Optical Tool Setting and Control for Precision Lathe

BY

Charles Geoffrey Haning

Submitted to the graduate degree program in Mechanical Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Chairperson

Committee members

Date Defended

The Thesis Committee for Charles G. Haning certifies that this is the approved

Version of the following thesis:

Optical Tool Setting and Control for Precision Lathe

Committee:

____
Chairperson

____

____

Date Approved: ___________________

Acknowledgements

I would like to thank my employer for the opportunity to participate in the Technical Fellowship Program, and the countless employees who helped along the way. I must especially thank John Fines and Todd Mills for the guidance they have provided.

I must thank my family for their love and support. Tiffany, I couldn't have done this without you and your dedication -- I Love You!

1. Introduction

1.1 Motivation

A commercial, fast, and accurate way of determining the tool nose radius necessary for machining complex profiles on a lathe via an in process technique does not exist. There are many different compensation methods that can be used on the machine (i.e. leadscrew and volumetric compensation), the part (i.e. dynamic fixture offsets), and the tool (i.e. laser tool setters, contact tool setters, offline tool measurement) to improve part tolerance and finish. Commercially available contact and non-contact tool setters do not facilitate the measurement of the actual tool nose radius, and in high precision contour work, the tool nose radius is a critical factor. Further, by utilizing the Application programming interface (API) of an open control system, the measurements and the corresponding values can be programmatically entered into the Computer Numerical Control (CNC) without user intervention. By developing and implementing a vision based measurement approach, tool nose radius can be calculated in addition to tool offsets, with the goal of reducing tolerance loss from incorrect tool nose radius and tool offset values.

1.2 Research Methodology

The focus of this research was on the development of a machine vision system to provide automatic tool offset and tool nose radius compensation on a lathe used on the shop floor. To accomplish this work, a T-bed lathe which utilized a PC-based control was used for development and testing. National Instrument Labview 8.5 with the Vision Development System as well as National Instruments Vision Builder for Automated Inspection was utilized to develop the machine vision software. Microsoft Visual Basic 2005 was used to write the interface between the vision software and the PC-based controller.

1.3 Thesis Structure

This thesis is organized into six chapters. The first chapter discusses motivation and introduces the methodology used to perform the research. The second chapter discusses the background information, including the systems, materials and machines used in addition to theory of operation. The third chapter is a summary of finding during the literature review. The fourth chapter discusses the Experimental Design and explains how data was recorded. The fifth chapter discusses the results of the work performed. The sixth chapter includes Conclusions and Recommendations of the current work.

2. Background

The focus of this research was two-fold:

1) Develop a machine vision system capable of resolving to 5 μm for the tool nose radius (TNR), X and Z offsets.

2) Develop interface software to allow the machine vision software to automatically transmit tool nose radius and offset values to a machine tool control (CNC).

2.1 Vision Hardware

There are three main components that make up the hardware necessary for a vision system: Camera, Optics and Lighting. Relevant theory and the basis for choosing the current hardware are explained here.

2.2 Camera

The Basler Pilot pia2400-12gm is a monochrome interline CCD camera, with a 5 megapixel resolution (2458x2040). It uses gigabit Ethernet as its digital interface with the PC. It is capable of 12 fps at full resolution.[1] It was selected because of its sustained frame rate, its sensitivity to light, and for the digital interface which facilitated the long cable distance between the camera and the CNC control computer.

Camera Theory

There are two main technologies behind current imaging electronics. These are Charge Coupled Device (CCD) and Complimentary Metal Oxide Semiconductor (CMOS). [2] The main differences in performance between CCD and CMOS sensors according to Basler are shown in Table 1.

Parameter	**CCD**	**CMOS**
Fill Factor	HIGH	MED
Dark Noise	LOW	MED
S/N Ratio	HIGH	MED
Dynamic Range	MED	HIGH

Table 1: Adapted from [2]

For the purposes of gauging, CCD presents several advantages, mainly: Higher Signal to Noise (S/N) ratio, lower dark noise, and higher fill factor. A higher signal to noise and lower dark noise ratio meant less noise would be introduced to the image data.[3] The higher fill factor can result in sharper pictures due to the ability of the camera to read the image from the sensor producing less motion blur. Additionally, CCD cameras have better response at lower wavelengths of light, which help improve overall system resolution. A plot of a CCD and CMOS image sensor relative response to different wavelengths of light can be seen in Figure 1.

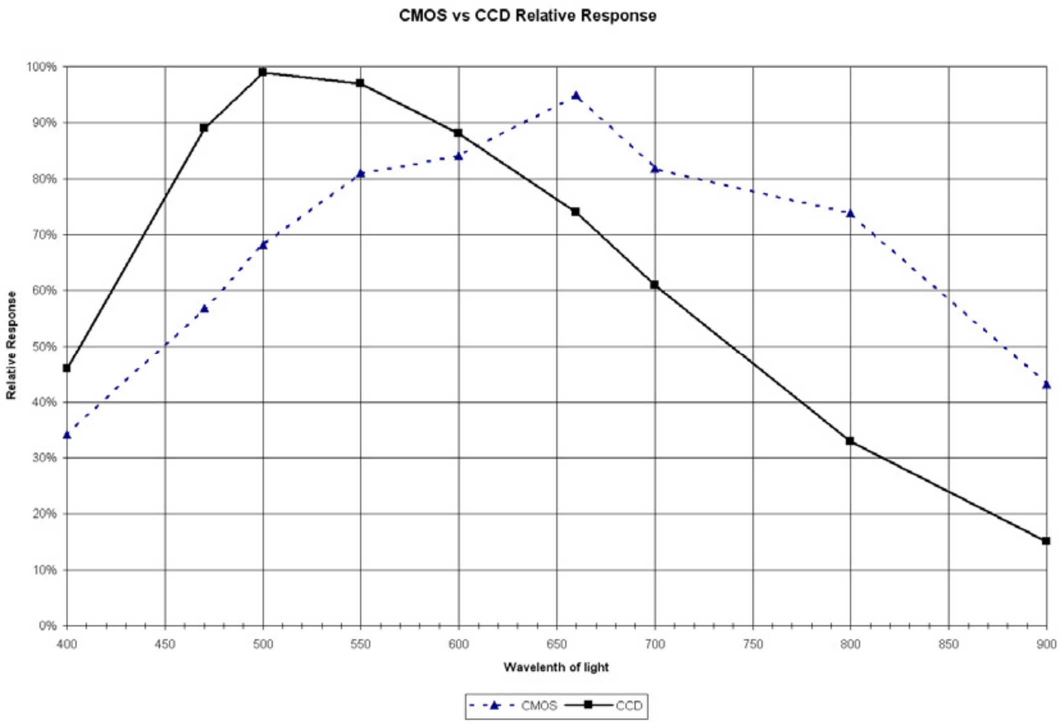


Figure 1: Camera Response Curve

CCD imagers acquire an image in a three step process. First, photodiodes or MOS photo capacitors are exposed to light. Second, Charge Transfer occurs thru the use of shift registers, which moves the packets of charge within the silicon substrate. Finally, output amplification via Variable Gain Control (VGC) and Charge-to-Voltage conversion via Analog to Digital conversion (ADC) take place. [4] [5] An example diagram of a CCD sensor is shown in Figure 2. With the particular CCD sensor utilized in the camera, the imager is broken down into left and right areas and pixels are clocked out from both the left and right side of the sensor simultaneously to aide in a high fill factor.

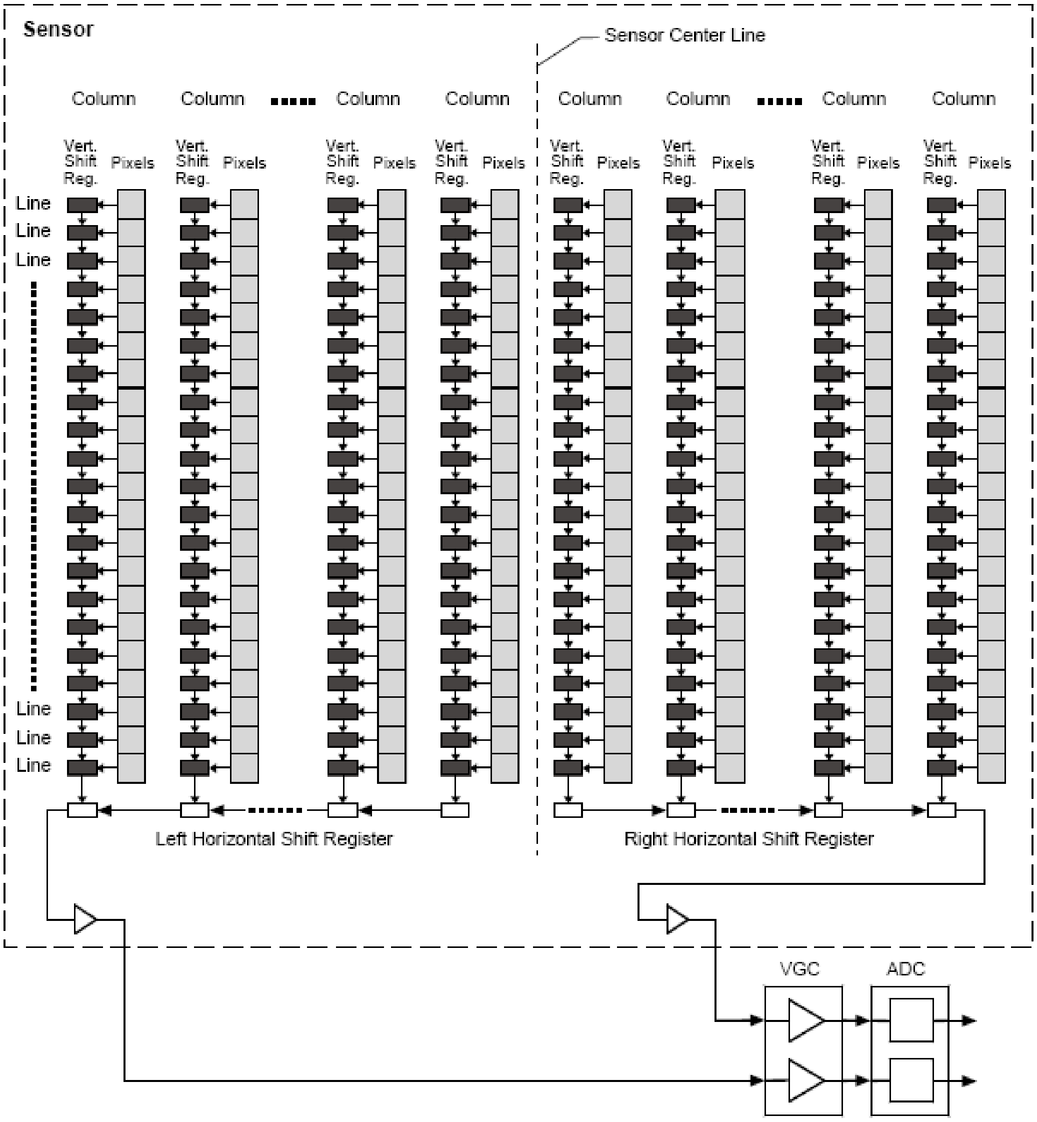


Figure 2: CCD Sensor

Digital Interface

In previous generations of machine vision cameras, output was analog video and required a frame grabber to digitize the signal. Many cameras now output data thru a digital interface because the camera performs the analog to digital conversion process internally. The result is improved noise immunity of the transmitted data. The particular camera used is capable of outputting quasi 16 bit (65536 gray levels), 12-

bit (4096 gray levels) and 8-bit pixel depth (256 gray levels) images at up to 12 frames per second (fps) at its full resolution of 2058x2456 pixels. It outputs this data over a digital interface called GigE, which is an implementation of gigabit Ethernet for industrial machine vision applications.

GigE for vision applications has been standardized by the Automated Imaging Association (AIA), and represents several standardized digital interfaces used for industrial machine vision, such as Firewire (IEEE 1394a and 1394b), and Camera Link (AIA Camera Link). The GigE standard ensures the behavior of the host and the camera, and also provides a standard approach to access any GigE compliant camera. GigE makes it possible for data transfer up to 100 meters.[6] GigE was used because of its relative low interface cost (a standard gigabit Ethernet adapter card is all that is needed), and for the distance of data transfer needed.

2.3 Optics System

The optics system is made up of three components. The first and second is a Zoom 6000 assembly from Navitar Inc. that included the zoom lens which creates adjustability in the field of view (FOV) and zoom of the system as well as a tube lens to focus the image onto the CCD sensor of the camera. The third is a Mitutoyo, Inc. 5X Infinity Corrected Flat Field Microscope objective. Together, these three components created a range in the system field of view (FOV) of (3.21mm x 2.41mm) at a pixel resolution of 2.1 microns to (0.77mm x 0.58mm) with a pixel resolution of 0.6 microns, respectively. [7]

The Mitutoyo lens is an infinity corrected objective. This means the image it produces is projected to infinity (parallel rays), and an additional lens is required to focus the image onto the camera sensor. [8] This additional lens is called a tube lens. A diagram labeling the components is shown in Figure 3, and a ray diagram of the lens system is shown in Figure 4.

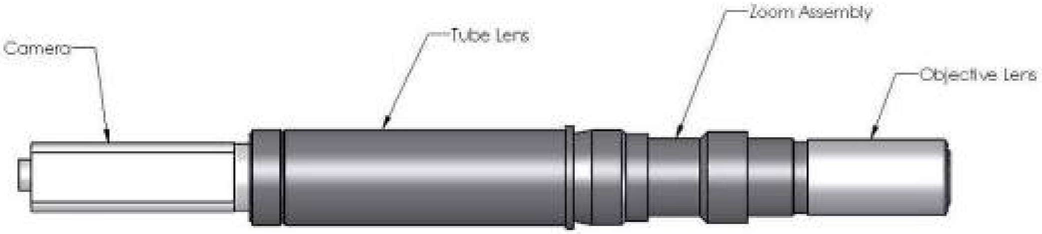


Figure 3: Lens Assembly

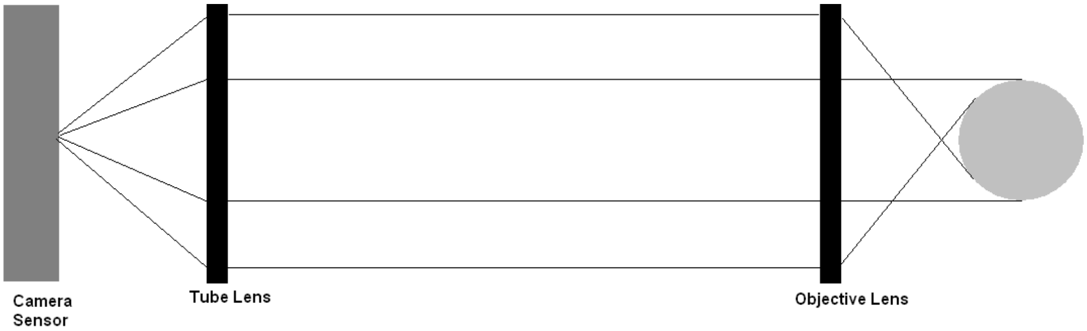


Figure 4: Lens Diagram

The Mitutoyo objective is also a plan achromat design. A plan objective corrects for color and spherical aberrations better than other designs and has a flat field over 95% of the imaging area.[9] The 95% area radius equals $R_{.95} = \sqrt{.95 \cdot R^2} \ or \ .9767 \cdot R$.

There were several parameters that had to be determined during the design process. A tradeoff had to be made between the FOV and system resolution because the two are inversely related. The optics system was designed to resolve features to 2 µm for the desired field of view (2.6mm x 2.0mm). This allowed the largest radius tool up to 1.5mm to fully be in the field of view of the camera. Additionally this provided the user a larger window to approach when positioning the tool within the FOV.

According to Zuech, accuracy and repeatability of a machine vision system can be improved by averaging several images together, and the improvement equals the square root of the number of images averaged. [5] This assumes negligible outside factors such as vibration and lighting variation.

Depth of focus (DOF) was an important consideration as it was 2 orders of magnitude smaller than the FOV, or 20 μm (0.000787 inches). The calculated DOF indicated the completed system would be dependant on accurate tool height adjustment prior to making tool nose radius and offset adjustment. [2] [10]

In all optic systems, there is a fundamental limit to the resolving power of the system imposed by diffraction. Diffraction occurs because of interference from obstacles result in secondary waves interfering with the original wave. The Rayleigh criteria describes the resolving power of the system as the minimal distance where two objects separated by some distance can still be distinguished as two objects through the imaging system. The Rayleigh criteria for microscope objectives is defined as $R = \frac{0.61\lambda}{NA}$, where the NA is equal to the numerical aperture of the lens, given by the lens manufacturer. This was found to be 2.05 μm for the designed system (NA=.14, λ=470 nm).[10]

2.4 Lighting

In gauging applications, backlighting is the preferred method of illumination when the object being measured has a 2-D profile that is planar. Backlighting provides high contrast differentiation for the object being measured by placing the light source inline with the camera, but behind the object. This type of lighting can be further enhanced by making the light output collimated. A collimated light source provides a parallel, unidirectional beam of light. This concept is illustrated in Figure 5.

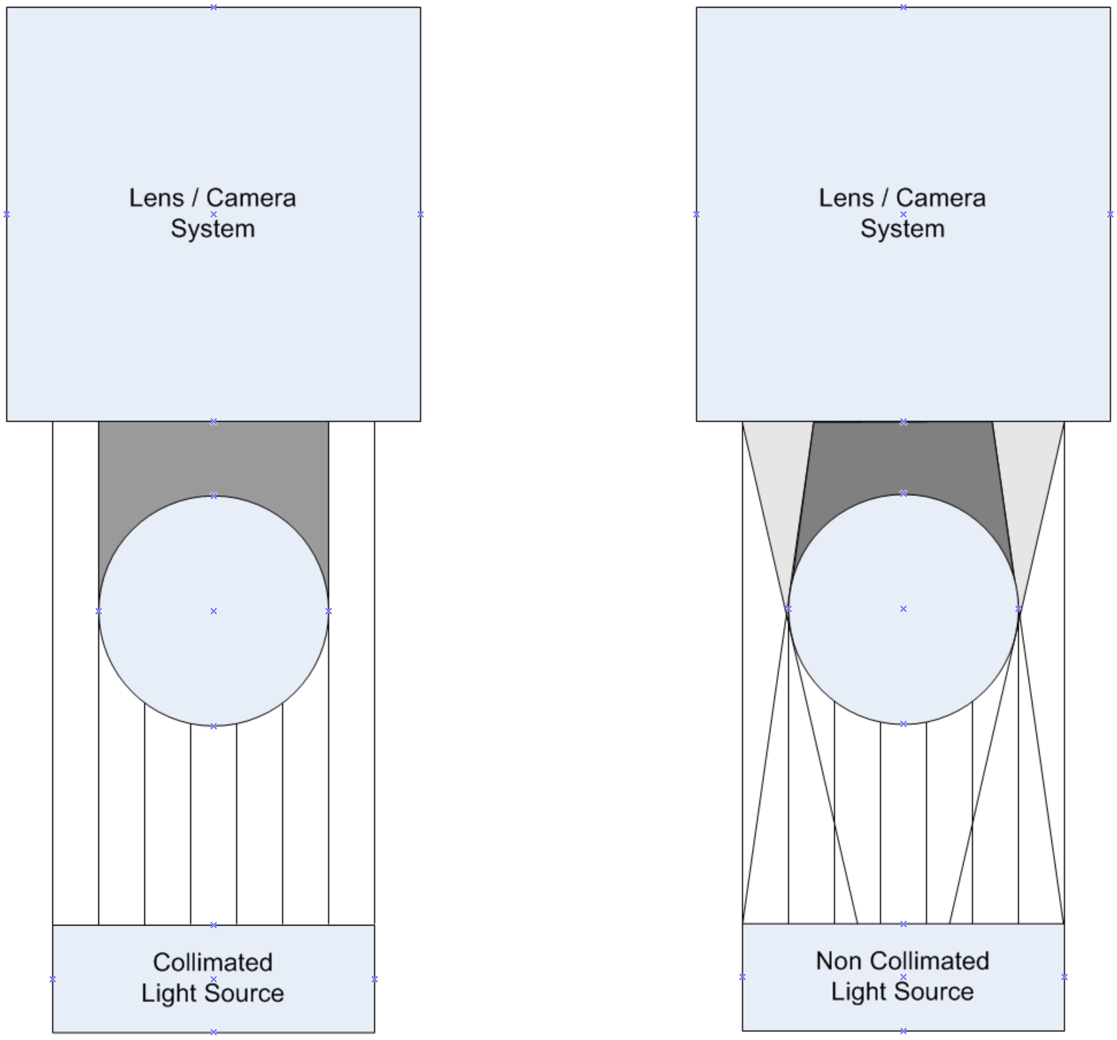


Figure 5: Collimated / Non-collimated light source

The result is a reduction in perspective errors independent of viewing distance, and an improved edge contrast, thus reducing size error. By considering the peak response of the Basler camera [1] (Figure 6), the optics system, the Rayleigh limit was determined to be 2.05 µm (0.00008 inches).

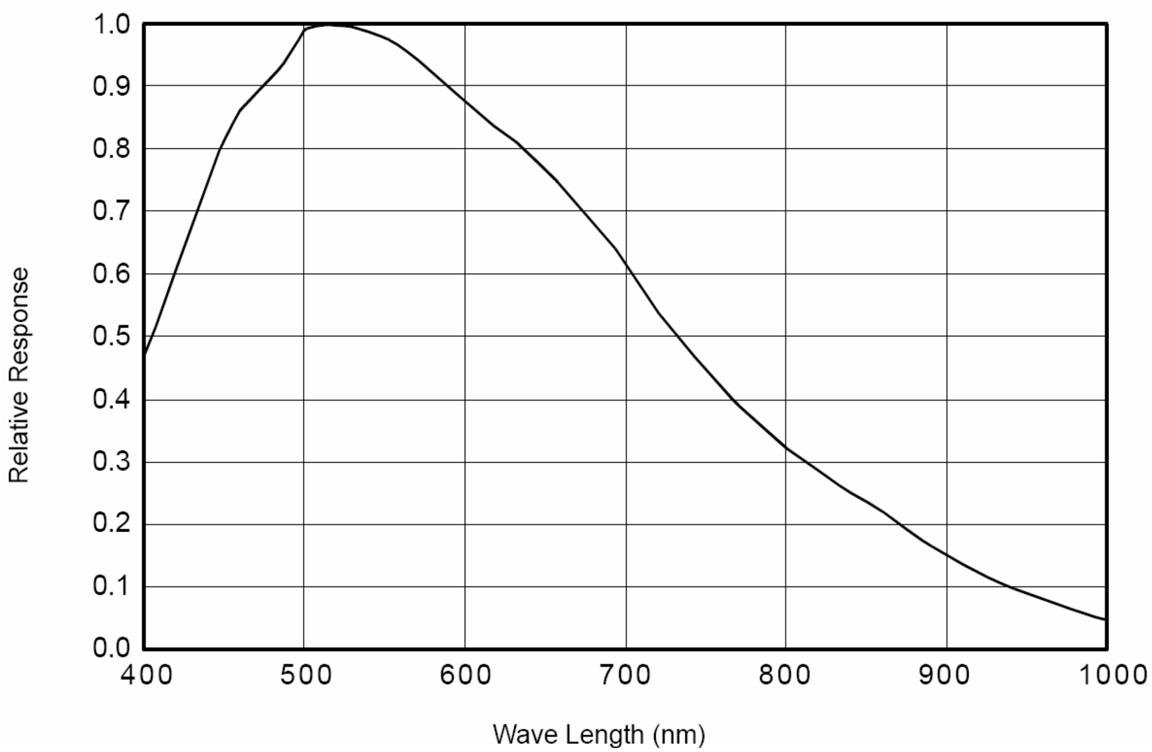


Figure 6: Camera Response [1]

2.5 Vision Software

NI Labview w/ Vision Toolkit

National Instruments Labview 8.5 with the Vision Toolkit and Vision Builder for Automated Inspection (VBAI) was used for its compatibility with a wide variety of machine vision cameras and the flexibility it provided in the design of the vision application. Many tools are available thru the vision toolkit, providing an easy way to create and deploy machine vision gauging applications. VBAI provides many of the tools available with Labview and the vision toolkit in a graphical programming environment by the use of flow charting and sequential functions. Utilizing VBAI reduced the complexity of the programming involved, and presented an improved software solution that involved less custom generated code than creating a similar application using a text based programming language such as Microsoft Visual Basic.

2.6 T-Bed Lathe

General Description

T-bed lathes are a type of 2 axis lathe, where the configuration is such that the X axis and Z axis are not physically attached, and the Z axis carry the spindle system. An example T-bed lathe is illustrated in Figure 7.

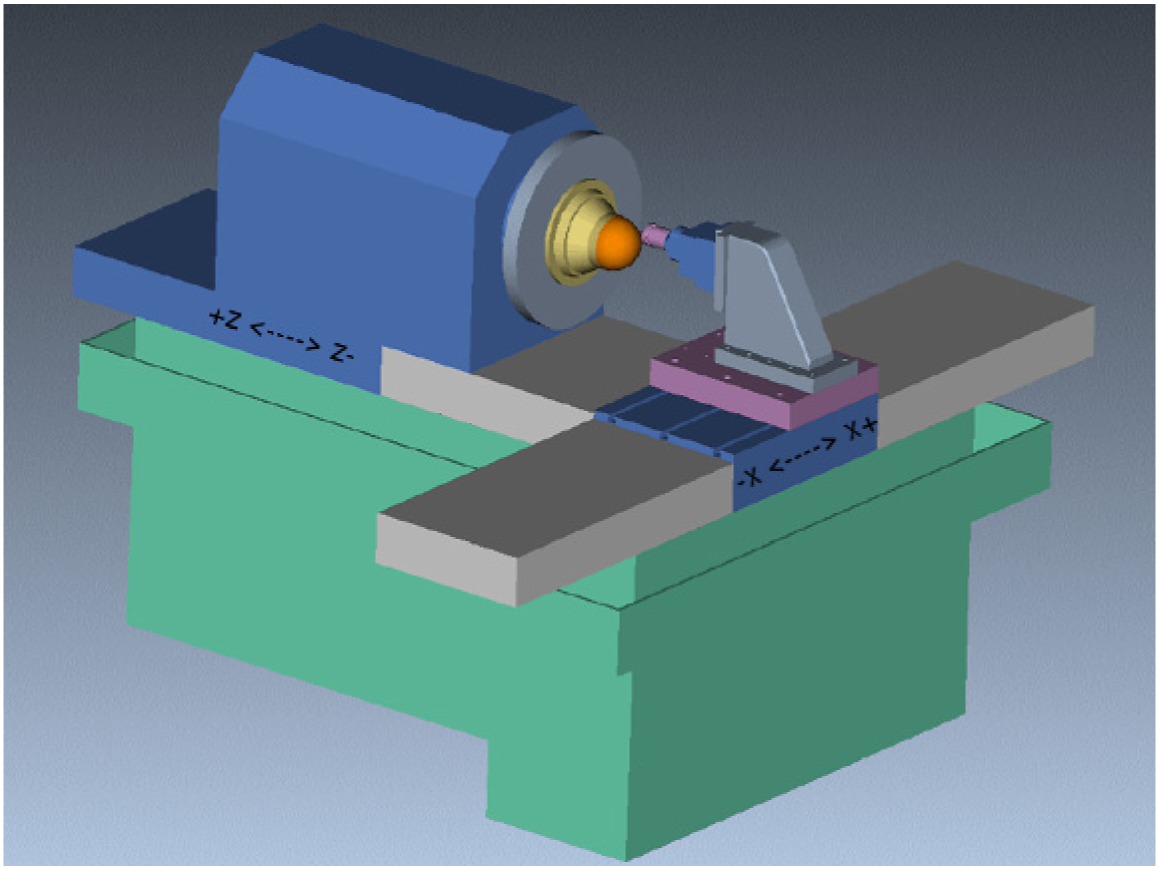


Figure 7: T-Bed Lathe

T-bed lathes are most often used in machining accurate profiles such as optics or spline based profiles. For this reason many contain hydrostatic or air bearing spindles to reduce vibrations and thermal growth concerns. Correct tool and nose radius offsets are necessary to achieve the desired accuracies.

Tool offsets and effect on cutting path

The ability to utilize tool offsets is a tool available on many CNC controllers. Tool offsets are used to correct for differences in individual tools. By utilizing this feature, part programs can be generic to the tool being used. Tool nose radius compensation goes one step further by also compensating the programmed path based on the tool radius value. Appendix D includes figures of offset errors as well as a diagram of the corresponding error transferred to the part.

2.7 Lathe Tool Inserts

Lathe tool inserts are the component performing material removal on a lathe. Because of costs related to custom manufacturing high tolerance tool inserts, a generic set are purchased through a commercial manufacturer. This lowers operational costs at the expense of relatively low tolerances for any given tool. The manufacturer guarantees the tool nose radius to ±0.0014 inches for the radius range of 0.002 in to 0.03125 inches.

2.8 Open Architecture Control

The controller on the T-bed lathe is a software product by MDSI named OpenCNC. OpenCNC is an unbundled, modular all-software based CNC control system.[12] OpenCNC runs on a generic IBM compatible PC running Microsoft Windows 2000 or Windows XP operating system and uses the add-on Real-Time Extensions from Ardence, Inc. to provide the deterministic response necessary for machine tool control. One of the advantages of OpenCNC is a published Application Programming Interface (API), allowing access to internal functions and variables for development of additional functionality such as the vision tool setter. The development tools are available through Microsoft C++ and Microsoft Visual Basic.

3. Literature Review

During the literature review, several works were examined but relatively few dealt with either tool offset or TNR measurement. Many focused on measuring tool wear only. For this reason, the literature review is split into three sections. The first is Tool Measurement, which presents an overview of tool offset and TNR measurement systems. The second section presents an overview of systems which perform tool wear monitoring with machine vision systems. The third section is an overview of commercially available devices and discussion of other concepts that were considered in the place of the currently presented system.

3.1 Tool Measurement

Maali [13] proposed using a 2048 pixel line scan CCD camera and a microcontroller to provide tool set-up capability by moving the tool thru the response line of the camera to build a 2048 x 2048 pixel image. By doing so and then looking for the highest pixel in the array, the Z and X offsets could be calculated from the set of data. Drawbacks were the amount of time required to iterate through the small displacements to generate a good data set and the tight integration between the control and the camera system. The system being presented eliminates these drawbacks by utilizing a full frame CCD camera to capture a full 2-D view of the tool edge.

Nobel [14] interfaced a standalone vision system with an Allen Bradley 8200 control thru the use of a custom interface computer to perform real time compensation of tool nose radius from a true radius. The limitations were the program could only react to current machine position, (it had no access to the path planning portion of the motion control system). This resulted in discontinuity errors on abrupt changes of motion such as corners and axis reversals. Through cutting test parts that avoided abrupt changes of motion, he was able to produce a small reduction in the overall form error of the part of approximately 10 µm.

Doiron [15] conducted his research using a CCD camera connected to a microscope and used the novel approach of including an additional object in the field of view to be used as a calibration gauge (see Figure 8). He also used a histogram to programmatically check if the light source was active. He found the accuracy of the system to be about 2.5 micron for a field of view of 1.4mm. The drawbacks of this system were that the calibration was only good for a small percentage of the viewable area, and thus larger radius tools could present problems. Also, the gauge object would need to be protected from any physical contact which could alter the shape of the gauge. Further, the tool could be damaged if it were to contact the gauge object. The system being presented eliminates these drawbacks. No gauge object is necessary, and a full field calibration is performed to remove any non-linearity from the lens system.

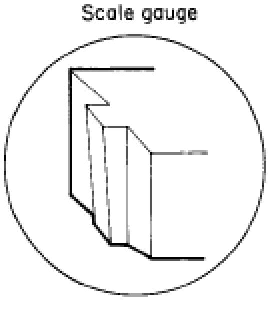


Figure 8: Doiron's Calibration Object

Reddy [16] developed a non-contact tool setting system to accurately position a micro-diamond tool using a CCD camera and a microscope. He tested various disturbances to the system, such as defocus, inclined tool and lighting on the accuracies of the system, and applied Chauvenet's criterion to eliminate spurious results. He found that digitization errors from not using sub-pixel processing, errors in the vision system calibration (from lens distortion), and machine vibration were the cause of most inaccuracies and non-repeatability of the system. The system being presented utilizes subpixel processing, is full field calibrated, and the shutter time of the CCD camera is very low, resulting in vibration being less of a concern.

Varga [17] investigated the use of a CCD camera system to compare the in use grinding wheel profile based on a known good profile. This was implemented on a machine with a separate computer to run the vision application. No geometrical accuracies were presented. He proposed this system could be used for tool wear monitoring.

3.2 Tool wear monitoring

There have been several authors who have worked with machine vision for the purpose of monitoring tool wear. This is not the focus of the presented research, and is provided for further reference to machine vision applications.

Kurada and Bradley [18] used a CCD camera to monitor wear of a lathe tool by monitoring wear pattern reflections and their corresponding grey level differences. Wong [19] used a CCD camera to monitor tool flank wear similar to Kurada and Bradley. Jurkovic [20] used a CCD camera and a scanning laser projected onto the tool to detect wear by analyzing the line pattern generated by the laser – thus enabling height detection of the tool wear. Kerr [21] developed and verified Kurada and Bradley's work.

3.3 Tool setting system

Laser

The principle of operation for commercially available non-contact laser tool setters is simply electronics monitoring for beam interruption. Renishaw NC4 non-contact laser tool setter can have accuracies as good as ±1.0 micron.[22] The benefits of this approach are that it is a commercially available product; it has high accuracy and can perform in a harsh environment. In order to perform tool nose radius measurements, the tool would have to be incrementally moved to trigger the beam to generate a series of points along the radius of the tool as shown in Figure 9.

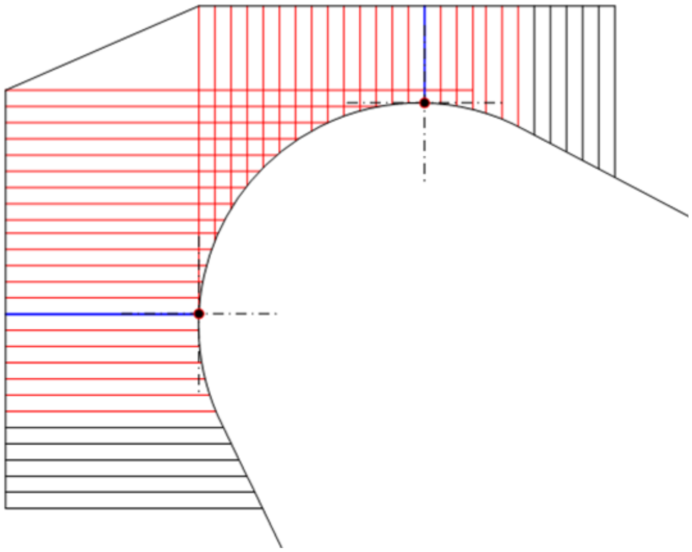


Figure 9: Example Search Pattern

However, laser non-contact sensors are incapable of detecting object below a certain size. The investigated tool setter claimed a minimum radius of 30 micron (.001 inches). For a .002 inch radius lathe tool, this would result in the tool having to be moved in more than .0001 inches past the actual end point of the tool. As this error would vary for different sized tools and different tool geometries, it would complicate any attempt to accurately measure the tool nose radius.

Optical Micrometer

Commercially available, optical micrometers provide high resolution up to ±0.5 micron. The principle of operation is that a led source is used with optical lensing to create a beam of light (think of a 2-D laser). The detector is generally a line scan CMOS or CCD camera.[23] An approach similar to the laser tool setter would be used, with the difference being the iteration would only have to use one axis of motion and then record both the distance to the first point and the thickness recorded by the optical micrometer. Drawbacks to this approach are the same as the laser toolsetter; the minimum detectable object size is greater at .002 inches. The entire tip of smaller tools (those with 0.002 inch radius) would be ignored before the optical micrometer would finally register that the tool had crossed the plane of measurement, and this would be unacceptable.

Contact

Renishaw, Blum, Marposs and Heidenhain all have physical contact based tool setting devices on the commercial market. The principle of operation is sensing a displacement applied to the unit, either thru mechanical, strain gauge or optical sensing methods. [24] [25] [26, 27] The drawbacks include the force required to detect the tool can damage fragile diamond tools. Additionally, only approximations of the tool nose radius can be made unless many touches are made along the edge of the tool using a perfectly round physical detector with a known radius. This would be time consuming, and further exaggerates any error present in the measurement device to the tool nose radius.

Offline

The currently employed method of checking the tool nose radius on the shop floor is to use an optical comparator to measure the radius of their tools offline before use. There are many drawbacks, including:

1. Inaccuracies when installing the tool on the machine
2. Operator is still forced to touch off the part to set X and Z offsets
3. Operator must use judgment on the radius of his tool

4. Experimental Setup

In order to get relevant results from the vision system, the appropriate software had to be created. One application was developed to communicate between VBAI and the MDSI controller. The other application performed all of the vision acquisition, data analysis, and displaying of results. Secondly, a calibration process had to be developed in order to get relevant information from the vision system. Experiments were then performed in order to check the results of the calibration process, and once this was satisfactory, actual tool offset and TNR measurements were made to judge the accuracy of the system. Experiments were performed with different sized tools as well as different tool materials to determine the accuracy and repeatability of the system.

4.1 Software

The largest portion of the research was devoted to programming the vision software. Two applications were developed to support the project goals. The first and largest application was the vision capture and analysis component. This was written using National Instruments Vision Builder for Automated Inspection (VBAI).

NI VBAI Code

The VBAI code development took a considerable amount of time and effort with a lot of trial and error to determine an inspection sequence that would work in almost all circumstances. The process was broken up into small tasks: Setup, Acquisition, Calibration, Inspection, Display, and Results

Setup consisted of starting the TCP Socket Server if it was not already running. Secondly, this step initializes local variables to their appropriate values.

Acquisition requested an image from the camera. A second step was included that evaluates the intensity of the image, and based on this data, could adjust the image until the intensity was within some target value. In this way, the intensity adjustment step would loop back to the Acquire step iteratively. Calibration simply loaded the calibration data from an image and applied it to the most recent image. Inspection is the largest step, it contains all the steps necessary to determine the radius and offset values from the calibrated image. First, it finds 4 points around the perimeter of the image, generates lines, bisectors and the intersecting point for those lines as shown in Step 1 of Figure 10.

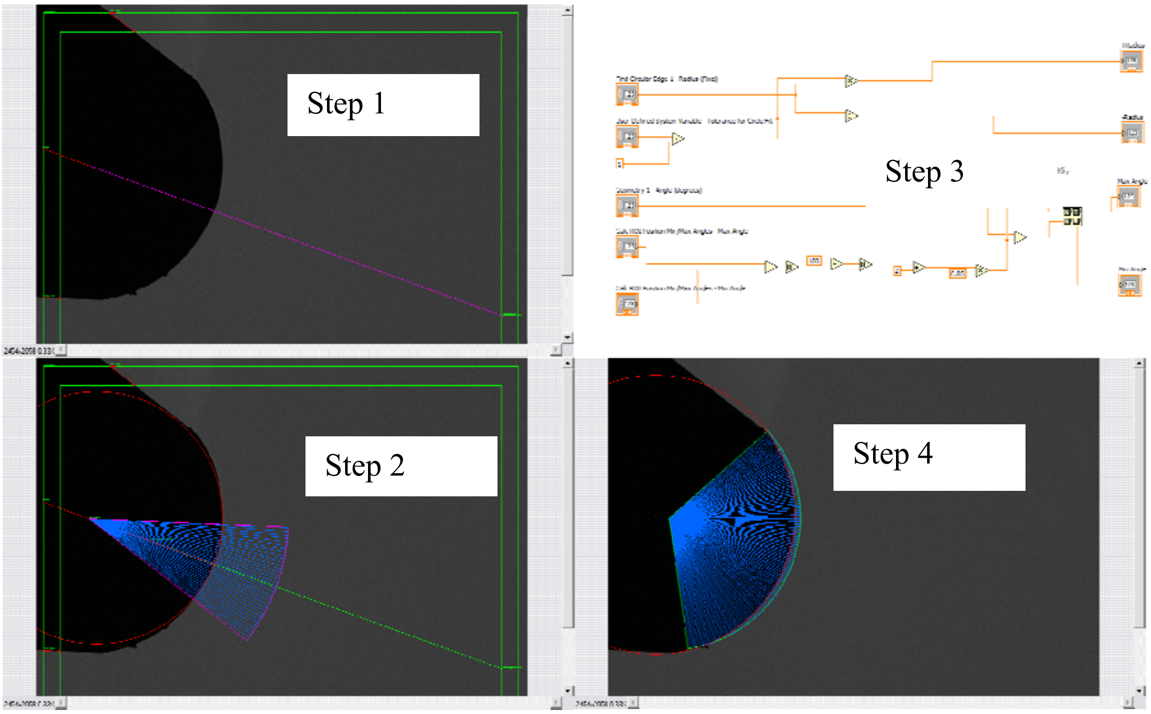


Figure 10: VBAI Steps

In Step 2, it rough calculates the tool nose radius (uses the bisector and the point at the end of the tool to find the edge of the radius), uses this information to create a new region of interest. This consists of setting a tolerance band for the radius measurement, setting angular ranges for the circle fit algorithm (shown in Step 3 of Figure 10). Finally it performs the final tool nose radius fit as shown in Step 4 of Figure 10. Further, the Acquisition, Calibration and Inspection steps are repeated a set number of times (configured by a local variable on startup); this was set to 10 for the purpose of this study. The Display step averages the inspection data and displays it on the screen. The Results step presents the operator a dialog box to accept or decline updating the tool offset table. This dialog also includes a number input box to set which offset register to store the data. If the operator accepts the tool offset values, this data is transferred to the appropriate tool offset register via the TCP Socket Server. See Appendix A for a flow chart of the code execution states, and a diagram of the various steps.

TCP Socket Server

The second application developed was a synchronous socket server written using Microsoft Visual Basic 2005. The purpose of this program was to handle communication between VBAI and OpenCNC using VBAI's internal TCP socket capability to communicate with MDSI OpenCNC. The program allows a connection from a remote client via TCP sockets. It also creates a connection to MDSI OpenCNC thru the use of an MDSI developed ActiveX control. It then facilitates communications between OpenCNC and the client, VBAI. A flow chart and the code itself are presented in Appendix B.

4.2 Calibration

There were three requirements of the calibration in order to make accurate tool offset measurements. These were: Calibration to real world units, the ability to account for angular errors and non-squareness to the lathe's XZ plane, and the physical offset between the machine coordinate system origin and the camera coordinate system origin. Two of these factors were accounted for simultaneously by using the built in calibration tool available in VBAI.

4.2.1 Calibration of vision system

The built in routines from VBAI were used to calibrate the vision system in order to account for angular errors in the alignment of the vision system as well as to provide real world units capability. The built in nonlinear calibration was chosen to provide a high degree of robustness in the calibration. According the National Instruments documentation on nonlinear calibration, *"The nonlinear algorithm computes pixel to real-world mappings in a rectangular region centered around each dot in the calibration grid"* and then "*estimates the mapping information around each dot based on its neighboring dots."*.[28] To most effectively calibrate the camera to the machine tool the optics assembly and the light source assembly were aligned such that the centerlines were parallel to each other to reduce any errors from the light source not being parallel with the optics. Secondly, the optics system was installed on the T-bed lathe and then aligned to the machine in X and Z directions. Finally, the built in calibration routine within VBAI was used to calibrate the system to account for any remaining squareness or angular errors.

4.2.2 Calibration of Camera Offset

By aligning the calibration grid shown in Figure 11 to be square with the machine axes, the calibration algorithm is also able to eliminate error from the camera not being square to the axes. The result of this step was a calibrated vision system capable of outputting real world units with rotation of the camera accounted for by the calibration process. This was a benefit to using the built in calibration as no additional math operations were needed to calculate true position of the tool.

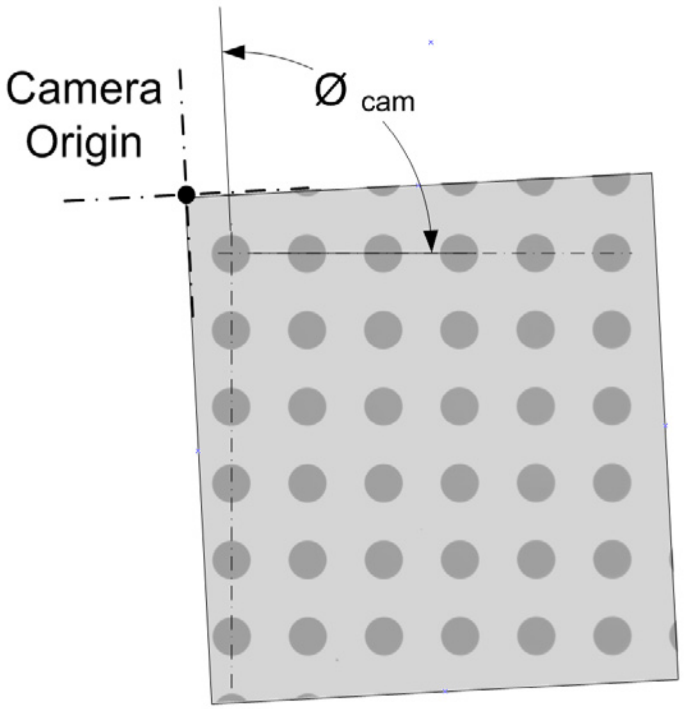


Figure 11: Calibration with Rotation

The second requirement was to calculate and set the offset from the machine tool coordinate system to the machine tool X centerline and Z offset locations. This was accomplished by using a tool insert that had a known tool nose radius, and moving the tool until contact is made on X and Z surfaces of the faceplate of the spindle. This position was set as the new machine origin as shown in Figure 12. The radius and the offset from the camera coordinate system origin were calculated (Xt and Zt in Figure 12). The offset from the set machine origin is Xm and Zm respectively. The calculation of the camera offset is then of the form:

$$X_{cam} = X_m - X_t \qquad Z_{cam} = Z_m - Z_t$$

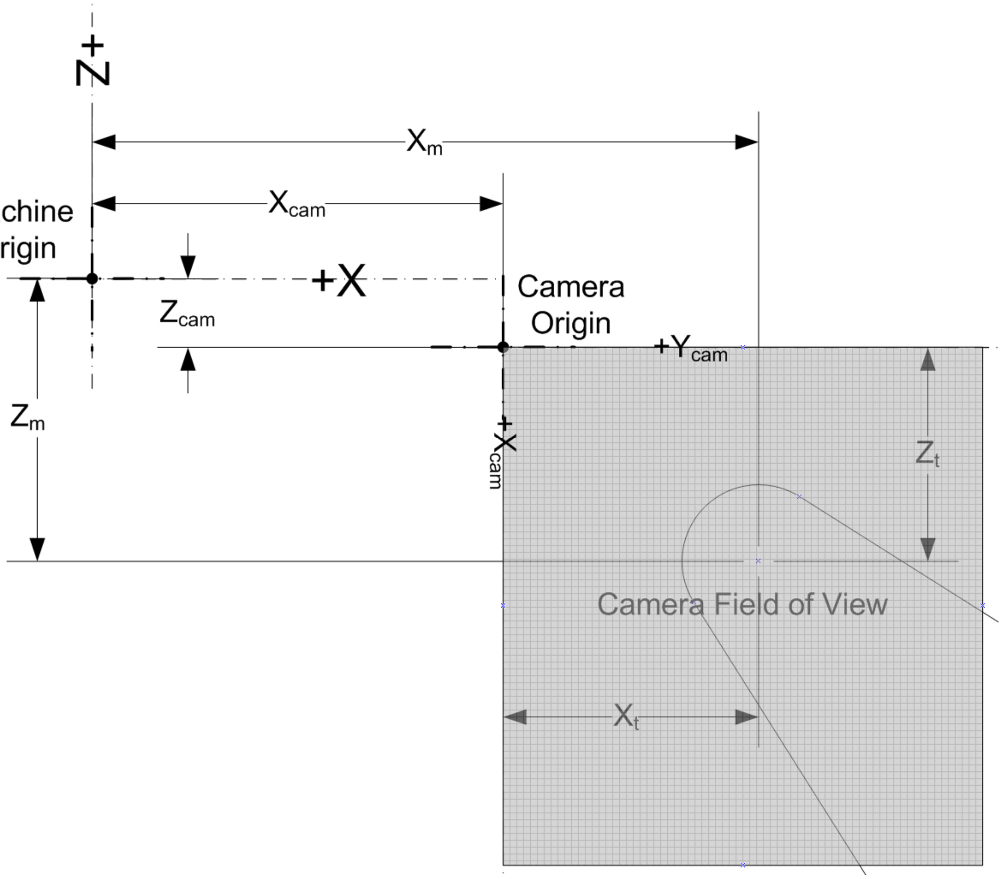


Figure 12: Coordinate Systems

Once this was calculated, it was stored in the vision system software for use when generating tool offsets.

4.3 Data Collection

The data collection from the measurement program was performed by adding a step to the inspection in VBAI which enables measurements to be logged to file. The first experiment was to check the advantage of blue wavelength light over white (multi-wavelength) and to compare the effects of using thresholding techniques to enhance edge contrast of the acquired image. Data sets of 100 individual measurements were used for each test. The second experiment was to evaluate the effect of defocus on the measurement system. Five different positions were evaluated corresponding to 0, +-0.001 and +-0.002 inches, representing the offset from the correct focus distance. A third experiment was performed to check the effect of light intensity on the measurement system. The final experiment was designed to check the accuracy of the calibration performed as well as analyze the performance of the vision system. After the vision system was calibrated, three measurements were taken (each measurement is an average of 10 individual calculations) with the tool at five different locations within the field of view. The calibration was evaluated based on the error of the measurements.

5. Results

White Light and Blue Light Source

To evaluate the effect of light wavelength and the use of thresholding on the results, 100 individual measurements were conducted for four cases:

- White light source without thresholding
- Blue light source without thresholding
- White light with thresholding
- Blue light with thresholding

The data shown in Table 2 is separated into three sections; radius, X and Z. Within each table, the average, max, min, maximum deviation (max-min) and standard deviation (sigma) was calculated for each of the four cases.

The maximum deviation was almost double for the experiments using thresholding. In the same way, the standard deviation was at least double for the thresholding cases.

Table 2: Light Source Data Results

	Blue (470nm)		White(~550nm)	
	No Threshold	Threshold	No Threshold	Threshold
Radius				
Average	802.4211	802.2337	802.6542	802.6728
Max	804.1668	804.9041	804.4132	805.2438
Min	801.8160	799.8366	801.6912	800.2114
Max-Min	2.3508	5.0674	2.7220	5.0324
Sigma	0.388737125	1.510792212	0.75888073	1.508002603
X				
Average	-172.1282	-176.6138	-172.1822	-175.0963
Max	-171.4014	-174.3567	-171.0979	-172.5606
Min	-173.8630	-179.0531	-173.9386	-177.5061
Max-Min	2.4616	4.6964	2.8407	4.9455
Sigma	0.421452865	1.402136342	0.707617003	1.40205409
Z				
Average	366.2560	354.9534	362.4653	352.5230
Max	366.5716	355.9623	362.9361	353.6189
Min	365.3821	353.7469	361.5213	351.2571
Max-Min	1.1895	2.2154	1.4147	2.3619
Sigma	0.189102627	0.715850167	0.379299618	0.772325092

The data was analyzed and a normal distribution was fit to the data. Results can be seen as a plot versus a Normal distribution in Figure 13. Blue light and no thresholding had considerably less variation than any other experiment data. Using the threshold function more than doubled the standard deviation of the measurements in every case, so thresholding was abandoned as a technique to be used. Further, the blue light source was used exclusively as a way to further improve measurement data.

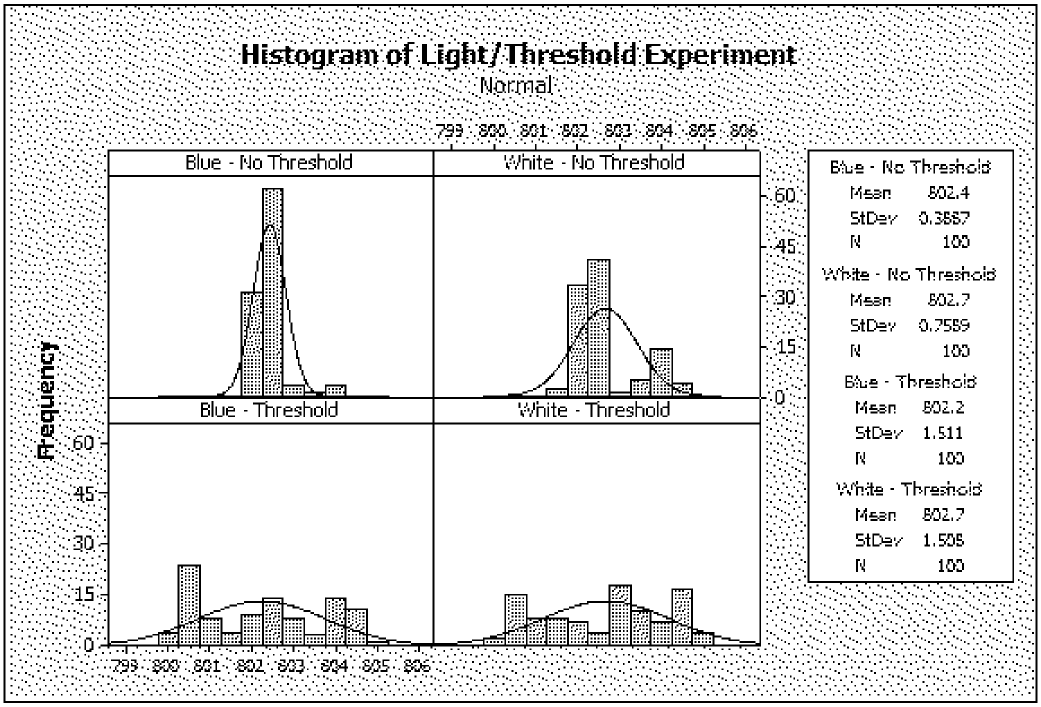


Figure 13: Results from Light/Threshold Experiment

A visual comparison of the blue light and white light images (Figure 14) clearly shows the Airy disk in the white light image as a series of concentric radiuses going toward the right from the edge of the tool. This was further assurance that the blue light would provide better measurement results.

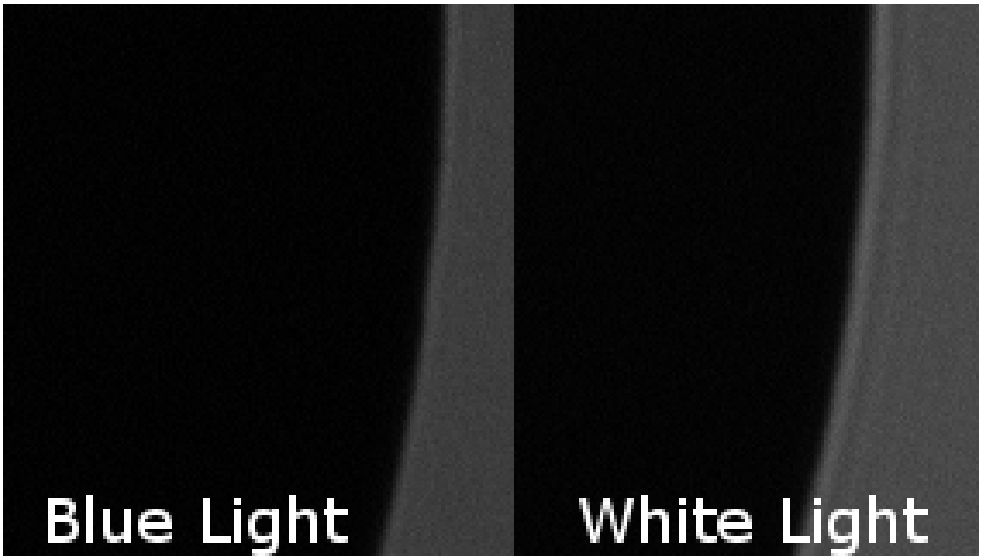


Figure 14: Blue Light vs. White Light Comparison

Depth of Focus

To check the vision system for error induced from the tool out of focus (truly an indication that the tool height was set improperly), a set of data was recorded and deviations and standard deviation calculated with the tool in 5 different positions [-0.0508 mm, -0.0254 mm, 0.000 mm, +0.0254 mm, +0.0508 mm]. These positions were chosen after determining the system had trouble detecting the radius at +-0.0508 mm. The results are presented in Table 3.

Table 3: Depth of Focus Data

	Deviation (µm)			Std Dev (µm)		
Focus Pt	Radius	X	Z	Radius	X	Z
-0.0508	0.5524	0.0668	-2.4067	0.2595	0.0657	0.4798
-0.0254	0.2370	-0.0962	-1.6153	0.1630	0.0499	0.3168
0.0000	0.0000	0.0000	0.0000	0.1457	0.0572	0.4130
0.0254	-0.0706	-0.0056	0.4645	0.2755	0.0534	0.4074
0.0508	1.4625	0.5115	-1.4689	0.6809	0.2097	0.9245

The maximum deviation of the calculated radius was 1.5 µm at a focus point of 0.0508 mm. X maximum deviation was 0.5 µm at 0.0508 mm, and Z maximum deviation was over 2.4 µm at -0.0508 mm. At +- 0.0508 mm out of focus, the inspection process of the vision software had trouble identifying a suitable edge, so more than ten measurements were attempted to get ten good measurements. When this occurred, the circle fit algorithm used a reduced set of data to perform its circle fit. This reduction in points is illustrated in Figure 15, where the captured data points appear as the bright dots at the edge of the dark radius representing the curve fit. The circle fit algorithm uses a minimum change in pixel value as a cutoff for determining what is an edge, and as a result of the out of focus condition, this cutoff wasn't met.

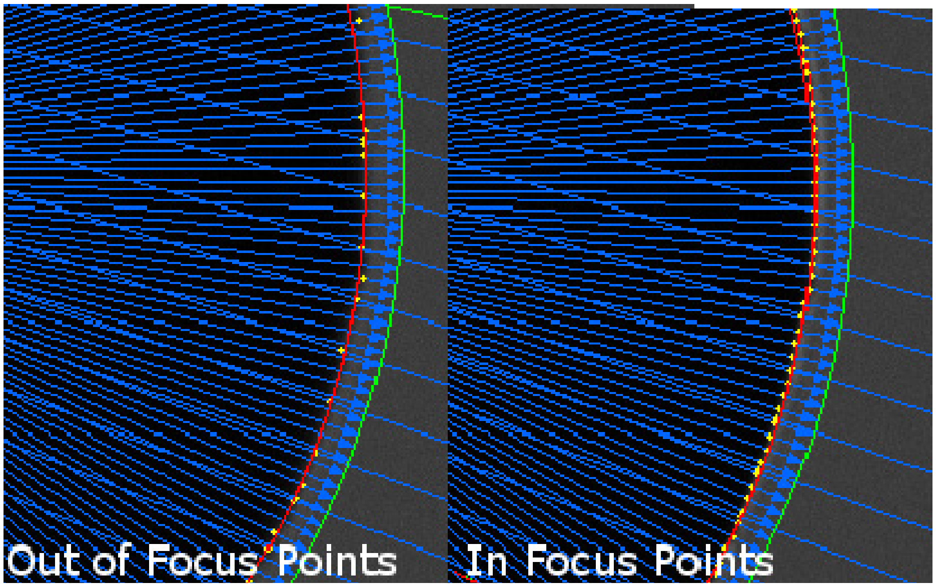


Figure 15: Depth of Focus Data Points

Calibration Checks

To check the calibration of the system, tools whose radius had been measured to five decimal places on a separate calibrated optical vision system were inspected in five different regions of the field of view as shown in Figure 16. Thirty measurements were taken at each of the five positions, and the data was analyzed for variation between the measurement positions. The error map produced by the calibration routine was evaluated to determine if there was any correlation between the data. This is shown as a graphic in Figure 17. The dark outer region represents the portion of the image that was not calibrated due to the spacing of the grid dots. From Figure 17, the calibration error map predicted at most .4 µm of error centered at Position 2. For a comparison to a distorted lens and corresponding error map from National Instruments, see Appendix C.

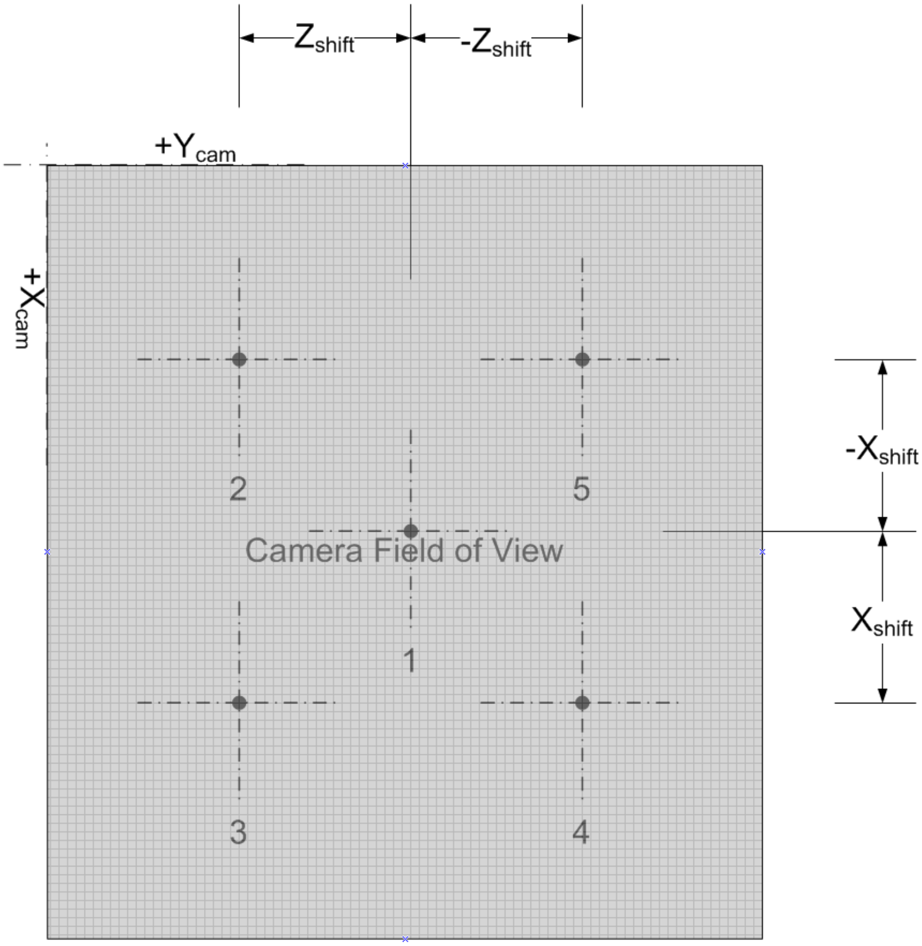


Figure 16: Vision System test locations

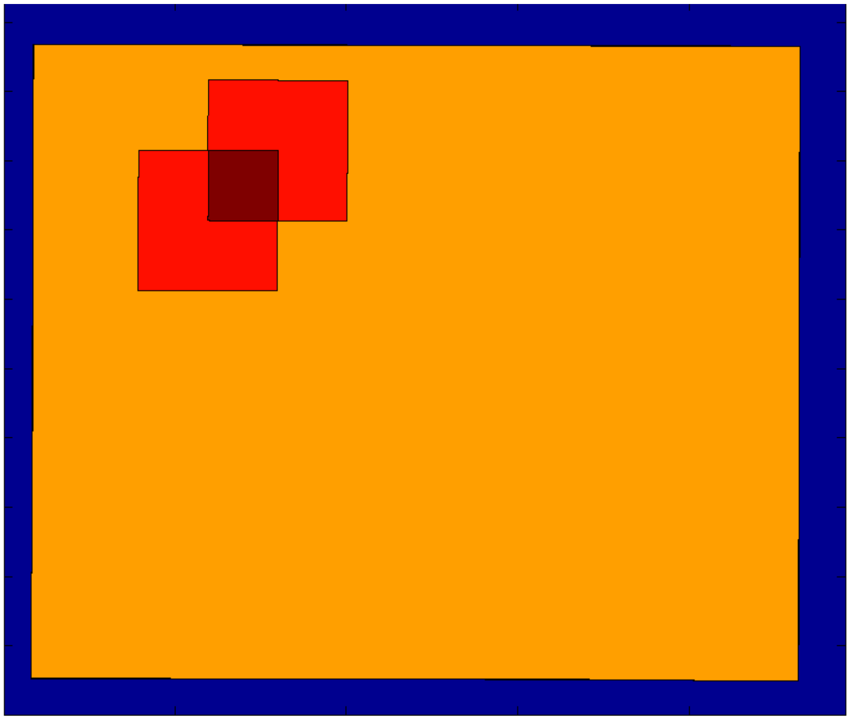


Figure 17: Calibration Error Map

Tool Radius and Offset Performance

The largest portion of testing was devoted to performing actual radius and offset measurements in order to characterize the entire system, including errors from the lathe, in order to get an idea of the attainable accuracy and repeatability. For this reason, no attempt was made to separate the machine errors from the vision system errors. In the first series of tests, tool inserts with radius values of 203.2 μm, 428.75 μm, and 792.48 μm were measured 30 times at the 5 positions used in the calibration tests. Deviations were calculated for each test, and are presented in Table 4. The data showed systematic errors for all measurements performed (see Mean Deviations listed in Table 4). When the data was analyzed at the individual positions, a trend did emerge in repeatability, with a mean repeatability of the radius value of 6.3 μm and a standard deviation of .33 μm.

Table 4: Tool deviations

	Measured Tool Deviations from actual (μm)								
	203.2			428.75			792.48		
Average of 10 runs	Radius (μm)	X (μm)	Z (μm)	Radius (μm)	X (μm)	Z (μm)	Radius (μm)	X (μm)	Z (μm)
Position 1	1.1288			-2.6197			4.2518		
Position 2	1.5488	-1.3980	-3.6998	-3.4106	-2.5357	-2.3369	2.8801	1.1119	-4.3641
Position 3	1.4350	-1.3980	1.2059	-1.4313	-0.4721	-4.9024	4.8366	0.5198	-1.3599
Position 4	1.2996	-2.6818	0.7742	-1.9506	0.0526	-4.9462	4.3667	0.6991	-0.4424
Position 5	0.9562	-3.8430	-0.7341	-3.1142	-0.5282	-1.5771	2.8443	0.6694	-5.2943
Mean Deviation	1.2737	-2.3302	-0.6134	-2.5053	-0.8708	-3.4406	3.8359	0.7501	-2.8652
σ Deviation	0.2367	1.1762	2.2193	0.8162	1.1403	1.7411	0.9156	0.2537	2.3298

The data was further analyzed over each position to get a representation of repeatability (max error – min error) at any one location. This is shown in Table 5.

Table 5: Position Repeatability

Repeatability at each Position	Radius (μm)	X (μm)	Z (μm)
Position 1	6.8715		
Position 2	6.2907	3.6476	2.0272
Position 3	6.2679	1.9178	6.1083
Position 4	6.3173	3.3809	5.7204
Position 5	5.9585	4.5124	4.5602
Mean Repeatability	6.3412	3.3647	4.6040
σ	0.3302	1.0788	1.8395

Finally, the mean was found to get an overall view of the accuracy and repeatability of the system, shown in Table 6.

Table 6: Overall Accuracy & Repeatability

All Positions	Radius (μm)	X (μm)	Z (μm)
Repeatability	8.2472	4.9550	6.5002
Mean Deviation	0.8681	-0.8170	-2.3064
σ Deviation	2.7775	1.5736	2.2965

Time Study

The previous method of determining tool insert radius was a manual process that involved walking to the nearest optical comparator, setting the comparator up, estimating the tool nose radius, and returning to the machine. The tool insert had to be installed into the tool holder, the tool height had to be set, and then the insert had to be touched off of the part to set the offset values. The total time averaged 11.5 minutes. The new process eliminates all but the manual insertion of the tool to the tool holder and setting of the tool height. Mean time during testing for swapping a tool, setting height, and obtaining tool offset values was 3.2 minutes.

6. Conclusion

A machine vision based optical tool setter was developed for checking the radius of lathe tools. The system had an average repeatability of 8.2 µm, with an accuracy average of 2.5 µm. The standard deviation of the measurements was improved by not applying thresholding to the image data and using a blue light source. A software based calibration routine was used to calibrate the camera to the full field of view. While the overall system did not meet the goal of 5 micron repeatability, there was still potential for improved part tolerance control. The system was also capable of automatically updating the CNC control tool offset register to reflect the measurements. Additionally, the system could eliminate 8.3 minutes from the manual tool process. Finally, a source of human error was eliminated as it was no longer necessary to use judgment in determining the tool nose radius value or while setting the touch off points on the part.

6.1 Recommendations

In order to obtain better calibration results, a calibration grid with a much finer grid spacing and dot size would need to be utilized. The ideal grid for the current camera and field of view according to the National Instruments documentation would be 15 µm dots spaced every 30 µm.

Further development of the software (also an increase in complexity) would allow current tool shapes to be stored in memory so it can later be recalled to check for tool wear and automatic shape recognition.

A zoom lens with powered zoom could be used to make more accurate measurements for smaller radiused tools. Alternatively, custom lens development services could be contracted to develop a lens solution to address resolution and telecentricity concerns to provide a very robust lens design – cost control would be a major concern.

Integration of techniques such as Pierce’s criterion to identify spurious data, or Hough transforms to identify curve data from line data.

Investigate White Light Interferometery for inclusion of tool height measurement concurrent with radius and offset measurement capabilities.

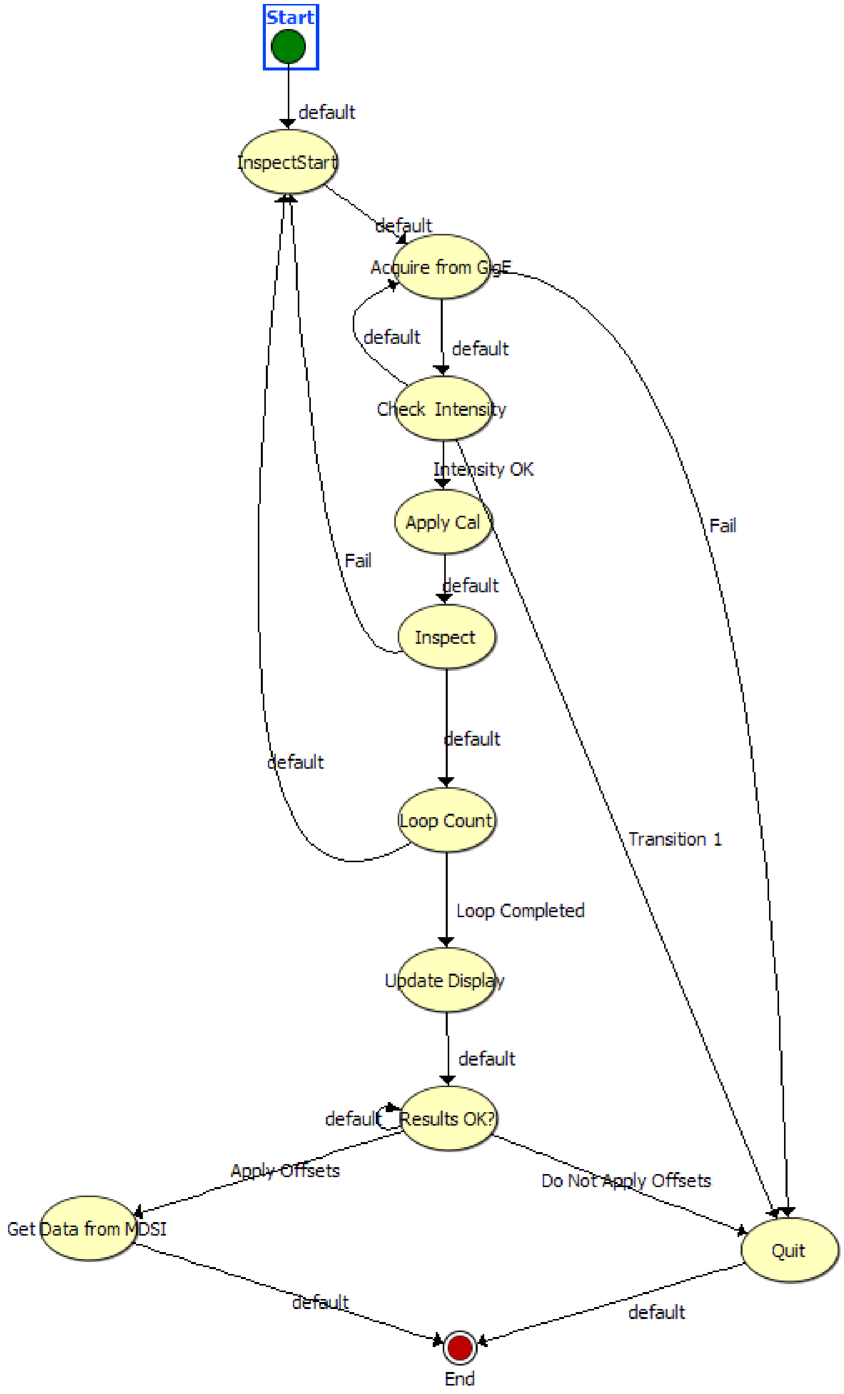
Bibliography

1. Basler, I., *Basler Pilot User Manual*. 2008.
2. Jahne, B., H. Becker, and P. Geibler, *Hand Book of Computer Vision & Application*. Vol. 1. 1999, San Diego, CA: Academic Press. 93-97 188-195.
3. Basler, I. *Image Sensor Technology*. 2007 [cited; Available from: http://www.baslerweb.com/beitraege/beitrag_en_23777.html.
4. Basler Inc., *Basler Pilot User Manual*.
5. Zuech, N., *Understanding & Applying Machine Vision, Second Edition*. 2000, New York, NY: Marcel Dekker, Inc.
6. National Instruments Inc., *National Instruments GigE Information*
7. Navitar Inc., *Lens System Performance Data*. 2007.
8. Menn, N., *Practical Optics*. 2004, Elsevier Academic PRess: Burlington, MA. p. 61-73.
9. Edmund Optics Inc. *Infinity corrected and Plan objectives*. [cited; Available from: http://www.edmundoptics.com/techSupport/DisplayArticle.cfm?articleid=277 .
10. Laikin, M., *Lens Design Third Edition, Revised and Expanded*. 2001, Marcel Dekker, Inc. p. 36-39, 133-134, 265.
11. MellesGriot. *Optics Info*. 2007 [cited.
12. MDSI Inc., *OpenCNC DataSheet*. 2008.
13. Maali, F., et al. *VISION BASED TECHNIQUE FOR AUTOMATIC TOOL SET-UP, IDENTIFICATION, AND WEAR MEASUREMENT IN CNC LATHE*. 1984. Las Vegas, NV, USA: ASME, New York, NY, USA.
14. Nobel, G.M., *Tool nose radius deviation error compensation simulation and implementation*. 1988.
15. Doiron, T.D., *Computer vision based station for tool setting and tool form measurement.* Precision Engineering, 1989. **11**(4): p. 231-238.
16. Reddy, G.G., *Accuracy of a vision based tool setting system for micro diamond tools*. 1994, Louisiana Tech University.
17. Varga, G., *Advantages of the CCD camera measurements for profile and wear of cutting tools.* Journal of Physics. Conference Series, 2005. **13**: p. 159-162.
18. Kurada, S. and C. Bradley, *Machine vision system for tool wear assessment.* Tribology International, 1997. **30**(4): p. 295-304.
19. Wong, Y.S., et al. *Machine vision monitoring of tool wear*. 1998. Boston, MA, USA: SPIE, Bellingham, WA, USA.
20. Jurkovic, J., M. Korosec, and J. Kopac, *New approach in tool wear measuring technique using CCD vision system.* International Journal of Machine Tools and Manufacture, 2005. **45**(9): p. 1023-1030.
21. Kerr, D., J. Pengilley, and R. Garwood, *Assessment and visualisation of machine tool wear using computer vision.* The International Journal of Advanced Manufacturing Technology, 2006. **28**(7): p. 781-791.
22. Renishaw Inc. *Renishaw NC4 Data Sheet*. [cited; Available from: http://www.renishaw.com/en/6099.aspx.

23. Keyence, *Optical Micrometer Data Sheet.* 2007.
24. Blum, *TC 76 Contact Tool Setter Info*. 2007.
25. Heidenhain, *TT-140 Contact Tool Setter Info*. 2007.
26. Marposs, *T18 Contact Tool Setter Info*. 2007.
27. Renishaw, *RP3 Contact Tool Setter Info*. 2007.
28. National Instruments Inc., *National Instruments Vision Concepts Manual*.

APPENDIX A: NI VBAI Code

Flow Chart

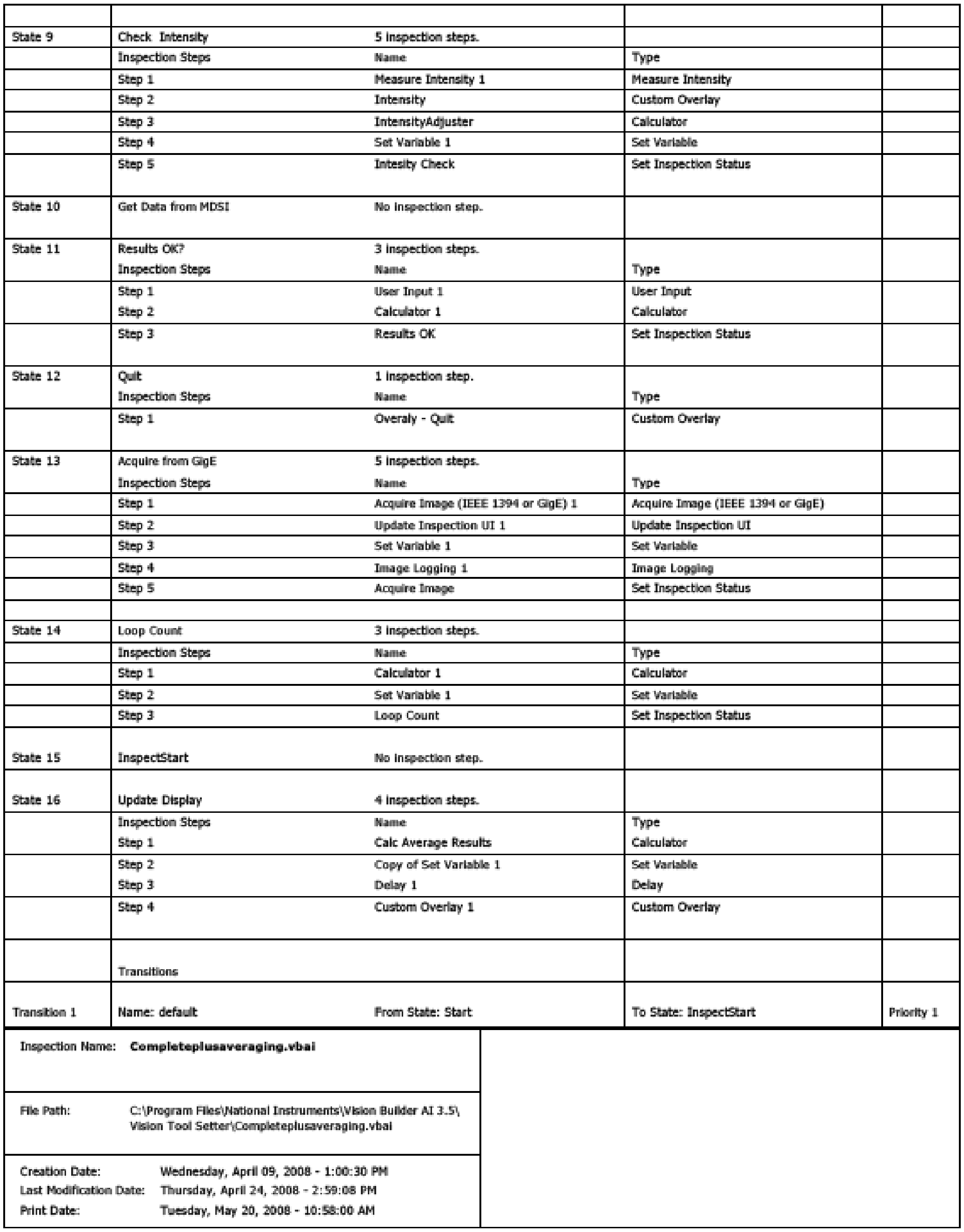


	States			
State 1	Select Inspection	No inspection step.		
State 2	Inspection Setup	2 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Run LabVIEW VI 1	Run LabVIEW VI	
	Step 2	Set Variable 1	Set Variable	
State 3	Inspection Cleanup	No inspection step.		
State 4	Start			
State 5	Inspect	21 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Find Edges 1	Find Edges	
	Step 2	Bisector	Geometry	
	Step 3	Intersecting Pt	Geometry	
	Step 4	Line1 Angle	Geometry	
	Step 5	Line2 Angle	Geometry	
	Step 6	Calc ROI Position Min/Max Angles	Calculator	
	Step 7	ROI Bisect to Intersect	Create Region of Interest	
	Step 8	Set Variable 2	Set Variable	
	Step 9	Find Edges 2	Find Edges	
	Step 10	Dist from Bisect to Edge	Geometry	
	Step 11	Geometry 1	Geometry	
	Step 12	RadiusFit1	Calculator	
	Step 13	ROI InitCircle	Create Region of Interest	
	Step 14	Find Circular Edge 1	Find Circular Edge	
	Step 15	Radius Tolerance Bands	Calculator	
	Step 16	ROI Final Circle Fit	Create Region of Interest	
	Step 17	Find Circular Edge 2	Find Circular Edge	
	Step 18	Set Variable 1	Set Variable	
	Step 19	OVerlay Results	Custom Overlay	
	Step 20	Data Logging 1	Data Logging	
	Step 21	Inspect	Set Inspection Status	
State 6	End			
State 7	Simulate Acquire	1 inspection step.		
	Inspection Steps	Name	Type	
	Step 1	Simulate Acquisition 1	Simulate Acquisition	
State 8	Apply Cal	2 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Calibrate Image 1	Calibrate Image	
	Step 2	Set Inspection Status 1	Set Inspection Status	

Inspection Name: **Completeplusaveraging.vbai**

File Path: C:\Program Files\National Instruments\Vision Builder AI 3.5\Vision Tool Setter\Completeplusaveraging.vbai

Creation Date: Wednesday, April 09, 2008 - 1:00:30 PM
Last Modification Date: Thursday, April 24, 2008 - 2:59:08 PM
Print Date: Tuesday, May 20, 2008 - 10:58:00 AM



State 9	Check Intensity	5 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Measure Intensity 1	Measure Intensity	
	Step 2	Intensity	Custom Overlay	
	Step 3	IntensityAdjuster	Calculator	
	Step 4	Set Variable 1	Set Variable	
	Step 5	Intesity Check	Set Inspection Status	
State 10	Get Data from MDSI	No inspection step.		
State 11	Results OK?	3 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	User Input 1	User Input	
	Step 2	Calculator 1	Calculator	
	Step 3	Results OK	Set Inspection Status	
State 12	Quit	1 inspection step.		
	Inspection Steps	Name	Type	
	Step 1	Overaly - Quit	Custom Overlay	
State 13	Acquire from GigE	5 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Acquire Image (IEEE 1394 or GigE) 1	Acquire Image (IEEE 1394 or GigE)	
	Step 2	Update Inspection UI 1	Update Inspection UI	
	Step 3	Set Variable 1	Set Variable	
	Step 4	Image Logging 1	Image Logging	
	Step 5	Acquire Image	Set Inspection Status	
State 14	Loop Count	3 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Calculator 1	Calculator	
	Step 2	Set Variable 1	Set Variable	
	Step 3	Loop Count	Set Inspection Status	
State 15	InspectStart	No inspection step.		
State 16	Update Display	4 inspection steps.		
	Inspection Steps	Name	Type	
	Step 1	Calc Average Results	Calculator	
	Step 2	Copy of Set Variable 1	Set Variable	
	Step 3	Delay 1	Delay	
	Step 4	Custom Overlay 1	Custom Overlay	
	Transitions			
Transition 1	Name: default	From State: Start	To State: InspectStart	Priority 1

Inspection Name: **Completeplusaveraging.vbai**

File Path: C:\Program Files\National Instruments\Vision Builder AI 3.5\
Vision Tool Setter\Completeplusaveraging.vbai

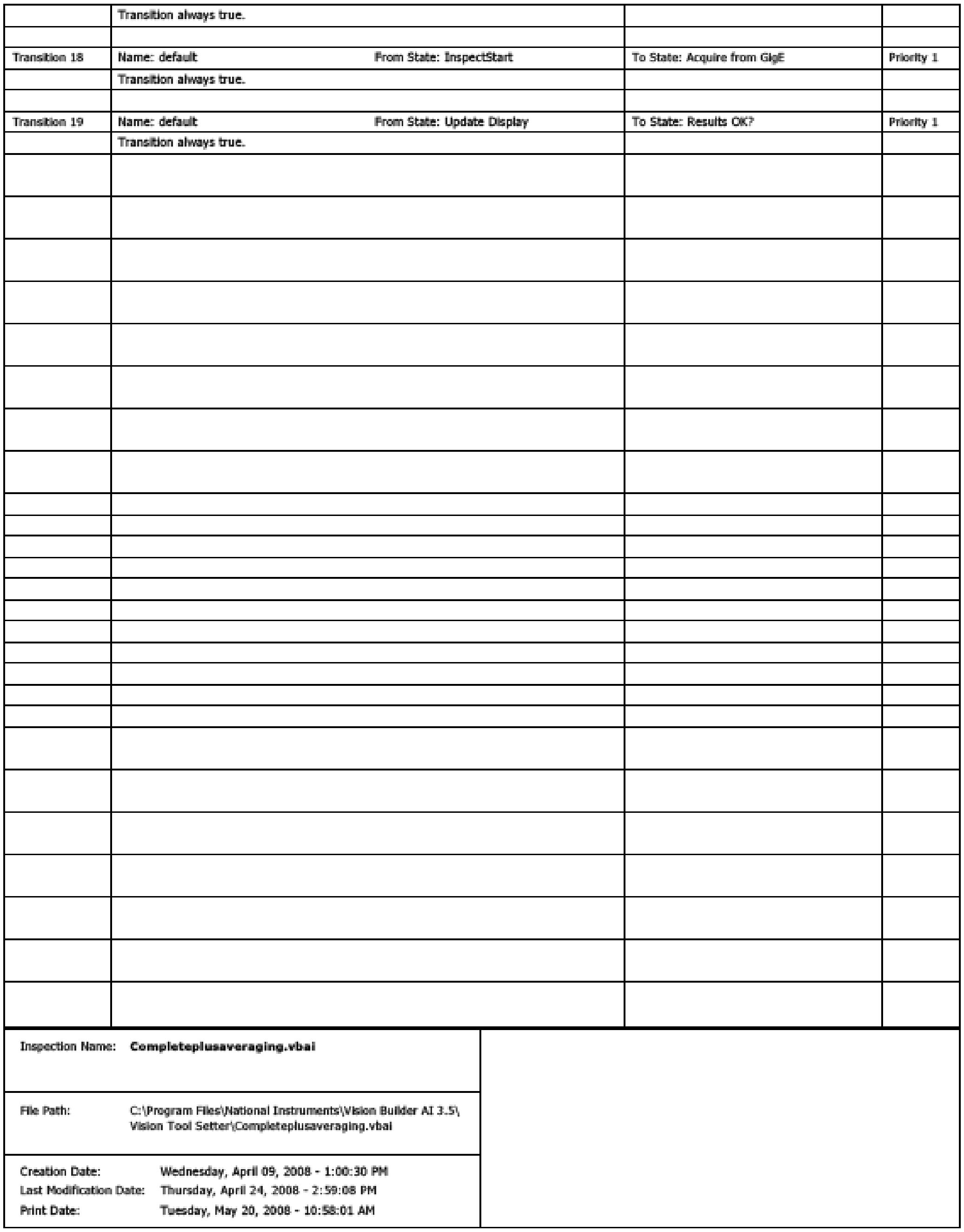
Creation Date: Wednesday, April 09, 2008 - 1:00:30 PM
Last Modification Date: Thursday, April 24, 2008 - 2:59:08 PM
Print Date: Tuesday, May 20, 2008 - 10:58:00 AM

	Transition always true.			
Transition 2	Name: Fail	From State: Inspect	To State: InspectStart	Priority 1
	Transition active if:	System Variable.Inspection Status	is FALSE	
Transition 3	Name: default	From State: Inspect	To State: Loop Count	Priority 2
	Transition always true.			
Transition 4	Name: default	From State: Simulate Acquire	To State: Simulate Acquire	Priority 1
	Transition always true.			
Transition 5	Name: default	From State: Apply Cal	To State: Inspect	Priority 1
	Transition always true.			
Transition 6	Name: Intensity OK	From State: Check Intensity	To State: Apply Cal	Priority 1
	Transition active if:	Intesity Check.Step Status	is TRUE	
Transition 7	Name: Transition 1	From State: Check Intensity	To State: Quit	Priority 2
	Transition always true.			
Transition 8	Name: default	From State: Check Intensity	To State: Acquire from GigE	Priority 3
	Transition always true.			
Transition 9	Name: default	From State: Get Data from MDSI	To State: End	Priority 1
	Transition always true.			
Transition 10	Name: Apply Offsets	From State: Results OK?	To State: Get Data from MDSI	Priority 1
	Transition active if:	Results OK.Step Status	is TRUE	
Transition 11	Name: Do Not Apply Offsets	From State: Results OK?	To State: Quit	Priority 2
	Transition active if:	Results OK.Step Status	is TRUE	
Transition 12	Name: default	From State: Results OK?	To State: Results OK?	Priority 3
	Transition always true.			
Transition 13	Name: default	From State: Quit	To State: End	Priority 1
	Transition always true.			
Transition 14	Name: Fail	From State: Acquire from GigE	To State: Quit	Priority 1
	Transition active if:	Acquire Image.Step Status	is FALSE	
Transition 15	Name: default	From State: Acquire from GigE	To State: Check Intensity	Priority 2
	Transition always true.			
Transition 16	Name: Loop Completed	From State: Loop Count	To State: Update Display	Priority 1
	Transition active if:	Calculator 1.	is TRUE	
Transition 17	Name: default	From State: Loop Count	To State: InspectStart	Priority 2

Inspection Name: **Completeplusaveraging.vbai**

File Path: C:\Program Files\National Instruments\Vision Builder AI 3.5\
Vision Tool Setter\Completeplusaveraging.vbai

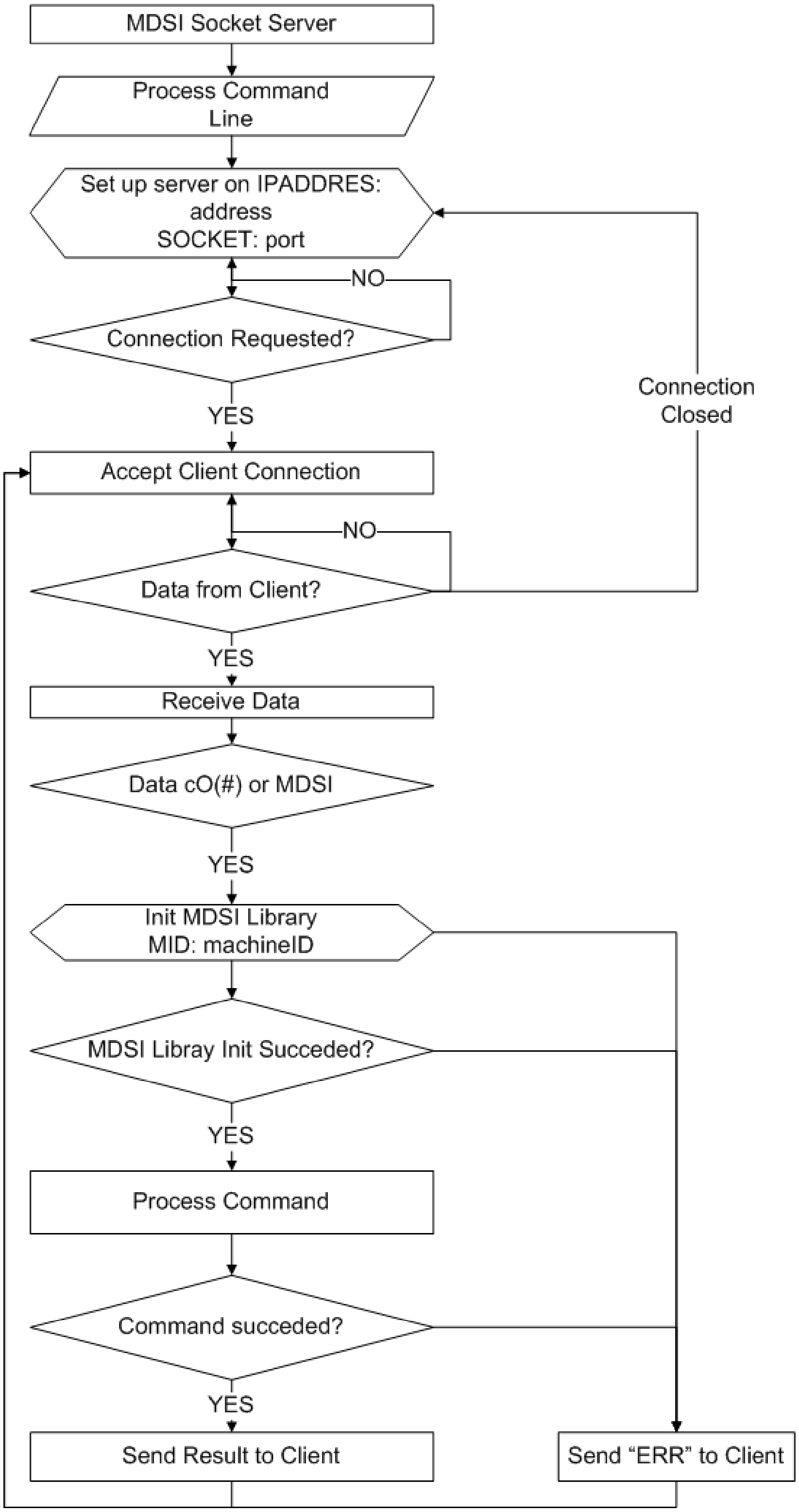
Creation Date: Wednesday, April 09, 2008 - 1:00:30 PM
Last Modification Date: Thursday, April 24, 2008 - 2:59:08 PM
Print Date: Tuesday, May 20, 2008 - 10:58:01 AM



	Transition always true.		
Transition 18	Name: default From State: InspectStart	To State: Acquire from GigE	Priority 1
	Transition always true.		
Transition 19	Name: default From State: Update Display	To State: Results OK?	Priority 1
	Transition always true.		

Inspection Name:	**Completeplusaveraging.vbai**
File Path:	C:\Program Files\National Instruments\Vision Builder AI 3.5\ Vision Tool Setter\Completeplusaveraging.vbai
Creation Date:	Wednesday, April 09, 2008 - 1:00:30 PM
Last Modification Date:	Thursday, April 24, 2008 - 2:59:08 PM
Print Date:	Tuesday, May 20, 2008 - 10:58:01 AM

APPENDIX B: TCP Socket Server Code



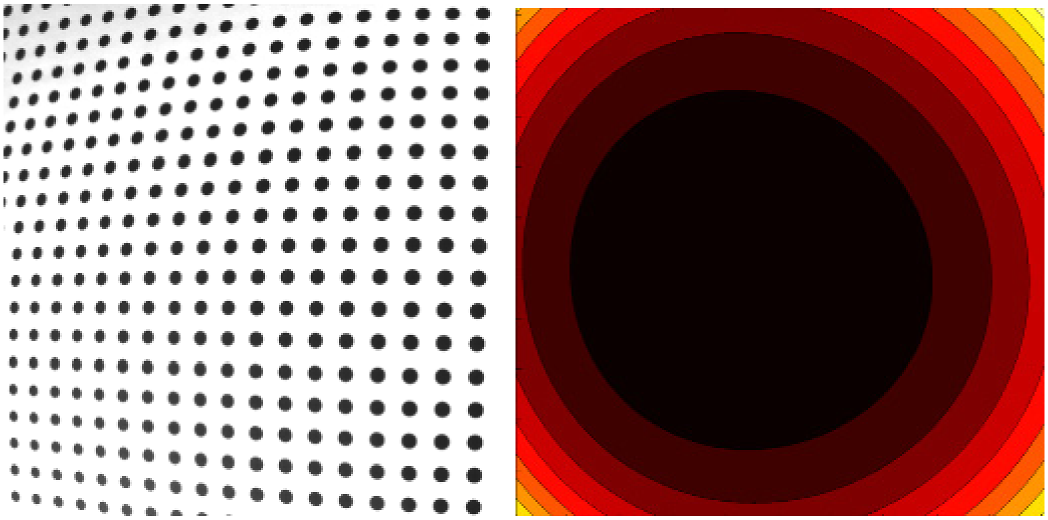
```
Imports System
Imports System.IO
Imports System.Net
Imports System.Net.Sockets
Imports System.Text
Imports mdsiMacroSupportVBX

Public Class MDSITcpServer
    Shared machineID As String
    Shared IP As String
    Shared Port As String
    Shared bDebug As Boolean = True
    Shared ret As mdsiMacroReturnTypes
    Shared mdsi As New mdsiMacroSupportX
    Shared units As mdsiUnitBasisModes
    Shared initret As Boolean

    Public Shared Sub ParseCmdLine()
        'This currently isn't checking for correctness of the command line arguments!!!

        Dim s As String = ""
        Dim j As Integer = 0
        For Each s In My.Application.CommandLineArgs()
            If s = "-m" Then
                machineID = My.Application.CommandLineArgs(j + 1)
            ElseIf s = "-a" Then
                IP = My.Application.CommandLineArgs(j + 1)
            ElseIf s = "-p" Then
                Port = My.Application.CommandLineArgs(j + 1)
            End If
            j += 1
        Next

    End Sub

    Public Shared Sub Main()
        Dim server As TcpListener
        server = Nothing
        Console.TreatControlCAsInput() = True

        Try
            ParseCmdLine() 'Get IP, Port and machine ID from Command Line

            Dim localAddr As IPAddress = IPAddress.Parse(IP)
            server = New TcpListener(localAddr, Port)
            ' Start listening for client requests.
            server.Start()

            ' Buffer for reading data
            Dim bytes(1024) As Byte
            Dim data As String = Nothing

            ' Enter the listening loop -- This loop continues until program is exited
            While (True) 'LISTENING LOOOP
                Console.WriteLine("Waiting for a connection at {0}", server.LocalEndpoint↙
    .ToString)
                ' Perform a blocking call to accept requests.You could also user server. ↙
    AcceptSocket() here.
                ' Wait for Connection here...
                Dim client As TcpClient = server.AcceptTcpClient()
                Console.WriteLine("Connected! from Client {0}", client.Client.          ↙
    LocalEndPoint.ToString)
                'If coming back into loop make sure data = Nothing
                data = Nothing
                ' Get a stream object for reading and writing
                Dim stream As NetworkStream = client.GetStream()
```

```
            Dim i As Int32 = 0
            ' Loop to receive all the data sent by the client. Use try Catch to      ↙
handle forced disconnect from client
            ' First read is done outside while loop.  Additional stream.Read are done↙
 inside while loop
            Try
                i = stream.Read(bytes, 0, bytes.Length)
            Catch serr As IOException
                Console.WriteLine("Socket Forcibly disconnected")
                i = 0
            End Try

            While (i <> 0) 'PROCESS BYTES
                ' Translate data bytes to a ASCII string.
                data = System.Text.Encoding.ASCII.GetString(bytes, 0, i)
                Console.WriteLine("Received bytes {0} = {1}", data.Length.ToString,  ↙
data.ToString)
                'Process Data thru Function parse_Data()
                'parse Data handles all checks on what type of response to give
                data = parse_Data(data)
                Dim msg As Byte() = System.Text.Encoding.ASCII.GetBytes(data)
                ' Send back a response.
                stream.Write(msg, 0, msg.Length)
                Console.WriteLine("Sent {0} bytes = {1}", data.Length.ToString, data.↙
ToString)
                ' Loop to receive all the data sent by the client. Use try Catch to  ↙
handle forced disconnect from client
                Try
                    i = stream.Read(bytes, 0, bytes.Length)
                Catch serr As IOException
                    Console.WriteLine("Socket Forcibly disconnected")
                    i = 0
                End Try

            End While 'PROCESS BYTES
            'Close IO stream, garbage collenct and end connection
            stream.Close()
            stream.Dispose()
            client.Close()

        End While 'LISTENING LOOP
    Catch e As SocketException
        Console.WriteLine("SocketException: {0}", e)
    Finally
        server.Stop()

    End Try

End Sub 'Main
Public Shared Function parse_Data(ByVal value As String) As String
    ' A. Executes a statement if data contains 'mdsiMacroObj' and '='
    ' B. Evaluates a statement if data contains 'mdsiMacroObj'
    ' C. If cO is present -- attempt to remove tool offset
    Dim exec_cmd As Boolean = value.Contains("=")
    parse Data = "Err"
    If value.Contains("mdsiMacroObj") Then
        initret = InitMdsiLibrary() 'Connect to MDSI Variable Database
        If initret Then
            Dim Script_Engine As New MSScriptControl.ScriptControl
            Script_Engine.AllowUI() = False
            Script_Engine.Language = "VBScript"
            Script_Engine.AddObject("mdsiMacroObj", mdsi)

            Try
                If exec_cmd Then
                    mdsi.mdsiReadFreshDataVB(False)
```

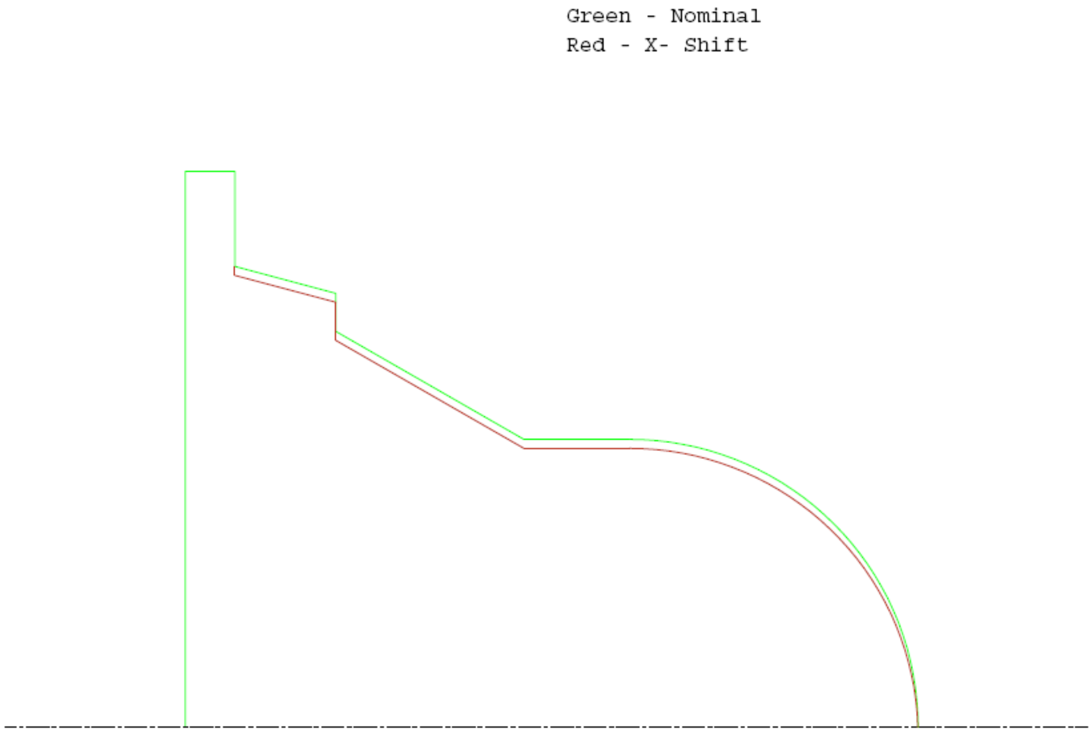
```
                        Script_Engine.ExecuteStatement(value)
                        parse_Data = "OK"
                    Else
                        mdsi.mdsiReadFreshDataVB(False)
                        parse_Data = Format(Script_Engine.Eval(value), "000.00000")
                    End If
                Catch ex As Exception
                    Console.WriteLine("Script Engine Returned: {0}", ex.Message.ToString)
                    parse_Data = "Err"
                End Try
                ' Reset scripting engine and uninitialize
                Script_Engine.Reset()
                Script_Engine = Nothing
                mdsi.unInitialize()
            End If

        ElseIf value.Contains("c0") Then
            'Format is form: c0#, where # is tool offset register to remove
            ' Do this to ensure data is o.k.
            value = value.Remove(0, 2)
            Console.WriteLine("Clearing Offset Table Row {0}", value)
            Dim j As Integer
            Dim k As Integer
            initret = InitMdsiLibrary() 'Connect to MDSI Variable Database
            If initret Then
                Dim num_axes = mdsi.axisCount
                Dim done As Double
                For j = 0 To num_axes - 1
                    For k = 1 To 4
                        Select Case k
                            Case 1, 2
                                mdsi.Offset(j, k, value.ToString, False) = 0.0
                                done += mdsi.Offset(j, k, value.ToString, False)  'Used  ↙
    to check if row actually clear
                            Case 3, 4
                                mdsi.Offset(0, k, value.ToString, False) = 0.0
                                done += mdsi.Offset(0, k, value.ToString, False)  'Used  ↙
    to check if row actually clear
                        End Select
                    Next
                Next
                If done = 0 Then
                    parse_Data = "OK"
                End If
            End If
            mdsi.unInitialize()
        End If

    End Function
    Public Shared Function InitMdsiLibrary() As Boolean
        Dim sinit As String = "mid=" & machineID & "jsID=0 locale=0"
        If mdsi.initialize(sinit, False, False) = -1 Then '-1 = Failure
            Console.WriteLine("MDSI VBX Init Failure")
            InitMdsiLibrary = False
        Else
            InitMdsiLibrary = True
            Console.WriteLine("MDSI VBX Init Success")
        End If
    End Function

End Class 'MDSITcpServer
```

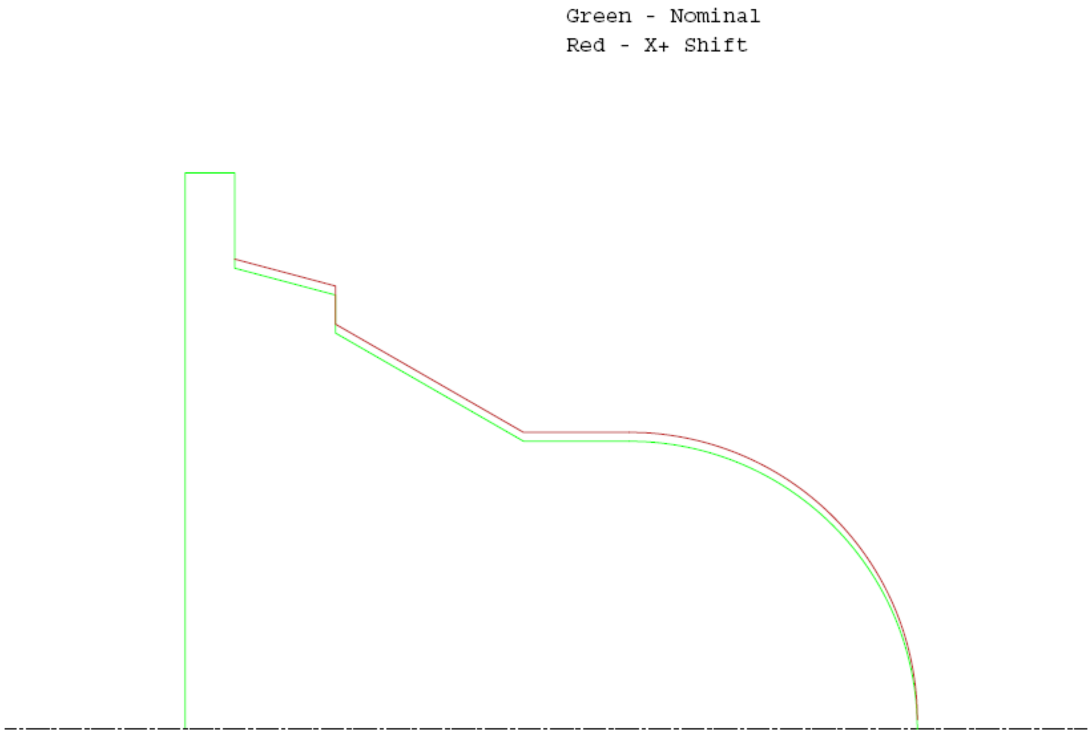
APPENDIX C Example of Calibration Error Map

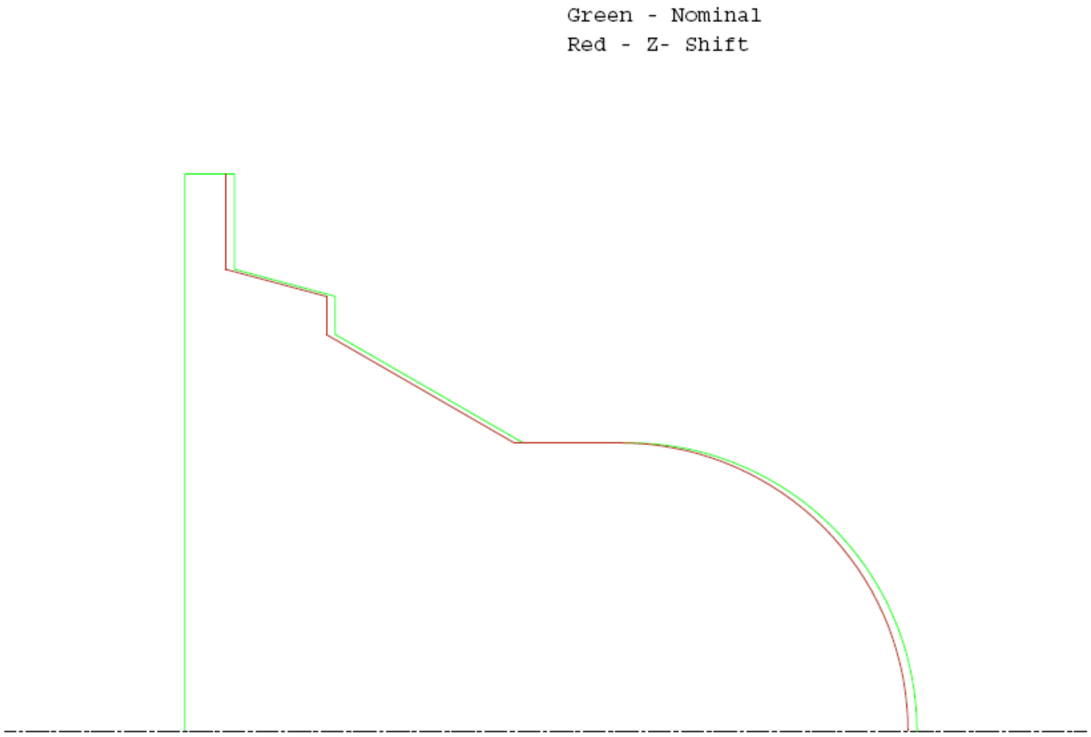
For a further example of calibration error mapping here is an example from National Instruments:

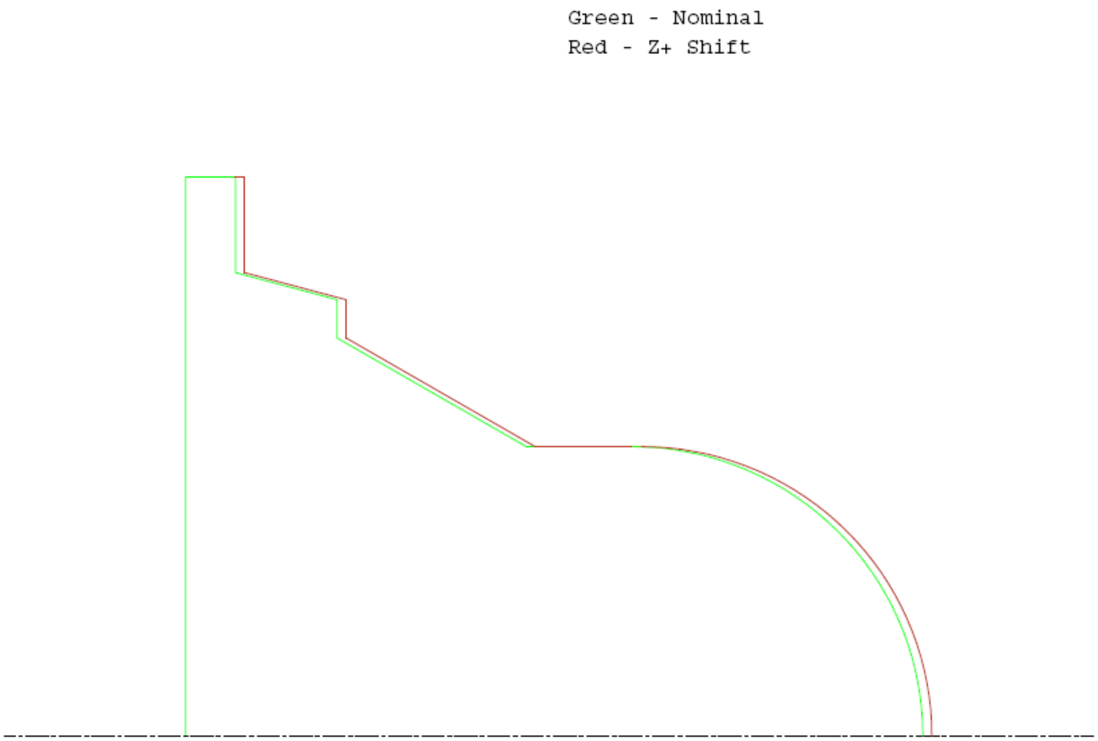


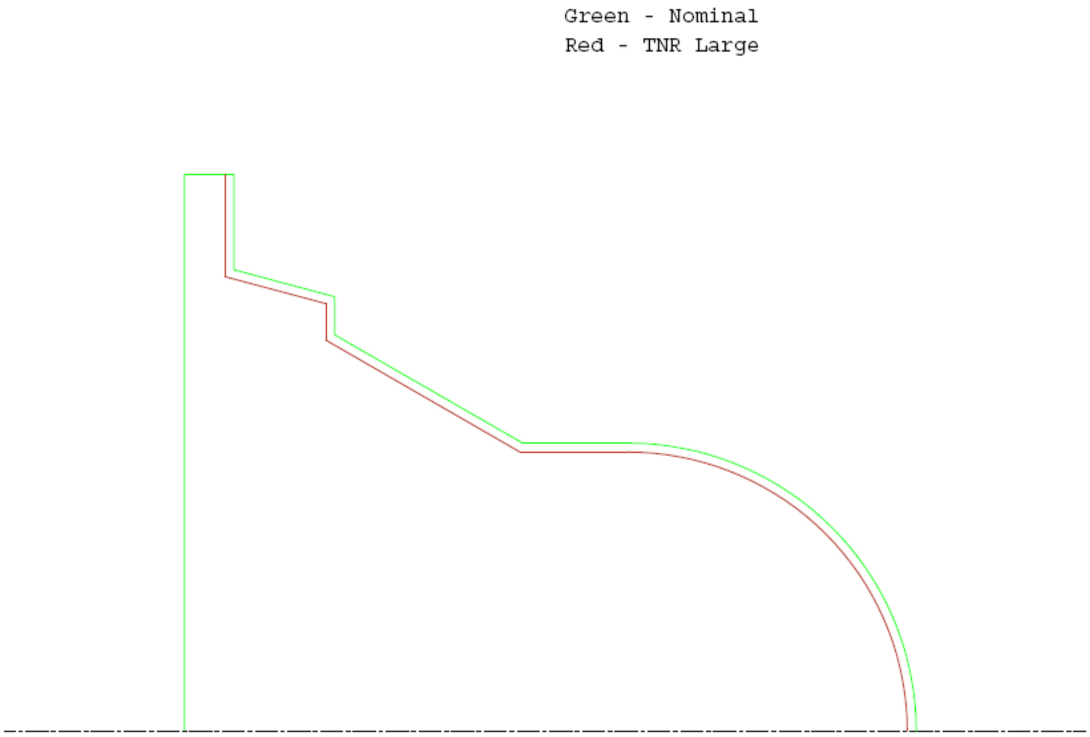
APPENDIX D: Tool Offset and Tool Nose Radius Errors




Green - Nominal
Red - X+ Shift


Green - Nominal
Red - Z- Shift


Green - Nominal
Red - Z+ Shift


Green - Nominal
Red - TNR Large

Green - Nominal
Red - TNR Small

Combined Error:
(Note similarity of Red and Blue lines
from different sources of error)
Green - Nominal
Red - X- shift
Blue - TNR Small AND Z+ shift