What I understand, or don't, about back-projection

M. Sauvage

Document change record		
Version	Date	Changes
D0.1	15/12/2010	Creation of the first draft
D0.2	16/12/2010	Incorporate first round of comments
D0.3	14/01/2011	Incorporate a description of MadMap's and Scanamorphos' mapmaking codes Examine the case of a wrong projection model
1.0	14/01/2011	First release

Reference documents			
Ref.	ID	Version	Title
RD1	PICC-NHSC-TR-028	2.0	PACS photometric check: MADmap vs. PhotProject
RD2	A&A Submitted		Scanamorphos: a map-making software for Herschel and similar bolometer arrays used in scan mode

Contents

1 Introduction

In light of the investigation on our possible discrepancy regarding extended source fluxes with other instruments operating at the same wavelengths, I investigate how we back-project our data into the sky. First I call projection the operation that consists in going from the sky to the observation, and back-projection the reverse operation, i.e. making the map from the detector samples. This is rather non-standard for observers, but in the perspective of mapmaking, we first have to model the signal acquisition phase, and then try to invert it. Therefore in that perspective, it makes sense to call this first operation "the projection". Furthermore it is also only in that direction that the mathematics of the operation can be written straight-forwardly. This document thus first puts in writing what I understand about the issues of projecting and back-projecting. With the help of many in the ICC, I have also looked at how back-projection is handled in the different codes that we use.

To summarize my conclusions (some of which are evident to you already):

1. Implementation of the back-projection cannot be blamed for a photometric discrepancy with other instrument.
2. `photProject` unnecessarily, and with no mathematical justification, introduces an extra-correction factor, called the active pixel fraction, f_{act} or the Okumura-Gastaud constant, in the back-projection equation that has the potential to create photometric catastrophes.
3. Such catastrophes do no occur yet because (1) we calibrate our reponsivity using maps made with `photProject` and thus introduce f_{act} in the response, which implies that level 1 data are expressed in Jy per ideal detector pixel (3.2 or 6.4 arcsec), and (2) alternate map-making softwares have so far ignored the geometry of our pixels as described in the calibration files either because they bypass the back-projection (MadMap) or do not have access to the files (Scanamorphos). This is likely a temporary situation.

In my opinion, the last two points require that we take action, either by removing f_{act} from our calibration and map-making scheme (which I would prefer) or by identifying how we can make sure that this information is extremely visible to our future users.

In a second part of this note, I investigate the consequences of a wrong assumption on the geometry of our detector (essentially a wrong assumption on the detector pixel size), to find that interestingly there are none because we have used a flux-conserving sky map estimation algorithm rather than a surface brightness conserving one. This part may be of interest to the reader who wants to know more about the inner workings of the map estimation algorithm.

2 Projection and back-projection: the art of map-making

In order to write down some equations, mapmakers usually resort to matrices representations. In these representations, maps and cubes are vectorialized (i.e. the pixels or samples are re-organized into 1D objects), and in the equations below they are represented by boldface characters. In this representation, if we call $\boldsymbol{x}$ the sky, and $\boldsymbol{y}$ the cube, they are related with:

$$\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{n}$ is the noise, considered as additive and that I will ignore throughout this note, and $\boldsymbol{P}$ is the key element, namely the projection matrix. Figure 1 lets you visualize this equation. The very important point is that to compute the elements of $\boldsymbol{P}$ we must have an accurate description of the detector effective geometry (i.e. the geometry of the detector pixels). This point is where some confusion may first be introduced, or where I'm most confused, thus I will dwell on it slightly.

We have a relatively good description of the physical geometry of our detectors (If we pretend to be able to make silicon grids of a few μm wide we'd better). We know that the full width of a pixel (including its walls) is 750 μm, while its inner width is 640 μm. The angular size of 3.2" that we quote for the blue pixels corresponds to the full width of the pixels. The ratio of the inner to the full surface of our pixel is what we

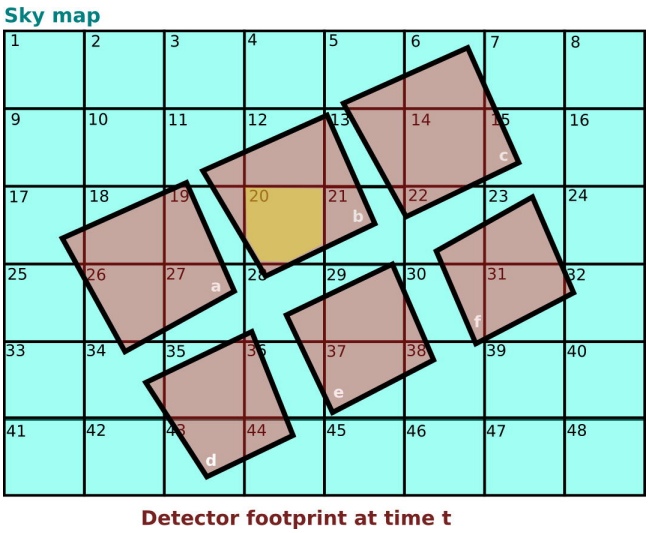

Figure 1: A sketch illustrating the relation between the sky and the detector timelines. The sky is represented by a fully sampled regular grid map, and I overlay the location of some of the detector pixel at a given sampling time t. I may have vastly exaggerated the optical distortion as well as the spacing between the detector pixels. In principle, the equations presented in this document are valid even if the detector pixels were undistorted and fully covered the sky. I've also placed myself in the more favorable (and recommended) situation where the sky map pixels are smaller than the detector pixels. In this figure I have numbered the sky pixels from 1 to 48 and the detector pixels from a to f. The solid angle in yellow is the overlap solid angle between pixel b of the detector and pixel 20 of the map at time t. The yellow solid angle is thus $P_{b,20}$.

call in the "active fraction" of the pixel (`activeFraction` in our projection code and in the metadata of the `subArrayArray` calibration file)[1], and is equal to 0.73 (**important note**: this is a ratio of surfaces). In our calibrated description of the array geometry, the size of the pixels is their inner size, thus indeed they do not project as jointed pixels on the sky.

Furthermore, we have the possibility to decide that the sensitive area of a pixel is not equal to its geometrical area, which is what the drizzling technique implements. This is a choice, and using a much smaller area can be interesting when one wants to minimize the amount of correlated noise between the pixels of the reconstructed map. We refer to this as the "pixel fraction" (or `pixFrac`). Note that for unknown reason we consider that the pixel fraction is the ratio of the width of the sensitive area to the width of the inner pixel, rather than a ratio of surfaces.

The bottom line however is that we have to have a self-consistent description of the detector: we measure the actual position of our pixels' centers, then compute the actual dimensions for each pixel, making use of the "active fraction" and the "pixel fraction". These dimensions are used to build $\boldsymbol{P}$. Thus adjusting our projection equations to reflect a change in the value of the "active" or "pixel" fraction cannot simply be done by applying some multiplicative factor on $\boldsymbol{y}$. The complete matrix $\boldsymbol{P}$ also has to be recomputed. In our current implementation of the projection scheme we adopt the geometry of the inner surface of the pixels (i.e. we have included the notion of the "active fraction" in our description of the detector). Thus Figure 1 is a plausible representation of our situation. By default, we also work with a pixel fraction of 1, but the code accepts other values, and correctly handles their impact on the detector geometry and thus on $\boldsymbol{P}$.

Of course when using equation (1), we have to make sure we get the units right. Although the unit of choice for the PACS sky maps is Jy/pixel[2], I will first reason with a sky map that has units of surface brightness in its pixels, i.e. Jy/Ω (Ω being a solid angle), and I will examine the other case later. In the Jy/Ω case the elements of the matrix $\boldsymbol{P}$ are the yellow areas of Figure 1 expressed as well in solid angle units, and equation (1) implies that

[1] familiarly the Okumura-Gastaud constant.
[2] which is simpler for photometry for instance, and can be seen as having deeper reasons.

the unit in the cube is Jy in detector pixel (effectively the inner surface of the pixels times the square of the pixel fraction). This is fortunate because we know that the bolometer signal is proportional to the incoming power, which can be computed from the spectral flux density, the spectral convention, and the system transmission, all these terms can be concatenated into the color correction, so in fact we can derive a responsivity factor to convert Volts per detector pixel to Jy per detector pixel using the sky map and equation (1).

As an illustration based on Figure 1 we can write that the number of Jy falling into pixel b, that I will call y_b is given by:

$$y_b(t) = P_{b,11}\times x_{11} + P_{b,12}\times x_{12} + P_{b,13}\times x_{13} + P_{b,19}\times x_{19} + P_{b,20}\times x_{20} + P_{b,21}\times x_{21} + P_{b,28}\times x_{28}, \quad (2)$$

all the other $P_{b,n}$ coefficients of the projection matrix being null. I recall that in this equation, x_j is in surface brightness units (Jy/Ω) and the $P_{i,j}$ coefficients are solid angles.

Now the big issue is to actually reconstruct the map, i.e. invert equation (1). In general the matrix is not inverted as it is a tremendous task and typically an approximate solution is built. Still using Figure 1 we can follow the typical way of getting to that approximate solution. In fact the term "approximate solution" is incorrect as in no way does the reconstructed map satisfies equation (1). It really is an estimate of the map. We can use the flux density values that are recorded in the detector pixels to estimate the surface brightness of each map pixel. For instance, the surface brightness in map pixel 20 can be estimated from the flux density value recorded in detector pixel b at time t assuming that the surface brightness is constant on the area that encompasses those two pixels:

$$x_{20}^{est}(t) = y_b(t)/S_b \quad (3)$$

where S_b is the solid angle subtended by detector pixel b, and the t in parenthesis on the left size simply states that it is the estimate of pixel 20 that can be made from readout at time t. Obviously we have many measurements in the $\boldsymbol{y}$ vector that will let us estimate the value of x_{20}, and we want to use them all to get our best estimate of x_{20}. However it is quite clear that we should do a weighted average because not all the detector pixels that intercept the map pixel of interest should be treated equally. Consider for instance in Figure 1 how small the intercept of pixel b is with map pixel 28. One possible weighting scheme is to weight by the ratio of the intercepted solid angle and the map pixel solid angle. In the example above, the weight would be $P_{b,20}/S_{20}$ (S_{20} is the map pixel solid angle, i.e the weight is 1 when the map pixel is fully covered).

Trying to write that in an equation would give:

$$x_{20}^{est} = \frac{\sum_{i,i\cap 20\neq 0} (P_{i,20}/S_{20}) \times y_i/S_i}{\sum_{i,i\cap 20\neq 0} (P_{i,20}/S_{20})}, \quad (4)$$

where the summation is on the detector pixels that have a non-zero intersection with the map pixel. We see that the surface of the sky map pixel disappears and we can rewrite this equation as:

$$x_{20}^{est} = \frac{\sum_{i,i\cap 20\neq 0} (P_{i,20} \times y_i)/S_i}{\sum_{i,i\cap 20\neq 0} (P_{i,20})}, \quad (5)$$

Now given that detector pixels that have a null intersection with the map pixel of interest have a corresponding component in the projection matrix equal to 0, the summation can be extended to all detector pixels:

$$x_{20}^{est} = \frac{\sum_{i} (P_{i,20} \times y_i)/S_i}{\sum_{i} (P_{i,20})}, \quad (6)$$

which leads to the more condensed representation:

$$\boldsymbol{x}_{PhP} = \frac{\boldsymbol{P}^T(\boldsymbol{y}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^T\boldsymbol{1}}, \quad (7)$$

where the label PhP refers to the PACS IA task `photProject` as I assume it is performing its estimation of the map in a very similar way (see section 3.1), and the division of vectors $\boldsymbol{y}/\boldsymbol{S}_{DetPix}$ is supposed to be made element per element. In equation (7) $\boldsymbol{S}_{DetPix}$ is not a constant: it is the actual solid angle of each of the detector pixels. This is known because it is the same surface that was used to compute the component of matrix $\boldsymbol{P}$.

Finally one can check that equation (7) is homogenous: $\boldsymbol{y}$ is in Jy, $\boldsymbol{P}$ and $\boldsymbol{S}_{DetPix}$ are solid angles, and $\boldsymbol{x}_{PhP}$ is in Jy per solid angle.

I now look at the alternative where we want to express the map in Jy/pixel. In that case I don't want to change the unit of y because in the detector space we essentially do not know what the pixels are and we should treat them simply as detectors, i.e. use units of Jy per detector. In that case we have:

$$\boldsymbol{y} = \boldsymbol{P'}\boldsymbol{x'} + \boldsymbol{n}, \tag{8}$$

where the prime is there to remind me of the change in sky map units. In that case the coefficients of $\boldsymbol{P'}$ are no longer the solid angle subtended by the the intersection between detector pixels and sky map pixels, but the ratio of this solid angle to the map pixel solid angle. Therefore:

$$P'_{i,j} = P_{i,j}/S_j, \tag{9}$$

where the i index runs in the detector space and the j index runs in the sky map space, and actually $S_j = S_{Sky}$ since the sky map should have regular pixels[3].

The same method can be then used to estimate the sky map from the detector samples. We make the same assumption, namely that the sky brightness is locally constant on an area encompassing the considered detector and sky pixels. In that case each detector pixel that has a non-zero intersection with the map pixel of interest provides an estimation of the flux in the map pixel as:

$$x'^{est}_j(t) = y_i(t) \times \frac{S_{Sky}}{S_i}, \tag{10}$$

where t is simply there to state that at time t pixel i of the $\boldsymbol{y}$ vector has an overlap with pixel j of the $\boldsymbol{x'}$ sky map. Again we can average all the measurements that provide an estimation of x'_j using a weighted scheme. The weight can obviously be related to how well the map pixel is covered by the detector pixel, which is completely captured by the corresponding coefficient of the projection matrix $\boldsymbol{P'}$. Thus in that case our estimation of a map pixel becomes:

$$x'^{est}_j = \frac{\sum_i P'_{i,j} \times y_i \times \frac{S_{Sky}}{S_i}}{\sum_i (P'_{i,j})}. \tag{11}$$

Summation can indeed be extended to all indices i of vector $\boldsymbol{y}$ as $P'_{i,j}$ is zero when the intersection between the detector pixel and the map pixel is null. Thus we can condense this in matrix form as:

$$\boldsymbol{x}'_{PhP} = S_{Sky} \times \frac{\boldsymbol{P}^{T'}(\boldsymbol{y}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^{T'}\boldsymbol{1}} \tag{12}$$

Since the sky pixel solid angle is essentially constant and appears at the denominator of every coefficient of $\boldsymbol{P'}$, we can factorize it and express $\boldsymbol{x}'_{PhP}$ as a function of projection matrix P. This gives:

$$\boldsymbol{x}'_{PhP} = S_{Sky} \times \frac{\boldsymbol{P}^{T}(\boldsymbol{y}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^{T}\boldsymbol{1}} = S_{Sky} \times \boldsymbol{x}_{PhP}, \tag{13}$$

Which is indeed expected.

3 Code checks

With the help of Hervé and Michael, we have looked into the code that is implemented to create maps from level 1 products.

[3]Strictly speaking, we cannot assume $S_j = S_{Sky}$: on a 10 square degree map, a relative variation of 10^{-3} is expected in the solid angle of sky pixels for the tangential projection we are using.

3.1 photProject

Reconstructing the algorithms used from the code shows first that the construction of matrix $\boldsymbol{P'}$ is fully consistent with the calibrated description of the detector geometry (position of pixel centers, location of pixel corners assuming their size is the inner pixel size) and with the chosen value of the pixel fraction. In other words, $\boldsymbol{P'}$ is built from the measured position of the pixel centers, using 640 μm as their effective with, and if necessary adapting that width with the value of the pixel fraction `pixFrac`[4].

However, the map estimation method does *not* implement equation (12) but a slightly modified one:

$$\boldsymbol{x}'_{PhP} = S_{Sky} \times f_{act} \times \frac{\boldsymbol{P}^{T'}(\boldsymbol{y}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^{T'}\boldsymbol{1}}, \tag{14}$$

where f_{act} is the active fraction or the Okumura-Gastaud constant. Essentially, this is wrong, or, to put it more diplomatically, there is no justification for this in the mathematics of projection. This error cancels out only because we have used the same code (`photProject`) to create maps in Volts from which we derive our responsivity. So in effect, we have introduced in our responsivity factor a term which is linked to the geometry of the pixel. I am not comfortable with this idea. If, for some reason we start deriving our responsivity from something else than the maps (e.g. the pixel timelines), there's a good chance we will get a different value for the responsivity and then have maps with wrong fluxes.

Also we have a unit inconsistency: at level 1, the data is in units of Jy/pixel, but the pixel are not those described in the calibration file, rather they are Jy per hypothetical camera pixel (3.2 or 6.4 arcsec in width depending on the channel). At level 2 the data is in units of Jy/pixel but this time these pixels are really those described by the map astrometry. Furthermore, the value in level 1 pixels is *not* the actual number of Jy we would measure in 3.2 or 6.4 arcsec-wide pixels, but an estimation of this, *knowing* that we measure flux in smaller pixels instead and *assuming* that the surface brightness is constant over the larger ideal camera pixels.

One can argue that it is not an issue as long as the whole system (our calibration scheme, the code we use to make maps, and our values of responsivity) stays self consistent (and most calibration are like that, they rely on some internal consistency). I would reply that given that the introduction of f_{act} in the map estimation has no justification from the mathematics of the projection, we are creating a serious weak point in our system.

The only justification for keeping it this way can be found in section 5.

3.2 MadMap

Actually MadMap has almost fallen through that trap. We have checked that the MadMap code completely ignores the existence of f_{act}. This is not too surprising given that the current implementation of the code does not properly handles the projection but rather introduces a one-to-one link between sky pixels and detector pixels by applying a nearest neighbor algorithm to the pixels' centers (i.e. the $\boldsymbol{P}$ or $\boldsymbol{P'}$ matrices are simply filled of 0's and 1's). At first this seemed to be exactly a recipe for photometric catastrophe given the units inconsistency noted above, but we were troubled by the surprising photometric agreement between MadMap and `photProject` (see RD1).

It turns out that MadMap, in order to allow reconstruction of maps with any pixel size, applies a renormalization to the flux it finds in the level 1 pixels. When making the time-ordered data, i.e. in the `MakeTodArray.java` code, a flux correction factor is applied to the level 1 data (i.e. the level 1 data are *divided* by this flux correction factor). This factor is the inverse of the square of the `scale` parameter that is provided to the task, which in turn is the ratio of the map pixel width to the ideal detector pixel width (e.g. 3.2 or 6.4 arcsec depending on the channel).

In essence what this means is that to build its ToD, MadMap assumes that what it finds in level 1 data are Jy per ideal detector pixel, and uses these values to estimate a flux in the associate map pixels by assuming that the surface brightness is constant on the detector pixels' areas so that the number of Jy scales with the ratio of surfaces. It turns out that *because we have introduced f_{act} in our responsivity factor*, what is in the level 1

[4] `pixFrac` also rightfully appears as a renormalization factor applied on the level 1 flux as `pixFrac` effectively changes the detector pixels' size and hence the power they should have collected

data can indeed be considered as Jy per ideal detector pixel. If MadMap were to use the actual description of the pixels that can be found in the calibration file, the maps would immediately be in the wrong scale.

Note of importance: the code in `MakeTodArray.java` only correctly handles the case of `scale` < 1. Whether or not it makes sense to make map with larger pixels, as of today the resulting maps would have an incorrect flux scale (see problem report `HCSS-12056`, fixed for version 6 of `hipe`).

3.3 Scanamorphos

Scanamorphos is a map-making code written and supported by H. Roussel (RD2) that implements $1/f$ noise filtering and second-level deglitching among other things, all in `IDL`. It is completely stand-alone and detached from Hipe. Thus it does not have access to our calibration files, and requires that level 1 data be prepared for it. This preparation simply consists in reshaping the masks, and adding the pixel coordinates to the level 1 data, something that is usually done on-the-fly by the Hipe map-makers.

Therefore, we have another map-making code that ignores the existence of f_{act} and yet produces maps that are photometrically consistent with the Hipe map-makers. Why is that so? Once again the maps are saved by the fact that *we have introduced f_{act} in our responsivity* and that scanamorphos cannot access the actual geometry of the pixels and therefore assumes that they are the ideal detector pixels (of 3.2/6.4 arcsec width for the blue/red channels). Scanamorphos estimates its maps by attributing the detector pixel a circular area slightly smaller than that of the ideal pixel. This works as well because our calibration scheme gives fluxes at level 1 as we would get them if the pixels where indeed the ideal detector pixels.

4 Conclusions, so far...

I believe this note now presents the mathematics of projection and back-projection in a way that is short yet consistent with better descriptions of the problem available in the literature. I also think that it clearly describes what information is stored in our calibration files and how this information is used (or not) by the different routes we use to perform map-making.

I also believe that it clearly highlights (1) the modification we have made to the standard algorithm used for back-projection, namely the introduction of the active fraction f_{act} in the back-projection equation of `photProject`, (2) that this modification is mathematically incorrect, and (3) that it opens the door to major inconsistencies with other map reconstruction methods *using the same level 1 data*. We are only saved from the consequences of this choice by the fact that (1) we are compensating for it in our responsivity, and (2) all other alternative map-makers consistently ignore the geometrical description of the arrays that we have in our calibration files (except for the position of pixels' centers), a situation that may/will not last forever.

Therefore, given that we are bound to revise our photometric calibration soon, I will argue here that it is time to re-consider whether the f_{act} term should stay in the `photProject` back-projection equation. I would strongly favor a scheme where we remove it, consider that units in level 1 data are Jy/detector with the detector geometry described in the calibration files. This scheme is in my opinion one that has a good chance of resisting the passing of time, and the associated dilution of knowledge. Right now we are only saved by the fact that our responsivity scale is established on maps made with a certain software, and that other softwares ignore part of the calibration information we have on our instrument. I do not consider that safe.

5 What if the projection model is wrong?

This section originates in a comment/question of N. Billot, remarking that it is hard to reconcile the *observation* that the absorption efficiency of the detector can reach very high values (close to 100%) and the *assumption* that the geometry of our detector is such that only the central 640 μm of the 750 μm-wide pixels are sensitive to light. Yet this assumption is central in our reference projection model.

I thus now investigate whether the possibility that this assumption is wrong is really of no consequence on our ability to restore photometrically correct maps, knowing that at first is seems rather counter-intuitive.

I will specifically address the question: what if our pixels really are 750 μm wide?

This means that the projection equation is now really:

$$\boldsymbol{y} = \boldsymbol{P''}\boldsymbol{x} + \boldsymbol{n}. \tag{15}$$

Compared to equation (1) it is really only the projection matrix that changes: the map is still the same map (expressed in surface brightness units, Jy/Ω, and of course unknown) and the measured signal vector is still the same vector (expressed in Jy/detector pixel, it is unchanged because it is our data!). What I am trying to say here is that the sky can obviously not be affected by my assumptions on the detector geometry and the signal is measured once and for all. Of course when I now use my previous example to see how the values of $\boldsymbol{y}$ are built up from the map, the expression is going to change, as can be visualized on Figure 2.

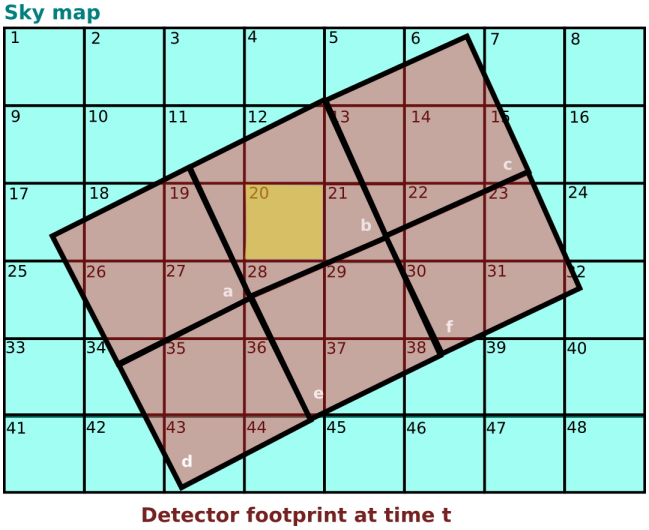

Figure 2: Same as figure 1, but this time assuming that the detector we are using has fully jointed pixels. Notice that map pixel 20 is now fully covered by detector pixel b at time t, that all the overlap areas with neighboring map pixels have increased and that pixel b at time t now also covers map pixel 27.

Indeed the relation between $y_b(t)$ and the map pixels is now:

$$y_b(t) = P''_{b,11}\times x_{11} + P''_{b,12}\times x_{12} + P''_{b,13}\times x_{13} + P''_{b,19}\times x_{19} + P''_{b,20}\times x_{20} + P''_{b,21}\times x_{21} + P''_{b,27}\times x_{27} + P''_{b,28}\times x_{28}, \tag{16}$$

which helps us realize (comparing in addition Figure 1 to Figure 2) that the differences between matrix $\boldsymbol{P''}$ and $\boldsymbol{P}$ are that (1) some matrix coefficients that were 0 in $\boldsymbol{P}$ are now non-zero in $\boldsymbol{P''}$, and (2) all the non-zero coefficients of $\boldsymbol{P}$ are now replaced by greater or equal values. How much greater? Given that pixels have increased on their edges, there's virtually no limit to the ratio $\boldsymbol{P''}i,j/\boldsymbol{P}i,j$, and the coefficients that increase less are those that were close to being equal to the map pixel solid angle. So it must be clear that it is very unlikely that we will find a couple of vectors $\boldsymbol{y}$ and $\boldsymbol{x}$ that would be simultaneously solution of equations (1) and (15).

To exemplify this more clearly on the case shown on Figure 2, our problem will originate from the fact that the relation between the recorded signal and the map is given by equation (16) when we assume it is:

$$y_b(t) = P_{b,11}\times x_{11} + P_{b,12}\times x_{12} + P_{b,13}\times x_{13} + P_{b,19}\times x_{19} + P_{b,20}\times x_{20} + P_{b,21}\times x_{21} + P_{b,28}\times x_{28}. \tag{17}$$

This actually let us see that our photometric error is not only created by a wrong assumption of the pixel size (definition and computation of $\boldsymbol{P}$ and $\boldsymbol{P''}$), which we could hope to correct using a clever calibration scheme, but also depends on the actual distribution of sky surface brightness (multiplication by $\boldsymbol{x}$)!

It also allows use to see how different brightness distribution may lead to different behavior of the photometric errors.

Let's first start with the case of an absolutely flat map ($x_j = x = cte$). In that case the actual surface brightness of the map can be factored out and what remain are sums of the $\boldsymbol{P}$ or $\boldsymbol{P''}$ coefficients for each detector sample. These are simple to compute: they are the surface of the detector pixels, S''_b or S_b (**note of some importance:** this is only true for detector pixels that fall on a fully covered area of the sky, i.e. on the edges of the map the sum of the $\boldsymbol{P}$ or $\boldsymbol{P''}$ coefficients for those detector pixels is less than their surface). Therefore the two equations above differ only by a multiplicative factor on the left side, and since this factor is the ratio of detector pixel surface (actual/assumed) we can reasonably hope that it will cancel out when we derive our response calibration from the map. Let's see whether it is the case.

Neglecting the edge-of-the-map effect mentioned above, all components of the $\boldsymbol{y}$ vector are equal to $S''_i \times\ x$, where i is the index running on the detector pixels. Therefore the equation used to estimate the map pixels becomes:

$$x_j^{est} = \frac{\sum\limits_{i,i\cap j\neq 0} (P_{i,j} \times y_i)/S_i}{\sum\limits_{i,i\cap j\neq 0} (P_{i,j})} = x \times \frac{\sum\limits_{i,i\cap j\neq 0} P_{i,j} \times S''_i/S_i}{\sum\limits_{i,i\cap j\neq 0} (P_{i,j})}, \tag{18}$$

or, in matrix form:

$$\boldsymbol{x}_{PhP} = x \times \frac{\boldsymbol{P}^T(\boldsymbol{S}''_{DetPix}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^T\boldsymbol{1}}. \tag{19}$$

In our particular case where S_i corresponds to the angular size of the inner 640 μm-wide absorbing grid, and S''_i is the angular size of the full 750 μm-wide pixel, one can safely assume that $S''_i/S_i = cte = 1/f_{act}$ and thus:

$$\boldsymbol{x}_{PhP} = x \times 1/f_{act} \times \frac{\boldsymbol{P}^T\boldsymbol{1}}{\boldsymbol{P}^T\boldsymbol{1}} = x/f_{act} \times \boldsymbol{1} = \boldsymbol{x}/f_{act}. \tag{20}$$

Therefore we indeed have reconstructed the exact sky map with a gain error, but this gain error is exactly $1/f_{act}$ (overestimation of the map) and it is likely that it is for this reason that the `photProject` code first multiplies $\boldsymbol{y}$ by f_{act} before performing the map estimation. If we substitute in equation (14) $\boldsymbol{y}$ by its expression in our particular case, we obtain:

$$\begin{aligned}
\boldsymbol{x}'_{PhP} &= S_{Sky} \times f_{act} \times \frac{\boldsymbol{P}^{T'}(\boldsymbol{y}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^{T'}\boldsymbol{1}} \\
&= S_{Sky} \times f_{act} \times x \times \frac{\boldsymbol{P}^{T'}(\boldsymbol{S}''_{DetPix}/\boldsymbol{S}_{DetPix})}{\boldsymbol{P}^{T'}\boldsymbol{1}} \\
&= S_{Sky} \times f_{act} \times x \times 1/f_{act} \times \frac{\boldsymbol{P}^{T'}\boldsymbol{1}}{\boldsymbol{P}^{T'}\boldsymbol{1}} \\
&= S_{Sky} \times \boldsymbol{x},
\end{aligned} \tag{21}$$

which is first dimensionally correct because $\boldsymbol{x}$ is in Jy/Ω and $\boldsymbol{x}'_{PhP}$ is in Jy per map pixel, and second is also photometrically correct! Therefore if we were only observing absolutely flat skies, the unnecessary introduction of the f_{act} normalization in the `photProject` code corrects the error of using $\boldsymbol{P}$ as a projection matrix when we should really be using $\boldsymbol{P''}$. It is fascinating to realize that we have implemented a projection scheme and added a correction factor in it as if we *knew* that our scheme was wrong!

But is it the end of the story? We're obviously not observing flat skies (how happy would we be if there was indeed a flat region of the sky to calibrate ourselves on..) so can we hope that a single normalization *scalar* can solve a projection model error in all situations.

Let us examine now another extreme case where only one pixel in the map has a non-zero value. I fully agree that it is a non-physical situation but it represents another mathematical extreme in which to test our approximate method of back-projection. Assuming it is pixel 20 that is illuminated, equation (16), the only one to represent the *actual* relation between the signal and the map, reduces to a single term, $P''_{b,20} \times x_{20}$.

What happens when we try to estimate x_{20} in the back projection? Remembering that we estimate the map

using projection model $\boldsymbol{P}$ while the signal has been obtained with projection model $\boldsymbol{P''}$, we have:

$$x_{20}^{est} = \frac{\sum_{i,i\cap 20\neq 0}(P_{i,20}\times y_i)/S_i}{\sum_{i,i\cap j\neq 0}(P_{i,20})} = x_{20}\times\frac{\sum_{i,i\cap 20\neq 0}(P_{i,20}\times P''_{i,20})/S_i}{\sum_{i,i\cap 20\neq 0}(P_{i,20})}. \quad (22)$$

Can we evaluate the term left of x_{20} and in particular, does it lead to an overestimation of x_{20}^{est}? What we have is a weighted mean of the terms $P''_{i,20}/S_i$. This mean is slightly particular in the sense that it is not realized over all the cases where $P''_{i,20}\neq 0$ but only on those cases where $P_{i,20}\neq 0$, which are slightly less numerous by construction. By construction as well, we have $P''_{i,20}/S_i\in\,]0, S_{sky}/S_i]$ since the area of overlap between a detector pixel and a sky pixel varies between 0 and the surface of the sky pixel. Thus we have an upper limit for the estimation at:

$$x_{20}^{est} < x_{20}\times(S_{sky}/S_{DetPix}), \quad (23)$$

where I have assumed that to a first order all detector pixels have the same projected surface (an assumption we actually make in the implementation of back-projection). Given that our weighting scheme is built to put more weight on the estimations performed with detector pixels that fully cover the sky pixel, the actual value should be closer to the limit than to 0.

Therefore our estimation of the pixel surface brightness is this time multiplied by a factor that depends on the ratio between the sky pixel and the assumed detector pixel surface. Since one usually build maps with small pixels, this can be smaller than 1 (i.e. underestimation). This could be worrying but it is not the end.

Indeed, what happens when we try to estimate sky pixels neighboring pixel 20? Any such pixel is mostly covered by detector pixels that did not see sky pixel 20, i.e. detector pixels that have zero flux and thus provide a zero estimate of the map surface brightness. However some of the detector pixels covering the map pixel of interest also covered sky pixel 20 and thus have non-zero flux. Taking an example from Figure 2, we can make the following estimate of the sky brightness in pixel 21:

$$x_{21}^{est}(t) = y_b(t)/S_b = (P''_{b,20}\times x_{20})/S_b, \quad (24)$$

i.e. estimates of the surface brightness of pixel 21 that are made from detector pixels that also overlap with pixel 20 provide a value that is not zero. This means that by construction of our estimator we distribute flux over an area around pixel 20. Given the expression above it should be clear that the flux we distribute this way is such that the surface brightness in estimated map pixels around pixel 20 is smaller than that estimated for map pixel 20.

All these computations and reasoning point in fact to an essential feature of the method used to estimate the sky map: *it does not conserve surface brightness*. By construction of the method, flux that was originally in a given area of the input sky map will get distributed around that area in the estimated map, i.e. the surface brightness spatial distribution has been modified. This raises immediately three questions: (1) what is the flux amplitude of this effect (i.e. if we compare pixel to pixel an input map to its estimated version, what kind of ratios can we observe) (2) how far does this spread-out effect reach and (3) is the total flux conserved (i.e. if we integrate the flux in the estimated map, what do we get)?

The first question is answered by equation (23): the amplitude of the variations can be of the order of the ratio between the sky pixel surface and the assumed detector pixel surface.

Can we then estimate the radius of the area on which we artificially distribute flux? In $\boldsymbol{y}$, the signal, any detector pixel that has an overlap with map pixel 20 has a non-zero value, i.e. any detector pixel which center falls within $\sim\left(\sqrt{S''_{DetPix}/2}+\sqrt{S_{sky}/2}\right)$ of pixel 20's center has a non-zero value. In turn when we back-project, any sky pixel that is fully or partially covered by a detector pixel that has a non-zero value will get a non-zero surface brightness. This means that any map pixel whose center falls within $\sim\left(\sqrt{S_{DetPix}/2}+\sqrt{S_{sky}/2}\right)$ of a non-zero detector pixel will have flux (note that here I'm using the wrong projector on purpose). Thus we can estimate that any sky pixel with a center distant of $\sim\left(\sqrt{S_{DetPix}/2}+\sqrt{S''_{DetPix}/2}+2\sqrt{S_{sky}/2}\right)$ will receive flux by construction of the sky map estimation. Using for a numerical application a red case with sky

pixels of 4″ in side, and correct/assumed detector pixels of 6.4″/5.5" (the implicit case in our implementation of `photProject`), this is a distance of 14.1" from the center of the illuminated sky pixel, i.e. a size that is significant w.r.t the sky map pixel size (and the PACS point spread function). This is shown on Figure 3 where we display the estimated map corresponding to an input map with a single illuminated pixel.

Finally it is not possible to analytically answer the third question, however a number of observation simulators exists and thanks to TAMASIS (provided by Pierre Chanial) we have simulated the case of a sky map with a single illuminated pixel. We consider a sky map sampled with pixels of 4″ in side, observed with a projector $\boldsymbol{P}''$ having pixels 6.4″ in side (red band photometer) and reconstructed with a projector $\boldsymbol{P}$ having pixels 5.5" in side (again the case implicitly implemented in our code). We then perform the total flux photometry in the resulting map and find again that is is overestimated w.r.t. to the input total flux by a factor $1/f_{act}$. Thus as far as total fluxes are concerned we are in the same situation as for the infinitely flat case, and the `photProject` renormalization will take care of that. This could in fact have been expected from the linearity of the projection and back-projection equations and the fact that a flat sky image is the sum of images having only one pixel illuminated. Total flux conservation on the flat sky thus implies total flux conservation on the elements of the sum of images making a flat sky.

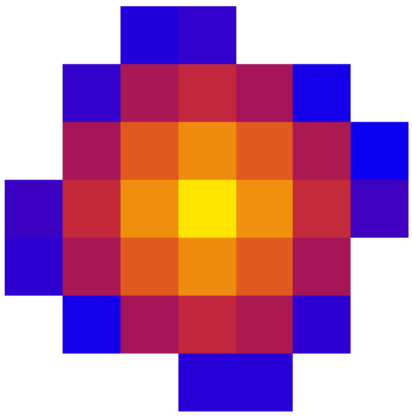

Figure 3: The estimated map when the input map has a single illuminate pixel (the central one). In the present case, the map pixel has a 4" side, the pixel in projector $\boldsymbol{P}$" has pixels 6.4" in side and the map is reconstructed assuming projector $\boldsymbol{P}$ with pixels 5.5" in side (corresponding to $f_{act} = 0.73$). The surface brightness in the original illuminated pixel was 1, and that in the central pixel of the estimated map is 0.351, while the upper limit estimated in equation (23) is 0.529 . The distance over which we predicted the spread-out effect to occur is 14.1" or 3.5 pixel, which is what we observe.

6 Conclusions, take 2

Even-though the previous section dealt with the case of a wrong projector (i.e. wrong assumption on the pixel size), there are a number of new elements that are valid for the ideal case (i.e. correct projector).

- The back-projection method we use to estimate the map does not preserve the surface brightness spatial distribution, i.e. the surface brightness that can be measured on pixel scales in the sky map has essentially no relation (or a very complex one) to the actual surface brightness in that area on the sky. This is because the projection–back-projection phenomenon leads to a convolution of the sky by a pixel response function that depends on the sampling choice for the sky map, the assumed and actual detector geometries and the scanning strategy.
- The convolution realized by the map estimation algorithm is not negligible with respect to the instrument's PSF. Note however that a measured PSF is in fact the convolution of the instrument PSF with the

projection–back-projection pixel response function of the PSF observation.

- More importantly, the algorithm chosen to estimate the sky map is flux conserving. This means that despite the convolution, the total flux of an object is preserved in the estimate, i.e. when using an identical projector for the projection and the back-projection, the total intensity of the pixel response function is 1. When using a wrong projector, the total flux is not strictly preserved but it is multiplied by a constant which is the ratio of the correct pixel size to the wrong pixel size. This can be taken care of by the calibration scheme if the maps are used to derive the responsivity.

Therefore as far as total fluxes are concerned, it essentially does not matter whether we know our detector geometry or not. This can be fixed by the calibration strategy. It remains to be investigated whether there are any particular precaution to use to compare maps estimated this way with other maps.

7 Acknowledgments

To write this note, I have benefited from extensive discussions with Hervé Aussel, Nicolas Barbey, Pierre Chanial, René Gastaud, Koryo Okumura. I have received useful feedback from Babar Ali, Nicolas Billot, Dieter Lutz and Paola Popesso. I also thank Cate Liu, Helène Roussel, and Michael Wetzstein for their help in specific software implementation areas.