Depth Image Segmentation Using Statistical Methods

An Engineering Paper

Presented to

Junior Science, Engineering and Humanities Symposium

Maryville University

by

Connor Monahan

Junior

559 E. Highway N of
Wentzville, Missouri
63385

September 6, 2015 – January 10, 2016

Dr. John Barr
Physics Teacher
Timberland High School

Acknowledgements

Dr. John Barr

Vincent Redman

Darrell Wodrich

JP Yagelowich

Wentzville School District

Abstract

This paper presents an algorithm that utilizes unsupervised machine learning methods for classifying *n*-objects in a spatial image. Other methods of spatial image segmentation, for example the Laplacian operator and support vector machines (SVM), solely return all visible edges within the image, which can then be used to find all objects within the frame; however, this algorithm returns all points belonging to a specific object in the image rather than the edges of the entire image. This allows the omission of further processing and categorizes all the present data into classes using a modification of the Otsu thresholding method, by using a spatial metric and optimizing the local minimum to the partial derivative of an *n*-modal distribution. Implementation of the algorithm can be specialized to work on any list of points, such as an entire image or a region of an image. The results of the algorithm return distinct information for each object in the image, allowing further processing to determine other characteristics about the object, such as the orientation.

Table of Contents

1. Introduction

One of the most important problems underlying computer vision is the distinction and recognition of objects in an image. A common issue in current object recognition is the detection of multiple similar overlapping objects in an image. Often when objects are stacked or placed behind one another, common algorithms combine them into one region in a computer vision

solution. Many different algorithms currently exist to find objects in images; some simply find edges and boundaries in an image, such as Canny and the Laplacian operator. Each has a unique set of disadvantages: Canny and Laplacian only return every edge possible in the image and finding contours requires initial filtering to separate the contours from the noise of the image. Three-dimensional images also contain another element not accounted for by traditional edge detectors. The distance information provides geometric information on the objects in the field, data not easily represented by traditional intensity images [1]. Therefore, segmentation in depth necessitates a combination of these operations as well as statistical processing. In this paper, a new approach to solving a classification problem is presented which is capable of differentiating between various objects or regions of an image based on their distance and their location in the image.

2. Background

Traditional applications of computer vision technologies retrieve image data from cameras based upon color or infrared light and then process this to find objects and output usable information. Recently, the development of new cameras pushed spatial images to common usage. Also known as depth maps, each value in the picture provides a value related to the distance for every pixel in the image. There are many ways for cameras to create these images. A popular camera technology is calculating the distance based on Time of Flight information [2], where the camera calculates how long infrared light takes to return to the camera to calculate the distance. The economical Kinect sensor projects a specific IR pattern into the environment, which another camera on the same sensor receives. The sensor then uses the disparity in the resulting image, compared to a reference image at a known distance, to calculate the depth to the target [3].

Figure 1. Depth Image

3. Related Work

A common forerunner to image segmentation, morphology applications also find use removing noise from input images. These algorithms, erode and dilate, find the minimum and maximum values respectively of an input image based on a static kernel, resulting in an image with almost total noise cancellation [4]. All camera inputs, including depth-imaging devices, have noise – the slight variations visible from image to image in a static environment. Spatial images have significantly higher noise than regular images, as visible through experimentation. However, morphology cannot be successfully applied to spatial images because they only operate on binary images – images where each pixel value is either full white or black. The precise variation in spatial images gives them their usefulness, as software can calculate distance to almost every pixel. These operations, therefore, create an unprocessable image, because the necessary edge detection algorithms cannot work on such an image. Hence, another solution is required for successful classification of objects in spatial images that does not disregard the important data available in these special images.

The current trend in academia and industry for image segmentation and classification is the use of deep learning algorithms to classify and segment images. Neural networks have experienced a revival in popularity due to the invention of the convolutional neural network [5] [6] [7]. There are numerous challenges to test the performance of classification algorithms; however, they do not include any depth images [8]. Their exclusion of depth images is due to the convolutional neural networks' inability to accurately operate on depth images.

Despite this shortcoming of convolutional neural networks, there have been a few instances of image segmentation and classification in depth images. The first notable one was Microsoft with their human pose recognition algorithm that utilized a random forest classification [9]. The other also used random forest classification to identify hand gestures in American Sign Language [10]. These methods proved to be very accurate; however, they require labelled data, which this problem does not have, thus requiring a new approach.

4. Materials

This algorithm was implemented onboard an embedded robotics environment. The setup allowed mobility and wireless interfacing. The entire apparatus consisted of a frame upon four wheels connected to motors controlled remotely. For vision, the setup included a depth camera and an onboard computer to run the program. The computer was able to process the input data at roughly 17 solutions per second, allowing real-time implementation of the algorithm

Multiple depth-capable cameras were tested with the program. The ASUS Xtion Pro Live camera is capable of 30 frames per second and interfaces with a computer through a USB 3.0 port. This camera is bus-powered; it does not require a secondary power supply – a big advantage for smaller environments as one only needs to consider powering the computer where voltage regulation is a concern. Another option is the Xbox Kinect camera. It works similarly to

the Xtion camera: 30 FPS, USB interface; however, it requires a second 12V power supply. It is also larger and heavier than the Xtion. Both devices are similarly inexpensive.

An ODROID-XU serves as the side vision computer for this experiment. This single-board computer provides a 1.6 GHz ARM Cortex-A15 processor and 4 GB of memory, which allows for quick parallel execution of the software and image capture. The device has a very small form factor, about the size of a cell phone, allowing for a smaller robot and higher concentration of other sensors and equipment. It also consumes very little power, allowing for longer operation on a single charge of a 12 V, 17 Ah battery that also powers the motors.

The Ubuntu Linux 14.04 LTS operating system runs aboard the computer and executes the computer vision cameras. This system is advantageous to this application in many areas. First, it is available free of charge for many different platforms. The ARM architecture of the computer limits the range of operating systems usable. It also handles interaction with the cameras and the networking equipment automatically, allowing more development and testing of the algorithm without worry to the underlying operation.

Reimplementation of several necessary algorithms for this process, such as the Laplacian operator, is not necessary due to the use of the Intel OpenCV library. This piece of software handles many common tasks in computer vision automatically, allowing more development focus on the new algorithm itself. It also contains highly efficient and optimized implementations of these, which increase the speed at which the program can process images. It also contains functions specifically suited to run applications in parallel on popular GPGPU devices, thereby increasing the potential for added speed on more powerful, yet commonly available and affordable consumer hardware. Reputable institutions around the world confirm the accuracy of,

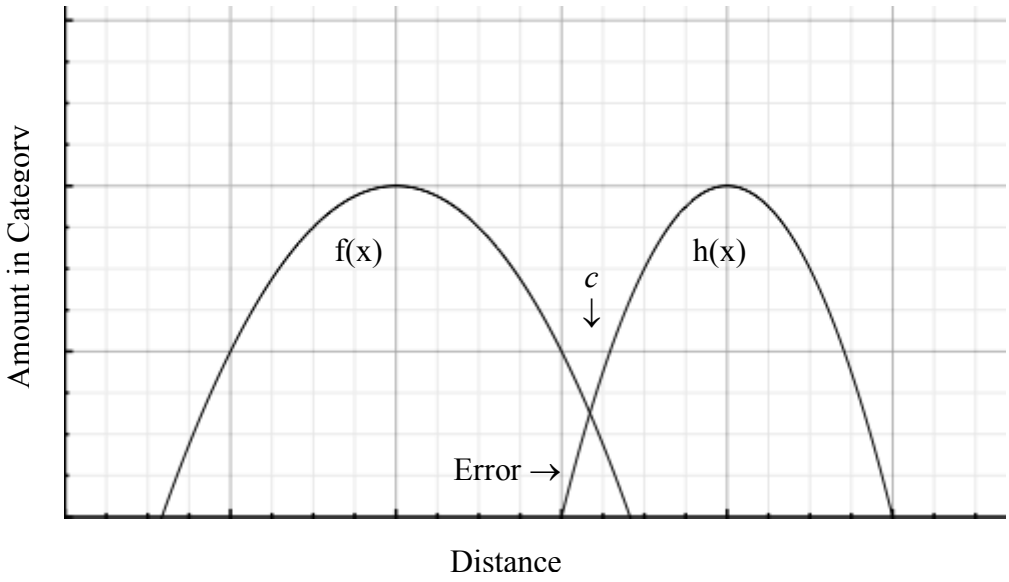
and contribute to, OpenCV. Defense contractors rely on the capability of this library due to the open source nature and heavy testing in a variety of environments.

5. Formation of Problem

Classification is essential to almost any computer vision problem. Often, these applications require the identification of certain objects or features in an environment based on their appearance in a two-dimensional image. The availability of three-dimensional images requires a new method of classification in order to recognize objects. Depth maps are structured in a way where pixel intensity in a single-channel image is directly proportional to the real world distance – the cameras encode the real distance using a linear formula to fit in a domain of 256 possible integer values. The Kinect can optionally encode the values in a range of 2048 for greater precision. Each model of depth camera uses a different formula to translate pixel values to real distance values, so this must be considered in the program's process as well. Regular images store color as the pixel intensity instead of a spatial value. This algorithm takes into account variation of pixel intensity, which corresponds to change in distance, to detect distinct objects at different distances and locations with respect to the camera.

The theory behind this algorithm comes from the popular Otsu method of thresholding an image [11]. Assume that the image has two objects in it.

Figure 2. Otsu Thresholding

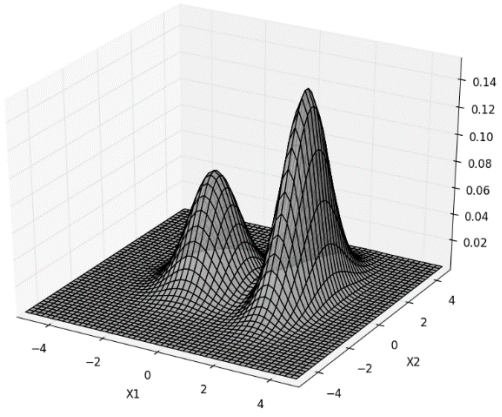


Otsu's thresholding method attempts to minimize the error in classification between the two objects, or between the two distributions, which is explained mathematically in (1)

$$\min_{c} \int_{0}^{c} \mathrm{h}(x)dx + \int_{c}^{\infty} f(x)dx \tag{1}$$

This problem becomes more complex when one considers depth and pixel coordinate instead of simply pixel intensity.

Figure 2. 3D Histogram of a Depth Image



Thus making the optimization problem become

$$\min_{c,d} \int_0^c \int_0^d h(x)dxdy + \int_c^\infty \int_d^\infty f(x)dxdy \tag{2}$$

This process was then generalized to accept an unknown number of underlying distributions.

6. Procedure

The first step to any computer vision process, apart from any setup overhead, is the retrieval of the image from the camera. With OpenCV and specific software such as OpenNI and OpenKinect, both tested cameras provide their output as a one-channel, 640x480 image. At time of capture, the depth data is encoded in a 16-bit format, providing higher precision, but unfortunately incompatible with the implementations of the next few functions in OpenCV.

Use of smoothing and averaging filters is a common first step of many computer vision applications, as they remove the issue of camera noise. All image streams have a certain amount of noticeable noise, caused mainly by infinitesimal changes in the environment. Infrared-based depth cameras also encounter higher amounts of noise than traditional cameras, which is visible in the differences between each output image. The Gaussian blur, a type of lowpass filter, removes high-frequency noise, which increases the stability of the first and second derivatives of intensity [12]. This ensures that tracked variation in the image is the result of real physical changes instead of randomness. The averaging feature of this operation also reduces the amount of information in the image and eliminates many sharp edges, which increases the precision of the edge detectors but must also be applied sparingly to prevent high loss of edge information. The Gaussian filter (3) uses a mask of a predefined size to smooth the image.

$$G_\sigma(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3}$$

After noise filtration, an edge detector is then applied to obtain the areas in the image with significant change in pixel intensity from one region to another. Note, however, that the algorithm described in this paper does not depend on the use of an edge detector, as it can work on entire images, but it cannot process entire images in real time. The Laplacian operator (4) is a common edge detector that works in both 2D and spatial images and is commonly used in academia. Edge detection continues along the path of reducing the information in the image, as its output consists of an image with a solid background and lines whose thickness determines the variation along the edge.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4}$$

The program then executes another procedure, a method of detecting contours from the visible lines in the image. This algorithm returns a list of the points forming the perimeter of each region as well as the ordering of overlapping regions [13]. Then, the area of each region must be calculated. This is achieved through Green's theorem (5), which states that contours can be integrated on the perimeter to find the area of the interior [14].

$$\frac{1}{2}\oint(-y\,dx + x\,dy) \tag{5}$$

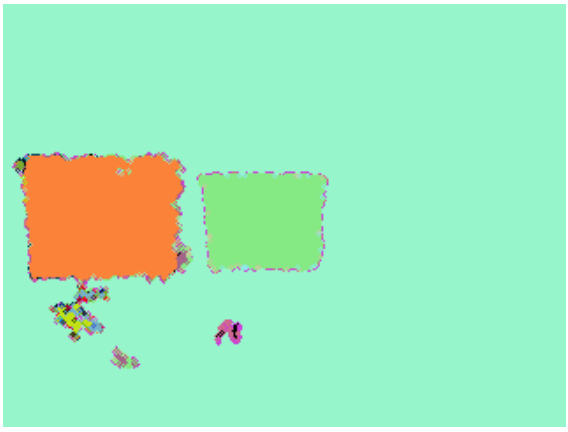
The program then filters this returned list, excluding all contours except for the innermost contour and removing contours with extremely small or large areas. In some conditions, objects distinguishable at this stage will be encompassed by a larger contour containing both, which must be eliminated in order to independently make calculations for each object.

The necessity of this algorithm comes from the fact that traditional classification and edge detectors solely account for visible edges in the image. This is acceptable for classifying color images, but it ignores the embedded value of a depth image – the depth image provides information about the texture of the object. Further processing is therefore necessary to take this variable into account as well. In this specific application, the input domain was {x | 0 <= x < 640; x ∈ Z}, {y | 0 <= y < 480; y ∈ Z}, however this algorithm is capable of generalizing to any *n*-dimensional input space. The proposed algorithm is described in algorithm 1.

Algorithm 1 Image segmentation

```
inputs ← pixels in image
classes ← ∅
while inputs do
    p ← inputs[0]
    for all c ∈ classes do
        dist ← dist(f(p))
        if |dist − Average(c)| < Stdev(c) + ϵ then
            c ← c ∪ p
            classified ← true
        end if
    end for
    if classified = false then
        c ← p
        classes ← classes ∪ c
    end if
    inputs ← inputs − p
end while
```

Figure 4. Graphical Output of Algorithm 1



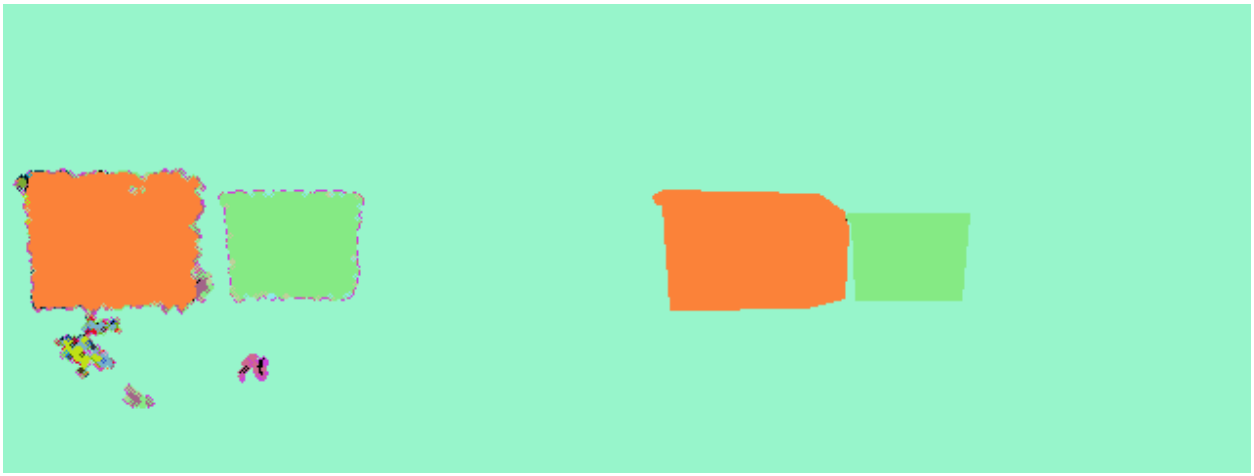
At this point, the algorithms return a list of unique objects in the image in the form of every pixel that makes up said object. This data is then used by an assortment of feature extraction algorithms to find application specific information and measurements. For example, the algorithm was tested with a post-procedure that counted the height of a stack of a specific object with known dimensions using geometry. Each test varies by application but they all function properly due to the accuracy of the data confirmed by the contour segmentation algorithm applied earlier.

7. Data

To test this algorithm, the result was compared to that of hand segmented depth images based on the color version. The error is given by (6)

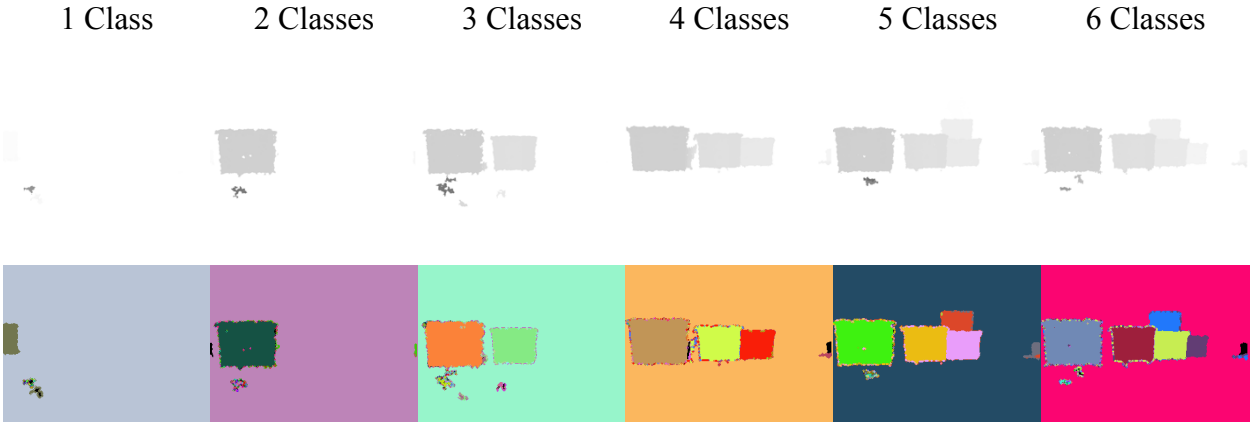
$$\frac{pixels\ misclassified}{total\ number\ of\ pixels\ in\ image} * 100 = Percent\ Error \tag{6}$$

Figure 5. Output vs Hand Threshold



As is shown in figure 5, the proposed algorithm preforms nearly identical to that of human performance. For a close range, optimal image with three classes (the background and two overlapping objects), the algorithm performed with 2.97% error in comparison to a hand-tuned threshold value, missing only a handful of pixels likely due to image noise. Testing was performed with two full images by comparing the contents of each region in both; no averaging of multiple images was performed before computation of contours in depth. Error can be further reduced by adding methods that deal with noise elimination, such as increased filtering and segmentation of objects based on dimensions.

Figure 6. Application of Algorithm 1 to Images of Varying Object Count



According to figure 6, the algorithm performs correctly on images with high amounts of classes, in roughly constant time considering image size is controlled. However, there exists a trend of increasing error almost linearly in relation to the number of classes in the image. A manual comparison of the hand-tuned threshold and the output image alludes that the spacing between the IR depth sensor on the camera and the color sensor results in slight shifts in the spatial image. The camera's computation of the depth map must have internal flaws as well, as only the face roughly parallel to the plane of the camera is visible. These sources of error can be overcome through the use of a different sensor or different calibration, and additional filtering.

8. Results

The accurate results of this algorithm prove the effectiveness of tracking several overlapping objects in a spatial image. This also expands the potential for depth camera application to many new fields that currently incur this problem of overlapping. Additionally, the low error in simpler conditions comes as a surprise because noise adversely affects many other computer vision solutions and makes them imprecise. Considering the algorithm runs

unsupervised, in which it has no base information to predict further calculations, it adds promise to this field of machine learning as well and encourages future work and study.

9. Conclusion

Accurate and precise calculations from depth images containing overlapping objects would be impossible without this algorithm. The original filtering step through Laplacian produced unsuitable results because the image processing steps could not work with objects combined in a single region. The data demonstrates the accuracy of the proposed algorithm. This method of image segmentation and classification could be applied to any image classification task, given that an appropriated special metric is chosen. Real time execution is not necessary, in fact, classifying entire images takes longer to process than individual contours. However, the algorithm is not wasteful or inefficient and completes quickly on modern hardware.

10. Future Work

Depth images are still an up and coming technology and institutions are slowly beginning to understand them. Not only may this method be applied to segmentation, but also it can potentially be applied in combination with convolutional neural networks to classify entire images. For instance, hospitals diagnose patients everyday based on X-Rays, MRIs, and various other biomedical imaging techniques. Future work with this proposed algorithm could potentially be able to diagnose biomedical images. Depth image segmentation has great potential for expansion in the future as well as protecting the health of patients worldwide.

Bibliography

[1] N. Yokoya and M. D. Levine, "Range Image Segmentation Based on Differential Geometry: A Hybrid Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1989.

[2] G. J. Iddan and G. Yahav, "3D Imaging in the Studio," in *Proceedings of SPIE*, San Jose, 2001.

[3] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. XXXVIII, no. 5, pp. 133-138, 29 August 2011.

[4] G. Bradski and A. Kaehler, Learning OpenCV, First Edition ed., M. Loukides, Ed., Sebastopol, California: O'Reilly Media, Inc., 2008, p. 135.

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in neural information processing systems,* pp. 1097-1105, 2012.

[6] N. Kalchbrenner, E. Grefenstette and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," Cornell University Library, Ithaca, 2014.

[7] D. C. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber, "Convolutional Neural Network Committees For Handwritten Character Classification," in *2011 International Conference on Document Analysis and Recognition*, Beijing, 2011.

[8] O. Russakovsky, J. Deng and H. Su, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV),* vol. 115, no. 3, pp. 211-252, 2015.

[9] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," *Microsoft Research Cambridge & Xbox Incubation,* 2011.

[10] A. Kuznetsova, L. Leal-Taixé and B. Rosenhahn, "Real-time sign language recognition using a consumer depth camera," in *ICCV*, Hannover, 2013.

[11] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 9, no. 1, pp. 62-66, 1979.

[12] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza, Introduction to Autonomous Mobile Robots, Second Edition, Cambridge, MA: The MIT Press, 2011.

[13] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," *CVGIP,* vol. 30, no. 1, pp. 32-46, 1985.

[14] L. Yang and F. Albregtsen, "Fast and exact computation of cartesian geometric moments using discrete Green's theorem," *Pattern Recognition,* vol. 29, no. 7, pp. 1061-1073, July 1996.