Andrew Fitzsimmons
11/20/2022
CSC481

Final Project
Hockey Puck Goal Detection - Final Paper

Abstract
The goal of this project was to evaluate whether or not a goal is being scored in an image of a hockey game. A variety of image processing techniques were used to interpret objects and their locations within the image in order to assess whether a goal was scored. These techniques include hue segmentation, histogram segmentation and thresholding, blurring, and polynomial regression. Ultimately, my program was able to successfully detect and identify a goal/no-goal condition in both photos that I used for interpretation. However, the camera angle and lens used for the image capture presents some challenges for reliability when detecting puck location at different heights in the 'z direction'. The object detection through contour interpretation is also vulnerable to selecting a non-puck object as the object that is used to represent the puck in evaluation.

Introduction
My final image processing project was to create a program that would interpret whether or not a goal was being scored in a hockey game based on an image taken from an in-goal camera. Figure 1 below is an example of the type of photo that was input into the program for goal determination. Using image processing techniques like hue segmentation, histogram segmentation and thresholding, blurring, and polynomial regression, I was able to create a program that would reliably interpret whether or not a goal was being scored, although it did have some limitations due to the techniques that were utilized. Automated goal detection in hockey, and other sports, is important because it removes the need for human judgment when making determinations on whether a scoring event occurred in sports. Whether we use automation to make the primary judgment on a scoring play, or simply as a redundancy to the referees call on the field, sports in general are moving more and more towards automated event detection methods. Major League Baseball for example, will be implementing automated ball/strike calls in 30 AAA parks (Minor Leagues) for the 2023 season. Replacing human judgment with programmatic decision making is becoming a trend in all sports, especially for plays that are repetitive and have a clear basis on which a decision is made (ball or strike, goal or no goal, etc).

Figure 1 - Positive ‘goal’ image

Background

The most similar implementation that I could find on the internet of automated goal line detection was in a 2013 paper titled “Non-Invasive Soccer Goal Line Technology” [1]. My program, for ease of programming implementation, utilizes only one ‘in goal’ camera view to make determinations on goal or no goal. In the “Non-Invasive Soccer Goal Line Technology” paper, they describe using 3 cameras per goal (6 cameras total on one field) in order to capture and interpret images. For each goal, a camera is positioned in line with the goal line for a side profile ‘depth’ view of the ‘goal event’. A single camera is also placed behind the goal, perpendicular to the goal line for better visibility to the z direction and the side to side location (axis running parallel with the goal line). Images from each of these 3 cameras are interpreted together in order to estimate the ball location in space (Figure 2), and ultimately determine whether a goal has occurred based on the boundaries of the goal posts and the cross bar.

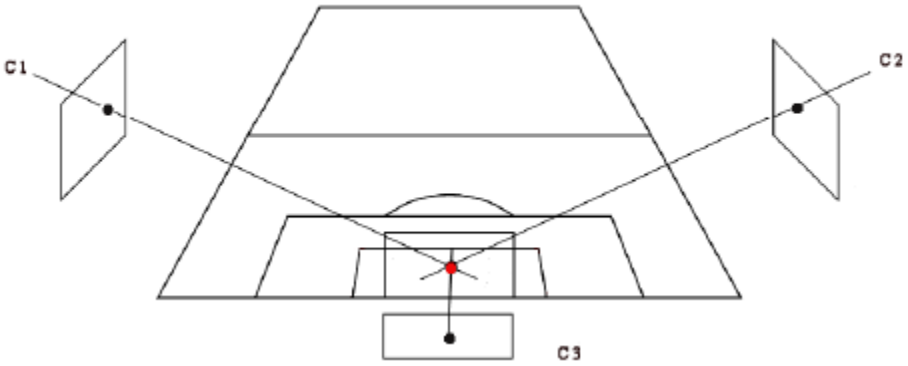

Figure 2 - The intersection of 3 camera projection lines produces the estimated ball position [1]

Methods

Below is a summary of my code for each of the primary functions that I wrote to perform the image processing tasks required for goal detection. My program was built in python, and utilized functions from the openCV image processing package.

Function 1 - cropImage()

Input: Full Image of Hockey Goal (Figure 1)

Methods Used: Row selection from original image matrix

Output: A cropped image showing only the 'evaluation area' for goal/no-goal detection. See Figure 3 below.

Description: The first function that is run that has a significant impact on the evaluation of the image is the cropImage() function. This function is simple in that it is just selecting a predefined range of row values within the image matrix. This is important because when attempting to establish object locations and goal boundaries, objects outside of the goal area that may be present in the original image can generate noise and lead to a higher rate of incorrectly detecting objects within the image.

Figure 3 - A cropped version of Figure 1 showing only the evaluation area for the goal in order to reduce noise caused by other objects in the larger image.

Function 2 - findGoalLine()

Input: A cropped image showing only the 'evaluation area' for goal/no-goal detection. See Figure 3.

Methods Used: Hue Segmentation, Polynomial Regression

Output: A polynomial regression model that describes the inside edge of the goal line.

Description: The findGoalLine() function is critical to goal/no-goal detection because it establishes a model representation of the goal line in Figure 3. I chose to use polynomial regression in this case due to the distortion of the goal line by the fisheye lens that was used to capture the image. The first step to building the polynomial regression model that defines the goal line

boundary is to perform hue segmentation on the original image so that you get a binary representation of all the ‘red’ objects in the image, which in this case corresponds to the pixels that make up the goal line (Figure 4).

Figure 4 - Hue segmentation on ‘Red’ objects in the evaluation image

Now, with all of the red pixels located in our image, I selected the highest row value ‘red’ pixel (pixel rows increase from 0 to N rows from top to bottom of the image) at each x value in the image (or for each column in the image matrix). The points that resulted from this selection represented the bottom most edge of the goal line which is what I ultimately used to generate a polynomial model of the goal line shown below in Figure 5.

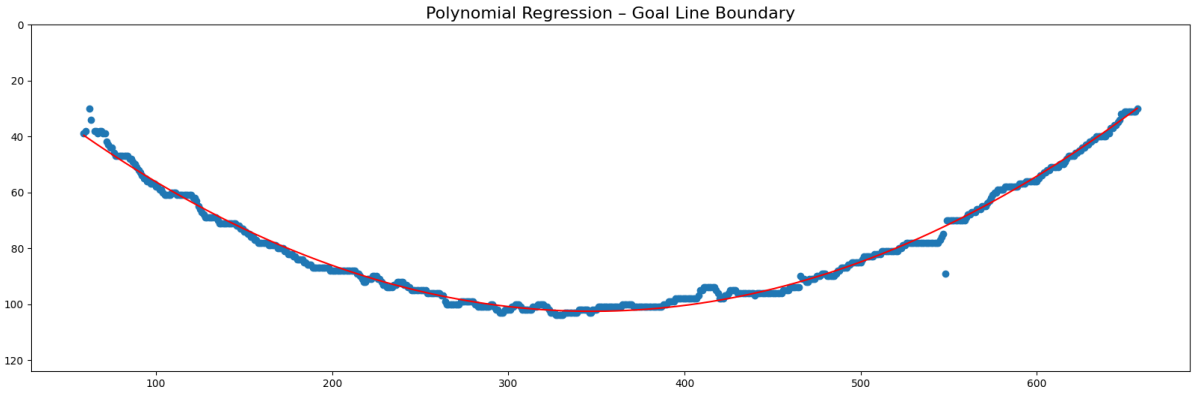

Figure 5 - Polynomial regression line fit to goal line boundary points

Function 3 - findPuck()

Input: A cropped image showing only the 'evaluation area' for goal/no-goal detection. See Figure 3.

Methods Used: Grayscale Thresholding/Histogram Segmentation, findContours()

Output: Image with Max Area contour identified (ideally this represents the hockey puck). Figure 7 below.

Description: The first step in the puck detection process is to perform thresholding of the grayscale image based on histogram segmentation values. On a negative representation of the grayscale image (so black appears as white), I created a threshold that made all pixels with intensity values > 220 appear as white, and everything else represented as black. Figure 6 below shows the resulting thresholded image.

Figure 6 - Grayscale Thresholding/Histogram Segmentation to identify the darkest black objects in the evaluation image

With the thresholded representation of the image, I was then able to run the 'findContours()' function from the openCV package. This function identifies large area groupings of white pixels, essentially looking for dense clusters of the same color pixel. Then, I select the contour object from the resulting set that has the largest overall area. In my testing, this usually returned the object that represented the puck (Figure 7), but this method can be vulnerable to picking up other large, dark objects in the image and misinterpreting them as the hockey puck.

Figure 7 - The Max Area contour object is selected as a representation of the hockey puck

Function 4 - getPuckCoordinates()

Input: Image with Max Area contour identified (ideally this represents the hockey puck). Figure 7.

Methods Used: RGB Segmentation

Output: Vectors representing X and Y coordinates of the points that make up the top edge of the hockey puck.

Description: Since we outlined our hockey puck object in green against the black and white thresholded image, it was fairly straightforward to iterate across the pixel values in the image and identify the x and y coordinates of every point that was 'green'. These points are critical since they represent the location of the hockey puck in space. We only record the lowest row value pixels (lower rows are higher up in the observed image in terms of coordinate values) since the top edge of the hockey puck is really what we're interested in if we want to know that the whole puck is within the goal line.

Function 5 - determineGoal()

Input: X and Y vectors for the hockey puck edges, polynomial model for the goal line

Methods Used: Polynomial Regression/Prediction

Output: return True if goal is detected, return False if no-goal is detected.

Description: This final function was fairly straightforward to determine whether or not a goal was scored in the image. Using the polynomial model of the goal line, I predicted the model y value of each point in the hockey puck x vector. If any predicted y value was higher than the associated value in the hockey puck y vector, then that would indicate at least a portion of the puck is outside the goal line and would return no goal. If no points fail the model prediction test, the function evaluates the image as a goal.

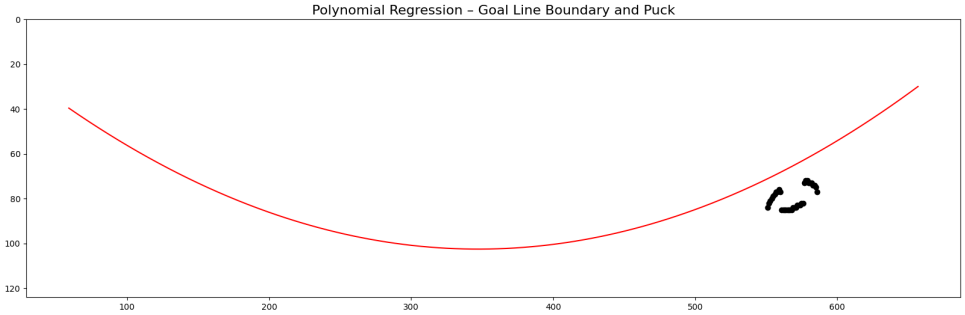

Figure 8 - Projection of hockey puck top edges against goal line model regression line

Results and Conclusion

I tested my program on two images from the same game, one represented a goal, the other represented a no-goal. The program was able to successfully determine goal line location, identify the hockey puck location, and make a correct assessment on whether or not the entire puck had crossed the goal line for both images.

There are, however, a few drawbacks to my program in the way that it detects objects for evaluation. If a similar program was going to be implemented for practical in-game use, a few challenges would need to be solved:

1) The camera angle in the images provides an artistic and dynamic visual of the inside of a hockey net during a goal scoring event. Unfortunately, the angle of the line of sight of the image, combined with the fisheye lens that was selected, result in distortion of the image and make it difficult to interpret depth. A solution to this problem might be to mount a strip of cameras in an array on the crossbar of the net, looking directly down. The new orientation of the camera would result in a more clear interpretation of puck depth in the net, losing some information regarding the height that the puck is off the ice, but that is not used for evaluation since the camera location would set the new max height. The array could be stitched together to give a flat 2D representation of the goal line with no distortion.

2) Obstructions when detecting the hockey puck object could result in an incorrect object being identified as the puck. This could be resolved potentially with a sensor in the puck itself, or perhaps redundant camera angles that could all evaluate puck location separately and compare locations in order to ‘agree’ on where the puck is in space.

References

[1] Spagnolo, Paolo, et al. *Non-Invasive Soccer Goal Line Technology: A Real Case Study*. https://www.researchgate.net/publication/261498240_Non-invasive_Soccer_Goal_Line_Technology_A_Real_Case_Study.

[2] Jalled, Far´es, and Ilia Voronkov. *Object Detection Using Image Processing*. https://arxiv.org/pdf/1611.07791.pdf.

[3] Vats, Kanav, et al. *Puck Localization and Multi-Task Event Recognition in Broadcast Hockey Videos*. https://openaccess.thecvf.com/content/CVPR2021W/CVSports/papers/Vats_Puck_Localization_and_Multi-Task_Event_Recognition_in_Broadcast_Hockey_Videos_CVPRW_2021_paper.pdf.

[4] Srinivas, P., et al. *Image Processing Edge Detection Technique Used in Traffic Control Problem*. https://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.7692.

[5] Li, Min, et al. “Color Image Segmentation Using Adaptive Hierarchical-Histogram Thresholding.” *PLOS ONE*, Public Library of Science, https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0226345#sec003.