GammaLib - Action #998

Introduce classes and readers for a cube-style IRF format

11/20/2013 02:48 PM - Mayer Michael

Status:	Closed	**Start date:**	11/20/2013
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	00-08-00		

Description

In order to analyse sources with different morphologies (from point-like to extended), we decided on a preliminary format for the instrument response which can be read by gammalib. The idea is to pass run-wise files to gammalib, which contain the effective area and the PSF (see example files attached).
The file format for the effective area is similar to the format of a GModelSpatialDiffuseCube. It can be stored in instrument coordinates (for unbinned analyses) or in sky coordinates (for binned analyses). The third axis of the cube is the energy.
The file for the PSF should be a binary table storing every required parameter of the PSF parametrisation as a function of instrument coordinates (for unbinned analyses) or sky coordinates (for binned analyses) and energy.
The goal is to read in these files, store their content in classes and access the values within GCTAResponse.

History

#1 - 11/20/2013 02:54 PM - Mayer Michael

The first thing we have to think about is wether we want to use the current base classes GCTAPsf and GCTAAeff. Their operator() takes the arguments lnE, theta, phi, zenith and azimuth. Do we want to use this base class but instead querying the return values of the derived classes e.g. with operator(double lnE, double x, double y)? This approach would simplify the code in GCTAResponse, I guess.
Furthermore, as far as I can see, the GCTAResponse::irf_*() and GCTAResponse::npred_*() - methods would have to be extended to allow the handling of the new classes.

#2 - 11/21/2013 10:08 AM - Knödlseder Jürgen

Mayer Michael wrote:

> The first thing we have to think about is wether we want to use the current base classes GCTAPsf and GCTAAeff. Their operator() takes the arguments lnE, theta, phi, zenith and azimuth. Do we want to use this base class but instead querying the return values of the derived classes e.g. with operator(double lnE, double x, double y)? This approach would simplify the code in GCTAResponse, I guess.
> Furthermore, as far as I can see, the GCTAResponse::irf_*() and GCTAResponse::npred_*() - methods would have to be extended to allow the handling of the new classes.

I think it should be able to read the new response using the operator()(lnE, theta, phi, zenith, azimuth), which uses the instrument system theta and phi. Then, no modifications to GCTAResponse::irf_*() and GCTAResponse::npred_*() are needed.

If you thing we cannot use the operator()(lnE, theta, phi, zenith, azimuth) interface we should discuss what exactly is needed. The goal was to have a general interface through which we can access all response formats.

#3 - 11/21/2013 10:15 AM - Mayer Michael

Alright great. I was just thinking way too complicated. Then I assume we just need the correct reader and a simple conversion from (det_x, det_y) to (theta, phi).

#4 - 11/21/2013 12:40 PM - Knödlseder Jürgen

I hope this is a trivial transformation. We can use the WCS classes for this. Do you have a document describing how DET_X and DET_Y are defined? GammaLib works internally always with spherical coordinates ...

#5 - 11/21/2013 01:22 PM - Mayer Michael

In H.E.S.S. the camera coordinates (nominal system) DET_X and DET_Y are defined in radians. Positive DET_X corresponds to positive altitude while positive DET_Y corresponds to positive azimuth. I don't know how this is handled elsewhere. There is a PhD thesis by Stefan Gillessen (in German) where all the coordinate system in H.E.S.S. are described (Chapter 4 at http://www.mpe.mpg.de/~ste/data/phd.pdf).

Do you think we would need to have something like a GCTAResponseTable3D to store and handle the new formats?

#6 - 11/21/2013 02:11 PM - Knödlseder Jürgen

Mayer Michael wrote:

> In H.E.S.S. the camera coordinates (nominal system) DET_X and DET_Y are defined in radians. Positive DET_X corresponds to positive altitude while positive DET_Y corresponds to positive azimuth. I don't know how this is handled elsewhere. There is a PhD thesis by Stefan Gillessen (in German) where all the coordinate system in H.E.S.S. are described (Chapter 4 at http://www.mpe.mpg.de/~ste/data/phd.pdf).

So it's a spherical system? (with X and Y I was assuming it were a cartesian system)

> Do you think we would need to have something like a GCTAResponseTable3D to store and handle the new formats?

I had not yet time to look into your files. Will do so asap.

#7 - 11/21/2013 02:42 PM - Mayer Michael

> So it's a spherical system? (with X and Y I was assuming it were a cartesian system)

No, it should be cartesian. I just wanted to point out the direction of the axes :). I am most certainly not an expert, but I think we would just need to convert from cartesian to polar coordinates there.

> I had not yet time to look into your files. Will do so asap.

These are the same files as we have discussed since the coding sprint in Toulouse. We would probably need the possibility to do 3D-interpolations of the IRFs (in theta, phi, energy)

#8 - 11/21/2013 03:09 PM - Knödlseder Jürgen

Mayer Michael wrote:

> So it's a spherical system? (with X and Y I was assuming it were a cartesian system)

No, it should be cartesian. I just wanted to point out the direction of the axes :). I am most certainly not an expert, but I think we would just need to convert from cartesian to polar coordinates there.

So I guess we may use GWcsCAR for this.

> I had not yet time to look into your files. Will do so asap.

> These are the same files as we have discussed since the coding sprint in Toulouse. We would probably need the possibility to do 3D-interpolations of the IRFs (in theta, phi, energy)

GSkymap allows already 2D. We can combine this with GNodeArray for getting the energy dimension. Should be straight forward.

#9 - 11/21/2013 03:15 PM - Mayer Michael

ok great. We could introduce a new class GCTAAeff3D and extend GCTAPsfKing. We might use the feature-specific discussion threads then - and close this one?

#10 - 11/21/2013 03:23 PM - Knödlseder Jürgen

I'm just wondering whether we should stick to 3D. Is there a chance that we'll expand this format in higher dimensions (e.g. event class as 4th dimension)? Would Cube be a better attribute for this, e.g. GCTAAeffCube?

Is PSF the same basic format (x,y,logE)? Is GCTAPsfCube a good name?

#11 - 11/21/2013 03:33 PM - Mayer Michael

Agreed. The effective area is should be named GCTAAeffCube.
For the PSF it is a bit more complex since we store the parametrisation of the PSF, i.e. the parameters "Gamma" and "Sigma" in case of the king profile in each cube pixel. Therefore, we would have two cubes for the GCTAPsfKing (one holding "Gamma" one holding "Sigma" as function of x, y, logE). We could still create GCTAPsfCube and make GCTAPsfKing a derived class.

#12 - 11/21/2013 03:41 PM - Knödlseder Jürgen

Ok. I was mainly thinking about the question of whether we can develop a generic class for data access that can support different implementations in the end. We may indeed handle this using a base class for the structure and a derived class for the implementation of the specific formula. Not sure that we need it in a first place (e.g. if we want to study right now many different PSF parametrisations). It's maybe easier to start with hard-coding the formulae we have in mind in the same class (maybe one method per formula)? We can always re-factor the classes later if needed.

#13 - 11/21/2013 04:11 PM - Mayer Michael

> It's maybe easier to start with hard-coding the formulae we have in mind in the same class (maybe one method per formula)? We can always re-factor the classes later if needed.

If we use a specific file format which stores parameters for a specific parametrisation, I can't think of a case where several formulae would apply. So for now, I assume we could stay with GCTAPsfKing which has its own read() method for this specific format. If we want to have e.g. triple Gaussian (as sometimes used in HESS), we need another format with more cubes for the several parameters. If we want to have something like this, I agree, we can still refactor later. So I will start with GCTAAeffCube and extend GCTAPsfKing very soon.

#14 - 11/21/2013 04:28 PM - Knödlseder Jürgen

Well, you could have a cube format, and then a specific set of cubes is provided for a given formula.

#15 - 11/21/2013 05:26 PM - Mayer Michael

Ok, you mean to have the possibility to use several PSF parameterisations within one class? That means reading in all available cubes and the class is able to determine which formula to take (e.g. having "Gamma" and "Sigma" signals to use the king profile, etc.)

#16 - 11/21/2013 05:29 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Ok, you mean to have the possibility to use several PSF parameterisations within one class? That means reading in all available cubes and the class is able to determine which formula to take (e.g. having "Gamma" and "Sigma" signals to use the king profile, etc.)

Right. But at the end this probably means that we need a generic GCTAResponseCube class that is then used by the various response classes at the end ...

#17 - 11/21/2013 05:39 PM - Mayer Michael

Ok right, I understand. The effective area is currently stored in an image, other than the PSF which is stored in a bintable. We probably should switch to a bintable for the effective area too. Then GCTAResponseCube should be a member of e.g. GCTAAeffCube or GCTAPsfCube. This class would be analogous to GCTAResponseTable in e.g. GCTAAeff2D right?

#18 - 11/21/2013 05:43 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Ok right, I understand. The effective area is currently stored in an image, other than the PSF which is stored in a bintable. We probably should switch to a bintable for the effective area too. Then GCTAResponseCube should be a member of e.g. GCTAAeffCube or GCTAPsfCube. This class would be analogous to GCTAResponseTable in e.g. GCTAAeff2D right?

Right, GCTAResponseCube would be analogous to GCTAResponseTable.

But now I'm confused. You say that we should switch to a bintable for the effective area. This would then by what is implemented by GCTAResponseTable which supports n-dimensional structures (at least in principle; would need to add operators for >2D access).

#19 - 11/22/2013 09:15 AM - Mayer Michael

Well, the PSF parameters are already stored in a bintable. There, in the "PSF" extension, we have 3D arrays for every parameter (stored as function of x,y,lnE). Probably, we have to differentiate between instrument coordinates and sky coordinates. For instrument coordinates, the bintable should be fine while I guess for sky coordinates, we would need an image, right?

If the functionality to read the bintable is already there, we could stick to instrument coordinates first and use the same reader for the effective area. The reason why the effective area is currently stored in an image is that you can easily check the files with DS9. Moving it to a bintable and instrument coordinates would be no problem, I guess.

As far as I can see, we have to decide wether we extend GCTAResponseTable for 3D access and interpolation or wether we introduce GCTAResponseCube for this.

#20 - 11/29/2013 04:08 PM - Mayer Michael

Mayer Michael wrote:

> As far as I can see, we have to decide wether we extend GCTAResponseTable for 3D access and interpolation or wether we introduce GCTAResponseCube for this.

I had a deeper look into GCTAResponsTable and I think that all functionality to read the bintable is already there. We could easily add the "*_LO" and "*_HI" columns to the IRF-files to be compliant with this type of table. I could start to try implementing 3D (or even higher dimension) access and interpolation. Should I make a new issue for this?

#21 - 11/29/2013 04:30 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Well, the PSF parameters are already stored in a bintable. There, in the "PSF" extension, we have 3D arrays for every parameter (stored as function of x,y,lnE). Probably, we have to differentiate between instrument coordinates and sky coordinates. For instrument coordinates, the bintable should be fine while I guess for sky coordinates, we would need an image, right?

The storage for a bintable or an image is at the end the same (just an array of double precision values), the main difference being that a FITS image is associated to world coordinates (through specific header keywords) while a bintable is only "indexed". Hence if we want to use sky coordinates (or any spherical coordinates), we need to encode the world coordinate information somehow. For an image this is done automatically (using the CRVAL etc keywords), for a bintable we would have to define our own format.

> If the functionality to read the bintable is already there, we could stick to instrument coordinates first and use the same reader for the effective area. The reason why the effective area is currently stored in an image is that you can easily check the files with DS9. Moving it to a bintable and instrument coordinates would be no problem, I guess.

Since we want to have "images" of response parameters at the end, maybe it's indeed better to move to a new GCTAResponseCube structure. The thing is that x and y are somehow different from the other parameters, as x and y are world coordinates (sky coordinates or instrument coordinates), while the other dimensions (e.g. logE) are dependencies. x and y need spherical coordinate transformations, while the other dimensions simply need linear interpolations in the specific parameter. Specifying _LO and _HI for x and y won't make much sense, this would probably just be pixel indices ...

> As far as I can see, we have to decide wether we extend GCTAResponseTable for 3D access and interpolation or wether we introduce GCTAResponseCube for this.

#22 - 11/30/2013 03:08 PM - Mayer Michael

Knödlseder Jürgen wrote:

> The storage for a bintable or an image is at the end the same (just an array of double precision values), the main difference being that a FITS image is associated to world coordinates (through specific header keywords) while a bintable is only "indexed". Hence if we want to use sky coordinates (or any spherical coordinates), we need to encode the world coordinate information somehow. For an image this is done automatically (using the CRVAL etc keywords), for a bintable we would have to define our own format.

Thanks for clarifying that. I agree, for storing the instrument response in sky coordinates, we need the WCS information.
However, which coordinate system would you use for the instrument coordinates? In gammalib there is "EQU", "GAL", "ECL" and "SGL" available. The instrument coordinate system has its origin in the camera center and is not connected to one of the available systems. Therefore, I am a bit unsure how GSkymap should deal with the instrument coordinates (probably I also don't really understand the coordinate systems correctly smile.png)

#23 - 11/30/2013 06:50 PM - Knödlseder Jürgen

I mean in fact projection. You said that the DETX DETY coordinates were cartesian, hence a cartesian projection, implemented by GWcsCAR would be appropriate. But we don't have to worry about which projection to use as GSkymap handles them transparently.

#24 - 11/30/2013 07:00 PM - Mayer Michael

I fully agree to use GWcsCar. However, I am asking more specifically about the image header keywords CTYPE1 and CTYPE2. As I follow the code, in GWcsLib::read an exception is thrown, when none of the known systems is given. Or is the choice of the system not important at this point?

#25 - 12/01/2013 09:43 PM - Knödlseder Jürgen

Indeed, I forgot about the coordinate system part. So one would need to artificially associate instrument coordinates with, for example, equatorial coordinates. This is definitely not very nice and potentially confusing.

The problem is that GSkymap are tied to world coordinates, which is logical in the sense that the sky map represents the sky (hence necessarily a sky system). This means that in our case we can not use GSkymap for storing the data. We may in fact use the GFitsImage class directly to access the pixel values, and use GWcsCAR to perform transformation from offset and azimuth angle to pixel coordinates. The only thing is that we need to re-implement the interpolation in pixel coordinates that is already available in GSkymap.

#26 - 12/01/2013 11:03 PM - Knödlseder Jürgen

... since we cannot simply re-use GSkymap for our purpose I was wondering whether we may not go back using GCTAResponseTable and invent our own header keywords that define the projection (as we cannot use WCS standard for this anyways).

#27 - 12/02/2013 05:03 PM - Mayer Michael

I completely agree to go back to GCTAResponseTable. Since the idea of the format with the GSkymap was mainly to have e.g. exposure cubes for binned analyses (where events in one spatial bin might come from different zenith angles) I guess for unbinned analyses, GCTAResponseTable is the best option to use.
After closely following the source code of GCTAResponseTable, I think this class has great potential to host all IRFs for unbinned analyses. Therefore, I would even propose to stay with e.g. GCTAAeff2D and GCTAPsf2D using the predefined format and see where this takes us. If the

results show that we need one more dimension (e.g. the zenith angle) it should be very easy to extend this within GCTAResponseTable as far as I can see.
Admittedly, I think this is what you proposed in the first place.

#28 - 12/03/2013 02:22 AM - Knödlseder Jürgen

Mayer Michael wrote:

> I completely agree to go back to GCTAResponseTable. Since the idea of the format with the GSkymap was mainly to have e.g. exposure cubes for binned analyses (where events in one spatial bin might come from different zenith angles) I guess for unbinned analyses, GCTAResponseTable is the best option to use.
> After closely following the source code of GCTAResponseTable, I think this class has great potential to host all IRFs for unbinned analyses. Therefore, I would even propose to stay with e.g. GCTAAeff2D and GCTAPsf2D using the predefined format and see where this takes us. If the results show that we need one more dimension (e.g. the zenith angle) it should be very easy to extend this within GCTAResponseTable as far as I can see.
> Admittedly, I think this is what you proposed in the first place.

Indeed, the GCTAResponseTable class has been created with flexibility in mind. So it should be possible to extent the class easily. I was even thinking to promote the class to the GammaLib core (as GResponseTable in the obs module), because also the LAT interface uses this class.

#29 - 12/03/2013 01:47 PM - Mayer Michael

Yes, that would indeed avoid code duplication. However, the GLATResponseTable seems to be very LAT specific. One would probably need to adapt some code in GLATResponse as well.
I will use the GCTAResponseTable for the implementation of GCTAPsfKing. When we change it to GResponseTable, it will be easily adjustable.

#30 - 12/03/2013 01:58 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Yes, that would indeed avoid code duplication. However, the GLATResponseTable seems to be very LAT specific. One would probably need to adapt some code in GLATResponse as well.
> I will use the GCTAResponseTable for the implementation of GCTAPsfKing. When we change it to GResponseTable, it will be easily adjustable.

Agree. There is no need to generalise now.

#31 - 01/08/2014 09:27 PM - Knödlseder Jürgen

Looks like this is now covered/done by #753? (Introduce GCTAPsfKing class). Michael, can you confirm?

#32 - 01/08/2014 09:45 PM - Mayer Michael

Confirmed. GCTAPsfKing and GCTAAeff2D provide all functionalities to conduct IACT analyses with sufficient precision (and level of detail).

#33 - 01/08/2014 10:31 PM - Knödlseder Jürgen

- Status changed from New to Closed

- % Done changed from 0 to 100

- Remaining (hours) set to 0.0

Okay, time to close the issue smile.png

Files

expcube_23544.fits	135 KB	11/20/2013	Mayer Michael
psf_23544.fits	135 KB	11/20/2013	Mayer Michael