Luggage Recognition using Image Processing

Rishub Jain

Period 5

June 16, 2015

Thomas Jefferson High School for Science and Technology

Mr. Stueben

Introduction

Currently, most of the luggage recognition is done through routing labels. A piece of plastic that has a barcode is attached to the handle of the luggage, and that serves as the only identification method for the people who are moving around the baggage. Although this works great most of the time, there are some major flaws that come with this method. First, if that label gets damaged, there is almost no other way that the airline can identify the baggage. Since luggage is often thrown around, and quickly put in and out of harsh climates, it is not uncommon for these tags to become damaged. Also, this method requires people to physically scan the barcode, and then physically place it in the correct category. Not only is this more expensive and less efficient, but also human error can lead to huge mistakes, such as accidentally loading the baggage on the wrong plane. Many airlines have issued RFID tags on the pieces of plastic attached to the luggage so that identification can become entirely automatic. However, these can be damaged even more easily, and are simply outdated for today’s technology. The goal of my project is to improve the luggage recognition inside airports using image processing techniques. My project provides a way to automate the sorting of luggage to reduce the human error introduced. Also, this program allows those who have lost their luggage to easily find where it is.

Body

My project proposes to originally identify the luggage by automatically taking a picture of it when it is checked in. Then, whenever the luggage must be separated, an image of the luggage is automatically taken again, and identified based on the original images taken of the luggage. Luggage would be automatically sorted this way, without any human involvement. Also, luggage is sometimes forgotten at airports, or dropped without noticing. If luggage is found out of place without any identification, an image can be taken of it, and using my program it can be identified and later can be given to its owner.

The fundamental problem of this project is to match two images of the same piece of luggage. This specific problem has not been completely solved before, yet the problem of matching two images displaying the same physical content is fairly common. Though general solutions have been created, such as in the Reverse Image Google Search, this tool does take into account the assumptions that can be made in this scenario, and the method of solving this problem can vastly change among different applications. In the case of luggage, I assumed that the images would be taken with a solid background, similar lighting, and that the distance from the camera to the base of the background would be constant, which is reasonable considering the images would be taken on a conveyor belt as the luggage is being transported to and from the plane. I originally had thought that I needed the orientation of the luggage to be consistent throughout the images, but I was able to change my project to allow for 90 degree rotations as long as the luggage was not tilted. This allowed for the colors and dimensions to generally remain the same across images of the same luggage.

To solve this problem, I gave each image two characteristics: the dimensions of the luggage and the dominant colors in that image. To first find the dimensions of the luggage, I generated the outline of the image using the Canny edge detection method, and then found the first and last black pixel values in the horizontal and vertical directions. This will not always work if the background is not solid, since the outline may provide stray black pixel values outside of the piece of luggage. However, I am just assuming that the background is solid.

I then found the number of dominant colors in the image. Using the k-means algorithm[1], I created a series of images in which the number of colors was restricted to a number from 2 to 7. I measured how different each image was from the original image. Then, I had explored a variety of different methods to choose the lowest number of colors that still was very close to the original image. That is, to choose the lowest value of *k* for which the original image can still be represented well enough.

The naïve method would be to graph variance of how each image that was restricted to a given number of colors differed from the original image, and then find where the variance graphed suddenly stops decreasing so steeply as the number of colors allowed increases, indicated by the circle in Figure 1. However, this is very inaccurate as finding this point can often be subjective.

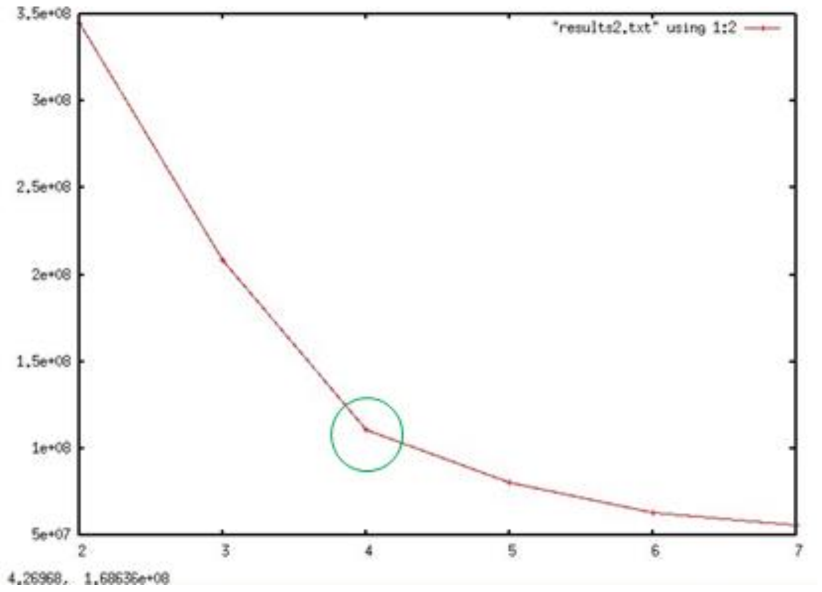

Figure 1: The variance graph and the correct *k* value

Another method, developed by Pham et al. (2005), finds this *k* value by simply finding the weights to an equation, and then just computing this equation. This runs very quickly, but it is highly inaccurate for small and medium sized data sets.

[1] The k-means clustering algorithm finds the optimal way to group a data set (pixel values) into a given number of clusters (colors), *k*. First, *k* different RGB values are chosen, and the image's pixels are divided into n different clusters by which RGB value they are closer to on the RGB color cube. Then, the average is found of each of these clusters, which becomes the new *k* different RGB values. The pixels are divided again into k different clusters, and this process repeats until the clusters do not change.

$$f(K) = \begin{cases} 1 & \text{if } K = 1 \\ \dfrac{S_K}{\alpha_K S_{K-1}} & \text{if } S_{K-1} \neq 0, \forall K > 1 \\ 1 & \text{if } S_{K-1} = 0, \forall K > 1 \end{cases}$$

$$\alpha_K = \begin{cases} 1 - \dfrac{3}{4N_\mathrm{d}} & \text{if } K = 2 \text{ and } N_\mathrm{d} > 1 \\ \\ \alpha_{K-1} + \dfrac{1 - \alpha_{K-1}}{6} & \text{if } K > 2 \text{ and } N_\mathrm{d} > 1 \end{cases}$$

Figure 2: Equation for the method by Pham et al. (2005)

A developed by Tibshirani and Walther (2005), called the prediction ptrength method, could also be used. This method randomly divides the data into two different sets, the Training and Test sets. The k means analysis is run on both of these sets for each value of *k*, and if the centroids of the clusters in the Training set are close to that of the Test set, that *k* value will have a high prediction strength. Although this method is more accurate than that of Pham et al., it is still inaccurate in specific cases, and it is very slow.

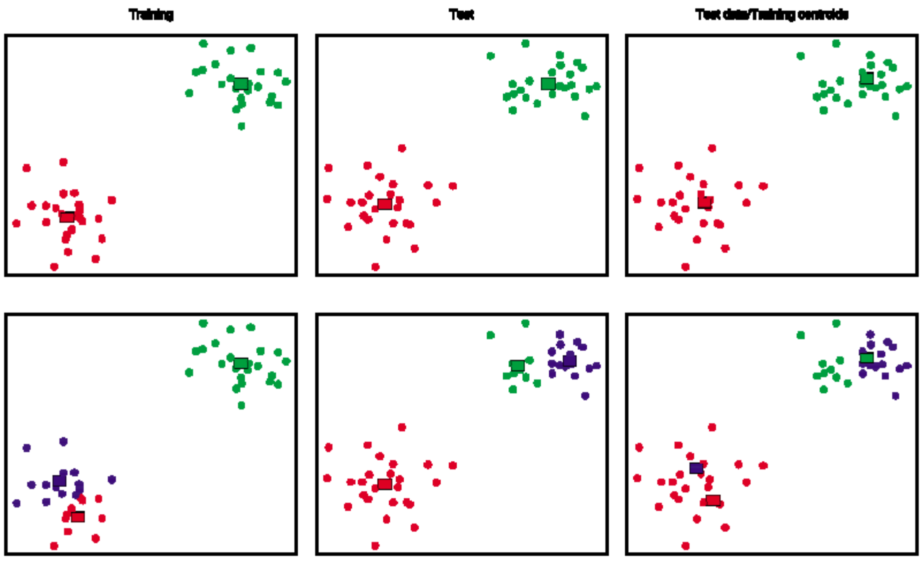

Figure 3: Illustration of the prediction strength method (Tibshirani et al., 2005)

Another method developed by Tibshirani et al. (2001) uses the gap statistic to find the correct *k* value. It generates a completely random data set in the same vicinity as the original data set, and then runs the k means analysis on both sets. It then calculates a function of the percent variance of each set as shown in Figure 4, similar to the elbow method. The method then selects the *k* value by finding when the difference between these two functions, or the gap statistic, suddenly increases. This method is much slower than that of Pham et al., but it takes is around the same time as the prediction strength method. Also, it is consistently accurate throughout many different values of *k* and data sizes.

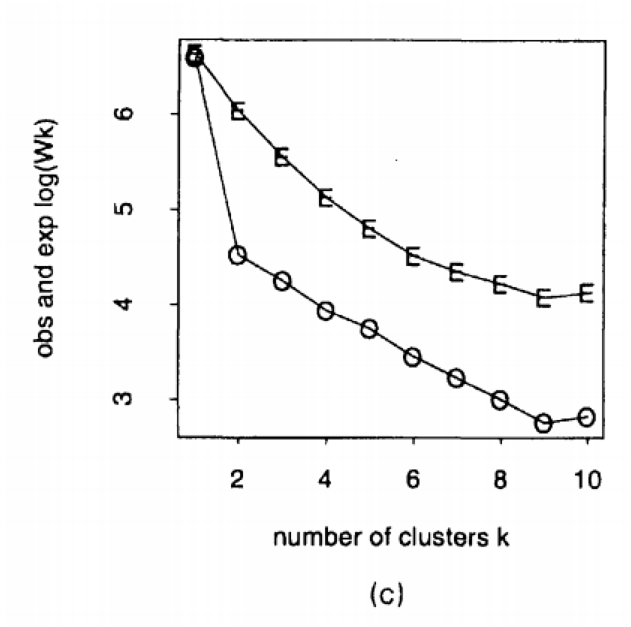

Figure 4: The variance of the k means analysis on the random data, shown by E, and the original data, shown by O (Tibshirani et al., 2001)

I chose to use the gap statistic method because my project needed that type of accuracy, and speed was not much of a factor because the database of images would be preprocessed. I then stored the colors associated with the image, along with the dimensions, and compared them to other images to find the same luggage in a different image.

Conclusion

There are other methods to find the number of dominant colors that I was not able to explore, such as the method of Krzanowski and Lai and the contrast statistic method, and these may be more suitable for this project. Also, though the gap statistic is very accurate, it sometimes takes several minutes to run, and this may cause this luggage recognition to become impractical because in many cases the images need to be processed quickly.

The biggest problem with this project is when two luggage exist that look exactly the same, which is common in airports. Since the luggage identification would not be able to know exactly what luggage it is from the image, the automation of the sorting of luggage is not yet practical. However, my program can provide a list of luggage that seem the most similar to the luggage provided, giving someone many options that may aid him in finding the correct owner of a forgotten piece of luggage.

References

Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *219*(1), 103-119.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411-423.

Tibshirani, R., & Walther, G. (2005). Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics*, *14*(3), 511-528.