EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN – AB DEPARTMENT

CERN-AB-Note-2008-024 (ABP)

Calibration and correction of the BTVs images

E.Benedetto, E.Bravin

Abstract

As the screen at the BTV monitor is not orthogonal to the optical axis of the camera, the image acquired by the CCD results distorted. An algorithm to correct for this distortion and other orientation errors is described in this note.

Geneva, Switzerland

24/04/08

1 Introduction

In addition to the monitor calibration, the images acquired at the BTVs should be corrected from the distortion due to the orientation of the OTR screens with respect to the CCD cameras. It was agreed [1] that the front-end should return in addition to the present set of data (i.e. the S1xS2 array with the signal on each pixel and two vectors of length S1 and S2 respectively, describing the raw pixel coordinates) another FESA property with the coordinates of 4 particular points (i.e. at the corners of the screen). This allows extracting the parameters to calibrate the image and correct for the distortion via a simple linear algorithm.

Figure 1 shows the image of the calibration pattern of TT10.BTV1018. The monitor is equipped with an alumina screen on which is marked a grid of $1cm$ x$\sqrt{2}cm$ pitch size. Since the screen is oriented at 45^o with respect to the optical axis, its projection on the orthogonal plane gives $1cm$ x$1cm$ pitch. For the set of monitors in the transfer line TT10, it is possible to profit of the grid and get the coordinates of four points on it. The algorithm to get, from the four points at the corners of a rectangular box, the parameters to convert the distorted raw image (in pixels) into the real image at the screen will be described in this note.

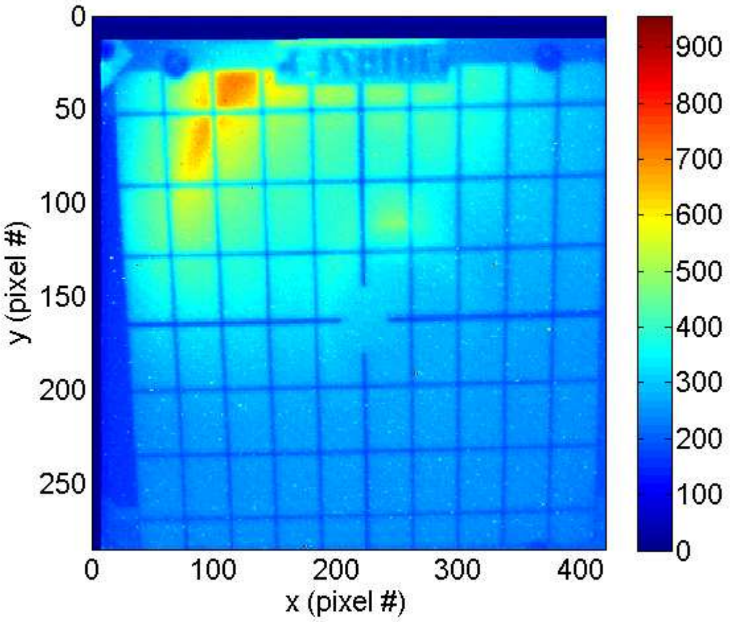

Figure 1: TT10.BTV1018 calibration image

2 The OTR monitors and the reference systems

Transition radiation is produced when the beam is crossing the surface of the screen conducting material. The backward directed radiation, which will be collected by a CCD camera, is emitted in the direction corresponding to the geometric reflection of the incident beam angle ϕ, as represented in Fig. 2. By geometrical considerations, ϕ is also the tilt angle of the screen with respect to the plane perpendicular to the axis of the OTR optical system.

For the purpose of calibrating and correcting the image, it is possible to identify three different coordinate systems:

- the screen (the object) $\rightarrow (x_S, y_S)$
- the image at the camera $\rightarrow (x_I, y_I)$
- the pixel digitalization $\rightarrow (x_p, y_p)$

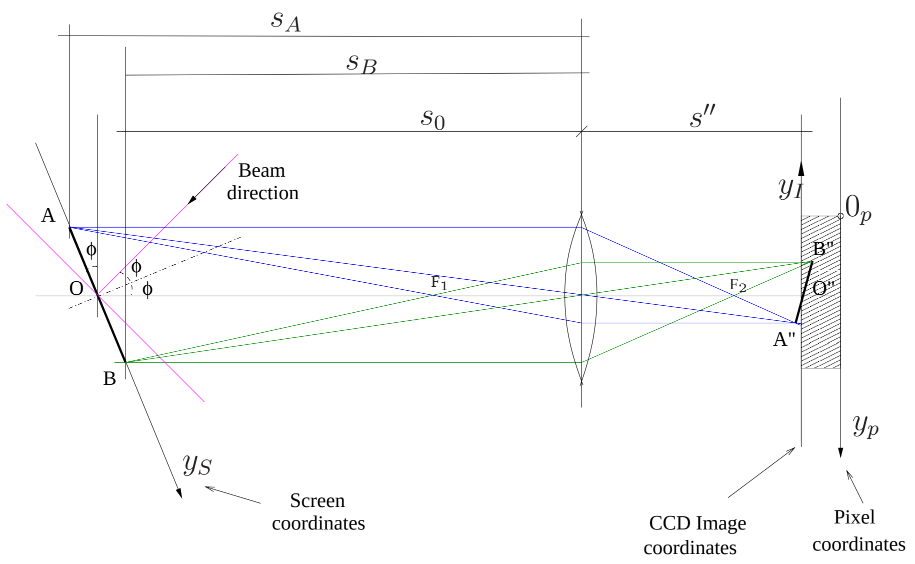

Figure 2: The OTR system and the three coordinates sets

We have access to the coordinates of the pixels which are integer numbers and we want to know the real coordinates at the screen, from which we can retrieve the ones of the beam that are (x_S, $y_S \cos\phi$).

The transformation from the screen to the image at the camera involves the geometrical optics equations and in particular the notion of magnification factor M with its dependence on the distance between the object and the lens and thus dependence on y_S. The digitalization of the image implies different horizontal and vertical scaling factors, due to the different sampling frequency in the two directions. Moreover, the origin of the coordinates in pixel is in the top left corner of the image, so that the sign for a variation in the y_p coordinate is inverted, and the camera image can be tilted.

In addition to that, for what concerns the orientation (left/right, top/bottom) it is important to have the same sign convention of MAD-X [2] for the BTV images. For this purpose, two flags have to be passed from the front-end to the acquisition application, so that it will be easy to check if this is the case and eventually change the sign of the horizontal/vertical coordinate.

The distorted image at a calibration screen is sketched in Fig. 3, left. The coordinates of the points P1, P2, P3, P4, at the corner of a rectangular box are given both in the pixel reference and in the screen reference frame.

3 Rotation and translation

First of all, the image can be tilted and needs to be rotated by an angle θ in order to make the box sides P1-P2 and P4-P3 (see Fig. 3, left) horizontal:

$$\theta = \text{mean}\left\{\arctan\frac{\mathrm{y_{P2}} - \mathrm{y_{P1}}}{\mathrm{x_{P2}} - \mathrm{x_{P1}}}, \arctan\frac{\mathrm{y_{P3}} - \mathrm{y_{P4}}}{\mathrm{x_{P3}} - \mathrm{x_{P4}}}\right\} \tag{1}$$

The rotation matrix to be applied is:

$$M_r = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2}$$

For convenience reasons, the y–axis is translated to the axis where there is no deformation. To find the "neutral" axis we proceed by noting that the sides P1-P4 and P2-P3 are not parallel. They are respectively laying on the lines $r1$ and $r2$ of equation:

$$\begin{aligned} r1: \quad & y_p = y_{p,1} + m_1\,(x_p - x_{p,1}) \\ r2: \quad & y_p = y_{p,2} + m_2\,(x_p - x_{p,2}) \end{aligned}$$

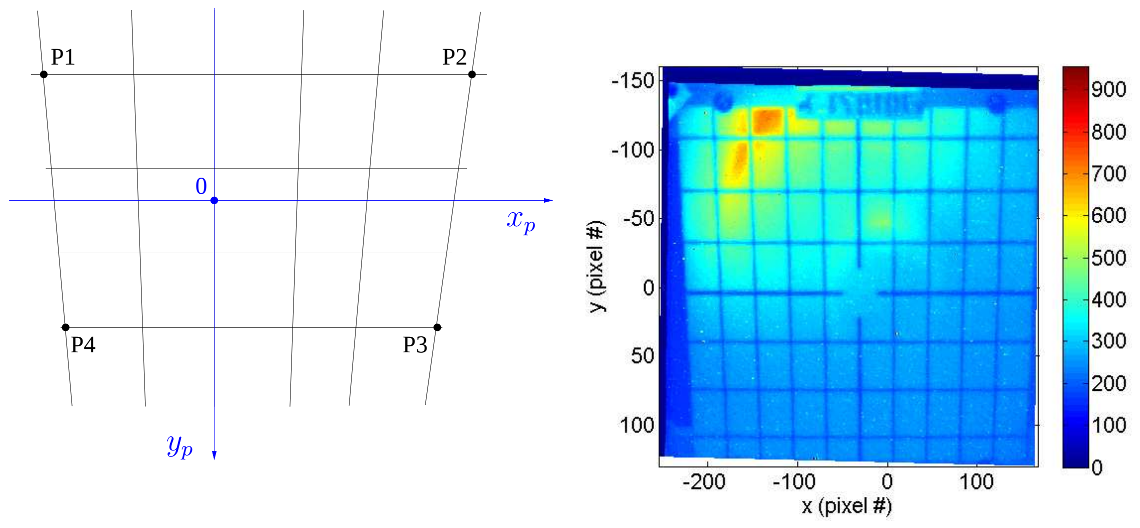

Figure 3: Left: Sketch of a deformed image used for calibration: the points P1, P2, P3, P4 coordinates are given. Rigth: TT10.BTV1018 calibration image, after rotation and translation.

where the coefficient m_1 and m_2 are:

$$m_1 = \frac{y_{p,4} - y_{p,1}}{x_{p,4} - x_{p,1}} \; ; \qquad m_2 = \frac{y_{p,3} - y_{p,2}}{x_{p,3} - x_{p,2}} \tag{3}$$

The point Q=$(x_{p,Q}, y_{p,Q})$, in which the two lines are crossing, and through which the "neutral" vertical axis is passing as well, is obtained by solving the system:

$$\begin{cases} y_{p,Q} = y_{p,1} + m_1 \left(x_{p,Q} - x_{p,1}\right) \\ y_{p,Q} = y_{p,2} + m_2 \left(x_{p,Q} - x_{p,2}\right) \end{cases}$$

The point O, which will be the new origin of the coordinates system after the translation, will therefore have the same x–coordinate of the point Q:

$$x_{p,O} = x_{p,Q} = \frac{y_{p,2} - y_{p,1} + m_1 x_{p,1} - m_2 x_{p,2}}{m_1 - m_2} \tag{4}$$

The y–coordinate of the point O can be chosen arbitrarily, but it is convenient to set it to:

$$y_{p,0} = \frac{y_{p,1} + y_{p,4}}{2} \tag{5}$$

leading to the coordinates axis shown in Fig. 3 (left).

From now on, the treatment will always consider the image after the rotation and after the translation to the point (x_0, y_0). In order to simplify the notation, the rotated and translated coordinates of the points will be indicated again by $x_{p,i}, y_{p,i}$.In Fig. 3 (right) is shown the calibration image of TT10.BTV1018 after the rotation and translation.

4 Image correction and calibration

4.1 A little bit of optics...

The magnification factor M, which is the ratio between the image (I) and the object (S) dimensions $h = x, y$, is:

$$M \equiv \frac{h_I}{h_S} = \frac{s''}{s} \tag{6}$$

so that:

$$x_I = M \cdot x_S \tag{7}$$

$$y_I = M \cdot y_S \cos\phi \tag{8}$$

s and s'' (see Fig. 2) are respectively the distance of the object and the image from the principal point of the lens and they are linked with the focal length f via the formula:

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s''} \tag{9}$$

By putting together Eqs. 6 and 9, the magnification factor can be written as:

$$M = \frac{f}{s-f} \tag{10}$$

In most of the BTVs in the LHC complex (transfer lines and SPS), the distance s between the camera and the screen is about $\sim (30 \div 50)$ cm, while the focal length is $f \sim 25$ mm, leading to a $M < 0.1$. This translates into a CCD sensor of $\sim 1/3''$ for a screen image of about ~ 10 cm.

The points on the screen, lying on a plane tilted by an angle ϕ (usually it is $\phi = 45^o$) with respect to the plane perpendicular to the optical axis, have different s, for different y_S coordinates on the screen:

$$s = s_0 + \Delta s = s_0 - y_S \sin\phi \tag{11}$$

In our case, we can assume s'' constant (i.e. the points A'', O'' and B'' of Fig. 2 lying on the same plane perpendicular to the optical axis) and:

$$s'' \sim f \ll s \tag{12}$$

We can also consider:

$$y_S \sin\phi \ll s_0 \tag{13}$$

and expand the magnification factor for $\left(\frac{y_S \sin\phi}{s_0 - f}\right) \ll 1$, keeping only the first order term:

$$M = \frac{f}{(s_0 - y_S \sin\phi) - f} \approx \frac{f}{s_0 - f}\left(1 + \frac{y_S \sin\phi}{s_0 - f}\right) \tag{14}$$

Finally, since the y_S coordinate on the screen is proportional to the image coordinate y_I (to first order) and:

$$y_S \propto y_I \propto -y_p \quad , \tag{15}$$

we can write:

$$M \approx C_0\,(1 - C_1 y_p) \tag{16}$$

The transformation from the screen coordinates to the CCD camera image coordinates is therefore:

$$x_I = M(y_p)x_S = C_0\,(1 - C_1 y_p)\,x_S \tag{17}$$

$$y_I = M(y_p)y_S \cos\phi = C_0\,(1 - C_1 y_p)\,y_S \cos\phi \tag{18}$$

with the coefficients C_0 and C_1 to be determined.

4.2 Image digitalization

The CCD camera has an analog reading. In the conversion to the digitized signal, two different coefficients D_x and D_y should be introduced, due to the different sampling frequency in the two directions, i.e. the pixels are not exactly square. Moreover, the sign of the y_p coordinate needs to be exchanged (see Fig. 2) to comply with the convention to have the zero in the top left corner and increasing y–coordinated toward the bottom of the image (see Fig. 1) .

$$x_p = D_x\,x_I \tag{19}$$

$$y_p = -D_y\,y_I = \alpha D_x\,y_I \; ; \quad \alpha = -D_y/D_x \tag{20}$$

4.3 How to determine the coefficients

To summarize, the equations to go from the coordinates at the screen to the coordinates of the pixels are:

$$x_p = D_x M(y_p)\, x_S = C_0^*\, x_S - C_0^*\, C_1 y_p x_S \tag{21}$$
$$y_p = -D_y M(y_p)\, y_S \cos\phi = \alpha C_0^*\, y_S \cos\phi - \alpha C_0^*\, C_1 y_p y_S \cos\phi\ , \tag{22}$$

being $C_0^* = (D_x C_0)$, and the inverse transformation (from the pixel to the real coordinates) is:

$$x_S = \frac{x_p}{C_0^* - C_0^*\, C_1 y_p} \tag{23}$$
$$y_S \cos\phi = \frac{y_p}{\alpha\,(C_0^* - C_0^*\, C_1 y_p)} \tag{24}$$

The beam coordinates, in which finally we are interested, are exactly $(x_S, y_S \cos\phi)$.

We need to determine 3 coefficients: C_0^*, C_1 and α, from the coordinates of the points P1,...P4 of the digitized image (already rotated by the angle θ and translated to the point (x_0, y_0)).

From Eq. 21, let's consider the differences $\Delta x_{p,21} = (x_{p,2} - x_{p,1})$ and $\Delta x_{p,34} = (x_{p,3} - x_{p,4})$ respectively between the x–coordinates of points P1–P2 and P3–P4, which on the screen are separated by the same $\Delta x_S = (x_{S,2} - x_{S,1}) = (x_{S,3} - x_{S,4})$, known:

$$\begin{cases} \Delta x_{p,21} = C_0^*\, \Delta x_S - C_0^*\, C_1\, y_{p,1}\, \Delta x_S \\ \Delta x_{p,34} = C_0^*\, \Delta x_S - C_0^*\, C_1\, y_{p,4}\, \Delta x_S \end{cases} \tag{25}$$

By solving the system in Eq. 25, it is:

$$C_0^* = \frac{(\Delta x_{p,21}\, y_{p,4} - \Delta x_{p,34}\, y_{p,1})}{(y_{p,4} - y_{p,1})\Delta x_S} \tag{26}$$

and

$$C_1 = \frac{(\Delta x_{p,21} - \Delta x_{p,34})}{(\Delta x_{p,21}\, y_{p,4} - \Delta x_{p,34}\, y_{p,1})} \tag{27}$$

Finally, using Eq. 22 to compute the difference between $y_{p,4}$ and $y_{p,1}$, it is:

$$(y_{p,4} - y_{p,1}) = \alpha C_0^* (y_{S,4} - y_{S,1}) \cos\phi - \alpha C_0^*\, C_1 (y_{p,4} y_{S,4} - y_{p,1} y_{S,1}) \cos\phi \tag{28}$$

from which the coefficient α is:

$$\alpha = \frac{(y_{p,4} - y_{p,1})^2 \Delta x_S\, /\, \cos\phi}{(\Delta x_{p,21}\, y_{p,4} - \Delta x_{p,34}\, y_{p,1})(y_{S,4} - y_{S,1}) - (\Delta x_{p,21} - \Delta x_{p,34})(y_{p,4}\, y_{S,4} - y_{p,1}\, y_{S,1})} \tag{29}$$

5 Pixel intensity correction

Because of the variation of the magnification with y_p, the intensity I_S must be scaled accordingly. If we look at Eq. 23 and 24 as a "y–dependent" scaling of the pixel dimensions, in order to preserve the collected light intensity, if the pixel size is decreased its intensity should be augmented of the same amount and vice versa:

$$I_S = I_p \left(\frac{D_x M(y_p)}{D_x C_0} \right) \left(\frac{D_y M(y_p)}{D_y C_0} \right) = I_p \,(1 - C_1 y_p)^2 \tag{30}$$

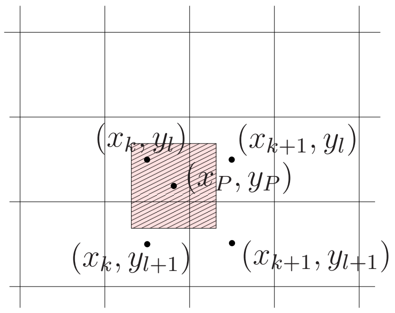

Figure 4: The pixel mapped on the new equispaced mesh

6 Mapping to a new array

By applying this algorithm, the notion of mapping a S_xxS_y image to another S_xxS_y raster has been abandoned, in favor of considering "pixels" of varying dimensions and intensity. This approach has the advantages of having no discontinuities and the intensity-density being automatically preserved, but on the practical point of view it is not possible any longer to compute easily the image projection on the x(y)–axis, by simply summing over the columns(rows) of the 2D pixel-intensity array.

In order to have the intensity values on an array, it is possible, starting from Eqs 23 and 24, to linearly interpolate the intensity values of the old pixels, which now have new coordinates, to a newly created equispaced mesh (i.e. of $\Delta x_{new} = 1/(D_x C_0)$ and $\Delta y_{new} = 1/\alpha/(D_x C_0)$) in analogy to what is done in the Particle In Cells and Cloud In Cells simulations algorithms [3]. The intensity of every pixel is distributed in the 4 surrounding cells so that, as in the case of Fig. 4:

$$\begin{aligned}
I_{k,l} &= I_p \left(1 - \frac{x_p - x_k}{x_{k+1} - x_k}\right)\left(1 - \frac{y_p - y_l}{y_{l+1} - y_l}\right)\\
I_{k,l+1} &= I_p \left(1 - \frac{x_p - x_k}{x_{k+1} - x_k}\right)\left(\frac{y_p - y_l}{y_{l+1} - y_l}\right)\\
I_{k+1,l} &= I_p \left(\frac{x_p - x_k}{x_{k+1} - x_k}\right)\left(1 - \frac{y_p - y_l}{y_{l+1} - y_l}\right)\\
I_{k+1,l+1} &= I_p \left(\frac{x_p - x_k}{x_{k+1} - x_k}\right)\left(\frac{y_p - y_l}{y_{l+1} - y_l}\right)
\end{aligned} \tag{31}$$

This algorithm is valid for "small" corrections, i.e. if the pixel dimensions are comparable to the size of the new cells $\Delta x_{new} = 1/C_0^*$:

$$\frac{\left|\left(\frac{1}{C_0^* - C_0^* C_1 y_p}\right) - \Delta x_{new}\right|}{\Delta x_{new}} \ll 1 \tag{32}$$

which is verified in our case since it requires $C_1 y_p \ll 1$ (Eq. 13 and following).

Figure 5 illustrates the two approaches. On the left side, the pixels are "enlarged" and their intensity is reduced accordingly, and vice versa, while on the right side, the pixels intensities with the new coordinates are interpolated on a equispaced grid. Figure 6 shows the resulting beam images after the application of the calibration algorithms, and again the differences between the two methods.

7 Conclusions and observations

An algorithm to calibrate the BTVs images and correct for the tilt angle between the screen and the optical axis has been written and it has been successfully tested for the OTR monitors installed in TT10.

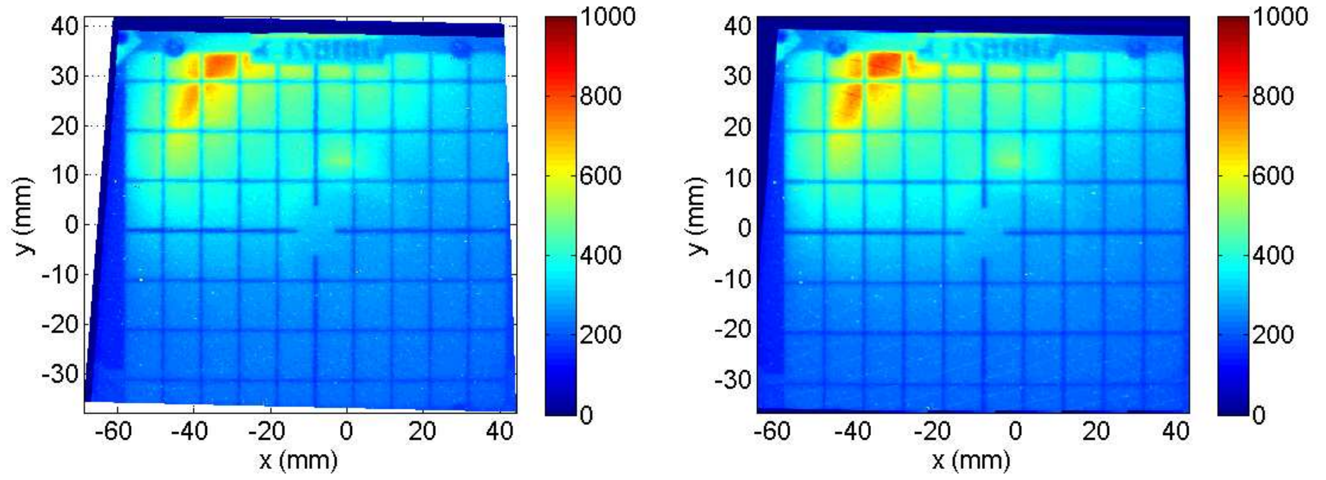

Figure 5: The corrected TT10.BTV1018 calibration image. Left: The image obtained from the pixel "deformation". Right: The result of the interpolation and mapping on a new mesh

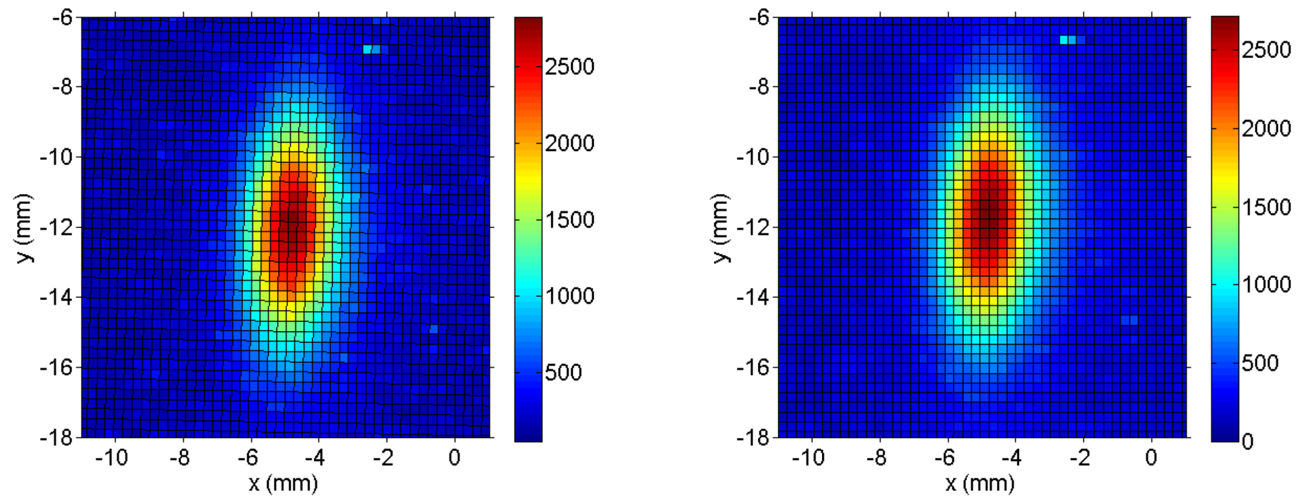

Figure 6: Left: Beam image at TT10.BTV1018 obtained from the pixel "deformation". Right: The result of the interpolation and mapping on a new mesh

For this correction, the front-end should return, in addition to the $S1$x$S2$ image array and the two pixel-coordinate vectors of length $S1$ and $S2$ respectively, another FESA property with the position of 4 particular points, both in the pixel coordinate system (x_p, y_p) and in real mm coordinates at the screen (x_S, y_S), and the tilt angle ϕ of the screen with respect to the plane perpendicular to the optical axis.

For what concerns the BTVs in TT10, the screens are tilted by an angle $\phi = 45^o$ and are equipped with an Alumina screen on which is marked a grid of 1cm x $\sqrt{2}/2$cm pitch size, so that the front-end can easily provide the coordinates of the 4 corners of a rectangular "box". Not all the BTVs have this grid, e.g. the monitors in TI2, and for them it will be necessary to identify other solutions.

The algorithm, which involves rotation, translation and dilatations of the pixel coordinates, computes a new set of coordinates for each pixel element. If only the 2D fit on the beam profile was required, it would be possible to stop at this point and work with pixels of variable size and intensity. This would be the best approach, since it does not imply interpolations and it automatically preserves the total intensity. However, on the practical point of view, it is useful to map the image on a newly-created equispaced mesh, by linearly interpolating the pixel intensities on the new grid. The result of this exercise gives back again two coordinate vectors of length $S1$ and $S2$ and the new $S1$x$S2$ image array, which is very easy to manipulate, i.e. the sum over the columns/rows gives the horizontal/vertical profile projection and it can be easily visualized e.g. in the standard Java application.

Concerning the 2D fit on the images, the two methods do not give appreciable differences in terms of beam sizes. Moreover, the image mapping on an equispaced grid has the advantage that it is straightforward to retrieve the “first guess” for the 2D fit directly from the Gaussian fits on the projections. The computational time which is required for this further step (i.e. the additional “for loop” over the $S1$x$S2$ pixels) at a first evaluation does not affect the speed of the application and for sure it is payed off by avoiding complications in the algorithms for the visualization of the profiles and the fits.

Another information which have to provided from the front-end is the orientation of the image (right/left, top/bottom). Two flags for that are already existing and it will be important to pass them to the acquisition application and check if they are consistent with MAD-X convention.

8 Acknowledgments

The authors would like to thank G. Arduini for encouraging the study and useful discussions, J.J. Gras, A. Guerrero, F. Follin, V. Kain for discussions and the implementation in the front-end and in the operation applications.

References

[1] meeting, 20/12/07

[2] MAD-X web page, in particular: http://mad.web.cern.ch/mad/Introduction/conventions.html

[3] R.W. Hockeney, J.W. Eastood, Computer simulation using particles, McGraw-Hill, New York, (1981)