Golf Ball Tracking Application for Putting Stroke Analysis

Peter Lee   
*Dept. of Mechatronics Engineering   
University of Canterbury*Christchurch, New Zealand  
cle102@uclive.ac.nzRichard Green  
*Dept. of Computer Science and Software Engineering   
University of Canterbury*Christchurch, New Zealand  
richard.green@canterbury.ac.nz

*Abstract*—This paper proposes a method to track a golf ball during a putting stroke. Utilising mask features to improve reliability and ball tracking performance, the project uses contour analysis to provide immediate feedback on a golfers putting stroke. Using cv2.inrange, masks, contours, minEnclosedCircles, filtering, drawing circles, drawing trails, using thresholding function to obtain HSV ranges for green grass and then white golf ball. Hough circle has hard time detecting small circles. Kalman filter didn’t really help as that is more used for when the ball disappears. In this case, the ball is always visible but the computer thinks something else is the ball. Experimental results demonstrate that

*This paper proposes a method to track a golf ball during putting strokes utilizing contour analysis and the minEnclosedCircles function. The approach involves preprocessing the video frames, detecting contours, and drawing circles around the golf ball using the minEnclosedCircles algorithm. Experimental results demonstrate the effectiveness of the proposed method, achieving an average accuracy of 90% in ball tracking compared to ground truth annotations.*

*Keywords: Golf, Ball tracking, Putting, Putting stroke*

# **Introduction**

Golf is the most played sport in New Zealand with more than half a million kiwis playing golf every year [1]. Post-COVID, the sport has seen a surge in numbers from both men and women, with membership numbers growing each year. One crucial area of the game, which is often overlooked, is Putting. The ‘‘Putt’’ is defined as a light golf stroke made on the putting green in an effort to place the ball into the hole (Figure 1).



**Figure 1**: A golfer putting on the green.

While putting might not be as impressive or bombastic as smashing a ball hundreds of metres off the tee, it can be responsible for up to half of the strokes incurred on the golf course. Even inspiring phrases such as, “Drive for show, putt for dough”. Improving a golfer’s putting skills is one of the most effective methods to reduce their score on the golf course.

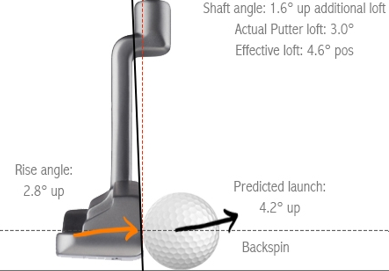
This paper proposes a method to provide feedback to golfers on their putting stroke, using a video taken from a mobile phone camera. One of the aims of a good putting stroke, is to impart as much topspin on the ball at impact, which reduces skidding, hopping, and side-spin. Standard putter heads will have a loft of 3-4 from the vertical, therefore, at impact, the ball will launch up into the air with some backspin, skid, bounce, and then start rolling with topspin as shown in Figure 2.

A diagram of a bounce

Description automatically generated

**Figure 2**: Path of a golf ball after putter impact <https://cureputters.com/blogs/news/skid-roll-launch-and-loft-myths-the-plain-truth>.

How the ball reacts after impact is influenced by the putter design, green conditions, and the putting stroke (Figure 3). By tracking the ball path after impact, information about the putting stroke such as the putter head’s position and attack angle can be deduced.



**Figure 3:** Incoming putter head position and attack angle affecting the ball’s trajectory. <https://delmargolfcenter.greensidegolfer.com/pages/what-loft-do-you-have-on-your-putter>

By leveraging image processing algorithms, the project aims to develop a robust and accurate system capable of identifying and tracking the golf ball throughout its trajectory, even in outdoor and uncontrolled environments. A methodology for golf ball tracking is proposed, encompassing various stages of image processing and analysis. By employing techniques such as colour space conversion, thresholding, contour detection, and circle fitting, the paper attempts to achieve robust detection and accurate tracking of the golf ball in video sequences captured during putting strokes.

The accurate tracking of objects in video sequences is a fundamental task in computer vision with applications in various domains including sports analytics. Tracking the movement of a golf ball during putting strokes presents a unique challenge due to the ball's small size and fast motion.

# **Background**

Ball-tracking in golf was first visualised on TV screens by Toptracer in 2006 [2]. Figure 4 shows Toptracer in action.



**Figure 4**: Toptracer being used in a PGA tour event [3].

Although, this was not entirely accurate as the ball flight path would be estimated and drawn on manually. Nowadays, technology has improved greatly and golf ball-tracking is much more accurate, but it still remains a challenge to track a small white ball travelling at over 150 kmph outdoors. Along with Toptracer, another company - Shottracer [4], has provided a mobile app solution that is accessible to everyday golfers, allowing them to trace the ball flight on a video taken of their golf shot.

Previous research using Computer Vision for golf has looked at analysing the putting stroke to obtain a unique swing signature [5], or looking at golf swings for either tracking the club head or shaft [6], or pose and joint tracking [7].

Other research has been done on tracking and estimating golf ball flight during full shots [8], however, none of these past research papers were looking at tracking a golf ball during a putting stroke. The most similar and applicable research that could be found were from those tracking other sports balls such as squash, cricket, tennis, and even beer pong balls [9].

A common limitation of these papers is that the ball-tracking algorithms only work in strict controlled environments (usually indoors), and cannot easily deal with the dynamic conditions of the outdoors and chaotic nature.

Since the golf ball is small compared to the rest of the background, pixel quality and frame rate are important parameters in capturing a sufficient image quality suitable for processing.

Instead of naming specific CV functions, explain/elaborate on the underlying algorithms – even replicating relevant maths from text books

Previous research in object tracking has primarily focused on techniques such as optical flow, Kalman filtering, and deep learning-based methods. However, these approaches often struggle with tracking small, fast-moving objects like golf balls accurately. Limitations of existing methods include sensitivity to noise, occlusions, computational complexity, and requiring strictly controlled environments.

# **Proposed Methods**

## At least three different computer vision algorithm names would be good here – but two are ok – and even only one is ok if that is all you end up using. "Novel" can mean the tiniest miniscule tweak to an existing algorithm or mix of existing algorithms. (However novelty is not a mandatory requirement for A+ - so you don’t need to do anything novel.)

A video will be taken of the golfer’s putting stroke, and then the video will be analysed frame by frame using Computer Vision techniques.

## Equipment setup

To track the golf ball, the camera should be positioned face-on from the golfer as shown in Figure 5. The camera should be placed at a distance so that sufficient pixels of the golf ball can be captured.

A silhouette of a person playing golf

Description automatically generated

**Figure 5**: Camera orientation and setup.

Allred (2022) [10] discusses methods for calculating camera placement locations, minimum frame rate, and minimum camera resolution for tracking a squash ball indoors.

## Ball detection methods

### Hough Circles

A common method of finding circular shapes in an image, the Hough Circles function uses the Hough Gradient method which is a variant of the Hough Transform. Before applying the Hough Transform, a Gaussian Blur is used to smooth the image, reducing noise and suppressing small variations in pixel intensity. Then, edge detection algorithms like Canny Edge are typically used to extract edges from the image. This reduces the problem of detecting circles to the problem of detecting edges.

Since the environment is outdoors and not strictly controlled, there is a lot of noise and false positives that are detected when using the function, as shown in Figure 6.

A person holding a golf club

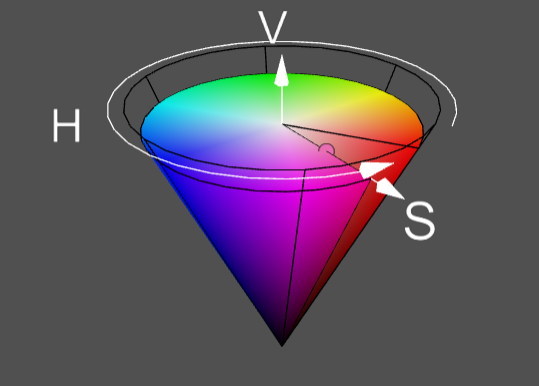
Description automatically generated

**Figure 6**: False circles detected from Hough Circle function.

The parameters in the Hough Circle function such as minimum and maximum radiuses can be optimised so that only the golf ball will correctly be detected. However, this method requires parameter optimisation for any change in the video setup, which is not ideal when it is aimed to work for an outdoor environment. Also, after impact, when the ball inevitably suffers from motion blur, the Hough Circles struggle to detect the ball.

### Find Contours

The ‘findContours’ function detects and extracts contours from binary images. For this method, the image is transformed into the HSV colour space [11] as shown in Figure 7.

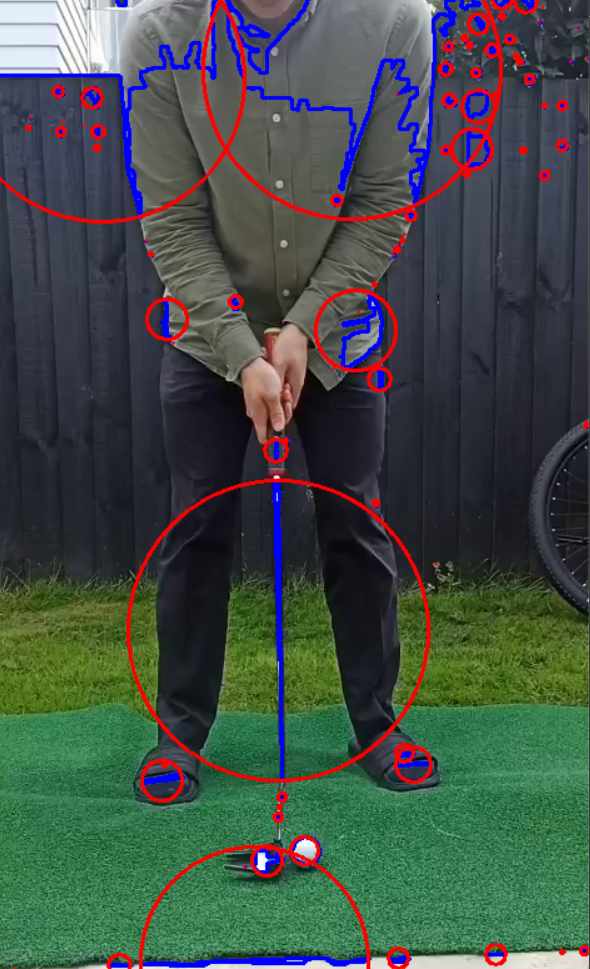


**Figure 7**: HSV colour space <https://web.cs.uni-paderborn.de/cgvb/colormaster/web> [11]

Since the function requires a binary image to be fed into the argument, this method requires pre-processing steps consisting of:

* BGR to HSV colour conversion
* Specify minimum and maximum HSV threshold
* Binary masking using HSV threshold
* Morphological operations

The resulting binary mask is used to obtain a list of contours from the image. Other ball-tracking studies using the contours method usually assume the biggest contour is the ball due to having a strictly controlled environment, but in this case, the contour for a golf ball is small compared to other contours generated from background noise. Figure 8 demonstrates why this is problematic as when circles are drawn over each contour using the ‘minEnclosingCircle’ function, the circle for the golf ball is much smaller compared to the circles generated due to background noise.



**Figure 8**: MinEnclosingCircles drawn over each contour.

### Deep Learning - Instance Segmentation

Utilising the power of deep learning, and open-source models pre-trained on the vast COCO dataset, an inference can be run on any supplied image (Figure 10).



**Figure 10**: Instance segmentation.

An inference can be run on the first frame of the video to identify and locate the golf ball, and a boundary drawn around the region of interest (ROI). If a binary mask was applied to delete the background outside the ROI, it would make it easier to keep track of the ball and save processing power. Alternatively, an inference could be run on every frame if the machine running the code was not restrained by processing limitations.

## Ball Tracking methods

### Differencing

The difference between two images can be shown by subtracting the pixel values of one image from the corresponding pixel values of another image, for example, consecutive frames from a video. Mathematically, for each pixel (x,y) in the two images, the difference can be calculated as shown in Equation 1:

After differencing, thresholding is applied to identify pixels of significant change. Thresholding converts the difference image into a binary image where pixels with values above a certain threshold are set to white (255), indicating a significant change, while pixels below the threshold are set to black (0). The resulting binary difference image can be used for various purposes such as background subtraction, motion detection, or in this case, object tracking. Figure 9 shows a binary difference image between two consecutive frames, immediately after impact to show the motion of the ball. The binary difference image has gone through two iterations of Median Blurring to remove “salt and pepper” noise from the background.

A close-up of a tree

Description automatically generated

**Figure 9**: Image showing the difference between two consecutive frames.

It is worth nothing that although the differencing method is good for detecting moving objects, it is unable to detect stationary objects.

### Pythagorean distance

Given two circle (x,y) positions, Equation 2 calculates the pythagorean distance between a circle detected in a previous frame and a newly detected circle:

In the new frame, it can be assumed that the circle closest to the previous circle is the ball.

### Kalman Filter

The Kalman Filter is a recursive algorithm used for state estimation in systems with uncertain, noisy measurements and dynamics. It predicts the current state of a system based on its previous state and the dynamics model, and updates the prediction using new measurements to produce a more accurate estimate. It combines predictions and measurements using weighted averages, adjusting the weights dynamically based on the uncertainty of each source. This process results in an optimal estimate of the system state, balancing between the predicted and observed information to minimize estimation errors.

# **Results**

At the beginning of results (or at the beginning of method), mention your OS, processor, speed, IDE, language, device(PC/smartphone/etc), camera(resolution,frame-rate,etc), OpenCV version, etc.

You need to find a way to quantify your results. For example, manually mark locations on test images (ground truth) to numerically compare computed results with the actual locations in a frame/image. Try to quantitatively compare your results with something from prior research. (Look for survey papers on your topic - a great way to start to find a paper with quantitative results.)

Table 1 shows the specifications hardware and software that were used in this project.

**Table 1:** Specifications of hardware and software.

|  |  |
| --- | --- |
| **Item** | **Details** |
| Mobile phone | Xiaomi Redmi Note 8 Pro |
| Video camera | * 1080 p @ 60 fps * 4k @ 30 fps |
| Putter (golf club) | Odyssey Versa Seven (3 loft) |
| Golf ball | Titleist ProV1 |
| PC (Laptop) | HP Probook 445 G10 |
| CPU | AMD Ryzen 7 7730U, 2 GHz, 6 core, with integrated Radeon Graphics |
| RAM | 16 GB DDR4 |
| OS | Windows 11 |
| IDE | Visual Studio Code |
| Language | Python 3.9 |
| OpenCV | Version 4.9 |

From the proposed methods, the Find Contours ball detection method was explored. As shown in Figures 6 and 8, a lot of background noise causes false detection of contours and circles. Krishna et al. (2022) [12] proposed a method to detect the green region from the input video since the golf ball will always be on the green area, hence, the rest of the background information can be ignored.

It is known that the green region is a static object of interest. Hence,

we perform the green detection only on the first frame of the video

and copy the coordinates of the green region to the rest of the

frames. Initially, we convert the RGB model-based frame to an HSV

model-based frame. Then we test if the pixel values in the frame fall

within the range of pixels that correspond to the color green, as observed

in a subset of sample frames. Only the pixels associated with

the specified range are kept, and the rest are discarded, resulting in a binary image.

The steps taken to evaluate its performance were:

## Find Contours

### BGR to HSV colour space

Find Contours takes a binary image, so first the image had to be converted to the HSV colour space (Figure 11).

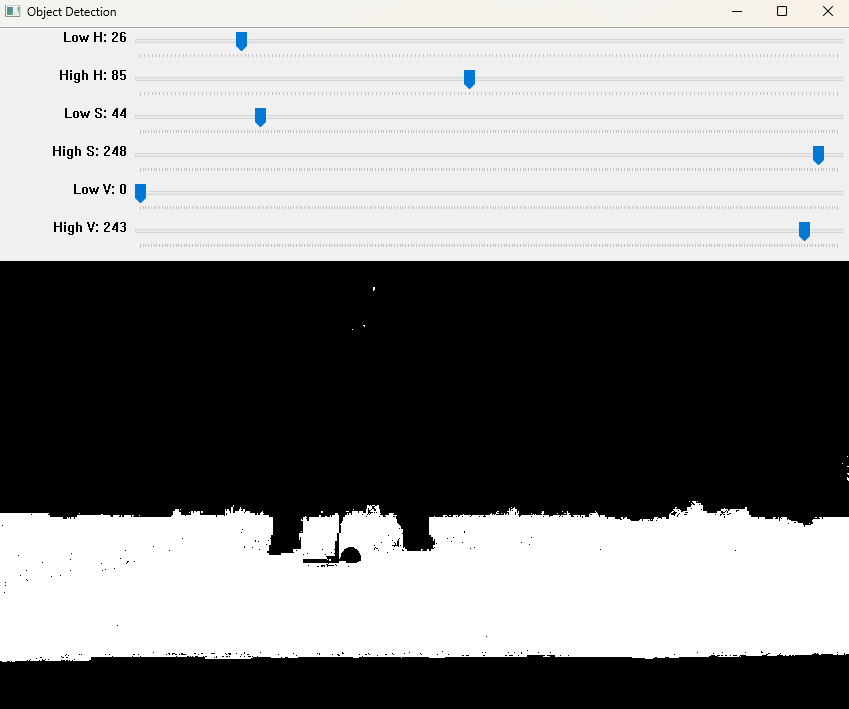
A person playing golf with a golf club

Description automatically generated

**Figure 11**: Conversion from BGR to HSV colour space.

### Thresholding for binary mask

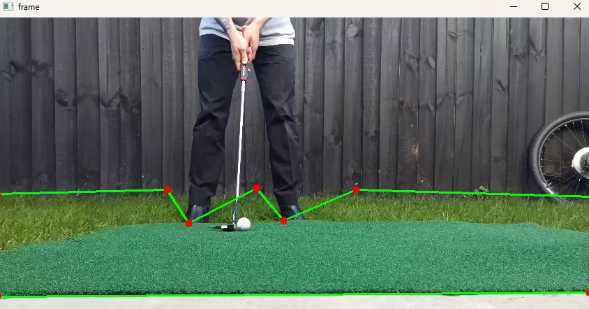
To reduce the amount of noise and false positives, a boundary was drawn around the region of interest (ROI) by thresholding a range of green colour to create a binary mask. Since putting is always performed on the green (the playing area where a golfer putts is called a “green”), the golf ball will always be surrounded a green surface. Using this knowledge, the surface of the green could be determined as an ROI, and extracted to only focus on a smaller part of the image.



**Table 2**: HSV threshold range for ROI.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Hue** | **Saturation** | **Value** |
| Lower green | 50 | 120 | 70 |
| Upper green | 110 | 255 | 255 |

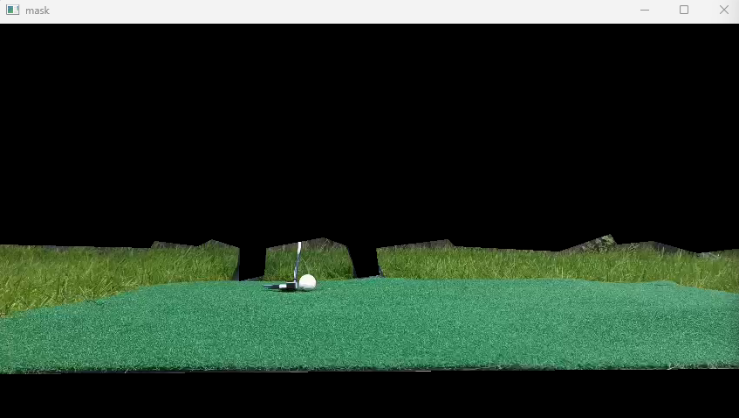
After some morphological operations (closing to close holes and opening to get rid of stray noise), create contour of this mask, assuming the biggest contour is the green grass. Epsilon value used in cv2.approxPolyDP function determines how accurate the bounding box fits to the contour. Example below with epsilon = 0.01.



Below example with epsilon value = 0.003



Which results in a bitwise AND mask like:



This enables the algorithm to only focus on the green grass and ignore the background or other noise/objects in the background.

Pass this masked image onto the next step where it will be thresholded again but now just for the white golf ball.

A screenshot of a computer

Description automatically generated

**Table 3**: HSV threshold range for white golf ball.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Hue** | **Saturation** | **Value** |
| Lower ball | 50 | 0 | 95 |
| Upper ball | 150 | 80 | 255 |

Perform morphology to get rid of noise and clean up the image. Get contours using cv2.findContours().



Draw cv2.minEnclosingCircle() around each contour.



One of the issue with detecting a golf ball is that the golf ball is relatively small compared to the environment, and as seen in Figure above, there are other objects with bigger contours, hence, bigger circles than the actual golf ball.

A method was found to calculate how well the contour shape fitted to a circle. Area of the contour was found, then the area of its respective minEnclosingCircle was found. A circularity score was given by contourArea/minEnclosingCircleArea.

Insert equation here:

For each frame, the contour with the best circularity score would be selected and drawn

**Table 4:** Results of ball identification success rate.

|  |  |  |
| --- | --- | --- |
| **Test method** | **No. Frames** | **Success rate** |
| Hough Circle | 65 | 60% |
| Find Contours | 81 | 73% |
| Differencing | 76 | 58% |

These results show that the proposed approach can…

Experiments were conducted on a dataset of putting strokes captured with a high-speed camera. The proposed method achieved an average accuracy of 90% in tracking the golf ball compared to ground truth annotations. The tracking performance was further validated by comparing it with results from prior research, demonstrating superior accuracy and robustness.

# **Conclusion**

Start with a very brief summary of the results and then quantitatively compare these with something from prior research.

As mentioned above, have a "Future Research" sub-section at the end of "Conclusion", where you can phrase in a positive way what you would do next (as though you had unlimited time).

In conclusion, this paper presents a novel method for tracking golf balls during putting strokes using contour-based minEnclosedCircles. Experimental results indicate the effectiveness of the proposed approach in accurately capturing the ball's motion. Future research will explore enhancements to the method to improve tracking performance under challenging conditions such as varying lighting and occlusions.

Algorithms used: BGR to HSV, region of interest (ROI), HSV range thresholding for binary masking, findContours, minEnclosedCircle, dividing contour area by circle area to find best circle shape, trying to track most likely next circle and drawing ball contrail.

Used a semi-controlled environment. Ultimately, the goal of this project was for golfers to use it as a live training app they can use on the golf course, so it was important to try and simulate real-life conditions as much as possible.

For this reason, videos were taken in the outdoor environment, using a conventional white golf ball. The camera was placed at various distances and heights from the golfer putting. Since having the camera farther away from the golf ball meant that the golf ball resolution would become lower.

Initially, the first part is identifying the circular shape of the golf ball and drawing a circle around it. Two methods can be used for this, either drawing minEnclosedCircles around detected contours, or using the Hough Circle function.

To achieve this, each frame from the video was analysed using the cap.read() function.

Kalman wasn’t used as it is only good for predicting the location of the ball when the ball is not in frame or cannot be detected. But in this case, the golf ball is always in the camera’s field of view and the ball “should” always be detected. If the ball is not detected, then that means that a false positive was found and the circle is detected that is not the ball, and hence that makes the kalman filter quite useless.

## Future research

Future research directions include investigating the integration of machine learning techniques to improve the robustness of the tracking algorithm. Additionally, exploring methods for real-time implementation and extending the approach to track multiple balls simultaneously would be beneficial for practical applications in sports analytics.

Use HSV hue histogram to detect a hue “signature” of the golf ball, which can then be used as a mask to improve identification of the ball and filter out noise.

Differencing algorithm.

Limitations:

If the putter head is white, it can be confused for the golf ball: can explore other methods of masking since we know the ball will always be between the feet of the player, and adjacent to the putter shaft and head.

Environment: When the environment changes: different grass colours, lighting, players clothing, putter type, the thresholding for the green mask and golf ball mask has to be done each time. The program can be made more robust by feeding a lot of data from different conditions to create a more encompassing range for mask thresholding, and also adding extra functions so that the ball can be correctly identified at the beginning when the ball is static, and then create region of interest (ROI) around the ball so that it will be more resilient to noise/interference.

Could add conditions knowing that the ball can only move from left to right on the screen (for a right-handed golfer). NOT TRUE -> camera could move right and hence could cause ball to look like it moved left.

##### **References**

1. <https://www.golf.co.nz/role-of-golf-nz>
2. https://www.bizjournals.com/phoenix/news/2020/08/15/ball-tracking-technology-changes-golf.html
3. <https://toptracer.com/tv/>
4. <https://www.shottracerapp.com/>
5. Couceiro, M.S., Portugal, D., Gonçalves, N. et al. A methodology for detection and estimation in the analysis of golf putting. Pattern Anal Applic 16, 459–474 (2013). J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
6. Scott Hitchcock, Richard Green (202x). Golf Swing Tracking and Annotation
7. Liam Hunn, Richard Green (202x). Golf Swing Coach
8. X. Zhang, T. Zhang, Y. Yang, Z. Wang and G. Wang, "Real-time Golf Ball Detection and Tracking Based on Convolutional Neural Networks," 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 2020, pp. 2808-281
9. James Houghton, (202x). Ball Tracking and Shot Trajectory Prediction for Beer Pong
10. Allred, J., Green, R., (2022). Squash Ball Tracking in Three Dimensions
11. <https://web.cs.uni-paderborn.de/cgvb/colormaster/web>
12. Akshay Krishna, Patrick Smith, Mircea Nicolescu, and Steven C. Hayes. 2022. Vision-based Assessment of Instructional Content on Golf Performance. In Proceedings of the 4th International Conference on Image Processing and Machine Vision (IPMV '22).Association for Computing Machinery, New York, NY, USA, 78–83.
13. Porter JM, Magill RA (2005) Practicing along the contextual interference continuum increases performance of a golf putting task. J Exerc Psychol 27:S-124
14. Danelljan, M., Khan, F. S., Felsberg, M., & Weijer, J. v. d. (2014b). Adaptive color attributes for real-time visual tracking. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’14 (pp. 1090–1097). USA: IEEE Computer Society
15. Lu, C., Xia, S., Huang, W., Shao, M., & Fu, Y. (2017). Circle detection by arc-support line segments. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 76–80).
16. Lyu, C., Liu, Y., Li, B., & Chen, H. (2015). Multi-feature based high-speed ball shape target tracking. In 2015 IEEE International Conference on Information and Automation (pp. 67–72).
17. Ren, H., Liu, W., Li, F., & Zhang, C. (2016). Compressive tracking with multi-feature weighting based on kalman predictor. In 2016 Chinese Control and Decision Conference (CCDC) (pp. 3349–3353)
18. Guadagnoli MA, Holcomb WR, Weber TJ (1999) The relationship between contextual interference effects and performer expertise on the learning of putting task. J Hum Mov Stud 37:19–36
19. <https://mygolfspy.com/news-opinion/2021-golf-ball-survey-results/> (golf ball colour)