Astro-**W**ise **E**nvironment

Software implementation of the photometric system

January 3, 2007

Contents

Chapter 1

The class model of the photometric pipeline

All the classes in the photometric sub-system ultimately derive from a set of base classes used throughout the Astro-Wise system. These classes are `DBObject`, `DataObject`, `BaseCatalog`, `Catalog`, and the mixin class `ProcessTarget`. In this chapter, the classes which are specific for the photometric system are discussed. The core classes that have been defined in the photometric system are given in Table 1.1. A high-level design diagram of the photometric system is shown in Fig. 1.1.

1.1 Class descriptions

1.1.1 PhotRefCatalog

One of the basic ingredients in the photometric pipeline is a catalog of reference stars. This catalog is represented by the `PhotRefCatalog` class. The most important public functionality of this class consists of an accessor for obtaining a dictionary of magnitudes for a selected photometric band (the *get_dict_of_magnitudes* functionality).

Associated class : PhotRefSource

The `PhotRefSource` class represents a single entry in a catalog of reference stars. The class is, therefore, closely associated with the `PhotRefCatalog` class where it can have any multiplicity. The attribute list of this class comprises, amongst others, the coordinates of the source and a list of magnitudes from various photometric systems (Johnson-Morgan-Cousins, Sloan). The class has a *get_magnitude* functionality that is only used internally by the `PhotRefCatalog` object in which the sources are stored. The `PhotRefSource` class mainly serves as a repository of knowledge about the content of a single source; it is through static methods defined on the `PhotRefSource` class that its validity can be checked by the system.

Associated classes : SourceFilterOrigin/PhotRefCatalogReader

Any `PhotRefCatalog` object has a reference to a `SourceFilterOrigin` object that can be used to restrict the view on the contents of the standard star catalog. Activating this filter ensures that only certain parts of the standard star catalog can be used by the system.

Underlying the `PhotRefCatalog` object is an LDAC file. This file is read by a dedicated `PhotRefCatalogReader` object associated with the standard star catalog. All method calls to

Table 1.1: The list of core classes defined in the photometric system.

Class name	Descripton
`PhotRefCatalog`	Represents a catalog of standard stars.
`PhotSrcCatalog`	Represents a catalog of *measured* standard stars.
`PhotTransformation`	Represents a table of color transformation coefficients.
`PhotExtinctionCurve`	Represents a standard extinction curve.
`PhotSkyBrightness`	Represents a table of the brightness of the sky as a function of lunar phase.
`PhotometricExtinctionReport`	Represents the results of monitoring the atmospheric stability throughout the night.
`PhotometricSkyReport`	Represents the results of deriving the sky spectrum.
`PhotometricParameters`	Represents the final output of the photometric pipeline containing the zeropoint and the extinction.
BaseAtmosphericExtinction	Represents the atmospheric extinction.
`IlluminationCorrection`	Represents the fit to the illumination correction pattern.
`IlluminationCorrectionFrame`	Represents the frame used in applying the illumination correction during processing.

the `PhotRefCatalog` object are routed through this reader. It is through this reader that any restrictions imposed by active filters on the standard star catalog are enforced.

1.1.2 PhotSrcCatalog

The `PhotSrcCatalog` class is the real 'working' class in the photometric system, because objects of this class are used throughout. A `PhotSrcCatalog` object is defined as the association between a source catalog derived from a `ReducedScienceFrame` object and a catalog of reference stars. Within the design of the photometric system, the `PhotSrcCatalog` class has the status of an (important) implementation class.

Two of the most important public functionalities of the `PhotSrcCatalog` class are the *make* and the *get_list_of_raw_zeropoints* functionalities. The *make* functionality of this class has three main dependencies: a `ReducedScienceFrame` object (attribute name: `frame`), an `AstrometricParameters` object (attribute name: `astrom_params`), and a `PhotRefCatalog` object (attribute name: `refcat`). An additional dependency, when needed, consists of a `PhotTransformation` object (attribute name: `transform`).

The *get_list_of_raw_zeropoints* functionality returns a list of all the raw zeropoints contained in the `PhotSrcCatalog` object. Every 2-tuple in this list contains the raw zeropoint from one of the sources in the catalog, and its associated error.

Associated class : PhotSrcSource

The `PhotSrcSource` class represents a single entry in a source catalog and is, therefore, closely associated with the `PhotSrcCatalog` class where it can have any multiplicity.

The class has several attributes. The most important ones are an `index` attribute for cross-referencing with a standard star catalog, the magnitude M of the star for a given photometric band with its error, and the measured instrumental magnitude m' of the star with its error. The *get_raw_zeropoint* functionality of the class returns the difference of the magnitude M and

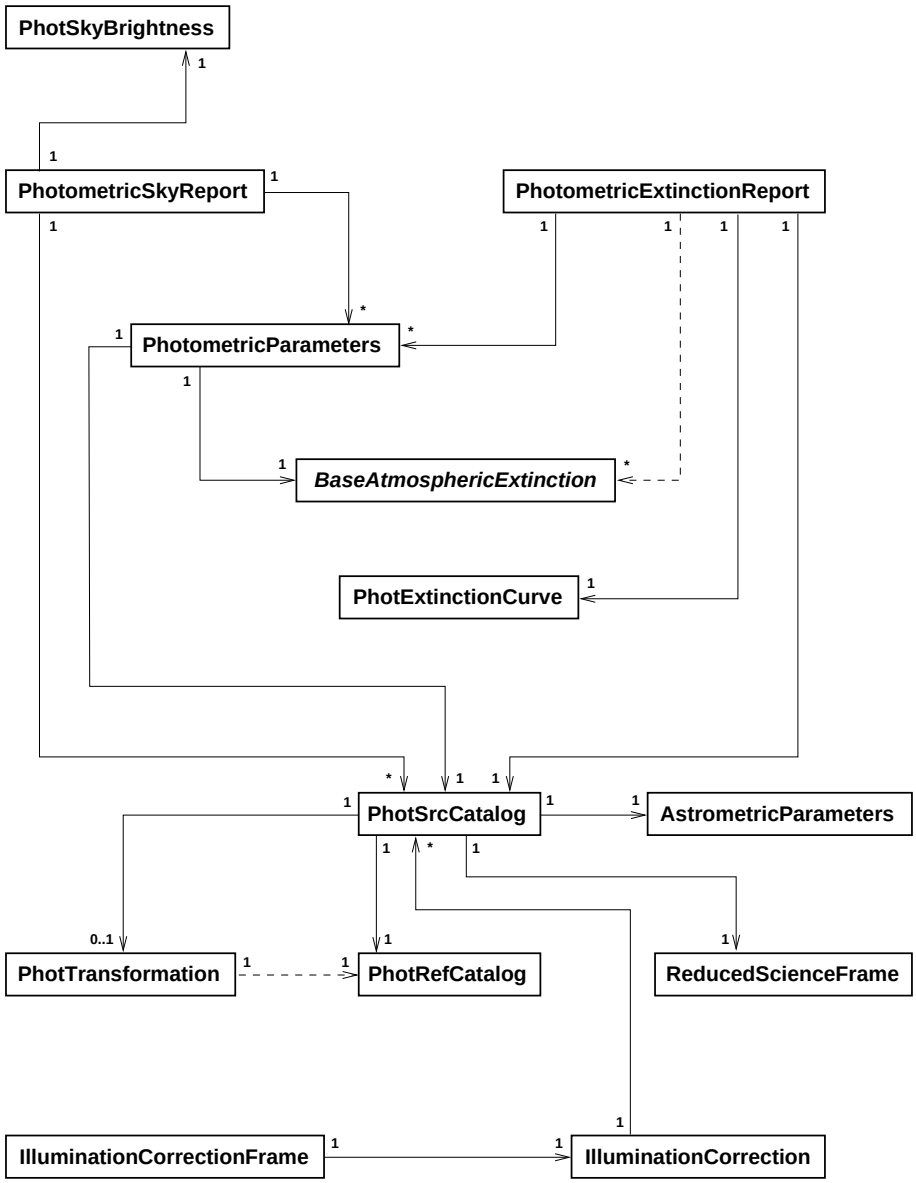

Figure 1.1: The core design diagram of the photometric pipeline

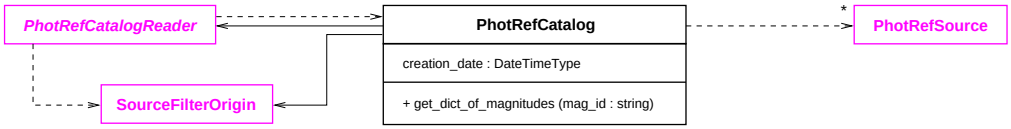

Figure 1.2: The definition of the PhotRefCatalog class and its associates.

the instrumental magnitude m', and its associated error ($M - m'$ and error). Take note that m' is **not** corrected for atmospheric extinction. The instrumental magnitude m' is in this system defined as:

$$m' = -2.50 \cdot \log\left(\frac{\sum \mathrm{ADU}}{exptime}\right), \tag{1.1}$$

with $\sum$ADU the number of ADU integrated over the source, and *exptime* the exposure time of the frame from which the sources were extracted in seconds.

1.1.3 PhotTransformation

The `PhotTransformation` class represents a table of parameters used to transform magnitudes of standard stars from one photometric system to another (color terms). The equation used to calculate this 'transformed' magnitude M_{T} from other bands is:

$$M_{\mathrm{T}} = M_{\mathrm{prm}} - CT \cdot (M_{\mathrm{scd}} - M_{\mathrm{thd}}) + \delta, \tag{1.2}$$

with CT the color term, and δ an additional shift that can be applied. The M_{prm}, M_{scd} and M_{thd} parameters can each separately be set to any of the bands for which magnitude information is available in the standard star catalog. A dictionary of transformed magnitudes can be retrieved through the *get_dict_of_transformed_magnitudes* functionality.

The class has several attributes. The `filter` attribute identifies the filter to which the transformation table applies, and the `primary_band`, `secondary_band` and `tertiary_band` attributes contain information about which of the photometric bands present in the standard star catalog should be used to provide the input magnitudes for the transformation (the M_{prm}, M_{scd} and M_{thd} magnitudes in Eqn. 1.2). The `coefficient` and `color_term` attributes hold the δ and CT used in the transformation equation (see Eqn. 1.2)

1.1.4 PhotExtinctionCurve

The `PhotExtinctionCurve` class represents an atmospheric extinction curve. The *get_extinction* functionality is the most important one of this class, and is used to get the monochromatic extinction for a given wavelength.

Associated class : PhotExtinctionPoint

The `PhotExtinctionPoint` class represents a single point in an extinction curve and is, therefore, closely associated with the `PhotExtinctionCurve` class where it can have any multiplicity. The class has two attributes: `wavelength` and `extinction`. The class does not have any functionalities.

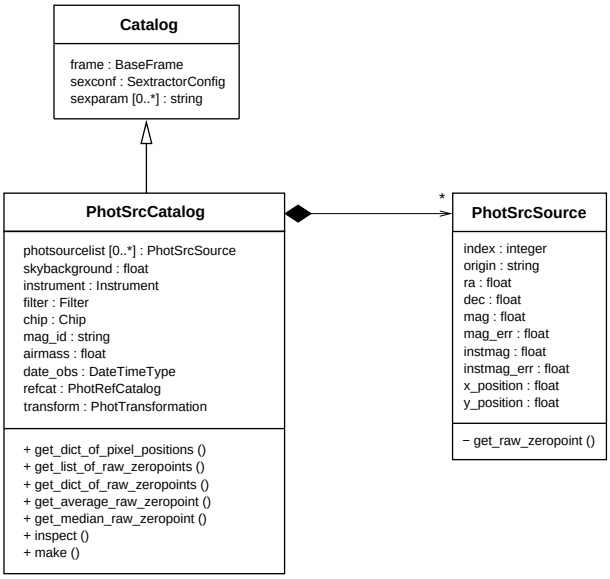

Figure 1.3: The definition of the PhotSrcCatalog and PhotSrcSource classes.

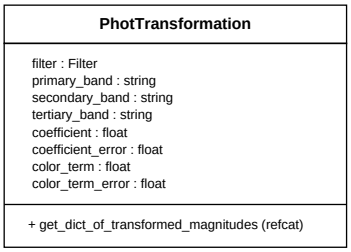

Figure 1.4: The definition of the PhotTransformation class.

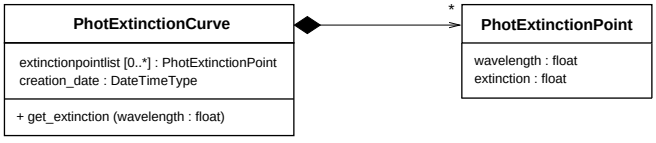

Figure 1.5: The definition of the PhotExtinctionCurve and PhotExtinctionPoint classes.

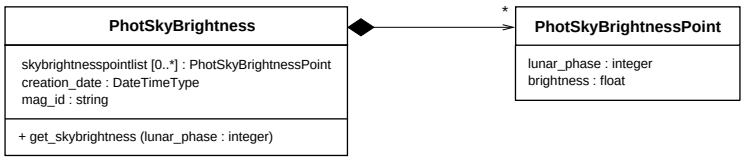

Figure 1.6: The definition of the PhotSkyBrightness and PhotSkyBrightnessPoint classes.

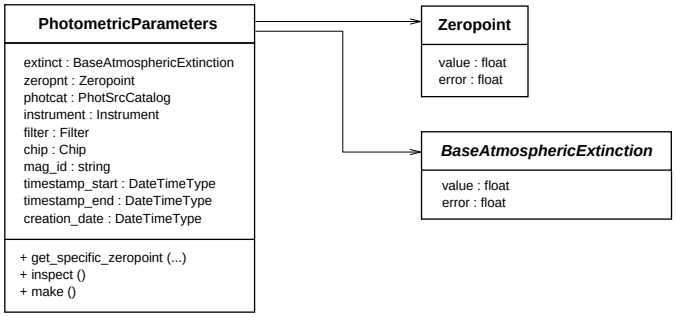

Figure 1.7: The definition of the PhotometricParameters class and its associates.

1.1.5 PhotSkyBrightness

The `PhotSkyBrightness` class represents a table of the brightness of the skybackground as a function of the number of days since the last new moon. A single `PhotSkyBrightness` object contains brightness data for only one photometric band (the one mentioned in its `mag_id` attribute). The skybrightness information is retrieved from the table through its public *get_skybrightness* functionality (see Fig. 1.6).

Associated class : PhotSkyBrightnessPoint

The `PhotSkyBrightnessPoint` class represents a single point in a skybrightness table and is, therefore, closely associated with the `PhotSkyBrightness` class where it can have any multiplicity. The class has two attributes: `lunar_phase` and `brightness`. The class does not have any functionalities.

1.1.6 PhotometricParameters

The `PhotometricParameters` class plays the central role in the photometric system because it represents the final output: the atmospheric extinction and the zeropoint for a specific key or user band.

The class has several public functionalities. The two most important of these are an accessor functionality for the *specific* zeropoint (*get_specific_zeropoint*), and a *make* functionality. The *make* functionality of this class has two dependencies. These are an instance of one of the child classes of the `BaseAtmosphericExtinction` class (attribute name: `extinct`), and a `PhotSrcCatalog` object (attribute name: `photcat`). The end result of using the *make* functionality is the creation of a `Zeropoint` object, that is assigned to the `zeropnt` attribute of the `PhotometricParameters` object (see Fig. 1.7).

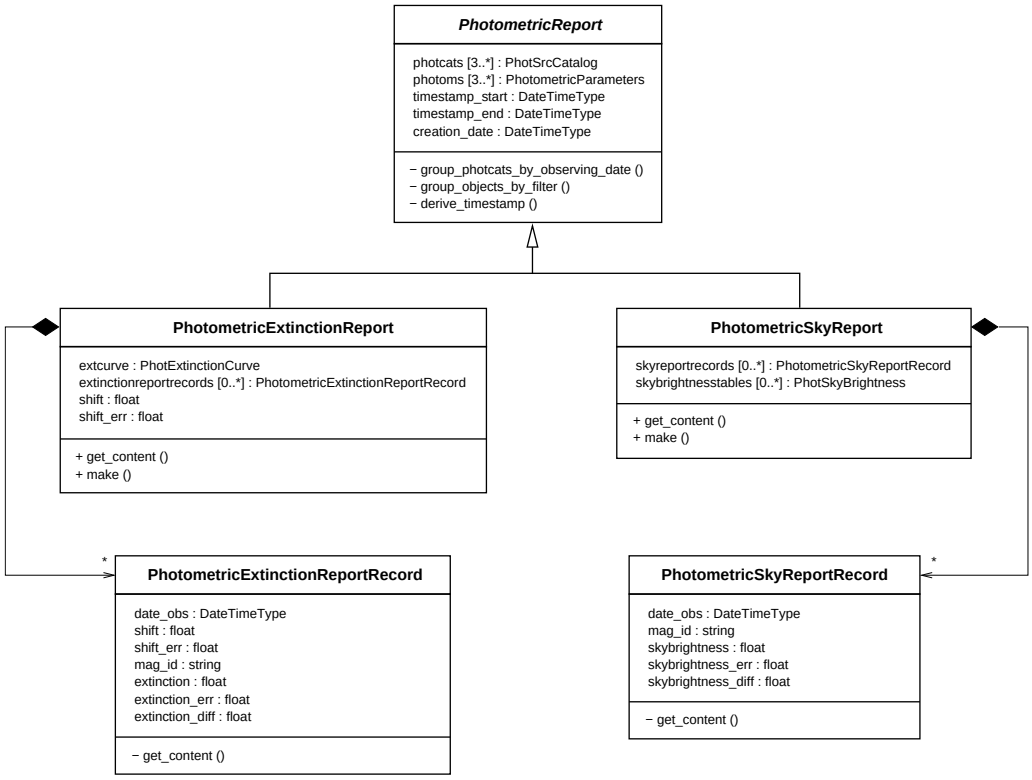

Figure 1.8: The inheritance structure of the family of classes representing the photometric reports.

1.1.7 PhotometricExtinctionReport

The `PhotometricExtinctionReport` class represents a report containing information about the transparency of the atmosphere as measured throughout the night.

The *make* functionality of the class has three dependencies: a list of `PhotSrcCatalog` objects (attribute name: `photcats`), a list of `PhotometricParameters` objects (attribute name: `photoms`), and a `PhotExtinctionCurve` object (attribute name: `extcurve`).

Associated class : PhotometricExtinctionReportRecord

The `PhotometricExtinctionReportRecord` class represents a single entry in an extinction report and is, therefore, closely associated with the `PhotometricExtinctionReport` class where it can have any multiplicity. The class has a *get_content* functionality that is only used internally by the `PhotometricExtinctionReport` object in which the records are stored.

1.1.8 PhotometricSkyReport

The `PhotometricSkyReport` class represents a report containing information about the low-resolution spectrum of the atmosphere as measured throughout the night.

The *make* functionality of the class has three dependencies: a list of `PhotSrcCatalog` objects (attribute name: `photcats`), a list of `PhotometricParameters` objects (attribute name: `photoms`), and a list of `PhotSkyBrightness` objects (attribute name: `skybrightnesstables`).

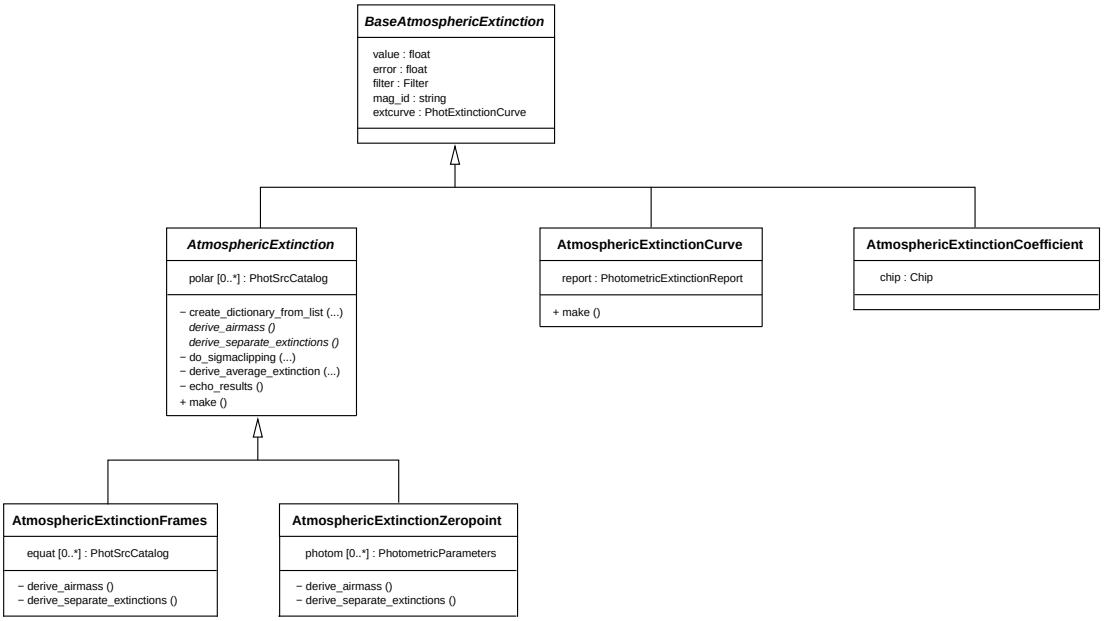

Figure 1.9: The inheritance structure of the family of classes representing the atmosperic extinction.

Associated class : PhotometricSkyReportRecord

The `PhotometricSkyReportRecord` class represents a single entry in a sky-spectrum report and is, therefore, closely associated with the `PhotometricSkyReport` class where it can have any multiplicity. The class has a *get_content* functionality that is only used internally by the `PhotometricSkyReport` object in which the records are stored.

1.1.9 AtmosphericExtinction

The title of this section is somewhat misleading in that it actually hides a complete family of classes all related to the atmospheric extinction. Four types of atmospheric extinction are discerned in the photometric system based on the way these are derived.

The 'classical' way to derive an atmospheric extinction, in which two images of standard fields at different airmasses are combined, is represented by the `AtmosphericExtinctionFrames` class. This class has two dependencies, both lists of `PhotSrcCatalog` objects; these are the `polar` and `equat` attributes of the class. The only (public) functionality of the class is the *make* functionality. This mode of deriving the extinction is only supported to meet the popular demand, but is not used in the default modus operandus of the photometric pipeline.

In case the zeropoint of the detector chain is already known, the atmospheric extinction can be derived using only one image of a standard field. This particular way of deriving the atmospheric extinction is covered by the `AtmosphericExtinctionZeropoint` class. Again, the only (public) functionality of the class is the *make* functionality. This functionality has two dependencies: a list of `PhotSrcCatalog` objects (attribute name: `polar`), and a list of `PhotometricParameters` objects (attribute name: `photom`). This way of deriving the atmospheric extinction is solely used by `PhotometricExtinctionReport` objects (see Fig. 1.1).

For verification purposes, objects of both these classes can be provided with an instance of

the `PhotExtinctionCurve` class. The attribute to which this object is assigned is `extcurve`. This curve is used as a form of sanity check on the derived atmospheric extinction when using the *verify* functionality.

The third way of deriving the atmospheric extinction supported by the photometric system is by the use of a (shifted) standard extinction curve. This 'type' of atmospheric extinction is represented by the `AtmosphericExtinctionCurve` class. The *make* functionality is the only (public) one of this class, which has three dependencies : a `PhotExtinctionCurve` object (attribute name: `extcurve`), a `PhotometricExtinctionReport` object (attribute name: `report`), and a `Filter` object (attribute name: `filter`). This last way of deriving the atmospheric extinction is the default used in the photometric pipeline.

The fourth way in which an atmospheric extinction can be 'derived' is by using a default value for the extinction coefficient. This is covered by the `AtmosphericExtinctionCoefficient` class. This class is not really a `ProcessTarget`. Properly constructed instances, therefore, have to be created 'by hand' before these can be used by the system.

The four *concrete* classes described above are tied together by two *abstract* base classes, the `AtmosphericExtinction` and `BaseAtmosphericExtinction` class, and form a hybrid of a *Strategy* and a *Template Method* pattern. For details of the full hierarchy, see Fig. 1.9.

1.1.10 IlluminationCorrection

The 2-dimensional map that describes the illumination variation across the chips of an instrument is represented by the `IlluminationCorrection` class. The class has a *make* functionality with one dependency : a list of `PhotSrcCatalog` objects (attribute name: `photcats`). The illumination variation is determined by fitting a 2-dimensional, second order polynomial to *all* the raw zeropoints of the standard stars contained in the separate `PhotSrcCatalog` objects simultaneously. The polynomial fit is of the form:

$$z(x, y) = C_0 + C_x \cdot x + C_y \cdot y + C_{xx} \cdot x^2 + C_{xy} \cdot xy + C_{yy} \cdot y^2, \tag{1.3}$$

with z the raw zeropoint from a standard star located at grid position (x, y), and C_α the coefficients to be determined. The overall, global fit is then 'translated' to local fitparameters for every separate chip and stored for future use. The fit parameters for a given chip can be retrieved from the `IlluminationCorrection` object throught its *get_fit_parameters* method. These fit parameters are in magnitudes.

It should be emphasized that the *make* functionality can only be used to make a quick-and-simple, rough estimate of the effects of stray light; a proper characterization of the illumination variation should be made in an interactive analyis. Also, a meaningful result can only be obtained if the input chips are densely and uniformly covered with standard stars.

Associated class : IlluminationCorrectionRecord

The `IlluminationCorrectionRecord` class represents a single entry in an illumination correction map and is, therefore, closely associated with the `IlluminationCorrection` class where it can have any multiplicity. The class has a *get_fit_parameters* and a *get_content* functionality, which are only used internally by the `IlluminationCorrection` object in which the records are stored. One `IlluminationCorrectionRecord` object is stored for every unique chip that went into making the illumination correction map (see Sect. 1.1.10).

1.1.11 IlluminationCorrectionFrame

The illumination correction is applied during processing using an illumination correction frame. This frame is represented in the photometric system by the `IlluminationCorrectionFrame`

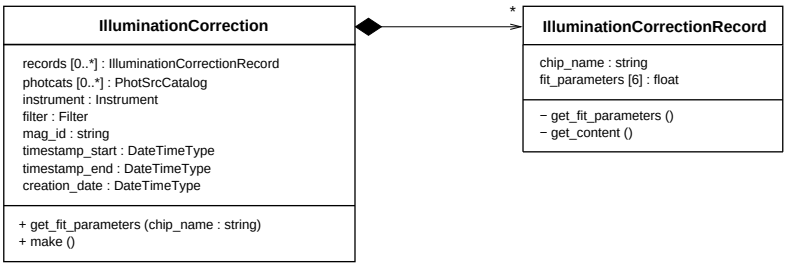

Figure 1.10: The detailed make-up of the IlluminationCorrection and IlluminationCorrectionRecord classes.

class. The class has a *make* functionality with two dependencies : an `IlluminationCorrection` object (attribute name : `illuminationcorrection`), and a `Chip` object (attribute name : `chip`).

1.1.12 Process configuration classes

Many of the classes with a *make* functionality come with a set of configuration parameters. The configuration parameters are attributes of dedicated parameter classes, of which an instance is assigned to the `process_params` attribute of the class it configures. The name of a configuration class always ends with `...Parameters`, and begins with the name of the class it configures. The class that contains the configurable parameters of the `PhotSrcCatalog` class, for example, is called `PhotSrcCatalogParameters`. A full overview of the process parameters available in the photometric pipeline can be found in appendix B.

1.1.13 Miscellaneous

Besides the classes which are specific for the photometric sub-system, the system also uses a set of classes that are borrowed from other parts of the Astro-Wise system. These include the `ReducedScienceFrame`, `AstrometricParameters`, `Filter`, `Chip` and `Instrument` classes.

The role of the Filter class in the photometric system

Of all the miscellaneous classes listed above, the `Filter` class (obviously) has a special relation to the photometric sub-system, which uses many of its attributes. Besides the use of its attributes, a `Filter` object is also an actual *dependency* of an `AtmosphericExtinctionCurve` object.

The `Filter` class has two attributes that are used by the photometic pipeline : a `mag_id` attribute, and a `central_wavelength` attribute. Of these, the `mag_id` attribute is the most important one for the proper functioning of the photometric sub-system, because it is the primary key through which the system identifies the photometric band to which the filter belongs. It is especially important for obtaining the correct magnitude information from a `PhotRefCatalog` object; the magnitude information contained within every separate `PhotRefSource` object is completely described in terms of this `mag_id` parameter (see Fig. 1.2). The same is also true for the `primary_band`, `secondary_band` and `tertiary_band` attributes of the `PhotTransformation` class (see Fig. 1.4). For a full list of the photometric bands known to the system see Table C.2.

Less important for the photometric sub-system is the `central_wavelength` attribute of the `Filter` class. The value of the `central_wavelength` attribute is used as parameter in the *get_extinction* functionality of the `PhotExtinctionCurve` class (see Fig. 1.5).

The role of the Chip class in the photometric system

The `Chip` class plays a more modest role in the photometric pipeline. Its use is mostly limited as a tracking mechanism of the internal data flow; almost everything in the pipeline is on a per-chip basis. However, there is one place where a `Chip` object is more important than that, and that is in the creation of an illumination correction frame (see Sect. 1.1.11). Here it is used as a dependency from which the right dimensions are retrieved of the frame to be made. These dimensions are retrieved through the *get_trimmed_size* method of the `Chip` object.

Appendix A

Glossary of dependencies and public methods

This appendix gives an overview of the *public* methods of the photometric classes and, where applicable, their dependencies.

A.1 PhotSrcCatalog

The dependencies and public methods of the `PhotSrcCatalog` class are given in Table A.1. Note that fullfilling the `transform` dependency is optional.

A.2 PhotRefCatalog

The `PhotRefCatalog` class has no *make* functionality and, therefore, no dependencies. The public methods are given in Table A.2.

A.3 PhotExtinctionCurve

The `PhotExtinctionCurve` class has no *make* functionality and, therefore, no dependencies. The public methods are given in Table A.3.

A.4 PhotSkyBrightness

The `PhotSkyBrightness` class has no *make* functionality and, therefore, no dependencies. The public methods are given in Table A.4.

A.5 PhotTransformation

The `PhotTransformation` class has no *make* functionality and, therefore, no dependencies. The public methods are given in Table A.5.

Table A.1: The public methods and dependencies of the `PhotSrcCatalog` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`inspect()`	no parameters	output to screen
`get_list_of_raw_zeropoints()`	no parameters	list of 2-tuples
`get_dict_of_raw_zeropoints()`	no parameters	dictionary of 2-tuples
`get_average_raw_zeropoint()`	no parameters	2-tuple of floats
`get_median_raw_zeropoint()`	no parameters	float
`get_dict_of_pixel_positions()`	no parameters	list of 2-tuples
`get_content_of_catalog()`	no parameters	list of 7-tuples
`get_source_attributes()`	no parameters	dictionary
`get_source_data(columns)`	`columns` : list of strings	dictionary of lists
`get_number_of_sources()`	no parameters	integer
`make_skycat()`	no parameters	NA
Dependency	**Input type**	
`frame`	one `ReducedScienceFrame` object	
`refcat`	one `PhotRefCatalog` object	
`astrom_params`	one `AstrometricParameters` object	
`transform`	one `PhotTransformation` object (optional)	

Table A.2: The public methods of the `PhotRefCatalog` class.

Method	Parameter description	Return type
`associate(incat, outcat)`	`incat` : string `outcat` : string	NA
`get_dict_of_magnitudes(mag_id)`	`mag_id` : string	dictionary of 2-tuples
`get_dict_of_origins()`	no parameters	dictionary of floats
`get_list_of_bands()`	no parameters	list of strings
`get_source_attributes()`	no parameters	dictionary
`get_source_data(columns)`	`columns` : list of strings	dictionary of lists
`get_number_of_sources()`	no parameters	integer
`make_skycat()`	no parameters	NA
`dump()`	no parameters	NA

Table A.3: The public methods of the `PhotExtinctionCurve` class.

Method	Parameter description	Return type
`get_wavelength_range()`	no parameters	2-tuple of floats
`get_extinction(wavelength)`	`wavelength` : float	float

Table A.4: The public methods of the `PhotSkyBrightness` class.

Method	Parameter description	Return type
`get_skybrightness(lunar_phase)`	`lunar_phase` : int	float

Table A.5: The public methods of the `PhotTransformation` class.

Method	Parameter description	Return type
`get_color_for_filter()`	no parameters	string
`get_dict_of_transformed_magnitudes (refcat)`	`refcat` : `PhotRefCatalog object`	dict. of 2-tuples

Table A.6: The public methods and dependencies of the `PhotometricParameters` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`inspect()`	no parameters	output to screen
`get_zeropoint()`	no parameters	2-tuple of floats
`get_specific_zeropoint(exptime, airmass)`	`exptime` : float	2-tuple of floats
	`airmass` : float	2-tuple of floats
`get_extinction()`	no parameters	2-tuple of floats

Dependency	Input type
`photcat`	one `PhotSrcCatalog` object
`extinct`	one *BaseAtmosphericExtinction* object

Table A.7: The public methods and dependencies of the `PhotometricExtinctionReport` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`inspect()`	no parameters	NA
`get_shift()`	no parameters	2-tuple of floats
`get_content()`	no parameters	list of 7-tuples

Dependency	Input type
`photcats`	list of `PhotSrcCatalog` objects
`photoms`	list of `PhotometricParameters` objects
`extcurve`	one `PhotExtinctionCurve` object

Table A.8: The public methods and dependencies of the `PhotometricSkyReport` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`get_content()`	no parameters	list of 5-tuples
Dependency	**Input type**	
`photcats`	list of `PhotSrcCatalog` objects	
`photoms`	list of `PhotometricParameters` objects	
`skybrightnesstables`	list of `PhotSkyBrightness` objects	

Table A.9: The public methods and dependencies of the `IlluminationCorrection` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`inspect()`	no parameters	NA
`get_fit_parameters(chip_name)`	`chip_name` : string	list of floats
`evaluate(chip_name, x_pos, y_pos)`	`chip_name` : string `x_pos` : float `y_pos` : float	single float
`get_content()`	no parameters	list of lists
Dependency	**Input type**	
`photcats`	list of `PhotSrcCatalog` objects	

Table A.10: The public methods and dependencies of the `IlluminationCorrectionFrame` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`apply(frame)`	`frame` : one `ReducedScienceFrame` object	NA
Dependency	**Input type**	
`illuminationcorrection`	one `IlluminationCorrection` object	
`chip`	one `Chip` object	

Table A.11: The public methods and dependencies of the `AtmosphericExtinctionCurve` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`get_extinction()`	no parameters	2-tuple of floats
Dependency	**Input type**	
`extcurve`	one `PhotExtinctionCurve` object	
`filter`	one `Filter` object	
`report`	one `PhotometricExtinctionReport` object	

Table A.12: The public methods and dependencies of the `AtmosphericExtinctionFrames` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`get_extinction()`	no parameters	2-tuple of floats
Dependency	**Input type**	
`polar`	list of `PhotSrcCatalog` objects	
`equat`	list of `PhotSrcCatalog` objects	
`extcurve`†	one `PhotExtinctionCurve` object	

† : is used by the `verify` method

A.6 PhotometricParameters

The dependencies and public methods of the `PhotometricParameters` class are given in Table A.6. The actual type of the object assigned to the `extinct` attribute is one of the concrete sub-types of the *BaseAtmosphericExtinction* class.

A.7 PhotometricExtinctionReport

The dependencies and public methods of the `PhotometricExtinctionReport` class are given in Table A.7.

A.8 PhotometricSkyReport

The dependencies and public methods of the `PhotometricSkyReport` class are given in Table A.8.

Table A.13: The public methods and dependencies of the `AtmosphericExtinctionZeropoint` class.

Method	Parameter description	Return type
`make()`	no parameters	NA
`get_extinction()`	no parameters	2-tuple of floats
Dependency	**Input type**	
`polar`	list of `PhotSrcCatalog` objects	
`photoms`	list of `PhotometricParameters` objects	
`extcurve`†	one `PhotExtinctionCurve` object	

† : is used by the `verify` method

A.9 IlluminationCorrection

The dependencies and public methods of the `IlluminationCorrection` class are given in Table A.9.

A.10 IlluminationCorrectionFrame

The dependencies and public methods of the `IlluminationCorrectionFrame` class are given in Table A.10.

A.11 AtmosphericExtinction

The dependencies and public methods of the concrete sub-types of the *BaseAtmosphericExtinction* class are given in Tables A.11 - A.13.

Appendix B

Configurable process parameters

The table in this appendix gives an overview of the process parameters that can be configured in the photometric pipeline. The first column gives the class to which the configuration applies, and the second column lists the available parameters. The third and fourth columns list the type and default value for the relevant parameter, respectively.

Class	Parameter	Type	Default
PhotSrcCatalog	FLUX_TYPE	STRING	FLUX_APER
	MAX_MAG_DIFF	FLOAT	0.5
PhotometricParameters	SIGCLIP_LEVEL	FLOAT	1.5
	MIN_NMBR_OF_STARS	INT	3
	MAX_ERROR	FLOAT	0.03 mag
PhotometricExtinctionReport	SIGCLIP_LEVEL	FLOAT	3.0
	MAX_ERROR	FLOAT	0.02 mag
PhotometricSkyReport	LUNAR_PHASE	INT	0
AtmosphericExtinction	SIGCLIP_LEVEL	FLOAT	2.0
	MAX_ERROR	FLOAT	0.05 mag
IlluminationCorrection	SIGCLIP_LEVEL	FLOAT	4.0

Appendix C

Data structures

Although the photometric pipeline is built to operate in a fully database-driven fashion, it is also required to function in a *non*-database environment in which the results and inputs are stored as files. In this appendix, a description is given of the structure of these files and how this structure maps onto the class definition. Except for the frames, all files produced or used by the photometric pipeline are in the LDAC fits format.

C.1 PhotSrcCatalog

The LDAC fits file representing a `PhotSrcCatalog` object has a `FIELDS` and an `OBJECTS` table, the latter storing the list of `PhotSrcSource` objects contained within the `PhotSrcCatalog` object. The file lay-out is given in Table C.1. The `date_obs` entry in the `FIELDS` table is defined in the `PhotSrcCatalog` class as a `DateTimeType` type.

C.2 PhotRefCatalog

The LDAC fits file representing a `PhotRefCatalog` object only has a `STDTAB` table. Every row in the table corresponds to one `PhotRefSource` object contained within the `PhotRefCatalog` object. The file lay-out is given in Table C.2.

C.3 PhotExtinctionCurve

The LDAC fits file representing a `PhotExtinctionCurve` object has a `FIELDS` and an `OBJECTS` table; the list of `PhotExtinctionPoint` objects contained within the `PhotExtinctionCurve` object is stored in the latter. The file lay-out is given in Table C.3.

C.4 PhotSkyBrightness

The LDAC fits file representing a `PhotSkyBrightness` object has a `FIELDS` and an `OBJECTS` table, the latter storing the list of `PhotSkyBrightnessPoint` objects contained within the `PhotSkyBrightness` object. The file lay-out is given in Table C.4.

Table C.1: File representation of a `PhotSrcCatalog` object.

FITS table	Column name	Type	Maps onto
FIELDS	INSTRUME	STRING	PhotSrcCatalog.instrument.name
	TELESCOP	STRING	PhotSrcCatalog.instrument.telescope_name
	CHIP_ID	STRING	PhotSrcCatalog.chip.name
	FILT_ID	STRING	PhotSrcCatalog.filter.name
	MAG_ID	STRING	PhotSrcCatalog.mag_id
			PhotSrcCatalog.filter.mag_id
	CWL	FLOAT	PhotSrcCatalog.filter.central_wavelength
	airmass	FLOAT	PhotSrcCatalog.airmass
	skybackground	FLOAT	PhotSrcCatalog.skybackground
	date_obs	STRING	PhotSrcCatalog.date_obs
	CRVAL1	FLOAT	PhotSrcCatalog.astrom_params.CRVAL1
	CRVAL2	FLOAT	PhotSrcCatalog.astrom_params.CRVAL2
	CD1_1	FLOAT	PhotSrcCatalog.astrom_params.CD1_1
	CD1_2	FLOAT	PhotSrcCatalog.astrom_params.CD1_2
	CD2_1	FLOAT	PhotSrcCatalog.astrom_params.CD2_1
	CD2_2	FLOAT	PhotSrcCatalog.astrom_params.CD2_2
OBJECTS	index	INT	PhotSrcSource.index
	origin	STRING	PhotSrcSource.origin
	Ra	FLOAT	PhotSrcSource.ra
	Dec	FLOAT	PhotSrcSource.dec
	mag	FLOAT	PhotSrcSource.mag
	mag_err	FLOAT	PhotSrcSource.mag_err
	instmag	FLOAT	PhotSrcSource.instmag
	instmag_err	FLOAT	PhotSrcSource.instmag_err
	Xpos	FLOAT	PhotSrcSource.x_position
	Ypos	FLOAT	PhotSrcSource.y_position

Table C.2: File representation of a `PhotRefCatalog` object.

FITS table	Column name	Type	Maps onto
STDTAB	SeqNr	INT	PhotRefSource.index
	origin	STRING	PhotRefSource.origin
	Name	STRING	PhotRefSource.star_id
	Ra	FLOAT	PhotRefSource.ra
	Ra_err	FLOAT	PhotRefSource.ra_err
	Dec	FLOAT	PhotRefSource.dec
	Dec_err	FLOAT	PhotRefSource.dec_err
	Epoch	FLOAT	PhotRefSource.epoch
	Flag	INT	PhotRefSource.flag
	JohnsonU	FLOAT	PhotRefSource.JohnsonU
	JohnsonU_err	FLOAT	PhotRefSource.JohnsonU_err
	JohnsonB	FLOAT	PhotRefSource.JohnsonB
	JohnsonB_err	FLOAT	PhotRefSource.JohnsonB_err
	JohnsonV	FLOAT	PhotRefSource.JohnsonV
	JohnsonV_err	FLOAT	PhotRefSource.JohnsonV_err
	CousinsR	FLOAT	PhotRefSource.CousinsR
	CousinsR_err	FLOAT	PhotRefSource.CousinsR_err
	CousinsI	FLOAT	PhotRefSource.CousinsI
	CousinsI_err	FLOAT	PhotRefSource.CousinsI_err
	SloanU	FLOAT	PhotRefSource.SloanU
	SloanU_err	FLOAT	PhotRefSource.SloanU_err
	SloanG	FLOAT	PhotRefSource.SloanG
	SloanG_err	FLOAT	PhotRefSource.SloanG_err
	SloanR	FLOAT	PhotRefSource.SloanR
	SloanR_err	FLOAT	PhotRefSource.SloanR_err
	SloanI	FLOAT	PhotRefSource.SloanI
	SloanI_err	FLOAT	PhotRefSource.SloanI_err
	SloanZ	FLOAT	PhotRefSource.SloanZ
	SloanZ_err	FLOAT	PhotRefSource.SloanZ_err

Table C.3: File representation of a `PhotExtinctionCurve` object.

FITS table	Column name	Type	Maps onto
FIELDS	wavel_incr	FLOAT	PhotExtinctionCurve.wavel_incr
OBJECTS	wavelength	FLOAT	PhotExtinctionPoint.wavelength
	extinction	FLOAT	PhotExtinctionPoint.extinction

Table C.4: File representation of a `PhotSkyBrightness` object.

FITS table	Column name	Type	Maps onto
FIELDS	MAG_ID	STRING	PhotSkyBrightness.mag_id
OBJECTS	lunar_phase	INT	PhotSkyBrightnessPoint.lunar_phase
	brightness	FLOAT	PhotSkyBrightnessPoint.brightness

Table C.5: File representation of a `PhotTransformation` object.

FITS table	Column name	Type	Maps onto
STDTAB	INSTRUME	STRING	PhotTransformation.instrument.name
	FILT_ID	STRING	PhotTransformation.filter.name
	timestamp_start	STRING	PhotTransformation.timestamp_start
	timestamp_end	STRING	PhotTransformation.timestamp_end
	primary_band	STRING	PhotTransformationElement.primary_band
	secondary_band	STRING	PhotTransformationElement.secondary_band
	tertiary_band	STRING	PhotTransformationElement.tertiary_band
	coefficient	FLOAT	PhotTransformationElement.coefficient
	coefficient_error	FLOAT	PhotTransformationElement.coefficient_error
	color_term	FLOAT	PhotTransformationElement.color_term
	color_term_error	FLOAT	PhotTransformationElement.color_term_error

Table C.6: File representation of a `PhotometricParameters` object.

FITS table	Column name	Type	Maps onto
`STDTAB`	`INSTRUME`	`STRING`	PhotometricParameters.instrument.name
	`CHIP_ID`	`STRING`	PhotometricParameters.chip.name
	`FILT_ID`	`STRING`	PhotometricParameters.filter.name
	`MAG_ID`	`STRING`	PhotometricParameters.filter.mag_id
			PhotometricParameters.mag_id
	`timestamp_start`	`STRING`	PhotometricParameters.timestamp_start
	`timestamp_end`	`STRING`	PhotometricParameters.timestamp_end
	`zeropnt`	`FLOAT`	PhotometricParameters.zeropnt.value
	`zeropnt_err`	`FLOAT`	PhotometricParameters.zeropnt.error
	`extinct`	`FLOAT`	PhotometricParameters.extinct.value
	`extinct_err`	`FLOAT`	PhotometricParameters.extinct.error

C.5 PhotTransformation

The LDAC fits file representing a `PhotRefCatalog` object only has a `STDTAB` table. The file lay-out is given in Table C.5. The `timestamp_start` and `timestamp_end` entries are defined in the `PhotTransformation` class as a `DateTimeType` type.

C.6 PhotometricParameters

The LDAC fits file representing a `PhotometricParameters` object only has a `STDTAB` table. The `timestamp_start` and `timestamp_end` entries are defined in the `PhotometricParameters` class as a `DateTimeType` type. The file lay-out is given in Table C.6.

C.7 PhotometricExtinctionReport

The LDAC fits file representing a `PhotometricExtinctionReport` object has a `FIELDS` and an `OBJECTS` table, the latter storing the list of `PhotometricExtinctionReportRecord` objects contained within the `PhotometricExtinctionReport` object. The file lay-out is given in Table C.7. The `date_obs` entry in the `OBJECTS` table is defined as a `DateTimeType` type in the `PhotometricExtinctionReportRecord` class. The same is true for the `timestamp_start` and `timestamp_end` entries in the `FIELDS` table.

C.8 PhotometricSkyReport

The LDAC fits file representing a `PhotometricSkyReport` object has a `FIELDS` and an `OBJECTS` table, the latter storing the list of `PhotometricSkyReportRecord` objects contained within the `PhotometricSkyReport` object. The file lay-out is given in Table C.8. The `date_obs` entry in the `OBJECTS` table is defined in the `PhotometricSkyReportRecord` class as a `DateTimeType` type. The same is true for the `timestamp_start` and `timestamp_end` entries in the `FIELDS` table.

Table C.7: File representation of a `PhotometricExtinctionReport` object.

FITS table	Column name	Type	Maps onto
FIELDS	INSTRUME	STRING	PhotometricExtinctionReport.instrument.name
	timestamp_start	STRING	PhotometricExtinctionReport.timestamp_start
	timestamp_end	STRING	PhotometricExtinctionReport.timestamp_end
	shift	FLOAT	PhotometricExtinctionReport.shift
	shift_err	FLOAT	PhotometricExtinctionReport.shift_err
OBJECTS	date_obs	STRING	PhotometricExtinctionReportRecord.date_obs
	shift	FLOAT	PhotometricExtinctionReportRecord.shift
	shift_err	FLOAT	PhotometricExtinctionReportRecord.shift_err
	MAG_ID	STRING	PhotometricExtinctionReportRecord.mag_id
	extinction	FLOAT	PhotometricExtinctionReportRecord.extinction
	extinction_err	FLOAT	PhotometricExtinctionReportRecord.extinction_err
	extinction_diff	FLOAT	PhotometricExtinctionReportRecord.extinction_diff

Table C.8: File representation of a `PhotometricSkyReport` object.

FITS table	Column name	Type	Maps onto
FIELDS	INSTRUME	STRING	PhotometricSkyReport.instrument.name
	timestamp_start	STRING	PhotometricSkyReport.timestamp_start
	timestamp_end	STRING	PhotometricSkyReport.timestamp_end
OBJECTS	date_obs	STRING	PhotometricSkyReportRecord.date_obs
	MAG_ID	STRING	PhotometricSkyReportRecord.mag_id
	skybackground	FLOAT	PhotometricSkyReportRecord.extinction
	skybackground_err	FLOAT	PhotometricSkyReportRecord.extinction_err
	skybackground_diff	FLOAT	PhotometricSkyReportRecord.extinction_diff

Table C.9: File representation of an `IlluminationCorrection` object.

FITS table	Column name	Type	Maps onto
FIELDS	FILT_ID	STRING	IlluminationCorrection.filter.name
	MAG_ID	STRING	IlluminationCorrection.filter.mag_id
			IlluminationCorrection.mag_id
	INSTRUME	STRING	IlluminationCorrection.instrument.name
	TELESCOP	STRING	IlluminationCorrection.instrument.telescope_name
	timestamp_start	STRING	IlluminationCorrection.timestamp_start
	timestamp_end	STRING	IlluminationCorrection.timestamp_end
	min_x	FLOAT	IlluminationCorrection.min_x
	max_x	FLOAT	IlluminationCorrection.max_x
	min_y	FLOAT	IlluminationCorrection.min_y
	max_y	FLOAT	IlluminationCorrection.max_y
	C_o	FLOAT	IlluminationCorrection.fitcoeffs
	C_x	FLOAT	IlluminationCorrection.fitcoeffs
	C_y	FLOAT	IlluminationCorrection.fitcoeffs
	C_xx	FLOAT	IlluminationCorrection.fitcoeffs
	C_xy	FLOAT	IlluminationCorrection.fitcoeffs
	C_yy	FLOAT	IlluminationCorrection.fitcoeffs
OBJECTS	chip_name	STRING	IlluminationCorrectionRecord.chip_name
	C_o	FLOAT	IlluminationCorrectionRecord.fit_parameters
	C_x	FLOAT	IlluminationCorrectionRecord.fit_parameters
	C_y	FLOAT	IlluminationCorrectionRecord.fit_parameters
	C_xx	FLOAT	IlluminationCorrectionRecord.fit_parameters
	C_xy	FLOAT	IlluminationCorrectionRecord.fit_parameters
	C_yy	FLOAT	IlluminationCorrectionRecord.fit_parameters

C.9 IlluminationCorrection

The LDAC fits file representing an `IlluminationCorrection` object has a `FIELDS` and an `OBJECTS` table, the latter storing the list of `IlluminationCorrectionRecord` objects contained within the `IlluminationCorrection` object. The file lay-out is given in Table C.9. The `timestamp_start` and `timestamp_end` entries in the `FIELDS` table are defined as a `DateTimeType` type in the `IlluminationCorrection` class.