

Automatic Fault Detection in Cheese using Computer Vision

David Wrangborg

Centre for Mathematical Sciences
Lund Institute of Technology (LTH)
Lund University

Supervisors: Karl Åström
Håkan Ardö

11th June 2007

Abstract

In production of cheese with eyes (bubbles of CO_2 often referred to as holes) there are occasionally problems with cracks in the cheese. These cracks can pose a problem when cutting up the cheese and they will, even though they are harmless, cause the cheese to appear less attractive for the consumer. Therefore the cheese producing industry is interested in the microbiological reasons behind the cracks. As one step to find these statistics of when the cracks appear, what they look like and where on the cheese they are needs to be gathered.

This master's thesis examines the possibility to use digital images taken when cutting up the cheese and automated image analysis to reach these statistics. This is done by taking a photo of the cut surface on all cheeses passing during the cutting-up process. The resulting images are then segmented by classifying the pixels as cheese or background according to their colour value and Bayes' Theorem. An ellipse is fitted to the cheese pixels. Everything inside the ellipse is considered to be cut surface and everything outside background and is therefore removed. The image is rectified to a top view. This allows the image to be taken with a bit tilted view.

Several different filters, designed to be sensitive to cracks, are applied on the rectified images. The output of these filters are used as features for the classification. The classification algorithm Support Vector Machine is then trained on training data consisting of three classes, flat cheese, eyes and cracks. The resulting classification algorithm is used to classify all the pixels in the images into those three classes.

The third interesting class, cracks, contains information about if there are any cracks and in that case how big, and where they are. The information can be specially treated with morphological operations to give a decision if the specific cheese has a crack of a certain size.

The results, from tests on images taken at a live industrial environment, are promising but future development is needed for the method to be usable commercially.

Acknowledgements

I would like to thank my supervisors at the Centre for Mathematical Sciences at Lund Institute of Technology (LTH), professor Karl Åström and Ph.D. student Håkan Ardö for their time, guidance and valuable advice. I would also like to thank Frans Bengt Nilsson at Skånemejerier and professor Ylva Ardö at Department of Food Science, University of Copenhagen for providing needed expertise on cheese.

Contents

1	Introduction	1
1.1	Background	1
1.2	Cheese production	1
1.3	Aim of the Thesis	2
1.4	Related Work	2
1.5	Overview of this Thesis	2
2	Theory	3
2.1	Bayes' Theorem	3
2.2	Morphology	3
2.2.1	Dilation	3
2.2.2	Erosion	4
2.2.3	Opening and Closing	4
2.3	Homogeneous Coordinates	6
2.4	Homography	6
2.5	Cholesky Decomposition	6
2.6	Canny Edge Detector	6
2.7	Harris Corner Detection	7
2.8	Support Vector Machine	8
3	Methods	10
3.1	Image acquisition	10
3.2	Image Segmentation	11
3.2.1	Attaining a Binary Cheese Image	11
3.2.2	Applying Morphological Operations	13
3.2.3	Fitting an Ellipse	14
3.3	Rectifying the Image	14
3.3.1	Finding a Homography	15
3.3.2	Rectification	17
3.4	Crack Detection	18
3.4.1	Cracks	18
3.4.2	Feature Selection	19
3.4.3	Using the SVM	22
3.5	Whole Cheese Classification	23
4	Results	24
4.1	Segmentation and Rectification	24
4.2	Crack Detection	25
4.3	Whole Cheese Classification	26
5	Conclusions and Future Work	28

1 Introduction

1.1 Background

When producing the typical Swedish cheese Grevé a certain degree of large round eyes (bubbles of CO_2 in the cheese often called holes in daily speech) are expected and desirable. The eyes originates from production of CO_2 in the fermentation process.

Unfortunately sometimes the CO_2 is produced later than usual and then arise when the cheese has already attained a more solid form. The result is that cracks appear. These cracks can cause trouble later in the process and will, even though they are harmless, cause the cheese to appear less attractive for the consumer. Therefore it's desirable to minimize the cracks while still attaining the typical eyes.

There has been and still are studies trying to figure out the microbiological reasons for this late CO_2 production. As of right now a few of the big cheese producing companies in Sweden have joined together in a research program to find the causes. As one step to reach this goal statistics of when the cracks occur, how they look and where on the cheese they are needs to be gathered. This thesis investigates the possibility to use a digital camera and image analysis to attain these statistics from the large quantities of cheese in the daily production. If this can be done it will be a cheap method with no interference in the production.

1.2 Cheese production

Grevé is a semi hard Swedish cheese with characteristically large round eyes/holes. It has its origin about 200 to 300 years ago when Swiss cheese makers were invited to countries all over Europe to teach the Swiss way of making cheese. Since the conditions in the new countries were different from the Swiss the cheese produced also became different from the typical Swiss. One of the resulting cheeses in Sweden was the Grevé.

Like most other cheeses Grevé's main ingredient is pasteurized cow milk. To coagulate the milk calf rennet is added. After the coagulation the substance called curd is cut, stirred and scalded in order to reach the appropriate level of moisture. The curd is then molded into pieces. These pieces will contain microscopic bubbles of air which will be the seeds for the eyes. The pieces are pressed for 1 to 2 h to press out the last free moisture and shape it into a desirable form (in this case a cylinder with diameter of approximately 350mm and thickness of 120mm). The cheese is then put in brine salt for a couple of days. Thereafter the cheese is coated with paraffin wax and put to storage. During storage the active bacteria¹ will convert citrate into CO_2 , diacetyl, acetoin and butanediol. The excess CO_2 will eventually begin to expand the microscopic air bubbles and form the eyes of the cheese. It's when this production of CO_2 is too late that the problem with cracks arises.

There are other faults but cracks that can arise during the manufacturing process. The most common is that the eyes are so close to each other that they overlap to a large degree. In this way the eyes are not complete. Then the form

¹In Grevé the active bacteria are: DL-starter culture: A mixture of several stains of lactic acid bacteria, which is in the milk from the start. PAB: Propionic Acis Bacteria, which is added during the process

of the eyes are torn and they are not completely round anymore. Another fault is the appearance of clusters of small holes instead of the typical large round holes.

After the cheese has been stored for an appropriate amount of time the cheese is cut and packed in smaller pieces adapted for sale.

For more background on cheese manufacturing see [1][ch. 12, 15, 17].

1.3 Aim of the Thesis

The aim of this thesis is to evaluate the possibility of detect cracks in cheese by studying images taken during the cutting process using a digital camera and computer vision.

1.4 Related Work

Image analysis and computer vision has been used with success in inspection of food products for some time. Its main use for food is quality assurance purposes were it has been applied increasingly frequent. Areas where computer vision has been in use for inspection include: meat, fish, vegetables, bakery products and many more. For a review see [3].

Computer vision has also been used for inspection of cheese before. For example the authors of [4] developed a computer vision method to determine the meltability of Cheddar and Mozzarella cheese. The same authors also evaluated the possibility to determine the oiling off property of cheese in [5].

A more similar work to this is done in [6] where a method to determine the surface area of a cheese slice occupied by gas holes was developed. The experimental setup in that project was quite different from the one here and a lot more ideal. With the images taken of cheese slices that were pre prepared the analysis could be done with quite elementary image analysis such as thresholding. The conclusion was that this works well.

1.5 Overview of this Thesis

In chapter 2 a couple of theoretical concepts are presented. These concepts are then needed in the following chapter 3 where the methods used in the project are described and explained. The methods are presented in a chronological order i.e. presented in the order they are applied to the image. In chapter 4 the results are discussed. A few problems that arose and were solved during the projects are also presented. In chapter 5 conclusions are drawn and future work discussed.

All the programming in this thesis was done in *MATLAB* which is a numerical computing environment created by *The MathWorks* (www.mathworks.com).

2 Theory

In this section some basic tools that are needed to understand the methods in the next chapter are presented.

2.1 Bayes' Theorem

One of the most used theorems in statistics was first formulated by an English priest called Bayes in the 18th century.

The theorem treats conditional probabilities. Especially it states the relationship between the probability of an event A conditional on another event H and the reverse, i.e. probability of H conditional on A.

The theorem states that:

If A and H_j are events, where $H_1 \cup H_2 \cup \dots \cup H_n = \Omega$ and $H_j \cap H_i = \emptyset$ for $i \neq j$ then

$$P(H_i|A) = \frac{P(H_i)P(A|H_i)}{\sum P(H_j)P(A|H_j)}. \quad (1)$$

So as long as the a priori probabilities $P(H_j)$ and all the conditional probabilities $P(A|H_j)$ are known the probability $P(H_i|A)$ can be calculated, [2][p. 31].

2.2 Morphology

Mathematical morphology provides important tools for extracting image components that are useful in representing and describing region shapes. The operations are particularly useful for binary images and common use include noise removal, image segmentation and image enhancement.

Here follows a short review of the basic concepts. For a thorough treatment see [7].

The language of morphology is set theory and most operations take two sets as input data. One, quite naturally, is the input image (usually binary), here denoted A, and the other is a structuring element, denoted B. The structuring element determines the precise effect of the morphological operation on the image. Typically the structuring element is translated to and superimposed on each pixel in the input image in turn. The value of corresponding pixel in the output image is determined by a comparison between the image and the structuring element. This comparison depends on the morphological operation in use.

The most basic operations are dilation and erosion. From these the more complex operations, opening and closing can be built.

2.2.1 Dilation

Dilation is the first of the two basic operations in mathematical morphology. The effect of dilation is to enlarge foreground regions i.e. expand the boundaries of and shrink holes within these regions.

The mathematical symbol for dilation is \oplus and its set theoretical definition (for binary sets)

$$A \oplus B = \{x | (\hat{B})_x \cap A \neq \emptyset\}, \quad (2)$$

where A denotes the input set, B the structural element, \hat{B} denotes a reflection of B , $(B)_x$ the translation of B by x and \emptyset is the empty set.

This means that each pixel in the output image is determined by superimposing (the reflection of) the structuring element on the corresponding pixel in the input image. If at least one pixel in the structuring element coincides with a foreground pixel in the input image the output pixel is set to be foreground. If all the corresponding pixels in the image are background the output pixel is set to background. For an example see Figure 1(d).

2.2.2 Erosion

The second basic operation of morphology is erosion and to some extent it is the opposite of dilation. As the name suggest, instead of enlarging the foreground regions like dilation, erosion erodes the boundaries of the region making it smaller.

The mathematical symbol for erosion is \ominus and it's set theoretical definition (for binary sets)

$$A \ominus B = \{x | (B)_x \subseteq A\}. \quad (3)$$

This means that, just as in dilation, each pixel in the output image is determined by superimposing the structuring element on the corresponding pixel in the input image. The difference is that for the output pixel to be considered foreground all the pixels in the input image which lies within the region of the structuring element must be foreground. If one or more of these pixels are background (in the input image) the output pixel will be labelled background as well. For an example see Figure 1(b).

2.2.3 Opening and Closing

So far we have two morphological operations one that enlarge foreground regions (dilation) and one that shrinks them (erosion). Now to get two new useful operations we combine these two.

The first, called opening, consist of applying an erosion followed by a dilation using the same structural element i.e.

$$A \circ B = (A \ominus B) \oplus B, \quad (4)$$

where \circ denotes opening.

The second, called closing is quite the opposite. It consists of a dilation followed by an erosion (still using the same structural element) i.e.

$$A \cdot B = (A \oplus B) \ominus B, \quad (5)$$

where \cdot denotes closing.

Both of these have a smoothing effect on contours of an image but otherwise their characteristics are quite different. Opening generally breaks narrow passages and eliminates small regions while closing generally fuses narrow breaks, eliminates small holes and fills gaps in the contour. See Figure 1(e)&(i).

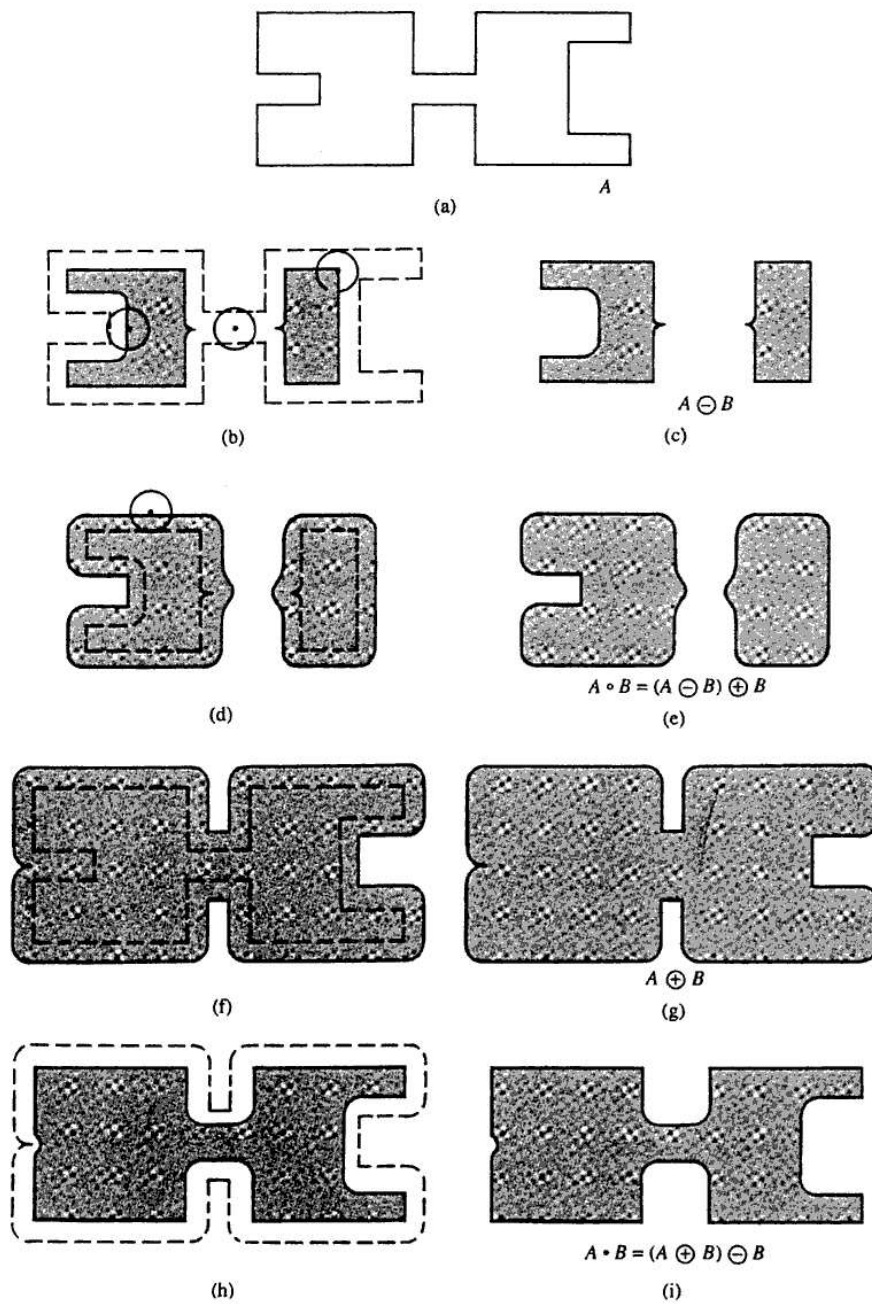


Figure 1: Illustration of morphological operations. (a) shows a set (b) shows a disk as the structuring element at different locations during erosion the result of the erosion can be seen in (c). In (d) dilation is applied to the eroded set, the result is shown in (e) which is an opening. (f) and (g) shows the result of dilation applied to the original set and in (h) and (i) erosion is applied to the result in (g) giving a closing. Source: [7]

2.3 Homogeneous Coordinates

In a two dimensional Euclidean coordinate system a point in the plane is represented by a pair of scalars $(x, y)^T$. One representing the position in the first dimension and the other the position in the second dimension.

A line in the plane is determined by an equation of the form $ax + by + c = 0$. Different values of a, b and c give rise to different lines. A line can therefore be represented by a vector $l = (a, b, c)^T$. This representation is not unique. Since the equation $(ka)x + (kb)y + (kc) = 0$ describe the same line it can also be represented by the vector $l = k(a, b, c)^T$ for an arbitrary choice of scalar, k.

A point $(x, y)^T$ lies on a line $l = (a, b, c)^T$ if and only if $ax + by + c = 0$. This can be written as $(x, y, 1)(a, b, c)^T = 0$. Here the point is represented by its Euclidean coordinates with an added 1 at the end. Since $k(a, b, c)^T$ describe the same line the equation can also be written $(kx, ky, k)(a, b, c)^T = 0$. It's therefore natural to let $(kx, ky, k)^T$ represent the same point $(x, y)^T$. This is the homogeneous representation of a point in 2-dimensional space. An homogeneous vector has the form $\mathbf{x}=(x_1, x_2, x_3)^T$ which represent the point $(x_1/x_3, x_2/x_3)$ in the 2-dimensional space.

One of the biggest gains with homogeneous coordinates is that with them, not only finite points can be represented, but also points at infinity. These points are characterized by that their third component is zero i.e. $(x, y, 0)^T$. This vector then represent a point at infinity in the direction of (x, y) .

One important application of points at infinity is in intersections of lines. In Euclidean geometry the intersection of parallel lines can not be represented since it is located at infinity. In homogeneous coordinates the intersection of all lines even parallel can be represented, [8].

2.4 Homography

A homography, H, in computer vision relates points in one view with points in another view by $p_{\text{new}} = Hp_{\text{old}}$. If the views are represented by images, the p's are the homogeneous vectors for the pixel coordinates in the old and new images respectively and H is a 3x3 matrix which relates these, [8], [11].

2.5 Cholesky Decomposition

A symmetric positive definite matrix, A can be decomposed as

$$A = U^T U, \tag{6}$$

where U is an upper triangular matrix. This is called the Cholesky Decomposition [9].

2.6 Canny Edge Detector

Canny edge detection was developed by J. F. Canny in 1986 and is one of many edge detectors. Canny's goal was to develop optimized edge detectors. To do so multiple stages are used in his detector.

The image is first convolved with a Gaussian mask to reduce noise. Then four different filters for detecting vertical, horizontal and diagonal edges are applied

and stored. For each pixel the largest value of these four is marked along with which direction it belongs to, an intensity gradient has been produced.

Assuming the edges to be continuous lines it is reasonable to use two different thresholds to find these, one higher and one lower. The higher is applied to find "seeds" or starting points for the lines. Once these seeds have been found the directional information stored from above can be used to follow the edge. Here the lower threshold is used. As long as the point in the direction of the previous point is higher then the lower threshold it's marked as edge and its direction is follow to the next point. When all the seeds have been followed to their end the edge detection is done.

There are three adjustable parameters in the Canny filter, the size of the Gaussian filter and the two thresholds.

2.7 Harris Corner Detection

Edge detectors most often fail at corners, because they assume the edges to be continuous lines. Instead to detect corners specially designed tools are used. One such tool is Harris corner detector.

Let $I(x, y)$ represent the grey-scale intensity at pixel coordinate (x, y) . Then consider the following matrix (where $\frac{\partial I}{\partial x}$ is the partial derivate in x direction) defined in a region surrounding a certain pixel

$$M = \begin{pmatrix} (\frac{\partial I}{\partial x})^2 & \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial y} \frac{\partial I}{\partial x} & (\frac{\partial I}{\partial y})^2 \end{pmatrix}. \quad (7)$$

The matrix gives a measure of how the intensity varies around the pixel. The eigenvalues of the matrix will behave differently depending on the region.

There are three major possibilities:

- Two small eigenvalues in areas with roughly constant intensity.
- One small and one big eigenvalues in areas where the intensity varies in one direction i.e. edges.
- Two big eigenvalues in areas where the intensity varies in both directions i.e. corners. [8]

Looking for corners can therefore be done by looking for a matrix with two big eigenvalues. One way to look for such a matrix without computing the eigenvalues is to compute (where k is a constant suggested by Harry to be 0.04)

$$R = \text{Det}(M) - k\text{Tr}(M)^2, \quad (8)$$

which is related to the eigenvalues α and β by

$$R = \alpha\beta - k(\alpha + \beta)^2. \quad (9)$$

R will be small if either of α or β is small and large if both α and β are large.

2.8 Support Vector Machine

Any classification task consists of two data sets:

- Training set: used to train the system.
- Testing set: used for testing the system.

Each instance in the training set have a number of *Features* which describes its properties and a *Class Label* stating what class the instance belongs to. The testing set only has the *Features* and it's the classification algorithm task to decide the most probable class for each instance in it.

There are many different algorithms to solve classification problems. One powerful, linear method is the Support Vector Machine. It is based on the concept of decision planes that determine the decision boundaries. The SVM simultaneously minimize the empirical classification error and maximize the geometric margin and is therefore also known as a maximum margin classifier.

The easiest way to understand the SVM is to start with a feature space with low dimensionality (2 or 3) and two classes which are separable. In the 2-dimensional case the SVM finds the line that separates the two classes with the largest margin. Then classification can easily be done by checking on what side of the line an instance lies. See Figure 2 for an illustration.

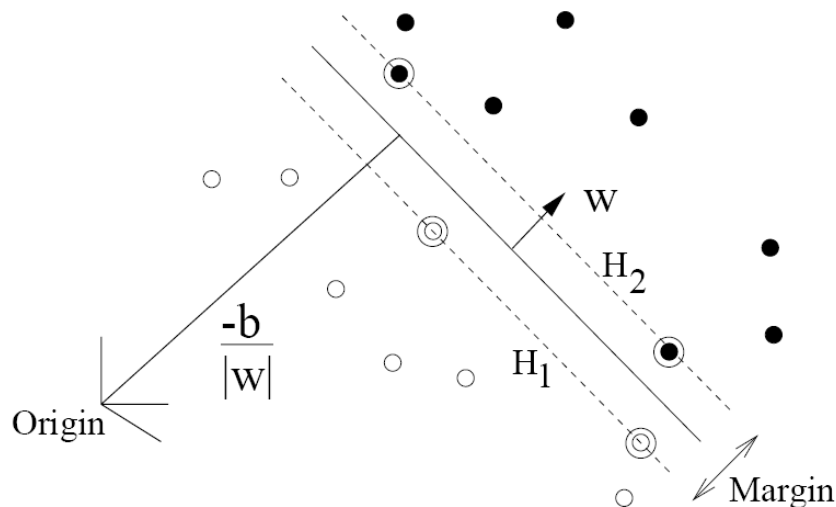


Figure 2: Decision boundary of a SVM in a 2 class problem (filled dots against none-filled dots). Source: [10]

In the 3-dimensional case the boundary is a plane and the classification can be done by checking on what side of the plane an instance lie.

In higher dimensions the same thing is done the difference is the dimension of the plane which is then higher than two. Then the plane is then referred to as a *hyperplane*.

The SVM can also handle more than 2 classes and cases with not completely separable classes. Confer [10] or [11] for a more detailed explanation of SVM.

3 Methods

3.1 Image acquisition

When the cheese is cut the first cut is made along the middle of the thickness perpendicular to the circular dimension. A circular cut surface is the result and this surface is exposed to the eye (or a camera) for a few seconds.

An Axis 207 digital camera (www.axis.se) was mounted so that an image of the cut surface could be taken. The camera could not be mounted directly above the cheese cut (which would give the best view) instead it had to be installed with a slightly slanting view. This result in that in the image the cheese cut appears elliptic instead of circular. The closer parts of the cheese cut appears bigger than the ones farther away. This is to be corrected later by a rectification.

Every night the process line is cleaned with pressure washer. Since Axis 207 isn't a water resistant camera it has to be removed during this. Therefore the camera needed to be put up and taken down once a day. This results in that the image of the cheese is taken with a slightly different view from day to day.

To get an image of the cut from each cheese a mechanical trigger is used. The trigger uses moving parts of the cutting machine to shortcut the trigger channel of the camera which then stores an image on a server.

The server used in this project was a Bubba-server from excito (www.excito.com) which is a very small Linux server.

In this manner an image (resolution 640x480 pixels) of the cut surface for each cheese was stored for later image analysis processing.

In Figure 3 the process line with the mounted camera can be seen and in Figure 4 an example of an image taken with it.



Figure 3: The process line with the digital camera

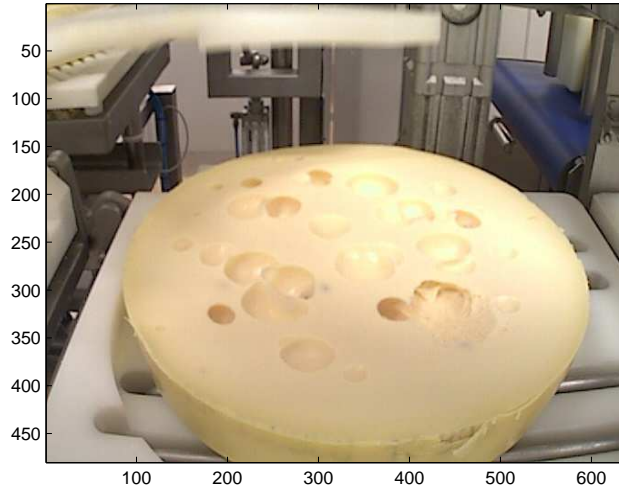


Figure 4: A typical image taken of the cut using the digital camera and the trigger.

3.2 Image Segmentation

When using image analysis techniques it's important to be able to distinguish the parts of the image containing interesting information from those that don't. For that reason it's important to be able to segment the image into two or more parts. One, or a couple, containing information and others that don't.

In this case the interesting part is the cheese cut. It's only in this parts the cracks appear and therefore only here that they can be detected. Everything outside this area contains no valuable information.

The first problem to be solved is therefore to segment the image into two parts. One consisting of the cheese cut (which contain the information for later analysis) and the other containing everything else (background).

3.2.1 Attaining a Binary Cheese Image

The main algorithm to find the cheese cut in the image is based of Bayes' theorem (see ch. 2.1). The idea is to use the theorem on the colour content in each pixel in the image.

The colours of the cheese pixels are hopefully distinctively different from the background pixels (more yellow). If this is the case it will be possible to decide if a pixel is cheese or not according to its colour.

The first step in reaching such a classification method is to find out what colour a typical cheese respectively none-cheese pixel has. This is done by manually selecting a large amount of pixels of both classes and storing the information of the colour content of these pixels.

The digital images (stored as .jpg) when loaded into Matlab have three colour channels according to respectively the RGB-scale (red, green, blue). Each of

these channels has a value between 0 and 255 representing how much of that colour the pixel contains.

To reduce the amount of information each colour channel is converted (linearly) to 16 values. Then the three dimensional RGB space is converted to a one dimensional by representing all the colour channels together by one number. If a pixel have colour values R_i , G_i and B_i the representing number is

$$X_i = \lfloor R_i/16 \rfloor 16^2 + \lfloor G_i/16 \rfloor 16 + \lfloor B_i/16 \rfloor. \quad (10)$$

(where $\lfloor x \rfloor$ is the floor function that gives the largest integer less than or equal to x). By this combination each unique combination of RGB is represented by a unique number X .

The information about the colour value of a typical cheese respectively background pixel can now be stored by making histograms over the manually selected pixels. The result is one histogram showing how the cheese pixels have been divided over the different colour values and one histogram for the background pixels (both stored as vectors in Matlab). Normalizing these histograms (i.e. dividing by the number of pixels used making the histogram) the result can be used as approximates of the conditional probabilities to get colour value X given that the pixel is cheese or background i.e. $P(X|\text{cheese})$ and $P(X|\text{background})$.

The a priori probabilities $P(\text{cheese})$ and $P(\text{background})$ are also needed. They are set to be equal, which means they are both 0.5. This makes the algorithm general. It could be made more specialized by looking at the occurrence of cheese pixels in a few images and then setting the a priori probabilities according to that.

The probability that a certain pixel is a cheese pixel can be calculated using Bayes' Theorem and the colour value X

$$P(\text{cheese}|X) = \frac{P(\text{cheese})P(X|\text{cheese})}{P(\text{cheese})P(X|\text{cheese}) + P(\text{background})P(X|\text{background})}. \quad (11)$$

If the probability that a pixel is a cheese pixel is greater than 0.5 the pixel is classified as cheese otherwise it's classified as background.

This classification is done for all the pixels in the image. The result is a binary cheese image. In this image 1 (white) represent cheese and 0 (black) represent background. For the algorithm to be useful most of the pixels within the cheese cut must be classified as cheese and most of the pixels outside the cheese cut must be classified as background. This is also the case. See Figure 5 for an example of the end product of the pixel classification.

The classification will not be perfect. There are a couple of reasons for this. One of the reasons is that since the image is taken in a live process line handling lots of cheeses it's very likely that the background will not be totally free from cheese. Cheeses passing before on the line might have left small bits and pieces of cheese. These bits and pieces will likely be classified as cheese since they show the characteristic colour of cheese.

Another reason for the classification to be imperfect lies within the algorithm itself. Some colour values might be so infrequent in the manual selection that there just aren't enough of them for good statistics to be reached (approximate conditional probabilities will be bad). The result might be that pixels of that colour are wrongly classified, either cheese pixel classified as background or background as cheese.

There are probably also colour values that appear in both the background and the cheese cut i.e. the colour of cheese pixels is not distinctively different from the background pixels. This will also result in wrongly classified pixels (the risk for this is greater because of the conversion to 16 value colour channels).

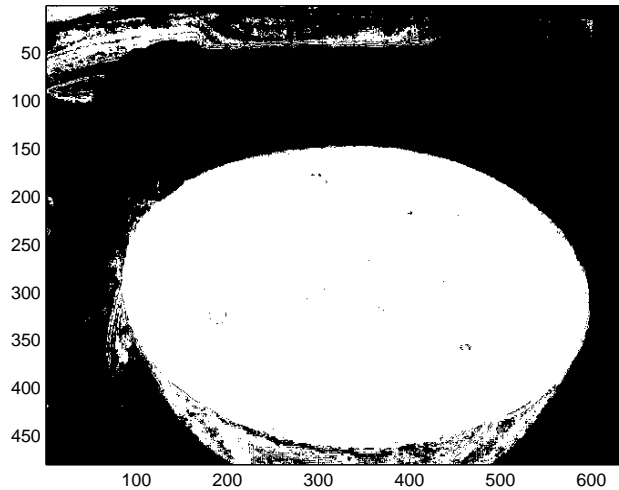


Figure 5: The end product of the classification applied to the image in Figure 4. White - Cheese, Black - Background.

3.2.2 Applying Morphological Operations

The binary cheese image which is the end product of the algorithm described in the last chapter will contain noise, since it has been classified on pixel level.

Most of the pixels classified as cheese will lie on the cheese cut itself although quite a few pixels outside of the cheese cut will also be wrongly classified as cheese. To get rid of these unwanted cheese pixels morphological functions are applied.

As structural element, a circle with a radius of 5 pixels, was selected. The choice fell on a circle since it's an easy form to describe which in a wide sense looks like the form which is sought (elliptic). To get rid of single pixels and smaller areas erode is first applied. This removes small areas and cuts other ones off from the main body. Then areas containing less pixels than a set threshold (1500 pixels) are removed (8-connectivity is used, see [7][p.41]). This hopefully only leaves the main body (although there might still be some other areas left if there is a lot of noise in the environment). Now dilate is applied to restore the main area to its original size. Finally to end the morphological operations a close operation is applied to get rid of spots in the main area.

3.2.3 Fitting an Ellipse

The cheese cut has in reality a circular form (or at least almost circular). Since the image of the cut is taken with a slanting view the cut in the image will appear in an elliptic form. It's inside this ellipse that all the useful information lies. For this reason it's reasonable to try to find an ellipse in the image that encloses the cheese cut.

For all the remaining regions the centre of mass, the major-, minor-axis and the orientation of an ellipse with the same second moment as the region is computed using the matlab command *regionprops*. The largest of these regions is assumed to be the cheese cut.

From the above quantities for the ellipse a matrix, E , that describes the ellipse can be computed. Points, p , which lies within the ellipse then fulfil the equation,

$$p^T E p \leq 0. \quad (12)$$

This concludes the segmentation. Everything inside the ellipse is considered to be a part of the cheese cut and everything outside it is considered background.

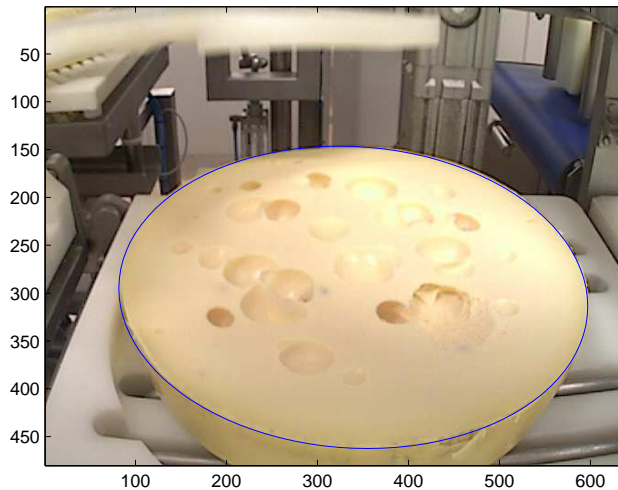


Figure 6: The original image (Figure 4) with the ellipse from the segmentation plotted (black line).

3.3 Rectifying the Image

For all areas to appear in their correct relative size in the image we need a view from right above the cheese. Since the camera could not be mounted right above the cheese this could not be achieved in the original image. Instead we need to correct this using math.

We are looking for a homography, H , that relates a (homogeneous) pixel coor-

dinate in the old image, p_{old} , to a pixel coordinate in the new image p_{new}

$$p_{new} = Hp_{old}. \quad (13)$$

When this homography has been found the image can be rectified by using this relation backwards. That is the origin of the new pixel can be found in the old by

$$p_{old} = H^{-1}p_{new}. \quad (14)$$

3.3.1 Finding a Homography

To do this we need some requirements that should be fulfilled in the rectified image.

One requirement can be created through manually marking pairs of lines that should appear parallel in the image if it is taken from a top view (this also means they should be parallel in reality). Fortunately such lines are easy to find in the images (as edges of machine parts). Such lines should never intersect i.e. should have their intersection at infinity.

A pair of lines that goes horizontally in the image and a pair that runs in the depth of the image were selected. In reality the horizontal lines are orthogonal to the depth lines. The intersections of the pairs were computed by taking the cross-product between the line parameters.

For the lines to appear parallel in the rectified image we want their intersection to be located at infinity. That is, if P_h and P_d is the intersection of the horizontal respectively depth lines, we want to find the homography, H_{lines} , such that: $H_{lines}P_h = (1, 0, 0)^T$ and $H_{lines}P_d = (0, 1, 0)^T$. The points $(1, 0, 0)^T$ and $(0, 1, 0)^T$ are chosen because they are elementary points at infinity representing two orthogonal directions, which we choose to be parallel to the x-axis and y-axis respectively. Together with for example the condition that the centre of the old image $(320, 240)$ to be located at $(0, 0)$ in the new i.e. $H_{lines}(320, 240, 1)^T = (0, 0, 1)^T$ all this can be written as

$$H_{lines} \begin{pmatrix} & 320 & \\ P_h & P_d & 240 \\ & & 1 \end{pmatrix} = I \Rightarrow H_{lines} = \begin{pmatrix} & 320 & \\ P_h & P_d & 240 \\ & & 1 \end{pmatrix}^{-1}, \quad (15)$$

where I is the identity matrix. Now a homography has been found that makes real parallel lines parallel in the image.

There is another requirement that can be put on the rectified image. As said before the cheese cut is circular (or at least almost circular). This means that in a top view, as in the rectified image, the cut should be circular.

From equation 12 and 14 it can be derived that the ellipse matrix in the new coordinates is

$$E_{new} = H_{lines}^{-T} E H_{lines}^{-1}, \quad (16)$$

this since

$$\begin{aligned} p_{old}^T E p_{old} &= (H^{-1}p_{new})^T E H^{-1}p_{new} = \\ &= p_{new}^T H^{-T} E H^{-1}p_{new} = p_{new}^T E_{new} p_{new}. \end{aligned} \quad (17)$$

The easiest way to check and make sure that the circular requirement is fulfilled is to rewrite the ellipse matrix in the unit circle form

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

The first step in doing so is to translate the centre of the circle/ellipse to the origin. To find the centre of the ellipse the inverse of the matrix, E_{new} is computed. It turns out that the centre can then be found in the last column of the resulting matrix, say (a,b,c). This is the centre in homogeneous coordinates. To convert it to Euclidean a division with the last element is performed resulting in (a/c,b/c). Now a translation matrix T can be created by

$$T_{\text{trans}} = \begin{pmatrix} 1 & 0 & a/c \\ 0 & 1 & b/c \\ 0 & 0 & 1 \end{pmatrix}.$$

The ellipse matrix with the centre translated to origo can be computed as

$$E_{\text{trans}} = T_{\text{trans}}^{-T} E_{\text{new}} T_{\text{trans}}^{-1}. \quad (18)$$

This new ellipse matrix has the form

$$E_{\text{trans}} = \begin{pmatrix} e_{11} & e_{12} & 0 \\ e_{21} & e_{22} & 0 \\ 0 & 0 & e_{33} \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & e_{33} \end{pmatrix}.$$

To transform this to a unit circle matrix the top left corner is Cholesky decomposed (see ch. 2.5) as

$$A = U^T U, \text{ with } U = \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}. \quad (19)$$

The final part of the homography is made by using this factorization and scaling the bottom left element to attain -1 in the final matrix. This is done by using

$$T_{\text{chol}} = \begin{pmatrix} U & 0 \\ 0 & \sqrt{-e_{33}} \end{pmatrix}$$

since

$$\begin{aligned} T_{\text{chol}}^{-T} E_{\text{trans}} T_{\text{chol}}^{-1} &= \\ &= \begin{pmatrix} U^{-T} & 0 \\ 0 & 1/\sqrt{-e_{33}} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & e_{33} \end{pmatrix} \begin{pmatrix} U^{-1} & 0 \\ 0 & 1/\sqrt{-e_{33}} \end{pmatrix} = \\ &= \begin{pmatrix} U^{-T} U^T U U^{-1} & 0 \\ 0 & e_{33}/(\sqrt{-e_{33}})^2 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & -1 \end{pmatrix}. \end{aligned} \quad (20)$$

So now forming $T_{\text{chol}}^{-T} E_{\text{trans}} T_{\text{chol}}^{-1}$ will result in the unit circle matrix. The total homography is therefore

$$H_{\text{total}} = T_{\text{chol}} T_{\text{trans}} H_{\text{lines}}. \quad (21)$$

Although to get the new image in a reasonable size and shape we translate so that the centre of the cheese is at (320,240) and scale it so that the cheese radius becomes 200 using transformation matrix,

$$S = \begin{pmatrix} 200 & 0 & 320 \\ 0 & 200 & 240 \\ 0 & 0 & 1 \end{pmatrix}.$$

The final homography is therefore $H = S H_{\text{total}} = S T_{\text{chol}} T_{\text{trans}} H_{\text{lines}}$.

3.3.2 Rectification

Once the homography, H , has been found the real rectification can be done. First the size of the new image has to be decided. Here it is natural to choose the same size as the original image (640x480 pixels).

The origin in the old image of a pixel in the new one can be computed by using the relation $p_{\text{old}} = H^{-1} p_{\text{new}}$. If this origin lies within the bounds of the old image (the first component between 1 and 640 and the second between 1 and 480) the corresponding pixel in the old image is the one closest to the origin (i.e. if the origin is rounded the coordinates of the corresponding pixel is attained). The colour value of the pixel in the new image is set to be the same as the corresponding old one.

If the origin lays outside the old image the colour value of the new pixel is set to 0 (for all three channels) which means black.

When this has been done for all the pixels in the new image a rectified image has been created. An example of a rectified image can be seen in Figure 7.

Everything in the rectified image is not interesting. It's only the part inside the segmented cut that is interesting. Therefore this part of the work is ended by removing everything outside the circle by using the requirement of equation 12. Then to save space the image is cropped to a smaller size that fits the radius of the cut. An example can be seen in Figure 8

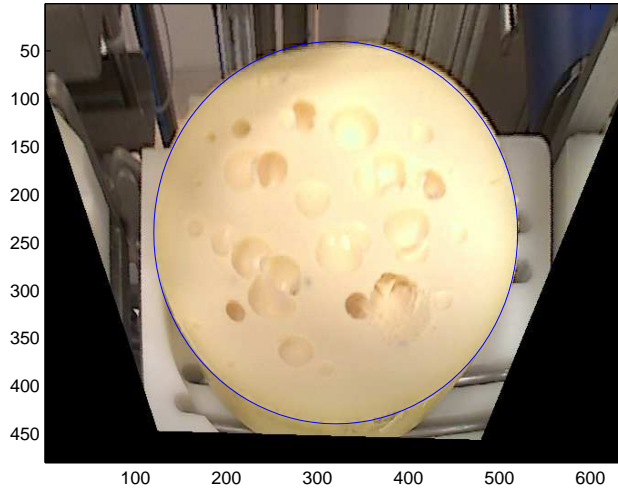


Figure 7: The rectified image with the transformed segmentation plotted.

3.4 Crack Detection

When the image has been segmented and rectified the actual work of detecting the cracks can begin.

To do this there has to be something that distinguishes regions with cracks from the rest of the cheese cut. To find such a distinction it needs to be established what a typical crack looks like and what about this look that makes it special from the rest of the cheese. When this has been done a feature space can be created. This feature space should contain features that separate regions with cracks from those that don't. Once a feature space has been decided upon a classification algorithm can first be trained on a selected training set and then used to classify the pixels in new images. The classification algorithm used in this thesis is a linear Support Vector Machine (see ch. 2.8).

3.4.1 Cracks

To be able to detect the cracks there has to be something that distinguish them from the rest of the cheese cut.

The cracks visible in the cut images can be divided into two categories: vertical and horizontal. The first appear as sharp, dark, elongated figures or even lines. The horizontal cracks can be seen as regions with more texture than the rest of the cut. The edges of these regions are also more uneven and jagged than the edges of eyes. For examples of cracks see Figures 9 and 10.

There seems to be no significant colour difference between normal regions and regions with cracks. Therefore the image is converted to a 256 grey colour scale and all the crack detection is done on the grey colour image. This saves a lot of extra computations.

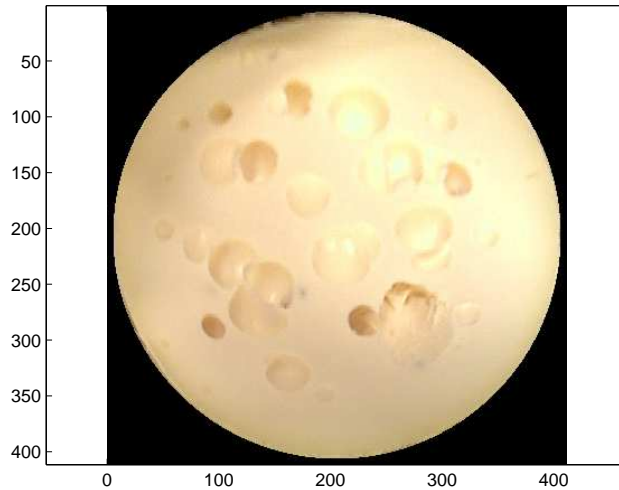


Figure 8: The end product of the segmentation. Everything outside the segmented region has been removed and the image has been cropped to a smaller size fitted to the segmentation.

3.4.2 Feature Selection

Two different approaches were tried to produce the feature space. One a more theoretical, brute force method and another more specially designed for this problem. The first one uses elementary operations such as derivatives (on different scales and directions) and Gaussian smoothing on the image. Then non-linear filters are applied to the result and all the combinations of these are used as feature space. The other uses specially designed filters that are programmed to be sensitive to the properties the cracks display and are therefore sensitive to regions with cracks. The result of these filters are then directly used as feature space.

Brute Force

The brute force method starts by using elementary operations on the images. The main operations were:

- Gaussian smoothing with a certain standard deviation.
- Derivative in x after applying Gaussian smoothing with a certain standard deviation a .
- Derivative in y after applying Gaussian smoothing with a certain standard deviation a .
- Canny edge detection

With all the different scales about 10 such operations were normally used in this project but in general say, n . Each of these results is stored in a matrix of the

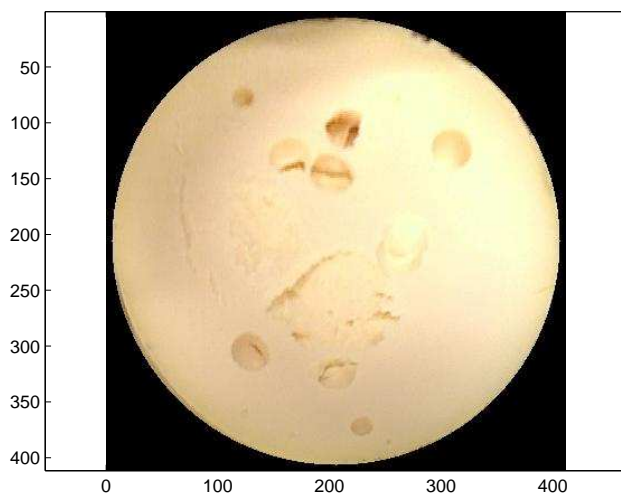


Figure 9: A cheese cut containing both vertical and horizontal cracks. The vertical can be seen as black lines and the horizontal as regions with more texture.

same size as the image with the filter response at the place of the pixel. Then two non linear functions were applied: $\max(0,x)$ and $\max(0,-x)$ which results in double the number of matrixes now with only positive entries. In general we now have $2n$ matrixes.

These matrixes were then multiplied element-by-element with each other in all possible ways (this should give approximate for second derivatives among others). Finally to get a more homogeneous feature space the resulting matrixes from above is convolved with a Gaussian (Gaussian smoothing). The result is a feature space of size $\sum_{i=1}^{2n} i$. In the normal case of 10 starting filters this means a total of 210 features per pixel which is quite a lot.

The idea is that this feature space will distinguish regions with a lot of textures from those lacking textures.

Special Design Filters

Just as the name suggests in this approach the feature space is produced by a number of specially designed filters. The filters are designed such that they will be sensitive to regions with cracks (most often giving a large output value for these regions). Some of them are based upon the difference in texture while others are based on the irregularity of the edges around cracks. Here follows a short description of the filters used.

Sum-Canny

In the cheese cut images since the cracks appear as regions with more texture one idea is that these regions should show up with a lot of edges when using an edge detector with the right scale on the image. A distinction between regions

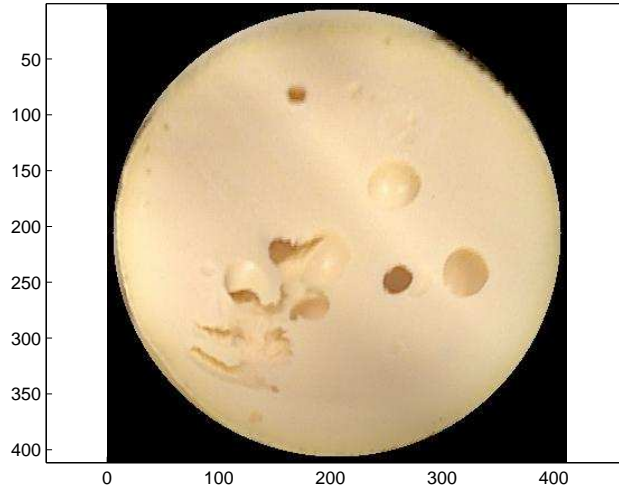


Figure 10: A cheese cut containing horizontal cracks. Seen to the lower left of the middle.

with normal cheese and cracks would then be the amount of edges in the region or the neighbourhood of a certain pixel.

To model this first a Canny filter (see ch. 2.6) was applied resulting in a binary image with ones representing edges. Then a sum of all the edges in a certain region around each pixel was computed and stored as the output of the filter. This will result in that smooth regions (i.e. no eyes or cracks) attain a small value, regions with eyes a slightly larger value (depending on the size and number of eyes) and regions with texture (cracks) an even higher value.

Sum-Corner

In regions with normal cheese (flat cheese and eyes) corners are not common (the eyes are round and therefore have no corners). The irregularity of the cracks on the other hand gives rise to a lot of corners. Especially the boundaries around the cracks have strong corners. For that reason one can find cracks by looking for regions with a lot of corners.

This filter starts with detecting corners using Harris corner detector (See ch. 2.7). Then the sum of all corners in a small region around each pixel is computed. The result should give high values for regions with cracks and small for normal cheese. Since most of the corners detected are on the boundaries of the cracks this filter will be most sensible to the boundaries of the cracks while the response inside a big crack might be small.

Median based filters

Applying a median filter on a greyscale image is a quite straight forward process. The pixels in the output image are set to be the median of all the pixels inside a specified region (usually a square) around the pixel. The result is a filter where

pixels with a large deviation from its surroundings don't have a large effect on the output. This means the filter is good at noise reduction and it's also its main application.

Here the median filter is manipulated to detect regions with texture, which can be interpreted as noisy regions. The way this is done depend on the specific filter.

Median-Median

In this filter the texture of the cracks are modelled as fast varying greyscale intensity. First a median filter with a certain square size is applied. That is each pixel is set to be the median of all the values in the square around it. Then the absolute difference between the original pixel value and the corresponding median value is computed by taking the original image (greyscale), deducting the median and taking the absolute value of the result. This will give a matrix with small values for smooth regions, larger for regions with texture and largest values for edges in the image since they differ a lot from the region around them. The texture areas will be represented with areas with semi high values.

Now another median filter is applied, this time with a bit larger size than the one before. Since the edge pixels usually are not very many their high values will not affect the median value much. Instead the semi-high values of the texture will give raise to the biggest values.

Median-Threshold

This filter starts just like the one above with computing the absolute difference between the (greyscale) image and a median filter of a square size applied to it. Instead of using another median filter to the result a threshold is used to get rid of the largest values which probably represent edges (around eyes and cracks) in the image. All values above this threshold are considered edges and therefore not interesting (since we are here looking for texture) and are set to 0. What remains are hopefully just values that comes from small deviations from the median value (i.e. texture). To get an output that represent the texture around the pixel all the values in a square (of a certain size) around the pixel are added together. The output of this filter should therefore ideally give a measurement of how much texture there are in the neighbourhood of the pixel.

A variation of the above filter is made by, for every pixel in the image, computing the median of the pixel values in a square around it. This median is then deducted from all the pixel values in the square. The absolute value of the difference is then compared to a threshold and all values within the square that are less than the threshold are added together to get the output for the current pixel.

3.4.3 Using the SVM

When a feature space have been decided upon it's time to train the SVM. This is done by manually selecting regions in a couple of images belonging to the different classes in use and train the SVM on these instances (here pixels).

The training was done on regions selected manually in a couple of different images containing cracks of different sorts. From these regions about 800 samples of each class were randomly selected to represent it in the training.

Once the SVM has been trained it can be used to classify all the pixels in

the cut images to detect cracks in these.

3.5 Whole Cheese Classification

To attain statistics on when cheeses with cracks appear it's only necessary to know if a cheese has a crack or not. For this to be done the cheeses passing the cut-up needs to be divided into two categories, contain crack and doesn't contain crack. To some degree it's only interesting to detect cracks of a certain size since small cracks pose a smaller or none-existing problem.

For this reason it's desirable to convert the information on pixel level attained from the SVM into information on cheese level (crack, none-crack).

The easiest way to do this would be to just check if there are any pixels in the image that are classified as cracks and if there are classify the whole cheese as a crack cheese. This will be a bad decision though, since a single wrongly classified pixel can make the decision on cheese level wrong. Actually this will lead to that most of the cheeses passing through are classified as containing cracks. A more stable algorithm is needed.

A attempt to program such an algorithm based on morphology was made. First the 3-class image which is the end result of the pixel classification was converted to a binary image where 1 represents crack and 0 none-crack.

A circle with radius 4 pixels was chosen as the structural element. First erosion is applied and immediately after that all regions containing less than a number of pixels are removed. All this to get rid of single pixels or small areas. Then dilate is applied to restore the remaining areas to their original size. After this the number of pixels still classified as crack are counted. The decision on whether a cheese contain a crack or not can then be made by thresholding the number of crack pixels needed to be classified as a crack cheese.

The threshold mostly used in this project was 0 (more than 0 pixels classified as crack).

4 Results

4.1 Segmentation and Rectification

In most cases the segmentation works very well. In some cases there are some problems.

The most common cause of problem is the outside of the cheese (paraffin wax) that can be seen in the lower part of the image under the cheese cut. This area has a similar colour to the cut and therefore there are always pixels on it that are classified as cheese in the classification process. This is usually not a problem though since the morphological operations remove these pixels. Although sometimes the pixels in this area are too many and too close to each other so that the morphological operations will not remove them all. This results in that the fit between the ellipse and the cheese cut is less good. The centre of the ellipse ends up a bit shifted downwards and its shape a bit warped so that its minor axis is too long (minor axis is almost parallel to the vertical axis).

In the image a part of the outside of the cheese in the lower region and a part just above the cheese cut will be inside the ellipse. This also means that these parts will show up in the rectified image after everything outside the circle has been removed. It should also have consequences for the rectification itself since the ellipse that is supposed to be transformed into a circle is wrongly fitted. It's hard to say what these consequences are but they are probably not very big.

To get rid of this problem trials with a third class as outside of cheese in the Bayesian-classification were performed. The result was good and if a higher accuracy is necessary this algorithm can be used. Although the algorithm with the third class added is a fair bit slower. For this reason and because the accuracy of the 2-class algorithm was considered good enough for this project it was used.

There are also problems with Grevé cheeses that had been stored a longer time than the normal Grevés usually studied here. These were easily distinguished from the rest by that the paraffin in use were coloured black so that the outside of the cheese appear black.

The problem with these cheeses was mainly that they have a slightly different colour than the normal ones. They are also a bit moister which makes them more likely to leave tracks and small cheese grains on the machine parts.

This results in bad classifications for these cheeses. So bad that the fit of the ellipse in this case is unsatisfactory.

It was quite easily solved by making new and specialized histograms for these specific cheeses in the same manner as the histograms for the normal ones were done (manual selection of the different classes). The result is an almost as good classification as the one for normal cheeses. It is not exactly as good because of the more disturbing environment with tracks and grains which in some cases still cause problems.

This solution of the problem with longer stored cheeses might itself be a problem if the system is to be fully automated. A decision on which of the two different sets of histograms have to be made either by using image analysis or by some other form of information, for example information on what kind of cheese that are processed on the line at the moment. As it's now this is needed anyway because not only Grevé but also other cheeses are cut on the same process line. In this project the images were visually examined and batches of Grevé selected.

4.2 Crack Detection

There were a few problems that arose during the crack detection. Some could easily be solved others were harder and required ad hoc solutions that in some cases might be hard to motivate theoretically. Most of the problems had to do with that the SVM was too sensitive and wrongly classified regions without cracks as containing cracks.

The two different methods to create the feature space were first tried parallel and they showed similar results with the difference that the brute force method needed a bit more computing time.

Time was also an issue for the work itself and to be able to accommodate a proper development and testing of the system focus had to be put on one of the two algorithms. The specially designed filter approach was chosen. Therefore, if nothing else is stated, the comments here after refer to that approach. Presumably the brute force method would produce similar results, problems and similar solutions would be applicable. However this can not be said for certain.

At first only two classes were used, normal cheese and cracks. It soon became apparent that at least one more class was needed. This because the eyes of the cheese had a strong tendency to be classified as cracks instead of normal cheese (even though eyes were apart of the class normal cheese). To solve this the two classes were transformed into three: flat cheese, eyes and cracks. This removed a large part of the previously wrongly classified regions as they now could be classified as eyes instead of being wrongly classified as cracks.

This was not enough. There were still quite a few wrongly classified pixels and some of them even in flat regions of the cheese cut. Finally the conclusion that some interior regions within the cracks appear similar to that of the flat cheese was drawn. What makes the human eye detect these regions as cracks is mostly the surrounding boundary. This boundary is quite different in looks from the normal cheese (flat regions and eyes) and it's therefore easy for us to detect. Once this boundary have been detected our perception led us to think that everything inside also is a crack region. Since the computer lacks this perception it can not detect these regions in the same way. Actually if these regions appear in the same manner as the flat regions they can not be detected at all. At least not on a local level.

The solution to this was to disregard these interior regions and instead concentrate on detecting the boundaries. This was easily done by only selecting the boundaries for the crack class in the training set.

This reduced the number of wrongly classified pixels but also results in that the SVM will not detect the close to flat regions within the cracks, only the pronounced boundaries. This unfortunately removes some of the information on how large area a certain crack covers. Another drawback is that while most cracks have pronounced boundaries some do not. These cracks will be hard for the algorithm to detect and most of them will therefore go by unnoticed.

Still after all this there were problems with the SVM being too sensitive. Sensitive in the sense that regions with just slightly abnormal eyes were often classified as cracks. To solve this experiments were made where the hyperplane (decision boundary) was moved in a direction such that fewer pixels were clas-

sified as cracks. Here a balance between being able to detect most cracks while still not getting to many false crack pixels were needed. The method used was trial and error.

Another problem with the crack detection is that it has a tendency to classify small deep eyes as cracks. These eyes are distinctly different from the other eyes in the way that they appear dark (since they are small and deep less light enters them and can be reflected). The reason they are classified as cracks are not totally clear but an educated guess is that the high contrast between the dark eye and the surrounding flat cheese give rise to detected corners and also affect the median based filters.

Possibly not enough of these small dark eyes were selected during the training. If this is the case maybe the problem can be solved there by including more of these specific eyes. Unfortunately not enough time was available to solve this problem.

As stated in section 1.2 there occur other faults than cracks in the cheese. These faults will have an appearance different from both the cracks and normal eyes. These regions were not present at all in the training of the SVM and it was therefore not certain how such regions would be classified. Testing a few cheeses that show the two other faults (overlapping eyes and many small eyes) showed that to a large degree these areas will be classified as cracks. This can be seen as both a drawback and an advantage.

It is a drawback in the sense that the goal was to detect cracks and establish statistics for these. This since now a portion of the cracks detected might actually be some other fault.

The advantage is that if you want to optimize the cheese produced then you probably want to minimize all faults not only cracks. Therefore it's a good thing that the other faults are also detected. In this case the name of the last class can be changed from cracks to plainly faults.

For an example of the final crack detection algorithm applied to the cheese cut in Figure 8 see Figure 11.

4.3 Whole Cheese Classification

A small test of the cheese classification was made using 15 handpicked images of normal cheeses and 15 handpicked images of cheeses containing cracks. The images were picked after the rectification and only cheeses which had a good result up until then were chosen. This to test only the crack detection itself.

From the 15 cheeses containing cracks the algorithm detected 11 (i.e. 4 false negative). From the 15 normal cheeses one was classified as containing a crack (i.e. 1 false positive).

Another test was made on 100 consecutive Grevé's passing the cut-up during a day were cracks were present (since cracks usually appear in groups most days have none). The images of these cheeses were visually examined and divided into two categories, containing crack and not containing cracks. There were 72 without cracks and 28 containing cracks.

The whole cheese classification was performed (i.e. all steps from segmentation to cheese classification). Of the 72 none-crack cheese 11 were classified as containing cracks (i.e. 11 false positive) and of the 28 cheeses containing cracks 17 were classified correctly (i.e. 11 false negative). 11 false positive is a large

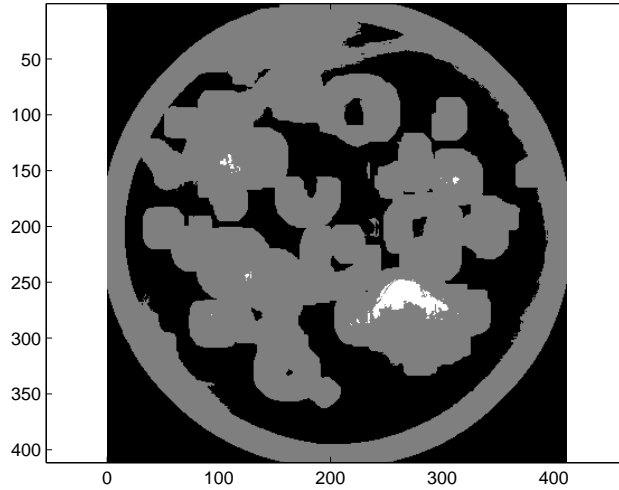


Figure 11: Result of crack detection applied to the cheese cut in Figure 8. Black - flat cheese, Grey - eyes, White - Crack. The upper half of the horizontal crack has been detected.

number. A closer look at the cheeses causing these false positive gives an explanation for the large number. At least 4 of the false positive can directly be related to regions with overlapping eyes and with some good will a few more (there seems to be some correlation between cracks and other faults so that they appear in the same batches). The remaining false positive seems to come from regions with dark eyes.

The 11 false negative are most of them cheeses which contain small cracks or cracks with undefined boundaries.

5 Conclusions and Future Work

The aim of this thesis was to evaluate the possibility to use image analysis and computer vision to reach statistics of cracks in cheese. The short conclusion of the evaluation is: yes it works. It's most certainly possible to develop a program that can detect cracks. Such a system has been developed in this project. Although the system developed is not ready for commercial use. The areas in which the system needs to be improved can be divided into there different fields.

- Computing time
- Sensitivity
- Automation

Computing time

Since this project was an evaluation of the possibility not an actual implementation little thought were put into the effectiveness and speed of the system. Some of the algorithms are therefore very time consuming and the result is that to go from an original image to the classified crack image takes about 1 minute. This is quite a lot of time especially if a realtime system is desirable. There are only a couple of seconds between two cheeses during cut-up. Though a realtime system is only needed if the information is to be used by the process line not if statistics is the goal. Also as said before not much thought were put into the effectiveness of the current system and therefore it's highly probable that the computing time can be drastically cut by a new implementation (maybe also on another platform since Matlab is not that fast).

Sensitivity

While the current system detects cracks it doesn't detect all. Depending on the requirements it could be that it detects too few. What's probably worse is the high occurrence of falsely detected cracks (false positive). In the current form the high number of these could surmount the number of real cracks detected making the statistics almost useless. Actually the current system should probably more be seen as a help devise for visually observing the images. That is someone should check the crack labelled images and sort out the ones that are false. Even just this should be a great help since instead of visually observing thousands of cheeses now only a few images needs to be examined and sorted.

There are quite a few ways to improve sensitivity. The algorithm can be improved by developing more crack sensitive filters to use in the feature extraction, adding specific troublesome regions (small, dark eyes for example) to the training set and further experimenting with the decision boundary. Another way to improve the sensitivity could be to manipulate the lighting of the camera setup. Nothing was done in this area in this project and the lighting of the images is not perfect. Setting up and experimenting with lamps and lighting from different angles could probably improve image quality and therefore also sensitivity.

Automation

As stated in the previous section because of the low sensitivity the system can't work properly without some extra control as it is now. There are also other

things that need to be changed for the system to become fully automated. First as it is now the camera is put up and taken down every day resulting in a slightly different view every day. Therefore the parallel lines needed for the rectification needs to be manually selected every day. This can easily be solved through buying a slightly more expensive but water resistant camera which can be permanently mounted. Another problem is that there are different kinds of cheeses processed by the same line and therefore all images are not taken of Grevé but other kinds of cheeses, that are uninteresting in this project. For this to be solved some external information about what kind of cheese is being processed needs to be added.

References

- [1] Edited by Hui, Y. H., Meunier-Goddick, L., Hansen, Å. S., Josephsen, J., Nip, W.-K., Stanfield, P. S., Toldr'a, F., Marcel Dekker *Handbook of Food and Beverage Fermentation Technology*, Marcel Dekker, Inc. 2004.
- [2] Blom, G., Enger, J., Englund, G., Grandell, J., Holst, L. *Sannolikhetssteori och statistikteori med tillämpningar*, Studentlitteratur, Lund, 2005.
- [3] Brosnan, T., Sun, D.-W. *Improving quality inspection of food products by computer vision-a review*, Journal of Food Engineering, Vol. 61, pp 3-16, 2004
- [4] Wang, H.-H., Sun, D.-W. *Correlation between Cheese Meltability Determined with a Computer Vision Method and with Arnott and Schreiber Tests*, Journal of Food Engineering, Vol. 67, Nr. 2, 2002
- [5] Wang, H.-H., Sun, D.-W. *Evaluation of the oiling off property of cheese with computer vision: Correlation with fat ring test*, Journal of Food Engineering, Vol. 61, pp 47-55, 2004
- [6] Caccamo, M., Melilli, C., Barbano, M., Portelli, G., Marino, G., Licitra, G. *Measurement of Gas Holes and Mechanical Openness in Cheese by image Analysis*, American Dairy Science Association, Vol. 87, pp 739-748, 2004
- [7] Gonzales, R. C., Woods, R. E. *Digital Image Processing*, Addison-Wesley Publishing Company, 1992
- [8] Hartley, R., Zisserman A. *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000
- [9] Spanne, S. *Föreläsningar i Matristeori*, Matematiska Institutionen Lund Tekniska Högskola, 1988
- [10] Burges, C. J.C *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery 2, pp 121-167, 1998
- [11] Forsyth, D. A. *Computer Vision a Modern Approach* Prentice Hall, 2003