

# A machine vision system for high speed sorting of small spots on grains

Tom Pearson · Dan Moore · Jim Pearson

Received: 30 September 2011 / Accepted: 11 October 2012 / Published online: 27 October 2012  
© Springer Science+Business Media New York (Outside the USA) 2012

**Abstract** A sorting system was developed to detect and remove individual grain kernels with small localized blemishes or defects. The system uses a color VGA sensor to capture images of the kernels at high speed as the grain drops off an inclined chute. The image data are directly input into a field-programmable gate array that performs image processing and classification in real time. Spot detection is accomplished by a combination of color information and a simple, nonlinear spatial filter that detects small dips in pixel intensity along an image line. Color information is combined with spatial filtering to achieve a high level of accuracy. Testing was performed on popcorn with blue-eye damage, which is characterized by a small blue blemish on the germ. A two-camera system was developed to inspect the opposite sides of each kernel as they slide off the end of a chute. The chute was designed such that the kernels slide down the chute without tumbling, increasing the probability that a spot will be in the field of view of one of the cameras. The system's accuracy is 89 % identification of blue-eye damaged kernels with a 6 % false positive rate. The throughput is approximately 180 kernels per second, or 100 kg/h.

**Keywords** FPGA · Camera · Color · Imaging

## Introduction

The detection of small localized spots (or blemishes) on agricultural products using machine vision has proven to be feasible [1–3]. However, this approach becomes complicated when one desires real-time detection and handling at an economically feasible cost. Commercial color sorters are widely used to separate grains, nuts, and other products by color, but they do not have the spatial resolution or image processing capability to detect small spots on kernels [4]. An imaging- and hardware-based processing system was developed that was integrated into a sorting system for the accurate separation of grains by color and surface texture [5]. However, this system could not detect single spots or blemishes on kernels. In an earlier work [4], it was attempted to address the detection of spots on popcorn caused by blue-eye fungal infestation. However, the accuracy of detecting blue-eye infested kernels was only 74 % with 9 % false positives, and the throughput was only approximately 35 kg/h. Feedback from the popcorn industry indicated that the detection accuracy for blue-eye infested kernels must be approximately 90 % with approximately 5 % false positives, and the throughput must be higher before a sorting method would be useful enough to be implemented.

Some of the problems in the initial effort to detect blue-eye infested kernels were caused by dark areas on the edges of the kernels and the random orientation of the kernels in that system [4]. Moreover, the study did not use color information to help differentiate blue-eye blemishes from other darker regions or spots on the kernels because the processing power of the hardware was insufficient.

The objective of this study was to improve the throughput and accuracy of a sorter used to detect blue-eye damaged popcorn kernels through improved lighting and

---

T. Pearson (✉)  
USDA-ARS-NPA-CGAHR, Manhattan, KS, USA  
e-mail: thomas.pearson@ars.usda.gov

D. Moore  
National Mfg., Lincoln, NE, USA

J. Pearson  
Short Dog Electronics, Corvallis, OR, USA

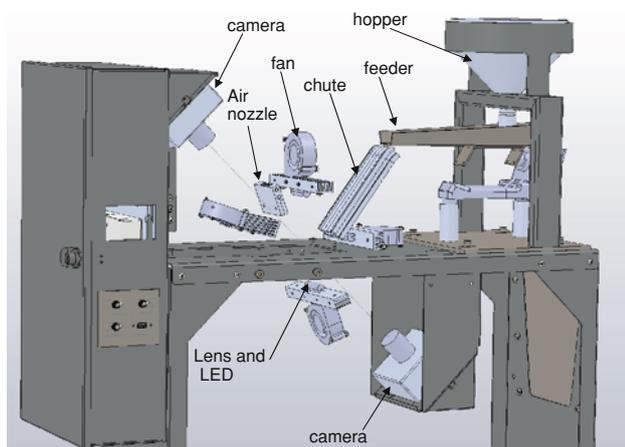
feeding systems, as well as additional processing enabled by an field-programmable gate array (FPGA) with more logic elements and memory to accommodate more mathematical operations on the image than previously possible. The goal was to produce a machine with the accuracy and throughput required for industry implementation.

## Materials and methods

A sorting system with three parallel chutes and two cameras was constructed as shown in Fig. 1. In this system, bulk kernels were placed into a hopper, and a vibratory feeder moves the kernels in a single layer onto a chute that is inclined at 45°. Images of the opposing sides of the kernels are acquired about 10 mm after the kernels slide off the chute. A two-camera system was chosen instead of the three-camera system as used in Pearson [4]. This allowed parallel chutes to be used to increase throughput, and made it possible to orient the kernels such that their germ faced one of the two cameras. This system is similar to that originally developed [5] except for four important differences: the feeding system was designed to prevent kernel tumbling and orient the kernels so the germ faced one of the cameras, the camera circuit board facilitates more image processing, the use of two cameras on opposing sides of the kernels, and the image algorithm used to detect small spots on the germ of the popcorn caused by blue-eye fungal infestation. Each of these differences is discussed in detail below.

### Chute design

It was observed that popcorn kernels, while on the vibratory feeder, almost always orient themselves with their germ

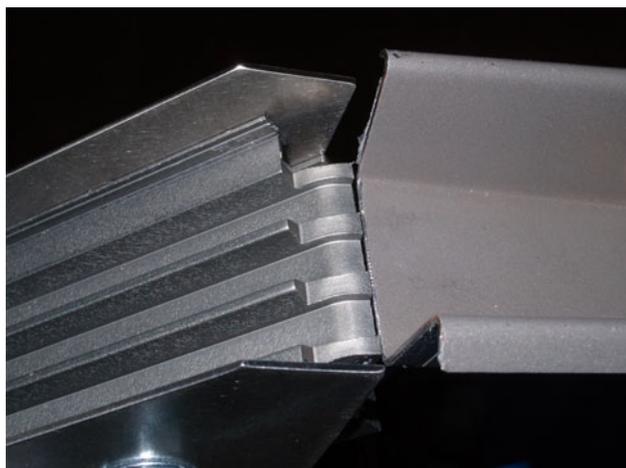


**Fig. 1** Sorting system for the detection of spots. Note that two cameras are used such that opposite sides of the kernels are imaged simultaneously

facing up or down. This is because the germ or endosperm sides of the kernel tend to be flat, and this orientation puts the kernel into the lowest center of gravity. However, sorting cannot be performed at the end of the vibrating feeder, as the kernels are too close to each other, which prohibits the efficient diversion of defective kernels. An inclined chute is required to accelerate and separate the kernels. An aluminum extrusion was designed and fabricated with flat bottom grooves of sufficient width such that the popcorn kernels could slide down the chute with their germ facing up or down, thereby matching their orientation on the vibrating feeder. The depth of the grooves was deep enough that kernels could slide on their germ or endosperm side yet not so deep that the kernels could tumble. Since the grooves of the chute were shallower than the length of the average kernel, a plastic cover, cut from transparency plotter film (17702T, Hewlett Packard, Palo Alto, CA) was placed over the chute and prevented most of the kernels from flipping completely over. The width and depth of the grooves in the chute were determined using the width and thickness of 300 randomly selected popcorn kernels from two different growing seasons and five different growing locations. Using these measurements, the groove width was set at 7.6 mm, which corresponded to the average width plus three standard deviations of the kernel width. The groove depth was set at 6 mm, which corresponded to the average kernel thickness plus three standard deviations. The average length of the popcorn kernels was 8.1 mm with a standard deviation of 0.9 mm. Therefore, it would be impossible for at least 95 % of kernels to tumble down the chute with this design. To further ensure that kernels remained in their germ up/down orientation, the top end of the chute was curved such that the bottom of the chute grooves follow a 25.4 mm radius, then it was cut so that the surface at the bottom of the chute grooves would be tangent to the end of the vibratory feeder as shown in Fig. 2. A normal practice for a vibratory feeder-chute transition would be to drop kernels from the feeder onto an inclined chute. However, this practice can initiate tumbling and prevent the germ from facing one of the cameras. Finally, the aluminum chute was polished and hard anodized to reduce friction and further reduce tumbling.

### Sorter design with two cameras

As indicated in Fig. 1, the two-camera arrangement required the placement of an air nozzle to divert blue-eye infested kernels in a slightly unusual direction. In most single channel sorting machines, the kernels were diverted roughly perpendicularly to their travel path. However, this was not possible with the two-camera system because it would cause the diverted kernels to pass over the lower camera's field of view. Therefore, the air valve was placed



**Fig. 2** Feeder-chute transition. Note that, normally, a plastic cover would be placed over the chute to prevent the kernels from tumbling. This cover was removed so that the chute-feeder transition was more visible

in an almost vertical orientation above the kernel flow stream and diverted the kernels straight down vertically, while accepted kernels continued on a path approximately 45° from vertical.

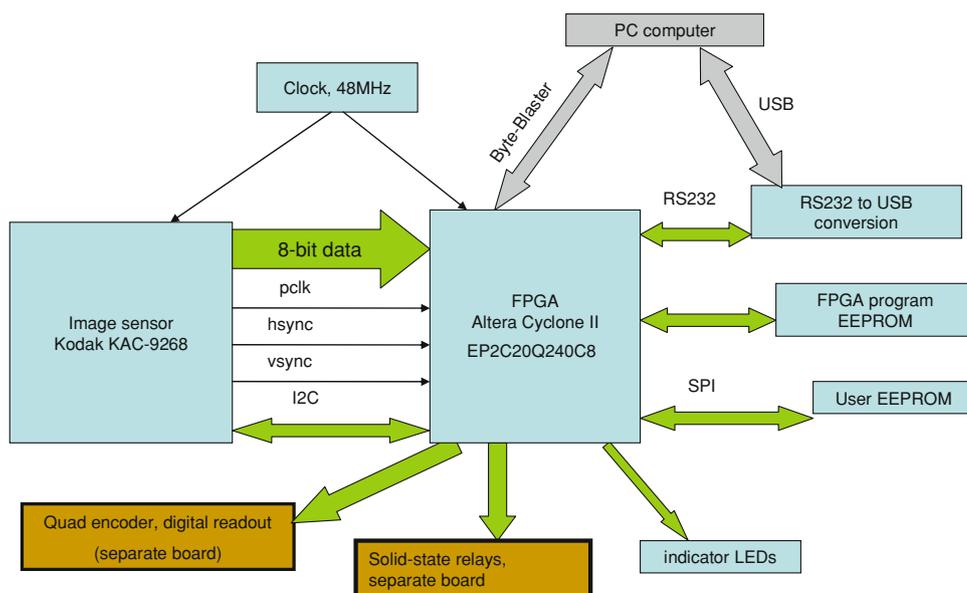
Lighting for this sorting system was provided by four white light emitting diode (LED) arrays placed around the kernel stream. One light array was placed under the chute. The LED arrays were constructed from a custom made aluminum clad printed circuit board (PCB-POOL IMS (Aluminum), PCB-Pool Beta Layout, San Jose, CA) and consisted of nine 5 W LEDs (LZ1-00CW05, LEDEngin, Santa Clara, CA) in a linear pattern spaced 5.5 mm apart. The aluminum clad PCB provided excellent heat conduction through the board to help dissipate heat generated

by the closely spaced LEDs. An aluminum heat sink (MM23600, M&M Metals, Carrollton, Texas) with a fan (#BG0702-B055-000, NMB Technologies Corporation, Chatsworth, CA) was subsequently mounted to the PCB on the side opposite the LEDs. The LEDs had a cool white color temperature of 5,500 K, which provided good color rendition, especially for enhancing blue (B) color. A linear lens was placed over the LED array (C-002, DBM Reflex Lighting Solutions, Laval, Québec) to diffuse the light from each LED and focus it onto the point of imaging.

Camera circuit board

A circuit board with an image sensor (KAC-9628, Eastman Kodak Co., Rochester, NY) directly linked to a FPGA (EP2C20Q240C8, Altera, San Jose, CA) was designed to perform the image analysis in real time. A schematic of the board is shown in Fig. 3. The FPGA programs were written in the Verilog HDL language and compiled in Quartus II, which was supplied by the FPGA manufacturer. Compiled programs were transferred from the PC to the FPGA using a special communications cable (ByteBlaster, Altera, San Jose, CA). The image sensor was mounted on the opposite side of the PCB board from the FPGA to minimize the distance the data would need to travel and reduce the potential of data corruption from noise. The support electronics for the image sensor and the FPGA are those specified by the manufacturer and were described in detail in [4]. The FPGA used in this design has 18,752 logic elements and 234 kbits of memory, as opposed to the FPGA used in [4], which has only 4,608 logic elements and 117 kbits of memory. The additional logic elements made it possible to perform more image processing operations,

**Fig. 3** Block diagram of the camera system



and the expanded memory enabled the capturing of higher resolution images and the use of buffers for filtering and color interpolation.

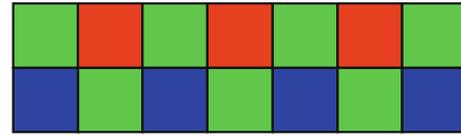
As in [4], the image sensor operates in near line-scan mode, which means that each frame in the image is composed of only two lines and successive frames are combined to form a two-dimensional image. When a kernel enters the field of view, the pixel intensity increases and triggers the FPGA to commence image capture. Next, the image is loaded into its on-chip memory. Each image is stored in a raw (un-interpolated) format with a size of  $124 \times 124$  pixels. Spatial resolution was approximately 0.06 mm/pixel. Each image, at 8 bits/pixel, requires 123 kbits. The memory structure is 8 bit words and requires the memory to be allocated in  $2^n$  bytes; therefore,  $2^{14}$  bytes were allocated, leaving 1,008 bytes for other image data, such as image features, that the FPGA extracts from the images in real time. If desired, the user can connect a PC to the FPGA through the USB interface and transfer the images and extra data from the FPGA memory to a PC for each kernel. As discussed below, the image sensor has 640 pixels per line with a field of view spread over three parallel channels that the grain slides down. Three different regions of interest are centered over the three channels; each has a width of 124 pixels.

The FPGA and the image sensor are timed with the same 48 MHz clock source. The image sensor divides this clock frequency by four to obtain a pixel clock frequency of 12 MHz. Because the FPGA clock rate is faster than the pixel clock rate, it is possible to perform mathematical operations on the image while pixels are being transferred from the sensor to the FPGA.

The camera board also has EEPROM memory that the user can transfer data to through the USB connection. This data is read by the FPGA during startup and can contain parameters for kernel classification (such as discriminate function coefficients) or parameters to adjust the image sensor at startup (such as the pixel clock rate). In addition, the camera board has an input for a switch, a quad encoder, and a pulse width modulated output to a digital LCD display. The FPGA was programmed to decode the encoder signal and adjust the PWM signal proportionally to the encoder number. This number can be used as a user-adjustable rejection threshold for classification and sorting. Finally, the camera board has buffered outputs to solid-state relay triggers that fire air solenoid valves to divert kernels as discussed in [4].

The image sensor (KAC-9628, Eastman Kodak Co., Rochester, NY) has  $640 \times 480$  pixels and uses a color Bayer filter to sense red (R), green (G), or B light on different pixels as shown in Fig. 4.

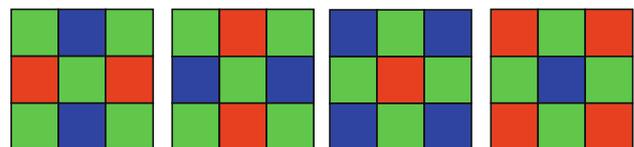
The pixel data was digitized on the image sensor chip and transferred to the FPGA without interpolating the colors.



**Fig. 4** Bayer filter pattern on the image sensor used. One line is comprised of *green* and *red* pixels, and the next line is *blue* and *green* pixels (Color figure online)

Normally, color interpolation is performed after an entire image is acquired, but this approach is not possible for real-time sorting because all of the image processing must be performed as the image is being acquired so that a decision can be made almost immediately after the kernel passes out of the field of view. This design allows the FPGA to be ready to process another kernel almost immediately. To interpolate the colors with minimal delay between kernels, two large first-in-first-out buffers were created using the FPGAs memory. The buffer lengths are two lines (1,280 pixels) and one image line (640 pixels) each. As each pixel is received from the image sensor, it goes into the buffers, and the pixel data from exactly two lines above the current pixel location is outputted by the buffers. The output of these buffers and the current pixel data are subsequently input into three revolving memory variables, forming a  $3 \times 3$  pixel array. Color interpolation is the process of using colors from adjacent pixels to compute a triplet of R, G, and B pixel values for all of the pixels in the image. This process can involve a large number of pixels and numerical operations [6]. To minimize the amount of computations, the interpolation algorithm used in this study simply averages the appropriate adjacent pixels in a  $3 \times 3$  pixel array. With a Bayer color pixel pattern, there are four color arrangement scenarios for pixels in a  $3 \times 3$  pixel array, as shown in Fig. 5.

The center pixel of the  $3 \times 3$  array is simply used as one of the colors for the RGB triplet. In the cases where a G pixel is centered, the average of the two adjacent R pixels and the average of the two adjacent B pixels are used to complete the RGB triplet. The averages are computed by summing the pixel values and subsequently right shifting 1 bit to obtain an average. It is known that this operation truncates the result instead of rounding it. If the center pixel is R or B, the average of the four adjacent pixels comprising the other two colors is used to complete the triplet. In this case, the sum of the 4 pixels is right shifted 2 bits to obtain the average color.



**Fig. 5** Possible *color* arrangements in a  $3 \times 3$  pixel array from raw *color* image data (Color figure online)

The summing and shifting color interpolation operations on the  $3 \times 3$  pixel arrays are efficient, can be performed between pixels being clocked into the FPGA, and do not slow any of the other image computations. The color of each pixel in the  $3 \times 3$  array is tracked by counting pixels from the start of each line and by their relation to the vertical sync and horizontal sync pulses from the image sensor. Image lines containing R and G pixels are preceded by both vertical and horizontal sync pulses, whereas the green–blue lines are preceded by only a horizontal sync pulse. The only delay comes from the two line image buffer and is only 0.08 ms. The kernels travel off the chute at a speed of approximately 2 m/s. Therefore, after the entire image is acquired, the kernel has only traveled 0.16 mm before all the processing is complete. Most of the gaps between kernels as they slide off the chute are greater than 1 mm. If kernels happen to be touching, the first 0.16 mm of the trailing kernel's image is truncated; however, this was considered to be an acceptable loss of data that probably would not cause the kernel to be misclassified.

#### Image processing and classification algorithm

Samples of popcorn from two growing years and from five different storage bins were collected so that a reasonable range of kernel color and kernel morphology could be studied. Samples were drawn from bins known to have high levels ( $\sim 5\%$ ) of blue-eye damage. Each sample was approximately 1 kg and was divided using a Boerner divider (#34, Seedbuco Co., Des Plaines, IL). One of the fractions was kept for sorter testing, whereas the other samples were hand inspected, and 100 undamaged and 100 damaged kernels were pulled from each sample for a total of 2,000 blue-eye and 2,000 undamaged kernels. Un-interpolated raw images of each kernel were then collected in the two-camera sorter prototype by feeding the kernels from the vibratory feeder in the same way as if they were being sorted. The images were color interpolated off-line using the same technique as discussed previously and were saved in BMP format for off-line analysis to develop an image processing algorithm for detecting the damaged kernels.

The first off-line analysis performed was to investigate use of color to distinguish blue-eye damaged regions from the other parts of the kernel. Each of the saved kernel images was opened in Adobe Photoshop, and RGB values were recorded for 10 pixels in the blue-eye region (if present) and 20 pixels in other areas of the kernel with similar R values to the blue-eye region. These other regions were usually located near the edge of the kernel or in areas with shadows, due to undulations on the kernel's surface. On the undamaged kernels, RGB values of edges and shadow areas were recorded from 30 pixels scattered around the kernels.

The RGB values for all the pixels were converted to hue, saturation, and value (hsv) and CIE Lab color values. In addition, the difference between two of the three color values for each pixel (R-G, R-B, and G-B) was computed. The averages and standard deviations of these data are shown in Table 1. All of the computed color values and RGB values were saved in a spreadsheet along with their association with blue-eye damaged regions, shadows, or other portions of the kernels. Stepwise discriminant analysis software (Number Cruncher Statistical Systems, Kaysville, UT) was used to select the single best color value for distinguishing pixels in blue-eye regions from other regions on the kernel. The stepwise procedure selected saturation as the best feature for distinguishing blue-eye pixels from the other regions of the kernel and selected R-B for distinguishing blue-eye regions from shadows. Figure 6 shows a color image of a blue-eye damaged kernel and an image displaying only the saturation component of the hsv image.

**Table 1** Average and standard deviation (in parenthesis) for sample pixels from different popcorn regions

Color feature	Blue-eye	Endosperm	Shadows
R	122.4 (17.7)	118.6 (43.8)	136.7 (10.6)
G	100.2 (16.1)	86.6 (20.3)	109.9 (9.4)
B	65.1 (10.7)	51.5 (14.8)	69.6 (7.4)
Hue	24.4 (2.4)	21.9 (3.0)	24.1 (1.7)
s	0.47 (0.03)	0.56 (0.07)	0.49 (0.03)
L	43.7 (6.5)	39.4 (10.5)	48.1 (3.8)
a	3.9 (2.0)	8.4 (11.8)	5.1 (1.8)
b	23.2 (3.4)	25.5 (8.3)	26.4 (2.5)
R-B	57.4 (8.6)	67.1 (37.1)	67.1 (6.5)
G-B	35.1 (6.6)	35.1 (7.3)	40.3 (4.8)
R-G	22.2 (4.2)	32.0 (35.2)	26.7 (3.9)



**Fig. 6** Color image of a popcorn kernel with blue-eye damage (left) and the saturation image (right) (Color figure online)

Next, the saturation image was processed to extract spots indicating blue-eye damage. Note that the germ and tip cap regions are high in saturation, whereas the blue-eye region was relatively low. Portions close to the edge of the kernel can have similar saturation levels to the blue-eye region. A simple nonlinear filter was developed to distinguish blue-eye regions from other regions having similar saturation levels that can be executed in real time.

The filter simply compares the saturation levels of 5 pixels on the same line, 1 pixel is centered between the other four, two are spaced only 1 pixel from the center, and the other two are separated from the center pixel by a larger gap of pixels. The pixels location in this filter can be described as  $\{x - \text{gap}, x - 1, x, x + 1, x + \text{gap}\}$  where  $x$  is the pixel location along the image line and  $\text{gap}$  is distance in pixels. If the saturation level of least two of the pixels at  $x - 1, x$ , or  $x + 1$  are less than both of the pixels at  $x - \text{gap}$  and  $x + \text{gap}$  by more than a preset offset value, then the center pixel is considered to be part of a blue-eye region. This effectively identifies a “dip” in the saturation levels across the pixels that is at least 2 pixels wide and less than the offset value from the surrounding pixels. Darker regions along the edge of the kernel are not counted as blue-eye as they do not have such a dip associated with them. This simple filter was applied to all the images saved with gaps ranging from 5 to 15 pixels in single pixel increments and offsets ranging from 10 to 30 levels of saturation in increments of four. For each combination of gap and offset, the pixels that were considered to be part of a blue-eye kernel were counted. Afterwards, stepwise discriminant analysis was used to select the best combination of gap and offset for distinguishing blue-eye damaged kernels from undamaged kernels. In addition to the counts, the averages and standard deviations of the image pixels classified as blue-eye from the R-B image were saved to help reduce false positive errors caused by shadows on undamaged kernels. It was observed that shadow regions of undamaged kernels had slightly higher levels in the R-B image and a much lower variance of R-B pixel values than did the blue-eye regions.

### FPGA implementation

The FPGA was programmed to compute saturation values from the RGB values through a modification of the standard conversion of the procedure [7] as outlined below.

$$\begin{aligned} \max &= \text{maximum}(R, G, B) \\ \min &= \text{minimum}(R, G, B) \\ s &= (\max - \min)/\max \end{aligned}$$

where  $R, G$ , and  $B$  are the interpolated red, green, and blue values of the pixel, and  $s$  is the saturation. However, for all pixels corresponding to the kernel in the test set,  $R$  is the

maximum and  $B$  is the minimum, which simplifies the saturation computation as shown in Eq. 1.

$$s = (R - B)/R \quad (1)$$

All of the  $R-B$  values were found to be less than 127 and so were stored in a 7 bit variable. The division was performed by left shifting the  $R-B$  value 9 bits to obtain a maximum of a 16 bit number, then dividing by  $R$  (an 8 bit number) using a division function supplied by the Quartus II software. This resulted in  $s$  being scaled between 0 and 255. The 48 MHz clock was used to perform the division and was completed under four clock cycles so this computation does not cause any delay to the image processing. Finally, to eliminate any effects of background pixels having similar saturation values to the kernel, all pixels with an  $R$  value less than 15 were set to a saturation value of zero.

The number of pixels classified by the filter as belonging to blue-eye regions was saved in addition to the sum of the  $R-B$  values and the sum squared of the  $R-B$  values of the pixels in these regions. The average and the variance of the pixels classified as blue-eye regions were then computed with Eq. 2 using division functions supplied by the Quartus II software.

$$\text{var}(\text{blue-eye}) = (\text{sum}^2 - \text{sum} \times \text{mean}) / (n - 1) \quad (2)$$

where  $\text{var}(\text{blue-eye})$  is the variance of the  $R-B$  image pixels that were classified as belonging to a blue-eye region,  $\text{sum}$ ,  $\text{sum}^2$ , and  $\text{mean}$  are the sum, sum squared, and mean of the  $R-B$  regions classified as blue-eye, respectively, and  $n$  is the number of pixels classified as belonging to a blue-eye region. Although the computation of variance requires the computation to be broken into three clock cycles, one for mean computation, one for multiplication and subtraction, and the division of a 22 bit number by a 12 bit number, the time required for the variance computation to execute after a complete image was acquired was less than 1  $\mu\text{s}$ . Therefore, for practical purposes, this method did not delay the ability of the sorter to respond in real time.

Three different image processing modules were programmed such that the three parallel kernel channels could be inspected independently of one another. The program required approximately 72 % of the logic elements available on the FPGA and 68 % of the memory. The EEPROM on the FPGA board was programmed with a default threshold level to classify kernels, and this level was read immediately after powering up. The threshold level was programmed to be adjustable with the attached quad encoder but was not used in the sorter testing experiments as discussed below.

### Sorter testing

After programming the FPGA, the 2,000 kernels used to develop the image algorithm were run individually down

the sorter a second time. As computed in real time by the FPGA, the count of blue-eye pixels, mean, and variance from the R-B image were recorded. These data were used to set threshold levels for separating the unsorted samples. Sorting was performed on the non-hand-picked 500 g portions of the samples that were collected from the five separate bins over two different years. After sorting, the accepted and rejected streams from each sample were inspected, and the number of blue-eye kernels in each fraction was recorded.

## Results

From the 2,000 images, it was determined that the optimal pixel gap and pixel saturation offset to use in the filter were 10 pixels and 24 levels of saturation, respectively. Applying these parameters to the training set, the blue-eye kernels had an average count of 146 blue-eye pixels, and the undamaged kernels had an average count of 30 blue-eye pixels. The minimum count for all of the kernels with blue-eye damage was 15, whereas 49 % of the undamaged kernels had counts less than 15. Inspection showed that 92 % of the blue-eye damaged kernels had counts greater than 60, whereas 87 % of the un-damaged kernels had counts less than or equal to 60, indicating a false positive rate of 13 % if a threshold of 60 was used. Most of the undamaged kernels with greater than 60 pixels classified as blue-eye were due to shadows on the kernels from irregular kernel surface morphology. The false positive rate could be reduced by using  $\text{var}(\text{blue-eye})$  in conjunction with the number of pixels that were classified as blue-eye. The study showed that 79 % of the undamaged kernels with blue-eye counts between 60 and 90 had  $\text{var}(\text{blue-eye})$  values of 35 or less. Conversely, 87 % of the actual blue-eye damaged kernels had  $\text{var}(\text{blue-eye})$  values greater than 35. Therefore, a three-step classification scheme was used as follows:

- (1) If the blue-eye pixel count  $>90$ , then classify the kernel as blue-eye damaged.
- (2) If the blue-eye pixel count is between 60 and 90, and  $\text{var}(\text{blue-eye}) >35$ , then classify it as blue-eye; otherwise classify it as undamaged.
- (3) If the blue-eye pixel count  $\leq 60$ , then classify it as undamaged.

This classification scheme correctly classified 92 % of the blue-eye damaged kernels and 94 % of the un-damaged kernels from the 2,000 hand-picked kernels from the training set.

After the classification scheme was programmed into the FPGA, the 10 un-picked 500 g popcorn samples were sorted. After inspecting these samples, it was found that, on

**Table 2** Sorting accuracies for the ten popcorn samples tested sorted from highest blue-eye accuracy to lowest

Sample	Years	Blue-eye accuracy (%)	Un-damaged accuracy (%)	Average accuracy (%)
1	2007	94	94	94
2	2008	93	93	93
3	2008	92	92	92
4	2008	90	93	92
5	2007	90	95	93
6	2007	89	93	91
7	2008	88	94	91
8	2007	87	94	91
9	2008	87	96	92
10	2007	86	96	91
Average		90	94	92

average, 90 % of the blue-eye damaged kernels were removed, and 94 % of the un-damaged kernels were accepted. The sorting accuracies among the ten samples ranged from 85 to 94 % for blue-eye damaged kernels and 92 to 96 % for undamaged kernels. Some kernels tended to be more round in shape, and they did not orient as well as the others; this was one reason for the sorting accuracy differences among the samples. The two samples with the most round-shaped kernels had the lowest accuracy for blue-eye damage but the highest accuracy for un-damaged kernels. The accuracies for all ten samples are shown in Table 2.

## Discussion

The accuracy of this sorter was comparable or better than previous technologies for separating blue-eye damaged popcorn. The original FPGA-based sorting system (Pearson [4]) achieved an average accuracy of 83 % compared with 92 % for the current study. The primary reasons for the better accuracy are the use of saturation and R-B values from the shadows; the original system used only the R pixels in the image. The LED lights used on this sorter assist with color fidelity; the original sorter used halogen lamps, which do not have much B light energy, and the color rendition in the original sorter was not as good. Finally, saturation and R-B are less prone to slight variations in lighting and kernel shade than absolute pixel intensity, which was used in the original sorter. There were some differences in sorting accuracy among samples from different bins and growing seasons. It is possible that slight adjustments to the thresholds could optimize the sorting accuracy for kernels of different sizes and shapes. Although the FPGA was programmed to allow threshold adjustment

through the quad encoder, no tests were performed to determine if this would be useful for some of the samples. Future testing is planned in a popcorn processing facility to determine the long-term robustness of the image processing algorithm and hardware, as well as adjustments to the thresholds to optimize sorting accuracy for a specific bin. This sorter has a much higher throughput than the original sorter (100 vs. 35 kg/h) because it has three parallel channels compared with one in the original sorter. The cost to fabricate this sorter is similar to, if not less than, that of the original sorter.

The average classification accuracies achieved by this sorter for blue-eye and undamaged kernels were only slightly lower (92 vs. 94 % overall) than those obtained by using frequency spectra features extracted from images and classifying them with a support vector machine (SVM) [8]. Furthermore, the Yorulmaz study used higher resolution images that were obtained from stationary kernels. Nevertheless, an FPGA is suitable for implementing an SVM, and this will be the focus of future study because the training of the SVM is more straightforward and may make this method of detecting small spots more easily adaptable to other commodities.

Although this study focused on blue-eye damaged popcorn, the sorter could possibly be used for other applications such as separating grains or legumes with small blemishes on them, for example, mottled lentils and insect infested beans and grains. The precise image filter and the thresholds used might need to be changed. The implementation of hsv chromaticity should also lead to an improvement in the sorting of grains by color over previous systems. Future studies will involve making the system more easily adaptable to other commodities.

## Conclusion

The goal of this study was to develop a sorter that can identify approximately 90 % of blue-eye damaged popcorn

kernels with a false positive rate under 5 %. This was almost accomplished, as this sorter could identify 90 % the blue-eye damaged kernels with a 6 % false positive rate. It is possible that the implementation of an SVM with more features could reduce the error rate and thereby achieve the accuracy goal. However, the current design is notably simple, and the image processing is expected to be robust because it is based on saturation and R-B images. Long-term testing should be performed in a popcorn processing facility to determine the sorter's actual performance over long periods of time.

## References

1. J.R. Mathiassen, E. Misimi, A. Skavhaug, A simple computer vision method for automatic detection of melanin spots in Atlantic salmon filets. *Machine vision and image processing conference, 2007. IMVIP 2007. International*. pp. 192–200 (2007)
2. K. Mertens, B. De Ketelaere, B. Kamers, F.R. Bamelis, B.J. Kemps, E.M. Verhoelst, J.G. De Baerdemaeker, E.M. Decuyper, Dirt detection on brown eggs by means of color computer vision. *Poult. Sci.* **84**, 1653–1659 (2005)
3. G. Venora, O. Grillo, R. Saccone, Quality assessment of durum wheat storage centres in Sicily: evaluation of vitreous, starchy and shrunken kernels using an image analysis system. *J. Cereal Sci.* **49**(3), 429–440 (2009)
4. T.C. Pearson, Hardware based image processing for high-speed inspection of grains. *Comput. Electron. Agric.* **69**(2009), 12–18 (2009)
5. T.C. Pearson, High-speed sorting of grains by color and surface texture. *Appl. Eng. Agric.* **26**(3), 499–505 (2010)
6. J. P. Allebach, Image scanning, sampling, and interpolation, in *Handbook of Image and Video Processing*, ed. by A. Bovik (Academic Press, San Diego, 2000), pp. 629–643
7. E. Reinhard, E.A. Khan, A.O. Akyuz, G.M. Johnson, *Color Imaging, Fundamentals and Applications* (A.K. Peters Ltd., Wellesley, 2008)
8. O. Yorulmaz, T.C. Pearson, A.E. Çetin, Cepstrum based feature extraction method for fungus detection. *Proc. SPIE* **8027**, 80270E (2011). doi:[10.1117/12.882406](https://doi.org/10.1117/12.882406)