Adaptive Augmented Reality for Competitive Swimming

Adam A. Oxner
University of Michigan
Ann Arbor, MI 48104
adoxner.com/contact.html

***Abstract*—This article outlines a method for finding a competitive swimming pool and its orientation in images that encapsulate an entire rectangular pool. This model only applies to cameras with limited radial distortion. The end result of a successful run of this algorithm is the four corners of the pool and its orientation with respect to lane lines. To achieve this result, the solution relies on multiple applications of filters, the Canny edge detector algorithm, and Hough transforms. To find the most likely pool candidate in an efficient way, the algorithm actively discards solutions that break spacial orientation assumptions. This paper explores the algorithm in detail, presenting thoughts on the most important variables and discussing areas that could be explored further in order to achieve more accurate results. The ultimate goal of this project is to present part of a solution to the larger problem of exactly labelling a pool and its lanes for labelling or other analysis.**

Fig. 1. Televised augmented reality at the 2012 Olympics [8]

I. Introduction

Labelling pools on live broadcasts is not a new idea: pools have been labelled on television broadcasts since the 1990's. Uses can include labelling lane number and swimmer placement, tracking swimmers, and displaying records as lines being chased by competitors. This technology has been inaccessible to the general public, however, because of expensive setups: visual markers, defined camera positions, etc. An example of televised augmented reality can be found in Figure 1.

The purpose of this project is to start developing a framework so that an average user can use this augmented reality technology themselves. The scope of this application is limited to finding and outlining a pool within a static image, and determining its orientation with respect to lane direction in order to label the lanes. This turned out to be a large enough task in itself, due to some challenges explained below.

Some possible applications of highly accurate pool and lane detection for the consumer market include augmented reality on mobile devices to help spectators find swimmers and compare paces in real-time, easier post-race analysis for swimmers and coaches, and increased availability to low-budget production outfits.

Besides the problem of large variability of input seen by the entire computer vision community, one problem that particularly plagues the detection of pools is the amount of visual noise found around natatoriums. Not just random noise, but noise that ends up matching up closely to features important in the detection of the pool: the color blue around the pool and a huge amount of lines. Lines are common around pools because of the necessity of items in the setting: tiles on the floor, walls (if indoors), stands, and backstroke flags could all become issues. Some previous work looking at solutions to similar problems can be found in section 2.

To deal with the 'line' noise, bluring and downsampling the image, using the Gaussian Filter and Pyramid, respectively, proves useful. Doing some math with RGB to bring out the blue of the pool, then setting a threshold to ignore the rest also helps ignore lines outside of the probable pool location (blue area). Dealing with 'color' noise is still a challenge, however. Measuring reflectivity of surfaces might be helpful, if difficult, in this regard. A technical discussion of this can be found in section 3, subsection B,1.

Because pools are so large, it is much more reliable to locate the edges of the pool rather than the corners. It is much more likely that a corner of a pool is obscured by an object (a person, maybe) than an entire 25 meter edge. For this reason, a Canny Edge Detector [3] in combination with Hough Transform [1], [6] for line detection was used to detect the outline of the pool. You can find a description of line-finding methods in section 3, subsection B,2. Other methods used to detect the pool outline largely involve selectively iterating through potential pool edges and comparing their geometric features to eliminate lines that are probably not pool edges. A technical discussion of this method is found in section 3, subsection B,3.

A similar method to this is used to find the lane lines in the pool. Because lane lines are often obscured, look like the lines on the bottom of the pool, the same color of the water, or not present at all, the it is more reliable to find the general direction

of all of the lane markers than find each one individually. Also, since they are the only lines in the pool and we already know the location of the pool, we don't have to worry about noise as much as when finding the pool itself. This method is described in section 3, subsection B,4.

An explanation of the experiments used to evaluate these methods can be found in section 4. The paper is concluded in section 5.

II. Related Work

A. Review of Related Work

Generalizing the problem of detecting rectangles with certain properties, we find that work focuses on *edge detection* and *line detection*.

Previous and well-known work on edge- and line-detection include Canny's work [3] and Hough Transforms's applications for finding shapes and lines [1], [6].

Other work used is Suzuki and Abe's contour-finding algorithm [11]. Their work is implimented in the OpenCV library [10] for contour detection, which this application uses.

Work on finding rectangles is applicable here. Jung and Schramm's work on using a windowed Hough Transform [9] was read into for possibilities. It uses a sliding window in the image to search for rectangles instead of searching the entire image at once. This work is applicable to things like building detection from satellite images.

Some more recent work by Bhaskar, Werghi, and Mansoori [2] compares Hough and Radon transforms in finding rectangles in images. Their experiments were focused on finding complete rectangles and found that the Radon transform was more robust to salt-and-pepper noise.

B. Comparison with Current Work

The Canny Edge Detector [3] and the Suzuki-Abe contour-finding algorithm [11] are used to find and outline the blue blob of interest (the pool). The Hough Transform applications [1], [6] are then used to outline it in a rectangular shape.

Although there is much literature on finding rectangles and shapes in images, there is not much on finding them without corners, which is necessary for this application. This project uses similar but not identical solutions, implimenting the Hough Transform to detect the shape of the pool.

The Jung/Schramm paper [9] uses an efficient and accurate method to detect relatively small rectangles that are oriented normally to the camera. This does not work in the case of the pool problem for the following reasons: the pool takes up a significant portion of the image and the angle at which it is photographed is not normal to the plane of the pool.

The Bhaskar/Werghi/Manssori paper [2] introduces the Radon transform [5] and suggests it for use to increase robustness against grain noise, but the type of noise this project is combating is not of this type. As previously stated, the noise mostly comes from color and lines. It also only looks at rectangles that are oriented normally with respect to the camera.

The work discussed in this paper creates a sythesis of the selection methods mentioned above. Arranging them and carefully adjusting their parameters, a robust solution to this problem was achieved.

III. Technical Discussion

A. Summary

1) Assumptions: To comply with limited temporal resources, some assumptions were made in order to limit the scope of this project.

The first, and most limiting, was to only consider images that completely encapsulate a pool. Competition pools are large objects to be photographing and are also somewhat rare (only a few per city at most), so the number of images that fit this criteria are limited.

Many of the images that did contain the entire pool were through a fish-eye lens, which introduced another limitation: only consider images with very limited radial distortion. This makes the goal more tangable: it is unlikely to find an image of an entire pool and have the camera be calibrated unless the experimenter takes it and calibrates the camera themselves.

Other assumptions include using a color image to detect the blue of the pool, the pool of interest being rectangular for edge detection, and the image being taken from a "reasonable" distance (i.e. no satellite photos).

2) Approach: The general approach to finding the pool is as follows:

- De-noise the image with filters
- Take advantage of pools all being blue to isolate the probable pool area
- Detect edges and lines in the image
- Find use the set of lines to find quadrilaterals that could be the pool
- Further eliminate quadrangles to find the pool candidate most likely to be the actual pool.

To find the orientation of the pool, the following steps are taken:

- De-noise the image with filters
- Detect edges and lines that are within the pool (as found previously)
- Find intersects with the edges of the pool- this information is used to determine which sides are the ends.

B. Algorithm

1) Filtering the image: First, to get rid of low-frequency lines, the source image is blurred with a Gaussian Blur. Because the source image can be of any size, a one-size-fits-all approach to the kernel size does not work. A large kernel over-blurs small images and vice-versa. Equation 1 is used to adjust for changes in image size. A simple if-statement is also used to make sure it is odd.

$$\left\lceil \frac{\min src.rows, src.cols}{10} \right\rceil \tag{1}$$

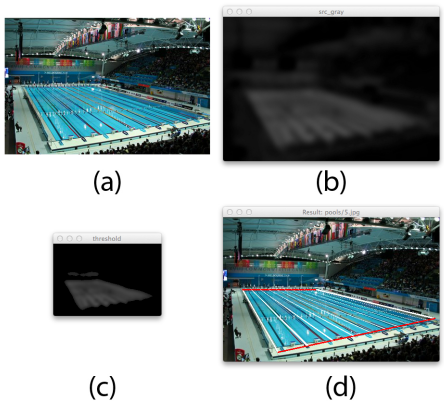

Fig. 2. The original image (a), with RGB manipulation (b), downsampled w/ binary threshold (c), and final result (d).

A Gaussian Pyramid downstep is then used to downsample the image. This further reduces noise. It also reduces computation time.

The small, blurry image is then split into RGB values. To bring out the pool, the red channel is subtracted from the green channel. You can see the result of this in figure 2(b). To ignore low-frequency noise, a binary threshold is set at the mean value plus an offset, which experiments determined should be set to 11. The mean is used because every picture has different levels. The offset is used because the mean level is generally not high enough to cut out enough low-frequency noise.

2) Finding pool candidates: After thresholding, the Canny Edge Detector algorithm [3] is implemented, whose threshold was carefully selected to be 7.2, to find the outlines of that threshold. Then, the contours of the image are found using the Suzuki-Abe contour-finding method [11]. Iterating through the contours, the largest one (by area) whose lower side of its bounding rectangle is in the lower half of the image is found. The lower-half reqirement was deduced from a large blue sky that sometimes caused erroneous results on images of outdoor pools during experimentation.

That contour is used to as input to a probabilistic Hough Transform line finding algorithm [6], [7]. The Hough threshold is set to 25 intersections, the minimum line length and maximum line gap is set using equations similar to 1, but with 30 and 80 in the denominators, respectively. The set of lines output from the transform is narrowed down to pairs of lines that (1) if extended, intersect within the bounding rectangle of the contour plus a 10% margin surrounding it and (2) are not outer lines that were perhaps detected at the edge of the image.

For convenience, this set of pairs of lines is called Set 1. Set 1 contains all possible line pairs that could form pool candidates.

3) Selecting the final pool: From Set 1, the final set of pool candidates, Set 2, is created. Each pair of lines in Set 1 is queried against another pair of lines from that same set. Together, they form a quadrangle. If the quadrangle meets a strict set of conditions, it will be added to Set 2. Those conditions are:

- All line intersects are within the bounding rectangle+10% of the largest condition-meeting contour.
- At least two of the sides are greater than or equal to one third of the largest image dimension.
- All sides are greater than one twenty-fourth of the largest image dimension.
- All of the inner angles are between 20°and 170°.
- The quadrangle must bound the moment of the qualifying contour.

The largest quadrangle in Set 2 is then selected as the pool.

These Set 2 conditions were compiled carefully through experimenting with many different inputs, which will be described further in the next section.

4) Getting the orientation: To find the orientation of the pool, the most glaring indicator of orientation is looked at: the lane lines. For this round of line detection, less blurring is needed: the blur kernel is set to 5 pixels. The rest is similar to before.

The image is split into RGB, then set to green channel subtracted from red. It is then blurred using a Gaussian blur. That image is used as input to the Canny Edge Detection [3] algorithm, whose threshold is set to 3.0. The output of the Canny algorithm is used as the input to the probabilistic Hough Transform line search function. The threshold of the transform is set to 20, the minimum line length is set using an equation similar to equation 1, but with 50 as the denominator, and the maximum line gap is set to three pixels.

The output of the Hough Transform is then filtered to only include lines where at least one end is inside the pool. Simultaneously, the number of these lines that intersect each side is counted. Each side's number of intersect is added with it's opposite side, then the two sums are compared. Whichever sum is greater is determined to be the sum from the two ends of the pool.

5) Labeling: For this project, the number of lanes is to be provided by the user. An image of lane lines and text, if desired, is dynamically created based on the user's input. In the future this could even be, if applied to video, a moving pace line.

The projective transform is then found and applied between the created image and the input image using the methods described in the Carlbom/Paciorek paper [4]. The result is the dynamically created image appearing to lay on top of the pool.

Please see figure 3 for the final output of the equation on image.

IV. Experiments

The internet was the primary source for pool images for the project. As stated in section 3, subsection A,1, most test images entirely encapsulated the pool and had minimal radial distortions. Some non-conforming images were included for comparison. The total test set size was 24 images. You can find these images attached to the document, if in digital format.

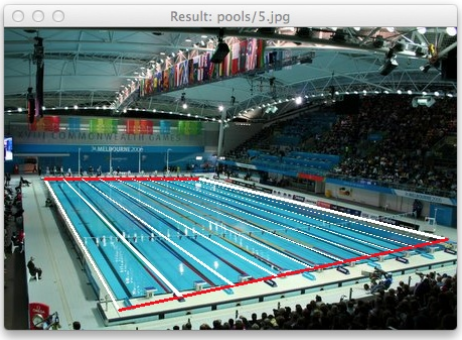

Fig. 3. Final output, including labeling. Red lines mark the pool ends, white lines divide the lanes. Input: image 5.jpg in the test set, 8 lanes.

Fig. 4. Experiments summary. Sample size: 24. Four images did not conform to the 'no radial distoration' rule.

Rules	Orientation Correct	>95% Accuracy	Oriented & Accurate	No detection
Strict	70%	60%	55%	10%
None	58%	50%	45%	25%

Much time during experimentation was devoted to finely tweaking things like the blur kenel (equation 1) and releted eqautions, Canny thresholds, and Hough transform parameters. The method was simple: tweak a variable, run through each image, and determine if the detection accuracy was improved on average. The changing of variables ended up being similar to binary search for the perfect value, except the variables were not independent. For example: if the blur kernel size was reduced, the bounding rectangle's margin needed to shrink in order to reduce false positives. This took time.

The other section that needed similar tweaking was quadrangle selection. The rules and values in section 3, subsection B,3 were adjusted and added throughout experimentation.

Because the test set was small and time was limited, there is still room for improvement in the above areas. A plurality of development time was devoted to small tweaks of these values.

Two quantitative qualities were measured for each image: orientation correctness (y/n) and normalized root mean square error (NRMSE) of the detected corners of the pool vs. manually picked corners. Please see equation 2 for the specific NRMSE equation used. Here, p hat is the predicted corner, p is the manually chosen corner, n is the number of corners (always 4), and the denominator is the diagonal distance of the image- this normalizes the error to image size.

$$\frac{\sqrt{\frac{\sum_{i=1}^{n} |p-\hat{p}|^2}{n}}}{\sqrt{src.rows^2 + src.cols^2}} \tag{2}$$

A summary of experimental results can be found in figure 4. For full experimental results, please consult the appendix.

V. Conclusions

Pools are a weathy source of organized information. People are divided into lanes, distances are well-known, and objects rarely enter or leave the pool. These properties make pools a perfect place for computer vision to address. Challenges arise due to the nature and variability of the setting: natatoriums are visually noisy spaces and pools can come in many shapes, sizes, and orientations.

This solution is one step toward a fully-implemented, open-source solution pool detection and labeling system. The application described in this paper helps with an important first step: locating the pool and finding its orientation. It does this with a fair amount of accuracy: greater than 50

With further refinement and more experimentation, more accurate and consistent results can be achieved. Specific areas the author believes have significant room for improvement are the filtering/line-finding methods (seciton 3, subsection B,1-2) and quadrangle filtering (section 3, subsection B,3) algorithms.

Topics related to pool detection project that deserve further investigation include detecting pools in images that do not fully encapsulate them and applying this method to moving images.

Appendix

Please find all relevent code and test images attached to this report, if in digital format. It can also be found at: https://github.com/adoxner/OpnSwim.

Exhaustive Experiment Results*:

Image	Orientation	NRMSE (%)
1	y	15.54
2	y	1.24
3	y	4.19
4	y	3.64
5	y	1.61
6	y	1.64
7	y	2.32
8	-	-
9	-	-
10	y	0.44
11	y	8.57
12	-	-
13	n	29.80
14	n	4.99
15	y	1.52
16	y	2.07
17	-	-
18	n	15.94
19	-	-
20	-	-
21	y	2.24
22	n	6.68
23	y	2.69
24	y	6.28

*Images where no pool is detected are marked by '-'

ACKNOWLEDGMENT

The author would like to thank Professor Matthew Johnson-Roberson at the University of Michigan for his oversight and continued consultation regarding the problem.

REFERENCES

[1] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):11-122, 1981.

[2] H. Bhaskar, N. Werghi, and S.A. Mansoori. Combined Spatial and Transform Domain Analysis for Rectangle Detection. *Information Fusion (FUSION)*, 13:1-7, 2010.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679-698, 1986.

[4] I. Carlbom and K. Paciorek. Planar Geometric Projections and Viewing Transformations. CSUR 10(4):465-502, 1978.

[5] C.L. Luengo Hendriks M. van Ginkel and L.J. van Vliet. A short introduction to the radon and hough transforms and how they relate to each other. Technical Report QI-2004-01, Delft University of Technology.

[6] R. Duda and P. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11-15, 1972.

[7] C. Galambos, J. Kittler, and J. Matas. Progressive probabilistic hough transform for line detection. *Computer Vision and Image Understanding*, 78(1):119-137, 2000.

[8] International Olympic Committee. Sun Yang Smashes Men's 1500m Freestyle World Record- London 2012 Olympics. Internet: http://www.youtube.com/watch?v=T5FlDy3YmDQ, August 4, 2012 [December 15, 2013].

[9] C. Jung and R. Schramm. Rectangle Detection based on a Windowed Hough Transform. *Brasilian Symposium on Gomputer Graphics and Image Processing (SIBGRAPI' 04)*, 17(20):113-120, 2004.

[10] Open Source Computer Vision Library. http://opencv.org.

[11] S. Suzuki and K. Abe. Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30(1):32-46, 1985.