Development of an Imaging Platform for Maize Kernel Morphology

University of Wisconsin - Madison

Thesis for M.S. Electrical Enginering

Author:
Nicholas Dimick

Supervisor:
Dr. Nicola Ferrier

Friday 14th June, 2013

I would like to thank Professor Ferrier, Professor Spalding, and my parents. Without their support, I certainly would not have made it very far in my pursuit of learning and growth.

Abstract

The field of computer vision can play a significant role in measuring features that describe the physical appearance of maize kernels. Extending a computer vision system to an almost automatic process can create a high througput solution that provides accurate data. Previously, such a system was designed in order to show a proof of concept. In order to fully realize that design, both positives and negatives needed to be considered to develop the fully functioning imaging platofmr. After developing and putting the system to use, it was discovered the original design was unable to account for all shapes, sizes, and colors of maize kernels. This lead to a redesign and implementation of the feature extraction method. The end result has led to an imaging platform that can provide accurate feature data at a high throughput rate.

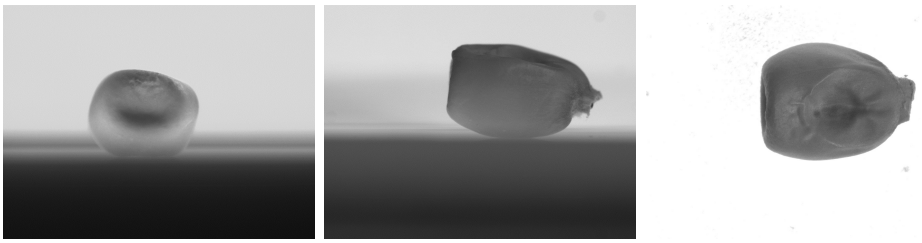
Contents

1 Introduction

Computer vision has become a strong field within Botany. The tools can be used to classify types of gains or identify plant species [11, 12]. More recently, computer vision is being used along with machine learning tools to find important shape and color features for maize kernels [8]. Combining these results with genetics and other physical features, it may be possible to identify favorable traits within maize kernels through new processes. In order to obtain these shape features, the process must remain the same, but it is not uncommon for maize kernels to exhibit varyingly different physical characteristics. I have focused on improving and upgrading a method used to gather shape and color features of maize kernels.

1.1 Maize

Maize kernels are the subject of this experiment. Three regions of the kernels were targeted for features as seen in figure 1. The top view of the kernel corresponds to the region of the germ. In order to create color features, images of the cap were also gathered. Finally, the side of the kernel was examined to obtain the three orthogonal views of the kernel. The tip of the kernel (side opposite of the cap) can be seen in the top or abgerminal view. While this was not a targeted part of the kernel, this portion did have an effect on the segmentation as seen later.



(a) Cap View (b) Side View (c) Abgerminal View

Figure 1: Imaged regions of the maize kernels.

1.2 Background

Previously, work completed by Navdeep [4] focused on creating an experimental setup to gather data and classify maize kernels. The process involved three steps. First, images of the kernels were gathered using three digital cameras. Next, a selection of common features used in classification were extracted from the images. This step involves segmenting the image so only the pixels representing the kernel are identified. Finally, using machine learning algorithms, the features were used as a method to classify the different kernels. This final step shows the features are useful in order to identify the different mutations.

The goal of my work was to develop a system that could process any maize kernel, regardless of shape, size, or color, to be imaged on the same platform. As a temporary solution, two different setups had been used. One with a dark background and another with a light background. This forced the users to adjust and recalibrate the setup. In addition, software written to extract features needed to distinguish if an image used a light kernel on a dark background or a dark kernel on a light background. Switching to a uniform setup speeds up the work flow and helps to remove bias created by having to frequently adjust the workstation setup.

1.3 Goals

The goal of my work here was to design, implement, and upgrade a more permanent system that would accomplish the same objective as laid out by Navdeep. This led to many adjustments such as mounting cameras to fixed locations and improving code to be faster and more reliable. Most significantly, these changes have led to a system that is more robust under a wider variety of kernel appearances.

In the original problem, all of the kernels used had a light color as seen in figure 2. An object with this appearance is almost ideal when it comes to segmentation. Placed on a black background, a computer has little trouble identifying which pixels refer to the background and which refer to the kernel.

The problem is the kernels used here do not accurately represent all the wide variety of possible kernel appearances. If a kernel has a darker color, it completely vanishes against a black background. Figure 3 shows this effect. Probably the most obvious solution would be to use a white background for the dark color seeds, but this has many problems which are described in

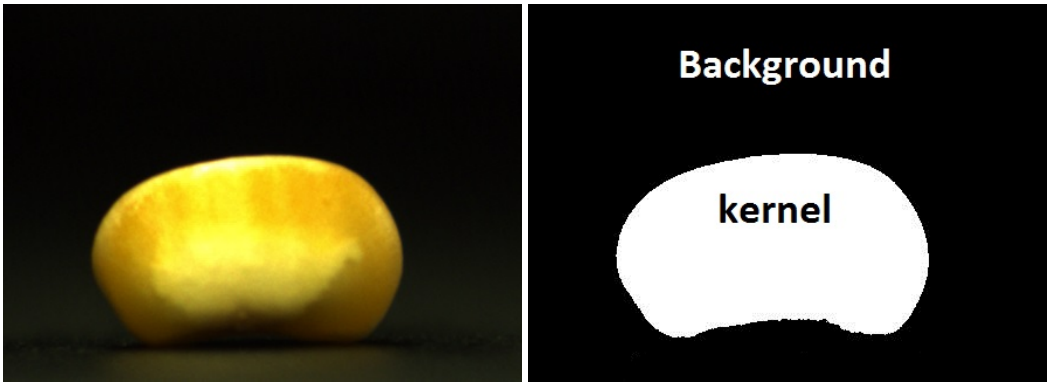
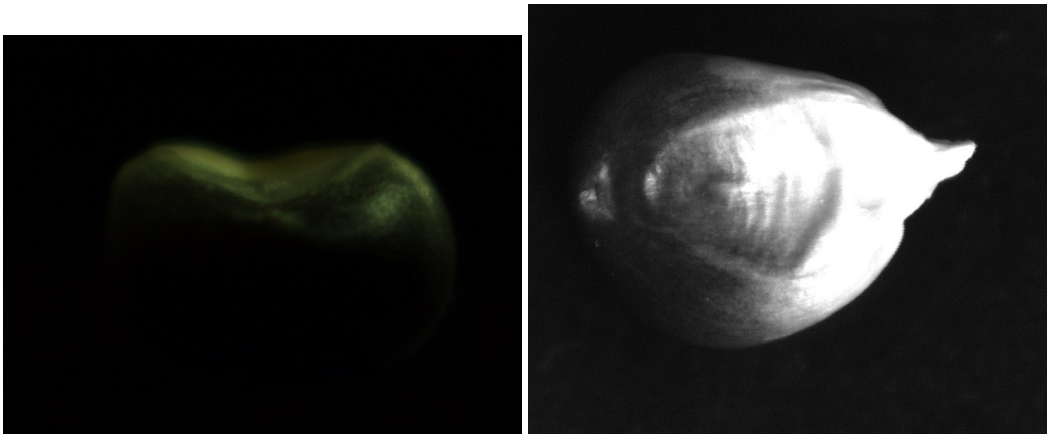


Figure 2: In previous work, the typical kernel used had a light color and placed against a dark background as seen here. The background and kernel pixels can then be identified.

a later section. Due to this new problem, my goal became to improve the imaging system such that it would be able to identify kernels of any shape, size, or color.



(a) View of the Cap (b) View of the germinal side of the kernel

Figure 3: How a dark colored kernel appears on a dark background.

2 Image Features

The purpose of this entire system is to extract a selection of features that can be used to describe the shape and color of every kernel imaged. These features then become not only the final result, but also a means of evaluation. All of the features can only be identified after creating a mask which will label which pixels are background and which are part of the maize kernels.

Several different types of features are commonly used in classification of plant seeds [8, 12]. Area, major axis length, minor axis length, and eccentricity make up the most intuitive features. In addition, six of the Hu Moment Invariants [5] are used to give a total of 10 shape features per view.

Aside from the shape, color can also provide features when classifying different varieties of maize kernels [1, 8]. In this project, average Red, Blue, Green, Hue, Saturation, and Value/Intensity are examined leading to a total of 6 color features. The last 4 features are referred to as Color Matching Features and histograms of the Hue and Saturation values.

2.1 Color Space

Three different color spaces were used in this work: RGB, HSV, and grayscale. A grayscale image was used most commonly. Each pixel has a single value ranging from 0-255. Thresholding, a method described later, can only be applied to a single color channel at a time, so all images are converted to grayscale at some point.

The RGB and HSV color spaces have three channels each. In an RGB image, the red, green, and blue channels can all range from 0-255. The saturation and value channels of HSV range from 0-255 as well, but hue can take a value from 0 to 360.

2.2 Thresholding

The threshold method converts a single channel image into a binary image with all pixel values having a value of 0 or 1 (or a maximum value such as 255). When applying a threshold, a limit value is chosen and under normal conditions, every pixel with a value above the limit is set to 1 (or a maximum value) and everything else is given a value of 0. Equation 1 is a formal description of this method.

$$J(x,y) = \begin{cases} maxVal & \text{if } I(x,y) > limit \\ 0 & otherwise \end{cases} \quad (1)$$

2.3 The Mask

In order to obtain all of the image features used to describe the kernels in this experiment, a mask image is created through thresholding. The mask is defined as a binary image where all the pixels have either a value of 1 or 0 as seen in figure 4. The process to compute this mask is described in section 2.2.

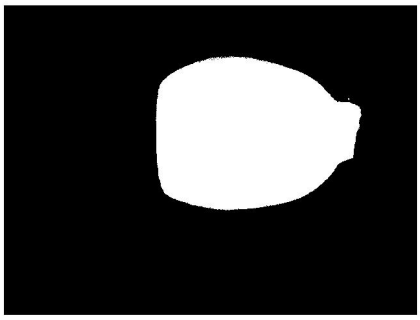


Figure 4: A mask image used for finding feature values.

The purpose of the mask is to limit feature analysis only to pixels that correspond to the object of interest, a kernel in this case. In the mask, any pixel that has a value of 1 represents a pixel that is part of the kernel in the image. If a pixel has a value of 0, then it is labeled as part of the background and can be easily ignored.

It is assumed the mask will always be a white object with a black background. If an object is brighter than its background, then this will be the result of applying a threshold. In the case the object is darker than the background, it is simple to negate the result since it is a binary image and will produce the expected result.

In the shape analysis, the features are found by using this mask image directly. Since pixels with value 0 contribute nothing to these features, it is important to correctly label the background pixels.

Unlike the shape analysis, the color values of the pixels are required. Instead of reading the mask directly, the analysis uses it as a guide. For each pixel in the mask, if the value is 1, that particular pixel in the color image is examined, but if the value is 0, it is skipped over.

2.4 Raw Moments

There are 10 different shape features for each view creating a total of 30 shape features. All of these features are based on raw image moments. The raw image moments can be described by the following equation

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \tag{2}$$

One key moment is M_{00} or area. The subscripts determine the power of the weight of each pixel, but in this case, since the subscripts are both 0, the weights will be 1. The above equation then simplifies to

$$M_{00} = \sum_x \sum_y I(x, y) \tag{3}$$

This can now be seen as simply summing over all pixels in the image. Taken one step further, if a threshold were to be applied to the image, then all pixels would either have a value of 0 or 1. Computing the value of M_{00} for this image becomes simply counting the white pixels. The same process can be looked other moments such as M_{01} or M_{11} moments. These raw image moments are used to describe the following shape features that are computed in this analysis: area, major axis, minor axis, and eccentricity.

2.5 Central Moments

The central moments are based on the raw moments and invariant to translation [5]. In addition, they are key to defining the Hu Moment Invariants used as image features. Using the raw moments described previously, the centroid of the 2D images can then be computed by the following equation.

$$(\bar{x}, \bar{y}) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \tag{4}$$

The centralized moments are computed in a similar manner, but instead of basing the weight on the distance from the (0,0) point, the centroid is used.

These are denoted by the symbol μ. Unlike the previous raw moments, the central moments are independent of translation due to being based on the centroid rather than a fixed point.

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y) \tag{5}$$

This results in μ_{01} and μ_{10} having a value of 0. The normalized central moments are computed by the following

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+(i+j)/2}} \tag{6}$$

Since these are normalized around μ_{00}, η_{00} is equal to 1.

Even though the central moments are invariant to translation, they can still be affected by scaling and rotation. A much more robust feature would be the Hu Moment Invariants. In addition to being invariant to translation like the normalized central moments, the Hu moments are invariant to scale and rotation transformations. Hu Moment Invariant I_7 in 10 is skew invariant [5].

2.6 Shape Features

The final result are the 10 unique shape features used for describing the images. Area of the object has already been described previously as the M_{00}. Major axis and minor axis are computed directly from the central moments by finding the eigenvalues of the following matrix.

$$A = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \tag{7}$$

$$\lambda_i = \frac{\mu_{20} + \mu_{02}}{2\mu_{00}} \pm \frac{\sqrt{(\mu_{20} - \mu_{02})^2 + 4 * \mu_{11}^2}}{2\mu_{00}} \tag{8}$$

and eccentricity from its definition

$$eccentricity = \sqrt{1 - \frac{\lambda_2}{\lambda_1}} \tag{9}$$

The final set of shape features are the Hu Moment Invariants [5] , which are defined by the normalized central moments.

$$
\begin{aligned}
I_1 = &\ \eta_{20} + \eta_{02} \\
I_2 = &\ (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 = &\ (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 = &\ (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 = &\ (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\ (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
I_6 = &\ (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 = &\ (3\eta_{21} - 3\eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - \\
&\ (\eta_{30} - 3\eta_{12})(\eta_{03} + \eta_{21})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}
\tag{10}
$$

2.7 Color Features

There are 6 features based directly on the color values developed by Navdeep [4]. These cover the mean Red value, mean Green value, mean Blue value, mean Hue, mean Saturation, and mean Value/Intensity. The image of the cap is captured as an RGB image, then converted to HSV to add three additional features.

For each image, only pixels corresponding to the maize kernel are used; background pixels are avoided with the use of a mask. The mask, being a binary image, will represent regions of interest in the original as described in section 2.3. If a pixel is part of the kernel in the image, it contains a value of 1, otherwise it is a value of 0. The important pixels can then be noticed by only using the color values when the same pixel in the mask has a value of 1.

For a single layer, the analysis starts by breaking its color values down into bins. In the case of the red, green, blue, saturation, and value layers, only 8 bins were used. These layers can have a maximum value of 255 creating bins of width 32, but since hue can have a max value of 360, this particular value was reduced to 9 bins. This has the effect of reducing the size of the color space, eliminating the total number of pixels with unique color combinations.

The final goal in examining the color layers is to compute the average value of pixels in each layer. In order to eliminate outliers, only the top 90% most common color combinations were used. Once the color values are binned, a list is made for each unique RGB or HSV color combinations. The number of pixels for each combination are counted then sorted. The desired

maximum number of pixels can then be picked out by selecting bins of unique colors, starting with the most common color, until the total number of pixels chosen added up at least the 90% limit. The final average color value was representative of which bin the average pixel would be attributed with.

2.8 Histogram Matching

The Histogram Matching features are based on metrics used to compare probability densities. A histogram for each color image was built using the OpenCV histogram routines from the hue and saturation layers. In addition, the same histogram was created of a reference image. This reference image chosen has been used for all the data. This image itself is not particularly important as long as it is used for all images.

The reference histogram will be denoted by H_0 and the histogram for each image will be denoted as H_1. Both of these are used together with four different metrics.

Pearson's Chi-Squared test is the first comparison used and is described by equation 11. The test can be used to measure goodness of fit [2].

$$d_{chi-squared}\left(H_0, H_1\right) = \sum_i \frac{\left(H_0(i) - H_1(i)\right)^2}{H_0(i)} \tag{11}$$

The second measure is the Bhattacharyya Distance [3]. This distance tests for similarity between the two histograms and follows a distribution similar to chi-squared, but the result is being used to compare other images and not as an assessment of one particular kernel. OpenCV is used to compute this distance, but use computes Hellinger Distance for this value as described by equation 12

$$d_{Hellinger}\left(H_0, H_1\right) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_0 \bar{H}_1 N^2}} \sum_i^N \sqrt{H_0(i) H_1(i)}} \tag{12}$$

The third matching feature is the sum of the intersection between H_0 and H_1 and is described by equation 13. The value is as straight forward as it seems. The two histograms are intersected, so each bin is set to the minimum of the two histograms then they are all summed together.

$$d_{Intersect} = \sum_{i}^{N} min\left(H_0(i), H_1(i)\right) \tag{13}$$

The Earth Mover's Distance is the final matching feature [10]. This describes the amount of work it takes to transform one distribution into another. Setting H_0 as a consumer and H_1 as the supplier, OpenCV is then used to implement the Earth Mover's Distance algorithm.

3 Data Collection

There are three steps involved with obtaining the image features mentioned in section 2. First, the images must be acquired using digital cameras. Next, each image must be thresholded in order to create the mask used in the feature extraction. Finally, this mask can be used to obtain the features. However, this mask is an important step because if it does not accurately portray the actual kernel, then the feature values will be off as well.

3.1 Equipment

The first step of the entire project would be the cameras. As seen in figure 5, three cameras were used to gather the data, each one is mounted orthogonal to the other. This allows images to be gathered from views along the x, y, and z-axis. The kernels are placed such that we gather data from the cap of the kernel, the germinal side, and the side of the kernel. Each of the cameras connect through a PC using Firewire. The cameras along the y and z-axis only produce grayscale images while the cap of the kernel in the x-axis is imaged in color.

In the original setup, the cameras were not mounted and free to move. Combined with clumsy tripods, bumping and moving the cameras was a common problem, so in this final iteration, each of the cameras are firmly mounted to adjustable platforms and locked in place. This allows for simple positioning of the cameras while preventing movement of the cameras that could be caused by a slight bump or vibration.

Slight adjustments in the kernel position can produce different results [4]. This has led to the development of the tool by Navdeep used specifically for placing the kernel as seen in figure 6. Each time a kernel is to be imaged, this

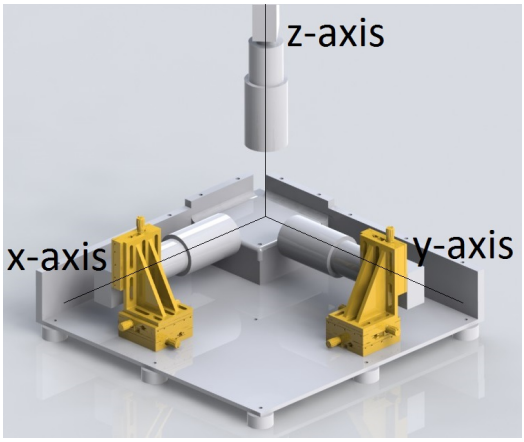


Figure 5: Model of the station used for gathering images.

tool is lined up with edges to ensure that everything is square. A small notch in the tool allows for placement of the kernel in the correction orientation and position.

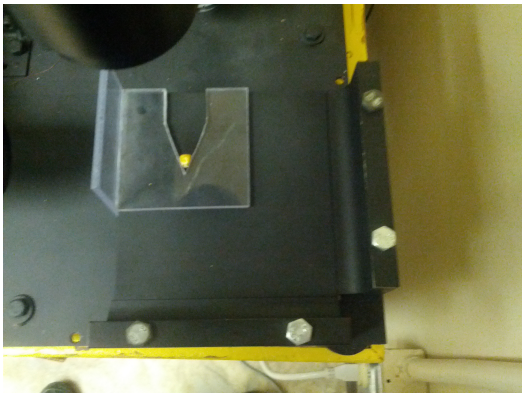


Figure 6: Tool built to position maize kernels.

The most difficult part of any experiment is to gather good data, so finding a solution that would allow the apparatus to be used equally for dark and light color kernels took several attempts to solve as described in section 4. The ideal computer vision setup uses a while object on a black background or vice versa. This allows for easy segmentation and feature extraction for the object being imaged.

Unfortunately, as the experiment proceed, a wider variety of kernels were

used. These had a wide color range which can be bright yellow to a dark green than the ones used in the original tests. This prevented the use of a solid background of either white or black. In addition, not only can ambient light affect some data such as color values, but it can also cast shadows, so the solution needed to be at least somewhat invariant to this effect. The final solution uses backlights behind the kernels using electroluminescent (EL) panels. There were several advantages to this approach.

The first advantage was that this can be used for both dark and light colored kernels. Instead of lighting up the kernel from a single direction, the kernel blocks out the light and creates a silhouette of the object. In addition, the panels are not extremely bright and allow us to also identify interior features of the kernels. While this is not currently taken advantage of this, it is definitely an area to expand on in the future.

The second advantage is the reduction/elimination of shadows, oil, and dirt. Previous efforts used a painted background with ambient or ring lights to eliminate external effects, but one of the major problems was when dirt/dust in the air or oil from a person's hands ended up on this background. This began to create spots and shadows in the images that could appear near the kernel, causing the segmentation to return an object that was larger than the actual kernel. By using a backlight, the light diffused around the oil, dirt, and shadows eliminating them from the segmentation results.

The major disadvantage comes from the color and direction of these panels. Due to the lights being behind the kernels, the kernel can still be influenced by ambient light. The effects of shadows are minimized, but there is still a noticeable difference in color analysis if the external lighting is adjusted. Finally, due to the nature of EL, it's difficult to obtain true white light panels. The ones used here have a slight blue tint regardless of being labeled as white light. While this does not affect the shape features, it does cause some problems with color analysis if trying to compare to kernels outside of this set.

3.2 Feature Extraction

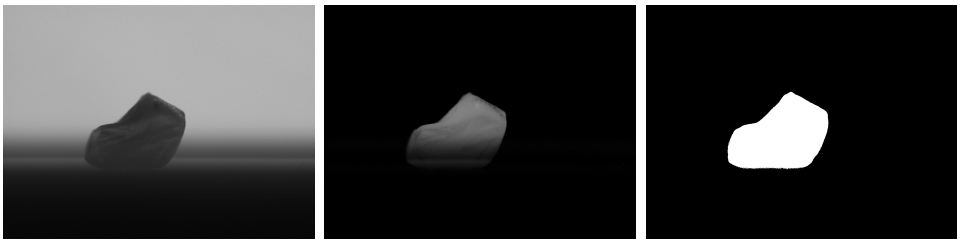
The second step to obtaining data is to perform segmentation and feature extraction on the gathered images. Segmentation is performed in three steps and relies on a mask image. The mask is a binary image that has a value of 1 when a pixel is important (part of the kernel) and 0 for all other background pixels. This allows the feature extraction implementation to only focus on

pixels that are part of the kernel and not the background.

In the final approach used, the background of the image is first removed. This is accomplished by keeping a set of empty background images with each set of kernels helps to eliminate the edge effect from the imaging setup. As seen in figure 7a, the kernel is placed on top of a platform, which creates an undesirable edge effect. In order to remove this, images of each view without kernels are captured beforehand and included with each set of kernels. Maintaining background images for each set is required due to the EL panels. Instead of suddenly failing, the panels slowly grow dimmer over time, so the background will, in effect, change between kernel sets. The effect will be more noticeable between 30-50 sets rather than just 1-2.

When creating the mask, this background image can be subtracted from the kernel image to give the result as seen in figure 7b. If a background image does not exist, variations in pixel values across a set of kernels can be used. Since a single dataset can contain 15 or more different kernels in a short time frame, the unnecessary background pixels can be considered constant throughout an entire dataset. This fact can be used to create a pseudo-background image and eliminate the unnecessary edge.

Next, a threshold is applied to the image. A static threshold level is used since the background of the image has been removed and can be considered almost little to no variation and close to zero. Since lighting is the same in every image, this also avoids boundary changes in the object. That is, if a variable threshold is used, then pixels with value X included in image A that should be included in image B might be zeroed due to a slightly different threshold value. The result of this threshold can be seen in figure 7c.



(a) An image of a kernel taken using the platform.

(b) Same image after subtracting out the empty background.

(c) A mask can then be created by thresholding the image.

Figure 7: Stages of segmenting an image

The final step is to ensure the only object in the binary image is the kernel itself. This is a necessary step in case there are extra artifacts in the image after applying the threshold. The kernel object is chosen simply by removing all blobs except the largest one. At this point, the shape and color analysis previously mentioned can be applied.

This process is followed for the front and side views of a kernel, but the top view is a much simpler process. Since the camera is looking directly at a seed with the EL panel covering the entire view, a simple threshold will segment out the kernel as seen in figure 8.

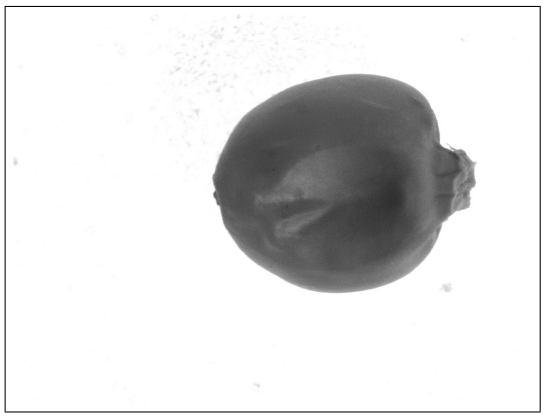


Figure 8: The pericap view of a kernel, a border around the image has been added to show the actual edges of the image.

4 Updating the Pipeline

The pipeline for this project is intended to be automatic. There should be no adjustment or assembly required to gather data for a large set of kernels. In the first version, the imaging station produced excellent results, but the selection of kernels used were limited. The ideal kernels had a solid, light color and a smooth, flat shape. Since its color is light, the kernel will create a sharp contrast against a black background. A smooth, flat kernel is ideal because it creates almost no shadows that affect image segmentation. The background, method of lighting, and segmentation algorithm can all have an effect on the final feature data.

4.1 Imaging Background

The ideal conditions for segmenting out a single object in an image involve either a dark object on a light background or a light object on a dark background. These two cases create a very sharp contrast between the edges of the object.

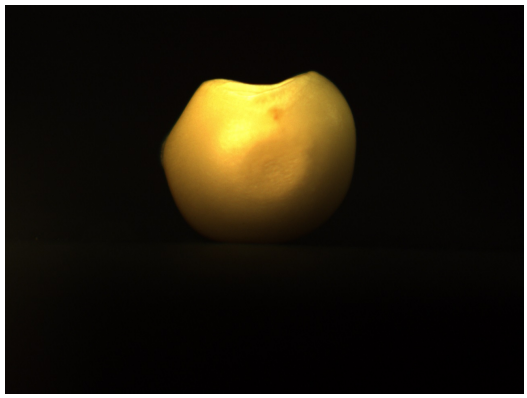
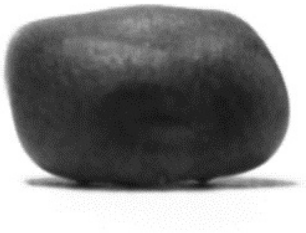


Figure 9: An example of a light colored object on a dark background.



(a) View of the pericap of a dark kernel.

(b) Top view, notice the tip.

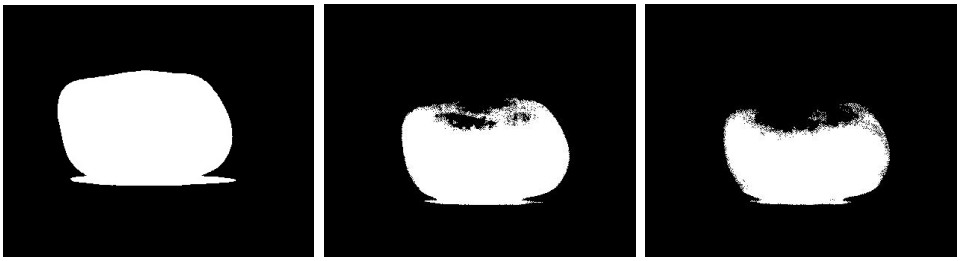
Figure 10: Example of dark objects on a light background.

Figure 9 and 10 show this sharp contrast, but there are several issues. First, with the light colored kernel, a significant portion of the bottom of the

kernel is lost as a result of the shape of the seed. This could be considered a shape feature in itself, but then many of the shape features described will not be reporting accurate values. This can be fixed by adjusting the lighting, but will be discussed in another section.

The alternative to a dark background is a white one, but this comes with problems of its own as well. The major one being shadows as seen in figure 10a. This is not something can simply be removed through thresholding. As seen in figure 11, the shadow can not be removed alone through the inverse threshold function. As the threshold limit is increased, portions of the kernel blob are also lost. The effect is even more significant with figure 10b. In this case, the shadow follows along the edge of the kernel, so in a binary image, this addition is not noticeable. The result is a larger area value.

Furthermore, despite the kernel being mostly a dark green color, the tip is still white. As seen in figure 12, the tip of the kernel is lost when thresholding an image before even trying to eliminate the shadow. Keep in mind, the final goal of the pipeline is to be able to classify the kernels based on the extracted features. If all dark kernels have a white tip, then overall, the dark kernels will have a shorter major axis in this view. This may lead to the feature being chosen when it is not an actual feature or it may not be considered when this should be a similar feature.



(a) t = 100 (b) t = 150 (c) t = 170

Figure 11: Shadows can not be removed through thresholding without destroying the desired blob.

Different backgrounds also introduce an issue of repeatability. In the case of the original imaging station, it was difficult to access the platform where the kernel would sit. All the extra movement of setting up different backgrounds for light/dark kernels increases the likelihood of bumping or moving any part of the setup. This results in additional calibration and even if everything is realigned, the experiment has already changed.

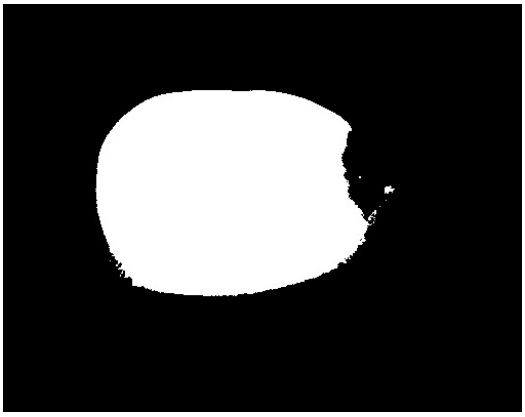


Figure 12: Thresholding dark kernel on a white background.

4.2 Lighting

Along with the background used when gathering images, the lighting effects around the setup can cause additional problems. Shadows are one such issue that has already been mentioned. Of course, this also means that adjusting the lighting of the imaging station could eliminate shadows. By applying direct lighting to the top view of the kernel and limiting the aperture of the camera, it was possible to obtain accurate shape features. It should be clear that despite the seed being absolutely saturated with light in figure 13, the shape is maintained even after thresholding. Detail of this part of the kernel is lost though. This part of the kernel is called the germ and is not used specifically in the analysis right now, but it is one of interest. Further down the line, additional analysis may prove useful, but it would not be possible in this approach.

Dirt and oil has a much more significant effect under this direct lighting. While the shape of the kernel is visible clear, the lighting also causes other reflective particles to appear as indistinguishable from the kernel. Normally, if these particles appear to the side of the kernel, there is no problem, but oil from a person's fingers can appear directly beneath the kernel. In this case, the oil will appear as an extension of the kernel, throwing the feature values off.

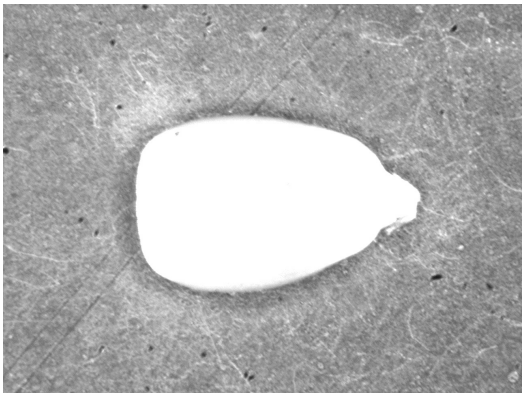


Figure 13: Flooding the kernel with light brings out the shape of the kernel due to reflection.

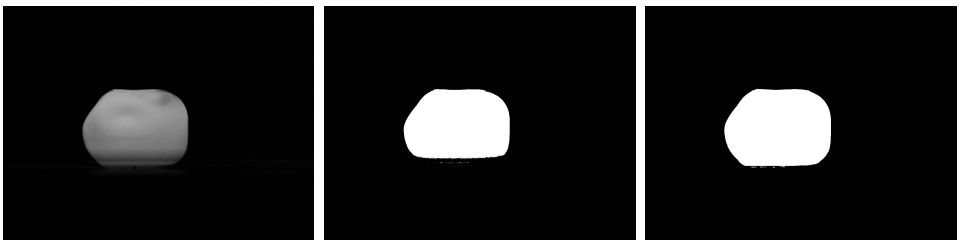
4.3 Thresholding

Despite the background and lighting causing problems, the most difficult challenge had to be applying an actual threshold level. Two types of thresholding were considered: manual and automatic.

In both cases, a single value is used as a limit for all pixels. If a pixel contains a value that is greater than the limit, that pixel is given a value of 1 (or 255). If a pixel contains a value that is less than or equal to the limit, then the pixel is given a value of 0. This creates a mask for the image and used for obtaining shape and color features.

Automatic thresholding was found using Otsu's Method [9]. This attempts to use pixel values in the image as a reference to find the threshold limit without assistance from the user. Unfortunately, this approach failed to produce useful results. In almost all the images, regardless of the kernel color, there exists many kernel pixels that have a value close to that of the background. Otsu's Method does not pick up the slight transition from the background to object. This, more often than not, results in a significant portion of the kernel being removed from the mask as seen in figure 14. In this particular example, using Otsu's method reduced the area of the kernel by 10%. Using an automatic method will also produce a different threshold limit for each image. This means a pixel in image A could be removed, but still appears in image B despite having the same value. This has the most significant effect near the edge of the kernel.

Since automatic thresholding has failed, the next step is to find a global



(a) subtracted image (b) Otsu Threshold (c) Fixed threshold

Figure 14: A typical result from using Otsu method.

threshold level that will work for all images that will be used. This allows for consistency across all images, but then the lighting and background must not change from one set of kernels to the next, regardless of shape or color. Since the goal of applying the threshold is to create a mask, then background subtraction is a strong candidate for reducing an image down to the point of being able to apply a fixed threshold limit.

4.4 Backlighting

The final solution was to apply backlighting to the kernels. This creates light from underneath/behind the kernel, creating an effect that works for both dark and light kernels. In an image, this background should appear white, but unlike a white background, shadows are eliminated and white portions of a kernel will not be lost. Any shadows cast on to the background will be diffused by the light produced. Previously, using a white background, it was easy to lose light kernels on a white background, but with a backlight, the kernel is actually blocking the light, so a light kernel will appear more distinct, making it an easy object to threshold out.

With the current setup, background subtraction is required for this approach. The region behind a kernel is lit up, but the platform the kernel rests upon is not, creating an edge effect as seen in figure 7a. Before images are gathered on a set (about 16-32 kernels), an image of only the background from each of the three cameras is recorded. Then, during the analysis, these images are subtracted from the images of kernels. Since the background is constant over a short period of time, those pixels have a value near 0 while the kernel pixels will have a positive value.

There exists two major problems with the EL panels used for backlight-

ing. First, it seems to be impossible to obtain white light panels. The "white light" panels used have a distinct blue tint, unfortunately. This tends to be reflected by the kernels causing the blue channel to be higher than previously imaged kernels. The move to these panels was performed between experiments, so while a comparison to previous images would not be appropriate due to bias, it will not affect future work.

The second problem is one of lifetime. Unlike a lot most electronic equipment, the EL panels have a theoretical infinte lifetime; they will never simply fail or stop working. The trade off is a gradual dimming of the light produced and after roughly 3000 hours, the light is too dim to produce comparable results. The trade off for a short lifetime is a low cost and easily replaced. Across a single set of kernels, the light intensity can be considered constant, but after 30 or 40 sets, this may not longer be true. As a result, a single background image for subtraction cannot be maintained. This has led to keeping a new set of background images along with each kernel set. In the case a background image does not exist, using one from a recent set or creating one using pixel intensity variations can be used.

5 Evaluation Techniques

The features measured here will be used along with many other measurements in order to identify unique characteristics of different kernel composition and genetics. One method to identify unique features is through machine learning algorithms. Classification can show us if features can be used to correctly identify different kernel types or mutants. Feature selection methods are excellent for reducing a large feature set and will work to pick out the important features.

5.1 Classification

Classification focuses on trying to identify a classes by a list of features given. These methods can then build a model that will be used to classify future sets that do not have a class label. In order to build the model, there exists supervised and unsupervised algorithms. In each case, a set is a list of examples, one per kernel, each with a list of features and a label describing how that kernel should be classified. In unsupervised learning, the labels are absent and the algorithm must determine those on its own.

The measure of success for these methods is typically accuracy, which is the percent of examples correctly labeled. In order to provide an accuracy value that's less biased, cross validation is used. This involves splitting a dataset into different N groups. One group is removed, while all of the others are used to build a model for classification. Using the model, the removed group is then classified and the accuracy is reported. This is repeated N times, one for each group. The final accuracy reported is the average over all iterations.

5.2 Feature Selection

The goal if feature extraction is to obtain a subset of features that will be the most optimal. This is different from classification which focuses on accuracy of labeling a test data set [7]. There are several methods for solving the problem of feature extraction.

A Filter Based Selection is one typical approach with the Relief Algorithm [6] being such an example. This method is a good option for picking out individual independent features that are relevant to the class, but tends to miss features that can be highly predictive when combined with other features. Redundant features are usually chosen as well rather than just an optimal subset.

A common feature selection approach is Wrapper Based Selection. Instead of finding strong, individual features, the selection process is treated as a search problem. The most widely used methods of wrapper selection are forward selection and backward elimination. In this case, forward selection was used, which tends to be the faster approach. Backward elimination is more likely to selected features that work well together [7].

Given a training set with N features, forward selection method begins by trying to classify the data using an empty feature set. Since the class label is given with a training set, the algorithm will stop at this point if only one label was given, otherwise it will continue. The algorithm continues by choosing the very first feature of the feature set and then creating a classification model with a specified algorithm (Bayes Net, SVM, etc.). The model is evaluated using the accuracy metric and then repeated for each feature in the feature set. Since there are N features, this creates N models, of which the best is chosen based on accuracy.

The chosen feature f_3 is then ranked as the best feature. Using this newly selected feature as its base set, the algorithm repeats its previous step,

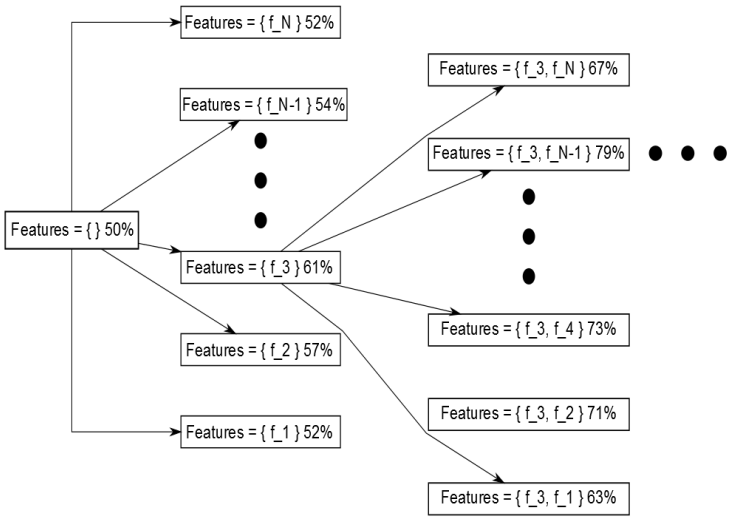


Figure 15: In each step of the forward selection method, multiple models using different features are created.

but this time. Instead of classifying with only one feature at a time, f_3 is included in each model as well. This time, N-1 models are created, the best feature pair is selected, and everything is repeated. The algorithm continues until classification accuracy reaches 100% or it reaches the end of the feature set. These steps can be seen in figure 15. Backward elimination is similar, except it starts by classifying with all features and removes features instead of adding them.

The forward selection algorithm focuses on features that have strong prediction power both as independent features and in combination with other features. This method can pick out redundant features, but it is only important that the features are not strongly dependent on each other.

5.3 WEKA

All of the analysis here is completed using the WEKA libraries (http://www.cs.waikato.ac.nz/ml/weka/). These are maintained by the University of Waikato. The API for Java allows for both classification and feature selection algorithms. In addition, to the Java API, there is a GUI that allows for a simpler use of the libraries.

SVM	Bayes Net	Neural Network	Naive Bayes
I_2 (front)	Area (side)	I_1 (front)	Area (side)
I_2 (side)	Major Axis (side)	I_3 (side)	Major Axis (side)
Major Axis (side)	I_2 (top)	Major Axis (side)	Minor Axis (side)
Minor Axis (side)	I_5 (top)	Eccentricity (side)	Eccentricity (top)
EMD	Red Value	Hue	Red Value
Red Value	Blue Value	Saturation	Hue
Green Value	Hue	Value/Intensity	Saturation
Blue Value	Value/Intensity	Green Value (strd)	
Hue			
Area (side, strd)			
Major Axis (front, norm)			
Blue Value (norm)			

Table 1: Selected features from 10 lines of kernels.

6 Results

The following results were obtained by classifying 10 different kernel lines. In addition to the 40 features discussed in section 2, the same features normalized and the same features standardized are included for a total of 120 features.

6.1 Feature Selection

Five different algorithms were used along with Greedy First Search to through forward feature selection. The features selected with four of the five methods are listed in table 1. The random forest method was used, but it was unable to produce any significant results, so it is not included in the results. Where applicable, labels denoting the view, normalization, and/or standardization are included.

Several of the features are repeated across different classifiers. The primary goal of the feature selection is to reduce a feature set, which this algorithm has accomplished.

It should be pointed out that several of these features such as major and minor axis for the side are repeated across classifiers. This could indicate that certain features are more unique and may have a stronger relation to other characteristics of the kernels.

Classified using...	Neural Network	Bayes Net	Random Forest	Naive Bayes	SVM
Naive Bayes	77.7%	80.41%	20.95%	85.81%	82.43%
Bayes Net	75.68%	79.05%	6.76%	80.41%	76.35%
Random Forest	83.11%	79.73%	18.92%	81.76%	79.73%
SVM	72.3%	77.7%	11.49%	87.16%	86.49%
Neural Network	83.11%	81.76%	18.92%	87.16%	88.51%

Table 2: Classification accuracies of the previous reduced sets using different classifiers.

6.2 Classification

As mentioned, each of these reduced sets are now classified with each of the different algorithms listed before with accuracy as the unit of comparison. The results are listed in Table 2.

Overall, most of the sets can be classified with nearly 80% accuracy. The random forest set shows poor results across the board. This is a result of the feature selection method as it only selected a total of two features before it was able to improve its accuracy.

A further breakdown can be found by looking at the reduced set and classifier that produced the highest accuracy. Each group only has 14-15 kernels, so even if just one kernel is misclassified, this can be a 6-7% reduction in accuracy.

6.3 Errors

Occasionally, errors in the images arise and cause problems. Typically, the resulting feature values will be extreme outliers, so finding these errors is relatively simply by viewing the statistical distribution of values. Sometimes, more difficult problems arise. If a seed is too large to fit within the view of the camera, part of it will be cut off by the edge of a camera. Since the images are gathered manually, this and similar problems are avoided by checking each individual image before it is saved, but not everything is caught. Automatic checking systems are developed when a new problem is found, but this can only be accomplished for mistakes that have been discovered.

7 Conclusion

Previously, a method had been developed to gather and identify features related to maize kernel characteristics. This process was still in its experimental stages when several problems involving segmentation had been discovered. Through the construction of a more permanent station and modifications to lighting, a more robust system has been developed. In addition to still providing useable results, the process can handle a greater variety of maize kernels.

There are two improvements that I would like to see happen with this design. First, an more automatic method of imaging the kernels should be explored. Currently, each kernel is placed by hand in the appropriate position in front of the three cameras. In terms of efficiency, this is a slow process. Instead, if a approach can be developed that only requires the user to load up the imaging station with a tray of kernels, this could cause a significant increase to the imaging process.

The second goes back identifying the kernel within an image. Even though this system is stable and works for a wide variety of kernels, it may be possible that a kernel with a new shape/color appears that will not comply with what has been designed. A new approach would be to develop features that actually define when a pixel is part of the kernel and when a pixel is part of the background. These features could then be used along with machine learning to identify not only current kernels, but possibly new ones that may cause problems in the future.

References

[1] Irfan S. Ahmad, John F. Reid, Marvin R. Paulsen, and James B. Sinclair. Color classifier for symptomatic soybean seeds using image processing. *Scientific Societies*, 83(4):320–327, 1999.

[2] Steve Baker and Robert D. Cousins. Clarification of the use of chi-square and likelihood functions in fits to histograms. *Nuclear Instruments and Methods in Physics Research*, 221(2):437 – 442, 1984.

[3] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhya: The Indian Journal of Statistics (1933-1960)*, 7(4):pp. 401–406, 1946.

[4] Navdeep Dahiya. Automatic classification of different genetic varieties of corn kernels based on phenotypic information. diploma thesis, University of Wisconsin - Madison, 2010.

[5] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[6] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the National Conference on Artificial Intelligence*, pages 129–129. John Wiley & Sons Ltd, 1992.

[7] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1???2):273 – 324, 1997. ¡ce:title¿Relevance¡/ce:title¿.

[8] X. Liu, Y. Wang, Q. Su, , and Z. Wang. IFIP International Federation for Information Processing. In D. Li and Z. Chunjiang, editors, *Computer and Computing Technologies in Agriculture II, Volume 3*, volume 295, pages 2207–2216. Boston: Springer, 2009.

[9] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Systems, Man, and Cybernetics Society*, 9:62–66, 1979.

[10] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66, 1998.

[11] David M Woebbecke, George E Meyer, Kenneth Von Bargen, and David A Mortensen. Plant species identification, size, and enumeration using machine vision techniques on near-binary images. In *Applications in Optical Science and Engineering*, pages 208–219. International Society for Optics and Photonics, 1993.

[12] Liu Zhao-yan, Cheng Fang, Ying Yi-bin, and Rao Xiu-qin. Identification of rice seed varieties using neural network. *Journal of Zhejiang University SCIENCE*, 6(11):1095–1100, 2005.