Image Analysis - TP6 - Background Subtraction

Jander Nascimento, Raquel Oliveira

May 25, 2011

1 Introduction

The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. [1]

Segmentation is mostly used for object recognition, differentiate those objects from the background, image compression or image editing.

In this practical work, we implemented an approach for image segmentation for background subtraction, with which is possible to identify the foreground region in an image given a set of images of the background.

2 The method

In this exercise we use a Gaussian model to build a statistical model of the background for each pixel in the image. For each colored pixel i, i.e., $i=(i_r, i_g, i_b)$, the probability density applied is described in equation 1:

$$p_b(i) = \frac{1}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}} exp(\frac{-1}{2}(i - i_m)^T \Sigma^{-1}(i - i_m)) \tag{1}$$

where i_m is the mean value with respect to the given set of N values i_n:

$$i_m = \frac{1}{N}\sum_{n=1}^{N} i_n \tag{2}$$

and Σ is the covariance matrix:

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(i_n - i_m)(i_n - i_m)^T \tag{3}$$

3 Results

We implemented the equation 1 step by step. First creating all the methods that would be useful to generate the model, as subtraction and multiplication between matrices; transpose, inverse and determinant of a matrix and the covariance matrix.

We calculate the mean matrix once, in the beginning of the method. Over there we stored the mean value for each pixel (i_m in the formula above) with respect to their colors in all the background images considered.

With the previous methods was possible to build the Gaussian model for all pixels in the images in the background. We were given 115 images of the static background, to build the model (Figure 1).

Figure 1: Example of background

We tested the statistical model with the 5 given foreground images. For each pixel, based on the model, we can determine if the pixel belongs either to a foreground or to a background of the image.

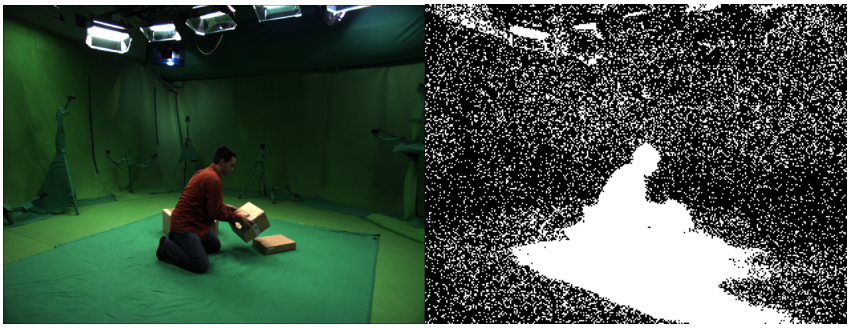
Given an input image with a foreground, we create a binary image with two regions: foreground and background. What determines which region the pixel belongs to is the probability given by the Gaussian model. To do so, it is needed to threshold the probability for each pixel.

The probability that a pixel belongs to a background is given by the formula described in equation 1. And the probability that a pixel belongs to a foreground is:

$$p_f(i) = 1 - p_b(i) \tag{4}$$

We tested some thresholds to determine which one would be more suitable to properly distribute the pixels in the 2 regions, and the value that gave better results was 0.01. So, if $p_b(i) > 0,01$, means that the pixel belongs to the background (which we painted in black). Otherwise, it belongs to the foreground (which we painted in white).

We can see in Figure 2 the results of the algorithm for a image with a foreground. To obtain this result, all the 115 background images were used to built the statistical model.



(a) Original Image (b) Binary Image

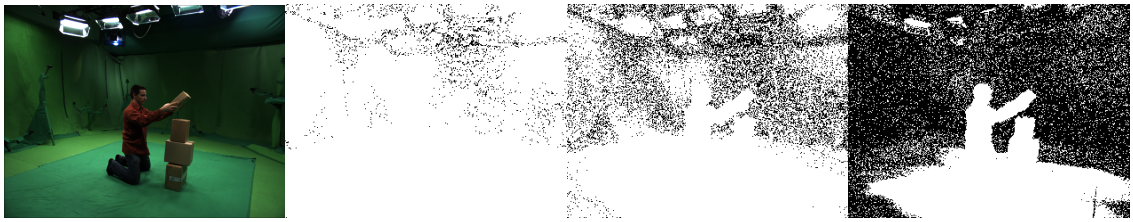
Figure 2: Background Subtraction

3.1 Effect of the threshold

We observed that the lower the threshold is, the better is the result achieved. In the Figure 3 we can see the effect of different thresholds (denoted by T) with the same number of background images (115).

We used the threshold in the following way:

```
T:=0.01
IF p(i)>T
        pixel belongs to background
ELSE
        pixel belongs to foreground
```

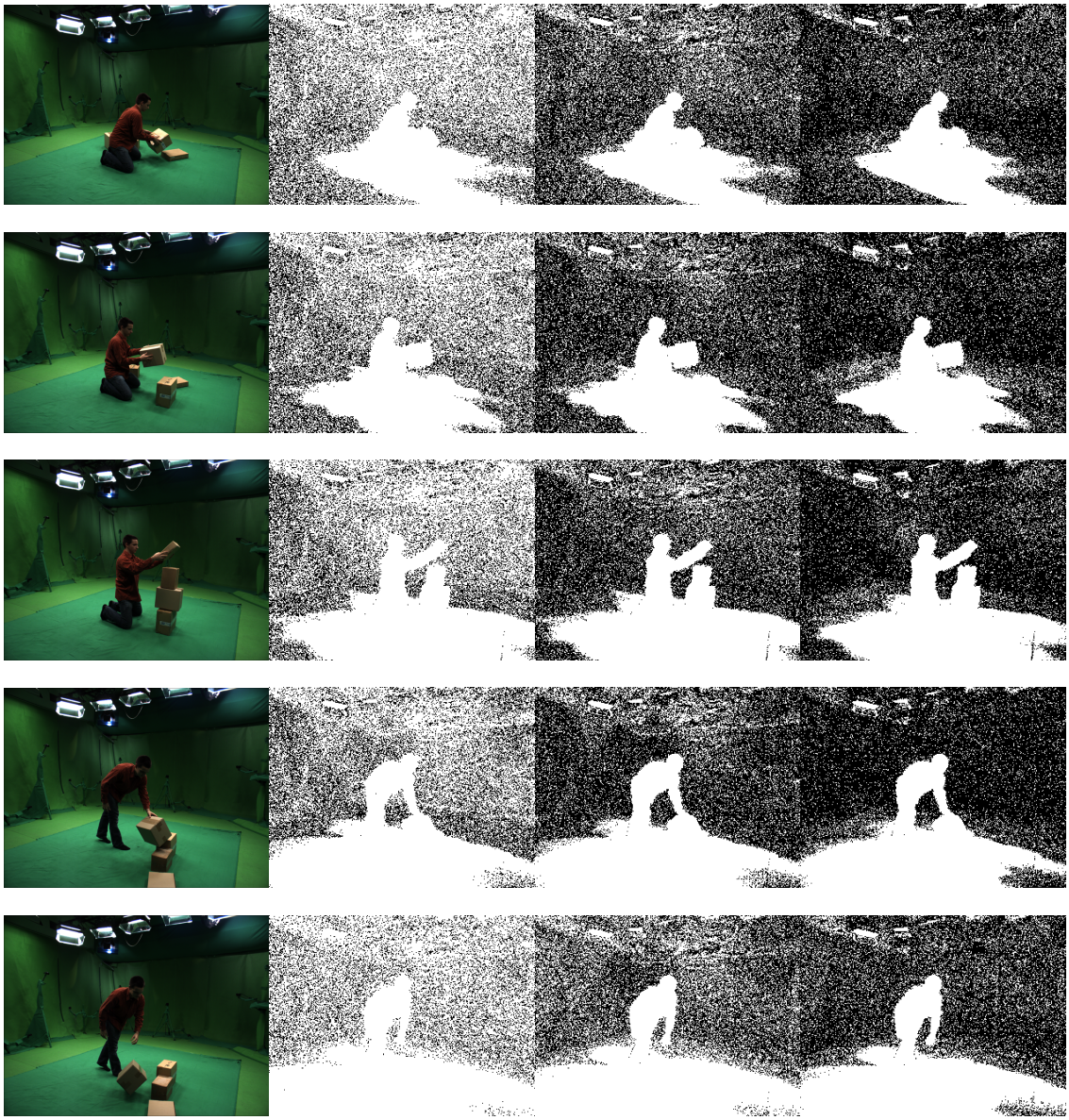


(a) Original Image (b) T=0,5 (c) T=0,1 (d) T=0,01

Figure 3: Effect of the threshold (T)

3.2 Effect of the number of background images

We observed that the more static background images you use to build the statistical model, better the result is. Which means that the pixels will be more correctly classified in their segments. We illustrate this in the Figure 4, with different number of background images being used (denoted by N). However, the computational cost increases proportionally with the number of background images used to build the statistic model.

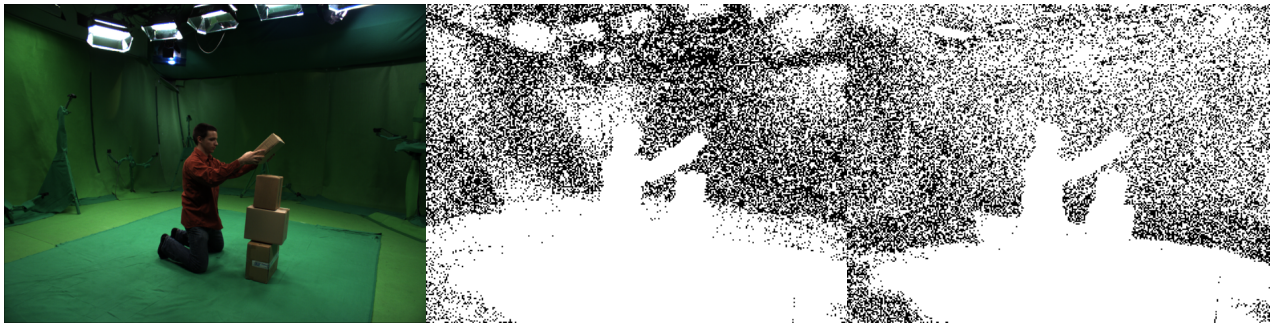


(q) Original Image (r) Binary Image N=5 (s) N=15 (t) N=115

Figure 4: Effect of the number of background images (N)

3.3 Comparison between the effects of the threshold and the number of background image

As said before, the computational cost of the algorithm increases proportionally with number of background images used to build the statistical model. To deal with this issue, it is possible to play with both parameters to improve the results. In the Figure 5 we show an example where both parameters are different. The image 5b is the result of applying the algorithm with 115 background images, and a threshold equal to 0,06. To optimize the time, you could decrease the number of background images to 5 and also decrease the threshold to 0,01 (as in image 5c), and the result will be equivalent.



(a) Original Image (b) N=115 T=0,06 (c) N=5 T=0,01

Figure 5: Effect of both threshold (T) and number of backgrounds (N)

4 Issues

This method consumes much memory. In order to build the Gaussian model, for each pixel of the foreground image all the background images are read, and each background is an image with dimensions 1224x1624, which leads to 1.987.776 pixels in memory for one single image. Since we are using all the 115 background images, in the end of the day it is necessary to keep in memory 228.594.240 pixels!

As a consequence, the computation required to separate foreground and background is very high. The standard implementation we used so far to read ppm images was too slow to apply this method. Our read method by default reads the whole image and stores its pixels in a matrix(in memory), which we can use for any computation.

To solve both problems of memory requirement and computation time, we changed our read image method to open a file and read the RGB values of only one pixel, instead of read and store all the pixels in a matrix, by calculating the position of the pixel inside a binary image file and positioning the file cursor in the exact pixel position. That change made possible to apply the method and achieve the results describe above.

This was possible just because we converted the given images in a binary format of PPM, instead of the original PNG format.

5 Equilikely foreground model

Without any particular knowledge of the foreground image colors, if we would like all colors to be equilikely in the foreground model, we could set:

$$p_f(i) = cst \tag{5}$$

In order to achieve this, we should change the algorithm in the following way: instead of calculate the equation 1 to build the Gaussian model, we just threshold p_f with a constant value cst, and set p_b to be:

$$p_b(i) = 1 - p_f(i) \tag{6}$$

6 How to run?

Steps to compile the application:

- svn checkout https://jfimageanalysis.googlecode.com/svn/trunk/TP6/ #download source code
- make #compiles the code

Example of usage:

- ./tp6

The resulting images will be printed in the standard output.

References

[1] Linda G. Shapiro and George C. Stockman *Computer Vision*. pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3 2001.