A Computational Approach to Object Classification

Austin Lubetkin
Florida Polytechnic University
Lakeland, FL, USA
bocaaust@gmail.com

***Abstract*—This paper describes a computational approach to object classification. This paper introduces the reasoning behind pursuing integrals as a mathematical foundation for object detection. In addition this paper outlines the potential steps in an algorithm that can detect objects. The key advantage of this algorithm is its computational complexity. The algorithm can be simplified to a series of subtractions which are O(1) operations versus the much more computationally complex convolutional neural network approach typically applied to classify objects.**

Keywords—component; formatting; style; styling; insert (key words)

I. Introduction

Many computer vision algorithms in use today stem from the work done to establish edge detection as an extension of the derivative of the Gaussian operators of an image [1]. Derivatives at their core are a measure of change, when computing derivatives abrupt changes would be significant in the first derivative which is why they are significant for edge detection. Integrals on the other hand are mathematically a measure of area. Furthermore, integrals can be expanded into a series of subtractions. By abstracting an object or image to a series of layers separated in color space you can perform integration to measure the area of said layers. Then it simply requires subtracting the desired layers of the object from the image to see if it is possible to fit the object within the image.

Addition and subtraction are highly efficient operations operating at O(1) complexity and this process would overall allow object classification to occur as an O(1) operation.

Another advantage of using this integration method is you can consider objects by the gradients of colors and details that make an object instead of solely considering an object as a function of its edges. This ideally can provide a more accurate form of detection by considering this additional data that would be typically ignored in object detection.

II. The Algorithm

A. *Defining the Subshapes*

The first step to performing object detection is defining a classification parameter for dividing the input sources being the image and source images of the object. This algorithm requires dividing source images into layers. This could be done by breaking an image into layers by thresholds of color space. I'd recommend working in HSV color space since using Hue and Saturation for object detection primarily will minimize interference from light conditions.

In addition, multiple source images of an object can be used to develop a range of values. This range can account for the variation of the objects and allow for a more generalized classification procedure.

B. *Integrating the Subshapes*

The second component to the algorithm is taking the layers of subshapes and integrating the Gaussian vectors of the subshapes into 3D space. This will generate the area of the shapes created from the Gaussian vectors of the layer.

C. *Performing the Subtraction*

The next step for the algorithm is compositing the layers of matching ranges between the two sources and performing subtraction. This at its core is a measure of how these layers are able to fit into each other. The percent match of an object would be relative to its ability to fit within an image. When working with a library of objects you are trying to classify, the relative percent matches can be used to cull a selection of objects that wouldn't be able to appear in the image.

D. *Quadrant Recursion of Training Images*

Depending on the size of the library of objects being classified, it may not be possible to cull all objects that definitively won't occur in an image. This additional set of procedures would allow further culling to occur to identify the best match of the object being classified.

First, the object library images are divided into quadrants. With the samples developed, I typically would divide into 4 equal quadrants but this can be adjusted based on the size of the dataset. Steps A, B, and C would then be recursively performed on the individual quadrants and their percent matches would be composited to further cull objects from the library.

Eventually, there should only be a top match remaining and this would be returned as a object classified in the image.

E. *Quadrant Recursion of Input Image*

This is an additional optional step available to use cases requiring bounding box detection of objects classified in an image. This process will additionally require a threshold parameter. Once the top object is detected, its possible to divide the input image into quadrants. Steps A, B, and C would be performed on the image. If the there is a quadrant that has a match greater than a threshold this sole quadrant would be divided again and the process would repeat. If there is not a quadrant detected that has a match greater then the set threshold then recursion ends. The quadrant returned by this algorithm would then be displayed as a bounding box of the object classified.

III. Applications

A. *Visual Odometry*

Visual Odometry describes understanding the speed of a moving object by having a camera on the moving object. The Mars rover recently tried tracking its speed using the way its wheels were turning but it struggled due to the dust and environment of Mars. [2]

The possibility of using the algorithm described in this paper to achieve visual odometry is as follows. First of all, the computational efficiency of this algorithm would allow it to run object detection in real time using modern processors. Ideally, you'd be able to run it on every frame of a 60 frame per second video. By comparing the transformation of area which is a value already calculated of objects classified in images between frames you can relatively understand transformations in 3D space.

Through an understanding of multiple objects and their transformations in 3D space its possible to infer the rate of transformation of the camera. This could be used to calculate speed from a camera and achieve the challenge of visual odometry.

B. *Facial Recognition*

Currently, visual facial recognition is a highly explored topic. Many approaches use facial keypoint detection and mathematical approximations of the features of a face. One area these methods have failed at is they are unable to distinguish a real physical face from a recording or a 2D image.

The way to approach this problem and improve upon existing facial recognition algorithms is as follows. First, a 3D scan can be made using a composite of images of a person's face. Then, facial recognition algorithms could be used on a persons front-facing face in the typical fashion used to authenticate a user initially. Once an initial validation has occurred, the user would be instructed to move their face in a random set of instructed directions. Each of these instructions would correspond to layers processed through the algorithm outlined in this paper interpreted from the 3D scan of the user's face. Then, the algorithm outlined in this paper could be used to classify the face against these 3D transformations and determine if the user has a real face in front of the camera or a 2D image.

Acknowledgment

I'd also like to thank James Andrews for the direction on the Visual Odometry application. I'd

also like to thank Dr. Muhammad Abid at Florida Polytechnic University for the direction on the more secure facial recognition application.

REFERENCES

[1] Canny, John. "A computational approach to edge detection." *Readings in computer vision*. Morgan Kaufmann, 1987. 184-203.

[2] Maimone, Mark, Yang Cheng, and Larry Matthies. "Two years of visual odometry on the mars exploration rovers." *Journal of Field Robotics* 24.3 (2007): 169-186.