Computer Vision : Snooker Project

Done by : Rahul Shah, Chadha Aouina **chadha@usi.ch , rahul.shah@usi.ch**

- Aim is to implement the concepts of computer-vision such as Hough-transform, DLT algorithm many more on the given video under name of WSC.mp4 by reconstructing snooker table and balls from side view with some predefined goals which are

  1. Pre-process video and filter only those frames that show side view
  2. Reconstruct camera position
  3. Reconstruct ball positions

---

1 Pre-process video and filter only those frames that show side view :

1.1 Pre-process video

- During this process, we have diagnosed the similarity by converting image from RGB to HSV format which is Intersection over Union (IoU) between the reference frame WSC sample.png and frames we have extracted from the video WSC.mp4.

Note :

IoU (Intersection over Union) is used to define the similarity of binary masks, such as in segmentation tasks in computer vision, which can be represented in our case as:

$$\text{IoU} = \frac{A \cap B}{A \cup B}$$

where A and B are the two binary masks (in our case, the green masks of the two frames).

- Having done analysis on the video we are able to extract around 15GB data which are frames with similarity greater than 0.95 and on evaluating we got around 367878 frames overall in the particular video. But for ease of evaluation of similarity we have used trimmed video for specific purpose of evaluation and with that we have created graph as shown in Fig 1 Which describes no of frames with greater than 0.95 threshold.

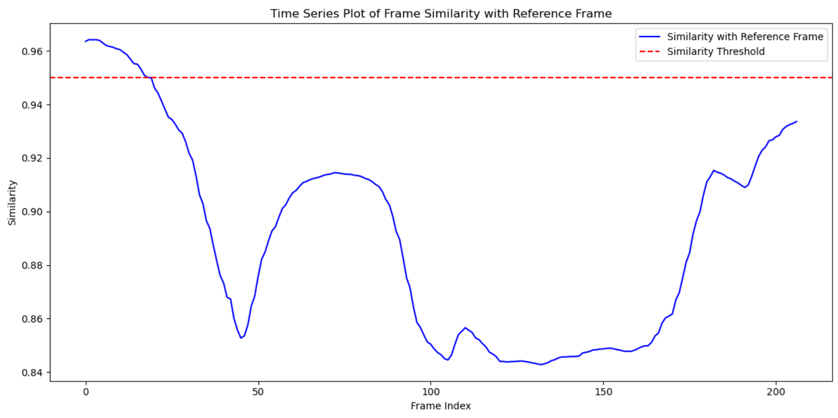


Figure 1: Frame Similarity with Reference Frame

1.2 Filtering

- Having the similarity we are now able to extract frames with certain threshold of similarity values (i.e. in our case is 0.95) and append those frames with similarity greater than 0.95 in to particular folder.

2 Reconstruct Camera :

2.1 Finding the Playing Area or Green Area Corner

- Firstly, we review the dimensions of the snooker table which will be used in determining the world coordinates during the DLT algorithm.
- Secondly, to find the specific corners of the table, we use the Hough Transform for visualizing edges or lines, which helps identify the corner coordinates by their intersections using Sobel gradients and Canny edge detection.

1. **Sobel gradients:** The Sobel operator is used in image processing and computer vision for edge detection by computing the gradient of the image intensity function. It calculates the gradient in both the x and y directions, which can then be combined to get the magnitude using the Euclidean distance formula $\sqrt{(\text{sobel}_x)^2 + (\text{sobel}_y)^2}$ and the direction of the edges.

   We use a grayscale image to simplify computation and improve the accuracy of the results, as well as reduce the dimensionality.

   Furthermore, thresholding is applied to create a binary image from the gradient magnitude image. Pixels with intensity values above a certain threshold are set to 255 (white), and all other pixels are set to 0 (black). This helps us find edges using the Canny edge detector.

2. **Hough Transform (lines):** The Hough Line Transform is a technique used to detect straight lines by pre-processing the edges found using Canny edge detection. With specific parameters, we can detect lines and draw out the coordinates of corners by their intersections.
   (a) **rho (ρ):** This is the distance from the origin to the line along a line perpendicular to the detected line. Essentially, it measures how far the line is from the origin.
   (b) **theta (θ):** This is the angle of the line relative to the x-axis. It represents the angle that the line makes with the horizontal axis of the image.
   (c) **Threshold:** This parameter is used to reduce noise or unwanted lines for a clearer visualization (we used 280 for a more perceptual view).

3. Above those parameters are used to determine the range from (ρ, θ), which represents all possible lines through that point. This is done by calculating the value of ρ using the formula

$$\rho = x\cos(\theta) + y\sin(\theta)$$

   for each edge point (x, y) (i.e., parameter space), and then passing it to an accumulator, which helps identify which (ρ, θ) pairs represent the most likely lines in the image (essentially a voting process for those pairs).

4. After calculating these, we can determine the lines and their intersections, which represent the four corners of the table. (Obtained image is shown below as Fig 2)

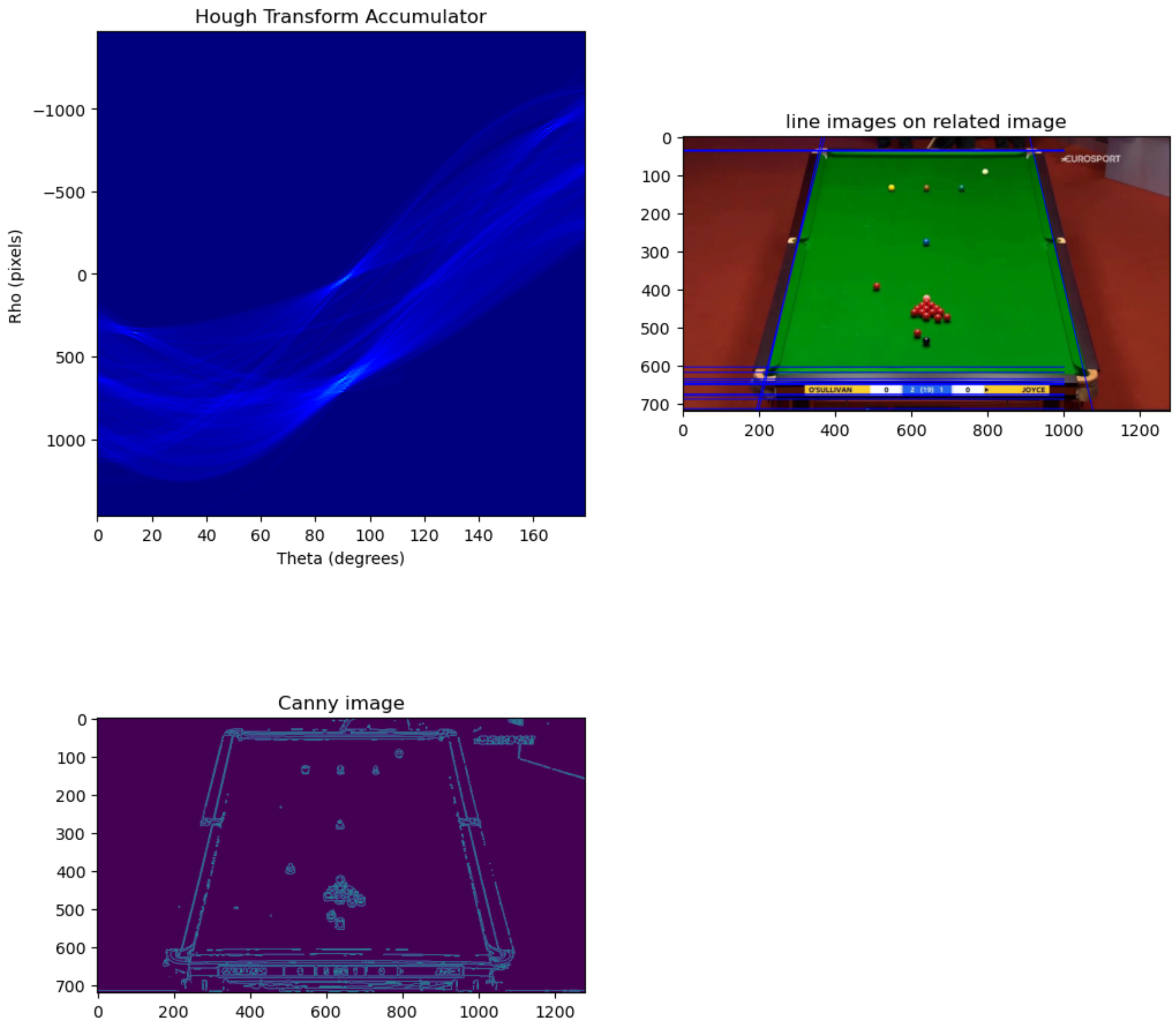


Figure 2: Hough transform

5. **Explanation of Hough tranform Image:** As we can see the accumulator image which represents Hough space which is calculated by ρ with a point at each angle from 0 to 180. The points in Hough space make a sinusoidal curve. as u can clearly see which is in light blue. The more curves intersect at a point, the more votes a line in image space will receive.

2.2 corresponding image points by ball marker pos :

- **Ball coordinates detection :** Here by , after detecting the corner coordinates we moved forward to detect specific colored spots [yellow, brown, pink, black, green, blue] on a snooker table from an image and calculates their coordinates. It uses color segmentation in the HSV color space, contour detection, and centroid calculation to determine spot positions. The coordinates are adjusted relative to the baulk line for consistency.

  Coordinates :

```
spots coordinates of YGBBPB are {'yellow': (405, 664), 'green': (638, 359), '
   brown': (1056, 664), 'blue': (659, 663), 'pink': (228, 31), 'black': (637,
   700)}
```

- **Explanantion of contur part of code :**

```
        if contours:
            largest_contour = max(contours, key=cv2.contourArea)
            M = cv2.moments(largest_contour)
            if M["m00"] != 0:
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                return (cX, cY)
```

  This code first checks if any contours are detected in an image. If contours are present, it identifies the largest contour based on its area. Then, it calculates the centroid of this largest contour by computing its moments. If the contour has a non-zero area (indicated by the zeroth moment m00), it calculates the centroid coordinates (cX, cY) by dividing the first order moments m10 and m01 by m00. Finally, it returns the centroid coordinates (cX, cY).

2.3 Use correspondence pairs to find camera matrix P by DLT algorithm :

- **Usage :** The idea of the DLT is to rewrite the perspective projection equation as a homogeneous linear equation and solve it by standard methods using the World points (3D coords) and image points (2D coords).

  1. The perspective projection equation describes how 3D points in the world coordinate system are projected onto a 2D image plane in the camera coordinate system. It is typically represented using homogeneous coordinates
  2. Standard methods such as Singular Value Decomposition (SVD) to estimate the parameters of the camera projection matrix P and using that decompose the obtained projection matrix P to extract intrinsic parameters (e.g., focal length, principal point) and extrinsic parameters (e.g., rotation, translation) of the camera.(i.e. in our case are K-Intrinsic matrix, R-Rotation matrix, C-Camera center). Also calulated corresponsing time which turns out DLT-2 has less computation time
  3. **Coords :**

     (a) **World points coords :** Those are selected manually with the given table dimensions.

```
world_coords = {'yellow': [0.292, 3.277, 0],'green': [0.292, 0.292, 0],
    'brown': [0.292, 1.778, 0],'blue': [1.778, 1.778, 0],'pink': [2.565,
     1.778, 0],'black': [3.261, 1.778, 0],'corner_top_left': [0, 0, 0],'
    corner_top_right': [3.569, 0, 0],'corner_bottom_left': [0, 3.569,
    0],'corner_bottom_right': [3.569, 3.569, 0]}
```

(b) **image points coords :** Those are calculated using above corners coordinates and ball marker pos in 2.1 and 2.3 respectively.

```
spots_coordinates = {'yellow': [405, 664],'green': [638, 359],'brown':
    [1056, 664],'blue': [659, 663],'pink': [228, 31],'black': [637,
    700],'corner_top_left': [356, 36],'corner_top_right': [356, 39],'
    corner_bottom_left': [362, 36],'corner_bottom_right': [362, 39]}
```

Note :

We have done two DLT algorithm (i.e. DLT-1 and DLT-2) one with coordinates found by resulting codes and other ones were given.

2.4 2D to 2D Transformation

During our attempt to warp the image using the homography matrix derived from the detected corners of the snooker table, we encountered difficulties achieving an accurate transformation. As a result, we were unable to obtain a satisfactory warped image that accurately represented the top-down view of the snooker table.

Given these challenges, we decided to utilize the coordinates provided by the professor, which allowed us to perform a 2D to 2D transformation directly. This method involved mapping the given coordinates to a new coordinate system that approximates a top-down view of the snooker table.

Additionally, to improve the visibility of the important areas in the image, we also had to zoom in and crop the image. Specifically, we cropped the image from $x = 225$ to $x = 1000$ and $y = 20$ to $y = 600$, and then resized it to the specified dimensions of 720×1445.

(Fig 3)

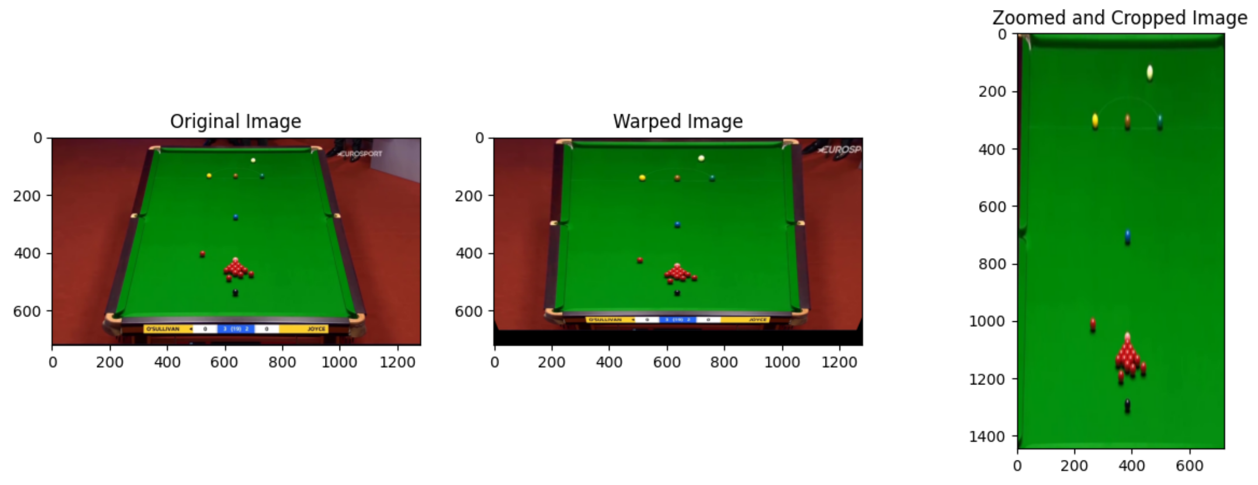


Figure 3: camera reconstruction

3 ball positions reconstruction

In this section, we describe the process of detecting red balls in the snooker table image, identifying highlights within these balls, and reconstructing their positions.

3.1 Finding Red Balls

The function `find_red_balls(image)` detects red balls in an image through the following steps:

1. **Convert the Image to HSV Color Space:** The image is converted from BGR to HSV color space because the HSV space makes it easier to filter specific colors.
2. **Define Red Color Ranges:** Red color can span two different ranges in the HSV color space. Two sets of lower and upper bounds are defined to capture all shades of red.
3. **Create and Combine Masks:** Masks for the two red ranges are created and then combined to isolate all red regions in the image.
4. **Apply Morphological Operations:** Morphological closing and opening operations are applied to remove noise and fill gaps in the combined mask. This step ensures that the detected red regions are smooth and continuous.
5. **Display the Mask:** The mask is displayed to visually confirm that the red regions are correctly identified.
6. **Find Contours and Enclosing Circles:** Contours are found in the mask to detect the shapes of the red regions. For each contour, the minimum enclosing circle is calculated. Only circles with a radius above a certain threshold are considered, filtering out small regions that are unlikely to be balls.

3.2 Finding Highlights within Red Balls

The function `find_highlights(image, balls)` identifies highlights within the detected red balls:

1. **Create a Mask for Each Ball:** For each detected ball, a mask is created to isolate the ball region from the rest of the image.
2. **Extract the Ball Region:** The ball region is extracted using bitwise operations with the mask.
3. **Convert to HSV Color Space:** The extracted ball region is converted to HSV color space.
4. **Thresholding to Identify Highlights:** A threshold is applied to the value channel to identify bright regions within the ball, which are likely to be highlights caused by reflections of light.
5. **Display the Thresholded Image:** The thresholded image is displayed to visually confirm the identified highlights.
6. **Find Contours of Highlights:** Contours of the white regions in the thresholded image are found. For each contour, the centroid is calculated and stored as a highlight position.

3.3 Reconstructing Ball Positions

The function `reconstruct_ball_position(highlight_positions, l_z, c_x, c_y, c_z)` reconstructs the positions of the balls based on the detected highlights:

1. **Approximate Ball Positions:** For each highlight position, the corresponding ball position is approximated in the 2D image plane. The x and y coordinates of the ball are assumed to be the same as the highlight position.
2. **Choosing the Maximum Number of Highlights:** Since multiple highlights can be detected within each ball, we decided to choose the maximum number of highlights detected as the final highlight positions for reconstructing the ball positions. This approach helps ensure the most accurate representation of the balls' positions.

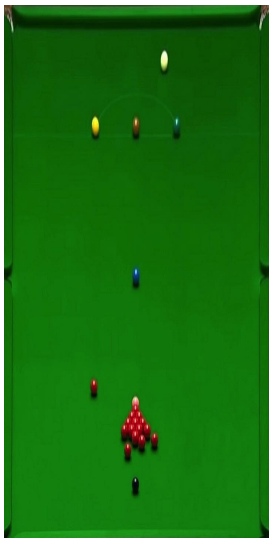
3.4 Reconstructing the Image

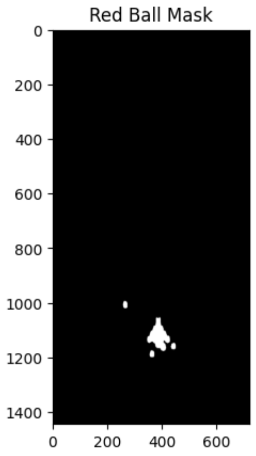
The function `reconstruct_image(ball_positions, image_size)` creates a new image to visualize the reconstructed ball positions:

1. **Create a Green Background:** A blank image with a green background is created to represent the snooker table.
2. **Draw Red Balls:** For each reconstructed ball position, a red circle is drawn on the green background to represent the ball.

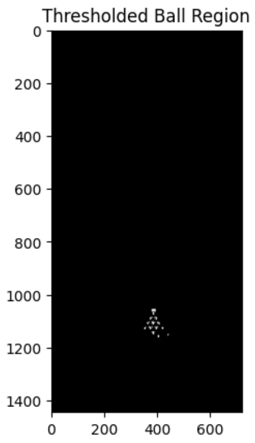
3.5 Example Usage

we used the final image from the camera reconstruction that was done previously

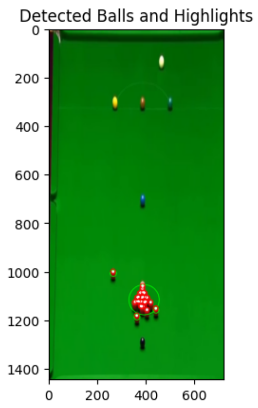

(a) Original Frame



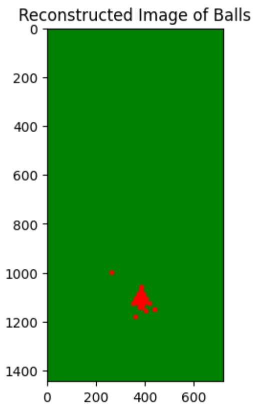
(b) applying mask



(c) applying threshold



(d) Detected Balls



(e) Reconstructed Frame

Figure 4: stages of ball reconstruction

4 Applying Methods to Filtered Video

In this section, we describe the process of applying all the previously discussed methods to the filtered video available in the zip document provided. This involves processing the video frame by frame to detect and reconstruct the positions of red balls.

4.1 Process Overview

The filtered video, named WSC_filtered.mp4, was processed using the same techniques described earlier in this report. The steps involved in this process are as follows:

1. **Loading the Video:** The video was loaded frame by frame for processing.
2. **2D to 2D Transformation:** For each frame, we applied the homography transformation to warp the image, ensuring the perspective is corrected to approximate a top-down view of the snooker table.
3. **Cropping and Zooming:** After warping, the image was cropped from $x = 225$ to $x = 1000$ and $y = 20$ to $y = 600$, and then resized to the specified dimensions of 720×1445 to enhance visibility.
4. **Detecting Red Balls:** The `find_red_balls` function was used to detect red balls in the zoomed frame by converting the image to HSV color space, creating masks for red color ranges, and finding contours.
5. **Finding Highlights:** For each detected ball, the `find_highlights` function identified highlights within the balls using thresholding and contour detection.
6. **Reconstructing Ball Positions:** Using the detected highlights, the `reconstruct_ball_position` function approximated the ball positions. We decided to use the maximum number of highlights detected to ensure accurate representation of the balls' positions.
7. **Reconstructing the Image:** Finally, the `reconstruct_image` function was used to create a new image with a green background, drawing red circles to represent the reconstructed ball positions.
8. **Saving the Output:** The processed frames were saved into a new video file named WSC_processed.mp4.

4.2 Implementation Details

- **Homography Matrix Calculation:** A homography matrix was computed using given 2D correspondences to transform the perspective of each frame.
- **Frame-by-Frame Processing:** Each frame of the video was processed individually. The frame was warped using the homography matrix, then cropped and resized to improve visibility.
- **Ball Detection and Highlight Identification:** The red balls were detected in each processed frame, and highlights within these balls were identified to reconstruct the ball positions.
- **Visualization:** The reconstructed ball positions were visualized by drawing red circles on a green background, representing the snooker table. This visualization was saved frame by frame into a new video.

4.3 Conclusion

By applying the previously discussed methods to the filtered video frame by frame, we were able to accurately detect and reconstruct the positions of red balls on the snooker table. The final processed video, WSC_processed.mp4, provides a clear and interpretable visual representation of the ball positions throughout the video.
This comprehensive approach ensures that all frames meeting the similarity threshold are processed, enhancing the accuracy and reliability of the ball position reconstruction.