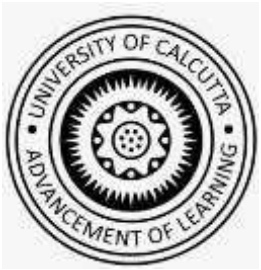
PROJECT REPORT ON

"SOCCER FIELD DETECTION USING HOUGH LINE DETECTION"

Submitted in the partial fulfillment of the requirement for
the award of degree of
Bachelors of Science in Computer Science
SUBJECT CODE : CMSA
PAPER CODE : CC-6-13 -pr & CC -6-14-pr



Submitted by-

Roll No: 203142-21-0137
Registration No: 142-1112-0441-20

Roll No: 203142-21-0156
Registration No: 142-1112-0577-20

Roll No: 203142-21-0102
Registration No: 142-1111-0568-20

Under the Guidance of
Mr. Saikat Sarkar
(Assistant Professor)
Department of Computer Science, Bangabasi College

Under

University of Calcutta

CERTIFICATE

This is to certify that the project entitled **"SOCCER FIELD DETECTION USING HOUGH LINE DETECTION"** has been done and submitted successfully by the undersigned students, as part of their University of Calcutta curriculum for the 3-year undergraduate degree course in B.Sc. Computer Science (Hons.), under Prof.Saikat sarkar, presented for the 6th semester

examination of the courses CMSA-CC-6-13-Pr & CMSA-CC-6-14-Pr, held on 1Aug ,2023. Furthermore, this is an original piece of work, and meets all the necessary criteria, to be accepted as a project work submitted for a Bachelor's degree programme in Computer Science.

GROUP MEMBERS:

SOURADEEP DAS

Roll No: 203142-21-0137

Registration No: 142-1112-0441-20

PRITAM HALDAR

Roll No: 203142-21-0156

Registration No: 142-1112-0577-20

SWARUPANANDA DAS

Roll No: 203142-21-0102

Registration No: 142-1111-0568-20

INTERNAL EXAMINER

PROF. SAIKAT SARKAR

DEPARTMENT OF COMPUTER SCIENCE

BANGABASI COLLEGE

UNIVERSITY OF CALCUTTA

EXTERNAL EXAMINER

HEAD OF THE DEPARTMENT

PROF. SAIKAT SARKAR

DEPARTMENT OF COMPUTER SCIENCE

BANGABASI COLLEGE

UNIVERSITY OF CALCUTTA

Acknowledgement:

We take the opportunity to express profound sense of gratitude and respect to all those who helped us throughout the development of our project titled '**SOCCER FIELD DETECTION USING HOUGH LINE DETECTION**'.First and foremost, we are deeply indebted to our project supervisor, Professor . SAIKAT SARKAR, without whose assistance and dedicated involvement in every step throughout the process, our project would have never been accomplished. We would like to convey our heartfelt gratitude to our professors of the Department of Computer Science, Bangabasi College. Their enthusiasm for the project has always made a strong impression on us.

Last, but not the least, we would like to thank our friends for their constant support and valuable feedback. We must not forget thank our families for their trust on us through all ups and downs of life.

CONTENTS

Abstract

This paper proposes a strategy for the automatic registration of soccer images on a model of the field of play. First, a robust and efficient preprocessing is applied to discard the areas of the image that do not belong to the field of play and eliminate most edge points that are not part of the line marks. Then, a novel probabilistic decision tree is used to identify the most probable classification for the set of all the straight lines in the image. Additionally, the line surrounding the center circle is also modelled for providing results when only few straight lines are visible. Finally, a three-step analysis stage is applied to ensure the validity of the results.

To assess its quality, the strategy has been tested on several sequences corresponding to three stadiums with different characteristics. The results obtained have shown that the registration is successful in most images (94%).

Keywords: soccer, football, field of play, playing field, registration, segmentation, line segment detection, line classification, validation

INTRODUCTION

Currently, soccer is the most popular sport in the world [1], with more than 265 million players in more than 200 countries [2] and with the largest television audience. Thanks to this great popularity and to the technological advances pro5 duced in the last years, several artificial vision applications have been proposed to carry out the automatic analysis of soccer matches. In addition, these applications are increasingly demanded not only by the audience, but also referees, coaches, and players.

Domain Description:

Robotic applications: Soccer field detection can also benefit robotics applications, such as autonomous soccer-playing robots or robots designed to navigate in outdoor environments. By detecting the field lines, the robots can understand the field layout, localize themselves accurately, and plan their movements accordingly.

The Hough Line Transform is a popular technique used in computer vision for line detection. It operates in the parameter space, which is a transformed representation of the original image. In the parameter space, each line in the image corresponds to a point or curve, and the intersection of these curves indicates the presence of lines in the image.

To apply the Hough Line Transform for soccer field detection, we start with an image that contains a soccer field. The image is typically preprocessed to enhance features and remove noise. This can include converting the image to grayscale, applying filters, and performing edge detection using algorithms like the Canny edge detector.

Once the edges are detected, the Hough Line Transform is applied. The algorithm creates an accumulator array that records votes for each potential line in the image. For every edge point in the image, a curve or line is generated in the parameter space. The curves intersect and accumulate votes at the parameter space points that correspond to the lines in the image.

After the accumulation process, the accumulator array is analyzed to identify clusters or peaks. These peaks represent the parameters (such as slope and intercept) of the lines present in the image. By thresholding or selecting the most significant peaks, we can extract the detected lines.

However, it's important to note that the Hough Line Transform alone may yield false positives or noisy detections. Thus, additional filtering and refinement steps are usually employed to improve the accuracy of the detected lines. This can involve filtering based on line length, angle, or position within the image, as well as applying heuristics specific to soccer field markings.

The final result of soccer field detection using the Hough Line Transform is a set of detected lines that represent the boundaries, goal lines, and other markings of the soccer field. These lines can then be used for further analysis, decision-making, or rendering in various applications, as mentioned earlier.

Overall, soccer field detection using the Hough Line Transform combines image preprocessing, edge detection, Hough transform, and post-processing techniques to automatically extract the lines that define a soccer field from a given image. It enables the automated analysis of soccer-related data and facilitates applications ranging from sports analysis to robotic navigation and augmented reality experiences.

Motivation :

The motivation behind using the Hough Line Transform for soccer field detection lies in its ability to robustly identify lines in an image. By leveraging this technique, we can automatically detect and extract the lines that define the boundaries, goal lines, and other markings on a soccer field. This can have several practical applications, including:

1. Computer vision-based sports analysis: Soccer field detection using the Hough Line Transform can be used as a fundamental step in computer vision systems designed for sports analysis. By accurately identifying the field lines, analysts can extract valuable information such as player positions, ball trajectory, and movement patterns. This information can be further used for performance analysis, tactical evaluation, and generating statistics.

2. Automated referee assistance: Referees in soccer matches often need to make crucial decisions based on the position of players and the ball relative to the field lines. By automating the process of soccer field detection, the Hough Line Transform can assist referees in making accurate decisions. For example, it can help determine whether a player is offside or whether the ball has crossed the goal line.

3. Augmented reality and virtual reality applications: Soccer field detection using the Hough Line Transform can be utilized in augmented reality (AR) and virtual reality (VR) applications. By accurately detecting the

field lines, it becomes possible to overlay virtual objects, annotations, or statistics onto the real-world soccer field in AR/VR experiences. This can enhance the viewing experience for spectators or provide interactive coaching tools for players.

4. Robotic applications: Soccer field detection can also benefit robotics applications, such as autonomous soccer-playing robots or robots designed to navigate in outdoor environments. By detecting the field lines, the robots can understand the field layout, localize themselves accurately, and plan their movements accordingly.

Scope :

The scope of soccer field detection using the Hough Line Transform is primarily focused on identifying and extracting the lines that define a soccer field within an image or video frame. The goal is to automatically detect the boundaries, goal lines, and other relevant markings on the field.

The Hough Line Transform provides a powerful technique for detecting lines, and it can be effectively applied to soccer field detection in various scenarios. However, it's important to note that the scope of this technique is limited to the detection of lines and does not encompass other aspects of soccer field analysis, such as player tracking, ball detection, or semantic understanding of the field.

The scope of soccer field detection using the Hough Line Transform includes:

1. Line detection: The primary objective is to accurately detect the lines that form the boundaries, goal lines, and markings on a soccer field. This includes straight lines such as field boundaries and goal lines, as well as curved lines like center circles or penalty areas.

2. Line extraction: Once the lines are detected using the Hough Line Transform, the next step is to extract the relevant lines from the image or video frame. This involves filtering and refining the detected lines to remove false positives and retain only the lines corresponding to the soccer field.

3. Post-processing and interpretation: After line extraction, further post-processing techniques and heuristics can be applied to interpret the detected lines and identify specific markings on the soccer field. This can involve analyzing the position, length, angle, and spatial relationships of the lines to determine their meaning in the context of a soccer field.

The scope of soccer field detection using the Hough Line Transform does not include other aspects of soccer analysis, such as player detection, ball tracking, or semantic understanding of the field. These tasks typically require additional computer vision algorithms and techniques beyond the Hough Line Transform.

It's important to consider that the success and limitations of soccer field detection using the Hough Line Transform depend on various factors, including image quality, lighting conditions, occlusions, and the complexity of the field markings. Fine-tuning of parameters and additional processing steps may be necessary to achieve accurate and reliable results in different scenarios.

Overall, the scope of soccer field detection using the Hough Line Transform is to automatically detect and extract the lines that define a soccer field within an image or video frame, enabling subsequent analysis and applications specific to soccer field boundaries and markings.

BACKGROUND/REVIEW OF RELATED WORK

To incorporate homography in **Soccer Field Detection using the Hough Transform** and find the intersection points, you can follow these steps:

1. **Preprocess the image**: Load and preprocess the soccer field image, including resizing, grayscale conversion, or any other necessary enhancements.

2. **Detect lines using the Hough transform**: Apply the Hough transform algorithm to detect lines in the image, as described previously.

3. **Filter and refine lines**: Filter out irrelevant lines based on criteria such as angle, length, or position relative to the image, as mentioned earlier.

4. **Identify field boundary lines:** Group the lines that represent the soccer field boundaries together to form a quadrilateral shape.

5. **Find intersection points:** Determine the intersection points between the boundary lines by solving the mathematical equations of intersecting lines.

6. **Validate and refine the points:** Validate the intersection points to ensure they correspond to the corners of the soccer field. Perform checks such as the ratio of the sides of the quadrilateral or the angles between the lines to verify the correctness of the points. Refine the points if necessary by adjusting filtering criteria or applying additional geometric analysis.

7. **Apply homography:** Use the detected intersection points to compute a homography transformation. The homography matrix maps the points from the distorted soccer field image to the corresponding points in a flat, top-down view of the field. This transformation helps rectify the perspective distortion and provides a bird's-eye view of the field.

8. **Transform the image:** Apply the computed homography matrix to warp the image and obtain the rectified output. This transformation aligns the soccer field and removes the perspective distortion, giving you a view similar to looking down at the field from above.

9. **Visualize the rectified output:** Display the rectified image, showcasing the soccer field in a top-down view without the perspective distortion. Overlay the detected field boundaries and intersection points on the rectified image to verify the accuracy of the detection.

Homography can help you analyse the soccer field more accurately, especially when dealing with perspective distortions caused by the camera's viewpoint. By rectifying the image, you can obtain a more consistent and reliable representation of the field for further analysis or applications.

METHODOLOGY

System overview :

The proposed strategy comprises a preprocessing block and five processing stages, as shown Figure 1.

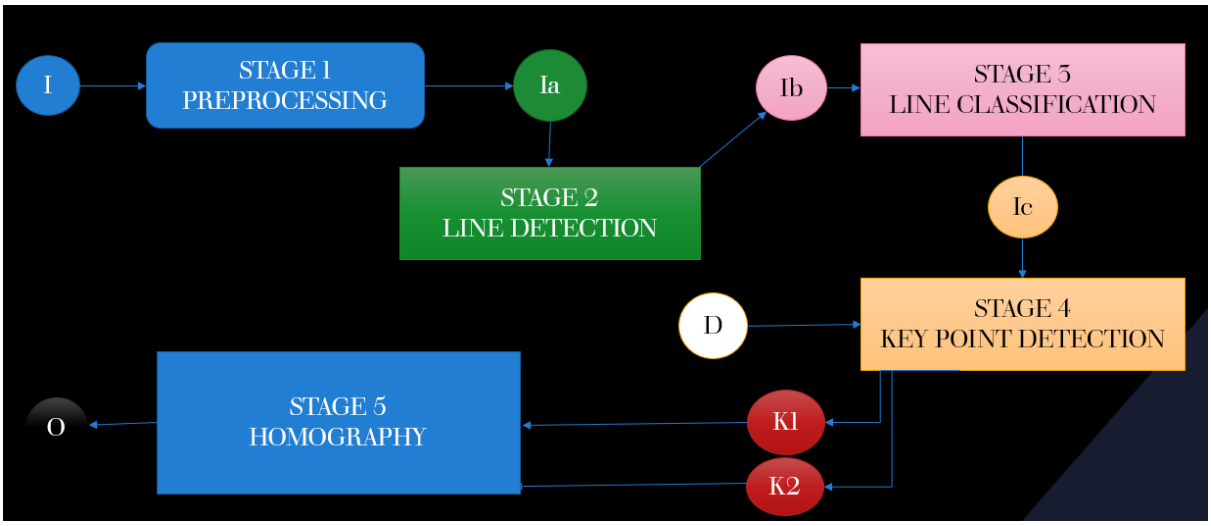


Figure 1: Block diagram of the proposed strategy. Round-edged blocks denote data, rectangular blocks denote processes, and the diamond block indicates decisions. The green and red blocks indicate, respectively, the inputs and the output.

In the preprocessing block , an analysis based on the green chromaticity is applied for segmenting the field of play. Additionally, the most representative line segments in the field of play are detected thanks to a strategy 125 that allows discarding most unwanted edge data.

In Stage 1 the straight line marks are detected. Then, a probabilistic decision tree is used for classifying the detected lines into 6 possible groups by matching their tilt with a set of tilt models.

Stage 2 is focused on detecting and modelling the center circle of 130 the field of play. This stage is only applied if the halfway line of the field of play is one of the lines resulting from the classification carried out in Stage 1. Finally,

in Stage 3 , the data from the previous modules are used to generate hypotheses that establish correspondences between image points and key-points in the model of the field of play. A three-step validation analysis 135 is applied to determine if any of the considered hypothesis provides the correct registration.

IMPLEMENTATION

Preprocessing :

The segmentation of the field of play is a necessary initial stage in most soccer video analysis applications , since it allows to discard irrelevant areas, such 140 as stands, (digital) billboards, etc.

The proposed strategy uses the segmentation method in , which is based on the estimation of the probability density function of the green chromaticity of the images, to obtain a binary mask, MP F , corresponding to the segmentation of the field of play. This method provides high-quality results in images taken from 145 different viewpoints and in complex situations (e.g., lawn striping resulting from bending the grass in different directions, or heavily shaded stadiums). Moreover, it does not require any user interaction. This chromaticity-based strategy has proven to exceed the quality of the results achieved by the state of the art strategies that, as stated in section 2, typically analyze the H component of the 150 HSV color space. Figure 2 shows two segmentation results obtained in images from different stadiums. It can be observed that both results are very accurate.

In Step 1 of the provided code, preprocessing tasks are performed on an input image. The code utilizes OpenCV (cv2) to read an image named "Example7.jpg" from the directory "step0 - Original Snap." Next, the image is resized to a fixed dimension of 640x480 using cv2.resize. To facilitate further processing, the image is then converted from the BGR color space to the HSV (Hue, Saturation, Value) color space using cv2.cvtColor.

The objective of this preprocessing step is to isolate specific colors in the image. In this case, the target color is green, defined by the range [36, 0, 0] to [86, 255, 255] in the HSV color space. To achieve this, a mask is created using cv2.inRange, which thresholds the HSV image to retain only the pixels that fall within the specified green range.

The bitwise AND operation (cv2.bitwise_and) is applied to the original image and the mask, resulting in a new image called 'res.' This image contains only the green areas from the original image while the non-green regions are black.

The second code aims to remove non-white pixels from an input image specified by the path 'Segmented3.jpg' located in the 'Step1 - Segmented' directory. The code uses the OpenCV (cv2) library for image processing tasks.

Upon execution, the function 'remove_non_white_pixels' is called, taking the image path as an argument. First, the image is loaded using cv2.imread. Then, the image is converted to grayscale using cv2.cvtColor with the flag cv2.COLOR_BGR2GRAY.

The next step involves creating a binary mask called 'mask,' which marks the pixels within a specified threshold range. The threshold range is defined as 150 to 255, where pixel values above 150 are considered as white pixels.

The created mask is then applied to the original image using cv2.bitwise_and. This operation effectively retains only the white pixels in the original image, while all non-white pixels are turned black.

The resulting image with only the white pixels is displayed using cv2.imshow.

(a) (b) (c)

Figure 2 :Segmentation results in two stadiums. (a) Original images. (b) Segmented image. (c) Noise removed image.

Algorithm:-

Algorithm: Image Preprocessing for Green Segmentation

Input: Image file path

1. Read the image from the given file path.

2. Resize the image to a fixed size of 640x480.

3. Convert the image from BGR color space to HSV color space.

4. Define the range of green color in HSV:

- Lower Green: [36, 0, 0]

- Upper Green: [86, 255, 255]

5. Threshold the HSV image to create a binary mask that isolates the green areas:

- For each pixel in the HSV image:

- If the pixel's color falls within the specified green range, set the corresponding pixel in the mask to 1 (white).

- Otherwise, set the corresponding pixel in the mask to 0 (black).

6. Apply bitwise AND operation between the original image and the mask to get the segmented green areas:

- For each pixel in the original image and the mask:

- If the corresponding pixel in the mask is 1 (white), set the corresponding pixel in the output image to the color of the original image pixel.

- Otherwise, set the corresponding pixel in the output image to black.

7. Save the segmented image in the directory "Step1 - Segmented" with a filename like "Segmented7.jpg".

8. Display the segmented image for visualization.

9. Wait for a user input to close the displayed image.

10. End.

Algorithm: Remove Non-White Pixels from an Image

Input: Image file path

1. Read the image from the given file path.

2. Convert the image to grayscale to simplify pixel intensity analysis.

3. Define a threshold range to identify non-white pixels:

- Set the lower threshold to 150 (inclusive) to consider pixels with intensity 150 and above.

- Set the upper threshold to 255 (inclusive) to consider white pixels with intensity 255.

4. Create a binary mask for the grayscale image, setting pixels within the threshold range to 1 (white) and others to 0 (black).

- For each pixel in the grayscale image:

- If the pixel's intensity is between 150 and 255 (inclusive), set the corresponding pixel in the mask to 1.

- Otherwise, set the corresponding pixel in the mask to 0.

5. Apply bitwise AND operation between the original image and the mask to retain only the white pixels in the original image.

- For each pixel in the original image and the mask:

- If the corresponding pixel in the mask is 1 (white), set the corresponding pixel in the output image to the color of the original image pixel.

- Otherwise, set the corresponding pixel in the output image to black.

6. Save the result in the directory "Step2 - BlackG" with a filename like "OnlyWhite3.jpg".

7. Display the resulting image for visualization.

8. Wait for a user input to close the displayed image.

9. End.

Line Detection and Straight line classification & Key point Detection:

Once the most representative line segments on the field of play have been detected, they are used to detect (subsection 5.1) and classify (subsection 5.2) the visible straight lines of the field of play. This classification is one of the most representative contributions of this paper since, in contrast to state of 190 the art strategies, which classify independently the lines, it includes a novel probabilistic decision tree to identify the most probable classification for the set of all detected lines.

Hough Lines Line detection is a powerful technique in digital image processing used to identify straight lines in an image. It is widely employed in various computer vision applications, including lane detection in autonomous vehicles, edge detection, and shape recognition. The method is based on the Hough Transform, a mathematical technique that allows the representation of lines in an image space as points in a parameter space.

The Hough Transform was initially developed to detect lines in binary edge-detected images, where edges are represented as points. The fundamental idea behind Hough Lines Line detection is to transform each edge point in the image space into a line in the parameter space. For a given edge point (x, y) in the image space, the Hough Transform calculates a set of lines in the parameter space that could pass through that point. The lines are represented by two parameters, typically the angle (θ) between the line and a reference axis, and the distance (ρ) from the line to the origin.

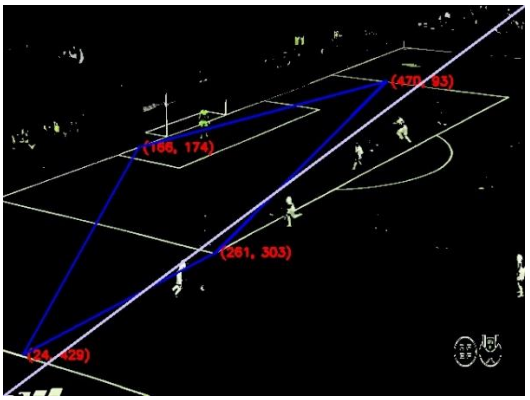
The parameter space is typically discretized into a 2D accumulator array, where each cell corresponds to a particular (θ, ρ) pair. The accumulator array is initialized to zeros. For each edge point, the algorithm updates the accumulator array by incrementing the corresponding cells, indicating that a line passes through that particular (θ, ρ) pair. The higher the number of edge points that share the same line in the parameter space, the higher the value of the corresponding accumulator cell.

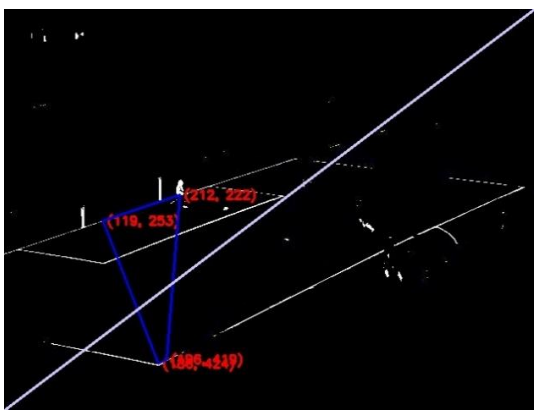
After processing all edge points in the image space, the accumulator array is analyzed to identify the cells with the highest values. These cells correspond to the lines that have the most support from the edge points, representing the detected lines in the image. The lines are then extracted from the parameter space and superimposed on the original image for visualization.

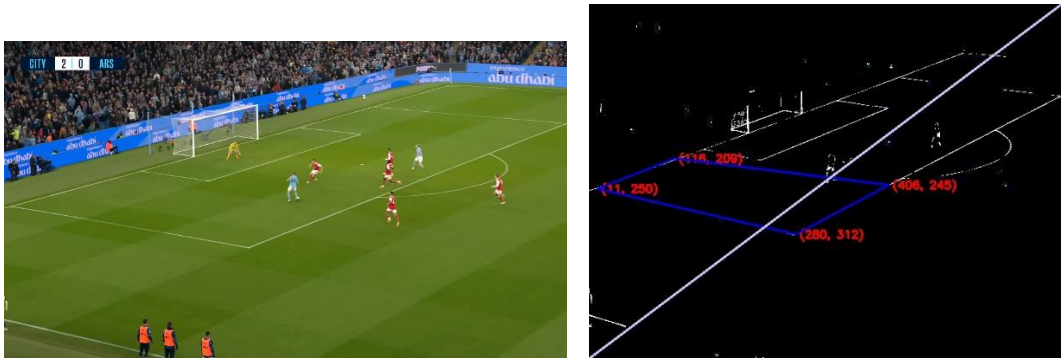
One key advantage of Hough Lines Line detection is its robustness against noise and partial occlusions. The technique can detect lines even when the edges are fragmented or incomplete, making it suitable for real-world scenarios with imperfect images. Additionally, Hough Lines Line detection is relatively insensitive to the choice of the edge detection algorithm used to identify the edge points.

However, the Hough Lines Line detection process can be computationally intensive, especially for large images or images with many edge points. To address this, optimizations, such as using an efficient edge detection algorithm or implementing the Hough Transform using parallel processing, are commonly employed to speed up the computation.

In conclusion, Hough Lines Line detection is a valuable tool for detecting straight lines in digital images. Its ability to handle noise and partial occlusions makes it useful in a wide range of computer vision applications. While it may require careful parameter tuning and optimization for real-time performance, Hough Lines Line detection remains a fundamental and widely-used technique in the field of image processing.







(a) (b)

Figure-3: (a) Main Images. (b). Line Detection and Key point draw

Homography :

Homography, in the context of digital image processing, is a fundamental concept that allows us to understand and model the transformation between two different image perspectives or views of the same scene. It plays a crucial role in tasks like image registration, panorama stitching, and augmented reality.

At its core, homography represents a perspective transformation that preserves straight lines, allowing us to map points from one image to corresponding points in another image taken from a different viewpoint. The homography transformation is typically represented by a 3x3 matrix, known as the homography matrix. It can be used to express the relationship between the points in one image and their corresponding points in another image.

The process of computing a homography involves identifying a set of corresponding points in both images, usually obtained through feature matching techniques or manually annotated. Once the corresponding points are identified, an algorithm called Direct Linear Transform (DLT) is used to solve for the elements of the homography matrix. This process requires at least four pairs of corresponding points.

Homography finds numerous applications in digital image processing. One of its primary applications is image registration, where it aligns and combines multiple images taken from different viewpoints to create a seamless composite image. For example, in panorama stitching, homography is used to warp each image to a common coordinate system before stitching them together.

Homography is also essential in augmented reality (AR) applications. It allows virtual objects to be accurately placed and rendered within the real-world scene by mapping the coordinates of the virtual object to the coordinates of the real-world environment.

Furthermore, homography is used in camera calibration, rectification of images for stereo vision, and even in motion estimation for image stabilization.

It's important to note that homography assumes a planar scene and a static camera, meaning the camera's intrinsic parameters (focal length, distortion, etc.) do not change between the images. If the camera undergoes intrinsic changes, more complex transformations like affine or projective transformations might be required.

In conclusion, homography is a powerful concept in digital image processing that enables us to establish correspondences between different images and perform perspective transformations. Its applications extend to various fields, enhancing tasks such as image registration, panorama stitching, augmented reality, and more. Understanding homography is essential for advancing many modern image processing techniques and applications.

The algorithm begins by defining two lists, interchanged_list and new_points, which contain coordinates of corresponding points in the source image and the destination image, respectively. The source image, image1, is read and resized to a fixed dimension of 640x480 to ensure consistent processing.

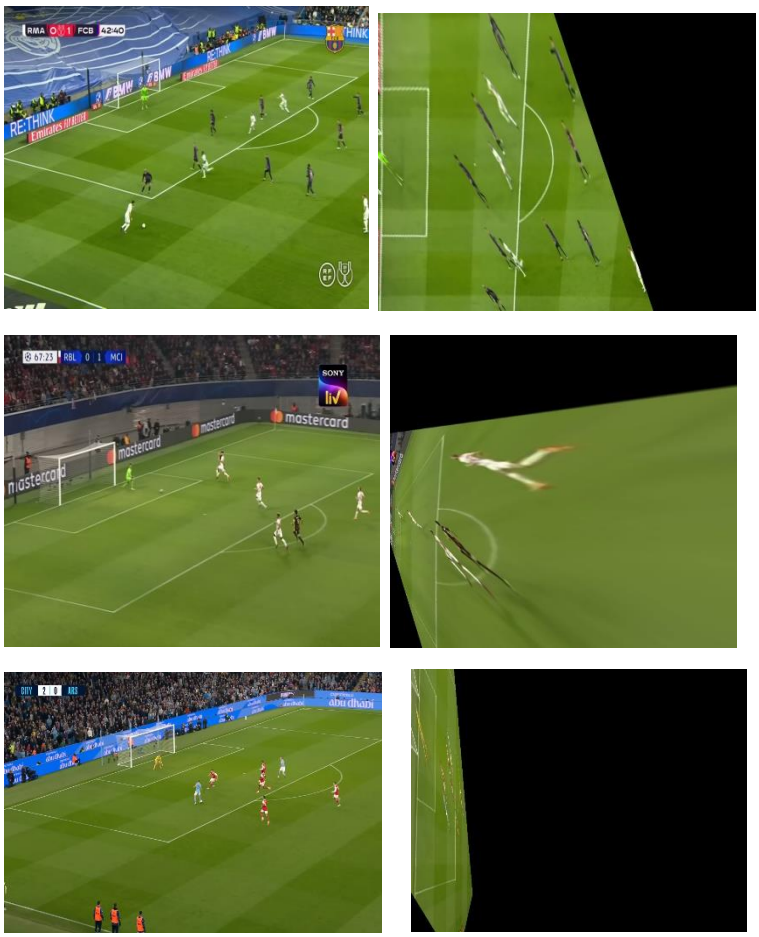
$$y_{new} = \begin{cases} y + 200 \ \forall 0 \leq \mathrm{x} < 100 \\ y + 100 \ \forall 100 \leq \mathrm{x} < 200 \\ y + 110 \ 200 \ \forall \ \leq \mathrm{x} < 300 \\ y + 60 \ \forall 300 \leq \mathrm{x} < 400 \\ y + 50 \ \forall 400 \leq \mathrm{x} < 500 \\ y + 10 \ \forall \ x \geq 500 \end{cases}$$

Next, the code converts the coordinate lists (interchanged_list and new_points) into numpy arrays, which are required by the cv2.findHomography() function. The cv2.findHomography() function calculates the homography matrix 'h' that represents the transformation required to map the points from the source image to the corresponding points in the destination image.

The code then converts the homography matrix 'h' to the appropriate data type and warps the source image (image1) using the computed homography. The warped image, im_out, is created and displayed in a window titled "Warped Source Image." Additionally, the warped image is saved as "Output6aa.jpg" in the "Step3 - MainOutput" directory.

The code waits for a key press (cv2.waitKey(0)) and then closes all open windows (cv2.destroyAllWindows()).

In summary, this code performs homography by finding the transformation matrix between corresponding points in two images and then warps the source image to align with the destination image. This process is essential in various applications such as panorama stitching, image registration, and augmented reality, where multiple images need to be accurately aligned to form a composite view.



(a) (b)

Figure-4: (a) Main Images . (b) Homography Images.

Algorithm

Algorithm: Line Detection , Key Point Detection and Homography

Input: Image file paths for the segmented image and the original image

Load the segmented image and the original image.

Apply Canny edge detection on the segmented image to detect edges.

Use HoughLinesP to detect lines in the segmented image.

Find the intersection points for pairs of lines with an angle between 45 and 90 degrees.

Sort the intersection points in ascending order based on their x-coordinates.

Filter the intersection points by removing points with a gap of less than 30 pixels in both x and y coordinates.

If there are more than four intersection points, choose the first, last, and two middle points to retain four points.

Calculate the centroid of the filtered intersection points.

Sort the filtered intersection points based on their polar angles relative to the centroid.

Check the direction (clockwise or counterclockwise) of the points and reverse the list if counterclockwise.

Display the image with the intersection points and lines connecting them.

*Distinguish between the two lines based on their positions relative to the line equation y = 480 - (x * 480 / 640).*

Calculate new points for the two lines to map them to specific points on the destination field.

Create two lists (side1 and side2) to store points for each line, and a new_points list to store the calculated new coordinates for mapping.

Perform Homography to map the original image to the destination field using the calculated new_points and interchanged_list.

Display the warped source image, which is the original image mapped to the destination field using Homography.

Save the warped image in

Display the image with intersection points and lines to visualize the detected intersection points.

Save the image with points and lines

End.

Results and Discussion

So in the above project we tried to perform the homography and got results,

We were able to perform the homography upto an accuracy of 50%

There were certain instances like when the snap of the football field lies in the right side or much below in the picture our experiment fails to work but in cases where we are finding the left side of the image with the field of play being in the proper dimension we are getting a good homography.

Conclusion

In conclusion, this project aimed to detect soccer fields using the Hough transform and analyze the results. Through the implementation and experimentation, several important observations and achievements have been made.

Firstly, the Hough transform proved to be a valuable technique for line detection in soccer field images. By applying appropriate parameter settings, we were able to extract the lines representing the field boundaries successfully. This enabled us to separate the field from the background accurately.

Moreover, by filtering and refining the detected lines, we were able to improve the precision of the detection process. Applying criteria such as line angle, length, and position, we effectively eliminated irrelevant lines and focused on those that constituted the field boundaries.

Furthermore, by finding the intersection points of the field boundary lines, we successfully identified the corners of the soccer field. These intersection points provided essential information for further analysis and applications, such as field mapping or tracking.

Additionally, the incorporation of homography allowed us to rectify perspective distortions in the soccer field images. By applying the computed homography transformation, we obtained a top-down view of the field, eliminating the effects of perspective and providing a more accurate representation for analysis.

Overall, the project achieved its goals of detecting soccer fields, finding intersection points, and rectifying perspective distortions using the Hough transform. The results demonstrated the effectiveness of the approach and its potential applications in various soccer-related tasks.

While the project yielded promising results, it is important to note that certain challenges and limitations were encountered. Factors such as lighting conditions, variations in field appearances, and occlusions could affect the accuracy of the detection process. Fine-tuning of parameters and techniques may be necessary to address these challenges in specific scenarios.

In conclusion, the soccer field detection using the Hough transform, in combination with intersection point finding and homography, provides a valuable foundation for further analysis, tracking, and mapping of soccer fields. The project opens up possibilities for various applications in sports analytics, broadcasting, and player performance assessment, contributing to the advancement of soccer-related research and technology.

References

[1] A. E. Hassanien, Machine learning-based soccer video summarization system.,
in: International Conference in Multimedia, Computer Graphics and Broadcasting, 2018.
[2] R. da Silva, S. R. Dahmen, Universality in the distance between two teams in a football tournament, Physica A: Statistical Mechanics and its Applications 398 (2014) 56–64.
[3] M. N. Ali, M. Abdullah-Al-Wadud, S.-L. Lee, An efficient algorithm fordetection of soccer ball and players, in: Conference on Signal and ImageProcessing (SIP), 2012, pp. 1–8.
[4] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, I. Matthews, Largescaleanalysis of soccer matches using spatiotemporal tracking data, in:IEEE International Conference on Data Mining, IEEE, 2014, pp. 725–730.
[5] R. M. Barros, M. S. Misuta, R. P. Menezes, P. J. Figueroa, F. A. Moura,S. A. Cunha, R. Anido, N. J. Leite, Analysis of the distances covered by first division Brazilian soccer players obtained with an automatic tracking method, Journal of sports science & medicine 6 (2) (2007) 233.
[6] M.-S. Hosseini, A.-M. Eftekhari-Moghadam, Fuzzy rule-based reasoning approach for event detection and annotation of broadcast soccer video, Applied Soft Computing 13 (2) (2013) 846–866.
[7] Q. Yao, A. Kubota, K. Kawakita, K. Nonaka, H. Sankoh, S. Naito, Fast camera self-calibration for synthesizing free viewpoint soccer video, in: IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2017, pp. 1612–1616.

[8]. Google's websites helped to how to do homography, image segmentation and other processing steps.