Development of Computer Vision-Enhanced Smart Golf Ball Retriever

Shun-Liang Lee[1], Pang-Chieh Lin[1], Sheng-Jie Lin[1] and Huang-Kuang Kung[2*]

(Received December 04, 2019)

Abstract

An automatic vehicle system was developed to assist golfers in collecting golf balls from a practice field. Computer vision methodology was utilized to enhance the detection of golf balls in shallow and/or deep grass regions. The free software OpenCV was used in this project because of its powerful features and supported repository.

The homemade golf ball picker was built with a smart recognition function for golf balls and can lock onto targets by itself. A set of field tests was completed in which the rate of golf ball recognition was as high as 95%. We report that this homemade smart golf ball picker can reduce the tremendous amount of labor associated with having to gather golf balls scattered throughout a practice field.

1. Introduction

Robot-assisted vehicles are a growing trend in the era of artificial intelligence (AI). The rapid growth of AI has enhanced the progress of autonomous vehicles with automatic functions such as the smart golf ball picker developed in this study.

In most golf courses, people usually practice chipping and putting on practice fields or greens. For convenience, hundreds of golf balls are used at a time. Since collecting these golf balls can be troublesome, we developed a smart golf ball picker that can locate and retrieve these golf balls automatically. However, due to the random heights of grass in the field, the shadow cast by the grass may partially or completely block the vision recognition of the picker. In order to make the golf ball picker become smarter, open source software for computer vision such as OpenCV is utilized to assist in the recognition of the golf ball locations.

In this study, different light conditions caused by sunshine, cloudy weather, and shaded areas under the trees are considered. We found that the accurate rate of golf ball recognition increased significantly so that the homemade golf ball picker was able to retrieve the majority of the golf balls.

2. Methods

2.1 Color isolation and selection techniques

OpenCV is the world's most popular computer vision repository and it is used extensively

1 Graduate Student, Institute of Mechatronic Engineering, Cheng Shiu University, Kaohsiung, Taiwan

2 Professor, Institute of Mechatronic Engineering, Cheng Shiu University, Kaohsiung, Taiwan

by developers and researchers due to its widespread support. [1] In order to successfully identify the golf balls, the color of the balls must be distinguished from the grassy background, keeping in mind that white and yellow golf balls are the most frequently used. In the practice field filled with scattered balls, images were taken for further analysis.

In computer vision, an image in RGB format can be divided into three color channels with different color intensity, i.e. the intensity and color information are mixed in RGB color space, but in HSV color space the color and intensity information is separated. This makes the HSV color space more robust and sensitive to lighting changes. [2]

A mask is simply a region of interest (ROI) of the image. In this case, we checked the HSV image, and filtered for the specific color that was selected from the lower and upper bounds on the intensity of the HSV color space. The ROI that matches the criteria was considered a mask variable. [3] Python was the programming language used in our case. Figure 2-1 shows the results of distinguishing and extracting original images for red, orange, yellow, purple, green, blue, and white colors.

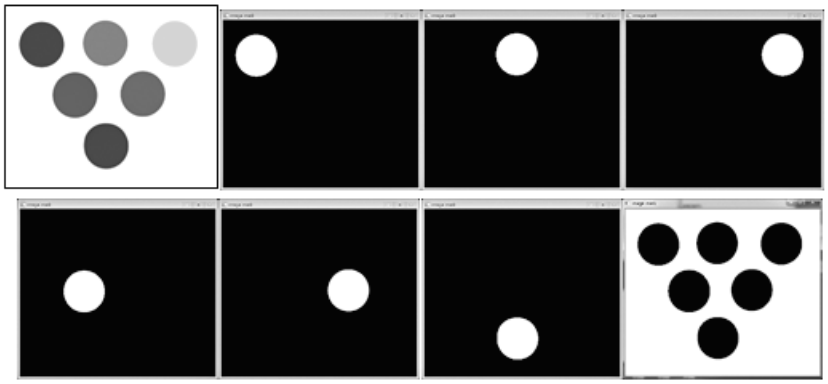

Figure 2-1. Distinguish and extract for red, orange, yellow, purple, green, blue, and white colors

2.2 Sobel and Laplacian operator to find image gradients

In order to detect the boundary of images, Sobel and Laplacian operators were utilized. Since the Sobel operator is a joint methodology of Gaussian smoothing and differentiation operations, it is more robust and resistant to imaging noise. Furthermore, this method comes with directivity. The direction of derivatives can be specified separately into vertical or horizontal components. Figure 2-2 shows the filtering results for the Sobel operator in vertical and horizontal directions. However, in this study, it seems that the differentiation in the x and y coordinates was trivial due to the circular feature of golf balls.

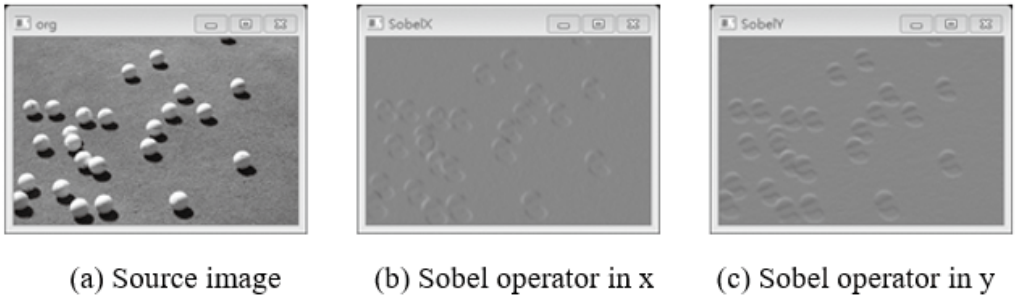

Figure 2-2. Sobel operator in vertical or horizontal directions

In order to find the edge of an object, the Laplacian operator takes second-order derivatives to find the maximum or the peak of image gradients. It calculates the Laplacian value of the image given by the relation, $\Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$, where each derivative is found using Sobel derivatives. The src variable represents the source image. In the parameter of the Laplacian operator, a kernel size can be used to select the effective pixel regions. The bigger the kernel size input, the larger the neighboring pixels involved in the calculation. This operation is known as a convolution in engineering mathematics.

Various kernel sizes were evaluated in the detection of the boundary of golf balls. Figure 2-3 shows the effect of kernel size on the filtering of the boundary of golf balls. Based on current resource images with the consideration of golf ball distance and pixel resolution, we found that kernel sizes of 3 or 5 were optimal.

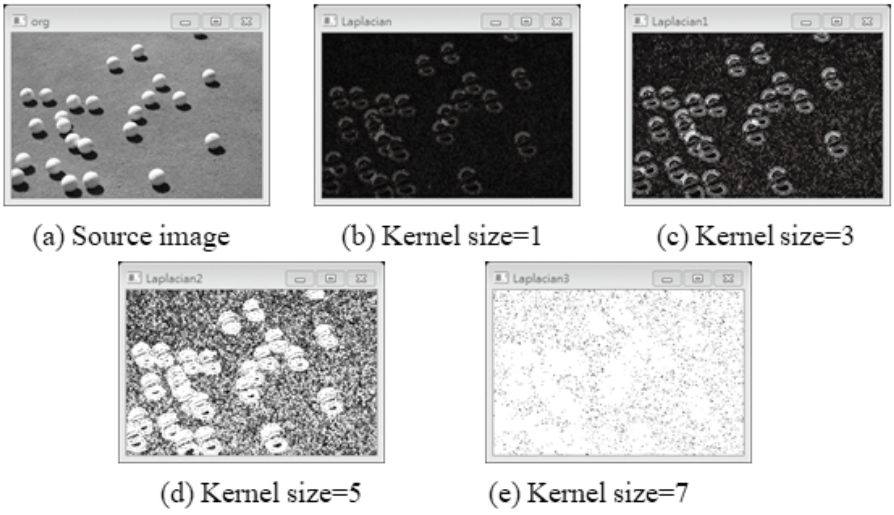

Figure 2-3. Effect of kernel size on the filtering of the boundary of golf balls

2.3 Canny edge detection for golf balls

The Canny edge detection algorithm is known to many as an optimal detector and satisfies three main criteria: low error rate for existent edges, good localization to minimize the distance between edge pixels detected and real edge pixels, and minimal response for only one detector per edge. In Canny edge detection, the first step is the use of a Gaussian filter to remove the noise. For the current case, a 5×5 kernel size was adopted. Next, the Sobel kernel

was chosen to obtain the first derivative in both the horizontal and vertical directions in order to find the maximum direction of gradient for each pixel. The true edges are determined by a hysteresis thresholding method. Therefore, it is very important to select the maximum and minimum thresholding values to obtain better results. Figure 2-4 shows the image recognition results of golf balls with their original image on the left and the recognition result image on the right.

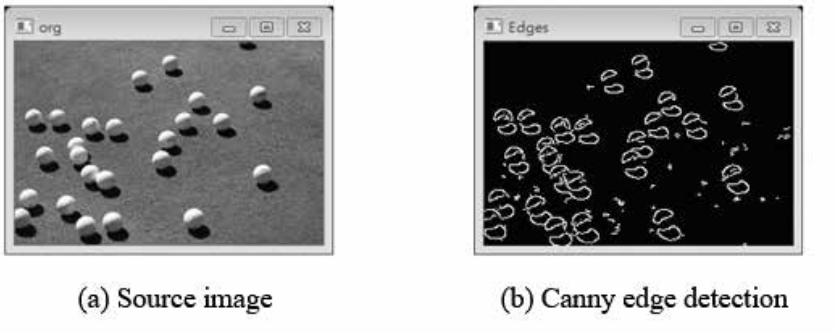

Figure 2-4. Canny edge detection results for golf ball image recognition

The configuration of the homemade golf ball picker is shown in Figure 2-5. The parallel plastic plates were used as the capture mechanism of the golf balls. The golf balls are caught by the friction forces of two plastic plates and lifted upward by rotating the plates. The golf balls can be released by the stopper and placed into the buckets. The camera is placed on the top of the golf ball picker.

Figure 2-5. Configuration of the homemade golf ball picker

The layout of the control algorithm is illustrated in Figure 2-6. The golf ball picker can be manipulated either by a cell phone app or a long-range remote controller. In the app, the controller can be switched between manual and automatic modes.

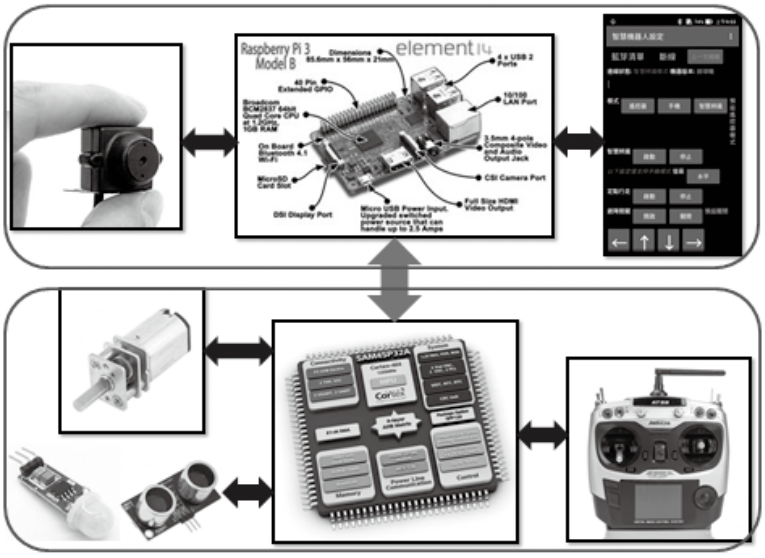

Figure 2-6. Layout of the control algorithm of the homemade smart golf ball picker

In order to acquire the distance and orientation from the golf ball locations in an image, a set of field tests was conducted to obtain their correlations. Figure 2-7 shows the layout for the testing field and also specifies the coordinate information for each of the golf balls. The measurement of the testing field was 5×5 m. The distance between the golf ball picker and the boundary of the testing field was 2 m. Therefore, the first row of golf balls was located 3 meters in front of the golf ball picker. Each point in the figure indicates the location of a golf ball.

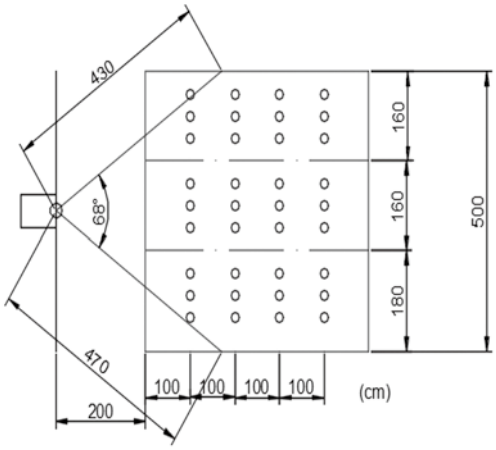

Figure 2-7. Layout of the testing field with golf balls

3. Results and Discussion

First, the golf balls have to be recognized by the golf ball picker in order for the picker to detect the distance and orientation of the balls. The previously mentioned OpenCV technique was used to recognize the golf balls on grass. Several of the golf ball images were taken under

different lighting conditions. A series of OpenCV techniques including GaussianBlur to filter the noise of the image and Canny to detect the golf ball boundary were utilized to process this image recognition. Figure 3-1 shows the results of Canny edge detection for typical golf balls.

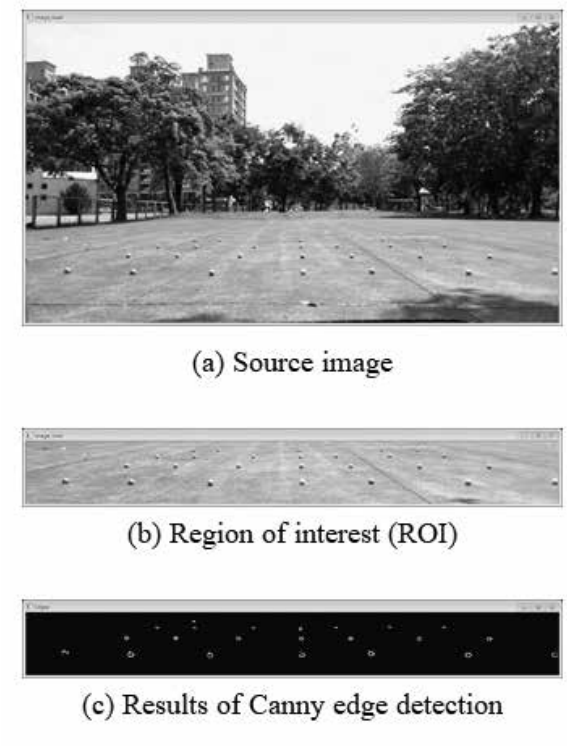

Figure 3-1. Canny edge detection procedure

The locations of each golf ball on the image can be further analyzed through their image center of gravity. This allows the corresponding distances and angles for each golf ball in the real field to be converted. The recognition results of golf balls versus the location coordinates in pixels are shown in Figure 3-2. The number indicated in the figure is the y coordinate of the pixel for each of the balls. The y coordinates for three rows of golf balls were found to be close to each other. This indicates that the estimation of vertical distance between the golf ball and ball picker was reasonably accurate. However, the x coordinates of the pixel are significantly distorted due to the vision of the lens. Fortunately, we know the information of the golf ball locations in advance. The relationship between the x coordinates of the pixel and the horizontal distance is found to require a second-order curve fitting equation.

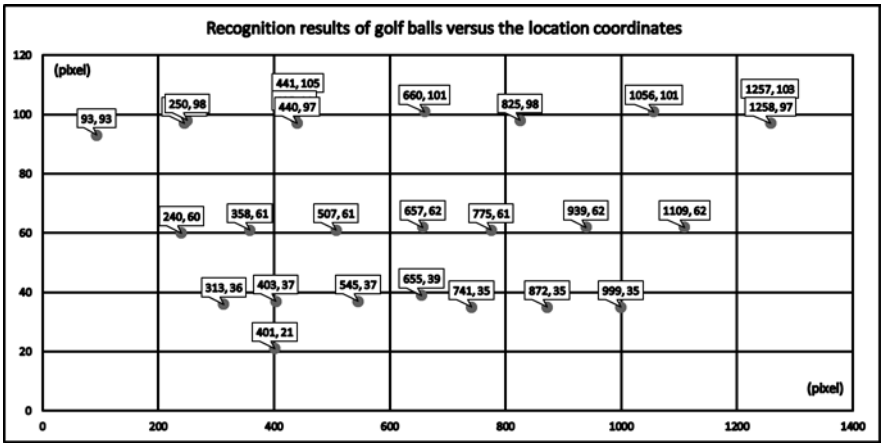

Figure 3-2. Recognition results of golf balls versus the location coordinates

The other important factor that affects the accuracy of golf ball recognition is the lighting conditions of the environment. With the aid of a light sensor, the rate of recognition for golf balls under various lighting conditions has reached a very satisfactory level, as shown in Figure 3-3.

The recognition rate of golf balls is evaluated by comparing the number of recognized golf balls to the overall number of golf balls in different grass field environments, lighting environments, and weather conditions. Based on the field tests, the detection rate of golf balls was as high as 95%. Most of the missing golf balls were those covered by long grass or those for which the recognized geometry was out of shape. In the future, this can be further improved by adding new criteria to our recognition algorithm.

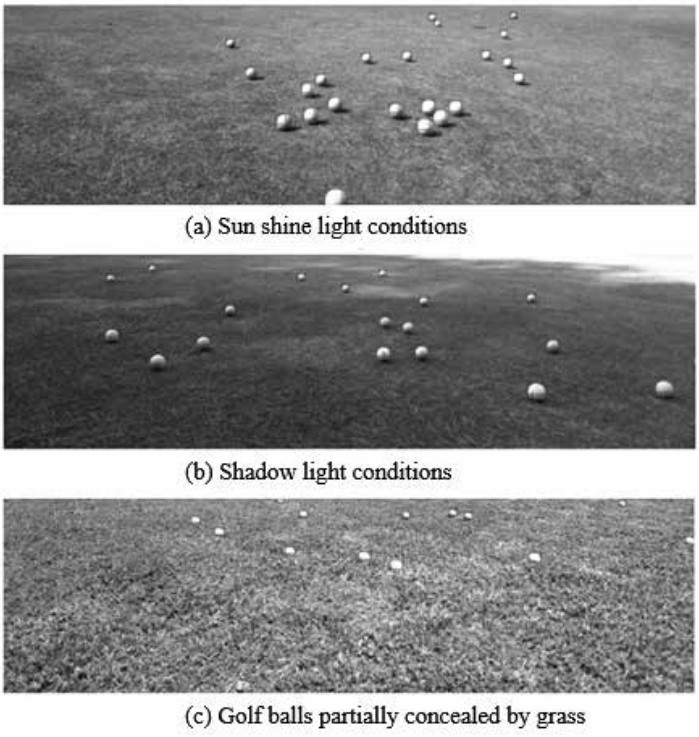

Figure 3-3. Recognition of golf balls under various lighting conditions

4. Conclusion

In this study, a homemade golf ball picker was developed to help golfers collect golf balls on a practice field or green. The distance and orientation relationship equations of golf balls and the golf ball picker can be obtained from the previous experiments. We found that the y coordinate in pixels is approximately proportional to the vertical distance and the relationship can be expressed in a linear polynomial equation. However, the x coordinate in pixels and horizontal distance is found to have a nonlinear correlation. A second-order curve fitting equation was therefore applied. With the aid of image recognition, the detection rate of golf balls was up to 95%, which will dramatically increase the acquisition rate of our homemade smart golf ball picker.

References

[1] Installing OpenCV 3 With Python On Mac OS X, https://prateekvjoshi.com/2015/10/09/installing-OpenCV-3-with-Python-on-mac-os-x/

[2] 16 OpenCV Functions to Start your Computer Vision journey (with Python code), https://www.analyticsvidhya.com/blog/2019/03/OpenCV-functions-computer-vision-Python/

[3] Color Detection in Python with OpenCV, https://henrydangprg.com/2016/06/26/color-detection-in-Python-with-OpenCV/

[4] OpenCV-Python Tutorials:Image Gradients, https://OpenCV-Python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_gradients/py_gradients.html

[5] OpenCV-Python Tutorials:Canny Edge Detection, https://OpenCV-Python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

[6] Learning OpenCV3 computer vision with Python (second edition), Joe Minichino and Joseph Howse, PACKT publishing.

[7] Programming computer vision with Python, JanErik Solen, O'REILLY publishing.