Fluoroscopy using the EPID

June 17

2010

This is the graduation paper of Tim Lustberg student Electronic Engineering. It explains all the work done for MAASTRO clinic.

The software interface between Matlab and the XRD-FG

Author: Tim Lustberg

Company supervisor: D. Emans

School supervisor: R. Jongen

Version: 1.0

Acknowledgement

Title	Fluoroscopy using the EPID
Graduation company	MAASTRO clinic
Address	Dr. Tanslaan 12
	6229 ET Maastricht
Mail address	Postbus 5800
	6202 AZ Maastricht
Telephone number	0031- (0)88 44 55 666
Fax number	0031- (0)88 44 55 667
E-mail	info@MAASTRO.nl
Company supervisors	D. Emans
	B. Nijsten
	L. Persoon
Department	Technical innovation and Quality control
School superviser	R. Jongen
Education	Hogeschool Zuyd Electronic Engineering
Author	Tim Lustberg
Education	Hogeschool Zuyd Electronic Engineering, Electronics and Telecommunication
Graduation date	24th of June 2010
Graduation period	1st of February 2010 till 24th of June 2010

Preface

My name is Tim Lustberg. This graduation report is the result of my internship at MAASTRO clinic to obtain my bachelor's degree in Electrical Engineering with a mayor in Electronics and Telecommunication at the Faculty of Engineering at the Hogeschool Zuyd in Heerlen.

I would like to thank Davy Emans for helping me finish this project successfully. His knowledge about working in projects helped with my project planning.

I would like to thank Lucas Persoon for helping me improve my C++ knowledge. Without him I would not have been able to complete the Matlab acquisition library.

I would like to thank Bas Nijsten for helping me understand the EPID mechanisms and for providing the Matlab code to test the image corrections.

I would like to thank Ramon Jongen who brought me in contact with MAASTRO clinic and helped me choose my graduation internship. Without his encouragement I would probably not have taken this risk of an assignment that is unusual for a student of Electronics and Telecommunications. I would have missed out on a great experience.

And last but not least I would like to thank the TIQc and KFG for their assistance with the equipment and all the questions they answered.

I hope you enjoy reading my graduation report.

Summary

Problem description

MAASTRO clinic wants to guarantee the quality of their treatment and continuously aims to improve it. Currently the Electronic Portal Image Device (EPID) is only used to verify the patient's position during treatment, but the EPID has much more potential than just position verification. The EPID can be used for multiple purposes and therefore MAASTRO clinic is developing their own software. Matlab is the most common software-developing -environment in MAASTRO clinic. Unfortunately, at this moment it is impossible to communicate with EPID in Matlab. This requires the use of a frame grabber to acquire frames from the EPID. This frame grabber is delivered with a C++ library to acquire frames. The graduation assignment is to use Matlab EXecutable files (MEX files) to build a set of functions to communicate with EPID in Matlab. To acquire these frames correctly it is necessary to know when the linear accelerator (linac) is sending photon beams. The linac is sending this beam pulsed, every time a trigger signal becomes low. The other part of the assignment is to investigate the possibilities of using this trigger signal which can be read out in the operating room of the linac.

Hardware interrupts

The linac trigger is not suitable to directly connect to the frame grabber's interrupt input. It needs to be scaled to TTL level and because the frame grabber is an expensive device it needs to be protected with an optic protection circuit. After investigating this matter it was determined that it is possible to use the linac trigger, but there was still much work to do. Because the graduation period was limited it was decided that the acquisition software had priority over the hardware. The work is documented and should be further investigated in another project.

Acquisition

The frame grabber is delivered with a C++ library. This library contains a set of functions that allows the user to communicate with the EPID and acquire frames. But MAASTRO clinic develops their applications in Matlab because it is best suited to their demands. To be able to use C++ code in Matlab, Matlab Executable files (MEX files) are needed. These MEX files are the bridge between Matlab and C++. The first step was to create a C++ program that could acquire frames successfully. The next and final step was realizing a communication structure between Matlab and C++, which made it possible to acquire EPID images in Matlab. The end result was the sequential acquisition of EPID images in Matlab. Every acquisition variable can be set in Matlab so the function can be used by all users to their own needs.

Corrections

The images acquired still have flaws due to the hardware properties. The cells of the EPID contain photodiodes and it is impossible to manufacture photodiodes with exactly the same characteristics. These corrections consist of: a dead pixel correction, an offset correction and a gain correct. A dead pixel correction is needed to correct for broken pixels. The offset correction is to correct the different bias currents of each cell. And the gain correction corrects the different gain of each cell. The gain of each cell is different because every photodiode reacts slightly different to the same amount of photons. The assignment was to enable the creation of the correction images need to correct for these flaws. The methods to correct these images are known within MAASTRO clinic but it was not yet possible to acquire these correction images in Matlab. The functions to create the correction images together with the image acquisition functions are the Matlab acquisition library which forms the end product of the graduation assignment.

Conclusions & Recommendations

The hardware interrupt design needs to be further evaluated in a different project. The design needs to be evaluated by an experienced hardware engineer before the production the hardware.

The acquisition functions enable the programmers of MAASTRO clinic to acquire images in Matlab. The continuous acquisition is not jet able to show the images real-time. The output containing the frames is not available during acquisition. This makes the function not suited for real time processing. The continuous acquisition function did lead to a set of offline reconstructions containing information that has great value to MAASTRO clinic.

It is possible to create correction images in Matlab. A mayor improvement would be a function that could create a set of correction functions with the different available camera modes and integration times.

The last recommendation is a personal recommendation to find an internship that is the challenge you have been looking for without being worried about the appendages.

Table of Content

1 Introduction

This chapter will provide the necessary background information to understand the purpose of the project and understand how patients are treated at MAASTRO clinic, what cancer is and what the effects of ionized radiation has on tumours. It explains the basics of a linear accelerator (linac) is and the Electronic Portal Image Device (EPID) as well.

1.1 Patients of MAASTRO Clinic

When a patient is diagnosed with cancer, several treatment options are: chemotherapy[1], surgery, radiation therapy of a combination of these options. MAASTRO clinic is a radiotherapy institute in Limburg and treats approximately 3000 patients a year.

The first step of the diagnostic trial in MAASTRO clinic is an intake appointment with the patient by a physician. The physician will explain what will happen during the treatment and will answer questions. The next step will be to create all the necessary CT-images of the patient for treatment planning. The CT-images form a 4D model of the insides patients. The position of the patient during the CT-scans is marked with waterproof ink on their body. The patient lying on the table while horizontal and vertical lasers mark the way they are positioned during the scan. In both the CT and treatment rooms a laser-system is used to position the patient on the markings. Treatment planning will be done by a physician and a radiation technologist. The physician determines where the tumour is located and marks the images. The tumour volume is determined and the Organs At Risk (OARs) near the tumour are delineated.

The radiation technologist continues with the treatment planning. A lot of guide lines need to be followed during this procedure. The dose[2] to the healthy tissue needs to minimized, the OARs need to be avoided while the dose in the tumour needs to be conform the prescription of the physician.

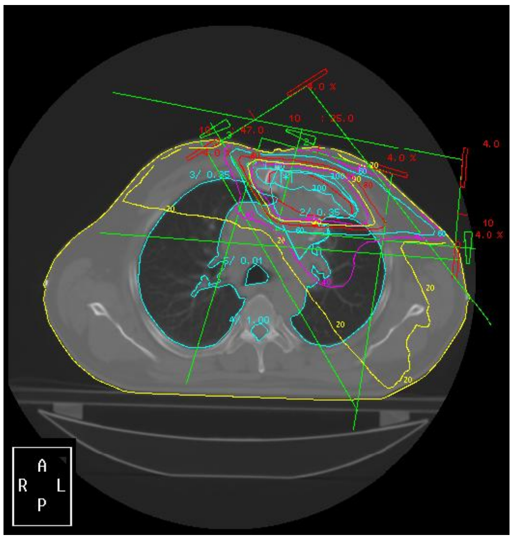


Figure 1 Example 2D CT image with planning markers

To save as much healthy tissue as possible the tumour is radiated from different angles while avoiding the OARs marked by the physician. When the radiation technologist is done a physician checks the complete treatment plan to be sure the patient will be treated correctly.

The next step is the actual treatment of the patient. To make sure the patient is positioned perfectly during the treatment, the patient is placed according the markings made during the preparation. The accelerators are equipped with an EPID which can be used to make static images using the photon beams of the treatment.

[1] *Chemotherapy* is the treatment of cancer patients with the use of cytostatics.

[2] Dose is the amount of radiation delivered to the patient. More information can be found in enclosure 1.

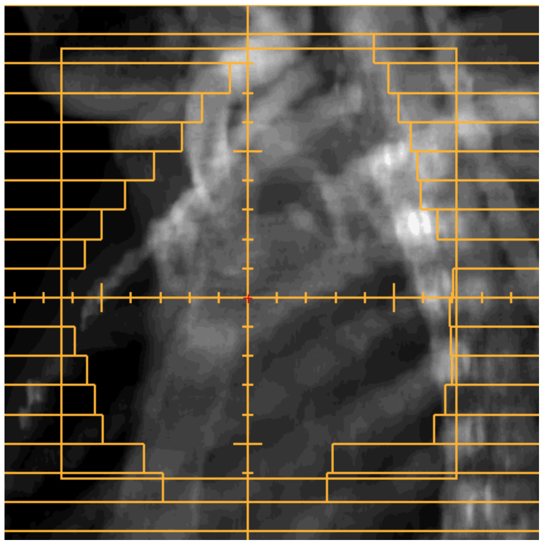


Figure 2 Digital Reconstructed Radiograph

The EPID is used to check the patient positioning by matching bony anatomy or fiducial markers. The set-up is verified by comparing the EPID image with CT-images (planning position compared with treatment position). Once the position is verified the treatment can start according to plan. Figures 2 and 3 give an example of 2 images that are compared to determine patient location.

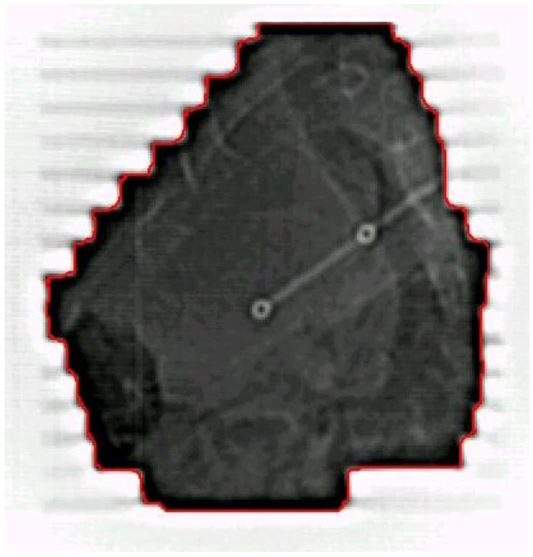


Figure 3 EPID image

1.2 Cancer and Radiation therapy

The definition of cancer is an abnormal growth of cells which tend to proliferate in an uncontrolled way and, in some cases, to metastasize (spread). This definition leads to two groups of cancer: Benign and malignant. A Benign tumour does not infiltrate surrounding tissue while a malignant tumour does, disabling the surround tissue to work the way it should. Worst case a malignant tumour can infiltrate and a blood vessel and admit cells into the bloodstream. This can cause another tumour somewhere else in the body. An example of both types can be found in Figure 4 and 5.

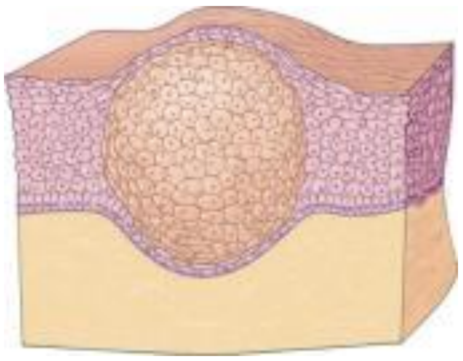


Figure 4 Benign tumour

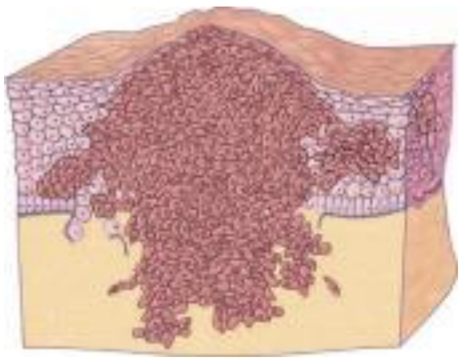


Figure 5 Malignant tumour

A mutation in the DNA causes the cells to divide rapidly. Radiation therapy damages DNA with photon beams in the MegaVolt (MV) spectrum. The chapter about the linear accelerator will give a more detailed explanation about the type of ionizing radiation used in radiotherapy. For healthy cells this damage means they will stop dividing and repair their DNA. Cancer cells have lost this ability and will die because of this radiation induced mutation. Figure 6 gives an illustration of the effect of radiation on cells.

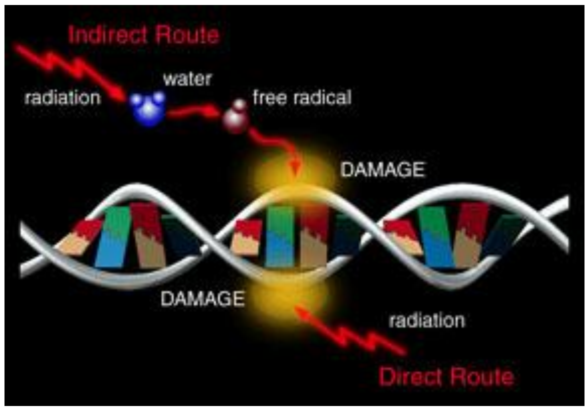


Figure 6 Damaging DNA with radiation

The most effective way to cure the cancer is to deliver small amounts of equivalent dose frequently. For example: The patient is radiated five times a week with a rest period in the weekend. This gives the healthy tissue more time to recover and thus decreasing the change of causing a new tumour because of the treatment.

1.3 Linear Accelerator

A linear accelerator is a device that produces high energy photons (MV) or electrons (MeV). The mean parts of the machine are:

1. Injector (Electron Gun)
2. Waveguide
3. The bending envelop
4. The target
5. The Collimator

Figure 7 and 8 illustrate these parts. The injector releases electrons in pulses. These electrons are accelerated by the waveguide, a tube which is connected to a RF generator. Because of the construction of the waveguide the electrons ride the RF wave, like a surfer rides a wave in the ocean, accelerating them to light speed. According to Einstein's $E = mc^2$ the mass of the

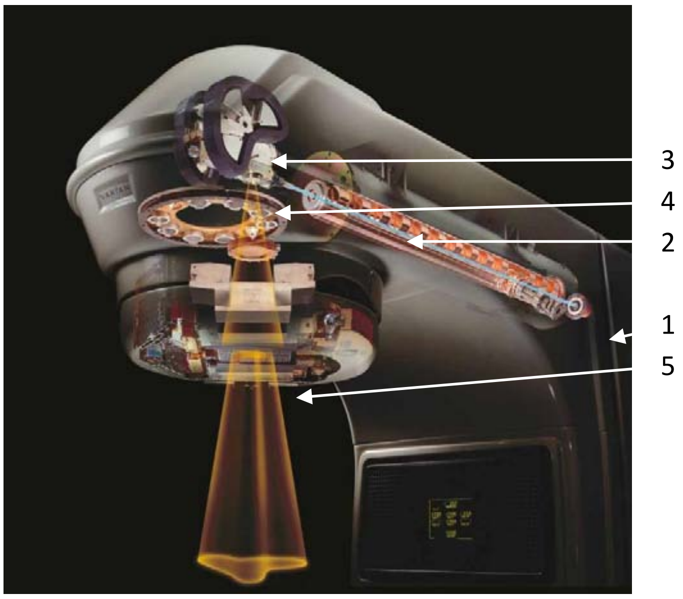


Figure 7 Linear Accelerator

electron will increase if energy is added after the electron reaches light speed. At the end of the waveguide all electrons contain huge amounts of energy. After leaving the waveguide the electrons enter the bending envelop. This bending envelop changes the course of the electrons using bending magnets and filters the beam. Electrons with the right amount of energy make 270° turn, the rest will not. This explains the 270° instead of 90°, the leaving electrons would go towards the patient if the beam would bend 90°. Next the beam will hit the target. The target contains heavy metal atoms which collide with the electrons. This collision produces photons with the same amount of energy as the electron which collided with the heavy metal. If the patient is treated with electron beams, the target absent.

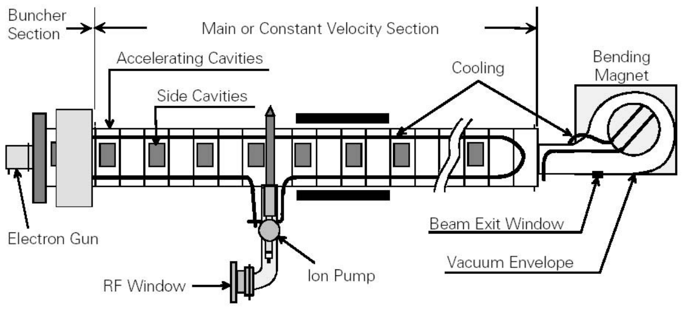


Figure 8 Accelerator components (schematically)

The beam now enters the collimator. The collimator shapes the beam. The equivalent dose needs to be homogenously distributed to the tumour. The multi leaf collimator shapes the beam to the form of the tumour to decrease the amount of radiated healthy tissue. Figure 9 is an illustration of a multi leaf collimator and figure 10 is one of the accelerators present at MAASTRO clinic.

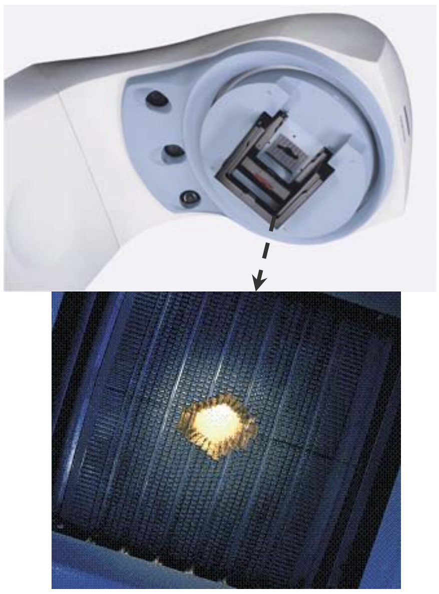


Figure 9 Multi Leaf Collimator

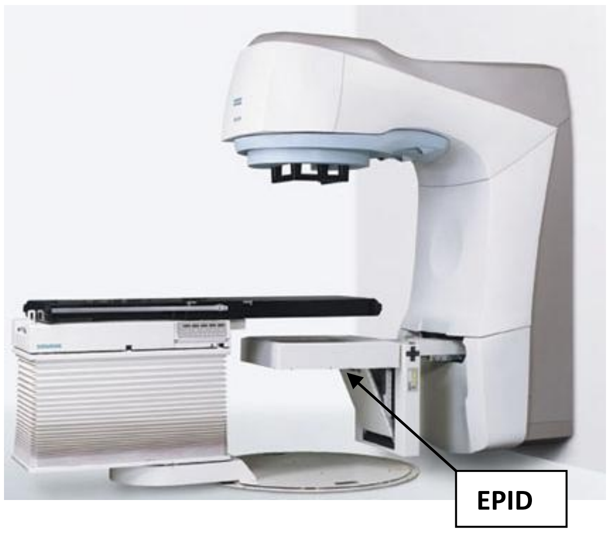


Figure 10 Siemens Artiste Linear Accelerator

1.4 EPID

The EPID is a panel which detects X-rays. In figure 10 the panel is located under the table in straigt under the head of the accelerator. The panel's internal logics are similar to a dynamic RAM (see figure 11).

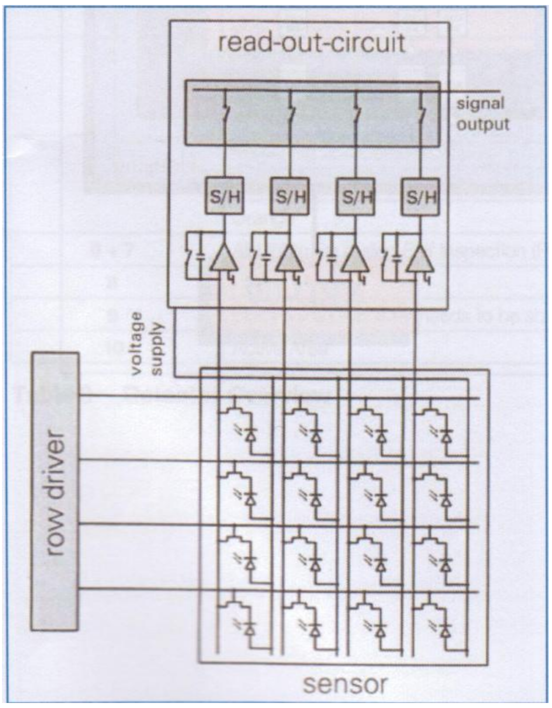


Figure 11 EPID internal logics

The row driver selects a row and every column a cell is read. These cells consist of a Field Effect Transistor (FET) used as switch and a special photodiode. This photodiode is not sensitive to photons in the visible spectrum but to photons in the x-ray spectrum. The amount of photons that goes through a cell determines the voltage it produces (like a capacitor). This voltage can not rise to infinite hights because it reach the maximum capacitor capacity. At the moment the cell is read out the capacitor will discharche and the cell is able again to detect new photons. It is important to periodically read every cell. This period is called the integration time. Converting the voltage produced by every cell into grayscale results in an image.

The EPID consist of several segments (in our case 16) which contain the structure as described above. So each segment has its own row drive and memory to store information. All segments combined result in an image like shown in figure 3. Figure 12 is what the EPID casing looks like. A more detailed description will follow in the corrections chapter.

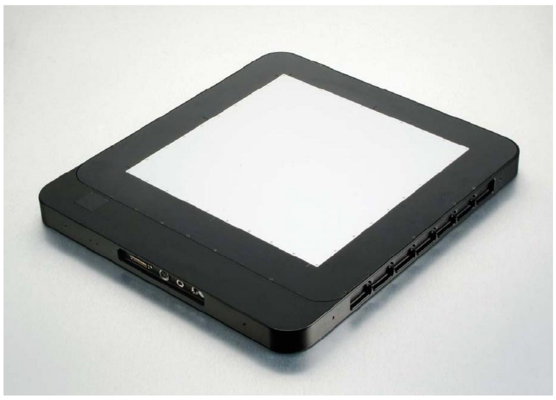


Figure 12 EPID

1.5 Frame grabber

For this project MAASTRO clinic purchased a XRD-FG [3] (figure 13) from Perkin Elmer. This frame grabber can be used to acquire images from the EPID. The manufacturer provides a simple software package to demonstrate all the functions available, called XIS[4]. XIS provides a C++ library as well allowing the user to create a custom acquisition program to their own liking.

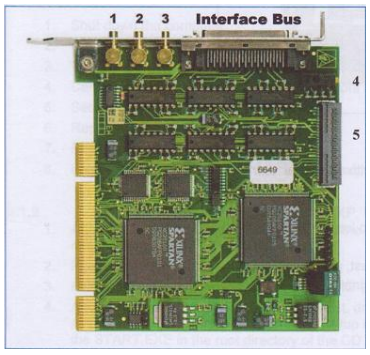


Figure 13 XRD-FG

[3] **X**-**R**ay **D**etector **F**rame **G**rabber
[4] **X**-ray **I**maging **S**oftware

2 Problem description

Within MAASTRO clinic the TIQc[5] guarantees the quality of the treatment by checking and adjusting the linear accelerators, CT-scanners and other medical equipment used. Some of these tests require a lot of preparation and measurement equipment, for example the use of a water tank. This is a large bin with water which is used to measure the dose delivered by the beam with specialized detectors. All this work could be (partially) simplified if the equipment is replaced with the EPID. But to effectively do this it is necessary to create a fluoroscopy[6] instead of static images. The images acquired with EPID are images with flaws that need to be corrected to be useful. After the use of dead pixel, gain and offset corrections the images can be useful in numerous applications. Finally to correctly acquire the images certain interrupt signals from the linear accelerator have to be processed. So the assignment consists of:

- Hardware interrupt interface
- Image acquisition
- Image corrections

2.1 Hardware interrupt interface

For the reconstruction[7] of the delivered dose out of EPID images it is very important to know when the linac starts and stops beaming. The linac itself starts giving pulses when the beam turns on. The frequency of these pulses represents the dose rate. The linac stops these pulses when the beam is turned off. The frame grabber mentioned in the introduction chapter has multiple trigger inputs. The question is of these pulses from the linac can be uses by the frame grabber to acquire images so that they are suitable for dose reconstruction.

[5] Department of **T**echnical **I**novation and **Q**uality **c**ontrol
[6] A film of x-ray images
[7] More information about dose reconstruction can be found in enclosure 1

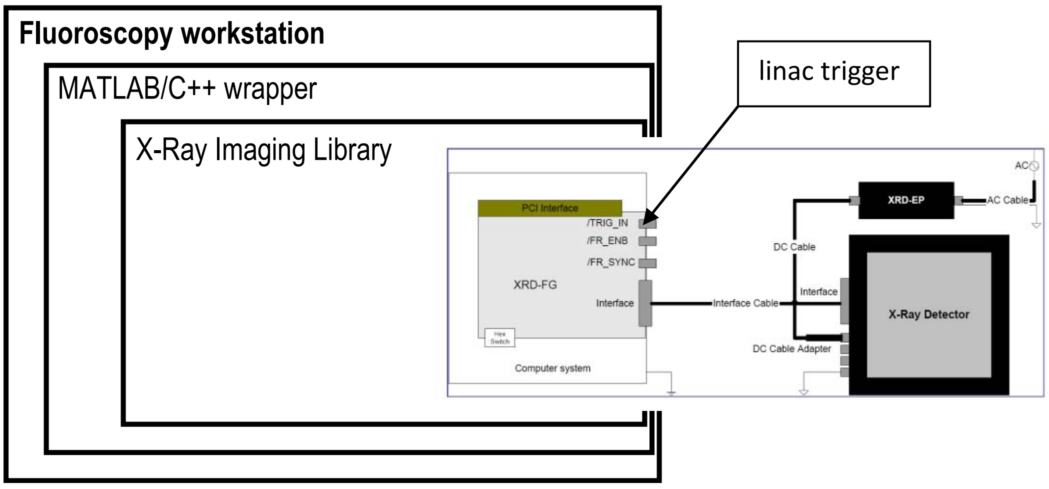


Figure 14 Schematic project description

The assignment is to investigate the possibilities and design the hardware interface needed to handle the interrupts. Figure 14 shows a schematic representation of the project's hard and software.

2.2 Image Acquisition

Because Matlab is very powerful tool for image processing and data structures almost all programming within MAASTRO clinic is done with Matlab.

To use the frame grabber easily in Matlab it is necessary to make all the acquisition parameters available in Matlab. This way the acquisition functions can be uses by anyone in any way they would like to acquire their images.

The two modes that need to be available are single frame and continuous acquisition. Both have to be adjustable so that the designer of the implementation software can achieve what he or she wants.

If there is time left at the end of the project a small example program can be made demonstrating the use of the created functions.

2.3 Image Correction

The EPID returns an uncorrected image during acquisition. Due to the internal logics of the EPID this image has three errors that need to be corrected:

- **Dead pixel correction**

Some cells don't react to photon beams anymore. The internal mechanism of the EPID detects these pixels and marks them. These pixels need to be filtered after acquiring the image.

- **Offset correction**

The cells of the EPID return a voltage even if they are not receiving any photon beams. This is called the offset. Unfortunately not all cells return the same offset. An offset image contains this offset value for every cell so that the offset correction can be applied.

- **Gain correction**

The relationship between the cell voltage and the intensity of the photon beam is linear. But the slope of every cell is different. A gain image contains the information needed to calculate the slope of each pixel.

To be able to correct these flaws in the acquired image it is needed to acquire correction images in Matlab.

3 Hardware Interrupt Interface

This chapter explains the work done for the design of the hardware interface. It contains an analysis of signal from the linac, a concept design to convert the signal to TTL level, and a recommendation for further research of the problem. The recommendation is needed because this part of the project had so much work involved that it was excluded from the graduation assignment. The Matlab functions had priority over hardware. This means that the designs in this chapter are concepts that need to be tested and revised by an experienced hardware engineer.

3.1 The linac signal

The linac provides a pulse every time a photon beam is released. This pulse is a spike from 400mV to -21V. The spike reaches 400mV again after only 4 microseconds long. Figure 15 is the simulation of what the pulse looks like (in red). The frequency of these pulses is called the dose rate.

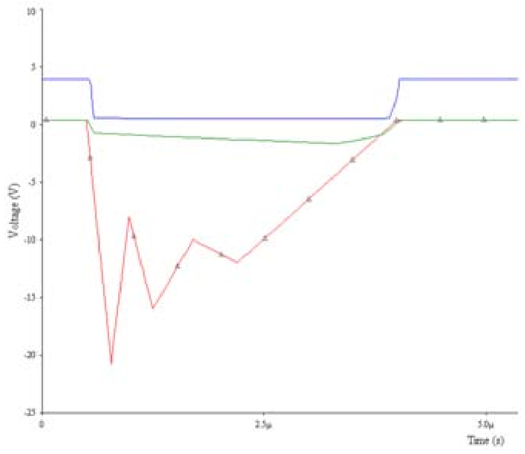


Figure 15 Pulse detection transient
in Red : Linac signal
in Green: : Pulse detection signal
in Blue: : Pulse detection signal converted to TTL

This pulse has to be connected to the trigger input of the frame grabber. However the signal specifications for the frame grabber trigger inputs are a low active TTL levelled pulse of at least 20 microseconds. The pulse needs to be adjusted before it can be used for the intended purpose.

3.2 Pulse adjustment

The linac signal needs two mayor adjustments. The voltage level of the pulse is incorrect and the length of the pulse is too short for the frame grabber to detect.

So the first problem is detecting the pulse and scaling it to TTL level. The initial design is to use a fast detection circuit containing a LT1190. This is an ultrahigh speed Operational Amplifier (OpAmp). This OpAmp circuit filters the peek out of the original pulse (figure 15, in green). After filtering the signal it is still not at TTL level. A second LT1190 is connected to the output of the first detection circuit. Because this OpAmp does not have any feedback it will immediately hit the voltage roof when the input signal is above 0 Volts and sink the output to 0 Volts if the input signal is below 0 Volts. This transforms the signal into a TTL compatible low active signal (figure 15 in blue).Now the pulse needs to be extended so that the frame grabber can actually detect it. A LM555 timer is an IC that extends the pulse. The length of the pulse is determined by the setup of the resistor capacity combination connected to the IC. Figure 16 shows the input in red and the extended (inverted) pulse in green. The purple signal is the threshold and the blue signal is the voltage over the capacitor. The pulse is extended until the capacitor reaches the threshold.

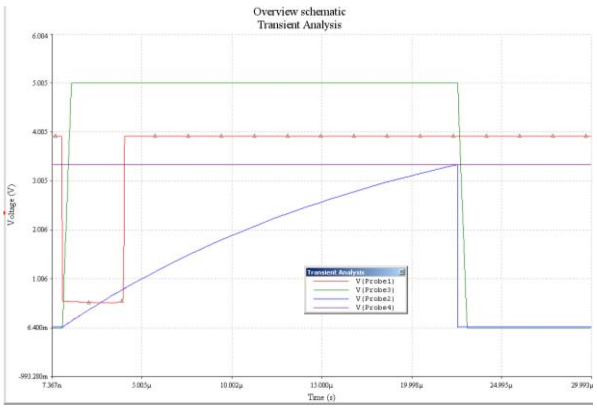


Figure 16 Timer signals transient
in Red : TTL Pulse detection signal
in Green : Extended inverted TTL pulse
in Blue : Timer capacity threshold
in Purple : Capacity voltage during extension

To invert the signal back to a low active signal it is send to the gate of a BS170 FET which is used as a digital switch connected to a pull-up. A high voltage on the gate activates the FET's channel so that the current can flow to the source of FET which is connected to the ground. Sinking the drain current, this leads to a voltage drop to almost 0 Volts. Not 0 Volts exactly because the channel resistance does not go to 0 Ohm so the voltage is divided over pull-up resistor and the channel resistance. When the gate is 0 Volts and the FET's channel is closed the drain voltage goes to almost 5 Volts. But again because the channel resistance is not infinite the voltage will be divided over the pull-up resister and the channel resistance. Figure 17 shows the pulse coming out of the pulse detection circuit in red, the extended pulse in blue, and the inverted pulse in green.

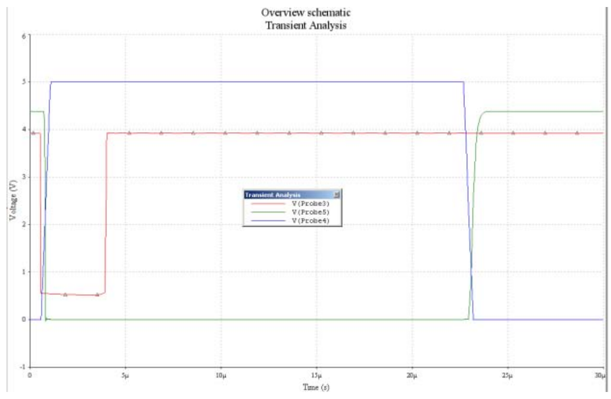


Figure 17 Pulse extension signals
in Red : TTL Pulse detection signal
in Blue : Extended inverted TTL pulse
in Green : Extended TTL pulse (behind FET)

The problem of the pulse adjustment circuit is that it delays the pulse. This may lead to loss of information which could lead to an error in the dose calculation. The end of chapter 3.3 will give more detailed information about the total delay of the entire pulse detection hardware.

3.3 Optic protection

Because the frame grabber is very expensive piece of hardware it is recommended to protect it against voltages out of TTL range. If for whatever reason the linac trigger signal would reach the trigger input of the frame grabber it could be destroyed. To do this the extended signal is send to a HCNR200 optocoupler which is set up like prescribed in the datasheet. The datasheet describes this setup as the high speed low cost circuit which minimizes the delay from in to output. Figure 18 shows the linac pulse in green and the converted, extended and optic separated pulse in red. Figure 19 shows the in- to output delay of the entire circuit. The total delay is about 0,5 microseconds. If this delay causes errors in the acquisition needs to be evaluated.

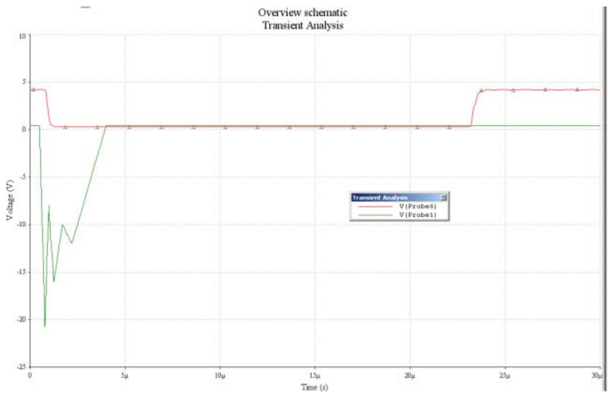
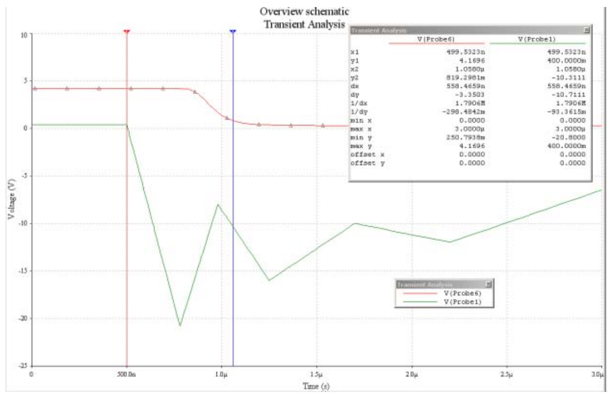


Figure 18 Hardware interrupt interface in and output
in Red : Output (TTL converted extended pulse)
in Green: : Input (linac pulse)



	V(Probe6)	V(Probe1)
x1	499.5323n	499.5323n
y1	4.1696	400.0000m
x2	1.0580µ	1.0580µ
y2	819.2961m	-10.3111
dx	558.4659n	558.4659n
dy	-3.3503	-10.7111
1/dx	1.7906M	1.7906M
1/dy	-298.4842m	-93.3615m
min x	0.0000	0.0000
max x	3.0000µ	3.0000µ
min y	250.7938m	-20.8000
max y	4.1696	400.0000m
offset x	0.0000	0.0000
offset y	0.0000	0.0000

Figure 19 Delay in to output
in Red : Output (TTL converted extended pulse)
in Green: : Input (linac pulse)

The complete schematics can be found in enclosure 2.

4 Acquisition

The mayor part of the graduation assignment was to create Matlab functions that acquire EPID images. This chapter explains the mechanisms of the acquisition functions created in C. This requires knowledge about **M**atlab **EX**ecutable files (MEX files) and the acquisition handles described by the XIS library. A MEX file is a C file with a Matlab include and a specific function header body.

4.1 MEX files

A MEX file is the bridge between Matlab and C. This is needed because the XIS library is written in C while MAASTRO clinic wants to acquire frames in Matlab. This paragraph only gives a short introduction to MEX files.

nrhs,nlhs, prhs[] and plhs[]

The 4 names mentioned above are the inputs of the MEX function given by Matlab.

- nrhs: The number of given inputs.
- nlhs: The expected amount of outputs
- prhs[]: An array containing the inputs. The inputs go from prhs[0] to prhs[nrhs-1].
- plhs[]: An array that can be used to store variables that need to be returned to Matlab.

Note that the types of the in- and outputs are not given. For example: If a MEX-functions with one input is used, and this input is an integer, the variable used in C++ to store this Matlab variable needs to be declared as in integer. To prevent errors in the application it is very important to keep track of input types. For the output it is less complicated. The MEX array created to store the outputs is declared in the C layer where the types of the variables are known.

The input is entered as a structure of variables. This means nrhs = 1 and that all variables are stored in prhs[0]. This input is a structure of the type mxStructArray and the variables are collected by name like in code sample 1.

```
//if the field name is BufferSize save in MexAcqVar.nBufferSize
if ((sNameCompare.compare("BufferSize"))==0)
{
    pMexFieldValue             = mxGetField(prhs[0], 0,pStringBuffer);
    pMexAcquisitionOptions     = mxGetPr(pMexFieldValue);
    MexAcqVar.nBufferSize      = (UINT)pMexAcquisitionOptions[0];
}
```

Code Sample 1

To return variables to Matlab mxArrays are needed again. The variables returned to Matlab are returned as a structure of variables like the input variables. Two structures are returned: The first one contains the acquired frames and the second contains information about the acquisition settings. An example of the output structure can be found in code sample 2.

```
strFieldNames[0] = "Columns";
mxAddField(plhs[1], strFieldNames[0]);
mxArray *pMxCOLUMNS;
pMxCOLUMNS = mxCreateDoubleMatrix(1,1,mxREAL);
double* pStructPartCOLUMNS;
pStructPartCOLUMNS = mxGetPr(pMxCOLUMNS);
pStructPartCOLUMNS[0] = (double)MexAcqVar.nColumns;
mxSetFieldByNumber(plhs[1],0,1,pMxCOLUMNS);
```

Code Sample 2

More information is provided by the Mathworks website in the sources section. All the functions and types are explained there.

4.2 Acquisition handles

The acquisition has a set of handles that needs to be kept in the physical memory during the entire acquisition. This requires a certain setup of programming. The acquisition handle is created at the start of every MEX function. This means the sensor needs to be initialized every time Matlab calls a MEX acquisition function. This introduces a ±1 second delay before being able to acquire frames, but ensures the existence of the acquisition handles during acquisition. The initial idea was to have a separate initialisation function in Matlab, making the frame acquisition MEX functions faster. However the acquisition handles went out of scope right after the initialisation MEX function terminated and caused the acquisition to fail.

4.3 XIS shared functions header

This header file was created to contain all code the MEX acquisition functions have in common. These are the acquisition structure, the initialisation routine and the acquisition callbacks.

Acquisition structure

Because there are several sub functions used in the code it is easiest to pass along the variables as part of a structure. The acquisition structure defined in the header file contains all necessary variables needed for successful acquisition. The most important variable is the pointer to the acquisition handle. After creating the handle at the start of every MEX function the pointer to this handle is stored in the acquisition structure so it can be accessed in every part of the acquisition process.

Initialisation

The initialisation process is the same for every type of acquisition which made it a suitable candidate for the header file. During the acquisition all information about the sensor is collected (i.e. the number of rows and columns). The user has two variables he or she needs to set. The sensor position and the sensor gain. There can be more sensors connected to the frame grabber so the user can choose the wanted sensor by setting the position variable. Since all linacs are only equipped with one sensor the position should be 0 by default. But the software is capable of initializing sensors in other positions as well.

Callback functions

The acquisition is interrupt based. The frame grabber asks for a frame after that the EPID reacts with an interrupt when the frame is ready. The time between these two interrupts is equal to the integration time of the panel. The integration time is the time that the panel keeps reading cells and integrating the cell voltage. The integrated values of every cell create the raw frame returned by the EPID.

To acquire the frames it is necessary to define how the software should react to the EPID's interrupts. These are the callback functions. There are two callback functions available. When a frame is collected and when the whole acquisition is completed. Both callback functions are the same for our application. At the start of the callback functions the first thing to do is to determine what kind of acquisition mode is used. The single frame acquisition requires a different action than for example the gain correction image acquisition.

After a frame is acquired it is possible to collect the frame header in a structure defined by the XIS library. This header information has all sorts of basic information about the sensor and the acquired frame. The interesting part was the extended header info, which contains data like the actual time the frame was integrated. This information could be very valuable when trying to calculate dose out of the acquired images. Unfortunately the EPIDs used in MAASTRO clinic do not yet support the extended header information feature.

4.4 Single Frame Acquisition

After explaining all the facts about the MEX functions, the acquisition handles and the shared function header it is possible to explain the acquisition routines, starting with the single frame acquisition. This function returns a single frame as a numeric matrix and the number of rows and columns in the form a structure. This structure with the number of rows and columns is just an example of how to return variables from C++ to Matlab. The user can add or remove variables he or she thinks are valuable to have in Matlab by following this example.

MEX function

After giving the Matlab command the program starts the MEX function. All in and output handling is done in this function. The first action is to collect all input variables and storing them in the acquisition variables structure. To program loops until a counter reaches the number of fields of the input structure. The loop checks the field names and adds the value of that field to the corresponding variable in the acquisition structure.

After collecting all input variables the acquisition handles are created and the pointers to these handles are stored in the acquisition structure. Before entering the actual acquisition routine the output mxArray needs to be created. The pointer to these mxArray is passed along to the acquisition routine together with the acquisition structure.

GetSingleFrame

The acquisition functions starts with initializing the sensor. After successfully initializing, the acquisition mode can be set. Only two of the four available modes are defines, the free running and internal trigger mode. Free running mode explains itself. When the internal trigger mode is selected a frame is captured with a user selected frequency. This should be in line with the selected integration. For single frame acquisition this does not have any effect but when continuously acquiring frames it very important. The paragraph about continuous acquisition will explain why.

The next step is creating the memory buffer to store the acquired frame and direct the XIS library to this buffer. The end of acquisition event is to let the main program know that the callback function is done. This event has to be declared before the program requests an image. The program waits until the callback sets the event. During the callback the frame is stored in the acquisition buffer and the end of acquisition event is set.

The program continues with moving the frame data from the acquisition buffer to the mxArray and returns to the MEX functions. The MEX function returns the structure with acquisition information, then closes the acquisition and returns to Matlab. The image can now be build from the numeric matrix with the Matlab imaging toolbox.

4.5 Continuous Acquisition

The continuous acquisition is not that different from the single frame acquisition. The real differences are that the frames are returned in a structure of numeric arrays instead a single numeric array and the event handling.

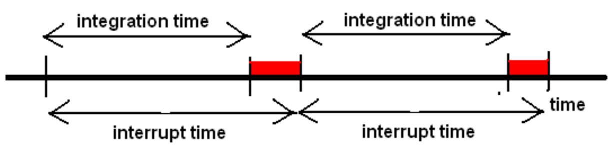


Figure 20 Timing of internal timer mode

The first thing to consider is the relation between integration time and internal trigger frequency. Figure 20 shows schematically this relation schematically. If any photon beams hit the EPID during the time marked in red they will not be detected. This leads to a loss in information and can lead to errors in the dose calculation. Ideally the integration time and interrupt time should be the same. This is only possible if one integration time is enough to process the frame before the next one arrives.

Like with single frame acquisition the mxArray to return the frames to Matlab is created before the acquisition. But it is a structure mxArray with one field instead of a numeric matrix. This field will contain the first frame acquired. Before acquisition the end of frame event is set instead of the end of acquisition event like in single frame acquisition. After starting the acquisition the program enters a loop where in the beginning the program waits for the end of frame event. When the event is set it means a new frame is ready to be processed. For the first frame it is directly moved to the first field of the output structure mxArray. For every other frame the program adds a now field to this array every time the end of frame event is set. The fieldnames consist of the word "Frame" and a number with is determined by a counter which registers how many frames have been collected.

When the number of frames set by the user are collected the acquisition is aborted. This means the program will not get the end of frame event anymore. The function that waits for this event had a timeout limit set to three times the interrupt time. If the timeout occurs it means the acquisition is done. It is impossible to wait for two events without multithreading. This is why the end of the acquisition cannot be detected with the end of acquisition event. For now it was best try to come up with a solution without having to use multithreading. But now when the timeout occurs it can mean two things. Besides the acquisition being complete, it is also possible that the communication with the EPID is lost. This means that it is impossible to detect this error because this mechanism. But because the reaction to both situations is to store the collected frames and return to Matlab, it was not a big issue.

It was not yet possible to create a start and stop sign for the continuous acquisition. This is why the number of frames has to be set before hand. The problem here is the limited amount of physical memory. The frames are stored in the physical memory of the computer and returned to Matlab. A cell value of the EPID is 2 bytes. This means an EPID with 1024 by 1024 pixels needs 2 Megabytes (Mb) of memory for one frame. The amount of memory on the test pc was rather limited, only 512 Mb. This memory is used to run windows and all your other applications as well. Even if the full 512 Mb was available the amount of frames that could be acquired is limited to only 256 frames. To prevent the user from selecting more frames than possible the program checks the available physical memory before starting acquisition. If the wanted number of frames exceeds the maximum amount of frames that can be stored in the physical memory, the program notifies the user and collects the maximum amount of frames possible.

To deal with this issue a second continuous acquisition function was created. Instead of returning the acquired frames to Matlab through physical memory it writes the acquired frames to the harddisk. This has one mayor disadvantage: Disk IO is slow. So the lowest integration times are impossible to use. Longer integration times increase the chance that the EPID is overexposed. But it was the best solution when acquiring frames with older hardware.

4.6 Results

Testing the acquisition functions was success. With the use of the continuous acquisition function that stores to hard drive it was possible to record an **I**ntensity **M**odulated **R**adiation **T**herapy (IRMT) sequence.

This treatment method beams small parts off the tumour in steps instead of beaming the entire tumour at once. Figure 21 shows the results of acquiring the IMRT sequence. The frames are presented from left to right. The frame in the right bottom shows the accumulation of all frames. This is approximates the shape of the tumour.

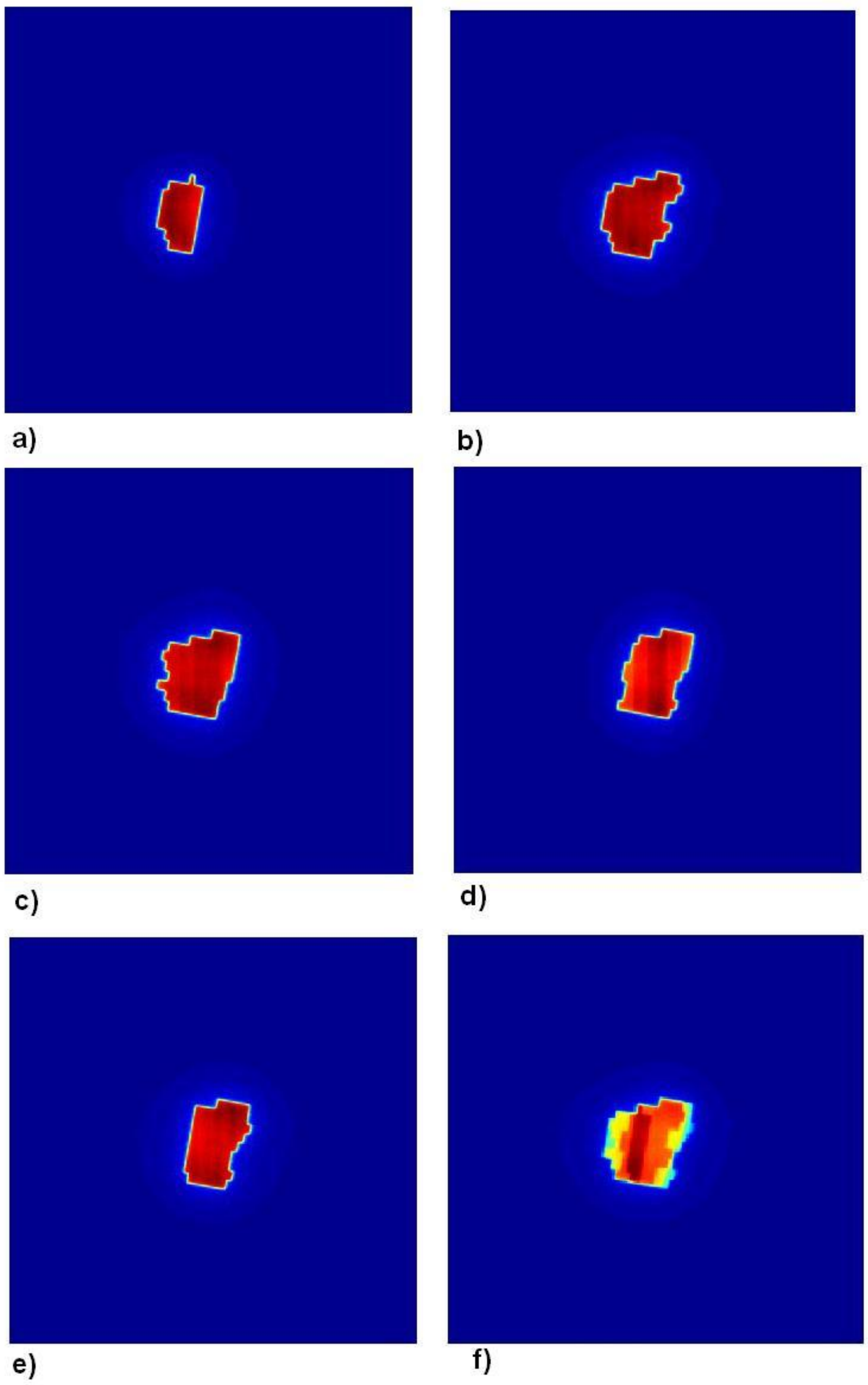


Figure 21 IMRT sequence
a) IMRT image 1
b) IMRT image 2
c) IMRT image 3
d) IMRT image 4
e) IMRT image 5
f) Accumulated image of image a, b, c, d, and e

5 Acquiring Correction Images

After being able to acquire images in Matlab the acquired frames or not yet ready for clinical use. The flaws in these pictures need to be corrected before for example a dose reconstruction or field analysis can be done. This chapter further explains the corrections mentioned in the problem description chapter.

5.1 Dead Pixel Map

When a cell of the EPID is not functioning any more, the panel's internal logic detects this "dead" cell. The cell value can scale from 0 to 0xFFFF and when a cell is dead the panel returns the value of the dead pixels as 0xFFFF. When scaling a returned black image[8] these cells look like white dots.

Before being able to correct for these dead pixels the software needs to know where they are located. To map these pixels a black image is acquired. The software then checks the value of every pixel. If the value is 0xFFFF it sets the corresponding pixel of the dead pixel map to 1 while all other pixels are set to 0. This creates a map of zeros and ones that indicate where the pixels are located. Figure 22 shows an example of what a dead pixel map would look like. The dead pixels are marked with a red circle.

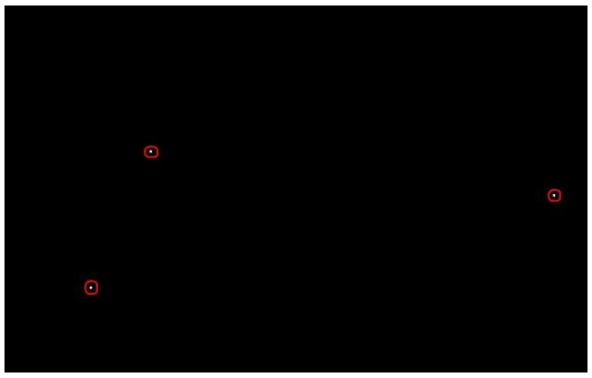


Figure 22 Dead Pixel Map (zoomed)

[8] Black image: An image created without any photon beams directed at the EPID

Now that the location of the dead pixels is know they can be corrected. The easiest way to do this is to check the pixels around that pixel. This results in a set of 9 values. The software checks if any of those pixels are dead. All good pixel values are added and divided by the number of good pixels around the dead pixel. This results a new value for this pixel that is derived out of its surrounding.

The software

The MEX function for creating a dead pixel map is almost exactly the same as the single frame function. The only difference is that after acquiring the frame the software does not return the acquired frame. The acquired frame is used to create the dead pixel map like described above.

5.2 Offset Correction

The offset is the initial value of the pixel. The cell values of a black image are not 0. The cell values should be 0 if no photon beam is directed at the EPID. Unfortunately this is not the case. The cells have a start value higher than 0 because of the biasing current of the photodiode. To make matters worse all cells have a slightly different offset value, resulting in an inhomogeneous image.

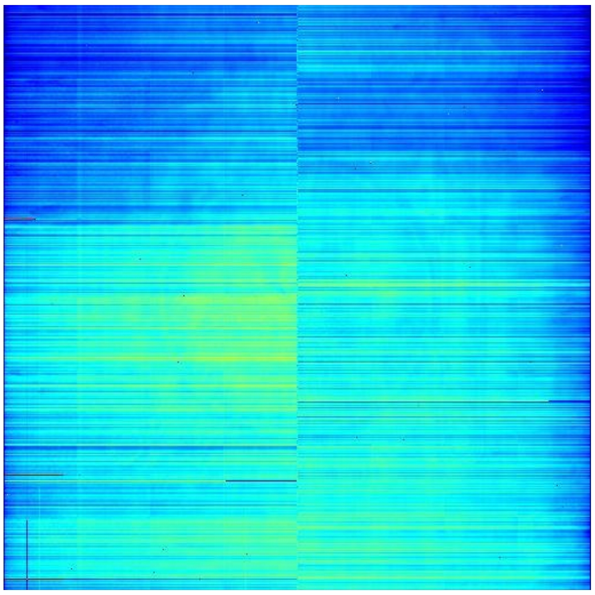


Figure 23 Offset Image

Figure 23 shows an example of an acquired offset image. The contrast is adjusted so that difference in bias values can be seen clearly.

To create a clear homogeneous image all these offset values need to be known so that when scaling the frame data to an image every pixel has the same initial value. This is what the offset correction does; it abstracts the offset values of all pixels so that a black image (corrected for dead pixels) looks like a homogeneous black picture.

The software

The MEX function for creating an offset image is almost the same as the MEX function to collect a single frame. However a different function is used. It is not necessary to define the destination buffers like with single mode acquisition. The pointer to the acquisition buffer is an input for the XIS defined offset acquisition function. The user can determine the number of frames that are averaged to create the offset image.

5.3 Gain Correction

The last correction is the gain correction. This correction is needed because besides having a different starting level, all cells react slightly different when being radiated with the same amount of photon beams. To create a homogeneous image it is necessary that all pixels that receive the same amount of photons have the same scaling value. Figure 24 gives an example of what a gain image looks like. Figure 25 is a plot that illustrates the offset and gain value of two cells with different gain and offset.

The software

The gain acquisition MEX function has very important difference. It returns a numeric array where each value is 4 bytes instead of 2. It also needs an extra input. Besides the necessary input structure it also requires an offset image. This needed to correctly acquire a gain image as prescribed in the XIS reference book.

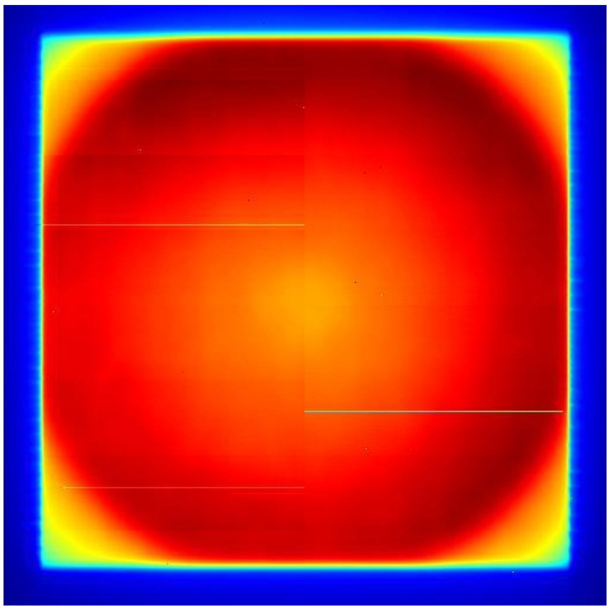
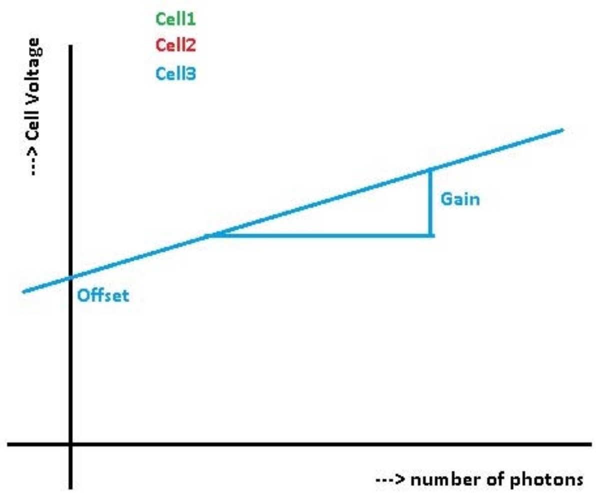


Figure 24 Gain image



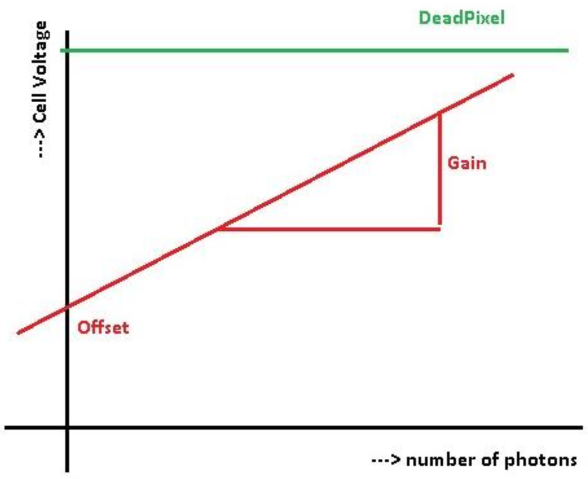


Figure 25 Correction plots

5.4 Results

With the correction MEX functions it was possible to create all correction images. This paragraph shows the effect of these corrections. The profile in figure 26 and 27 is the projection row 476 of the EPID image. This row shows all benefits of the corrections.

The dead pixel correction removes the peak between column 400 and 500. The corrected image starts at 0 due to the offset correction. The shape of the phantom is clearer due to the gain correction

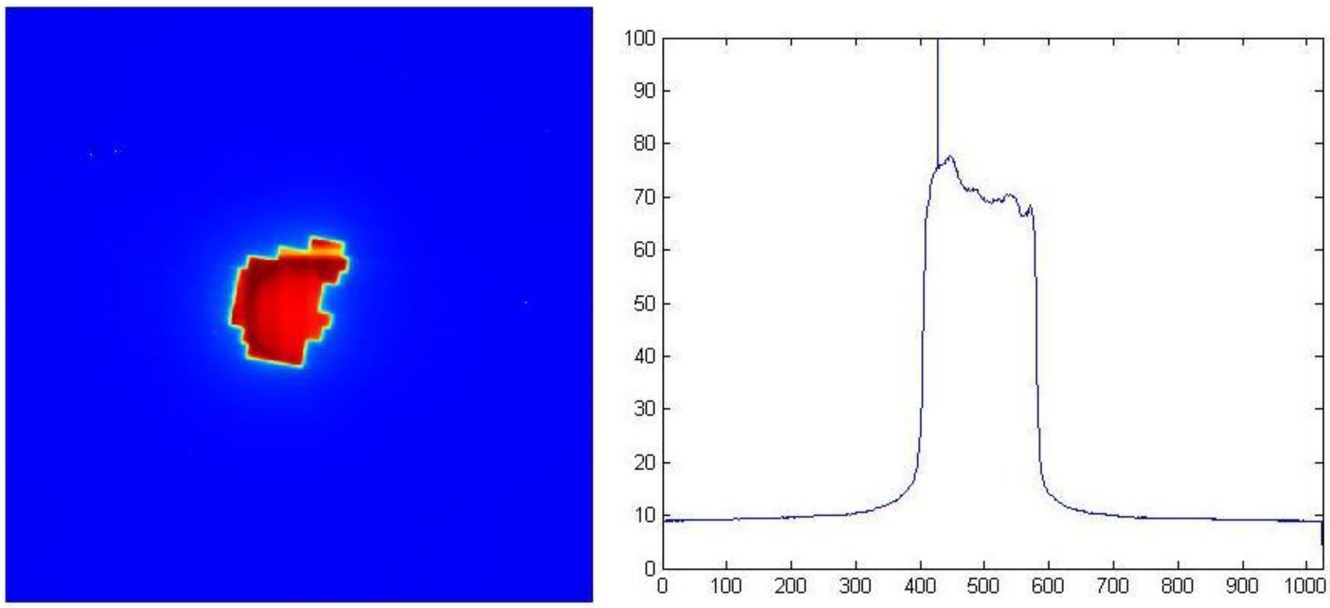


Figure 26 Uncorrected EPID image and profile

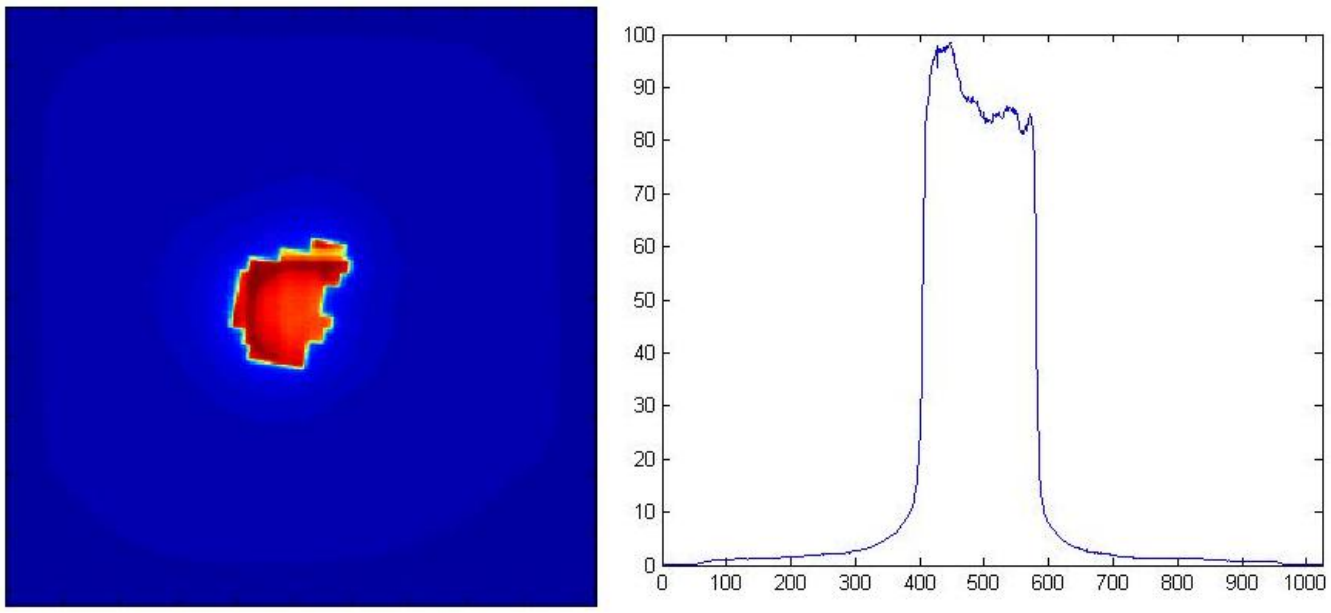


Figure 27 Corrected EPID Image and profile

6 Conclusion & Recommendations

Hardware Interrupts
Because of the lack of time this part of the project is not completed. After some research it was decided that this part should be evaluated in a different project. But the findings suggest that the linac trigger can be used to acquire frames using the external interrupt mode. But the linac trigger needs to be converted into a suitable trigger for the frame grabber. The in to output delay of the circuit is 0,5 microseconds (simulated). It needs to be evaluated if this can lead to information loss.

Because my component knowledge is fairly limited there might be better components available for the hardware interrupt interface. Before creating this interface the design should be reviewed by a senior Electronics Engineer. It is also wise to further investigate the possibilities of the trigger in- and outputs of the frame grabber. To collect frames correctly using these interrupts it might be necessary to include a microcontroller or FPGA in the hardware interface design.

Acquisition
Creating the Matlab Acquisition library is completed successfully. Using the MEX functions created for the MV fluoroscopy library it is possible to acquire EPID images in Matlab. It is possible to acquire a single frame resembling the images used for patient position verification. The continuous acquisition mode works but not the way MAASTRO clinic would have wanted. After entering a MEX-function Matlab stops responding. Like described in the paragraph about continuous acquisition frames are added to a structure that is returned to Matlab or written to .MAT file. It was expected that the frames would be available after being added to either the structure or .MAT file. This is not the case. All data is not available until the MEX function ends. This is a problem that needs to be fixed if the continuous acquisition is to be used for real time processing.

A mayor improvement of the software would be making the acquisition handles persistent in Matlab even after the MEX functions are executed. This would mean the initialisation and the closing procedure for acquisition can become separate MEX functions. This saves time when using the frame acquisition functions. The current methods were chosen to ensure the project would be successful.

The relation between trigger time and integration time is still just an assumption. This has not been thoroughly measured. When starting to use the acquisition functions for dose reconstructions this is a weakness in the program that could produce an error in the amount of dose calculated.

The other acquisition mode that has not been created yet is the external trigger mode. This makes the frame grabber request a frame when a certain combination of signals is presented to the three trigger inputs. The timing suggested by the XIS reference book needs further elaboration. Before continuing the design of the hardware interrupt circuits it would be best to investigate the possibilities of the external trigger mode and its advantages.

Corrections
The creation of correction images is possible in Matlab as well. The dead pixel, offset and gain images are available for further processing.

An improvement to this software would a sequenced offset of gain acquisition. Every integration time and camera sensitivity requires their own set of correction images. A function that would create a set of correction images instead of just a single correction image could be useful to improve the user friendliness of the correction functions.

Personal

My last recommendation is not related to the project for MAASTRO clinic but meant for students that still need to pick their graduation assignment. Do not be affected by appendages when picking your graduation assignment. Try to make sure that assignment is the challenge you are looking for and that the company you have chosen fits your personality. Try to not let matters like payment or distances affect your choice. In spite of missing out on internship payment I had a great internship. My specialization in Electrical Engineering in Electronics and Telecommunication and because of this my software engineering skills were limited. The challenge of overcoming this lack of knowledge and successfully finishing my graduation motivated me to show what I am capable of.

7 Sources

1. "*Portal dosimetry in Radiotherapy*" by Sebastiaan M.J.J.G. Nijsten
2. The "*KWF kanker bestrijding*" website with the general information needed about cancer. (www.kwfkankerbestrijding.nl)
3. The *EhealthMD* website with the information about radiation therapy. (http://www.ehealthmd.com/library/radiationtherapy/RT_how.html)
4. The Matworks website with the information about the use of MEX functions and Matlab code in C++ (http://www.mathworks.com/access/helpdesk/help/techdoc/apiref/bqoqnz0.html)
5. The XRD-FG manual by Perkin Elmer Optoelectronics: "*XIS Software reference book*"

8 List of figures

9 Enclosures

1) Extended background information
2) Hardware schematics
3) Frame header information
4) Acquisition Flowchart
5) Corrections Flowcharts

1) Extended background information

1.1 CANCER AND RADIOTHERAPY

Cancer is one of the major public health problems in Europe, the United States and other countries in the western world. In 2006, there were over two million incident cases of cancer in the European Union and over one million cancer deaths[1]. In the United States, one out of four deaths is due to cancer as estimated for 2007[2]. Prostate cancer (20%-29%) and lung cancer (15%-17%) are the most common forms of cancer in men while for women, breast cancer (26%-29%) is diagnosed most frequently[1,2]. Lung cancer causes most cancer deaths in men with an estimate of 26.3% of all cancer deaths in Europe in 2006. In that year, breast cancer was the most common cause of cancer death in women (16.7%)[1].

Radiotherapy is one of the main treatment methods for cancer next to surgery, chemotherapy and hormone therapy. Radiotherapy (alone or in combination with other treatments) is a curative treatment for 40% of the cancer patients receiving it[3]. The goal of radiotherapy is to irradiate tumor tissue to a high dose while sparing the surrounding normal healthy tissue as much as possible to limit the complications of a patient treatment. In general, irradiation is done by using *external beam radiotherapy* or *brachytherapy*.

In case of *external beam radiotherapy*, a radiation beam is pointed at a particular part of the body. By using multiple beams in an optimum beam angle configuration, the dose in healthy tissue can be diminished. Different types of radiation have different interactions with tissue and will cause more or less biological damage. Most commonly used radiation types are megavoltage X-rays (Megavolt or MV photon beams), electrons and protons.

In case of *brachytherapy*, a radioactive source is placed inside or next to the area requiring treatment. Examples of brachytherapy are interstitial brachytherapy with iodine-125 seeds implanted in the prostate to treat localized prostate cancer disease and intracavitary brachytherapy with iridium-192 placed in the cervix to irradiate cervical tumor tissue.

1.2 EXTERNAL MEGAVOLT PHOTON BEAM TREATMENT

Worldwide, the most frequently applied radiotherapy technique is an external MV photon beam treatment with linear accelerators (LINACs)[4-6]. With these devices, electrons are generated and accelerated to high energies of 4-25 MeV. The accelerated electrons produce X-rays when they collide with a tungsten target and the resulting photon beam can be used for treatment after additional filtering, collimation and shielding of the beam in the treatment head. Beam shielding is a prerequisite in high dose-high precision radiotherapy in order to obtain dose distributions that conform to the tumor volume while sparing neighbouring healthy tissue. Both individual moulded blocks and a multileaf collimator (MLC) can be used for beam shielding. The latter device is located inside the treatment head of a LINAC and consists typically of a series

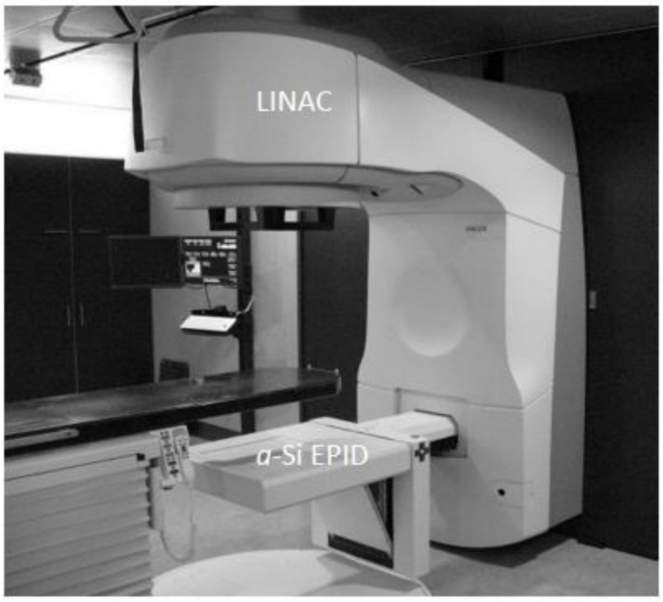


Figure 1: A Siemens Oncor medical linear accelerator (Siemens Medical Solutions, Concord, CA) equipped with an amorphous silicon (a-Si) electronic portal imaging device.

of 80 to 160 metallic leaves which can be positioned individually to shape the beam aperture. Furthermore, modern LINACs are usually equipped with an electronic portal imaging device (EPID) which allows imaging of the high energetic MV photon beam that exits the patient during treatment[7,8]. These images can be used for patient set-up verification or detection of organ motion but also for dosimetric verification of a treatment which is called *portal dosimetry*. In Fig. 1, a Siemens Oncor medical linear accelerator (Siemens Medical Solutions, Concord, CA) is shown equipped with an amorphous silicon (a-Si) electronic portal imaging device.

Before the treatment can be applied to a patient, an individual treatment plan is generated using a treatment planning system (TPS)[9]. This system uses three-dimensional (3D) imaging information of a patient to model the position and shape of both tumor and healthy tissues. Based on this localization, an optimum beam configuration including photon beam energies, field sizes, shielding, beam directions and relative beam weighting can be determined. The end result is a definition of beam parameters that is needed to set-up the linear accelerator and the 3D dose distribution inside the patient that documents the prescribed radiotherapy treatment.

1.3 TREATMENT VERIFICATION USING DOSE MEASUREMENTS

By administering radiotherapy in a fractionated manner (i.e. giving every day a small dose of radiation), a small difference in biological radiation sensitivity between tumor and healthy tissue can be exploited to increase the overall tumor dose. Hence, tumor control probability can be increased with acceptable normal tissue complication probability. Consequently, every fraction should be given in a reproducible way and the dose delivery in a patient should be as close as possible to the prescribed dose as calculated with the TPS. For instance, Mijnheer *et al.*[10] proposed an accuracy requirement in absorbed dose delivery at the dose specification point of 3.5% (1 SD) for external beam treatments based on the steepness of dose-effect curves for local tumor control and normal tissue damage. In order to achieve such a high accuracy, it is of utmost importance that the dose delivery is verified during external MV photon beam treatment.

In case of *image-guided radiation therapy (IGRT)*, imaging devices are used to determine the position of tumor and healthy tissues just prior to or during treatment and to calculate positional corrections based on measured images. By applying for instance a correction to the patient position, differences in patient set-up, and hence inaccuracies in delivered dose, should consequently decrease. Examples of IGRT devices in the treatment room are EPIDs, megavoltage or kilovoltage cone-beam computed tomography (CT) scanners and CT on rails[7,8,11-14].

In case of *dose-guided radiation therapy (DGRT)*, a treatment is adapted based on *direct dose measurements* in combination with the results of IGRT procedures. Dose measurements can be done prior to treatment (pre-treatment measurements) and during treatment (*in vivo* measurements) using several dose verification devices. Diodes, thermoluminescent dosimeters (TLDs) and metal oxide semiconductor field-effect transistors (MOSFETs)[15-18] are point detectors, only allowing a single point measurement. However, these detectors have a long history of clinical use, their dosimetric characteristics are well known and they are adequate for verification of conventional treatment techniques[19-21]. Radiographic and radiochromic film[22] can be used to verify dose distributions in two dimensions and have a high spatial resolution. Therefore, film has especially been used for the verification of intensity-modulated radiation therapy (IMRT) treatments where high dose gradients are present in the plane of the beam[23]. The disadvantage of film is that the dose evaluation is time consuming and is prone to errors during processing, digitizing and analyzing. Gel dosimetry[24,25] is one of the few dosimetry systems currently available allowing a measurement of the 3D dose distribution. However, the method is limited by the complex preparation process and the expensive analysis using a magnetic resonance imaging (MRI) scanner.

Finally, different types of EPIDs[7,8] have been developed in the past and their use for dosimetry has been investigated[26]. Compared with other dosimetry devices, EPIDs are already attached to a linear accelerator and no additional hardware is needed to perform portal dosimetry. EPID measurements can be performed with minimum set-up requirements and a 2D dose conversion can be done immediately using the digital images. Furthermore, the same EPIDs are already used for IGRT applications, e.g. for patient set-up verification using conventional portal imaging or megavoltage cone-beam CT. Consequently, one measurement device can be used to obtain different types of information during treatment. Although an EPID image contains 2D and not 3D information, it is still possible to reconstruct the 3D dose distribution inside a patient by using a back-projection procedure of the measured portal dose image (PDI) into three dimensions[27-29].

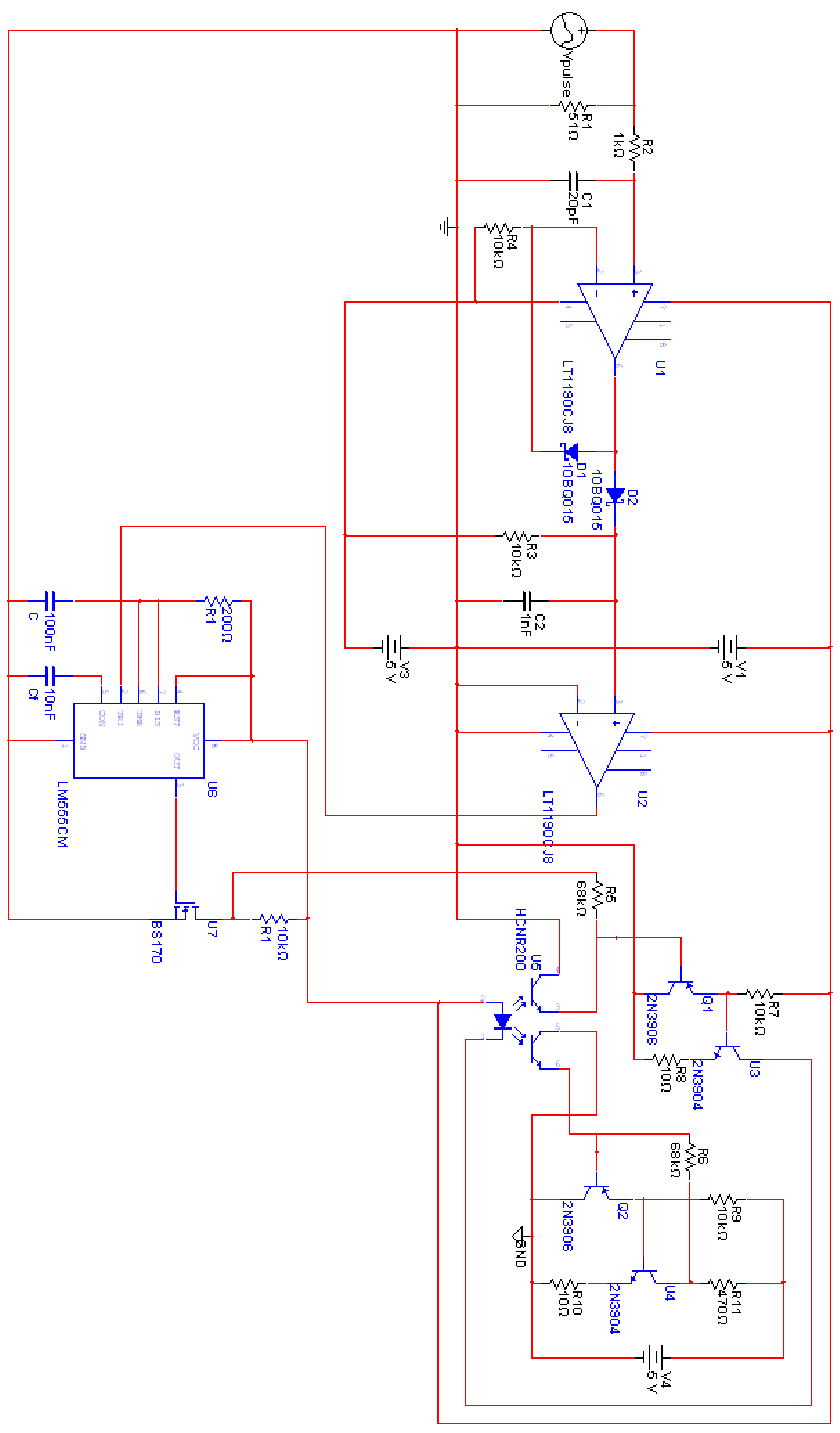
1.4 AIM OF THIS WORK

The use of electronic portal imaging devices for treatment verification using dose measurements looks very promising. Until now, EPIDs have not been used clinically on a large scale for dosimetric verification and hence, clinical experience is limited. Because point detectors have a long history of clinical use, dose differences resulting from these measurements are widely reported and understood. Moving directly from these measurements to 3D dose reconstruction using EPIDs is a large step, requiring not only complex mathematical models but also a new way of comparing and visualizing dose verification results. Although a comparison between a prescribed and delivered 3D dose distribution is the ultimate goal of portal dosimetry, a more gradual clinical transition may be worthwhile[30]. Starting with verification procedures based on portal dose measurements in one point offers the possibility to directly compare the EPID results to the results of diode, TLD or MOSFET measurements[19-21]. Also, pitfalls of the new procedures can be more easily recognized and eliminated in this case. For departments already performing *in vivo* dosimetry measurements, it is a small step to move to an EPID based dose verification procedure that takes less time on a linear accelerator than the traditional point dose measurements.

The aim of this work was to develop dose verification methods using portal dosimetry based on point dose measurements and dose distributions in two dimensions. Both dosimetric calibration models for different types of EPIDs and prediction models to obtain reference dose values had to be implemented. All models should be based on absolute dose in water measured under full-scatter conditions, simplifying a future back-projection procedure of the measured portal dose into three dimensions. The accuracy and clinical applicability of the dose verification methods was investigated and results are presented from large-scale clinical use. The final goal was to show that routine portal dosimetry is feasible and that detection of a wider variety of dose delivery errors than the traditional point dosimeters can be accomplished.

The text above is copied from the introduction chapter of the Thesis "*Portal dosimetry in Radiotherapy*" by **Sebastiaan M.J.J.G. Nijsten**

2) Hardware schematics



3) Frame Header example

7.2.38 CHwHeaderInfo

Structure that is used to retrieve the contents of detector's hardware header by Acquisition_GetHwHeader.

```
typedef struct
{
        DWORD       dwPROMID;
        DWORD       dwHeaderID;
        BOOL        bAddRow;
        BOOL        bPwrSave;
        DWORD       dwNrRows;
        DWORD       dwNrColumns;
        DWORD       dwZoomULRow;
        DWORD       dwZoomULColumn;
        DWORD       dwZoomBRRow;
        DWORD       dwZoomBRColumn;
        DWORD       dwFrmNrRows;
        DWORD       dwFrmRowType;
        DWORD       dwFrmFillRowIntervalls;
        DWORD       dwNrOfFillingRows;
        DWORD       dwDataType;
        DWORD       dwDataSorting;
        DWORD       dwTiming;
        DWORD       dwAcqMode;
        DWORD       dwGain;
        DWORD       dwOffset;
        BOOL        bSyncMode;
        DWORD       dwBias;
        DWORD       dwLeakRows;
} CHwHeaderInfo;
```

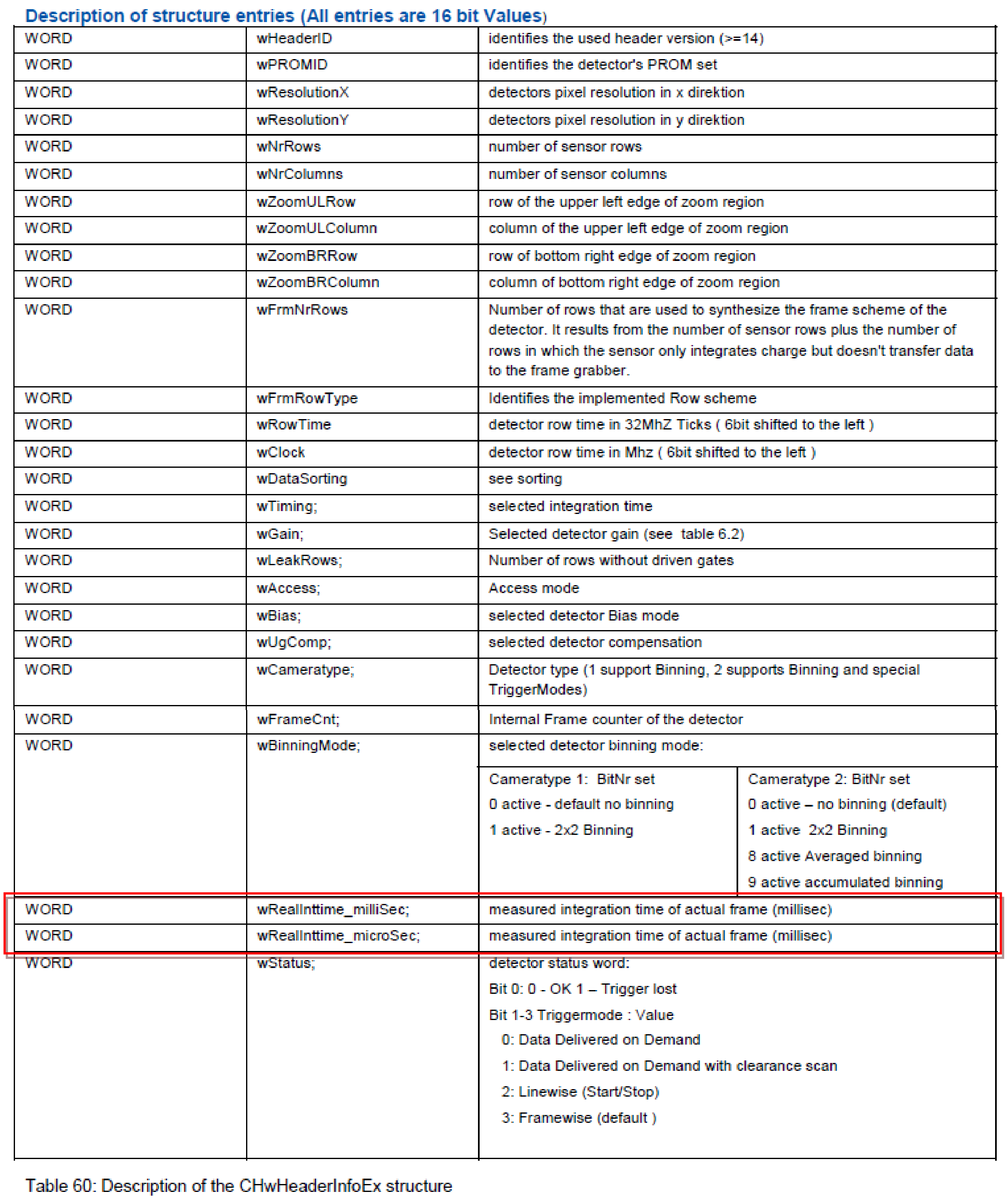
DWORD	dwPROMID;	identifies the detector's PROM set
DWORD	dwHeaderID;	identification number of this header type. Header types: 10 all Detectors except AM/AN 11 XRD 16x0 AM 12 | 13 XRD 16x0 AN, 14 XRD 1621 AN
BOOL	bAddRow;	indicates if an additional row is transferred
BOOL	bPwrSave;	indicates if detector is in power safe mode
DWORD	dwNrRows;	number of sensor rows
DWORD	dwNrColumns;	number of sensor columns
DWORD	dwZoomULRow;	row of the upper left edge of zoom region
DWORD	dwZoomULColumn;	column of the upper left edge of zoom region
DWORD	dwZoomBRRow;	row of bottom right edge of zoom region
DWORD	dwZoomBRColumn;	column of bottom right edge of zoom region
DWORD	dwFrmNrRows;	Number of rows that are used to synthesize the frame scheme of the detector. It results from the number of sensor rows plus the number of rows in which the sensor only integrates charge but doesn't transfer data to the frame grabber plus the number of filling rows.
DWORD	dwFrmRowType;	see Row Types
DWORD	dwFrmFillRowIntervalls;	Intervals of 10 nanoseconds to synthesize a frame (see description of hardware header (Header ID <11 otherwise used for int time detection in some header versions)
DWORD	dwNrOfFillingRows	Number of rows of the above mentioned row time.
DWORD	dwDataType;	used for int time detection in some header versions
DWORD	dwDataSorting;	see sorting
DWORD	dwTiming;	selected integration time
DWORD	dwAcqMode;	fixed mode (0), sync mode (1) with fixed frame regime (Header ID <11)
DWORD	dwGain;	1 for Header ID < 11 otherwise Gain mode
DWORD	dwOffset;	
BOOL	bSyncMode	1 if the detector operates in triggered mode else 0.
DWORD	dwBias;	10 V * SensorBias / 255 (Header ID <11)
DWORD	DwLeakRows	Number of rows without driven gates

Table 57: Description of the ChwHeaderInfo structure

7.2.49 CHwHeaderInfoEx

Structure that is used to retrieve the extended detector hardware header by Acquisition_GetHwHeaderINfoEx(..) or GetLatestFrameHeader(..). This function is only available for detectors with HeaderID 14 or above. (1621, 0820)

```
typedef struct{
          WORD  wHeaderID;
          WORD  wPROMID;
          WORD  wResolutionX;
          WORD  wResolutionY;
          WORD  wNrRows;
          WORD  wNrColumns;
          WORD  wZoomULRow;
          WORD  wZoomULColumn;
          WORD  wZoomBRRow;
          WORD  wZoomBRColumn;
          WORD  wFrmNrRows;
          WORD  wFrmRowType;
          WORD  wRowTime;
          WORD  wClock;
          WORD  wDataSorting;
          WORD  wTiming;
          WORD  wGain;
          WORD  wLeakRows;
          WORD  wAccess;
          WORD  wBias;
          WORD  wUgComp;
          WORD  wCameratype;
          WORD  wFrameCnt;
          WORD  wBinningMode;
          WORD  wRealInttime_milliSec;
          WORD  wRealInttime_microSec;
          WORD  wStatus;
}         CHwHeaderInfoEx;
```



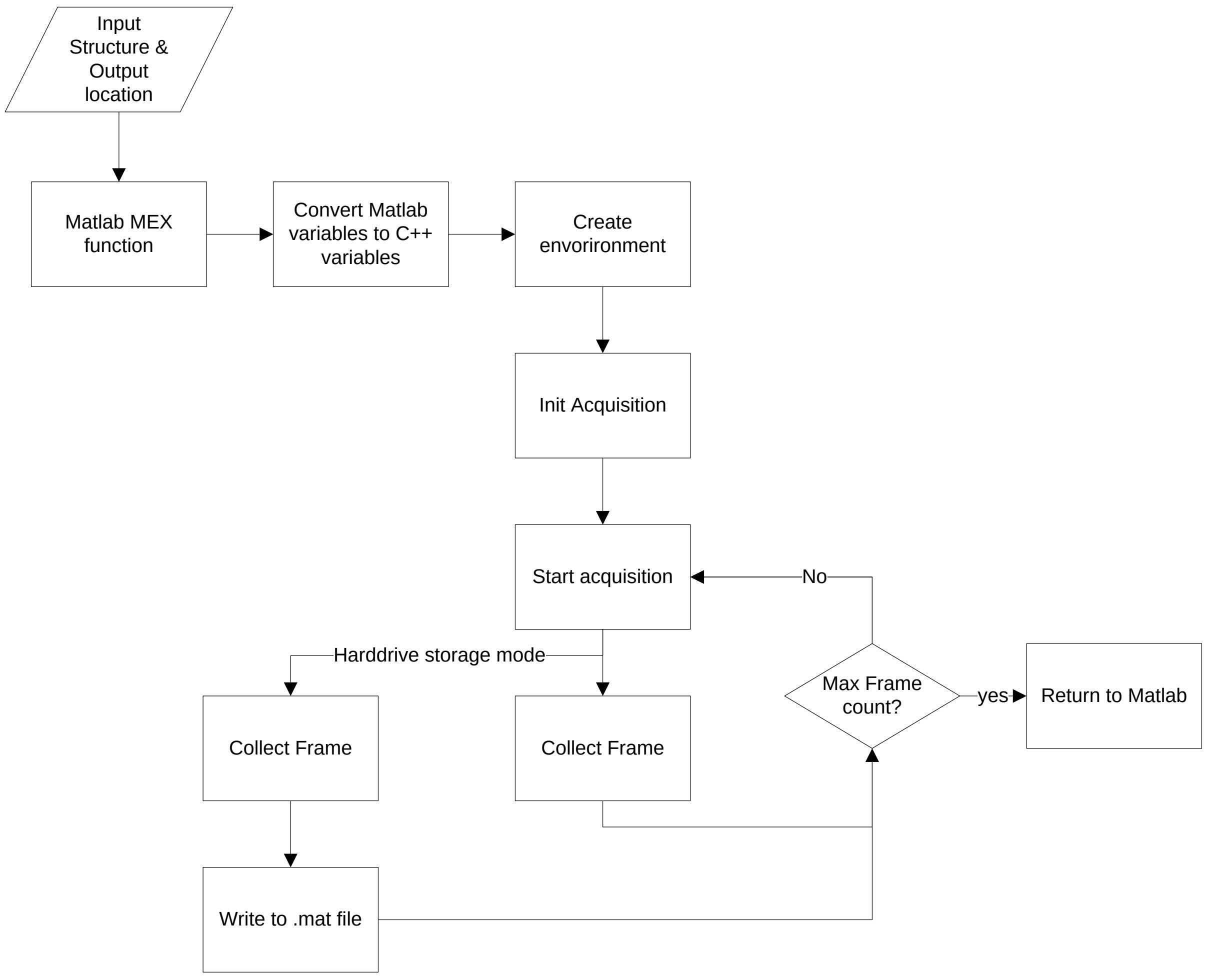
Description of structure entries (All entries are 16 bit Values)

WORD	wHeaderID	identifies the used header version (>=14)
WORD	wPROMID	identifies the detector's PROM set
WORD	wResolutionX	detectors pixel resolution in x direktion
WORD	wResolutionY	detectors pixel resolution in y direktion
WORD	wNrRows	number of sensor rows
WORD	wNrColumns	number of sensor columns
WORD	wZoomULRow	row of the upper left edge of zoom region
WORD	wZoomULColumn	column of the upper left edge of zoom region
WORD	wZoomBRRow	row of bottom right edge of zoom region
WORD	wZoomBRColumn	column of bottom right edge of zoom region
WORD	wFrmNrRows	Number of rows that are used to synthesize the frame scheme of the detector. It results from the number of sensor rows plus the number of rows in which the sensor only integrates charge but doesn't transfer data to the frame grabber.
WORD	wFrmRowType	Identifies the implemented Row scheme
WORD	wRowTime	detector row time in 32MhZ Ticks (6bit shifted to the left)
WORD	wClock	detector row time in Mhz (6bit shifted to the left)
WORD	wDataSorting	see sorting
WORD	wTiming;	selected integration time
WORD	wGain;	Selected detector gain (see table 6.2)
WORD	wLeakRows;	Number of rows without driven gates
WORD	wAccess;	Access mode
WORD	wBias;	selected detector Bias mode
WORD	wUgComp;	selected detector compensation
WORD	wCameratype;	Detector type (1 support Binning, 2 supports Binning and special TriggerModes)
WORD	wFrameCnt;	Internal Frame counter of the detector
WORD	wBinningMode;	selected detector binning mode: Cameratype 1: BitNr set 0 active - default no binning 1 active - 2x2 Binning Cameratype 2: BitNr set 0 active – no binning (default) 1 active 2x2 Binning 8 active Averaged binning 9 active accumulated binning
WORD	wRealInttime_milliSec;	measured integration time of actual frame (millisec)
WORD	wRealInttime_microSec;	measured integration time of actual frame (millisec)
WORD	wStatus;	detector status word: Bit 0: 0 - OK 1 – Trigger lost Bit 1-3 Triggermode : Value 0: Data Delivered on Demand 1: Data Delivered on Demand with clearance scan 2: Linewise (Start/Stop) 3: Framewise (default)

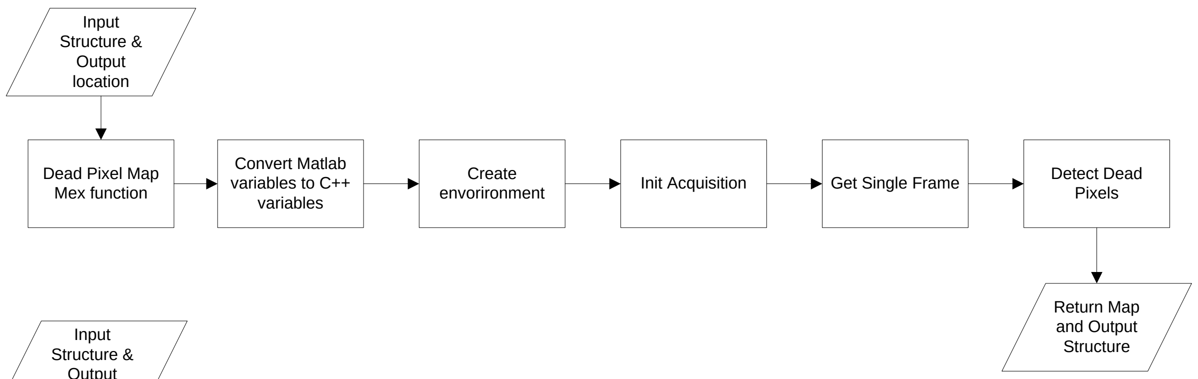
Table 60: Description of the CHwHeaderInfoEx structure

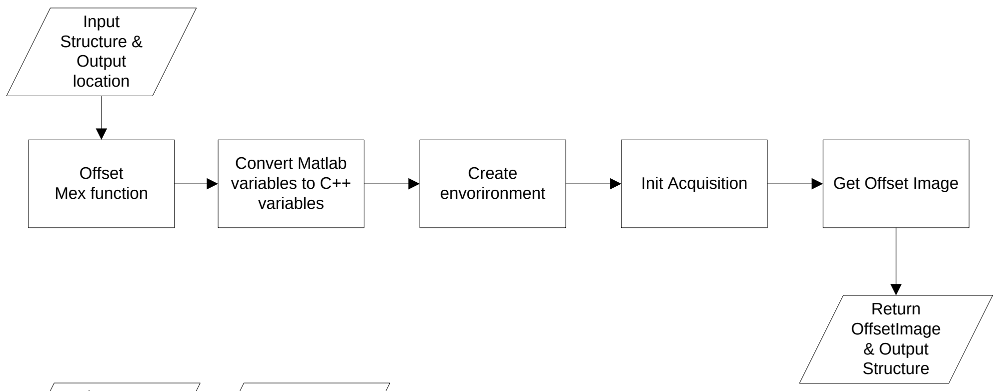
Source: The Perkin Elmer Optoelectronics "*XIS Software reference book*"

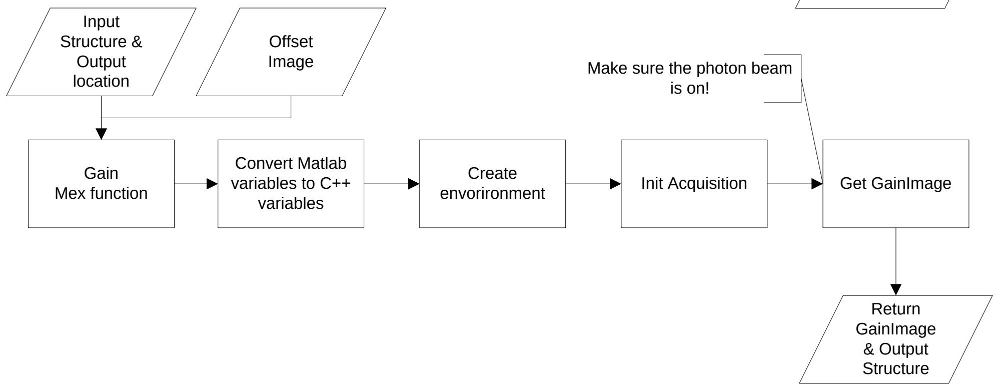
4) Acquisition flowchart



5) Corrections flowcharts


Input Structure & Output location
Dead Pixel Map Mex function
Convert Matlab variables to C++ variables
Create envorironment
Init Acquisition
Get Single Frame
Detect Dead Pixels
Return Map and Output Structure


Input Structure & Output location
Offset Mex function
Convert Matlab variables to C++ variables
Create envorironment
Init Acquisition
Get Offset Image
Return OffsetImage & Output Structure


Input Structure & Output location
Offset Image
Make sure the photon beam is on!
Gain Mex function
Convert Matlab variables to C++ variables
Create envorironment
Init Acquisition
Get GainImage
Return GainImage & Output Structure