Image Blob Detection: A Machine Learning Approach

Andrew Colello

March 18, 2016

Abstract

The field of computer vision has developed several methods of detecting contiguous circular pixel areas, or blobs. While both blob detection and computer vision are both still in their infancy, they are already being used in a variety of applications with significant successes. This study examined the effectiveness of classification-based machine learning algorithims on blobs and compared several models with a baseline to measure their performance. The system's objective was to take an image of a golf course as input and output the location of a golf ball in the image, if one is detected. The system took a raw image file as input, scanned it for blobs using image processing software, and then classified each blob as positive or negative as output. Results showed that several classification models, especially tree-based models such as Random Forest, could be used to classify blobs with statistically significant success.

Contents

List of Figures

List of Tables

1 Introduction

Computer vision is a broad field that is closely related to multiple sceintific fields such as mathematics, big data, and machine learning. Currently, computer vision is being used with marked success for applications such as facial recognition software [7] and self-driving car operating systems [12]. Blob detection is a specific field in image identification that focuses on identifying circular shapes in an image that have a definable edge and contiguous color. Machine learning can be used to classify blobs according to their features such as size and color. This project aimed to combine blob detection and machine learning by finding blobs in an input image and classifying each blob based on specific traits. The overarching goal is to use these findings to create a mobile application that can detect a golf ball sitting in the grass using a raw image taken by the golfer.

2 Background

Often times golfers will lose sight of their golf balls on the fairways or in rougher parts of the golf course due to the distance and environment. Lost balls result in a penalty of one stroke, as well as the expense of having to buy a new ball. Today's smartphones are portable, powerful, and ubiquitous. Smartphones possess the capability of finding a lost golf ball automatically by capturing the general location as a photo image, then employing computer vision and other processes to locate the golf ball in the image before outputting that location to the user. Research centered on computer vision applications and solutions, and eventually gravitated toward data mining and machine learning studies as well.

2.1 Alternative Solutions

Several solutions have been proposed to detect blobs, with limited success. One system uses similar elements to solve the same problem as this project. The Ballfinder Scout scans the playing field and looks for the unique color signature (visible spectrum) of white golf balls [9] using a normal digital image and processing it into a thermal image. It uses the "thermal signature consisting of a circular object distinct from its surroundings or two or more concentric circular objects distinct from their surroundings" [9] to

locate the golf ball, but does not use any machine learning for processing. As a result, it has not been well reviewed and has not been a commercial success. I submit that a similar image processing approach with an advanced machine learning processor will show a marked improvement in golf ball location, because of the many distinct objects in an environment that may cause false positives. Other solutions relied on golf balls modified with electronic or other special software and/or hardware [6].

2.2 Related Work

Most automatic image identification uses some form of image processing to aid in its overall system. Lanitis and others created an automatic facial identification system that used image processing to "deform each face image to the mean shape in such a way that changes in grey-level intensities are kept to a minimum" [7]. This way, the three-dimensional color image is changed into a two-dimensional greyscale image that later could be processed by other system elements. Lanitis and others' study shows that images can be successful when combined with other elements, such as machine learning, to create accurate image classification systems.

Sun, Bebis, and Miller created a system for self-driving cars that incorporates advanced tracking and image recognition software to parse and process what the car sees. Their system preprocesses the image based symmetry, color, contrast, texture, and quadrant location on the X-Y-Z plane to create an output that is then sent to a machine learning processor to determine the proper path that the car should take [12]. The image identification system showed success when using contrast and quadrant location to process images, and then used that processed picture as input for a machine learning mechanism to identify each object within sight of the self-driving car.

Moon and others built a computer vision application that uses blob detection on medical images of breast cancer patients to find blobs that may or may not be cancerous [8]. This study attempted to find blobs on the candidate images, and then attempted to classify the blobs based on features such as blobness, internal echo, and morphology. According the the report, "results suggest that [computer-aided decection] systems based on multi-scale blob detection can be used to detect breast tumors in...images" [8].

2.3 Research Evolution

Research was initially focused on image processing using advanced filters and pixel maximization. To identify the possible locations, an open source program ImageJ was used to find pixels that have the highest contrast with neighboring pixels. In the API for ImageJ, the object called MaximumFinder found the maxima pixels of an image and outputted those onscreen [2]. After ImageJ located the candidate areas, the computer language called R was going to be used as the feature extraction and machine learning aspects of the system.

The main problem was that R had a somewhat out-of-date ecosystem for image processing, and it is cumbersome to learn. Also, ImageJ's API was equally rough around the edges, and the resulting initial system did not progress past early ImageJ data mining steps before being refactored several times over. Eventually, Python was chosen as the language of choice for most of the code because it has a rich and diverse number of packages that were used in nearly every step of the system. Python's machine learning library called scikit-learn was used on several feature extraction and blob detection steps, and initially was used for blob classification. After strugging to create proper inputs for scikit-learn, a Java-based machine learning software suite called weka was used for the heavy lifting in classification because of its simple API, massive library of classification models, and its easy to use ARFF format.

3 System Process

The system process began by accessing the image sharing website known as Flickr (http://www.flickr.com). The first step was data mining through target queries with Flickr's REST APIs. Next, the system applied blob detection algorithms to each response image so that a list of candidate blobs were generated. Several features were extracted from each blob and then the data was parsed and recorded in a format that could be classified by a machine learning model. Each blob was then classified as positive or negative based on the training data and type of algorithm being used. Models were finally evaluated based on percentage of correct positive classifications and then compared against a baseline to establish statistical significance.

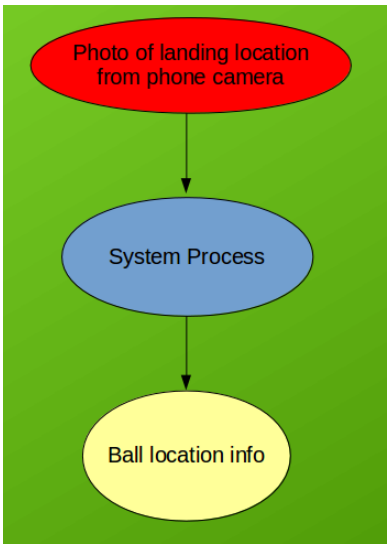

Figure 1: A high-level view of the system process.

3.1 Data Mining

The data mining step was handled by the python module called Scrapy. First, the process would kick off by accessing Flickr through its REST API. The responses were then mined and processed by the system until it was decided that search results were no longer returning useful images. Combined with the inital query and other queries used for training, a total of 214 images were mined for this project.

The first step in data mining was defining the specific features to be extracted from each image. Scrapy uses objects called items to declare features as fields. After items are defined, Scrapy uses an object called a spider to start at a user-defined url as a request source and then ”crawl” through the request source until told to stop. Each response is then packaged into an item which is promptly populated according to the spider’s parse logic. After the response is parsed, the item is finally yielded to an item pipeline controlled by Scrapy where it is outputted according to process logic contained in the code.

3.1.1 Spider

The spider contained the most of the logic for quering flickr and then parsing the data in the response. Two text queries were used: ”golf course rough grass” was used to find pictures of balls in the rough, while ”golf

course grass plain" was used for results that did not contain golf balls. The spider then iterated through each response, parsed the response images into items, and then yielded the items to the pipeline until a sufficient number of data was collected.

Parsing each response was a two-step process. First, each image was fed to a blob detection function that generated a set of candidate blobs in the image. Second, each candidate blob was scanned for features using the Python Image Library (PIL). The features were added to the item fields until every possible field was populated with data. After parsing was complete, the spider yielded the populated item to the pipeline.

3.1.2 Item Fields

Items had eleven fields: source url, blob detection algorithm, x-coordinate of blob center, y-coordinate of blob center, mean pixel value of the blob, median pixel value of the blob, mode pixel value of the blob, blob radius, the ratio of blob radius to total image height, the ratio of radius to total width, and class which could be either "POSITIVE" or "NEGATIVE". During runtime the spider created a blank item for each response, and then gradually filled the item with data from the image in the response.

3.1.3 Pipeline

The pipeline was a simple script that created a new file to write to for output during initialization. Whenever an item was yielded by the spider, the pipeline would receive it and then handle it according to the logic of the code. The pipeline accessed each item as if it was a dictionary, and then used string formatting to write the data to the target file in a specified form. Weka's ARFF format was used as the output type to make the spider functional without the need for a database or message broker software.

3.2 Blob Detection

Research indicated that the golf ball problem can be reduced to the problem of detecting a blob and then classifying that blob based on its attributes. Blob detection is a small subfield of computer vision and statistics that aims at detecting regions in an image that differ from surrounding regions. Blobs are defined by have contiguous and constant properties, such as color, contrast, brightness, or other empirical properties

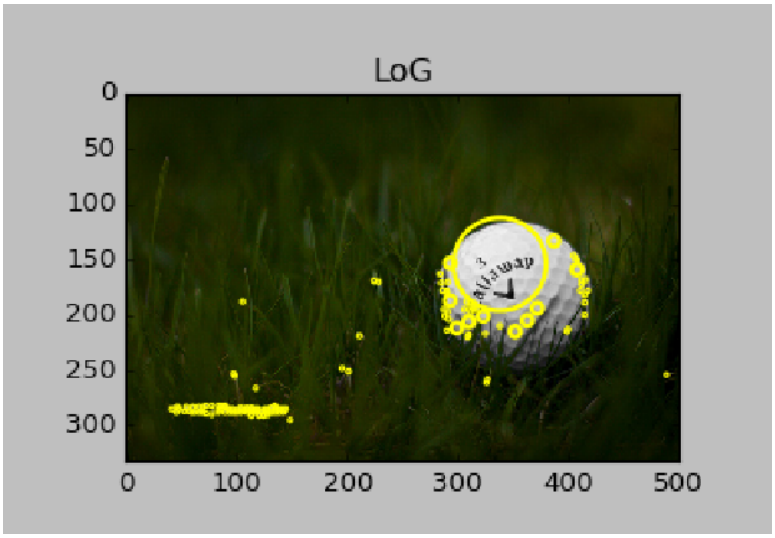

Figure 2: A view of the blob detection step. Blobs are outputed on a graph where the image is superimposed, (0,0) represents the top left corner, and (image-width, image-height) represents the bottom right corner of the image.

of the image [8].

3.2.1 Implementation

To identify candidate blobs in each image, the python module called skimage was used for its built-in algorithms. Specifically, the blob detector that was used for this project was based on finding the Laplacian of the Gaussian for candidate areas in the image. Jain and Park used the Laplacian of Gaussian to detect facial marks such as freckles, moles, and scars in their facial recognition study [4]. Laplacian of Gaussian works by finding a gradient from a given central location, and attempting to define an edge of the blob based on differences in the gradient [4]. The final blob is given a central coordinate and a radius, as it is always circular in the implementation that was used through sklearn. In short, each image was converted to greyscale and then represented as an array. This array was then sent as input to the laplacian of gaussian function in skimage. The output was a list of (x,y,r) coordinates, with (x,y) indicating the center of the blob and r represents the radius of the blob.

3.2.2 Feature Extraction

After the list of blobs were detected and recorded, each (x,y,r) item in the list was iterated through and features were extracted from the image for each blob. All eleven features in the scrapy item were extracted for every blob that was detected in each image. Some features, such as the x-coodinate, y-coordinate, and radius, were mostly or completely found by previous steps of the spider and therefore required little to no processing to populate the item. Features that required actual image processing such as mean pixel values used the Python Image Library or PIL for extraction and advanced image manipulation.

3.3 Blob Classification

The final step of the system process was classifying each blob as golf balls or noise. The data mining step was designed so that it outputs an ARFF file after completion, which is the default file format for the machine learning suite called weka. Weka was used to create several classification models, train those models, then use cross-validation to predict classes and analyze the prediction accuracy. Also, weka was used to compare the effectiveness of several machine learning models with each other, and to establish statistical significance using t-tests.

3.3.1 Machine Learning Overview

Machine learning is a subfield of computer science that uses computational power and algorithms to identify meaningful statistics, such as patterns, in data. Machine learning is closely related to pattern recognition and artificial intelligence, and branched from those two fields in its inception. This is evident when systems using machine learning find patterns or groupings in inputted data, then attempt to alter their own behavior to learn from the new patterns that have been discovered. Arthur Samuel once said that machine learning is a "field of study that gives computers the ability to learn without being explicitly programmed" [11]. While that statement is not entirely true in contemporary computing, machine learning provides an extra resource for data scientists to discover interesting information in the data that is being studied.

With enough data and accurate methods, machine learning can show patterns and relationships between attributes that were not obvious at first glance. On the converse side, machine learning frequently

highlights patterns that are glaringly obvious, or that do not exist. It is incredibly difficult to measure causation with computers and statistics, yet easy to measure correlation [1]. Machine learning algorithms, called models, frequently use mathematical statistic and probability principles to analyze data and then output predictions or decisions based on those statistics. Learning methods for machine learning models include classification, clustering, and numeric prediction.

Methods can be described as supervised or unsupervised learning. Supervised learning presents its system with a training set of preprocessed data that includes inputs and their sought after outputs; the system then attempts to learn rules that map any input to its proper output [1]. Unsupervised learning does not use any preprocessed data, leaving the system to draw its own conclusions about the data [1]. Reinforcement is a third type of method learning, similar to unsupervised learning in that it does not have any training set data; however, reinforcement learning uses input that constantly changes, such as input for the vision system of a self-driving car [12].

Classification learning typically predicts a class attribute by using a preprocessed "training" dataset to build a classification model [1]. Classification algorithms can use decision trees, rules, probability, and many other methods to build their models. The accuracy of each model is measured by the number of correct classifications that the model makes for each instance. Classification typically uses supervised learning to predict distinct categories given a training set of data that has its categories manually discovered [5]. Classification is applicable with data types that are binary, categorical, or ordinal [5].

Clustering groups attributes by assigning each one a coefficient and then pairing similar coefficients together until distinct clusters are formed [1]. Clustering algorithms find this coefficient in several ways. Hierarchical clustering calculates the distance of one object to another and then clustering objects that are near to one another. Centroid-based clustering represents each cluster by a central vector and then calculates the distance of each object to each cluster. Clustering is typically unsupervised and best used when data does not have distinct categories that can be used by a classifier [1].

Numeric prediction learning typically uses linear or multiple regressions to create a mathematical model for the dataset, then predicts a numeric attribute using that model [1]. Often, probabilistic outcomes of several predictions are calculated before the final numeric attribute is predicted. Numeric prediction is

typically supervised learning and is best used when working with data that consists of only real numbers [1].

3.3.2 Classification Algorithms

Classification models were the object of this study, as it was attempting a binary classification for each blob entry. I used four types of classification models: tree-based decision forest algorithms, rule-based algorithms, lazy evaluation algorithms, and baysean networks.

Tree-based algorithms classify an input by sorting it based on feature values at the various nodes in the tree. Each node represents a feature, and the input starts at the root node and continues down the tree until it reaches a classificaiton result [5]. Criminisi, Antonio, and Shotton studied the application of decision forests, or classifiers with sets of trees, for computer vision and medical image analysis [3]. In their study, they describe how any form of decision forests can be applied for image classification, especially when dealing with static input of high contrast. However, they note that some decision forests have tendencies to overfit training data. Leonardis and others used decision forests to identify corners during high speed car travel. They also found that in the presence of noise, decision forests tend to overfit data and make poor decisions [10]. Furthermore, Leonardis' study concluded that decision forests can be used for image identification as long as noise is controlled, as "the performance drops quite rapidly with increasing noise to start with," but then as noise increases past a certain point, decision forests outperform other machine learning algorithms [10].

Rule-based algorithms construct a model by creating rules, similar to if-then chains based on training data. The goal of rule-based models is to create a minimal set of rules that are "directly induced from the training set" [5]. They are typically comparable in performance to decision trees, but can suffer from incomplete or inconsistent rule sets. Zero-R is the simplest rule-based model; it works by creating no rules and simply choosing the class that is most common in the training set. One-R works similarly by creating a single rule based on the most significant feature, and then classifying based on that.

Lazy learning, or instance-based learning, requires minimal computation time compared to other algorithms but are often not as accurate [5]. In lazy learning, the sorting process is not performed until after

classification; thus, the model is continually adjusted according to the accuracy of previous predictions [5].

Baysean networks classify by creating a statistical probability model with the training set, and then computing the probability that an instance belongs to a class based on the model. Baysean algorithms can form several nodes that classify an input based on complex nodes that are influenced by probability outputs [5].

3.3.3 Training Data

Training data was created using a combination of manual classification using ImageJ, and blob detection on images that did not contain a golf ball. There were 54 instances classified as ”POSITIVE”, 99 instances classified as ”NEGATIVE”, and 61 missing classifications for a total of 214 instances and 153 verified instances.

To get positive results, the system used the query ”golf ball rough grass”. The result URLS of the query were recorded, and then each image was downloaded to a local compter. The ellipse analysis tools of ImageJ were then used so that the (x,y) coordinates of the ball's center and the radius of the ball were recorded based on manual analysis of the image. Each url was then fed through the scrapy data mining tool to extract features about the candidate blobs in the image. Finally, candidate blobs were classified using a script that compared the (x,y) coordinates of the candidate blobs with those of the manually classified blobs. Candidate blobs were considered ”POSITIVE” if their centers were within a predetermined threshold of the manually processed blobs; to limit results, that threshold was kept constant at 50 pixels.

Negative results were acquired by querying for ”golf course grass plain”. The results were then sent directly to the scrapy spider without any human interaction, and any blobs detected were immediately recorded as ”NEGATIVE”.

4 Results

Zero-R was established as the baseline on which to compare all other results. All models performed significantly better than Zero-R, with Random Forest reporting the highest accuracy at 89.80%. All models were verified using 10-fold cross-validation, and all models used the same dataset.

Model	Type	Percent Correct
Zero-R	Rule-Based	64.75
One-R	Rule-Based	86.33
JRip	Rule-Based	87.44
k-N-N	Lazy	88.45
Naive Bayes	Probability	85.68
J48	Tree-Based	86.20
Random Forest	Tree-Based	89.80

Table 1: Classificaion Model Comparison

4.1 Tree-Based

Two tree-based models were used for experimentation: J48 and Random Forest. Of all models used for testing, Random Forest was the most accurate. It created a random forest of 100 trees, each constructed using 4 random features. Random Forest did take the longest to complete by a signifcant margin. Figure 3 shows the resultant pruned tree from the J48 training exercise; note that there are 11 leaves and the size of the tree is 21.

4.2 Rule-Based

In addition to Zero-R, the rule based algorithms One-R and JRip were used as well. Zero-R had the worst results with 64.75% correct, as expected by the behavior of the simple algorithm and by the size and composition of the dataset. One-R was used as a secondary baseline to compare the effectiveness of complicated models. It is worth noting that One-R achieved 86.33% correct, which is very statistically close to the non-baseline algorithms.

4.3 Lazy Evalutation and Naive Bayes

One lazy evaluation algorithm called k-Nearest-Neighbor, or IBk, was used for this study. A Naive Bayes classification model was also used so that there were diverse types of classificaiton algorithms to compare with each other. The lazy model was by far the fastest, as expected, with an actual build time of 0 seconds meaning that it build faster than the computer could time it. By comparison, the slowest model (Random

```
J48 pruned tree
------------------

radius <= 36
|   ycenter <= 185: NEGATIVE (62.0)
|   ycenter > 185
|   |   medianpx <= 164
|   |   |   xcenter <= 342: NEGATIVE (20.0)
|   |   |   xcenter > 342
|   |   |   |   meanpx <= 112.445578: NEGATIVE (2.0)
|   |   |   |   meanpx > 112.445578: POSITIVE (2.0)
|   |   medianpx > 164
|   |   |   medianpx <= 253
|   |   |   |   radiuswidthpct <= 0.058824: POSITIVE (6.0)
|   |   |   |   radiuswidthpct > 0.058824
|   |   |   |   |   xcenter <= 108: POSITIVE (3.0)
|   |   |   |   |   xcenter > 108: NEGATIVE (2.0)
|   |   |   medianpx > 253: NEGATIVE (2.0)
radius > 36
|   ycenter <= 177
|   |   modepx <= 204: NEGATIVE (11.0/1.0)
|   |   modepx > 204: POSITIVE (4.0/1.0)
|   ycenter > 177: POSITIVE (39.0)

Number of Leaves  :     11

Size of the tree :      21
```

Figure 3: Resultant tree for J48 algorithm.

Model	Number Correct	Number Incorrect
IBk	137	16
Naive Bayes	131	22

Table 2: Number correct for IBk and Naive Bayes

Forest) took 0.15 seconds on average to train and classify. IBk performed comparatively well with 89.5425% correctly classified instances. Naive Bayes was the worst performer, with 85.68% correctly classified instances.

5 Conclusions and Future Work

When compared with Zero-R as the baseline, all other algorithms were statistically significance with a significance value of 0.05. Based on the findings, IBk would likely be the most logical choice for a model in a mobile app because of its combination of accuracy and low system load. Based on the research, it

is of little surprise that Random Forest performed the best, as Criminisi, Antonio, and Shotton medical imaage computer vision analysis showed that decision trees can be the right choice for similar tasks [3]. Because of the high accuracy of One-R and the low statistical difference between it and the other non-baseline algorithms, I am worried that the models are overfitting the sample data. There are only 150 images that occasionally have unrelated or random results from Flickr, so more research is needed in the effectiveness of classification on blobs detected using compter vision. Moving forward, the mobile app will be developed and the training set will be grown so that the model will have a more robust decision processing engine.

References

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Research Services Branch. Imagej, June 2015.

[3] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[4] Anil K Jain and Unsang Park. Facial marks: Soft biometric for face recognition. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 37–40. IEEE, 2009.

[5] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[6] E.H. Kuesters. Golf ball locator, September 5 2000. US Patent 6,113,504.

[7] Andreas Lanitis, Christopher J Taylor, and Timothy F Cootes. Automatic face identification system using flexible appearance models. *Image and vision computing*, 13(5):393–401, 1995.

[8] Woo Kyung Moon, Yi-Wei Shen, Min Sun Bae, Chiun-Sheng Huang, Jeon-Hor Chen, and Ruey-Feng Chang. Computer-aided tumor detection based on multi-scale blob detection algorithm in automated breast ultrasound images. *Medical Imaging, IEEE Transactions on*, 32(7):1191–1200, 2013.

[9] A. Nejah. Golf ball finder, January 5 2012. US Patent App. 12/830,225.

[10] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006.

[11] P. Simon. *Too Big to Ignore: The Business Case for Big Data*. Wiley and SAS Business Series. Wiley, 2013.

[12] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, 2006.