Review on image processing algorithms used in PITZ applications

DESY summer student program 2013

Weijia Xiong

Tsinghua University, CHINA

Supervisor

Marek Otevrel

1st of September 2013

Abstract

PITZ was built to develop and test free electron laser (FEL) photo injector system capable of producing high charge short electron bunches of lowest possible transverse emittance for the optimization of FEL performance. Images of the beams are taken at the observation station for further analysis and optimization. To deal with the images, different image processing algorithms are adopted. These image processing algorithms are discussed in this paper for providing a basic understanding and serving as the first step towards unification of image processing within PITZ group.

Contents

I. Introduction

The Photo Injector Test facility at DESY, Zeuthen(PITZ) site was built to develop, test and optimize high quality beam source for Free Electron Laser(FEL) like FLASH and European XFEL. The main requirement for a FEL electron injector is the reliable ability to generate electron beams with good brightness and a very small transverse emittance and a reasonably small longitudinal emittance. Images of the beams are taken by cameras to reconstruct phase space and calculate emittance. However, the original images have all kinds of noises, coming out of the background, the camera, or induced by the signal itself. Image processing filters are applied to the original images to get rid of the unwanted noises. The following chapters deal with the image processing algorithms presently used in all PITZ applications, including Video Client3, Emcalc, TOMO, MAMA and Slice Emittance.

II. Image processing algorithms used in PITZ applications

i. Videoclient3

The videoclient 3 uses C/C++ as its programming language. The goal of videoclient3 is to measure the electron beam position and the transverse profile of the beam at different locations along the beam line [1]. Basically it includes 4 procedures:

- Receiving images;
- X-ray filtering;
- Background subtraction;
- Normalization;

The data we get from camera is called "raw data". It consists of **an array of bytes**, including the information like Width, Height, Bpp (bits per pixel) of the image. The basic idea here is to extract the significant data out of raw data. (All the images discussed below are of 8 bpp.)

1) X-ray filtering

5	8	20
18	250	7
12	60	200

We compare the central value with surrounding values to check if the central point is a noise and needs to be filtered. We don't directly compare all the surroundings with the central one, instead, we define a double: *AwayAboveSurroundings*, which varies from 1.0 to 3.0 to calculate the value is to be compared later. Alternatively, we have another variable: percentage, which is given by *AwayAboveSurroundings* - 1. In this way, we can choose the percentage to determine to which extent we want the filter intensity to be. The algorithm can be understood from following:

AwayAboveSurroundings: [1.0, 3.0] *//Default:1.5//*
(Alternative input: *Percentage= AwayAboveSurroundings* -1
Default *Percentage*: 0.5)
ALevelNumSurroundings: *Maximum allowed counted number of surrounding pixels*
//Default aLevelNumSurroundings: 4//

cmp =central pixel× (1/ *AwayAboveSurroundings*)
Amount=0
For all surrounded pixles:
If value (surrounding pixel) < *cmp*
Amount+=1
End
If Amount >= *aLevelNumSurroundings*
Central pixel=*Average* (surrounding pixels)

In this case, *cmp*=250× (1/1.5) =166
Only 1 surrounding pixel value is larger than 166, i.e. *Amount*=7 > *ALevelNumSurroundings.*
Thus, central pixel= *Average* (surrounding pixels) = *Average* (5, 18, 12, 60, 200, 7, 20, 8) =47
For now we can see that X-ray filter in videoclient 3 is to compare the central pixel with surroundings to see if the central pixel needs to be replaced, and we have a percentage to control how intensive the filtering process is.

2) Background subtraction

There're two solutions to do background subtraction: average and envelope.
Let's consider following few pixels as an example:
Background:

Bkg.1

100	80
20	50

Bkg.2

110	50
30	40

Bkg.3

120	40
100	80

Image:

200	50
120	80

a. Average

In this way, we calculate the average of the background pixel values and then use Image pixel values to subtract the average background pixel values (results less than 0 are set to 0).

$$Bkg\_avg = \frac{1}{N}\sum Bkg\,(k)$$

Bkg_avg:

110	56
50	56

Result (Image-Bkg_avg):

90	0
70	24

b. Envelope

In this way, we take the maximum pixel values of all the backgrounds counted and use image pixels to subtract background pixels.

$$Bkg\_env = max\{Bkg(k), \forall k\}$$

Bkg_env:

120	80
100	80

Result (Image-Bkg_env):

80	0
20	0

3) Normalization

To make the image more conducive to human eyes, normalization is applied. Normalization is also called histogram equalization, which effectively spreads out the most frequent intensity values and can thus gain a higher global contrast of the image.
For example we have a few pixels as below:

20	15	87	75	150

Top-down normalization which means spreading out the pixle values to the whole space, i.e.0 to 255 is done. In this case, we have *MinValue* =15---->0, while *MaxValue*=150---->255, Constant *AbsMaxValue*=255. By solving the simple equations, we get:

$$Scale = AbsMaxValue/(MaxValue - MinValue)$$

$$Offset = Sacle \times MinValue \times (-1)$$

The pixel values are transformed to normalized ones through following transformation:

$$NormalizedValue = OldValue \times scale + Offset$$

In our case, we have the *Scale*=1.89 and the *Offset*=-28.33. A normalized pixel values are as follows:

9	0	136	113	255

ii. Emcalc

In order to measure the size and the emittance of an electron beam accurately, filters are applied to the beam's images. Images are collected by PITZ video system and are stored in ".imc "files as arrays of bytes. Emcalc contains 3 kinds of image filters. For each filter, a number of consecutive images are taken to get better filtered.

The three kinds of filters are denoded as follows:

- The so-called EMSY filter is applied to images taken at the Emittance Measurement System (EMSY) station where the moving slit is located.
- The so-called MOI filter is applied to images taken at the observation screen station to create a mask for beamlets.(Here *mask* is a logical matrix)
- The Beamlets filter is applied to beamlets images taken at the observation screen station.

The screen station positions are briefly shown in Figure 2.1.

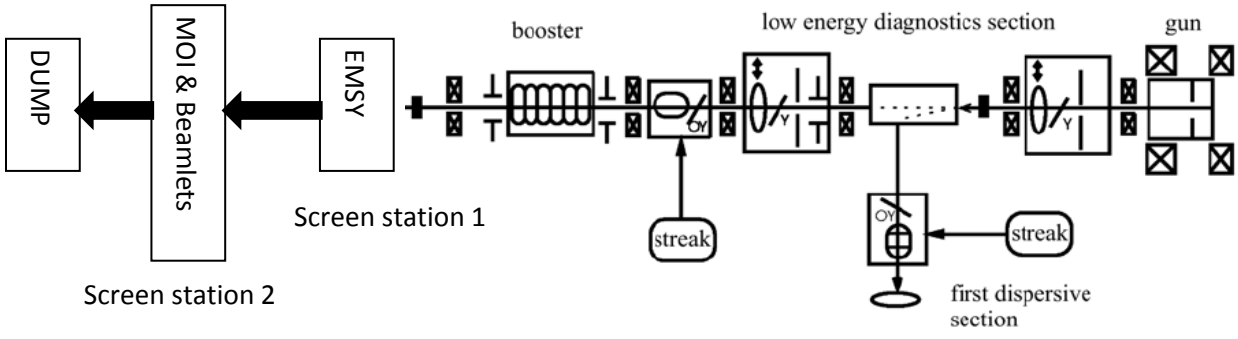

Figure 2.1: Screen station positions (from right to left)

1) EMSY filter

First we load several EMSY signal and background images, computing the average of signal and background images respectively. Then we subtract the average background from the average signal; if the result is negative, it is set to 0.

Sig_without_Bkc (i, j) = Average_Sig (i, j) - Average_Bkc (i, j);
If Sig_without_Bkc (i, j) < 0, Sig_without_Bkc (i, j) = 0;

Next we calculate the RMS of the background images to create a mask. Whenever the signal without background is larger than the root mean square (RMS) of the set of the background images, we set it to 1, else 0.

$$Mask = \begin{cases} 1, if\ Sig\_without\_Bkc\ (i,j)\ >\ RMS\_Bkc\ (i,j) \\ 0, if\ Sig\_without\_Bkc\ (i,j)\ \leq\ RMS\_Bkc\ (i,j) \end{cases}$$

Now we can start to create a filter. The filter is made up of two parts. First we implement a multiplicative filter to cut the mask to get rid of the unwanted spots which remain after the background subtraction, and then we restore the surviving signal. Both cutting and restoring are done three times.

a) Cutting

Let's consider the matrix below to represent the mask.

$$Mask = \begin{pmatrix} \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\ \vdots & \cdots & \ddots & \cdots & \cdots & \iddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & x(i,j) & x(i,j+1) & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & x(i+1,j) & x(i+1,j+1) & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \end{pmatrix}$$

According to the algorithm, the pixel $x(i,j)$ is replaced by the multiplication of the surrounding pixels.

For k=1:3

$$Mask(i,j) = x(i,j) \times x(i+1,j) \times x(i,j+1) \times x(i+1,j+1)$$

End

b) Restoring

If we apply the *Mask* to the image now, the filtering would be too aggressive. Therefore, we have to do restoration to restore some pixels that are actually in the original signal but are cut out in the above cutting procedure. Let's consider the below matrix to represent the image after being cut.

$$Mask = \begin{pmatrix} \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\ \vdots & \cdots & \cdots & \cdots & \cdots & x(i,j) & x(i,j+1) & \cdots & \vdots \\ \vdots & \cdots & x(i+1,j-3) & x(i+1,j-2) & x(i+1,j-1) & x(i+1,j) & x(i+1,j+1) & \cdots & \vdots \\ \vdots & \iddots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \end{pmatrix}$$

To restore, the pixel $x(i,j)$ is replaced by the sum of the surrounding pixels shown in the above matrix *Mask*.

For k=1:3

$$Mask = x(i,j) + x(i,j+1) + x(i+1,j-3) + x(i+1,j-2) + x(i+1,j-1) + x(i+1,j) + x(i+1,j+1)$$

End

Finally we get the mask we need:

$$EMSY\_Mask = \begin{cases} 1, if\ Mask \geq 1 \\ 0, if\ Mask < 1 \end{cases}$$

So far, the cutting and the restoring procedures have been done. And the restored mask is applied to the image to get the filtered one.

$$Filt\_EMSY = Sig\_without\_Bkc \times EMSYMask$$

According to past experiences, there are always noises in the marginal areas of the image and cannot be cut due to cameras' own characteristics; worse still, they can get restored to a bigger scale. For this reason, some pixels in the edges need to be cut out as well (set the pixel value to be 0). Thus, another 4 inputs are added: offset from all sides.

After above processing procedures, we should never forget to flip the image. We want to have the image in right-handed coordinate system with positive z-axis pointing to the beam's travel direction while what we have now is 2D image of the opposite x-axis and y-axis. The coordinate before and after flipping is shown in Figure 2.2.

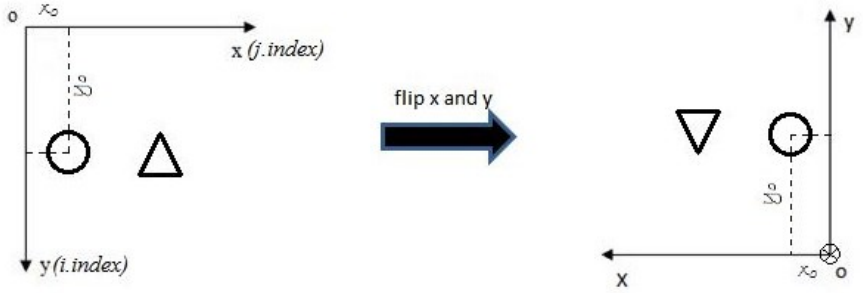

Figure 2.2: the coordinate before and after flipping (from left to right)

2) MOI filter

The so-called Mask of Interest filter is applied to deal with the full beam image taken at the observation screen station. We have following steps to deal with MOI images:

1) Load files;
2) Subtract the background
3) Create mask—in the same way as in EMSY filter.

After these steps, we get a matrix ***MOIMask*** which is going to be used as a mask in the following Beamlets filter.

3) Beamlets filter

Beamlets filter is quite different from EMSY and MOI filters. In EMSY and MOI, we get a piles of signal images and background images and we compute the average of them; while in Beamlets, we simply have single beamlets signal image and beamlets background for every slit position.

To do filtering, files are loaded; the average and the RMS of the background are calculated (as mentioned above, we don't have statistic records for background, but this is the best we can do to achieve filtering). By subtracting the average background and the RMS of background, a *BletMask* is created.

*Blet {k} =Blet {*k*}-AverageBletBkc;*

NewBlet {k} = Blet {k}-RMSBletBkg;
BletMask {k} = (NewBlet {k}>0);
//k represents the number of a beamlet image//

Before applying a filter, we have to make sure that the beamlets do fit with *MOIMask.*

$$BletMask\{k\} = MOIMask \times BletMask\{k\};$$

The beamlets are now ready to be filtered. The compositions of cutting and restoring pixels are a bit different, but both are still done 3 times.

a) Cutting

Taking the matrix below as a mask of a beamlet:

$$\text{BletMask}\{k\} = \begin{pmatrix} \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\ \vdots & \ddots & \cdots & \cdots & \cdots & \iddots & \vdots \\ \vdots & \cdots & x(i-1,j-1) & x(i-1,j) & x(i-1,j+1) & \cdots & \vdots \\ \vdots & \cdots & x(i,j-1) & x(i,j) & x(i,j+1) & \cdots & \vdots \\ \vdots & \cdots & x(i+1,j-1) & x(i+1,j) & x(i+1,j+1) & \cdots & \vdots \\ \vdots & \iddots & \cdots & \cdots & \cdots & \ddots & \vdots \\ \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \end{pmatrix}$$

Still using multiplication to replace the central value, we have:

For m=1:3

$$BletMask = x(i,j) \times x(i-1,j-1) \times x(i-1,j) \times x(i-1,j+1) \times x(i,j-1) \times x(i,j+1) \times x(i+1,j-1) \times x(i+1,j) \times x(i+1,j+1)$$

End

b) Restoring

The cutting is so intensive that we have to do restoration to restore some pixels that are actually in the original signal but are cut out in the above cutting process.

We calculate the sum of all the surrounding pixels and set $x(i,j)$ to 1 if *BletMask* >1, else it's 0.

For m=1:3

$$BletMask\{k\} = x(i,j) + x(i-1,j-1) + x(i-1,j) + x(i-1,j+1) + x(i,j-1) + x(i,j+1) + x(i+1,j-1) + x(i+1,j) + x(i+1,j+1)$$

End

After cutting and restoring, we finally get the mask we need:

$$ResBletMask\{k\} = \begin{cases} 1, if\ BletMask \geq 1 \\ 0, if\ BletMask < 1 \end{cases}$$

For the same reasons as in the case of the EMSY filter, we need to cut the marginal pixels (set the marginal pixels to zero) and flip the image. Finally, we get the filtered image by $Filtered\_Beamlets = ResBletMask \times NewBlet$, and we use it to reconstruct the phase space and do the emittance calculation.

iii. TOMO

For detailed analysis of the transverse phase-space density distribution of the electron beam, a phase space tomography section is installed. The tomography section at PITZ (Figure 3.1) consists of three FODO cells and four diagnostic stations for measuring the spatial beam density distributions. Several groups of quadruple magnets distributed upstream along the PITZ beam line are used to match the beam to the tomography section[2].

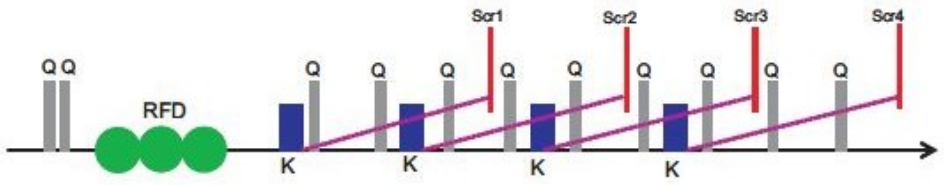

Figure 3.1: Simplified layout of the RF deflector followed by tomography section

This chapter will be dealing with the image processing algorithms used in TOMO.
The code itself is written in C++. After the ".imc" files (raw data stored in an array of byte) are loaded, there're two ways of subtracting the background noise from the original signal (Only one of them is applied.)

1a. Clean over averaged

After grabbing N frames images and calculating the average of both signals and backgrounds, the average of backgrounds is subtracted from the average of the signals. The result of this is a single cleaned frame.

$$Result = AverageSignal - AverageBackground$$

1b. Clean over single frame

In this case, the average of the backgrounds is first calculated and it is then subtracted from each signal frame.

$$Result(k) = Signal(k) - AverageBackground$$

Images cleaned this way undergo further cleaning steps and after that the result is obtained by means of averaging as follows:

$$Result = \frac{\sum_{k=1}^{N} Result(k)}{N}$$

If there's no other cleaning applied, the result of both above described methods would be the same. The reason why we consider the second cleaning is that the background noise is not stable but constantly shifting. Therefore it won't be enough for us to simply use the Clean over averaged.

Following cleaning procedures are available:

2. Sigma compared

The RMS of background noise is calculated over the number of frames and compared with the result in the previous step. If the RMS of background noise is larger than the result, the result pixel value is set to 0.

$$For\ all\ pixels\ seperately, \quad If\ RMS\_bkg > Reult, Result = 0$$
$$//Otherwise, Result\ unchanged//$$

3. Sigma subtracted

The RMS of background noise is calculated from the statistics over the number of frames. The RMS is subtracted from each pixel from the result of the proceeding step.

$$For\ all\ pixels\ seperately, \quad Result = Result - RMS\_bkg$$
$$If\ Result < 0, \quad Result = 0$$

It's important to point out that only one of these two processes can be done. If we do Sigma compared, we don't do Sigma subtracted, and vice versa.

4. Neighbors filtered

A pixel is treated as noise if the multiplication of surrounding $N \times N$ matrix is 0. The value of N indicates the number of rows/columns surrounding a pixel that would be taken into account. *N* can be any odd number and is controllable.
Let's take *N=3* for example. Consider the below matrix:

$$\begin{pmatrix} \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\ \vdots & \ddots & \cdots & \cdots & \cdots & \iddots & \vdots \\ \vdots & \cdots & x(i-1,j-1) & x(i-1,j) & x(i-1,j+1) & \cdots & \vdots \\ \vdots & \cdots & x(i,j-1) & x(i,j) & x(i,j+1) & \cdots & \vdots \\ \vdots & \cdots & x(i+1,j-1) & x(i+1,j) & x(i+1,j+1) & \cdots & \vdots \\ \vdots & \iddots & \cdots & \cdots & \cdots & \ddots & \vdots \\ \cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \end{pmatrix}$$

If $x(i-1,j-1) \times x(i-1,j) \times x(i-1,j+1) \times x(i,j-1) \times x(i,j+1) \times x(i+1,j-1) \times x(i+1,j) \times x(i+1,j+1) = 0$

$x(i,j) = 0$

$//otherwise, Result\ unchanged//$

5. N RMS cleaned

Region of Interest (ROI) is defined as the elliptical area having its center at the center of the mass of the full image distribution processed until now and extends to *N* RMS range. The *N* is an input set to 3 by default while *N=5* is usually considered as the conservative option. The pixels which are out of ROI are set to 0. The figure 3.2 shows the ROI.

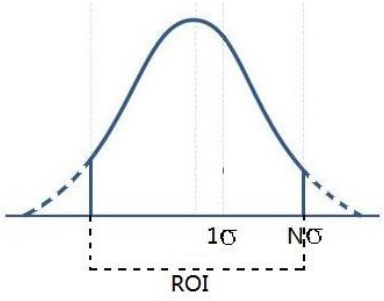

Figure 3.2: Schematic explanation of ROI construction and N-RMS cleaning procedure.

6. Median filter

A pixel is substituted with the median value of its immediate neighbors (3×3 matrix).

$$x(i,j) = Median(x(i,j), x(i-1,j-1), x(i-1,j), x(i-1,j+1), x(i,j-1), x(i,j+1), x(i+1,j-1), x(i+1,j), x(i+1,j+1))$$

The *median* is the "middle number" in a sorted list of numbers. To determine the median value in a sequence of numbers, the numbers must first be arranged in value order from lowest to highest. In the case of an odd amount of numbers, the median value is the value that is in the middle. Otherwise, the median value is the average of the middle pair.

7. Adaptive median

This is a median like filter where the new pixel value is calculated over a window with a variable size. First we calculate the median pixel value of the 3×3 matrix, and then the 5×5 matrix and so on. The iterations are not stopped until the difference between latest two new pixel values is smaller than a certain value.

$For\ i = 1{:}\ \infty$

$m = 2i + 1;$

$NewPixel(i) = Median(m \times m\ matrix);$

$IfAbs\big(\ NewPixel(i) - NewPixel(i-1)\big) < Level$

$Break;$

End

End

We can apply either median filter or adaptive median filter, not both. However, the adaptive median procedure is considerably more time-assuming.

iv. Momentum Analysis

The momentum distribution is obtained from the projection of deflected beam. Since we deal with image projection, the result is in general less sensitive to the image processing method applied. The current working version of the code is written in MATLAB.

1) Background preprocessing

The same way of treating background is used as in videoclient 3. Either the average of the backgrounds or the maximum of the backgrounds is taken into account.

$$Average{:}\ Bkg = \frac{\sum_{k=1}^{N} Bkg(k)}{N}$$

$$Envelope{:}\ Bkg = Max\{Bkg(k), \forall k = [1, N]\}$$

2) Simple Gamma-spike Cleaning

First the signal without background is obtained by subtracting the result from the previous step. Then a logical matrix is created where 1 represents the maximum pixel value while 0 represents those who're not above the saturation value (the max pixel value). Next, the logical matrix is applied to the signal without background. If the pixel value is not a maximum, it remains the same. Otherwise, it's replaced by its left side pixel value.

3) Baseline subtracting (applied to projections only)

Due to the noise induced by the signal itself, it's not enough for us to simply do background subtraction. Here we create a linear baseline by connecting the first and the last point of the image distribution curve and it is subtracted from the image distribution. See the Figure 4.1 below.

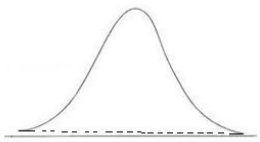

Figure 4.3: The dashed line represents the baseline to be subtracted for a projection

v. Slice Emittance

The code is written in C++. There are three image processing steps in this application. First the average background is calculated and subtracted from the signal. If the signal without background is smaller than *M* times RMS of the background, it is set to 0; otherwise it's unchanged. (*M* is an input variable)

$$If\ Sig\_without\_Bkg < M \times RMS\ \_Bkg, Sig\_without\_Bkg = 0$$
$$otherwise, Sig\_without\_Bkg\ unchanged$$

Next we define a variable *T* for each pixel representing threshold to do X-ray filtering. *T* is given by the average of the surrounding pixels plus 5 times RMS of the surrounding pixels. First we compare all the pixel values to their corresponding *T (i, j),* and mark all those above *T (i, j)* . Then we use the average of the good neighboring pixel values(less than their corresponding *T (i, j)*) to replace the marked ones. The depth of the neighborhood is adjustable.

$$T(i,j) = Average\_Sig + 5 \times RMS\_Sig$$
$$If\ C(i,j) > T(i,j), C(i,j) = Average(GoodNeighbours)$$

If we take the table below into consideration, then we have a new C1 (in the case of considering 3×3neighborhood):

G1	G2	G3
G4	C1	G5
G6	G7	G8

$$C1 = \frac{\sum_{m=1}^{7} Gm}{7}$$

$G8 > T(G8)$, therefore not taken into account

The last step is so-called anti-flickering filtering. For each pixel, if the signal without background is 0 for 70% of the frames, it is set to 0 for each pixel. Moving islands of signal are consequently reduced by applying this filter. Afterwards, an adjustable depth of border is restored in the same way as done for beamlets in the Emcalc.

Acknoledgements

I'd like to thank Marek Otevrel, my supervisor, for his kind guidance and support. I would also like to thank PITZ group for the kindness and lively scientific discussions. Especially thank to Stefan Sweisse and Galina Asova for their willingness and patience to answer all my questions. I want to thank Karl Jansen for his devotion to the summer students and all the people who involved in the organization of the summer students program. Lastly, I want to thank DESY for giving me the opportunity to take part in the summer students program.

Reference

[1] Weisse.S and V.Miltchev., Video Client 2 User Documentation., http://adweb.desy.de/mcs/tine/VideoSystem/vsv2doc/Video%20Client%202%20User%20Documentation_rev2.pdf .,01-04-2004.

[2] G.Asova et al., "New beam diagnostic developments at the Photo-Injector Test Facility PITZ," *Particle Accelerator Conference*, 2007. PAC. IEEE , vol., no., pp.3967,3969, 25-29 June 2007
doi: 10.1109/PAC.2007.4439943.