Exercise in Computer Vision
A Comparison of Thresholding Methods
–NTNU–

Per Christian Henden

20th November 2004

Abstract

This paper contains a comparison of common, simple thresholding methods. Basic thresholding, two-band thresholding, optimal thresholding (Calvard Riddler), adaptive thresholding, and p-tile thresholding is compared.

The different thresholding methods has been implemented in the programming language c, using the image analysis library Xite[1]. The program sources should accompany this paper.

1 Methods of thresholding

Basic thresholding. Basic thresholding is done by visiting each pixel site in the image, and set the pixel to maximum value if its value is above or equal to a given threshold value and to the minimum value if the threshold value is below the pixels value. Basic thresholding is often used as a step in other thresholding algorithms.
Implemented by the function *threshold* in thresholding.h

Band thresholding. Band thresholding is similar to basic thresholding, but has two threshold values, and set the pixel site to maximum value if the pixels intensity value is between or at the threshold values, else it it set to minimum.
Implemented by the function *bandthresholding2* in thresholding.h

P-tile thresholding. P-tile is a method for choosing the threshold value to input to the "basic thresholding" algorithm. P-tile means "Percentile", and the threshold is chosen to be the intensity value where the cumulative sum of pixel intensities is closest to the percentile.
Implemented by the function *ptileThreshold* in thresholding.h

Optimal thresholding. Optimal thresholding selects a threshold value that is statistically optimal, based on the contents of the image. Algorithm, due to Calvard and Riddler:

[1] http://www.ifi.uio.no/forskning/grupper/dsb/Programvare/Xite/

1. Assuming no knowledge about the exact location of objects,
   consider as a first approximation that the four corners of the image contain background pixels only and the remainder contains object pixels.

2. At step t, compute μ_B^t and μ_O^t as the mean background and object gray-level, respectively, where segmentation into background and objects at step t is defined by the threshold value T^t determined in the previous step.

$$\mu_B^t = \frac{\sum_{(i,j)\in background} f(i,j)}{\#background\_pixels} \tag{1}$$

$$\mu_O^t = \frac{\sum_{(i,j)\in objects} f(i,j)}{\#object\_pixels} \tag{2}$$

3. Set

$$T^{(t+1)} = \frac{\mu_B^t + \mu_O^t}{2} \tag{3}$$

   $T^{(t+1)}$ now provides an updated background–object distinction.

4. If $T^{(t+1)} = T^t$, halt, otherwise return to step 2.

Implemented by the function *findThreshold* in thresholding.h

Adaptive thresholding. Adaptive thresholding divides the image into patches, and each patch is thresholded by a threshold value that depends on the patch contents (the threshold *adapts*). The threshold value of a patch is chosen to be a weighted sum of the mean intensity value of the patch and a global threshold value. The global threshold value is chosen to be the optimal threshold (see above).
Implemented by the function *adaptiveThreshold* in thresholding.h

2 Comparison

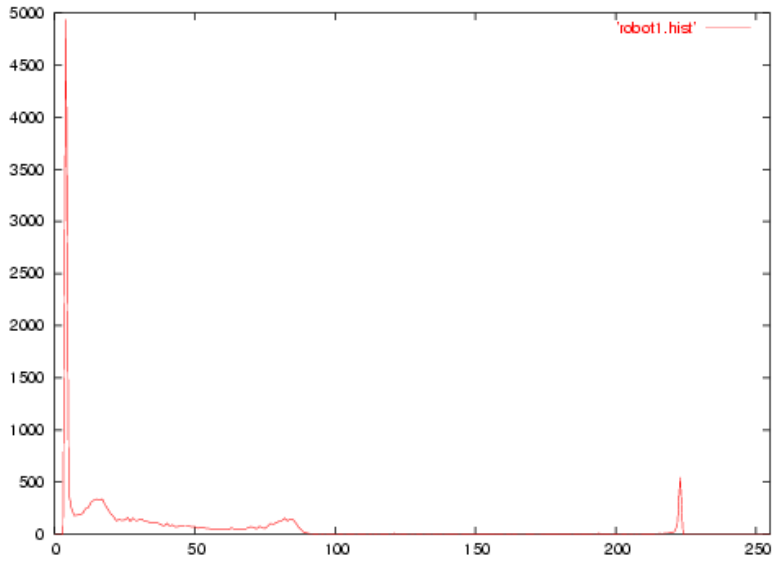
The different methods of thresholding have been applied to three images, which illustrate the shortcomings and advantages to the different methods. The various algorithm parameters have been optimized for each image.

First out, we have an image from the view of a robot (see figure 2 on page 4). There are different tools on the table, but varying lighting and other effects make the tools difficult to detect. The histogram of the image is shown in figure 2 on the next page. We can tell from the image that the unwanted information in the image, the light effects, have a different color than the objects. Because of this, basic thresholding works well, and solves the problem of segmentation entirely.

When using **basic** threshold, a threshold value has to be selected somehow. This was done manually. Interestingly, the method of **optimal** thresholding selected a better threshold value than what was found manually. **Band** thresholding is not meaningful to do for this image, because the histogram does not need to be divided into more than two to do correct

thresholding. **Adaptive** thresholding manages to get the borders of the shapes slightly more correct, but also produces a little more junk. <*step*> is the parameter that decides how many patches the image is divided into along each axis. A step value of 32 in this image, which is of size 128×128, means that patch size is 4×4 pixels. **Percentile** (Ptile) thresholding can be used for automatic threshold selection if a priori knowledge about the image is known, and the objects we are interested in is in the first P% of the area below the histogram curve.

Figure 1: Histogram of robot image

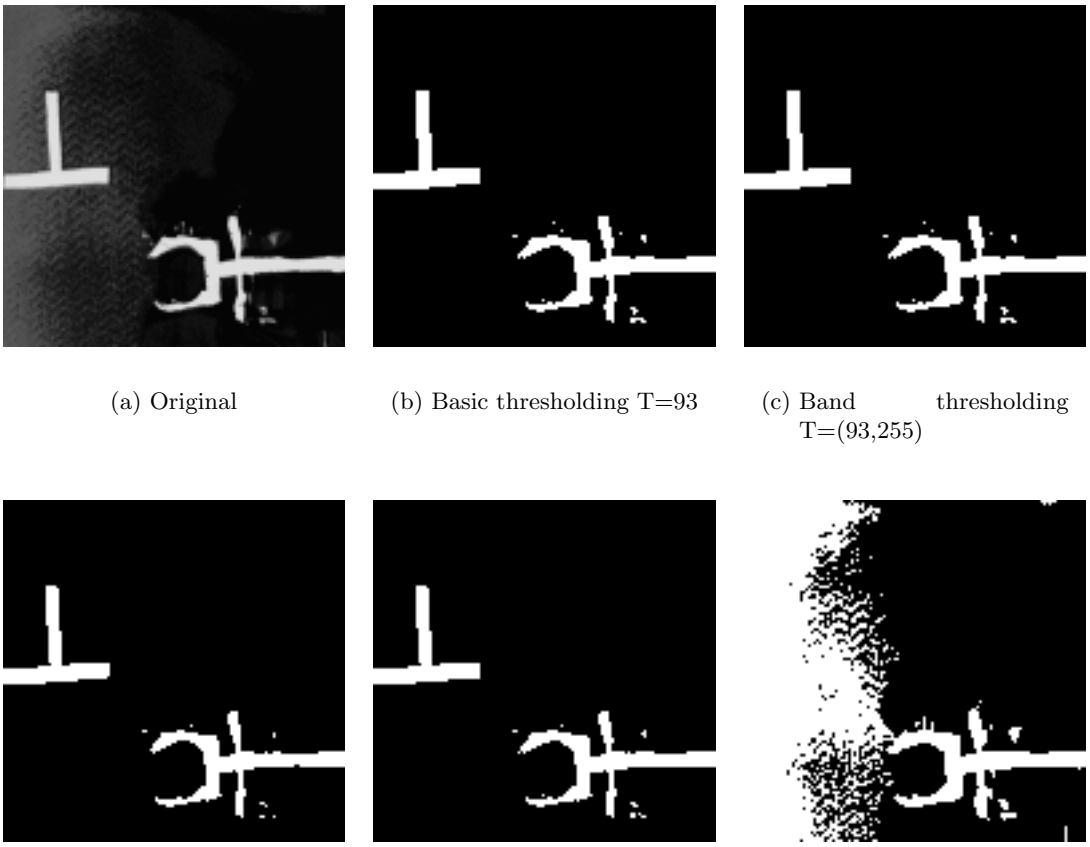


Our second example is the image (see figure 4 on page 5) of a electronic chip. The chip is connected to a surface that is connected to another surface. The task is now to separate the chip in the image from the rest of the image.

Global, **basic** thresholding cannot solve this problem. The chip's colours has an intensity that lies between the colours of the two surfaces. As we see from the image's histogram in figure 2 on page 5, a single threshold value cannot separate the second peak from the first and the third. **Optimal** and **Ptile** depends on basic thresholding, and fails for this reason. **Band** thresholding is needed, and is the only method of the ones examined that produced the correct result. The vertical line on the left is a remnant of the border between the surfaces. **Adaptive** thresholding takes local values into account, but that does not make for correct segmentation, because varying the threshold with image location cannot divide the input in three, as is needed. The reason Ptile and adaptive look so similar is that the adaptive one is heavily (weight 0.90) based on local values.

Our final example image (see figure 6 on page 6) is an image of a crest where the lighting varies from bright to dark in the direction left to right. The histogram (see figure 2 on page 6) shows four peaks. The last peak is the white frame on the left, and the first and second peaks, respectively, are the darkest parts of the left and the right side of the image. The third peak is the woman in the center of the image. We want both the dots and the woman to stay white.

Because of the light-effect will **basic** thresholding not produce the same effect on the right half



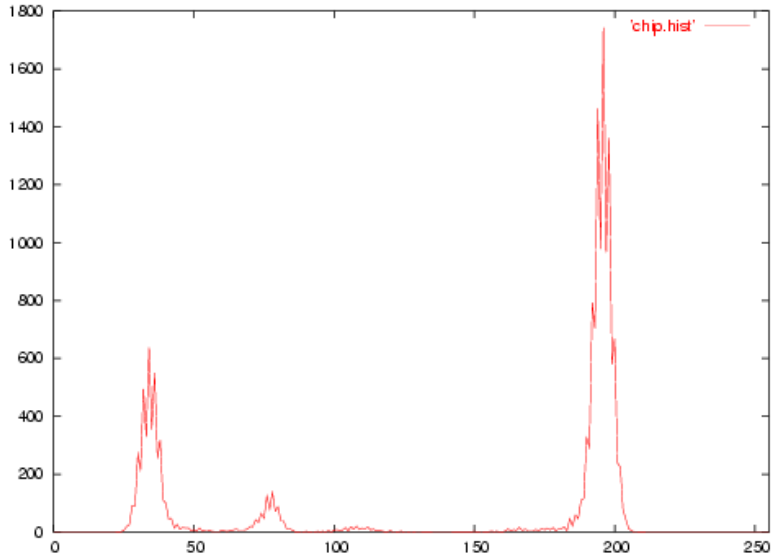
(a) Original (b) Basic thresholding T=93 (c) Band thresholding T=(93,255)

(d) Optimal (e) Adaptive thresholding, step=32, weight=0.20 (f) Ptile thresholding, T=65%

Figure 2: Robot image

of the image as on the left part, resulting in incorrect segmentation. This means that **Ptile**, and **optimal** thresholding will not succeed, because they are just means to automatically find the single threshold value (basic threshold value). **Band** thresholding suffers from the same problem as basic thresholding. It is however, the perfect task for **adaptive** thresholding. By varying the threshold value, the method *adapts* to the lighting, and produce a good result.

Figure 3: Histogram of chip image



(a) Original

(b) Basic thresholding T=128

(c) Band thresholding T=60,90

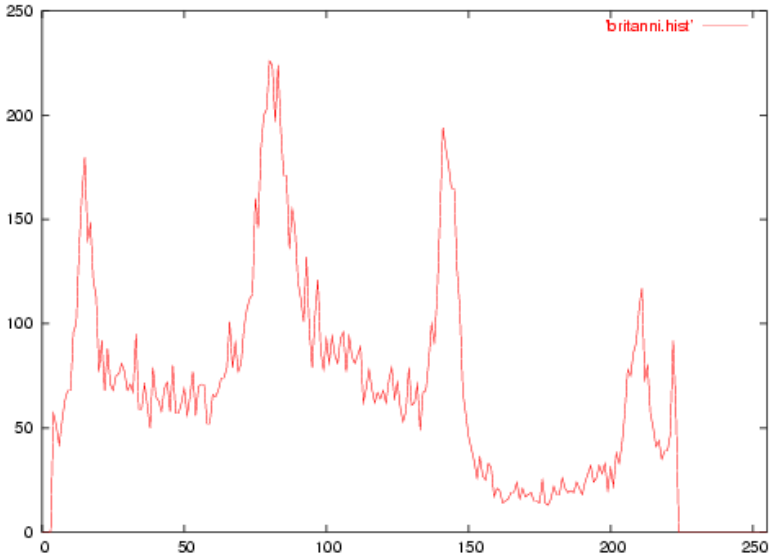
(d) Optimal

(e) Adaptive thresholding, step=32, weight=0.90

(f) Ptile thresholding, T=25%

Figure 4: Chip image

Figure 5: Histogram of Britannia image



(a) Original

(b) Basic thresholding T=110

(c) Band thresholding T=(65,160)

(d) Optimal

(e) Adaptive thresholding, step=32, weight=0.70

(f) Ptile thresholding, T=65%

Figure 6: Britannia image

3 Discussion and summary

Basic thresholding is a good choice if objects and background can be separated in the histogram by a vertical line.

Optimal threshold selection (Calvard and Riddler) and P-tile threshold selection are two methods to find this line by automatic means. Optimal threshold selection will always find this line if it exists, and if it does not it will make a good (optimal) choice anyway, and the results are often good in that case too, but they are not guaranteed to be good (Sonka et. al 1999, Image Processing, Analysis, and Machine vision, p. 129).

Ptile thresholding can be used for automatic threshold selection if a priori knowledge about the image is known, and the objects we are interested in is in the first P% of the area below the histogram curve. If these conditions are fulfilled, then the method will in all cases be better than optimal threshold selection because the algorithm is computationally cheaper, easier to implement, and has an upper limit on the worst-case runtime, in contrast to the optimal (Calvard and Riddler) one.

Band thresholding, which divides the image into three or more bands, is sometimes necessary to extract the information you want.

All of the methods above can break on different lighting in the image, as in figure 6 on the preceding page. Adaptive thresholding solves the problem varying light presents by adapting to the local lighting conditions. It is the most sophisticated method of the ones discussed.