KOREA ASTRONOMY AND SPACE SCIENCE INSTITUTE

기술보고서 No. 13-001-111

MIRIS 우주관측카메라 자료처리 파이프라인 사용 설명서

표정현, 박원기, 김일중, 정웅섭, 이대희, 박영식, 문봉곤, 이덕행, 한원용

776 Daedukdae-ro, Yuseong-gu, Daejeon 305-348, Korea

요 약

<table>
<tr><td rowspan="2">제 목</td><td>국문</td><td colspan="3">MIRIS 우주관측카메라 자료처리 파이프라인 사용 설명서</td></tr>
<tr><td>영문</td><td colspan="3">User Manual of the MIRIS SOC Data Reduction Pipeline</td></tr>
<tr><td>관련과제명</td><td colspan="4">과학기술위성 3호 주탑재체 개발(6차년도) (2011282000)</td></tr>
<tr><td>저 자 명</td><td colspan="4">표정현, 박원기, 김일중, 정웅섭, 이대희, 박영식, 문봉곤, 이덕행, 한원용</td></tr>
<tr><td>작성일자</td><td colspan="2">2013년 4월 1일</td><td>KMS 공개</td><td>공개(O), 비공개( )</td></tr>
<tr><td>기술분류1</td><td colspan="2">기술보고서</td><td>기술분류2</td><td>XE2</td></tr>
<tr><td>Keyword</td><td colspan="4">MIRIS, SOC, 자료처리, 소프트웨어, 파이썬</td></tr>
<tr><td>초록</td><td colspan="4">과학기술위성 3호(STSAT-3)의 주탑재체인 다목적 적외선 영상 시스템(MIRIS)의 우주 관측 카메라(SOC)는 한국천문연구원에서 처음으로 개발하는 위성용 적외선 영상 관측 기기이다. 관측 영상에서 과학적으로 의미 있는 결과를 얻기 위해서는 우선 자료 처리 과정을 통하여 관측 기기의 특성이 보정된 영상을 얻어야 한다. 이러한 과정을 수행하는 프로그램을 자료처리 파이프라인이라 한다. MIRIS의 SOC를 위하여 개발된 자료처리 파이프라인 소프트웨어 SOCdr은 고급 프로그래밍 언어인 파이썬(Python)과, 파이썬을 기반으로 한 천문학용 라이브러리인 Astropy를 이용하여 개발되었다. 본 보고서는 SOCdr의 사용 방법에 대해 설명한다.
소프트웨어 및 시험에 활용한 영상은 별도의 CD에 저장하여 제출한다.</td></tr>
</table>

목 차

그림 목차

표 목차

약어 및 기호 정의

명 칭	의 미
CASU	Cambridge Astronomy Survey Unit
FITS	Flexible Image Transport System
HDU	Header Data Unit
IPAC	Infrared Processing and Analysis Center
MIRIS	Multipurpose Infrared Imaging System, 다목적 적외선 영상 시스템
PyPI	Python Package Index
SIP	Simple Imaging Polynomial
SOC	Space Observation Camera, 우주 관측 카메라
STSAT-3	Science and Technology Satellite-3, 과학기술위성 3호
WCS	World Coordinate System

1. 소 개

SOCdr은 MIRIS SOC의 관측 자료를 처리하기 위한 여러 요소들로 이루어진 파이프라인 소프트웨어이다. 이 소프트웨어는 순수하게 파이썬으로 만들어졌으며 여러 써드파티(third-party) 파이썬 패키지들과 외부 프로그램에 의존한다. 파이썬으로 짜여진 부분은 파이썬의 객체 지향(object-oriented) 특성을 활용하여 효율적으로 작동하도록 만들어졌다. 즉, 자료를 처리할 때마다 공통적으로 필요한 자료나 계산을 미리 읽어들이거나 수행하여 클래스 내에 저장해 두는 것이다. 또한 자료 처리 시 수행되는 복잡한 계산 등은 사용자에게 직접 드러나지 않도록 숨겨 쉽게 사용할 수 있도록 하였다.

본 보고서의 2장에서는 SOCdr이 의존하는 써드파티 패키지와 외부 프로그램, 그리고 설치 방법에 대하여 설명한다. 3장에서는 SOCdr을 구성하는 각 자료 처리 요소에 대하여 설명한다. 4장에서는 SOCdr의 사용 방법에 대하여 기술하고 5장에서는 시험 수행 결과에 대해 논한다.

2. 설 치

2.1. 작동 환경

MIRIS SOC의 자료처리 파이프라인(SOCdr)은 파이썬을 기반으로 하지만, 일부 기능은 리눅스에서만 작동하는 외부 프로그램에 의존한다. 따라서 SOCdr은 기본적으로 리눅스 환경, 또는 이와 유사한 환경에서만 작동한다. 이 보고서는 Debian testing (2013년 3월 현재) 환경을 기준으로 설명한다. 한편, SOCdr에는 MySQL 데이터베이스에 접근하여 raw 데이터를 가져와 FITS로 변환하는 기능이 있는데, 이 경우에는 SOC의 raw 데이터가 저장되어 있고 데이터베이스가 있는 환경에서만 작동한다. 만일 raw 데이터 파일에서 변환된 FITS 파일을 가지고 있어 SOCdr의 mainproc 모듈과 postproc 모듈만 사용한다면 아무 컴퓨터에서나 사용할 수 있다.

한편, SOCdr은 파이썬 2.7.3 버전을 기반으로 개발되었으며, 2.7.x 버전의 파이썬에서는 문제없이 동작한다. 하지만 파이썬 3.x 버전에서는 문법이 대폭 변경되어 그 아래 버전의 프로그램과는 호환되지 않으며, SOCdr도 파이선 3.x 버전에서는 동작하지 않으므로 주의해야 한다.

2.2. 의존 소프트웨어

SOCdr은 파이썬의 빌트인(built-in) 라이브러리 외에 아래와 같은 써드파티 파이썬 패키지에 의존한다.

2.2.1. ConfigObj

ConfigObj는 SOCdr의 설정 파일인 socdr.cfg을 읽는데 사용된다. Debian testing의 경우, python-configobj 패키지를 설치하면 된다.

홈페이지: http://www.voidspace.org.uk/python/configobj.html

PyPI 홈페이지: https://pypi.python.org/pypi/configobj/

2.2.2. MySQL_python

MySQL_python은 파이썬에서 MySQL 데이터베이스에 접근하는데 사용되며, 1.2.3 버전 이상을 설치해야 한다. Debian testing의 경우, python-mysqldb 패키지를 설치

하면 된다.

홈페이지: http://mysql-python.sourceforge.net/

PyPI 홈페이지: https://pypi.python.org/pypi/MySQL-python/

2.2.3. PyEphem

PyEphem은 좌표 변환, 행성 좌표 계산 등 여러 천문학용 계산에 활용할 수 있는 라이브러리로, 3.7.5.1 버전 이상을 설치해야 한다. SOCdr에서는 태양과 달의 좌표 계산, 그리고 이들로부터의 회피각(avoidance angle)을 계산하는데 활용한다.

홈페이지: http://rhodesmill.org/pyephem/

PyPI 홈페이지: http://pypi.python.org/pypi/pyephem/

2.2.4. NumPy

NumPy는 산술 계산 등에 쓰이는 범용 파이썬 라이브러리로, 1.6 버전 이상을 설치해야 한다. Debian testing의 경우, python-numpy 패키지를 설치하면 된다. 이 라이브러리는 크기가 크고 소스에서부터 설치하기에는 시간이 오래 걸리므로 자신의 환경에 맞는 바이너리 패키지를 구하여 설치하기를 권한다.

홈페이지: http://www.numpy.org

PyPI 홈페이지: https://pypi.python.org/pypi/numpy

2.2.5. SciPy

SciPy는 여러 과학 계산에 쓰이는 라이브러리로, 0.10 버전 이상을 설치해야 한다. Debian testing의 경우, python-scipy 패키지를 설치하면 된다. 이 라이브러리도 소스에서부터 설치하기에는 시간이 오래 걸리므로 자신의 환경에 맞는 바이너리 패키지를 구하여 설치하기를 권한다.

홈페이지: http://www.scipy.org

PyPI홈페이지: https://pypi.python.org/pypi/scipy

2.2.6. Astropy

Astropy는 그동안 흩어져있던 천문학 관련 파이썬 라이브러리를 통합하고 파이썬에서 천문학 관련 계산을 하는 데 편의를 제공하는 라이브러리이다. 0.2 버전 이상을

설치해야 한다.

홈페이지: http://www.astropy.org/

PyPI 홈페이지: https://pypi.python.org/pypi/astropy

2.2.7. montage-wrapper

montage-wrapper는 아래에서 설명할 Montage 프로그램을 파이썬에서 접근할 수 있도록 해주는 라이브러리이다. 기존에는 python-montage로 알려져 있었으나 0.9.5 버전 이상에서는 Astropy의 제휴 프로그램으로 편입되면서 montage-wrapper로 이름을 바꾸었다. 0.9.5 버전 이상을 설치해야 한다.

홈페이지: http://www.astropy.org/montage-wrapper/

PyPI 홈페이지: https://pypi.python.org/pypi/montage-wrapper

위 패키지들은 모두 PyPI에 등록되어 있기 때문에 2.3절에서 설명하듯이 pip로 SOCdr을 설치하면, 컴퓨터가 인터넷에 연결되어 있는 경우, 자동으로 설치한다. 다만 위에서 설명하였듯이 NumPy와 SciPy는 크기가 크기 때문에 각 리눅스 배포판의 해당하는 패키지로 미리 설치해 두기를 권한다.

SOCdr은 위의 파이썬 패키지들 외에 아래와 같은 외부 프로그램을 활용한다. 이 외부 프로그램들은 파이썬 패키지들과 달리 수동으로 설치해야 한다.

2.2.8. SExtractor

SExtractor는 천문학 영상으로부터 광원들을 검출하여 카탈로그를 작성하는 프로그램으로, 최신 버전은 2.8.6이다. SOCdr에서는 WCS 정보를 보정할 때 SOC 영상에서 광원을 검출하는데 사용한다. SExtractor의 실행 파일은 sextractor라는 이름으로 사용자의 실행파일 검색 경로(PATH 환경 변수 확인)에 존재해야 한다. Debian testing에서는 sextractor 패키지를 설치하면 된다.

홈페이지: http://www.astromatic.net/software/sextractor

2.2.9. Montage

Montage는 IPAC에서 개발한 천문 영상 합성 프로그램으로, 최신 버전은 3.3이다. 홈페이지에서 소스코드의 압축파일 Montage_v3.3.tar.gz를 다운로드하여 압축을 풀고, Montage_v3.3/ 디렉터리의 README 파일을 참고하여 컴파일한다. 만일 MPI 버전의

실행파일도 만들려면 OpenMPI 컴파일러와 관련 라이브러리를 설치하고 Montage/Makefile.LINUX 파일을 편집하여 아래 두 줄의 주석 처리를 해제하고 컴파일한다.

```
MPICC = mpicc
BINS = $(SBINS) $(MBINS)
```

컴파일이 완료된 후, bins/ 디렉터리에 있는 실행파일들을 사용자의 실행파일 검색 경로로 옮긴다.

홈페이지: http://montage.ipac.caltech.edu/

2.3. 설치 방법

SOCdr을 설치하는 데에는 distribute 패키지가 필요하다. Debian testing에서는 python-setuptools와 python-pkg-resources 패키지를 설치하면 된다. 만일 distribute 패키지가 설치되어 있지 않으면 SOCdr을 설치하는 과정에서 인터넷 연결을 통하여 자동으로 설치된다. SOCdr 설치에는 pip 프로그램을 사용하면 편리하다. 내려받은 압축파일 socdr-x.x.x.tar.gz(x.x.x는 SOCdr의 버전)에 대하여, 시스템 디렉터리(/usr/local/lib/pythonx.x/dist-packages/)에 설치할 때에는 루트 권한으로 아래와 같이 실행하고

```
# pip install socdr-x.x.x.tar.gz
```

사용자 디렉터리(~/.local/lib/pythonx.x/site-packages/)에 설치할 때에는 아래와 같이 실행한다.

```
$ pip install --user socdr-x.x.x.tar.gz
```

pip 프로그램은 Debian testing의 경우, python-pip 패키지에 포함되어 있다.

2.4. 제거 방법

pip로 설치한 경우, 아래와 같이 실행하여 손쉽게 삭제할 수 있다.

```
$ pip uninstall socdr
```

시스템 디렉터리에 설치한 경우에는 루트 권한으로 실행해야 한다.

3. 자료 처리 과정

SOCdr의 전체적인 자료 처리 구성 요소의 구성과 자료의 흐름은 그림 1과 같다. 다만, 여기에서 위성에서 전송된 자료 파일인 MIRIS.dat를 MySQL 데이터베이스에 등록하는 기능은 SOCdr에 포함되어 있지 않다.

SOCdr은 크게 전처리(preprocessing), 주처리(main processing), 후처리(postprocessing) 부분으로 나눌 수 있고, 이들은 각각 socdr 패키지의 preproc, mainproc, postproc 모듈에 구현되어 있다. 이 외에 base와 utils 모듈이 있는데, 여기에는 내부적으로 사용하는 유틸리티 함수와 클래스가 정의되어 있고 일반 사용자는 직접적으로 사용하지 않는다.

전처리는 데이터베이스에 질의하여 raw 자료와 관련 메타자료를 가져오고, raw 자료를 FITS 형식으로 바꾼다. 주처리에는 통상적인 적외선 영상 처리 과정들이 구현되어 있다. 여기에는 가) 검출기의 배드 픽셀 가리기, 나) 검출기의 비선형성 보정, 다) 영상 차분, 라) 영상 평탄화, 마) 영상의 왜곡 보정이 포함된다. 후처리는 전처리 과정에서 영상에 삽입한 WCS 정보를 수정하고, 영상을 합성한다.

데이터베이스에서 자료를 가져오는 요소를 제외한 모든 요소들은 처리된 결과를 astropy.io.fits.HDUList 오브젝트로 넘겨주고, 만일 이들 요소를 실행할 때 writeto라는 키워드 인자에 파일 이름을 주면 이 파일에 FITS 형식으로 저장한다. 하지만 사용자는 되도록 각 자료 처리 구성 요소에 접근하기 보다는 전처리 구성 요소를 통합하는 함수인 getfits(), 주 처리 구성 요소를 통합하는 클래스인 MainProcessing(), 후처리 구성 요소를 통합하는 클래스인 PostProcessing()을 이용하길 권한다. 이들의 구체적인 사용 방법에 대해서는 4장에서 다룬다.

SOCdr에서 다루는 HDUList 오브젝트나 저장하는 FITS 파일에는 여러 개의 HDU가 들어있는 다확장 FITS(multi-extension FITS) 형식이다. 하나의 HDUList 오브젝트 또는 FITS 파일은 한 번의 SOC 관측에 해당한다. 여기에는 헤더 정보만 들어있는 주 HDU(astropy.io.fits.PrimaryHDU) 하나와 실제 영상 자료가 들어 있는 영상 HDU(astropy.io.fits.ImageHDU) 여럿이 들어간다. 영상을 처리하는 과정에서 일부 영상 HDU에서는 영상이 제거되고 헤더만 남게 될 수도 있다. 한 HDUList에 들어있는 영상들을 합성한 후에도 HDUList 오브젝트를 넘겨주는데, 여기에는 영상을 중간값으로 합성한 결과(확장이름 MEDIAN), 평균으로 합성한 결과(확장이름 MEAN), 그리고 합성 영상의 각 픽셀에서 대푯값을 얻는데 이용된 값의 개수(확장이름 NUMPNTS)가 ImageHDU로 저장되어 있다.

이 장에서는 SOC 관측자료가 등록되어 있는 데이터베이스의 구조와 각 자료 처리 구성 요소에 대하여 설명한다.

그림 1. SOCdr의 구성 및 자료 흐름.

3.1. SOC raw 자료 데이터베이스의 구조

SOC의 raw 자료 데이터베이스는 MySQL(5.5.28 버전)을 기반으로 하며, 현재(2013년 3월) SOCdr을 개발하는데 사용한 워크스테이션(203.253.237.100)에 구축되어 있다. 이 데이터베이스의 이름은 miris_db이며, soc_rawdata 테이블에 정의되어 있다. 일반 사용자는 soc_user 계정을 이용하여 읽기 전용으로 접근할 수 있다. 이 테이블에는 22개의 열이 표 1과 같이 정의되어 있다.

표 1. SOC raw 자료 테이블의 구조.

열	열 이름	자료 타입	FITS 키	설명	소스[1]
1	obsid	CHAR(12)	OBS-ID	관측 ID	프로그램
2	observer	CHAR(32)	OBSERVER	관측자	사용자
3	proposal	CHAR(6)	PROPOSAL	관측제안 ID	사용자
4	category	ENUM(...)[2]	OBS-CAT	관측 카테고리	사용자
5	object	CHAR(16)	OBJECT	관측 천체	사용자
6	filter	CHAR(8)	n/a	관측 필터	사용자
7	exposure	FLOAT	n/a	노출 시간 (단위: 초)	사용자
8	object_ra	DOUBLE	RA-OBJ	관측 천체의 적경 (단위: 도)	사용자
9	object_dec	DOUBLE	DEC-OBJ	관측 천체의 적위 (단위: 도)	사용자
10	ra_set	DOUBLE	RA-SET	설정 적경 (단위: 도)	사용자
11	dec_set	DOUBLE	DEC-SET	설정 적위 (단위: 도)	사용자
12	galon_set	DOUBLE	n/a	설정 은경 (단위: 도)	프로그램
13	galat_set	DOUBLE	n/a	설정 은위 (단위: 도)	프로그램
14	q1_set	DOUBLE	Q1-SET	설정 사원수의 첫 번째 성분	사용자
15	q2_set	DOUBLE	Q2-SET	설정 사원수의 두 번째 성분	사용자
16	q3_set	DOUBLE	Q3-SET	설정 사원수의 세 번째 성분	사용자
17	q4_set	DOUBLE	Q4-SET	설정 사원수의 네 번째 성분	사용자
18	time	DATETIME	n/a	관측 시작 시각	프로그램
19	nframes	SMALLINT UNSIGNED	n/a	관측 프레임의 수	프로그램
20	flag	ENUM(...)[3]	n/a	관측 상태 플래그	사용자
21	time_retrv	DATETIME	n/a	자료를 받은 시각	프로그램
22	filename	CHAR(255)	n/a	raw 자료 파일의 절대 경로	프로그램

1) 소스는 값을 어디에서 가져오는 지를 나타낸다. '사용자'는 자료를 등록하는 과정에서 직접 제공해야 하는 값을, '프로그램'은 자료를 등록하는 프로그램에서 계산해서 넣는 값을 의미한다.

2) category의 타입은 ENUM('XX', 'PT', 'DT', 'MT', 'OT')이다. 'XX'는 정의되지 않은 경우이고, 'PT', 'DT', 'MT', 'OT'는 각각 performance verification time, director time, mission time, open time을

soc_rawdata 테이블에서 원하는 자료를 검색할 때에는 socdr.preproc.searchdb.search_db() 함수를 사용한다. 이 함수에 SQL SELECT문의 WHERE 뒤에 들어갈 질의문을 주면 테이블을 검색한 결과를 넘겨준다. 다만, 이 함수를 사용하려면 테이블에 접근하기 위한 soc_user 계정의 암호를 입력해야 한다. 기본적으로 검색 결과로는, 주어진 질의문에 맞는 자료 파일들의 절대 경로(filename)를 넘겨준다. 만일 표 1의 모든 정보를 얻으려면 search_db() 함수의 full_output 키워드 인수에 True 값을 지정한다. 이 경우에는 검색 결과를 astropy.table.Table 오브젝트로 넘긴다. 이 표의 각 열은 표 1과 같이 정의되어 있다.

3.2. Raw 자료를 FITS 형식으로 변환

SOC의 raw 자료를 FITS 형식으로 변환하는 데에는 socdr.preproc.raw2fits.raw2fits() 함수를 이용한다. 이 함수에는 raw 자료 파일의 경로를 인수로 준다. 이 외에 키워드 인수로 auxinfo에 파이썬의 사전(dictionary)과 같은 오브젝트를 줄 수 있다. auxinfo는 FITS 파일의 주 HDU의 헤더에 일부 값을 채워 넣는데 사용된다. auxinfo를 이용하여 값을 넣는 헤더 정보에 대해서는 표 1의 "FITS 키" 열에 정리되어 있다. auxinfo는 주어지지 않아도 되지만 일부 헤더 값에 의미 없는 값이 들어간다. 일반적으로는 search_db()에 full_output 인수에 True 값을 주어 얻은 Table 오브젝트를 auxinfo 인수의 값으로 주면 된다.

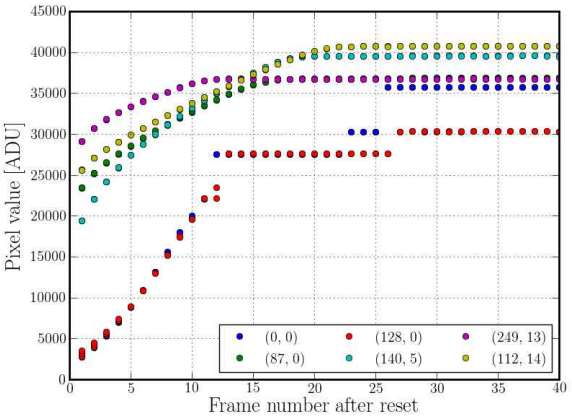
변환 과정에 대한 자세한 내용은 별도의 기술노트(김일중, 표정현, 박원기 등, 2013)를 참고한다. raw2fits() 함수는 앞에서 설명하였듯이 하나의 주 HDU와 여러 개의 영상 HDU로 이루어진 HDUList를 넘겨준다. 다만 리셋 영상의 경우에는 영상 자료는 제거되고 헤더만 갖는 영상 HDU가 된다.

3.3. 배드 픽셀 가리기

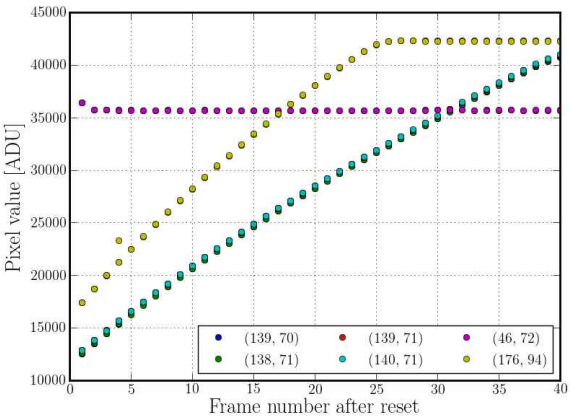
SOC 검출기의 선형성을 조사하는 과정(3.4절 참조)에서 픽셀들의 반응 곡선을 보고 이상한 반응 곡선을 보이는 픽셀들을 찾아냈다. 배드 픽셀로 판단한 기준은 아래와 같다.

의미한다.

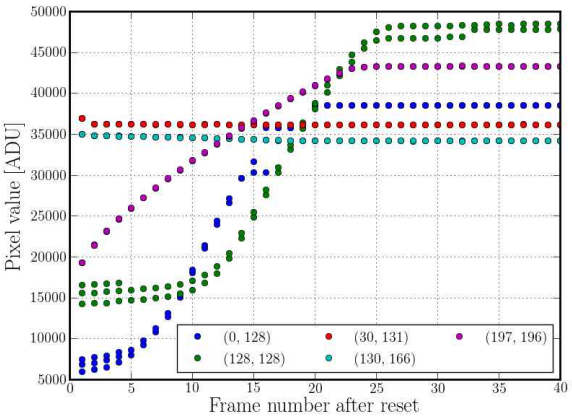
3) flag의 타입은 ENUM('SUCC', 'FAIL', 'LOST', 'UNKNOWN')이다. 'SUCC'는 성공, 'FAIL'은 실패, 'LOST'는 관측이 수행되지 않았음, 'UNKNOWN'은 정의되지 않았음을 의미한다.



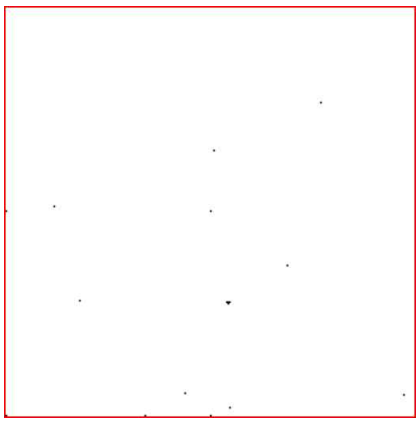
(가) 배드 픽셀의 반응 곡선 (1/3).



(나) 배드 픽셀의 반응 곡선 (2/3).



(다) 배드 픽셀의 반응 곡선 (3/3).



(라) 영상에서 배드 픽셀의 위치.

그림 2. 배드 픽셀의 반응 곡선(가–다) 및 SOC 영상에서 배드 픽셀의 위치(라).

1. 반응 곡선에서 노출 계량이 10보다 작은 부분을 1차 함수로 맞추었을 때, 아래 조건 중 하나라도 만족한다.
   - 기울기가 0이라는 귀무가설의 유의 확률(p–value)이 10^{-17}보다 크다.
   - 상관 계수(correlation coefficient) 0.98보다 작다.
   - 표준 오차(standard error)가 25보다 크다.
2. 포화값이 41,000 ADU보다 작은 픽셀들 중 반응 곡선이 이상하다.

여기에 더하여 좌표[4]가 (138, 71), (139, 70), (139, 71), (140, 71)인 픽셀들은 다른 픽셀들과 비교하여 유달리 낮은 반응성을 보여서 이들도 배드 픽셀로 판단하였

4) 이 문서의 픽셀 좌표는 영상의 가장 왼쪽 아래 픽셀의 좌표를 (0, 0)으로 하는 관습을 따른다. 참고로 FITS 표준과 ds9은 이 픽셀의 좌표를 (1, 1)로 하는 관습을 따르므로 주의한다.

다. 위와 같은 조건으로 찾은 배드 픽셀은 총 17개였다. 이들의 위치와 반응 곡선은 그림 2와 같다.

SOCdr에서 배드 픽셀의 위치는 FITS 형식 파일로 저장되어 있으며, 이 파일의 이름 및 경로는 파이프라인 설정 파일에 정의되어 있다(부록 1 참조). 이 FITS 파일은 SOC 영상과 같은 크기를 가지고 있는데, 값이 0보다 큰 픽셀이 배드 픽셀이다.

배드 픽셀을 가리는 기능은 socdr.mainproc.MaskPixels() 클래스에 구현되어 있다. 이 클래스는 초기화되었을 때 배드 픽셀의 위치가 저장된 영상 파일을 읽어 저장한다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 영상 HDU의 영상 자료에서 배드 픽셀에 해당하는 픽셀들에 NumPy의 NaN 값을 지정한다. 그 후, 수정된 HDUList를 넘겨준다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 MaskPixels() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 MASKBADP 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 MaskPixels() 클래스가 초기화될 때, 그리고 불러질 때 준 인자들이 기록된다.

3.4. 비선형성 보정

SOC 검출기의 비선형성을 보정하기 위하여 SOC의 필터를 H 밴드로, 노출시간을 1.5초로 고정하고 픽셀값이 초과될 때까지 촬영한 영상을 활용하였다. SOC의 검출기는 리셋을 하기 전까지는 이전 리셋 이후에 쌓인 전자를 계속 쌓아두면서 전자의 수를 노출시간 단위로 읽는 방식으로 작동한다. 그러므로 검출기에 입사한 총 광량은 광원의 플럭스가 일정하다고 가정할 때, 리셋 이후 영상 숫자(영상 HDU 헤더의 FRAMERST 값)에 비례하게 되고, 이런 특성을 이용하여 비선형성을 조사할 수 있다. 동일한 실험을 두 번 반복하였기 때문에 각 FRAMERST에 대하여 두 개의 자료값이 존재한다. 따라서 이 두 자료값의 평균을 이용하여 분석하였다.

검출기의 선형성을 조사하기에 앞서 포화값을 정하였다. 위에서 설명한 실험 자료의 경우, 포화는 FRAMERST가 대개 30이상일 때 발생하며 35 이상일 때는 거의 대부분의 픽셀이 포화된다. 이를 바탕으로 아래의 방법을 통하여 포화값을 조사하였다.

1. FRAMERST가 최대(40)인 경우의 평균값을 35인 경우의 평균값으로 나눈 것이 1.01보다 작은 경우, 포화된 것으로 판단하였다.

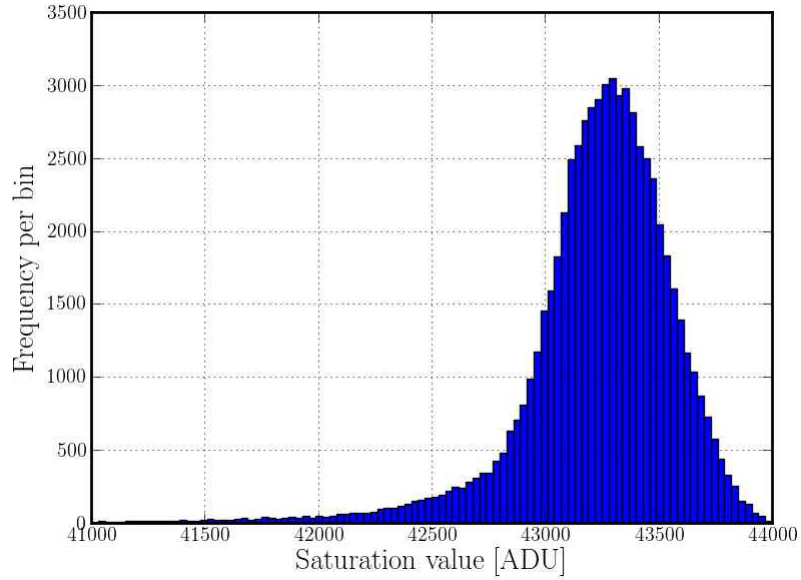


그림 3. 포화값의 히스토그램.

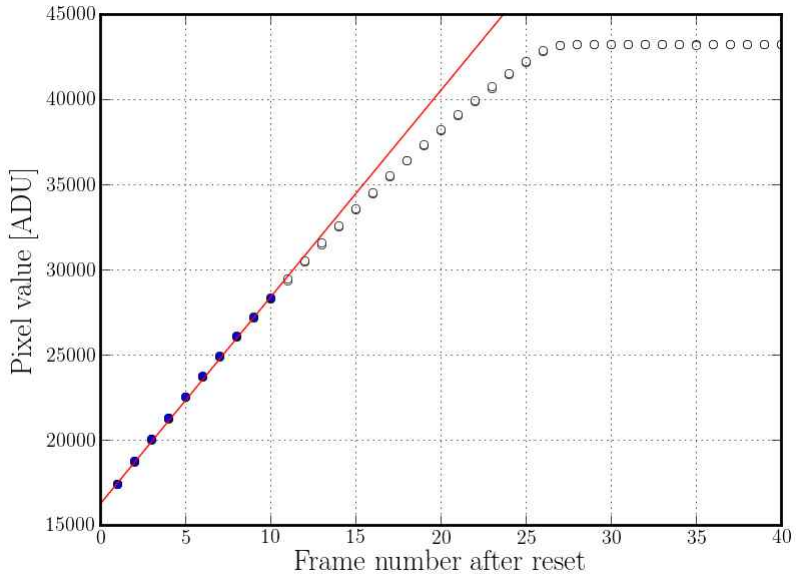


그림 4. 픽셀 (224, 23)의 반응 곡선(점)에 대한 선형 맞춤 결과(빨간 선). 파란 점은 맞춤에 이용된 것들이고 열린 점은 그렇지 않은 것들이다.

2. 위 조건을 만족하는 픽셀들에 대하여 FRAMERST가 35보다 크거나 같은 경우의 값들 중 가장 작은 것을 포화값으로 택하였다.

이렇게 계산한 포화값들의 3σ−clipping 후 평균은 47275.37, 표준편차는 256.50이었다. 이들의 히스토그램은 그림 3과 같다.

여기에서 포화값은 충분히 작게 잡아 41,000 ADU로 정하였다. 위와 같이 얻은 포화값들(총 65,530개) 중 41,000 ADU보다 작은 것은 173개로 0.26% 정도이다. 따라서 포화값을 41,000 ADU로 정하였을 때, 포화하였는데도 포화하지 않은 것으로 호도될 수 있는 픽셀의 수는 전체의 0.26% 정도이다. 한편 약 99.74%의 픽셀은 포화값이 41,000 ADU보다 크므로 이들은 아직 포화하지 않았는데도 포화한 것으로 판단될 수 있다. 그러나 SOC를 활용한 연구는 어두운 퍼진 천체를 대상으로 하므로 포화값을 실제보다 작게 정해도 문제가 없을 것이다. 다만 밝기 검보정을 할 경우, 이 포화값보다 어두운 점광원을 활용하도록 주의하여야 한다.

검출기의 비선형성을 조사하는 데에는 전체 픽셀에서 1) 배드 픽셀(bad pixel), 2) 127번째와 255번째 열에 있는 픽셀들, 그리고 3) 127번째와 255번째 열에 있는 픽셀들을 제외한 모든 픽셀을 사용하였다. 그림 4에서 볼 수 있듯이 픽셀의 반응성은 FRAMERST가 10일 때까지 선형적이다. 따라서 각 픽셀 (i, j)에 대하여 FRAMERST의 값 x에 따른 측정값 y_{ij}의 변화를 선형 함수 $a_{ij}x + b_{ij}$로 맞추고 이를 기대되는 측정값 $Y_{ij}(x)$라 한다. 그림 4는 한 예로 픽셀 (224, 23)에 대하여 맞춘 결과이다. 여기에서 점은 노출 계량에 따른 측정값의 변화를 나타나는데, 파란점이 맞춤에 이용된 것들이다. 빨간 실선은 맞춤 결과이다.

그림 5는 분석에 사용한 모든 픽셀에 대하여 측정값 y_{ij}와 기대되는 측정값 Y_{ij}의 비를 측정값의 함수로 나타낸 것이다. 여기에서 포화값인 41,000 ADU보다 큰 값을 갖는 점들을 제외하고 나머지를 이용하여 4차 다항식 $P(y) = Ay^4 + By^3 + Cy^2 + Dy + E$로 맞추었다. 그 결과 얻은 다항식 계수의 값은 아래와 같고, 이 계수들을 이용하여 $P(y)$을 그림 5에 보라색 선으로 나타내었다.

$$
\begin{aligned}
A &= -5.77388971 \times 10^{-19},\\
B &= \ \ 7.00289087 \times 10^{-14},\\
C &= -3.36973917 \times 10^{-9},\\
D &= \ \ 7.14695134 \times 10^{-5},\\
E &= \ \ 0.453013462
\end{aligned}
$$

측정값을 이 다항함수로 나누면 검출기의 비선형성을 보정할 수 있다. 그림 6은 비선형성 보정 후, 측정 픽셀값의 기대 픽셀값에 대한 편차가 얼마나 개선되었는지 보여준다. 픽셀값이 포화값보다 작은 모든 범위에서 픽셀값의 2% 내에서 비선형성이 보정되는 것을 확인할 수 있다.

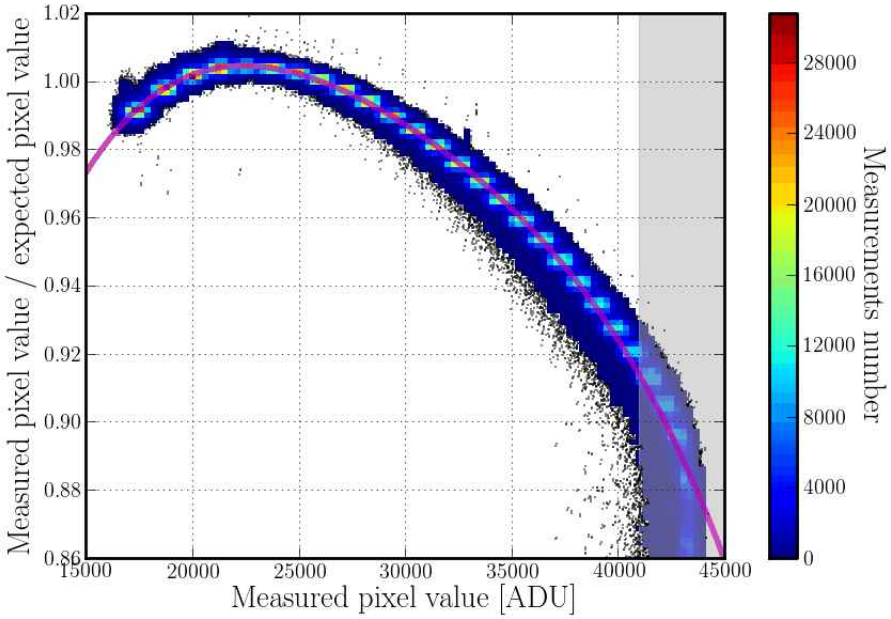
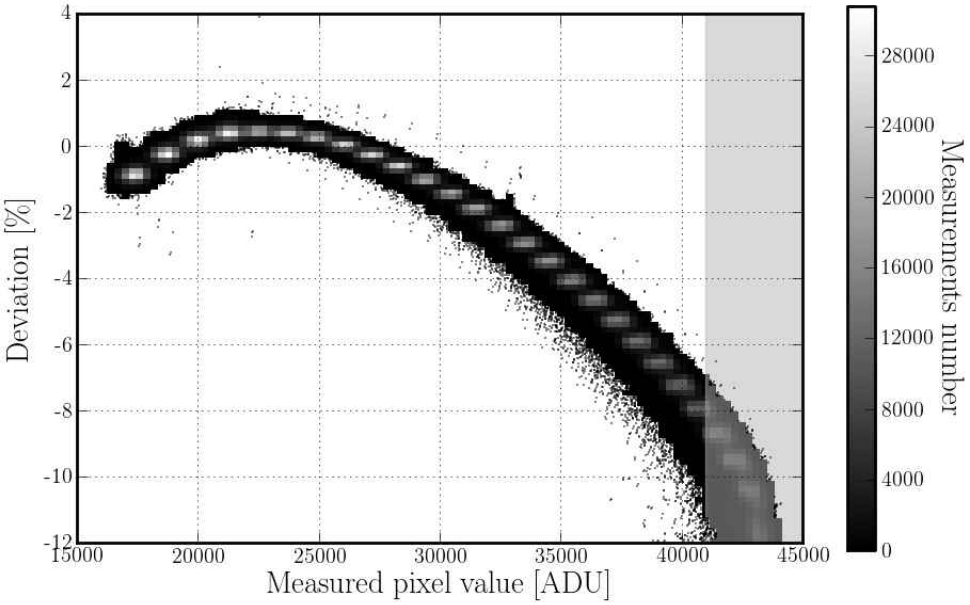
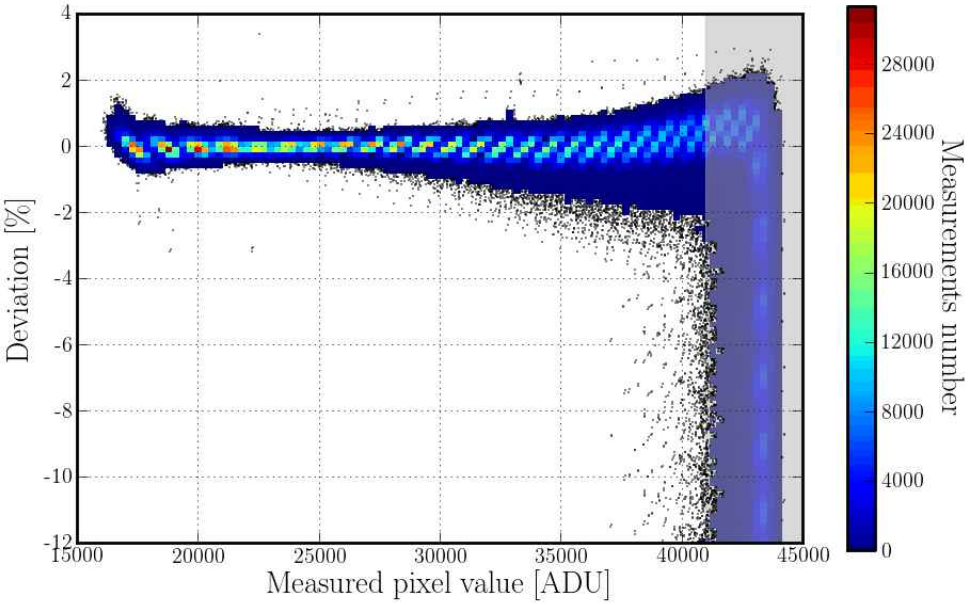


그림 5. 모든 픽셀에 대한 실제 측정값과 기대되는 측정값의 비. 이 그림에서 자료값을 모두 점으로 나타내는 대신 점의 밀도를 색으로 나타내었다. 밀도를 계산하는 데 사용한 격자의 크기는 가로 300 ADU, 세로 0.0016이다. 회색으로 칠해진 부분은 측정값이 포화값보다 큰 영역을 나타낸다.

비선형성을 보정하는 기능은 socdr.mainproc.LinearityCorrection() 클래스에 구현되어 있다. 이 클래스는 초기화되었을 때 파이프라인 설정에서 초과값과 보정 다항함수의 계수를 읽어 들인다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 영상 HDU의 영상 자료에서 초과값을 넘는 값을 가진 픽셀들에 NumPy의 NaN 값을 지정하고, 그 외의 픽셀들에서는 보정 다항함수를 나눈다. 그 후, 수정된 HDUList를 넘겨준다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 LinearityCorrection() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 LINCORR 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 LinearityCorrection() 클래스가 초기화될 때, 그리고 불러질 때 준 인자들이 기록된다.



(가) 비선형성 보정 전.



(나) 비선형성 보정 후.

그림 6. 비선형성 보정 전(가)과 후(나)의 측정 픽셀값의 기대 픽셀값에 대한 편차.

3.5. 영상 차분

SOC의 검출기는 전자를 쌓아가며 일정 시간 간격(노출 시간)으로 값을 읽고, 쌓인 전자는 일정한 영상 간격으로 리셋 전압을 걸어 제거하게 된다. 따라서 리셋과 리셋 사이의 영상들 사이에 차이를 구해야 지정한 노출 시간 동안의 밝기를 얻을 수 있다.

영상을 차분하는 기능은 socdr.mainproc.Differentiation() 클래스에 구현되어 있다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 각 영상 HDU의 영상 자료에서 그 전 영상을 뺀다. 하지만 가) 전 영상이 리셋 영상인 경우, 나) 지상 전송 시의 문제 등으로 전 영상이 같은 리셋 시퀀스에 있지 않은 경우(즉, 두 영상에서 헤더의 RESETCNT 값을 조사하여 그 값이 서로 다른 경우), 다) 지상 전송 시의 문제 등으로 자료 상의 전 영상이 실제로는 전 영상이 아니고 그 전의 영상인 경우(즉, 두 영상에서 헤더의 FRAMERST 값을 조사하여 그 값의 차가 1이 아닌 경우)에는 그 영상 HDU에서 자료를 제거하고 헤더만 남겨둔다. 그 후, 수정된 HDUList를 넘겨준다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 Differentiation() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 DIFFERN 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 Differentiation() 클래스가 초기화될 때, 그리고 불러질 때 준 인자들이 기록된다.

3.6. 영상 평탄화

검출기의 픽셀에 따른 반응성 보정을 위하여 적분구로 촬영한 영상 자료를 활용하였다. 이 자료를 위에서 설명한 배드(bad) 픽셀 가리기, 비선형성 보정, 그리고 영상 차분 과정을 거쳐 처리한 후, 이렇게 얻어진 영상들을 밴드 별로 합성하였다. 영상을 합성할 때에는 중간값을 취하였다. 합성 후에는 영상을 평균값으로 나누어 정규화하였다. 정규화할 때 너무 크거나 너무 작은 값을 배제하기 위하여 평균에 비하여 표준편차의 5배보다 크거나 작은 값은 배제하였다. 이 과정은 더 이상 배제할 값이 없어질 때까지 반복하였고, 이렇게 배제된 값들은 영상에서 NaN 값으로 대체하였다. 최종적으로 얻은 바닥 고르기용 자료의 영상은 그림 7과 같다.

여기에서 I밴드의 경우에는 균일한 광원이 입사하지 않은 것으로 보인다. 따라서 H밴드와 같은 자료로 평탄화할 예정이다. 각 밴드별 평탄화 영상자료는 FITS 형식으

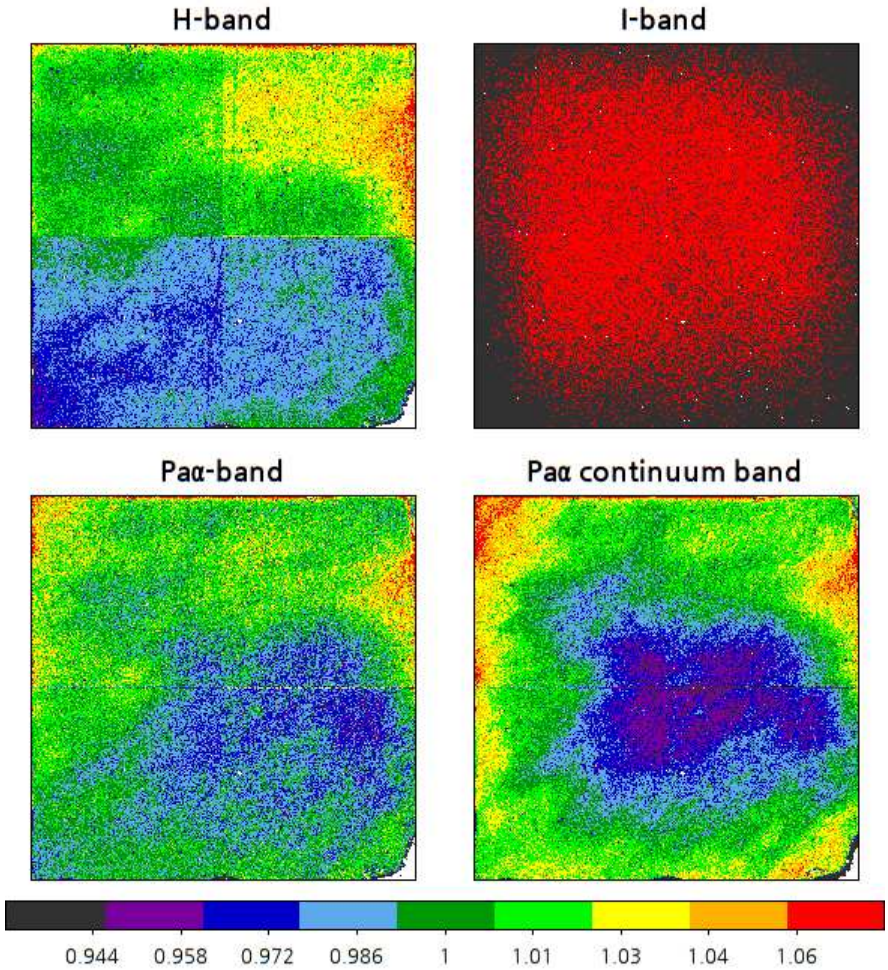


그림 7. 평탄도 보정을 위한 영상.

로 저장되어 있으며, 이들 파일의 경로는 파이프라인 설정파일에 정의되어 있다(부록 1 참조).

영상을 평탄화하는 기능은 socdr.mainproc.FlatFieldCorrection() 클래스에 구현되어 있다. 이 클래스는 초기화되었을 때 파이프라인 설정에 저장된 평탄화 영상의 경로에서 영상들을 읽어 들인다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 각 영상 HDU의 영상 자료에서 평탄도 보정 영상을 나누고 수정된 HDUList를 넘겨준다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 FlatFieldCorrection() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 FLATCORR 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 FlatFieldCorrection() 클래스가 초기화될 때, 그리

고 불러질 때 준 인자들이 기록된다.

3.7. 영상 왜곡 보정

기기에 의하여 발생하는 각종 광학 왜곡을 보정한다. 광학 왜곡을 정의하는 데에는 SIP(Shupe, D. L., Moshir, M., Li, J., et al., 2005)를 따랐으며, 보정 알고리즘은 IRAF의 geotran 태스크를 따랐다.

광학 왜곡을 보정하는 변환은 영상의 기준 픽셀을 중심으로 한 픽셀 상대 좌표의 함수로 나타낼 수 있는데, 대개 다항함수를 이용한다. 따라서 다항함수의 계수를 이용하여 광학 왜곡을 정의할 수 있다. 이 계수들은 파이프라인 설정 파일의 distcorr 섹션에 저장한다. 계수들을 저장하는 데에는 SIP의 컨벤션을 따른다. SIP 컨벤션은 영상 왜곡의 보정식을 FITS 영상 헤더에 저장하기 위해 개발되었지만, SOC의 자료처리 파이프라인에서는 이를 설정 파일에 저장하기 위해서 사용하고 FITS 파일의 헤더에는

```
[distcorr]
A_ORDER = 2
A_0_2 = 1.569E-05
A_1_1 = 5.232E-05
A_2_0 = 3.31E-05
B_ORDER = 2
B_0_2 = 4.172E-05
B_1_1 = 2.213E-05
B_2_0 = -9.819E-07
AP_ORDER = 2
AP_0_1 = 5.677E-05
AP_0_2 = -1.569E-05
AP_1_0 = 5.871E-05
AP_1_1 = -5.231E-05
AP_2_0 = -3.309E-05
BP_ORDER = 2
BP_0_1 = 4.432E-05
BP_0_2 = -4.172E-05
BP_1_0 = 2.091E-05
BP_1_1 = -2.213E-05
BP_2_0 = 9.814E-07
CRPIX = 128., 128.
```

그림 8. 광학 왜곡을 정의하는 파이프라인 설정의 예.

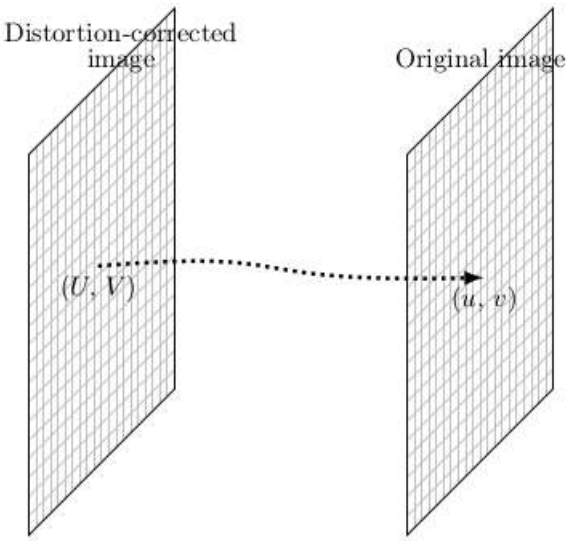


그림 9. 광학 왜곡 보정을 위한 픽셀 매핑.

저장하지 않는다.

그림 8은 Shupe, D. L., Moshir, M., Li, J., et al. (2005)의 Spitzer IRAC 채널4 예제를 파이프라인 설정 파일의 형식으로 쓴 예이다. 위에서 A_나 B_로 시작하는 파라미터들은 영상의 픽셀 좌표를 천구 상의 가상의 픽셀 좌표로 변환(왜곡 보정; 순방향 변환)하는데 사용되고 AP_나 BP_로 시작하는 파라미터들은 그 반대 방향의 변환(왜곡 적용; 역방향 변환)에 사용된다. A_나 AP_로 시작하는 파라미터들은 가로 방향 좌표에 대한 것들이고 B_나 BP_로 시작하는 파라미터들은 세로 방향 좌표에 대한 것들이다. CRPIX는 다항식의 변수를 정의할 때 사용한 기준 픽셀의 좌표로, 이 문서에서 사용된 것과 다르게 FITS 표준을 따르는 좌표이다.

파이프라인에서 광학 왜곡을 보정하는 데에는 역방향 변환만 사용된다. 따라서 광학 왜곡 보정을 위해서는 AP_나 BP_로 시작하는 파라미터들을 정의해야 한다. 만약 네 개의 _ORDER로 끝나는 파라미터 중 0으로 설정되었거나 누락된 경우에는 그에 해당하는 왜곡은 없는 것으로 판단한다. 현재 버전에서는 넷 모두 0으로 되어있어 광학 왜곡이 전혀 없다. 한편, 왜곡을 정의하는 데 사용된 기준 픽셀의 좌표를 나타내는 CRPIX는 광학 왜곡이 없더라도 반드시 정의해야 한다.

광학 왜곡이 보정된 영상의 픽셀 (U, V)의 값은 아래와 같이 구한다.

1. 광학 왜곡을 보정하는 변환의 역변환을 이용하여 픽셀 좌표 (U, V)를 원본 영상의 픽셀 (u, v)로 변환한다(그림 9 참조).
2. 원본 영상에서 픽셀 (u, v)의 값을 얻는다. 이때 (u, v)가 정수가 아닐 수 있으므로 2차원 선형 내삽(bilinear interpolation)을 이용하여 값을 얻는다.

3. 이 값을 광학 왜곡이 보정된 영상의 픽셀 (U, V)에 지정한다.
4. 변환에 따른 픽셀 면적의 변화를 보정하기 위해 야코비안(Jacobian)을 곱한다.

위 과정은 IRAF의 geotran 태스크에서 interpolant 옵션을 linear로 지정한 경우의 알고리즘을 따른 것이다.

영상의 광학 왜곡을 보정하는 기능은 socdr.mainproc.DistortionCorrection() 클래스에 구현되어 있다. 이 클래스는 초기화되었을 때 파이프라인 설정에 저장된 SIP 계수를 이용하여 보정에 필요한 계산을 수행한다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 각 영상 HDU의 영상 자료의 왜곡을 보정하고 수정된 HDUList를 넘겨준다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 DistortionCorrection() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 DISTCORR 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 DistortionCorrection() 클래스가 초기화될 때, 그리고 불러질 때 준 인자들이 기록된다.

3.8. WCS 정보 수정

SOC 영상에는 기본적으로 위성체의 궤도와 자세에서 계산한 적도 좌표계의 WCS 정보가 들어있다. 하지만 영상을 합성하기 위해서는 정밀한 WCS 정보가 필요하다. WCS 정보를 수정하는 방법은 CASU에 적용된 방법을 이용하였다.

우선, 주어진 FITS 파일의 영상이 차지하는 영역에 대하여 VizieR 카탈로그 서비스를 제공하는 서버에 질의하여 이 영역에 존재하는 별의 목록을 2MASS 점광원 전천 카탈로그(2MASS All-Sky Catalog of Point Sources)에서 가져온다. 기본적으로는 CDS(Centre de Données astronomiques de Strasbourg) 서버 (http://webviz.u-strasbg.fr/)에 접속한다. 이때 SOC 영상의 시야는 매우 넓기 때문에 목록에 존재하는 모든 별의 목록을 가져오면 크기가 너무 크기 때문에 H 등급이 10등급보다 밝은 것들만 가져온다.

그 다음에는 각 영상에 대하여 SExtractor를 실행하여 영상에 존재하는 광원의 목록을 작성한다. 2.2.8절에서 설명했듯이 SExtractor의 실행 파일은 sextractor라는 이름으로 사용자의 실행파일 검색 경로에 존재해야 한다. SExtractor를 실행할 때 영상 파일 등을 /tmp/ 폴더에 임시로 저장한다. 따라서 /tmp/ 폴더에 1 MB 정도의 빈 공

간이 있어야 한다. SExtractor를 실행하는데 필요한 설정 파일들의 경로는 파이프라인 설정 파일의 seconf 섹션에 정의되어 있다.

온라인에서 받은 2MASS의 카탈로그와 영상의 카탈로그를 비교하여 두 카탈로그의 광원을 매치한다. 이를 통하여 영상의 픽셀 좌표와 천구 상의 좌표를 비교할 수 있다. 두 좌표를 비교하여 얻은 WCS 정보를 FITS 헤더에 기록한다. 이때 업데이트되는 FITS 헤더 항목은 CRVAL1, CRVAL2, CRPIX1, CRPIX2, CD1_1, CD1_2, CD2_1, CD2_2이다. 업데이트 전의 원래 값은 각각 O_CRVAL1, O_CRVAL2, O_CRPIX1, O_CRPIX2, O_CD1_1, O_CD1_2, O_CD2_1, O_CD2_2에 저장된다. 이외에 WCS 보정의 오차는 STDCRMS에, WCS 보정에 이용된 광원의 개수는 NUMBRMS에 저장된다.

WCS 정보를 수정하는 기능은 socdr.postproc.WCSCorrection() 클래스에 구현되어 있다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르면, 여기에 들어있는 각 영상 HDU의 헤더에 수정된 WCS 정보를 기록한다. 영상 자료는 수정하지 않는다. HDUList 오브젝트를 인자로 준 경우, 메모리 효율성을 위하여 인자로 주어진 오브젝트를 바로 수정한다. 만일 인자를 수정하고 싶지 않은 경우에는 WCSCorrection() 클래스를 초기화할 때 copy 키워드 인자에 True 값을 준다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더에는 WCSCORR 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 WCSCorrection() 클래스가 초기화될 때, 그리고 불려질 때 준 인자들이 기록된다.

3.9. 영상 합성

한 관측 동안 촬영한 영상들을 합성한다. 영상을 합성하는 데는 외부 프로그램인 Montage를 이용한다. 따라서 Montage의 실행 파일들이 사용자의 실행파일 검색 경로에 존재해야 한다. 한편, Montage를 실행하는 동안 다량의 임시 파일들을 /tmp/ 디렉터리에 생성하므로 이 디렉터리에 최소한 700 MB 이상의 빈 공간을 확보해야 한다. 합성 후에는 HDUList 오브젝트를 넘겨주는데, 여기에는 중간값으로 합성한 결과(확장이름 MEDIAN), 평균으로 합성한 결과(확장이름 MEAN), 그리고 합성 영상의 각 픽셀에서 대푯값을 얻는데 이용된 값의 개수(확장이름 NUMPNTS)가 영상 HDU로 저장되어 있다. 이 영상의 크기는 주어진 FITS 파일 또는 HDUList에 저장된 영상들 모두를 커버할 만큼 크다.

영상을 합성하는 기능은 socdr.postproc.StackFrames() 클래스에 구현되어 있다. 이 클래스에 인자로 SOC의 영상 FITS 파일의 경로나 HDUList 오브젝트를 주어 부르

면, 여기에 들어있는 영상들을 합성한다. HDUList 오브젝트를 인자로 준 경우, 이 클래스는 주어진 인자를 수정하지 않는다. 따라서 초기화할 때 copy 키워드 인자를 받지 않는다. 대신 mpi 키워드 인자를 받는데, 이 인자에 True 값을 주면 영상을 합성할 때 MPI 버전의 Montage 실행 파일들을 이용한다. 이 클래스가 넘겨주는 HDUList의 주 HDU 헤더는 STACKFRM 항목이 추가되고 이 항목에는 'T' 값이 지정된다. 그리고 HISTORY 항목에는 그 HDUList가 처리된 시간과 StackFrames() 클래스가 초기화될 때, 그리고 불러질 때 준 인자들이 기록된다.

4. 사용 방법

SOCdr은 파이썬 프롬프트에서 인터랙티브하게 사용할 수도 있고 스크립트를 만들어 사용할 수도 있다. 하지만 여기에서는 파이썬 프롬프트에서 사용하는 것을 가정하고 사용 방법을 설명한다.

SOCdr을 사용하려면 우선 socdr 모듈을 가져와야 한다.

```
>>> import socdr
```

그러고 나서 목적에 맞게 MainProcessing() 클래스와 PostProcessing() 클래스를 초기화한다.

```
>>> mainproc = socdr.MainProcessing(flat=True, distcorr=True)
>>> postproc = socdr.PostProcessing(stack=True, mpi=False)
```

MainProcessing() 클래스를 초기화할 때에는 flat과 distcorr 키워드 인자를 줄 수 있다. 이 둘의 기본값은 True이다. flat 인자에 False 값을 주면 영상 평탄화 처리를, distcorr 인자에 False 값을 주면 영상 왜곡 보정 처리를 하지 않는다. PostProcessing() 클래스를 초기화할 때에는 stack과 mpi 키워드 인자를 줄 수 있다. stack 인자의 기본값은 True, mpi 인자의 기본값은 False이다. stack 인자에 False 값을 주면 영상을 합성하지 않는다. mpi 인자에 True 값을 주면 영상을 합성할 때 MPI 버전의 Montage 프로그램을 이용한다(3.9절 참고).

SOC의 관측 자료는 getfits() 함수를 이용하여 가져올 수 있다.

```
>>> socfitsfile = socdr.getfits(query)
```

위 명령은 MIRIS 관측 자료의 데이터베이스가 구축되어 있는 컴퓨터에서만 의미 있다. 만일 이미 FITS 형식으로 변환된 SOC 관측 자료를 가지고 있다면 다음 단계로 넘어간다. 위 명령에서 query는 그대로 search_db() 함수의 인자로 넘겨지는데, 찾고자 하는 자료에 대한 질의문이다(3.1절 참고). 만일 이 질의에 해당하는 관측 자료가 없으면 socfitsfile에는 None이 넘어오고, 하나만 존재하면 하나의 HDUList 오브젝트만 넘어오고, 여럿 존재하면 HDUList 오브젝트의 리스트가 넘어온다. 따라서 getfits() 함수를 실행한 후에는 결과값의 형식을 조사해야 한다.

만일 socfitsfile이 하나의 HDUList라면 아래와 같이 주처리 및 후처리한 결과를 얻을 수 있다.

```
>>> socfitsfile1 = mainproc(socfitsfile)
>>> socfitsfile2 = postproc(socfitsfile1)
```

위와 같이 실행할 때 주의할 점은, 실행 후 socfitsfile과 socfitsfile1은 사실상 같은 오브젝트이고, 주처리된 결과가 아니라 WCS 수정까지 이루어진 결과라는 점이다. 만일 PostProcessing() 클래스를 초기화할 때 stack 인자에 False 값을 주었다면 socfitsfile2까지도 같은 오브젝트가 된다. 이는 3장에서 여러 번 설명하였듯이 각 처리 과정을 거치면서 메모리 효율성을 위하여 인자로 주어진 오브젝트를 수정하기 때문이다. 만일 이런 상황을 피하고 싶다면 아래와 같이 mainproc과 postproc을 부를 때 copy 키워드 인자에 True 값을 주면 된다.

```
>>> socfitsfile1 = mainproc(socfitsfile, copy=True)
>>> socfitsfile2 = postproc(socfitsfile1, copy=True)
```

copy 키워드 인자 외에 writeto 키워드 인자와 clobber 키워드 인자를 줄 수 있다. writeto 인자에 파일 경로를 주면 이 파일에 처리 결과를 저장한다. 기본값으로는 None이 지정되어 있으며 파일에 저장하지 않는다. clobber 인자는 기본값으로 False 값이 지정되어 있으며, 만일 writeto 인자의 파일 경로에 이미 파일이 존재하면 파일을 덮어쓰지 않는다. 이 인자에 True 값을 주면 파일을 덮어쓴다. writeto 인자를 통하지 않고 명령 실행 결과로 얻은 HDUList를 FITS 파일로 저장하려면 아래와 같이 실행한다.

```
>>> socfitsfile2.writeto(fitspath)
```

이 경우에도 기본적으로는 파일이 이미 존재하면 파일을 덮어쓰지 않고 IOError 예외를 발생시킨다. 만일 덮어쓰려면 위에서 설명한 바와 같이 clobber 키워드 인자에 True 값을 주어 실행한다.

한편, 위에 설명하였듯이 getfits() 함수의 실행 결과는 질의에 맞는 관측 자료의 개수에 따라 달라지기 때문에 결과값의 형식을 확인할 필요가 있다. 이것은 아래와 같이 구현할 수 있다.

```
>>> from astropy.io import fits
>>> socfitsfile = socdr.getfits(query)
>>> if socfitsfile == None:
>>>     pass
>>> elif isinstance(socfitsfile, fits.HDUList):
>>>     socfitsfile = mainproc(socfitsfile)
>>>     socfitsfile = postproc(socfitsfile)
>>> elif isinstance(socfitsfile, list):
>>>     socfitsfile = [mainproc(x) for x in socfitsfile]
>>>     socfitsfile = [postproc(x) for x in socfitsfile]
```

5. 시험 결과 및 결론

파이프라인의 시험은 2MASS의 전천 탐사 영상 중 황북극을 중심으로 하는 4.2° × 4.2° 영역 내의 영상들을 받아 합성하여 만든 모의 자료를 이용하여 수행하였다. 다만, 전처리 부분을 시험하는데 필요한 이진 자료를 작성하기는 어렵기 때문에 FITS 형식의 자료로 시험할 수 있는 주처리와 후처리 부분만 시험하였다. 모의 자료는 1초 노출로 10분 간 관측하고 10 프레임마다 한 번씩 검출기를 리셋하는 경우를 가정하여 작성하였다. 여기에 실제 관측 전 후에 암전류 측정을 위한 관측을 1분 동안씩 수행한 것으로 가정하였다. 이 모의 자료에는 총 721개의 영상이 들어 있다.

이 자료를 이용하여 파이프라인을 시험하기 위한 스크립트는 아래와 같다.

```
import time
import socdr

mainproc = socdr.MainProcessing()
postproc = socdr.PostProcessing(mpi=False)

t0 = time.time()
simim = mainproc("SOC_NEP01_H.fits")
simim = postproc(simim)
t1 = time.time()
print "Processing took {0} sec w/ mpi=False.".format(t1 - t0)

postproc.mpi = True
t0 = time.time()
simim = mainproc("SOC_NEP01_H.fits")
simim = postproc(simim)
t1 = time.time()
print "Processing took {0} sec w/ mpi=True.".format(t1 - t0)
```

실행 결과는 Intel® Core™ i7-950 (8 MB 캐쉬, 3.06 GHz, 4 코어, 8 쓰레드) 프로세서 기준, 아래와 같다.

```
Processing took 533.657499075 sec w/ mpi=False.
Processing took 271.712661982 sec w/ mpi=True.
```

한편, AMD Opteron™ 2218 (1 MB 캐쉬, 2.6 GHz, 2 코어, 4 쓰레드) 프로세서의 경우에는 아래와 같다.

```
Processing took 1026.58064294 sec w/ mpi=False.
Processing took 484.372719049 sec w/ mpi=True.
```

실행 시간의 대부분은 후처리에 걸리는 시간이며 주처리는 수 초 내지 수십 초 내에 실행된다.

SOCdr은 파이썬을 이용하여 작성되었으며 비교적 짧은 개발 기간 동안 개발 완료되었다. 또한 직접 구현하기 쉽지 않은 기능을 외부 프로그램인 SExtractor 및 Montage 등과, 최근 활발하게 개발되고 있는 천문학용 파이썬 라이브러리인 Astropy에 맡김으로써 개발의 효율성과 동시에 신뢰성을 향상시킬 수 있었다. 덧붙여 파이썬의 객체 지향성을 적극 활용하여 각 기능을 독립적인 모듈로 구현하였다. 이를 통하여 사용자에게는 사용상의 편의성을 제공하고 개발자에게는 개발 및 유지의 편리성을 제공한다.

참고문헌

1. 김일중, 표정현, 박원기 등, 2013, MIRIS SOC 데이터 포맷팅 소프트웨어 개발, 기술노트 No. 13-###-###, 한국천문연구원

2. Shupe, D. L., Moshir, M., Li, J., et al., 2005, The SIP Convention for Representing Distortion in FITS Image Headers, Astronomical Data Analysis Software and Systems XIV (P. L. Shopbell, M. C. Britton, & R. Ebert eds) vol. 30 of ASP Conference Series

부록

1. 파이프라인 설정 파일

파이프라인의 각종 설정은 ConfigObj 모듈이 읽을 수 있는 형식으로 socdr.cfg 파일에 저장되어 있다. 파이프라인 0.1.1 버전에 포함된 socdr.cfg 파일은 아래와 같다.

```
[flat]
H = data/flat/flat_H-1.0.1.fits
# Currently, I-band shares flat-field with H-band.
I = data/flat/flat_H-1.0.1.fits
PAAL = data/flat/flat_PaA-1.0.1.fits
PAAC = data/flat/flat_PaAcnt-1.0.1.fits

[badpixels]
BadPixelsImg = data/badpixels/badpixels-2.1.fits

[lincorr]
SaturationValue = 41000
PolyCoeffs = -5.77388971e-19, 7.00289087e-14, -3.36973917e-09, 7.14695134e-05,
4.53013462e-01

[distcorr]
# SIP convention distortion coefficients
# To define distortion, AP_* and BP_* coefficients are mandatory.
A_ORDER = 0
B_ORDER = 0
AP_ORDER = 0
BP_ORDER = 0
# FITS standard (1-based) pixel coordinates of reference pixel
CRPIX = 128., 128.

[seconf]
# WARNING: DO NOT CHANGE THIS SECTION IF YOU DO NOT KNOW WHAT YOU ARE
DOING
SEConfFile = data/seconf/soc_phot.sex
PARAMETERS_NAME = data/seconf/soc_phot.param
FILTER_NAME = data/seconf/soc_phot.conv
STARNNW_NAME = data/seconf/soc_phot.nnw
```

이 파일은 socdr 모듈을 가져올(import) 때 읽히며 socdr.SOCDRCFG를 통하여 접근

할 수 있다. 이 오브젝트는 configobj.ConfigObj 형식으로, 파이썬의 사전과 같은 방식으로 값을 가져올 수 있다. 예를 들어 badpixels 섹션 BadPixelsImg 항목의 값을 가져오려면 socdr.SOCDRCFG['badpixels']['BadPixelsImg']의 값을 확인하면 된다. 설정 항목 중 일부는 파일의 경로를 정의하는데, 이 경로는 SOCdr이 설치된 디렉터리에 대한 상대 경로이다. 이런 경우, 파일에 접근할 때에는 pkg_resources 모듈을 이용하면 편리하다. 예를 들어 배드 픽셀의 위치가 정의되어 있는 FITS 파일을 읽는 경우, 아래와 같이 수행할 수 있다.

```
>>> # import modules
>>> import socdr
>>> from pkg_resources import resource_stream
>>> from astropy.io import fits
>>>
>>> # load badpixel image
>>> badpixs = fits.getdata(resource_stream('socdr',
...                                        socdr.SOCDRCFG['badpixels']['BadPixelsImg']))
```

위에서 resource_stream은 해당 파일을 열어 파이썬 file 오브젝트로 넘겨준다. 만일 전체 파일 경로를 알고 싶다면 resource_filename을 사용한다.

설정 파일의 각 항목의 의미는 아래와 같다.

1. flat 섹션
   가. 이 섹션의 각 항목은 그 항목의 이름에 해당하는 밴드(H는 H 밴드, I는 I 밴드, PAAL은 Paα선 밴드, PAAC는 Paα선 연속선 밴드)의 영상 평탄화 템플레이트의 FITS 파일 경로이다.
2. badpixels 섹션
   가. BadPixelsImg: 배드 픽셀의 위치를 정의한 FITS 파일의 경로.
3. lincorr 섹션
   가. SaturationValue: ADU 단위의 초과값.
   나. PolyCoeffs: 비선형성 보정을 위한 다항식의 계수. 높은 차수항의 계수부터 낮은 차수항의 계수 순으로 나열한다.
4. distcorr 섹션: 3.7절 참고.
5. seconf 섹션
   가. SEConfFile: SExtractor 실행을 위한 설정 파일의 경로.
   나. PARAMETERS_NAME, FILTER_NAME, STARNNW_NAME: SExtractor 실행을 위한 설정. SExtractor의 매뉴얼 참조.

2. 설치 디렉터리 구조 및 파일

0.1.1 버전 기준, 파이프라인 소프트웨어의 설치 디렉터리 구조 및 파일은 다음과 같다.

socdr/	
├── __init__.py	모듈 정의 및 초기화
├── data/	자료 디렉터리
│ ├── README.rst	빈 파일(placeholder)
│ ├── badpixels/	
│ │ └── badpixels-2.1.fits	배드 픽셀 위치 영상 파일
│ ├── flat/	
│ │ ├── flat_H-1.0.1.fits	H 밴드 평탄화 템플레이트
│ │ ├── flat_PaA-1.0.1.fits	Paα 밴드 평탄화 템플레이트
│ │ └── flat_PaAcnt-1.0.1.fits	Paα 연속선 밴드 평탄화 템플레이트
│ └── seconf/	SExtractor 설정 파일 디렉터리
│ ├── soc_phot.conv	
│ ├── soc_phot.nnw	
│ ├── soc_phot.param	
│ └── soc_phot.sex	
├── mainproc.py	주처리 모듈
├── postproc.py	후처리 모듈
├── preproc/	전처리 모듈
│ ├── __init__.py	전처리 모듈 정의 및 초기화
│ ├── raw2fits.py	Raw 자료 FITS 형식 변환
│ └── searchdb.py	데이터베이스 질의
├── socdr.cfg	설정 파일
└── utils.py	유틸리티 모듈