広島大学望遠鏡用全天スカイモニターと自動観測スケジュール機能の開発

保田 知則
広島大学理学部物理科学科
1479066B
高エネルギー宇宙・素粒子実験研究室
主査 深沢泰司　副査 両角卓也

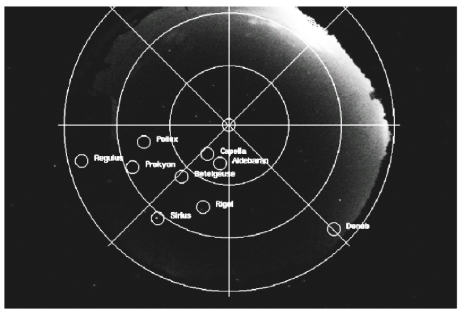
2005年 2月10日

概 要

我々は 1.5m 可視近赤外線望遠鏡を用いてガンマ線衛星 GLAST や X 線衛星 SUZAKU と連携した高エネルギー天体の多波長観測研究を推進している。高エネルギー天体は突発的、一過性のものが多く、観測を効率的に行う為には、空の雲の状況をリアルタイムで判定するスカイモニターの実装と、次の観測対象として最も有効な天体を選定するスケジュール機能とが求められる。そこで私は独自に全天スカイモニターを製作すると共に、スケジュール機能の開発を試みた。

全天スカイモニターは、魚眼レンズや画像積分可能な小型 CCD カメラ等で構成され、linuxPC で制御される。私はその筐体を含む装置設計、組み上げ、データ取り込み用ソフトウェア開発を行った。設計では低価格でかつ性能面においても信頼性のあるものを目指した。 組み上げ後、国立天文台岡山に既設のスカイモニターと同時観測を行うなどして、所期の性能を持つことが確認できた。

これまでに東広島天文台サイトにおいて、全天の雲の変動の様子を複数晩にわたり撮影した。画像では月のある無しに関わらず雲が判定できると共に、3 等星程度の恒星まで写っている。私は天体位置計算ソフトを自作して、各画像上の観測天体位置を表示し、目視で雲領域と比較可能にした。 自動観測に用いるには雲領域を自動的に検知してマッピングし、観測可能天体を自動判定できるようなソフトウェア改良が求められる。



目 次

第1章　目的

1.1　広島大学東広島天文台計画

国立天文台三鷹構内に設置されていた赤外シミュレーター (口径 1.5m 光学赤外線望遠鏡) が 2004 年に広島大学に移管され、2006 年 4 月に東広島市内に移設される予定となっている。この赤外シミュレーターは、国立天文台ハワイ観測所にある大型光学赤外線望遠鏡「すばる」で使用する観測装置の開発・評価をする目的で建設されたものであるが、すばるの第一期観測装置開発がすべて終了したのち、シュミレーターの需要が減ってきていることも考慮され、本格的な天文・宇宙研究へ活用する計画を提案した広島大学に移管された。現在、全国的にも高いレベルのシーイング環境を有したサイト (福成寺) にて移設準備が進められている。この赤外シミュミレーターを用いた主な研究目的は、アラート対応の高エネルギー突発天体観測を行うことである。具体的には 2007 年に NASA から打ち上げられるガンマ 線衛星 GLAST 衛星などが天体を検出したときに発信するアラートから天体の座標を取得し、可視光で追跡観測することを計画している。また、GLAST や 2005 年に宇宙航空研究開発機構から打ち上げられたX線衛星 SUZAKU と連携することで同時多波長観測 (可視〜ガンマ線) を可能とし、宇宙における様々な高エネルギー宇宙現象の研究分野に貢献することも目的としている。

1.2　地上望遠鏡による観測の流れ

1.2.1　1.5m 可視近赤外望遠鏡 (赤外シミュレーター)

1.5m 望遠鏡 (赤外シミュレーター) はすばる望遠鏡で使用される観測装置の実験や、すばる望遠鏡立ち上げの予行演習の他に、赤外線や可視域における天体観測も考慮して建設された口径 1.5m の反射望遠鏡である。この望遠鏡は 10 分間の露光時間の分光観測で約 17 等級、測光観測では約 20 等級までの観測が可能である。この望遠鏡で狙う主なサイエンスは、高エネルギー天体における物理過程の解明である。高エネルギー天体には、ガンマ線バーストなどの急激な時間変動を示す突発現象があり、可視光でもその様子を観測することができる場合がある。突発天体は、いつどこで起こるか分からないため、現象発見直後を観測することは難しい。広島大学では、本格的な望遠鏡ながら小型であるという長所を活かし、これら突発天体に特化した望遠鏡化を目指す。架台・駆動系は実績のある名古屋大学と協力して突発天体発生後 30 秒以内の観測が可能な機動性の獲得を目標にしている。以下図 1.1 と表 1.1 が 1.5m 望遠鏡の全体図および仕様である。

1.2.2　通常観測の流れ

通常、望遠鏡は降雨や湿気、風を防ぐ為に自動開閉式のドームの中に収められている。観測者は夕方の薄明時に現在晴れているかを確認し、晴れていればドームを開け、晴れて

図 1.1: 1.5m 望遠鏡 (赤外シミュレーター) 全体図

光学系	Ritchey-Chretien 光学系
主鏡	有効径 1500 mm 主鏡の F 比 F= 2.0
焦点モード	カセグレン焦点 1 つ ナスミス焦点 1 つ 合成 F 比 カセグレン、ナスミス共に F = 12.2 焦点距離 18,300mm
星像の分解能	1′′FWHM
視野	$15'\phi$
最大駆動速度	1° /sec 以上
最大加速度	0.5° / $\sec^2$ 以上
架台の方式	経緯台方式

表 1.1: 望遠鏡の仕様

図 1.2: 岡山天体物理観測所全天スカイモニター全体図

いる領域での観測可能天体を把握する。そして複数の候補天体に対して観測のスケジューリングを行う。観測者は手動で望遠鏡を操作し、観測スケジュールに沿って星に望遠鏡を向け、翌朝の薄明開始まで観測を行うのである。その間、適時観測者は気象情報や実際の空を見ることで天候や空の状況を把握し、観測スケジュールの修正を行うこともある。

1.3 即応自動観測と遠隔観測

我々は天文台計画の一貫として、効率的な観測を行うことを目的とした観測の自動化、さらに天文台に常駐しなくとも広島大学構内において観測が行えるよう遠隔化を目標にしている。その為には望遠鏡の操作と制御、周辺機器の自動化を行う必要がある。周辺機器には観測天体を天体の移動に応じて自動追尾するオートガイダーや天文台サイトの温度、湿度などといった情報を観測者に提供する気象モニター等がある。その中でも私は全天スカイモニターの開発を行った。この装置は天文台サイトの空の様子を監視し、雲を自動検知することにより観測可能天体を特定し、観測のスケジュールを自動で組み、さらに適時調整を行うものである。これにより人が適時ドーム外へ出て暗闇に目を馴染ませた後、空の様子を見るという一連の手間と危険を回避できるだけでなく、観測の自動化、遠隔化の大きな進歩となる。

岡山天体物理観測所に同様のスカイモニターがあり (図 1.2)、こちらは空の画像を 10 分に 1 枚撮影し、気象データとも併せて後日のデータ解析の際の参考資料に用い、それをウェブブラウザに掲載し毎回更新されている。さらにその一部はアーカイブに保存される。私が開発する全天スカイモニターは雲を検知したり、観測スケジューリング機能を持たせるという点において、岡山のものをさらに発展させたものとなっている。

1.4 全天スカイモニターと要求される機能

全天スカイモニターは、天文台上空をほぼ収められる広い視野と高感度を兼ね備えているものである必要がある。また、周辺機器であり予算に余裕がないことから、できるだけ安価に抑えることを目標にする。

一方、本開発にあたり C 言語、API(application programmers interface) である GTK(GIMP TOOL KIT) による機器の制御、ソフトウェア開発について学ぶことも目的のひとつである。全天スカイモニターは、撮像、PC への画像表示及び画像の自動更新機能、雲の自動

検知機能を有し、さらに本開発の中軸である観測天体位置の特定とその位置の雲の有無、雲の流れの予測による観測スケジュール機能を有したものが求められる。その結果がすべて望遠鏡や他の周辺機器にフィードバックされることで観測が自動化されるのである。さらに、これらの機能は一つのソフトウェアとして統括されることにより、観測地外で遠隔操作可能となることが期待される。また本来の目的からは外れるが、さらに全天スカイモニターの用途として昼夜自動撮影を可能にし、仕事や余暇などで現地の天候をリモートで判別したい市民を対象とした地域市民支援目的の要素も盛りこむ。

本研究の主目的は、広島大学独自の全天スカイモニターと観測スケジューリングソフトウェアの開発であり、これにより突発天体に即応した追跡観測にとって効果的となる即応自動観測化と遠隔化を大きく進める事が期待されるものである。

第2章　観測装置の製作

本論文で開発される全天スカイモニターは、天文台建設後は定期的な点検、部品交換を行うとき以外は安定に稼働するものでありたい。

なお、本全天スカイモニターのハードウェアの設計には約1週間、物品の調達と加工に1ヶ月、組み上げに数日ほどを費した。

2.1　装置構成

図 2.1: 広島大学全天スカイモニター

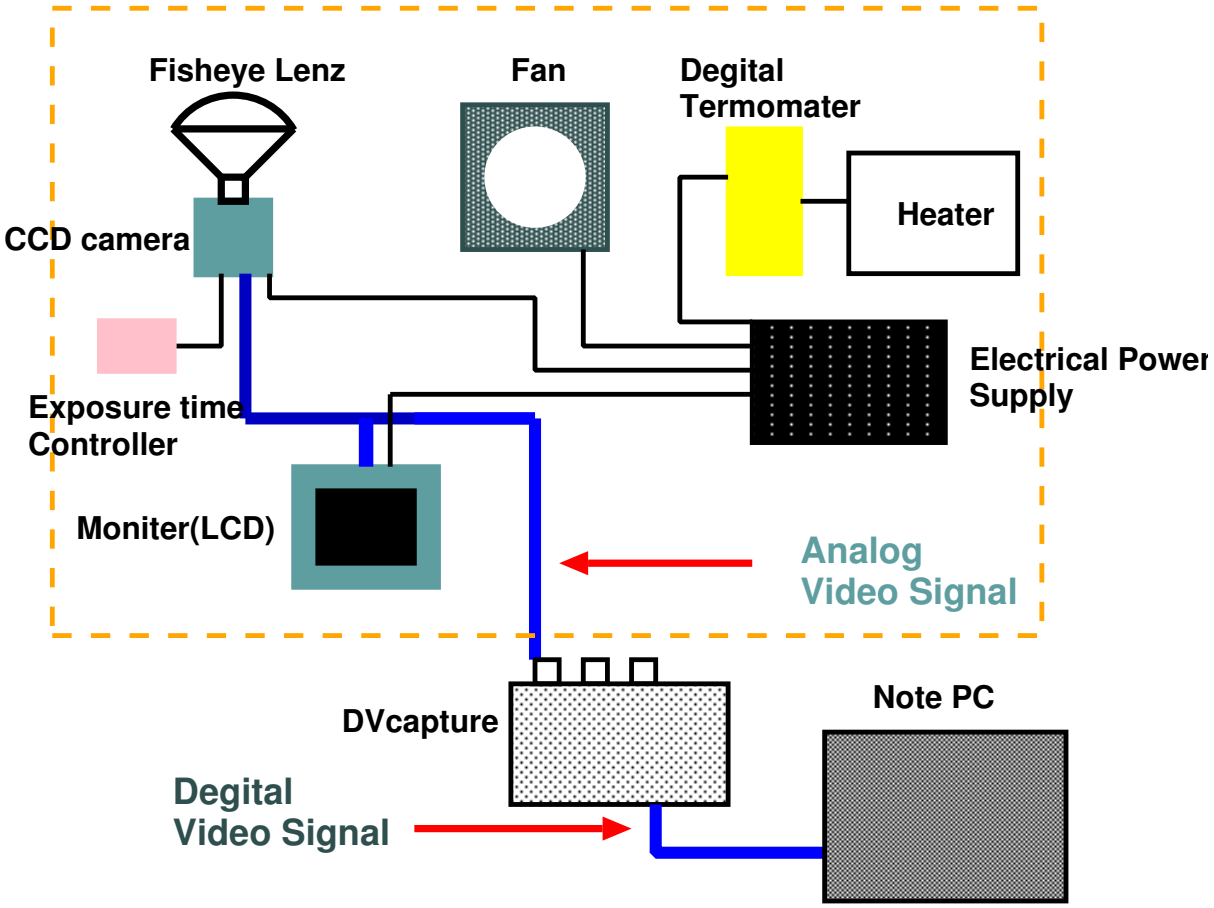


図 2.2: 装置構成図

図 2.1 が製作した全天スカイモニターの全体図である。内部の装置構成 (図 2.2) は破線で囲まれる部分が筐体内部であり、内部には検出部となる魚眼レンズと CCD モノクロカメラを組み合わせたもの、12V 電源、監視用モニター、温度センサー、ファンが収納されている。検出部と周辺機器は 12V 電源に全て接続され、太線のケーブルは CCD モノクロカメラからの信号の経路を表している。出力されたアナログビデオ信号は DV キャプチャーによりデジタル信号に変換され、インタフェース IEEE1394 を介し、DELL の B5 のノートパソコンに送られる。

DV キャプチャーは CCD カメラから出力されたアナログビデオ信号をデジタル信号に変換する機器であり、ここではカノープス ADVC － 100 を用いた。

2.2 光学系と検出器及び周辺機器

2.2.1 検出部

全天スカイモニターの光学系と検出部は低価格で広視野で高感度な信頼性のあるものを目指した。レンズには 185 °の広視野で高解像度である 1/2 インチサイズ用 C マウント魚

図 2.3: 魚眼レンズ

図 2.4: WAT-120NCCD モノクロカメラ

眼レンズ (Fujinon　FE185C057HA-1) を使用している (図 2.3)。夜間に図 2.1 のように全天 (2πstr) の撮影をするのに十分であり、さらに 20 万円程度と安価である。 高解像度であるために空冷型の高感度 CCD モノクロカメラ (WAT-120N, 図 2.4) と組み合わせることが可能である。このカメラは有効画素数 768×494 画素である。リモコン操作が可能であり、CCD カメラの積算時間と受光量の増幅率であるゲインの調整、ガンマ特性の変更が可能である。魚眼レンズに入射する光量が多過ぎるため本来 CCD カメラでは撮像不可能な昼間の空も撮像可能になるであろう。一日中自動測定ができ、その画像をウェブで公開することで地域社会支援にもつながる。この CCD モノクロカメラも 4 万程度と安価なものを選定してある。ここでガンマ特性について述べておく。CCD カメラに入射する光量と撮像するデータとのピクセルごとの信号の比はある比例関係をもっており、$y = x^{\gamma}$ と表される。この γ は通常 $\gamma = 1$ であるが、これを $\gamma = 0.45$ などに変更することで明るい領域の色の諧調が鮮やかになる。将来的には、リモコンを改造し計算機からリモートで変更できるようにする。

魚眼レンズと CCD カメラを組み合わせたものが検出部であるが、デフォルトのマウントではフォーカスが合わなかったため、2.3m 厚のスペーサーを理学部金属加工室に依頼して製作し、取り付けた。季節の変化等状況に応じてフォーカスの調整を行う可能性があるためスペーサーは 2.1, 2.2,⋯,2.5mm の計 5 個を製作した。

メーカー	Fujinon
焦点距離	1.4 (mm)
画角 (H×V)	185° ×185°(ϕ4.6mm)
レンズしぼり	F/1.4∼F/16

表 2.1: 魚眼レンズの仕様

メーカー	WATEC
有効画素数	768(H)×494(V)
ユニットセルサイズ	8.4μm(H)×9.8μ m(V)
走査方向	2：1 インタレ-ス
解像度	570TV 本
最低被写体照度	0.001Lux F1.4
AGC(ゲインコントロール) (S/N 比)	8∼38 dB 52 dB (OFF 時)
ガンマ特性	High≃0.45 Low≃0.6 OFF≃1.0
積算時間	OFF, 0.033, 0.067, 0.13, 0.27, 0.53, 1.1, 2.1, 4.3, 8.5 (sec)

表 2.2: CCD モノクロカメラの仕様

2.2.2 筐体

全天スカイモニターは天文台屋上に設置する予定である。屋外にさらすため、湿気、結露、降雨などに対する対策と温度調節が必要となる。気象用に作られた密封性の高い密封ケース (図 2.5) に監視用の半球の透明アクリル製ドーム (図 2.6) を接着したものに装置全般を収納し、湿気や降雨の対策を取った。

さらに結露の対策として筐体を暖めるヒ-タ-を設置し、そのスイッチを ON/OFF 切替え可能であるデジタル温度コントローラ (図 2.7) と、効率的な暖房をするためにファン (図 2.9) をとりつけた。ヒ-タ-にはシリコンラバ-ヒ-タ-(図 2.8) を採用した。デジタル温度計の温度センサーはアルミ基板に設置する。ファンには 11 年間使用可能なものを選定した。

図 2.5: 気象用密封ケース

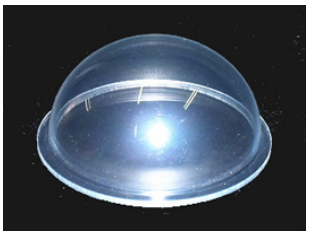


図 2.6: 透明アクリルドーム

材質	プラスチック
寸法 (W×H×L)	290×190×340 (mm)

表 2.3: 気象用密封ケースの仕様

材質	アクリル
メ-カ-	はざい屋
形状	厚み 2 mm 外径 100 ϕ フランジ巾 10 mm

表 2.4: 透明アクリルドームの仕様

図 2.7: OMRON 温度制御用デジタルサーモ

図 2.8: 坂口電熱シリコンラバーヒーター

図 2.9: DC ファンモータ

2.3 装置設計と製作、組み上げ

全天スカイモニターは、密封ケースに透明アクリルドームを接着した筐体に、魚眼レンズ、モノクロ CCD カメラを組み合わせたものと周辺機器を収納する。密封ケースへの追加工および部品を装着するためのアルミ基板の設計は、3 次元 CADSolid Works を用いて行った。アクリルドームはフランジ巾にネジ穴を空けネジにより固定した。雨水の侵入を防ぐためドームフランジ巾の外側 3mm の密封ケースとのすき間に O リング (ニトリル製) をはさんだ。また液体防水シールをネジ穴と、フランジ巾とケースのすき間に流しこんだ。

機器は密封ケース内にアルミ基板を設置し、そこにネジで接合した。設計に関しては Solid Works を用いて応力シミュレーションによる耐荷重テストや、効率的な空気循環を考慮したアルミ基板の設計を行った。

・Solid Works

部品モデルを描く、図面を書くなどに便利なソフトウェアであり、ネジ穴や板の切抜きなどが容易にでき、応力による歪みのシミュレーションも行える。以下図 2.10 が設計したアルミ基板である。Solid Works では作成した部品モデルを合成することもでき、光学系を設置する光学台を合成した図が図 2.11 である。アルミ基板の設計図を (図 2.12) に載せる。

密封ケース、アルミ基板の加工は広島大学理学研究所特殊加工技術開発室金属素材応用部門に委託した。私はケース、アルミ基板の設計及び装置の組み上げを主導で行った。

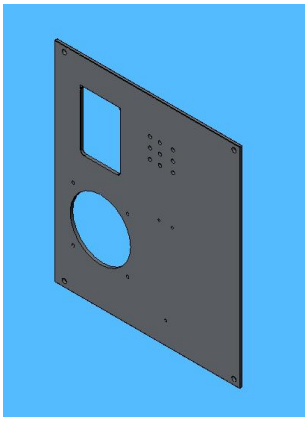


図 2.10: アルミ基板

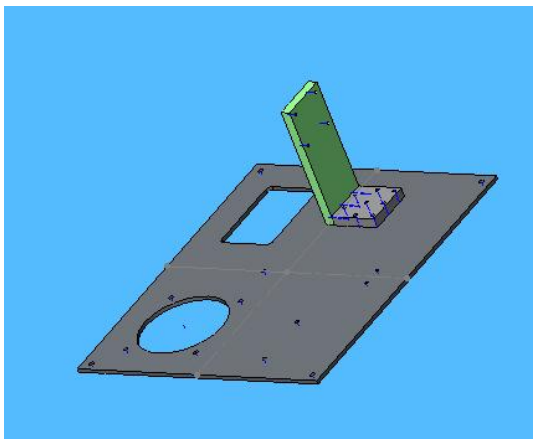


図 2.11: アルミ基板と光学台の合成図

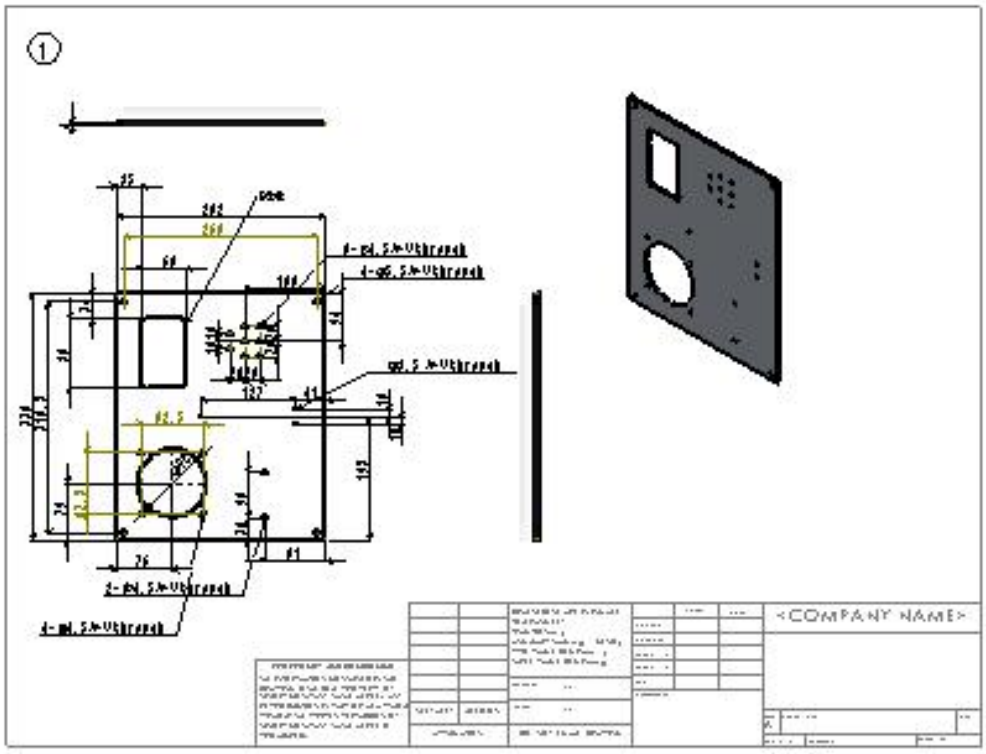


図 2.12: 図面

図 2.13 は完成した筐体の内部である。12V 電源が中心に据え付けられている (図 2.14)。表 2.5 はその仕様である。装置全般はこれに接続される。12V 電源の入力電圧は 100V であるため、まだ高圧電源が通じていない場所での観測時には 12V 電源バッテリーから 100V インバーターを介して電気を取ることになる。

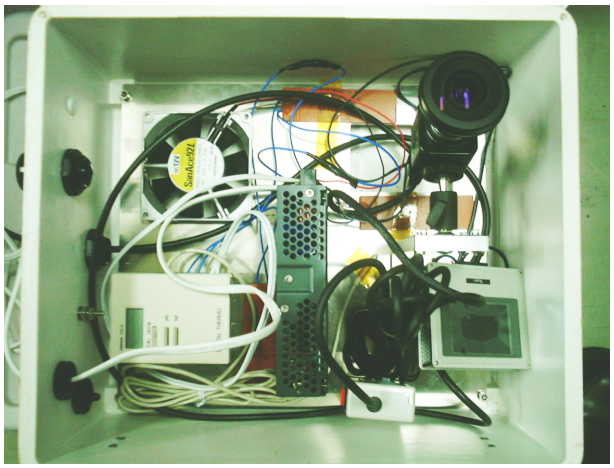


図 2.13: 筐体内部

図 2.14: 12V 電源 EWS50-12

メーカー	DENSEI-LAMBDA 社
定格直流出力電圧	12V
定格直流出力電流	4.4A
入力電流	1.2A (100VAC)
動作周囲温度	-10 ℃ ～+60 ℃
冷却方法	自然空冷
質量	450g

表 2.5: 電源 EWS50-12 の仕様

2.4 データ取得ソフトウェア開発

私は試験稼働に際し、データ取得のための簡易なソフトウェア開発を主としてC言語を用いて行った。作成されたソフトウェアは全天スカイモニターのソフトウェアの根幹となる PC への画像取込と画像表示、画像の自動更新、アーカイブに自動で保存する機能を有するものである。

今回の試験では10秒ごとに画像を取り込み、ppm形式の1枚画像に変換し、パソコン上にそれを表示する。ソフトの調整を行うために10秒ごとに画像を取り込む必要があるが、後の画像上の雲や天体を用いた画像処理は一定時間ごとの画像で十分である。そこでパソコンのディスク空き容量の関係上画像は表示後に破棄し、3分ごとに画像を表示後アーカイブに保存するものとした。以下はそのソフトウェア開発で用いたものである。

2.4.1 OS:Fedora Core2(kernel 2.6.9-FC2)

FedoraはRedHat社がRedHat Linux9を最後にコンシューマ向けLinuxの開発・販売を中止した事からその開発の継続を請け負ったLinuxディストリビューション製作プロジェクトである。私がこのOSを用いた主な理由は広島天文台シーイングモニターのソフトウェアに利用されたものであり、全天スカイモニター開発にも同様の利便性が期待できるからである。細かな理由としては、開発環境はLinux上が望ましく、Fedora core2のkernelバージョン2.6.6以上でのインターフェースの取扱いが容易であるためである。

2.4.2 API：GTK+ version2.4

GTK(GimpToolKit)はグラフィカルユーザーインターフェースを作成するためのライブラリである。GTKは元々General Image Manipulation Program(GIMP)の開発用に作られたもので、GNU Network Object Model Environment(GNOME)を含む多くのソフトウェアプロジェクトで使われているApplication programmers interface(API)である。

このGTKはLinux上でC言語で記述される。この言語を用いたのは後述のdvgrabとtranscodeがC言語上でから容易に機能させることができるからである。さらにGTKはGDK(GIMP Drawing Kit)上で実装されている。GDKは基礎となるウィンドウ関数へアクセスする低レベル関数を包むWrapperであり、画像の作成、画像処理などはGDKの関数であるGDKPixbufferを用いて行った。

2.4.3 画像取り込み：dvgrab version1.5

dvgrab version1.5はLinuxで機能するソフトウェアで、Fedora Core2のインストール時に選択することで実装され、DVキャプチャー器から送られるデジタル信号をIEEE1394を介しパソコンに取り込むソフトである。Unixのシェルからコマンドとして操作でき、C言語からのコントロールも容易である。

2.4.4 画像変換：transcode version0.6.4

transcodeはLinux用コマンド入力型高機能画像変換ソフトウェアである。全天スカイモニターではdvgrabで取り込んだムービーを1フレームの非圧縮静止画ファイル(ppm)に変化することを目的に導入した。このソフトは相性が難しく、正常に動作しないマシンも多いようであるが、本学シーイングモニターにて

・avi2yuv version 0.9.7-1
・gtk+-1.2(1.2.4以上のバージョン)
・gtk+-2.4

・glib+-1.2(1.2.4 以上のバージョン)
・glib+-2.4
・atk-1.2
・pango-1.5
http://www.gtk.org/
http://libdv.sourceforge.net/

を、kernel version を変更しインストールすることで正常に動くことが判明しているので、同様の操作を行った。また、transcode でデジタル信号を扱えるように以下のライブラリを拡張した。

・libdv version1.2
このライブラリは transcode をインストールする前に必ずインストールする必要がある。

2.5 試験稼働

装置が一式完成したところで試験稼働を本学高エネルギー宇宙素粒子実験研究室内において行った。図 2.15 はその試験画像である。フォーカス合わせが不十分であったためぼやけている。雲の状況を収めるために重要なのは広い視野と淡い光の領域を捉える高感度であるが、この試験ではゲインを低くおさえ、レンズのしぼりを小さくするなどして感度を低く設定して、視野のみを確認した。この試験により全天スカイモニターの光学系は 180 度を越える視野を持つことが確認できた。また 2 時間ソフトウェアを稼働し続けることができ、ソフトウェアが安定稼働することも確認ができた。

図 2.15: テスト撮影画像

2.5.1 データ処理の流れ

185 °の広角魚眼レンズにより CCD(撮像装置) に全天が映し出される。この映像はまずビデオカメラでアナログビデオ信号に出力される。このアナログビデオ信号は NTSC 方式である。NTSC はアメリカ、日本、台湾、韓国などで利用されているテレビのカラー映像方式であり、秒間 29.97 フレーム、走査線 525 本である。液晶モニター (WATEC:WAT ー LCD25) は、撮像をテストするときなど撮像状況の監視用に取り付けられている。その後、DV キャプチャーにより、デジタルビデオ信号に変換され、IEEE1394 インタフェースを介してパソコンに取り込まれる。

その後 dvgrab を用いてデジタル信号を約 0.1 秒間の映像ファイル (dv 形式) として取り込み、transcode を用いて個々の非圧縮静止画ファイル (ppm 形式) に変換する。作成された静止画ファイルは GTK 上で実装されている GDK(GIMP Drawing Kit) により window 上に表示される。

図 2.16: 実験測定の様子

図 2.16 は実際に実験測定時に使用する装置の写真である。現在は 12V バッテリーや 100V インバーター、ノート PC 等は湿気よけのプラスチックボックスに収まっている。天文台建設後は天文台より 100V 電源ケーブルをひき、アナログビデオ信号ケーブルは天文台内にひかれ、設置された DV キャプチャー、IEEE1394 インタフェースを介して天文台内の PC により制御される予定である。

第3章　測定

3.1　岡山天体物理観測所における予備測定

図 3.1: 1.88m 望遠鏡ドーム

岡山天体物理観測所に観測補助として行く機会を得たため、2005 年 11 月 8 日から 9 日にかけて岡山天体物理観測所において本機を用いて夜空の初観測を行い、同観測所に常設されているスカイモニター画像との比較を試みた。機器を 1.88m 望遠鏡ドーム前に設置し、時刻 23：30～ 翌朝 5：50 薄明開始まで 30 分おきに 10 回測定を行った。最適な観測状態を導く事も目的にしていたので、ゲインと積算時間と魚眼レンズのしぼりを切替え、フォーカス合わせを行いながら測定を行った。

図 3.2 に 11 月 9 日午前 3 時 7 分の広島大学全天スカイモニターで撮像した画像と、比較のため同時刻に岡山天体物理観測所スカイモニターが撮像した画像 (図 3.3) を示す。 図 3.2 において左側が暗い理由は、ドーム (図 3.1) が暗く写っているからである。右端の明りは街明りである。4 等級程度の明るさの恒星が写っている。岡山スカイモニターと比較すると星座の位置などから広大全天スカイモニターはより広視野に夜空を写していることが分かる。以上より、視野及び感度の両面で広大全天スカイモニターは岡山天体物理観測所既存のスカイモニターを凌ぐ十分な性能を持っていると言える。

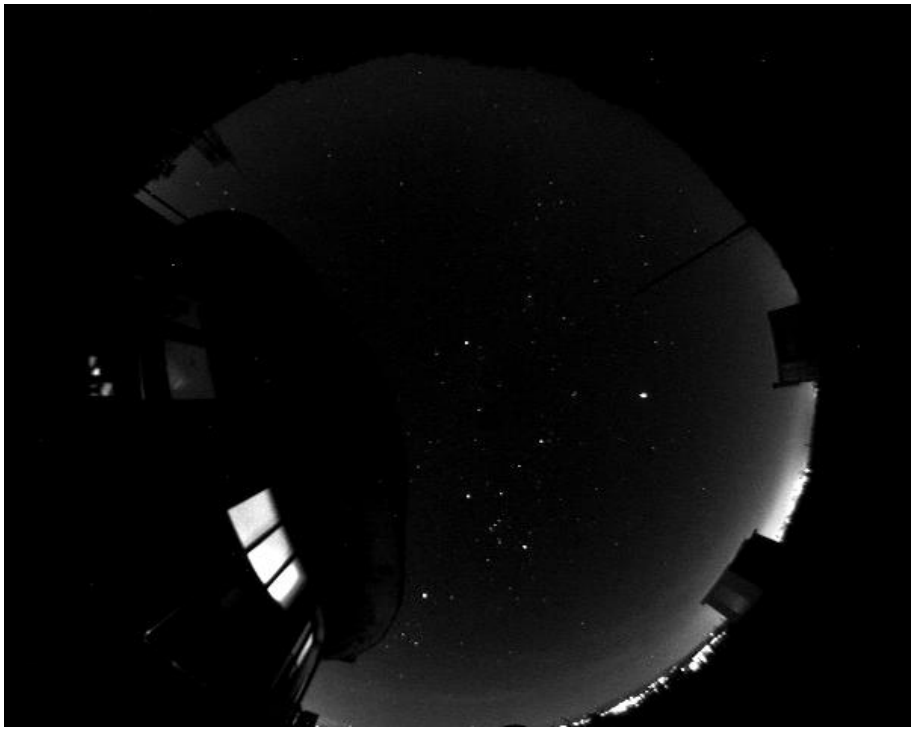


図 3.2: 夜空画像：広大全天スカイモニター

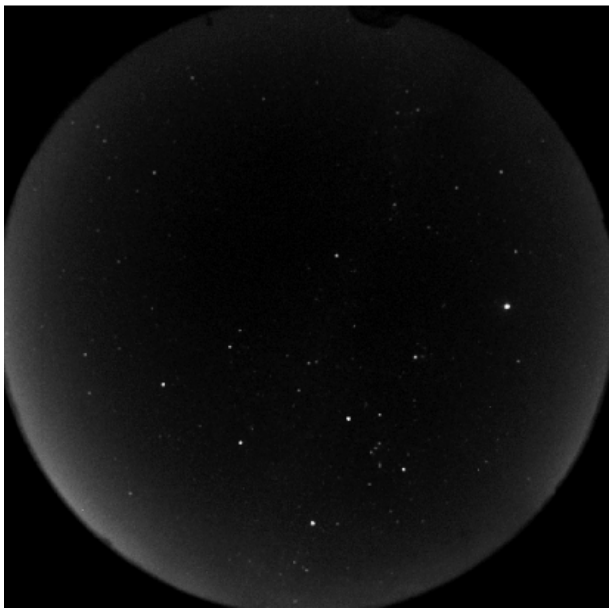


図 3.3: 夜空画像：岡山天文台スカイモニター

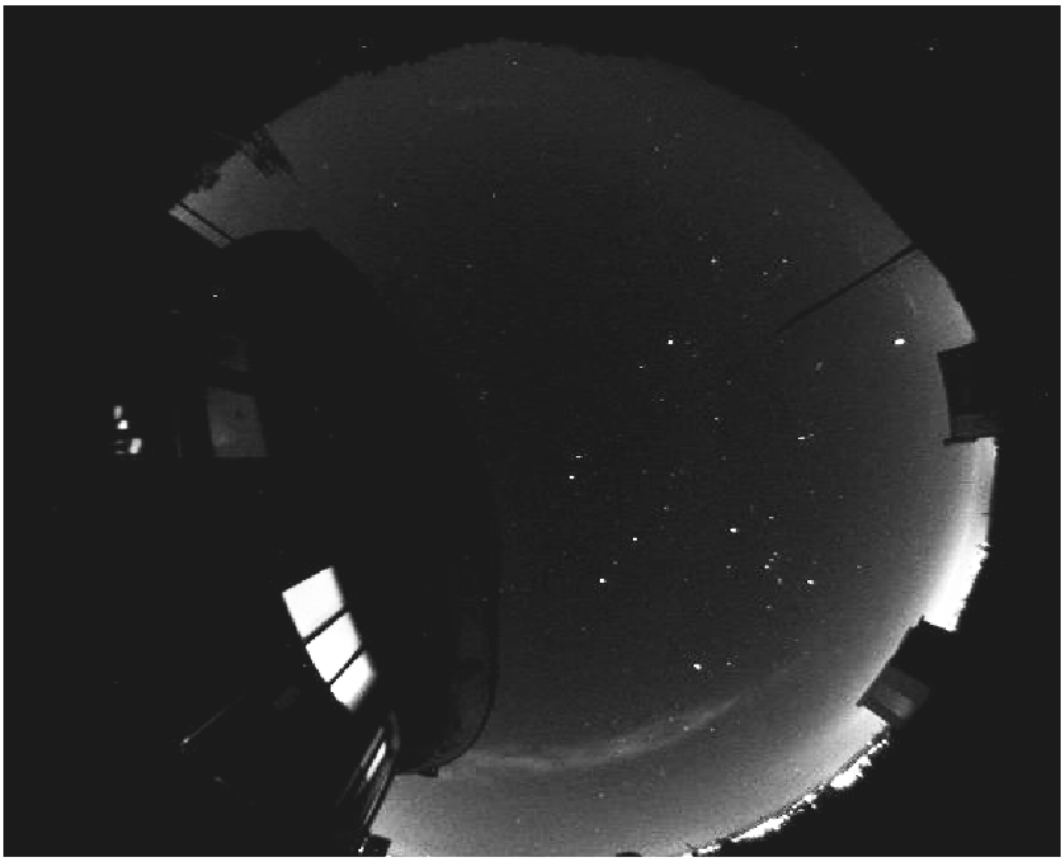


図 3.4: 夜空画像 (雲を含む)：広大全天スカイモニター

なお、この晩はかなりの晴天であったため雲が写っていたフレームはごく一部しかなかった。図 3.4 は雲を写しているものであるが、下部に淡くむらのある雲が写っている。雲検知機能では街明りを反射し輝く淡くゆらぎの多い雲を画像処理により特定し、雲領域のマッピングを行う。

3.2　東広島市天文台サイトにおける本測定

図 3.5: 広島大学天文台建設地

私は天文台建設地 (福成寺) において全天スカイモニターの装置の本測定を複数晩にわたり行った。本測定の目的は、レンズのしぼり、フォーカス合わせ、ゲイン、露光時間等の測定パラメータのセットアップを調整しながら行い、夜間に空の状態を撮影するために最適なパラメータを調査することと、空の雲の変動の様子を撮影し、加えて恒星をできるだけ写すことである。装置、ソフトウェア両面での安定稼働試験も兼ねている。ソフトの設定としてはは 30 秒に 1 枚撮像し、3 分に一度アーカイブに保存する。

セットアップを行う際、注意を払うことは像の鮮明さと明るさを同時に最大限得るこ

図 3.6: セットアップの様子

とと、CCD のサチュレーション (飽和) をさせないことである。受光量を増やし過ぎると CCD の受光限界を越えてしまい、サチュレーションして真の光量の測定が難しい状態になるので、これを避けるようセットアップを行わなければならない。表 3.1 には各測定パラメータの CCD カメラの撮像に対する影響や効果をまとめた。セットアップ時にこれらの調整を行った。

期間は 1 月初旬から 1 月下旬までである。表 3.1 にその詳細を載せる。

フォーカス (スペーサー)	像の鮮明さを変える
レンズのしぼり	F/1.4 ⇔ F/16 受光量： 増加 ⇔ 減少 フォーカス合わせの容易さ： 合いにくい ⇔ 合わせやすい
積算時間	0 sec ⇔ 8.5 sec 受光量： 減少 ⇔ 増加 時間分解能： 高い ⇔ 低い
ゲイン	LOW(5dB) ⇔ HIGH(38dB) 受光量： 減少 ⇔ 増加 ノイズ： 小 ⇔ 大

表 3.1: 測定パラメータリスト

3.2.1 1 月 12 日

観測日時	1 月 12 日 19：10~22：30
天候	曇天 (降雨無し)
月齢	12
フォーカス	スペーサー 2.3mm
レンズのしぼり	F/7
積算時間	8.5 sec
ゲイン	26dB(MID)

表 3.2: 1 月 12 日 観測環境

測定開始直後、うろこ状の雲に空が覆われており、強い風が吹いていた。一時晴れはしたが、測定終了時まで雲が出ていたので、雲の時間変動を長時間にわたり連続して測定することができた。図 3.7 はその時間変動の図である。雲は画像から目視で判別可能であり、表 3.2 の測定パラメータで雲を十分測定できる程度の撮影を行うことができることが分かった。ゲインは半分程度の設定であり、さらに雲のむらを強調するためにはゲインをより大きくして測定すれば良い。フォーカスが合っておらず、像がぼやけていた。

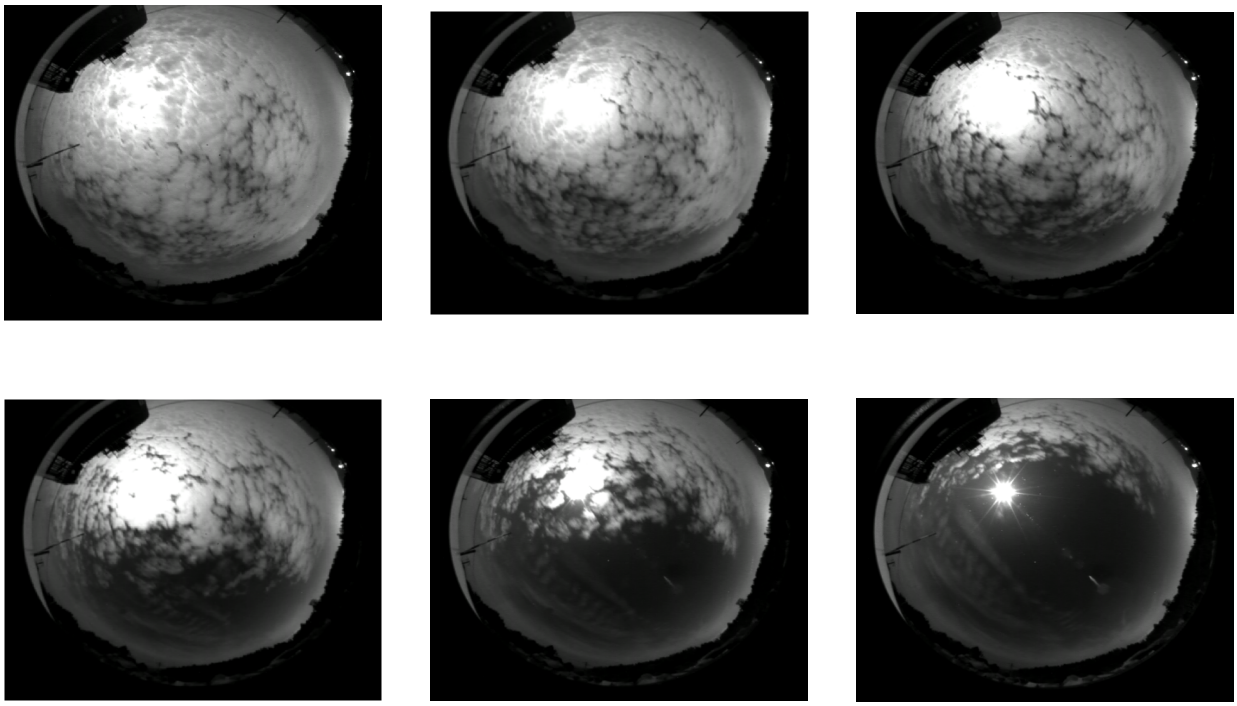


図 3.7: 11 月 12 日 20 時 12 分 ~20 時 27 分 天文台サイトにおける雲の時間変動

3.2.2　1 月 18 日

観測日時	1 月 18 日 18：18~30：36
天候	晴れ
月齢	18
フォーカス	スペーサー　2.2mm
レンズのしぼり	F/10
積算時間	8.5　sec
ゲイン	38dB(HIGH)

表 3.3: 1 月 18 日 観測環境

この日はよく晴れた。表 3.3 のパラメータ設定で、１等星程度の恒星まで写すことができた。また、初めて一晩中自動撮影をすることができた。2.2mm スペーサーを使用したところ、フォーカスが合い鮮明な像が得られた。以後これを使用する。

図 3.8 は 19 時 21 分から 1 時間ごとの画像を載せたものであるが、かなり暗い像になってしまっているのが分かる。これは主にレンズをしぼりすぎたことによる。明け方には空一面が明るい状態でかつ光量もちょうど良いデータが得られたのでバッドピクセルを特定することができた。バッドピクセルとは他のピクセルよりもノイズが大きく、CCD のピクセルの中で、受光の有る無しに関わらず常に明るい、感度が低い、もしくは動作をしないピクセルのことである。画像処理の際にこれを画像から補正する処理を行う。今回の測定の画像により得られたバッドピクセルテーブルを用いて、今後は自動的に画像を補正して見やすく誤判断がないようにソフトウェアを改良した。

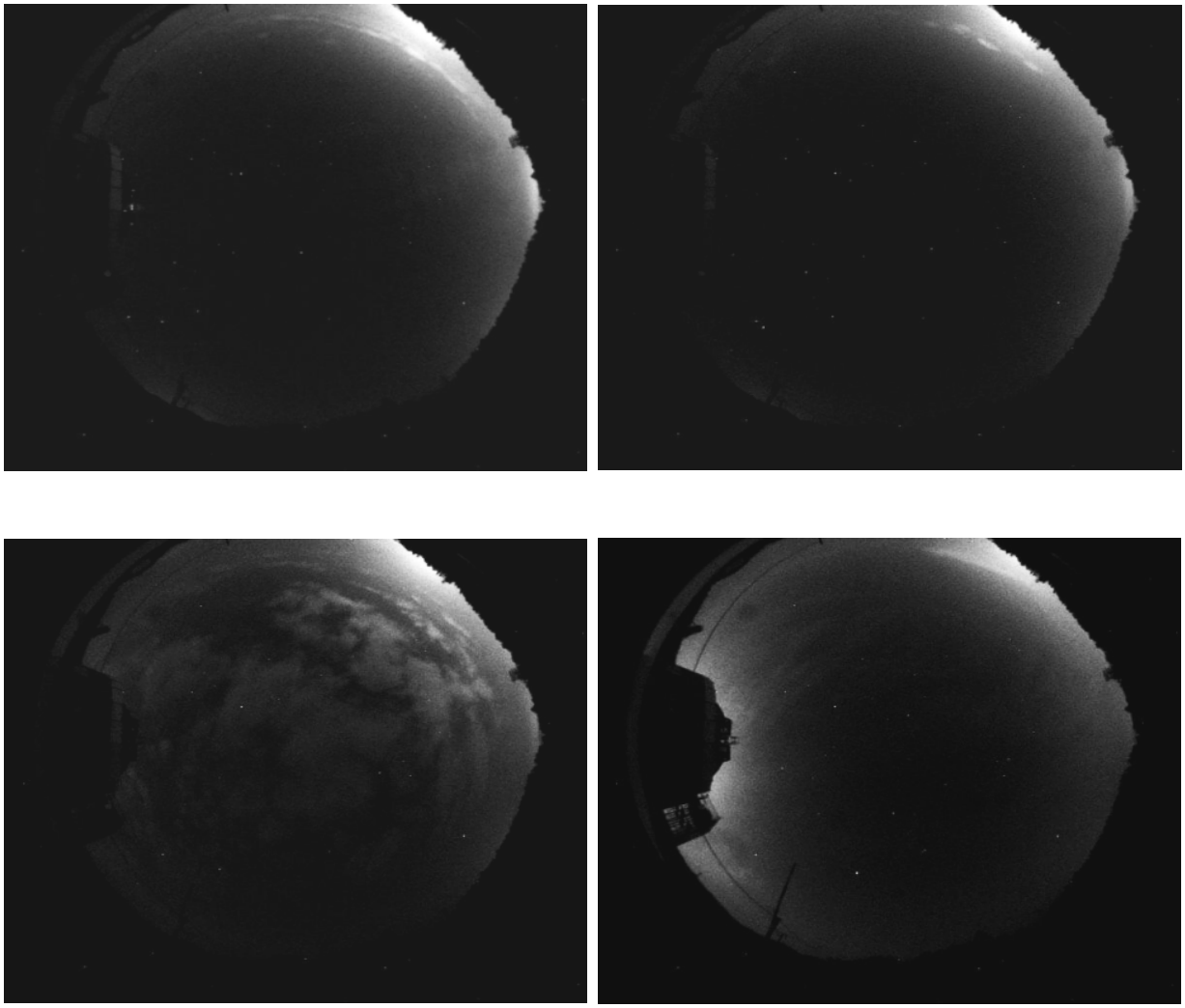


図 3.8: 1 月 18 日 19:21∼22:18 まで 1 時間毎の画像

3.2.3　1 月 24 日

観測日時	1 月 24 日 18：30∼8：00
天候	晴れ
月齢	24
フォーカス	スペーサー　2.2mm
レンズのしぼり	F/4
積算時間	8.5　sec
ゲイン	38dB(HIGH)

表 3.4: 1 月 24 日 観測環境

18 日の測定では 1 等程度の恒星が写っていたが、今回は 3 等程度の恒星まで写すことを目標にした。約 3 等程度の恒星を撮像するには約 6 倍の光量が必要になる。ゲインと積算時間は最大にしているのでしぼりを調整した。しぼりはレンズの受光面積を増減するもので、目盛を F/10 から F/4 にすると約 6 倍の光量が得られる。

図 3.9 は 21 時に撮影した画像であり、オリオン座、ふたご座等の星座も見て取れる。北

極星も写っているので天頂計算の際に役に立つ。また図 3.10 は 19 時から 1 時間ごとの画像を左から順に並べたものである。18 時台の画像は薄明が終了しておらず、CCD がサチュレーションを起こしている。翌朝 6 時過ぎから薄明が開始し、またサチュレーションを起こしていた。図 3.9 を見ると夜間は 3 等程度の星まで写っており、1 等程度の星はとても明るく写っている。大気中の微粒子などが反射する街明りも明るく写っており、スカイと呼ばれる背景光度を求めることも可能である。

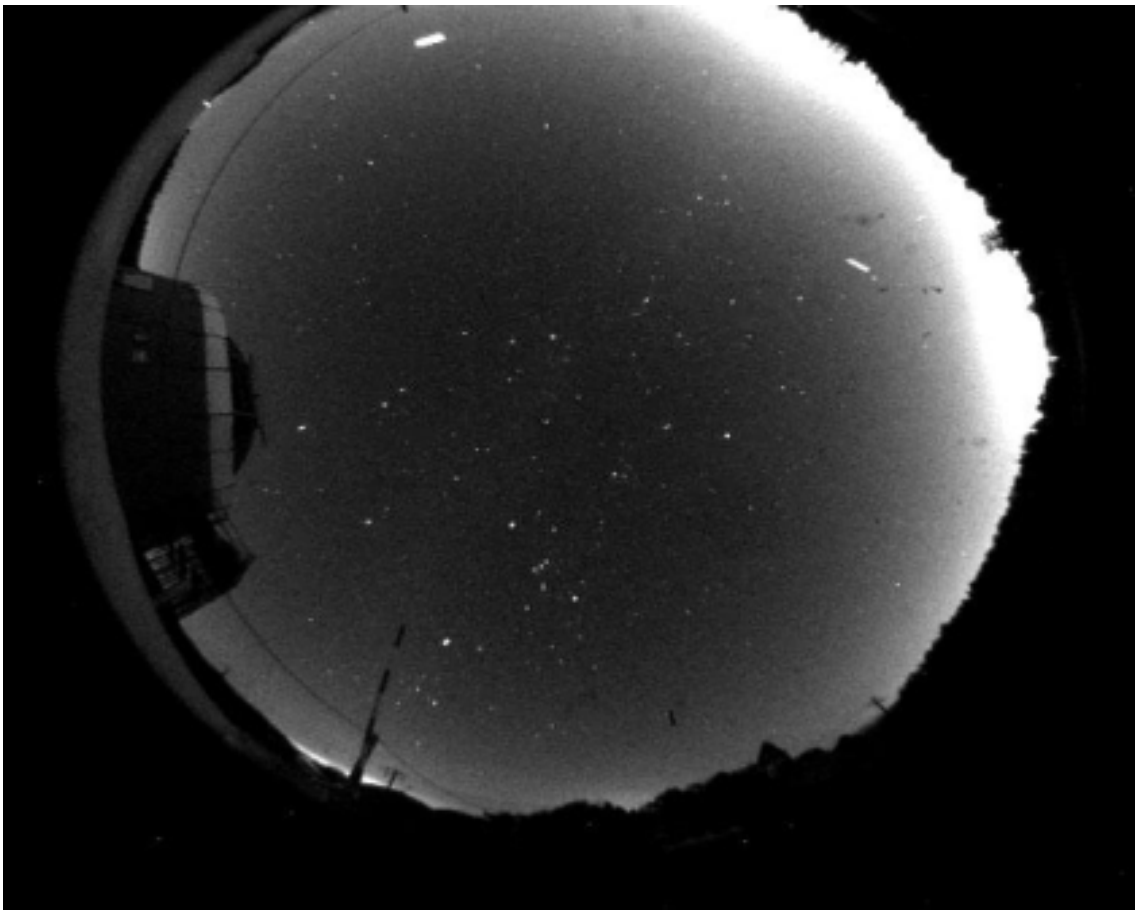
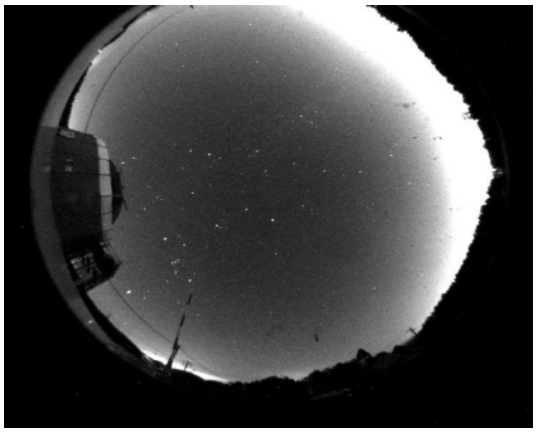


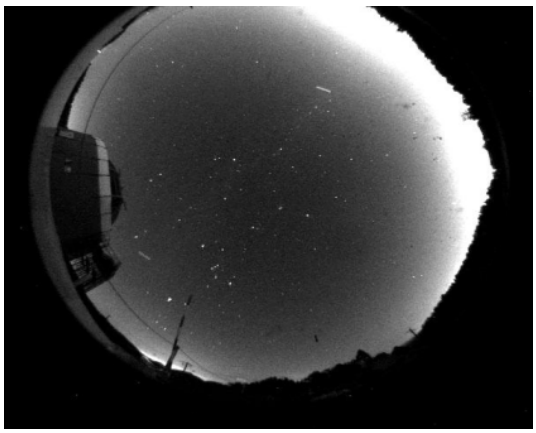
図 3.9: 1 月 24 日 21:00 撮影画像

3.2.4 まとめ

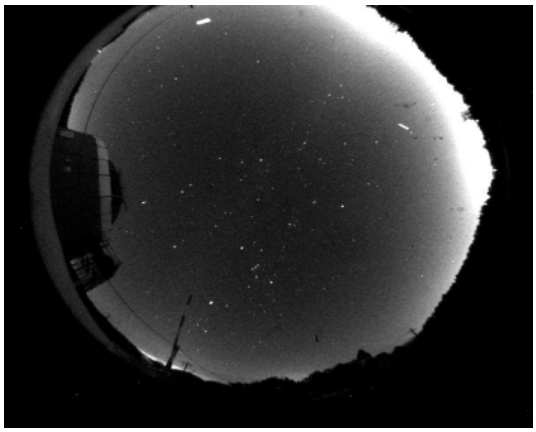
以上の測定からほぼ最適なセットアップ環境が決められ、安定した自動稼働が確認できた。画像処理に用いる雲の変動画像と 3 等程度の恒星を写した画像をアーカイブに保存できた。また、バッドピクセルの自動補正機能も加えることができた。天文台サイトにおける本測定において、雲の変動の様子と 3 等程度の恒星を鮮明に撮影することに成功したことにより、装置が十分な性能をもつと評価することができる。



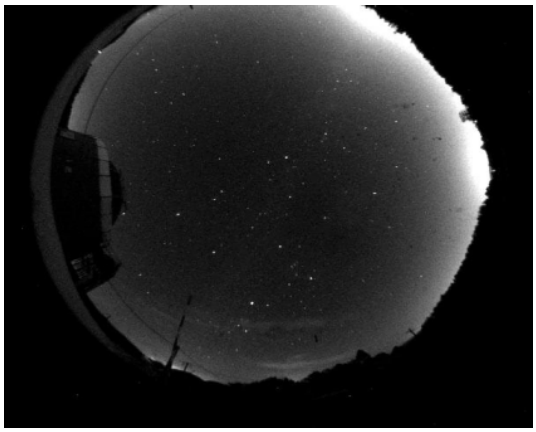
19 時



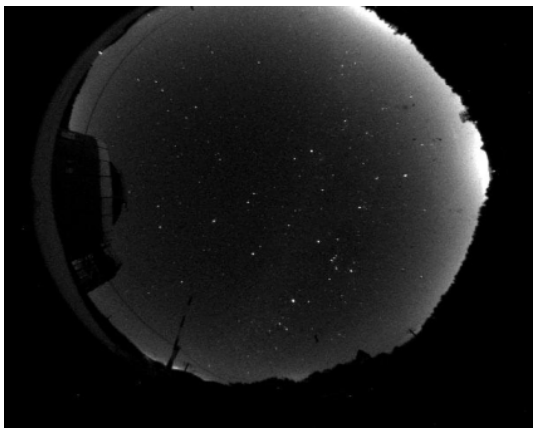
20 時



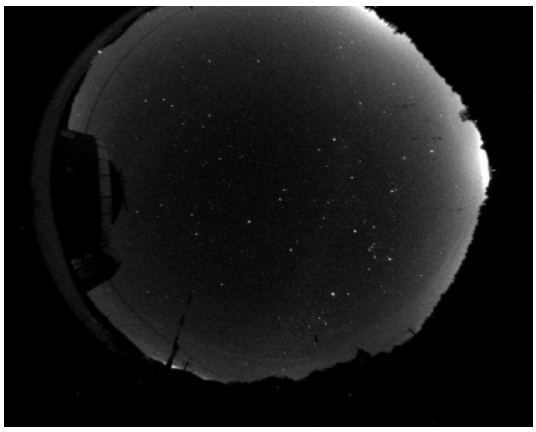
21 時



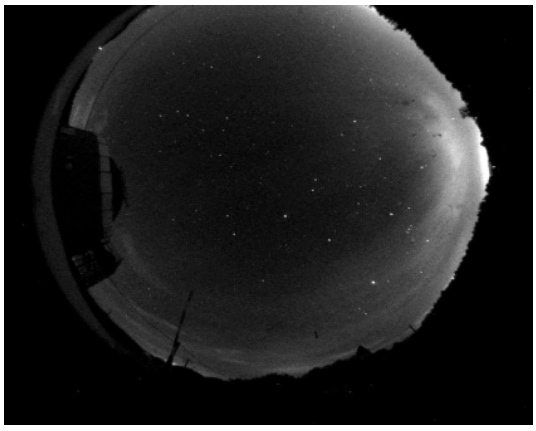
22 時



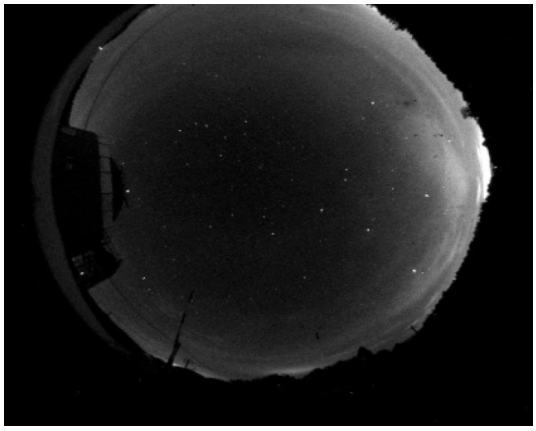
23 時



24 時



25 時



26 時

図 3.10: 1 月 24 日 1 時間おきの夜空画像

第4章　自動雲検知アルゴリズムの開発と観測スケジュール機能

4.1　観測天体位置との照合

観測天体位置の雲の有無を照合するために、観測天体の画像上でのマッピングを行う必要がある。検出部は天球面を2次元円として撮像しているため、球面三角法による2次元座標変換を行う。それを雲マップと同じものに天体位置をマッピングし照合する。以下ではその計算の方法について述べる。なお計算の都合上紙面で表す角度は全てラジアン単位とするが、表示する際には一般的によく用いられる度数表示や時間表示に変換する。

4.1.1　MJD(Modified Julian Day)

Julian Day(ユリウス日) とは紀元前4713年1月1日の正午からの通し日数であり、2000年1月1日正午は2,451,545日と数えられる。これは世界時 (UT) を使用している。この通し日数の特徴として正午を起点にしているため、同日の24時は0.5日足した2451545.5日となる。この Julian Day を用いる理由は、通常の年月日表示では、十進法で時刻が連続しておらず都合が悪いためである。

MJD(Modified Julian Day) とは Julian Day は桁数の多く計算上の扱いが困難であるため、MJD = JD(Julian Day) - 2400000.5 としたものであり、別の呼び方で準ユリウス日とも呼ばれるものである。式4.1はその計算式で、西暦 y 年 m 月 d 日 hour 時 min 分 sec 秒の MJD の計算をする。

1月、2月は前年を基準に計算するため、m = m + 12, y = y - 1 とする。

$$\begin{aligned}\text{MJD [日]} = 365.25 \times \mathrm{y} + \mathrm{y}/400.0 + \mathrm{y}/100.0 + 30.59 \times (\mathrm{m} - 2.0)] + \mathrm{d} - 678912 \\ +\text{hour}/24 + \text{min}/1440.0 + \text{sec}/86400.0 \quad (4.1)\end{aligned}$$

最後の3項は時、分、秒の日換算であり1時間は1÷も、24日、1分は1/(24×60)日、1秒は1/(24×3600)日となる。これにより MJD を実数表記できるようになる。

4.1.2 赤道座標系

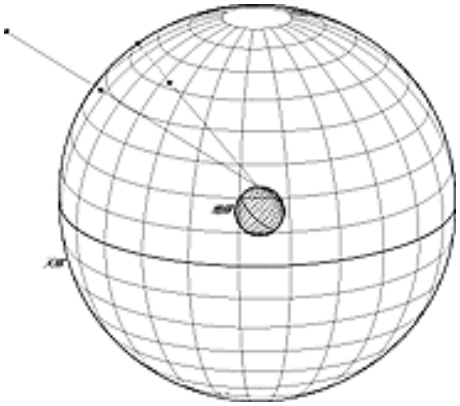


図 4.1: 天球

我々が夜空で見る星の地球からの距離は様々だが、距離が遠いため図 4.1 のような 2 次元天球面上に点在しているとみなせる。赤道座標系は地球上で緯度・経度を用いて地球上での座標を記述するように天球上で星の座標を記述する座標系の一つである。この座標系は地球の自転軸を延長した天球との交点を天の北極、南極とし経線・緯線を描く。赤道座標系は赤経 (Right Ascension)・赤緯 (Declination) を用いて記述される (図 4.2)。赤経は春分点を基点にし東回り 360° である。赤緯は赤道面を天球に延長した天の赤道を基点に北側に 0° ∼90°、南側に-0° ∼-90° である。

地球の歳差運動により天体の赤道座標はずれていくため、いつの時点の星の座標であるかという原点の定義が必要になる。これを分点という。私は J2000(2000 年分点) を用いた。

4.1.3 Local Sidereal Time(LST)

Local Sidereal Time(地方恒星時) を説明するためにグリニッジ恒星時について述べる。グリニッジ恒星時とは経度 0 °において南中している天体の赤経である。地方恒星時は観測地点での南中している天体の赤経である。よってグリニッジ恒星時に観測地の緯度による補正値 (東経+、西経-) を加えると地方恒星時を求めることができる。地方恒星時の計算では、θ を式 4.2 のように定義し、その少数部分のみ式 4.3 に代入して求める。

$$\theta \;=\; 0.671262 + 1.002737909 \times (\mathrm{MJD} - 40000.0) + \mathrm{lon}/2\pi \tag{4.2}$$

MJD : Modified Julian Day
lon : 観測地の緯度 (単位：rad)

$$\mathrm{LST}\ [\mathrm{hh:mm:ss}] \;=\; 24 \times \theta \tag{4.3}$$

地方恒星時は観測可能な天体を選定する重要な指標である。

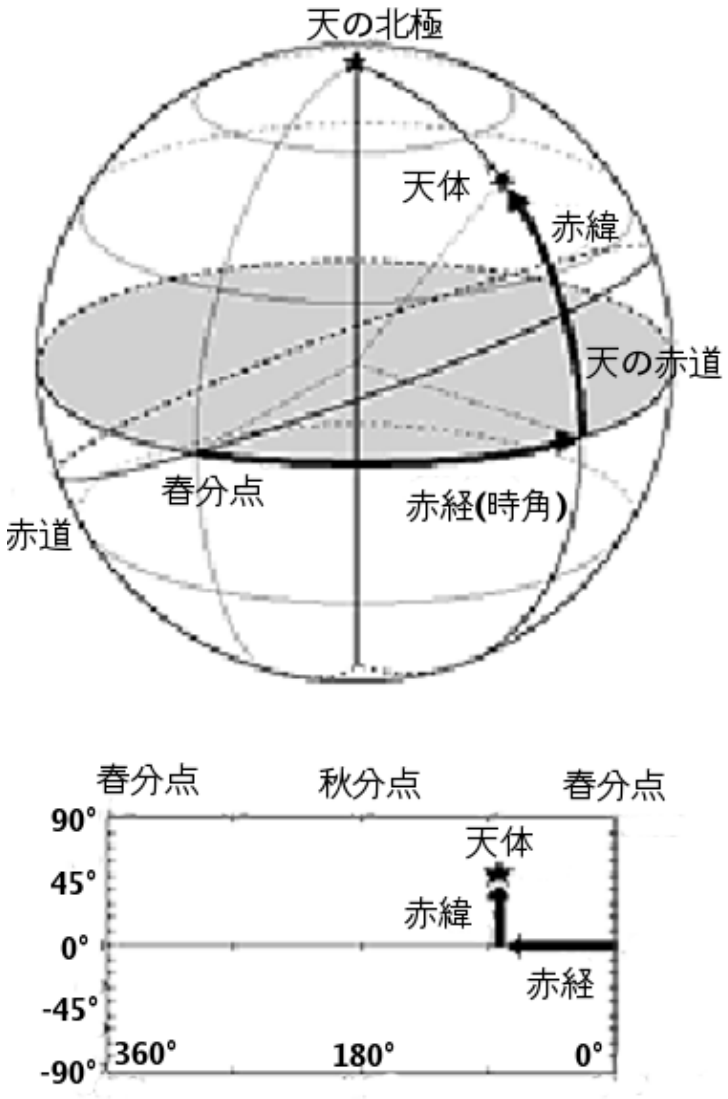


図 4.2: 赤道座標

4.1.4 Horizontal coordinate system(地平座標系)

3 章の図 3.2 のような画像上で星の位置を特定するために、地平座標系を導入する。地平座標系は観測地点を中心とした地平面を基点とした方位角 (Azimuth) と高度 (Altitude) で座標を表示する (図 4.3)。以下は任意の天体の赤経・赤緯を用いて天体の高度、方位角を求める計算である。

Hour Angle(時角)

第一に求めるのは時角である。時角とは天の北極-真南-天の南極を通る大円からの西回りに 0~24 時で表される角度のことであり、任意の時刻での南中している天体からの観測天体への角度差を表す。その計算を式 4.4 に示す。

$$\mathrm{ha}\ [\mathrm{hh} : \mathrm{mm} : \mathrm{ss}] = \mathrm{LST} - \mathrm{RA} \tag{4.4}$$

ha ： 時角
LST ： 地方恒星時
RA ： 観測天体の赤経

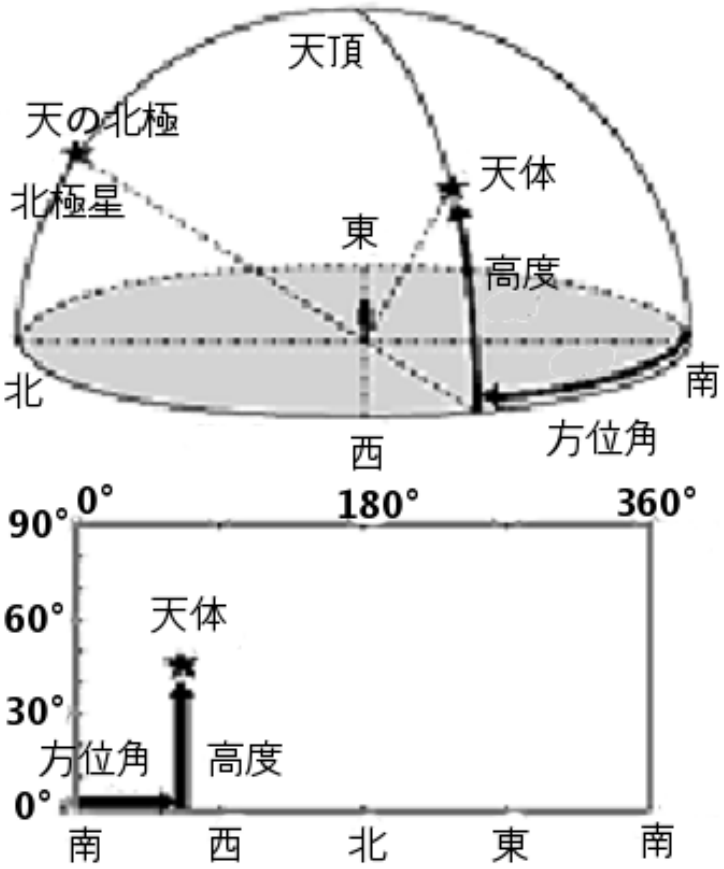


図 4.3: 地平座標

Altitude(高度)

次に高度を求める。高度は地平線を基点として観測地上空の天頂方向へ 0° ～ 90°、逆方向へ -0° ～ -90° である。計算式は式 4.5 で与えられる。実際画像で天体位置を計算する際には、天頂からの距離 (zenith distance) の方が便利である (式 4.6)。

$$\mathrm{Alt[d:mm:ss]} = \arcsin(\sin(\mathrm{Decl})\sin(\mathrm{lat}) + \cos(\mathrm{Decl})\cos(\mathrm{ha})\cos(\mathrm{lat})) \tag{4.5}$$

Alt ： 高度, Decl ： 観測天体の赤緯
lat ： 観測地の緯度 , ha ： 観測天体の時角

$$\mathrm{Z[d:mm:ss]} = \pi/2 - \mathrm{Alt} \tag{4.6}$$

Z ： 天頂距離 (Zenith Distance)
Alt：天体の高度

Azimath(方位角)

最後に方位角を計算する。方位角は南を 0 °とし、西回りに 0° ～360° で表す。

$$\mathrm{Azimuth[d:mm:ss]} = \arcsin(\sin(\mathrm{ha})\cos(\mathrm{Decl})/\cos(\mathrm{Alt})) \tag{4.7}$$

Azimuth ： 方位角 、ha ： 観測天体の時角
Decl ： 観測天体の赤緯、Alt ： 観測天体の高度

4.1.5 画像の天体位置

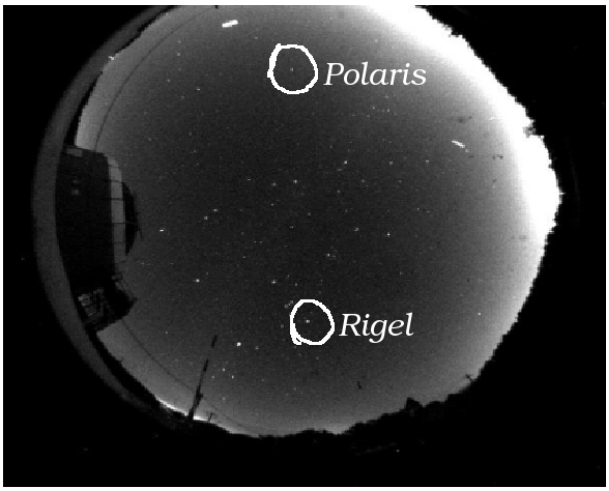


図 4.4: 北極星とリゲルの画像上位置

図 4.6 は実際の撮像されたデータと画像の対応関係であるが、天球上の天体は魚眼レンズで取り込まれ、画像上で 2 次元平面表示される。実際の画像上の天体位置を特定するために (高度, 方位角) ⇔ (画像上の X,Y) という二次元平面への変換を行う必要がある。北極星 (Polaris) はほぼ常に高度は観測地の緯度程度、方位角は 180 °の位置に位置している。まず北極星と任意の時刻に南中 (方位角 0 °) に位置している天体の座標を特定し、それぞれの天頂距離の比から画像上の天頂 (Zenith) を推定する。画像上の座標は図 4.5 のように画像の左上の点が原点である。水平右向きに x 軸正方向、垂直下向きに y 軸正方向である。

私はこの計算に 1 月 24 日 21:09:48 の画像 (図 4.4) 上の北極星と南中したオリオン座に位置するリゲル (Rigel) を用いた。後に述べるが、実際の天頂距離と画像上での距離はリニアな相関をもっており、実際の天頂距離を先に述べた方法を用いて計算しておけば、その比を画像上の距離と比較することで天頂距離を同定できる。

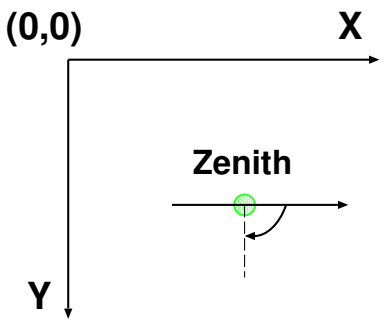


図 4.5: 画像上の座標系

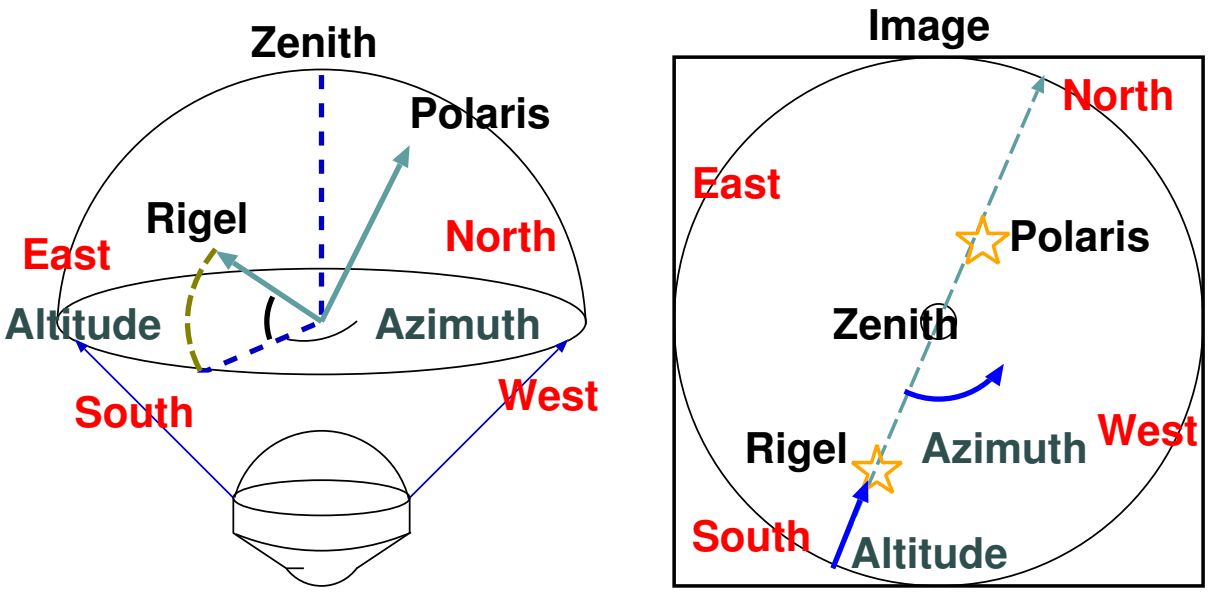


図 4.6: 撮像する天球と画像との対応図

次に同日の画像に写り込んでいる 1 等程度の星の画像での座標を西、南、東それぞれ 6 点程度サンプリングし、ファイルにまとめた。画像上の天頂距離と方位角をサンプリングした天体座標と天頂の座標からそれぞれの天体について計算した。天頂距離は画像上の天頂からのピクセル単位の距離、方位角は天頂から x が正の方向を 0 °とし、0° ~360° とした。私は任意の時刻の画像から計算した天頂距離と同時刻の天体の赤経、赤緯から計算した天頂距離を gnuplot[1] を用いてプロットした。図 4.7 においては x 軸は画像の天頂距離であり、y 軸は赤経、赤緯から導かれた天頂距離をとった。フィッティングは 1 次関数で充分であった。また方位角についても図 4.8 のようにフィッティングを 1 次関数で行った。x 軸が画像での方位角、y 軸が赤経、赤緯から導いた方位角である。これにより図 4.6 のような対応関係のある実際の方位角と高度に対する画像上での天体位置との変換パラメータを求めることができた。

3 章で 3 晩にわたり観測を行ったことを述べたが、それぞれセットアップを変えている。作成したソフトでは、観測日ごとにそれぞれパラメータのファイルを作成し、プログラムに読み込ませるようにしている。

[1]gnuplot は 2 次元や 3 次元のグラフを描くツールであり、テキストファイルにあるパラメータデータをプロットし、最小二乗法でそのプロットに対し関数で近似曲線を描くことができる。これをフィッティングという。

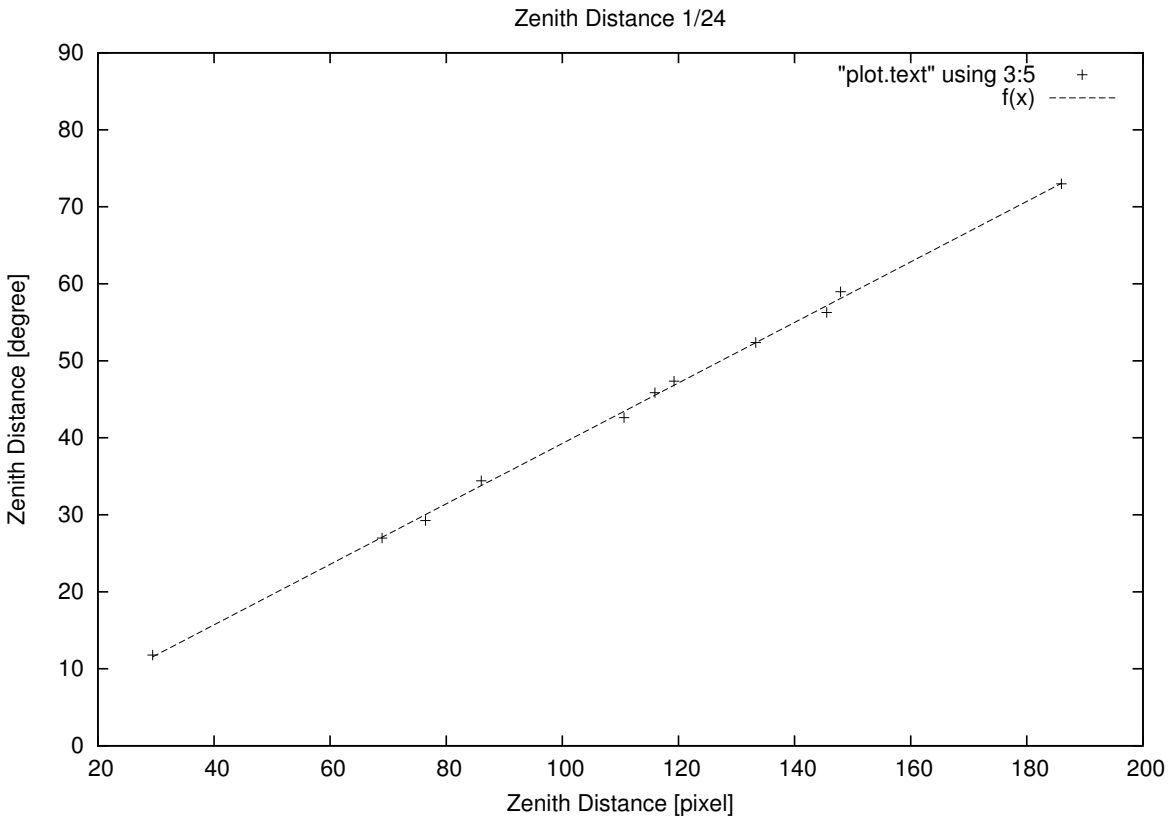


図 4.7: 天頂距離の対応関係

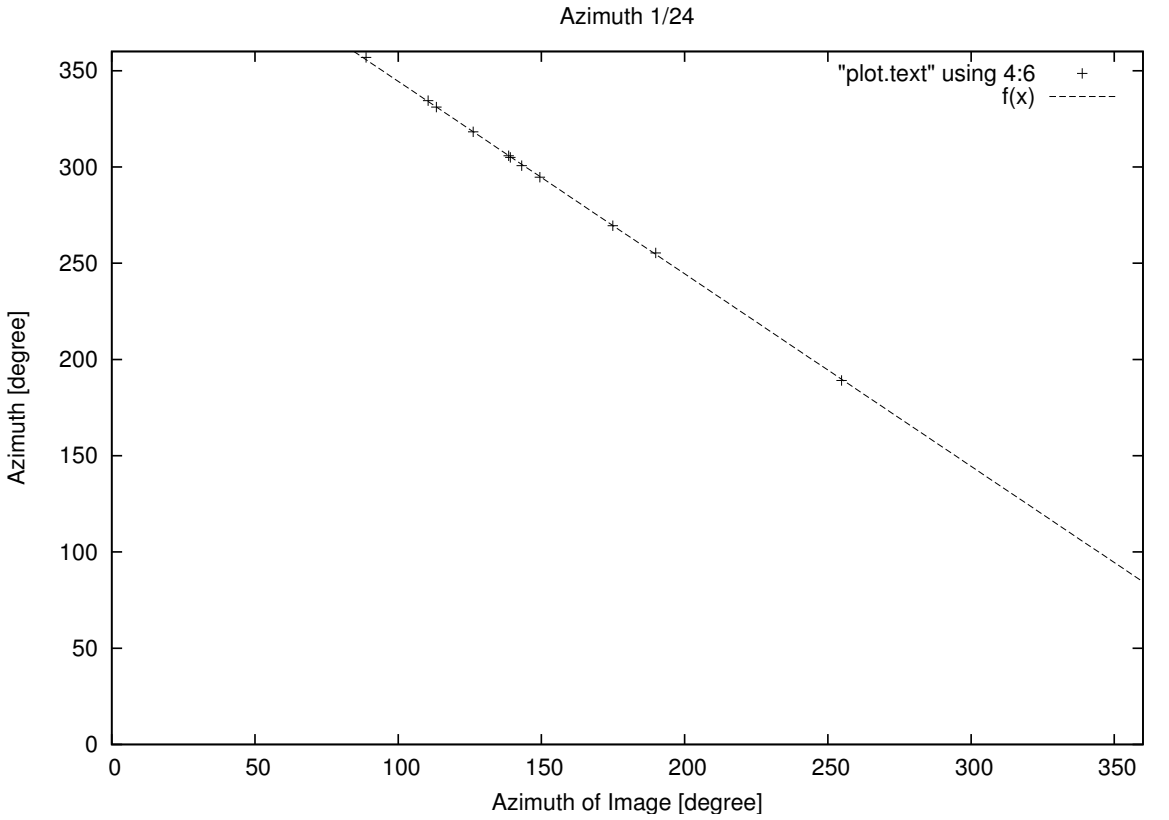


図 4.8: 方位角の対応関係

図 4.9: 位置計算ソフトメインウィンドウ

図 4.9 が作成した天体位置計算ソフトのメインウィンドウである。画像上に目視で天体位置が把握できるように、天頂から高度 60° 線、30° 線、0° 線の円を描き、方位角 0° 方向から 45° ずつ直線を引き、正距方位図法的なマップを描画するようにしている。天体リストには１等星程度の有名な星のものを用意した。それをプログラムに読みこみ、計算した天体位置を画像上に描画するようになっている。画像上での天体位置が計算できるようになったことで、自動雲検知機能による雲判定結果との比較が可能になった。このマップは様々用途に合わせ、実画像モード、マップ表示モード、画像差分モードを有している。また画面上をクリックするとその点の座標、ピクセルカウント、高度、方位角、赤経、赤緯を画像左下及び右下に表示する。

4.2 全天の領域分け

雲検知アルゴリズムを用いて雲領域のマッピングを行うために、画像を区分する。まず通常望遠鏡で観測する高度が 30 °以上であるため、高度 30° 以上の範囲を 15×15(ピクセル単位) の正方形で区分けする。その処理の単純化のために正方形領域を高度 30 °以上の領域が全て収まる範囲でとり、その四方の領域は除く処理をする。図 4.10 はその概略図である。

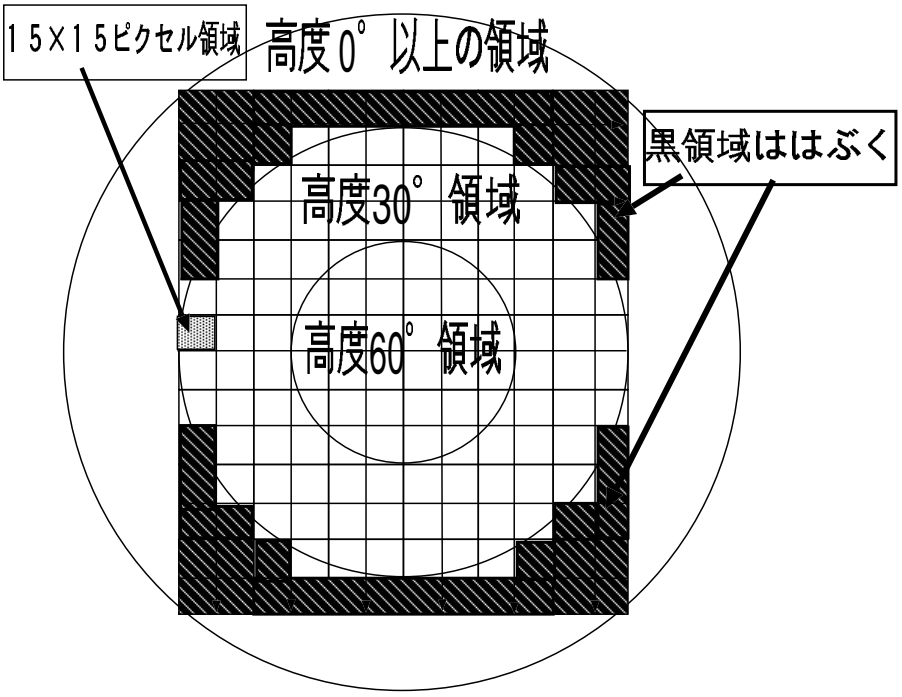


図 4.10: 画像領域区分図

また画像上で区分した領域を描画し、目視で各領域の雲の有無などの状況を判別できるようにした (図 4.11)。雲と判定する条件を設定することで条件に当てはまる領域だけ描画することも可能である。

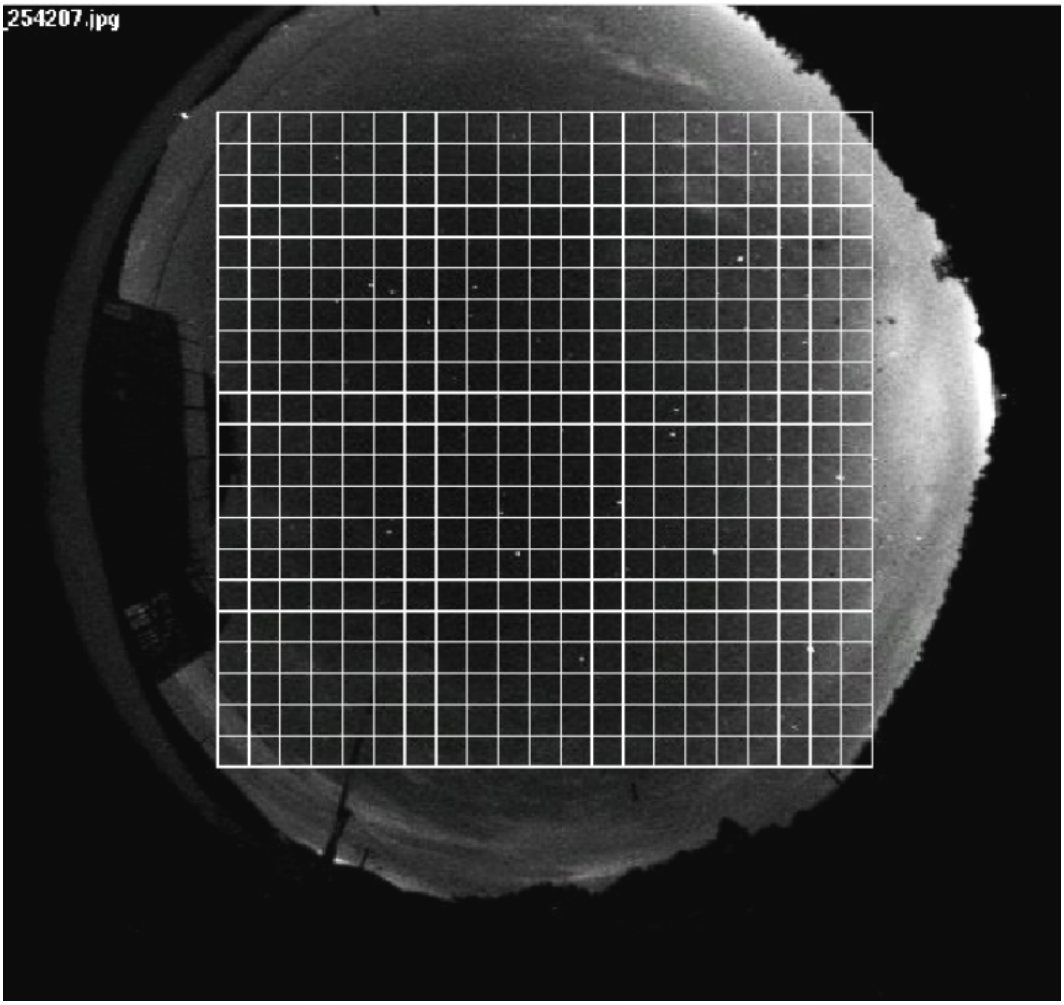


図 4.11: 実画像への領域区分

4.3 自動雲検知アルゴリズムの開発

雲は一様ではなく流動的であり、大きさも一様ではなく分離、合体を繰り返している。そのため雲を一個体として判別するのは困難である。私が考案した自動雲検知アルゴリズムは、領域区分した全天画像の各領域のピクセルカウントを用いて、領域ごとに雲判定パラメータを設定し、パラメータ全てに所定の値をとった領域を雲と判定し、雲があると判定された領域のみ画像上に描画することを基本としている。この方法で雲領域をマッピングする。

まず、区分した領域のピクセルカウントを用いて諸パラメータを作り、それを計算する機能を追加した。そして 1 月 12 日と 1 月 24 日の測定で得られた画像に対してパラメータを求めた。

デジタル信号から dvgrab と transcode により非圧縮静止画ファイルとなったデータのピクセルカウントの扱いについて述べる。私は画像を表示するために GDK に属する GDKPixbuffer を用いた。GDKPixbuffer は画像を扱う API であるが、その画像描画システムは取得したデータの光量ピクセルカウント GDK-DRAWING-AREA という window 上の描画領域に 1：1 のピクセル比でデータのピクセルカウントを格納していく。格納される

カウントの最大値は 255 であり、最小値は 0 である。

雲は街明りを反射したり、背後の月の光を散乱し輝くのでその厚さに応じて一定の光量カウントが得られる。このカウントがある一定値以上であることを雲の条件とする。その中でもピクセルごとの揺らぎが大きいものを雲と判定する。
私はまず区分した領域でのピクセルごとのカウントの平均値 ($\overline{X}$)、標準偏差 (σ)、中央値 (X_m) を計算する機能を追加した。式 4.8、4.9 はそれぞれの定義式である。

・平均値 $\overline{X}$

$$\overline{\mathrm{X}} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\mathrm{C_i(x,y)}}{\mathrm{N}} \tag{4.8}$$

・標準偏差 σ

$$\sigma^2 = \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{(\mathrm{C_i(x,y)} - \overline{\mathrm{X}})^2}{\mathrm{N}} \tag{4.9}$$

ここでの N は足し合わせたピクセル数、$C_i(x,y)$ は任意の座標でのピクセルのカウント数である。
X_m は領域内のピクセルカウントを最低値から最高値まで並べたその中央の値である。

ピクセルカウントの中にはバッドピクセルや、宇宙線入射、飛行機、人工衛星等により最大値に近い高い値を取るものがある。15×15 の領域において平均値はそれら偶発的な事象に大きく依存する場合がある。X_m はそれらの影響はないので、$\overline{X}$ ではなく X_m をその領域のカウント平均とみなす。また同様にそれらの事象の影響を除いた σ を求めるために、並び替えたピクセルカウントのうち、最大、最小から各 10 個を除いた X_m からの標準偏差 (σ_m) を計算する機能を加えた。
また雲は非一様であり時間的な各領域のピクセルカウントの揺らぎも大きい。そこでピクセルカウントを 30 秒前に撮像したデータのカウントから差し引きをする差分画像を用いて、差分した各領域のピクセルカウントの中央値 ($X_{m,d}$) と、並び替えたピクセルカウントの最大、最小から各 10 個ずつを除いた $X_{m,d}$ からの標準偏差 ($\sigma_{m,d}$) を計算する機能を加えた。差分画像の時間差に 30 秒を採用した理由は、長い時間差の差分では後述の月の影響が大きくなり、雲判定が非常に難しくなったことと、これより短い時間差では雲の変化が少なかったことである。

ここで差分画像の具体的な例を載せておく。図 4.12 と 4.13 は 1 月 12 日に撮影した実画像と差分画像を並べたものである。画像を 30 秒前にアーカイブに保存された画像とで差分したものである。図 4.12 は雲が厚くむらの多い雲であったために時間的ゆらぎが大きく、雲領域はカウント高とカウント低の領域がランダムに広がっている。また図 4.13 は薄い雲が一部にのみ掛かっていたときであり、雲領域部分のみ差分でカウント高になっている事が分かる。また月が写り込んでいるが、月などの天体は光量がほぼ一定であるため、差分画像ではもっともピクセルカウントが低い。
差分画像全体が明るくなっている理由は、2 枚の画像の差分後、ピクセルカウントが負の値を取らないように差分後の各ピクセルカウントに 255 を足し 2 で割る処理をしている

ためである。この処理を行っているため、差分後のピクセルカウントは 127.0 が原点 (差分がゼロ) とする。つまり 127.0 以上ならばカウント増加、それ以下ならカウント減少である。

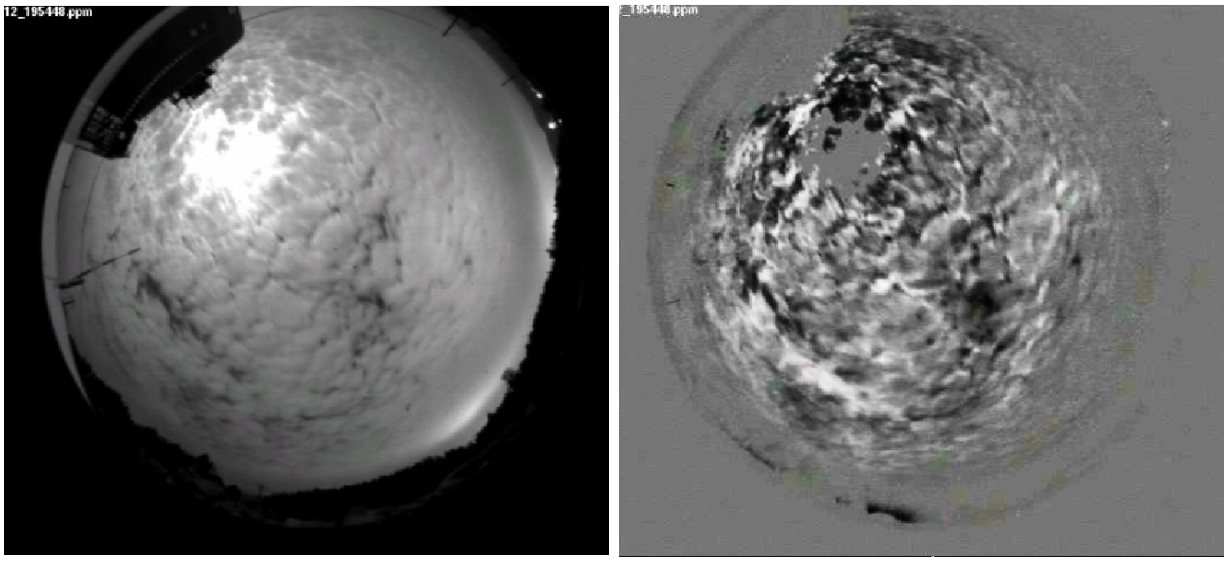


図 4.12: 実画像と差分画像の比較 (厚い雲)

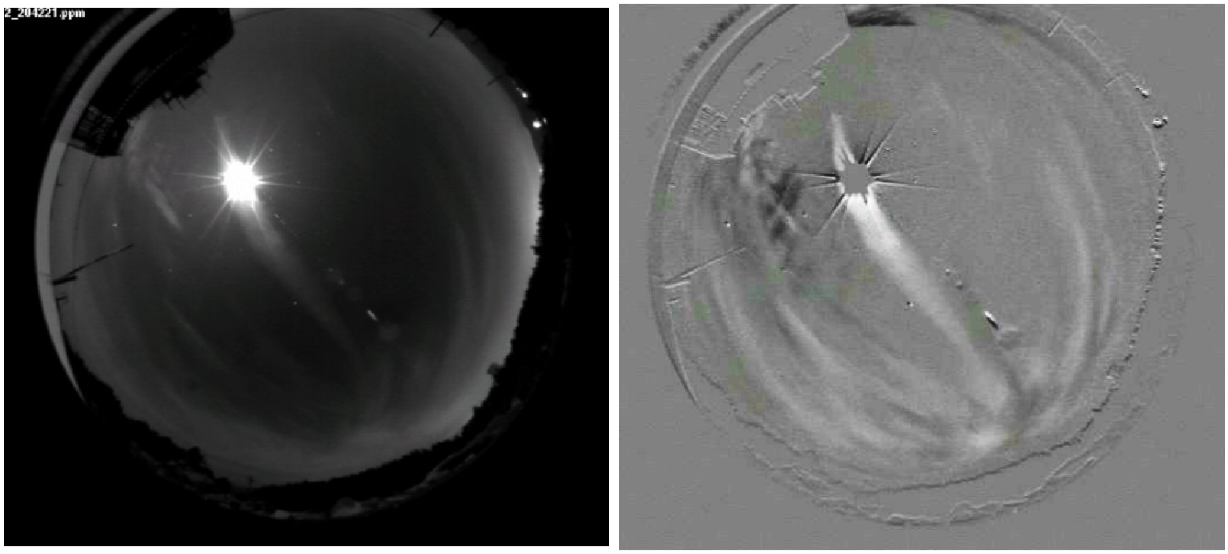


図 4.13: 実画像と差分画像の比較 (薄い雲)

私は全天画像の区分した領域に対して各領域の X_m、σ_m、$X_{m,d}$、$\sigma_{m,d}$ を計算した後、表 4.1 の領域を画像を目視で判別し、その領域のピクセルカウントの実測値を計算して整理することで、雲判定のためのパラメータを導入してみた。画像は月があり様々な形態の雲が測定できた 1 月 12 日と、月が無く 3 等程度の恒星を測定できた 1 月 24 日のデータを用いて、表 4.1 のような場合分けをして判定を試みた。

	雲の濃さ	月の有無
case 1	厚い雲	あり
case 2	薄い雲	あり
case 4	雲と空の境界	あり
case 3	雲無し	あり
case 5	薄い雲	無し
case 6	雲無し	無し

表 4.1: 空の様子の場合分け

	薄い雲	雲無し
X_m	50.0~65.0	9.0~50.0
σ_m	5.6 程度 (2.75~7.24)	4.0 程度 (2.36~5.21)
$X_{m,d}$	127.2 程度	127.4 程度
$\sigma_{m,d}$	1.49~4.34	1.88~4.14

表 4.2: 月がないときの雲パラメータ の実測値

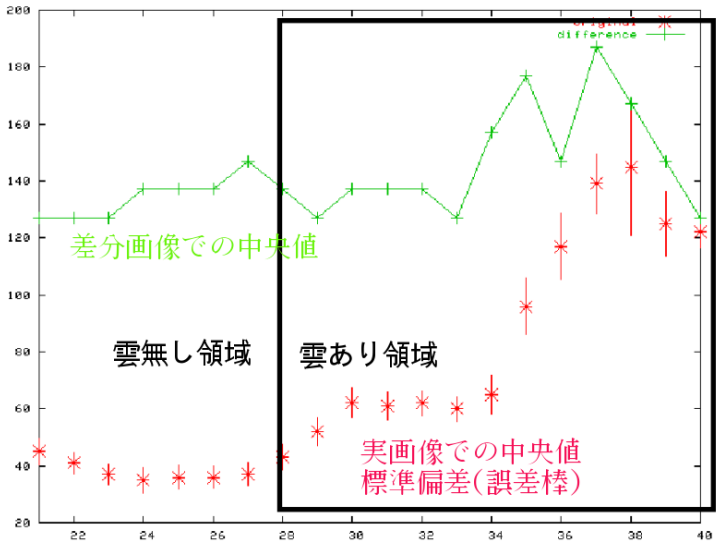


図 4.14: 月が無い時の区分領域一列のプロット

月の無い時の各領域の統計量は表 4.2 となった。また図 4.14 は月が無いときの画像の領域区分した一列の領域の各種統計量の値をプロットしたものである。赤色線は X_m、その誤差棒として σ_m、また緑線は $X_{m,d}$ である。正方形で囲んでいる領域が薄い雲がある領域であり、横軸は左端から番号をふった領域の番号、縦軸は中央値の値である。右端の領域で X_m の値が高いのは、街明りが強い領域であることが理由である。表 4.2 では X_m 以外の統計量はあまり差異がなく見えるが、各領域ごとの値のばらつき方が異なる。雲が無い領域ではほぼ一定値をとるが、薄い雲のある領域では値はばらついていた。

	厚い雲	薄い雲	空と雲の境界	雲無し
X_m(月周辺)	147.0~254.0	155.0~254.0		147.0~254.0
(他の領域)	58.0~147.0	45.0~147.0	53.0~180.0	1.5~45.0
σ_m(月周辺)	0.56~33.25	0.56~35.96		0.0~65.0
(他の領域)	1.74~17.09	1.26~24.69	4.00~38.47	0.56~3.72
$X_{m,d}$(月周辺)	111.0~156.0	78.0~168.0		119.0~133.0
(他の領域)	112.0~137.0	116.0~131.0	93.0~207.0	127.0
$\sigma_{m,d}$(月周辺)	0.56~25.89	3.00~33.82		0.00~10.33
(他の領域)	0.92~10.0	3.99~24.65	2.70~18.70	0.55~1.26

表 4.3: 月があるときの雲パラメータの実測値

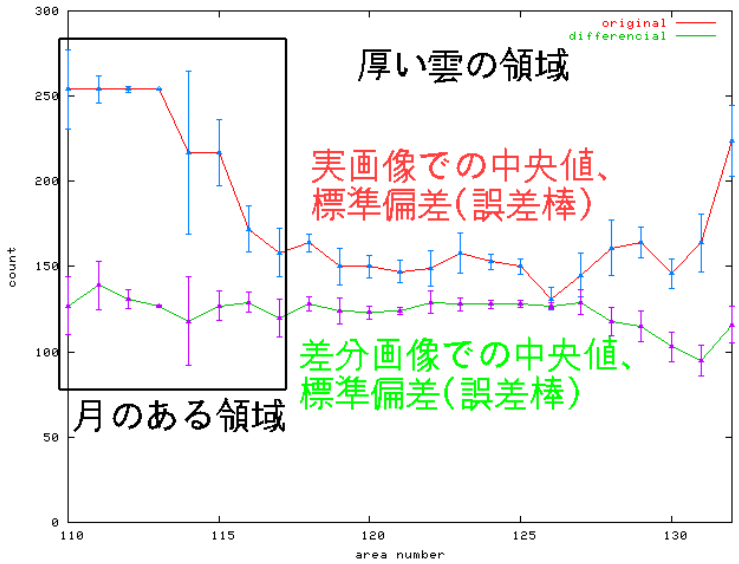


図 4.15: 月がある時の厚い雲領域一列のプロット

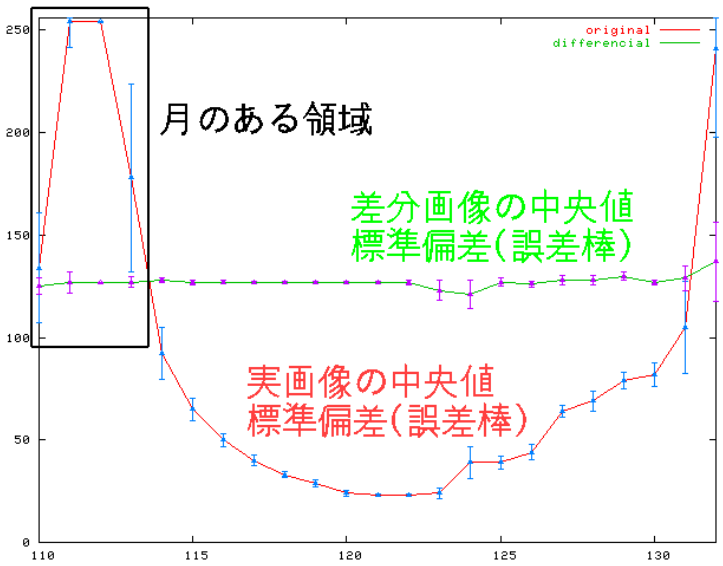


図 4.16: 月がある時の雲のない領域一列のプロット

図 4.15 は月があるときに厚い雲広がっているときの区分領域一列の諸統計量の値をプロットしたものであり、図 4.16 は月があり雲が無いときの諸統計量をプロットしたものである。月が無い場合のグラフと同様に赤線が X_m、誤差棒が σ_m、緑線が $X_{m,d}$、誤差棒が $\sigma_{m,d}$ である。横軸は左端の領域からの番号であり、縦軸は中央値の値である。月がある領域を長方形で囲っている。月がある時は月の影響が大きく、月が背後にある領域、月周辺の領域は統計量が他と大きく異なっているのが分かる。月がある領域は特異な統計量となるため、雲判定が難しく、実際の望遠鏡観測自体も行われない。月周辺の実測値から月を判定するパラメータを設定し、月周辺の領域を省く処理をする。月が無い領域は X_m の値は 45.0 が雲領域の下限であった。σ_m、$X_{m,d}$、$\sigma_{m,d}$ は雲が無い領域はほぼ一定値になり、雲がある領域はばらついた。

4.3.1 雲判定条件

月が無い場合の雲判定パラメータとして、$X_m = 45.0$、$\sigma_m = 2.8$ を下限に設定し、周囲の領域の標準偏差の値が 2.8~7.2 の間でばらついている時に雲と判定する方法が考えられる。周囲の領域と比較する理由は、雲領域は常に標準偏差が高い値ではなく低い値をとる場合もあるからである。

月がある場合の雲判定パラメータとしては、まず月周辺では X_m が空の様子に関係なく 147.0~254.0 の範囲の値を取り、$X_{m,d}$ が 115 以下または 135 以上を取る領域がその領域の周囲にある時、月近傍であるとして省く処理をする。これには、別途用意する月位置計算ソフトウェアの併用も見込んでいる。

次に雲判定であるが、$X_m = 45.0$ 以上であることを第一条件とし、第一条件を満たしたものの中で σ_m が 3.72 以上または $\sigma_{m,d}$ が 2.0 以上の値で且つ、周囲に同条件を満たす領域がある時その領域を雲と判定する。なお $X_{m,d}$ は雲と空の境界では雲の移動方向に対し前方であれば値は増加し、後方であれば減少する。連続して出力される画像に対して 2 枚ずつ画像差分を行い比較することで、雲の移動方向や雲の速度の推定も可能であると考える。

雲判定条件により高度 30 °以上の雲のある領域をマッピングする機能を追加した。図 4.17 がその図である。位置計算機能と同時に雲領域のマッピングが可能である。

雲判定パラメータは 2006 年 1 月現在の夜空の雲の傾向であり、月齢や季節、雲の種類、観測地の風が強い等の条件により異なる可能性がある。今後は観測地での測定を測定を繰り返し、より校正を重ねる必要があるだろう。

図 4.17: 雲領域マッピング画面

4.4　観測スケジュール機能の提案

観測者は通常観測対象天体リストを元に、 空の現在の状況、気象予報、西側の空の様子から観測可能かどうかをそれぞれ見積る。それと観測天体の優先順位、天体の明るさに応じた必要観測時間、および天体高度が 30° 以上で保つ観測可能時間を見積り、比較検討を行うことで、観測スケジューリングが可能となる。空の状態等の情報は適時更新を行い、その都度観測スケジュールは検討、変更を行う。私は観測スケジュール機能の開発までは至らなかったが、私が開発したソフトウェアの出力情報と天体リストの情報 (座標、高度、観測モード) とを組み合わせることでこういった観測スケジューリングは可能であると期待できるので、観測周辺機器として実用化を実現するために求められる観測スケジュール機能構想の提案を行う。

§4.1 において全天画像を区分し、§4.2 で各領域内でのカウントの各種統計量を計算する機能を追加し、その整理を行った。それから雲判定パラメータを考案し、雲領域のマッピングを行った。観測スケジュール機能の開発には自動雲検知アルゴリズムの完成が必須であり、今後は私が考案した雲判定パラメータの校正を重ね、月領域や、雲の種類に対応したものにする必要がある。

観測スケジュール機能ではまず観測対象天体リストを作成し、それらの優先順位、必要観測時間、赤経、赤緯等のデータを載せておく。全天スカイモニターにより出力される画像に対し、ソフトウェアの自動雲検知機能で判定された雲領域のマッピングを行い、それを観測天体位置と照合し、現在の高度、方位角、雲の有無とその濃さを観測対象天体のデータファイルに与える。さらに雲領域のマップにおいて雲無しと判定された領域では観測可能な残り時間を、雲ありと判定された領域では予測可能なときは次回観測可能になるまでの時間をデータファイルに与える。これらのデータを観測対象天体の必要観測時間、優先順位、観測可能時間と照合することで観測スケジューリングを行い、次に観測すべき天体を決定して、望遠鏡制御ソフトウェアへフィードバックする。(図 4.18)　観測スケジュール機能にはこれら種々のデータを元にした条件別に場合分けをした観測スケジュールを組むアルゴリズムの開発も同時に求められる。

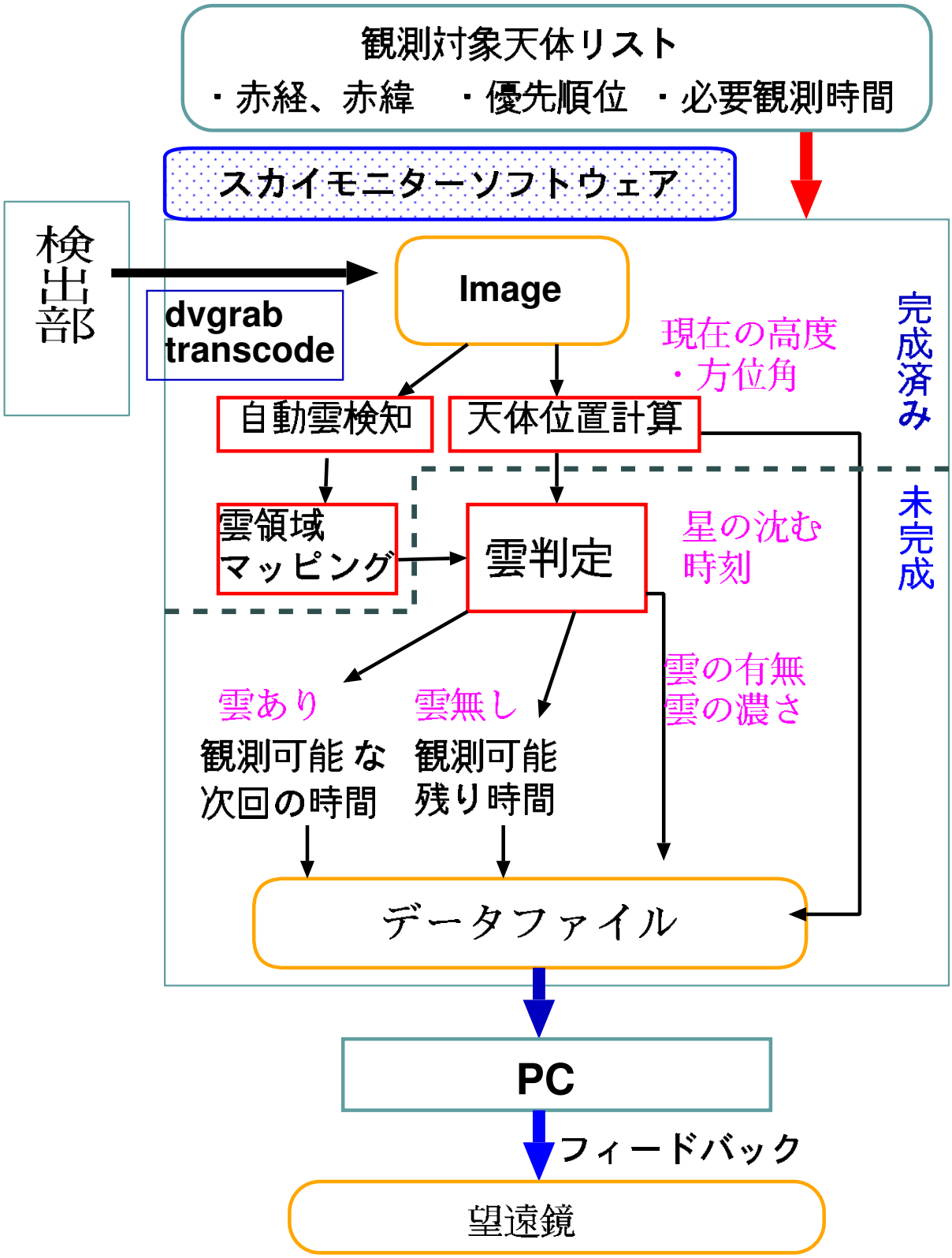


図 4.18: 観測スケジュール機能の流れ

第5章　まとめ

私は広島大学東広島天文台計画の一貫として、即応自動観測化と遠隔化を推進するために、望遠鏡周辺機器の一つである全天スカイモニターの製作と観測スケジュール機能の開発に取り組んだ。

私は全天スカイモニターの設計、組み上げを行うと共に、GTK を用いた画像取り込み、PC 上への画像表示、その自動更新、アーカイブへの自動データ保存機能を有したソフトウェアを作成した。複数晩にわたり天文台サイトの空の雲の様子を測定し、雲の変動の様子と 3 等程度の恒星の撮像することに成功した。これにより装置の安定稼働を確認し、広い視野と高感度を兼ね備えた所定の性能をもつことが分かった。

次に 3 日間の測定で得られた画像を用いて、雲領域と観測天体位置を照合するために観測対象天体位置を時刻、観測対象天体の赤経、赤緯から計算し、画像上の座標に変換する機能を加えた。さらに作成した天体リストから画像上の天体位置を計算し、マッピングする機能をソフトに追加することで、雲領域との照合を可能にした。

また雲検知アルゴリズムの開発を目指して全天の領域分けを行った。区分領域のカウントの各種統計量を計算する機能の追加とその整理を行い、雲判定パラメータを考案し、雲領域をマッピング可能にした。

今後は、継続的に天文台サイトにおける測定を行い、季節や雲の形等の条件に依らない雲判定パラメータの決定を行う。これにより観測者は全天の雲領域と観測対象天体位置との照合が確実になり、観測効率はかなり上昇する。また、全天スカイモニターの実用化に向けた私が構想した自動観測スケジュール機能の開発に着手したい。

付 録A　ソースコード

A.1　画像の自動更新プログラム

```
#include <gtk/gtk.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <math.h>
#include <gdk-pixbuf/gdk-pixbuf.h>
#include <sys/types.h>
#include <time.h>

#define IMG_HGT 480
#define IMG_WID 720
#define PPMPATH "/home/skymon/data/ppm/"    //Current ppm file path
#define DEF_TIMER 30000  //Default time interval of dv read-out

//Time struct
typedef struct{
  int year, month, day, hour, min, sec;
}myTime;

//Time functions
int get_time( myTime *ttime );  //Get current time
int get_localtime( myTime *ttime ); //Get local time

/****Global struct****/
typedef struct{
  int time_reload;       //Interval for reload dv in msec
  char dvfilepath[256];  //Path of .ppm file(convert from dv image)
  myTime ctime;          //Time
}Skymon;

void set_ppm_path( void );                  //Set path for image read
void dvgrab( void );                        //Image read from DV
int load_pix_buffer( void );                //Load pix buffer
int copy_to_gray_buffer( void );            //Copy pixbuf to graybuffer
int init_cap_window( GtkWidget *window_cap );//Initialization
void redraw_image( void );                  //Redraw image
void reload_image( gpointer* pointer );
void timeout_reload_image( gpointer pointer );
void starttimer( void );
void stoptimer( void );
void callback_open_window_cap ( GtkWidget *widget, gpointer data );
void callback_hide_window_cap ( GtkWidget *widget, gpointer data );
gboolean on_darea_expose ( GtkWidget *widget, GdkEventExpose *event,
  gpointer user_data ); //Callback for expose_event

gint delete_event( GtkWidget *widget, GdkEvent *event, gpointer data );

/****Global variables****/
guchar* graybuf;               //pointer for gray-scale image buffer
int wid, hgt;                 //image size
GtkWidget *g_image_area;        //captured image area, use in cap_window
GtkWidget *window_capture;       //capture window
gint timertag;                   //timer tag, used in cap_window
GdkPixbuf *pixbuf;

/* Generic global settings */
Skymon mskymon;

int get_time( myTime *ttime ){ //Get curremt time
  int now, yr;
  int d, t, i, flg1;
  int mday[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

  //2002.01.01
  //"8" is a leap years
  d = ( int )( ( ( 2001-1970 )*365 + 8 )*24*3600 );
  now = time( NULL );
  now -= d;
  t = now;
  yr = 2001;
  i = 1;
  flg1 = 1;
  while( flg1 ){
    t -= mday[i]*24*3600;
    if( t<0 ){ flg1 = 0;
    }else{
       i++;
      if( i>12 ){
yr++;
i = 1;
if( ( yr%4 ) == 0 ){
  mday[2] = 29;
}else{
           mday[2] = 28;
        }
       }
     now = t;
   }
  }
  ttime->year = yr;
  ttime->month = i;
  ttime->day = now/( 24*3600 ) + 1;
  now -= ( ttime->day-1 )*24*3600;
  ttime->hour = now/3600;
  now -= ttime->hour*3600;
  ttime->min = now/60;
  now -= ttime->min*60;
  ttime->sec = now;

  return( EXIT_SUCCESS );
}

int get_localtime( myTime *ttime ){  //Get current local time
  int ret;

  ret = get_time(ttime);
  ttime->hour += 9;          //Time-zone difference
  return(ret);
}

void set_ppm_path( void ){  //Set show path
  strcpy( mskymon.dvfilepath, PPMPATH );
  strcat( mskymon.dvfilepath, "skymon000000.ppm" );
}

/**************Main Function*****************/
int main( int argc, char *argv[] ){
  GtkWidget *window;
  GtkWidget *window_cap;
  GtkWidget *button, *vbox, *hbox;

  /* This is called in all GTK applications. */
  gtk_init( &argc, &argv );

  //Set defaults
  timertag = -1;
  set_ppm_path();
  mskymon.time_reload = DEF_TIMER;

  window = gtk_window_new( GTK_WINDOW_TOPLEVEL );
  window_cap = gtk_window_new( GTK_WINDOW_TOPLEVEL );

  gtk_window_set_title( GTK_WINDOW( window ), "Skymon:Main" );
  gtk_window_set_title( GTK_WINDOW( window_cap ), "Skymon:Capture" );

  gtk_signal_connect( GTK_OBJECT( window ), "delete_event",
     GTK_SIGNAL_FUNC( delete_event ), NULL );

  gtk_container_set_border_width( GTK_CONTAINER( window ), 2 );
  gtk_container_set_border_width( GTK_CONTAINER( window_cap ), 10 );

  //Making contents of Main Window
  vbox = gtk_vbox_new( FALSE, 5 );
  hbox = gtk_hbox_new( FALSE, 0 );

  gtk_container_add( GTK_CONTAINER( window ), vbox );
  gtk_box_pack_start( GTK_BOX( vbox ), hbox, TRUE, TRUE, 0 );

  //Button for capture window
  button = gtk_button_new_with_label( "capture window" );
  gtk_signal_connect( GTK_OBJECT( button ), "clicked",
     GTK_SIGNAL_FUNC( callback_open_window_cap ),
     ( gpointer )window_cap );

  //We pack this button into the invisible box  packing into the window
  gtk_box_pack_start( GTK_BOX( hbox ), button, TRUE, TRUE, 0 );
  gtk_widget_show( button );

  //Do these same steps again to create a closebutton
  button = gtk_button_new_with_label( "close" );
  gtk_signal_connect( GTK_OBJECT( button ), "clicked",
     GTK_SIGNAL_FUNC( delete_event ), NULL );

  gtk_box_pack_start( GTK_BOX( hbox ), button, TRUE, TRUE, 0 );
  gtk_widget_show_all( window );

  //Making contents of capture window
  init_cap_window( window_cap );
  window_capture = window_cap;

  printf( "now starting skymonitor..\n" );
  gtk_main();

  return 0;
}

void dvgrab( void ){   //Image read from DV

  /*Shell skymon (Bottom of this source) includes with
     dvgrab and transcode made ppm files */
  system( "skymon.sh" );
```

```
  //It send ppm files to current ppm path
  system( "cp skymon000000.ppm PPMPATH" );
}

int load_pix_buffer( void ){ //Load pix buffer

  if( pixbuf != NULL ){  //Release and destroy previous pixbuf
    g_object_unref( pixbuf );
  }
     //Load from file preliminary.
  if((pixbuf = gdk_pixbuf_new_from_file(mskymon.dvfilepath, NULL))
== NULL ){
      printf( "failed opening ppm file.path:%s\n",mskymon.dvfilepath );
      return(EXIT_FAILURE);
  }
    //Add reference
  wid = gdk_pixbuf_get_width( pixbuf );
  hgt = gdk_pixbuf_get_height( pixbuf );

  return( EXIT_SUCCESS );
}

int copy_to_gray_buffer( void ){ //Pixbuf to graybuffer
  int i, j;
  guchar *inpix, *outpix;

  if( ( inpix = gdk_pixbuf_get_pixels( pixbuf ) ) == NULL ){
    printf( "error in reading pixbuf.\n" );
    return( EXIT_FAILURE );
  }
  outpix = graybuf;

  for( i=0;i<hgt;i++ ){
    for( j=0;j<wid;j++ ){
      *outpix = ( guchar )(( *inpix + ( *inpix + 1 ) + ( *inpix + 2 ))/3);
      inpix += 3;
      outpix++;
    }
  }
  return( EXIT_SUCCESS );
}

void redraw_image( void ){  //Redraw function

  gdk_draw_gray_image( GDK_DRAWABLE( g_image_area->window ),
        g_image_area->style->fg_gc[GTK_STATE_NORMAL],
      0, 0, wid, hgt, GDK_RGB_DITHER_MAX, graybuf, wid );
}

void reload_image( gpointer* pointer ){ //Reload image to pixbuf
  GtkWidget *w;
  w = *pointer;
  myTime t;
  char buf[30], buf2[30], com_buf[40], com_buf2[30], buf_dir[30];
  int last_min = 999;

  get_localtime( &t );

  //Save ppm files once three minutes to Archive
  if( ( t.min%3 ) == 0 && last_min != t.min ){
    dvgrab();
    sprintf( buf, "%04d%02d%02d_%02d%02d%02d", t.year, t.month,
    t.day, t.hour, t.min, t.sec );   //Time stump name

    // Because of capacity of my PC, files arec converted to JPEG type
    sprintf( com_buf, "convert skymon000000.ppm %s.jpg", buf );
    system( com_buf );
    sprintf( buf_dir, "%04d%02d%02d", t.year, t.month, t.day );

    if( ( access( buf_dir, 0 ) ) == -1 ){ //Access the directry
      sprintf( buf2, "mkdir %s", buf_dir );
      system( buf2 );
    }
    sprintf( com_buf2, "mv *.jpg %s", buf_dir );
    system( com_buf2 );
    last_min = t.min;

  }else{            // normal reload mode
    dvgrab();
    system( "rm *.ppm" );
  }
  if( ( load_pix_buffer() ) ){  //load to pixbuffer
    printf( "error in reading from capture image\n" );
    return;
  }
  if( !( copy_to_gray_buffer() ) ){
    redraw_image();
  }
}

void timeout_reload_image( gpointer pointer ){
   reload_image( &pointer );
}

void starttimer( void ){  //Timer for reloading image
  timertag = gtk_timeout_add( mskymon.time_reload,
     ( GtkFunction )timeout_reload_image,
     g_image_area );
}

void stoptimer( void ){   // Stop Timer
  gtk_timeout_remove( timertag );
  timertag = -1;
}
```

```
//It's important for self-reloading system. This is expose-event
gboolean on_darea_expose( GtkWidget *widget, GdkEventExpose *event,
 gpointer user_data ){

  if( ( copy_to_gray_buffer() ) ){
     gdk_draw_gray_image( GDK_DRAWABLE( widget->window ),
widget->style->fg_gc[GTK_STATE_NORMAL], 0, 0,
wid, hgt, GDK_RGB_DITHER_MAX, graybuf, wid );
  }
  return TRUE;
}

//Open capture window
void callback_open_window_cap( GtkWidget *widget, gpointer data ){

  gtk_widget_show( ( GtkWidget* )data );
  if( timertag == -1 ){
    starttimer();
  }
}

//Hide capture window
void callback_hide_window_cap( GtkWidget *widget, gpointer data ){

  gtk_widget_hide( ( GtkWidget* )data );
  stoptimer();
}

//Initialization of cap_window
int init_cap_window( GtkWidget *window_cap ){
  GtkWidget *button, *box_cap;

  box_cap = gtk_vbox_new( FALSE, 0 );
  gtk_container_add( GTK_CONTAINER( window_cap ), box_cap );

  pixbuf = NULL;
  if( ( load_pix_buffer() ) ){
    printf( "error in reading from capture image\n" );
    return( EXIT_FAILURE );
  }
  if( ( graybuf = ( guchar* )malloc( sizeof(unsigned char)*wid*hgt ) )
== NULL ){
    printf( "failed in allocating memory for image buffer.\n" );
    exit( EXIT_FAILURE );
  }

  gdk_rgb_init();  //Init of GdkRGB

  g_image_area = gtk_drawing_area_new();
  gtk_drawing_area_size( GTK_DRAWING_AREA( g_image_area ), wid, hgt );
  gtk_signal_connect( GTK_OBJECT( g_image_area ), "expose-event",
      GTK_SIGNAL_FUNC( on_darea_expose ), NULL );

  gtk_box_pack_start( GTK_BOX( box_cap ), g_image_area, TRUE, TRUE, 0 );
  gtk_widget_show( g_image_area );

  //Close button for capture window
  button = gtk_button_new_with_label( "Close" );
  g_signal_connect( G_OBJECT( button ), "clicked",
    G_CALLBACK( callback_hide_window_cap ), window_cap );

  gtk_box_pack_start( GTK_BOX( box_cap ), button, FALSE, FALSE, 0 );

  gtk_widget_show( button );
  gtk_widget_show( box_cap );

  return(EXIT_SUCCESS);
}

gint delete_event( GtkWidget *widget, GdkEvent *event, gpointer data ){
  GtkWidget *dialog;
  gint reply;

  dialog = gtk_message_dialog_new( NULL, GTK_DIALOG_MODAL,
  GTK_MESSAGE_QUESTION, GTK_BUTTONS_YES_NO,
  "Are you sure to quit Skymonitor?" );

  reply = gtk_dialog_run( GTK_DIALOG( dialog ) );
  gtk_widget_destroy( dialog );

  if( reply == GTK_RESPONSE_YES ){
    gtk_main_quit();
  }
  return 0;
}
```

```
[skymon.sh] //Shell grab a single dv frame
rm /home/skymon/sam/skymon001.dv
dvgrab --format raw --frame 1 --duration 0.1 skymon
transcode -i skymon001.dv -x dv,null -y ppm,null -o skymon
```

謝辞

本研究にあたり突風の吹く氷点下近い気温の天文台サイトへの移動、装置の設置、回収につきあっていただいたり、分からない用語や装置製作の方法など懇切丁寧に教えていただいた川端先生に、心から感謝いたします。また、科学論文のいろはやソフトウェア作成のアドバイスなどをしてくれた山下先生、深沢先生、植村さん、自身の論文作成で忙しい中ソフトウェア開発環境の立ち上げや論文作成の際、数々のアドバイスをくれた千代延さん、高橋さん、沢本さんにも感謝します。

環境面におきましては、研究室のみなさまの明るい雰囲気のおかげで疲労で辛いときも気力だけは充実した生活が送れました。夜通し一緒に研究にせいを出したみなさま本当にありがとうございました。

最後になりましたが、事務室で色々な面で研究を支えてくれた上原さん、石井さんどうもありがとうございました。

関連図書

[1] 岩田、http://www.kusastro.kyoto-u.ac.jp/%7Eiwata/newtel/seemon2/ (2003)

[2] 千代延真吾、広島大学 1.5m 望遠鏡移設地シーイングのモニター装置開発と測定、2004 広島大学卒業論文

[3] 国立天文台岡山天体物理観測所 40 周年記念誌 (2001)

[4] Martin.V.Zombeck:*Handbook of space astronomy and astrophysics* **110** , Cambridge University Press, 1990

[5] 高橋 麻奈、やさしい C、 風工舎 (2003)

[6] 国立天文台編 理科年表 (1996)、 **38**, No. 11, 1427

[7] Tony Gale, IAn Main, *GTK v1.2 tutorial* http://www.kitanet.ne.jp/ asler/linux/gtk/

[8] Sky Catalogue 2000.0, Eds. A. Hirshfeld and R. W. Sinnott, Cambridge University Press (1982)