Work-in-Progress: An Educational Tool to Support Learning Robot Vision

Dr. Fernando Garcia Gonzalez, Florida Golf Coast University

Dr. Fernando Gonzalez joined FGCU as an Assistant Professor in the Software Engineering Program in the fall of 2013. Previously he has worked at Texas A&M International University in Laredo, Texas, the U.S. Department of Energy at Los Alamos National Laboratory in Los Alamos, New Mexico and at the University of Central Florida in Orlando, Florida. Dr. Gonzalez graduated from the University of Illinois in 1997 with a Ph.D. in Electrical Engineering. He received his Master’s degree in Electrical Engineering and his Bachelor’s degree in Computer Science from Florida International University in 1992 and 1989. Dr. Gonzalez research interest includes the intelligent control of large scale autonomous systems, autonomous vehicles, discrete-event modeling and simulation and human signature verification.

Dr. Janusz Zalewski, Florida Gulf Coast University

Janusz Zalewski, Ph.D., is a professor of computer science and software engineering at Florida Gulf Coast University. Prior to an academic appointment, he worked for various nuclear research institutions, including the Data Acquisition Group of Superconducting Super Collider and Computer Safety and Reliability Center at Lawrence Livermore National Laboratory. He also worked on projects and consulted for a number of private companies, including Lockheed Martin, Harris, and Boeing. Zalewski served as a chairman of the International Federation for Information Processing Working Group 5.4 on Industrial Software Quality, and of an International Federation of Automatic Control Technical Committee on Safety of Computer Control Systems. His major research interests include safety related, real-time embedded and cyberphysical computer systems, and computing education.

Mr. Gerardo Javier Pinzon P.E., Texas A&M International University
With the rising popularity of robotics in our modern world there is an increase in engineering programs that offer an introductory course in robotics. This common introductory robotics course generally covers the fundamental theory of robotics including robot kinematics, dynamics, differential movements, trajectory planning and basic computer vision algorithms commonly used in the field of robotics. To teach robotic vision the student is generally exposed to a variety of vision algorithms where they learn how to combine them along with the selection of their parameters to produce a functional vision software system as specified by the course instructor.

This paper introduces an educational software tool that allows the student to create their software robotic vision system. The tool consists of an integrated development environment (IDE) where the student practices with the different methods and parameters and learns what combination works best for their purpose. While this may appear like a common image processing tool, it’s quite different in that the tool only performs the basic methods studied in the course. This includes image histogram plots, the image threshold operation, low and high-pass Fast Fourier Transform (FFT) based filters, 3x3 mask convolution, Hit-Miss morphological transform, Hough transforms to detect Lines, and Hough transforms to detect circles, random noise addition, image moment computations, and image negative generation. The engineering of the vision system is not performed by the tool, it is only assisting in the process.

**Keywords**: Robotics education, robotic vision simulator, image processing.

1. Introduction

1.1. General Background

This tool is specifically designed to teach the basic Introduction to Robotics course. Many robotics books such as1-8 over this material. The course generally covers robotics fundamentals including history, robot types, and degrees of freedom, robot kinematics including the transformation matrix, forward and inverse kinematics, and the Denavit-Hartenberg (D-H) parameters, differential motions, robot dynamics, trajectory planning, actuators and sensors, and robot vision. In this course robotic vision includes some basic fundamental image processing and computer vision algorithms related to robotics. Generally only between 4 to 8 classes are devoted to this topic as it is only one of several components included in the Introduction to Robotics course. For example in the book by Niku7 only 73 of the 441 pages in the book are devoted to robot vision and therefore it is not intended to be a survey of all possible vision routines. It basically consists of learning about a collection of low level algorithms that perform specific operations on an image or extract specific information from an image.

This course is generally a first course in robotics and may be an undergraduate or graduate level course. In many programs, this course is offered as the first in a sequence of several robotics courses in a robotics program with the goal of graduating students who specialize in robotics. This course has gained a reputation of being very interesting and attractive to students which has led many institutions to offer this course as an independent single course.
with an alternate goal not related to producing robotics experts. The goals range from teaching students how different systems work together such as in the Systems Engineering Program at Texas A&M International University to simply providing an elective course such as in the Software Engineering Program at Florida Gulf Coast University.

The learning outcomes this tool is designed to support are listed below.

After completing this section of the course and with the help of this tool the student will:
1. Be able to perform image cleaning to remove random isolated pixels from the image,
2. Be able to use image histogram and the threshold operation to support image processing.
3. Be able to perform basic low and high-pass filtering using both Fourier transforms and convolution based algorithms,
4. Be able to find edges in an image using convolution,
5. Be able to find the orientation and location of a known object in an image using the image moments,
6. Be able to use Hough transforms to find geometric shapes in an image such as lines and circles to support extraction of information from an image,
7. Be able to use morphological transformation to support image processing.

This tool has been recently merged with another tool that supports the student in learning the other areas of this course not related to robotic vision. The overall tool supports the student in learning the complete course however this paper only presents the vision part of the tool. This part of the tool only supports the students in learning the topics related to robot vision as described in the learning outcomes listed above.

This tool differs from other existing tools designed to support robotic vision in that it is not a library of routines which requires the student to create programs but rather it is a standalone tool design to be used with no programming. Peter Corke\(^8\) has developed a library of MATLAB functions and has made it available free\(^9\). This library is very popular but requires the student to write programs in MATLAB. MATLAB has its own image processing toolbox\(^12\) as well. While writing programs is far superior in helping the student learn robotics, it is not always feasible especially at institutions that do not have a robotics program. For example in the Systems Engineering Program at Texas A&M International University programming is a very small part of the engineering curriculum where students are not expected to be able to create whole programs yet they still offer the robotics course as required for the major. It is not reasonable to add learning how to program to the robotics course contents. This course in the Systems Engineering Program was the initial motivation for creating our tool. OpenCV\(^9\) is another very popular tool used for computer vision but it is also a library of functions written in C++. There are many standalone tools that do not require programming such as Irfanview\(^11\) and Gimp\(^13\) however these tools are not specifically designed for learning the fundamental routines but rather they are designed to actually do the image processing. When designing this tool we took care not to have the tool do too much. For example, in Irfanview, to sharpen an image you simply select the sharpening option in the image menu and it does all the work. In contract in the presented tool sharpening requires the student to select the proper filter and its parameters.

In our tool each algorithm needs to be given parameters to get it to perform the desired tasks. The student must learn how to select the combination of algorithms and how to select the
parameters for each in order to accomplish the desired tasks. Due to their simplicity, implementing the algorithms has minimal educational benefit and yet requires significant amount of time. Since the main educational objective is to learn how to find the combination of methods each with their own parameters that produces the best result for their assigned task, the best tool is one that provides all of the algorithms and allows the student to simply use the algorithms and their parameters.

The tool consists of an image rendering panel with ports to each of the image processing routines. The graphical user interface (GUI) is shown in Figure 1. The instructor assigns a task and either provides the images or allows the student to acquire them. A typical task will be to have the student identify a sequence of operations along with their parameters to identify the object among a selection of possible objects. The task may also require the student to locate and determine the orientation of the object. The student enters the image into the tool and then begins to investigate the different sequence of routines with their associated parameters. The student selects a routine to perform and provides the parameters. The image then is processed by the routine. This allows the student to use routines without implementing them. The routines are elementary image processing routines that are only as good as the selection of input parameters. This is not like a commercial image processing tool where the routines are very complex, carefully designed to perform some well-defined task in an optimal way. For example, one cannot tell the tool to clean an image. Rather one tells the tool to perform a series of elementary operations with specified parameters which will then hopefully clean the image. The engineering is not performed by the tool but left to the student.

2. Tool description

This section describes the tool in detail. It includes a list of operations the tool can perform with figures illustrating the result of each operation.

An image is a two dimensional array of pixels representing grey levels or colors. In robotics it’s very common to use grey levels only and therefore the simplest format is to use one byte per pixel representing the grey level or darkness of the pixel.

The IDE can read an image into the tool but it must be in portable graymap format (PGM). This image file format is part of the Netpbm format family that stores the image as a simple uncompressed ASCII file using 1 byte per pixel. Therefore each pixel can represent a value between 0 and 255 with 0 being black and 255 being white. Most file formats can be converted to PGM using free software such as Irfanview11.

The tool includes functions that implement the following algorithms that are described in detail further below. The elementary operations the tool can perform at this time are as follows:

- Histogram
- Image threshold
- Low and high-pass FFT based filters
- 3x3 mask convolution,
- Hit-Miss morphological transform
- Hough transforms to detect Lines
- Hough transforms to detect circles
- Add random noise
- Image moment computations and image negative.

In each case the IDE displays the image on the left and a control panel on the right. The student loads an image then performs a sequence of operations. The engineering is in the selection of the sequence of operations and the parameters each operation needs. Following is a description of all the operations and the required parameters each must have.

Figure 1: The graphical user interface of the IDE. Default values are present where possible.

2.1 Example

Consider the following example assignment. The student will use the tool to learn how to design a vision system that can determine the location and orientation of the handle of the screwdriver in the image so that the robot’s hand can pick it up. In Figure 2 (a) the original grey scale image is shown. The first step is to perform an image threshold application to
convert the image from grey scale to black and white. The student may perform a histogram to see the distribution of the grey levels to determine a good threshold value. In Figure 2 (b) the histogram of the image is shown. The values range from 0 to 255. The student may try different values and see the result. In Figure 3 (a) the image with a threshold of 128 is shown. The student can see that there is some background grey areas that were darker than the threshold value of 128 which means it had a value less than the threshold. The student may try a smaller value. In Figure 3 (b) the image with a threshold of 96 is shown. The student can see that only the really dark pixels that are part of the screwdriver are left black and therefore learned that the threshold for this image of less than the midpoint but greater than the quarter point or around 96. The histogram and threshold operations are designed to support learning outcome 2, “The student will be able to use image histogram and thresholding to support image processing.”

![Figure 2: (a) The original image. (b) The histogram of the image.](image)

Next the student wishes to determine the orientation of the screwdriver by using the Hough transform to detect lines. Unlike the other algorithms that are relatively simple and not worth having the student spend time on implementing those, implementing the Hough transform is more challenging. It is more beneficial for the student to implement the algorithm which varies depending on the shape it is to detect. However in the situations where this is not feasible, instead of simply skipping this learning outcome the tool can be used to show the student the type of results the transform can give. A parameter of the algorithm is to determine the number of shapes to detect which the student can learn. In Figure 4 (a) the Hough transform for lines is applied and the top 20 line is superimposed on the image. The student can see that some lines are not aligned with the screwdriver and sees a potential problem if this transform is used to guild the robot to use the correct angle of the hand. The student may realize that perhaps the top line is the most aligned and the others become less aligned since there should be more pixels in the line that is most aligned. Next the student selects the routine to only show the top line. Figure 4 (b) shows the image with only the top
line superimposed. This feature supports learning outcome 6 “The student will be able to use Hough transforms to find geometric shapes in an image such as lines and circles to support extraction of information from an image.” The red line runs along the length of the tool and the angle of this line with respect to the horizontal axis is the screwdriver’s orientation. The equation of the line is not displayed at this time.

![Figure 3: (a) The image after a threshold operation with a parameter of 128. (b) The image after a threshold operation with a parameter of 96.](image)

Now the student wants to find the location of the screwdriver. He studied the information that can be generated by computing the moments of the image. The theory shows that the centroid of the screwdriver is computed by

\[
\frac{M_{1,0}}{M_{0,0}} = \bar{x}, \quad \frac{M_{0,1}}{M_{0,0}} = \bar{y}
\]

Where \( M_{1,0} \) is the first moment about \( x \), \( M_{0,1} \) is the first moment about \( y \) and \( M_{0,0} \) is the zeroth moment. Figure 5 (a) shows the moment report generated by the tool. From here the student can use the values of the needed moments. Figure 5 (b) shows the centroid highlighted by the red dot that was computed using the moments. This feature supports the student with learning outcome 5, “Be able to find the orientation and location of a known object in an image using the image moments.”

The student learns by trying different routines in series with different parameter until something useful is discovered. For example after trying several combinations of the convolution and other routines it was discovered that using the threshold with a value of 96 followed by an Emphasis-Horizontal convolution results in the image shown in Figure 6 (a). Negating the image then results in an outline of the image as shown in Figure 6 (b). This is not needed for the location and orientation of the screwdriver but could be used to determine
how wide to open the gripper. This features supports learning outcome 4 “The student will be able to find edges in an image using convolution.”

![Image with Hough transform results](image.png)

Figure 4: (a) The image with the top 20 lines found by the Hough transform. (b) The binary image with the top line found by using the Hough transform.

One can see that the student can learn a lot by using the different routines to perform some useful function without the need to write programs. While having the students write the routines or even write a program that simply calls the routines may be more effective to learning robot vision, it should be obvious that this tool is effective in the situation where the students will not write any programs. The following is a list of the different features and their parameters.

### 2.2. Histogram

A histogram is a chart representing the total number of pixels in an image with a particular value. See Figure 7. The horizontal axis is the pixel value from 0 (black) on the left to 255 (white) to the right and the vertical axis is the number of pixels with that value. This information is useful for many techniques including the image threshold operation presented next. Supports learning outcome 2 “The student will be able to use image histogram and thresholding to support image processing.”

### 2.3. Image threshold operation

The image threshold operation is the function of visiting every pixel and making it 0 if the value of the pixel is less than the threshold and 255 if it is greater. See Figure 8. The threshold value is a function of the image and depends on many factors such as the darkness, the value of the information you want to keep and other factors. A poor choice may return an image that is almost all white or black. The student needs to determine what value to use. The use of a histogram may provide some information to help determine this threshold value.
Supports learning outcome 2 “The student will be able to use image histogram and the threshold operation to support image processing.”

Figure 5: (a) The moment report generated by the tool. (b) A red dot superimposed at the centroid of the black area calculated by using moments.

Figure 6: (a) The binary image with an Emphasis-Horizontal convolution applied. (b) The same image negated.
Figure 7: Left, the original image, right the histogram of the image.

Figure 8: The original image of the screwdriver after a threshold operation with limits of (a) 10, (b) 50, (c) 100, and (d) 150.

2.4. Low and high-pass FFT based filter

Filtering is the process of removing a group of frequency components from the image. The low frequencies tend to represent smooth areas of the image such as the background while the high frequencies represent the larger abrupt changes. A high-pass filter can be used to emphasize the edges by removing the frequencies that do not contribute to the edges. Filtering frequencies is tricky because the frequencies are integrated. That is you cannot
simply chop off all frequencies below some threshold. This produces an image with waves. A Butterworth filter has a gradual cutoff.

Image filtering can be performed by using the Fourier transform. The algorithm basically performs the convolution of a Butterworth filter and the image. First a Fast Fourier Transform (FFT) is performed on the image producing an image frequency spectrum. Then the spectrum is multiplied by the Butterworth filter and then an inverse FFT is performed resulting in the original image convoluted with the filter. The discrete Fourier transform is

\[
F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp \left( \frac{-2j\pi(ux + vy)}{N} \right)
\]

(1)

Where \(x\) and \(y\) are the column and row of the pixels and \(u\) and \(v\) are the coordinates in the frequency spectrum. The inverse transform is

\[
f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp \left( \frac{2j\pi(ux + vy)}{N} \right)
\]

(2)

Note the image must be square and of size \(2^n \times 2^n\) where \(n\) is an integer. All the images shown in this paper are 512 x 512 pixels. The low-pass Butterworth filter is

\[
H(u, v) = \frac{1}{1 + \sqrt{2} - 1 \left[ \frac{D(u, v)}{D_0} \right]^{2\pi}}
\]

(3)

And the high-pass is

\[
H(u, v) = \frac{1}{1 + \sqrt{2} - 1 \left[ \frac{D_0}{D(u, v)} \right]^{2\pi}}
\]

(4)

Where \(D_0\) is the cutoff frequency and \(D(u, v)\) is

\[
D(u, v) = \sqrt{u^2 + v^2}
\]

(5)

Note, the frequency spectrum gives information about the frequency components in the image. However frequency is a function of time and the unit Hz is cycles per second. Since in an image there is space instead of time we assume the sample rate is \(d = \frac{1}{512}\) second as a default so that the frequency components range from 0 Hz to 255 Hz in increments of \(\frac{1}{Nd} = \frac{1}{512 \times 1/512} = 1\) since there image is 512 x 512. Since half of the frequencies are negative and identical to the positive in magnitude the highest frequency is then \(\frac{N/2}{2d} = \frac{1}{2d} = 255\).
In Figure 9 the original and filtered image is shown. The filter is a 10th order high-pass and the cutoff frequency is 50 Hz. In Figure 10 the frequency spectrum of the original image is shown in (a) and the cross section of the positive side is shown in (b). Note the lower frequency components have more magnitude. This is very normal for most all images. In Figure 11 the filter is shown (a) and its cross section is shown in (b). Note the black is 0 and the white 255 so the hole in the middle is where the lower frequency components are in the spectrum and they get zeroed when the filter is multiplied by the spectrum. Note the cross section of the filter is not sharp. This gradual change from 0 to 255 over some distance is governed by the order and is a parameter the student needs to determine. In Figure 12 (a) the frequency spectrum of the filtered image is shown in (a) and the cross section of the positive side is shown in (b). The cutoff frequency is the other parameter that needs to be determined.

The order (1st, 2nd or 3rd) of the filter determines sharpness of the edge of the filter. The cut-off frequency determines the frequency boundary to filter. One can perform a low-pass filter where frequencies above the cut-off frequency are attenuated or a high-pass filter for frequencies below the cut-off. The student needs to learn how to select the parameters to perform desired filtering. The student can run algorithms using different parameter combinations and observe the result.

Figure 9: (a) The original filter. (b) The image after an application of a 10th order high-pass filter with cut-off frequency of 50Hz.
Figure 10: (a) The frequency spectrum of the original image. (b) The cross-section of this spectrum.

Figure 11: (a) The frequency spectrum of the Butterworth filter. (b) The cross-section of this spectrum.
Next we present several combinations varying the order and the cutoff frequency of both high-pass and low-pass filtering. In general the high-pass filter is used to emphasize the edges of the image. In this case we want to emphasize the edges of the screwdriver. Note for this particular image the filter produced bands around the object. There are other filtering techniques such as convolution that work better for this image. In general the FFT filtering works better with scenes. In Figure 13 (a) the 10th order filter with a cutoff frequency of 50 Hz is shown, (b) the cross section is shown wand in (c) the filtered image are shown. Figure 14 shows the same using a 2nd order filter with the same cutoff of 50Hz. In Figure 15 the cutoff was increased to 150 Hz while keeping the order at 10. The low-pass filter produces images that are better suited for images of discrete parts. However the convolution method still produces a better result. In Figure 16 a 10th order filter with a 50Hz cutoff frequency is shown. Figure 17 shows the same but using a 2nd order. Finally in Figure 18 we present the same 10th order filter with a cutoff of 150 Hz. This feature supports learning outcome 3 “The student will be able to perform basic low and high-pass filtering using both Fourier transforms and convolution based algorithms.”

2.5. Mask convolution

Convolution is a family of algorithms used for filtering, edge detection, morphology, testing for connectivity, etc. The algorithm consists of passing a 3x3 pixel mask over the image visiting each pixel and replacing the pixel in the image directly under the center pixel of the mask with a value generated by multiplying each of the 9 pixels in the mask with the corresponding pixel in the image that is directly under it.
Figure 13: A high-pass filter with an order of 10 and a cutoff frequency of 50 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.

Figure 14: A high-pass filter with an order of 2 and a cutoff frequency of 50 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.

Figure 15: A high-pass filter with an order of 10 and a cutoff frequency of 150 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.
Figure 16: A low-pass filter with an order of 10 and a cutoff frequency of 50 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.

Figure 17: A low-pass filter with an order of 2 and a cutoff frequency of 50 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.

Figure 18: A low-pass filter with an order of 10 and a cutoff frequency of 150 Hz. (a) The Butterworth filter. (b) The cross section of the filter. (c) The filtered image.
To understand why convolution can be used to filter an image, it should help to know that multiplying the complex conjugate of an image by a filter spectrum like the Butterworth filter presented above is equivalent to performing the convolution of the image with the inverse FFT of the filter. The filter is created in the frequency domain so an inverse FFT will transform the filter to the spatial domain. The equation below expresses this concept.

\[ f(x, y) \circ h(x, y) \Leftrightarrow F^{-1}(u, v)H(u, v) \]  

Mask convolution is not the same as convolution between the two images since the mask is only 3x3 and not the same direction as the filter image. With the images presented here this smaller convolution works better.

The process that this algorithm performs, whether it’s filtering, edge detection or another task is dependent only on the mask used. The engineering the student must learn is in the selection of the optimal mask (choosing 9 values in the mask). Most textbooks present a few masks that perform specific processes. Several of the most common masks are loaded into the tool and the student can select them. Presented are some of the common masks and their effects on the image. In Figure 19 (a) the Laplacian-1 high-pass filter mask is shown and in (b) the filtered image is shown. In Figure 20 (a) the Laplacian-2 high-pass filter mask is shown and in (b) the filtered image is shown. The orientation of the screwdriver has an effect on the difference between these two filters. In Figure 21 (a) the Sharpen-Low high-pass filter mask is shown and in (b) the filtered image is shown. In Figure 22 (a) the Roberts-Horizontal edge-detection mask is shown and in (b) the convoluted image is shown. In Figure 23 (a) the Prewitt-Horizontal edge detection mask is shown and in (b) the convoluted image is shown. These features supports learning outcome 3 “The student will be able to perform basic low and high-pass filtering using both Fourier transforms and convolution based algorithms” as well as learning outcome 4 “The student will be able to find edges in an image using convolution.”

![Figure 19: (a) The Laplacian-1 high-pass filter. (b) The original image after convoluted with the mask.](image-url)
Figure 20: (a) The Laplacian-2 high-pass filter. (b) The original image after convoluted with the mask.

\[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
\]

Figure 21: (a) The Sharpen-Low high-pass filter. (b) The original image after convoluted with the mask.

\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0 \\
\end{bmatrix}
\]
2.6. Hit-Miss morphological transform

Morphological transformations are a family of algorithms that change the shape of the figure in the image\textsuperscript{14}. This algorithm is similar to convolution but involves a collection of masks. Like the convolution, the Hit-miss algorithm slides the masks over the image making the center pixel under the mask on or off depending on whether the pattern in the neighboring pixels matches one of the 256 masks. The masks are specified in a 16 x 16 array where a 1 indicates to include the corresponding mask into the set. A set of masks must be used.
specified for each morphological operation. Since it is impractical to have 256 text boxes in
the GUI, the student does not create the set of masks rather they select which set to use from
a list. Currently only the thinning set is included in the tool.

It is required to thin an image to perform many other image processing techniques. For
example, Figure 24 shows how an image changes after using 7 applications of the Hit-miss
algorithm using the thinning mask set. Each application peals a layer of the shape off. The
algorithm guarantees not to disconnect an object while thinning. This feature supports
learning outcome 7 “The student will be able to use morphological transformation to support
image processing.”

![Figure 24: (a) An image of some lines and circles. (b) The same image thinned using 7
applications of the thinning Hit-miss algorithm.](image)

2.7. Hough transforms to detect specific shapes

The Hough transform is a family of algorithms that are used to detect geometric shapes in
an image. The engineering is in the design of the transform, determining the image
processing needed before the algorithm can be applied to the image, and determining what
can be done with the generated information. To find a particular shape the student needs to
find the equation of the shape, and then put it into a form where the constants are the
variables and the variables become the constants and finally solve it for the constants. A
system of counters needs to be designed to count the number of pixels that contribute the
each shape. Since implementing this algorithm from scratch may be too difficult or time
consuming for an introductory course, two algorithms are included in the tool, to detect line
and to detect circles. The engineering then becomes the design of the preprocessing of the
image and the appropriateness of the particular image. For example, Figure 25 (a) shows how
running the Hough transform to detect lines produces a set of the top 20 lines which are then
superimposed on top of the original image (b). Knowing how tricky these algorithms are is
also part of the learning process. This feature supports learning outcome 6 “The student will
be able to use Hough transforms to find geometric shapes in an image such as lines and circles to support extraction of information from an image.”

Figure 25: (a) The thinned image. (b) The image with the first 20 lines the Hough Transform found superimposed.

2.8. Adding random noise

To help the student practice cleaning an image the tool provides a function to add random noise to the image. The student enters the percent of the pixels that will become white. Figure 26 shows how making 2% (a) and 10% (b) of the pixel white affects the image. The pixels are selected randomly.

Figure 26: (a) White pixels are added to 1% of the image selected randomly. (b) White pixels are added to 10% of the image selected randomly.
2.9. Image Moments

The moments of an image is defined as

\[
M_{a,b} = \sum_{x,y} x^a y^b I_{x,y}
\]  

(7)

Where \( I_{x,y} \) is 1 if the pixel at coordinate \((x, y)\) in the image is black and 0 if the pixel is white. In the image black is considered to be part of the object and white is part of the background. Each moment has some particular information about the object in the image. Note this process does not change the image but rather it only produces numbers associated with each moment. Some of the more popular moments are listed in Table 1 below. If the object moves within the image then some of these 2nd and higher moments change values and any correlation created with those values will be lost so we sometimes use central moments that are position invariant as defined in Equation 8 below.

\[
\mu_{a,b} = \sum_{x,y} (x - \bar{x})^a (y - \bar{y})^b I_{x,y}
\]  

(8)

Table 1: Moments and their description.

<table>
<thead>
<tr>
<th>Moment</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_{0,0} = \sum x^0 y^0 )</td>
<td>The number of black pixels in the image. The area of the object.</td>
</tr>
<tr>
<td>( M_{0,1} = \sum y )</td>
<td>The sum of all the y coordinates of the black pixels in the image.</td>
</tr>
<tr>
<td>( M_{1,0} = \sum x )</td>
<td>The sum of all the x coordinate of the black pixels in the image.</td>
</tr>
<tr>
<td>( M_{1,1} = \sum xy )</td>
<td>The product of inertia of the area.</td>
</tr>
<tr>
<td>( \frac{M_{1,0}}{M_{0,0}} = \bar{x}, \quad \frac{M_{0,1}}{M_{0,0}} = \bar{y} )</td>
<td>The centroid of the object. The coordinate of the center of mass of the object.</td>
</tr>
<tr>
<td>( M_{2,0} = \sum x^2 )</td>
<td>The 2nd moment of the area relative to the x-axis.</td>
</tr>
<tr>
<td>( M_{0,2} = \sum y^2 )</td>
<td>The 2nd moment of the area relative to the y-axis.</td>
</tr>
</tbody>
</table>

The second and higher moments changes as the shape rotates and so a correlation between the value and the rotation can be used to measure the angle of rotation of the object. Figure 27 (a) shows the four 0th and 1st moments that are generated and in (b) the image is shown with a red dot at the centroid. This feature supports learning outcome 5 “The student will be able to find the orientation and location of a known object in an image using the image moments.”
3. Assessments

This tool has not yet been used with a class. We are planning to use it during the current semester and will assess its appropriateness and usefulness in practice. However please note that all the other tools design to support this specific course require the students to have previous programming skills making it not suitable to programs where the students do not have this programming skill. Consider also that some instructors do not want to have the students invest the extra time it will take to have them write programs even if they know how. This tool is not competing with those other tools but rather filling a gap where no other tool exist with this focused application. Since all the students must be treated equally, separating the students into two groups and giving each group a different tool will lead to some student having to write programs while the other does not. Therefore a survey is the only practical assessment tool we can use. The survey will ask the basic question “did this tool support your learning in robot vision” in addition to the questions related to how they like the tool and how easy it was to use. The questions follow. The answer choices are (strongly agree, agree, neutral, disagree, and strongly disagree) numbered from 4 down to 0.

1. This tool gave me support in learning robotic vision?
2. This tool was not aligned with the course content.
3. This tool was difficult to use.
4. Assignments using this tool was better than simple pencil and paper homework problems.
5. I learned how to use the tool very quickly.
6. The tool did not help me in understanding robotic vision.
7. I prefer to use this tool than to write a program using a library of image processing routines.
8. Using this tool was easy.
9. I spent a lot to time learning to use the tool.
10. I would have preferred to write a program using a library of routines than using this tool.
Note many question are redundant and phrased differently to make sure they do not change their view based on how the question is worded. The survey will be given after they complete the vision assignment which is towards the end of the course so the results will not be ready in time for this publication but will be presented in the presentation.

4. Future Enhancements

This tool was recently merged with another tool that is used to support the same course. That other tool is used to support teaching robot kinematics and programming. The two tools have been merged to a single executable however they have not been integrated. A future enhancement is to integrate the vision with the robot movement so that the student can identify, locate and determine the orientation of the part then program the arm to move to the part and grab it. Another enhancement is to add an equation editor that will allow the student to type an equation and the tool will compile and evaluate the equation. This will allow the student to create more complex moment equations such as invariant moments that are made of a function of standard moments. Other programming features will then be added such as to implement loops and if statements. This will allow the student to create more complex programs that may include a sequence of image processing steps and analysis. Ideally the student will create a program that given any image of a set of images will identify and determine the location and orientation of the object.

5. Conclusion

Presented in this paper is a tool used to support the robotic vision portion of the basic Introduction to Robotics course. The tool allows students to run many common robotic vision algorithms without having to implement each one in code. The tool only performs the basic operations leaving the student to design the system by selecting which operations to run and by selecting their parameters.

References
9. Peter Corke MATLAB tool box for robotics. URL: http://www.petercorke.com/RVC/top/toolboxes/
10. OpenCV URL: http://opencv.org/
13. GNU Image Manipulation Program. URL: http://www.gimp.org/