるよう、より優れたソフトウェアへと改良を加えていくことが今後の課題であり、目標である。