Field Detection Using MATLAB

Daniel Grissom[1] and Joseph Tarango[2]

[1,2]*Department of Computer Science and Engineering, University of California, Riverside*
900 University Ave. Riverside CA 92507 USA
[1]daniel.grissom@email.ucr.edu, [2]joseph.tarango@email.ucr.edu

Abstract— **Segmentation is an essential process of image processing which locates and separates objects in images. MATLAB offers tools for image processing; however, detecting specific objects is non-trivial. In this paper, we introduce an algorithm to detect and segment athletic fields from color Google Earth images. The MATLAB implementation is introduced and analyzed based on a set of 25 images taken from Google Earth.**

Keywords— **Google Earth, Google Maps, image processing, MATLAB, object detection, segmentation.**

I. INTRODUCTION

Segmentation is the process of separating objects within a given image and allows regions of interest to be identified as objects using pixel processing (e.g., edge detection, color extraction, sharpening). By applying the appropriate pixel processing algorithms in a specific order, objects can be discovered and classified using distinguishing criteria. In this paper, we propose a naive segmentation algorithm called Grissom-Tarango Athletic-Field Detection (GTAFD). The algorithm is domain specific and uses criteria known from athletic fields to classify the objects. The algorithm is implemented and tested in MATLAB.

II. MOTIVATION

As smart phones become ubiquitous, the use of Google Maps/Earth is becoming increasingly popular and the need to automatically identify areas of interest becomes more desirable [1][2]. Being able to identify areas of interest, such as athletic fields, will allow users to search the nearest to their location. For example, given someone is near the park seen in **Figure 1**, they could identify and find the athletic fields at this park.

Figure 1: Park with several athletic fields (football field on the left and soccer field on the right).

III. METHODOLOGY

A. *Object Detection*

Table 1 gives a high-level description of the GTAFD algorithm. The MATLAB code in its entirety can be downloaded at [3].

```
1.  For all pixels
        if RGB is not within "athletic-field" green threshold then
            set RGB values to 0
        else
            set RGB values to 1
2.  Remove all regions bordering edges of image
3.  Perform morphological closing
4.  Remove connected regions less than region threshold
5.  Fill regions
6.  Trace regions
7.  Extract shape measurements from regions into objects
8.  For all objects
        if object is within known constraints then
            mark object as athletic field
```

Table #1: GTAFD Algorithm

Processing an image to find a particular object requires knowing the distinguishable features of that object. In our case, we search for non-baseball/softball athletic fields. This primarily includes football and soccer fields, but also includes unmarked, general-purpose athletic fields that may often be used for other sports such as Frisbee or rugby.

A training set of 10 images containing athletic fields and stadiums, manually taken from Google Maps, was used to learn the key features of athletic fields. From our analysis of these images, we learned that athletic fields share the following properties: they are some shade(s) of green in color, rectangular in shape, and predominately solid in color.

Intuitively, it makes sense to focus only on green areas of an image since nearly all athletic fields are some shade of green. This will exclude fields that are not well kept and appear brown due to high wear and low maintenance, but will still allow many athletic fields to be found. To create a color base-line for our processing, we surveyed the initial 10 images containing several athletic fields to pinpoint the range of acceptable red, green, and blue (RGB) values for most fields. Color values within the range were kept and other values were discarded. The result of this step is a binary (black and white) image in which white pixels indicate that the pixel from the original image fell within our range of green; black pixels indicate non-green pixels in

the original image. Thus, we have a new image containing only the green areas (**Figure 5A/B**).

Partial fields can be seen at this point, so we remove all regions bordering the edges of the image since there is not enough information to accurately determine if the green area is an athletic field (**Figure 5C**). The regions remaining are of interest, so we perform morphological closing (dilation followed by erosion). Morphological closing is an image processing method used on shapes (connected pixel blobs) where individual pixels are compared with neighbouring pixels and assigned a value based on thresholds (**Figure 5D**).

The regions can still have noise after morphological closing, so removal of connected objects of few pixel counts is necessary. We remove objects containing less than 6K pixels (**Figure 5E**), which puts a lower bound on the elevation of the images we can process. With this 6K pixel threshold, we have successfully found fields where the Google Maps "measurement stick" is 200ft. Images zoomed out further than this were not tested since it is increasingly difficult to spot fields with the human eye as the elevation of the images increases.

Next, the holes in the (green) regions are filled, resulting in cleaner and fuller pixel regions (**Figure 5F**). Athletic fields are typically marked with white lines and numbers which mark the boundaries and regions of the field. Filling the holes in these regions fills in the non-green parts of the field to give a more solid object.

The remaining regions are potential candidates for athletic fields. In order to process these regions into objects, we use the MATLAB function *bwboundaries* (**Figure 5G**). This function traces regions on their exterior and interior boundaries to identify them as distinguishable shapes. Each of these shapes are processed into objects where shape measurements are calculated. The shape measurements include area, length, width, and several other properties.

B. *Classification*

After object detection, we have a list of distinct green objects from the image which we must classify as athletic fields or not. We use the rectangular property of athletic fields to help classify. The ratios for a football field with green end zones, a football field with painted (non-green) end zones, and a soccer field are 0.44, 0.53, and 0.69, respectively. However, fields almost always have additional green on the outside of the boundaries, and satellite imagery is rarely taken from directly above the field, so the fields are usually elongated in aerial images. From the training images, we experimented with the ratios of the fields and decided to filter out any objects that did not have a ratio between 0.4 and 0.9, as seen in **Figure 2**.

We use the property that athletic fields are predominately solid in color to finalize our classification. Some of the objects may have the correct rectangular ratio, but may have significant areas of non-green pixels in the middle (e.g. a park with sidewalks cutting through). To account for regions like this, we sweep the filled-area ratio threshold to filter out non-field objects. The filled-area ratio is the percentage of the box/shape bounding an object that is actually filled with pixels. Any objects remaining at this point are classified as athletic fields. **Figure 3** shows the classification of the entire dataset when the filled area ratio threshold is 0.8. The ROC graph in **Figure 4** shows the true/false positive rates for the GTAFD as the filled-area ratio sweeps from 0 to 1.

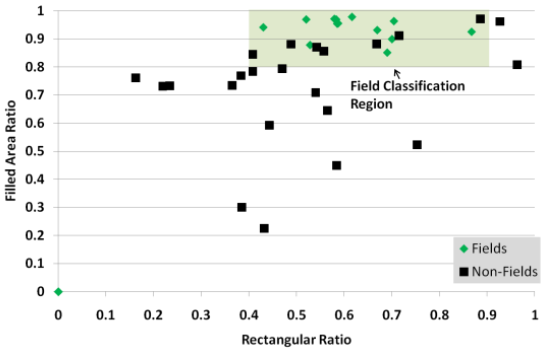


Figure 2: Training data from 10 images showing the field classification region. The rect. ratio is bounded between 0.4-0.9; the filled-area ratio sweeps (shown here at 0.8).

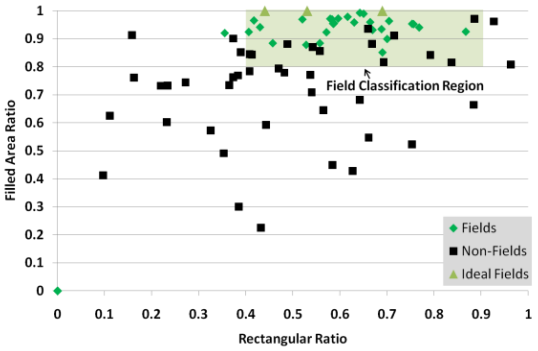


Figure 3: All data from 25 images showing the field classification region. The rectangular ratio is bounded between 0.4-0.9; the filled-area ratio sweeps (shown here at 0.8).

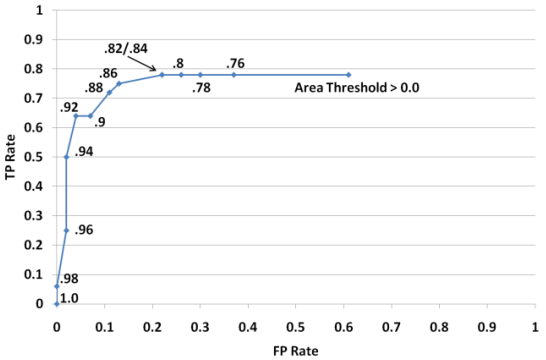
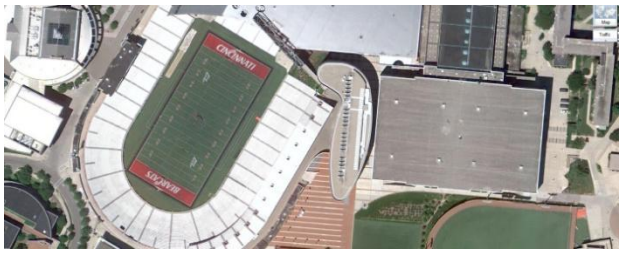
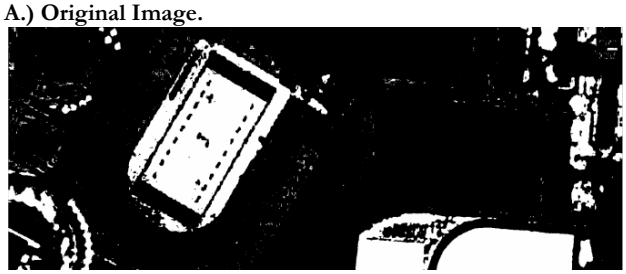


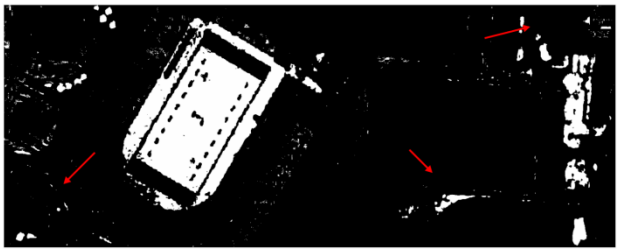
Figure 4: The ROC curve generated by sweeping the filled-area ratio threshold from 0 to 1.



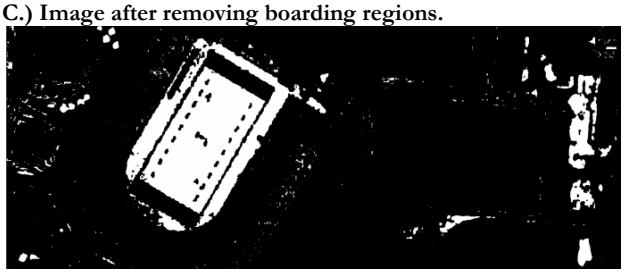
A.) Original Image.



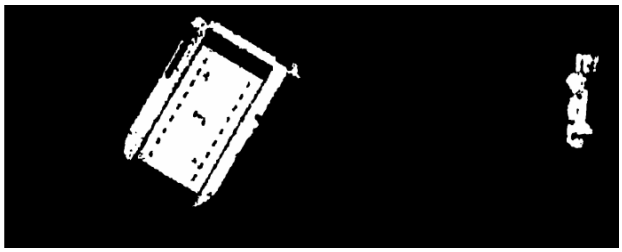
B.) Extracted "green" image from RGB thresholds.



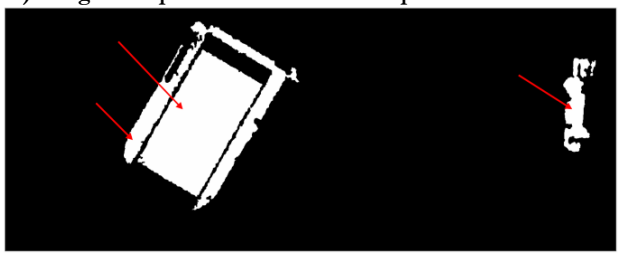
C.) Image after removing boarding regions.



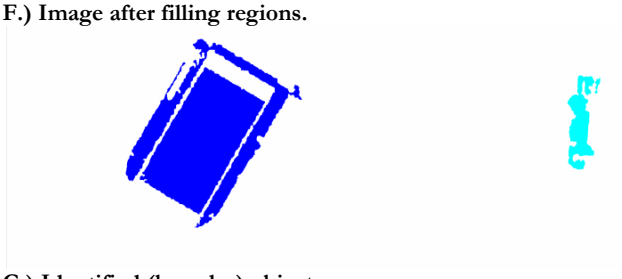
D.) Image after morphological closing.



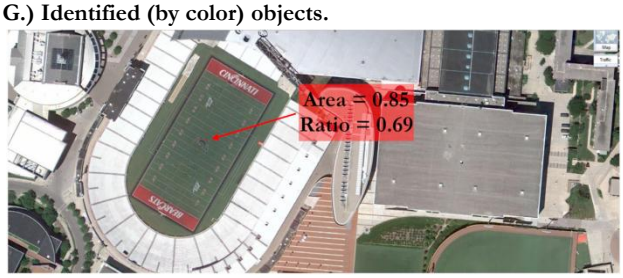
E.) Image with pixel blobs less than 6k pixels removed.



F.) Image after filling regions.



G.) Identified (by color) objects.



H.) Detected regions based on athletic field constraints.

Figure 5: An example showing how an image is processed to identify objects. Red arrows annotate areas that changed (not all changes annotated)

IV. EXPERIMENTAL SETUP

The experiment was conducted on an x86 server using MATLAB and hand-selected JPEG images from Google Maps. The images were selected containing various athletic fields and fields similar to athletic fields. All results are reproducible and the MATLAB code and dataset images are published online [3].

Implementation of the GTAFD algorithm is relatively straight forward; however, due to non ideal images from Google Maps, modifications were necessary. These modifications were necessary because images used in Google Maps are taken by satellites, aerial vehicles, and GIS 3d Globe in non-ideal angles, resolutions, and weather conditions [2]. In order to create a more-uniform delectability among images, we had to reduce noise by applying the Gaussian transformation before running the GTAFD algorithm.

V. DISCUSSION

From the ROC graph (**Figure 4**), we determine that the 0.86 filled-area threshold is the appropriate operating point. This operating point provides the best true positive (0.75) to false positive rate (0.13) for our algorithm. The GTAFD algorithm is able to detect athletic fields despite orientations; however, field obstructions within the image make detection difficult. Future work could be to develop a generic training algorithm to find basic characteristics between databases of objects to detect any specific objects within an image.

VI. REFERENCES

[1] http://en.wikipedia.org/wiki/Google_Maps
[2] http://en.wikipedia.org/wiki/Google_Earth
[3] http://www.cs.ucr.edu/~grissomd/courses/CS235.html