Determining bark content in wood-chips for pulping with computerized image analysis

Kristin Norell

1st February 2005

Abstract

This thesis examines the possibility to use computerized image analysis for quality control of wood-chips used for pulping. The quality control in this case means controlling the amount of bark in the chips, since bark can create damages in the finished product. At present the chips quality at Stora Enso Packaging Boards, Skoghall Mill is controlled with samples where the amount of bark is evaluated. The evaluation is done by sorting out the bark in the sample and calculating the weight percentage.

To examine if this controll is possible to do with computerized image analysis two different sets of images were used. Image series 1 was used for developing and evaluating the method and image series 2 was used only for evaluation. The results from image series 1 are good and shows that it is possible to separate the bark from the rest of the pieces. In image series 2 the results are not quite as good. Not all bark pieces are found, and some other pieces are instead identified as bark. The reason for this is that the image material for developing the algorithm was not large enough. Continuing to develop the method with more images from different sets of wood-chips would probably give even better results.

Contents

1 Introduction

Pulp is made from small pieces of wood, called chips. This thesis examine the possibility of evaluating the chip quality with computerized image analysis.

The procedure of making wood-pulp is sensitive to different kinds of disturbances. One of these disturbances is the amount of bark in the pulp. In the ideal situation there would be no bark at all, but that is not the case. Making wood-pulp begins with de-barking the wood. The de-barking is done while the wood is still in the form of logs and it could for example be done in a big drum where logs are fed in together with water. As the drum rotates most of the bark will wear off. After the de-barking the logs are made into chips by feeding them into a big chopper. The chips vary in size but are often about 3×3 cm and about 1 cm thick. A bad de-barking results in bark together with the chips and that can cause problems in the pulp. In the CTMP (chemi-termomechanical) process, used for example at Stora Enso Packaging Boards, Skoghall Mill, bark can create colored dots in the resulting product. If the wood is not fresh there is also problems with the bast, that is, the outermost layer of wood under the bark. When the wood is fresh the bast have the same color as the wood, but as the wood gets older the bast gets a color more like the bark and therefore the bast too can create problems in the CTMP process. The CTMP process is a combination of two methods in making pulp: mechanical pulping and chemical pulping. The mechanical pulping is a method where the fibers are separated by refining, in which the chips are grinded against a stone. In the chemical process chemicals are used to separate the fibers. In the CTMP process the fibers are first separated in a chemical way, and then with mechanical refining. For more information on how paper is made see [1].

Parts of the chips used at Skoghall Mill, are de-barked and chipped at a sawmill located elsewhere and transported to Skoghall with trucks. The quality of these wood chips vary between different saw mills and also from time to time at the same saw mill. At present the chips quality at Skoghall Mill is controlled in different samples where for example the amount of bark is evaluated. The evaluation is done by sorting out the bark in the sample and calculating the weight percentage. This is a time consuming process and when the result is available the chips may already be consumed in the pulp process.

2 Problem description

The purpose of this thesis was to examine the possibility of using image analysis for quality control of the incoming wood chips. Instead of (or as a complement to) the samples taken from the trucks, a system with a digital camera and a computer could be used. The camera would be placed above the trucks as they stop at the measurement station at Skoghall Mill and digital images would be taken from above on the wood chips. The images would then be processed in a computer with the image analysis program. The process should be totally automatic and would result in an evaluation of the amount of bark in the truck. The image analysis program should identify the bark and calculate the amount of bark in an image. By assuming that the fraction of bark is uniformly distributed in the truck, an estimate about the chips quality can be made.

One difficulty with the segmentation of bark, that is finding the bark in the

images, is that the images contain bast and a lot of shadows between the chips. The shadows are dark and both the shadows and the bast have the same color as the bark and thus it is not trivial to separate between these. However, they do differ in texture, so separation is possible.

3 Data

Two different sets of color images were used. Images series 1 was used for developing the method and for evaluation. Image series 2 was used only for evaluation. The two different image series are taken from different loads of chips with different freshness and different pieces of bark.

The images have a pixel depth of 24 bits and the size is 2704×4064 pixels. 100 pixels in the images correspond to about 8.5 mm of chips. Figure 1 shows an image from image series 1. All images used were created in a photo studio with the equipment seen in table 1. The exposure was 1/250 seconds, diaphragm 22 and the distance from the camera to the chips was 60 cm.

Table 1: Equipment

Camera:	Canon EOS 1-DS
Lens:	Canon compact-Macro Lens EF 50 mm 1:2,5
4 Flash generators:	PRO-7B 1200
4 Flash heads:	PRO-7

Figure 1: An image from image series 1.

4 Theory

4.1 Digital images

A digital image is an image with discrete values both in its spatial coordinates and in the intensity. Each pair of coordinates in the image represents a picture element, a pixel, and every pixel has an intensity-value. In a gray scale image a low value is a dark pixel and a high value is a bright pixel. The value depends on the number of bits that can be represented. For an 8-bit image the highest possible value is 255, which corresponds to a white pixel and the lowest value is 0, which correspond to a black pixel.

4.2 Relationships between pixels

A pixel, p, with coordinates (x,y) has two horizontal and two vertical neighbors with the coordinates (x-1,y), (x+1,y), (x,y-1) and (x,y+1). These pixels are called the 4-neighbors of p. Besides the 4-neighbors, p has also four diagonal neighbors given by the coordinates (x-1,y-1), (x+1,y-1), (x+1,y+1) and (x-1,y+1). These pixels together with the 4-neighbors are called the 8-neighbors of p.

Two pixels, p and q, are connected if they are neighbors and if their values satisfy a specified condition, for example, if they have the same grey value. If S is a subset of an image, two pixels, p and q, are connected in S if there exists a path between p and q consisting entirely of pixels in S. The set of pixels connected to a pixel, p in S, is the connected component of S.

4.3 Color Images

There are many ways to represent colors in an image. Some are more abstract and oriented at hardware and some more oriented towards the way that humans see colors. One common thing with these standard methods is that the color in one pixel is not represented by only one number as in the gray scale images, but usually by three different values each corresponding to a certain property of that color.

4.3.1 The RGB model

The RGB (red, green, blue) model is often used in color monitors and color video cameras. By combining light from the three primary colors red, green and blue most colors can be represented. The amount of one primary color in a pixel is given by an intensity-value similar to the one in gray scale images. This means that the three different colors red, green and blue each create their own gray scale image and combining them results in a color image.

The RGB model is often illustrated with a color cube (see Figure 2) where the color intensity has been scaled down to vary only between 0 and 1. Each primary color is shown on a principal axes in the coordinate system. Black is placed in the origin and white in position$(1, 1)$, which is the opposite corner from black. The grey scale is found on the diagonal between the black and white corners (shown as a dotted line in Figure 2). For each of the grey scale values the primary colors have identical intensities. That is, if the red, green and blue component all have the same value, that color is on the grey scale. It

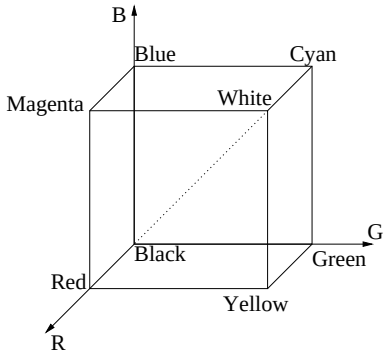


Figure 2: The RGB color cube. The dotted line shows the grey scale.

is easy to see that white is achieved by combining the three primary colors with maximum intensity.

The number of bits used to represent each pixel in RGB space is called the pixel depth. If the R, G and B values each are represented by an 8 bit value the color pixels are said to have a depth of 24 bits (3×8 bits).

4.3.2 HSI model

Another way to represent a color image is with the HSI (hue, saturation and intensity) model. This model is perhaps more intuitive since, it is similar to how we are used to think about colors. A color is separated into its hue, saturation and intensity. The hue is the color tone, such as yellow, blue or orange. Saturation is the degree of white in the color tone. A high value in saturation means much white. The intensity value tells the brightness of the color.

A variation of the HSI model is the HSV (hue, saturation and value) model. Hue and saturation are the same as in the HSI model but the value V is not the same as the intensity, instead it also depends on the saturation.

The hue, saturation, intensity and value can be calculated from the RGB values in the following way (see [2] and [3] for more information):

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases},$$

with

$$\theta = \arccos\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{\frac{1}{2}}} \right\},$$
$$S = 1 - \frac{3}{(R+G+B)}[min(R,G,B)],$$
$$I = \frac{1}{3}(R+G+B),$$
$$V = max(R,G,B).$$

The RGB values are assumed to vary between 0 and 1 and so does the saturation, intensity and value. The hue varies between 0 and 360°.

4.4 Thresholding

Thresholding is a simple method for segmentation. The segmentation of an image means separating the interesting parts, in this case the bark, from the rest of the image, the background. Thresholding is a discrete function taking a grey scale image as input and resulting in a binary image. A certain thresholding value is chosen. If a pixel value in the grey scale image is larger than the threshold it is set to 1, and if it is smaller it is set to 0. Thus

$$g(x,y) = \begin{cases} 1 & \text{for } f(x,y) \geq T \\ 0 & \text{for } f(x,y) < T \end{cases},$$

where f is the image and g is the output image.

Sometimes more than one variable are used to characterize a pixel, such as the RGB values in a color image. In that case multi-spectral thresholding can be used. When thresholding each component at a time the result is a box in the RGB space that defines the thresholding values. With multi-spectral thresholding other properties can be used such as a pixels closeness to a certain cluster of pixels. In that case the volume in the RGB space can have different shape which often lead to better results.

4.5 Histogram

The histogram is a discrete function that shows the distribution of different intensities in a grey scale image. In the histogram the number of pixels per intensity level are plotted against the intensity level. The histogram is usually normalized with the total number of pixels, n, in the image. For a grey level image with values in the range [0,L] the probability for grey level k is

$$p(k) = \frac{n_k}{n} \qquad \text{for } k = 0, 1, \cdots, L,$$

where n_k is the number of pixels having grey level k.

For an image with a dark background and a bright object the resulting histogram will have two peaks, one representing the background and the other representing the object. Thresholding with the threshold value in the valley between the peeks will then result in a segmented object (see Figure 3).

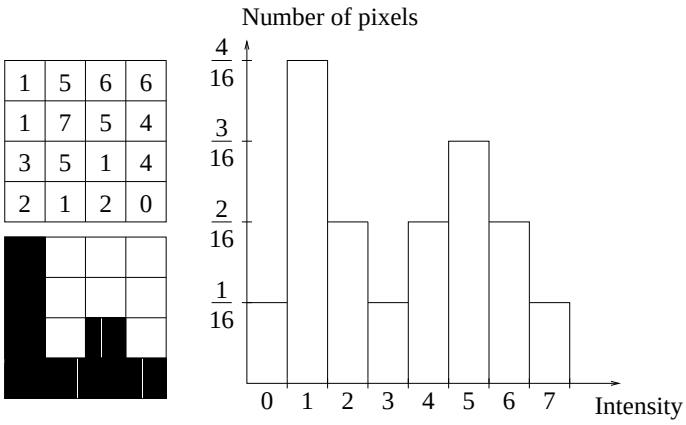


Figure 3: Original image (top left), thresholded image (bottom left) and histogram (right).

4.6 Spatial Filtering

Filtering images can be useful for a number of different applications such as noise reduction or edge detection. When filtering an image in the spatial domain the filter is in the form of a matrix (a mask) and the matrix values are called weights. This is illustrated in Figure 4. When filtering an image with a filter with an odd size, the center of the mask, $w(x, y)$, is placed over a pixel, $f(x, y)$, and the pixel values are multiplied by the overlaying weights and summed together. The resulting value is the new value, $g(x, y)$, for the filtered pixel. The filter is moved to the next pixel and the same procedure is done. All the calculations are made with the original pixel values. Filtering a pixel can be described as:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t),$$

where x and y are the coordinates for the filtered pixel, g is the resulting value, w is the filter weight, f is the input image value and a and b are related to the filter size such as if the filter is $m \times n$ pixels and m and n are odd numbers $a = (m-1)/2$ and $b = (n-1)/2$.

After filtering an image the resulting intensity levels can be too large to represent in the image, or they can be negative. That is, if the original image is an 8 bit image with values between 0 and 255, some of the values after filtering may be negative or larger than 255. One solution to that problem is to rescale all the values. If, for example, the values vary between -65 and 512 this can be done by adding 65 to all the values, dividing all values with $(512 + 65)$ and at last multiply every value with 255 and rounding to the nearest integer. All values will now be in the range $0 - 255$.

One thing to think about when filtering is what to do near the edges of the image. What happens when the filter is partly outside the image? One way to deal with that is to filter only the pixels where the whole filter fits inside the image. That is, if the filter is $5 * 5$ pixels the two out-most rows and columns

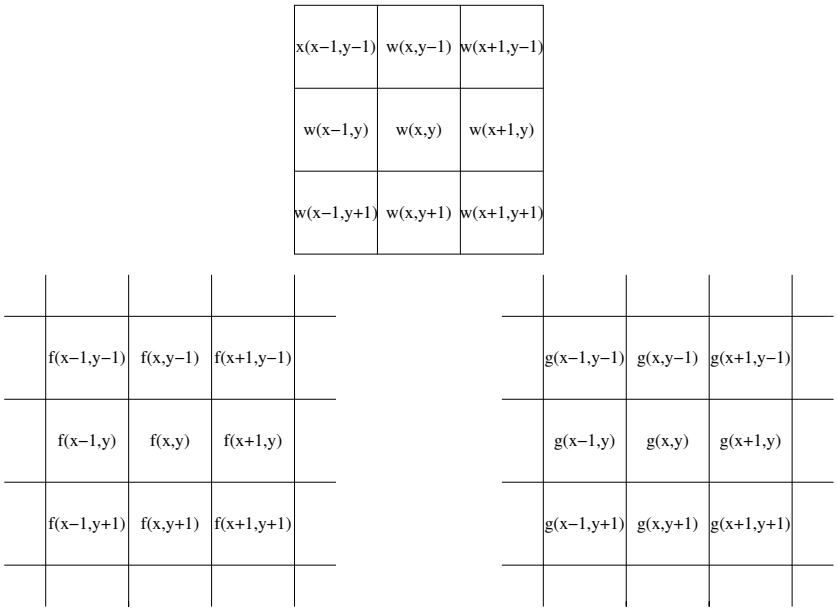


Figure 4: Spatial filtering. Original image (left), filter mask (above) and filtered image(right).

are ignored and the filtering starts two pixels in from the edge. That will result in a smaller image though, since some pixels on the edge of the image are in that case not filtered at all. Another way is only to use the filter values that are inside the image when the filter is centered over a pixel. That means if one row of the filter is outside the image, that row will be ignored in the calculations. This will result in an image of the same size as the original image. A third way is to enlarge the original image in some way, for example by mirroring the out-most rows and columns in the image edge. If the filter mask again is $5 * 5$ pixels the image should be enlarged with two rows or columns on each of the four sides. Then the image can be filtered as in the first method above, and the resulting image will still have the same size as the original image.

4.6.1 Laplace Filter

To detect edges in an image is the same thing as to detect discontinuities in the pixel values. There are a number of different ways to do this, e.g., using the first- or second order derivative. The sharper an edge is, the larger is the difference in pixel-values in the near-lying pixels defining the edge.

The Laplace filter is a method that use the second order derivative. It has two main properties. One is that the sum of all weights in the filter is zero. The other property is that the filter is isotropic, that is the response is independent of the direction of the discontinuities. The Laplacian is given by [4]

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

The discrete form of the second order derivative can be defined as

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

With two variables the following notation is used

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y).$$

The discrete Laplacian will then look like

$$\begin{aligned}\nabla^2 f &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\ &= f(x+1, y) + f(x-1, y) - 2f(x, y) + f(x, y+1) + f(x, y-1) - 2f(x, y) \\ &= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y).\end{aligned}$$

Applying this to a filter mask results in the filter shown in Figure 5.

0	1	0
1	−4	1
0	1	0

Figure 5: Laplace filter, 3×3 pixels.

The Laplacian is sensitive not only to edges but also to single pixels having a different value compared to the near-lying pixels. This is a result of the fact that the filter is isotropic. A comparison with the Sobel filters in Figure 6 shows a big difference. Sobel filters use the first order derivative and are not

1	0	−1
2	0	−2
1	0	−1

−1	−2	−1
0	0	0
1	2	1

Figure 6: Sobel filter masks for detection of vertical edges (left) and horizontal edges (right).

isotropic. Instead one filter is needed for each direction in which edge detection is wanted. The Sobel coefficients are more fit to identify a line than the Laplace filter, where the coefficients are rather formed to identify a dot. This can be a problem with a noisy image since the noise will be amplified. Figure 7 shows an image filtered with a vertical Sobel filter and with a Laplace filter.

Figure 7: Original image (left), result after filtering with a vertical Sobel filter (middle) and result after filtering with a Laplace filter (right).

5 Method Description

The key to get a good approximation of the amount of bark in an image is a good segmentation. If the segmentation is bad, the result will be too. Therefore the main part of the work has been concentrated on the problem of segmentation. There are three steps in the segmentation process: thresholding, filtering and calculating and analyzing the histograms. Analyzing the histograms is the most important step to separate between bark, bast and shadows. After the segmentation some area calculations were made to evaluate the amount of bark in the image. The structure of the algorithm is shown in Appendix A. Each step will be explained further in the following sections.

5.1 Segmentation

5.1.1 Thresholding

Since the bark pieces are dark and most of the wood is bright the first step was to threshold the image. The thresholding was done in the HSV space, in the value image. The value image was chosen, because this was the only color space coordinate observed where the shadows and bast did not look exactly the same as the bark. The threshold value was after some experiments chosen to 0.75. With this value all the bark was accepted as objects which was good, but so was most of the shadows and the bast. The bright wood was not accepted as bark. The value 0.75 is sensitive. Changing it with ± 0.05 will give a considerable difference in the result. Figure 8 shows an example of an original image and the thresholded image. In the original image the bark pieces have been marked with a rectangle. The white pixels in the thresholded image are the objects found after thresholding. It is obvious from Figure 8 that thresholding is not enough for segmentation of the bark.

The pixel values was also plotted in the RGB space to examine the possibility of a better result using multi-spectral thresholding, but the bark, bast and shadows have the same color and are therefore not separable with multi-spectral thresholding.

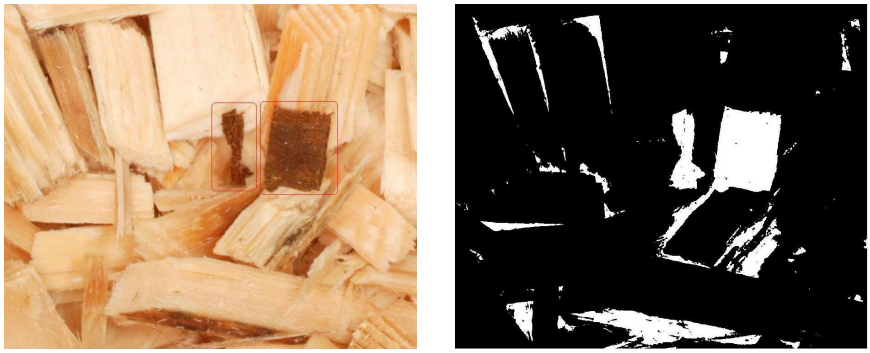


Figure 8: Original image with the bark marked with rectangles (left) and thresholded image (right).

5.1.2 Filtering

The problem with same colors for both bark, bast and shadows make thresholding not enough to separate between these. However, looking at the image filtered with an edge detecting filter indicates that the shadows are quite smooth, while the bark and bast are not. The idea of using the edge detection filter is therefore to get rid of some of the shadows. The filter used was a Laplace filter of size 5×5 pixels with the weights in figure 9. This filter have the properties

−1	−1	0	−1	−1
−1	0	2	0	−1
0	2	4	2	0
−1	0	2	0	−1
−1	−1	0	−1	−1

Figure 9: The 5×5 Laplace filter.

mentioned in section 4.6.1, that the filter is isotropic and that the sum of the weights is zero. The coefficients have opposite signs to the coefficients in the filter in Figure 5, meaning that the filter will find dots that have a high value compared to the surrounding pixels, instead of finding dots with lower values.

The filtering was done separately for the hue, saturation and value images. The values under zero or above 255 was set to zero and 255 respectively. After the filtering, the three resulting grey scale images were combined to form a color image. The result is shown in Figure 10 where the bright pixels represent edges. The brighter the pixel, the sharper the edge.

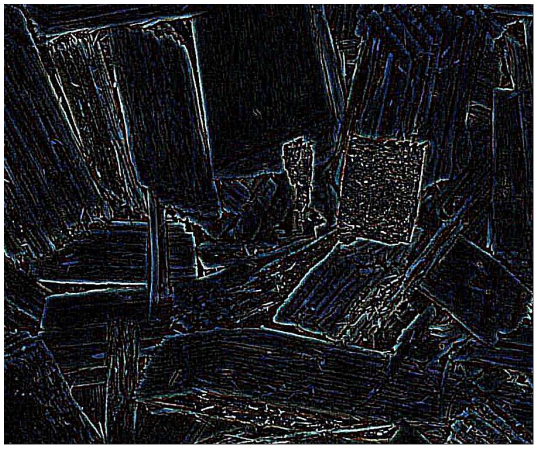


Figure 10: Filtered image. The filtering is done separately for the hue, saturation, and value images and then combined to form a color image. The original image is shown in figure 8.

A combination of thresholding and filtering gave a better result than thresholding alone and was done in the following way: A small part of the image, 15×15 pixels, was selected and analyzed both in the filtered image and in the thresholded image. At first it was analyzed in the filtered image. This was done by dividing the filtered pixels into four different categories. These were:

- White pixels (k): The pixels that have the value 255 in all three filtered images.
- Black pixels (n): The pixels that have the value 0 in all three filtered images.
- Low intensity edge pixels (p): The pixels that have a value under 55 in all three filtered images. The value 55 was found with the trial and error method and it is sensitive to changes in the order of ± 10.
- The rest of the pixels (m).

The criteria used for deciding if the analyzed area was bark or not was:

$$k > 0 \textbf{ or } (p < m < n)$$

The first criteria, $k > 0$, says that there should be white pixels in the area. This is perhaps intuitive since white pixels represents sharp edges. The condition $p < m$ can be thought of as a case where there should be fewer low intensity edge pixels than other edge pixels (still not totally white). The m pixels may still have low intensity values in some of the three images, but must at least be larger than 55 in one of the images. That means the edge is sharp in at least one image. The condition $m < n$ might seem a little strange. It simply says that there should be fewer pixels with some high intensity value in the filtered image, than there are black pixels. This condition is a result of extensive testing, as

are also the combination of the conditions. There were, in fact, much trial and error involved in this part of the segmentation.

If the conclusion after the previous analysis was that if the area had the right combination of edge pixels, it was assumed to be bark. In that case every pixel in this little part was then looked at in the thresholded image. If the thresholding result said a pixel belonged to a piece of bark, it was still assumed to be bark, but if it was set as background in the thresholding it was still thought of as background, even thought the filtering indicated that it could be bark. This procedure removed some of the shadows, but mainly it helped to find some boundaries for the bark pieces. A piece of bark that had a shadow beside it could in this way be segmented without the shadow. Without this combination of thresholding and filtering the shadow often was assumed to be a part of the bark piece. The difference seemed therefore not so big, but it is important for a good segmentation result. Figure 11 shows the thresholded image and the image resulting from combination of filtering and thresholding.

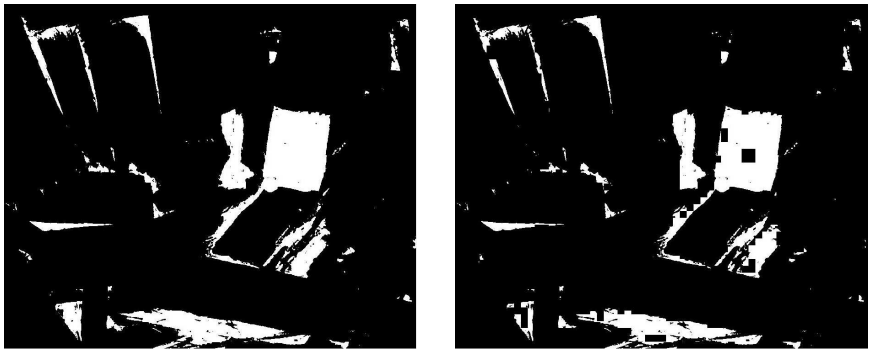
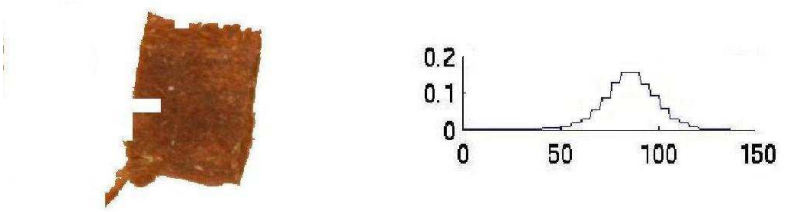


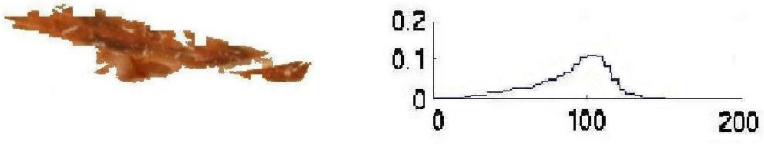
Figure 11: Left: Thresholded image. Right: Thresholding and filtering combined. The original image is shown in figure 8.

5.1.3 Histogram analysis

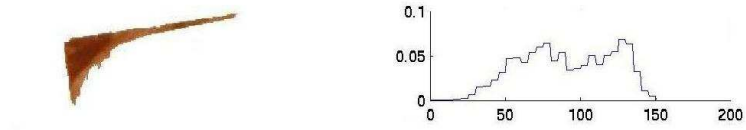
For each of the remaining connected components the histograms were calculated. After analyzing the histograms some differences were found between bark, bast and shadow. An example is shown in Figure 12. The bark components generally had a more symmetric histogram, even though there were many exceptions. The bast is often uneven in its colors and therefore the histogram sometimes have two or more peaks. For bark the maximum value in the histogram was often more centered than in bast or shadows.



(a) Bark



(b) Bast



(c) Shadow

Figure 12: An example of three different components and their histograms.

Using these observations made it possible to separate between bark, bast and shadows. The conditions used for separation of the objects were:

- The difference between the maximum value and the closest neighbors is smaller than 0.3 (see Figure 13(a)). This criterion tests if the maximum is much larger than the surrounding values.

- If the histogram has more than one maximum, a sum is made with all the values in the histogram that increase towards a maximum that is not the global maximum. The sum must be under 0.2 on each side of the global maximum. This condition is illustrated in Figure 13(b) and handles the situation with two or more peaks in the histogram.

- Two straight lines are drawn between the global maximum and the first zeros on each side of the maximum. If the histogram values are larger than the values on that line the difference between those values were summed together. The following conditions were used regarding the maximum difference between the line and the histogram, m, and the total sum of the differences, s:

  $$s < 0.01 \textbf{ or } m < 0.01 \textbf{ or}$$
  $$(s < 0.055 \textbf{ and } m > 0.01) \textbf{ or}$$
  $$(0.055 < s < 0.07 \textbf{ and } m > 0.015).$$

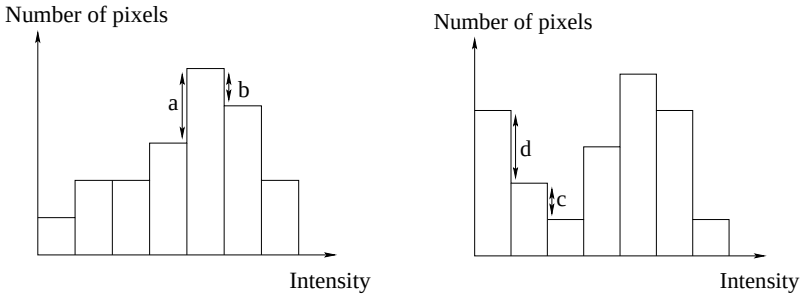
  This condition also takes care of the situation with local maxima as well as histograms like the one in Figure 13(c).

- The maximum value in the histogram must not be next to an intensity that is not represented by any pixels, that is the histogram value for that intensity is zero. This condition takes care of small components and asymmetric histograms.

All the conditions above were found through trial and error and they are sensitive to small changes. To save a connected component as a piece of bark, all the conditions above must be true for the histogram.

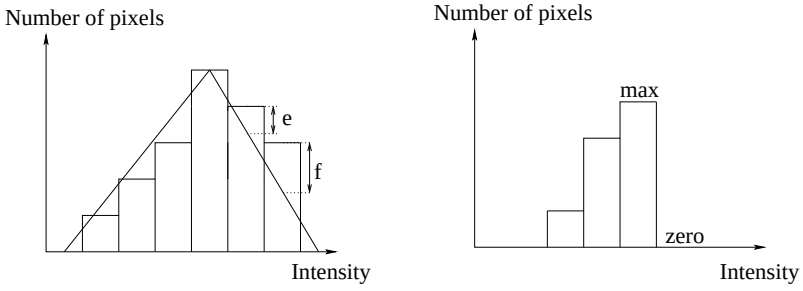
Analyzing the histogram was a very important procedure in segmentation of the bark. It removed the main part of shadows and bast without removing the bark. Figure 14 shows the result before analyzing the histograms and after.

When analyzing the histograms it is important not to have any shadows attached to the bark. Such a histogram will not have the form of a bark histogram, but will be disturbed by the contribution from the shadows. This is the reason why the combination of filtering and thresholding is important since it separates the shadows from the bark.



(a) $a < 0.3$ and $b < 0.3$

(b) $c + d < 0.2$



(c) Conditions on the sum of e and f and on the largest value, f. See text.

(d) The maximum cannot be next to a zero.

Figure 13: Conditions for analyzing the histograms.

Figure 14: Before (left) and after (right) removing the pieces by analyzing the histograms. The original image is shown in figure 8.

5.2 Area calculation

When the pieces of bark are found an assumption is made that the holes in the pieces are also bark. Therefore all the white pixels that are surrounded by bark are also set as bark. This will be true in most cases.

The next step is to calculate the total amount of bark in the image. The fact that the chips are in many layers and therefore creates a lot of shadows in the images is something to consider in the area calculations. The shadows cannot be assumed to be wood and therefor it is not ideal to compare the amount of bark with the total number of pixels in the image.

In the program the amount of bark is calculated in two different ways. One is comparing the amount of bark with the total number of pixels. The other area calculation is made by a comparison between the found bark and the pixels not found after filtering and thresholding. This is an approximation of the uppermost layer of chips in the image.

6 Experiments and Results

Two different sets of images were used in the experiments. The first set was used both for developing the method and for evaluation and the second set was used only for evaluation. The experiments were mainly in the form of trial and error. The methods used were not advanced and the main work was to make experiments to establish good values for thresholding, determining when to think of a piece like smooth or not smooth and a big effort was put down in the part where the histograms are analyzed and sorted out. Many different measurements have been tried and after some testing the ones used in the algorithm were picked out.

In figure 17 to 23 in Appendix B some images and the results are shown. In the original images the bark is marked with a rectangle.

6.1 Image series 1

Figure 17 to 20 shows some images and the results in the form of found bark. The result is good in these images and it is easy to see that the bark pieces are found and nothing else is assumed by the method to be bark, except for some small pieces of bast or shadows in figure 20.

6.2 Image series 2

Figure 21 to 23 shows some images and results from image series 2. These results is also fairly good, but in figure 22 some shadows and bast is also identified as bark, which is not good.

There are also some other problems in image series 2. Figure 15 shows a bluish piece of wood that has been identified as bark. These bluish pieces appear in old wood and it is not bark, only miss-colored wood.

Another problem is the bright piece of bark seen in figure 16. It is not identified as bark by the algorithm and that is of course wrong.

Figure 15: A bluish piece that is not bark (left), but identified as bark by the method (right).

Figure 16: Bark not found by the method.

7 Conclusions

The results from image series 1 are good and shows that it is possible to segment the bark from the rest of the pieces and the shadows.

In image series 2 the results are not as good. Not all bark pieces are found, and some other pieces are instead identified as bark. It should be noted, though, that it works well on pieces similar to the bark in the images in series 1. That means that the algorithm can identify those pieces of bark. The other pieces, not used for developing the algorithm, are not correctly identified by the algorithm. Also, the same thing occur with the bluish pieces. They were not present at the development and therefore the algorithm does not know that it is wood, not bark. However, the blue pieces are different in color from the bark and it should be easy to separate them by thresholding in the hue image.

The reason for the problems with bark not found and bluish pieces is that the image material for developing the algorithm was not large enough. Continuing to develop the method with more images from different sets of wood-chips would give better results.

The threshold value is sensitive to disturbances in color. Therefore, it is important that the images are taken under the same conditions every time, so that the colors are as similar as possible between the images. For this reason, calibration of the threshold is needed for every new set up of hardware.

The combination of filtering and thresholding was not perfect. In the result images the bark pieces are rugged. That depends on the way that the combination is done. Since an area of 15×15 pixels are examined at a time there can be squares left where the conclusion is that it is not bark, although that square is on the edge of a bark piece. In that case the result is a black square on the edge of a bark piece. Also there are still some small pieces of bast and shadows sticking to the bark pieces. Also this can be seen in the result images and shows the difficulty in separating between bark, bast and shadow.

This method is no exact way to establish the amount of bark in a truck with wood-chips, nor in an image. The 2-D format makes it difficult to make any assumptions on the thickness of the chips and therefore the volume cannot be analyzed, only the area. Since the volume cannot be measured the method also does not consider the densities of wood and bark. That makes the comparison with the existing method of weight measurements impossible. There is no reason why the different methods would show a similar result. Hopefully, there should be a correlation, but the percentage numbers have no reason to be the same.

To improve the algorithm more images with different kinds of bark and bast are needed. Doing this will probably give good results also for not yet analyzed pieces of bark.

8 Acknowledgments

I would like to thank KG Paulsson and Ulf Höglind who initiated and supported this project through Höglind Marketing HB, in cooperation with Stora Enso Packaging Boards, Skoghall Mill. Thanks also to PhD Mats Erikson and Prof. Gunilla Borgefors for supervising and feedback during my work, and Prof. Mats Nylinder for answering my questions about wood-chips. Finally I would like to thank the staff at CBA for helping me with various problems during my work

and making my time at CBA very pleasant.

References

[1] Sappi. http://www.ideaexchange.sappi.com/uploads/506_PaperMaking.pdf, Dec 2004.

[2] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, chapter 6.2.3. Prentice Hall, 2nd edition, 2002. www.prenhall.com/gonzalezwoods.

[3] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/HSV_color_space, Nov 2004.

[4] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, chapter 3.7.2. Prentice Hall, 2nd edition, 2002.

A Algorithm

Algorithm 1: The program

```
read the input image;
threshold the image;
filter the image;
for 15 * 15 pixels at a time do
    analyze the group of pixels in the filtered image;
    if it is not smooth then
        analyze pixel by pixel in the thresholded image;
        if the pixel has the value 1 then
            set the pixel as an object;
        else
            ignore the pixel;
        end
    else
        ignore the group of pixels;
    end
end
for every object identified do
    calculate the histogram for that object;
    if the histogram looks like bark then
        keep the object;
    else
        ignore the object;
    end
end
fill the holes in found bark pieces;
area calculations;
save the result as an image;
```

B Results

(a) Original image

(b) Result

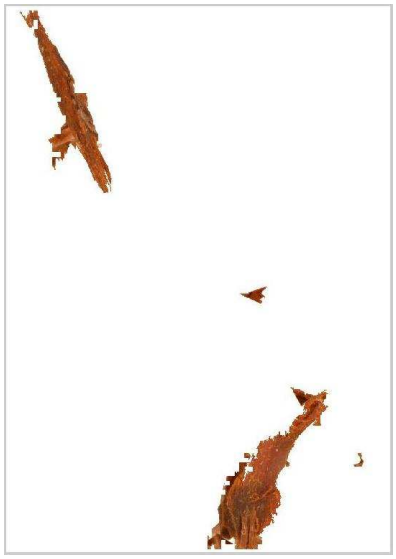
Figure 17: Image series 1

(a) Original image

(b) Result

Figure 18: Image series 1

(a) Original image



(b) Result

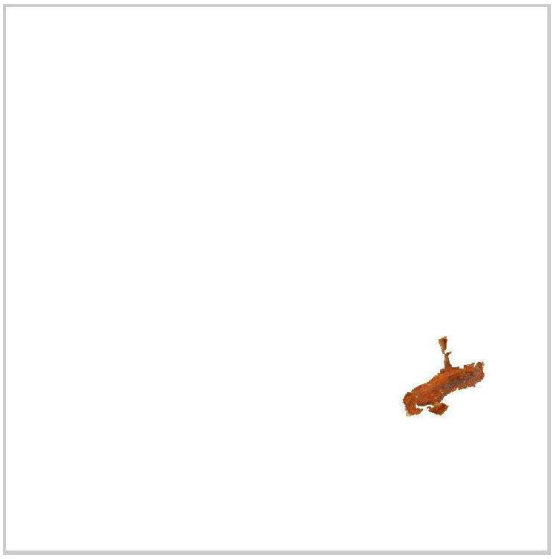
Figure 19: Image series 1

(a) Original image

(b) Result

Figure 20: Image series 1

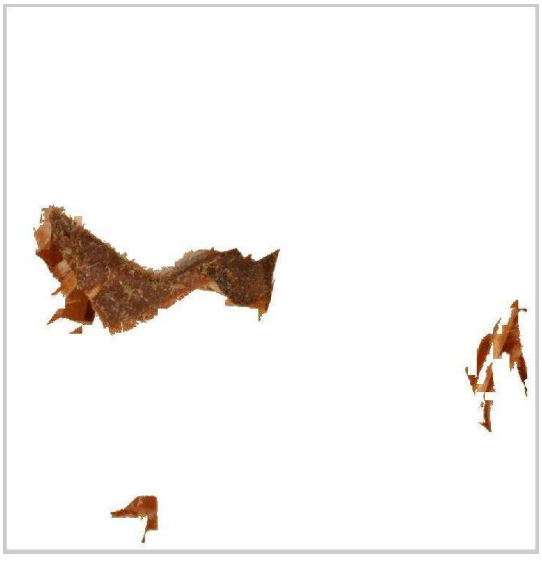
(a) Original image



(b) Result

Figure 21: Image series 2

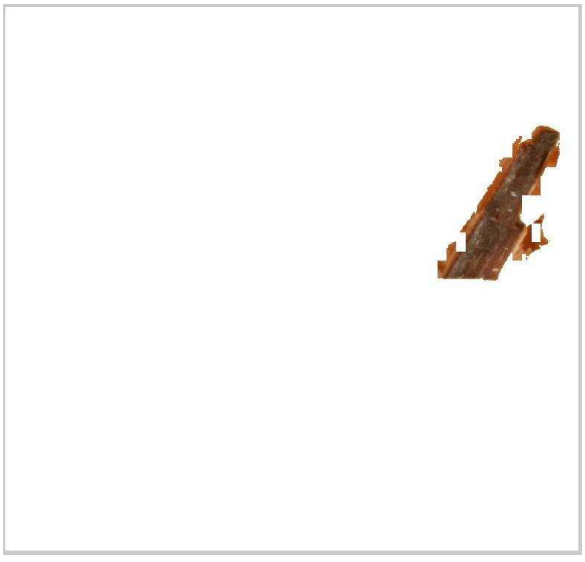
(a) Original image



(b) Result

Figure 22: Image series 2

(a) Original image



(b) Result

Figure 23: Image series 2