2015

Measurement of solid rocket propellant burning rate using x-ray imaging

Matthew D. Denny

MEASUREMENT OF SOLID ROCKET PROPELLANT BURNING RATE USING X-RAY IMAGING

by

MATTHEW D. DENNY

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in
The Department of Mechanical and Aerospace Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, AL

2015

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Matthew D Denny — 5/20/2015

Matthew D. Denny (Date)

THESIS APPROVAL FORM

Submitted by Matthew Denny in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Systems Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Systems Engineering.

Robert A. Frederick 5/20/15 Committee Chair
Dr. Robert A. Frederick (Date)

David M. Lineberry 5/20/15
Dr. David M. Lineberry (Date)

George J. Nelson 5/11/15
Dr. George J. Nelson (Date)

D. K. Hollingsworth 5/21/15 Department Chair
Dr. Keith Hollingsworth (Date)

Jennifer M. English 5/21/15 College Dean
Dr. Shankar Mahalingam (Date)

5/20/15 Graduate Dean
Dr. David Berkowitz (Date)

ABSTRACT

The School of Graduate Studies
The University of Alabama in Huntsville

Degree Master of Science in Aerospace Systems Engineering

College/Dept. Mechanical and Aerospace Engineering

Name of Candidate Matthew D. Denny

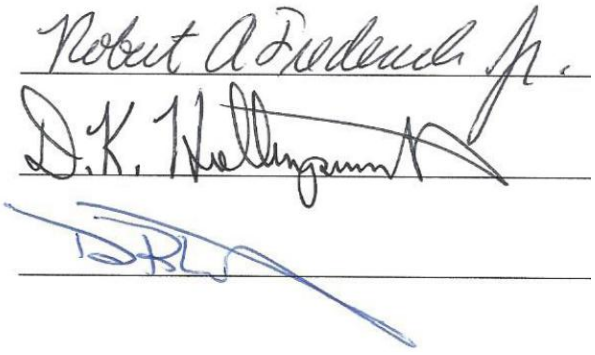
Title Measurement of Solid Rocket Propellant Burning Rate Using X-ray Imaging

The burning rate of solid propellants can be difficult to measure for unusual burning surface geometries, but X-ray imaging can be used to measure burning rate. The objectives of this work were to measure the baseline burning rate of an electrically-controlled solid propellant (ESP) formulation with real-time X-ray radiography and to determine the uncertainty of the measurements. Two edge detection algorithms were written to track the burning surface in X-ray videos. The edge detection algorithms were informed by intensity profiles of simulated 2-D X-ray images. With a 95% confidence level, the burning rates measured by the Projected-Slope Intersection algorithm in the two combustion experiments conducted were 0.0839 in/s ±2.86% at an average pressure of 407 psi ±3.6% and 0.0882 in/s ±3.04% at 410 psi ±3.9%. The uncertainty percentages were based on the statistics of a Monte Carlo analysis on burning rate.

Abstract Approval: Committee Chair

Department Chair

Graduate Dean



ACKNOWLEDGMENTS

There are several people who made this work possible and deserve recognition and especially my gratitude. I want to thank Dr. Robert Frederick for his skilled oversight of the research and his extensive, detailed guidance in designing and conducting experiments, in studying data, and in producing this thesis. I would also like to thank Dr. David Lineberry for providing a multitude of insights into experiment designs and data analysis techniques. I also want to thank Dr. George Nelson for sharing his expertise in X-ray imaging techniques and data analysis. All three of these professors gave generously of their time to invest in the project and to instruct me personally.

I am very grateful to Ph.D students Andrew Hiatt and Joe Buckley for being committed, excellent teachers and coworkers in the laboratory. I am also very grateful to Mr. Tony Hall for providing extensive training and assistance with the design, construction, and operation of experiment apparatus and materials. I had the fortune of working alongside of Mr. Hiatt, Mr. Buckley, Mr. Hall, and Dr. Lineberry to construct experiment apparatus and to conduct various experiments in the laboratory.

Several other people have my appreciation and thanks. Mr. Anthony Edmondson provided project administration and assistance in the procurement of equipment and materials. Ph.D student Matthew Hitt provided assistance in the construction of the combustion bomb for the X-ray experiments. Mr. Jason Winningham provided assistance in the electrical wiring of the X-ray equipment. Mr. Royal Ritchie, Mr. Steve Collins, and Mr. Tony Doll provided machining services and guidance. At last, I want to thank my family for their endless encouragement, interest, and support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER 1

INTRODUCTION

1.1 Introduction

Many missiles and satellites rely on small thrusters for flight control. Missiles used by the U.S. Navy must have propellants that can be safely stored and operated aboard sea-faring vessels [1]. Liquid propellant thrusters have been effective for some Divert/Attitude Control Systems (DACS) including that of the Space Shuttle [2], but liquid propellants are not desirable in Navy applications because of the storage difficulty and health hazards [3]. Also, solid rocket motors generally have higher mass fractions, but they generally cannot be throttled without the addition of equipment that reduces the mass fraction [2].

Electrically-controlled solid propellants (ESP) can be throttled, extinguished, and re-ignited, and they are generally inert at atmospheric pressure or vacuum unless activated by electrical potential [3]. Therefore, these propellants can have the advantages of both liquid and solid propellants. Electrically-controlled solid propellants are a desirable rising technology, but they are largely still in development [4]. Because of the geometries required to provide electrical influence to the propellant, ESP can have unusual regression contours that are difficult to track, which means that several common

methods of burning rate determination are difficult to use with ESP. This difficulty has led to the use of real-time X-ray imaging to measure the burning rate of the propellant.

As a part of this study, a new X-ray imaging system was set up at the University of Alabama in Huntsville (UAH) to determine burning rates and behavior of solid propellants in a laboratory setting. As later described in the literature review, real-time X-ray imaging has been used to characterize the burning rate of solid and hybrid propellants by other researchers in the past, but it is believed that this work represents the first historical use of X-ray imaging to measure the burning rate of an ESP. The accuracy of the measurements has been assessed, and the burning rates obtained from the X-ray system have been compared to burning rates obtained for the same propellant by other techniques. The experimental approach and results are presented, and recommendations for improvement and future research are given.

1.2 Objectives

There were two objectives of the research described herein. The first objective was to measure the burning rate of a solid propellant using an X-ray imaging system that was constructed in the High Pressure Laboratory (HPL) at UAH. The second objective was to determine the uncertainties of the burning rate measurements and compare them to the uncertainty of solid propellant burning rate measurements obtained by X-ray imaging techniques by other researchers in the past.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Characterization of Burning Rate

In the design of a solid rocket motor, it is essential to know the burning rate of its solid propellant. The burning rate is the rate at which the burning surface regresses in the direction perpendicular to the plane of the burning surface. The prediction of a thrust-time curve is very important for planning the trajectory of a rocket vehicle, and it is heavily dependent upon the solid propellant burning rate. According to Fry [5], if the burning rate is known to an accuracy of ±1%, then the thrust-time prediction accuracy is typically limited to about 1.5-2%. Theoretical models of the burning rate of solid propellants do not usually have that level of accuracy, so the burning rate must be experimentally determined. Even if models with such accuracy exist for some propellants, a new propellant would still need to be experimentally tested to prove that the burning rate prediction was accurate. As described by Fry, several methods of experimental burning rate measurement have been studied such as pressure-time correlations in motors, wired strand burners, optics, ultrasonic waves, X-rays, Gamma rays, and others.

The burning rate of solid propellants varies with chamber pressure in the motor case. Therefore, the burning rate, chamber pressure, and nozzle geometry are all coupled parameters. The burning rate equation, or St. Robert's law, is given by Hessler [6] as

$$r = ap^n, \tag{2.1}$$

where r is the burning rate, a is the temperature coefficient, p is the chamber pressure, and n is the pressure exponent.

The values of the temperature coefficient a and the pressure exponent n are generally constant for a given initial propellant temperature and a given propellant formulation. The values are determined through experimental testing at different pressure ranges. Regression rates are plotted versus pressure, and the values of a and n are adjusted until a curve is fitted to the data. The value of n for most propellants is greater than zero and less than one. An example plot of burning rate versus pressure data is given in Figure 1 from Sutton [2].

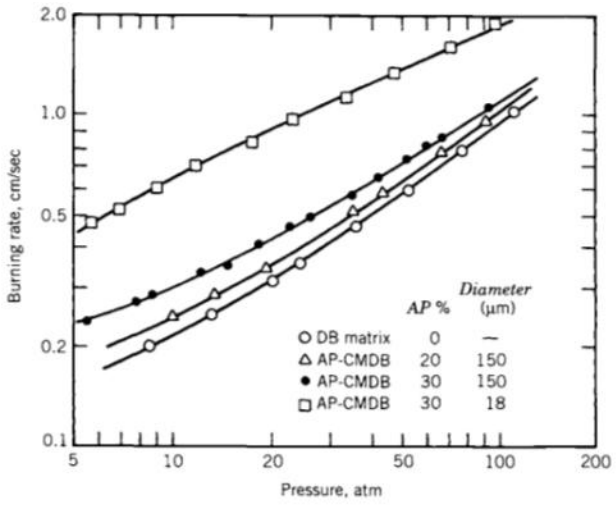


Figure 1: Example Plot of Burning Rate and Pressure Data with Curve-Fitting [2]

The initial temperature of a solid propellant usually affects its burning rate. As shown by Sutton [2], the temperature sensitivity of burning rate can be written as

$$\sigma_p = \left(\frac{\partial \ln r}{\partial T_b}\right)_{p_1} = \frac{1}{r}\left(\frac{\partial r}{\partial T_b}\right)_{p_1}, \quad \textbf{(2.2)}$$

where T_b is initial propellant temperature and p_1 is chamber pressure. The temperature sensitivity of pressure can be written as

$$\pi_K = \left(\frac{\partial \ln p_1}{\partial T_b}\right)_K = \frac{1}{p_1}\left(\frac{\partial p_1}{\partial T_b}\right)_K, \quad \textbf{(2.3)}$$

where *K* is the ratio of the burning surface area to the throat area. In the full characterization of a solid propellant, the values of these sensitivities should be experimentally determined.

A motor environment can influence the burning rate of solid propellant such that a formulation that has been fully characterized in a laboratory might burn differently in a motor. Erosive burning, or the augmentation of propellant burning rate due to cross-flow velocity, often occurs in a solid rocket motor [5]. Propellants can exhibit local burning rate augmentation due to heat transfer by the presence of inserted wires or by metal fuel agglomerates thrown to the burning surface by the spin of a vehicle [7]. Cracks in propellants may propagate under the influence of changing stress fields caused by pressure or structural deformations and cause the motor to burn in an unexpected manner [8]. Unstable combustion and pressure oscillations can also cause unexpected burning rates. In order to fully understand the burning rate of a solid propellant in its final implementation, it would be necessary to study all of these conditions.

Many techniques have been used to measure the burning rate of solid propellant. The most basic method is to burn the propellant in a motor configuration and examine the pressure trace [5]. More complicated methods involve direct measurement of the regression rate in addition to the measurement of pressure. Regression rate measurement

is performed with wired strand burners [9], ultrasonics [10], lasers and photographic optics [5], electrical capacitance [5], microwaves [5], translation motors with feedback control [11], real-time X-ray radioscopy [12], and Gamma rays [5]. If propellant is burned in a closed combustion chamber, a range of pressures and burning rates may be observed in a single test as the chamber pressurizes with combustion products so that a curve can be fitted [9]. A table, shown in Table 1, has been constructed by Cauty, et al. [13] that offers a high-level comparison between various methods.

Table 1: Summary of Burning Rate Measurement Techniques; Credit [13]

Table 2 Summary of advanced methods for regression-rate measurements

Technique	Purpose	Limitations	Accuracy	Cost	Maturity	Advantages, remarks
Microwave	Burning rate	Complex	Good	High	Medium	Direct measurement
X ray	Failure diagnostics	Metallic case possible depending on source	Medium	Very high	Medium	Large motor, visual data
	Burning rate	Metallic case needs special set-up and analysis	Fair	High	Medium	Laboratory setup
	High transient	Same remark	Good	High	High	Laboratory setup
PCG	Burning rate	Physical understanding	Good	Low	Medium	Flight integration Local measurement
	Ablation	Physical understanding	Fair	Low	Medium	Small size, local measurement
Ultrasonic	Burning rate for propellants	Coupling material	Very good	Low	Medium	Cost time, local measurement
	Ablation	Variation in speed of sound	Fair	Low	Medium	Flight integration
	Fluxmeter	Variation in speed of sound	Good	Low	Fair	
	De bonding		Good	Low	Low	Local measurement
Resistance	Burning rate for solid fuels	May require additional development for exotic fuels	Very good	Very low	High	Very low total cost, local measurements, small size, simple and small flight design
	Burning rate for solid propellants	Not yet mature, shorting problem	Fair	Very low	Low	Very low total cost, local measurements, small size, simple and small flight design
	Ablation	Not yet mature, limited experimental database	Good (with very limited data set)	Very low	Low	Very low total cost, local measurements, small size, simple and small flight design

2.2 Fundamentals of X-ray Imaging

X-ray imaging, also called real-time radiography (RTR) or radioscopy, is the production of images in real time from X-ray energy that has passed through a subject of interest. X-rays are electromagnetic wave energy similar to light, but they represent a frequency range of 3×10^{16} - 3×10^{19} Hz in the electromagnetic spectrum, which

corresponds to the wavelength range of 0.01-10 nm. An illustration of the electromagnetic spectrum is given in Figure 2.

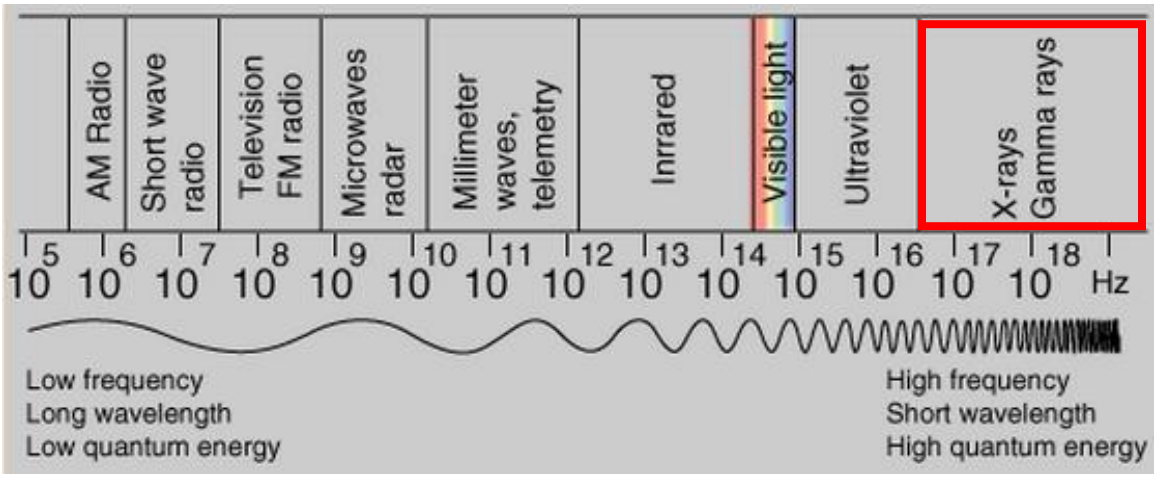


Figure 2: X-rays in the Electromagnetic Spectrum [14]

X-rays are attenuated by materials according to the attenuation properties and density of the materials. Every material has an attenuation coefficient that depends on the X-ray energy. X-rays with a large amount of energy will more easily pass through a given material than X-rays with a smaller amount of energy. The attenuation coefficient of a given material will be higher when the X-ray energy is low, and it will be lower when the X-ray energy is high. A relationship between initial and final X-ray energies and their attenuation by materials is given by [15]

$$I = I_0 \exp\left(-\sum_{i=1}^{n}(\mu_i \rho_i x_i)\right), \tag{2.4}$$

where I is the X-ray intensity at the detector, I_0 is the intensity at the source, μ is the attenuation coefficient of a material, ρ is the density of a material, x is the path-length through a material, and n is the total number of materials through which an X-ray passes and experiences attenuation. The equation is known as Beer's law or Lambert's law [15].

A variation of Beer's law is given by Frederick [16] to include the inverse-square law for a diverging beam. It can be written as

$$I = \frac{I_0}{R^2} \exp\left(-\sum_{i=1}^{n} (\mu_i \rho_i x_i) \right), \tag{2.5}$$

where R is the distance between the source and the detector, and the final intensity I has units of intensity per unit area of the detector.

The essential components of an X-ray imaging system are an X-ray source, X-ray detector, and the object of interest to be visualized. An illustration of a basic X-ray system is given in Figure 3. The X-ray detector might be one device that converts X-rays to a digital signal in an array of sensors that can be used to generate an image, or the detector might be comprised of two or more components. A flat-panel detector with X-ray sensitive semi-conductors is an example of a device that converts X-ray energy directly to a signal. An X-ray detector often consists of a scintillating screen and a camera. The scintillating screen converts X-ray energy into visible light, and the camera captures the visible light. An image intensifier is a commonly used device for converting X-ray energy to light and enhancing the light to be suitable for photography.

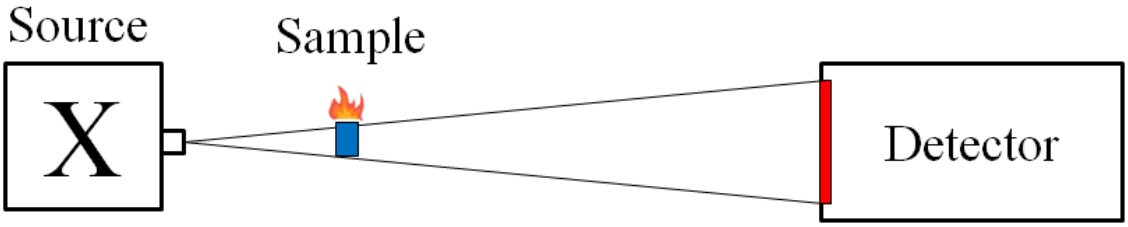


Figure 3: Basic Components of an X-ray Imaging System

2.2.1 Magnification

When the X-ray beam is a diverging beam as shown in Figure 3, the image of the sample of interest will be magnified according to the layout of the X-ray components.

The ratio of the distance between the source and the sample to the distance between the source and the detector will be equal to the ratio of the physical height of the sample to the projected height of the image of the sample at the detector. This is illustrated in Figure 4, where

$$\frac{B}{C} = \frac{H}{D}. \tag{2.6}$$

Thus the magnification factor is *C/B*.

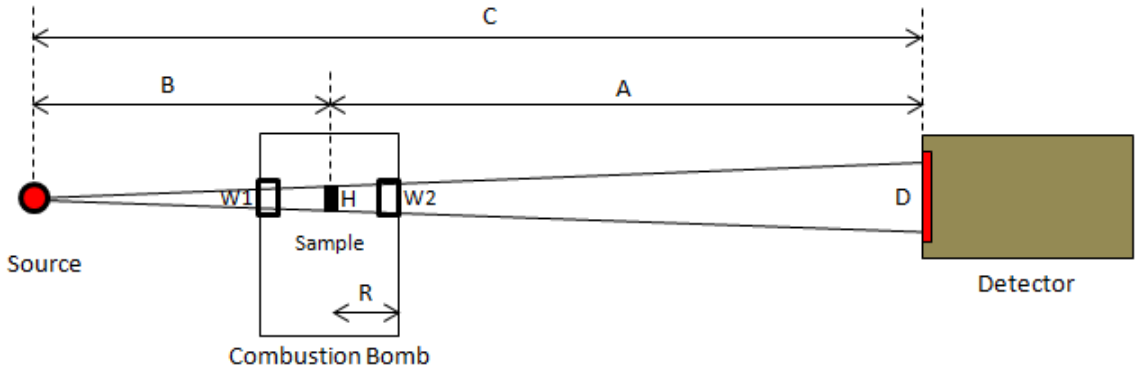


Figure 4: X-ray Image Magnification

2.2.2 **Penumbra**

Penumbra is a blurring effect caused by a finite focal spot size. An infinitesimally small focal spot would not produce penumbra. Since the X-rays are emitted from the focal spot in a diverging manner, multiple ray lines may be drawn from any point on the focal spot to one common point on the sample of interest. These multiple lines will have different angles, thereby producing multiple projections of that point at the detector. This causes blurring. Penumbra is illustrated in Figure 5, and it can be measured as the distance between the two widest points of the projection of a single point on the sample. The maximum penumbra would be *(D2-D1)/2).*

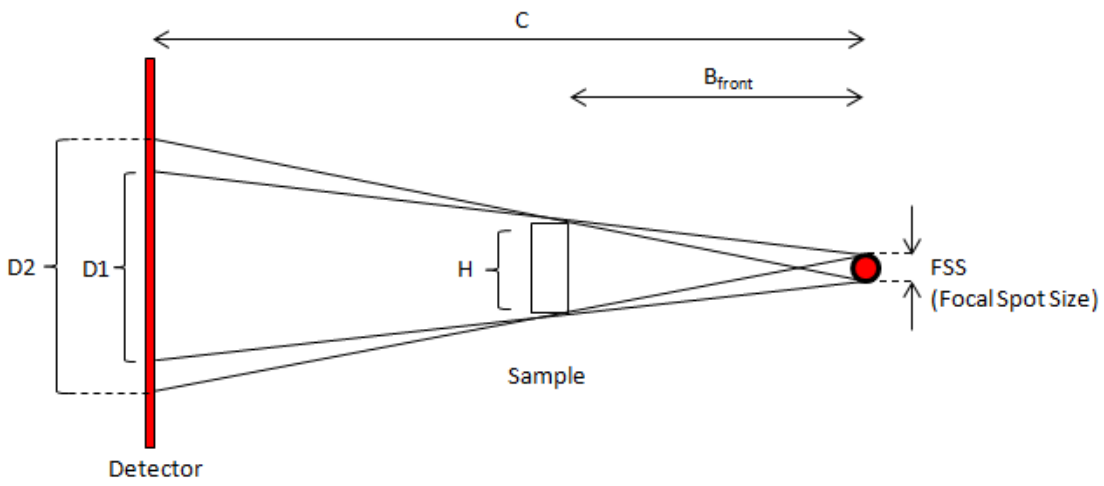


Figure 5: X-ray Image Penumbra

2.3 Historical Use of X-ray Imaging to Measure Solid Propellant Burning Rate and Other Regression Rates

2.3.1 Historical Measurements of Solid Propellant Burning Rate

X-ray imaging has been used to visibly track the burning surface of a solid propellant in the past. X-ray imaging can be used to track burning surfaces in many motor configurations or surface geometries. This is an advantage of the technique over many other techniques for regression rate measurement. It is very useful for applications with complex regression geometries such as nozzles or electric solid propellants. Using X-ray imaging, measurement of regression rate can be performed not only by tracking a surface but also by examining the time-rate-of-change of color intensity of the propellant from a side view. For example, as a propellant regresses from the inside out of a center-perforated cylinder, X-rays passing through the thickness of the propellant will be less and less attenuated. This time rate of change of intensity can reveal the burning rate. The use of X-ray imaging for solid propellant burning diagnostics has been studied by multiple groups of researchers all over the world beginning in the 1970's.

In 1978, a report was published by T. Godai in Japan on the use of pulsed radiography to measure the burning rate of a solid propellant [12]. The report was translated into English in 1987. Radiographs were obtained at a pulse rate of 8 fps. The measurement error was found to be 3%. The spatial resolution was described by a length measurement uncertainty of 0.3 mm. The work involved a 150 keV X-ray source for a 5-cm diameter solid propellant end-burning motor. A set of radiographs produced sequentially over a motor burn is shown in Figure 6.

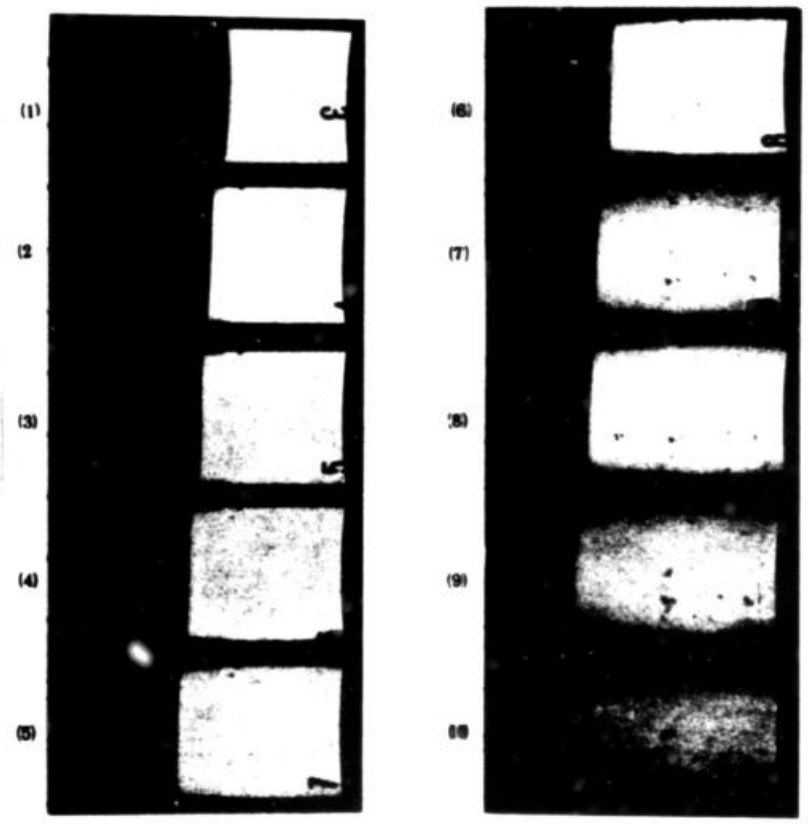


Figure 6: Solid propellant regression from pulsed radiography [12]

Kenneth Kuo, W. H. Hsieh, and S. J. Ritchie reported a study of Very High Burning Rate (VHBR) propellants using real-time radiography in 1991. Real-time x-ray radiography was used to study very high burning rate (VHBR) propellants. VHBR

propellants generally have burning rates ranging from 1 m/s to 500 m/s in closed vessel tests. Explosives generally burn above 2000 m/s. X-ray methods were used as a replacement for studying a pressure-time trace of a burn test in a closed bomb in order to find burn rate. There were two x-ray system configurations. One was called a medium-pressure (MP) test rig, which had a maximum pressure of 172 MPa. The other was called a high-pressure double-windowed (HPDW) test rig, which had a maximum pressure of 345 MPa.

An image intensifier was used to convert incident x-radiation to visible light which could be recorded with a high-speed camera at up to 12,000 pps (pictures per second). In the MP rig, a fiberglass or carbon-fiber tube was used to hold the propellant. The inner diameter of the tubes was 2.86 cm (1.125 in.) and an outer diameter of 4.13 or 4.76 cm (1.625 or 1.875 in.). An end-burning cylindrical propellant grain was X-ray recorded at 4000 fps. In the HPDW rig, a filament-wound fiberglass tube was used to hold the propellant. The inner diameter of the tube was 6.99 cm (2.75 in.) and the outer diameter was 12.06 cm (4.75 in.). The propellant grains tested were typically 3.38 cm (1.33 in.) long with a 0.63 cm (0.25 in.) perforation diameter. The propellant grain mass was 110 gm, so its loading density was 0.34 gm/cc (cc=cubic cm). A lead diaphragm and collimators were used to block scattering as shown in Figure 7 [17]. In 1996, Kenneth Kuo published more similar work on VHBR propellants. Another iteration of a similar radiography test schematic layout that he used is shown in Figure 8.

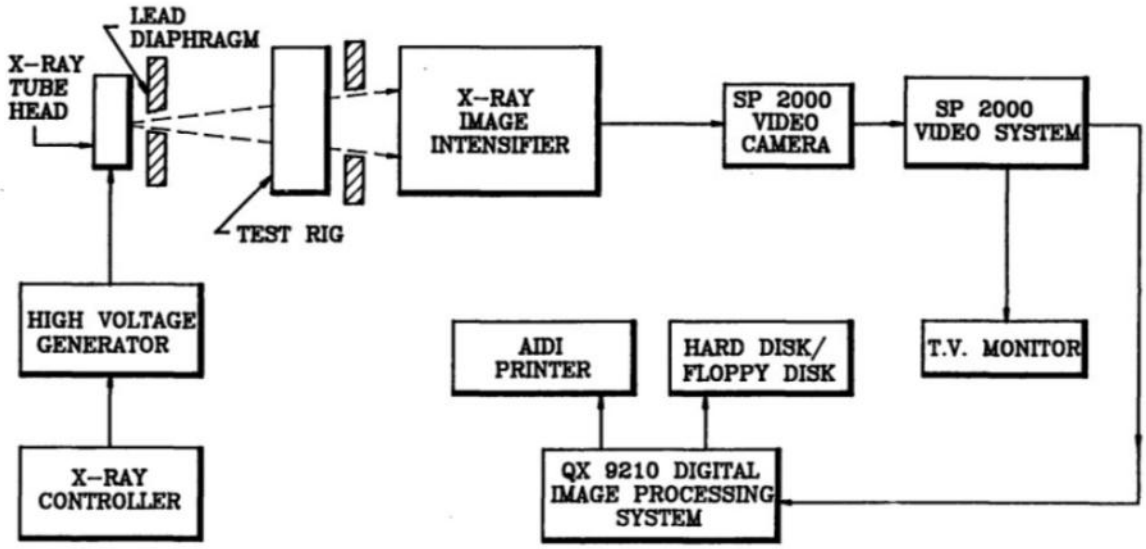


Figure 7: X-ray System Layout 1991 [17]

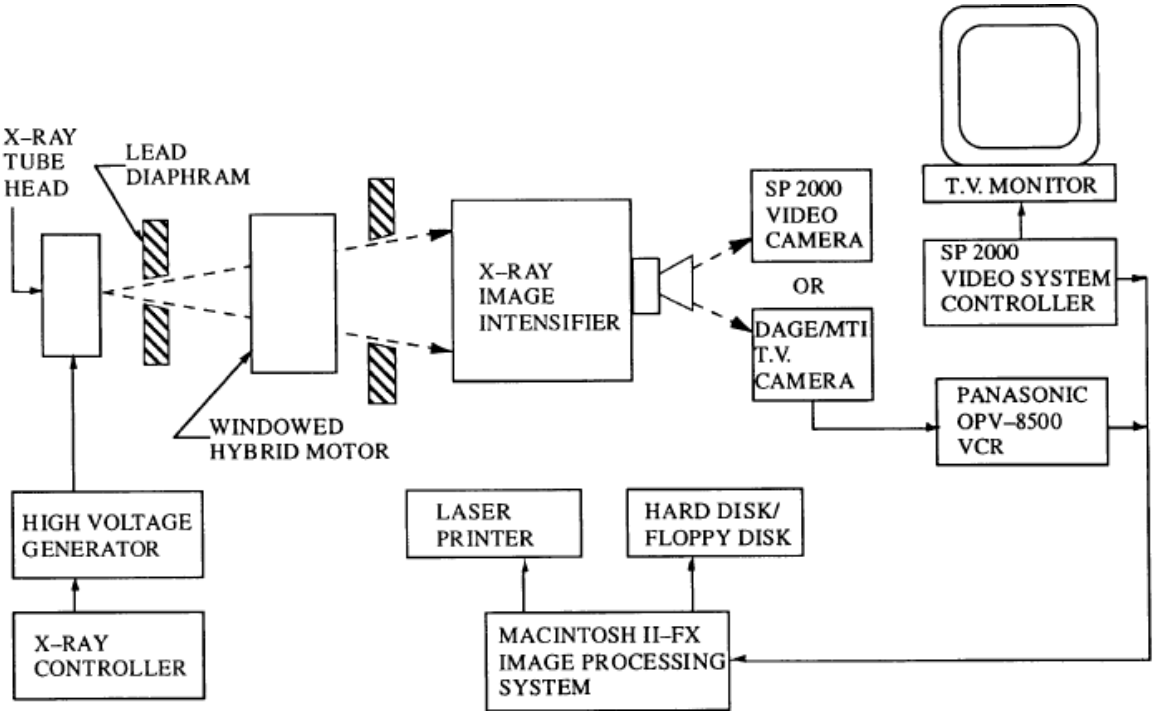


Figure 8: Radiography System Schematic 1996 [18]

In 1995, a document was produced by M. G. Anderson, J. F. Seely, and T. W. Hayes in which a dual-measurement system was described for measuring solid propellant burning rate [19]. Burning rate data was simultaneously obtained from two measurement systems, one being a real-time radiography system and the other being an ultrasound system. Burning rate data was captured for three subscale motor tests and for three full scale motor tests. For the subscale tests, a 6 MeV linear accelerator was used with a dose rate of 300 Rad/min. For the full scale tests, a 9 MeV source was used at a dose rate of 3000 Rad/min. The burn rate was successfully measured from both methods. The ultrasound equipment was found to be beneficial for accurately measuring the burning rate throughout the entirety of the motor burn, whereas the RTR method was incapable of capturing the full duration of the event because of the large size of the full scale rocket motors and the limited field of view of the RTR system. The RTR method was found to have the advantage of safety in that it was able to capture an entire failure event whereas some of the ultrasonic equipment was destroyed in the motor failure event.

In “Solid Rocket Propellant Behavior During Static Firing Test Using Real Time X-Ray Radioscopy”, a paper presented at a conference of the Advisory Group for Aerospace Research and Development (AGARD) in October of 1997, J. M. Tauzia and P. Lamarque describe the results of radioscopic testing of large rocket motors [20]. A solid propellant motor of 600-mm (23.6 in) diameter was burned under radioscopic imaging. The radioscopy was performed at 25 fps in one case, and 250 fps was achieved in another case at the cost of degraded image quality. The spatial resolution of the system in the case of 25 fps was found to be about 1 mm. The document presents radioscopic images with overlays of theoretical contour lines of the predicted burning contours. The

images are presented at multiple time steps so as to see the solid propellant surface regression. The images are given below in Figure 9.

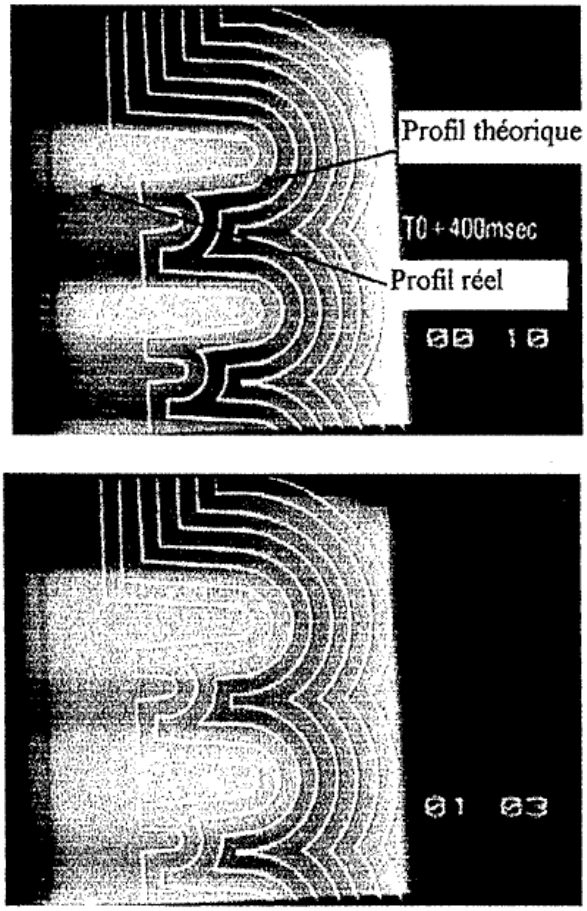


Figure 9: Radioscopy with theoretical burning regression contour lines [20]

Tauzia's document also contains observations about voids. The radiographs show that voids in the propellant, visible from the side as density gradients, appear to deform when under compression. Images prior to burning and during burning show the voids being squeezed under pressure. The voids are said to result in the formation of cracks when under this compression, which can cause chamber pressure spikes that were unexpected even if the voids were found in pre-test inspection, because the cracks can provide much more surface area than merely the voids. This observation is an excellent

example of the great value of real-time X-ray imaging, because this phenomenon could not be easily observed by other means. Images of void deformation are provided in Figure 10.

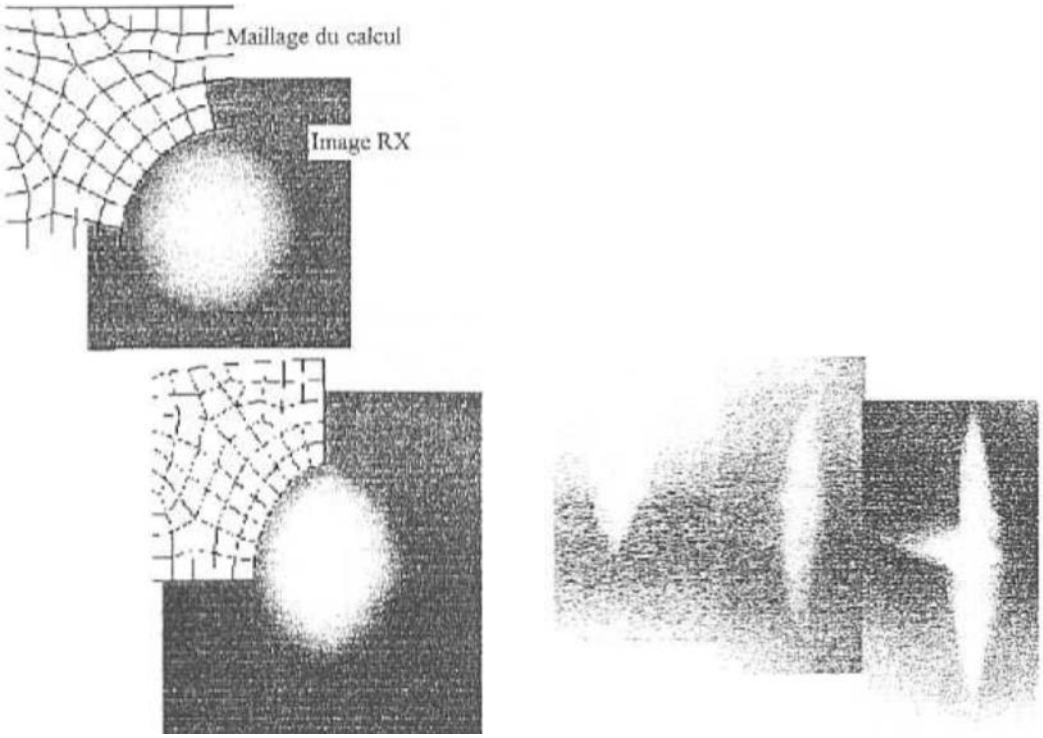


Figure 10: Small Void Deformation (Left), Void Deformation Leading to a Crack (Right) [20]

In reporting the X-ray capabilities at Los Alamos National Laboratory (LANL) in New Mexico in 1999, D. A. Fry writes, "In the 1980's Los Alamos pioneered the combination of high speed video cameras with fast output phosphor x-ray image intensifiers to enable radiographic imaging at speeds up to 2000 images/sec over extended time frames. The original capability used Spin Physics high speed video systems which recorded to special high speed video tape. This capability was used for a number of years in various applications until recently replaced by two electronic-memory high speed video systems." (p 117) These are the Kodak EktaPro 4540 and 1000HR systems. "The 4540 switches to partial frames or windows of less than the full 256x256

pixels in order to enable frame rates faster than 4500 per second. This goes down to 64x64 pixels when operating at 40,500 frames per second; however, since the partial frames take up less memory than the full the frames more partial frames than the 1024 full frames can be stored."

LANL has been a leader in the use of the most advanced radiography equipment for industrial inspection and imaging [21]. One of the most powerful machines at LANL can produce 9000 Rad/min/m at 20 MVp (peak MegaVolts). The lab has several other x-ray machines of varying energy, several of which are portable. The L&W Research Portac-6 is said to be the primary machine used for real-time radiography at the lab. It has an output of 20-425 R/min/m with a 2-mm focal spot size. The lab has three microfocus x-ray machines that each has a focal spot size of 5 microns, which can be used for high magnification. The lab has used amorphous silicon detector panels to successfully replicate the radiograph quality of film in certain circumstances. The lab also used CsI and NaI scintillator plates to create images that were recorded by cameras. Fast output phosphor image intensifiers are used to produce radiographs with very fast response time in conjunction with high speed cameras.

For an end burning propellant test, it is important to know whether the propellant burned only with a planar contour on its end or the burn also spread to the sides of the solid propellant along the motor case. If the propellant also burned down the sides, the pressure would be higher than is expected, which might cause error in burning rate measurement. Real-time X-ray radiography may be used to monitor the burning profile as described so as to ensure that a solid propellant test burned as intended. The same problem may occur for other propellant grain shapes such as a hollow bore. R. M.

Salizzoni, W. H. Hsieh, K. K. Kuo, and A. A. Juhasz report this problem in "Study of Combustion Behavior of Very High Burning Rate Using a Real-Time X-Ray Radiography System" [22]. The report includes several images of radiographs from the testing, which involved Very High Burning Rate (VBRH) propellants. The radiographs are shown below in Figure 11.

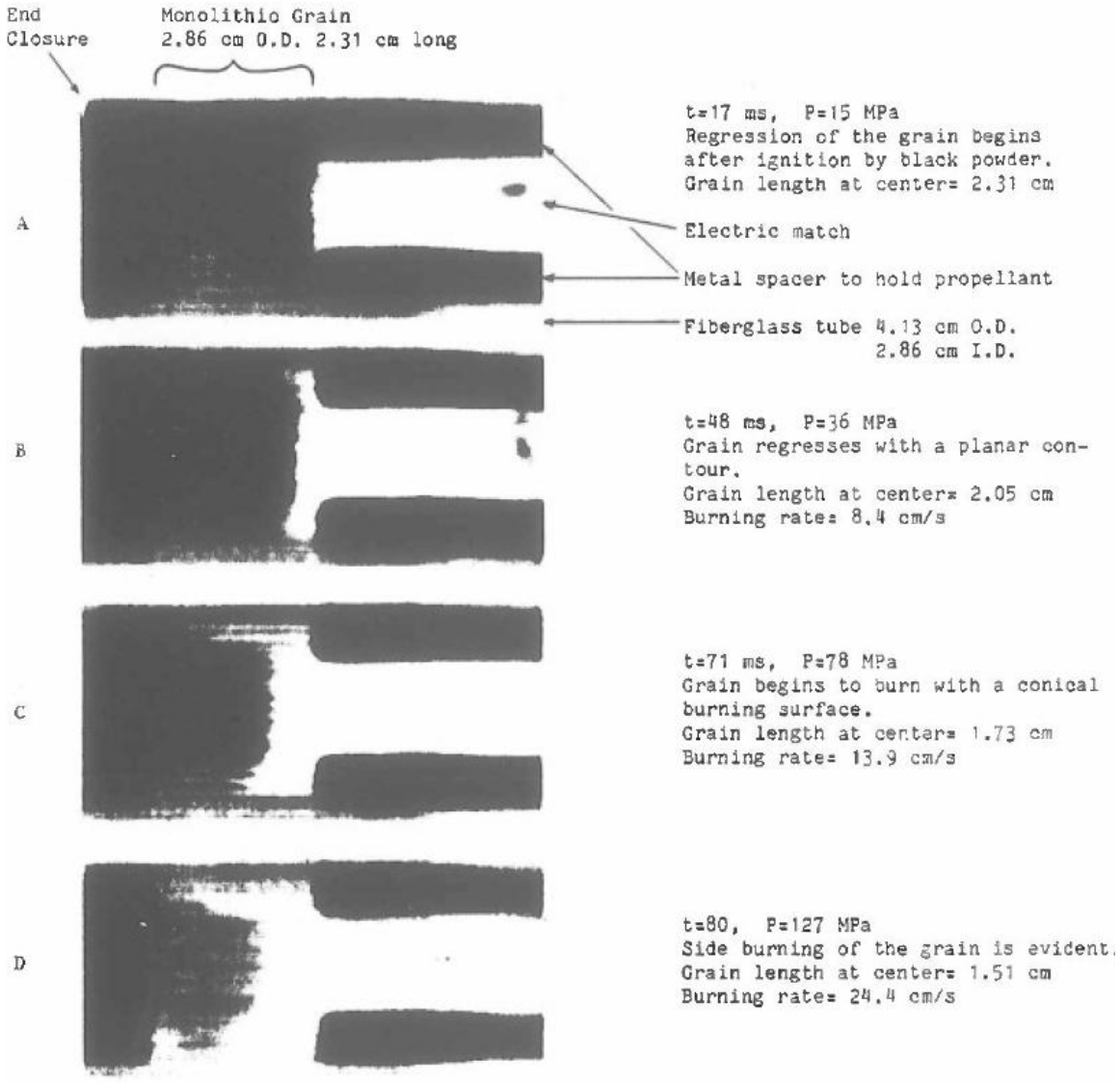


Figure 11: Burning Contours [22]

As can be seen in the radiographs, the propellant burning surface begins in the middle of the image and burns from right to left. However, the burning surface also progresses down the sides of the propellant grain along the motor case wall. If the grain was meant to be inhibited so as to burn only on the end, then this side regression could cause an artificially high burning rate measurement from a pressure trace, but the X-ray video reveals the side burning behavior.

M. Chiaverini reported his work using radioscopy to determine the regression rate of the solid portion of the hybrid motor [23]. His team found that their burn rate data from a radioscopy test matched the data from an ultrasonic pulse-echo test with an error of 4%. The regression-rate data is said to fall within a ±5% error band around a mathematical correlation prediction of regression rate. This might imply that the regression-rate data itself was contained within a relative ±5% error band, although the report does not specify whether the regression-rate data used to form the correlations were obtained with both X-rays and ultrasounds or with only one technique. The radioscopy system viewed a 19-cm section of a hybrid motor during a burn test. The spatial resolution was 140 μm. In order to allow penetration of the X-rays through the motor, a 19-mm thick Lexan outer window was used in combination with a 13-mm inner graphite window in the viewing ports of the rocket motor. The Lexan window provided strength, whereas the graphite window provided thermal protection. X-ray images were captured at 30 frames-per-second. A schematic of the motor setup is given in Figure 12.

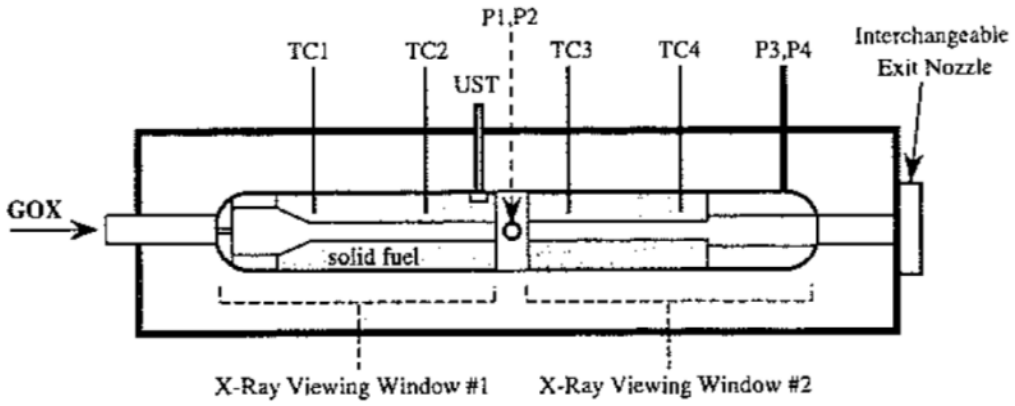


Figure 12: Schematic of Hybrid Motor Test Apparatus [23]

2.3.2 Experimentation related to X-ray Imaging Surface Tracking

In 1966, Osborn reported the use of gamma rays to measure burning rate [11]. Instead of using gamma rays to produce images, the gamma ray dose rate was measured as a burning sample of propellant was continually translated forward across the beam of gamma rays. The goal was to translate the propellant at the same rate as its burning regression. The gamma ray dose signal was amplified and used as a feedback control on the motor that translated the propellant sample, such that the motor kept the burning surface of the propellant sample in a constant location. As this was accomplished, the speed of the motor shaft was measured, and that speed represented the burning rate of the propellant. The largest deviation of any datum point from the least-squares curve fit of Robert's law on the data was less than 2.5%.

In Anderson's description [24] of the radioscopy testing of Titan IV Lightweight Analog Motors (LAM) in 1992, he reports that the propellant surface of the solid propellant was tracked with radioscopy at 30 fps so as to measure expansion from case pressurization. The uncertainty associated with the distance measurements was 0.017 inches. This was for a test motor case made of composite materials and a 6-MeV pulsed

real-time radioscopy X-ray source. The propellant grain dimensions were 10 inches in diameter and 17 inches in length. A schematic of the region of the test apparatus that was penetrated with X-rays is shown in Figure 13. A sample set of the radiographs from the primary data view is shown in Figure 14.

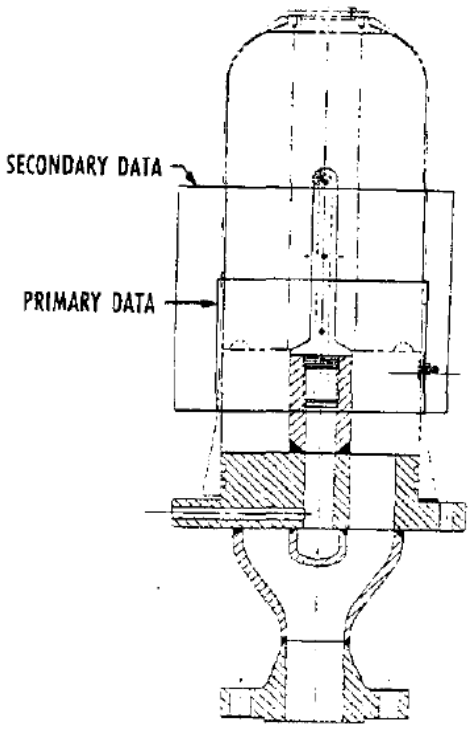


Figure 13: Apparatus Schematic for X-ray radioscopy [24]

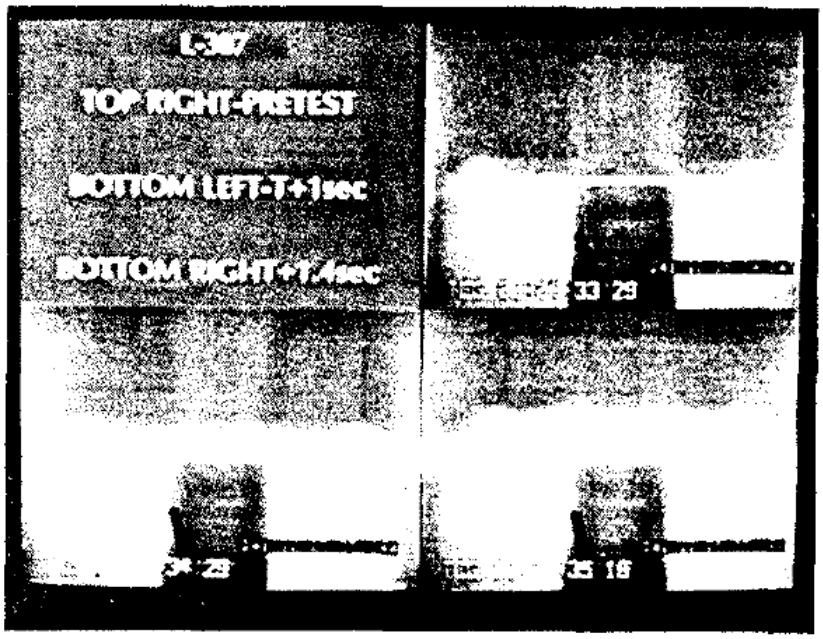


Figure 14: Sample radiographs of solid propellant regression from primary data view [24]

Pressley and Glick [25] reported positional accuracy of a radiograph computer simulation of 0.1 mm. This accuracy was said to be beyond what was achievable for their proportions in reality with the system parameters given in the computer because scattering and screens would cause additional blurring not accounted for by the computer radiography simulation program.

Bisola Olaniyi [26] reported a research project in 2010 in which an experiment was conducted to simulate the measurement of solid propellant burning rate using a continuous X-ray source. The test included a translating plexiglass cylinder that was penetrated by a radiography system at AEDC Arnold Air Force Base in Tennessee. The experiment was designed to measure the accuracy of the radiography system in tracking a surface in hope of using the system to measure the burning surface of a solid rocket propellant motor. The testing at AEDC involved a continuous X-ray source Gemini-2000

at 320kV. The screen was zinc cadmium sulphate (MCI optronix) and fluoresced with X-ray with a 1/16” aluminum plate. A mirror was placed at an angle to the screen so the camera could record the image.

The outside dimensions of the Plexiglas motor cylinder were 7.75” by 2.5”, and the cylinder had several stepped internal diameters, and it had a curved inner region as shown in Figure 15(a). The motor cylinder was translated across the x-ray view in order to simulate a burn rate test while also validating the system’s ability to accurately measure the size variations in the cross section of the cylinder. A sample plot of the incident X-ray intensity is provided in Figure 15(b). In the plot, the intensity can be seen to be at its peak value outside the case on the left and right edges of the plot, and then the intensity dips to a low value on both sides where the walls of the case are roughly in line for part of the case curvature. The intensity arcs upward towards the middle of the plot as the case walls become increasingly perpendicular to the view point, meaning that the effective wall thickness attenuation decreases towards the center of the plot.

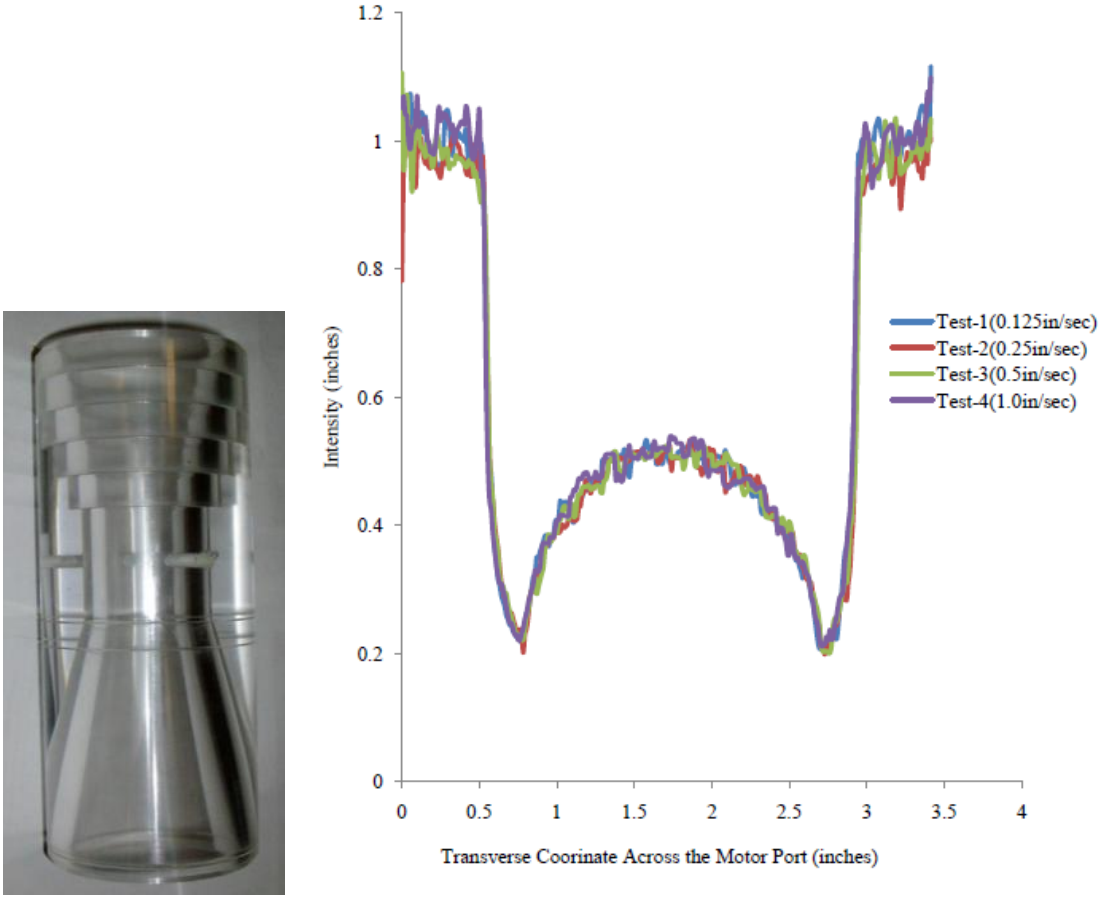


Figure 15: (a) Left: Plexiglas Motor Cylinder (b) Right: Sample plot of intensity data across one lateral plane of the cylinder. [26]

The moving motor cylinder surface was tracked with an error of 1.79% error for motor case wall thickness of less than 0.75 inches. The test results showed a deviation of 6.67% for a thickness of 0.75 inches at one point on the motor case cylinder. Olaniyi also reported that H.M. Pressley, Jr., is said to attain 3% error in burn rate measurement using radioscopy.

In 2012, Andrew C. Cortopassi of the Pennsylvania State University published his dissertation on the study of nozzle erosion using real-time radioscopy [27]. He used an alignment tool to calibrate the X-ray radioscopy system. The tool had holes on two sides that show whether the object is centered in front of the X-ray beam or not. A radiograph

of the alignment tool is shown in Figure 16. Radiographs of nozzle erosion from his work are shown in Figure 17.

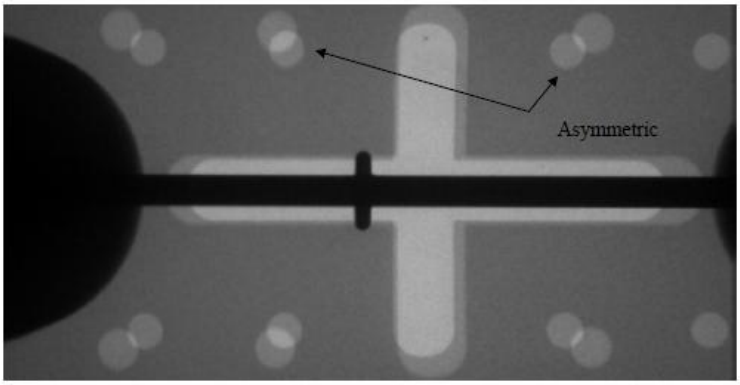


Figure 16: Radiograph of alignment tool showing misalignment [27]

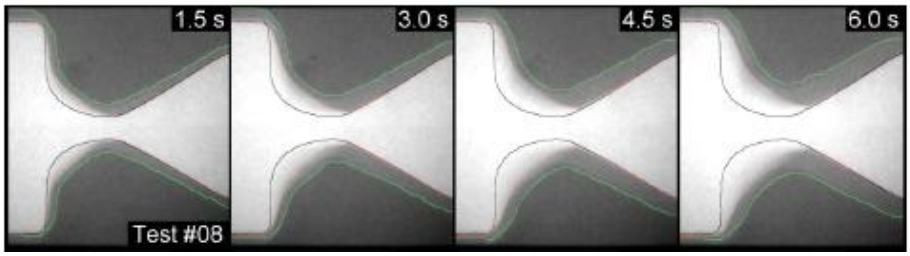


Figure 17: Successive radiographs showing nozzle erosion [27]

Also in 2012, V. G. Efimov [28] reported his geometric analysis for using x-ray radiography to measure propellant burning surfaces in order to measure burn rate. No burn test was reported, but the calculations that would be used to estimate burning rate and even for reconstruction are described in detail. An x-ray machine was used with a 250-mm CsJ(TI) monocrystal screen and CCD array. Relative measurement error was 1%.

2.4 Summary Table of Burning Rate Measurements With X-ray Imaging

The following table contains a summary of the reports of several parameters commonly involved with experiments and testing. Does rate or intensity, X-ray energy,

temporal (time) resolution, spatial resolution, and burning rate percent error are tabulated from past experiments which used X-ray radiography to measure propellant burning rate of solid rocket motors. Certain experiment parameters were not reported in some of the historical documentation of the tests and experiments.

Table 2: Historical X-ray System Parameters

Reference	Application	X-ray tube voltage	Temporal Resolution	Spatial Resolution	Burning Rate % Error or StDev
Burning Rate Determination With X-ray Imaging					
Godai [12]	Small: 5-cm diameter motor	150 kV	8 fps	0.3 mm	3%
Anderson [19]	Large: subscale and full scale	6 MV, 9 MV	-	-	-
Chiaverini [23]	Small: *RTR data matched ultrasonic data by 4%.	-	30 fps	140 μm	±4%*
Tauzia [20]	Large: 600 mm diameter motor	8000	25 fps	1 mm	-
Kuo 1991 [17]	Small: VHBR propellants	-	4000 fps	-	-
Kuo 1996 [18]	Small: HTPB	-	30 fps	-	±5%
Pressley, Olaniyi [26] pg 13	Solid motor	-	-	-	3%
Fry [21]	Small	-	40500 fps	3.9 lp/mm	-
Simulations and Other Applications					
Anderson [24]	Small: propellant pressurization deformation	6 MV pulsed	30 fps	0.017 in	-
Olaniyi [26]	Small: tracking motor cylinder	-	-	1.79%	-
Cortopassi [27]	Nozzle erosion (slower rates)	320	75.5; 15.15 with avging	3.34%	SD 0.21
Frederick [16]	Simulation	-	-	106 px/in, 10%	-
Pressley [25]	Simulation	-	-	0.1 mm	-
Efimov [29]	Radiography Demonstration	-	-	1% error	-

CHAPTER 3

EXPERIMENTAL AND ANALYTICAL APPROACH

3.1 Experimental Approach

3.1.1 Overview of Experimental Design

In order to characterize the burning rate of the electric solid propellant, the broad goal of the experiments was to obtain values of *a* and *n* as described in St. Robert's Law, which is given in Equation (2.1). The propellant samples were burned inside of a combustion bomb, and a real-time X-ray radioscopy system was used to capture video of the samples as they burned. The burning rate of the samples was measured by examination of the digital X-ray videos with a MATLAB code. The MATLAB code also performed a Monte Carlo analysis of the burning rates to yield uncertainty bounds.

While the work of characterizing the ESP is still being carried out, the X-ray technique has been studied and used to measure the baseline burning rate of some ESP samples. Multiple propellant sample configurations were originally designed, and the test apparatus were designed to operate at multiple pressures, but the X-ray analysis has focused on two tests conducted with one configuration of propellant samples with one target pressure. The target pressure was 400 psi, which was to be the lower bound of pressures at which the propellant samples would be tested. The target voltage applied to the ESP samples was 200 V for both tests.

3.1.2 **Propellant Samples**

The electric solid propellant samples were constructed in a joint effort between UAH and Digital Solid State Propulsion (DSSP). The sample cases were built at UAH and shipped to DSSP where they were filled with propellant. The propellant samples were then shipped back to UAH for experimentation. The propellant samples that were studied in this work were called Ignition Configuration samples.

The burning rates of the ESP samples were determined by a MATLAB code that tracked the propellant burning surface in the X-ray videos. The ignition configuration was designed with the intent that the propellant would be allowed to burn on its own with no further supply of electrical power after initial ignition. The propellant was ignited by flowing current through the propellant, but, once ignited, the propellant would quickly burn away from the source of electrical power on the propellant. The source of positive voltage was a small, bare tip of a wire inserted into the top surface of the propellant as shown in Figure 19. The aluminum tube served as the negative electrode for the brief flow of electrical current for ignition.

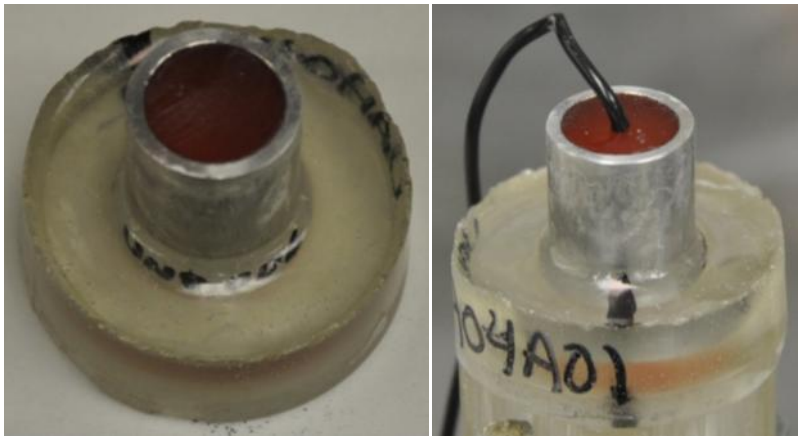


Figure 18: Ignition Configuration Propellant Sample Example Images

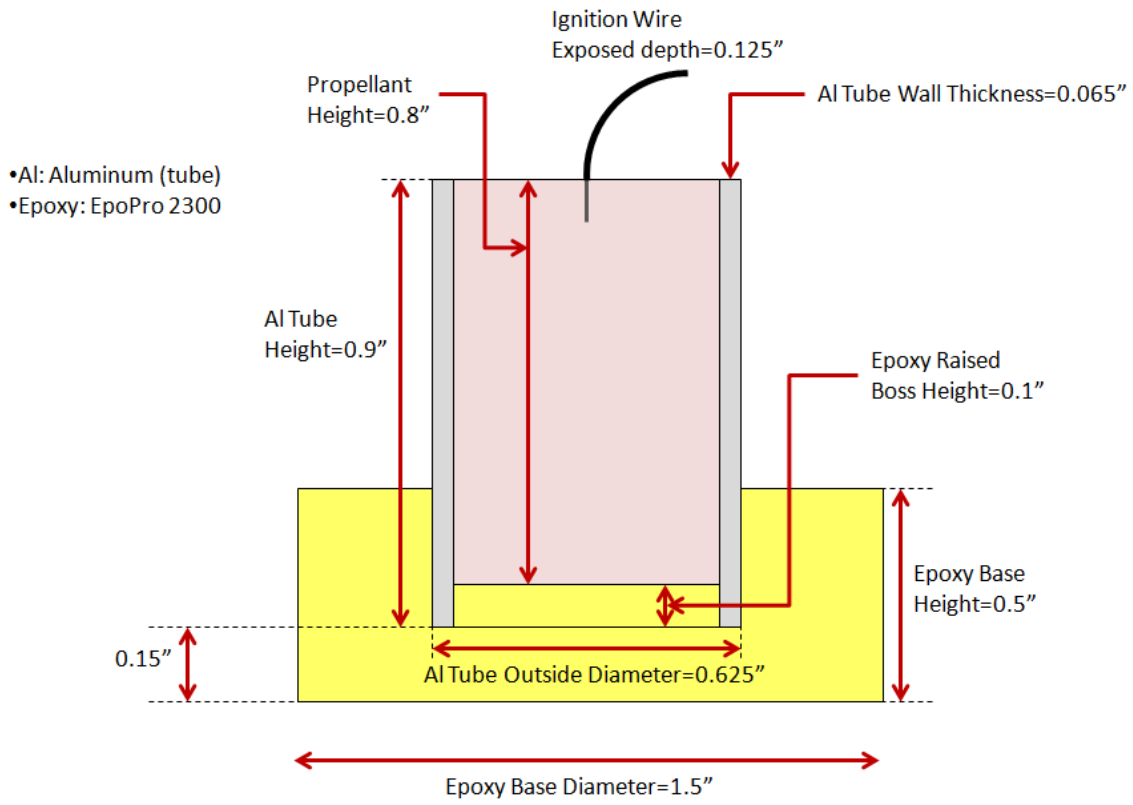


Figure 19: Ignition Configuration Propellant Sample Schematic

In order to measure the burning rate of the propellant, each sample was burned individually in a windowed combustion bomb, and X-rays were used to create radioscopic video of the propellant sample as it burned. In each test, pressure was measured, and the power supplied to the sample was also measured. Tests were conducted in a constant-pressure environment in order to isolate the effects of electrical augmentation of burning rate. Tests were planned at multiple target pressures, but the tests reported herein were all conducted at 400 psi.

Before a sample was mounted in the combustion bomb, it was marked with a permanent black ink marker to indicate the rotational orientation. The black line drawn

on the epoxy base of the sample was made to point in the direction of the X-ray detector when the sample was fixed to the phenolic-epoxy support structure.

3.1.3 **Combustion Bomb**

A windowed combustion bomb composed of copper and aluminum was used to contain propellant samples while they burned. The combustion bomb is shown in Figure 20. Its hardware was originally designed as a liquid propellant rocket engine chamber, and it was used in the past by other students at UAH for liquid propellant rocket engine experiments [30]. For the ESP experiments, it was assembled with only one flow port used for all flow operations including pre-pressurization, expansion of combustion products, and venting. The pressure inside of the bomb was measured by two pressure transducers and an analog gauge mounted on the tubing at the flow port as shown in the picture. Maximum pressure ratings for various parts of the combustion bomb were estimated in calculations performed by Ryan Saffell. His calculations can be found in Appendix C of Standard Operating Procedure PRC-SOP-HPL-010-0-B.

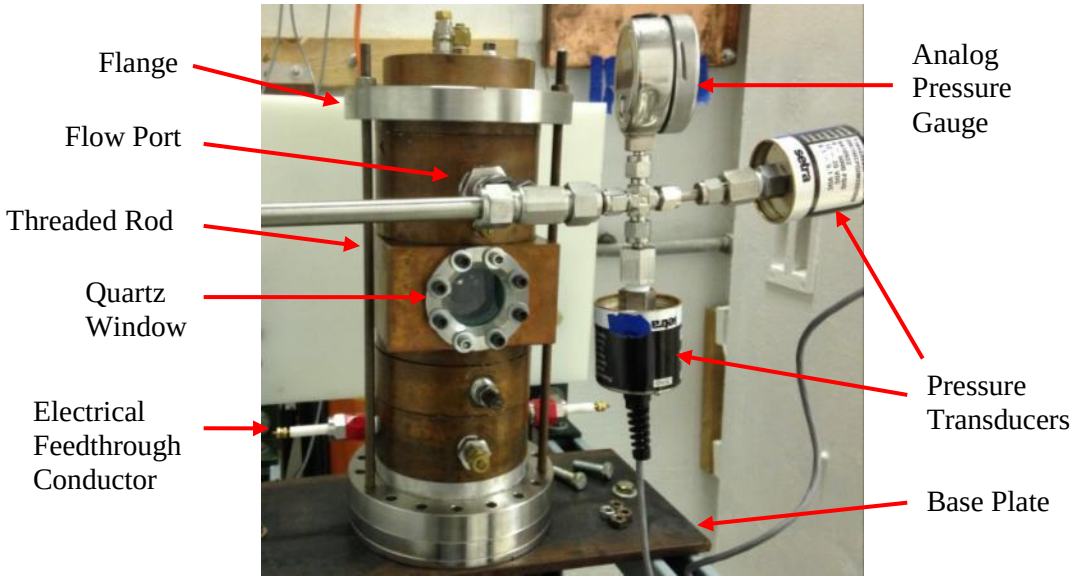


Figure 20: Combustion Bomb

The combustion bomb was assembled from seven chamber sections. The sections were sealed against each other with size-238 rubber O-rings. The sections were compressed together between flanges and a steel base plate with nuts and threaded rods. The bomb was placed upon two stacked flanges which held the bomb off of the base plate, and the rods passed through holes in the base plate that were aligned with the flanges. Nuts were fastened above the top flange to compress the bomb from the top, and the other set of nuts were fastened onto the rods underneath the base plate to compress the combustion bomb sections together from the bottom. The inside diameter of the combustion bomb was 2.125 inches. The outside diameter of the bomb was 5 inches. The total height inside of the combustion bomb was 10.79 inches. The internal volume of the bomb while empty was 38.27 in^3.

In order to hold the propellant samples in a fixed position inside of the combustion bomb, a support structure was built using epoxy and phenolic. A phenolic disk was made to be fastened with nuts onto a set of four threaded rods. The rods were threaded into the aluminum bottom end cap of the combustion bomb and secured with nuts locked onto the inside surface of the end cap. Since the diameter of the propellant samples was wider than the rod spacing, an epoxy riser was placed on top of the phenolic disk to lift the sample above the rods. The epoxy riser was cast as a cylinder and cut into a cross shape in order to make space for the rods and nuts. A five-minute epoxy, Elmer's Super Fast Epoxy Cement, was used to permanently fasten the riser to the top of the phenolic disk. A hole was drilled down the center of the phenolic and epoxy pieces to allow a space through which to connect to the bottom of the center electrode rod of the propellant sample. When a propellant sample was placed on top of the support structure

for a burn test, a small amount of five-minute epoxy was used between the sample base and the riser to secure the sample in position. The phenolic-epoxy structure is shown in Figure 21.

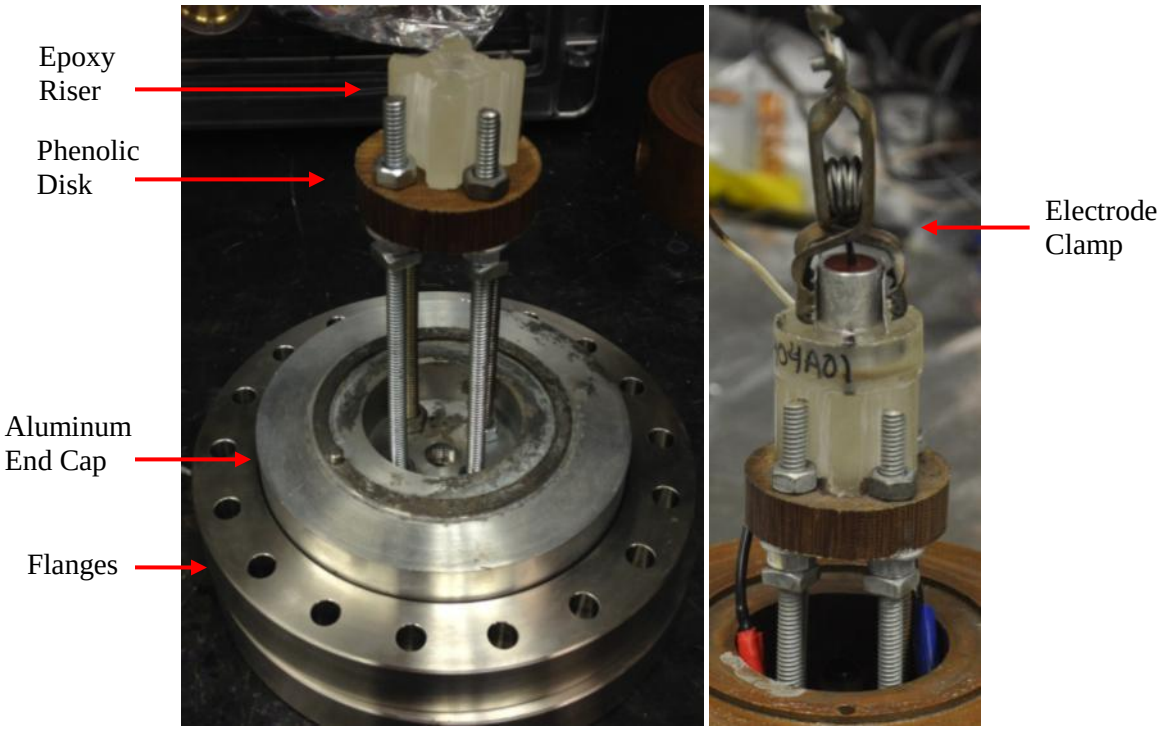


Figure 21: Phenolic-Epoxy Sample Support Structure Inside of the Combustion Bomb (left), Example of Mounted Sample (right)

The positive power wire inserted into the top surface of the propellant sample was connected to the positive electrical pass-through fitting inside of the combustion bomb. A clamp was used to connect the negative power wire to the propellant sample tube as shown in Figure 21. The negative power wire was connected to the negative electrical pass-through fitting inside of the combustion bomb.

Electrical power for the propellant samples was passed into the combustion bomb through insulated fittings in one of the copper chamber sections. A general picture of the electrical pass-through fitting product series is shown in Figure 22, but the exact model used had smaller electrode rods and corresponded to the Type "A" drawing in the figure.

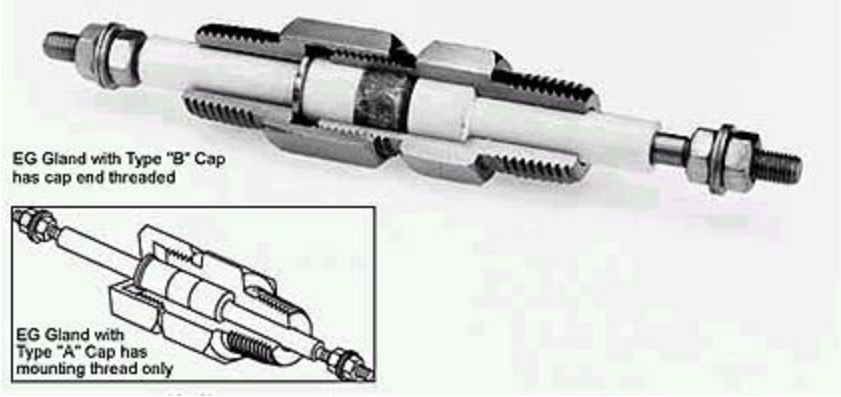


Figure 22: Electrical Pass-Through Fitting for Combustion Bomb [31]

3.1.4 **Pressure System**

When solid propellant samples were burned in the combustion bomb, high pressure nitrogen was used to pre-pressurize the combustion bomb. Nitrogen was also used to purge combustion products out of the combustion bomb after a burn test. In order to control the flows of nitrogen and combustion products, high pressure stainless steel tubing and pneumatic valves were installed. A surge tank was also used to create a constant-pressure environment in the combustion bomb. More equipment details are provided in APPENDIX A.

For conventional solid propellants that burn with pressure dependence according to St. Robert's Law, a constant pressure environment provides an accurate measurement of the burning rate at a given pressure, but several tests at various pressures are required in order to determine the dependence of burning rate upon pressure. With conventional propellants, a closed bomb experiment with constant volume can be useful because it provides a range of instantaneous regression rates correlated to a range of pressures. This reduces the number of tests necessary to find the coefficients a and n. However, the constant-pressure environment is beneficial for studying ESP because it allows the effects of electrical augmentation of burning rate to be observed distinctly compared to an

unpowered configuration with no electrical effects, because there is no pressure rise contributing to the augmentation of burning rate. A schematic of the constant pressure system is shown in Figure 23. A picture of the constant-pressure system is given in Figure 24.

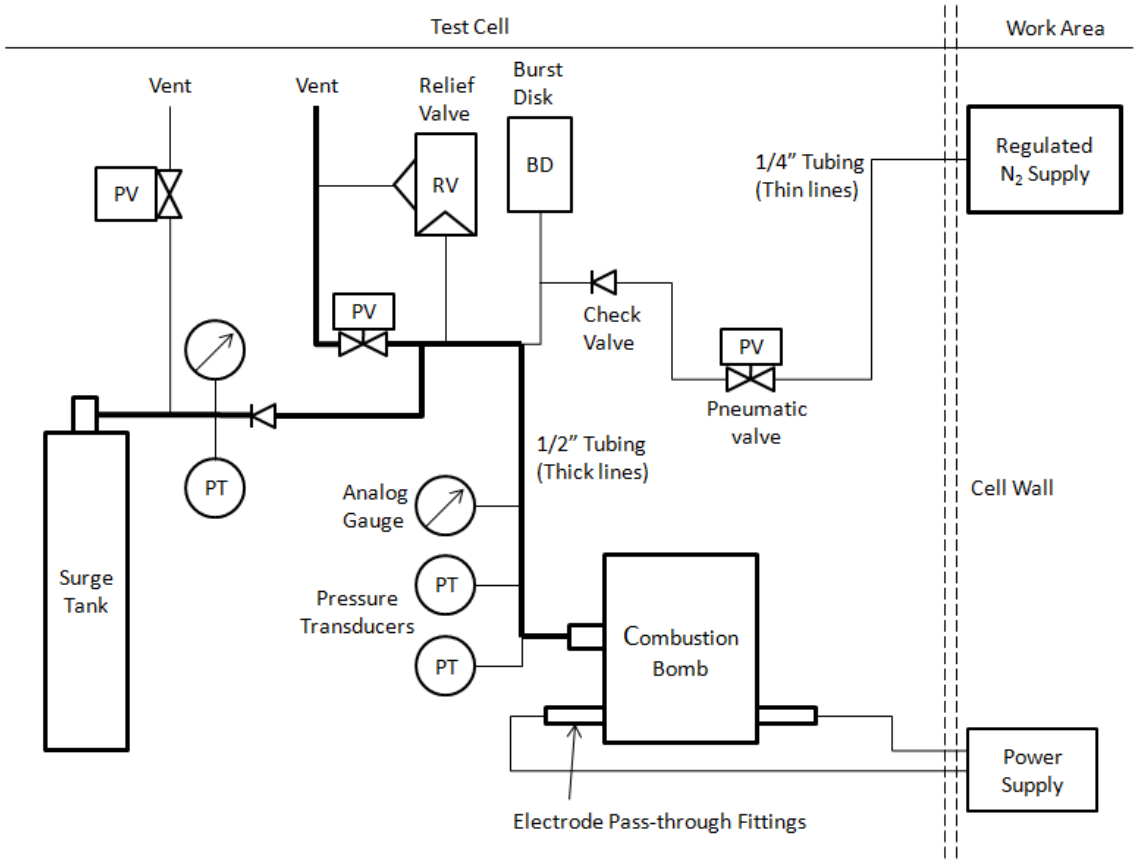


Figure 23: Schematic of High Pressure System in X-ray Test Cell

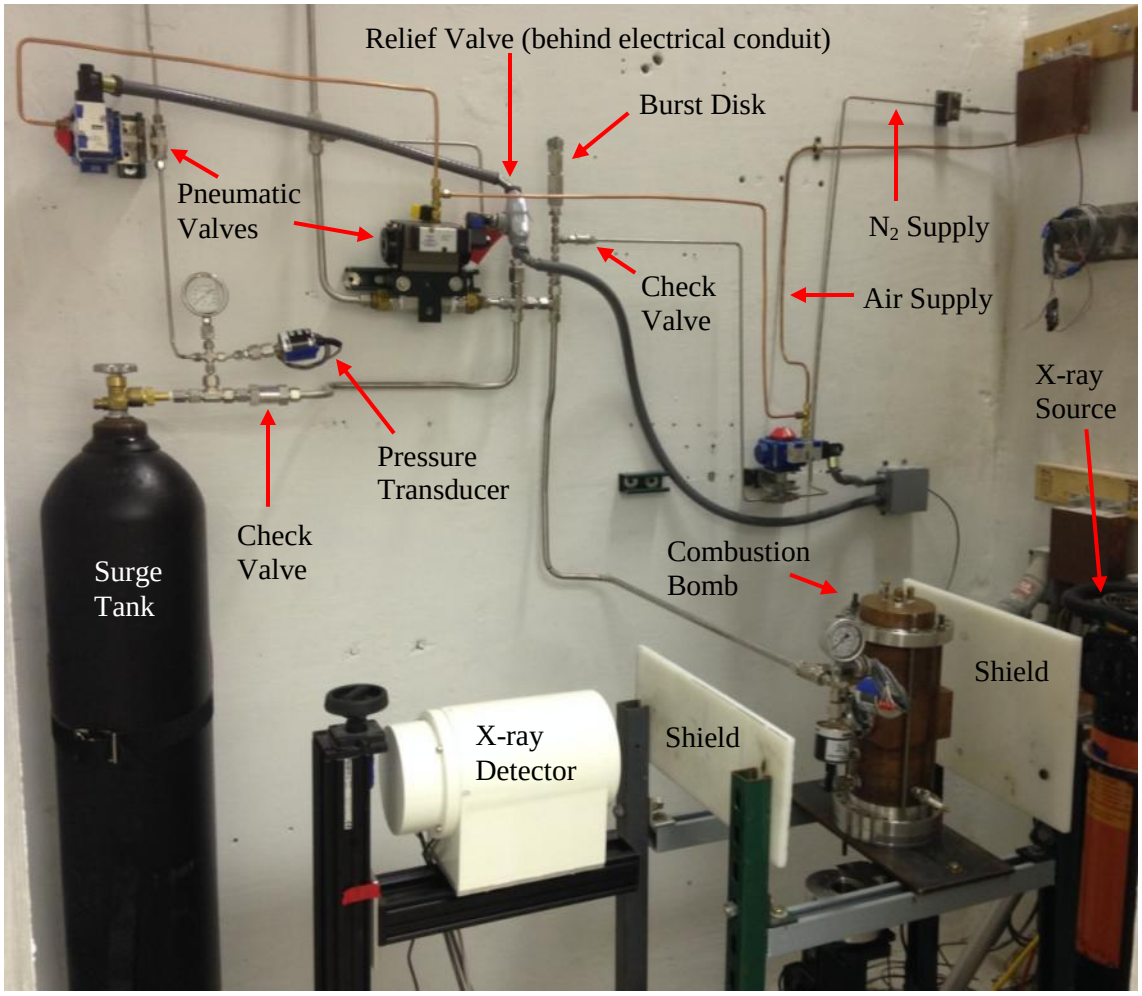


Figure 24: High Pressure System in X-ray Test Cell

The pressure system was equipped with two components designed to relieve pressure in the unlikely event of an over-pressurization. A relief valve allowed pressure to be vented from the system if the pressure ever reached the cracking pressure of the relief valve. The cracking pressure was set manually by personnel using a self-venting regulator. The valve was set to crack at 550 psi. The other pressure relief component was a burst disk, also known as a rupture diaphragm. The nominal rupture pressure of the burst disk was 900 psi. Plastic shields were mounted in between the bomb windows and the X-ray equipment. The shield thickness was one-half inch.

A programmable logic controller (PLC) was used to remotely toggle pneumatic valves to control the flow of gases. The PLC is pictured in Figure 25. The PLC was also used to send the electrical power to the propellant samples. It had several physical and virtual lockout switches to prevent unintentional ignition of the propellant sample.

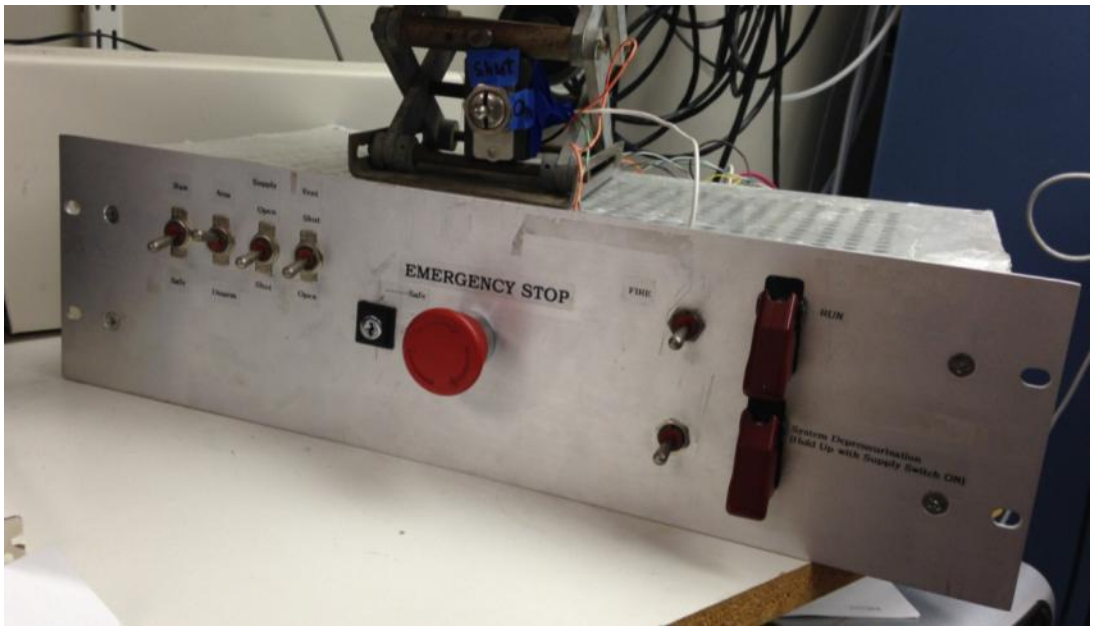


Figure 25: Programmable Logic Controller

3.1.5 X-ray Equipment

X-ray video was recorded while each propellant sample burned. The X-ray source that produced the X-rays was the ERESCO 200 MF4-R made by GE. It is shown in Figure 26. It was capable of producing X-rays at a tube voltage of 10-200 kV. It was determined that 100 kV was the tube voltage with 3.0 mA of tube current at which the contrast appeared best for the propellant samples within the combustion bomb, so 100 kV and 3.0 mA were used for all tests. To produce X-ray images, an image intensifier and digital camera were integrated to act as an X-ray detector, shown in Figure 27. The image intensifier was the Toshiba E5877J-P1. The digital camera was the Kappa HiRes3-XR. The image intensifier contained a scintillating screen that produced light in response to activation by

X-rays. The digital camera captured the light produced by the scintillating screen inside of the detector. For all tests, the gain of the digital camera in the X-ray detector was set to 300. The X-ray equipment was spaced such that the X-ray images had a magnification factor of about two. The distances between the components are listed in Table 3.

Figure 26: X-ray Source (left) X-ray controller (right)

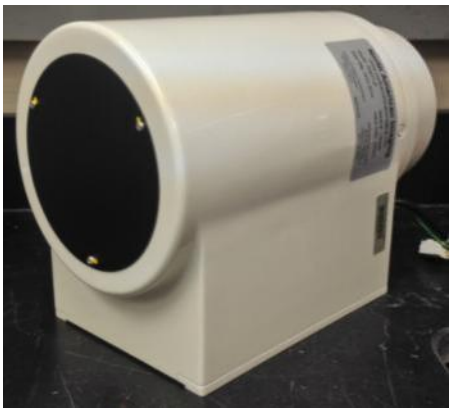


Figure 27: X-ray Detector

Table 3: Spacing of X-ray System Components

Description	Distance (inches)
Detector to shield	6.125
Detector shield to face of bomb	10.625
Windowed section width	5
Source shield to face of bomb	4.25
Source to shield	13.875
Shield thickness	0.5

3.2 Analytical Approach

3.2.1 Predicted X-ray Images and Strategy for Surface Tracking

The plan for performing the burning rate calculations was to use a MATLAB code to track the burning surface in the X-ray video frames. The MATLAB code would read the transmitted intensity gray-scale values from the X-ray images and compare them for consecutive images. Olaniyi's X-ray intensity plot in Figure 15(b) is an example of the sort of intensity plot that would be used in the burning surface tracking code. The plot shows low points where the X-ray path length through the Plexiglas wall was the thickest at the edge of the inner bore, and there is a local maximum at the center of the inner bore where the wall path length through the wall was minimal. If the cylinder was burning on the inside surface as a center-perforated solid rocket motor, then one could expect that the location of those low points on the intensity plot would move horizontally away from each other as the inside surface of the bore regressed radially toward the outside of the case. One would also expect that the peak at the center of the plot would move higher and higher as the center bore burned because the wall thickness would be decreasing, thereby decreasing the amount of attenuation of the X-rays. These changes could be tracked by a computer.

In conjunction with the analysis of X-ray videos from the ESP burn testing at UAH, a MATLAB simulation of an X-ray image of an end-burning motor or propellant sample with a bowl-shaped burning surface was created as shown in Figure 28. The mathematics in the code were heavily borrowed from a 3-D modeling tool for predicting X-ray radiographs created by Dr. Robert Frederick in 1991 [8], [16]. The MATLAB code used 2-D analytic, geometric formulas to calculate X-ray path-lengths through the various parts of the propellant sample. It took inputs of energy level, densities and X-ray absorption coefficients for various materials, and distances and dimensions of the sample. The code is provided in APPENDIX E along with a table of input parameters.

The code first simulated a solid propellant that was at about half of its burning time within an aluminum tube case. The image had the darkest gray value where the X-rays were most attenuated, which was through the middle of the propellant-filled region, and the lightest colors were outside of the tube and within the region where the propellant had burned away inside of the aluminum tube.

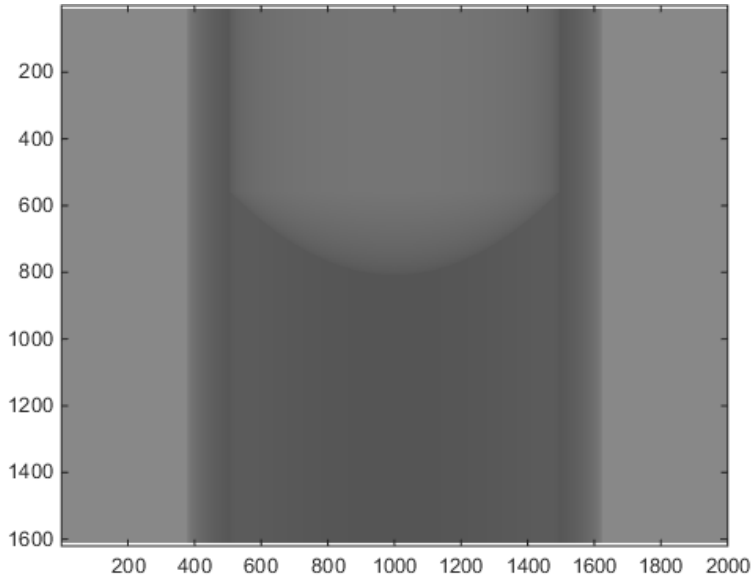


Figure 28: Computer-Generated X-ray Image Simulation of a Propellant Sample

The intensity along the center column of pixels in the simulation image was plotted as shown in Figure 29. The various regions of the propellant can be clearly identified on the intensity profile plot. Moving from left to right on the plot, which corresponds to moving from top to bottom of the image, the intensity is constant until it reaches the bowl-shaped burning surface. The bowl shape has a downward curve of intensity values until they reach a constant value within the unburned propellant region. Since the bowl shape is clearly visible on the intensity profile plot, the plot implies that a computer program could automatically track the movement of the burning surface by comparing the intensity plots of consecutive X-ray video frames to each other. Simulations of possible consecutive images are shown in Figure 30, and their corresponding intensities of the center column of pixels are plotted in Figure 31.

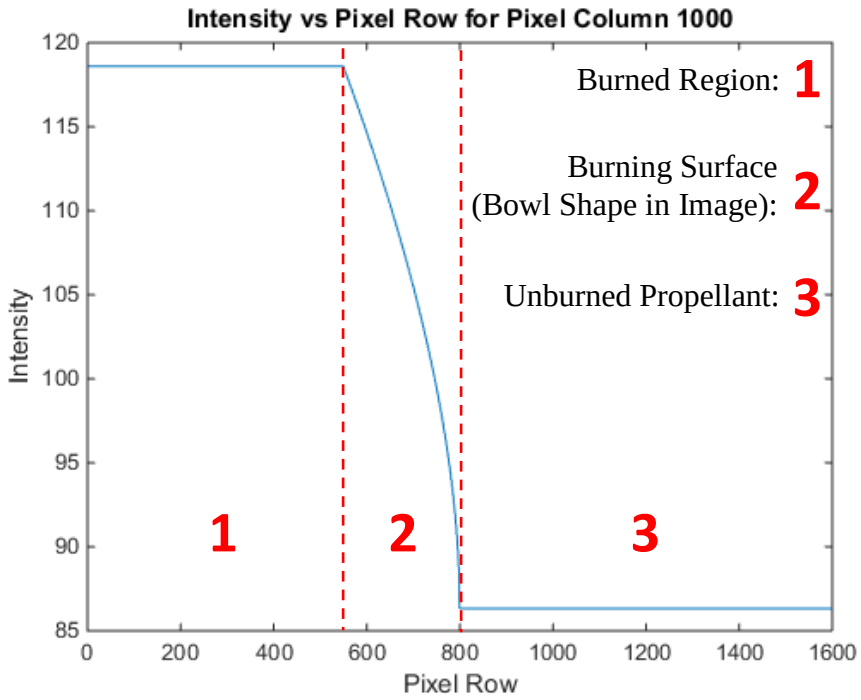


Figure 29: Intensity Plot for the Center Column of Pixels in the Computer-Generated X-ray Image Simulation of a Propellant Sample

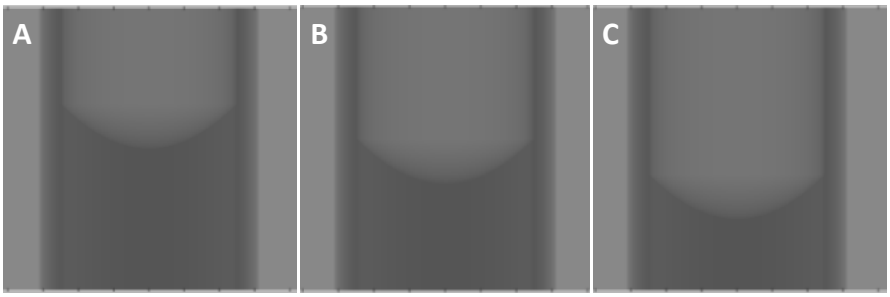


Figure 30: Simulated Consecutive X-ray Frames

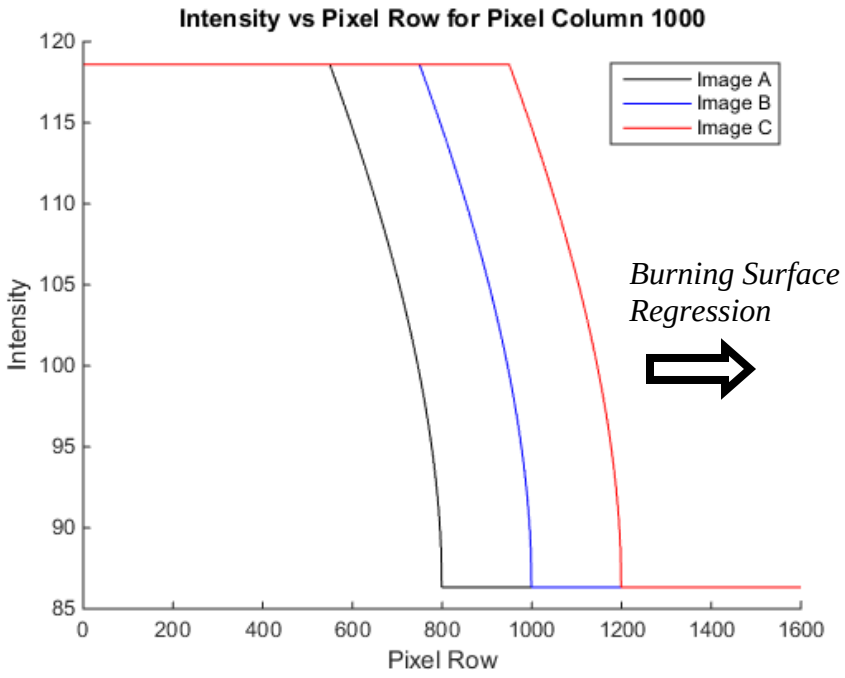


Figure 31: Intensity Plot for the Center Column of Pixels for Multiple Frames of a Propellant Sample Simulation

The moving contour of the burning surface shows that the bottom edge of the burning surface could be tracked over time by tracking the pixel row at which the intensity reached a constant value. This could be done by finding a zero-crossing of a slope derivative which would mark the location where the intensity plot turned to the horizontal direction. Alternatively, the surface could be tracked by tracking the constant intensity value itself corresponding to the bottom edge of the burning surface.

3.2.2 **Least Square Error Line Fit**

If the burning surface could be tracked in every frame, then the burning surface locations could be plotted versus time to find the regression rate. In a constant pressure environment, the observed burning rate should be constant, meaning that the burning surface locations plot should be linear with time. Then a line fit could be applied to the data by the finding the least square sum of error of the data from a fitted line. The absolute value of the slope of the fitted line would then represent the burning rate. An illustration of burning surface locations plotted versus time with a least squares fitted line is provided in Figure 32.

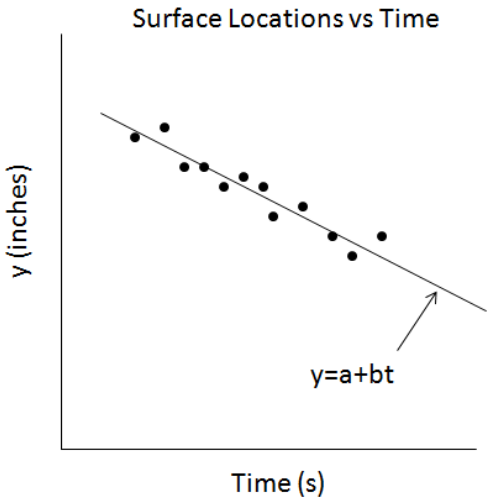


Figure 32: Least Squares Line Fit to Burning Surface Locations

The slope of the fitted line is the coefficient b [32] in the equation

$$y = a + bt, \tag{3.1}$$

where y is the dependent variable representing the location of the burning surface, t is the independent variable representing time, and a is the value of y when $t = 0$. The time step

was *n* times 1/30th of a second, since the digital camera captured images at 30 frames per second. The coefficient a can be calculated [32] as

$$a = \frac{\sum y_i \sum {t_i}^2 - \sum t_i \sum t_i y_i}{N \sum {t_i}^2 - (\sum t_i)^2} \ , \tag{3.2}$$

where y_i is the i^{th} frame, and N is the total number of terms. The coefficient b can be calculated [32] as

$$b = \frac{N \sum t_i y_i - \sum t_i \sum y_i}{N \sum {t_i}^2 - (\sum t_i)^2} \ . \tag{3.3}$$

The squared correlation coefficient can be calculated as

$$R^2 = \frac{\sum [y(x_i) - y_m]^2}{\sum [y_i - y(x_i)]^2 - \sum [y(x_i) - y_m]^2} \ , \tag{3.4}$$

where $y(x_i)$ is the result of evaluating the fitted line equation at x_i, and y_m is the mean value of all of the y_i.

3.2.3 **Monte Carlo Burning Rate Analysis**

In order to determine the propellant burning rates from the X-ray videos of the combustion experiments, a code was written in MATLAB to track the burning surface of the propellant. After tracking the location of the burning surface in every sequential frame, the code used a least squares line fit to obtain the burning rate from the sequential burning surface locations. A Monte Carlo analysis was performed by randomly varying certain parameters such as the burning surface locations found and then calculating the burning rate for a large number of iterations. The statistics of the resulting distribution of burning rates produced the burning rate reported for each test and its corresponding uncertainty bounds.

CHAPTER 4

EXPERIMENT DATA ANALYSIS

4.1 Overview of Experiment Data Analysis

The MATLAB edge tracking code calculated the burning rate for each test by tracking the location of the burning surface through every sequential frame in a selected range of frames of the X-ray video. The code performed a Monte Carlo analysis on the burning rate for each test to account for various biases and potential errors. In the Monte Carlo analysis, random variations were applied to certain parameters, the burning surface locations were determined, and the least squares line fit slope was calculated for a large number of iterations such that a distribution of burning rates was produced. The mean of the burning rate distribution for each test was reported as the final burning rate value for that test, and the bounds corresponding to two standard deviations of the burning rate distribution were reported as the uncertainty bounds with 95% confidence.

4.2 Edge Tracking Operations

4.2.1 X-ray Image Enhancement

An example X-ray image is shown in Figure 33. The image is a raw frame from an X-ray video while the sample was burning. The bowl shape of the burning surface is visible, and it had regressed away from the ignition wire. The burned and unburned regions of propellant are identified in the image.

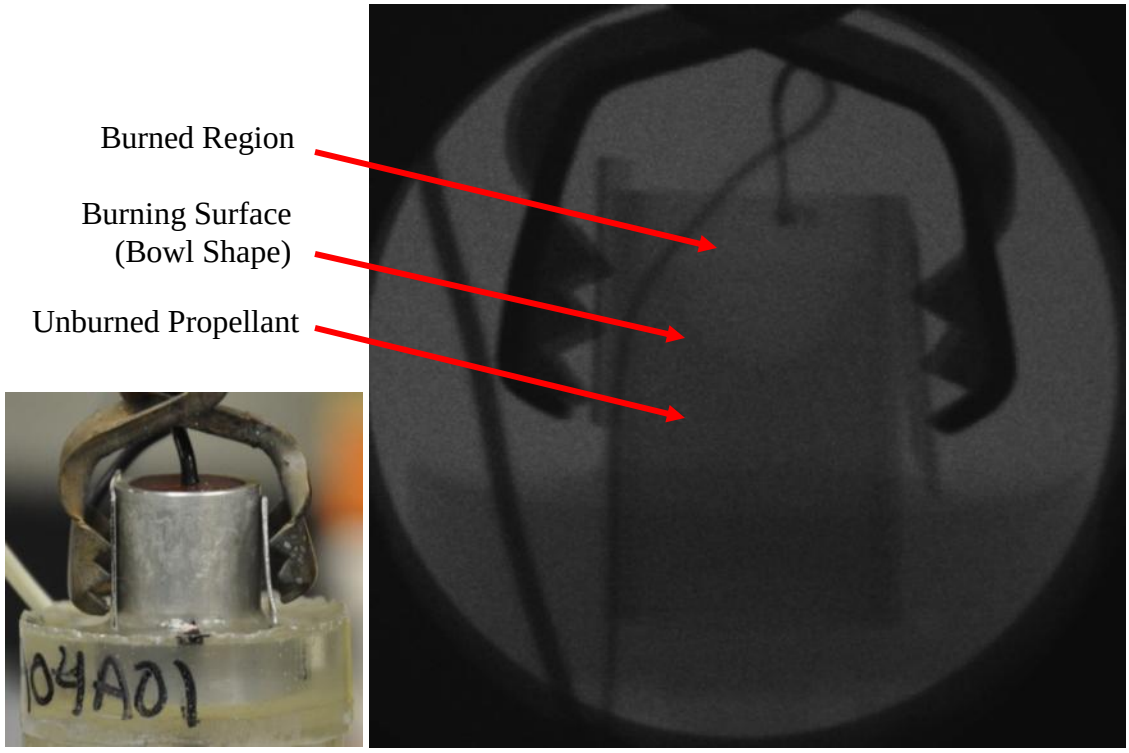


**Figure 33: Example Images: Pre-test Picture (Left),
X-ray Video Frame While Burning (Right)**

The MATLAB edge tracking code first enhanced the images by performing a smoothing operation, cropping, and a gray-scaling operation on the X-ray frames before it tracked the burning surface. The smoothing operation was performed by a median filter using a neighborhood of 25x25 pixels around each pixel. Each pixel took the median value of its 25x25 neighbors. A block of 25x25 pixels has been blackened for reference in Figure 34-D. The smoothing filter caused some blurring of the images, but the salt-and-pepper noise in the images was removed so that the propellant in the image had smooth gradients of transmitted intensity or gray values across the burned and unburned regions. Since the noise was removed, the smooth gradients could be more easily identified on a plot of the intensity. An example of an image that was smoothed is shown in Figure 34-B.

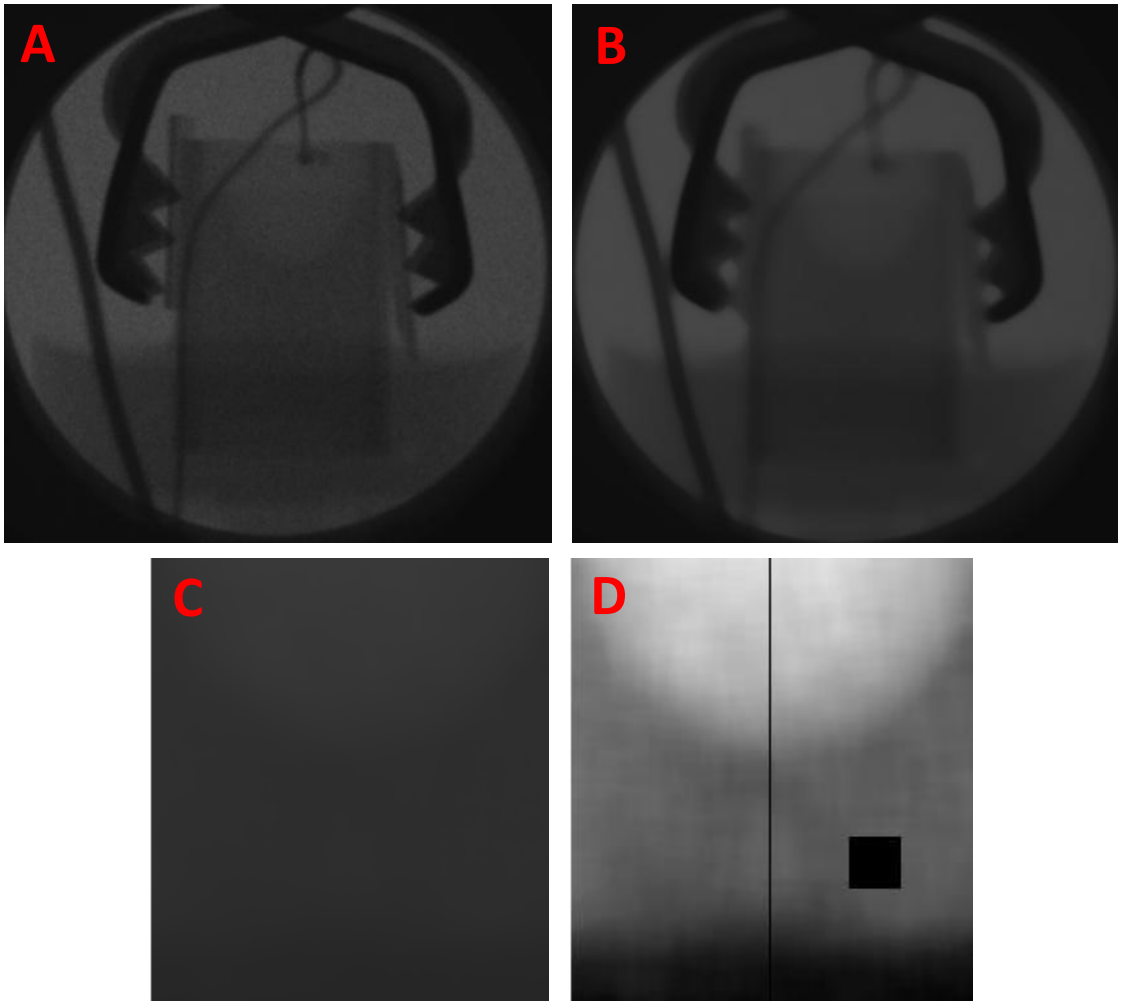


Figure 34: Example X-ray Images From One Video Frame: Raw Image (A), After Filtering (B), After Cropping (C), After Scaling (D)

After each image was smoothed with the median filter, the images were cropped in order to focus on a region of interest containing the burning surface. This saved computation time later in the code, and more importantly, the cropped images were gray scaled to use the full range of possible gray values allowed by the bit depth of the images. This increased the contrast of the burning surface. An example series of cropped images is shown in Figure 35.

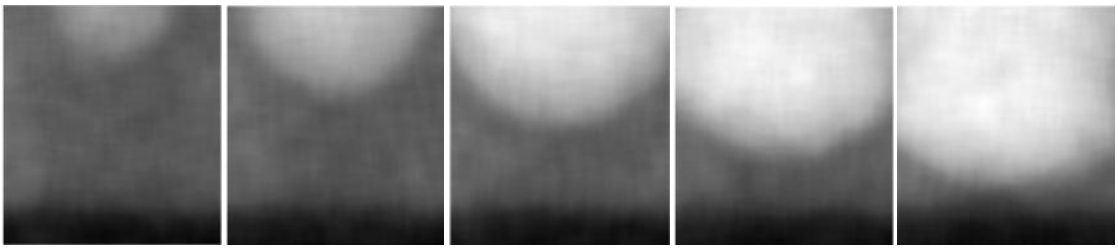


Figure 35: Example Cropped X-ray Frames

The gray-scale values of the images were scaled across the full possible range of gray values. The images were all gray-scaled together based on the same minimum and maximum values. The minimum gray value out of all of the images was determined, and then, for each image, all of the gray values were shifted down by subtracting the total minimum gray value out of all of the cropped images such that the total minimum value became zero. Then all of the gray values of the images were multiplied by a scale factor such that the maximum value out of all of the images was brought up to the maximum possible gray value for the bit depth of the images. The gray-scaling factor is given in Equation (4.1). A gray-scaled image in MATLAB is shown in Figure 34-D.

$$Gray\ Scale\ Factor = \frac{max\ gray\ value\ for\ bit\ depth}{\max(all\ images) - \min(all\ images)}\ , \tag{4.1}$$

After the images were filtered, cropped, and scaled, the code plotted the transmitted intensity for one column of pixels near the center of the image. The column of pixels that was most centered with the bottom of the bowl-shaped burning surface was manually selected by the user, as demonstrated by the line in Figure 34-D.

An intensity plot from the example image in Figure 34-D is given in Figure 36. The various regions of the image from both Figure 33 and Figure 34 are indicated on the plot. Across the gray scale, the darkest black color had a value of 0 and the brightest white color had the highest possible value of the bit depth. For example, if the images

were saved in 8-bit format, then the brightest gray value was 255. On the plot, the intensity exhibits a downward curve on the burning surface, because the burning surface is a three-dimensional bowl shape. Due to the bowl shape, there was more unburned propellant around the bottom point of the bowl surface than there was around the top of the bowl, so the X-rays were more attenuated near the bottom than at the top. Near the bottom of the unburned propellant, the color intensity drops because the epoxy base causes additional attenuation of the X-rays. Examples of intensity plots before and after the image smoothing are shown in Figure 37.

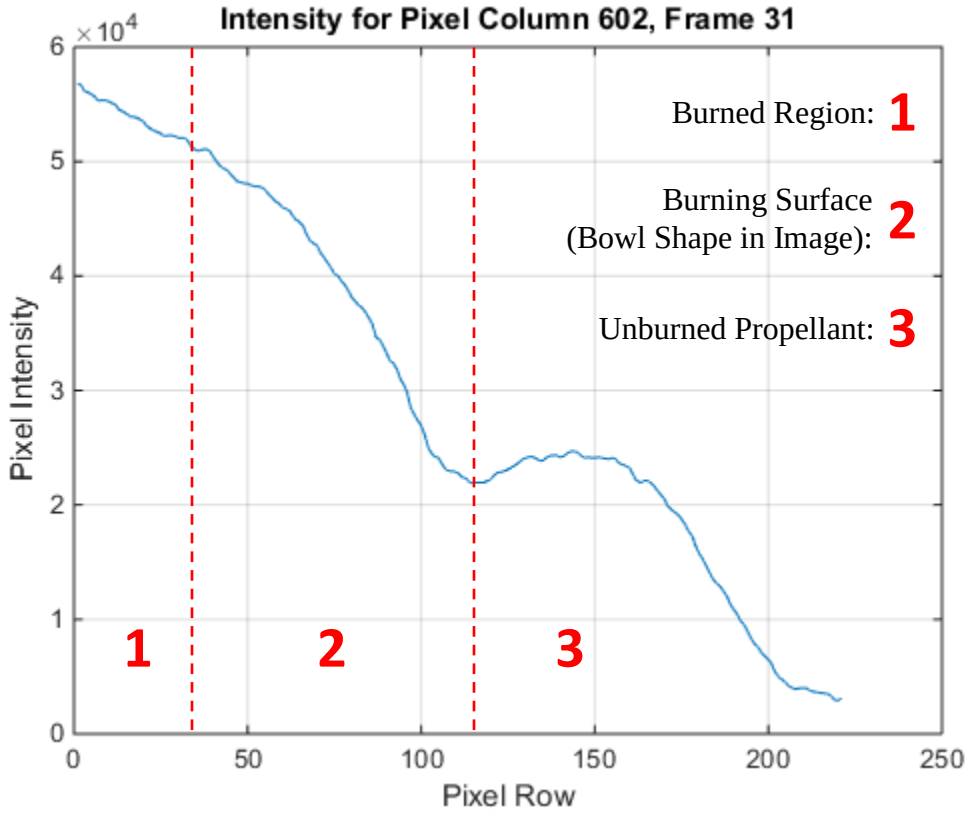


Figure 36: Example Pixel Intensity Plot from One Cropped X-ray Video Frame

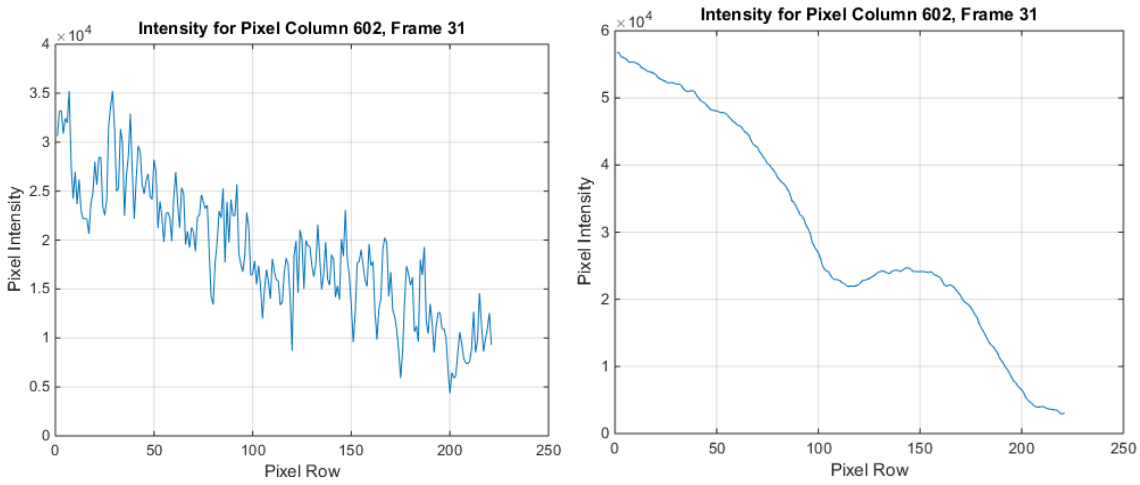


Figure 37: Example X-ray Intensity Plot of One Column of Pixels Before Filtering but Gray-Scaled (Left), After Filtering and Gray-Scaling (Right)

The code produced a plot showing the numerical intensity values for the center column of pixels of every video frame or every n^{th} video frame, where *n* was usually one, but *n* could be set to any value for better visualization. The exact column of pixels used in the code was selected by the user for what appeared to be the pixel column in the center of the bowl-shaped burning surface, and it was different for every test. An example plot of the intensities for multiple frames is shown in Figure 38. On the plot, the regression of the burning surface is represented by the spacing of the lines from left to right since the horizontal axis of the plot corresponds to the pixel row number in X-ray image. The frames that were used by the code were selected manually by the user for a range of consecutive frames that appeared to have a well-defined burning surface. For the final burning rate calculations, no frames were omitted within the selected range of frames.

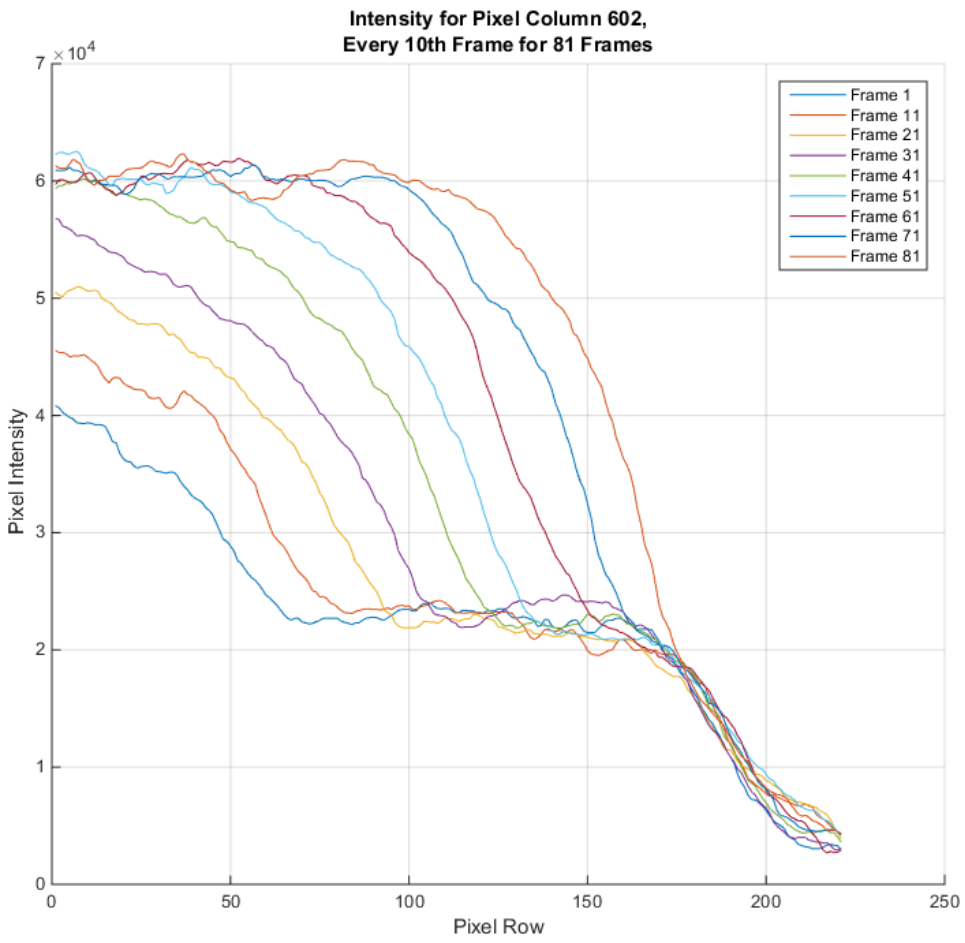


Figure 38: Example X-ray Intensity Plot: One Column of Pixels for Several Frames

4.2.2 Method 1: Constant Target Intensity Value

In order to calculate the burning rate, the burning surface had to be tracked in each X-ray frame by the MATLAB code. As previously discussed, a derivative zero-crossing method was considered in which the burning surface would be tracked as the location where the slope of the intensity line turned to a horizontal direction. However, this method would be unreliable with the data obtained in the experiments. Sometimes, the intensity contours experienced a small horizontal turn midway down the curve, and sometimes they never turned to a horizontal value at all. Therefore the derivative zero-crossing method would have large errors.

The other method previously mentioned was to track the constant intensity value at the bottom tip of the bowl. For each frame, the MATLAB code could step along the intensity curve until it found the point on the curve where the intensity fell to a specific target value that had been given to the code as an input. Intensity plots like the plot in Figure 38 showed that the intensity curves did not all turn horizontal at one consistent intensity value for the bottom of the bowl, so the constant intensity value selected would need to be above the average value at which the curves turned as shown in Figure 39. Otherwise, some intensity curves would turn before reaching the target intensity value and cause large error. This target intensity tracking technique was the first method used to track the burning surface and measure burning rates since it was assumed that a point slightly above the burning surface would be moving at the same regression rate as the true burning surface.

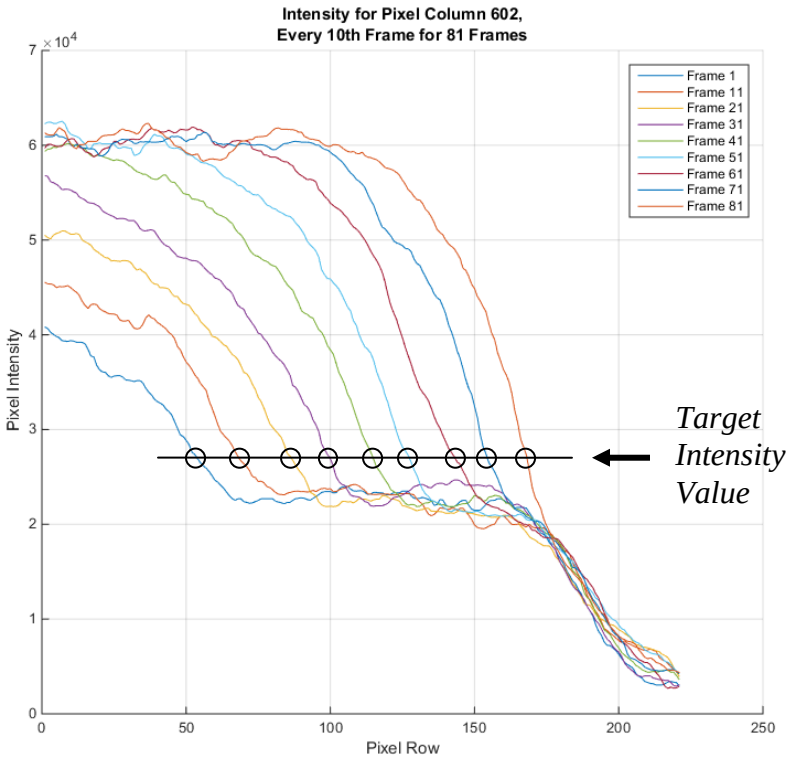


Figure 39: Tracking the Constant Target Intensity Value

After some time, a bias error in the target intensity tracking technique was discovered due to the widening of the bowl shape of the burning surface over time. The bias error resulted from the change in intensity gradient near the burning surface as the bowl widened. The bowl shape was expected to widen over time since the burning surface of a solid propellant regresses perpendicular to the surface. The simulated regressing images and plots from Figure 30 and Figure 31 do not show the widening of the propellant burning surface, but this widening effect was clearly observed in the experiment videos, and it is visible on the example intensity plot in Figure 38 as represented by the changing of the slopes of the lines in the bowl region. The slopes become steeper from frame to frame.

Another simulation code was written to illustrate the changing intensity contours of a widening bowl shape. The output plots of the code are pictured in Figure 40. The code plotted 2-D analytical contours representing the 3-D surfaces and calculated the relative X-ray attenuation through every cross-section of each contour with Beer's law, which is given in Equation (2.4). The MATLAB code is provided in APPENDIX E. The plots show that there could be a bias in the burning rate measurement leading to an artificially high burning rate if a constant target intensity value was tracked.

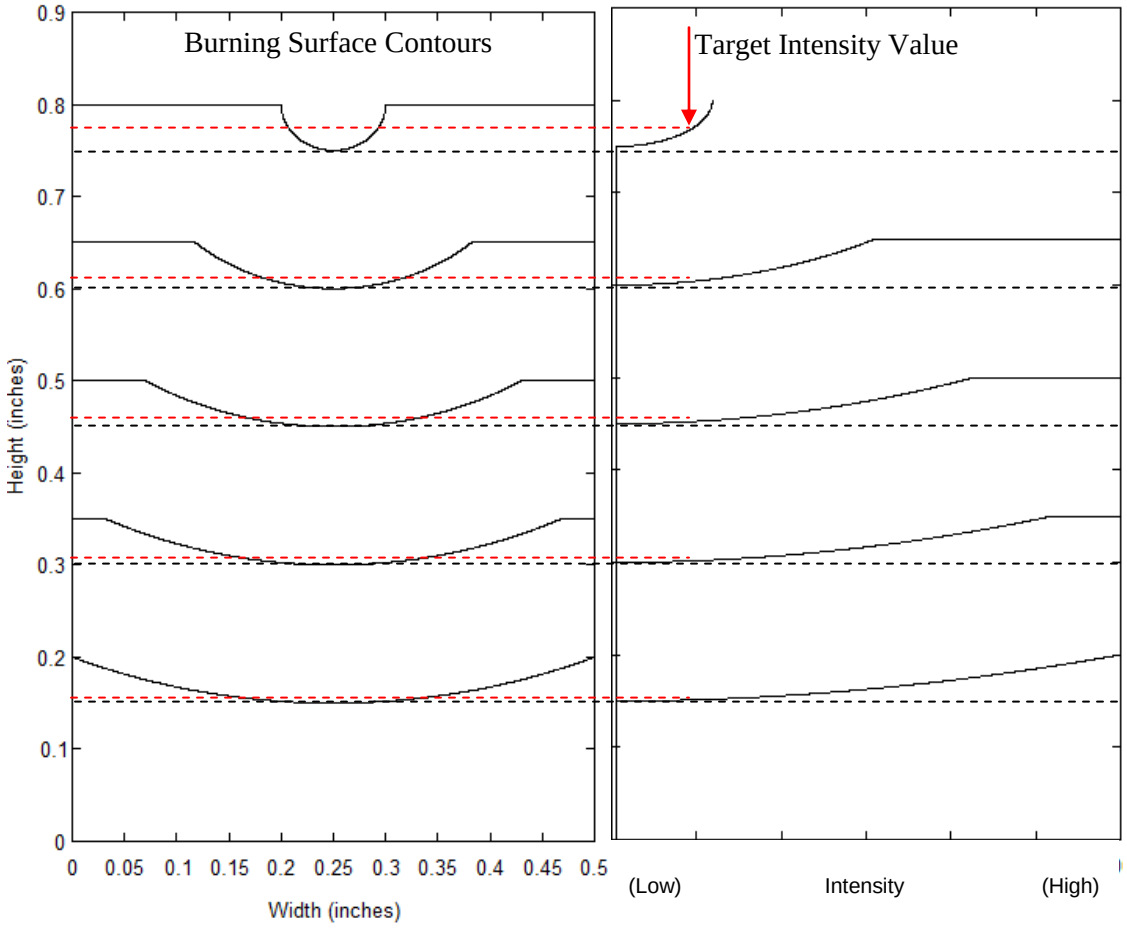


Figure 40: Inconsistent Location of Target Intensity Value Relative to Burning Surface Due to Changing Shape of Burning Surface

The left side of Figure 40 shows the regression of a spherical burning surface. The right side of the figure shows the relative X-ray intensity increasing from left to right for X-rays projected horizontally through the propellant. The constant target intensity value is indicated by the red arrow, and it corresponds to a point that is above the bottom tip of the burning surface in the first contour. The red dotted lines indicate the location of the target intensity value in the propellant. The black dotted lines indicate the location of the bottom of the bowl-shaped burning surface. In the sequential contours, the location of the constant target intensity value moves closer and closer to the bottom edge of the bowl-shaped burning surface, because the X-ray path length through the propellant must be

held constant to achieve the constant intensity value. This is visible as the red dotted line moves closer and closer to the black dotted line. This movement of the target intensity value relative to the burning surface would cause an artificially high burning rate measurement.

4.2.3 **Method 2: Projected-Slope Intersections**

In order to prevent the intensity gradient bias error, an improved technique of locating the burning surface was used in the MATLAB edge tracking code. Instead of tracking a constant intensity value on each intensity curve itself, the code applied a line fit to the intensity slope of each frame and calculated the location of the intersection of the projected line fit with a constant target intensity value marking the average perceived location of the burning surface. An example plot showing the line projections and intersections is given in Figure 41. This approach allowed the target intensity value to be selected at the bottom tip of the bowl-shaped burning surface instead of at a location well above the bottom of the bowl. With the target intensity value marking the bottom tip of bowl, the intensity gradient bias due to bowl widening was avoided, because the projected slope line fit of the intensity curve accounted for the changing intensity gradient.

The improved edge tracking technique was incorporated into the Monte Carlo analysis to calculate the burning rate. For each X-ray frame, the projected slope line took the slope of the downward curve of the intensity plot in the bowl region, which is region 2 in Figure 36. The slope was calculated by finding the pixel rows corresponding to two constant intensity thresholds on the intensity curve, with one threshold near the top of the curve and one threshold near the bottom of the curve. The thresholds are represented as

black horizontal lines in Figure 41. The slope of a line drawn between the two threshold points on the curve was projected down to the region on the plot where the curve turned to a relatively horizontal direction, since the location of that turn was considered to be the bottom of the bowl-shaped burning surface. The intersection of the projected slope line with the constant intensity value at the horizontal turn was marked as the location of the bottom of the burning surface for that frame. These three threshold intensity values were selected by the user as constant values to be used for all frames for a given test. The constant intensity value marking the horizontal turn is shown as the dashed line across the plot in Figure 41.

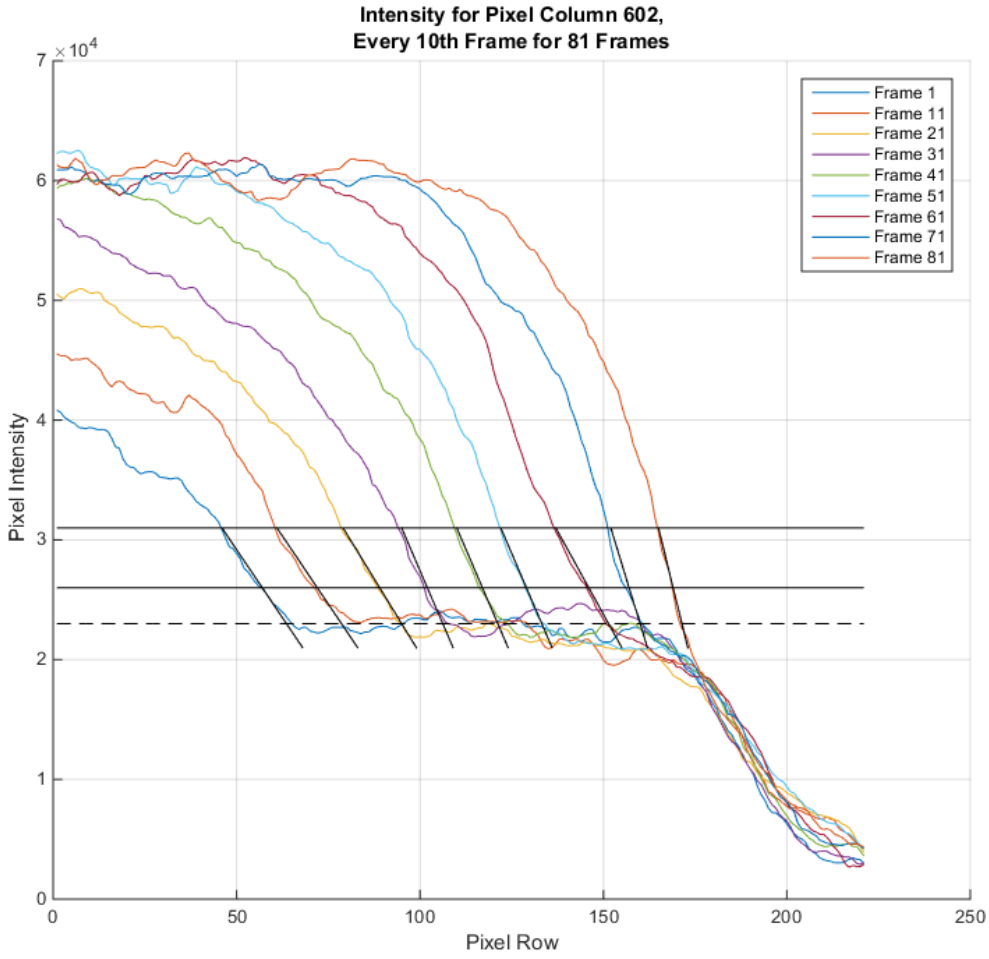


Figure 41: Example X-ray Intensity Plot: One Column of Pixels for Several Frames with Intensity Thresholds and Projected Slope Lines

4.3 Monte Carlo Analysis Inputs

The two-step task of calculating the burning rate involved the projected-slope intersections and the least squares line fit as described. To estimate the burning rate uncertainty, the MATLAB edge tracking code performed this two-step task 2,000,000 times while randomly varying a few parameters in each iteration. This technique is known as a Monte Carlo analysis. The parameters that were randomly varied were the intensity threshold values, the pixel column, the distance scale factor, and the row number determined after the projected-slope intersection routine. The input ranges for the random variations that were applied to these parameters are summarized in Table 6. The final burning rate reported for each test was the mean value of the 2,000,000 burning rates calculated in the Monte Carlo analysis. The standard deviation of the burning rate distribution was also calculated for each test, and the bounds of two standard deviations were taken as the uncertainty bounds of the burning rate for roughly a 95% confidence level. The 95% confidence level was applied because, for each test, two standard deviations of the Monte Carlo burning rate distribution contained about 95% of the burning rate values. Association of this confidence level with two standard deviations is characteristic of a normal distribution [32].

Table 4: Monte Carlo Input Variations

Parameter	UAH04A01 Variation	UAH04A02 Variation
Upper Slope Intensity Threshold	±1000 Uniformly Distributed	±13 (Uniformly Distributed)
Lower Slope Intensity Threshold	±1000 Uniformly Distributed	±13 (Uniformly Distributed)
Intensity Threshold of the Perceived Burning Surface	±1000 Uniformly Distributed	0
Pixel Column	±3 Uniformly Distributed (rounded to integers)	±3 Uniformly Distributed (rounded to integers)
Scale Factor	±4.98 Uniformly Distributed	±9.32 Uniformly Distributed
Intersection Row	±3.640 Normally Distributed	±2.953 Normally Distributed

4.3.1 **Variation of the Intensity Thresholds**

The intensity thresholds were varied within a range where the user might select a threshold value from examination of the intensity plots. These values were different between tests. This difference is due to the difference in the bit depths of the X-ray videos as described in Chapter 5. The intensity variation value of 13 for UAH04A02 was chosen because that was the smallest increment of the scaled resolution of the intensity, which had been gray-scaled to fit 8-bit format from 6-bit data. This variation represented a larger range of the UAH04A02 intensity curves than the value of 1000 represented on the range of the UAH04A01 intensity curves. The intensity threshold of the perceived location of the burning surface was not varied for UAH04A02 because the turn of the intensity curves was clearly concentrated around the value that was selected, and the low resolution of gray values would cause much wider variation than would be appropriate. Any error from this lack of variation of the bottom intensity threshold for UAH04A02 is also considered balanced by the fact that the variation of the slope intensity thresholds for UAH04A02 had a somewhat wider range than that of UAH04A01.

4.3.2 **Variation of the Pixel Column**

The determination of the pixel column at the center of the bowl-shaped burning surface for each test was made by visual examination of the X-ray video. Potential error could have arisen from a poor choice of pixel column because the burning rate result would be an artificially high value if a pixel column was selected in any region besides the center. This is due to the fact that, as the bowl-shaped burning surface regressed downward, the bowl shape itself also widened. The bowl shape widened because the surface was actually burning in all directions away from the point of ignition, which was

the small wire inserted into the top of the propellant, so the burning shape and direction could be thought of as spherical. Since the code measured regression in only the vertical direction, a surface tracking measurement in a pixel column away from the center column would have incorporated the widening of the bowl in addition to the downward regression, which would result in an artificially high burning rate. In order to ensure that the potential error from the choice of center pixel column was addressed, the Monte Carlo code randomly varied the pixel column by up to ±3 pixels with each iteration so that the variation due to pixel column choice would be captured in the uncertainty bounds. The pixel column variation was kept relatively small because, if a pixel column was selected too far from the center of the bowl, then the burning rate measurement would be artificially high due to the widening of the bowl.

4.3.3 **Variation of the Scale Factor**

The uncertainties of the pixel-to-inch scaling for each test were assessed and incorporated into the Monte Carlo analysis as inputs for variation of scale factor in the code. For a function *a=f(x,y),* Beckwith et al., [32] give the uncertainty U_a of the function *a* as

$$U_a = \sqrt{\left(\frac{\partial a}{\partial x}U_x\right)^2 + \left(\frac{\partial a}{\partial y}U_y\right)^2} \quad \textbf{(4.2)}$$

The final calculation for burning rate was to divide the burning rate in pixels-per-second by a scale factor of pixels-per-inch. The scale factor was calculated as the quotient of pixel distance measured in the X-ray video frames over a known distance in the X-ray images. For UAH04A01, the known distance used was the height of the aluminum tube. For UAH04A02, the known distance used was the outside diameter of the aluminum tube. Then the scale factor was,

$$SF = \frac{p}{d}, \tag{4.3}$$

where p represents the number of pixels and d represents the tube height for UAH04A01 and the tube outside diameter for UAH04A02. The uncertainties for the pixel measurements p were calculated with 95% confidence as two standard deviations of a set of 30 measurements of each dimension. The measurements and their statistics are shown in Table 5.

Table 5: Pixel Measurement Statistics for Distance Calibration

UAH04A01		
Aluminum Tube Height=0.902 inch		
top	bottom	span
364	818	454
365	817	452
363	816	453
362	814	452
363	817	454
363	816	453
365	816	451
364	816	452
364	816	452
363	814	451
363	815	452
363	815	452
364	816	452
363	816	453
362	814	452
363	816	453
363	814	451
363	816	453
363	816	453
362	816	454
363	814	451
362	814	452
363	817	454
363	816	453
363	815	452

UAH04A02		
Tube diameter=0.625 inch		
left	right	span
405	719	314
405	721	316
407	719	312
405	716	311
416	723	307
407	720	313
404	720	316
406	720	314
405	718	313
405	718	313
405	719	314
406	718	312
406	722	316
404	721	317
404	722	318
406	722	316
407	723	316
406	722	316
406	721	315
407	720	313
406	721	315
406	720	314
407	720	313
406	720	314
406	720	314

362	816	454	
362	817	455	
362	816	454	
362	815	453	
362	817	455	
Mean =		452.73	pixels
Standard Deviation (σ) =		1.1427	pixels
U_p = ±2*σ =		±2.2854	pixels
SF =		501.7	ppi

406	719	313	
407	722	315	
406	721	315	
406	721	315	
405	721	316	
Mean =		314.2	pixels
Standard Deviation (σ) =		2.1075	pixels
U_p = ±2*σ =		±4.2149	pixels
SF =		502.7	ppi

The resulting scale factors at the bottom of Table 5 were calculated as the mean pixel distance divided by the known distance. The uncertainty in the height of the tube was calculated using Student's t-distribution [32] as described in APPENDIX B. The uncertainty in the outside diameter of the aluminum tube came from the distributor, McMaster-Carr, as ±0.008 inch. The uncertainty in scale factor was calculated as follows:

$$U_{SF} = \sqrt{\left(\frac{\partial SF}{\partial p} U_p\right)^2 + \left(\frac{\partial SF}{\partial d} U_d\right)^2}. \quad \textbf{(4.4)}$$

Substituting the partial derivatives, the uncertainty became

$$U_{SF} = \sqrt{\left(\frac{1}{d}\ U_p\right)^2 + \left(\frac{p}{d^2}\ U_d\right)^2}. \quad \textbf{(4.5)}$$

For UAH04A01, the scale factor uncertainty was

$$U_{SF} = \sqrt{\left(\frac{1}{0.902}\ 2.285\right)^2 + \left(\frac{452.7}{0.902^2}\ 0.0077\right)^2} = 4.98\ \frac{pixels}{inch} \quad \textbf{(4.6)}$$

For UAH04A02, the scale factor uncertainty was

$$U_{SF} = \sqrt{\left(\frac{1}{0.625}\ 4.215\right)^2 + \left(\frac{314.2}{0.625^2}\ 0.008\right)^2} = 9.32\ \frac{pixels}{inch} \quad \textbf{(4.7)}$$

In each iteration of the Monte Carlo burning rate analysis, the scale factor was varied by a random amount up to the corresponding scale factor uncertainty for that test. The random variation was uniformly distributed in order to be conservative since the use of Student's t-test only assumed that the underlying tube length distribution was normally distributed.

4.3.4 **Variation of the Intersection Row**

In each iteration of the Monte Carlo analysis, a random variation was applied to the row number determined by the code for the burning surface location in each frame. This variation was applied after the projected-slope intersection was calculated for each frame. The range of numbers from which a random amount of variation was selected was determined by a study of an inert epoxy model of a propellant sample.

The model of the ESP propellant samples was made with epoxy and an aluminum tube as shown in Figure 42. The model had a bowl shape in the epoxy to simulate the burning surface of the propellant. The model was X-rayed and studied in order to determine the consistency with which the MATLAB code identified the location of the burning surface over multiple frames. The aluminum tube was slightly thinner than the tube used in the ignition ESP samples. It had a wall thickness of 0.035 inch, but the outside diameter was 0.625 inch as for the ignition samples. The tube was filled with 12-hour epoxy, and a rounded cone-shaped die was used to mold a bowl shape into the epoxy. The epoxy propellant model was placed inside of the combustion bomb for X-ray imaging so that it would be exposed to the exact same levels of radiation as the real propellant samples. A picture of the model's orientation in the combustion bomb is

shown in Figure 43. Example frames of the cropped and scaled images are shown in Figure 44.

Eighty X-ray frames of the epoxy propellant model were captured with 14-bit gray-scale video, and eighty more frames were captured with 6-bit gray-scale video. Each set of eighty frames was analyzed in a MATLAB code to determine the location of the bottom of the epoxy bowl by the same projected-slope intersection method used for the real burning rate data. Intensity plots are shown in Figure 45 and Figure 48. Plots showing the locations found by the code are given in Figure 46 and Figure 49. As can be seen in the plots, the locations found by the code had significant variation even though the epoxy bowl surface was not moving. Therefore these pixel ranges had to be considered in the Monte Carlo analyses on burning surface locations. Since the epoxy propellant model was placed in the combustion bomb in the same way as the propellant samples, it had roughly the same pixel-to-distance scale as the propellant samples, so no adjustment was made to the values for scale factor. The determined locations had approximately normal distributions as shown by the percentage of values within two standard deviations in the histograms in Figure 47 and Figure 50.

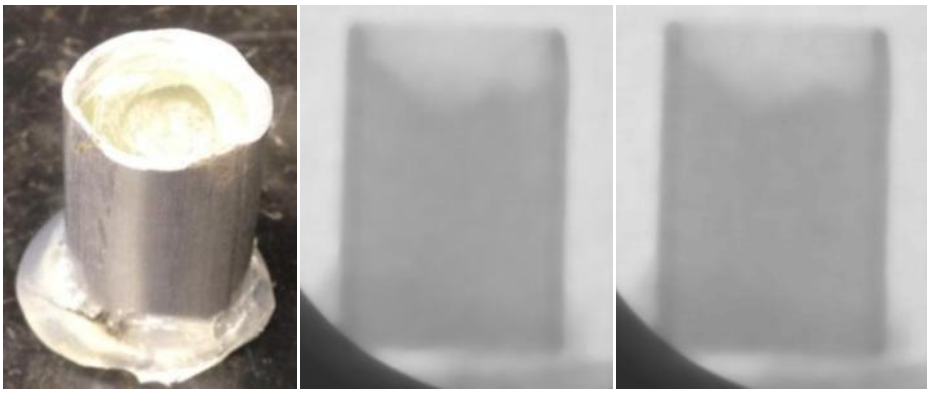


Figure 42: Epoxy Propellant Model: Picture (Left), 14-bit X-ray Image (Middle), 6-bit X-ray Image (Right)

Figure 43: Epoxy Propellant Model Inside of Combustion Bomb for X-ray Imaging

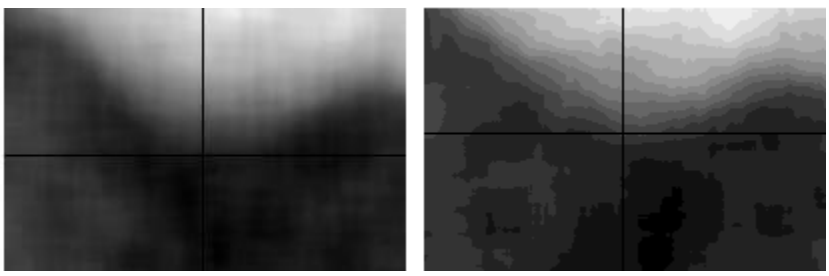


Figure 44: Epoxy Model Cropped X-ray Images: 14-bit (Left), 6-bit (Right)

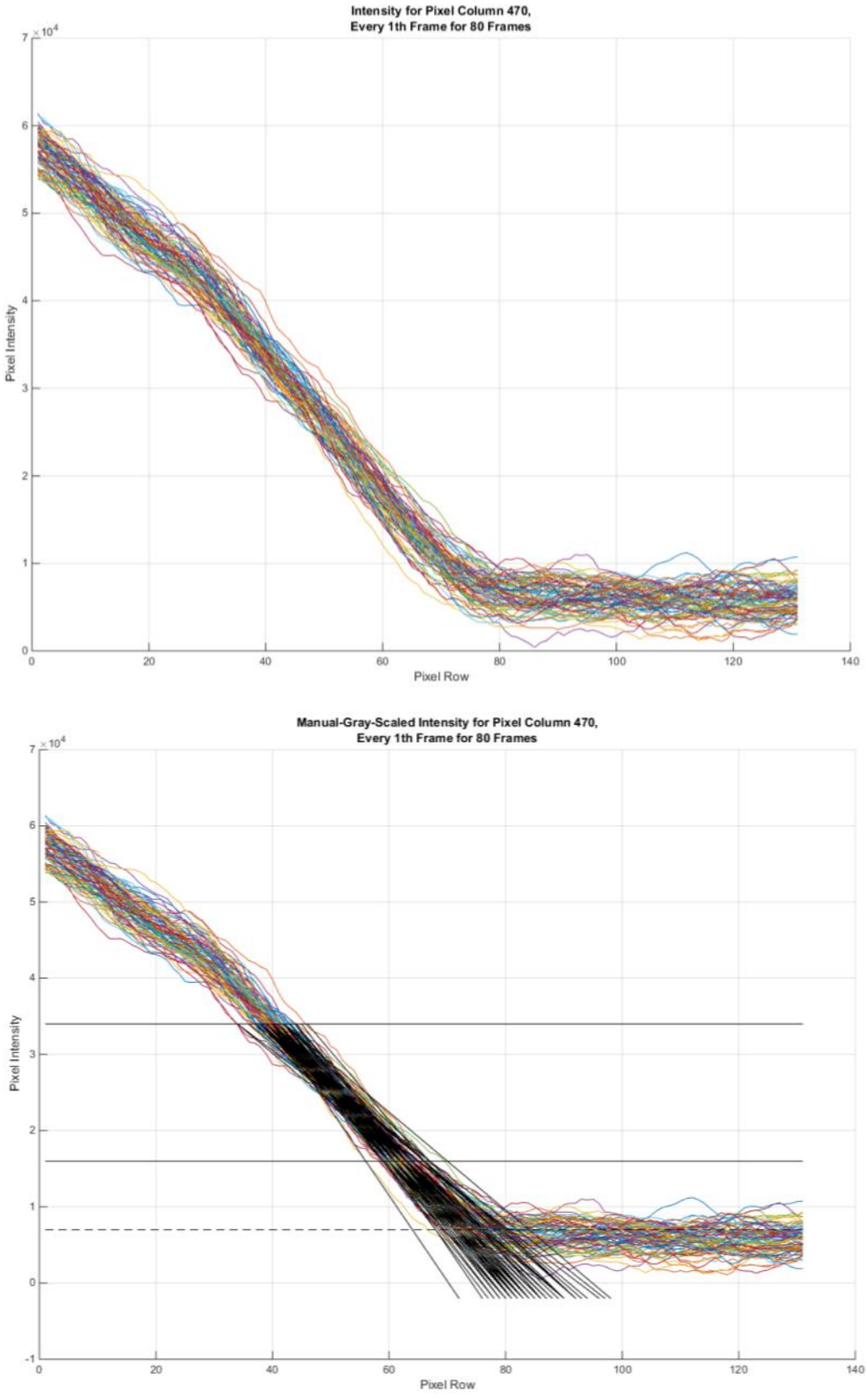


Figure 45: Intensity Plots for the Motionless Epoxy Propellant Model (14-bit)

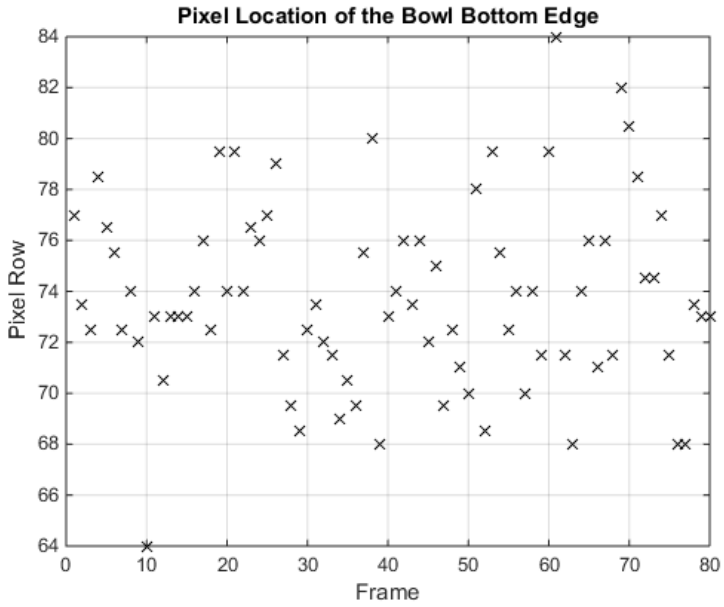


Figure 46: Bowl Edge Pixel Locations Determined by the Code for 80 Frames of the Motionless Epoxy Propellant Sample with 14-bit Gray-Scale Video

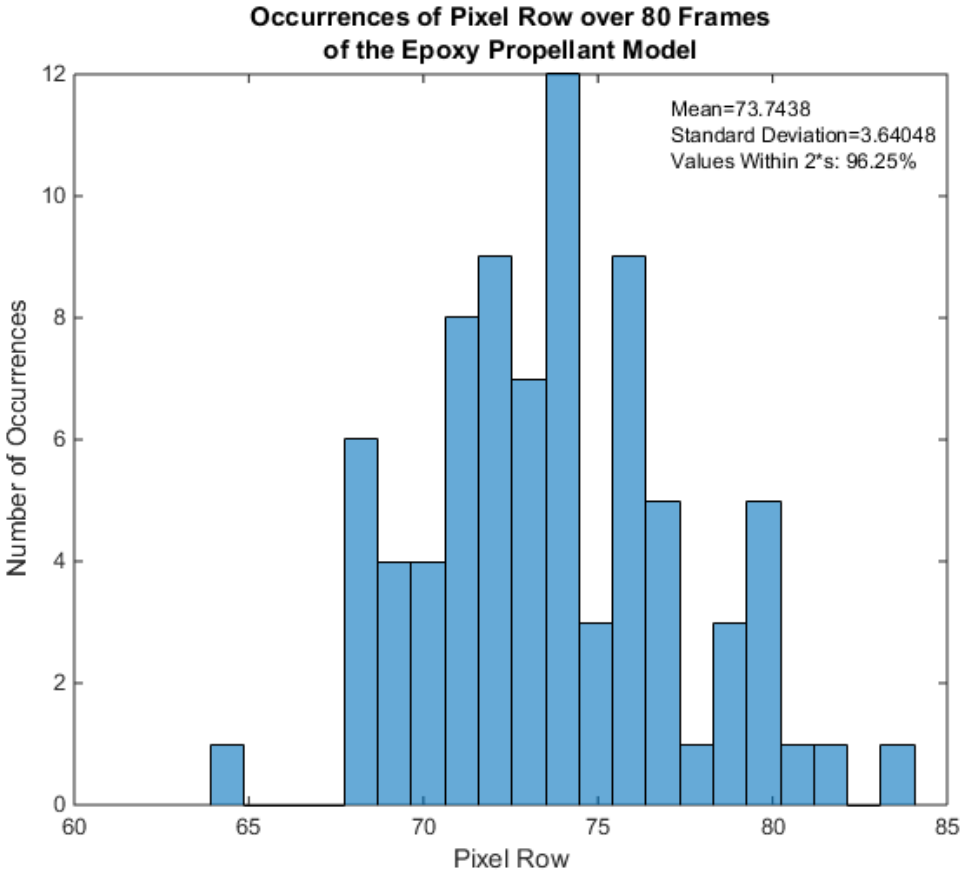


Figure 47: Histogram of 14-bit Epoxy Bowl Edge Locations

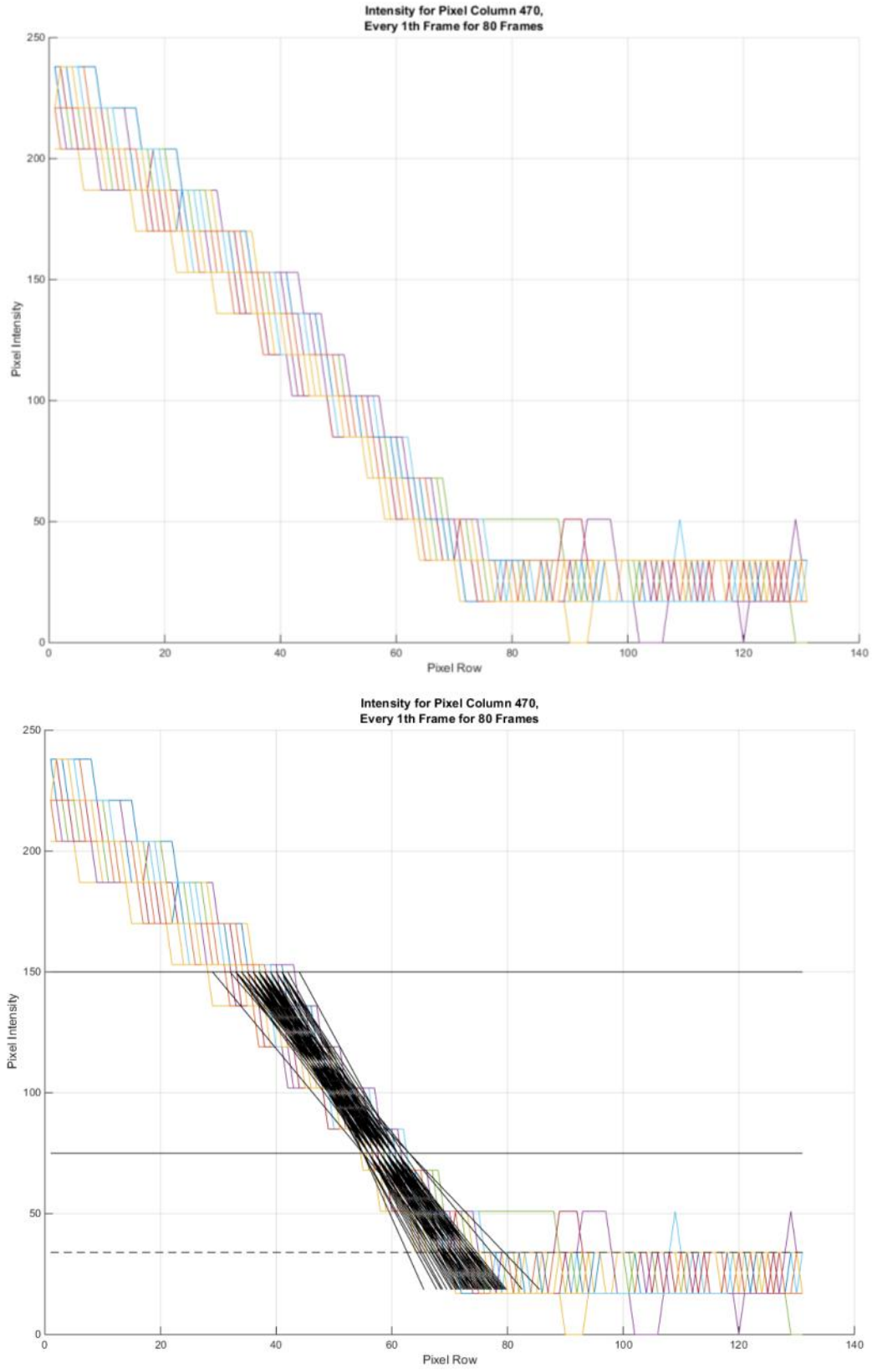


Figure 48: Intensity Plots for the Motionless Epoxy Propellant Model (6-bit)

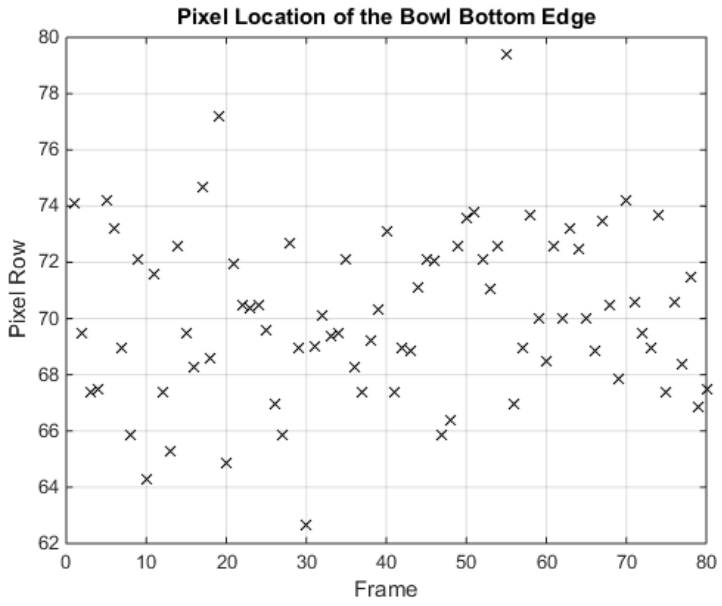


Figure 49: Bowl Edge Pixel Locations Determined by the Code for 80 Frames of the Motionless Epoxy Propellant Sample with 6-bit Gray-Scale Video

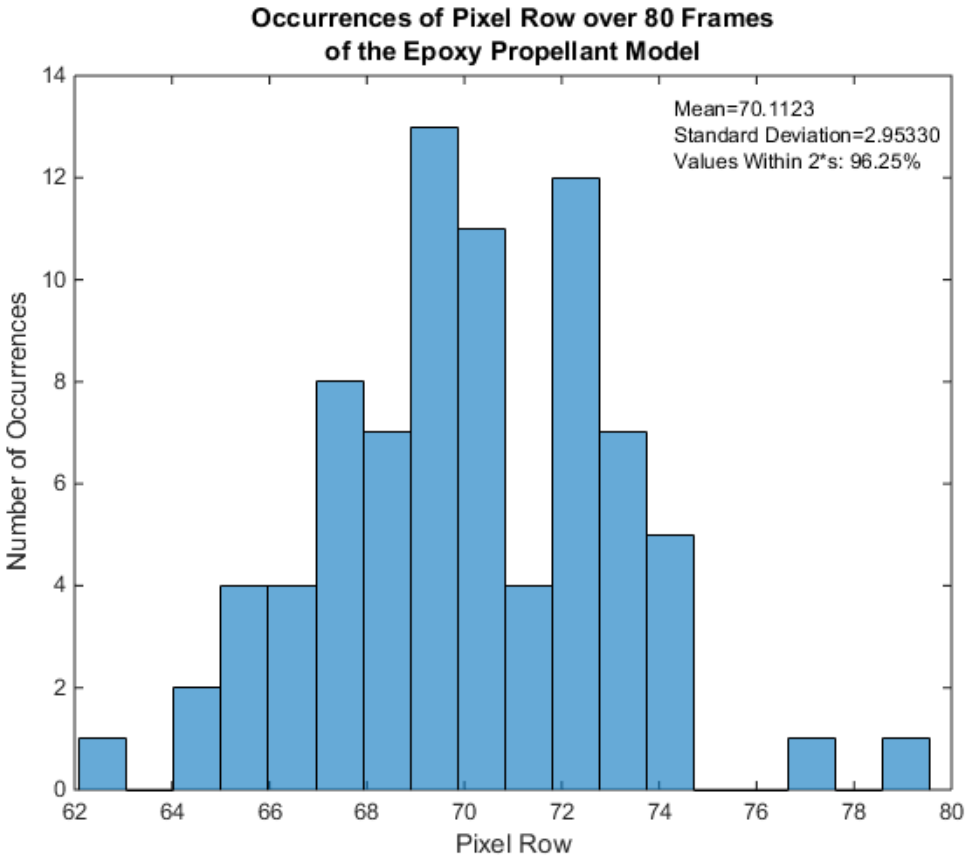


Figure 50: Histogram of 6-bit Epoxy Bowl Edge Locations

4.4 Discussion of Other Sources of Possible Error

4.4.1 Visibility of the True Bottom Surface

If the bottom edge of the burning surface was not visible in the videos, or if the visibility of the bottom edge of the burning surface changed throughout the burn, then there could be error in the burning rate measurements. While a bowl shape was visible in the videos, it may be that a gap in the propellant would be invisible if it was small enough. The bottom of the bowl in a horizontal plane is essentially just such a gap, so it might be invisible if the bowl shape is sufficiently steep.

Two separate studies were performed to assess the capability of the X-ray system to detect the bottom tip of the bowl-shaped burning surface in the propellant. One study was a computer-generated simulation of the X-ray images of the propellant samples. The other study was X-ray imaging of a physical model of the propellant sample made with epoxy to simulate the propellant. Neither of the studies indicated that the X-ray system was incapable of detecting the edge of the propellant burning surface.

Computer-Generated Simulation of X-ray Images

The MATLAB simulation code described in Chapter 3 was used to simulate the bowl shape of the burning surface of the propellant observed in the combustion experiments conducted at UAH. The code used a square-root geometric function to define the burning surface contour. A simulated image is shown in Figure 28 and Figure 53. The image was analyzed in MATLAB by applying the user's best guess of the location of the bottom of the burning surface and comparing the guess to the known location of the bottom tip of the burning surface to the known location written into the

code that created the simulated image. The equation used by the code to create the bowl-shaped burning surface was

$$R = 0.7\sqrt{0.4 - z}\,, \tag{4.8}$$

where R was the radius of the burning surface, and z was the depth from the top of the simulated propellant sample. By this equation, the bottom tip of the simulated burning surface must be located at $z = 0.4$ where $R = 0$. Therefore the lowest point on the burning surface is located 0.4 inch from the top of the propellant sample in Figure 51, which happens to be exactly in the center of the height of the sample.

A trace tool in MATLAB was used to locate the pixel of the user's best guess of the bottom of the burning surface. The trace tool is shown with its coordinates for the selected burning surface edge in Figure 51. The first row of pixels for which the intensity of the following rows of pixels did not change was row 809, so row 808 was selected as the bottom edge of the burning surface. In other words, the last pixel before the flat, horizontal low-level line on the intensity plot of the center column of pixels in Figure 52 was selected as the location of the bottom of the burning surface. There was a thin band of ten pixels above the top edge of the simulated propellant sample aluminum case, so the starting pixel row of the simulated propellant before ignition would have been row 11. Therefore, the user's estimate of the location of the bottom of the burning surface was 798 pixels below the top of the case. The true bottom edge of the burning surface is at 0.4 inch from the top, which is 800 pixels below the top of the case. The difference of 2 pixels between the guessed location and the actual location of the bottom of the burning surface corresponds to a distance of 0.001 inch. When scaled to the propellant burn tests, 0.001 inch corresponds to about half of a pixel. This discrepancy is a product of the

image resolution, and it would have been the same for all of the X-ray frames, so its effect would be canceled by sequential location measurements to find a rate.

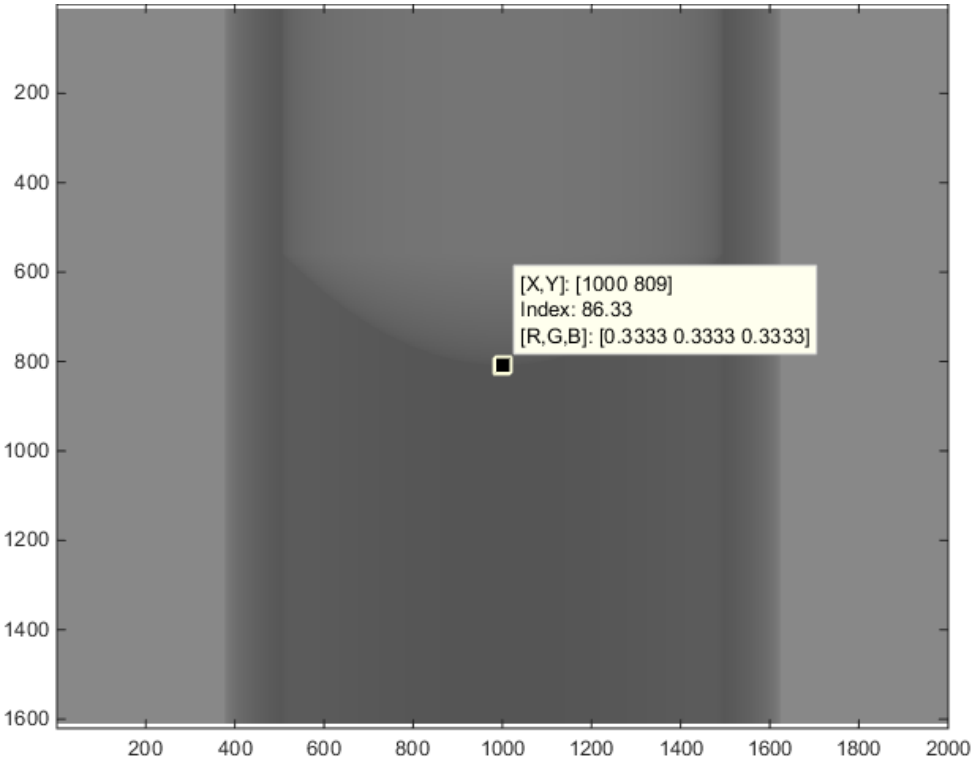


Figure 51: Trace Tool in MATLAB Computer-Generated X-ray Image Simulation

The burned region, bowl region, and unburned region of the simulated propellant are easily visible in the plot of color intensity of the center column of pixels from the simulated X-ray image in Figure 52. The burned region is the high horizontal line in the plot, the bowl region is the curve in the plot, and the unburned region is the lower horizontal line. The simulated burning surface bowl shape is most similar to the bowl shape observed near the end of the burn in the ESP combustion experiments rather than the beginning of the burn, because the simulated bowl shape is relatively wide since it has no top level flat burning surface. The bowl shape has widened completely across the propellant sample.

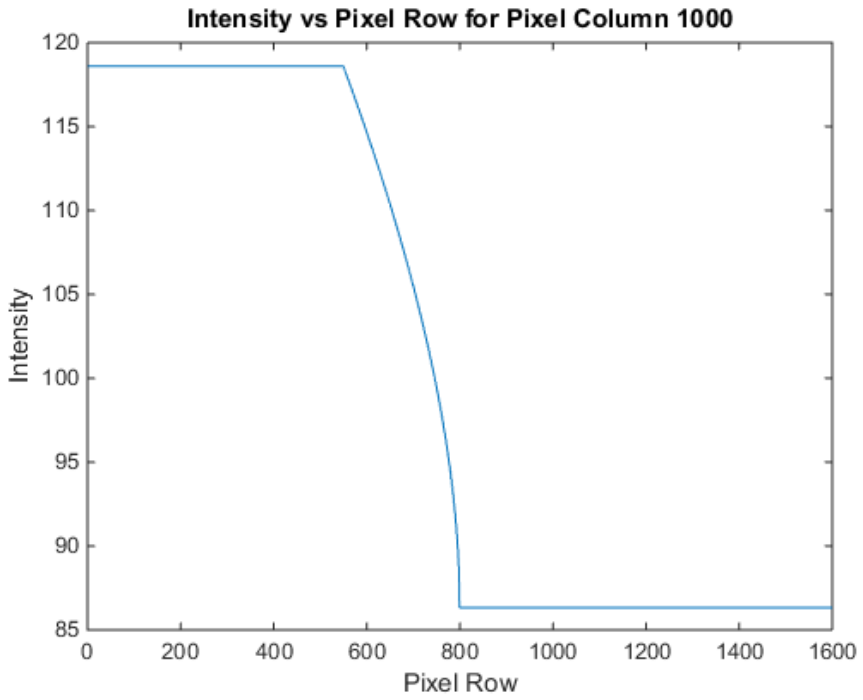


Figure 52: Intensity Plot for the Center Column of Pixels in the Computer-Generated X-ray Image Simulation of a Propellant Sample

Epoxy Propellant Model

The epoxy propellant was also used to investigate whether or not the true bottom edge of the bowl shape was visible in the X-ray images. It was considered that the X-rays may have been attenuated such that the bottom edge might appear to be higher in the X-ray images than the true edge. The distance from the bottom edge of the epoxy bowl to the top of the aluminum tube was measured with the depth gauge on a pair of calipers and found to be 0.25±0.01 inch. That distance was examined in the X-ray image in MATLAB as shown in Figure 53, and the user's best visual estimate of the row containing the bottom surface of the epoxy bowl was row 581. This estimate corresponds to the point where the curve flattens on the plot in Figure 48, which shows the color intensity plot from the 80 X-ray images of the motionless epoxy propellant model. The pixel row containing the top edge of the aluminum tube in the X-ray image was row 453, so, using

the prior row 452, the pixel distance from the top of the tube to the bottom of the bowl was 581-452=129 pixels. The X-ray images of the epoxy propellant model had a distance calibration of about 0.002 inch per pixel, so the pixel distance represented 0.258 inch from the top of the tube. That distance is within the caliper measurement of 0.25±0.01 inch.

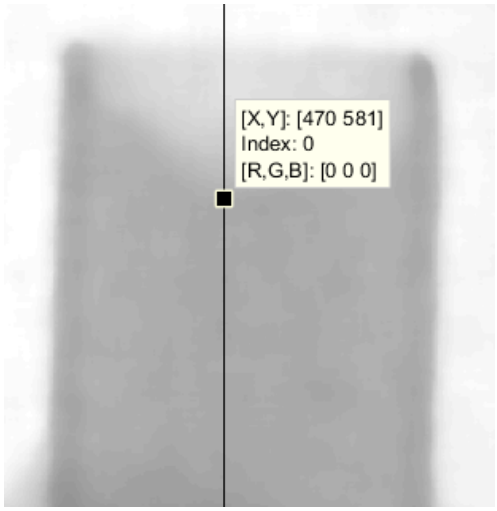


Figure 53: Trace Tool in MATLAB to Estimate the Pixel Row of the Bottom Edge of the Epoxy Propellant Model Bowl

4.4.2 **Pixel Bias**

If the X-ray detector had pixel biases, then the color intensities could have had error, which could have produced error in tracking the location of the target intensity value representing the burning surface. However, the use of the median filter caused any individual pixel biases to become unimportant, because the MATLAB code filtered the images before tracking the surface regression through the images. Some pixel bias was observed in the raw, unfiltered X-ray images as shown in the example X-ray image in Figure 54. The bias in the pixels is discussed in more detail in APPENDIX D.

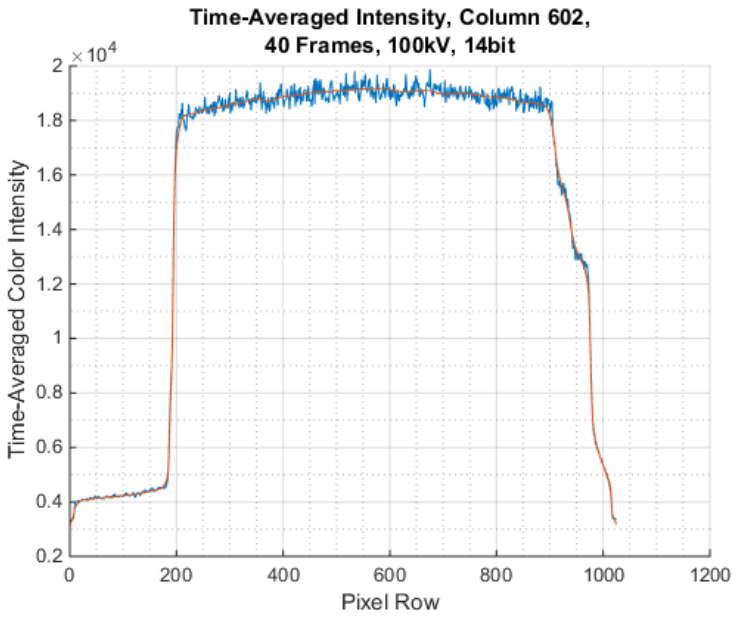


Figure 54: Example Plot Showing Patterned Noise through Bomb Windows, 100 kV, Gain 300, 14-bit, Column 602. Unfiltered (blue), Filtered (red)

4.4.3 **Filter Size**

The median filter, which removed noise from the X-ray images before the burning rate was calculated, used a square neighborhood of pixels. For each pixel in an image, the filter calculated the median gray value of a square neighborhood of pixels and assigned that median value to the pixel of interest. When a pixel was on or near the edge of an image, the square neighborhood of pixels was padded with zeros past the edge of the image. This caused some darkening around the edges of the full images, but the images were cropped down to a region of interest around the burning surface after the filter was applied, so the edge darkening was irrelevant. For most of the analyses, a neighborhood of 25-by-25 pixels was used. This filter size resulted in sufficiently smooth gradients of gray values without causing extreme blurring. Higher filter sizes would cause blurring and distortion of the propellant surface, and lower filter sizes resulted in noisy intensity plots that were difficult for the code to analyze.

4.4.4 **X-ray Geometry Effects**

Penumbra, caused by the use of a diverging X-ray beam and finite focal spot size, could cause blurring of X-ray images. Penumbra is discussed in more detail in the Fundamentals of X-ray Imaging section of Chapter 1. In these experiments, the focal spot size was one millimeter, which is quite small, and the sample was small enough and far enough from the source that there was no clear observation of penumbra in the X-ray videos. For what little amount of penumbra blurring may have been present, the effect of blurring the bottom edge of the burning surface is accounted for in other error studies previously discussed.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Summary of Combustion Experiment Results

Two burn tests were performed with ignition configuration ESP samples according to Standard Operating Procedure PRC-SOP-HPL-010-0-B. The results of the two tests are summarized in Table 6 from the Method 2 projected-slope intersection tracking method. The masses shown in the table were provided by the manufacturer. Both samples contained formulation HIPEP 501a propellant from the same batch, and they were packaged by DSSP on December 6, 2013. Test UAH04A01 was conducted on March 2, 2015, and test UAH04A02 was conducted on March 4, 2015.

Table 6: Summary of Combustion Experiment Method 2 Final Results

Serial Number	Propellant Mass	Average Pressure	Ignition Voltage	Average Burning Rate	Bounds for 95% confidence	Percent of Bounds
UAH04A01	3.9115 g	407 psi ±3.6%	200 V	0.0839 in/s	±0.00240 in/s	±2.86%
UAH04A02	4.0358 g	410 psi ±3.9%	200 V	0.0882 in/s	±0.00268 in/s	±3.04%

5.2 Chronological Experiment Descriptions

5.2.1 UAH04A01 Raw Data

Sample UAH04A01 was an ignition sample with a thick-wall tube. It successfully burned from top to bottom. When power was supplied, the propellant burned in all

directions from the bare metal wire inserted into the top surface of the propellant. This caused the burning surface to form a bowl shape. The bowl shape widened and became increasingly flatter as the surface regressed. The mass of the propellant was 3.9115 g. The burn time was approximately 7.367 seconds as determined from the X-ray videos. The average pressure of the test was found to be 407 psi ±3.6% with 95% confidence as shown in APPENDIX C. This was the combined mean of data from two pressure transducers from a range of three seconds in the middle of the burning time, because the pressure was very stable. The pressure and power data are shown graphically in Figure 57. Pictures of the sample are shown in Figure 55.

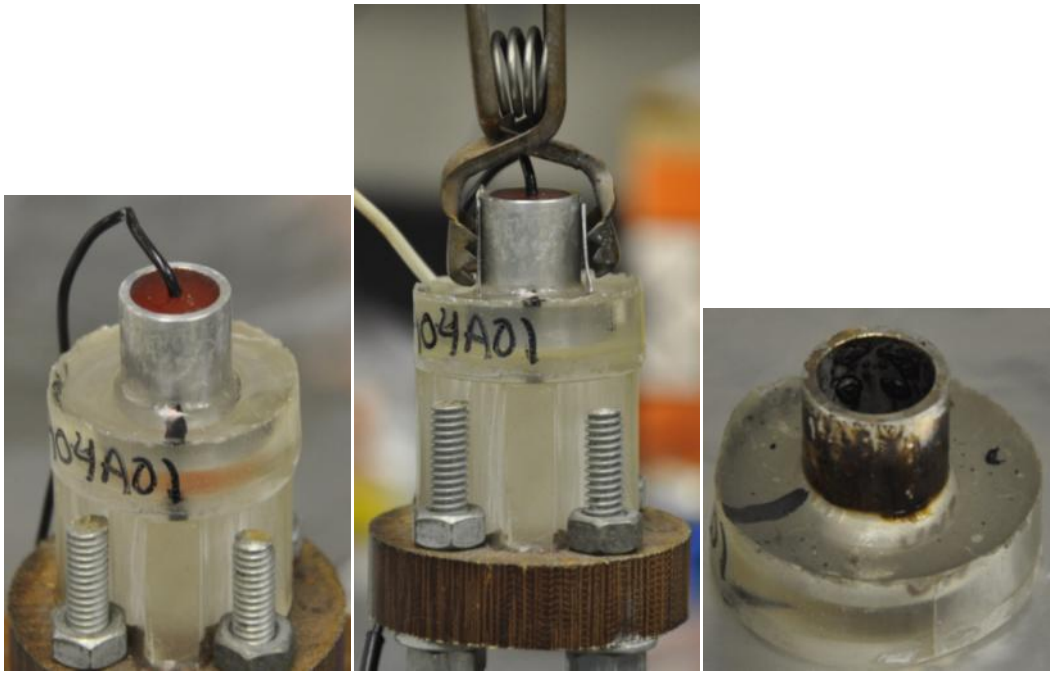


Figure 55: UAH04A01 Experiment Pictures:
Pre-test (left and center), Post-test (right)

Raw frames of the X-ray video are shown in Figure 56. The X-ray frames were captured with a bit depth of 14, meaning that there were 16384 possible gray values.

Although the bit depth was 14, the images were saved in 16-bit format because it is a more standard format, but this did not increase the resolution.

Figure 56: UAH04A01 X-ray Video Frame Series

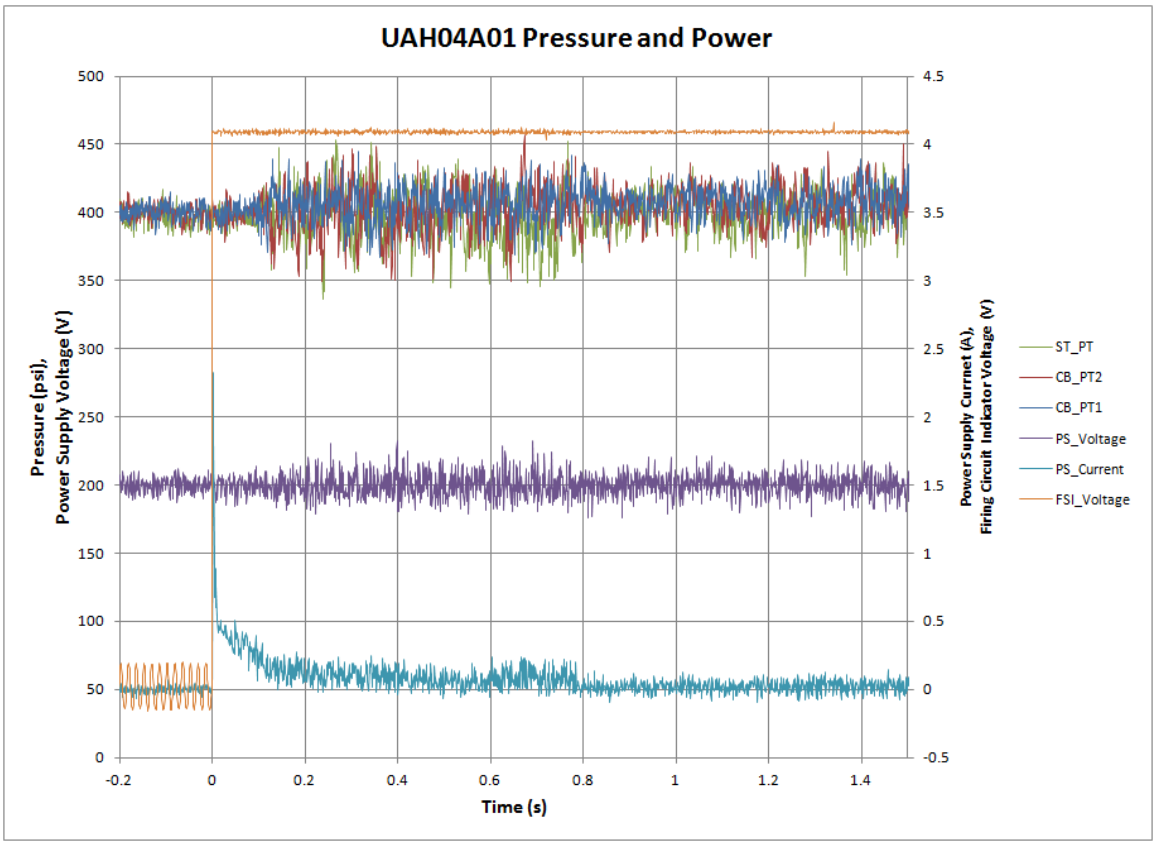


Figure 57: UAH04A01 Pressure and Power Data

5.2.2 UAH04A01 Method 1 Results

The burning rate calculated with the Method 1 analysis was 0.0847 in/s. The uncertainty was calculated as ±0.0071 in/s, which represented ±8.42% of the burning rate a 95% confidence level. An example picture of an X-ray frame from this test that was filtered and auto-scaled with a MATLAB gray-scaling function is shown in Figure 58. Pixel column 602 was used to track the regression, and it has been blackened in the image. A plot of the burning surfaces with a least squares line fit for the preliminary analysis is shown in Figure 60.

A variation of the MATLAB edge tracking code was used to generate a plot of the contour of the burning surface across a range of pixel columns within the images for a series of frames of this test. The contour for each selected frame was identified by

running the constant target intensity technique across each pixel column for the range of selected pixel columns indicated by the two vertical black lines in each frame of Figure 59.

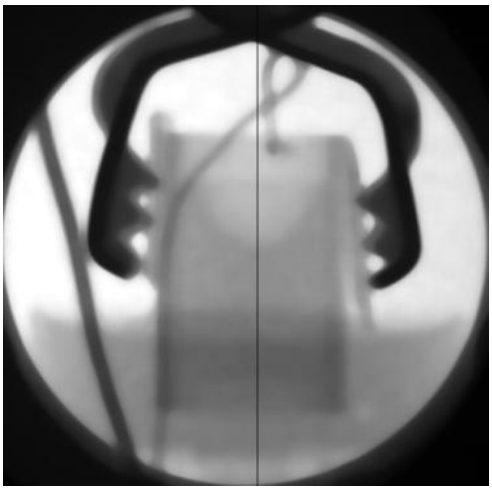


Figure 58: UAH04A01 Example Filtered and Auto-Scaled Video Frame

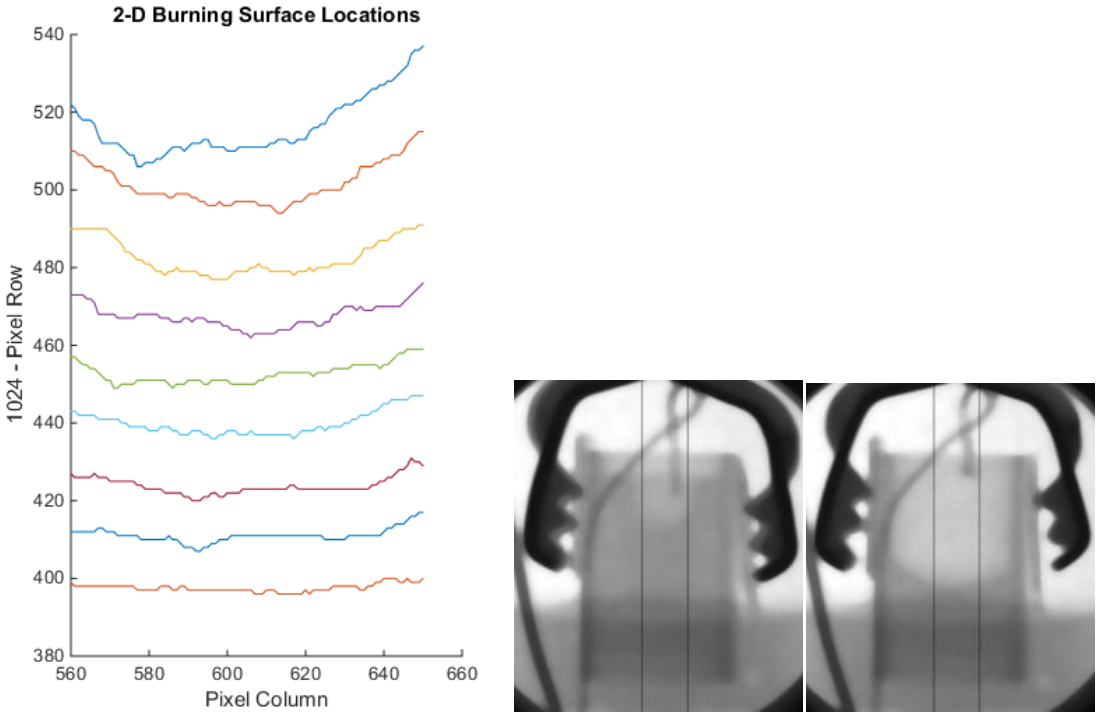


Figure 59: UAH04A01 Method 1 2-D Burning Surface Contour Detection Plot (left) with First (middle) and Last (right) Cropped Frames Used Spanning 3 Seconds

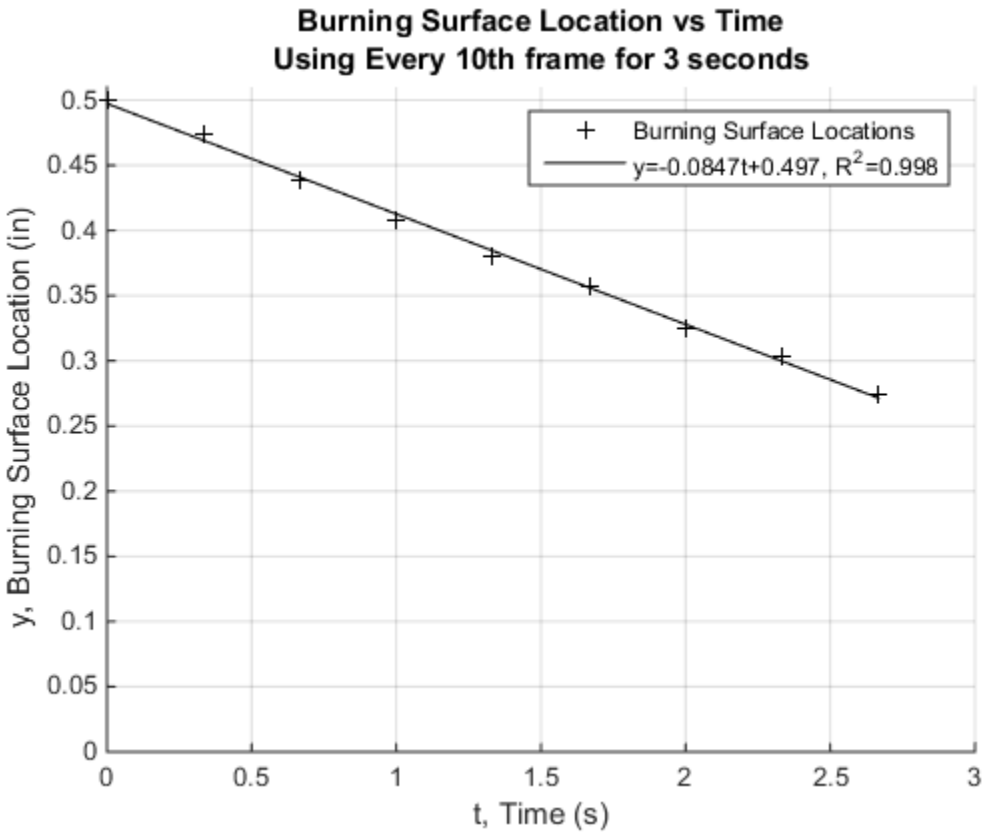


Figure 60: UAH04A01 Method 1 Burning Surface Locations with Least Squares Line Fit

5.2.3 UAH04A01 Method 2 Results

The mean burning rate from the MATLAB edge tracking code was found to be 0.0839 in/s. The quantity of two standard deviations was found to be 0.00240 in/s, which was 2.86% of the mean burning rate. This uncertainty bounded 95.46% of the burning rates calculated in the Monte Carlo analysis. Therefore it was appropriate to describe the burning rate distribution as a normal distribution and apply a 95% level of confidence to the burning rate uncertainty. A histogram of the burning rate distribution is shown in Figure 66. The pixel column selected as the center column of pixels for this test was

column 602 in the full frames, which became column 100 in the cropped frames. The scale factor was 501.7 pixels-per-inch (ppi).

As in the preliminary Method 1 analysis, the burning surface contours for a range of pixel columns were determined in the final analysis and plotted in Figure 61. The range of pixel columns plotted is indicated by the two black lines in each cropped frame of Figure 61. In the plot of the cropped images, pixel column 100 is the same as pixel column 602 from the full frames, and it is the exact center column of the cropped images.

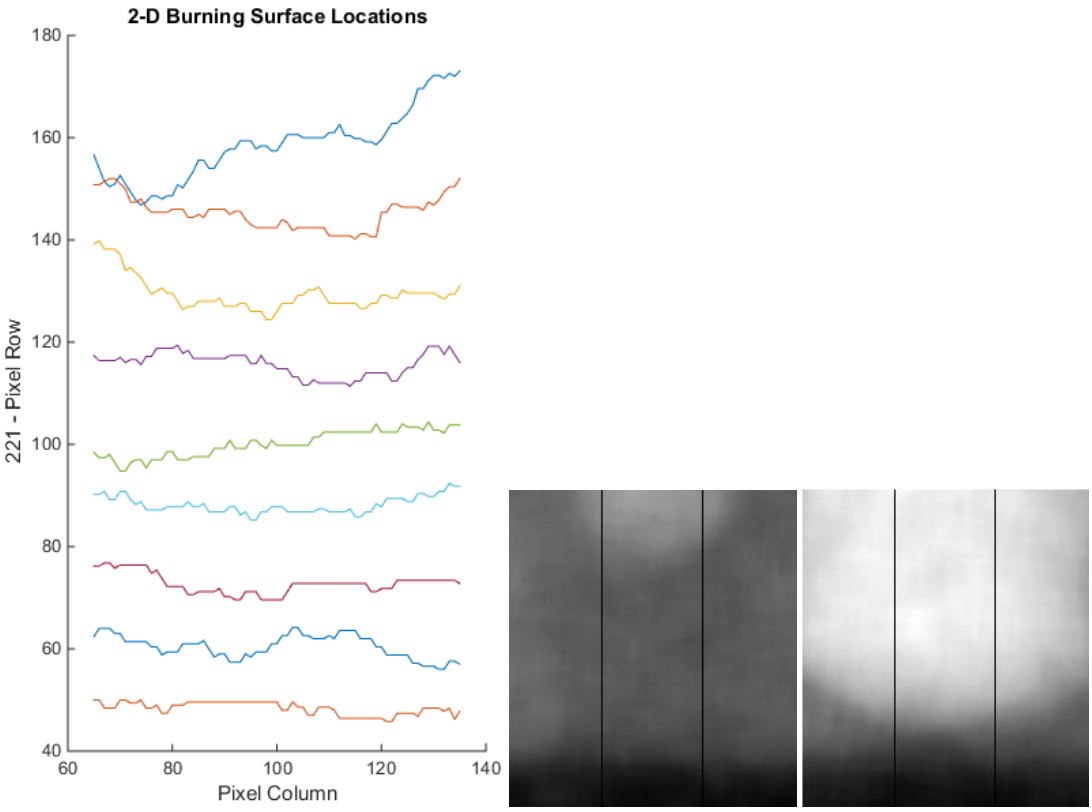


Figure 61: UAH04A01 Method 2 2-D Burning Surface Contour Detection Example Plot with First (middle) and Last (right) Cropped Frames Used Spanning 3 Seconds

Example intensity plots of multiple frames of the center column of pixels are shown in Figure 62 and Figure 63. An example plot of the location of the burning surface versus time with a least squares line fit is given in Figure 64. An example of a series of

cropped frames showing the burning surface location marked by the MATLAB edge tracking code is given in Figure 65.

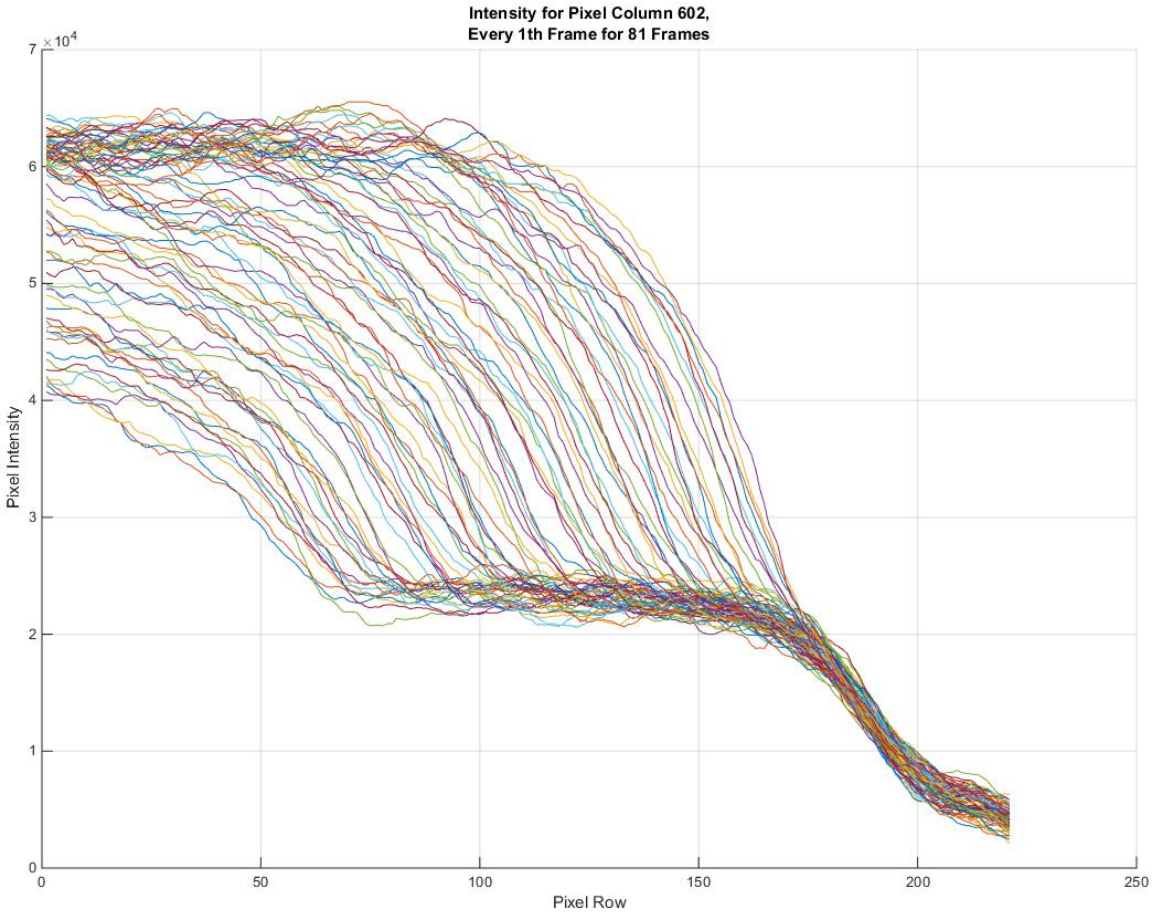


Figure 62: UAH04A01 Method 2 Example Intensity Plot over All Cropped Frames

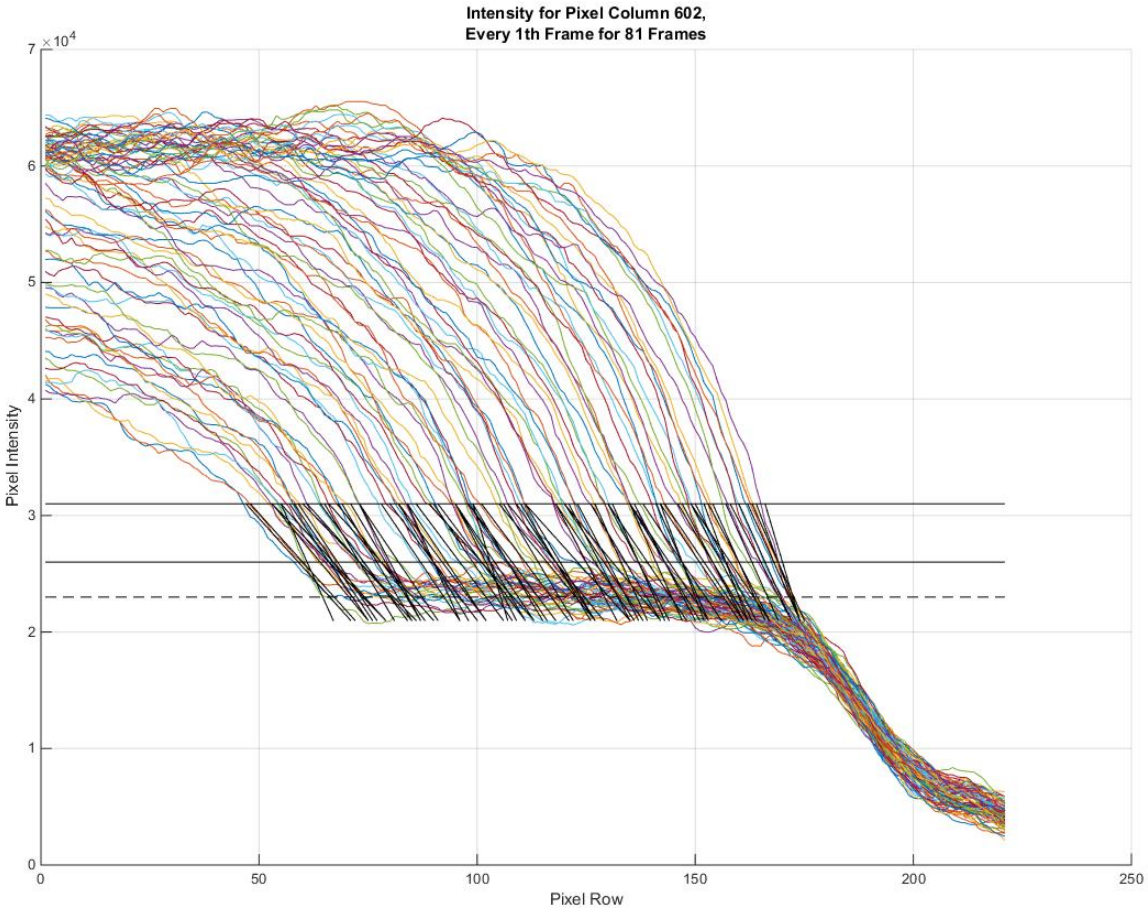


Figure 63: UAH04A01 Method 2 Example Intensity Plot over All Cropped Frames Showing Slope-Projections and Intensity Threshold Lines

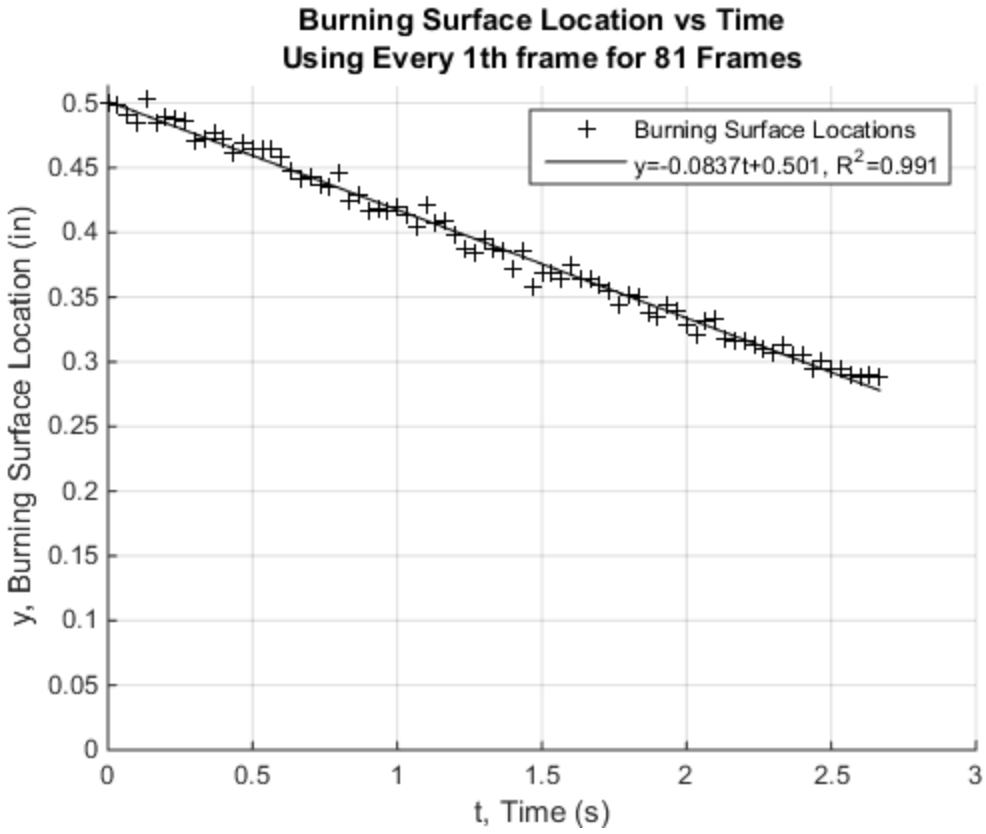


Figure 64: UAH04A01 Method 2 Example Plot of Least Squares Line Fit with Burning Surface Locations vs Time

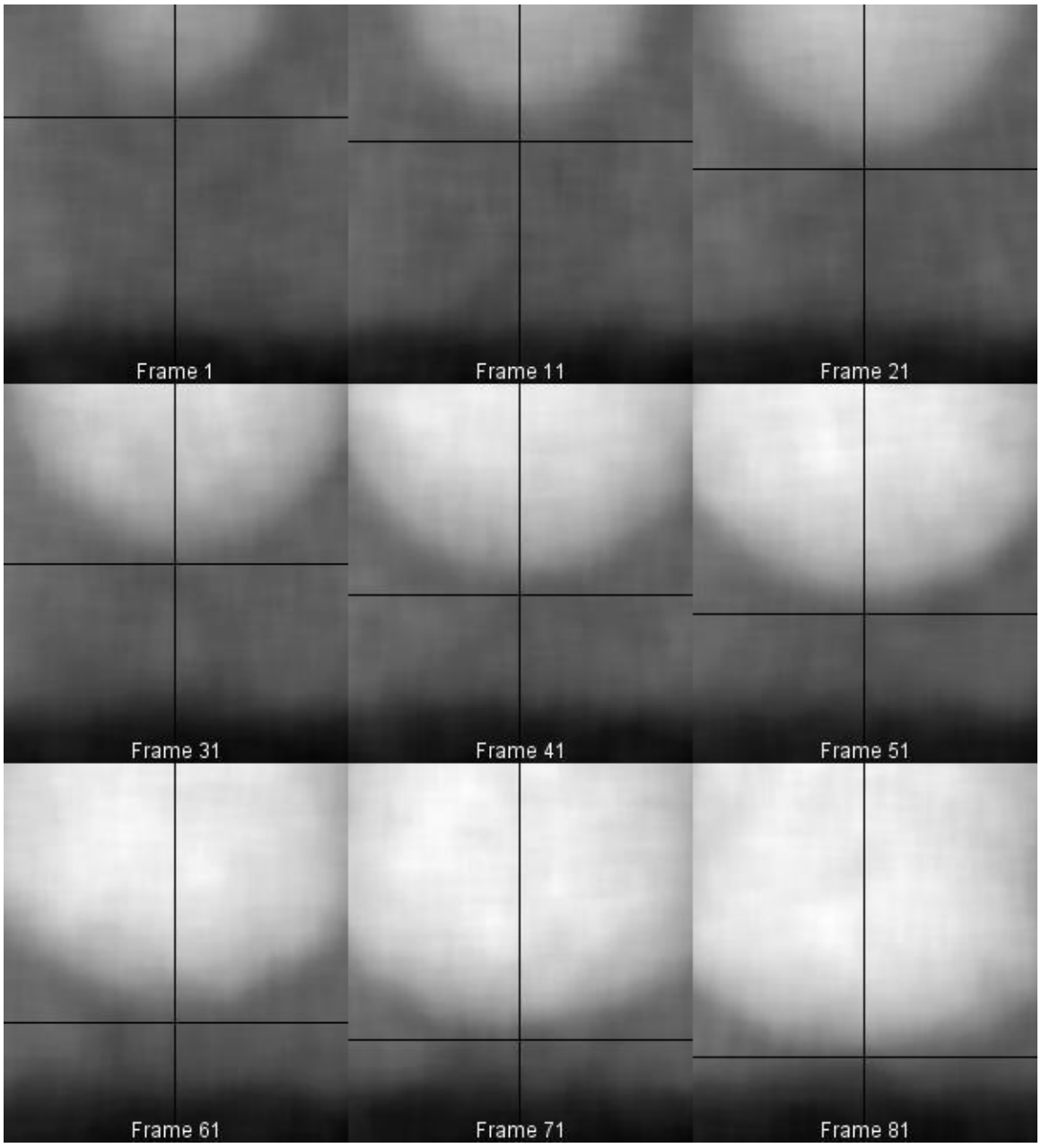


Figure 65: UAH04A01 Method 2 Example Series of Cropped Frames Showing the Burning Surface Locations Marked by the MATLAB Edge Tracking Code

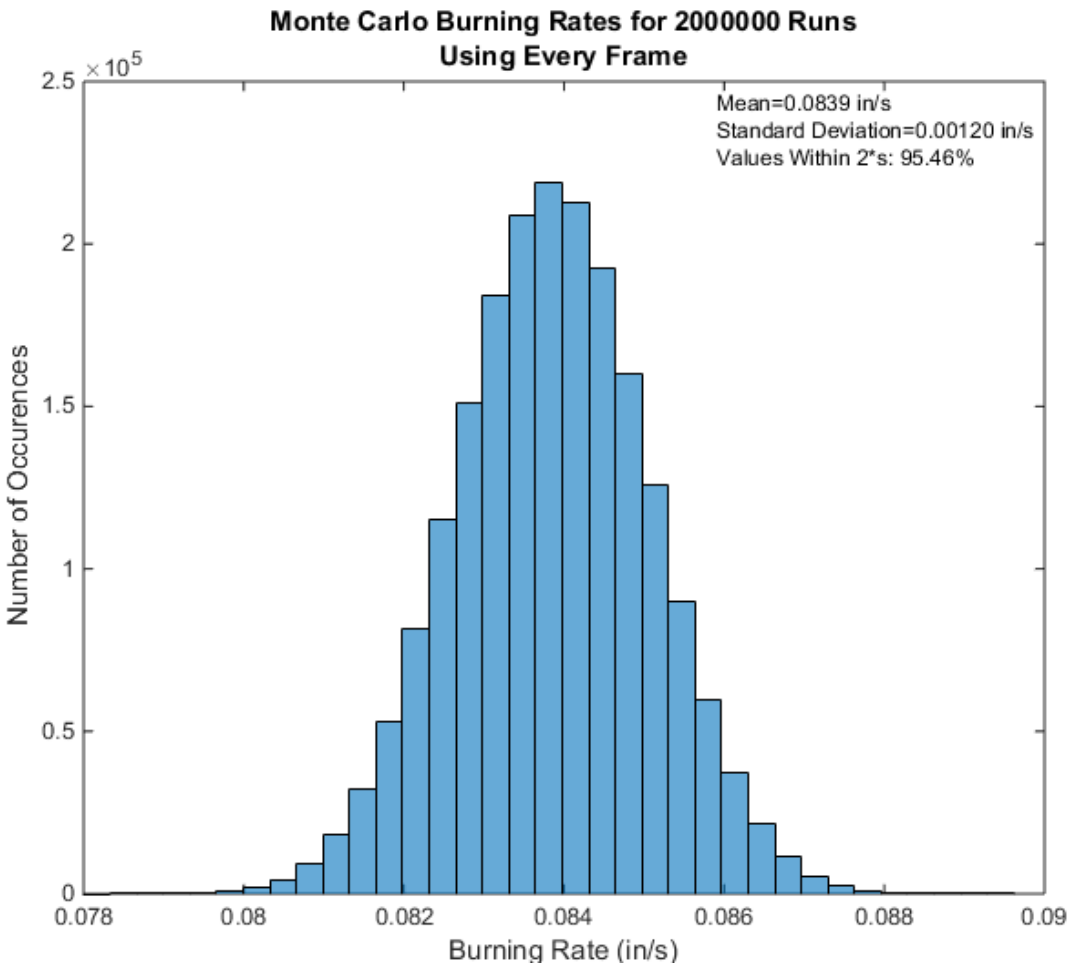


Figure 66: UAH04A01 Method 2 Burning Rate Distribution from Monte Carlo Analysis

5.2.4 **UAH04A02 Raw Data**

Sample UAH04A02 was the second ignition configuration sample tested. Like the first ignition sample, this sample burned from top to bottom with a widening bowl shape, but a small sliver of propellant jumped up from the epoxy base of the sample near the end of the burn. The MATLAB code analyzed the burning rate in the time before the propellant ejected. The propellant mass was 4.0358 g. The burn time was approximately 6.9 seconds as determined from the X-ray videos. The average pressure was found to be 410 psi ±3.9% with 95% confidence as shown in APPENDIX C. This was the combined mean of data from two pressure transducers from a range of three seconds in the middle of the burning time, because the pressure was very stable. The pressure and power data are shown graphically in Figure 69. Pictures of the sample are shown in Figure 67.

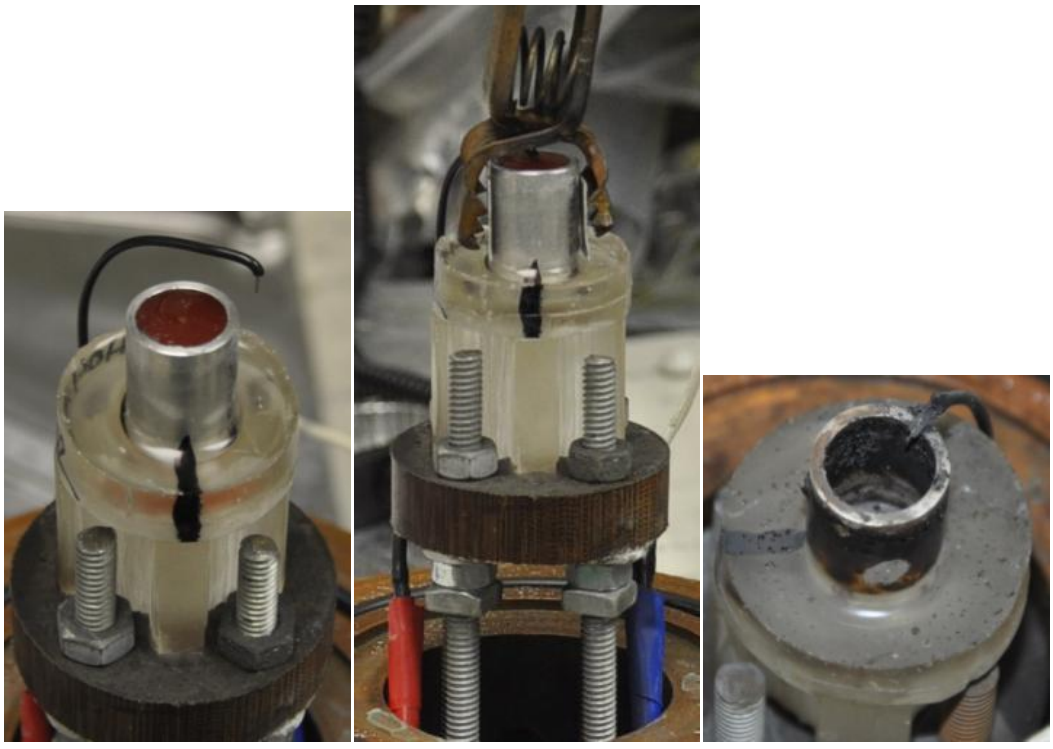


Figure 67: UAH04A02 Experiment Pictures:
Pre-test (left and center), Post-test (right)

Raw frames of the X-ray video are shown in Figure 68. The X-ray video for this test was captured with a color bit depth of 6, but the images were saved in 8-bit format. This was a reduction in bit depth from the value of 14 in the first test. The reduction was chosen due to the tendency of the recording software to occasionally fill its buffer and skip frames while operating in 14-bit mode. The highest possible number of gray values for 6-bit gray-scale is 64. The bit depth could not be assigned a value other than 6 or 14.

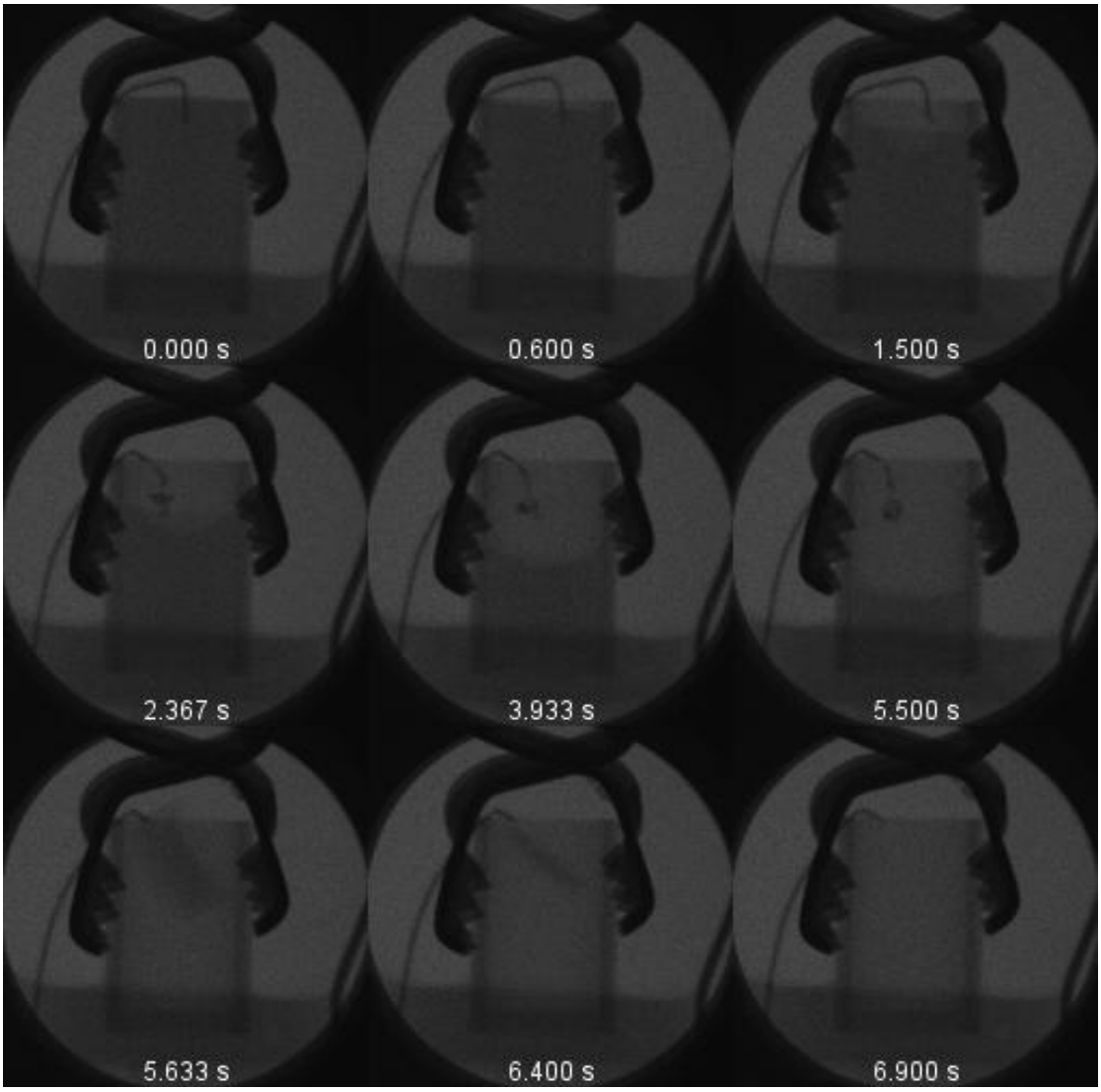


Figure 68: UAH04A02 X-ray Video Frames

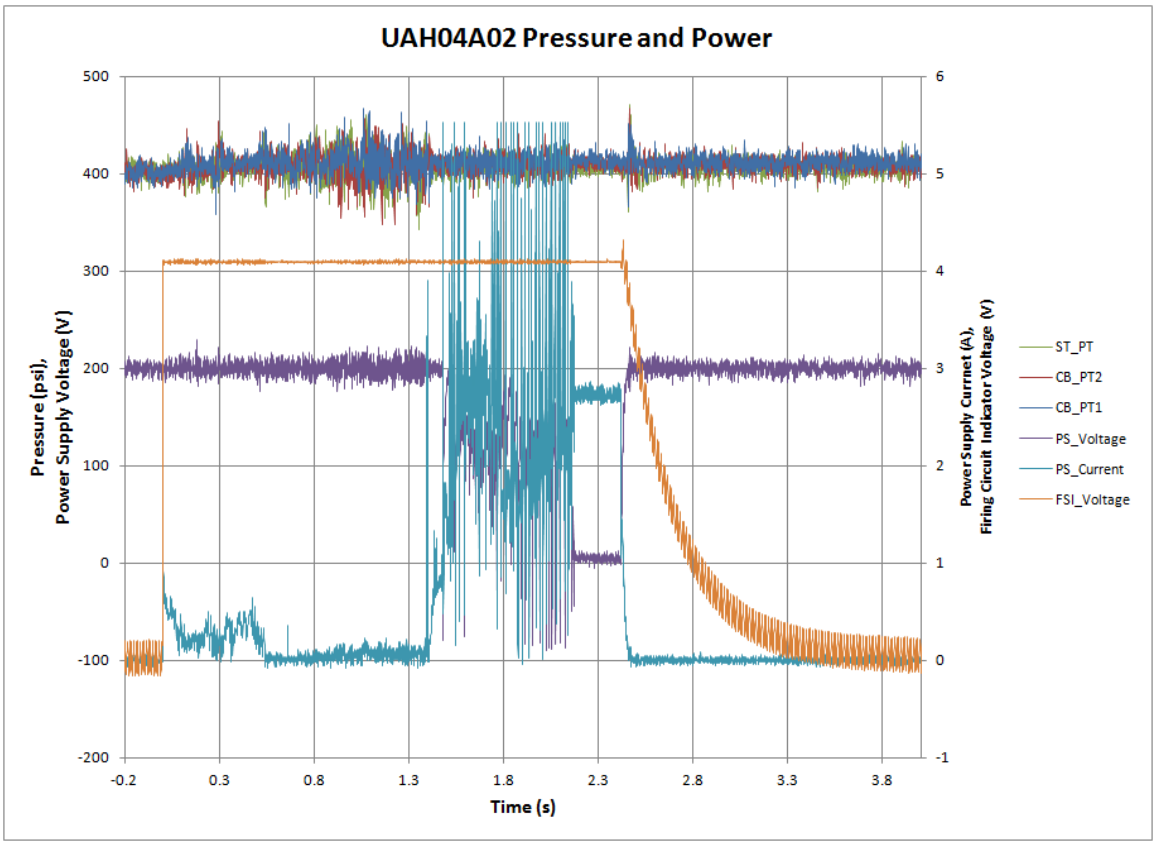


Figure 69: UAH04A02 Pressure and Power Data

5.2.5 UAH04A02 Method 1 Results

The burning rate calculated in the Method 1 analysis was 0.0913 in/s. The uncertainty was calculated as ±0.0060 in/s, which represented ±6.55% of the burning rate. An example picture of an X-ray frame from this test that was filtered and auto-scaled with a MATLAB gray-scaling function is shown in Figure 70. Pixel column 575 was used to track the regression, and it has been blackened in the image. The image shows a line on the column of pixels used to track the regression. A plot of the 2-D burning surface contours identified in the preliminary analysis for this test is shown in Figure 71 with frames that have black lines indicating the range of pixel columns plotted.

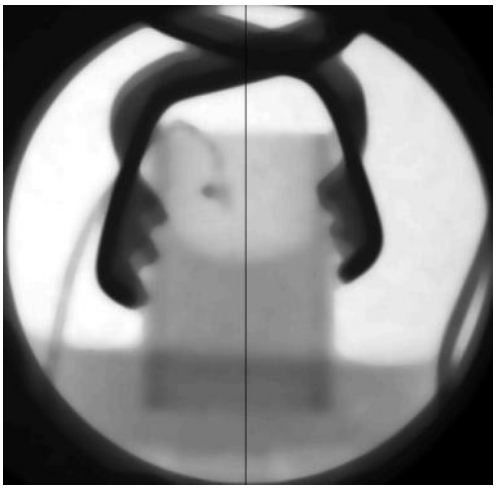


Figure 70: UAH04A02 Example Filtered and Auto-Scaled Video Frame

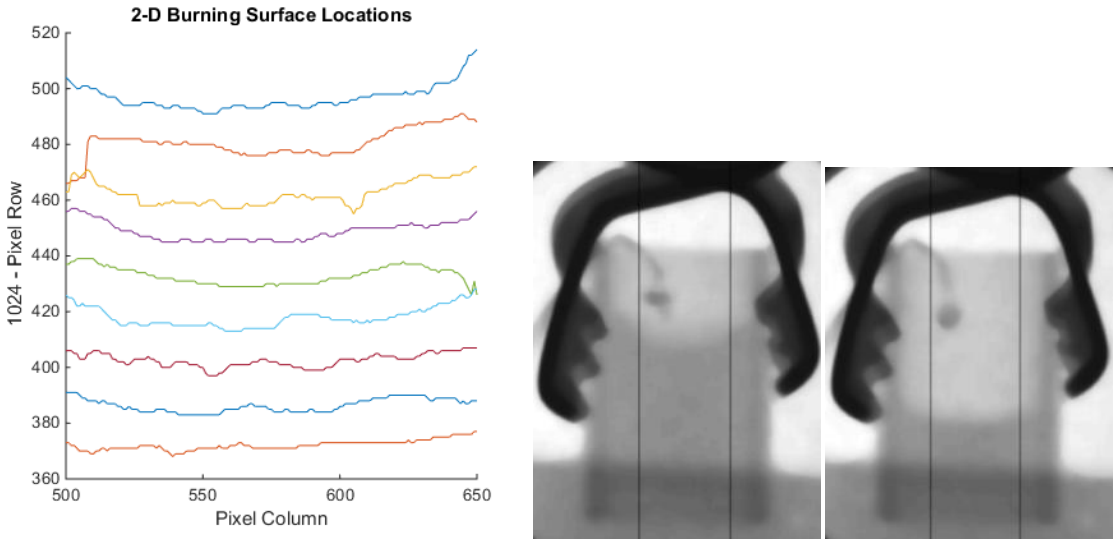


Figure 71: UAH04A02 Method 1 2-D Burning Surface Contour Detection Plot (left) with First (middle) and Last (right) Cropped Frames Used Spanning 3 Seconds

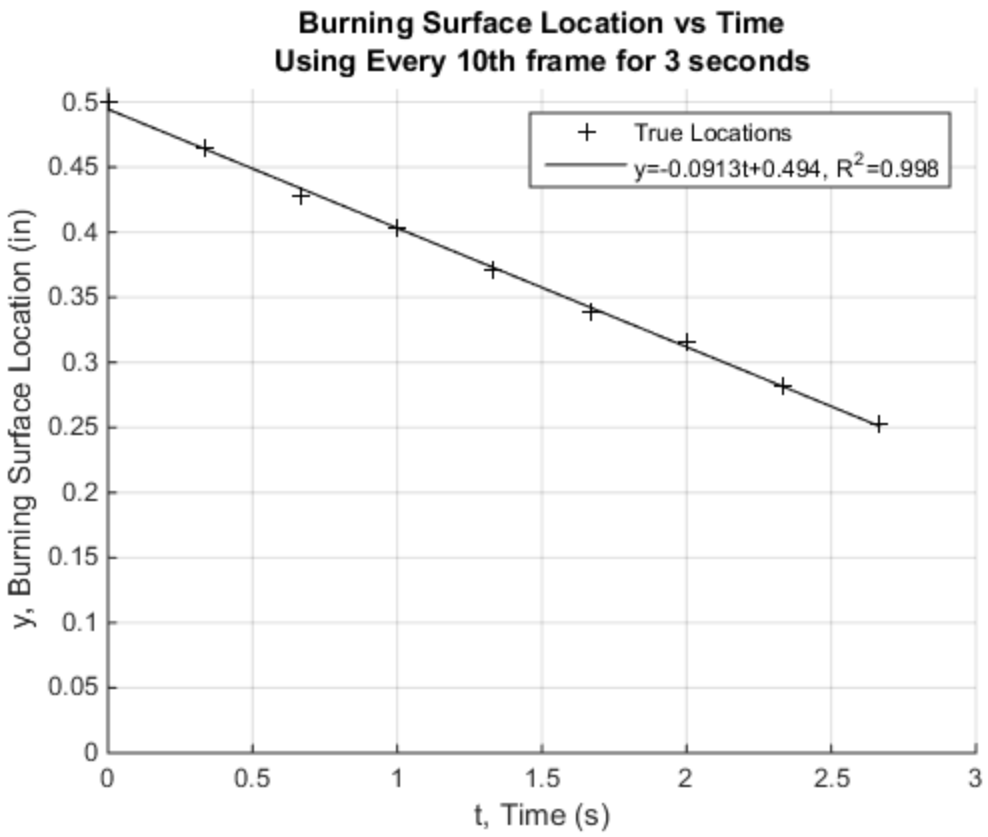


Figure 72: UAH04A02 Method 1 Burning Surface Locations with Least Squares Line Fit

5.2.6 UAH04A02 Method 2 Results

The mean burning rate was found to be 0.0882 in/s. The quantity of two standard deviations was found to be 0.00268 in/s, which was 3.04% of the mean burning rate. This uncertainty bounded 95.59% of the burning rates calculated in the Monte Carlo analysis. Therefore it was appropriate to describe the burning rate distribution as a normal distribution and apply a 95% level of confidence to the burning rate uncertainty. A histogram of the burning rate distribution is shown Figure 78. The pixel column selected as the center column of pixels for this test was column 575 in the full frames, which became column 100 in the cropped frames. The scale factor was 502.7 pixels-per-inch (ppi).

As in the first test, the burning surface contours for a range of pixel columns were determined in the final analysis for this test and plotted in Figure 73. The range of pixel columns plotted is indicated by the two black lines in each cropped frame of Figure 61. In the plot of the cropped images, pixel column 100 is the same as pixel column 575 from the full frames, and it is the exact center column of the cropped images.

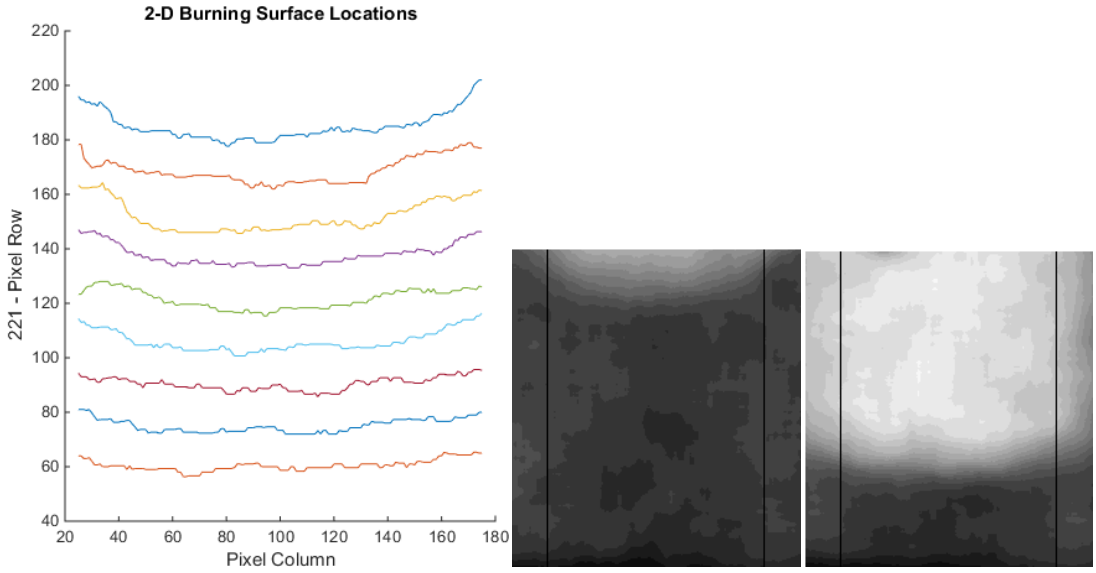


Figure 73: UAH04A02 Method 2 2-D Burning Surface Contour Detection Example Plot with First (middle) and Last (right) Cropped Frames Used Spanning 3 Seconds

Example intensity plots of multiple frames of the center column of pixels are shown in Figure 74 and Figure 75. An example plot of the locations of the burning surface versus time with a least squares line fit is given in Figure 76. An example of a series of cropped frames showing the burning surface location marked by the MATLAB edge tracking code is given in Figure 77.

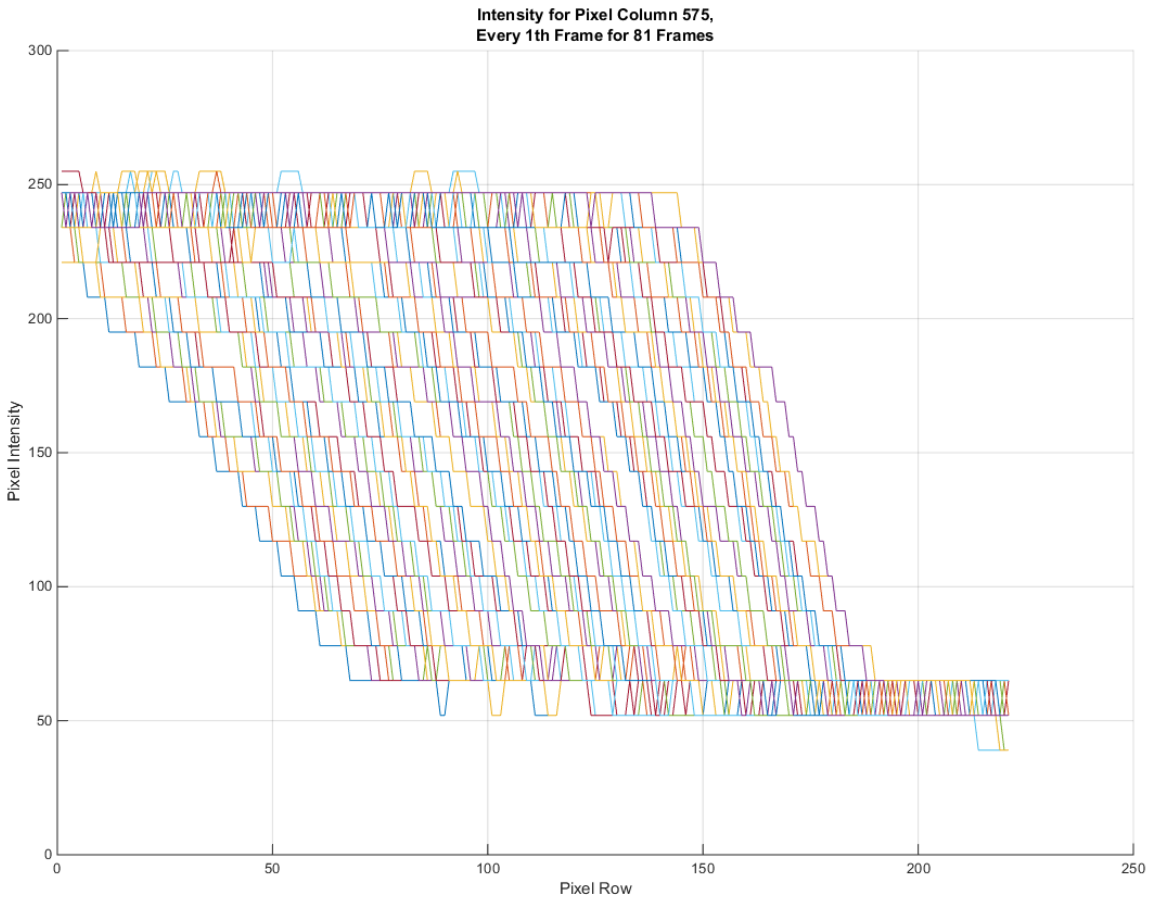


Figure 74: UAH04A02 Method 2 Example Intensity Plot over All Cropped Frames

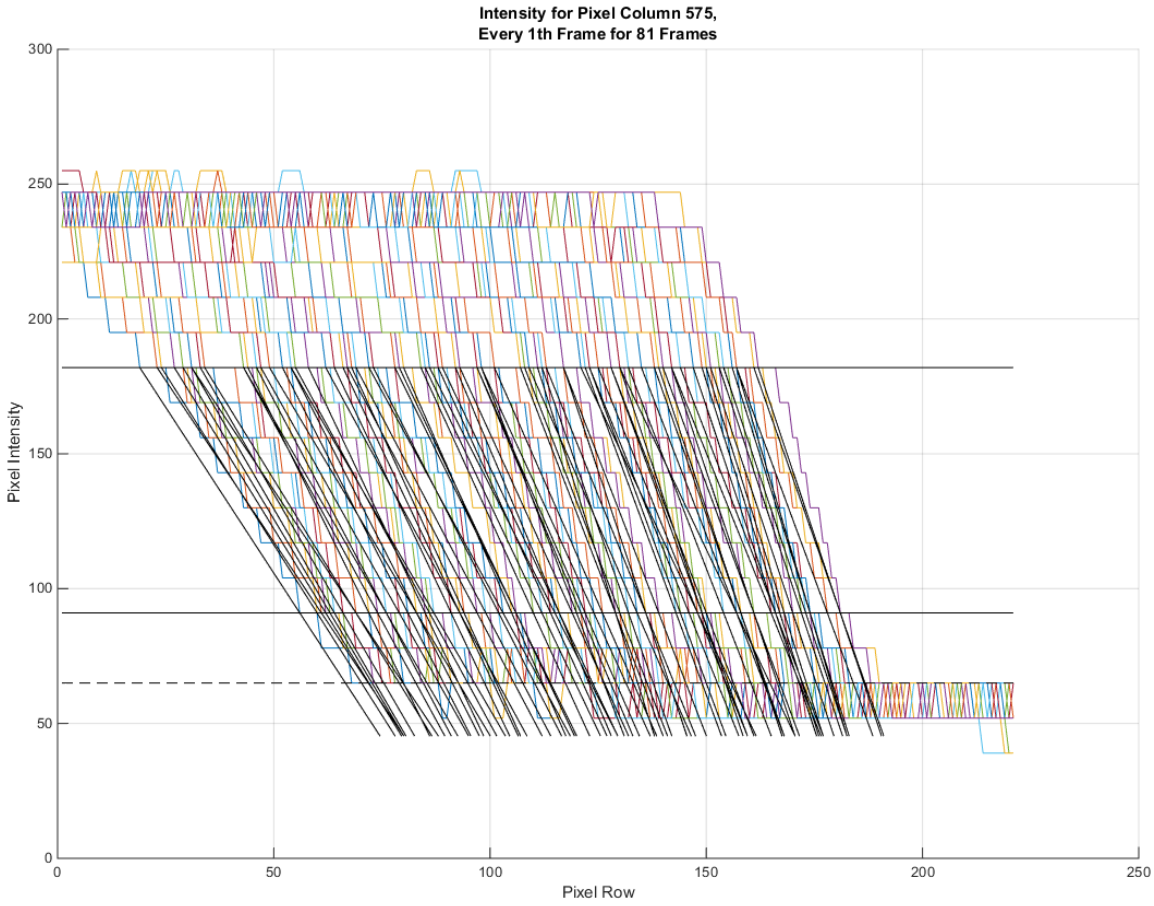


Figure 75: UAH04A02 Method 2 Example Intensity Plot over All Cropped Frames Showing Slope-Projections and Intensity Threshold Lines

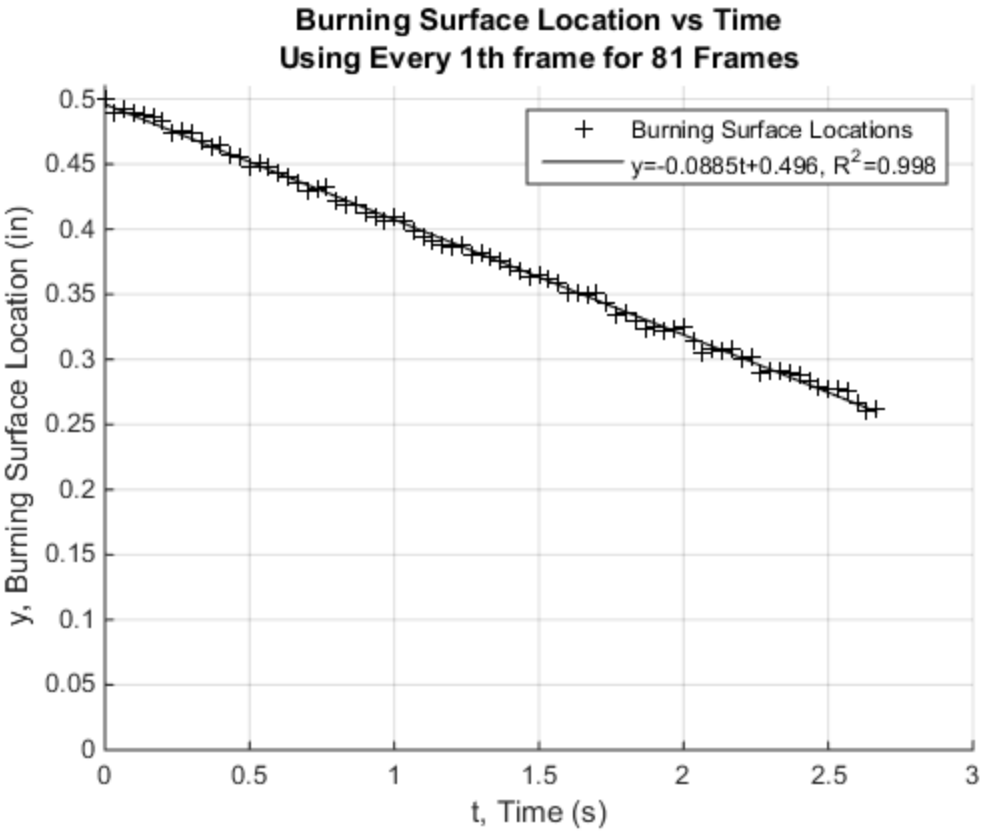


Figure 76: UAH04A02 Method 2 Example Plot of Least Squares Line Fit with Burning Surface Locations vs Time

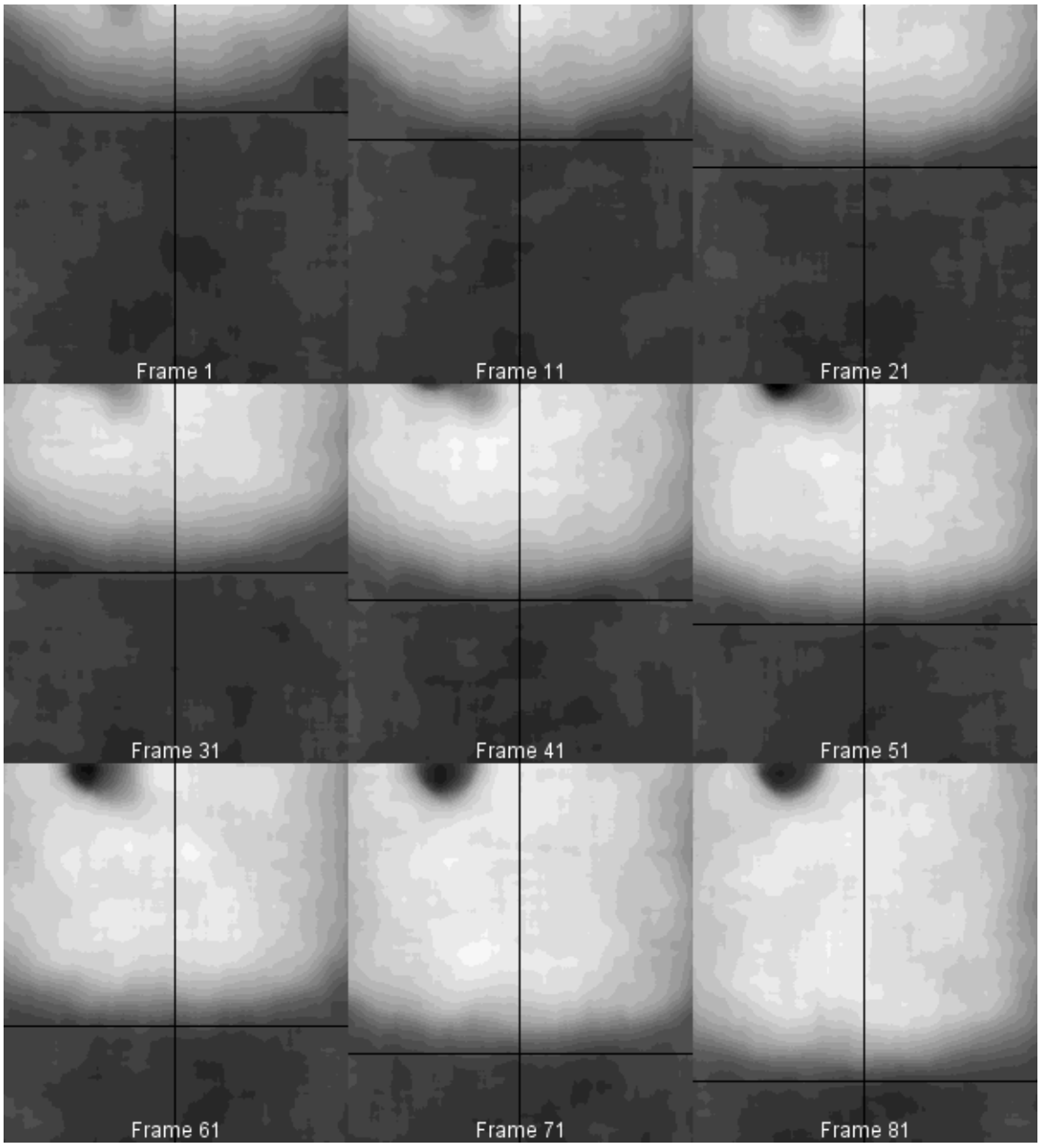


Figure 77: UAH04A02 Method 2 Example Series of Cropped Frames Showing the Burning Surface Locations Marked by the MATLAB Edge Tracking Code

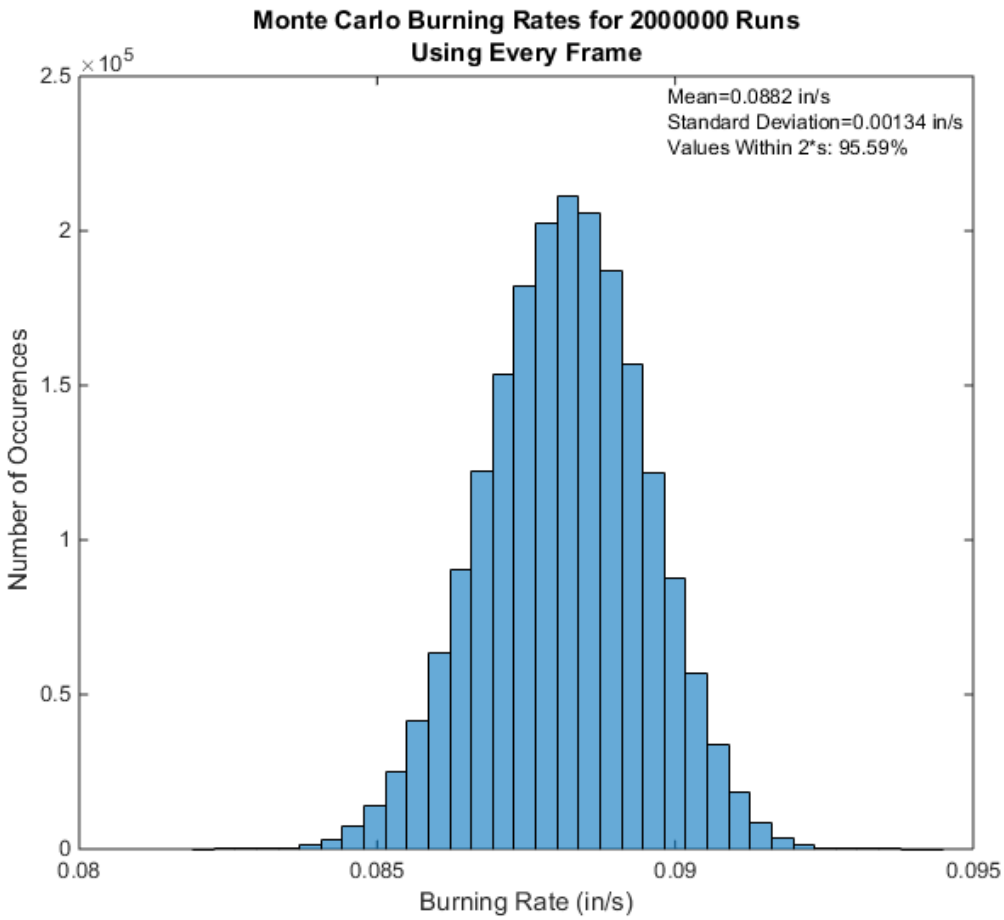


Figure 78: UAH04A02 Method 2 Burning Rate Distribution from Monte Carlo Analysis

5.3 Discussion and Comparison of Tracking Methods

In both tests, the final Method 2 analysis produced a lower burning rate than the preliminary Method 1 analysis. This was to be expected because the Method 2 analysis accounted for the intensity gradient bias due to bowl widening that would theoretically cause artificially high burning rate measurements in the Method 1 analysis, whereas the Method 1 analysis merely attempted to capture the bias error within its uncertainty bounds. In the Method 1 analysis, the selection of center pixel column was not varied, and only every 10th frame was used. All frames were used in the Method 2 analysis. Table 7 shows a comparison of the final Method 2 burning rate results with the results of the Method 1 analysis. For both analyses, the uncertainty bounds from each test were overlapping.

Table 7: Comparison of Final Results with Preliminary Results

Analysis	Average Pressure	Average Burning Rate	Bounds for 95% confidence	Percent of Bounds
UAH04A01 Method 1	407 psi	0.0847 in/s	±0.0071 in/s	±8.42%
UAH04A01 Method 2	407 psi	0.0839 in/s	±0.00240 in/s	±2.86%
UAH04A02 Method 1	410 psi	0.0913 in/s	±0.0060 in/s	±6.55%
UAH04A02 Method 2	410 psi	0.0882 in/s	±0.00268 in/s	±3.04%

5.4 Comparison of X-ray Results to Other Results

The burning rate results from the two tests have been plotted with data from burn strip experiments conducted in the past at UAH as well as burn strip experiments conducted by the ESP manufacturer, DSSP, as shown in Figure 79. The burning rates of the burn strips tested by UAH were determined optically with a high-speed camera viewing the burn strips through a window in a combustion bomb. The burning rates of the burn strips tested by DSSP were determined by analysis of pressure measurements and optics. The burning rates obtained from the X-ray measurements are lower than those

obtained from the burn strip tests, although no uncertainty was known to be reported for the burn strips. This discrepancy was not expected, because the X-ray propellant samples had the same propellant formulation as the burn strips. However, several factors have been identified which could cause differences. The propellant shape in the burn strips was smaller than the X-ray samples, so they may have experienced different temperature effects from different flows of combustion products, and the grain shape could have set differently. It is also noted that, while the formulation was the same, the propellant batch used in the X-ray samples was not the same propellant batch used in the burn strips. The X-ray samples were over one year old when they were burned, but the burst strips were not as old when they were burned. The UAH burn strips were burned in a purged environment which could have caused interaction with the burning surface.

Table 8: Burning Rate Comparison

	Burning Rate (in/s)	
Pressure (psi)	**DSSP Burn Strips (Batch 1)**	**DSSP Burn Strips (Batch 2)**
250	0.09	-
325	0.115	-
425	0.13	0.14
525	0.18	-
625	0.3	-
825	0.42	0.405
950	0.425	-
1100	0.495	-
1200	0.53	-
1300	0.525	0.505

	Burning Rate (in/s)	
Pressure (psi)	**UAH Burn Strips (Batch 3)**	**UAH X-ray Samples (Batch 4)**
400	0.133	-
400	0.134	-
400	0.152	-
407	-	0.0839
410	-	0.0882
800	0.394	-
800	0.441	-

Curve fits of the form $r=aP^n$ have been applied to the burn strip data on the plot. The DSSP burn strip data appears to have a slope break around 600 psi, so a curve fit was applied to the low-pressure points and to the high-pressure points separately from each other in addition to a curve fit applied to all of the DSSP data together. The UAH data is

not grouped in any curve fits with the DSSP data, and the X-ray data is not included in any curve fits. The X-ray samples gave lower burning rates than the burn strips, which might be because they were from different propellant batches. The values of the burning rates from various experiments are given in Table 8.

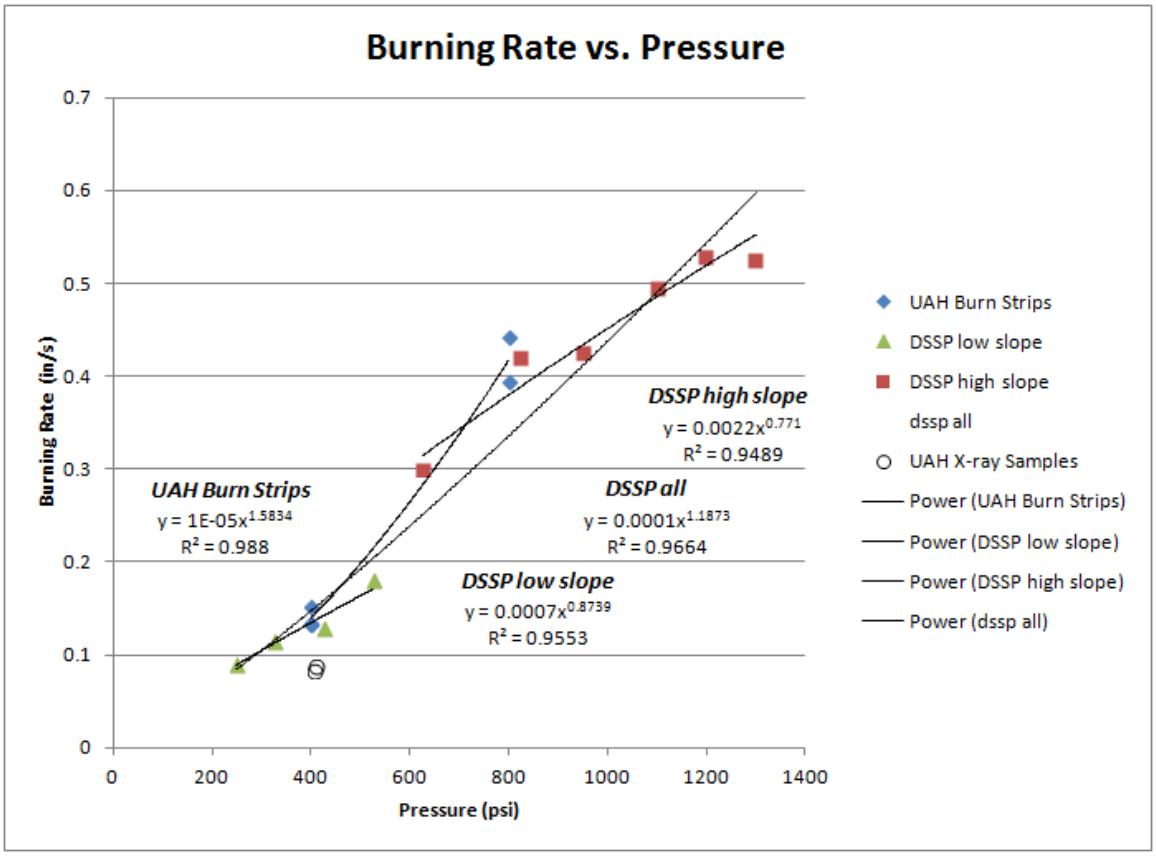


Figure 79: Plot of X-ray Burning Rates with Other UAH and DSSP Experiments

5.5 Comparison of Uncertainty of UAH X-ray Results to Historical Reports

The smallest uncertainty calculated on a test conducted at UAH with X-ray imaging of the propellant regression was found to be ±2.86% from UAH04A01. Test UAH04A02 had an uncertainty of ±3.04%. These uncertainty percentages suggest that the quality of the data is about the same as the data from experiments documented in the

past in which ±3-5% accuracy was reported by Pressley [26], Kuo [18], and Godai [12] as shown in Table 2.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The baseline burning rates of the propellant samples were found to be 0.0839 in/s ±2.86% at an average pressure of 407 psi ±3.6% and 0.0882 in/s ±3.04% at 410 psi ±3.9% for the conditions investigated in the Method 2 analysis. The uncertainty bounds applied for a 95% level of confidence. These uncertainties compared well with uncertainty and error values reported by other researchers in the past. The bounds of uncertainty on the burning rates do not enclose both mean burning rates, but the bounds overlap each other. These values were calculated in the Method 2 analysis.

The burning rates determined from the X-ray videos were different from the burning rates obtained for the burn strips of the same propellant formulation at 400 psi. The DSSP burn strip test at 0.09 in/s at 250 psi was the closest burn strip data point to the X-ray burning rates, but the burning rates from the 400-psi burn strip tests were well above the burning rates of the X-ray tests. This discrepancy may be due to a number of factors. There were significant shape differences between the X-ray samples and the burn strips which could have resulted in difference grain structures and different temperature propagations. Also, both of the X-ray propellant samples were over one year old when they were burned, and the burn strips were not as old. The UAH burn strips were

combusted in a purged environment in which nitrogen gas flowed through the combustion chamber during combustion. The X-ray samples were from a different propellant batch than the burn strips, but the formulation was the same and the batch is not highly suspected as the source of discrepancy.

The spatial resolution for UAH04A01 was 501.7 pixels-per-inch. The spatial resolution for UAH04A02 was 502.7 pixels-per-inch. The temporal resolution was 30 frames-per-second. While the frame-rate was slow compared to some frame-rates achieved in X-ray applications described in Chapter 2, these resolutions combined were as good as or better than the historical resolutions used specifically for the measurement of solid propellant burning rate using X-ray imaging.

The source of the largest known error in the analysis was the intensity fluctuations from frame-to-frame, which were most evident in the intensity plots of the motionless epoxy propellant model. Over multiple frames, the intensity corresponding to the bottom edge of the epoxy bowl was found to be in various places over a range of several pixel rows even though the model did not move. This variation was used as an input for the Monte Carlo analysis, which contributed directly to the final uncertainty bounds. The transmitted X-ray intensity fluctuation with time could be a result of possible fluctuations in X-ray source intensity itself.

6.2 Recommendations

Assessment of the consistency of the X-ray source intensity with time is recommended. Assessment of the X-ray detector black values while no X-rays are present is also recommended. The X-ray video quality might become improved if the pixel biases were studied in more depth. Each individual pixel could be calibrated to a

solid-color view, such that a color scale factor could be applied to every pixel for every video frame. If the pixel biases could be quantified or adjusted, then it might be possible to use less extensive filtering of the X-ray images, which could help to reduce the uncertainty of the burning rate measurements. Since a median filter was used for all of the images before analysis, the results may or may not be improved by studying the pixel bias. If the ESP formulation is studied more in the future, an aging study of the ESP is recommended.

If the 6-bit capturing technique must be used in the future, gray-scale smoothing is recommended for that data in order to provide higher resolution even though the gray values might be somewhat artificial in that case. It is also recommended to perform sensitivity studies on the various parameters that were varied as inputs for the Monte Carlo analysis. The pixel column cannot be varied extensively because it will cause artificially high burning rate measurements, but the intensity thresholds and resulting pixel row could be further investigated. The filter neighborhood size could also be studied in more depth to see if less filtering could be sufficient. Smoothing of an intensity plot itself from an unfiltered image could be investigated as well.

The surface tracking code could possibly be made more accurate by adding a reconstruction algorithm. If the code could create a smooth line of 2-D points to fit the bowl shaped burning surface, then the bottom tip of the reconstructed virtual bowl line could be tracked from frame to frame. The nature of forcing a line to smoothly fit the contour of the burning surface could possibly reduce the burning rate uncertainty associated with limiting the calculations to within one column of pixels.

APPENDIX A

DETAILED SUMMARY OF EQUIPMENT

A summary table is given below in Table 9. All handheld pictures were taken with either an iPhone 5C or a Nikon D90 with a Nikon AF-S NIKKOR 18-105mm 1:3.5-5.6G DX VR ED lens. ImageJ was used to compile montages of X-ray video frames. This manuscript was written in Microsoft Word 2007.

Table 9: Summary of Equipment

Item	Model	Company
Pressure		
¼" Pneumatic Valve	4F-B6LJ2-SS-51AC-1B	Total Hose
¼" Pneumatic Valve	4F-B6LJ2-SS-51AO-1B	Total Hose
1"Pneumatic Valve		McMaster
Analog Gauges	311D-254R	Total Hose
Check Valve	4A-C4L-1-T-SS	Parker; Swagelok
tubing and basic fittings		Parker; Swagelok
relief valve	4A-RH4A-VT-SSK3	Parker; Swagelok
burst disks		Zook
burst disk assembly		High Pressure Equipment
surge tank	CY-NI 300	Airgas
pressure transducers	206 150CP…	Setra
Power		
DC Power Supply	N5772A	Agilent Technologies
Electrical Pass-through fitting	EG-125-A-CU-V	Conax
X-ray		
X-ray Detector	Toshiba E5877J-P1 Image Intensifier	North American

	Kappa HiRes3-XR Digital Camera	Imaging
X-ray Source	ERESCO 200 MF4-R	General Electric
Software		
Programmable logic controller interface	CX Programmer 7.03	OMRON Corp.
Virtual instrument program	LabVIEW 8.0.1	National Instruments
MATLAB	MATLAB v2014b	MathWorks
Image capture program	KCC XRay 1.0.0.11121	Kappa
Image manipulation program	ImageJ	
Document editor	Word, Office 2007	Microsoft
Spreadsheet editor	Excel, Office 2007	Microsoft
Chart editor	PowerPoint, Office 2007	Microsoft
Miscellaneous		
24-hour epoxy	EpoPro 2300, Hardener HY 956 EN/US	Specialty Polymers and Services
5-minute epoxy	Super Fast Epoxy Cement	Elmer's

APPENDIX B

UNCERTAINY OF SCALE FACTOR

The uncertainty of the height measurement of the aluminum tube for UAH04A01 could not be directly measured since it was cast into an epoxy base, so the height was calculated with Student's t-distribution [32], which allows the statistical analysis of small sample sizes from a population. The use of this distribution requires that the underlying population has a normal distribution, so it was assumed that the population of the height of the tubes was normally distributed. Eleven spare aluminum tubes were made at the same time as the tubes that were used to make the propellant samples. The 11 spare tubes were not cast into epoxy like the propellant samples, so the lengths of the spare tubes were measured with a pair of calipers. The measured lengths are shown in Table 10.

The bound for the confidence limit of a value around the mean for the t-distribution is [32]

$$U_d = t \cdot s \, ,$$

where *t* is the ratio of the difference between the mean values of a small sample and its underlying population over the standard deviation of the mean value of the small sample, and *s* is the standard deviation of the sample. The value of *t* is found from a standard table of t-values. For α=0.05 for a two-sided distribution at a 95% confidence level with

11 samples, the value of *t* is 2.201. The statistics of the length measurements and the resulting uncertainty are shown at the bottom of Table 10.

Table 10: Measurements of Aluminum Tube Lengths

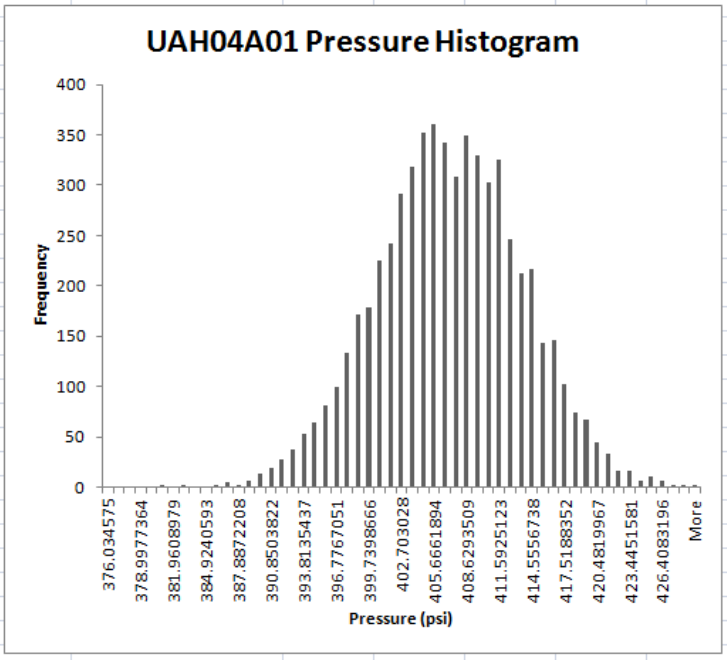
Length (inch)	
	0.899
	0.901
	0.904
	0.899
	0.899
	0.909
	0.903
	0.901
	0.899
	0.906
	0.906
Mean=	0.902364
s=	0.003501
U_d=	0.007705

APPENDIX C

UNCERTAINTY OF PRESSURE

The uncertainties of the mean pressures of the combustion bomb during the middle three seconds of each combustion test were calculated as the root-sum-square of the manufacturer's specification for accuracy and the value of two standard deviations of the bomb pressure data from two pressure transducers. The pressure uncertainty analysis is shown below for each test. The uncertainty bounds apply for a 95% confidence level because about 95% of the values fall within two standard deviations of the mean for each test as shown below.

Bin	Frequency	Cumulative %
376.0346	1	0.02%
377.0223	0	0.02%
378.01	0	0.02%
378.9977	0	0.02%
379.9855	0	0.02%
380.9732	2	0.05%
381.9609	0	0.05%
382.9486	3	0.10%
383.9363	0	0.10%
384.9241	1	0.12%
385.9118	2	0.15%
386.8995	5	0.23%
387.8872	3	0.28%
388.8749	7	0.40%
389.8627	14	0.63%
390.8504	19	0.95%
391.8381	27	1.40%
392.8258	38	2.03%
393.8135	53	2.92%
394.8013	65	4.00%
395.789	81	5.35%
396.7767	100	7.01%
397.7644	134	9.25%
398.7521	171	12.10%
399.7399	178	15.06%
400.7276	225	18.81%
401.7153	242	22.84%
402.703	291	27.69%
403.6907	319	33.01%
404.6785	352	38.87%
405.6662	360	44.87%
406.6539	342	50.57%
407.6416	309	55.71%
408.6294	350	61.55%
409.6171	329	67.03%
410.6048	303	72.08%
411.5925	326	77.51%
412.5802	246	81.61%
413.568	213	85.15%
414.5557	217	88.77%
415.5434	144	91.17%
416.5311	146	93.60%
417.5188	103	95.32%
418.5066	74	96.55%
419.4943	67	97.67%
420.482	45	98.42%
421.4697	34	98.98%
422.4574	16	99.25%
423.4452	16	99.52%
424.4329	6	99.62%
425.4206	11	99.80%
426.4083	6	99.90%
427.396	2	99.93%
428.3838	2	99.97%
More	2	100.00%



For UAH04A01 pressures P1 and P2 from 2-5 seconds:		
m=	406.5612	psi
σ=	6.638955	psi
2σ=	13.27791	psi
m-2σ=	393.2833	psi
m+2σ=	419.8391	psi
Percent in ±2σ=	95.63479	%

Setra 206 Transducer Accuracy Specification:

Accuracy=	±0.13% FS	

http://www.setra.com/products/pressure/model-206-industrial-pressure-transducer

http://cdn2.hubspot.net/hubfs/211498/Product_Data_Sheets/206_DS.pdf?t=1432155618233

FS (Full Scale output)=	5000	psi
Accuracy=0.0013*5000=	±6.5	psi

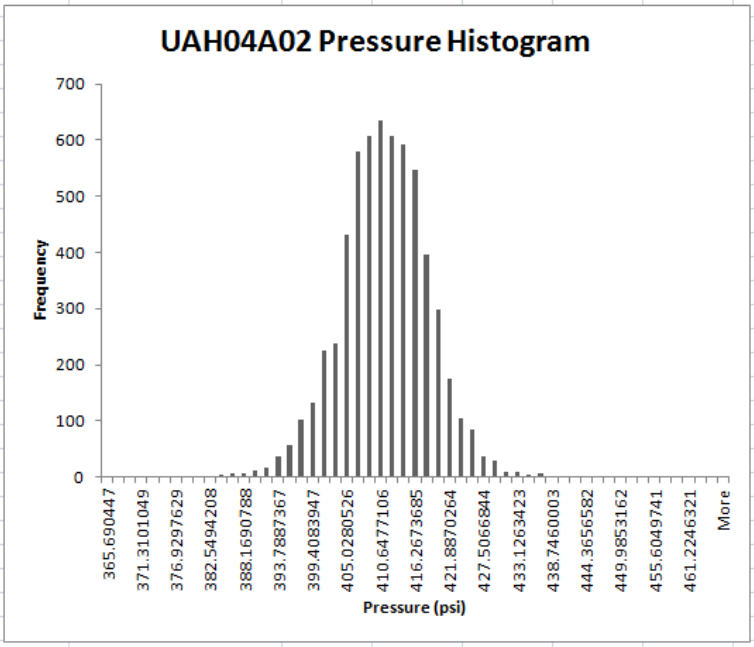
Uncertainty in Pressure[†]:

$$U_P = \sqrt{Accuracy^2 + (2\sigma)^2} = \sqrt{6.5^2 + (13.27791)^2}$$

U_P=	14.78353	psi (±)
U_P%=100*U_P/m=	3.636239	% (±)

[†]: Beckwith, T.G., Marangoni, R.D., Lienhard, J.H. V, "Mechanical Measurements", 6th Ed., Pearson Education, Inc., Upper Saddle River, NJ, 2007.

Bin	Frequency	Cumulative %
365.6904	1	0.02%
367.5637	0	0.02%
369.4369	0	0.02%
371.3101	0	0.02%
373.1833	0	0.02%
375.0565	0	0.02%
376.9298	0	0.02%
378.803	0	0.02%
380.6762	1	0.03%
382.5494	1	0.05%
384.4226	3	0.10%
386.2959	6	0.20%
388.1691	6	0.30%
390.0423	12	0.50%
391.9155	16	0.77%
393.7887	36	1.37%
395.662	57	2.32%
397.5352	103	4.03%
399.4084	133	6.25%
401.2816	226	10.01%
403.1548	239	14.00%
405.0281	431	21.18%
406.9013	579	30.82%
408.7745	608	40.95%
410.6477	635	51.53%
412.5209	608	61.66%
414.3941	592	71.53%
416.2674	547	80.64%
418.1406	397	87.25%
420.0138	298	92.22%
421.887	174	95.12%
423.7602	105	96.87%
425.6335	84	98.27%
427.5067	36	98.87%
429.3799	30	99.37%
431.2531	10	99.53%
433.1263	8	99.67%
434.9996	5	99.75%
436.8728	7	99.87%
438.746	0	99.87%
440.6192	0	99.87%
442.4924	2	99.90%
444.3657	1	99.92%
446.2389	2	99.95%
448.1121	0	99.95%
449.9853	0	99.95%
451.8585	2	99.98%
453.7318	0	99.98%
455.605	0	99.98%
457.4782	0	99.98%
459.3514	0	99.98%
461.2246	0	99.98%
463.0979	0	99.98%
464.9711	0	99.98%
More	1	100.00%



For UAH04A02 pressures P1 and P2 from 2-5 seconds:		
m=	410.3732	psi
σ=	7.280212	psi
2σ=	14.56042	psi
m-2σ=	395.8128	psi
m+2σ=	424.9336	psi
Percent in ±2σ=	95.5015	%

Setra 206 Transducer Accuracy Specification:

Accuracy= ±0.13% FS

http://www.setra.com/products/pressure/model-206-industrial-pressure-transducer

http://cdn2.hubspot.net/hubfs/211498/Product_Data_Sheets/206_DS.pdf?t=1432155618233

FS (Full Scale output)= 5000 psi

Accuracy=0.0013*5000= ±6.5 psi

Uncertainty in Pressure†:

$$U_P = \sqrt{Accuracy^2 + (2\sigma)^2} = \sqrt{6.5^2 + (14.56042)^2}$$

U_P=	15.94541	psi (±)
U_P%=100*U_P/m=	3.885587	% (±)

†: Beckwith, T.G., Marangoni, R.D., Lienhard, J.H. V, "Mechanical Measurements", 6th Ed., Pearson Education, Inc., Upper Saddle River, NJ, 2007.

APPENDIX D

PIXEL BIAS

In order to determine whether or not the detector had pixel biases, images were captured with nothing in front of the detector at two different X-ray tube voltages, and images were captured through the combustion bomb windows while there was no propellant sample mounted inside of the bomb. Examples of an image from each set are shown in Figure 80 and Figure 84. For each of these three sets of images, intensity values from filtered images were plotted with intensity values of the unfiltered images. Before the intensity values were plotted, the filtered images were averaged together and the non-filtered images were averaged together. The intensity plots for these averaged background images are given for pixel column 602 in Figure 81, Figure 82, and Figure 83 and for pixel column 575 in Figure 85, Figure 86, and Figure 87.

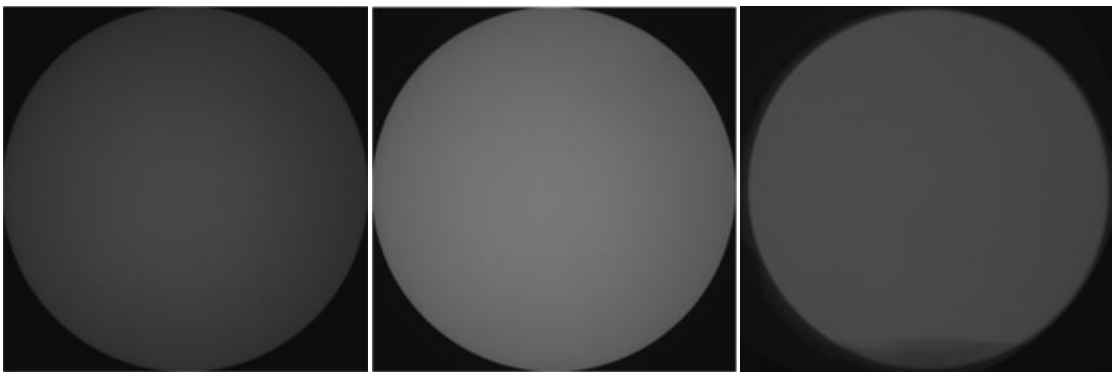


Figure 80: Example X-ray Images, 14-bit; Blank Background 20 kV Gain 300 (left), 35 kV Gain 0 (middle), 100 kV Gain 300 (right)

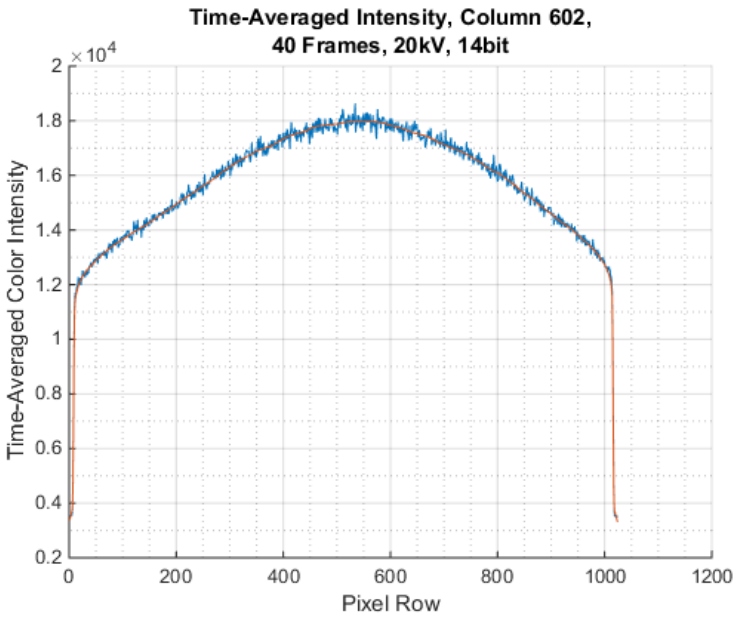


Figure 81: Intensity Plot, Blank Detector, 20 kV, Gain 300, 14-bit, Column 602

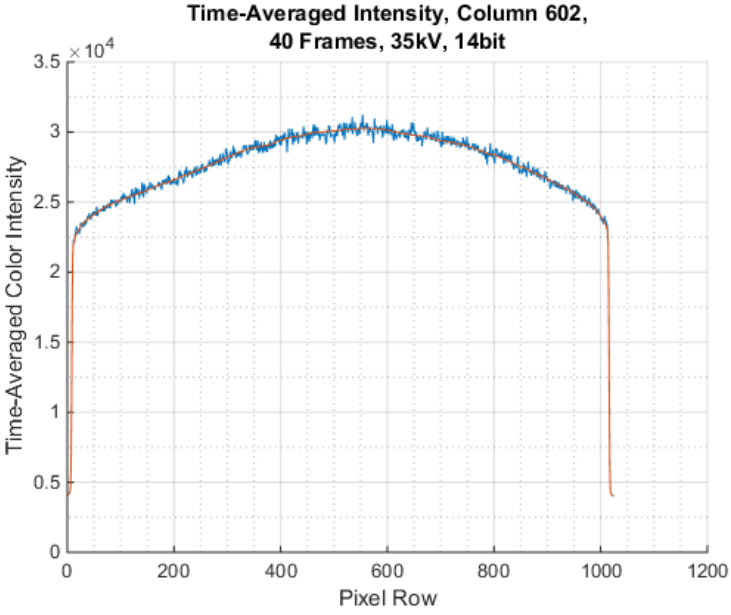


Figure 82: Intensity Plot, Blank Detector, 35 kV, Gain 0, 14-bit, Column 602

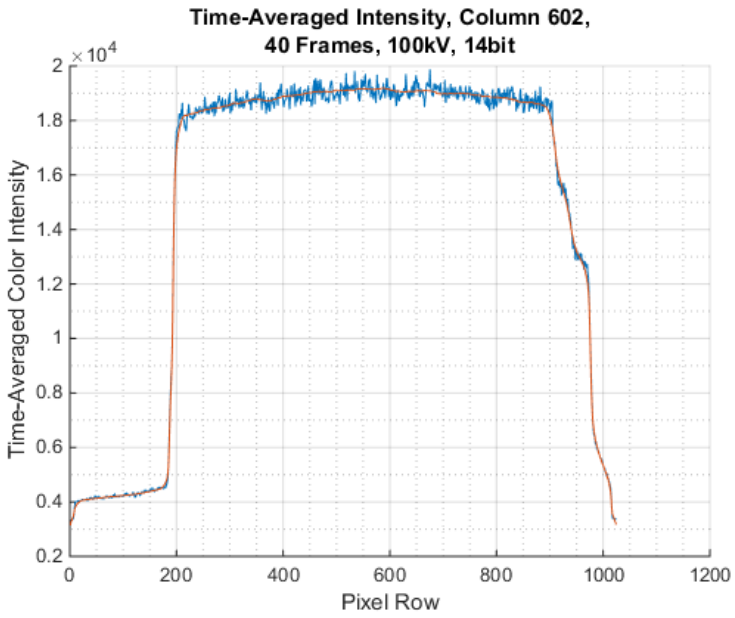


Figure 83: Intensity Plot, Bomb Windows, 100 kV, Gain 300, 14-bit, Column 602

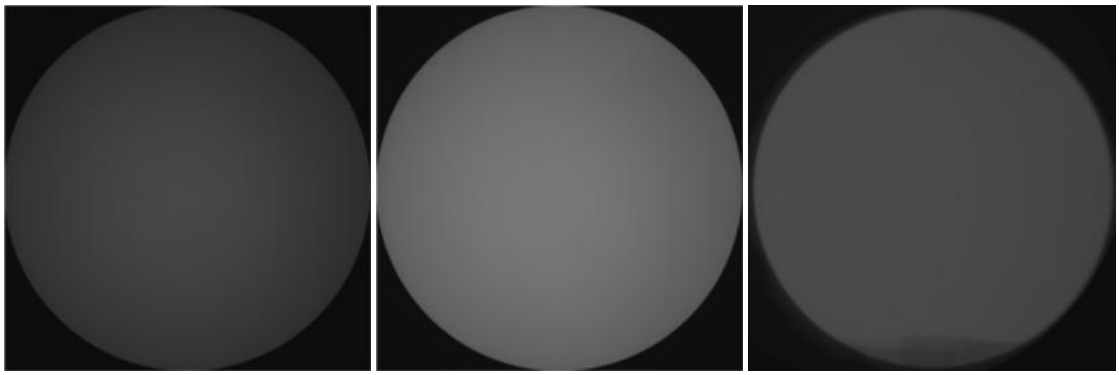


Figure 84: Example X-ray Images, 6-bit; Blank Background 20 kV Gain 300 (left), 35 kV Gain 0 (middle), 100 kV Gain 300 (right)

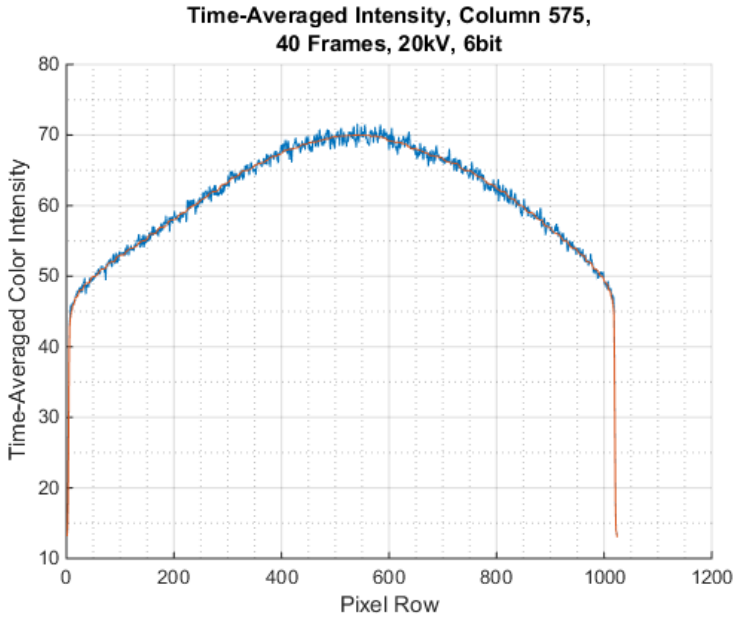


Figure 85: Intensity Plot, Blank Detector, 20 kV, Gain 300, 6-bit, Column 575

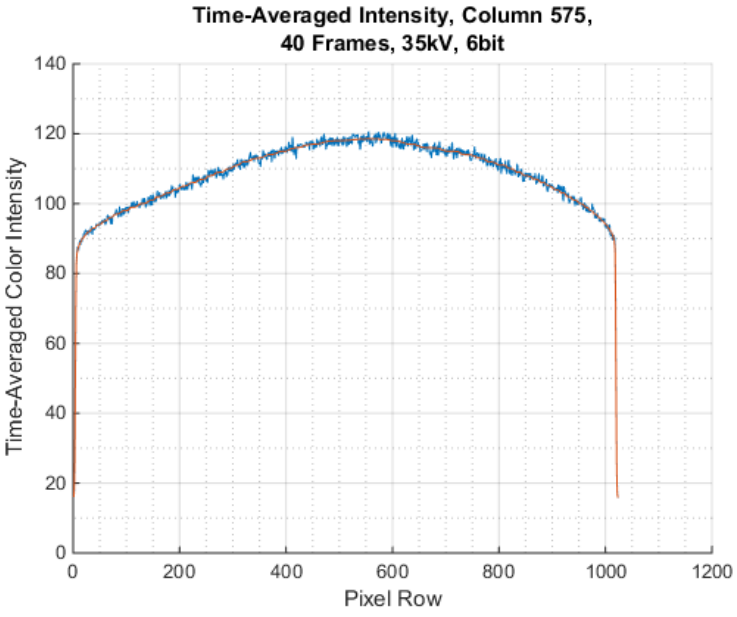


Figure 86: Intensity Plot, Blank Detector, 35 kV, Gain 0, 6-bit, Column 575

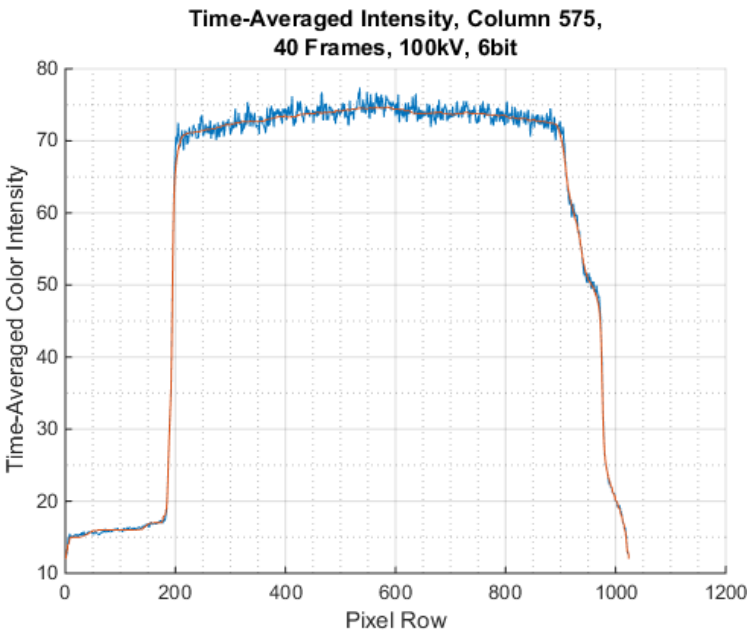


Figure 87: Intensity Plot, Bomb Windows, 100 kV, Gain 300, 6-bit, Column 575

Pixel bias is suggested by the fact that the unfiltered, averaged intensity plot is still a noisy line. The difference between colors in the averaged plots means that there is some consistency in the differing color responses between pixels for the same excitation. The biases were subtracted from the filtered intensities, and the percent biases were plotted as percentages of the highest possible gray values as shown in Figure 88 and Figure 89. Only rows 490-670 were plotted in the percent bias plots because those rows are the only rows in which the code searches for the target intensity value. Many of the peaks visible in the bias plots are present for all of the sets of images, indicating the possible presence of pixel bias in the raw images.

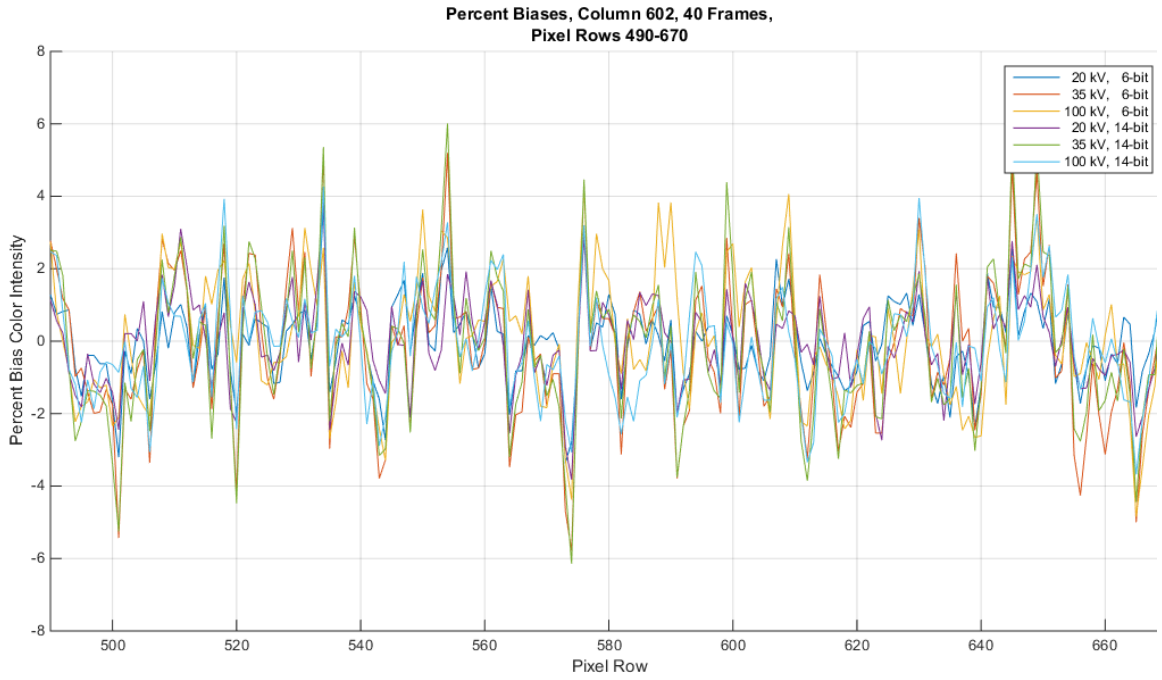


Figure 88: Percent Bias, Column 602

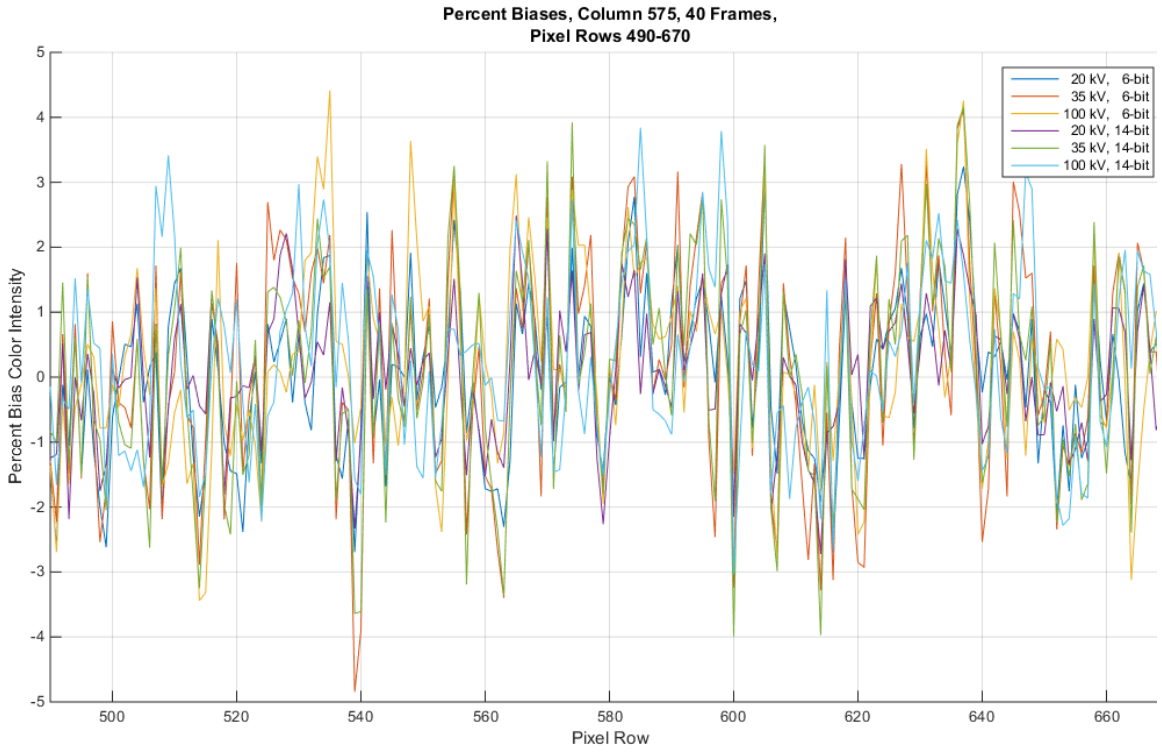


Figure 89: Percent Bias, Column 575

APPENDIX E

MATLAB SCRIPTS

Edge_Tracking_ESP_Xray_MONTE_CARLO_PixelShiftIntersectionRow.m

```
% Edge_Tracking_ESP_Xray_MONTE_CARLO_PixelShiftIntersectionRow
% Matthew Denny
%{
INSTRUCTIONS:
Choose a set of frames in TIFF format from the image sequence of the
video
in which the propellant has already formed a discernible burning
surface
through the end of the time when the burning surface is easily
discernible,
and save these images in the MATLAB folder or the folder containing
this
script. In this script, reset the variables under the heading
"INITIALIZE
THESE VARIABLES BEFORE RUNNING". In the first while loop, set the
ithFileName command to fetch the correct file names. For each run,
adjust
the plot labels as appropriate. To find out what vartype should be, use
a
command like i1=imread('whateverfilename.tif') and see what type of
variable it gives in the "Workspace" table. Calibrate the distance for
a
new series of images using the imshow command or MSPaint.
The average regression rate is reported in the Command Window.
Examine the log file called "X-ray_MATLAB_LogFile.txt". The code for
creating the file is at the end of this script. The file is placed in
the
current directory, which is usually the MATLAB folder.
%

%}
%_____
____

clc
clear all
close all
```

```
% INITIALIZE THESE VARIABLES BEFORE RUNNING
testn=2; % UAH04A01 is 1, UAH04A02 is 2
framecount=81; % total number of frames in the directory for analysis
fint=1; % interval of frames selected. (Use every fint'th frame)
bsmpline=[602;575]; % 602,575
bsmp=bsmpline(testn); % Pixel corresponding to the burn surface
midpoint
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise (25
works)
vartype='uint8'; % the variable type for the image data
fps=30; % frames per second
SF=[501.7;502.7]; % ppi, scale factor
U_SF=[4.98;9.32]; % ppi, Uncertainty in scale factor
%distcal=1/SF(testn); % in/pixel, distance calibration
%intenI=38540; % 38540, 147, target intensity value (TIV) for full
frames
intenIint1vect=[31000;182]; % 30000,first TIV for cropped frames
intenIint2vect=[26000;91]; % 25500,second TIV for cropped frames
intenIint1=intenIint1vect(testn);
intenIint2=intenIint2vect(testn);
intenPERC=[23000;65];
c=intenPERC(testn);
rowstart=490; % 490 the row at which to start the search range
rowend=670; % 670 the row at which to stop the search range
linefactor=[2;1.5]; % for projected line length on plots
bitdq=[2^16;2^8]; % bit depth quantity
xspan=[502:702;475:675]; % span of region of interest
yspan=[460:680;470:690]; % height of region of interest
xspanL=length(xspan(testn,:)); % length of vector
yspanL=length(yspan(testn,:)); % length of vector
bsmpint=bsmp-xspan(testn,1); % center pixel column of bowl for cropped
images

% Plot the intensities along one column for the frames specified by
fint
I=zeros(pxcount,pxcount,framecount,vartype);
Iint=zeros(yspanL,xspanL,framecount,vartype); % region of interest
%Iints=zeros(yspanL,xspanL,framecount,vartype);
usedcount=0;
figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
while i<=framecount
    ithFileName=sprintf('UAH04A02 (%d).tif', i); % 'pic' or 'UAH04A02'
    I(:,:,i)=imread(ithFileName); % store the image in a matrix
    % Use a filter to remove noise from the image:
    I(:,:,i)=medfilt2(I(:,:,i),[filtsize filtsize]);
    %Iints(:,:,i)=imadjust(I(yspan(testn,:),xspan(testn,:),i));
    Iint(:,:,i)=I(yspan(testn,:),xspan(testn,:),i);
    %plot(Iints(:,bsmpint,i));
    i=i+fint;
    usedcount=usedcount+1;
end
% Create plot labels
title(sprintf('Auto-Gray-Scaled Intensity for Pixel Column %d,\nEvery
%dth Frame for 3 Seconds',bsmp,fint));
xlabel('Pixel Row');
```

```
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);

%
% Manual Gray-Scaling
maxI=zeros(usedcount,1,vartype);
maxIint=zeros(usedcount,1,vartype);
minI=zeros(usedcount,1,vartype);
minIint=zeros(usedcount,1,vartype);
gs=1;
i=1;
while i<=framecount
    maxI(gs)=max(max(I(:,:,i)));
    maxIint(gs)=max(max(Iint(:,:,i)));
    minI(gs)=min(min(I(:,:,i)));
    minIint(gs)=min(min(Iint(:,:,i)));
    gs=gs+1;
    i=i+fint;
end
% Manually Scale
I=I-min(minI);
I=I.*(bitdq(testn)./(max(maxI)-min(minI)));
Iint=Iint-min(minIint);
Iint=Iint.*(bitdq(testn)./(max(maxIint)-min(minIint)));
hold off; figure; hold on;
i=1;
while i<=framecount
    plot(Iint(:,bsmpint,i));
    i=i+fint;
end
% Create plot labels
title(sprintf('Manual-Gray-Scaled Intensity for Pixel Column %d,\nEvery
%dth Frame for 3 Seconds',bsmp,fint));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
legend(legendcell);
%
```

```
% MONTE CARLO BURNING RATE DETERMINATION
% Calculate burning rate: least squares line fit, y=a+bt
% The variables rowstart and rowend define the range of rows in which
to
% search for a known intensity marking the burning surface to be
tracked.
% Vary the pixel column, scale factor, and target intensity choices.
mccount=2000000; % 2,000,000, number of Monte Carlo iterations
t=0:(1/fps)*fint:(1/fps)*fint*(usedcount-1); % s, say the first frame
occurs at time=0
t=t'; % s, time vector, invert the array
BSLOCpix=zeros(usedcount,mccount); % pixels, burning surface locations
BSLOCpixMC=zeros(usedcount,mccount);% pixels, burning surface locations
MCBR=zeros(mccount,1); % in/s, Monte Carlo Burning Rate
rTIV1=[1000;13]; % intensity addition
rTIV2=[1000;13]; % intensity addition
rTIV3=[1000;0]; % intensity addition
rSF=U_SF(testn)/2; % ppi addition
rcol=3.49999; % 3.49999, pixel column addition, use decimals since the
result gets rounded
stdpixn=[3.640;2.953]; % 3.640;2.953 standard deviation of rows from
the epoxy propellant model
stdpix=stdpixn(testn); % standard deviation of rows from the epoxy
propellant model

% Monte Carlo Loop
for mc=1:mccount
    % Calculate random additions
    randTIV1=random('Uniform',-rTIV1(testn),rTIV1(testn)); % intensity
addition
    randTIV2=random('Uniform',-rTIV2(testn),rTIV2(testn)); % intensity
addition
    randTIV3=random('Uniform',-rTIV3(testn),rTIV3(testn)); % intensity
addition
    randSF=random('Uniform',-rSF,rSF); % scale factor addition
    randcol=round(random('Uniform',-rcol,rcol)); % pixel column
addition
    % Apply random additions
    intenIint1r=intenIint1+randTIV1;
    intenIint2r=intenIint2+randTIV2;
    cr=c+randTIV3;
    SFr=SF(testn)+randSF;
    distcal=1/SFr;
    bsmpintr=bsmpint+randcol;

    % Find two points on the curve in the bowl-shape in order to find
the slope
    reportr1=zeros(framecount,1); % holds the row numbers at which the
first TIV is located
    reportr2=zeros(framecount,1); % holds the row numbers at which the
second TIV is located
    % Find the pixel rows corresponding to the locations of the target
intensity values
    i=1;
    while i<=framecount
```

```
        row=1;
        while reportr2(i)==0 && row<=yspanL
            if (Iint(row,bsmpintr,i)<=intenIint1r) && (reportr1(i)==0)
                reportr1(i)=row; % pixels, the pixel row of the first
TIV for the ith frame
            end
            if (Iint(row,bsmpintr,i)<=intenIint2r) && (reportr2(i)==0)
                reportr2(i)=row; % pixels, holds the row numbers at
which the second TIV is located
            end
            row=row+1;
        end
        i=i+fint;
    end
    % reportr1 and reportr2 now hold the pixel row locations of the
TIVs

    % Find the slope of each line and find its projected intersection
with the
    % intensity corresponding to the perceived location of the surface
    BSLOCvect=zeros(usedcount,1); % pixels, burning surface locations
    BSLOCvectr=zeros(usedcount,1); % pixels, randomly varied burning
surface locations
    i=1;
    j=1;
    % Least square error line fit with p=ax+b
    while i<=framecount
        x=[reportr1(i);reportr2(i)];
        p=[intenIint1r;intenIint2r];
        n=length(p);
        a=(sum(p)*sum(x.^2)-sum(x)*sum(x.*p))/(n*sum(x.^2)-sum(x)^2); %
in, y-intercept
        b=(n*sum(x.*p)-sum(x)*sum(p))/(n*sum(x.^2)-sum(x)^2); % in/s,
slope
        BSLOCvect(j)=(cr-a)/b; % pixel row of the intersection
        randpix=random('Normal',0,stdpix); % pixel row addition
        BSLOCvectr(j)=BSLOCvect(j)+randpix; % pixels, apply the random
variation
        %begin plot
        %longx=[x(1);x(1)+linefactor(testn)*(x(2)-x(1))];
        %projline=b*longx+a;
        %plot(longx,projline,'k-');
        %end plot
        j=j+1;
        i=i+fint;
    end % Now the intersection points are known. These points are the
burning surface locations.

    % Re-order the burning surface locations to descending order
instead of ascending order
    %BSLOC=zeros(usedcount,1); % inches
    i=1;
    Lstart=0.5; % in, % arbitrary starting location measured from the
bottom of the sample tube
    for i=1:usedcount
```

```
        BSLOCpix(i,mc)=600-(BSLOCvectr(i)-BSLOCvectr(1)); % pixels,
burn surface location, first value is 600
        %BSLOC(i)=Lstart-(BSLOCvect(i)-BSLOCvect(1))*distcalr; % in,
first value is Lstart
    end

    % Find the burning rate: Least Squares Line Fit through burning
surface locations
    y=BSLOCpix(:,mc); % pixels, locations of the burning surface
    n=length(y);
    %coefa=(sum(y)*sum(t.^2)-sum(t)*sum(t.*y))/(n*sum(t.^2)-sum(t)^2);
% pixels, y-intercept
    coefb=(n*sum(t.*y)-sum(t)*sum(y))/(n*sum(t.^2)-sum(t)^2); %
pixels/s, slope
    MCBR(mc)=abs(coefb); % pixels/s
    MCBR(mc)=MCBR(mc)*distcal; % convert to inches/s
    %fprintf('%d\n',mc);
end



mBR=mean(MCBR);
sdBR=std(MCBR);
twosigma=2*sdBR;
Uperc=100*twosigma/mBR;
outsidecount=0;
twosigmaupperbound=mBR+twosigma;
twosigmalowerbound=mBR-twosigma;
for mc=1:mccount
    if MCBR(mc)>twosigmaupperbound || MCBR(mc)<twosigmalowerbound
        outsidecount=outsidecount+1;
    end
end
conflevel=(1-(outsidecount/mccount))*100; % percent, values within
2*sigma

figure; histogram(MCBR,35)
title(sprintf('Monte Carlo Burning Rates for %d Runs\nUsing Every
Frame',mccount));
xlabel('Burning Rate (in/s)');
ylabel('Number of Occurences');
annotstring=sprintf('Mean=%.4f in/s\n\rStandard Deviation=%0.5f
in/s\n\rValues Within 2*s: %.2f%%',mBR,sdBR,conflevel);
tb1=annotation('textbox',[0.63,0.82,0.45,0.1],'string',annotstring);
tb1.FontSize=9;
tb1.LineStyle='none';

fprintf('Mean Burning Rate: %.5f\r',mBR);
fprintf('Burning Rate Standard Deviation: %.5f\r',sdBR);
fprintf('Burning Rate Uncertainty (which is +-2*sigma): +-
%.5f\r',twosigma);
fprintf('percent count of values within 2*sigma = %.4f\r',conflevel);
fprintf('Percent Uncertainty: +-%.4f\r\n',Uperc)
```

Edge_Tracking_Xray_ESP_MANUALSCALE_PLOTS.m

```
% Edge_Tracking_Xray_ESP_MANUALSCALE_PLOTS
% Matthew Denny
%{
INSTRUCTIONS:
Choose a set of frames in TIFF format from the image sequence of the
video
in which the propellant has already formed a discernible burning
surface
through the end of the time when the burning surface is easily
discernible,
and save these images in the MATLAB folder or the folder containing
this
script. In this script, reset the variables under the heading
"INITIALIZE
THESE VARIABLES BEFORE RUNNING". In the first while loop, set the
ithFileName command to fetch the correct file names. For each run,
adjust
the plot labels as appropriate. To find out what vartype should be, use
a
command like i1=imread('whateverfilename.tif') and see what type of
variable it gives in the "Workspace" table. Calibrate the distance for
a
new series of images using the imshow command or MSPaint.
The average regression rate is reported in the Command Window.
Examine the log file called "X-ray_MATLAB_LogFile.txt". The code for
creating the file is at the end of this script. The file is placed in
the
current directory, which is usually the MATLAB folder.
%

%}
%_____
____

clc
clear all
close all

% INITIALIZE THESE VARIABLES BEFORE RUNNING
testn=1; % UAH04A01 is 1, UAH04A02 is 2
framecount=81; % total number of frames in the directory for analysis
fint=10; % interval of frames selected. (Use every fint'th frame)
bsmpline=[602;575]; % 602,575
bsmp=bsmpline(testn); % Pixel corresponding to the burn surface
midpoint
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise (25
works)
vartype='uint16'; % the variable type for the image data
fps=30; % frames per second
SF=[501.7;502.7]; % ppi, scale factor
U_SF=[4.977;9.321]; % ppi, Uncertainty in scale factor
Usf=U_SF(testn); % ppi
distcal=1/SF(testn); % in/pixel, distance calibration
```

```
intenIint1vect=[31000;182]; % first TIV for cropped frames
intenIint2vect=[26000;91]; % second TIV for cropped frames
intenIint1=intenIint1vect(testn);
intenIint2=intenIint2vect(testn);
intenPERC=[23000;65];
c=intenPERC(testn);
rowstart=490; % 490 the row at which to start the search range
rowend=670; % 670 the row at which to stop the search range
linefactor=[2;1.5]; % for projected line length on plots
bitdq=[2^16;2^8]; % bit depth quantity
xspan=[502:702;475:675]; % span of region of interest
yspan=[460:680;470:690]; % height of region of interest
xspanL=length(xspan(testn,:)); % length of vector
yspanL=length(yspan(testn,:)); % length of vector
bsmpint=bsmp-xspan(testn,1); % center pixel column of bowl for cropped
images

% Plot the intensities along one column for the frames specified by
fint
I=zeros(pxcount,pxcount,framecount,vartype);
Iint=zeros(yspanL,xspanL,framecount,vartype); % region of interest
Iints=zeros(yspanL,xspanL,framecount,vartype);
usedcount=0;
figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
while i<=framecount
    ithFileName=sprintf('pic (%d).tif', i); % 'pic' or 'UAH04A02'
    I(:,:,i)=imread(ithFileName); % store the image in a matrix
    % Use a filter to remove noise from the image:
    I(:,:,i)=medfilt2(I(:,:,i),[filtsize filtsize]);
    Iints(:,:,i)=imadjust(I(yspan(testn,:),xspan(testn,:),i));
    Iint(:,:,i)=I(yspan(testn,:),xspan(testn,:),i);
    plot(Iints(:,bsmpint,i))
    i=i+fint;
    usedcount=usedcount+1;
end
% Create plot labels
title(sprintf('Auto-Gray-Scaled Intensity for Pixel Column %d,\nEvery
%dth Frame for %d frames',bsmp,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);

figure;
imshow(Iint(:,:,31));
```

```
%
% Manual Gray-Scaling
maxI=zeros(usedcount,1,vartype);
maxIint=zeros(usedcount,1,vartype);
minI=zeros(usedcount,1,vartype);
minIint=zeros(usedcount,1,vartype);
gs=1;
i=1;
while i<=framecount
    maxI(gs)=max(max(I(:,:,i)));
    maxIint(gs)=max(max(Iint(:,:,i)));
    minI(gs)=min(min(I(:,:,i)));
    minIint(gs)=min(min(Iint(:,:,i)));
    gs=gs+1;
    i=i+fint;
end
% Plot the Maximum Gray-Scale values per frame
hold off; figure; hold on;
plot(maxI,'b+');
plot(maxIint,'r+');
title('Maximum Gray Values for X-ray Frames');
ylabel('Maximum Intensity');
xlabel('Frame');
% Plot the Minimum Gray-Scale values per frame
hold off;figure; hold on;
plot(minI,'b+');
plot(minIint,'r+');
title('Minimum Gray Values for X-ray Frames');
ylabel('Minimum Intensity');
xlabel('Frame');
% Manually Scale
I=I-min(minI);
I=I.*(bitdq(testn)./(max(maxI)-min(minI)));
Iint=Iint-min(minIint);
Iint=Iint.*(bitdq(testn)./(max(maxIint)-min(minIint)));
hold off; figure; hold on;
i=1;
while i<=framecount
    plot(Iint(:,bsmpint,i));
    i=i+fint;
end
% Create plot labels
title(sprintf('Intensity for Pixel Column %d,\nEvery %dth Frame for %d
Frames',bsmp,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);
```

```
%

% BURNING RATE DETERMINATION
% Now the images have been enhanced or subtracted, so the regression
% calculations can be performed from the plots or derivative data.
% The variables rowstart and rowend define the range of rows in which
to
% search for a known intensity marking the burning surface to be
tracked.

reportr1=zeros(framecount,1); % holds the row numbers at which the
first TIV is located
reportr2=zeros(framecount,1); % holds the row numbers at which the
second TIV is located
% Find the pixel rows corresponding to the locations of the target
% intensity values
i=1;
while i<=framecount
    for row=1:yspanL
        if (Iint(row,bsmpint,i)<=intenIint1) && (reportr1(i)==0)
            reportr1(i)=row; % pixels, the pixel row of the first TIV
for the ith frame
        end
        if (Iint(row,bsmpint,i)<=intenIint2) && (reportr2(i)==0)
            reportr2(i)=row; % pixels, holds the row numbers at which
the second TIV is located
        end
    end
    i=i+fint;
end
% reportr1 and reportr2 now hold the pixel row locations of the TIVs

% Replot the column intensities so as to overlay the projected lines
hold off; figure; hold on;
i=1;
while i<=framecount
    plot(Iint(:,bsmpint,i));
    i=i+fint;
end
% Create plot labels
title(sprintf('Intensity for Pixel Column %d,\nEvery %dth Frame for %d
Frames',bsmp,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);
% Plot lines across the target intensity values
```

```
intenvectIint1=zeros(yspanL,1)+intenIint1;
intenvectIint2=zeros(yspanL,1)+intenIint2;
intenvectIint3=zeros(yspanL,1)+c;
plot(1:yspanL,intenvectIint1,'k-');
plot(1:yspanL,intenvectIint2,'k-');
plot(1:yspanL,intenvectIint3,'k--');
% Projected lines are about to be plotted

% Find the slope of each line and find its projected intersection with
the
% intensity corresponding to the perceived location of the surface
BSLOCvect=zeros(usedcount,1); % pixels, burning surface locations
i=1;
j=1;
% Least square error line fit with p=ax+b
while i<=framecount
    x=[reportr1(i);reportr2(i)];
    p=[intenIint1;intenIint2];
    n=length(p);
    a=(sum(p)*sum(x.^2)-sum(x)*sum(x.*p))/(n*sum(x.^2)-sum(x)^2); % in,
y-intercept
    b=(n*sum(x.*p)-sum(x)*sum(p))/(n*sum(x.^2)-sum(x)^2); % in/s, slope
    BSLOCvect(j)=(c-a)/b;
    %begin plot
    longx=[x(1);x(1)+linefactor(testn)*(x(2)-x(1))];
    projline=b*longx+a;
    plot(longx,projline,'k-');
    %end plot
    j=j+1;
    i=i+fint;
end

% Show the cropped images with burning surface locations and center
line
i=1;
j=1;
while i<=framecount
    Iinttrace=Iint(:,:,i);
    Iinttrace(round(BSLOCvect(j)),:)=0;
    Iinttrace(:,bsmpint)=0;
    figure; imshow(Iinttrace);
    i=i+fint;
    j=j+1;
end
% Show one cropped image with an example block of 25x25 pixels
blackened
Iinttrace=Iint(:,:,31);
Iinttrace(:,bsmpint)=0; % center line
Iinttrace(140:165,140:165)=0; % black square, filter neighborhood size
figure; imshow(Iinttrace);

% Re-order the burning surface locations to descending order
BSLOCpix=zeros(usedcount,1); % pixels, burning surface locations
BSLOC=zeros(usedcount,1); % inches
i=1;
```

```
Lstart=0.5; % in, % arbitrary starting location measured from the
bottom of the sample tube
for i=1:usedcount
    BSLOCpix(i)=600-(BSLOCvect(i)-BSLOCvect(1)); % pixels, burn surface
location, first value is 600
    BSLOC(i)=Lstart-(BSLOCvect(i)-BSLOCvect(1))*distcal; % in, first
value is Lstart
end

% Calculate burning rate: least squares line fit, y=a+bt
keepers=usedcount;
t=0:(1/fps)*fint:(1/fps)*fint*(keepers-1); % s, say the first frame
occurs at time=0
t=t'; % s, time vector, invert the array
y=BSLOC; % in, locations of the burning surface
n=length(y);
coefa=(sum(y)*sum(t.^2)-sum(t)*sum(t.*y))/(n*sum(t.^2)-sum(t)^2); % in,
y-intercept
coefb=(n*sum(t.*y)-sum(t)*sum(y))/(n*sum(t.^2)-sum(t)^2); % in/s, slope
% Calculate theoretical linear y values on the fitted line at each t(i)
yfit=coefa+coefb.*t; % in, theoretical linear burning surface locations
Ssq=sum((y-yfit).^2); % in^2, sum of squared deviations of the data
from the fitted line
% Calculate the correlation coefficient r in squared form, r^2=Rsq
ydif=sum((yfit-mean(y)).^2); % in^2
Rsq=ydif/(Ssq+ydif); % unitless, minimum=0, maximum=1
fprintf('Least Squares Burning Rate = %.4f in/s\n',abs(coefb)) % print
the result to the command line
fprintf('Squared Correlation Coefficient = %.4f\n',Rsq) % print the
result to the command line

% Plot location of burning surface and the least squares line fit
hold off; figure; hold on;
plot(t,BSLOC,'k+')
plot(t,yfit,'k-')
title(sprintf('Burning Surface Location vs Time\nUsing Every %dth frame
for %d Frames',fint,framecount));
xlabel('t, Time (s)');
ylabel('y, Burning Surface Location (in)');
ymaxval=max(BSLOC);
ylim([0 (0.01+ymaxval)]);
legend('Burning Surface Locations',sprintf('y=%.4ft+%.3f,
R^2=%.3f',coefb,coefa,Rsq));
grid;

% Show an image with the pixel column blackened
I(:,bsmp,1)=0;
figure; imshow(I(:,:,1));
Iint(:,bsmpint,1)=0;
figure;imshow(Iint(:,:,1));

% CREATE LOG FILE
fileID=fopen('X-ray_MATLAB_LogFile.txt','w');
fprintf(fileID, '%20s %20d\r\n','framecount',framecount);
fprintf(fileID, '%20s %20d\r\n','fint',fint);
fprintf(fileID, '%20s %20d\r\n','bsmp',bsmp);
```

```
fprintf(fileID, '%20s %20d\r\n','pxcount',pxcount);
fprintf(fileID, '%20s %20d\r\n','filtsize',filtsize);
fprintf(fileID, '%20s %20s\r\n','vartype',vartype);
fprintf(fileID, '%20s %20f\r\n','SF',SF);
fprintf(fileID, '%20s %20d\r\n','rowstart',rowstart);
fprintf(fileID, '%20s %20d\r\n','rowend',rowend);
fprintf(fileID, '%20s %20s\r\n','ithFileName',ithFileName);
fprintf(fileID, '%20s %20f in/s\r\n','Least Squares Burn
Rate',abs(coefb));
fprintf(fileID, '%20s %20f in/s\r\n','Squared Correlation
Coefficient',Rsq);
fclose(fileID);
```

Epoxy_Propellant_Model_study_PROJECTIONS.m

```
% Epoxy_Propellant_Model_study_PROJECTIONS
% Matthew Denny
%{
INSTRUCTIONS:
Choose a set of frames in TIFF format from the image sequence of the
video
in which the propellant has already formed a discernible burning
surface
through the end of the time when the burning surface is easily
discernible,
and save these images in the MATLAB folder or the folder containing
this
script. In this script, reset the variables under the heading
"INITIALIZE
THESE VARIABLES BEFORE RUNNING". In the first while loop, set the
ithFileName command to fetch the correct file names. For each run,
adjust
the plot labels as appropriate. To find out what vartype should be, use
a
command like i1=imread('whateverfilename.tif') and see what type of
variable it gives in the "Workspace" table. Calibrate the distance for
a
new series of images using the imshow command or MSPaint.
The average regression rate is reported in the Command Window.
Examine the log file called "X-ray_MATLAB_LogFile.txt". The code for
creating the file is at the end of this script. The file is placed in
the
current directory, which is usually the MATLAB folder.
%

%}
%_____
____

clc
clear all
close all

% INITIALIZE THESE VARIABLES BEFORE RUNNING
testn=2; % epoxypic16bit is 1, epoxy pic8bit is 2
framecount=80; % total number of frames in the directory for analysis
fint=1; % interval of frames selected. (Use every fint'th frame)
bsmpline=[470;470]; % 602,575
bsmp=bsmpline(testn); % Pixel corresponding to the burn surface
midpoint
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise (25
works)
vartype='uint8'; % the variable type for the image data
fps=30; % frames per second
%SF=[501.7;502.7]; % ppi, scale factor
%U_SF=[4.977;9.321]; % ppi, Uncertainty in scale factor
%Usf=U_SF(testn); % ppi
distcal=0.625/310; % in/pixel, distance calibration
```

```
intenIint1vect=[34000;150]; % 31000,182, first TIV for cropped frames
intenIint2vect=[16000;75]; % second TIV for cropped frames
intenIint1=intenIint1vect(testn);
intenIint2=intenIint2vect(testn);
intenPERC=[7000;34];
c=intenPERC(testn);
rowstart=530; % 490 the row at which to start the search range
rowend=620; % 670 the row at which to stop the search range
linefactor=[2;1.75]; % for projected line length on plots
bitdq=[2^16;2^8]; % bit depth quantity
xspan=[370:570;370:570]; % span of region of interest
yspan=[510:640;510:640]; % height of region of interest
xspanL=length(xspan(testn,:)); % length of vector
yspanL=length(yspan(testn,:)); % length of vector
bsmpint=bsmp-xspan(testn,1); % center pixel column of bowl for cropped
images

% Plot the intensities along one column for the frames specified by
fint
I=zeros(pxcount,pxcount,framecount,vartype);
Iint=zeros(yspanL,xspanL,framecount,vartype); % region of interest
%Iints=zeros(yspanL,xspanL,framecount,vartype);
usedcount=0;
figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
while i<=framecount
    ithFileName=sprintf('epoxy_bowl_8bit (%d).tif', i); % 16bit or 8bit
    I(:,:,i)=imread(ithFileName); % store the image in a matrix
    % Use a filter to remove noise from the image:
    I(:,:,i)=medfilt2(I(:,:,i),[filtsize filtsize]);
    %Iints(:,:,i)=imadjust(I(yspan(testn,:),xspan(testn,:),i));
    Iint(:,:,i)=I(yspan(testn,:),xspan(testn,:),i);
    %plot(Iints(:,bsmpint,i))
    i=i+fint;
    usedcount=usedcount+1;
end

%
% Manual Gray-Scaling
maxI=zeros(usedcount,1,vartype);
maxIint=zeros(usedcount,1,vartype);
minI=zeros(usedcount,1,vartype);
minIint=zeros(usedcount,1,vartype);
gs=1;
i=1;
while i<=framecount
    maxI(gs)=max(max(I(:,:,i)));
    maxIint(gs)=max(max(Iint(:,:,i)));
    minI(gs)=min(min(I(:,:,i)));
    minIint(gs)=min(min(Iint(:,:,i)));
    gs=gs+1;
    i=i+fint;
end
% Plot the Maximum Gray-Scale values per frame
hold off; figure; hold on;
plot(maxI,'b+');
```

```
plot(maxIint,'r+');
title(sprintf('Maximum Gray Values in Raw Frames\nFull Frames (Blue),
Cropped Frames (Red)'));
ylabel('Maximum Intensity');
xlabel('Frame');
% Plot the Minimum Gray-Scale values per frame
hold off;figure; hold on;
plot(minI,'b+');
plot(minIint,'r+');
title(sprintf('Minimum Gray Values in Raw Frames\nFull Frames (Blue),
Cropped Frames (Red)'));
ylabel('Minimum Intensity');
xlabel('Frame');
% Manually Scale
I=I-min(minI);
I=I.*(bitdq(testn)./(max(maxI)-min(minI)));
Iint=Iint-min(minIint);
Iint=Iint.*(bitdq(testn)./(max(maxIint)-min(minIint)));
hold off; figure; hold on;
i=1;
while i<=framecount
    plot(Iint(:,bsmpint,i));
    i=i+fint;
end
% Create plot labels
title(sprintf('Intensity for Pixel Column %d,\nEvery %dth Frame for %d
Frames',bsmp,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);
%

% Find the pixel rows corresponding to the locations of the target
% intensity value
% The variables rowstart and rowend define the range of rows in which
to
% search for a known intensity marking the burning surface to be
tracked.
reportr1=zeros(framecount,1); % holds the row numbers at which the
first TIV is located
reportr2=zeros(framecount,1); % holds the row numbers at which the
second TIV is located
% Find the pixel rows corresponding to the locations of the target
% intensity values
i=1;
while i<=framecount
    for row=1:yspanL
```

```
        if (Iint(row,bsmpint,i)<=intenIint1) && (reportr1(i)==0)
            reportr1(i)=row; % pixels, the pixel row of the first TIV
for the ith frame
        end
        if (Iint(row,bsmpint,i)<=intenIint2) && (reportr2(i)==0)
            reportr2(i)=row; % pixels, holds the row numbers at which
the second TIV is located
        end
    end
    i=i+fint;
end
% reportr1 and reportr2 now hold the pixel row locations of the TIVs

% Replot the column intensities so as to overlay the projected lines
hold off; figure; hold on;
i=1;
while i<=framecount
    plot(Iint(:,bsmpint,i));
    i=i+fint;
end
% Create plot labels
title(sprintf('Intensity for Pixel Column %d,\nEvery %dth Frame for %d
Frames',bsmp,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
%legend(legendcell);
% Plot lines across the target intensity values
intenvectIint1=zeros(yspanL,1)+intenIint1;
intenvectIint2=zeros(yspanL,1)+intenIint2;
intenvectIint3=zeros(yspanL,1)+c;
plot(1:yspanL,intenvectIint1,'k-');
plot(1:yspanL,intenvectIint2,'k-');
plot(1:yspanL,intenvectIint3,'k--');
% Projected lines are about to be plotted

% Find the slope of each line and find its projected intersection with
the
% intensity corresponding to the perceived location of the surface
BSLOCvect=zeros(usedcount,1); % pixels, burning surface locations
i=1;
j=1;
% Least square error line fit with p=ax+b
while i<=framecount
    x=[reportr1(i);reportr2(i)];
    p=[intenIint1;intenIint2];
    n=length(p);
```

```
    a=(sum(p)*sum(x.^2)-sum(x)*sum(x.*p))/(n*sum(x.^2)-sum(x)^2); % in,
y-intercept
    b=(n*sum(x.*p)-sum(x)*sum(p))/(n*sum(x.^2)-sum(x)^2); % in/s, slope
    BSLOCvect(j)=(c-a)/b;
    %begin plot
    longx=[x(1);x(1)+linefactor(testn)*(x(2)-x(1))];
    projline=b*longx+a;
    plot(longx,projline,'k-');
    %end plot
    j=j+1;
    i=i+fint;
end

BSLOCpix=BSLOCvect;
figure; plot(BSLOCpix,'kx');
xlabel('Frame');
ylabel('Pixel Row');
title('Pixel Location of the Bowl Bottom Edge');
grid;

meanrow=mean(BSLOCpix);
stdrows=std(BSLOCpix);
uppb=meanrow+2*stdrows; % upper bound
lowb=meanrow-2*stdrows; % lower bound
outsidecount=0;
for ii=1:usedcount
    rowi=BSLOCpix(ii);
    if rowi>uppb || rowi<lowb
        outsidecount=outsidecount+1;
    end
end
conflevel=100*(1-(outsidecount/usedcount));
fprintf('Mean Row: %.4f\r',meanrow);
fprintf('Standard Deviation of Rows: %.4f\r',stdrows);
fprintf('2*sigma: %.4f\r',stdrows*2);
fprintf('Confidence Level: %.4f\r\n',conflevel);

figure;
histogram(BSLOCpix,18); % 21,18
title(sprintf('Occurrences of Pixel Row over %d Frames\nof the Epoxy
Propellant Model',framecount));
ylabel('Number of Occurrences');
xlabel('Pixel Row');
annotstring=sprintf('Mean=%.4f\n\rStandard Deviation=%0.5f\n\rValues
Within 2*s: %.2f%%',meanrow,stdrows,conflevel);
tb1=annotation('textbox',[0.65,0.80,0.45,0.1],'string',annotstring);
tb1.FontSize=9;
tb1.LineStyle='none';

% Show an image
I(:,bsmp,1)=0;
figure; imshow(I(:,:,1));

% Show the cropped images with burning surface locations and center
line
i=1;
```

```
j=1;
while i<=framecount
    Iinttrace=Iint(:,:,i);
    Iinttrace(round(BSLOCpix(j)),:)=0;
    Iinttrace(:,bsmpint)=0;
    figure; imshow(Iinttrace);
    i=i+fint;
    j=j+1;
end
```

Edge_Tracking_ESP_Xray_SURFACE.m

```
% Edge_Tracking_ESP_Xray_SURFACE
% Matthew Denny
%{
INSTRUCTIONS:
Choose a set of frames in TIFF format from the image sequence of the
video
in which the propellant has already formed a discernible burning
surface
through the end of the time when the burning surface is easily
discernible,
and save these images in the MATLAB folder or the folder containing
this
script. In this script, reset the variables under the heading
"INITIALIZE
THESE VARIABLES BEFORE RUNNING". In the first while loop, set the
ithFileName command to fetch the correct file names. For each run,
adjust
the plot labels as appropriate. To find out what vartype should be, use
a
command like i1=imread('whateverfilename.tif') and see what type of
variable it gives in the "Workspace" table. Calibrate the distance for
a
new series of images using the imshow command or MSPaint.
The average regression rate is reported in the Command Window.
Examine the log file called "X-ray_MATLAB_LogFile.txt". The code for
creating the file is at the end of this script. The file is placed in
the
current directory, which is usually the MATLAB folder.
%
NOTES:
This code plots the burning surface locations identified across a range
of
pixel columns.
%}
%_____
____

clc
clear all
close all

% INITIALIZE THESE VARIABLES BEFORE RUNNING
testn=2; % UAH04A01 is 1. UAH04A02 is 2.
framecount=81; % total number of frames in the directory for analysis
fint=10; % interval of frames selected. (Use every fint'th frame)
bsmpline=[602;575];
bsmp=bsmpline(testn); % 602,575pixel corresponding to the burn surface
midpoint (bsmp)
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise (25
works)
if testn==1
    vartype='uint16'; % the variable type for the image data
else if testn==2
```

```
        vartype='uint8'; % the variable type for the image data
    end
end
fps=30; % frames per second
% altubeOD=0.625; % inches, outside diameter of aluminum tube case
% pixelsacrossaltubeOD=314; % 314,313pixels
% distcal=altubeOD/pixelsacrossaltubeOD; % in/pixel, distance
calibration
SF=[501.7;502.7]; % 501.7,502.7 ppi, scale factor
distcal=1/SF(testn); % in/pixel, distance calibration
%inten=147; % 38540, 147,156
%intenI=38540; % 38540, 147, target intensity value (TIV) for full
frames
intenIint1vect=[31000;117]; % 182, first TIV for cropped frames
intenIint2vect=[26000;78]; % second TIV for cropped frames
intenIint1=intenIint1vect(testn);
intenIint2=intenIint2vect(testn);
intenPERC=[23000;65];
c=intenPERC(testn);
rowstart=490; % 460,500 the row at which to start the search range
rowend=670; % 670 the row at which to stop the search range
bitdq=[2^16;2^8]; % bit depth quantity
xspan=[502:702;475:675]; % span of region of interest
yspan=[460:680;491:711]; % height of region of interest
xspanL=length(xspan(testn,:)); % length of vector
yspanL=length(yspan(testn,:)); % length of vector
bsmpint=bsmp-xspan(testn,1); % center pixel column of bowl for cropped
images



I=zeros(pxcount,pxcount,framecount,vartype);
Iint=zeros(yspanL,xspanL,framecount,vartype); % region of interest
Iints=zeros(yspanL,xspanL,framecount,vartype);
usedcount=0;
%figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
while i<=framecount
    ithFileName=sprintf('UAH04A02 (%d).tif', i); % 'pic' or 'UAH04A02'
    I(:,:,i)=imread(ithFileName); % store the image in a matrix
    % Use a filter to remove noise from the image:
    I(:,:,i)=medfilt2(I(:,:,i),[filtsize filtsize]);
    Iints(:,:,i)=imadjust(I(yspan(testn,:),xspan(testn,:),i));
    Iint(:,:,i)=I(yspan(testn,:),xspan(testn,:),i);
    %plot(Iints(:,bsmpint,i))
    %line1=zeros(yspanL,1)+intenIint1;
    %line2=zeros(yspanL,1)+intenIint2;
    %figure;hold on; plot(Iints(:,100,i)); plot(1:yspanL,line1);
plot(1:yspanL,line2)
    %hold off;
    i=i+fint;
    usedcount=usedcount+1;
end

% Manual Gray-Scaling
maxI=zeros(usedcount,1,vartype);
maxIint=zeros(usedcount,1,vartype);
```

```
minI=zeros(usedcount,1,vartype);
minIint=zeros(usedcount,1,vartype);
gs=1;
i=1;
while i<=framecount
    maxI(gs)=max(max(I(:,:,i)));
    maxIint(gs)=max(max(Iint(:,:,i)));
    minI(gs)=min(min(I(:,:,i)));
    minIint(gs)=min(min(Iint(:,:,i)));
    gs=gs+1;
    i=i+fint;
end
% Manually Scale
I=I-min(minI);
I=I.*(bitdq(testn)./(max(maxI)-min(minI)));
Iint=Iint-min(minIint);
Iint=Iint.*(bitdq(testn)./(max(maxIint)-min(minIint)));
%

% Plot one column of pixels for every frame after manual scaling
figure; hold on;
i=1;
studycol=50;
while i<=framecount
    plot(Iint(:,studycol,i))
    i=i+fint;
end
line1=zeros(yspanL,1)+intenIint1;
line2=zeros(yspanL,1)+intenIint2;
line3=zeros(yspanL,1)+c;
plot(1:yspanL,line1,'k');
plot(1:yspanL,line2,'k');
plot(1:yspanL,line3,'k--');
% Create plot labels
title(sprintf('Intensity for Pixel Column %d,\nEvery %dth Frame for %d
Frames',studycol,fint,framecount));
xlabel('Pixel Row');
ylabel('Pixel Intensity');
grid;
% Create a legend
legendcell=cell(usedcount,1);
i=1;
j=1;
while i<=framecount
    legendcell(j)=cellstr(sprintf('Frame %d',i));
    i=i+fint;
    j=j+1;
end
legend(legendcell);


% ITERATE THROUGH PIXEL COLUMNS
if testn==1
    pcol=65:135; % pixel columns to use (UAH04A01:560-650,
UAH04A02:500-650)
else if testn==2
```

```
        pcol=25:175;
    end
end
pcolnum=length(pcol);
pc=1; % counting index
BSLOC=zeros(usedcount,pcolnum); % burning surface location
for bsmpint=min(pcol):max(pcol)


% BURNING RATE DETERMINATION
% The variables rowstart and rowend define the range of rows in which
to
% search for a known intensity marking the burning surface to be
tracked.

%%%
reportr1=zeros(framecount,1); % holds the row numbers at which the
first TIV is located
reportr2=zeros(framecount,1); % holds the row numbers at which the
second TIV is located
% Find the pixel rows corresponding to the locations of the target
% intensity values
i=1;
while i<=framecount
    for row=1:yspanL
        if (Iint(row,bsmpint,i)<=intenIint1) && (reportr1(i)==0)
            reportr1(i)=row; % pixels, the pixel row of the first TIV
for the ith frame
        end
        if (Iint(row,bsmpint,i)<=intenIint2) && (reportr2(i)==0)
            reportr2(i)=row; % pixels, holds the row numbers at which
the second TIV is located
        end
    end
    i=i+fint;
end
% reportr1 and reportr2 now hold the pixel row locations of the TIVs

% Find the slope of each line and find its projected intersection with
the
% intensity corresponding to the perceived location of the surface
BSLOCvect=zeros(usedcount,1); % pixels, burning surface locations
i=1;
j=1;
% Least square error line fit with p=ax+b
while i<=framecount
    x=[reportr1(i);reportr2(i)];
    p=[intenIint1;intenIint2];
    n=length(p);
    a=(sum(p)*sum(x.^2)-sum(x)*sum(x.*p))/(n*sum(x.^2)-sum(x)^2); % in,
y-intercept
    b=(n*sum(x.*p)-sum(x)*sum(p))/(n*sum(x.^2)-sum(x)^2); % in/s, slope
    BSLOCvect(j)=(c-a)/b;
    BSLOC(j,pc)=yspanL-BSLOCvect(j);
    j=j+1;
    i=i+fint;
```

```
end

%%%

pc=pc+1;
end

i=1;
figure; hold on;
for i=1:usedcount
    plot(pcol,BSLOC(i,:))
end
title('2-D Burning Surface Locations');
ylabel(sprintf('%d - Pixel Row',yspanL));
xlabel('Pixel Column')


%{
% Calculate burning rate with least squares line fit, y=a+bt
t=0:(1/fps)*fint:(1/fps)*fint*(keepers-1); % s, say the first frame
occurs at time=0
t=t'; % s, time vector, invert the array
y=BSLOC; % in, locations of the burning surface
n=length(y);
coefa=(sum(y)*sum(t.^2)-sum(t)*sum(t.*y))/(n*sum(t.^2)-sum(t)^2); % in,
y-intercept
coefb=(n*sum(t.*y)-sum(t)*sum(y))/(n*sum(t.^2)-sum(t)^2); % in/s, slope
% Calculate theoretical linear y values on the fitted line at each t(i)
yfit=coefa+coefb.*t; % in, theoretical linear burning surface locations
Ssq=sum((y-yfit).^2); % in^2, sum of squared deviations of the data
from the fitted line
% Calculate the correlation coefficient r in squared form, r^2=Rsq
ydif=sum((yfit-mean(y)).^2); % in^2
Rsq=ydif/(Ssq+ydif); % unitless, minimum=0, maximum=1
%fprintf('Least Squares Burning Rate = %.4f in/s\n',abs(coefb)) % print
the result to the command line FIX THE
DECIMALS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%fprintf('Squared Correlation Coefficient = %.4f\n',Rsq) % print the
result to the command line FIX THE
DECIMALS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%}

%{
xline=[bsmpline bsmpline];
yline=[0.088 0.102]; % [0.082 0.1] or [0.088 0.102]
plot(xline,yline,'k-');
grid;
ylabel('Burning Rate (in/s)');
xlabel('Pixel Column')
title('Burning Rate vs Pixel Column')
legend('Every Frame','Every 5th Frame','Every 10th Frame','Every 20th
Frame',sprintf('Column %d',bsmpline));
%}
Iint(:,min(pcol),1)=0;
Iint(:,max(pcol),1)=0;
Iint(:,min(pcol),41)=0;
```

```
Iint(:,max(pcol),41)=0;
Iint(:,min(pcol),81)=0;
Iint(:,max(pcol),81)=0;
figure; imshow(Iint(:,:,1))
figure; imshow(Iint(:,:,81))
figure; imshow(Iint(:,:,41))
```

Background_Noise_Study.m

```
% Background_Noise_Study
% Matthew Denny
%{
INSTRUCTIONS:
Place the TIFFS of the video frames in the MATLAB directory, and adjust
the
file name in the first loop to read those files. Initialize framcount,
fint, bsmp, pxcount, filtsize, and vartype. Adjust the titles of the
plots
to include the acquisition conditions. To find out what vartype should
be, use a
command like i1=imread('whateverfilename.tif') and see what type of
variable it gives in the "Workspace" table.

The following block of notes is left over from Edge-
Tracking_ESP_Xray.m,
but they might be applicable here too.
%}
%_____________________________________________________________________
____
%{
%
NOTES:
3/3/2015
I could record the location of a certain plot feature for every picture
and
then calculate the burning rate from that. It might be easier to
isolate
that plot feature if I first subract each pair of sequential images to
obtain a picture of just the burning surface and obtain the burning
rate
from that. Either way, the burning rate comes from finding the
interface on
sequential pictures and calculating the spatial difference between
them.
%
In this code, the suffix f refers to filtering, and s refers to
scaling.
%
One possible method of tracking would be to find the maximum peak of
white
at the subtracted burning surface, and then find the points above and
below
at which the value is 0 (black), then define the midpoint between the
black
regions as the burning surface to be tracked between sequential images.
%
It is interesting to compare the plot of data before and after imadjust
is
used. On scaled plots, the curves are nearly the same, but more
features
are visible on the before-imadjusted plot.
%
3/9/15
```

```
Need to manually calibrate the distance by finding the number of pixels
across the case which correspond to 0.625 inches. This could possibly
be
automated, but it would still need the user to make sure that the row
of
pixels is a good row for finding the edges of the case.
%
4/7/15
Trouble solution: I was having a problem where the time-averaged
values of the pixel intensities were all approaching zero as the
framecount
got higher. The averaged image was all black. I made Isum average only
one
pixel, and I found that Isum was not summing properly, because it could
not
exceed 255 no matter how many frames there were. This was because Isum
is
an 8-bit variable, which can't count higher than 255 (starting from 0).
%}
%_____________________________________________________________________
____

clc
clear all
close all

% INITIALIZE THESE VARIABLES BEFORE RUNNING
framecount=40; % total number of frames in the directory for analysis
fint=1; % interval of frames selected. (Use every fint'th frame)
bsmp=575; % pixel corresponding to the burn surface midpoint (bsmp)
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise
vartype='uint16'; % the variable type for the image data
fps=30; % frames per second
altubeOD=0.625; % inches, outside diameter of aluminum tube case
pixelsacrossaltubeOD=314; % pixels
distcal=altubeOD/pixelsacrossaltubeOD; % in/pixel, distance calibration
inten=38540; % the intensity to find in order to track the surface,
this
%...number is found from using the coordinate tool on the first plot
that
%...this code generates
rowstart=400; % the row at which to start the search range
rowend=700; % the row at which to stop the search range
kV=35; % used in filename and plot titles

if strcmp(vartype,'uint8') % no need to initialize here
    bitd=8; % bit-depth, used later in plot titles
end
if strcmp(vartype,'uint16')
    bitd=16; % bit-depth, used later in plot titles
end

% Plot the intensities along one column for the frames specified by
fint
I=zeros(pxcount,pxcount,framecount,vartype);
```

```
figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
usedcount=0; % counts how many frames get used in the while loop
while i<=framecount
    ithFileName=sprintf('%dkV%dbit (%d).tif',kV,bitd,i); % determine
file name
    I(:,:,i)=imread(ithFileName); % store the image in a matrix
    % The above command (imread) stores the rows in the y coordinates
and
    %...the columns in the x coordinates due to the syntax (:,:,i).
    % Use a filter to remove noise from the image:
    %I(:,:,i)=medfilt2(I(:,:,i),[filtsize filtsize]); % median filter
uses
    %...the median value of the neighboring pixels with a neighborhood
size
    %...of "filtsize"
    %I(:,:,i)=imadjust(I(:,:,i)); % scales the image values to use the
full
    %...color range defined by "vartype"
    plot(I(:,bsmp,i))
    i=i+fint;
    usedcount=usedcount+1;
end

% Create plot labels
title(sprintf('Individual Intensities, Column %d,\n%d Frames, %dkV,
%dbit',bsmp,framecount,kV,bitd));
xlabel('Pixel Row');
ylabel('Color Intensity');

% AVERAGING LOOP
i=1;
Isum=zeros(pxcount,pxcount);
while i<=framecount
    Isum=Isum+double(I(:,:,i)); % the double function converts to
variable type 'double'. See note at top.
    i=i+fint;
end
Iavg=Isum./usedcount; % each pixel in the column/row is averaged with
the same
%...pixel in every frame

hold off; figure; % prepare a new plot figure
plot(Iavg(:,bsmp))
grid on; grid minor;
title(sprintf('Time-Averaged Intensity, Column %d,\n%d Frames, %dkV,
%dbit',bsmp,framecount,kV,bitd));
xlabel('Pixel Row');
ylabel('Time-Averaged Color Intensity');

%
% IMAGE GENERATOR

if strcmp(vartype,'uint8')
    Iavguint8=uint8(Iavg);
    figure; imshow(Iavguint8);
```

```
    Iavguint8adj=imadjust(Iavguint8);
    figure; imshow(Iavguint8adj);
end
if strcmp(vartype,'uint16')
    Iavguint16=uint16(Iavg);
    figure; imshow(Iavguint16);
    Iavguint16adj=imadjust(Iavguint16);
    figure; imshow(Iavguint16adj);
end
%
```

Bias_Isolation.m

```
% Bias_Isolation
% Matthew Denny
%{
The goal of this script is to subtract the primary, clean underlying
signal
from the time-averaged image in order to isolate the noise. The
isolated
noise will have positive and negative values which will require signed
variables.

INSTRUCTIONS:
Place the TIFFS of the video frames in the MATLAB directory, and adjust
the
file name in the first loop to read those files. Initialize framcount,
fint, bsmp, pxcount, filtsize, and vartype. Some of those are vectors
corresponding to different sets of images, so make sure they are the
same
length of vectors. Adjust the titles of the plots to include the
acquisition conditions. To find out what vartype should be, use a
command
like i1=imread('whateverfilename.tif') and see what type of variable it
gives in the "Workspace" table.

The following block of notes is left over from Edge-
Tracking_ESP_Xray.m,
but they might be applicable here too.
%}
%_____
____
%{
%
NOTES:
3/3/2015
I could record the location of a certain plot feature for every picture
and
then calculate the burning rate from that. It might be easier to
isolate
that plot feature if I first subract each pair of sequential images to
obtain a picture of just the burning surface and obtain the burning
rate
from that. Either way, the burning rate comes from finding the
interface on
sequential pictures and calculating the spatial difference between
them.
%
In this code, the suffix f refers to filtering, and s refers to
scaling.
%
One possible method of tracking would be to find the maximum peak of
white
at the subtracted burning surface, and then find the points above and
below
at which the value is 0 (black), then define the midpoint between the
black
```

```
regions as the burning surface to be tracked between sequential images.
%
It is interesting to compare the plot of data before and after imadjust
is
used. On scaled plots, the curves are nearly the same, but more
features
are visible on the before-imadjusted plot.
%
3/9/15
Need to manually calibrate the distance by finding the number of pixels
across the case which correspond to 0.625 inches. This could possibly
be
automated, but it would still need the user to make sure that the row
of
pixels is a good row for finding the edges of the case.
%
4/7/15
Trouble solution: I was having a problem where the time-averaged
values of the pixel intensities were all approaching zero as the
framecount
got higher. The averaged image was all black. I made Isum average only
one
pixel, and I found that Isum was not summing properly, because it could
not
exceed 255 no matter how many frames there were. This was because Isum
is
an 8-bit variable, which can't count higher than 255 (starting from 0).
4/10/15
Remember that the camera data is originally 14-bit color values that
are
stored in either 8-bit 16-bit format.
%}
%_____________________________________________________________________
____

clc
clear all
close all

% INITIALIZE THESE VARIABLES BEFORE RUNNING
framecount=40; % total number of frames in the directory for analysis
fint=1; % interval of frames selected. (Use every fint'th frame)
bsmp=602; % pixel corresponding to the burn surface midpoint (bsmp)
pxcount=1024; % pixel resolution of images (one dimension)
filtsize=25; % size of neighborhood for filter to remove noise
vartype='uint16'; % the variable type for the image data
bitd=[8 8 8 16 16 16];
truebitd=[6 6 6 14 14 14];
fps=30; % frames per second
altubeOD=0.625; % inches, outside diameter of aluminum tube case
pixelsacrossaltubeOD=314; % pixels
distcal=altubeOD/pixelsacrossaltubeOD; % in/pixel, distance calibration
inten=38540; % the intensity to find in order to track the surface,
this
%...number is found from using the coordinate tool on the first plot
that
```

```
%...this code generates
rowstart=490; % the row at which to start the search range
rowend=670; % the row at which to stop the search range
kV=[20 35 100 20 35 100]; % used in filename and plot titles
%{
if strcmp(vartype,'uint8') % no need to initialize here
    bitd=8; % bit-depth, used later in plot titles
end
if strcmp(vartype,'uint16')
    bitd=16; % bit-depth, used later in plot titles
end
%}
sets=length(kV);
I=zeros(pxcount,pxcount,framecount,sets,vartype);
Ifilt=zeros(pxcount,pxcount,framecount,sets,vartype);
% FOR LOOP ITERATES THE SEQUENCE OF IMAGES USED. It only changes the
voltage.
% Hold bit-depth and pixel column constant for now.
for seti=1:sets


% FILTER EACH IMAGE AND THEN AVERAGE THE FILTERED IMAGES
% Plot the intensities along one column for the frames specified by
fint
figure; hold on; % Prepare a figure for the plots
i=1; % start the count variable
usedcount=0; % counts how many frames get used in the while loop
while i<=framecount
    ithFileName=sprintf('%dkV%dbit (%d).tif',kV(seti),bitd(seti),i); %
determine file name
    I(:,:,i,seti)=imread(ithFileName); % store the image in a matrix
    % The above command (imread) stores the rows in the y coordinates
and
    %...the columns in the x coordinates due to the syntax (:,:,i).
    % Use a filter to remove noise from the image:
    Ifilt(:,:,i,seti)=medfilt2(I(:,:,i,seti),[filtsize filtsize]); %
median filter uses
    %...the median value of the neighboring pixels with a neighborhood
size
    %...of "filtsize"
    %I(:,:,i)=imadjust(I(:,:,i)); % scales the image values to use the
full
    %...color range defined by "vartype"
    plot(I(:,bsmp,i,seti))
    i=i+fint;
    usedcount=usedcount+1;
end


% Create plot labels
title(sprintf('Individual Intensities, Column %d,\n%d Frames, %dkV,
%dbit',bsmp,framecount,kV(seti),truebitd(seti)));
xlabel('Pixel Row');
ylabel('Color Intensity');

end % END THE SET LOOP
```

```
Isum=zeros(pxcount,pxcount,sets);
Ifiltsum=zeros(pxcount,pxcount,sets);
Iavg=zeros(pxcount,pxcount,sets);
Ifiltavg=zeros(pxcount,pxcount,sets);

for seti=1:sets

% AVERAGING LOOP
i=1;
while i<=framecount
    Isum(:,:,seti)=Isum(:,:,seti)+double(I(:,:,i,seti)); % the double
function converts to variable type 'double'. See note at top.
    Ifiltsum(:,:,seti)=Ifiltsum(:,:,seti)+double(Ifilt(:,:,i,seti));
    i=i+fint;
end
Iavg(:,:,seti)=Isum(:,:,seti)./usedcount; % each pixel in the
column/row is averaged with the same
%...pixel in every frame
Ifiltavg(:,:,seti)=Ifiltsum(:,:,seti)./usedcount;

hold off; figure; hold on; % prepare a new plot figure
plot(Iavg(:,bsmp,seti))
plot(Ifiltavg(:,bsmp,seti))
grid on; grid minor;
title(sprintf('Time-Averaged Intensity, Column %d,\n%d Frames, %dkV,
%dbit',bsmp,framecount,kV(seti),truebitd(seti)));
xlabel('Pixel Row');
ylabel('Time-Averaged Color Intensity');

end % END THE SET LOOP

% ISOLATE THE BIAS
Ibias=zeros(pxcount,pxcount,sets);
for seti=1:sets
    Ibias(:,:,seti)=Iavg(:,:,seti)-Ifiltavg(:,:,seti);
    hold off; figure; hold on;
    plot(Ibias(:,bsmp,seti))
    title(sprintf('Bias, Column %d,\n%d Frames, %dkV,
%dbit',bsmp,framecount,kV(seti),truebitd(seti)));
    xlabel('Pixel Row');
    ylabel('Bias Color Intensity');
end % END THE SET LOOP


% PLOT THE PERCENT BIAS
Ipercbias=zeros(pxcount,pxcount,sets);
hold off; figure; hold on;
for seti=1:sets
    Ipercbias(:,:,seti)=100.*Ibias(:,:,seti)./(2^truebitd(seti));
    plot(rowstart:rowend,Ipercbias(rowstart:rowend,bsmp,seti))
end
grid;
xlim([rowstart rowend]);
```

```
title(sprintf('Percent Biases, Column %d, %d Frames,\nPixel Rows %d-
%d',bsmp,framecount,rowstart,rowend));
xlabel('Pixel Row');
ylabel('Percent Bias Color Intensity');
% Make the legend
leg=cell(sets,1);
for seti=1:sets
    if kV(seti)==100
        if truebitd(seti)==14
            leg(seti)=cellstr(sprintf('%d kV, %d-
bit',kV(seti),truebitd(seti)));
        else % when kV=100 and truebitd=8
            leg(seti)=cellstr(sprintf('%d kV,   %d-
bit',kV(seti),truebitd(seti)));
        end

    else % where kV is not 100
        if truebitd(seti)==14
            leg(seti)=cellstr(sprintf('  %d kV, %d-
bit',kV(seti),truebitd(seti)));
        else % when kV is not 100 and truebitd=8
            leg(seti)=cellstr(sprintf('  %d kV,   %d-
bit',kV(seti),truebitd(seti)));
        end
    end
end
legend(leg);

%{
% IMAGE GENERATOR

if strcmp(vartype,'uint8')
    Iavguint8=uint8(Iavg);
    figure; imshow(Iavguint8);
    Iavguint8adj=imadjust(Iavguint8);
    figure; imshow(Iavguint8adj);
end
if strcmp(vartype,'uint16')
    Iavguint16=uint16(Iavg);
    figure; imshow(Iavguint16);
    Iavguint16adj=imadjust(Iavguint16);
    figure; imshow(Iavguint16adj);
end
%}
```

Denny_Radiography_Modeling_bowl_MULTIPLES.m

Input Parameters	
Source to Center-of-Sample Distance	20 in
Detector to Center-of-Sample Distance	20 in
Sample Case Outside Diameter	0.625 in
Sample Case Wall Thickness	0.065 in
Sample Length	0.8 in
Plexiglass absorption coefficient	0.1641 cm^2/g
Propellant absorption coefficient (magnesium)	0.1686 cm^2/g
Air absorption coefficient	0.2638 cm^2/g
Aluminum absorption coefficient	0.1704 cm^2/g
Quartz absorption coefficient	0.1684 cm^2/g
Plexiglass density	1.19 g/cm^3
Propellant density	1.5 g/cm^3
Air density	0.001225 g/cm^3
Aluminum density	2.7 g/cm^3
Quartz density	2.65 g/cm^3
Quartz window thickness (2 windows)	1.0 in (2.0 in for 2 windows)

```
%{
% Radioscopy Modeling - Translating Simulation with bowl burn surface
% Matthew Denny
% Advisor: Dr. Robert Frederick
% Propulsion Research Center
% The University of Alabama in Huntsville
%}

%{
% Notes:
% Code Description: This code is different from the original code in
that it puts a loop around the entire set of original loops and runs
the original
% set of loops for only one column pixels at a time. The diameter of
the burned surface of the propellant is changed with every new column
of pixels.
% When the loops begin, the parameter z is used differently from the
coordinate frame that defines the use of z for the original code. The
difference is
% that the z in the early loops in this code is used with zero located
at the top of the sample, whereas the zero point for z was the center
of the
% sample in the original code.


% The variable "zend" is not needed in this bowl code because the X-
rays are always perpendicular to the z-axis.

% The magnification factor "mag" is defined right before the loops, and
it currently must be an integer. This is not ideal.
```

```
% I added white space on both ends of the sample by concatenating white
columns that I created by estimating the unattenuated intensity as
I0/(L^2). This
% is not a great way to add white space but it worked.

% This code involves magnification from the distances, but it currently
magnifies only the vertical x axis. So even though the sample diameter
is 0.625",
% it measures twice that (1.25") in the result because of roughly 2x
magnification. This causes problems in scaling for the z-axis.

% When the windows are not used, 150000 is a good value for the I0.

% Put in a time-saver check for symmetry to copy one side of the
intensity map to the other side

%}

clc;
close all;
clear all;

energy=4000; % keV, the energy of the X-ray source
I0=254/60; % R/min, X-ray intensity at the source
I0=70000000; % R/min 70000000,150000

% Attenuation Properties
mu_px=0.1641; % cm^2/gram, X-ray absorption coefficient for plexiglass
mu_pr=0.1686; % cm^2/gram, X-ray absorption coefficient for magnesium
(similar to propellant)
mu_Al=0.1704; % cm^2/gram, X-ray absorption coefficient for aluminum
mu_air=0.2638; % cm^2/gram, X-ray absorption coefficient for air
mu_qz=0.1684;  % cm^2/gram, X-ray absorption coefficient for quartz
rho_px=1.19; % gram/cm^3, density of plexiglass, see "X-ray
Predictions.xlsx"
rho_pr=1.5; % gram/cm^3, density of propellant (green book), for
density of magnesium (1.74) see "X-ray Predictions.xlsx"
rho_Al=2.7;  % gram/cm^3, density of aluminum, see "X-ray
Predictions.xlsx"
rho_air=0.001225; % gram/cm^3, density of air, see "X-ray
Predictions.xlsx"
rho_qz=2.65; % gram/cm^3, density of quartz, see "X-ray
Predictions.xlsx"
att_px=mu_px*rho_px; % attenuation product for plexiglass
att_pr=mu_pr*rho_pr; % attenuation product for magnesium
att_Al=mu_Al*rho_Al; % attenuation product for aluminum
att_air=mu_air*rho_air; % cm^-1, attenuation product for air
att_qz=mu_qz*rho_qz; % cm^-1, attenuation product for quartz

% Define Cylinder Geometry
D1=0.625; % in, outside diameter of cylinder
D2=0.495; % in, diameter of cylinder wall closest to outside wall
% D3 is not defined here because the radius is used later in the first
loop as a function of z position. See R3.
D4=0; % in, nothing in the center. It's a bowl-shaped burning contour.
```

```
R1=D1/2; % in, outside radius of cylinder
R2=D2/2; % in, radius of cylinder wall closest to outside wall
% R3=D3/2; % in, radius of burned propellant surface, calculated in the
first loop
R4=D4/2; % in
L=0.8; % in, cylinder length
zend=L/2; % in, half the cylinder length, NOT USED IN THIS CODE
s=20; % in, y-component of distance from source to center of cylinder
cx=0; % in, x-coordinate of the center of the cylinder
cy=-s; % in, y-coordinate of the center of the cylinder
cz=0; % in, z-coordinate of the center of the cylinder

% Window Geometry of combustion bomb
tw=1*2.54; % cm, thickness of the windows. Each window is one inch
thick.
dw=1.25*2.54; % cm, diameter of the windows. For simplification, assume
the window attenuation applies to all points. Then this number is not
needed.

% Define Detector Geometry
H=2; % in, height of detector sensitive area2
W=2; % in, width of detector sensitive area THIS CODE DOES NOT NEED W
(EXCEPT THAT IT WILL CRASH UNLESS YOU COMMENT OUT ALL USES OF
W!!!!!!!!!!!!!!!!!!!!
t=20; % in, distance between detector and the center of the cylinder
Y=t+s; % in, distance between the source and the detector
dy=t; % in, y-coordinate of the detector
dx=W/2; % in, x-coordinate of the detector edge
dz=H/2; % in, z-coordinate of the detector edge
pixres=1000; % pixels/inch, pixel
resolution!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
halfp=(1/pixres)/2; % in, half of the width of a pixel to locate the
center of a pixel as the target for each X-ray

figure; hold on;
dcounter=1;
dcolor=['k' 'b' 'r'];
for depth=[0.4 0.5 0.6]

% Initialize the intensity matrix for the translating code
mag=Y/s; % magnification factor, FOR THIS CODE IT NEEDS TO BE AN
INTEGER, SO ADJUST s AND t AS NEEDED TO MAKE AN
INTEGER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
blanktopend=10; % This is used to make blank space before the
propellant sample.
Itrans=zeros(H*pixres,blanktopend+(L*pixres*mag));
Itrans(:,1:blanktopend)=I0/(Y^2); % The first columns of pixels of the
image is unattenuated radiation. They should
% be solid white just like all the places where the rays miss the
sample. Itrans is concatenated later at the end of every loop through a
column.

% New Loop: one iteration for each column of i rows
R3max=R2; % in, maximum radius of burned propellant
R3min=R4; % in, minimum radius of burned propellant
```

```
ztotal=L*pixres; % number of pixels in the length of the sample
zlevel=0.1; % in, distance of the top burning surface of the propellant
from the top of the sample
%R3f=@(z) R4+(R3max./zfinal).*z; % propellant radius as a function of z
steps down the center axis
bowlSF=0.7; % bowl Shape Factor is used in the function for R3 (next
line).
R3f=@(z) bowlSF.*(depth-z).^(1/2); % propellant radius as a function of
z steps down the center axis
for j=1:ztotal % the steps in the z direction (center axis)
    zpos=j/pixres; % the current position of z starting from the top of
the sample
    R3=R3f(zpos); % in, value of the function used to define the radius
of the contour of the burned surface of propellant
    % Adjustments
    if zpos<zlevel
        R3=R3max; % put empty space where propellant has burned away
    else
        if R3>R3max % Check if the contour function value exceeds the
case wall radius
            R3=R3max; % in, radius of burned surface of propellant
        else if R3<R4 % Check if the contour function value is less
than the radius of the center rod
                R3=R3min; % in, radius of burned surface of propellant
            end
        end
    end

    % Trace each ray from source to pixel to find pathlengths and then
intensity
    I=zeros(H*pixres,1); % Create a pixel vector of zero intensity at
each pixel
    detcount=(length(I)-1); % The number of rows of pixels of the
detector minus one for clean iteration.
    for i=0:1:detcount % For each row of pixels
        px=((length(I(:,1))/2)-i)/pixres-halfp; % in, Calculate the x-
coordinate of the pixel
        %for j=0:1:detcount % For each column of pixels
            pz=0; % in, Calculate the z-coordinate of the pixel
            % Calculate intermediate angles and total pathlength from
source to center of pixel
            if px==0
                phi=pi/2;
                hyp=abs(pz);
                theta=atan(hyp/Y);
                Rs=hyp/sin(theta); % Total pathlength from source to
pixel
                if pz==0
                Rs=Y;
                end
            else
                phi=atan(abs(pz/px));
                hyp=abs(px)/cos(phi); % Length of the hypotenuse of the
triangle formed by angle phi
                theta=atan(hyp/Y);
                Rs=hyp/sin(theta); % Total pathlength from source to
pixel
```

```
            end

            % For the outside diameter, D1:
            % Use the quadratic equation to calculate the pathlengths,
m, to the ray-cylinder intersection points for an infinite cylinder
            % Define the coefficients of the quadratic equation
            A=(sin(theta)^2)*(cos(phi)^2)+(cos(theta)^2);
            B=-2*s*cos(theta);
            C=s^2-R1^2;
            % For the positive square root in quadratic equation:
            mpos=(-B+sqrt(B^2-4*A*C))/(2*A);
            % For the negative square root in quadratic equation:
            mneg=(-B-sqrt(B^2-4*A*C))/(2*A);
            % Address the three possible cases for the results of the
quadratic equation: m can be real single-valued, real double-valued, or
complex for the outside diameter circle
            if mpos==mneg
                mcyl1=0; % If the ray is tangent to the cylinder, say
it misses the cylinder. Then it also misses the rest of the cylinders
inside of this one.
                mcyl2=0; % If the ray is tangent to the cylinder, say
it misses the cylinder.
                mcyl3=0; % If the ray is tangent to the cylinder, say
it misses the cylinder.
                mcyl4=0; % If the ray is tangent to the cylinder, say
it misses the cylinder.
            else if isreal(mpos) % For real double-valued m
                    z2intersect=mpos*sin(theta)*sin(phi); % in,
Calculate the length in the z-direction of the theoretical intersection
point between the circle and the ray for the point farther from the
source
                    z1intersect=mneg*sin(theta)*sin(phi); % in,
Calculate the length in the z-direction of the theoretical intersection
point between the circle and the ray for the point closer to the source
                    if z1intersect>zend % Check whether the ray passes
within the limited length of the actual cylinder
                        mcyl1=0; % in, Pathlength through the cylinder
for the outside diameter
                    else if z2intersect>zend % Check whether the ray
passes out of the end of the cylinder
                        m2=zend/(sin(theta)*sin(phi));
                        m1=mneg;
                        mcyl1=m2-m1; % in, Pathlength through the
cylinder for the outside diameter
                    else % Here the ray must be enclosed between the
round walls of the cylinder
                        m2=mpos;
                        m1=mneg;
                        mcyl1=m2-m1; % in, Pathlength through the
cylinder for the outside diameter
                        end
                    end

                    % For the next wall diameter,
D2:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
                    % Use the quadratic equation to calculate the
pathlengths, m, to the ray-cylinder intersection points for an infinite
cylinder
                    % Redefine the third coefficient of the quadratic
equation for the new diameter
                    C=s^2-R2^2;
                    % For the positive square root in quadratic
equation:
                    mpos=(-B+sqrt(B^2-4*A*C))/(2*A);
                    % For the negative square root in quadratic
equation:
                    mneg=(-B-sqrt(B^2-4*A*C))/(2*A);%
                    % Address the three possible cases for the results
of the quadratic equation: m can be real single-valued, real double-
valued, or complex for the second diameter circle
                    if mpos==mneg
                        mcyl2=0; % If the ray is tangent to the
cylinder, say it misses the cylinder.
                        mcyl3=0; % If the ray is tangent to the
cylinder, say it misses the cylinder.
                        mcyl4=0; % If the ray is tangent to the
cylinder, say it misses the cylinder.
                    else if isreal(mpos) % For real double-valued m
                            z2intersect=mpos*sin(theta)*sin(phi); % in,
Calculate the length in the z-direction of the theoretical intersection
point between the circle and the ray for the point farther from the
source
                            z1intersect=mneg*sin(theta)*sin(phi); % in,
Calculate the length in the z-direction of the theoretical intersection
point between the circle and the ray for the point closer to the source
                            if z1intersect>zend % Check whether the ray
passes within the limited length of the actual cylinder
                                mcyl2=0; % in, Pathlength through the
cylinder for the outside diameter
                            else if z2intersect>zend % Check whether
the ray passes out of the end of the cylinder
                                m2=zend/(sin(theta)*sin(phi));
                                m1=mneg;
                                mcyl2=m2-m1; % in, Pathlength through
the cylinder for the second diameter
                            else % Here the ray must be enclosed
between the round walls of the cylinder
                                m2=mpos;
                                m1=mneg;
                                mcyl2=m2-m1; % in, Pathlength through
the cylinder for the second diameter
                                end
                            end

                            % For the next wall diameter,
D3:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                            % Use the quadratic equation to calculate
the pathlengths, m, to the ray-cylinder intersection points for an
infinite cylinder
                            % Redefine the third coefficient of the
quadratic equation for the new diameter
                            C=s^2-R3^2;
```

```
                            % For the positive square root in quadratic
equation:
                            mpos=(-B+sqrt(B^2-4*A*C))/(2*A);
                            % For the negative square root in quadratic
equation:
                            mneg=(-B-sqrt(B^2-4*A*C))/(2*A);%
                            % Address the three possible cases for the
results of the quadratic equation: m can be real single-valued, real
double-valued, or complex for the outside diameter circle
                            if mpos==mneg
                                mcyl3=0; % If the ray is tangent to the
cylinder, say it misses the cylinder.
                                mcyl4=0; % If the ray is tangent to the
cylinder, say it misses the cylinder.
                            else if isreal(mpos) % For real double-
valued m

z2intersect=mpos*sin(theta)*sin(phi); % in, Calculate the length in the
z-direction of the theoretical intersection point between the circle
and the ray for the point farther from the source

z1intersect=mneg*sin(theta)*sin(phi); % in, Calculate the length in the
z-direction of the theoretical intersection point between the circle
and the ray for the point closer to the source
                                    if z1intersect>zend % Check whether
the ray passes within the limited length of the actual cylinder
                                        mcyl3=0; % in, Pathlength
through the cylinder for the outside diameter
                                    else if z2intersect>zend % Check
whether the ray passes out of the end of the cylinder

m2=zend/(sin(theta)*sin(phi));
                                            m1=mneg;
                                            mcyl3=m2-m1; % in,
Pathlength through the cylinder for the outside diameter
                                        else % Here the ray must be
enclosed between the round walls of the cylinder
                                            m2=mpos;
                                            m1=mneg;
                                            mcyl3=m2-m1; % in,
Pathlength through the cylinder for the outside diameter
                                        end
                                    end


                                    % For the next wall diameter,
D4:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                    % Use the quadratic equation to
calculate the pathlengths, m, to the ray-cylinder intersection points
for an infinite cylinder
                                    % Redefine the third coefficient of
the quadratic equation for the new diameter
                                    C=s^2-R4^2;
                                    % For the positive square root in
quadratic equation:
                                    mpos=(-B+sqrt(B^2-4*A*C))/(2*A);
                                    % For the negative square root in
quadratic equation:
```

```
                                    mneg=(-B-sqrt(B^2-4*A*C))/(2*A);
                                    % Address the three possible cases
for the results of the quadratic equation: m can be real single-valued,
real double-valued, or complex for the second diameter circle
                                    if mpos==mneg
                                        mcyl4=0; % If the ray is
tangent to the cylinder, say it misses the cylinder.
                                    else if isreal(mpos) % For real
double-valued m

z2intersect=mpos*sin(theta)*sin(phi); % in, Calculate the length in the
z-direction of the theoretical intersection point between the circle
and the ray for the point farther from the source

z1intersect=mneg*sin(theta)*sin(phi); % in, Calculate the length in the
z-direction of the theoretical intersection point between the circle
and the ray for the point closer to the source
                                            if z1intersect>zend % Check
whether the ray passes within the limited length of the actual cylinder
                                                mcyl4=0; % in,
Pathlength through the cylinder for the outside diameter
                                            else if z2intersect>zend %
Check whether the ray passes out of the end of the cylinder

m2=zend/(sin(theta)*sin(phi));
                                                m1=mneg;
                                                mcyl4=m2-m1; % in,
Pathlength through the cylinder for the outside diameter
                                                else % Here the ray
must be enclosed between the round walls of the cylinder
                                                    m2=mpos;
                                                    m1=mneg;
                                                    mcyl4=m2-m1; % in,
Pathlength through the cylinder for the outside diameter
                                                end
                                            end
                                    else
                                    mcyl4=0; % in, Pathlength through
the cylinder, the ray misses the cylinder
                                        end
                                    end


                                else
                                    mcyl3=0; % in, Pathlength through
the cylinder, the ray misses the cylinder
                                    mcyl4=0; % in, Pathlength through
the cylinder, the ray misses the cylinder
                                end
                            end


                        else
                            mcyl2=0; % in, Pathlength through the
cylinder, the ray misses the cylinder
                            mcyl3=0; % in, Pathlength through the
cylinder, the ray misses the cylinder
```

```
                            mcyl4=0; % in, Pathlength through the
cylinder, the ray misses the cylinder
                        end
                    end

                else
                    mcyl1=0; % in, Pathlength through the cylinder, the
ray misses the cylinder
                    mcyl2=0; % in, Pathlength through the cylinder, the
ray misses the cylinder
                    mcyl3=0; % in, Pathlength through the cylinder, the
ray misses the cylinder
                    mcyl4=0; % in, Pathlength through the cylinder, the
ray misses the cylinder
                end
            end

            % Pathlength through the quartz windows
            pw=2.54*2*tw/cos(theta); % cm, pathlength through the 2
windows of the combustion bomb.

            % Here choose the materials composing the various walls
(layers)
            mWall1=(mcyl1-mcyl2)*2.54; % cm, Pathlength through the
material of the first (outside) layer
            mWall2=(mcyl2-mcyl3)*2.54; % cm, Pathlength through the
material of the second layer
            mWall3=(mcyl3-mcyl4)*2.54; % cm, Pathlength through the
material of the third layer
            mWall4=mcyl4*2.54;         % cm, Pathlength through the
material of the fourth layer
            attWall1=att_Al*mWall1; % unitless, Total attenuation
coefficient times pathlength of the first (outside) layer
            attWall2=att_pr*mWall2; % unitless, Total attenuation
coefficient times pathlength of the second layer
            attWall3=att_air*mWall3; % unitless, Total attenuation
coefficient times pathlength of the third layer
            attWall4=att_Al*mWall4; % unitless, Total attenuation
coefficient times pathlength of the third layer
            attWindows=att_qz*pw; % unitless, Total attenuation
coefficient times pathlength of the windows
            sumOfAtt=attWall1 + attWall2 + attWall3 + attWall4 +
attWindows; % Sum of attenuation
            I(i+1,1)=(I0/(Rs^2))*exp(-sumOfAtt); % R/min, X-ray
intensity at target screen.
        %end % This "end" corresponds to the iteration across columns.
There is only one column per loop in this code, so the "for" loop is
eliminated.
    end
    abc=blanktopend+(mag*(j-1)); % counting variable used to put
multiple identical columns of the current I vector
    % into Itrans. It uses multiple columns of Itrans according to the
extent of magnification, mag.
    for magcount=1:mag
        Itrans(:,abc+magcount)=I;
    end
```

```
%     Imag=zeros(length(I),mag);
%     for magcount=1:mag % concatenate multiple identical columns of I
in order to account for magnification from the equipment spacing. Mag
must be integer.
%         Imag(:,magcount)=I;
%     end
%     Itrans=horzcat(Itrans,Imag);
end

Itrans_end=zeros(H*pixres,10);
Itrans_end(:,:)=I0/(Y^2);
Itrans=horzcat(Itrans,Itrans_end);

%{
% imwrite(I,'xrayimage1.png')
% imshow(xImage1)
%ximage1=image(I,[0 255]);
%mat2gray(ximage1)
%I2=histeq(I);
%imshow(I2);
%I3=uint8(round(I-1));
%}

% This was edited from I to Itrans for the translating code
% Imin=min(min(Itrans));
% Imax=max(max(Itrans));
% Inew=round((Itrans-Imin)*256/(Imax-Imin));
% image(Inew)
Itrans=Itrans';
%image(Itrans)
%colormap(gray(256))
%axis image % sets the aspect ratio to give square pixels

%figure;
centercolumn=1000;
plot(Itrans(11:1610,1000),dcolor(dcounter)) % 11:1610,1000
dcounter=dcounter+1;
end
xlabel('Pixel Row');
ylabel('Intensity');
title(sprintf('Intensity vs Pixel Row for Pixel Column
%d',centercolumn));
legend('Image A','Image B','Image C');
```

artificial augmentation by regression target intensity.m

```
% Artificial augmentation by regression target intensity

clc
clear all
close all

% Notes:
% The number of burn surfaces can be varied by changing scount.
% The outside diameter of the propellant is sod.
% There are two resolutions: xres and yres.

scount=5; % number of burn surfaces
sod=0.5; % in, outside diameter of burn surfaces
xres=0.0001; % in, step size for span
xstepcount=sod/xres; % number of steps across the diameter
xvect=xres:xres:sod; % vector containing horizontal distance of every
increment
empspace=10; % makes empty space
s=zeros(xstepcount,scount); % initialize
sinc=0.15; % in, regression increment distance of the burn surface
sstart=0.8; % in, starting height of burning surface, ie total unburned
height of propellant
sbot=0; % in, bottom of the case, which is the lowest point to which
the burning surface can go
I0=1000; % R/min, initial intensity

% Attenuation Properties
mu_px=0.1641; % cm^2/gram, X-ray absorption coefficient for plexiglass
mu_pr=0.1686; % cm^2/gram, X-ray absorption coefficient for magnesium
(similar to propellant)
mu_Al=0.1704; % cm^2/gram, X-ray absorption coefficient for aluminum
mu_air=0.2638; % cm^2/gram, X-ray absorption coefficient for air
rho_px=1.19; % gram/cm^3, density of plexiglass, see "X-ray
Predictions.xlsx"
rho_pr=1.5; % gram/cm^3, density of propellant (green book), for
density of magnesium (1.74) see "X-ray Predictions.xlsx"
rho_Al=2.7;  % gram/cm^3, density of aluminum, see "X-ray
Predictions.xlsx"
rho_air=0.001225; % gram/cm^3, density of air, see "X-ray
Predictions.xlsx"
att_px=mu_px*rho_px; % attenuation product for plexiglass
att_pr=mu_pr*rho_pr; % attenuation product for magnesium
att_Al=mu_Al*rho_Al; % attenuation product for aluminum
att_air=mu_air*rho_air; % cm^-1, attenuation product for air



% Variables for plotting the intensity through each burning surface
yc=@(x,R) -sqrt(R^2-x^2); % the y position on the circular contour
region
yres=xres; % in, step size for intensity plot (independent variable
plotted in horizontal axis with dependent variable running vertically
in order to be able to line it up with the burning surface lines)
ybot=0; % in, lowest point reached by lowest burning surface iteration
```

```
ystepcount=(sstart-ybot)/yres; % total number of steps in the y-axis
intf=@(x) I0*exp(-att_pr*x); % R/min, Intensity function, Beer's Law
for intensity attenuation
% Create yvect. A vector cannot be created counting downward (like
sstart:yres:ybot), so use a for loop
yvect=zeros(ystepcount,1);
for y=1:ystepcount
    yvect(y)=sstart-(y-1)*yres; % in, vertical position
end

I=zeros(ystepcount,scount); % intensity vector
p1=0;
p2=0;


% This loop calculates the contours and the intensities
for i=1:scount
    rc=0.05+(i-1)*sinc; % in, radius of the circle of the current burn
surface.
    sheight=sstart-(i-1)*sinc; % in, depth of the top of the current
burn surface
    s(:,i)=sheight; % in, burning surface contour flat across before
the circle is calculated
    for j=1:xstepcount % this loop creates the circular contour
        xc=(xstepcount/2-j)*xres; % in, xc is measured from the center
of the sample for each surface
        if isreal(yc(xc,rc))
            s(j,i)=sstart+yc(xc,rc);
        end
        if s(j,i)>sheight
            s(j,i)=sheight; % in
        end

    end

    % Intensity for the current burn surface

    for y=1:ystepcount
        path=-1; % meaningless initial value, used as a check later in
this loop
        if s(1,i)<yvect(y)
            path=0; % in, pathlength through propellant for X-ray
attenuation
        else if min(s(:,i))>yvect(y)
                path=sod; % in, pathlength through propellant for X-ray
attenuation
            else % the vertical position must be within the bowl region
                for j=1:xstepcount
                    if s(j,i)<=yvect(y) && path==-1 % find the cross-
section at the current depth. yvect might not have the resolution to
equal every s(j,i) or even to fit within [yvect(y) yvect(y+1)], so the
cross-section must be found by "less-than-or-equal-to" comparison
                        if s(j+1,i)~=s(j,i) % if the next height over
is not the same height
```

```
                            if j+1>(xstepcount/2) % if the next value
over is more than half of the sample width (which could happen at the
bottom of the bowl)
                                path=sod; % in, pathlength through
propellant for X-ray attenuation
                            else
                                path=2*j*xres; % in, pathlength through
propellant for X-ray attenuation, the bowl is symmetric so stop at the
first good edge and double that web thickness
                            end
                        end
                    end
                end
            end
        end

        I(y,i)=intf(path); % R/min, intensity value after attenuation
through the propellant cross-section

    end % closes the intensity loop
end % closes big loop

% Plot the contours
figure; hold on; % prepare a plot figure
for i=1:scount
    plot(xvect,s(:,i),'k-')
end
xlim([0 0.5]);
ylim([0 0.9]);
xlabel('Width (inches)');
ylabel('Height (inches)');

% Plot the intensities
figure; hold on; % prepare another plot figure
for i=1:scount
    plot(yvect,I(:,i),'k-')
end


% Plot the intensities
figure; hold on; % prepare another plot figure
for i=1:scount
    plot(I(:,i),yvect,'k-')
end
xlabel('Intensity (R/min)');
ylabel('Height (inches)');
```

REFERENCES

[1] Sawka, Wayne, “Highly Insensitive, Solid State, Electrically-Controlled Propellants for Divert Propulsion”, Digital Solid State Propulsion, LLC. <http://www.virtualacquisitionshowcase.com/document/1430/briefing>.

[2] Sutton, G. P., and Biblarz, O., "Rocket Propulsion Elements", 8th ed., John Wiley & Sons, Inc., Hoboken, New Jersey, 2010.

[3] Sawka, Wayne, “Safe Electric Solid Propellants”. Digital Solid State Propulsion, LLC, Navy Opportunity Forum, June 2009. <www.dtic.mil/ndia/.../WednesdayLandmarkAWayneSawka.pdf>.

[4] Nicholas, A., Sawka, W., “SpinSat Mission Overview”, Small Satellite Conference, Technical Session I: Advanced Technologies I, 2013.

[5] Fry, R.S., "Solid Propellant Subscale Burning Rate Analysis Methods for U.S. and Selected NATO Facilities”, AD-A399 211, Jan. 2002.

[6] Hessler, R. O., "Consistent Definitions for Burning-Rate Measurement in Solid-Rocket Motors," Combustion, Explosion, and Shock Waves, Vol. 36, No. 1, 2000.

[7] Frederick, R.A., "Slag accumulation and slosh measurements with real-time radioscopy",, Journal of Flow Visualization & Image Processing, Vol. 3, Issue 2, pp. 165-176, 1996.

[8] Frederick, R. A., Jr., Williams, B. M., Bean, S. B., "Predicting X-Rays for Dynamic

Flaw Detection in Solid Rockets", SAE Paper 911152, April 1991.

[9] Juhasz, A.A., "Round Robin results of the closed bomb and strand burner", JANNAF Combustion Subcommittee, CPIA 361, July 1982.

[10] Frederick, R., and Traineau, J-C, "Evaluation of Methods for Solid Propellant Burning Rate Measurements", Chapter 5, "Non-Intrusive Techniques", NATO/RTO Advisory Report, AVT Working Group 016, edited by Ronald S. Fry, JHU/CPIA, January 2002.

[11] Osborn, J.R., Burick, R.J., and Panella, R.F., "Continuous Measurement of Solid Propellant Burning Rates", Review of Scientific Instruments, Vol. 37, No. 1, 1966, pp. 86-92.

[12] Godai, T., "Studies of solid propellant combustion with pulsed radiography", NASW 4005, August 1987.

[13] Cauty, F., Seryn, N., and Gramer, D., "Solid-Fuel Pyrolysis Phenomena and Regression Rate, Part 2: Measurement Techniques," Fundamentals of Hybrid Rocket Combustion and Propulsion, edited by Chiaverini, M. and Kuo, K., Vol. 218,, Progress in Astronautics and Aeronautics, AIAA, Reston, VA, 2007.

[14] "The Electromagnetic Spectrum", HyperPhysics, Georgia State University. <http://hyperphysics.phy-astr.gsu.edu/hbase/ems1.html#c1>.

[15] Oaks, A. E., "Nondestructive testing of Scout rocket motors", NASA CR-2013 20111116, 1972.

[16] Frederick, R. A., "Predicting X-Ray Images for Burning Solid-Propellant Rocket Motors", AIAA 91-0305, Jan 1991.

[17] Kuo, K.K., Hsieh, W.H., Ritchie, S.J., “Study of Dynamic Behavior and Hazards of VHBR, LOVA, And PSS Propellants under Rapid Ignition And Combustion Conditions by Real-Time X-Ray Radiography”, Dec 1991.

[18] Kuo, K. K., “Fundamental Phenomena on Fuel Decomposition and Boundary-Layer Combustion Processes with Applications to Hybrid Rocket Motors. Part 1; Experimental Investigation” NASA-CR-201843, June 1996.

[19] Anderson, M.G., Seely, J.F., Hayes, T.W. "Concurrent Application of Real-Time Radiography and Ultrasonics During SRM Test." AEDC-PA95-232, ASNT Fall Conference and Quality Testing Show, Dallas, TX, Oct. 16-20, 1995.

[20] Tauzia, J.M., Lamarque, P., "Solid Rocket Propellant Behavior During Static Firing Test Using Real Time X-Ray Radioscopy", AGARD-CP-598 Conference Proceedings “Advanced Non-Intrusive Instrumentation for Propulsion Engines”, Brussels, Oct. 20-24, 1997.

[21] Fry, D.A., "Recent developments in electronic radiography at Los Alamos." Penetrating Radiation Systems and Applications, SPIE Vol. 3769, 0277-786X/99, July 1999, pp 111-123.

[22] Salizzoni, R. M., "Study of Combustion Behavior of Very High Burning Rate Using a Real-Time X-Ray Radiography System", The Pennsylvania State University and U.S. Army Ballistic Research Laboratory, 1991.

[23] Chiaverini, M. J., "Regression Rate Behavior of Hybrid Rocket Solid Fuels", Journal of Propulsion and Power, Vol. 16, No. 1, Jan-Feb 2000.

[24] Anderson, M. "Real-time radiography support for Titan LAM." AIAA-92-3823, July

1992.

[25] Pressley, H., "Measurement of Burning Rate in Solid Rocket Motors", AIAA-83-0481, Jan 1983.

[26] Olaniyi, Bisola, "Accuracy of real time radiography burning rate measurement: a thesis", The University of Alabama in Huntsville, 2010.

[27] Cortopassi, A., “Erosion of Carbon-cloth Phenolic Nozzles in Rocket Motors with Aluminized Solid Propellant”, Ph.D Dissertation, Pennsylvania State University, Dec. 2012.

[28] Efimov, V. G., "Multi-view radiation imaging: New opportunities of testing the internal structure of solid propellant propulsion systems", Nondestructive Testing and Evaluation, Vol. 27, Issue 4, Dec. 1, 2012, pp 327-333.

[29] Efimov, V.G., “Geometric analysis of image contours formation on the X-ray screen” Nondestructive Testing and Evaluation, Vol. 19, Issue 1-2, March 2003, pp 43-52.

[30] Saffell, Ryan, “Characterization of Gaseous Oxygen and Methane Injectors Under High Pressure Combustion”, M.S. Thesis, The University of Alabama in Huntsville, Huntsville, AL, 2009.

[31] "Conax EG Series Electrode Sealing Glands," [Online]. Available: http://www.electro-meters.com/conax/eg_glands/. [Accessed 17 03 2015].

[32] Beckwith, T.G., Marangoni, R.D., Lienhard, J.H. V, “Mechanical Measurements”, 6th Ed., Pearson Education, Inc., Upper Saddle River, NJ, 2007.

[33] Doering, E.R., Basart, J.P., Gray, J.N., "Three-dimensional flaw reconstruction and

dimensional analysis using a real-time X-ray imaging system", NDT & E International, Vol. 26, Issue 1, Feb. 1993, Pages 7-17.

[34] Pressley, H. M., Glick, R. L., “In-Situ Burning Rate Determination Using Flash Radiography”, 21st JANNAF Combustion Meeting, CPIA Publication 412, Vol. I, Chemical Propulsion Information Agency, Laurel, MD, Oct. 1984, pp. 261-273.

[35] Ravindran, V. R., Sreelakshmi, C.V., “Development of CT and 3D-CT using flat panel detector based real-time digital radiography system”, International Conference on the Applications of Computerized Tomography, CT 2008;, Kanpur; India; Vol. 1050, February 1.

[36] Ravindran, V. R., Sreelakshmi, C., Vibinkumar, S., “Digital radiography-based 3D-CT imaging for the NDE of solid rocket propellant systems”, Insight: Non-Destructive Testing & Condition Monitoring., Vol. 50, Issue 10, Oct. 2008, pp564-568.

[37] Armistead, R. A., "Advanced X-ray systems for nondestructive inspection and contraband detection", Penetrating Radiation Systems and Applications, SPIE Vol. 3769, 0277-786X/99, July 1999.

[38] Peugeot, R. S., "Real Time Radioscopic Examination", National Board BULLETIN, 58th General Meeting, 1989.

[39] White, K. J., McCoy, D. G., "Flash X-ray with image enhancement applied to combustion events", US Army Ballistic Research Laboratory, Memorandum Report BRL-MR-3457, Aug 1985.