

Realtime Focal Spot Characterization

M. Heavner

Advised by Dr. Christian Stoeckl

Laboratory for Laser Energetics

University of Rochester

250 East River Road

Rochester, NY 14623

March 13, 2007

Abstract

Short-pulse laser-plasma interaction experiments are carried out at the University of Rochester's Laboratory for Laser Energetics (LLE). In these experiments a short (< 1 ps) laser pulse is focused to a small (< 4 μm) spot on a solid target, rapidly creating a plasma. During alignment, the focal spot has to be located and characterized. A CCD camera provides high-resolution image data of the focal spot in analog format that is converted into a digital signal with a framegrabber, allowing it to be used by software. A program has been developed using the C and Java programming languages that accesses the framegrabber, acquires images, and interprets acquired data. It is able to locate the focal spot from the CCD data, graph the data, zoom into and autoscale the image, and infer the size of the focal spot.

1 Introduction

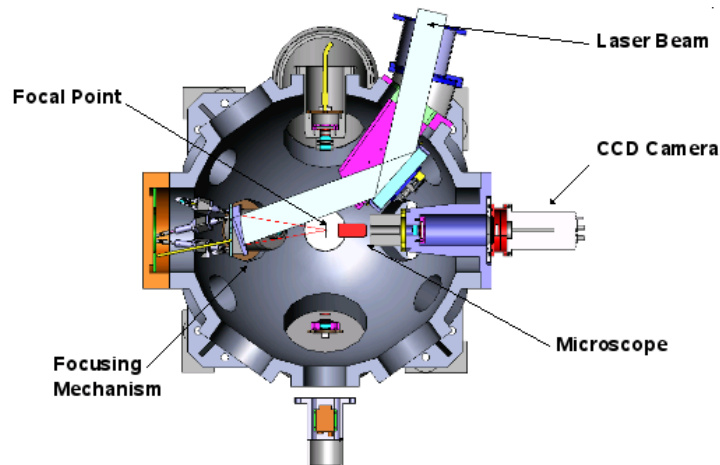


Figure 1.1: LLE's MTW Experimental Setup

In the University of Rochester's Laboratory for Laser Energetics (LLE), short pulse laser-plasma interaction experiments are carried out. In LLE's experimental setup (shown in **Figure 1.1**), a laser beam is sent into a chamber, reflected off a flat mirror and an off-axis parabolic mirror, and focused by a focusing mechanism on a small target. In the alignment of this setup, a requirement to locate and characterize laser focus points arises. Charge-coupled device (CCD) cameras, connected to microscopes, are used to provide real time, high-resolution image data of these points.

CCD cameras are used extensively within LLE experimental setups. CCDs provide high-resolution images and are up to 70% more photosensitive to incident light than photographic film. A CCD is composed of an array of photosensitive capacitors. During exposure, charge is transferred to non-photosensitive cells that are located in between active rows. The charge in these storage cells is shifted along the row one row at a time to form a video signal. This occurs while the active cells are being exposed to produce the next frame.

To access and interpret image data, a framegrabber must be installed in the client computer that is accessing the CCD camera. A framegrabber converts the analog video signal from the CCD to a digital signal, as shown in **Figure 1.2**, that is usable by a computer. The framegrabber used in the development of this program was a DT3155, created by Data Translation, Inc.¹ The DT3155

provides four multiplexed monochrome inputs, 8-bit digitization, programmable black and white levels, real-time image thresholding and image scaling and clipping.¹



Figure 1.2: A CCD camera, DT3155 framegrabber and DTControl in use^{2,3,4}

A program has been developed to actively locate and characterize a laser focal point using data acquired from a framegrabber that is attached to a CCD camera. It uses both the Java and C programming languages and the open-source Linux DT3155 driver⁵, modified to fit LLE needs. This program (DTControl) has been designed to be as self-intuitive as possible, allowing a scientist to start using it without previous knowledge of its inner workings and to still acquire necessary and useful information. Tools and features have also been added to allow the program to adapt to different experimental environments. The program has been designed to accommodate future upgrades.

2 Programming Technologies

Many operating systems are in use within LLE. Because of this, program code must be portable so that it may be used on different platforms. This program was designed and tested on a Linux machine. The GNU/Linux operating system⁶ has proven to be very successful and stable, and is used frequently within LLE because it allows code to be portable. The Java and C programming languages, along with the Java Native Interface, were utilized in the creation of this program. Java, which began as a Sun Microsystems project termed Oak that sought to provide an alternative to C/C++⁷, allows for the creation of code that can be used on many platforms. This is possible through the use of the Java Virtual Machine (JVM), which itself is platform-specific. Java code is created

and, when run, is interpreted by the JVM and executed on the desired platform. Methods to access hardware were intentionally left out of the Java programming language in order to provide portability across varying hardware. The C programming language, developed in 1972 in the Bell Telephone Laboratories by Dennis Ritchie for use on the Unix operating system⁸, is known as a lower-level programming language that provides direct access to computer memory and other intricacies. Code developed in C is platform-specific and is often more efficient than code written in other languages, such as Java, as it doesn't have overhead code, such as Java's JVM. It is useful when interfacing hardware or operating systems directly.

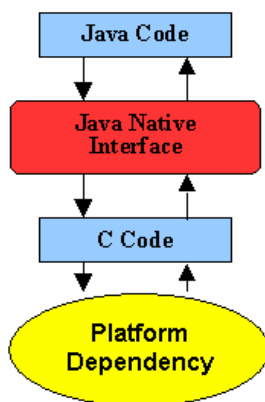


Figure 2.1: The Java Native Interface allows native languages to communicate with Java

In the event that a Java program must access platform-specific features, or code written in a lower-level language, the Java Native Interface must be used (see **Figure 2.1**). The Java Native Interface provides a way for Java code to utilize software libraries written in a native language, e.g., C or C++⁹. In the case of DTControl, it was used to provide the Java code with access to the DT3155 framegrabber.

3 Program Requirements

Requirements of the program needed by the user (LLE scientists) include the ability to locate and track a laser focal point, to measure and characterize such a point via graphing and interpretation

of data, to save and load images, and to adjust to various environments. The user must also be given the option of changing the accuracy of these features. Internally, the program must be robust and accurate, as many other process and tools are used concurrently on LLE computers.

4 Algorithm Design and Analysis

Algorithms were developed to allow for zooming, the location of a laser focal point, the measurement of a laser focal point, and auto-scaling of the image intensities to allow for better contrast.

4.1 Digital Zoom Algorithm

Upon a request from a user to zoom into a certain portion of an image, a boxed area of data surrounding the clicked point is partitioned. Pixels found within this area are expanded so that the selected portion of the image fills the entire viewing area. This may only be done a certain maximum amount of times, as zooming into a portion of the image too far will make it appear overly pixelated.

4.2 Focus-location Algorithm

Other algorithms and program aspects are dependent upon an accurate focal-spot center being found. This algorithm must be efficient because it is run very frequently (defaulted at every five frames). The process of finding the center involves two stages. First, an approximate center is located. This result is used in the second step that narrows the given information into an accurate result.

The first stage, location of the approximate center, works as follows: First, every n^{th} line (an undersampling rate of n) is checked for pixels of intensities greater than a certain value. These values are stored in an ArrayList, a dynamic storage structure. Second, the surrounding intensity (defined as the average over a six by six pixel area) of each previously found pixel is calculated. The

pixel with the greatest surrounding intensity is the approximate center.

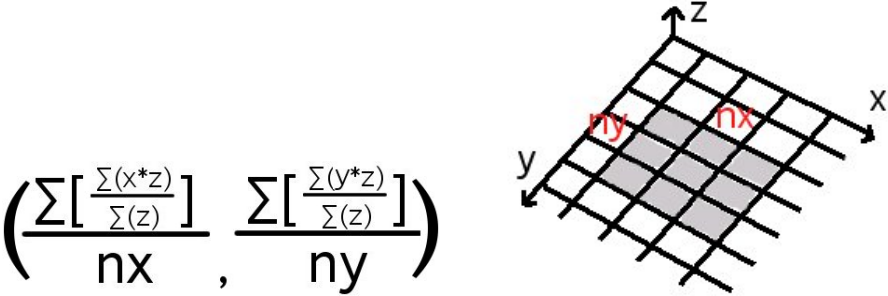


Figure 4.1: Second stage of the focus location algorithm

The location of this approximate center is then sent to the second stage of the process, which uses the formula shown in **Figure 4.1** to obtain the spot center (x,y) as the centroid (weighted average) over a region of nx by ny pixels. Here, x represents the X coordinate, y the Y coordinate, z the pixel intensity, and nx and ny the number of rows and columns checked, respectively. This method has been found to be highly accurate.

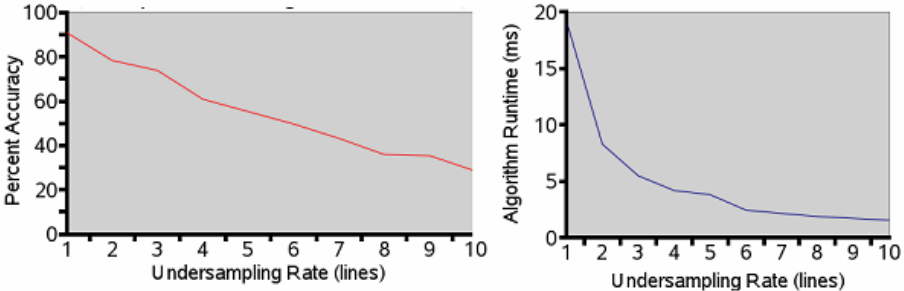


Figure 4.2: Accuracy and runtime for spots with a varying undersampling rate

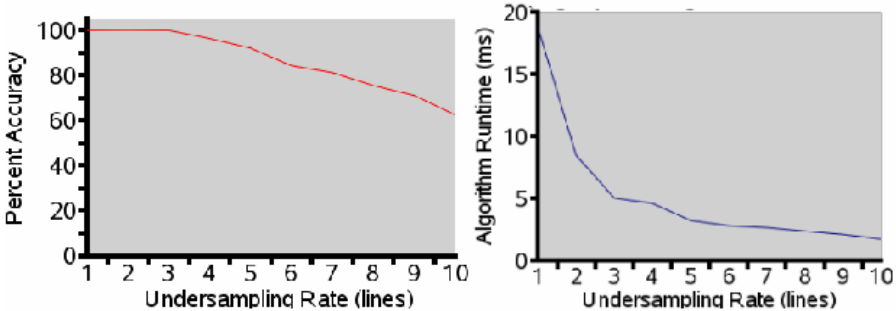


Figure 4.3: Accuracy and runtime for large spots with a varying undersampling rate

The main factor determining the efficiency and runtime of this algorithm is the undersampling rate n. The program allows the user to set the rate up to 10, which means that every 10th line is

checked. The algorithm was run one thousand times at each undersampling rate, with noise levels and spot-center locations randomized. **Figure 6.2** shows two generated images used in this testing process. The percent accuracy, or average error over the thousand runs when attempting to locate the center, and the associated runtime are shown in **Figures 4.2** and **4.3** as functions of the undersampling rate. A balance between efficiency and accuracy must be determined. Clearly, the larger the focal spot, the greater one can afford to increase the undersampling rate and still remain accurate, thereby saving CPU cycles.

4.3 Focal-measurement Algorithm

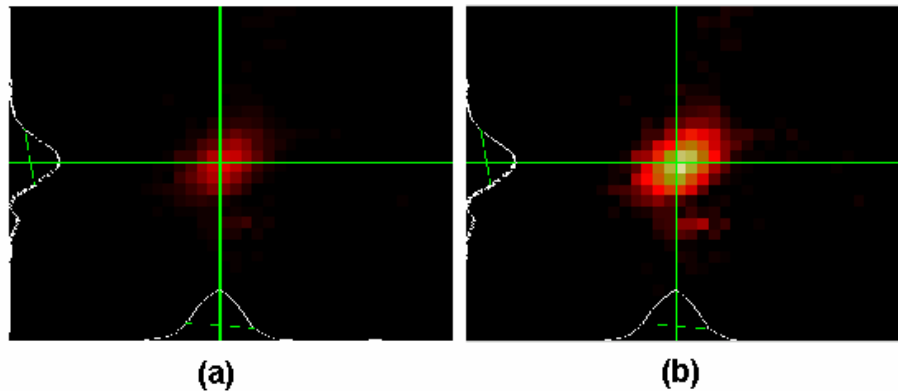


Figure 4.4(a): Experimental image of a focal spot, indicating the center of the spot and lineouts through the center of the spot in the horizontal and vertical directions

(b): Autoscaled image, with the center intensity scaled to 256.

An experimental focal spot is shown in **Figure 4.4(a)**, needing to be measured and autoscaled so that it might have a contrast that is easier to view.



Figure 4.5: Full width at half maximum for a given curve¹⁰

When characterizing a focal point, knowledge of the full width at half maximum (FWHM) is often very useful. This value, indicated graphically in **Figure 4.5**, is the distance between two points on a curve whose value is half that of the maximum value of the curve.¹¹ The algorithm goes through the following process: First, the intensity at the located focal center is obtained and the half-maximum value is calculated. Second, the algorithm traverses through the data vertically and horizontally until the points equal to or less than the half-maximum value are found. Noise is avoided in this step by checking points beyond the point in question to see if a decrease in intensity is consistent or rather noise-induced. Third, the distance between these two located points is calculated. The thin green lines shown in **Figure 4.4** indicate the FWHM.

The ability of the program to auto-scale image data is often imperative to providing a viewable image contrast. An algorithm to auto-scale pixel intensities has been developed. It calculates the auto-scaled intensity I' as $I \cdot (256/m)$ where I is the original intensity and m is the intensity at the focal center. The auto-scaled image of **Figure 4.4(a)** is shown in **Figure 4.4(b)**.

5 Program Design

DTControl follows the object-oriented model of programming, creating various objects that carry out different tasks. **Figure 5.1** shows DTControl's class layout, subdivided into classes of similar purpose.

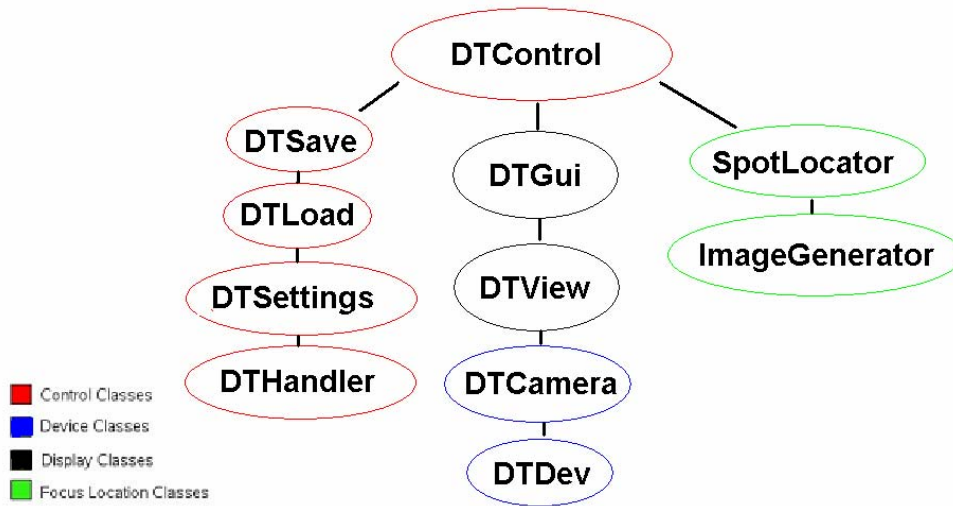


Figure 5.1: Program class hierarchy, subdivided into objects serving various purposes

The device classes provide an interface for accessing a CCD camera. They include DTCamera, which represents a CCD camera, containing data structures to hold image data and abstractions to acquire images, and DTDev, which provides access through the Java Native Interface to a C library. This library interfaces with a Linux driver that provides access to a DT3155 framegrabber.

The display classes are associated with the proper interpretation and display of data acquired from device classes. They include DTGui and DTView. DTView is responsible for displaying the information from one DTCamera. DTGui represents the entire user interface of the program, and assembles and displays a DTView and other components.

The focus location classes are responsible for the implementation of the focus location algorithm. SpotLocator provides algorithms for locating the laser spot and for calculating the FWHM values. ImageGenerator generates random images that are used to simulate laser focal spots.

The central classes control and synchronize other program features. DTSave and DTLoad save and load still images and DTSettings provides access to many program settings. Finally, DTControl controls all other classes and synchronizes GUI and camera updates.

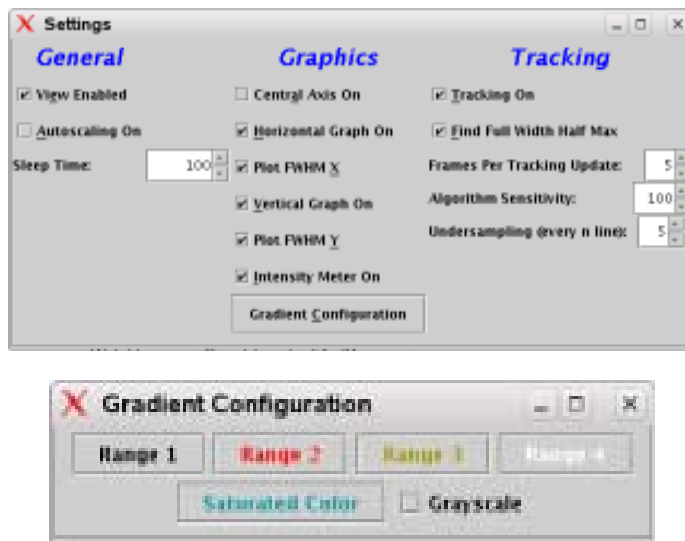


Figure 5.2: The user interface and options, designed to be self-intuitive

A user interface has been developed that allows the user to select many features. Selecting an appropriate color gradient for images such as **Figure 4.4** is important, as protective glasses used in laboratories often block certain colors. Color gradients, as well as the undersampling rate and algorithm sensitivity, can be changed by the user. Other features, such as a pixel intensity bar, displayed graphs, the calculation of FWHM values, the display of axes, and more, can also be enabled and disabled.

6 Program Testing

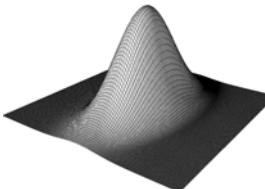
$$I = I_0 e^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)}$$


Figure 6.1: A two-dimensional Gaussian function¹²

A process whereby the program could be tested outside the real environment was needed

because it wasn't always possible to test the program in the real environment. The program was given the capability to generate virtual images to display from a virtual CCD camera. These images were composed of a two-dimensional Gaussian function (see **Figure 6.1**) with added noise, which produces a very natural distribution that appears as an ideal focal spot. The noise was added by augmenting a random number of pixels, between 0 and 1/5 of the image's area, by a random value between 1 and 255. Examples of such images are shown in **Figure 6.2**, **6.2(a)** with tracking enabled and **6.2(b)** with tracking disabled.

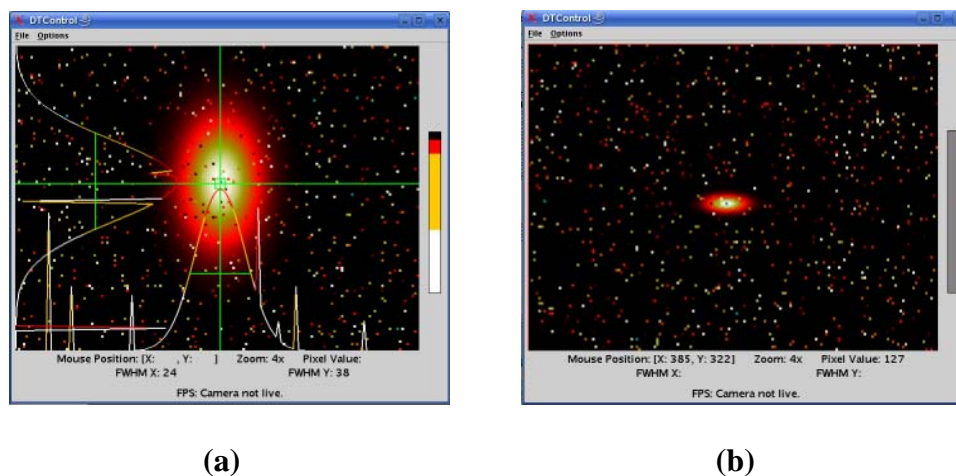


Figure 6.2: Test images produced from two-dimensional Gaussian functions with added noise.

(a) Tracking enabled; (b) tracking disabled.

7 Conclusion

Short-pulse laser plasma interaction experiments are carried out within LLE. They require a program that can locate and characterize a laser focal spot in real time. Such a program has been developed using the C and Java programming languages. The design of the program assigns different roles to various objects, creating a structured design that is easy to upgrade or alter in the future. The program and its various algorithms have been tested and proven to be accurate in both simulated and real laboratory environments. The program has already been used in the alignment and testing of lasers within LLE's Laser Development Laboratory and will continue to be used

during experiments at LLE.

8 Acknowledgments

I would like to thank my advisor Dr. Christian Stoeckl for his guidance, hints and words of advice during the progression of this project. I would also like to thank Dr. R. Stephen Craxton for providing the opportunity to pursue research at The University of Rochester's Laboratory for Laser Energetics this past summer.

9 References and Sources

1. DT3155 – Full Specification. Data Translation, Inc. 16 Aug. 2006
<http://www.datx.com/products_hardware/prod_dt3155_spec.htm>.
2. Online Image. CCD Camera. 6 Oct. 2006
<http://www.ni.com/third_party/sony/pdf/xcsT70_ce.pdf#search=%22xcst70_ce%22>.
3. DT3155 – Full Specification. Data Translation, Inc. 16 Aug. 2006
<http://www.datx.com/products_hardware/prod_dt3155_spec.htm>.
4. Online Image. Computer Clip Art. 14 Aug. 2006
<<http://www.sbac.edu/~tpl/clipart/Technology%20and%20Machines/computer%2001.jpg>>.
5. DT3155 Framegrabber Linux Device Driver. Version 1.8. <<http://sourceforge.net/projects/dt3155a>>.
6. The Linux Kernel Archives. <<http://www.kernel.org>>.
7. Kabutz, Dr. Heinz. Java History 101: Once Upon an Oak. 30 Jan., 2003. 24 Aug. 2006
<<http://www.devx.com/Java/Article/10686/0/page/1>>.
8. Aitken, Peter. Learning C. Carmel: SAMS, 1991.
9. Stearns, Beth. Trail: Java Native Interface. 21 Aug. 2006 <<http://www.iut-info.univ-lille1.fr/docs/tutorial/native1.1/index.html>>.
10. Online Image. Full-width at half-maximum. 10 Aug. 2006
<http://cfao.ucolick.org/EO/steinb/education_outreach/demoweb/home/FWHM.gif>.
11. Weisstein, Eric W. "Full Width at Half Maximum." MathWorld-A Wolfram Web Resource. 10 Aug. 2006
<<http://mathworld.wolfram.com/FullWidthatHalfMaximum.html>>.
12. Online Image. 2d gaussian. 16 Aug. 2006
<<http://local.wasp.uwa.edu.au/~pbourke/other/distributions/gaussian6.gif>>.