



## **LabVIEW Data Acquisition for Optical Parametric Synthesis**

Yousuf Hemani, University of Eastern Finland, Finland

July 19 – September 8, 2016

### **Abstract**

To generate attosecond pulses, high harmonic generation is an important process. For HHG to work, it is important to have synthesized waveforms generated by optical parametric wave synthesizer. An important component for this process is Optical parametric chirped pulse amplifiers. When working with OPCPA's in the lab, it is always critical for long running experiments like these to monitor the constancy of the characteristics of the laser beam exiting the regenerative amplifier such as the light spectrum and the beam profile. It is convenient to measure the spectrum with the help of a portable spectrometer and the beam profile with the help of an inexpensive camera. To acquire data continuously from these devices, software like LabVIEW is needed which can acquire data over long periods continuously from the devices. This summer project report contains all the necessary details on how to configure these devices and how to work with these devices in LabVIEW. Two different programs have been created, one for spectrometer acquisition and one for camera acquisition; to monitor the spectrum, the beam waist, and beam spot position. The advantages of such programs are that they can be replicated with slight changes for multiple spectrometers or multiple cameras to check for any inconsistencies or drifts in the laser beam at different places in a big experiment.

# Contents

LabVIEW Data Acquisition for Optical Parametric Synthesis .....	1
Contents .....	2
1. Introduction.....	3
2. Spectrometer Program .....	4
2.1. Working with the Spectrometer.....	4
2.2. Installation of Software and Drivers:.....	4
2.3. Working of the Spectrometer LabVIEW Program: .....	6
2.4. MATLAB code for processing: .....	8
3. The Camera Program.....	9
3.1. Working with the Camera .....	9
3.2. Installation and Configuration of Camera: .....	9
3.3. Working of the Camera LabVIEW Program: .....	12
3.4. Testing of Program and Experimental Setup:.....	15
References.....	18
Acknowledgements .....	19
Appendix A .....	20

# 1. Introduction

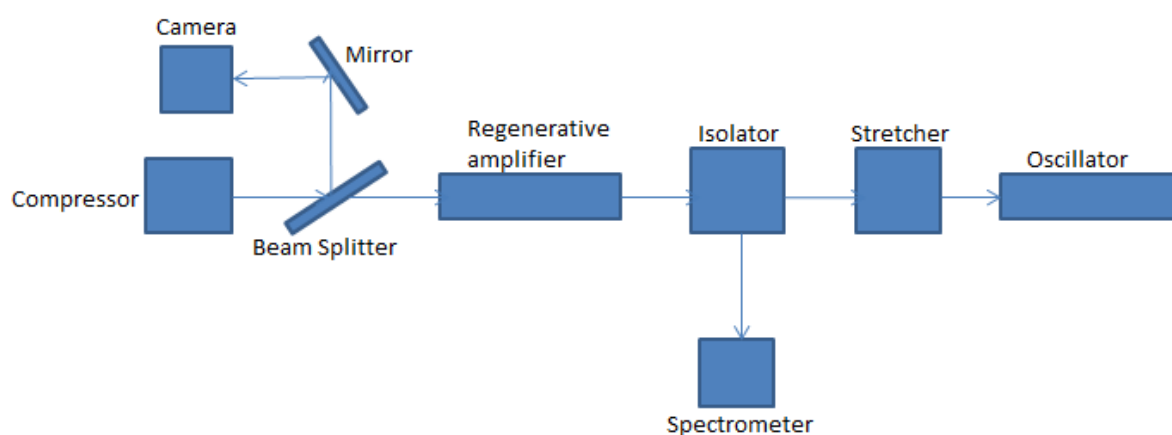
There are a lot modern applications and research concerning laser science which demands the generation of shorter light pulses. This requirement lead to the development of a light amplification technique called optical parametric chirped pulse amplifier. This phenomenon gives a broad gain bandwidth and higher pulse energy during the same time while keeping the thermal issues at bay (1). The OPCPA can directly generate few cycle high energy pulses. In OPCPA, for e.g. a femtosecond pulse having only a few cycles is stretched to picosecond or even to nanosecond durations for amplification and this is done by using narrow band high energy pump lasers and then the pulse is recompressed to its original duration. This high intensity laser pulse is suitable for strong field physics experiments (2). An important component of the OPCPA is the regenerative amplifier which is used to drive the OPCPA and it is important that the output of this amplifier is stable in power, pointing and beam parameters. A few advantages of OPCPA are that the parametric gain from a single pass through a nonlinear crystal is sometimes enough or it needs very few passes so the system can be compact and also the amplification is possible in a wide range of wavelengths (3).

The attosecond light sources have enabled the researchers to study atomic and intra-atomic processes at a sub femto second time scale. To generate attosecond pulses, synthesis of few long femtosecond pulses has to be done which results in sub cycle waveform. High harmonic generation is an established method to generate isolated attosecond pulses but synthesized waveforms are important for producing attosecond pulses with HHG method. The coherent pulse synthesis of high energy, few cycle pulses is the most efficient method to generate isolated attosecond pulses with flexibility in spectral shaping and scalability in spectrum and energy. Carrier envelope phase stability of the driving pulsed light source is crucial because HHG is driven by the electric field (4).

OPCPA's are a critical component to perform Optical Parametric Synthesis. Optical parametric wave synthesizer needs these steps (5):

- Generation of a broadband seed source which is CEP stabilized
- Amplification of different spectral regions separately in an OPCPA
- Combination of different pulses and compression

The project consists of deploying a spectrometer and multiple cameras to check for spectrum and beam characteristics at the output of the regenerative amplifier contained in the OPCPA and various other places inside the experiment. The data from the spectrometer and the camera is acquired in the computer for processing through LabVIEW acquisition programs. This report comprises of the details about the configuration and working of these two devices and their respective programs.

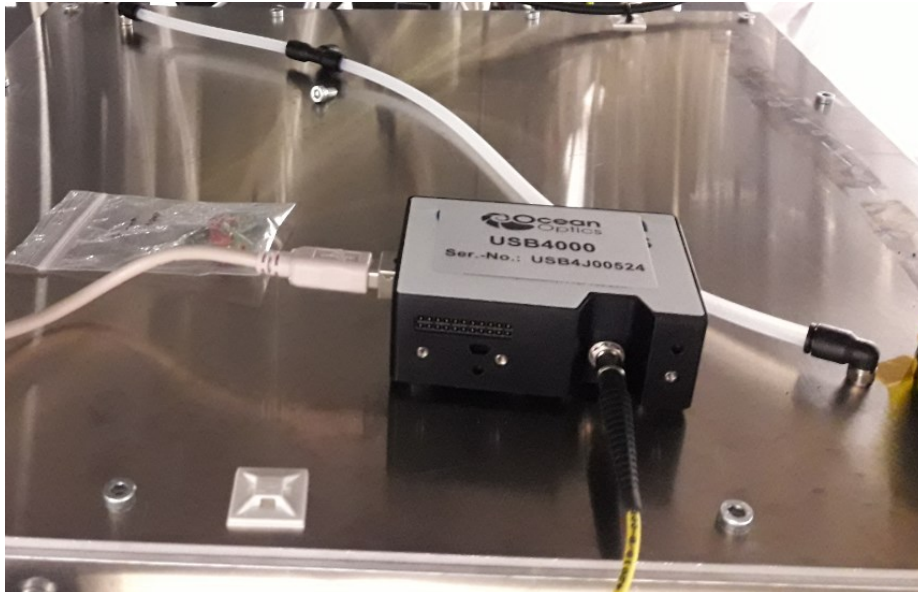


**Figure 1.1: An example diagram of OPCPA showing where the spectrometer and the camera will be placed**

## 2. Spectrometer Program

### 2.1. Working with the Spectrometer

To measure the spectra for the laser experiment, the Ocean Optics USB Spectrometer 4000 is used, and data acquisition is done in LabVIEW. The USB4000 is an economical and versatile spectrometer. The CCD array and high-speed electronics provide high spectral response and good optical resolution. It is a compact and flexible system with no moving parts. It covers the 200-1100nm range of wavelength and connects to light sources and optical fibers.



**Figure 2.1: Ocean Optics Spectrometer configured in the lab**

The data acquisition from can be done in LabVIEW by using the Omni driver and SPAM libraries which come with the Ocean Optics Software. Here, Omnidriver has the functions that are needed to develop applications which communicate with ocean optics spectrometers and control it as well. For e.g. it enables the user to connect to spectrometers, set acquisition parameters such as integration time and also acquire the spectra. SPAM(Spectral processing and Math) is a add on to the Omnidriver architecture and lets the user to do some processing on the acquired data for e.g., finding the peak point etc. More information on this can be found in the Omnidriver programming manual.

Omnidriver32.dll (or Omnidriver64.dll) contains all these functions which can be used in LabVIEW for data acquisition from spectrometer.

SPAM32.dll (or SPAM64.dll) contains higher level functions for analyzing the data acquired from the spectrometer.

### 2.2. Installation of Software and Drivers:

It is important to install all the software to run the spectrometer with LabVIEW.

The software needed to install are:

- OmnidriverSpam: Contains the Omnidriver and SPAM libraries

- Dotnetframework: Contains Microsoft net framework software
- Overture: The spectrometer software included with ocean optics
- Signaltonoisebroadband: contains predefined examples in several programming languages
- Usb programmerx64WinUSB: contains the usb driver for 64 bit version of windows

The first step is to connect the spectrometer to the computer and let it find the relevant driver. In most cases it will fail, where then the user can select the usb driver installed for 64 bit windows. After installing all of the above mentioned software, the user can check by running the Spectrumtest32.exe (or Spectrumtest64.exe) included in the installed files, which will show the no. of spectrometers connected to the computer. Also, running the overture software can also display if the spectrometer is acquiring data or not. More information on this can be found in the Omnidriver Startup file.

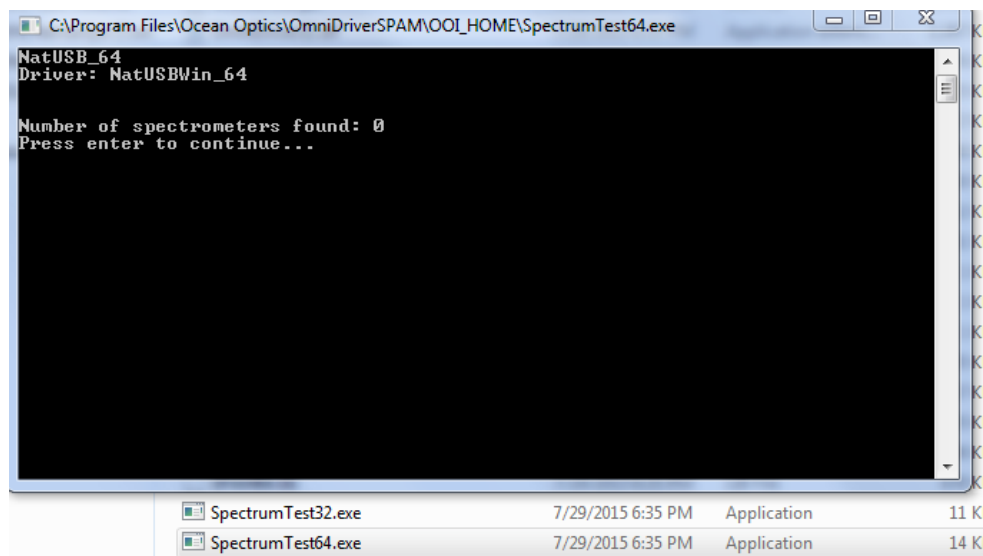


Figure 2.2: Check for spectrometers by using Spectrumtest64.exe

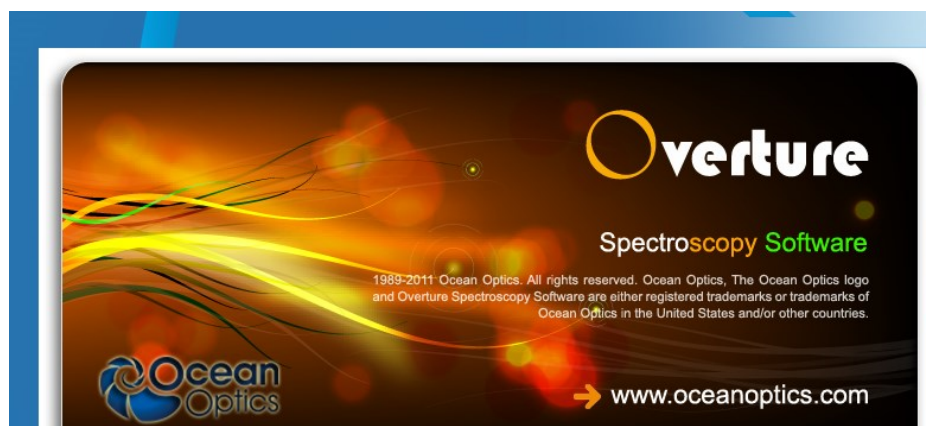


Figure 2.3: Overture software to check operation of the spectrometer

### 2.3. Working of the Spectrometer LabVIEW Program:

The spectrometer program is made in LabVIEW to acquire data from the ocean optics spectrometer. The program is implemented with a stacked sequence structure. It makes the use of wrapper.llb which contains selective VI functions to do basic programming on LabVIEW. It is important to create a wrapper first as all access to the spectrometer is granted by initializing a wrapper object. For example to create a wrapper and open all spectrometers; right click on the functions palette, go to Select a VI, choose wrapper.llb and there will be list of functions to choose from like open all spectrometers, close all spectrometers, create wrapper, destroy wrapper etc.

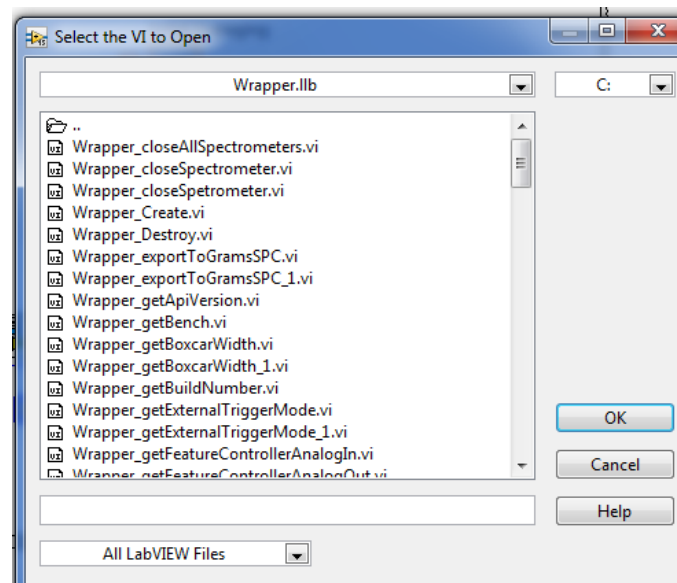


Figure 2.4: Wrapper Library for Spectrometer functions in LabVIEW

The program acquires no. of pixels, no. of wavelengths, removes the background noise and gives a live spectra reading, where the user can define the integration time. The program is designed to take a reference reading or load a reference reading and display the spectra. Then the user can define the no. of reading to be taken and no. of files to be created and the measurement interval.

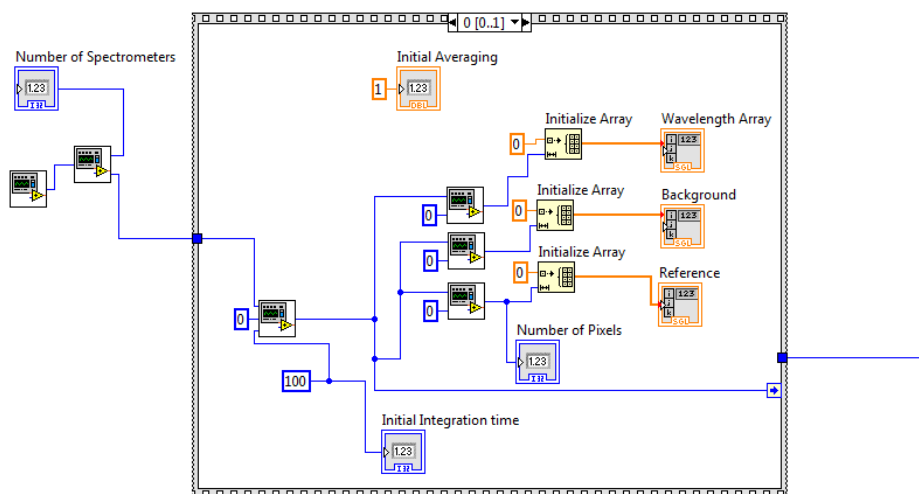
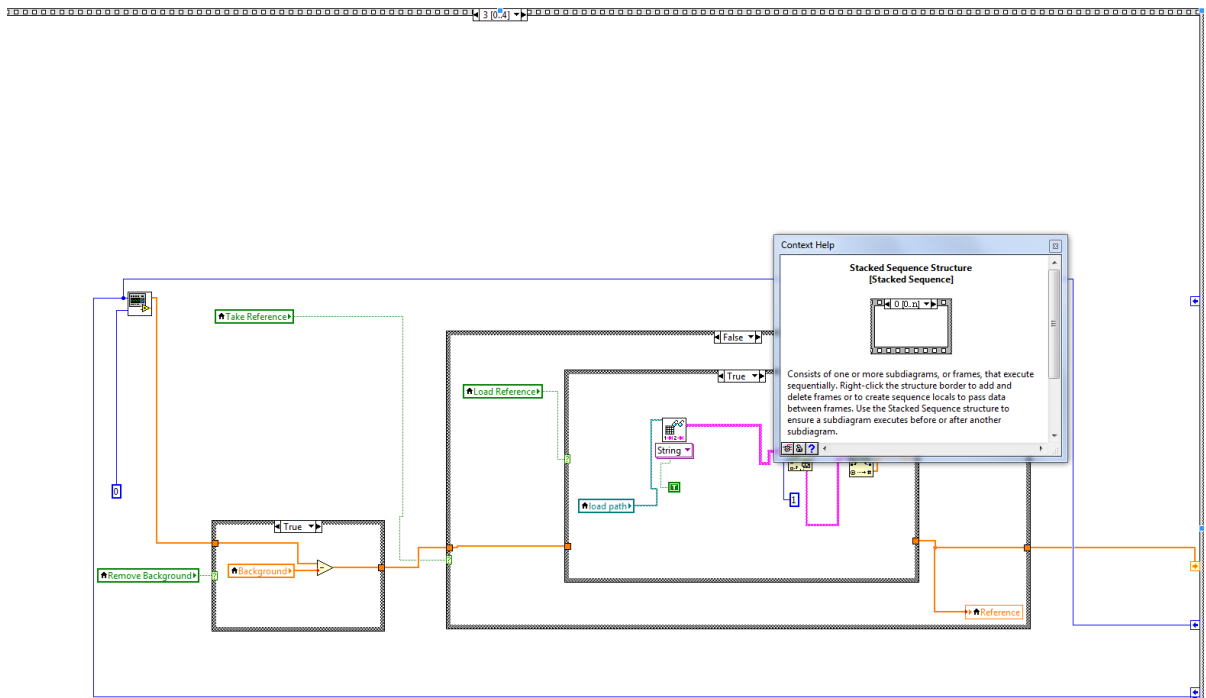


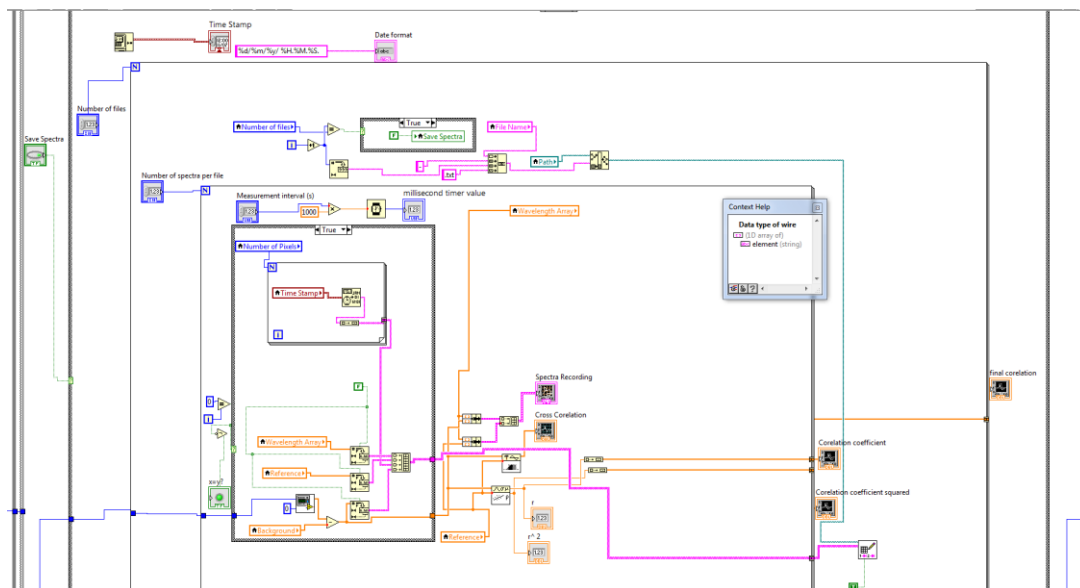
Figure 2.5: Initializing of spectrometer



**Figure 2.6: Taking/ Loading Reference**

The program takes all the measurements and writes them in a text file which contains the time stamps, the reference readings, the spectra readings and the wavelengths. The program also calculates co-relation coefficient and cross co-relation and puts them in the same text file. The program was tested in the lab by connecting the spectrometer to the experiment. The laser light was transmitted to the spectrometer with an optical fiber which was connected to the spectrometer. The program works fine with generating the results in the text file.

To process the text file, a MATLAB program has been generated which will be discussed in the next section.



**Figure 2.7: Recording Spectra**





Figure 2.8: Front Panel for the Program

debug-1.txt - Notepad															
File	Edit	Format	View	Help											
798.074219	-77.822845	8/17/2016	12:30:45	PM	-53.247217	8/17/2016	12:30:45	PM	-22.527686	8/17/2016	12:30:45	PM	-88.062686		
798.169983	-77.822845	8/17/2016	12:30:45	PM	-53.247217	8/17/2016	12:30:45	PM	-22.527686	8/17/2016	12:30:45	PM	-88.062686		
798.265686	-77.822845	8/17/2016	12:30:45	PM	-53.247217	8/17/2016	12:30:45	PM	-22.527686	8/17/2016	12:30:45	PM	-88.062686		
798.361389	104.446396	8/17/2016	12:30:45	PM	65.534990	8/17/2016	12:30:45	PM	-14.335791	8/17/2016	12:30:45	PM	75.774834		
798.457092	-110.590317	8/17/2016	12:30:45	PM	-108.542349	8/17/2016	12:30:45	PM	-77.822817	8/17/2016	12:30:45	PM	-116.734224		
798.552795	26.623604	8/17/2016	12:30:45	PM	6.143916	8/17/2016	12:30:45	PM	65.535010	8/17/2016	12:30:45	PM	12.287822		
798.648499	-88.062686	8/17/2016	12:30:45	PM	-43.007354	8/17/2016	12:30:45	PM	-16.383760	8/17/2016	12:30:45	PM	-28.671572		
798.744141	-22.527672	8/17/2016	12:30:45	PM	32.767485	8/17/2016	12:30:45	PM	-124.926108	8/17/2016	12:30:45	PM	-34.815483		
798.839844	-110.590363	8/17/2016	12:30:45	PM	-43.007393	8/17/2016	12:30:45	PM	-16.383799	8/17/2016	12:30:45	PM	2.047920		
798.935486	24.575630	8/17/2016	12:30:45	PM	-34.815464	8/17/2016	12:30:45	PM	-28.671558	8/17/2016	12:30:45	PM	-30.719526		
799.031067	-69.630943	8/17/2016	12:30:45	PM	-38.911411	8/17/2016	12:30:45	PM	-92.158999	8/17/2016	12:30:45	PM	-47.103286		
799.126709	12.287773	8/17/2016	12:30:45	PM	-36.863477	8/17/2016	12:30:45	PM	-73.726914	8/17/2016	12:30:45	PM	2.047930		
799.222290	-18.431713	8/17/2016	12:30:45	PM	67.582974	8/17/2016	12:30:45	PM	51.199224	8/17/2016	12:30:45	PM	51.199224		
799.317932	-59.391045	8/17/2016	12:30:45	PM	-8.191826	8/17/2016	12:30:45	PM	-106.494326	8/17/2016	12:30:45	PM	-45.055264		
799.413513	81.918793	8/17/2016	12:30:45	PM	43.007388	8/17/2016	12:30:45	PM	182.269263	8/17/2016	12:30:45	PM	51.199263		
799.509033	26.623613	8/17/2016	12:30:45	PM	-71.678887	8/17/2016	12:30:45	PM	24.575645	8/17/2016	12:30:45	PM	47.103301		
799.604614	-98.302513	8/17/2016	12:30:45	PM	-10.239858	8/17/2016	12:30:45	PM	-90.110640	8/17/2016	12:30:45	PM	-32.767515		
799.700134	-2.047920	8/17/2016	12:30:45	PM	-16.383701	8/17/2016	12:30:45	PM	22.527705	8/17/2016	12:30:45	PM	-75.774795		
799.795654	-75.774857	8/17/2016	12:30:45	PM	18.431709	8/17/2016	12:30:45	PM	61.439053	8/17/2016	12:30:45	PM	75.774834		
799.891174	-36.863392	8/17/2016	12:30:45	PM	-100.350425	8/17/2016	12:30:45	PM	-116.734175	8/17/2016	12:30:45	PM	-137.213862		
799.986694	-59.391079	8/17/2016	12:30:45	PM	-51.199204	8/17/2016	12:30:45	PM	-102.398423	8/17/2016	12:30:45	PM	-47.103267		
800.082155	-24.575634	8/17/2016	12:30:45	PM	-0.000010	8/17/2016	12:30:45	PM	-12.287822	8/17/2016	12:30:45	PM	-32.767510		
800.177673	61.439037	8/17/2016	12:30:45	PM	61.439038	8/17/2016	12:30:45	PM	24.575601	8/17/2016	12:30:45	PM	65.534976		
800.273132	131.070007	8/17/2016	12:30:45	PM	61.439063	8/17/2016	12:30:45	PM	86.014688	8/17/2016	12:30:45	PM	98.302500		
800.368591	77.822754	8/17/2016	12:30:45	PM	96.254473	8/17/2016	12:30:45	PM	20.479629	8/17/2016	12:30:45	PM	-10.239902		
800.463989	81.918739	8/17/2016	12:30:45	PM	49.151240	8/17/2016	12:30:45	PM	20.479678	8/17/2016	12:30:45	PM	14.335771		
800.559448	-28.671524	8/17/2016	12:30:45	PM	-28.671523	8/17/2016	12:30:45	PM	-26.623555	8/17/2016	12:30:45	PM	18.431758		
800.654846	20.479652	8/17/2016	12:30:45	PM	79.870747	8/17/2016	12:30:45	PM	94.206528	8/17/2016	12:30:45	PM	26.623560		
800.750344	77.822845	8/17/2016	12:30:45	PM	-22.527686	8/17/2016	12:30:45	PM	-75.774834	8/17/2016	12:30:45	PM	-116.734224		

Figure 2.9: An Example of the text file generated by LabVIEW

## 2.4. MATLAB code for processing:

Say for example the user has more than 100 different spectra readings and at different time intervals, this means that there are lots of different columns that are needed to be read and analyzed. Therefore a MATLAB program was necessary to process the measured data. The program takes the file name and the no. of spectra as the user argument and it will load the data which is in string format and will convert it to numerical format which is usable in MATLAB. When the program is run, it will generate columns with names automatically in a loop and will store the respective data in them. In the end we have the acquired data in numerical form in columns where we can analyze it by making graphs and such. The code is included in Appendix A.



### 3. The Camera Program

#### 3.1. Working with the Camera

To run the diagnostics on this experiment, we are using a Point Grey Black Fly PGE -13E4M-CS Camera, and doing data acquisition in LabVIEW. These kinds of camera manufactured by Point Grey, have highly sensitive and reliable sensors, are less expensive and can be described by two special characteristics name GigE Vision and PoE.

GigE Vision is a standard interface for high performance of industrial cameras. It provides the framework for transmission of video and control data at higher speeds over the Ethernet which is around 125 MB/s.

Power over Ethernet (POE) is a technology which powers the camera through network cables carrying electrical power. This provides a flexible, safe and cost saving alternative over other cameras.

Some important specifications for the camera we are using are that it is a 1.3 MP, monochromatic camera with a frame rate of 60 FPS (Frames per second) and a pixel size of 5.3 Micrometers.

Video and image data can be easily collected by NI Vision acquisition software compatible with NI LabVIEW. The software includes the NI IMAQ and the NI-IMAQdx libraries which can be used with GigE Vision Cameras, to do multiple operations such as grabbing and processing the image data.



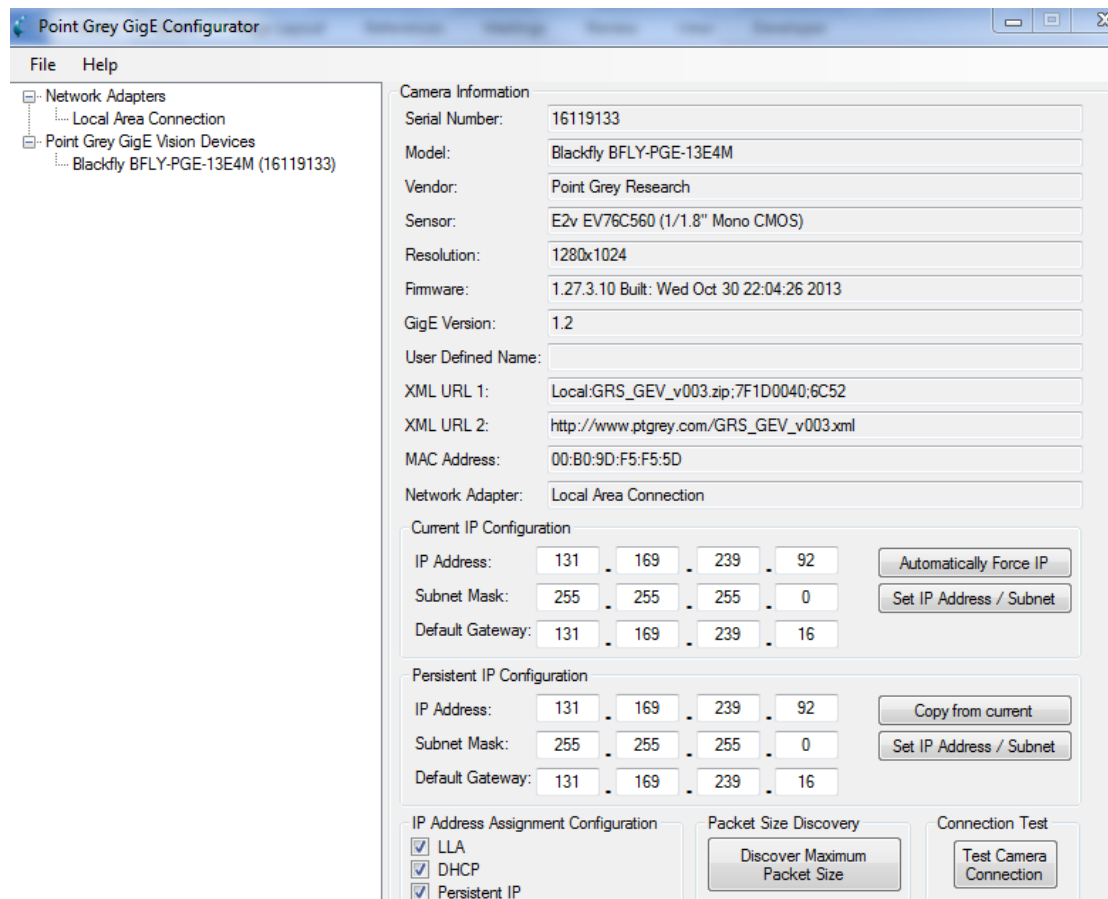
Figure 3.1: Point Grey Black Fly Camera

#### 3.2. Installation and Configuration of Camera:

It is critical that NI MAX, LabVIEW and NI Vision Acquisition Software are installed on the computer. Following steps describe how to install and configure the camera.

Step 1: Download the Camera software to the PC from Point grey website. Three Important things to download are: Latest FlyCapture2 Full SDK, FlyCapture2 - Windows GTK Runtimes, Blackfly PGE Firmware. It is important that all these software are downloaded for the respective windows version 32/64 bit. Except the firmware, download and install all the packages.

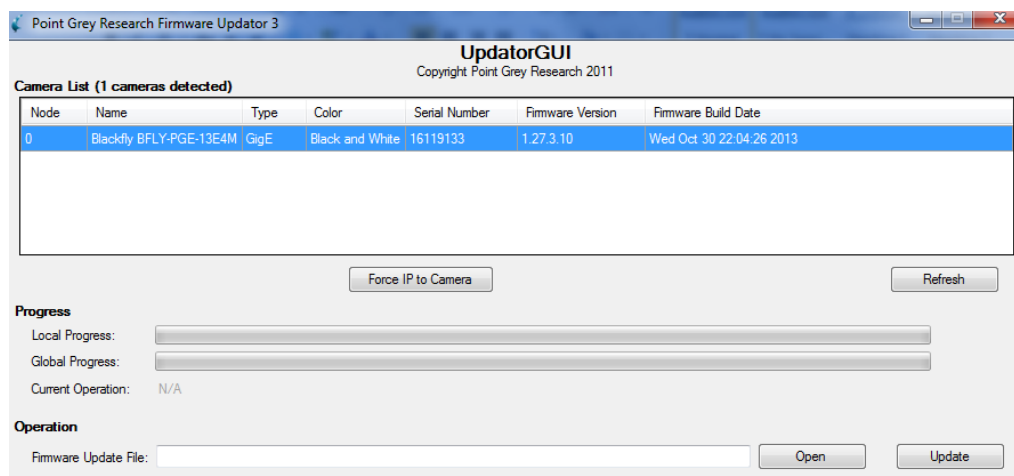
Step 2: Connect the camera to the main Ethernet port. Open the Gig-E Configurator software which comes with the camera software package. It is important here that the camera IP should be assigned in such a way that the camera and the computer are on the same subnet.



**Figure 3.2: GigE Configurator for configuring the camera ip**

Clicking on Automatically Force IP, assigns a temporary IP to the camera. Using this software the camera can be configured and a persistent IP can be assigned to the camera which is important for LabVIEW acquisition. Pressing test camera connection shows if the camera is properly communicating with the computer or not.

Step 3: Now connect the camera to the computer Ethernet port. Open the updater GUI3 included in the camera software package and update the camera firmware. Choose the firmware file downloaded in the first step and click update.



**Figure 3.3: Updater GUI3 – Firmware Updater**

Step 4: Check for Camera acquisition in FLY Cap2 Software and NI MAX. NI MAX can be used to set camera attributes and acquisition attributes for LabVIEW. It can also be used to view live video and snap images.

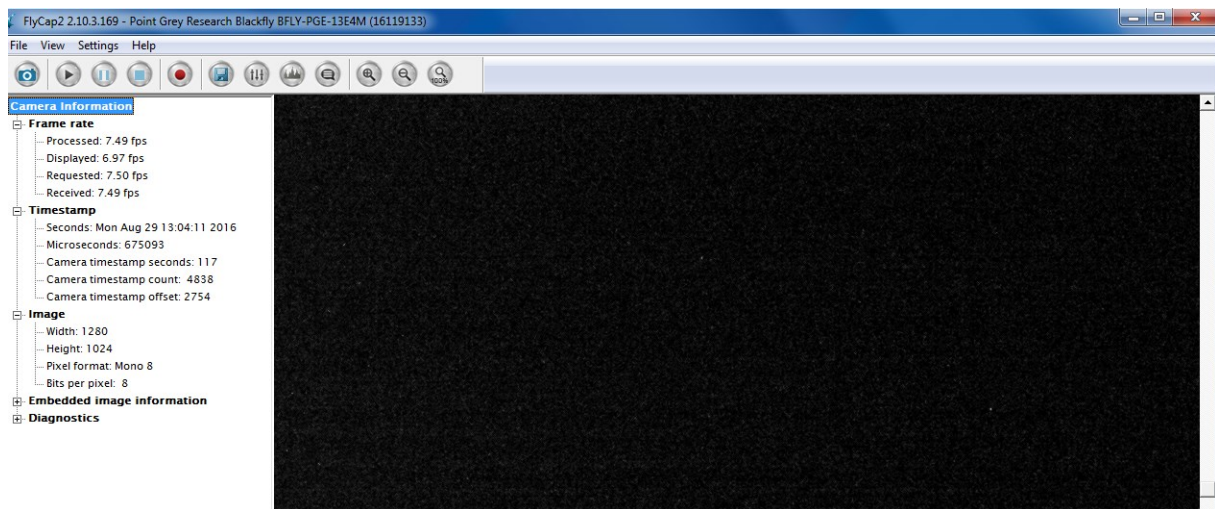


Figure 3.4: Fly Cap2 Software for Acquisition Preview

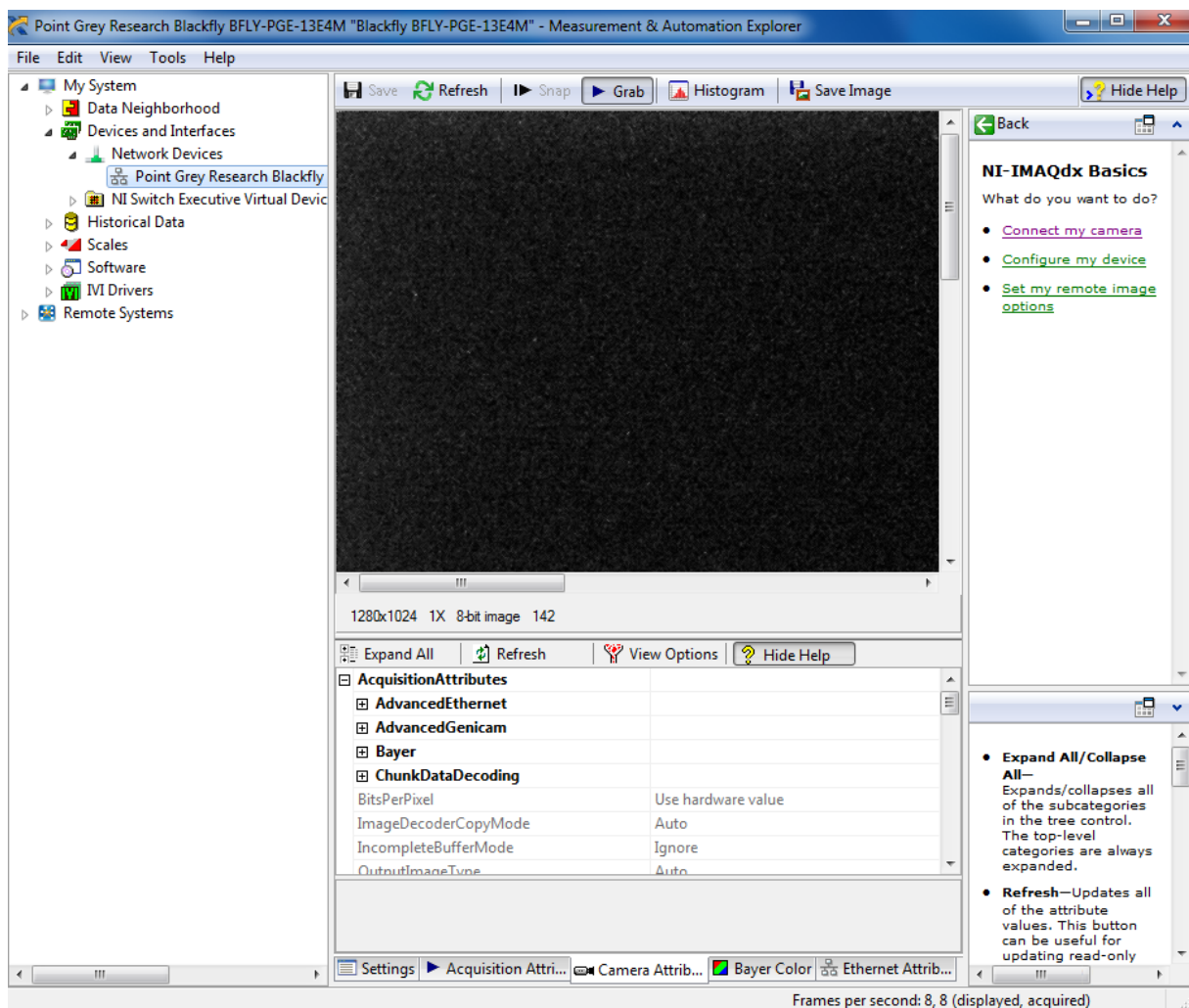
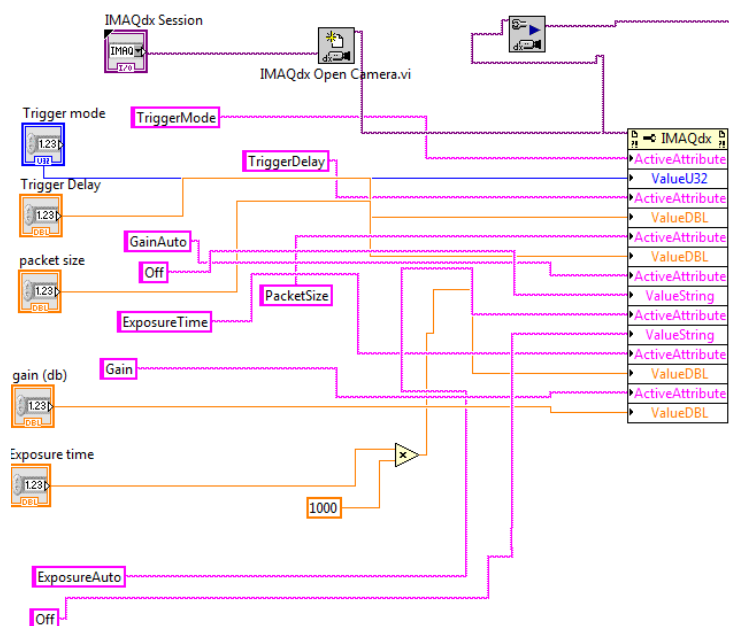


Figure 3.5: LabVIEW NI MAX for setting attributes and viewing live feed

### 3.3. Working of the Camera LabVIEW Program:

The vision library in LabVIEW has many different functions to choose from for data acquisition and processing the image data from the camera, which have been utilized to make a working LabVIEW program for a single camera. The program has two main components: Acquisition and Processing. Acquisition performs tasks such as grabbing live video, taking a reference image, taking multiple readings and comparing them. The aspect of noise removal is also added to the program. When used in the Laboratory with the laser experiment, the program computes the beam waist and beam spot position and saves the data in a text file. It also shows the intensity profile of the beam. The processing part of the program performs a few tasks on a single image such as, extraction of a certain part of an image, calculating line profile or distance between two points on an image. An explanation of the working and the architecture of the program and the library functions are mentioned in details below.

The acquisition starts with opening a camera session and using the function IMAQdx Open Camera. The first thing program does is to set attributes for video and image acquisition. Using a Property node and the function IMAQdx Configure Grab, camera attributes like trigger mode, trigger delay, gain, exposure time, packet size, etc. can be set by the user using the front panel.



**Figure 3.6: Setting attributes using property node**

After that is the live video module which uses the IMAQdx Grab and IMAQ AVI functions to create and save a video file. The user can input the no. of frames per second. The front panel shows the live video and intensity profile of the beam. User needs to stop the live video in order to record images.



values should be set to zero as the program is being used for a single camera and not for multiple cameras.

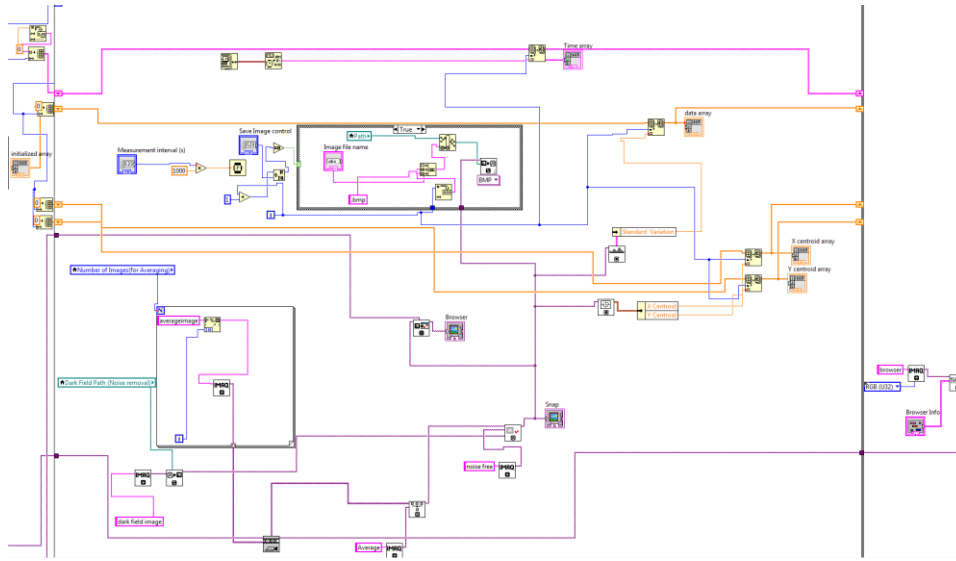


Figure 3.9: Example of recording images

When Save data is pressed, the program compares the beam waist and centroids for reference and the measurements and saves it in a text file to be processed later.

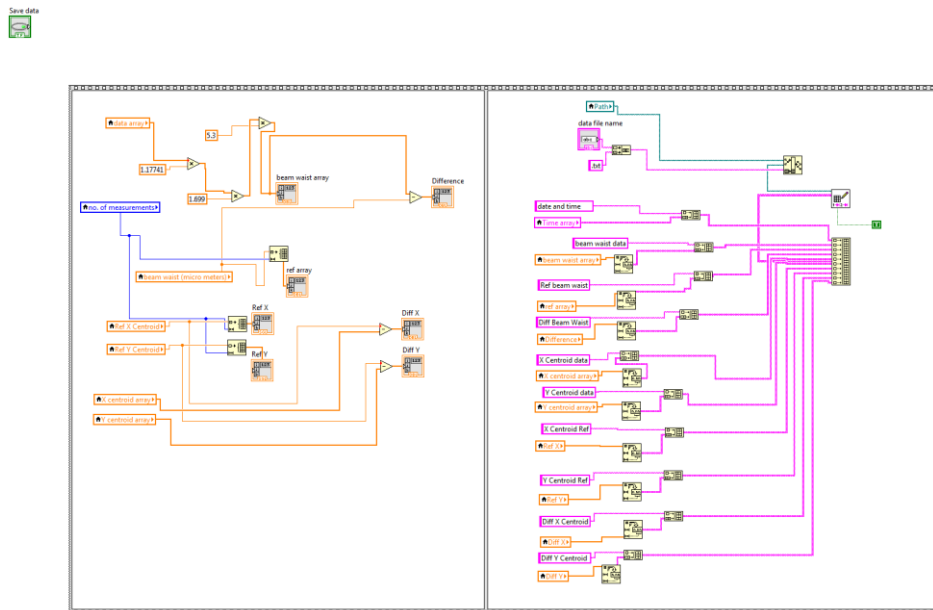


Figure 3.10: Example of saving data

The beam waist is calculated by using the standard variation provided by Imaq histogram function. FWHM can be calculated by the formula:

$$FWHM = \sqrt{2 \ln 2} \times \sigma$$

And beam waist at  $\frac{1}{e^2}$  is calculated by using the formula:

$$w = 1.699 \times FWHM$$

And the centroid is calculated by using the function IMAQ centroid. All the measurements can be seen in the image browser window which is created with the help of IMAQ browser functions.

The program can also be used to do some basic processing by loading a single image. The program can extract a part of image with the use of functions IMAQ Extract and the IMAQ ROI (Region of Interest) functions, calculate the distance from one point to another on an image using the distance formula and IMAQ ROI function

$$Distance = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

Also calculates the line profile and histogram of the whole or certain part of the image using IMAQ line profile and IMAQdx histogram functions when the user defines a region of interest and uses the IMAQ light meter to process an area selected on the image.

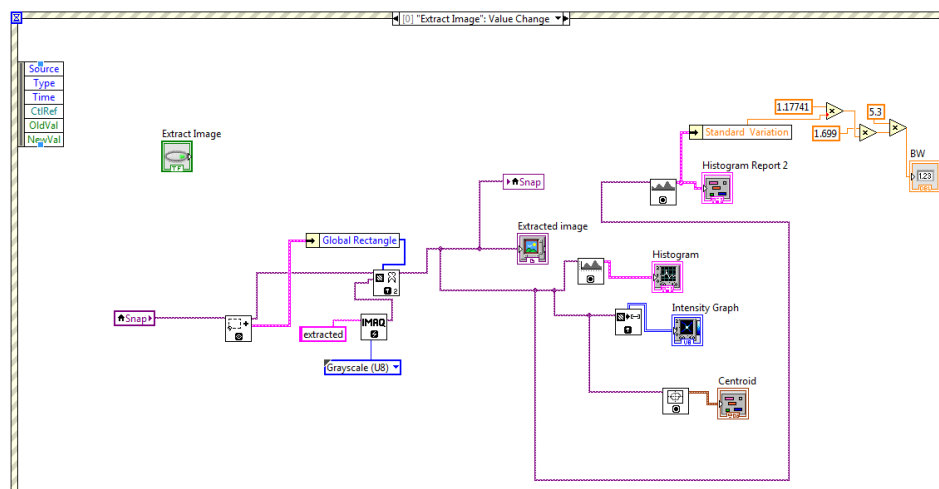


Figure 3.11: Example of Extracting image by defining a region of interest on the original image

### 3.4. Testing of Program and Experimental Setup:

To test the program in the lab, an experimental setup was made with a few optical components. The infrared filter was removed from the camera and lens tube was fitted into it to eliminate the surrounding light. A simple He-Ne laser was used with two mirrors to direct laser light in the direction of the optical wedge where beams are split into two parts so to make use of multiple cameras. The original beam goes through the optical filter to attenuate the transmission coefficient and then goes to the camera. A Plano convex lens can be used to focus the laser light on the camera.

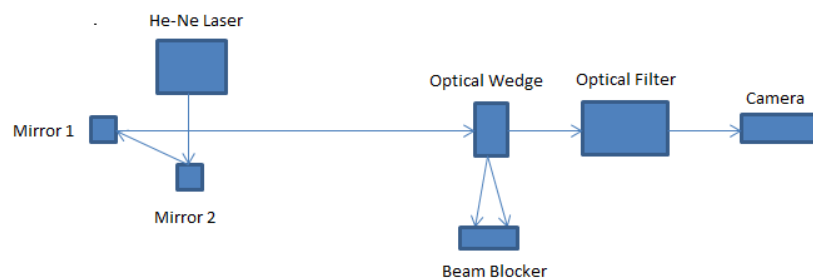


Figure 3.12: Experimental Setup



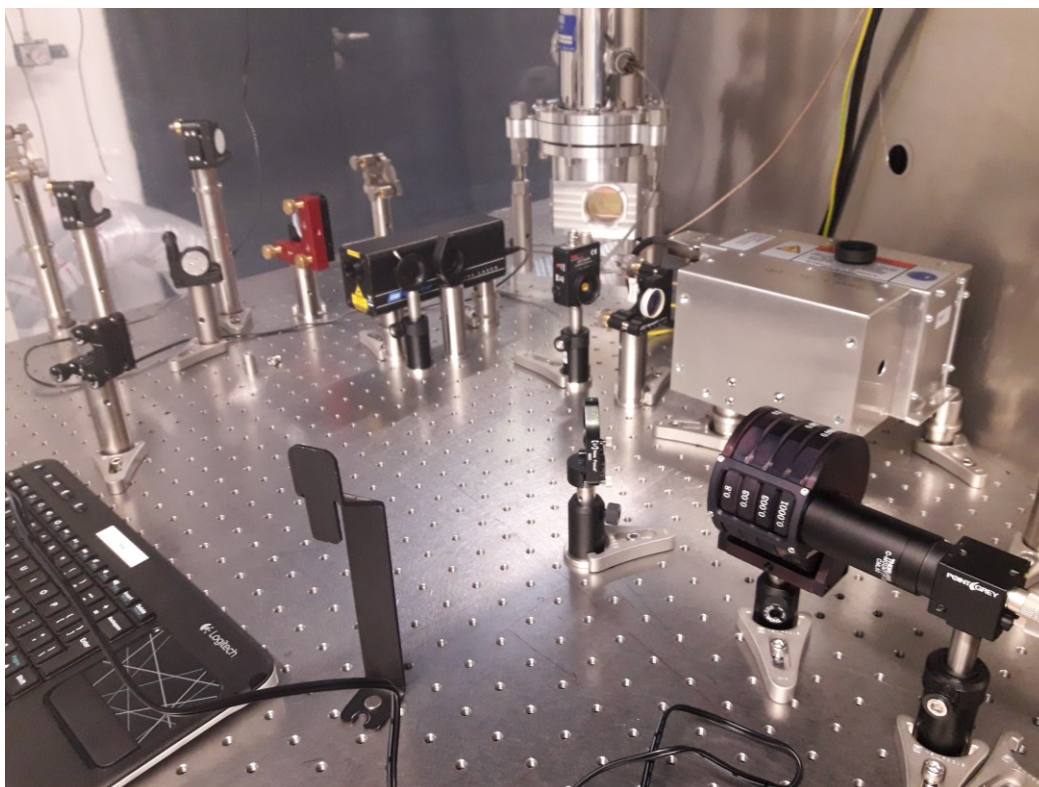


Figure 3.13: Experimental setup for testing the Camera program in the lab

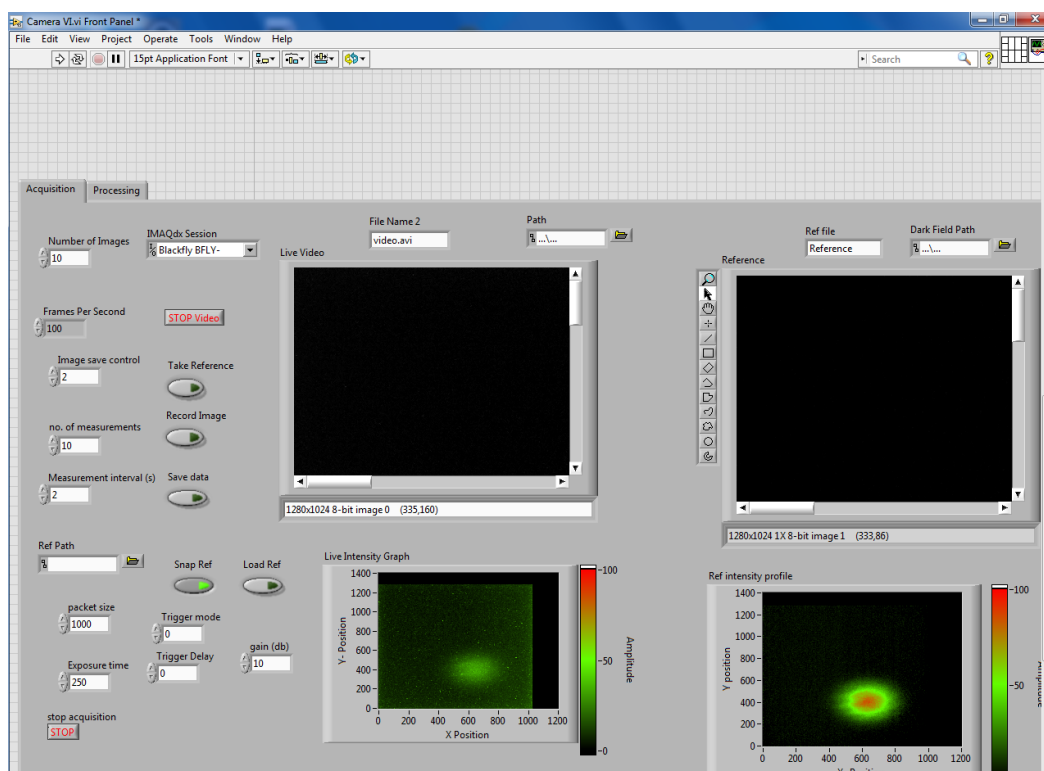


Figure 3.14: Front Panel of Acquisition showing user input parameters and live video

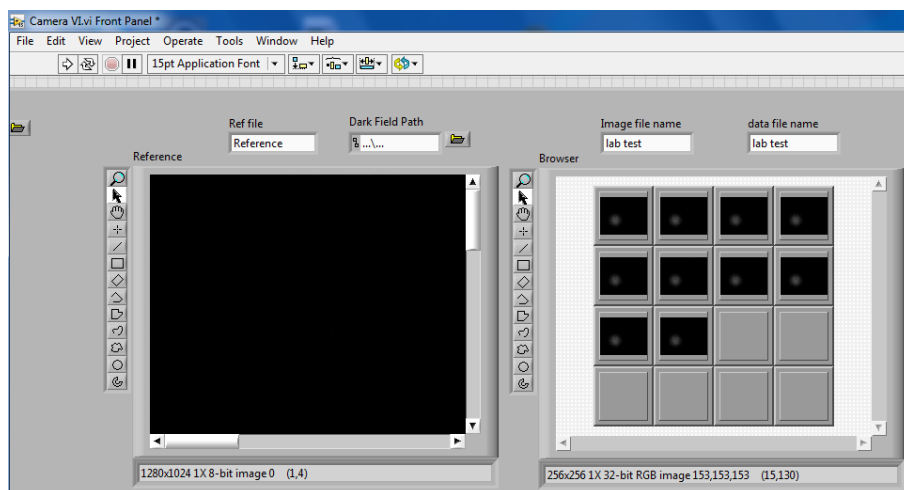


Figure 3.15: Front Panel of Acquisition showing no. of readings in the image browser

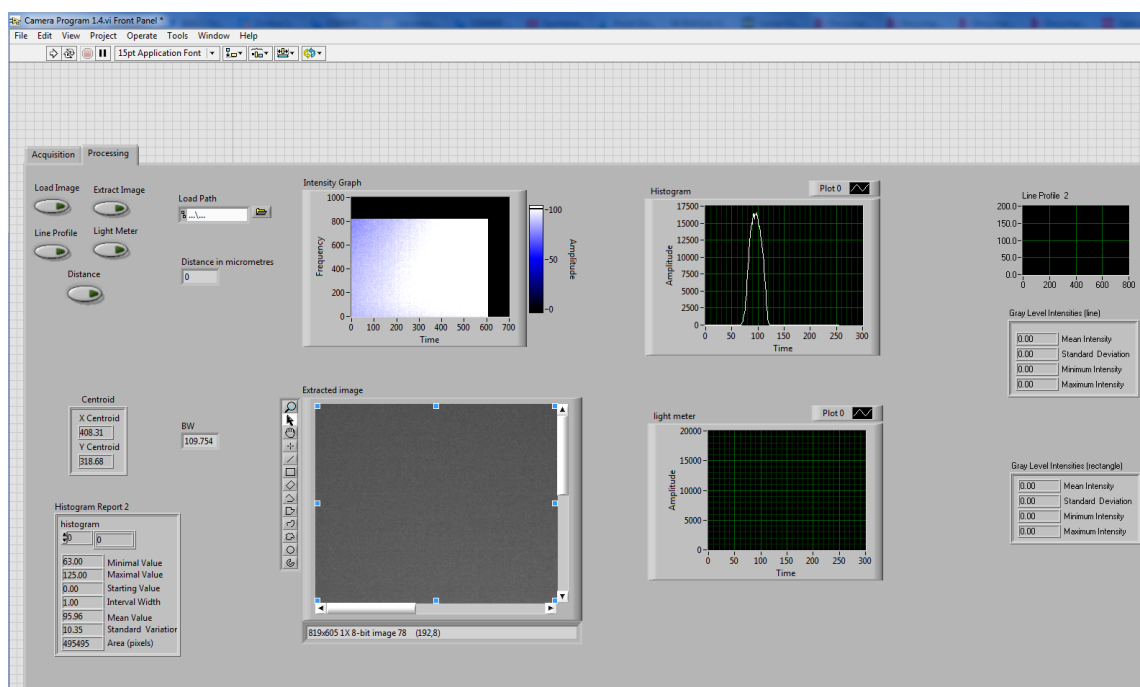


Figure 3.16: Front Panel of Processing

analysis6.txt - Notepad

Date and time	beam waist data	Ref beam waist	Diff Beam waist	X Centroid data	Y Centroid data	X Centroid Ref	Y Centroid Ref	Diff X Centroid	Diff Y Centroid
8/24/2016 4:07:29 PM	104.876715	144.839439	-39.962724	624.725159	511.164276	642.667358	514.737305	17.942200	3.573029
8/24/2016 4:07:31 PM	93.115257	144.839439	-51.724182	625.661865	521.900757	642.667358	514.737305	17.005493	-7.163452
8/24/2016 4:07:34 PM	110.671409	144.839439	-34.168031	603.970581	533.015869	642.667358	514.737305	38.696777	-18.278564
8/24/2016 4:07:36 PM	99.166560	144.839439	-45.672880	578.788391	585.656250	642.667358	514.737305	63.878967	-70.918945
8/24/2016 4:07:38 PM	102.487156	144.839439	-42.352283	578.831787	585.909302	642.667358	514.737305	63.835571	-71.171997
8/24/2016 4:07:41 PM	102.358867	144.839439	-42.480573	578.704041	585.207092	642.667358	514.737305	63.963318	-70.469788
8/24/2016 4:07:43 PM	102.654353	144.839439	-42.185087	578.841858	585.502075	642.667358	514.737305	63.825500	-70.764771
8/24/2016 4:07:45 PM	102.291021	144.839439	-42.548418	578.294556	585.851135	642.667358	514.737305	64.372803	-71.113831
8/24/2016 4:07:47 PM	102.489370	144.839439	-42.350069	578.512146	585.641418	642.667358	514.737305	64.155212	-70.904114
8/24/2016 4:07:50 PM	102.755504	144.839439	-42.083936	578.650940	585.694519	642.667358	514.737305	64.016418	-70.957214

Figure 3.17: Example of Text file generated by the Camera Program

More information on the NI IMAQ and NI IMAQdx functions is provided in the respective manuals provided by LabVIEW.

## References

1. *High-energy kHz Yb:KYW dual-crystal*. **Calendron, Anne-Laure, Çankaya, Hüseyin and Kärtner, Franz X.** s.l. : Optics Express, 2014, Vol. 20.
2. Few-Cycle Parametric Amplifiers and Sub-Cycle Waveform Synthesizers. [https://ufox.cfel.de/research/few\\_cycle\\_parametric\\_amplifiers\\_and\\_sub\\_cycle\\_waveform\\_synthesizers](https://ufox.cfel.de/research/few_cycle_parametric_amplifiers_and_sub_cycle_waveform_synthesizers). [Online]
3. Optical Parametric Chirped-pulse Amplification. [https://www.rp-photonics.com/optical\\_parametric\\_chirped\\_pulse\\_amplification.html](https://www.rp-photonics.com/optical_parametric_chirped_pulse_amplification.html). [Online]
4. *Temporal Characterization of Front-End for Yb-Based*. **Çankaya, Hüseyin, et al., et al.**
5. **Cankaya, Huseyin.** *Towards Table-Top Sub-Femtosecond X-Ray Sources*.

## Acknowledgements

This summer internship has been an exciting experience for me. I have learned new concepts relating to lasers and nonlinear optics and have gained new skills by working on LabVIEW and testing the programs in the laser lab by making an experimental setup. I would like to thank the following people for helping me and guiding me during my summer internship:

**Huseyin Cankaya:** For being a wonderful supervisor, teaching me lots of new concepts, taking me through the project step by step and guiding me towards making a better program.

**Nicholas Matlis:** For helping me understand concepts regarding the camera program and guiding me about features of the camera program

**Tobias Kroh:** for helping me with ideas and advice regarding different aspects of LabVIEW programming

**Adrian Pagel:** for helping with all the IT related stuff, and solving problems with the camera ip configuration.

**Christine Berber:** for helping me with all the administrative procedures.

## Appendix A

```
%% Import data from text file.
% Script for importing data from the following text file:
%
%     C:\Users\hemaniyo\Matlab\yousuftest11-1.txt
%
% To extend the code to different selected data or a different text file,
% generate a function instead of a script.

%% Initialize variables.
filename = 'C:\Users\hemaniyo\Matlab\testfile-1.txt';
delimiter = '\t';
prompt= 'Enter no. of spectra:    ';
x= input(prompt);
y=(2*x)+2;

for i=1:y
    if i==1
        myformat='%f';

    else if mod(i,2)==0
        myformat=strcat(myformat, '%f');

    else

        myformat=strcat(myformat, '%{dd.MM.yyyy HH:mm:ss}D');
    end
end
end

myformat=strcat(myformat, '%[^\n\r]');

myformat

%formatSpec = '%{dd.MM.yyyy HH:mm:ss}D%f%f%f%f%f{dd.MM.yyyy
HH:mm:ss}D%f%f%f{dd.MM.yyyy HH:mm:ss}D%f%f%f[^\n\r]';

%% Open the text file.
fileID = fopen(filename, 'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code.
dataArray = textscan(fileID, myformat, 'Delimiter', delimiter, 'EmptyValue',
NaN, 'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Allocate imported array to column variable names
for i=1:y
    eval(['column_' num2str(i) '=dataArray(:,i);']);
end

%% Clear temporary variables
clearvars filename delimiter myformat fileID dataArray ans;
```