GammaLib INST

Instrument specific interface

Version draft
15 February 2011

Author: Jürgen Knödlseder
Approved by: Jürgen Knödlseder

Institut de Recherche en Astrophysique et Planétologie (IRAP)
9, avenue du Colonel-Roche
31028 Toulouse Cedex 4
FRANCE

This page intentionally left blank

Contents

1 Introduction

The present document describes the instrument specific interface of the `GammaLib` toolbox and provides guideslines of how to add a new interface to the toolbox.

The general philosophy of `GammaLib` is that instrument specific interfaces are implemented as classes that derive from abstract base classes (i.e. the base class can not be instantiated). The base class will implement as most methods as possible to reduce the workload for integration of new instrument specific interfaces. Most base class methods, however, are declared as `virtual` and may be overloaded by instrument specific methods that implement more efficient computations adapted to the specific telescope.

The instrument specific aspects that have to be considered are the implementation of the instrument response functions (base class `GResponse`) and of the instrument data (base classes `GEvent` and `GEvents`). Each instrument may come with its own "data space", and methods are needed that provide an abstract representation of this data space (base classes `GInstDir` and `GRoi`). Also, instruments may point the sky in various ways and strategies (in particular when comparing ground based telescopes to satellites), and also here an abstract representation is needed of these strategies (base class `GPointing`). Finally, the data related to the observation with a given instrument differ, and an abstract interface is needed to combined those (base class `GObservation`).

Instrument specific classes are defined by inserting the instrument code `XXX` between the capital `G` and the remaining part of the name. For example, the `GResponse` class for `CTA` becomes `GCTAResponse`. Throughout this document, we will indicate derived class names by inserting the instrument code `XXX` in the name. Instrument codes should always be 3 letters, unless name confusion imposes the use of more letters.

The following table summarizes the C++ classes that have to be implemented for each instrument. Examples for the names of the derived classes are given for `CTA`.

Table 1: Instrument specific classes that have to be implemented.

Name	CTA name	Section	Usage
`GEventAtom`	`GCTAEventAtom`	??	Implements individual events for unbinned analysis
`GEventBin`	`GCTAEventBin`	??	Implements data space bins for binned analysis
`GEventCube`	`GCTAEventCube`	??	Container of data space bins (event cube)
`GEventList`	`GCTAEventList`	??	Container of events (event list)
`GInstDir`	`GCTAInstDir`	??	Photon "direction" in instrument coordinates
`GObservation`	`GCTAObservation`	??	Implements instrument specific observation
`GPointing`	`GCTAPointing`	??	Implements instrument specific pointing
`GResponse`	`GCTAResponse`	??	Implements instrument specific response function
`GRoi`	`GCTARoi`	??	Implements instrument specific data selection

To start from scratch the implementation of a new instrument interface, we strongly recommend to use the corresponding classes in the `template` directory of the `GammaLib` distribution (instrument code `XXX`).

2 Instrument response

2.1 Response model

2.1.1 General instrument response function

The general instrument response function $R(\vec{p'}, E', t'|\vec{d}, \vec{p}, E, t)$ provides the effective detection area per time, energy and solid angle (in units of cm^2 s^{-1} MeV^{-1} sr^{-1}) for measuring a photon at position $\vec{p'}$ with an energy of E' and at the time t' if it arrives from direction $\vec{p}$ with an energy of E at time t on the instrument that is pointed towards $\vec{d}$.

The photon arrival direction $\vec{p}$ is expressed by a coordinate on the celestial sphere, for example Right Ascension α and Declination δ. For imaging instruments, the measured photon position $\vec{p'}$ is likely also a coordinate on the celestial sphere, while for non-imaging instruments (such as coded masks or Compton telescopes), $\vec{p'}$ will be typically the pixel number of the detector that measured the photon. The definition of $\vec{p'}$ needs to be implemented for each instrument as a derived class from the abstract base class `GInstDir`.

Assuming that the photon intensity received from a gamma-ray source is described by the source model $S(\vec{p}, E, t)$ (in units of photons cm^{-2} s^{-1} MeV^{-1} sr^{-1}) the probability of measuring an event at position $\vec{p'}$ with an energy of E' at the time t' from the source is given by

$$P(\vec{p'}, E', t'|\vec{d}) = \int_0^{t'+\Delta t} \int_{E'-\Delta E}^{\infty} \int_{\Omega} S(\vec{p}, E, t)\, R(\vec{p'}, E', t'|\vec{d}, \vec{p}, E, t)\, \mathrm{d}\vec{p}\, \mathrm{d}E\, \mathrm{d}t \tag{1}$$

(in units of counts s^{-1} MeV^{-1} sr^{-1}). The terms Δt and ΔE account for the statistical jitter related to the measurement process and are of the order of a few time the rms in the time and energy measurements.

Computation of Eq. (??) in this general form is implemented in `GammaLib` by the virtual method `GObservation::model()`. The following diagram gives the call tree of this method:

```
GObservation::model()          - compute Eq. (1) for all source models
  GModelSky::eval_gradients()  - compute Eq. (1) for one source model
    GModelSky::temporal()      - perform integration over time (dt)
      GModelSky::spectral()    - perform integration over energy (dE)
        GModelSky::spatial()   - compute sky position integrated value
          GResponse::irf()     - perform integration over sky positions (dp)
```

The integration over sky positions $\vec{p}$, expressed as a zenith angle θ and an azimuth angle ϕ, is given by

$$P_{\vec{p}}(\vec{p'}, E', t'|\vec{d}, E, t) = \int_{\theta,\phi} S(\theta, \phi, E, t)\, R(\vec{p'}, E', t'|\vec{d}, \theta, \phi, E, t) \sin\theta\, \mathrm{d}\theta\, \mathrm{d}\phi \tag{2}$$

which has to be implemented for each instrument by the virtual method `GResponse::irf()`.[1]

For unbinned analysis we also need to compute the predicted number of events for each source model (for binned analysis we obtained this predicted number by summing over all data space bins). The predicted number of events that are on average detected from a source is given by

$$N_{\mathrm{pred}} = \int_{\mathrm{GTI}} \int_{E_{\mathrm{bounds}}} \int_{\mathrm{ROI}} P(\vec{p'}, E', t'|\vec{d})\, \mathrm{d}\vec{p'}\, \mathrm{d}E'\, \mathrm{d}t' \tag{3}$$

Here, ROI defines an event selection region (such as, e.g., an acceptance cone). The definition of this event selection region is instrument specific, and needs to be implemented for each instrument as a derived class

[1] **TODO:** *Why should we implement this for each instrument? What we need for each instrument is a point source IRF. The integration could be done in the general case using a dumb (and slow) integration method. Thus, we need an instrument dependent point source IRF (where the GModel argument is replaced by GSkyDir) and we need a default implementation of the GModel method.*

from the abstract base class `GRoi`. $E_{\rm bounds}$ are defines a set of energy intervals for which events are to be detected, and GTI specifies the intervals of good time (Good Time Intervals) during which the events are detected.

Computation of Eq. (??) in this general form is implemented in `GammaLib` by the virtual method `GObservation::npred()`. The following diagram gives the call tree of this method:

```
GObservation::npred()                              - compute Eq. (3) for all source models
  GObservation::npred_temp()                       - compute GTI integrated source model
    GObservation::npred_kern_spec::eval()          - time integration kernel
      GObservation::npred_spec()                   - compute Ebounds integrated source model
        GObservation::npred_kern_spat::eval()      - energy integration kernel
          GModelSky::npred()                       - compute ROI integrated source model
            GResponse::npred()                     - perform integration over ROI
```

The integration over the ROI is given by

$$N_{\rm ROI}(E',t') = \int_{\rm ROI} P(\vec{p}', E', t'|\vec{d}) \, {\rm d}\vec{p}' \tag{4}$$

which has to be implemented for each instrument by the virtual method `GResponse::npred()`.

The correspondance between the parameters given in the above equations and `GammaLib` C++ classes and methods is summarized in Table ??.

Table 2: Correspondance between response parameters and `GammaLib` C++ classes and methods. The last column indicates if an instrument specific implementation is required (**yes**) or if it is optional or not possible.

Designation	Equation	Class (::method)	Instrument specific
P	(??)	`GObservation::model()`	optional
$P_{\vec{p}}$	(??)	`GResponse::irf()`	**yes**
$N_{\rm pred}$	(??)	`GObservation::npred()`	optional
$N_{\rm ROI}$	(??)	`GResponse::npred)`	**yes**
S		`GModel`	no
$\vec{p}'$		`GInstDir`	**yes**
E'		`GEnergy`	no
t'		`GTime`	no
d		`GPointing`	**yes**
$\vec{p}$		`GSkyDir`	no
E		`GEnergy`	no
t		`GTime`	no
ROI		`GRoi`	**yes**
GTI		`GGti`	no

2.1.2 Factorised source model

In many situations, the source model can be factorised into a position dependent, an energy dependent, and a time dependent term, i.e.

$$S(\vec{p}, E, t) = S_{\rm S}(\vec{p}) \times S_{\rm E}(E) \times S_{\rm T}(t) \tag{5}$$

2.2 Events

Data are comprised of events. The are two types of events: event atoms and event bins. Events are combined together in `GEvents`. `GEvents` implements an iteration that allows iterating over event atoms or event bins (the differenciation between both is transparent for the user). `GEvents` has an abstract method `pointer(int index)` that returns a pointer to an event. This method, which has to be implement for each instrument, assigns information to all events. Note that in general, only a single event can be accessed at a time (any subsequent class to the `pointer()` method makes the old pointer invalid.

3 `GammaLib` instrument interface

3.1 Response class `GResponse`

The interface for the instrument response function is defined by the abstract base class `GResponse`. A derived class `GXXXResponse` (where `XXX` is the instrument code) needs to be implemented to provide response computations.

The interface of `GResponse` is shown below:

```
virtual void        clear(void) = 0;
virtual GResponse*  clone(void) const = 0;
virtual bool        hasedisp(void) const = 0;
virtual bool        hastdisp(void) const = 0;
virtual double      irf(const GEvent& event, const GModelSky& model,
                        const GEnergy& srcEng, const GTime& srcTime,
                        const GObservation& obs) const = 0;
virtual double      npred(const GModelSky& model, const GEnergy& srcEng,
                          const GTime& srcTime, const GObservation& obs) const = 0;
virtual std::string print(void) const = 0;
```

3.1.1 `GXXXResponse::clear()`

This method should set the instance to a clean initial state.

3.1.2 `GXXXResponse::clone()`

This method should return a pointer to a freshly allocated copy of the response instance.

3.1.3 `GXXXResponse::hasedisp()`

This method should return `true` if the response implements energy dispersions, `false` otherwise.

3.1.4 `GXXXResponse::hastdisp()`

This method should return `true` if the response implements time dispersions, `false` otherwise.

3.1.5 `GXXXResponse::irf()`

This method implements the computation of $P_{\vec{p}}(\vec{p'}, E', t'|\vec{d}, E, t)$ following Eq. (**??**).

3.1.6 `GXXXResponse::npred()`

This method implements the computation of $N_{\rm ROI}(E', t')$ following Eq. (**??**).

3.1.7 Members

No members are implemented by `GResponse`.

3.2 Response support classes

3.2.1 `GXXXInstDir`

Needs to be implemented.

3.2.2 `GXXXRoi`

Region in instrument directions. Needs to be implemented.

3.2.3 `GXXXPointing`

Needs to be implemented.

3.3 Event containers `GEventList` and `GEventCube`

The following methods can **optionally** be implemented to compute $N(\vec{p}', E', t')$ more efficiently that is done by the generic method `GEvents::model()`.

3.3.1 `GXXXEventList::model()`

3.3.2 `GXXXEventCube::model()`

3.4 Event classes `GEventAtom` and `GEventBin`

3.4.1 `GXXXEventAtom::model()`

3.4.2 `GXXXEventBin::model()`

3.5 Observation class `GObservation`

```
virtual void          clear(void) = 0;
virtual GObservation* clone(void) const = 0;
virtual GResponse*    response(void) const = 0;
virtual GPointing*    pointing(const GTime& time) const = 0;
virtual std::string   instrument(void) const = 0;
virtual std::string   print(void) const = 0;
virtual double        model(const GModels& models, const GEvent& event,
                            GVector* gradient = NULL) const;
virtual double        npred(const GModels& models, GVector* gradient = NULL) const;
```

3.5.1 `GXXXObservation::clear()`

3.5.2 `GXXXObservation::clone()`

3.5.3 `GXXXObservation::response()`

3.5.4 `GXXXObservation::pointing()`

3.5.5 `GXXXObservation::instrument()`

3.5.6 `GXXXObservation::print()`

3.5.7 `GXXXObservation::model()` **(optional)**

3.5.8 `GXXXObservation::npred()` **(optional)**

3.5.9 Members

`GObservation` implements the following protected members:

```
std::string m_name;         //!< Name of observation
std::string m_statistics;   //!< Optimizer statistics (default=poisson)
GEvents*    m_events;       //!< Pointer to event container
```

`m_name` is the name of the observation, and can be any arbitrary string of characters. The precise value of this string is not relevant.

`m_statistics` defines the statistics (`Poisson` or `Gaussian`) that should be used for parameter optimization.

`m_events` provides a pointer to the event container.

3.6 Source models

3.6.1 Factorized sky model (`GModelSky`)

A special type of source model is given by the factorisation

$$S(\vec{p}, E, t) = M(\vec{p}) \times P(E) \times V(t) \tag{6}$$

where

$M(\vec{p})$ is a map of the intensity distribution (in units of sr^{-1}),

$P(E)$ is the source spectrum (in units of photons cm^{-2} s^{-1} MeV^{-1}), and

$V(t)$ is a source modulation function (unitless).

In this case, Eq. (**??**) becomes

$$N(\vec{p'}, E', t'|\vec{d}) = \int_0^{t'+\Delta t} \int_{E'-\Delta E}^{\infty} \int_{\Omega} M(\vec{p}) \times P(E) \times V(t) \times L(\vec{d}, \vec{p}, E, t) \times A_{\mathrm{eff}}(\vec{d}, \vec{p}, E, t) \times PSF(\vec{p'}|\vec{d}, \vec{p}, E, t) \times E_{\mathrm{disp}}(E'|\vec{d}, \vec{p}, E, t) T_{\mathrm{disp}}(t'|\vec{d}, \vec{p}, E, t) \, \mathrm{d}\vec{p} \, \mathrm{d}E \, \mathrm{d}t \, . \tag{7}$$

3.6.2 Data model (`GModelData`)

3.7 Sky directions and maps, energies and times

3.7.1 `GSkyDir`

3.7.2 `GEnergy`

3.7.3 `GEbounds`

3.7.4 `GTime`

3.7.5 `GGti`

3.7.6 Sky maps

3.8 Other support classes

3.8.1 Numerical integration (`GIntegral`)

3.8.2 Numerical derivatives (`GDerivative`)

4 CTA interface

4.1 CTA response

The CTA response function depends on the following parameters

- the radial offset θ in the camera from the telescope pointing $\vec{d}$,
- the polar angle ϕ in the camera from the telescope pointing $\vec{d}$,
- the zenith angle Θ of the telescope pointing,
- the azimuth angle Φ of the telescope pointing,
- the optical efficiency $\epsilon_{\rm opt}$, and
- the array configuration C.

Furthermore, the angular separation between true and measued photon direction is denoted by δ. The relation between the angular parameters is illustrated in Fig. **??**.

Using these parameters, the CTA response function can be factorised as follows:

$$\begin{aligned} & R(\theta, E'|\theta, \phi, \Theta, \Phi, C, \epsilon_{\rm opt}, E) \\ & \quad = \quad A_{\rm eff}(\theta, \phi, \Theta, \Phi, C, \epsilon_{\rm opt}, E)\, PSF(\delta|\theta, \phi, \Theta, \Phi, C, \epsilon_{\rm opt}, E)\, E_{\rm disp}(E'|\theta, \phi, \Theta, \Phi, C, \epsilon_{\rm opt}, E) \end{aligned} \tag{8}$$

where

- $A_{\rm eff}(\theta, \phi, \Theta, \Phi, C, \epsilon_{\rm opt}, E)$ is the effective area (in units of counts photons^{-1} cm^2),

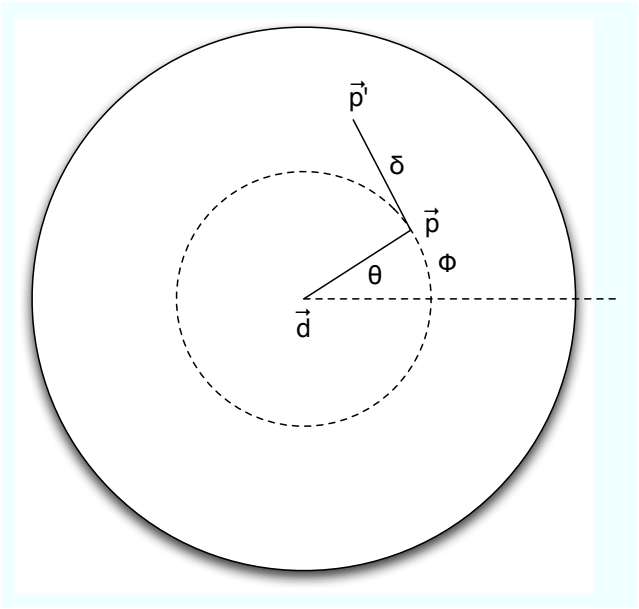

Figure 1: Relation between the pointing direction $\vec{d}$, the source position $\vec{p}$ the observed photon arrival direction $\vec{p'}$, the radial offset and polar angles θ and ϕ, and the angular separation δ between true and observed photon arrival direction. The large circle indicates the field of view, while the dashed circle indicates the line of constant offset angle θ. The dashed line indicates the direction for which $\phi = 0$.

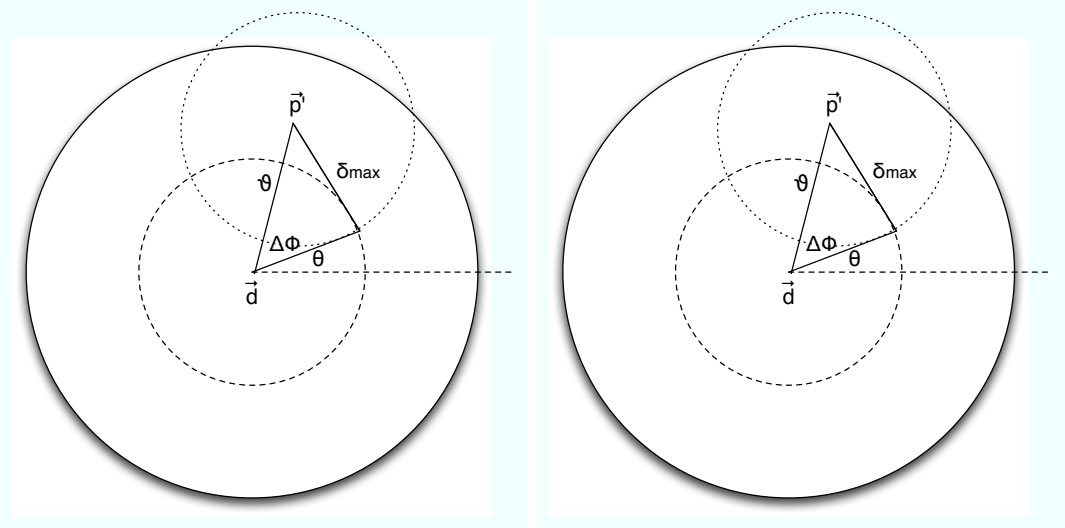

Figure 2: TBD.

- $PSF(\delta|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E)$ is the point spread function (in units of $\rm sr^{-1}$), and
- $E_{\rm disp}(E'|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E)$ is the energy dispersion (in units of $\rm MeV^{-1}$).

Note that

$$1 = 2\pi \int_0^{\pi} PSF(\delta|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E) \sin\delta \,{\rm d}\delta \tag{9}$$

and

$$1 = \int_0^{\infty} E_{\rm disp}(E'|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E) \,{\rm d}E' \tag{10}$$

Equation (**??**) can then be written as

$$\begin{aligned} & P_{\vec{p}}(\vec{p'},E',t'|\Theta,\Phi,C,\epsilon_{\rm opt},E) \\ &= \int_0^{\pi}\int_0^{2\pi} S(\theta,\phi,E,t)\, A_{\rm eff}\,(\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E) \\ &\times \; PSF(\delta|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E)\, E_{\rm disp}(E'|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E)\, \sin\theta \,{\rm d}\theta\,{\rm d}\phi \end{aligned} \tag{11}$$

where the source intensity is for convenience given as function of offset angle θ and polar angle ϕ around the pointing direction $\vec{d}$ (cf. Fig. **??**).

As $PSF(\delta|\theta,\phi,\Theta,\Phi,C,\epsilon_{\rm opt},E)$ falls strongly with increasing δ, we may define a maximum angle $\delta_{\rm max}$ beyond which the PSF drops to zero. Evaluation of Eq. (**??**) can thus be limited to a cicular region around $\vec{p'}$ with radius $\delta_{\rm max}$ (cf. left panel of Fig. **??**). The azimuthal integration is then limited to an interval $[\phi_0 - \Delta\phi, \phi_0 + \Delta\phi]$ around the position angle

$$\phi_0 = \arctan\left(\frac{\sin(\alpha_{\vec{p'}} - \alpha_{\vec{d}})}{\cos\delta_{\vec{d}} \times \tan\delta_{\vec{p'}} - \sin\delta_{\vec{d}} \times \cos(\alpha_{\vec{p'}} - \alpha_{\vec{d}})}\right) \tag{12}$$

of $\vec{p'}$ with respect to $\vec{d}$, where $(\alpha_{\vec{d}}, \delta_{\vec{d}})$ and $(\alpha_{\vec{p'}}, \delta_{\vec{p'}})$ are the Right Ascension and Declination of $\vec{d}$ and $\vec{p'}$, respectively. The half-width of the interval is given by

$$\Delta\phi = \begin{cases} 0 & \text{if } \vartheta = 0 \text{ and } \theta > \delta_{\rm max} \text{ or} \\ & \theta = 0 \text{ and } \vartheta > \delta_{\rm max} \text{ or} \\ & \vartheta \geq \theta + \delta_{\rm max} \\ \pi & \text{if } \vartheta = 0 \text{ and } \theta \leq \delta_{\rm max} \text{ or} \\ & \theta = 0 \text{ and } \vartheta \leq \delta_{\rm max} \text{ or} \\ & \theta \leq \delta_{\rm max} - \vartheta \\ \arccos\left(\dfrac{\cos\delta_{\rm max} - \cos\vartheta\cos\theta}{\sin\vartheta\sin\theta}\right) & \text{else} \end{cases} \tag{13}$$

where ϑ is the angular separation between $\vec{d}$ and $\vec{p'}$. Note that Eq. (??) is implemented by `GSkyDir::posang()` while Eq. (??) is implemented by the support function `cta_roi_arclength()` which returns $2\Delta\phi$ in units of radians.

Furthemore, if the source model is only non-zero within a circular region around $\vec{p}$, the integration can be further restricted to the overlapping region between the maximum PSF extent and this circular region.

Since only $S(\theta, \phi, E, t)$ and $PSF(\theta|\epsilon_{\rm opt}, \Theta, \Phi, \theta, C, E)$ depend on ϕ, we can separate the azimuthal integration from the offset angle integration. We thus integrate first the product of source model and PSF over all azimuth angles using

$$S_{\rm PSF}(\theta|\epsilon_{\rm opt}, \Theta, \Phi, C, E) = \int_0^{2\pi} S(\theta, \phi, E, t)\, PSF(\delta|\epsilon_{\rm opt}, \Theta, \Phi, \theta, C, E)\, {\rm d}\phi \tag{14}$$

and then perform the offset angle integration using

$$P_{\vec{p}}(\vec{p'}, E', t'|\epsilon_{\rm opt}, \Theta, \Phi, C, E) = \int_0^{\pi} S_{\rm PSF}(\theta|\epsilon_{\rm opt}, \Theta, \Phi, C, E)\, A_{\rm eff}(\epsilon_{\rm opt}, \Theta, \Phi, \theta, C, E)\, E_{\rm disp}(E'|\epsilon_{\rm opt}, \Theta, \Phi, \theta, C, E)\, \sin\theta\, {\rm d}\theta \tag{15}$$

Equation (??) can then be written as

$$P(\vec{p'}, E', t'|\vec{d}) = \int_{E'-\Delta E}^{\infty} \int_{\Omega} S(\vec{p}, E, t')\, A_{\rm eff}(\vec{d}, \vec{p}, E, t')\, PSF(\vec{p'}|\vec{d}, \vec{p}, E)\, E_{\rm disp}(E'|\vec{d}, \vec{p}, E)\, {\rm d}\vec{p}\, {\rm d}E \tag{16}$$

Since for many astrophysical sources $S(\vec{p}, E, t)$ drops quickly with energy, only photons near E' contribute effectively to the integral. For this reason, energy dispersion can also often be neglected, and $E_{\rm disp}(E'|\vec{d}, \vec{p}, E)$ can be approximated by a δ-function, leading to

$$P(\vec{p'}, E', t'|\vec{d}) = \int_{\Omega} S(\vec{p}, E', t')\, A_{\rm eff}(\vec{d}, \vec{p}, E', t)\, PSF(\vec{p'}|\vec{d}, \vec{p}, E')\, {\rm d}\vec{p}. \tag{17}$$

These simplifications may be considered when implementing `GResponse::irf()` for a specific instrument.

the computation of $P(\vec{p'}, E', t'|\vec{d})$ for a specific instrument.

Using $N(\vec{p'}, E', t'|\vec{d})$ given by Eq. (??), $N_{\rm pred}$, the total number of predicted events, can be written as

$$N_{\rm pred} = \int_0^{\infty} \int_0^{\infty} \int_{\Omega} S(\vec{p}, E, t)\, L(\vec{d}, \vec{p}, E, t)\, A_{\rm eff}(\vec{d}, \vec{p}, E, t)\, N_{\rm PSF}(\vec{d}, \vec{p}, E, t)\, N_{\rm Edisp}(\vec{d}, \vec{p}, E, t)\, N_{\rm Tdisp}(\vec{d}, \vec{p}, E, t)\, {\rm d}\vec{p}\, {\rm d}E\, {\rm d}t \tag{18}$$

where the three integrals are to be taken over all possible photon arrival times t, true energies E, and arrivial directions $\vec{p}$. The integrations over the data space selections have been compactly written as

$$N_{\rm PSF}(\vec{d}, \vec{p}, E, t) = \int_{\rm ROI} PSF(\vec{p'}|\vec{d}, \vec{p}, E, t)\, {\rm d}\vec{p'} \tag{19}$$

$$N_{\rm Edisp}(\vec{d},\vec{p},E,t) = \int_{E_{\rm bounds}} E_{\rm disp}(E'|\vec{d},\vec{p},E,t)\,{\rm d}E' \tag{20}$$

$$N_{\rm Tdisp}(\vec{d},\vec{p},E,t) = \int_{\rm GTI} T_{\rm disp}(t'|\vec{d},\vec{p},E,t)\,{\rm d}t' \,. \tag{21}$$

By combining all terms related to the instrument response function in

$$N_{\rm R}(\vec{d},\vec{p},E,t) = L(\vec{d},\vec{p},E,t)\,A_{\rm eff}(\vec{d},\vec{p},E,t)\,N_{\rm PSF}(\vec{d},\vec{p},E,t)\,N_{\rm Edisp}(\vec{d},\vec{p},E,t)\,N_{\rm Tdisp}(\vec{d},\vec{p},E,t) \tag{22}$$

Eq. (**??**) simplifies to

$$N_{\rm pred} = \int_0^\infty \int_0^\infty \int_\Omega S(\vec{p},E,t)\,N_{\rm R}(\vec{d},\vec{p},E,t)\,{\rm d}\vec{p}{\rm d}E\;{\rm d}t \tag{23}$$