Gebze Technical University

Computer Engineering

Introduction to Computer Vision(CSE 463)

Homework #1

Berkan AKIN

171044073

Part A

Requirements

1. Take a picture of the target soccer field with your cell phone.
2. To find the homography, you need 4 points. Use the intersection of the parallel lines of the soccer field as the 4th point. To distinguish the corners, you may print different markers for each corner, such as numbers.
3. Estimate a homography between the input image and the model soccer field. OpenCV functions **findHomography** or **getPerspectiveTransform** will be helpful.
4. Using the new transform, convert the input image and display the output as shown above. You may use the OpenCV function **perspectiveTransform** for this.

Desing

This code outlines the process of detecting the four corners of a soccer field in an image, calculating a homography matrix using these corners, and then applying this homography to transform the image into a flat view. Below is a detailed explanation of the steps, suitable for inclusion in your report:

Image Loading and Preprocessing

- **Loading the Image:** The image named '18.jpeg' is loaded using the OpenCV library with the `cv2.imread` function.
- **Converting to Grayscale:** The loaded image is converted from BGR (Blue-Green-Red) color space to grayscale using the `cv2.cvtColor` function. This step facilitates edge detection by reducing the complexity of the image.

Edge Detection and Contour Finding

- **Edge Detection:** The edges in the image are detected using the Canny edge detection method. The `cv2.Canny` function is utilized for this purpose, with specified lower and upper threshold values for detection.
- **Finding Contours:** Contours are found over the detected edges using the `cv2.findContours` function. Among these contours, the largest one (presumably the soccer field itself) is selected.

Corner Detection and Homography Calculation

- **Corner Detection:** A rectangle approximation is made on the found contour using the `cv2.approxPolyDP` function. This approximates the four corners of the contour.
- **Homography Matrix Calculation:** The four detected corner points and the targeted four corner points (the corners in a flat view) are used to calculate a homography matrix with the `cv2.findHomography` function.

Applying Homography and Displaying Results

- **Applying Homography:** The calculated homography matrix is applied to the original image using the `cv2.warpPerspective` function, transforming the image into a flat view.
- **Displaying Results:** The original image with marked corners and the image with applied homography are displayed on the screen using the matplotlib library. In the original image, the corner points are marked in green.

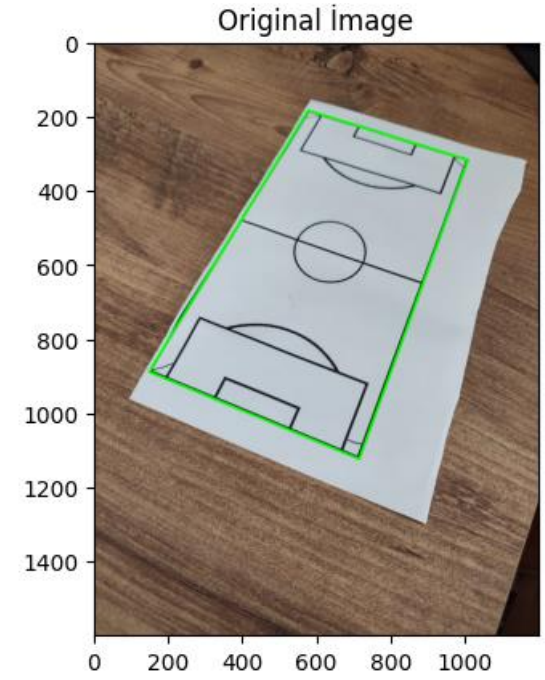
This process, especially useful for images of objects with specific geometric structures like a soccer field, allows for correcting the geometry of the image. Applying homography corrects perspective distortions on the image and enables obtaining a top-down view of the object. These procedures are frequently utilized in image processing, augmented reality applications, and object recognition fields.

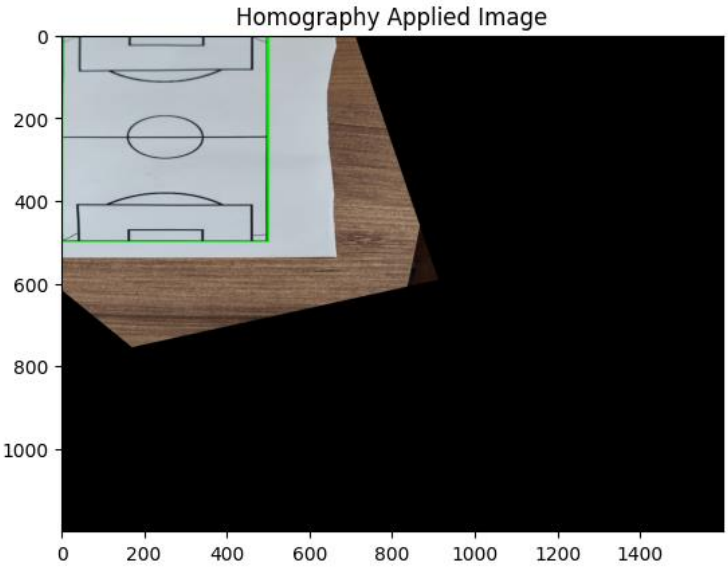
Part B

Examples

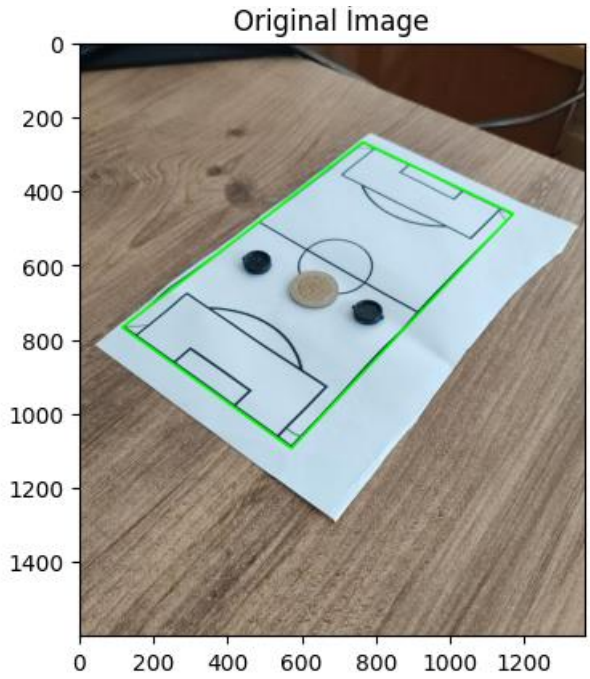
The homography images of photographs taken from different angles are presented below.

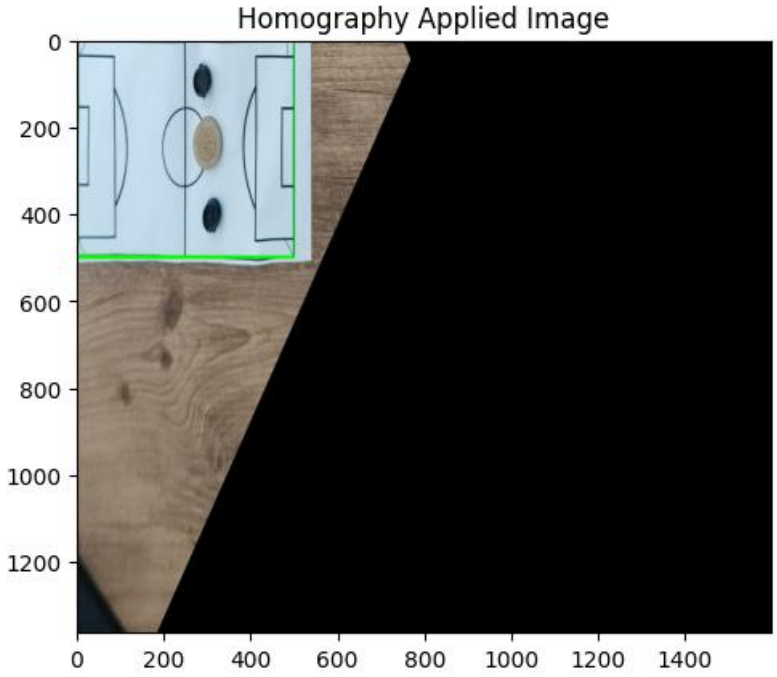
#1





#2





Note 1: Sample image outputs are included in the file attachment.

Note 2: For the homography transformation, 4 points have been used as the corner points of the field; the 4th point has not been used as the intersection of lines.

Requirements

1. Take images of the same soccer board with players on it.
2. Transform the input image to the model soccer field. Note that the players will look very odd on this model because they are not part of the homography plane.
3. Draw a circle around the players and transform the image back to its original form. This step is tricky; you will need to figure out some problems.
4. Do this with a different color for each player. You are supposed to detect the players automatically and draw the circles automatically without any user help. Be careful not to overpaint the players.

Desing

This script demonstrates an advanced image processing technique involving the detection of the corners of a soccer field, application of homography to transform the image to a top-down view, detection of blue objects within this transformed view, and finally, the drawing of circles around these detected objects. Below is a detailed explanation suitable for a report:

Image Loading and Initial Display

- The script begins by loading an image ('18.jpeg') using OpenCV's **cv2.imread** function.
- The original image is displayed with its colors converted from BGR to RGB for accurate color representation in Matplotlib.

Preprocessing and Edge Detection

- The image is converted to grayscale to simplify the edge detection process.
- Canny edge detection is applied to the grayscale image to identify edges. This is a crucial step for contour detection.

Contour Detection and Corner Identification

- Contours are detected in the image using the **cv2.findContours** method. The largest contour is presumed to be the soccer field, based on its area.
- A quadrilateral approximation of this largest contour is performed to identify the corner points of the field. If exactly four corners are identified, the script proceeds.

Homography Calculation and Application

- With the corner points detected, the script defines destination points representing the expected positions of these corners in a top-down view.
- A homography matrix is computed using the source (detected corners) and destination points. This matrix is used to warp the image to a top-down perspective of the soccer field.
- The transformed image is displayed, showing the soccer field from a top-down view.

Detection of Blue Objects

- The script then focuses on detecting blue objects within the transformed image by converting it to the HSV color space and applying a mask that isolates objects within a specific blue color range.
- The mask is displayed, highlighting the blue objects.

Drawing Circles Around Detected Objects

- Contours of the detected blue objects are found in the mask. For each contour, a circle is drawn around it in the top-down transformed image, indicating the location of blue objects (such as players or equipment).

- These circles are drawn with a significant thickness to ensure visibility.

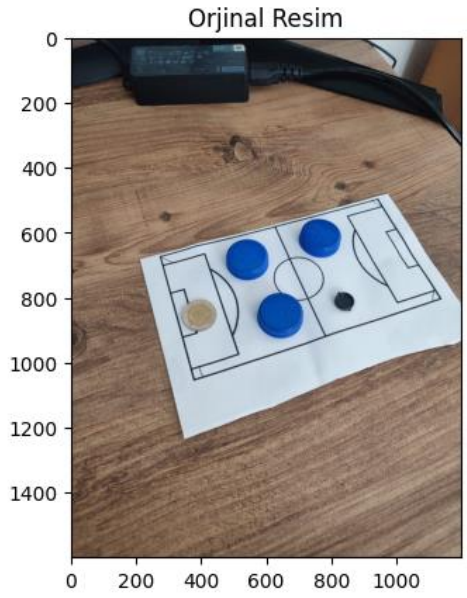
Image Restoration

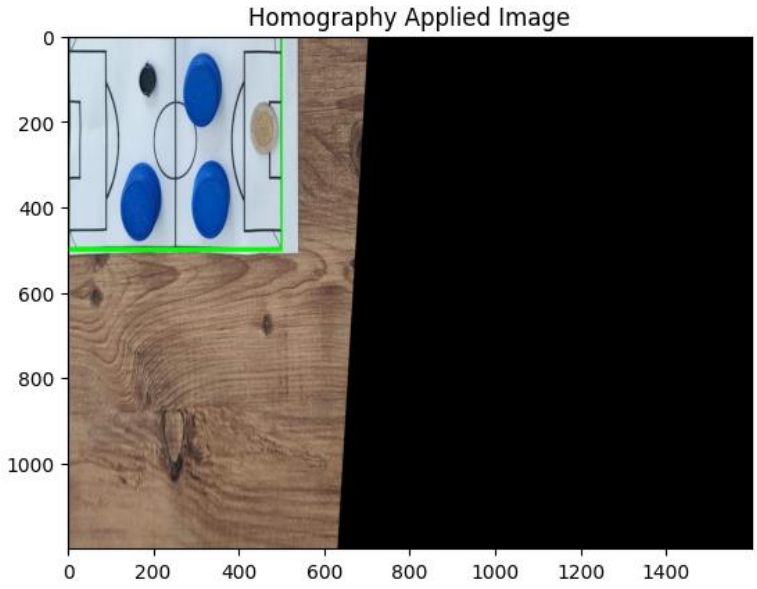
- If the initial homography application was successful, the script calculates an inverse homography matrix to transform the image back to its original perspective while retaining the circles drawn around the blue objects.
- The final image, showing the original view with circles drawn around detected blue objects, is displayed.

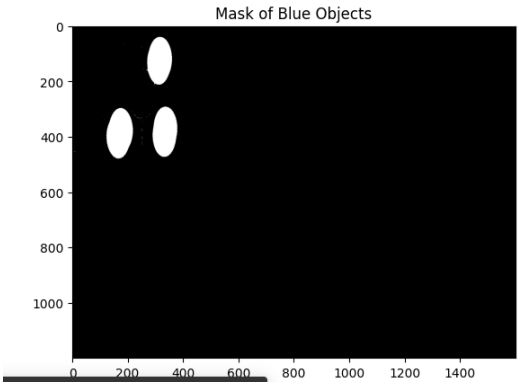
This process showcases the combination of several image processing techniques: contour detection for identifying the boundaries of objects, homography for perspective transformation, color space conversion for object detection based on color, and finally, geometrical transformations to annotate and emphasize detected objects. This approach is particularly useful in applications such as sports analytics, where it is necessary to identify and track objects or players on a field from different camera angles.

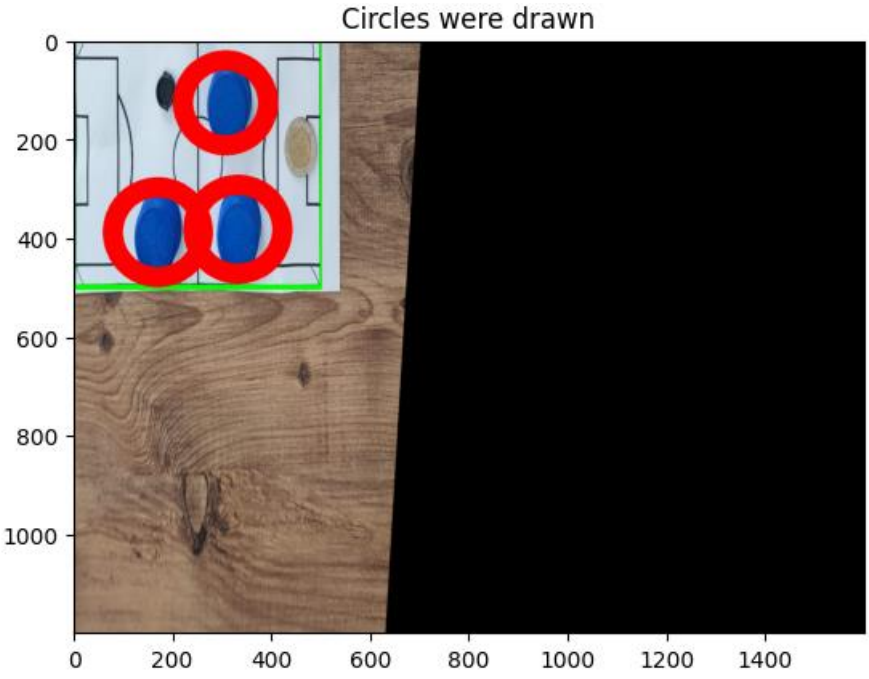
Examples

First, we apply homography to the image. After applying homography, we pass the image through a blue HSV filter to detect objects on the soccer field. Then, we apply the inverse homography to revert the image back to its original state.






Mask of Blue Objects
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


Circles were drawn
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

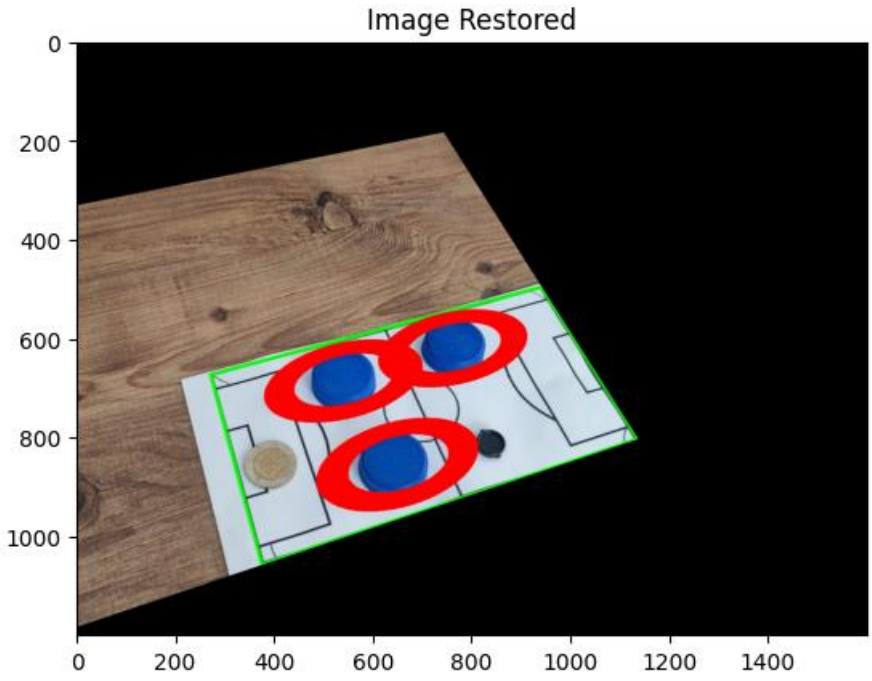
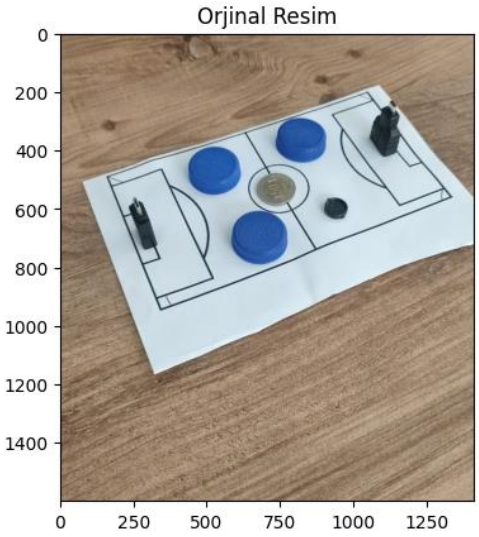
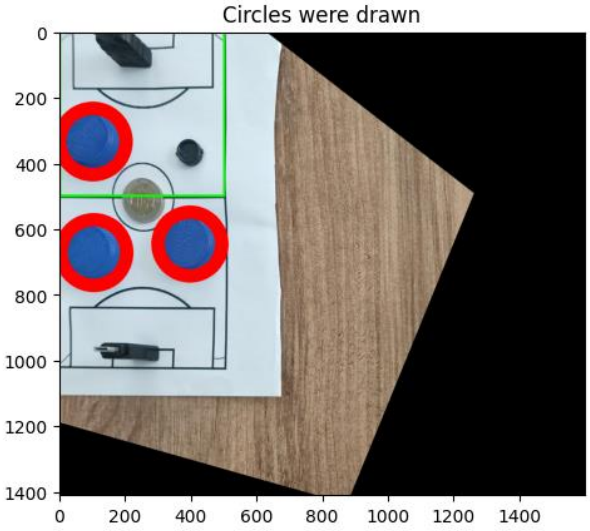
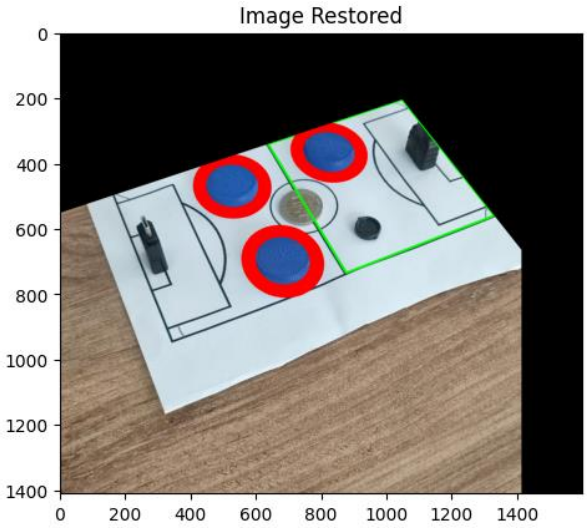


Image Restored
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

#2


Orjinal Resim
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250


Circles were drawn
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000
1200
1400



Note 1: While reverting the image back to its original state, there have been pixel losses.

Areas Where ChatGPT is Used

The topics given to ChatGPT are listed below.

1) How to Perform Corner Detection with OpenCV
2) Example homography application with OpenCV in Python
3) Example command to detect blue objects in an image using OpenCV with Python