Recursive Median Sample-based Blob Detection

Thomas Lux

Abstract—In this paper a method for detecting convex patches of similar colors, otherwise referred to as blobs, is proposed. The method is aimed at computational efficiency and hence relies on repeatedly sampling images to gather information.

Index Terms—Computer Vision, Blob Detection, Sample Based, Relative

1 Introduction

THIS algorithm is aimed at producing information about visual surroundings for the purpose of single-camera robotic navigation. The initial inspiration for this algorithm came from the abundance of image-processing based blob detection algorithms that required much of the image to be processed when only select portions were needed. Being sample based implies a certain level of inaccuracy in this algorithm. This approach is a means of gathering relative information about the current scene. Later in the paper a proper demonstration of where this type of algorithm would be useful is presented, along with image samples of its performance.

1.1 Related Works

Many different blob detection algorithms have been written in the past. The purpose of these algorithms is generally to identify patches in images with a specific set of characteristics. A readily available blob detection algorithm in the OpenCV library called "simpleBlobDetector" produces the desired results by applying a set of various thresholding algorithms. Other blob detection methods involve using mathematical formulas similar to a Harris-Detector and image gradients to find blobs and their relative orientations.

• *T. Lux is a student of Computer Science and Mathematics, Roanoke College, Salem, VA, 24153.*
E-mail: thlux@mail.roanoke.edu

Some of the more successful blob detection schematics rely on image pyramids or scaled gaussian filtered images and structure tensors to calculate the centroid and pose of blobs in an image.

The primary difference between this proposed method and all standing blob detectors is that in current methods a specific type of blob is desired at the beginning of the search. In current methods the blobs detected are precalculated generally desired for tracking. The aim of this algorithm however is to simply track all potential blobs in a scene. This method yields suitable results for purposes such as real time robotic navigation.

2 Algorithm

This blob detection algorithm relies on three primary operations to produce a set of blob centriods from a given image.

1) Sampling the image
2) Performing a recursive median search
3) Filtering and merging produced blobs

2.1 Sampling the Image

The most effective means of sampling an image is still undetermined for the purpose of this paper. The sampling method used is a uniform and linearly displaced set from the full image size.

```
Sudo Code:
step = (Rows*Cols / Desired samples)
for (sample = 0,
     sample < number of pixels,
```

```
    sample += step)
tetra median search
```

2.2 Recursive Median Search

The goal of the recursive median search is to identify the bounds and the center of the current blob being sampled. After testing a tetra median search yeilded results with the most balance between consistancy and accuracy. In the tetra median case there are four searches done in the following order: North South, East West, Northeast Southwest, and Northwest Southeast. At the starting sample point, the edges of the blob are found in the North South direction, the bounds of the blob are updated, and the median of those two points is used as the starting point for the next search. This process is repeated in the order aforementioned. Hence, the title recursive arises from the nature of the algorithm using the previous result as the next starting point. This search can be expanded to any desired even number of searches, as stated a tetra median search simply yielded the best initial data.

2.3 Filtering and Merging Blobs

This process is done dynamically during the production of blobs in the image. This was done in order to reduce the average case runtime for storing blobs. Two blobs are considered the same if they both overlap more than one quarter of their volume and are similar in color. The former is determined by the bounds of one blob overlapping the center point of another, color difference is defined by a single channel of the two colors differing by more than a set threshold amount.

3 Conclusion

These results demonstrate that a valid blob detection algorithm can be produced with mere image sampling. The primary detriment of using a sampling based algorithm is deacreased accuracy, but this is intentionally sacrificed for speed of computation. The proposed blob detection algorithm has the potential to quickly identify similarly colored patches of images and their approximate center.

3.1 Applications

This algorithm can be applied for computationally efficient relative robotic navigation. Although perfect obstacle avoidance cannot be guarenteed with this algorithm alone, in combination with other sensors this algorithm could provide a substantial foundation for navigating. This algorithm would only require a single camera to operate.

For any instance where similarly colored patches need to be identified in an image a recursive median blob search could find them quickly. This has the potential to narrow the search regions for object recognition algorithms and can also quickly identify non-textured surfaces.

3.2 Future Work

A recursive medain search has the potential for creating navigable data for single camera robotic platforms. Before code of that complexity is pursued there is also a great potential for a more intelligent sampling algorithm. It is likely that the performance of this algorithm could increase if sampling of an image happened mostly outside of regions known to be part of a blob.

Acknowledgments

I would like to thank Doctor Durrell Bouchard for his continued guidance and feedback throughout this research. Without his insights and ability to measure the relevance of present work, this algorithm may have never been discovered. I would also like to thank Doctor Anil Shende for his persistance in encouraging students at Roanoke College to perfect ideas before ever turning them into code. Had I not spent hours planning and testing before typing a character I may have never come to these results. Lastly I would ike to thank my peers Derek LaFever and Randall Pittman for constant feedback and assistance through all of the little problems that plagued my day.

[2] [3] [1]

REFERENCES

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 2:3, 2001.

[3] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

PLACE PHOTO HERE

Thomas Lux Thomas Lux is a rising Junior Computer Science and Mathematics major at Roanoke College with a minor in Physics. T. Lux first began his interests in the robotics and computer vision fields over the summer of 2013. A project on various sensors used for robotic mapping provided an introduction to OpenCV and the field of computer vision that inspired this work.