IT University of Göteborg

CHALMERS | GÖTEBORG UNIVERSITY

Masteruppsats i Intelligent Systems Design
Master thesis in Intelligent Systems Design

REPORT NO. 2008:083
ISSN: 1651-4769

Department of Applied Information Technology or
Department of Computer Science

Automatic detection of images containing nudity

Image detection using artificial neural networks and statistical methods

ANDREAS CARLSSON
ANDREAS ERIKSSON
MIKAEL ISIK

CHALMERS | UNIVERSITY OF GOTHENBURG

IT University of Göteborg
Chalmers University of Technology and University of Gothenburg
Göteborg, Sweden 2008

Abstract

This thesis discusses the possibilities of detecting images containing nudity using computer algorithms. We are only focusing on sexually explicit images. Our approach is to extract features such as skin, faces and regions, which can be used to classify images. We have investigated the advantages of the two color spaces RGB and IHLS when detecting skin. The difference in performance between the two, are illustrated in ROC graphs. The technology used is artificial neural networks, statistical methods and advanced image processing. Artificial neural networks are used for skin pixel segmentation, face detection, and image classification. Gaussian mixture models have been tested, but was too computationally heavy and was also outperformed by artificial neural networks. The separate parts performs well, but our approach using an artificial neural network with features as input does not perform as well as expected in its current state, and needs some modifications, which are proposed in the section future work.

The research problem was proposed by NetClean Technologies Sweden AB.

Keywords:
Artificial neural network, Gaussian mixture models, image processing, skin detection, image classification, CBIR

Sammanfattning

Den här uppsatsen diskuterar möjligheterna att automatiskt detektera bilder med naket innehåll, med hjälp av datoralgoritmer. Vi har fokuserat på sexuellt laddade bilder. Vårt tillvägagångssätt är att extrahera egenskaper så som hud, ansikten och regioner, som kan användas för att klassificera bilder. Vi har undersökt fördelar med de två färgrymderna RGB och IHLS, där skillnaden mellan dem illustreras i ROC-grafer. Tekniker som används är artificiella neurala nät, statistiska metoder och avancerad bildbehandling. Artificiella neurala när används för hudpixelsegmentering, ansiktsdetektering och bildklassificering. Gaussiska mixturmodeller har testats, men används inte, då tester visade att metoden ger sämre resultat än artificiella neurala nät. De olika delarna ger bra resultat, men vår metod att använda artificiella neurala nät även för de individuella egenskaperna ger ett sämre resultat än förväntat i sin nuvarande form. Därmed behövs vissa modifikationer, vilka föreslås i avsnittet “future work”

Examensarbetet föreslogs av och genomfördes hos NetClean Technologies Sweden AB.

Nyckelord:
Artificiella neurala nät, Gaussiska mixturmodeller, bildbehandling, hudigenkänning, bildklassificering, CBIR

Acknowledgments

We would like to thank everybody at NetClean Technologies for giving us the opportunity to do our thesis work, as well as their support throughout the project.

We would also like to thank our supervisor, Claes Strannegård for feedback on our work and thesis.

Abbreviations

ANN	Artificial neural network
fp rate	false positive rate
GMM	Gaussian mixture model
HLS	Hue, lightness, saturation, colorspace
HSV	Hue, saturation, value, colorspace
IHLS	Improved hue, lightness, saturation, colorspace
PCA	Principal component analysis
RGB	Red, green, blue, colorspace
ROC	Receiver operating characteristics
SVM	Support vector machine
tp rate	true positive rate
YCbCr	Luminance (Y), blue chrominance (Cb), red chrominance (Cr), colorspace
YUV	Luminance (Y), two chrominance (UV)

Table of Contents

1 Introduction

The use for automatic image classification is ever increasing, since more and more images are used, due to the use of computers in every day life. Our problem domain is to look into the possibilities of automatic classification of images containing nudity.

1.1 Legal limitations

NetClean Technologies Sweden AB and the authors have signed a legal agreement. Due to legal matters and confidential information regarding the application during the development, no detailed information about the main application, outside our problem domain, may be included in this report.

1.2 Research problem

Our system will be used as intelligence in a larger system, developed by NetClean Technologies Sweden AB. Our problem domain is to look into the possibility of detecting undesired or unsuitable pictures. The aim is to develop an approach to classify images, based on the level of unsuitability.

1.3 Research question

Due to the problem behind the research in this project, where the images should be the classified, based on the content of the images, the research question for this thesis is stated as:
How can images containing nudity be detected?

The first step is to define an undesired image. We are solely focusing on images that are sexually explicit. Thus, we do not take other objects, such as symbols and other offensive images, into consideration. Our approach is to extract a number of features from the pictures. Every feature is supposed to give some information of the content of the image. These features should then be analyzed, to see which ones to use in the final detection algorithms, and if they can be implemented in the limited period of about 20 weeks we have to do our thesis work and create our system.

1.4 Classification features

In order to come up with features, we started on a high level with fuzzy descriptions. The descriptions can then be divided into sub-features which can be found in images.
The three high level features we came up with are
- Nudity
- Shape
- Object classifier

1.4.1 Nudity

To detect nudity, the key is to be able to detect skin. To avoid delimiting the skin detection accuracy, a system should be able to detect skin with different skin color, and different lighting conditions as it must be taken into account that the pictures can be taken in different environments and with different cameras, including webcams.

1.4.2 Shape

To detect shape, some kind of template matching is a way to go. The easiest example is the use of a template which a program tries to match in an image by stepping through it, scanning every possible location. If the program can match the template in the image, the object is marked as detected.
To detect several objects with non-constant size, a number of templates are created and a program can then scan the picture for different scales of the templates, to make it possible to detect different sizes of the object. Rotation can also be used to detect objects that are not placed in an upright position.

1.4.3 Object classifier

An object classifier is meant as a system which searches for a specific object. Examples of objects to classify which might be useful when detecting unsuitable images are faces and breasts. If faces are the objects of interest, a successful classifier should find all faces in an image.

1.5 Purpose

The purpose is to look into the possibilities of detecting nudity in images, using computer algorithms. The target groups of the thesis are people with an interest in image processing, mainly concentrating on skin detection, classification and CBIR (content based image retrieval) as well as those with an interest in artificial neural networks and statistical methods.

1.6 Related work

Looking into previous work made in the problem domain, there are a number projects, that have dealt with similar problems. It is noticeable that there are a number of different approaches used to get a similar result. Some methods that can be noticed in a number of projects are Gaussian distribution, Gaussian mixture models, artificial neural network, Bayesian networks, histograms, geometrical shape and thresholds.

In [Brown et al 2001] the authors are using a self-organizing map, SOM (introduced by Kohonen), to detect skin and the aim is to be able to detect skin using a real-time hardware based implementation of the system. In their work, they found out that their system using a SOM-network was better, though just by 1-2%, than Gaussian mixture models, for detecting skin pixels. The samples shown in the report shows a too large amount of both falsely detected skin and skin that is not detected, for our application.

In [Lee and Yoo 2002], an approach is described where an elliptical boundary model is used to classify skin chrominance from non-skin chrominance. Six different chrominance spaces are tested. The authors focus on the comparison between elliptical boundary model and Gaussian mixture models, and the result they present is that the latter gives a worse result. It is not clear what the training data is. 2000 skin images and 4000 non-skin images is used but no segmentation is described and nor is the output. It is thereby unclear if the output is classified images or pixel-wise classification. The results is about 95% true positive rate, but this number is reached with a high false positive rate at about 35%.

Jones and Rehg [Jones and Rehg 1999] are looking into statistical methods for skin detection. The color space they are working in is RGB, and they point out that other color spaces, like YCbCr or HSV will not improve the skin detection performance. The performance is measured by ROC-graphs. Their key findings are that histogram models are superior to Gaussian mixture models and that skin color may be a very good cue when classifying naked people in an image.
They are also trying to create an adult image detection using neural networks, where a number of features are used as input. The features used for the training are manually classified from a large image set.

In [Forsyth/Fleck et al. 1996] a system is described which aims to detect naked people. It is divided into two steps. The first is to find areas with skin color in an image. The second step is to

find elongated regions of the skin areas detected in the first step and try to conclude if the groups are connected human limbs. The authors state: “Images containing sufficiently large skin-colored groups of possible limbs are reported as potentially containing naked people”. The skin filter uses a log-opponent representation of RGB and multiple median filters are used to extract smoothed texture. The article was published in 1996 and the timing constrains have changed. The system described in the article can classify about 10 images per hour and it is impossible to speculate how much faster such a system is today. As described it is not an option for a background component, which is supposed to run in near real-time, aiming at only a short delay of a few seconds.

The approach described in [Zheng et al. 2004] is to detect skin by building a Maximum Entropy Model for the skin distribution. The model uses the constraints on color gradients of adjacent pixels and is then combined with Bethe Tree and Belief Propagation. The output from the skin detection is a skin map in grayscale where the brightness is equal to the probability of skin. The next step is to extract features and fit ellipses, global fit ellipse and local fit ellipse. These features are then used in a pattern recognition system, consisting of an feedforward artificial neural network with one hidden layer. There are no performance numbers published in the article, but a visual interpretation of an ROC-graph in the article shows a true positive rate at about 80% when the false positive rate is about 10% and 90% true positive rate when the false positive rate is about 18%.

2 Image processing

This section describes the theory behind the technology used in this project.

2.1 Color theory

To give a brief background, a picture can be described as a matrix of pixels and the picture can be in color, grayscale or black and white. There are a number of ways to describe the colors, where the most common way is to use RGB. An RGB picture can be described as a matrix with a size of m x n pixels and 3 color channels, i.e. red, green and blue.

2.1.1 Color depth

The term color depth refers to how many bits (a bit is 0 or 1), that is used to represent a colored pixel. The larger number of bits used, the more accurate color. However an increased color depth is proportionally more memory consuming.

1- bit	*Monochrome or simple black and white. A pixel can solely be black or white.*
8-bit grayscale	*Only one color channel. $2^8 = 256$ possible shades of gray.*
8-bit indexed color	*A palette of colors is used to save memory. The maximum number of colors $2^8 = 256$ colors*
15-bit color	*Referred to as HighColor. The components can be described in $2^5 * 2^5 * 2^5 = 32^3 = 32\ 768$ colors.*
16-bit color	*Also referred to as HighColor. The difference from the model used above is that the green channel gets 6 bits and thereby twice the number of possible colors. This alteration is made due to the fact that the human eye is more sensitive to green. The total number of colors 9 is $2^5 \times 2^6 \times 2^5 = 32*64*32 = 65\ 536$ colors.*
24-bit color	*Referred to as TrueColor. Each channel has 8 bits, which implies $2^8 \times 2^8 \times 2^8 = 256^3 = 16\ 777\ 216$ different colors.*

Table 2.1. Description of some common color depths.

A matrix for 24-bit RGB color representation could look as seen in figure 2.1. There are 3 separate color channels with values $\in$ [0,255] which are equal to hexadecimal values 0x0 to 0xFF. A red pixel gets the value [255,0,0], a blue pixel [0,0,255] and a yellow is a mix of red and green => [255,255,0]. If all of the three RGB-components have the same value, the pixel will be a some shade of gray, and the shade will fade from black to white, where larger values will imply a lighter shade, i.e. close to white.

red			*green*			*blue*			
255	*255*	*255*	*0*	*255*	*255*	*0*	*0*	*255*	
0	*255*	*128*	*255*	*0*	*128*	*0*	*255*	*128*	
0	*0*	*0*	*0*	*255*	*0*	*255*	*255*	*0*	

Figure 2.1. A 3x3 matrix to show how the values ∈ [0,255] in each of the three RGB-channels is combined and examples of the resulting colors. For example, the pixel in the middle has a red value of 255, a green value of 0 and a blue value of 255. This implies the color magenta. The right column has equally large RGB values and thus, there can only be shades of gray.

Due to the nature of the RGB color space, a cube is one way to depict the color model. A cube is also a good way to show the dependence between the different axes. In figure 2.2, the red and green axes are given as perpendicular on the X-axis and Y-axis. The blue axis is perpendicular to both the red and green axes, and thus it can be seen as a Z-axis. In the origin, where all three axis start the value is [0,0,0]. This means there is no color in any of the color channels and the result is the color black.

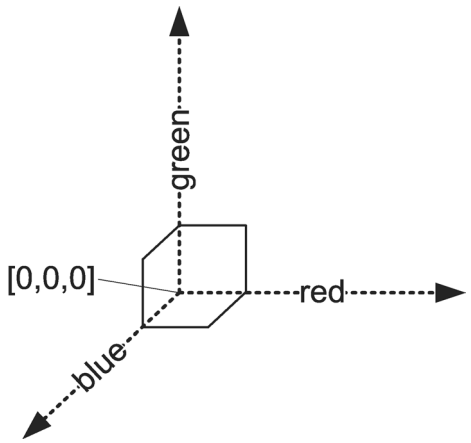
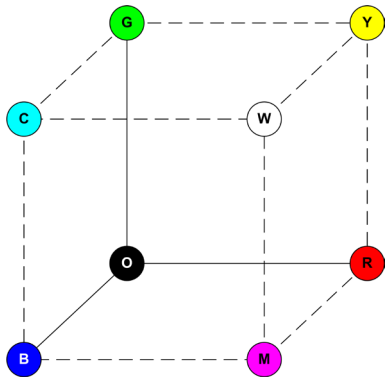


Figure 2.2. The red, green and blue colors depicted as X-, Y- and Z-axes.

To create a cube, two hypothetical lines can be drawn in parallel with the green and one the blue axes. The point where the lines meet will be equal to the color cyan. A new hypothetical line can then be drawn in parallel with the red axis and a new line parallel to the green axis. This intersection point will be equal to color yellow. In a similar way, the red and blue intersection lines will be equal to the color magenta. A new set of imaginary lines from the new intersection points will close the cube and this last intersection point is marked by a white dot. The concept is shown in figure 2.3.



R	*Red*
G	*Green*
B	*Blue*
C	*Cyan*
M	*Magenta*
Y	*Yellow*
W	*White*
O	*Origin = Black*

Figure 2.3. When drawing lines parallel to the red, green and blue axes, a cube occurs. The colored dots on the corners refers to the colors on the axes and are described in the table to the right.

2.1.2 Achromatic axis

When the concept of a cube is introduced, the achromatic axis can be depicted in an easy way. As described earlier, pixels with the exact same value for all three channels in the RGB color space, will imply a shade of grayscale. R,G,B = [0,0,0] will be equal to a black pixel and R,G,B = [255,255,255] is a white pixel. In other words, when the distance from the origin will increase equally for all channels, it will only produce grayscale. An imagined axis from the origin, through cube to the opposite upper corner will solely contain grayscale shades. Because of this fact, it is referred to as the achromatic axis. Please note that the axis stretches through the three-dimensional cube in an angle of 45° from the R-, G- and B-axis respectively.

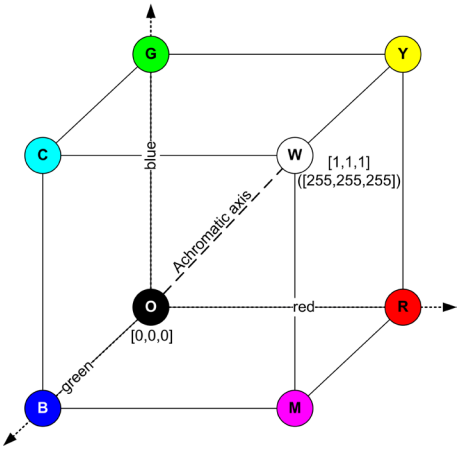


Figure 2.4. Please note the achromatic axis which starts in the origin (equal to black) and ends in the white corner. Along the achromatic axis, only shades of gray are defined.

2.1.3 CMYK

It it well worth to mention the CMYK color space when talking about the color cube. CMYK is an abbreviation for cyan, magenta, yellow and key, where key is the color black. It is used for prints, both in home printers and for industrial use. It is commonly referred to as four color or process color. The main difference between RGB and CMYK is that RGB starts from black and then red, green and blue is added to create the colors wanted, and thereby, RGB is an additive color model.

CMYK on the other hand is a subtractive color model. When a print process starts, there is only a white paper. The more colored ink added to the paper, the darker the color gets. Thus, CMYK is called subtractive colors, it removes or subtracts from white, in opposite to RGB, which add colors. Cyan, Magenta and Yellow can be used, but by introducing the Key, the amount of ink needed is decreased when printing dark areas of a picture or document. The problem when converting between the two concepts is that different devices will have its individual color properties, and therefore, it is hard to make a generic formula. This problem is less significant if color profiles are used.

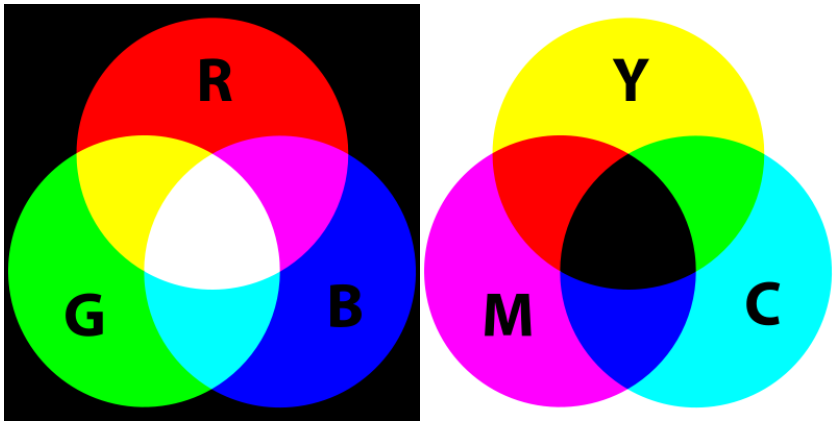


Figure 2.5. Left: Additive colors, like RGB start from black, and the more colors added, the brighter the combined color gets. Right: Subtractive colors start from white. For every added color the result gets darker. When cyan, magenta and yellow is fully saturated, it results in black.

2.1.4 HLS and HSV

Both HLS and HSV, sometimes written as HSL and HVS respectively, are commonly used in image processing. HLS is an abbreviation for hue, lightness and saturation. It is a color space, which describes the color in a different way than RGB. The base color is chosen as an angle in a circular disc. The saturation is given by the distance from origin, where the saturation increases with the distance from the origin. This can be represented as a two-dimensional circular map, but when introducing the lightness, the model needs a third dimension. This is done by adding an axis perpendicular to the hue and saturation, i.e. a Z-axis. The model is called 3D-polar coordinate color space.

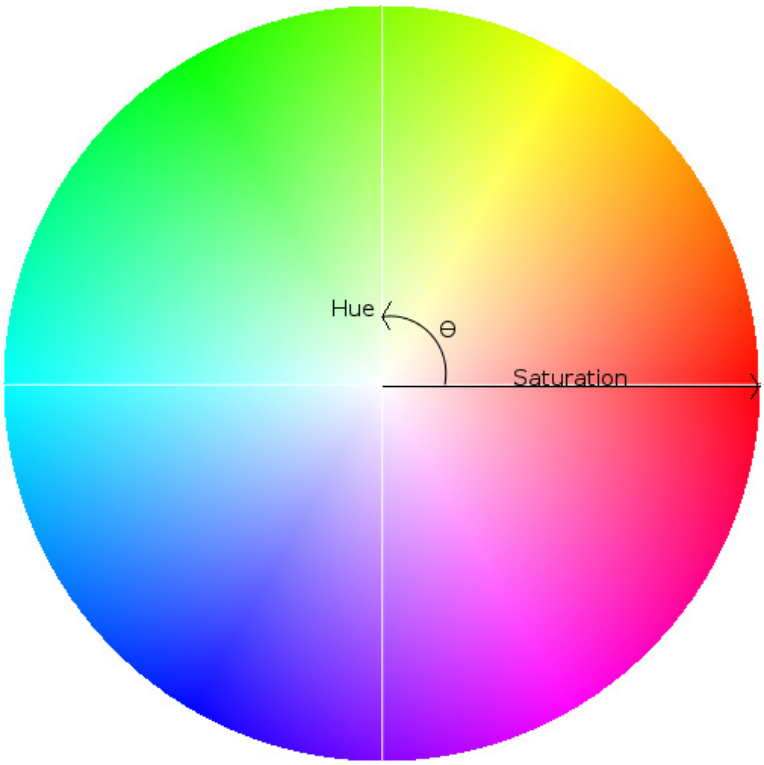


Figure 2.6. The figure depicts how the hue is given as an angle, and varies from red equal to 0° over the color spectrum as the angle increases. The saturation changes with the distance from the origin.

HSV represents the colors in a similar way as HSL. The abbreviation means Hue Saturation Value, where the value component replaces the lightness from HSL.
The hue component is given by equation 2.1 and equation 2.2.

$$H^* = \arccos\left(\frac{R - \frac{G}{2}\frac{B}{2}}{\sqrt{R^2 + G^2 + B^2 - RG - RB - BG}}\right) \qquad \textit{(eq. 2.1)}$$

$$H = \begin{cases} 360° - H^* \\ H^* \end{cases} \qquad \textit{(eq. 2.2)}$$

2.1.5 IHLS

IHLS is a modified color space presented in [Hanbury & Serra 2004], where the I before HLS stands for improved. The idea is to use the properties of the HLS color space, but with a significant difference. In HLS and HSV, the saturation depends on the luminance. In IHLS, the saturation is defined in a way, where it does not depend on the value of the luminance [Hanbury & Serra 2004, pp 2-8].

Saturation definitions for HLS, HSV and IHLS is shown in eq. 2.3, 2.4 and 2.5 respectively.

$$S_{HLS} = \begin{cases} 0 & \textit{if } \max(R,G,B) = \min\left(R,G,B\right) \\ \dfrac{\max\left(R,G,B\right) - \min\left(R,G,B\right)}{\max\left(R,G,B\right) + \min\left(R,G,B\right)} & \textit{if } L_{HLS} \le 0.5 \\ \dfrac{\max\left(R,G,B\right) - \min\left(R,G,B\right)}{2 - \left(\max\left(R,G,B\right)\min\left(R,G,B\right)\right)} & \textit{otherwise} \end{cases} \qquad \textit{(eq 2.3)}$$

$$S_{HSV} = 1 - \frac{\min(R,G,B)}{L_{HSV}}, \textit{where } L_{HSV} = \max(R,G,B) \qquad \textit{(eq. 2.4)}$$

$$S_{IHLS} = \max(R,G,B) - \min(R,G,B) \qquad \textit{(eq. 2.5)}$$

2.1.5.1 IHLS Conversion

When applying the IHLS color space, the red, green and blue values in RGB are converted to hue, saturation and luminance values. The conversion formula for hue can be seen in equation 2.1-2.2, and saturation is defined in equation 2.5. The lightness is in fact the color RGB-picture converted to grayscale. To convert grayscale, the red, green and blue components are weighted and added to a new vector. The sum of the red, green and blue values, used in IHLS, is given as

$$0.2126 \times red + 0.7152 \times green + 0.0722 \times blue \qquad \textit{(eq. 2.6)}$$

for every RGB-pixel in the image. These specific values are recommended for HDTV standards by [ITU 2002]. Some computer applications have slightly different weights but the results are similar.

2.1.6 YCbCr and YUV

YCbCr is a color model used in video. Unfortunately it is often confused with YUV. When people state the use of the YUV color space in image processing, YCbCr is what they usually mean. YUV is used for analog video (e.g. PAL) and not for digital processing.
In YCbCr, Y refers to the luminance component. The Cr and Cb components refer to the red and blue chrominance, as how much each components deviate from gray.

2.1.7 Brightness, luminance and lightness

There are a few different ways in which the brightness of an image can be described. Due to the similarity, the different expressions can sometimes be confused.

2.1.7.1 Brightness

Brightness is perceptual and can not be measured in any standardized way. This means the brightness differs in how bright something is perceived. CIE, the Commission Internationale de l´Eclairage (International Commission on Illumination) defines it as “the attribute of a visual sensation according to which an area appears to emit more or less light”.

2.1.7.2 Luminance

Luminance, often denoted as Y, is the luminous intensity (measured in Candela) per surface area, or the SI-unit cd/m². The luminance can be computed as a weighted sum, varying with a number of standards. IHLS uses a sum described in eq. 2.6.

2.1.7.3 Lightness

The human eye’s brightness response is nonlinear and this is taken into account in the concept of lightness. For example, if there are two light sources, and one has a luminance 18% lower than the other, it appear to be half as bright [Lightness]. Lightness is one of the components in the two color spaces Lab and Luv [Luv], both created by CIE [Fairchild 2005].

2.2 Preprocessing

Preprocessing can help to restore the original color or illumination of images, it can also bring out contours and details in otherwise blurry images. Preprocessing of images is important when images of different origin are used.

2.2.1 Histogram equalization

Histogram equalization is often used to enhance the contrast in images by redistributing the intensities in the image. The process is not computational heavy and fully reversible if the equalization function is known. When using histogram equalization on color images it can be a good idea to first convert the image to a color space other than RGB. Not doing so can otherwise result in dramatic changes in the color balance when the process is performed on the RGB channels separately. The Lab color space or the HLS are recommended as these will not affect the color balance in the image.

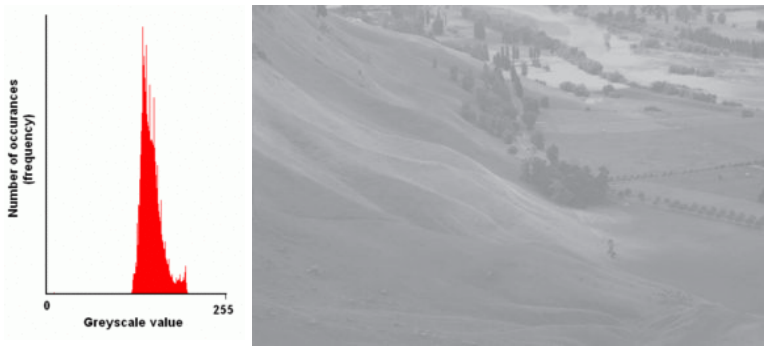


Figure 2.8.. A histogram and the original, unprocessed image.

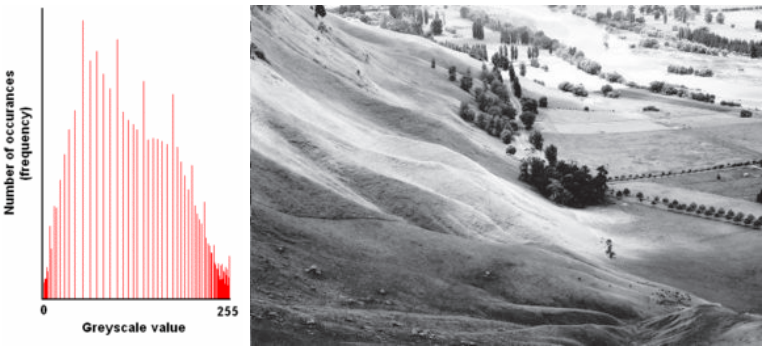


Figure 2.9. After the histogram equalization, the image appears much clearer, as the color spectrum is re-distributed.

2.2.2 Lighting Correction

Lighting correction is a technique that can for instance remove shadows or adjust dark areas in an image. Especially faces that suffer from poor lightning conditions such as shadows on half the face can benefit from lightning correction.

2.2.3 Noise reduction

Noise in images can be removed using smoothing filters. These filters removes unwanted pixels by adapting them to their surrounding pixels. The negative effect of this is that it can also remove important details. Some common techniques are averaging mask and median filter.

2.2.4 Sharpening

Some images that are blurry may have to be enhanced before they can be used. The blurriness can come from an error or as the natural effect of a special image acquisition method. The sharpening process finds edged and points that are of interest and enhances their contrast. A method that is commonly used is the Laplacian mask.

2.3 Segmentation

Segmentation, in the area of image processing, means one tries to extract the parts of the image that satisfies pre-determined conditions. Segmentation can be used in pattern recognition, medical imaging etc., where one or more parts of an image is interesting, not the entire image.
We have separated segmentation into pixel-wise segmentation and region-based segmentation.

2.3.1 Pixel-wise segmentation

Pixel-wise segmentation methods considers only one pixel at a time without regard to surrounding pixels.

2.3.1.1 Static thresholds

The simplest segmentation uses a static threshold. Its basic function is to check if the input is larger than, or in some cases smaller than, a threshold value. When working with controlled environments, this is possible and very predictable. In uncontrolled environments however, the input is often too complex for a system using static thresholds. If the input is an image, a controlled environment would be equivalent to the same place, the same objects and the same lighting conditions. If anything is changed, the risk is that the input value is changed and the threshold value is too large or too small.

2.3.1.2 Gaussian mixture models

Gaussian mixture models are good to use when you are working with incomplete data and when the data is spread in a way that makes it impossible to describe it with only one Gaussian distribution. This is the case with skin-pixels. Each distribution in a Gaussian mixture model is called a component (Γ). Each component has three unique parameters, the weight (ω), the mean-vector ($\boldsymbol{\mu}$) and the covariance matrix ($\boldsymbol{\Sigma}$). The probability density function for multivariate Gaussian distributions is used (equation 2.4).

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \boldsymbol{\Sigma}}} e^{-\frac{1}{2}((\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))} \qquad \textit{(eq. 2.4)}$$

When training a GMM each data has to be fit to a component. This can be done in several ways and the most common one is the expectation-maximization algorithm. The expectation-maximization algorithm consists of two steps. The first step (expectation, equation 2.5) is to calculate the probability that data x_j was generated from component k.

$$p_{jk} = \frac{\boldsymbol{\omega}_k P(\mathbf{x}_j | \boldsymbol{\Gamma}_k)}{\sum_{i=1}^{K} \boldsymbol{\omega}_i P(\mathbf{x}_j | \boldsymbol{\Gamma}_i)} \qquad \textit{(eq. 2.5)}$$

In the second step (maximization, equation 2.6-2.8) all parameters are updated. These two steps are repeated until the algorithm finds a local maximum.

$$\boldsymbol{\omega}_k^{new} = \frac{1}{N} \sum_{j=1}^{N} p_{jk} \qquad \textit{(eq. 2.6)}$$

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{j=1}^{N} p_{jk}\mathbf{x}_j}{\sum_{j=1}^{N} p_{jk}} \qquad \textit{(eq. 2.7)}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{j=1}^{N} p_{jk}(\mathbf{x}_j - \boldsymbol{\mu}_k^{new})(\mathbf{x}_j - \boldsymbol{\mu}_k^{new})^T}{\sum_{j=1}^{N} p_{jk}} \qquad \textit{(eq. 2.8)}$$

One way to know when the algorithm is finished is to calculate the log-likelihood (equation 2.9), which is faster [Mardia et al 1979], for the probability density function (equation 2.4). If the log-likelihood does not change much over two iterations then the algorithm has probably found a local maximum.

$$l(\mathbf{x}) = -\frac{n}{2}\log\left|2\pi\boldsymbol{\Sigma}\right| - \frac{n}{2}tr(\boldsymbol{\Sigma}^{-1}\mathbf{S}) - \frac{n}{2}(\bar{\mathbf{x}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu}) \qquad \textit{(eq 2.9)}$$

$$\mathbf{S} = \frac{\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T}{n} \qquad \textit{(eq. 2.10)}$$

2.3.2 Regions based segmentation

Region based segmentation means, regions of several pixels are used, instead of just process a single pixel at a time.

2.3.2.1 Region growing methods

The first region growing methods used seeds. Seeds in the image is expanded by iteratively comparing the neighboring pixels intensity with the regions mean. The placement of the initial seeds are often very important and has to be carefully selected. Other methods use the same technique but without the seeds. It starts with the first pixel and assigns it to a region, it then checks all neighboring pixels and compares them. If they are to different a new region is created for the second pixel. The algorithm continues until all pixels belong to a region.

2.3.2.2 Region splitting and merging

A different approach to finding regions is the merge and split technique. It starts with one big region for the whole image and then splits it into four quadrants if any of the four quadrant differs from the original region. Similar regions are then merged if they are of the same size and adjacent to each other. The process is stopped when no more merging or splitting is possible. Many variations on the merge and split technique has been developed that uses different merge and split conditions.

2.3.2.3 Active contours, deformable models, and gradient vector flow

Snakes can be very helpful when trying to find the boundaries of an object. It can be described as a computer generated curve that moves within an object and either grows or shrinks in order to fit the boundaries. The previous versions of snakes had problems with moving into boundary concavities or indentations and also moving towards objects that are to far away. This problem seems to have been solved by Chenyang Xu and Jerry L. Prince and their use of Gradient Vector Flow. They acquire a force field by applying a diffusion matrix. This force field then draws the snake towards the boundaries in the image. Jerry L. Prince and their use of Gradient Vector Flow.

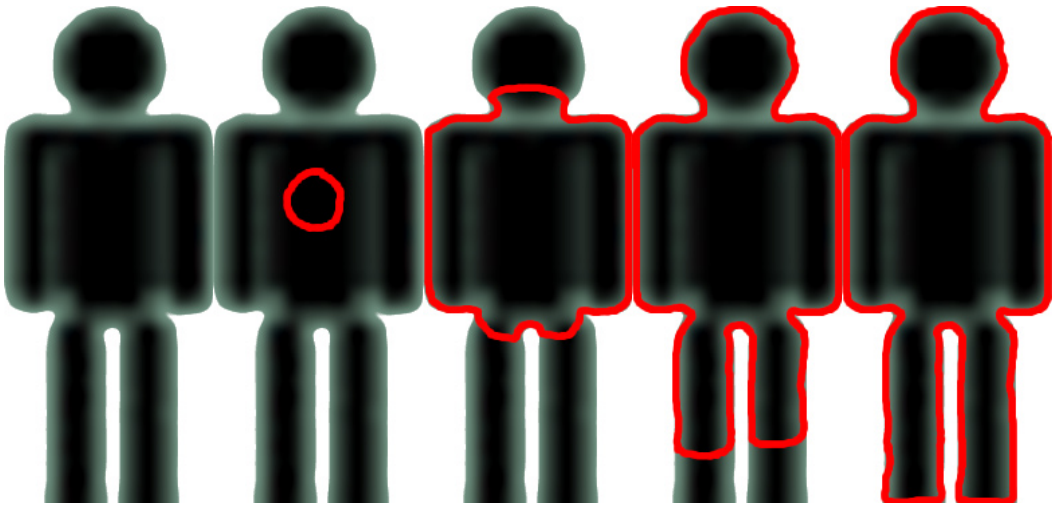


Figure 2.7. A curve is created and expands within the object and stopping at the outer edge. When finished, the outlines of the object is covered, which means the shape of the object is known.

2.4 Texture filter

Human skin is very smooth and that is a good property to use when removing pixels that was falsely classified as skin-pixels by the pixel-wise classification. The texture-filter calculates the standard deviation for the luminance in all adjacent pixels. If the standard deviation is above a threshold it will be classified as a non-skin-pixel.

2.5 Morphological image processing

Morphological image processing is a set of tools that are useful when working with regions. Skeletons, boundaries and convex hull are examples on morphological operations to describe region shape. Dilation, erosion and pruning are examples on morphological operations that are used for pre- or postprocessing. Grayscale images can be used for morphological image processing but the most common method is to use binary images.

2.5.1 Dilation and erosion

Dilation and erosion are two fundamental morphological operations. Other more advanced morphological operations are based on these two. Dilation means to grow a region and erosion means to shrink a region. A structuring element is traversed through the image and the shape of the structuring element is what determines the effect of the dilation or the erosion.

2.5.1.1 Dilation algorithm

A structuring element is traversed through the image. If the structuring element's origin is over a region pixel then convert all the structuring element's pixels to region pixels. Figure 2.12 shows a region of size d x d before the structuring element of size d/5 x d/5 is traversed through it. Figure 2.13 shows the new region after the dilation. The dotted line in figure 2.13 is the old region's boundaries.

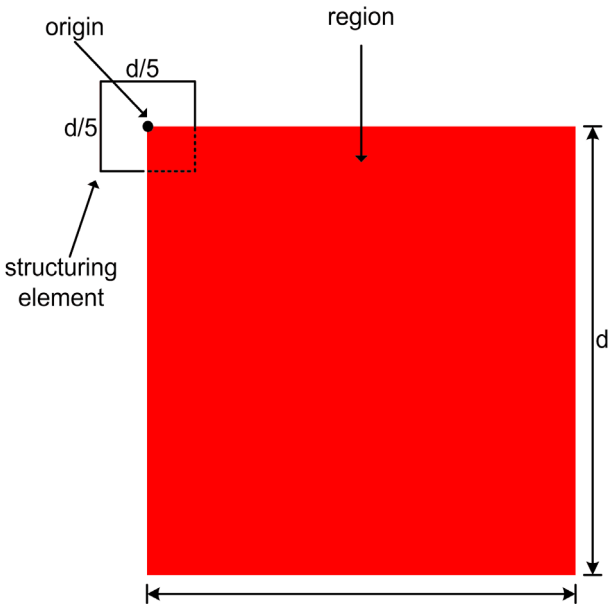


Figure 2.12. A square structuring element starts to traverse through a region with size d x d.

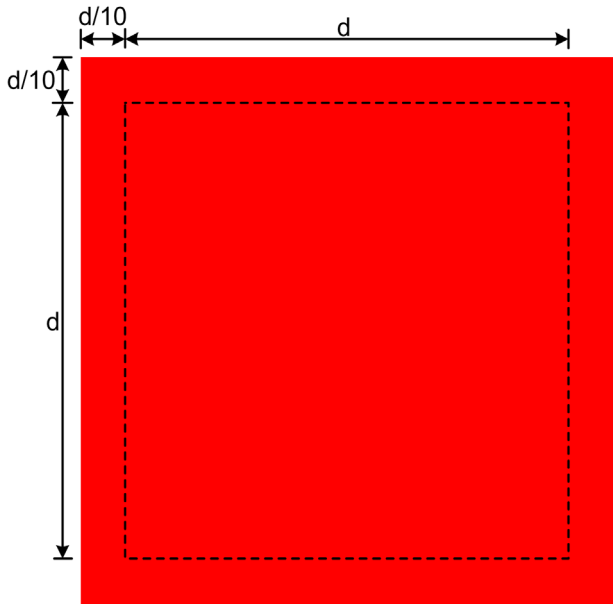


Figure 2.13. After dilation, the region has grown by half the size of the structuring element. The original size of the region is shown by the dotted line.

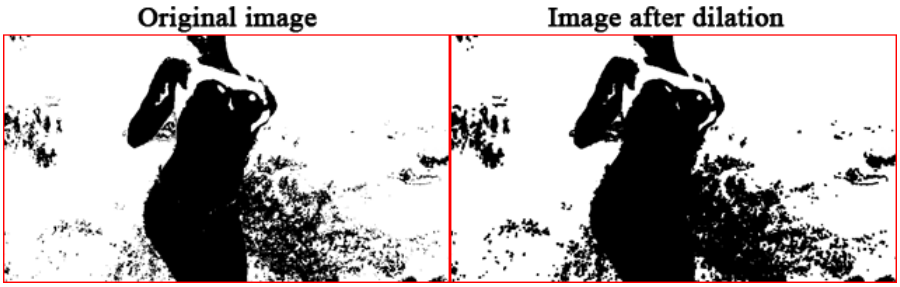


Figure 2.14 shows the original image, and the image after dilation is tested in Matlab. The difference is seen by looking at the smallest regions in the original image and comparing them to the correspondent regions in the right image.

2.5.1.2 Erosion algorithm

A structuring element is traversed through the image. If the structuring element's origin is a region pixel and at least one of the other pixels in the structuring element is a background then convert the origin to a background pixel. Figure 2.12 shows a region of size d x d before the structuring element of size d/5 x d/5 is traversed through it. Figure 2.15 shows the new region after the erosion. The dotted line in figure 2.15 is the old region's boundaries.

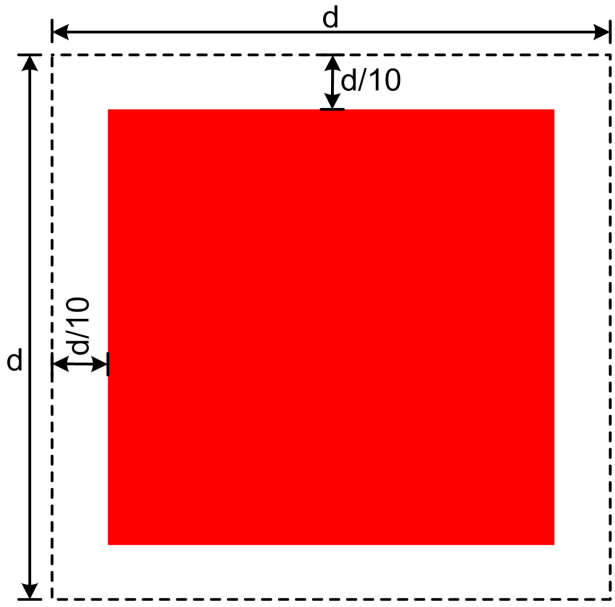


Figure 2.15. The region after erosion. The original size of the region is shown by the dotted line.

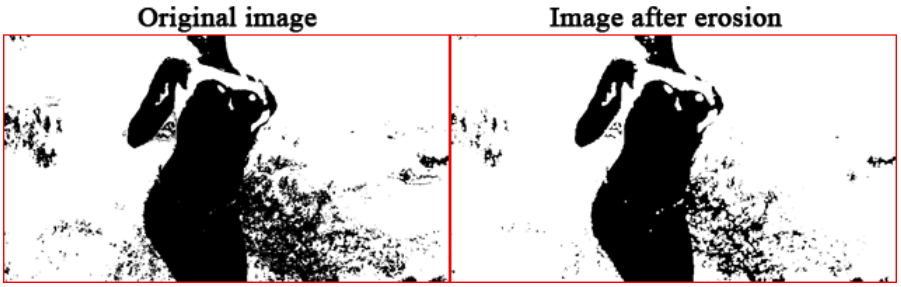


Figure 2.16. The original image and the image after erosion. Small regions which is removed. By adjusting the structuring element or running the algorithm more than once, it is possible the remove regions of other sizes.

2.5.2 Opening and closing

Dilation and erosion are both used in the morphological operations opening and closing. Opening is used for eliminating small regions and smoothing the contours of a region. Closing is used for eliminating small holes in regions and filling gaps in the contour. Opening is done by first doing a erosion and then a dilation. Closing is done by first doing a dilation and then a erosion.

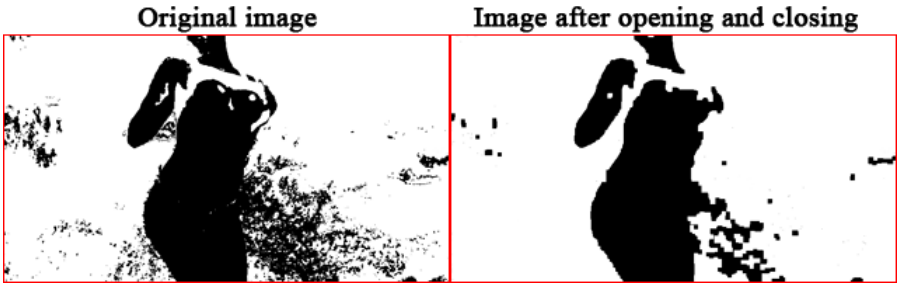


Figure 2.17. By combining opening and closing operations, a lot of the noise in an image can be removed, without destroying the shape of the region of interest.

2.6 Artificial neural networks

An artificial neural network (ANN) can be briefly described as the programming equivalent of a brain. It tries to reproduce the function of a brain by mimicking the layout. An ANN consists of several neurons that are interconnected. Each connection between the neurons have a specific weight which effect how much the output from the neuron will effect the input to the next neuron. Each neuron often also has a weight of its own called a bias term which determines the effect of the neuron itself. The information in an ANN is stored in its weights, training an ANN therefore requires the weights to be determined. There are several existing algorithms that deals with this process. Backpropagation is a common method that is used for feed forward networks which are the ones used in this project. Feed forward networks are divided into layers where each neuron has a connection to all neurons in the next layer.

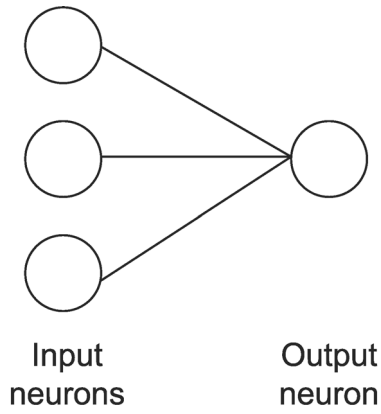


Figure 2.10. The simplest possible ANN with 3 input neurons and 1 out put neuron.

The simplest example is a network with a few inputs directly connected to an output neuron. The problem with such a simple ANN is that it is unable to be trained for a bit complex problems. The most common networks consists of three layers. An ANN can be trained to produce a pre-determined output when fed with the corresponding input. ANNs has a very broad usage area and can reproduce almost any linear function. They are especially useful when it is hard to describe the difference between input categories by simple rules. One example could be to classify people by age. For a human this is relatively simple. We can in most cases determine the persons age by a quick look at the face. Age can however be hard to describe in rules to a computer. This is where the ANN becomes useful. By training the ANN on many different examples it can often learn to distinguish between different categories if the training set is large enough.

2.6.1 Squashing function

To get a graded response from the network, a squashing function is introduced. Several different functions can be used but one common function is the sigmoid function is defined in eq 2.1 and is shown in figure 2.1.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \textit{(eq. 2.1)}$$

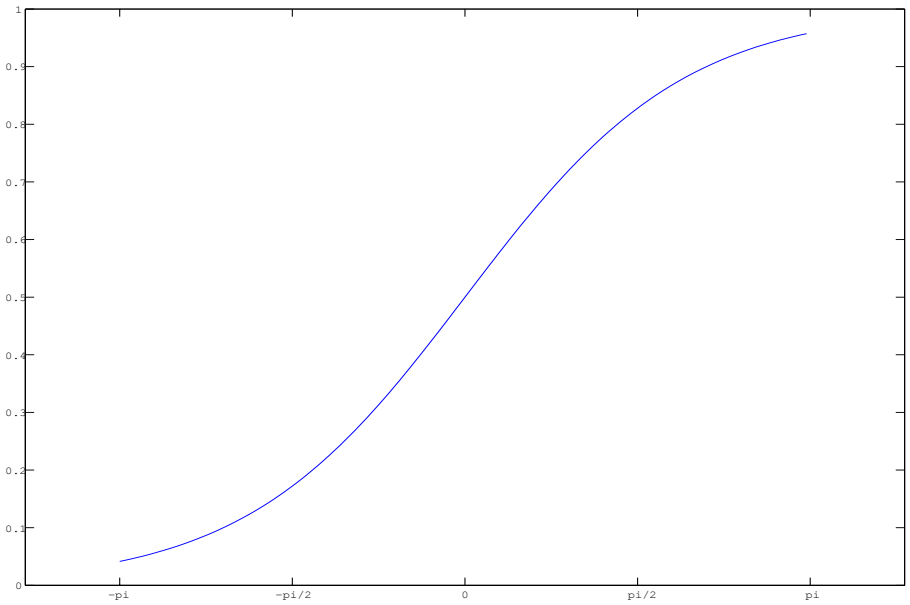


Figure 2.1. A sigmoid function, defined in eq. 2.1, which is often used as a squashing function in ANNs.

2.6.2 Backpropagation

Backpropagation calculates the error on each of the training pairs and adjusts the weights to fit the desired output. This is done in several epochs until the total error on the training set is small enough or when the error stops to decrease.

2.6.3 Overfitting

When overfitting occurs the performance on the training set increases while the performance for unseen data decreases. It can occur when the training set is to small in regard to the number of inputs or when the correlation between the inputs in the training set is high. Overfitting can often be avoided by using a large enough training set or by using cross-validation. There are also other techniques such as model selection, jittering, early stopping, weight decay and Bayesian learning that can be used to prevent overfitting. Model selection is related to the number of neurons and is discussed by [Moody, 1992] where he claim that more neurons could amplify noise in the target.

2.6.4 Hidden neurons

The number of hidden neurons in an ANN is often hard to choose and there exists no general rule of thumb. [Blum, 1992] suggests that the number of hidden neurons should be somewhere between the number of input Neurons and the number of output neurons whereas [Swingler, 1996] and [Berry and Linoff, 1997] says that you never require more than twice as many hidden neurons as you have inputs. [Lawrence et al, 1996, 1997] showed that it is difficult to reduce the training error to the global optimum even when using more weights than training cases. More weights can however make it easier for backpropagation to find the global optimum. Other guidelines apply if weight decay, Bayesian learning or early stopping is used and this is discussed by [Neal, 1996] and [Sarle, 1995]

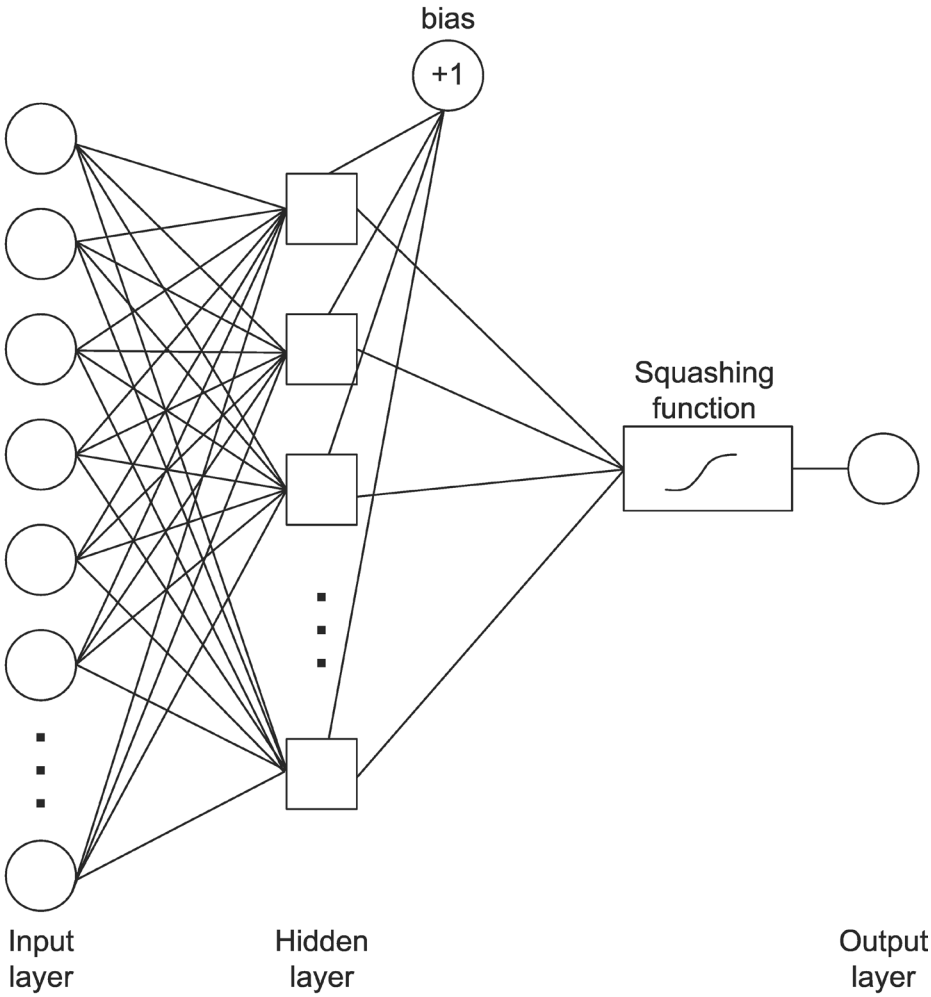


Figure 2.11. An ANN with one hidden layer and a squashing function, also known as activation function before the output neuron.

2.7 Face detection

A successful face detection algorithm should be able to detect faces regardless of orientation, facial expression, pose, structural components (facial hair, glasses), occlusion and image conditions. There are four basic types of face detection systems, Knowledge based methods, feature invariant approaches, template matching methods and appearance based methods.

2.7.1 Knowledge based methods

Describes a face from a human perspective with simple rules, e.g. a face has two symmetric eyes a nose and a mouth. The knowledge based method often starts with a simple rule that works on low resolution images such as that the face consists of a few brighter pixels grouped together. When areas of interest has been found the method continues to search for validation for the regions by scanning for face features such as eyebrows or eyes.

It is relatively easy to come up with a basic set of rules for describing a face. It can however be very difficult to get accurate results on images with cluttered backgrounds. It is hard to write rules that are good enough. A rule that is too general might detect too many false positives while a rule that is to precise or detailed will discard many faces. It is also hard to write rules that applies to all the different poses a face can have with different degrees of occlusion.

2.7.2 Feature based method

While knowledge based methods searches for face candidates first, feature based methods do the opposite and search for face features first. It then tries to group these features together by using random graph search. This method is invariant to pose or orientation but can give poor results with feature rich backgrounds or when the image has poor illumination or contains other image noise.

2.7.3 Template matching methods

Template matching is a relatively simple approach. The template is often hand made or calculated from a data set and is often based on the relativity between the brightness of different regions in the image. An example could be that the eyes often are darker than their surrounding regions. Template matching is not very good at handling different poses and often requires more then one template to give satisfying results. Template matching is more suited to detect objects of a certain type with the same pose and orientation than it is at detecting something as variation-rich as human faces.

2.7.4 Appearance based methods

Appearance based methods involves a classifier that is trained on a set of training data. The training data usually consist of both positive and negative samples. Common methods include ANNs, principal component analysis (PCA) support vector machines (SVM), naïve Bayes classifier, AdaBoost and many more. Many of the different methods take advantage of preprocessing of the images. Histogram equalization is one of the preprocessing methods that are often used. The performance of the above mentioned methods can often be improved by running the methods on training data and adding all false detections into the training set.
Most methods use some sort of scanning algorithm where the image is scaled to different sizes and then scanned. The speed of the algorithm relies heavily upon the speed of the classifier.

[Viola and Jones 2004] suggested a new method that involves many simple classifiers. They start with a classifier that have very simple conditions and then move on to more advanced classifiers if the image meets the conditions. The classifiers was sorted using AdaBoost.

2.7.4.1 Bootstrapping

It is sometimes hard to acquire a good collection of negative images. This can be avoided according to [Rowley et al, 1998] by using Bootstrapping. Bootstrapping is a simple procedure that adds the false detections from the training set to the training set. The false detections can be drastically decreased by using this method and often helps to produce a better training set.

3 Method

The method section describes the different methods we intended to evaluate, in order to classify images as containing nudity or not.

3.1 Gaussian mixture models

We have trained several different Gaussian mixture models (GMMs) by varying the number of components for both the RGB and IHLS color space. For each of the color spaces, we have trained 11 GMMs with 2, 3, 4, 5, 6, 7, 8, 10, 12, 14 and 16 components. On the following pages, scatter diagrams of the skin pixels and the GMM trained for the RGB color space using 2 components, in three different angles are shown.

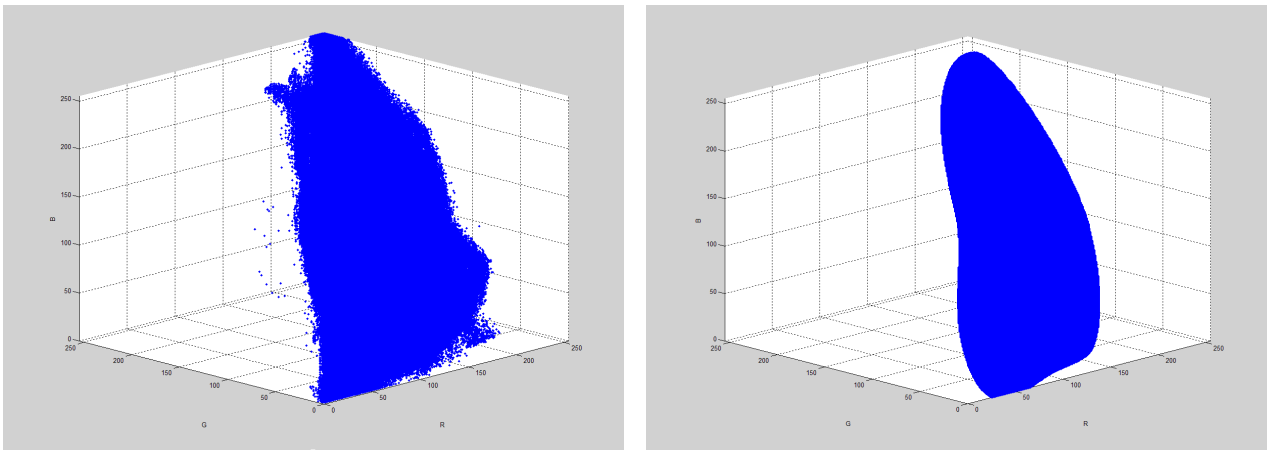


Figure 3.1. The left scatter diagram shows the skin pixels from the training data used in this project. The right scatter diagram shows the output from a GMM trained with two components on the same skin pixel data. The RGB color space is used.

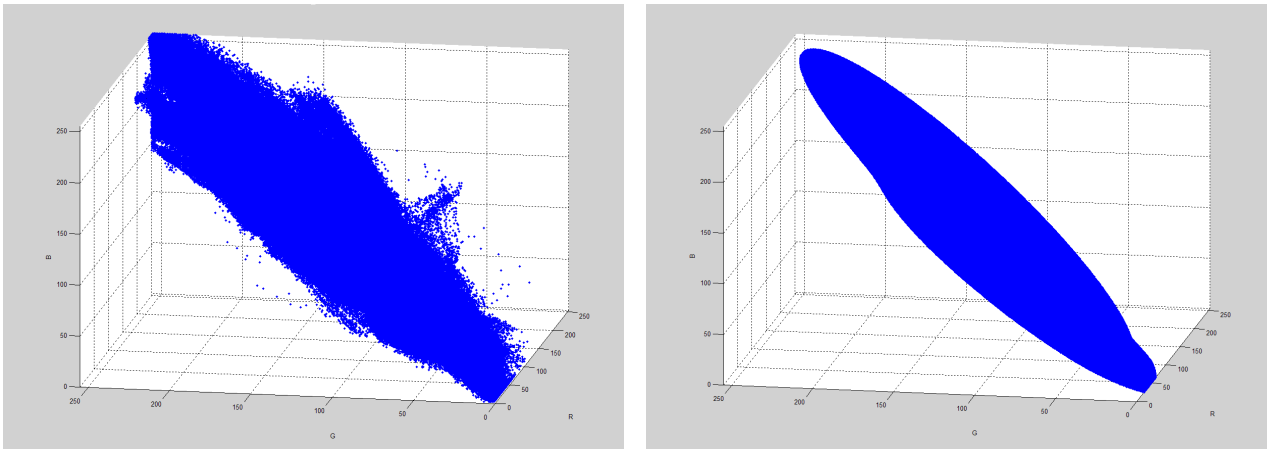


Figure 3.2. The figure shows the same as figure 3.1, but from an other viewing angle. Please note the cylindrical shape.

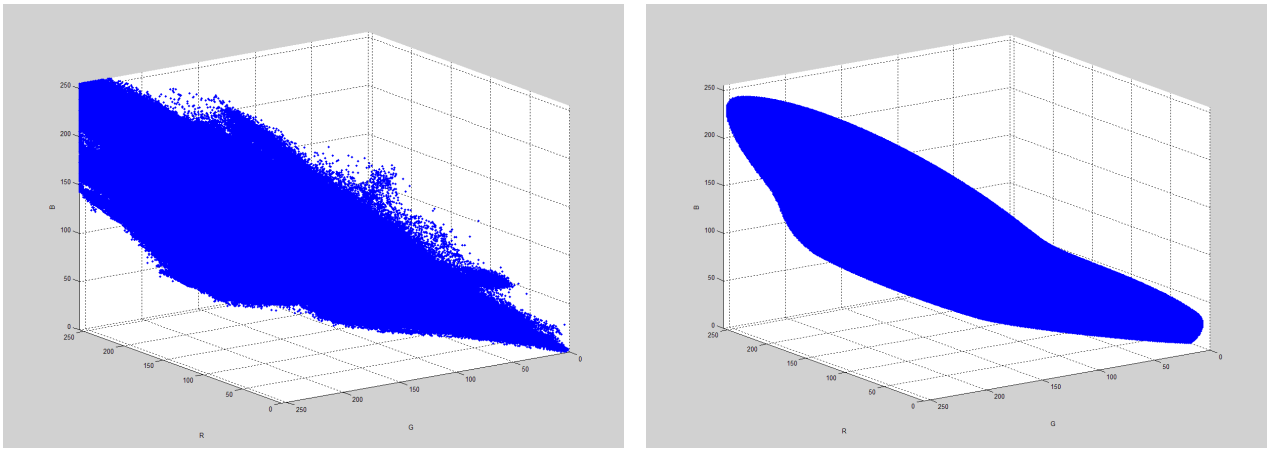


Figure 3.3. The figure shows the same as figure 3.1 from yet an other viewing angle.

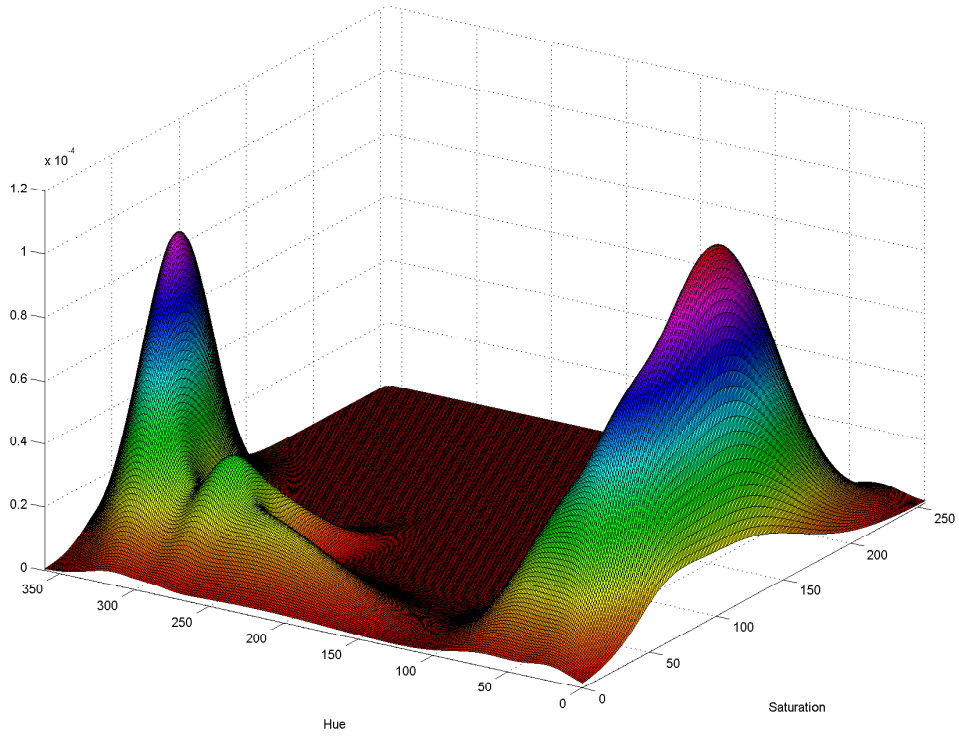


Figure 3.4. A surface plot of a GMM with five components, showing hue and saturation on the X-axis and Y-axis and the probability of skin on the Z-axis.

The formula for calculating the probability that a pixel is a skin-pixel requires a lot of processor power and images today usually have a high resolution. That makes it impossible to use GMMs if the computation can not take more than a couple of seconds. One way to solve this problem is to use a look-up table. The skin-probability for each pixel in the RGB color space is computed and saved in a look-up table. The look-up table for RGB would require a matrix with the size 256 x 256 x 256, thus memory usage will not be a problem on a standard desktop computer.

3.2 Skin segmentation using ANN

Instead of using thresholds for detecting skin pixels in an image it is possible to use an artificial neural network. The ANN will in theory give a better result than using only thresholds for each channel as the network also takes into account the relationship between the channels. This ANN was trained using manually segmented images. The distribution between skin and non skin pixel where close to 50-50 so that none of the categories would be favoured during training of the ANN. The ANN for the skin pixel detection could be made very small with only three input neurons; one for each channel in the RGB color space, a couple of hidden neurons and a single output neuron. For simplicity, all values was divided by 255 before entering them as input to the ANN. The training set contained over 46 million pixels.

3.3 Skin segmentation using 9-pixel ANN

An artificial neural network could also be trained to detect if larger areas contain skin, for example by using a region of 3 x 3 pixels, which would involve 9 pixels rather than looking at a single one. Of course, even bigger regions could be tested. This would in theory mean that the network would take the regions texture and not only the color of the pixel into account. It might also detect pixels that would not be classified as skin by the single skin pixel network as this network would take into account the neighboring pixels. These are all just theories that will have to be confirmed with tests.

3.4 Skin segmentation using multiple-pixel output ANN

As we had already successfully tested to detect skin using both a single pixel and a region of pixels as input to an ANN we decided to test it was possible to use a bigger ANN that take the whole image as input and output the segmented image. As this kind of ANN would be enormous if the images was not downscaled first we decided to test it with the size and width of only 30 pixels. The output would be of the same size as the input and would be on the form of 0-1 where a bigger number means more likely to be skin. The already segmented images used for training the pixel network was used to train the ANN. The number of hidden neurons was varied from 100 to 900.

Figure 3.5. To the left is the input to the multiple-pixel output ANN, and to the right is the supposed output from the ANN.

3.5 ANN trained on non-segmented images

We decided to try to use neural networks in order to both detect skin and to detect inappropriate images using the ANN as the only classifier, without any form of segmentation or processing. Early testing showed that the network reached over 90 % correct matches when trained with a large set of sexually explicit images and non sexually explicit images. The whole set contained almost 30 000 images where the distribution of sexual and others where about 50-50. The distribution between the images are important for the network to train correctly. Overpopulation of one category might make the network favor that category and adapt to make those images count for more. It can however in some cases be preferable to have more negative samples in order to reduce false detections. All images where scaled down to 30x40 pixels before training with them. The pictures where of very different quality and origin and no preprocessing was done before they where used other then resizing them. The pictures was not each checked manually before they where used so there might

have been some pictures that belonged to the wrong category. Thanks to the very large data set this could be neglected.

3.6 Feature extraction

A lot of different features are extracted from each image. All features have been chosen based on what we believe might matter when recognizing images containing nudity. Some might matter more then other and some might not be of any help at all. Which ones that matter the most can be checked by randomly switching one input to the final network at a time and calculating the error. The one with the largest error should be the one that matters the most if the input does not depend too much on each other.

3.6.1 Largest skin-region

The size of the largest skin region could be of importance. It is calculated in percentage of the whole image. In case there are many small regions detected as skin this feature would help in the classification.

$$largest\ skin\ region = \frac{skin\ pixels\ in\ region}{total\ number\ of\ \ pixels} \qquad (eq.\ 3.1)$$

3.6.2 Number of faces

The number of faces in an image could give useful information about the number of persons in an image.

$$number\ of\ \ faces = \frac{1}{number\ of\ \ faces} \qquad (eq.\ 3.2)$$

3.6.3 Output from ANN

The ANN is good at detecting inappropriate images and might even be the most important feature. The ANN however make some false detections and will probably benefit from having more features to rely on. Furthermore it is also impossible to know exactly what the ANN is trained to detect. This depends heavily upon the training data. If the training data is skewed in any way the ANN might train to react on something entirely different then supposed to.

3.6.4 Face percentage

Images containing head shots also contain a high percentage of skin. It is possible to classify these images as non undesired if the detected skin region also contains a face. Images containing many faces but few other skin regions can also be classified as non undesired with this technique.

$$face\ percentage = \frac{face\ pixels}{total\ number\ of\ \ pixels} \qquad (eq.\ 3.3)$$

3.6.5 Number of regions

The number of detected skin regions could indicate whether or not the image is undesired. The algorithm sort out the regions that are under a certain size as these are regarded as noise and only counts the larger ones.

3.6.6 Skin percentage in image

Much skin indicates that the image might contain nudity. Images without any skin at all could most often be discarded directly without further processing.

$$skin\ percentage = \frac{skin\ pixels}{total\ number\ of\ \ pixels} \qquad (eq.\ 3.4)$$

3.6.7 Largest skin-region's position

The position of the largest skin-region might make a difference. Images containing nudity seems to have the target at the center of the image. This area might then contain a lot of skin, whereas a faulty detected walls etc often is located closer to the proximity of the image.

$$largest\ skin\text{-}region's\ position = \frac{mean\ position[x, y]}{image\ size[width, height]} \qquad (eq.\ 3.5)$$

3.7 Face detection

An artificial neural network forms the basis in our face detection system. It evaluates images of the size of 24x24 pixels. The neural network can be trained both with and without color. In order to detects faces of different sizes the original image needs to be scaled. We use a method were we first scale down the original image to a maximum height or width of 24 pixels, the detector is then moved across the image. If a face is detected its position is stored. The image is then scaled with a factor of 1.25 and the detector is moved across the image scanning for faces where no faces has previously been detected. Many other systems use some sort of merging for detected faces to increase the accuracy of the detections if detected faces overlap. The algorithm then detects which one has the best fit and remove the rest of the overlapping detections.

The ANN is an advanced classifier and requires quite a lot of computational power. It can take several seconds to scan all possible locations of an image of size 400x300 pixels. In order to speed up the process we made attempts with a preclassifier that counts the number of skin pixels before the sector is passed along to the Neural Network. This proved to be able to speed up the algorithm considerably although this method only work on color images. The negative effect is that images that have distorted colors or is affected by background light or other image noise will cause the preclassifier to sort out potential faces.

The algorithm is not fast enough to run in real time on a desktop computer, each frame take around 5 seconds to compute. Our early tests showed good results at detecting faces but also a very high false detection rate. The training set consisted of just over 1000 faces and 1000 non-faces. The non-faces should be increased in order to prevent false detections. By using bootstrapping and adding false detections to the training set the results where improved.

3.8 Feature network

The feature network takes all the features as normalized input with values between 0-1. We have chosen to invert the input if the input is larger than 1, e.g. the number of faces feature is defined in equation 3.2. The ANN has eight input neurons and one output neuron. The number of hidden neurons that will be used in the final version will be acquired through testing and analysis of the results. Features are extracted from a set of images and this set is then divided into training and validation where training consists of 70% or approximately 3300 images.

3.9 Evaluation

Evaluation is one of the key element when working with different methods. To be able to compare different methods and get some comparable data, it is important to have a good measurement. The method we use to compare the different techniques is called receiver operating characteristics graphs, or ROC graphs.

3.9.1 ROC

Receiver operating characteristics (ROC) graphs are a useful way of visualizing the performance of different classifiers. Each output from the classifier has four possible outcomes. True positive is when the classifier correctly classified a positive instance as positive. If the instance is positive and classified as negative it is counted as false negative. If a negative instance is classified as negative it is counted as true negative. A negative instance classified as positive is counted as false positive. When given a set of test instances a two-by-two confusion matrix is formed.

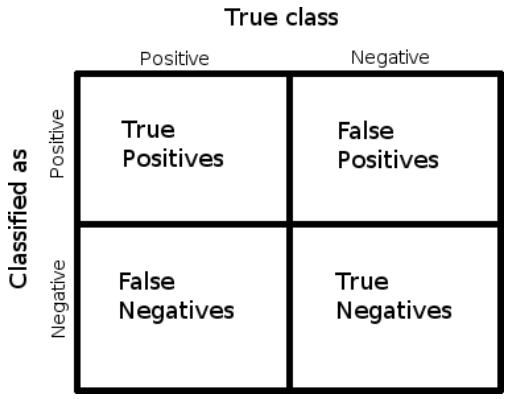


Figure 3.6. A confusion matrix.

From the confusion matrix many different metrics can be calculated. Among the two most important ones are true positive rate (tp rate), given in equation 3.6, and false positive rate (fp rate), given in equation 3.7.

$$tp\,rate = \frac{true\ positives}{positives} \qquad \textit{(eq. 3.6)}$$

$$fp\,rate = \frac{false\ positives}{negatives} \qquad \textit{(eq. 3.7)}$$

An ROC graph is a two-dimensional graph with tp rate on the Y-axis and fp rate on the X-axis. This results in an easy way of visualizing the trade-off between correctly classified instances and incorrectly classified instances. A perfect classifier would result in a point at the top left in the graph. A classifier that ends up on the minor diagonal line where x = y can be labeled as guessing as it has the same tp rate as fp rate. Points that are close to x = y should also be investigated as they might just be a lucky guess from the classifier. A classifier that end up in the lower right part of the graph is usually very bad as it has a high fp rate and low tp rate. This would mean that negating the classifier would yield better results; thus ending up in the top left part of the graph.

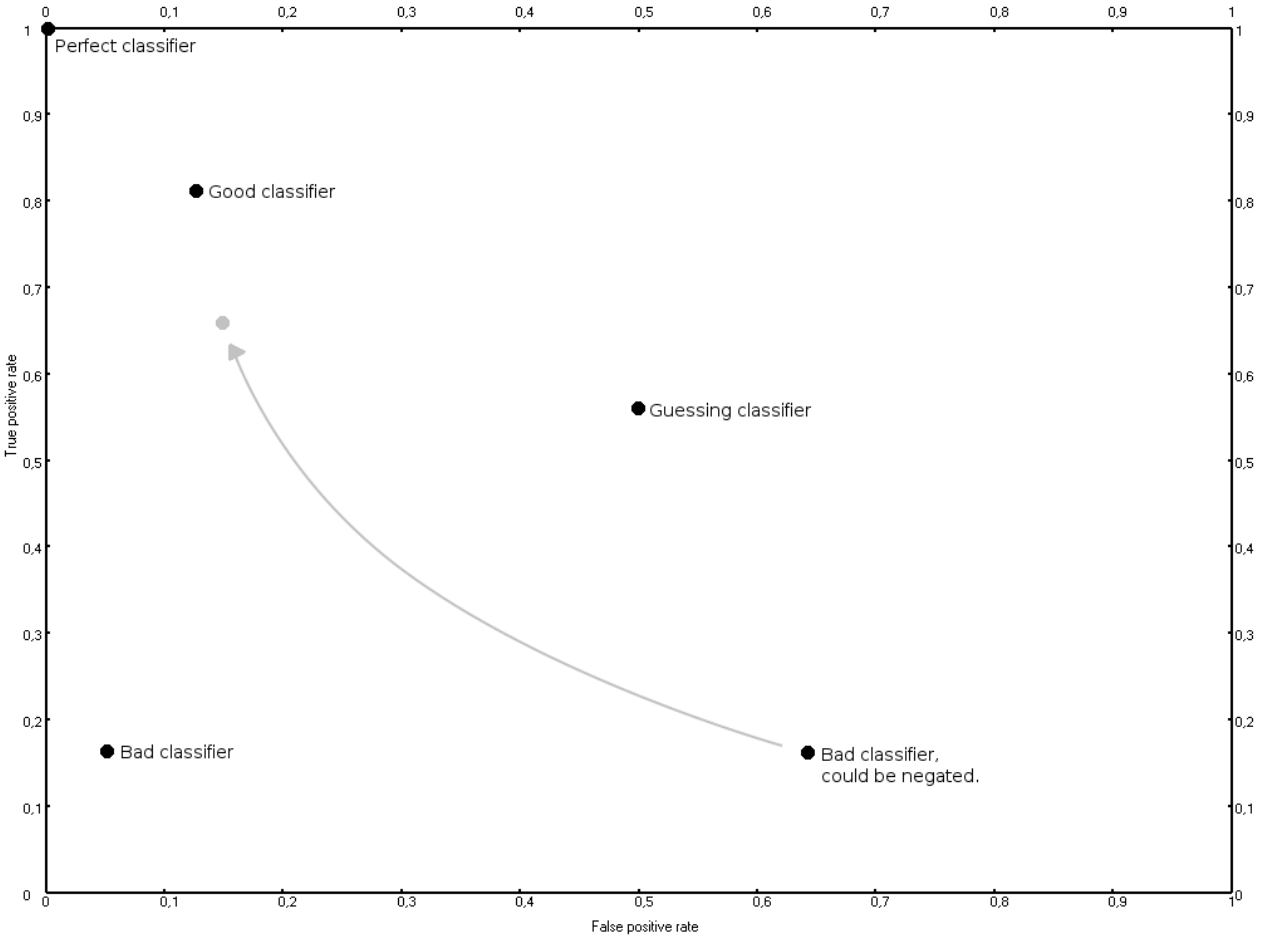


Figure . The figure above shows an ROC-graph. The data from the confusion matrix is calculated into false positive rate (X-axis) and true positive rate (Y-axis).

In order to evaluate and compare the output from the algorithms an evaluation tool was built. It simply measures the difference between a manually segmented image and the segmented image from the algorithms. The tools makes it much easier to see the difference between the algorithms and also different settings. The evaluation can be done on a single image or on a set of images. The result is displayed using an ROC graph for each classifier. Each graph contains multiple points where each point represent a specific threshold. The evaluation process requires a starting threshold, a step and a stopping threshold. With this technique it is easy to see witch threshold that has the best performance. Both the GMMs and the Neural network for pixels output values in the range of 0 to 1. The GMMs however in general require a lower threshold than the ANN.

3.9.2 Modified evaluation using ROC

An alternative way to evaluate the skin detection methods is to use the tp rate (Y-axis) from the normal ROC graph but exchange the fp rate (X-axis) with the method's volume in the color space that is used. The volume is normalized to values between 0 and 1 so the choice of color space does not affect the evaluation. Just like normal ROC graphs good results are those in the upper-left corner. Skin color is not a unique color so things that are not skin but have the same color will be classified as false positives. A low volume and a high tp rate means that the method is both accurate and it will not have many false positives, as it occupies little space in the volume of the color space. This evaluation method will not replace the normal ROC method but it will work as a complement.

4 Results

This section contains the results for the methods that has been evaluated, and are described in the method section. The overall results are good and shows that most of the individual parts performs good and can be used as an indicator for detecting nudity in images.

4.1 Skin detection ANN

The pixel network reached very satisfying results with a true positive rate of 80% with 13% false positive rate. The network reaches 90% true positive rate with a false positive rate of 23%. These result should be considered as good as the only input is the R, G and B values for a single pixel. The method is very fast and can easily process every frame from a webcam in real-time on a standard computer.

4.2 9-pixel skin detection ANN

The 9-pixel ANN used the same foundation as the ANN for single pixels, but instead of 3 inputs, 27 was used, three for each of the 9 pixels. This Network was trained using the same data as the single pixel network. Our early evaluation showed that this new network performed similar to the single pixel network. It requires more processing power and is noticeably slower.

The network performs almost identical to the network for single pixels when trained with the same data. It is however slower and the single pixel network is therefor preferred.

4.3 Multiple-pixel output ANN

Instead of using only a single pixel or a few as input to an ANN we decided to experiment using a whole image as input to see if we could get the skin-pixels as output. We used the images that we had already segmented and used them to train the network. During the first experimental trainings the network converged on the same solution all the time, namely to always mark the pixels in the center of the image as skin no matter what as this gave a good result on the training set since almost all of them had skin only in this region. To prevent this behavior more training images containing no skin was added to the training set.

The ANN used had 900 input, 900 hidden and 900 output neurons which takes an image of 30x30 pixels as input. The output is a 30x30 matrix with the likelihood for each point to be skin.
The results from the ANN was not satisfying. After the images containing non skin had been added to the training set the network instead marked all pixels as non skin as this gave a fairly good result on the training set with a true positive rate of 65%. This however proved to be the result of marking everything as not skin as 65% of the pixels in the training set was just that, non skin. Some images was now removed from the training set and the network was retrained in order to decrease the amount of non skin in the training set. The network now reverted back to its previous behavior and got stuck on a local optimum again. Something clearly has to be changed for this network to be more effective in training.

4.4 Image based network

The image based network performs really well but can on occasion give strange results on certain images. On the training set the tp rate is close to 90% with an fp rate of only 1.1%. In order to verify these results, the network needs to be tested on a large verification set. The performance on the training set as well as the speed of the image based network is very promising and a standard computer should be able to process several images per second.

4.5 Face detection

The face detection ANN implemented detects faces in images, but it classifies too much false objects as faces. It also misses some faces that are too small, tilted or taken from the side. This is probably due to the training data. By using more faces, as well as non-faces as training data, the results would probably be better and the risk of overfitting would decrease.

4.6 Feature Network

The feature network was trained with over 3 000 training cases. This is a fairly large amount on a ANN that only consists of 13 neurons, 8 input, 4 hidden and 1 output. After extensive training the network did not reach satisfying results. It never went under a medium error of 0.1 for the training data and performed very bad when tested on our validation data that consisted of 1400 cases. Out of 813 images containing nudity only 211 was marked as undesired when a threshold of 0.8 was used. It performed better on the images that did not contain nudity only classifying 31 out of 617 as undesired.

4.7 Evaluation of skin segmentation

Evaluation is done for the different methods that detects skin pixels. Each method is evaluated by comparing its resulting skin map to a manually segmented one, used as reference. The evaluation set consisted of 238 images and their respective manually segmented reference. None of the images in the evaluation set was used during training. The results of the evaluation are presented in ROC-graphs.

4.7.1 ROC

As seen in figure 4.1, the best GMM for the RGB color space, is the one using 2 components. In figure 4.2, the results are harder to distinguish. The GMMs for IHLS performs better on average than the ones using RGB. The best one is using 5 components. Figure 4.3 shows the performance of the pixel network, both with and without the use of a texture filter. As seen in the graph the texture filter in its current configuration removes too many true positives, but is slightly better at higher threshold values.

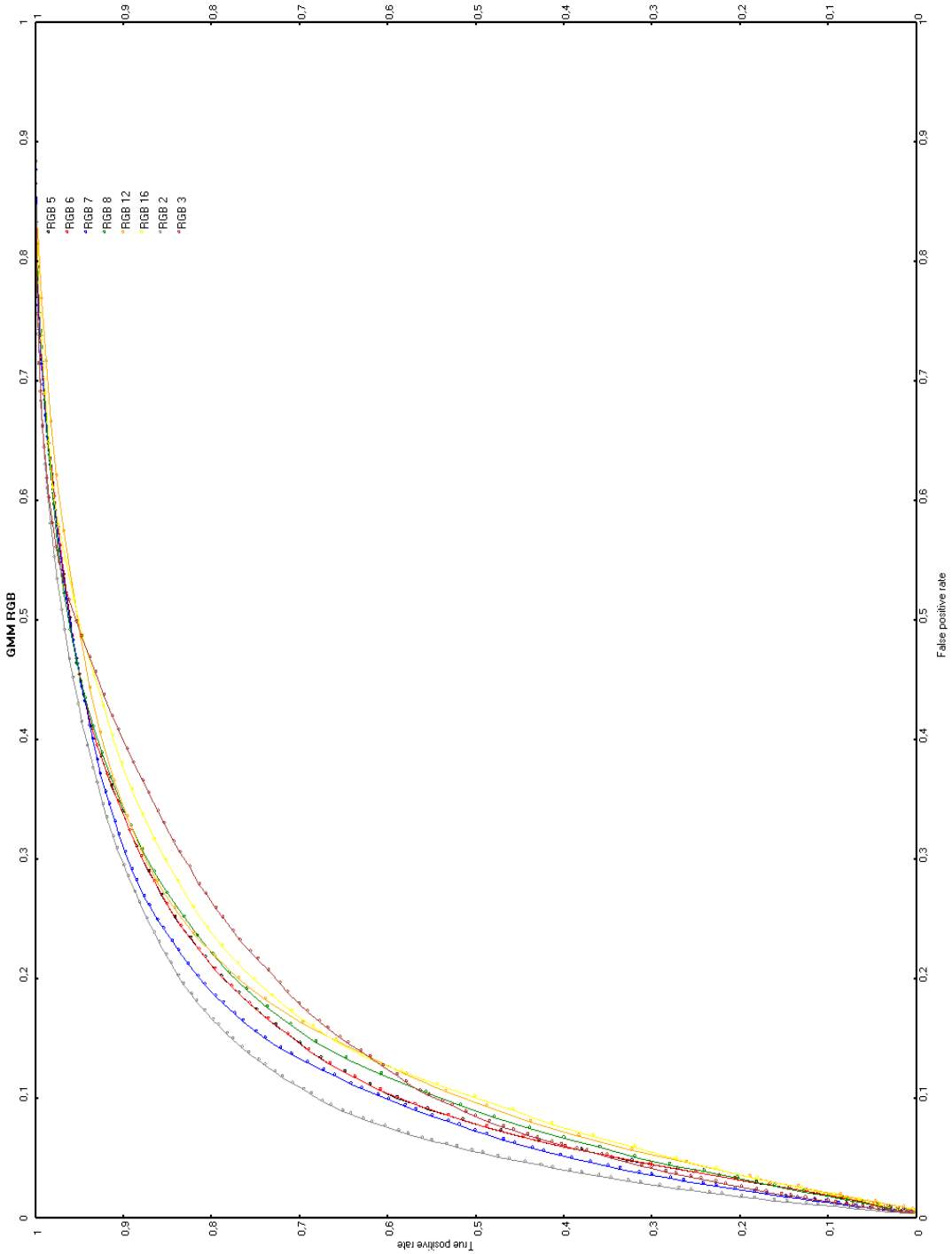


Figure 4.1. ROC-graph showing GMMs with different number of components, for RGB.

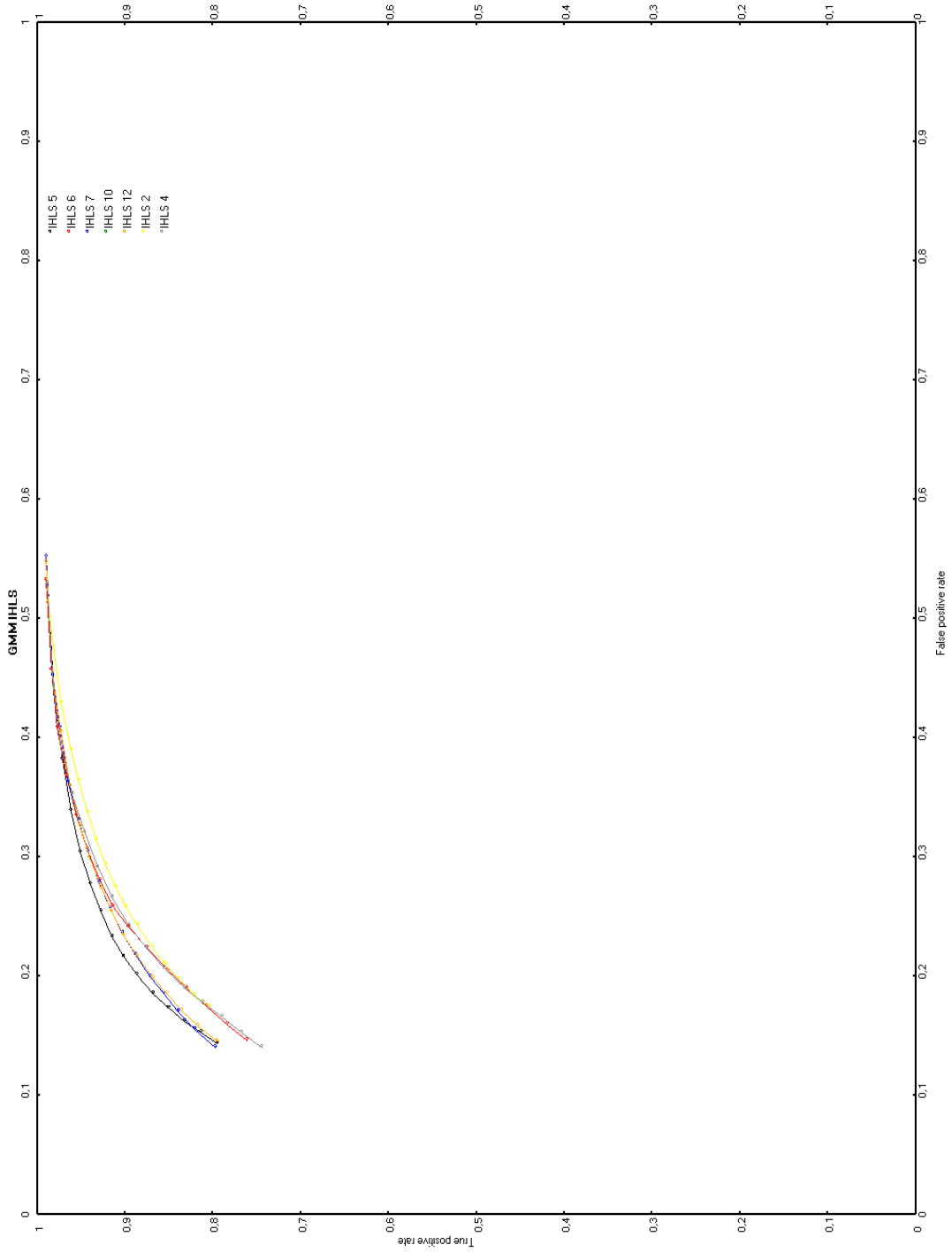


Figure 4.2. ROC-graph showing GMMs with different number of components, for IHLS.

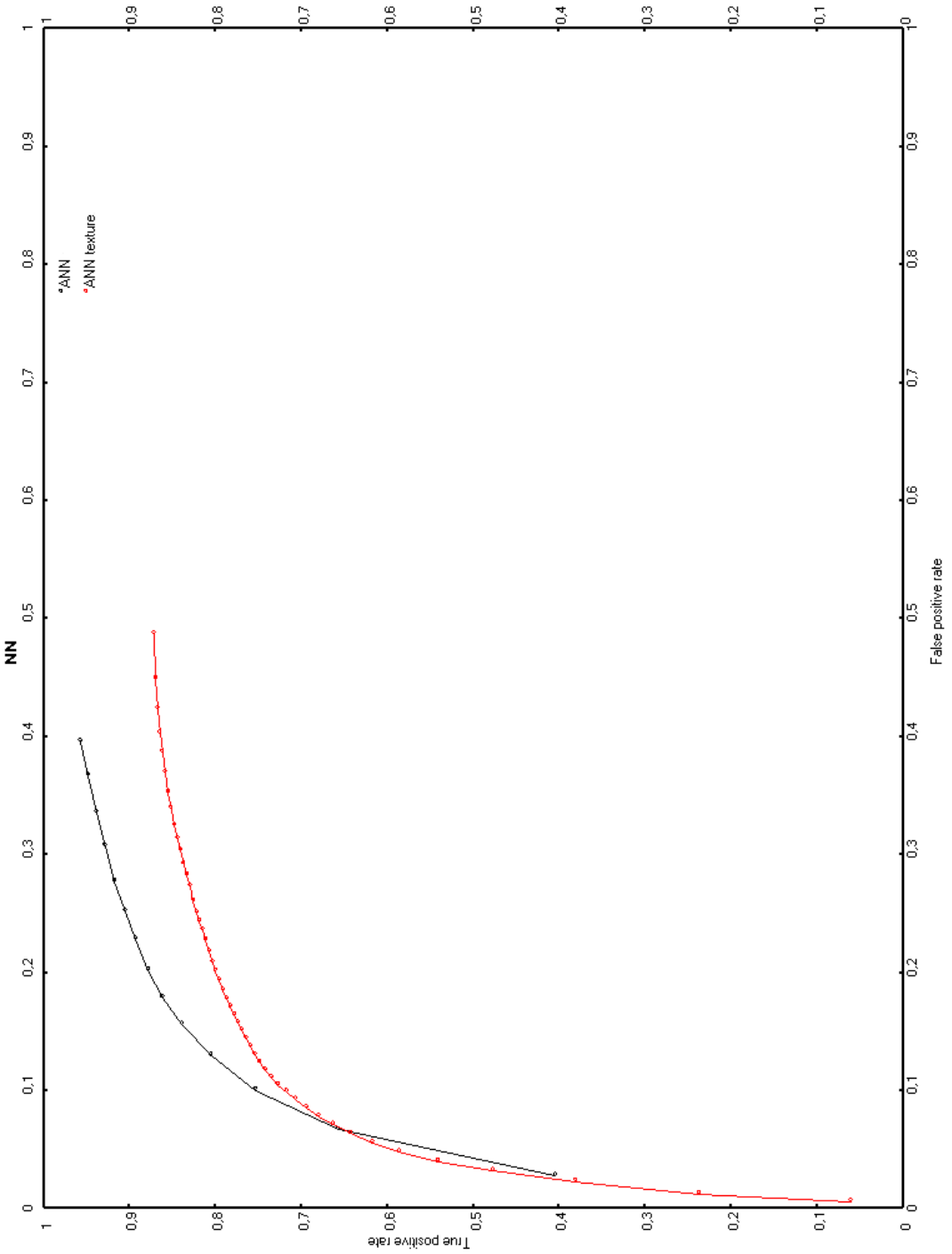


Figure 4.3. ROC-graph showing artificial neural network with and without texture filter.

4.8 Comparison of methods

In figure 4.4, the best alternatives for the different methods are shown. The artificial neural network for pixels, denoted as ANN in the figure, is slightly better than the best GMMs for both RGB and IHLS.

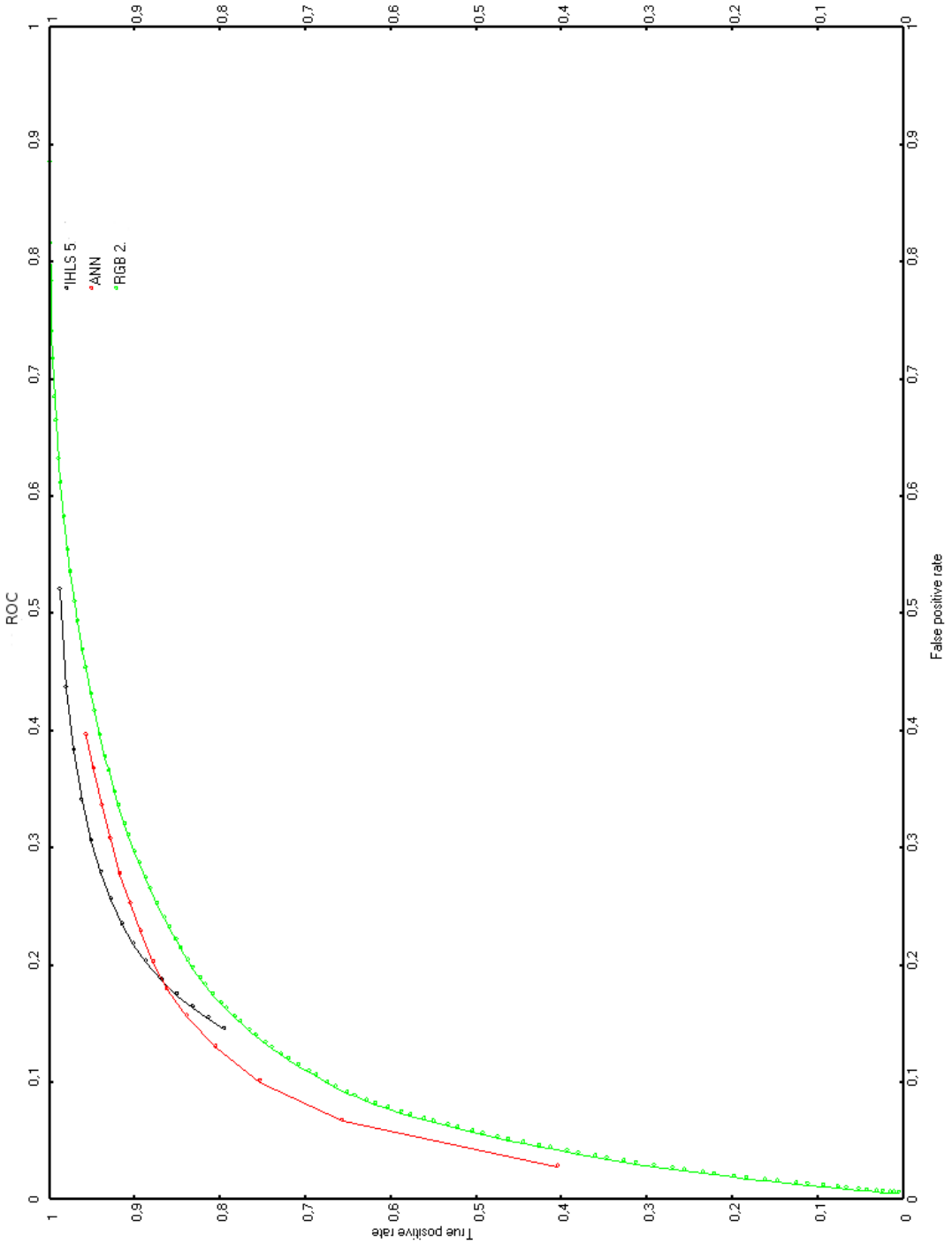


Figure 4.4. ROC-graphs over the best components for the different segmentation methods.

4.9 Modified evaluation using ROC

The results we got from the modified evaluation using ROC shows that the different methods used do not differ much. A larger difference between the methods would prove that one method is better than the other but when the difference is so small it could be a matter of variance in the choice of data used in the evaluation.

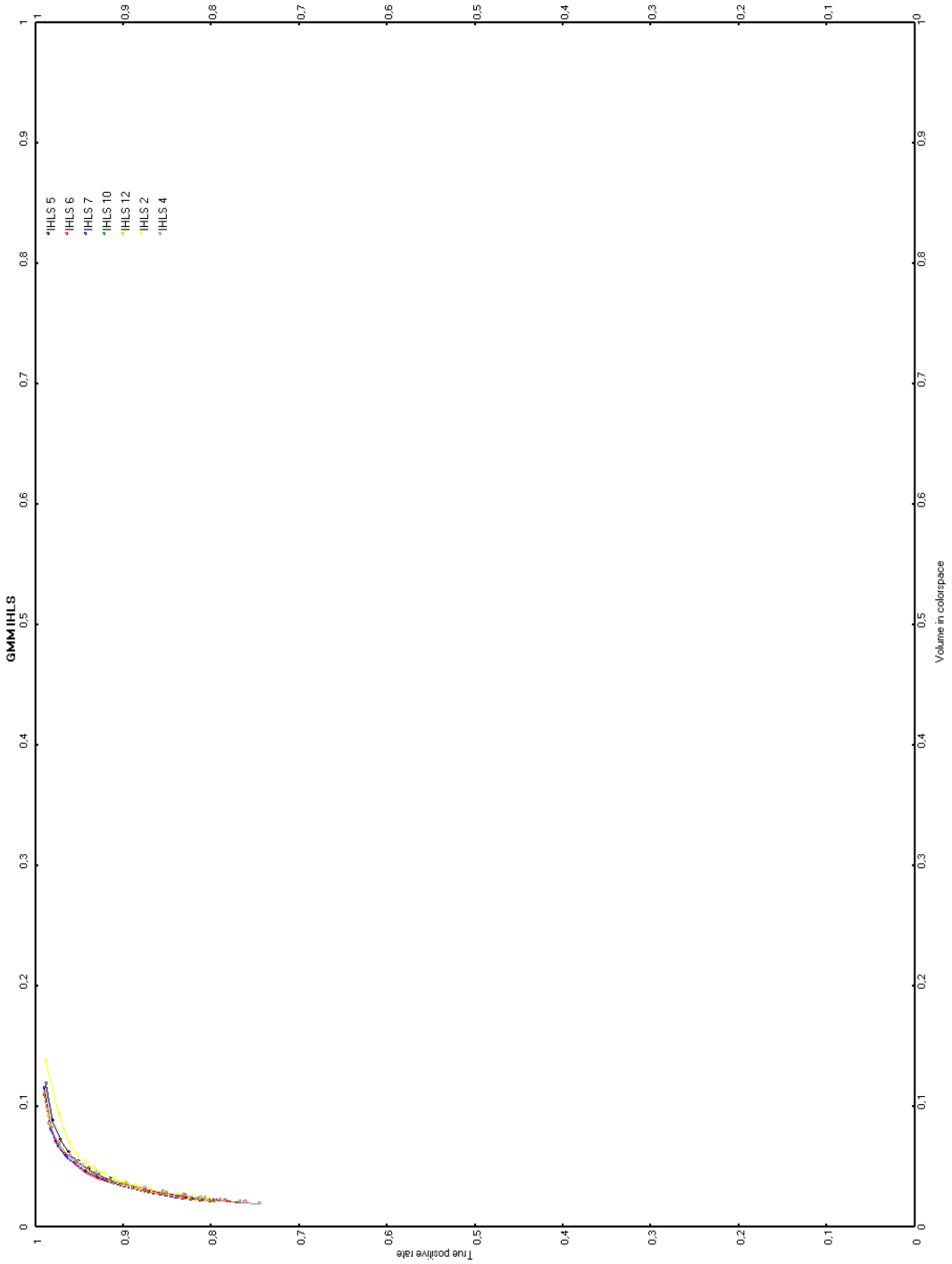


Figure 4.5. ROC-graph using the modified evaluation of GMMs for IHLS.

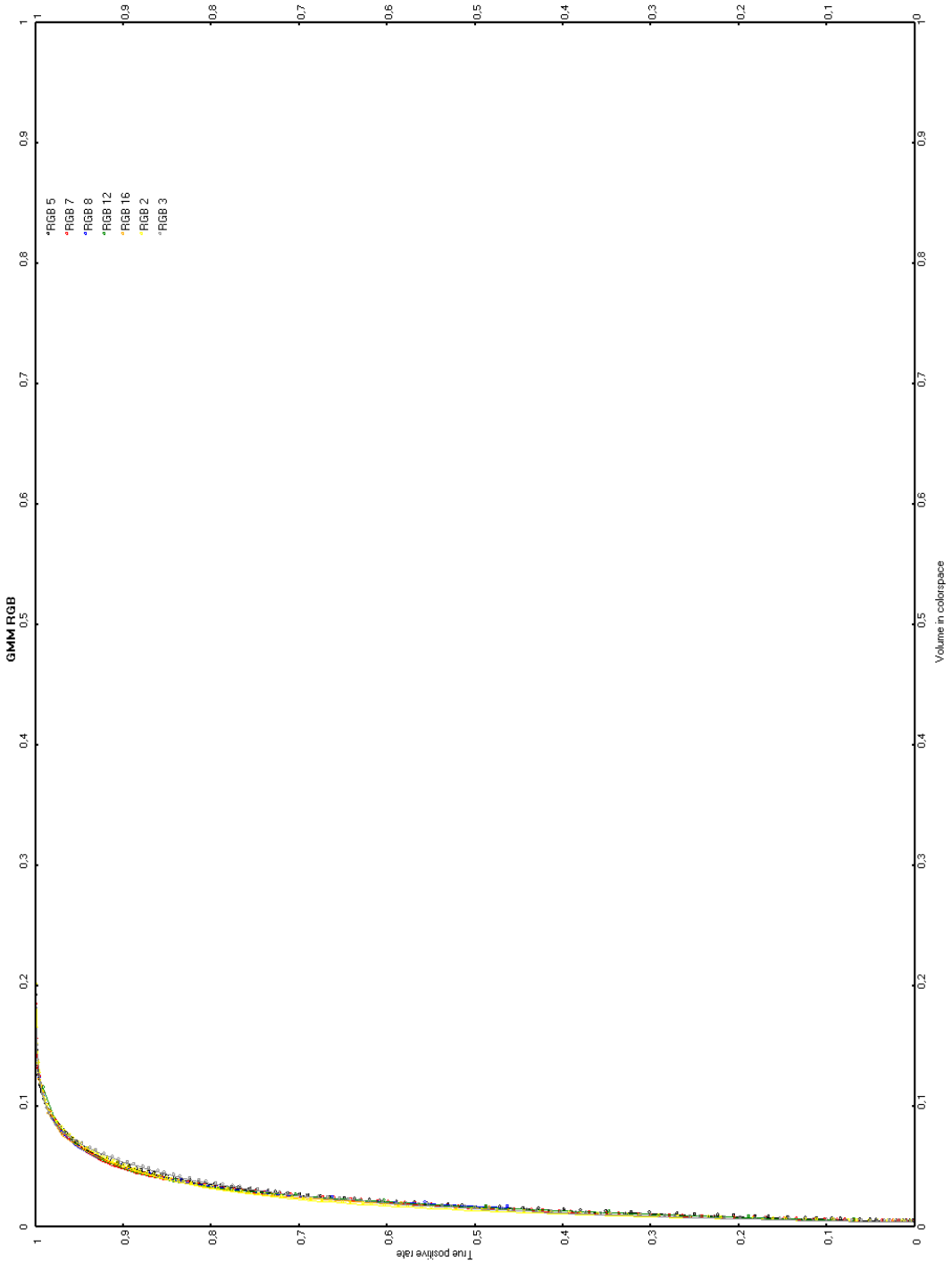


Figure 4.6. ROC-graph showing the modified evaluation with GMMs for IHLS

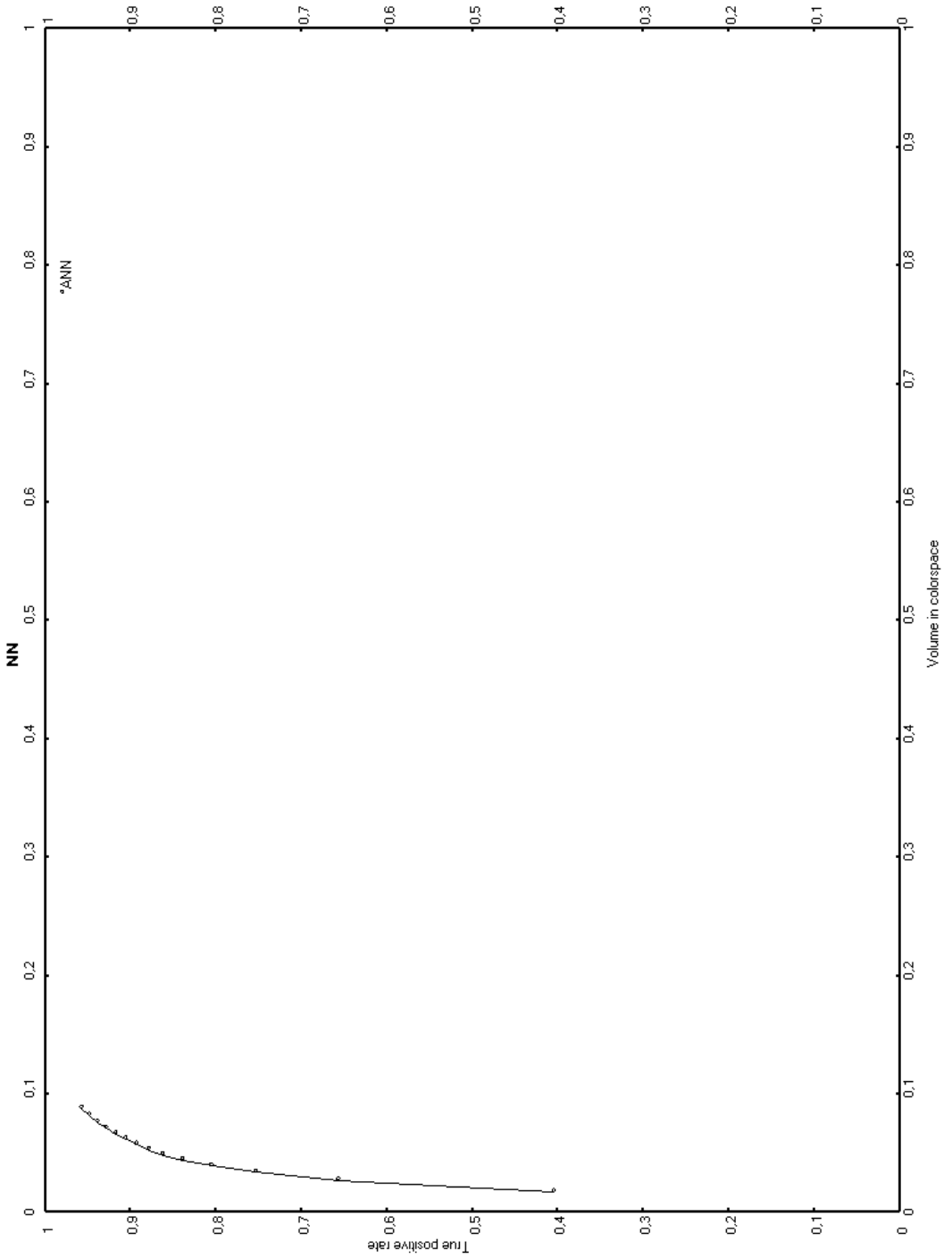


Figure 4.7. The ROC-graph for an artificial neural network, with the modified evaluation.

5 Discussion

A lot of image processing applications are used in controlled environments. This project was about classifying all possible images, used as input to the system, as containing nudity or not. All images are from uncontrolled environments, meaning we can not assume good conditions, such as good color distribution or steady lighting, and we do not have any information about the image before we receive it. The only common factor in the images that we could come up with was that it would contain skin, so removing everything except skin was one of the steps in our method. Skin colors do vary a lot, and it is not a unique color. It is therefore hard to remove everything except skin. The next step is to use the segmented image and extract features from it, features that could be used to classify the image. We could not see any clear patterns in segmented unsuitable images so that is why we came up with features, which we thought an ANN (or another classifier) could find patterns in.

5.1 Images containing nudity

The line between a suitable image and an unsuitable image is not always clear. A perfectly suitable image in one country/culture may not be very suitable in other countries/cultures. While working with this project we had to manually classify a big homemade image collection in suitable or unsuitable images. We are three persons, all males, around the same age and grown up in the same community but we would still not always agree in which category some pictures should belong. This shows that classifying images is very subjective and this is one of the reasons why this is such a difficult task to translate into computer algorithms.

5.2 Speed

The speed of the final algorithm is of different importance depending on the situation where it is supposed to be used. Different techniques require varied computational time and some that are rather slow in comparison might not be as useful as others that require much less calculations. These properties must also be considered when deciding which techniques to use. Many parts of the algorithm; such as the face detection, could be adapted to use multiple processors. This would most certainly improve performance as many new computers has two or more cores. Image analysis is in general very easy to adapt to image processing as many techniques allows the image two be divided into pieces before being processed.

When dealing with ANNs the speed of the ANN is proportional to the amount of neurons. Less neurons means a faster ANN. The number of input neurons is often relatively easy to decide upon. When dealing with images a simple rule is that if a human can distinguish between the types at a certain resolution the ANN should be able to handle it as well. The same applies to whether or not to use color in the input. There re no real guidelines on how many hidden neurons that are needed for any arbitrary problem so guessing seems to be the most commonly used method today. None of the reports that we have read gives an explanation other then that they used their experience when deciding the amount of hidden neurons. One thing that is clear is however that a more complex problem requires more hidden neurons while a simple problem might suffice with just a few.

5.3 Artificial neural network

Correct training data is very important for an ANN. There are many examples where incorrect data has been used resulting in a wrongly trained ANN. One of the most famous one involves a military experiment that tried to teach an ANN to detect camouflaged tanks in aerial pictures of forests. All went well until it was discovered that all training images without tanks was taken on a cloudy day and all training pictures that did contain tanks had been taken on a sunny day. This led to an ANN that could distinguish between cloudy and sunny days instead of tanks or no tanks.

5.3.1 Skin segmentation using ANN

The ANN that detects skin-pixels is very easy to train due to its many training cases and very few weights. We experimented with using from one to a hundred hidden neurons and everything over six seemed to be unnecessary. The training set consisted of over 46 million pixels so the chance of overfitting was very small when only a total of 31 weights where used. It was not surprising to see that the ANN for skin-pixel segmentation was the best performing out of the methods that we tested as it is very good at adapting to its training cases. It also takes into account the number of skin and non skin-pixels for each unique color combination that exists in the training set.

This method will never give a perfect result, but a tp rate over 80% with an fp rate lower than 10% is actually higher than we anticipated. The algorithm depends heavily on lightning conditions and does not work on black and white images. It will give poor results on images that contains objects with skin like colors such as sand, wood or light hair. The method is however fairly fast even for high resolution images and can be run in real-time on a stream of images aquired from a webcam, with a resolution of 320x240 pixels at 30 frames per second, on most modern computer desktops.

5.3.2 Skin segmentation using 9-pixel ANN

The 9-network used the same training set as the single pixel network and had similar performance. It is a bit slower than the pixel based network since the network has many more weights. Additional training cases should be added to the training set in order to make it discard texture rich areas such as hair and grass. The pixels in the training set that is not skin consists solely of whole images, not containing any skin or other human features at all. This could be a disadvantage since the network does not have a chance to train on hair or clothes. Especially hair was what we where hoping to discard using the 9-network but tests did not show any satisfying results.

5.3.3 Image based network

An ANN with 900 input, 100 hidden and a single output seems to be sufficient for detecting images containing nudity. The use of only an image of 30 by 30 pixels as input should be sufficient as most images can be classified manually when resizing images to this size. This will of course prevent the ANN to detect small undesired objects, but these seems to be very rare and can be neglected. As the total number of weights for this network is over 90 000 it might be necessary to use an even larger training set. At least twice as many is recommended by some while others recommend as much as 30 times the number of weights. The latter would in our case require approximately 2.7 million images in the training set. Our 40 000 images used for training might therefore not be enough. Using 2.7 million training cases would however result in a very slow training process and would require days, weeks or even longer to finish. As of now with 40 000 training cases the backpropagation algorithm can run about 100 epochs in 24 hours on a standard computer, with 2.7 million training cases this would require at least 90 times more processing time.

When investigating images that are falsely detected as containing nudity, it seems to be mostly headshots with lots of skin or other totally random images. These random images might be possible to reject using another ANN trained with slight differences in training data or even just a different number of hidden neurons. Many of the images that does not contain humans could probably be discarded by counting skin-percentage or detecting headshots.

It can be worth to mention that most of the images in the training set where either pornographic or completely lacked human presence. This could have had an negative effect on the result as the network will most likely not be so good at recognizing images that are close to the borderline. Our thought behind this was that the network would give a lower output for these images but still high enough to be classified as containing nudity with a lower threshold.

It is very difficult to compare our system with others as they use different data sets for evaluation and training. Some use images that have been acquired under controlled conditions and without access to their complete set of images it is almost impossible to make a decent comparison.

5.3.4 Feature ANN

The performance of the feature ANN was far worse than expected. This could be a result of bad performance on any of the features that are used as input to the ANN. An ANN for combining feature may not be the best way to proceed. Simple cascading conditions may be better suited for our purpose. This would most likely make the algorithm faster as not all features have to be calculated for every image. A start could be to check for skin-pixels and then discard all images that not contain enough skin at a first step. Images that do contain enough skin move on to the next classification step until it is either rejected or has passed all steps.

5.4 Texture filter

The texture filter shows promising results during testing but has to be validated further in order to verify its usage in this project. It removes a lot of hair, grass and other noise but also removes some skin, especially at edges. The filter is quite slow since it checks each pixel and also its neighboring pixels, especially at high resolution images. We designed the filter to be flexible and it is fully customizable in terms of thresholds and area of regions to check. The filter perform differently with different settings and these has to be evaluated in order to find the best performing ones. Due to time constraints we did not have time to evaluate the texture filter any further.

5.5 Performance in nudity detection

It is very difficult to compare our system with others as it uses totally different sets of images for evaluation. There are also different constraints on different systems, depending on the environment properties and accuracy demands. Thus, a comparison between different systems are limited to separate features.

5.6 Improving skin segmentation

Skin segmentation is only one part of this project, but with more time it could be improved by using preprocessing methods, like histogram equalization and color or lighting correction. An other method which might be useful is face detection. It would enable the detection of the specific skin colors in the image, thus adjusting the skin color range to the skin colors extracted from the face. This could potentially improve the performance of the skin segmentation in images with poor color reproduction.

Pixels that have the same color as skin but are not skin, will be classified as skin if only a pixel-wise segmentation is used. Experiments with a couple of methods that could possibly remove some of the false positives, was conducted. A texture filter and the morphological operations, opening and closing, was evaluated and the results seemed promising but they also removed some of the correctly detected skin pixels. However, we did not have time to evaluate these methods more accurate to get a fair result.

5.7 Morphological image processing

The dilation and erosion methods showed satisfying results during testing and removed much noise and also filled small gaps in the skin map. We did not however have time to evaluate whether or not these functions actually improved the segmentation or if it only was an ambiguous step in the process. Both dilation and erosion only requires one pass each over the skin map and is therefore very fast. The usage of the functions is also dependant on the previous segmentation and such steps as the recursive region function would make the dilation and erosion more or less superfluous

5.8 Face detection

Face detection with an ANN is in general as slow as any other approach. As most approaches uses a technique that requires the image to be scanned at every possible position and size, the speed of the classifier is very important, as it is can easily go over 50,000 scans in a single image. Cascading classifiers seems to be the fastest technique on the market. Using this technique as a preclassifier to our face-detection might help reduce the time required for every image significantly. Their cascading classifiers however only detect upright frontal faces. Our system is trained to detect both tilted and faces from the side so we would need other types of classifiers. The one we use now rely on color information so it is absolutely useless on black and white images. It would likely be possible to construct a preclassifier that reacts to edges instead. This could eliminate all positions in an image containing blank walls or blue sky. It might even be possible to discard locations with to much texture; such as grass or bushes, with this technique. Faces also tend to have distinguished borders on both sides and this could be an easy thing to check.

The ANN used for face detection in this project had approximately 25 000 weights. The training set contained only 3 000 cases which could easily have led to overfitting. To increase the performance more cases should be added to the training set. It might also be possible to reduce the number of neurons. We scanned an area of 24 x 24 pixels in the search for a face but this could probably be decreased to around 20 x 20 pixels or even lower. Many other researchers seems to have used sizes of around 20 x 20 pixels. This could lead to around 12 000 weights if the hidden neurons where decreased as well.

6 Conclusion

Detecting images containing nudity with the help of an artificial neural network (ANN) that takes features of an image as input was much harder then we had expected. The ANN works satisfactory on the training set but fails miserably on the validation set with only a few hits on images containing nudity. This could be the result of overfitting or an error in one of the features. Detecting images containing nudity is also a very complex problem. Images can be of very different origin and have noise or other degrading artifacts such as different lighting conditions and color distribution. It is interesting to see that the ANN trained on whole images outperforms the ANN trained on features and is also much faster. This proves that an ANN can be used for such a complex problem, if it is trained with enough training cases. The output from the image ANN is used as a feature and one might think that the ANN using features would train only to react on this feature and disregard the rest. The problem is that the image ANN was trained to detect pornographic images and not images containing nudity. This means that it does not react to the same things as the feature ANN is supposed to do. Adding training cases to the image ANN and retraining it should improve the performance of the ANN.
The image ANN will not be perfect and will most likely make mistakes on some images. Many of these mistakes may be averted by looking at simpler features such as skin percentage as images containing no skin will most likely not contain nudity.

The features that are extracted from the images work very well on their own, at least in the validation tests that we performed. Skin segmentation which uses only the color of the pixel will never be perfect but a true positive rate of over 80% with a false positive rate of 10% is very impressive since many images contain materials with skin-like colors.
Face detection also has satisfactory results even though it sometimes has a little too many false detections The overall false detection rate is small but since so many scans are performed on every image a few false detections is almost impossible to prevent.

Future work

It would be very interesting to see if it would be possible to detect humans in images and also their poses. It would then be possible to scan the human for clothes and establish a degree of nudity. It would then also be possible to see which body parts that are not covered with clothes and which ones that are. A naked torso would then be seen as inappropriate whereas a naked arm or face would be seen as normal. Such an algorithm would however face many challenges such as occlusion of bodies and unlimited variations in clothing and accessories.

7 References

Barron, A.R. (1993), “Universal approximation bounds for superpositions of a sigmoid function,” IEEE Transactions on Information Theory, 39, pp. 930-945.

Berry, M. J. A. and Linoff, G. (1997), “Data Mining Techniques”, New York: John Wiley & Sons.

Blum, A. (1992). “Neural Networks in C++”. New York: Wiley.

Boger, Z., and Guterman, H. (1997), “Knowledge extraction from artificial neural network models,” IEEE Systems, Man, and Cybernetics Conference, Orlando, Florida.

Brown, D. A., Craw, I. and Lewthwaite, J. (2001). “A SOM Based Approach to Skin Detection with Application in Real Time Systems”. The British Machine Vision Conference 2001.

Canny, J. (1986). “A Computational Approach to Edge Detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, no. 6 June 1986, pp. 679-698.

Chester, D.L. (1990), “Why Two Hidden Layers are Better than One,” IJCNN-90-WASH-DC, Lawrence Erlbaum, volume 1 1990, pp. 265-268.

Chetverikov, D. and Lerch, A. (1993). “Multiresolution Face Detection,” Theoretical Foundations of Computer Vision, vol. 69, pp. 131-140.

Craw, I. Ellis, H. and Lishman, J. (1987). “Automatic Extraction of Face Features,” Pattern Recognition Letters, vol. 5, pp. 183-187.

Craw, I. Tock, D. and Bennett, A. (1992). “Finding Face Features,” Proc. Second European Conf. Computer Vision, pp. 92-96.

DeVore, R.A., Howard, R., and Micchelli, C.A. (1989), “Optimal nonlinear approximation,” Manuscripta Mathematica, 63, pp. 469-478.

Elisseeff, A., and Paugam-Moisy, H. (1997), “Size of multilayer networks for exact learning: analytic approach,” in Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) Advances in Neural Information Processing Systems 9, Cambridge, MA: The MIT Press, pp.162-168.

Fairchild, M. D. (2005). Color Appearance Models, 2nd edition. Wiley, John and Sons Inc.

Fawcett, T. (2004). ROC Graphs: Notes and Practical Considerations for Researchers.

Fleck, M.University of Iowa, City of Iowa, Iowa, Forsyth, D. A., Bregler, C. (1996). “Finding naked people”. U.C. Berkeley, Berkeley, California.

Geman, S., Bienenstock, E. and Doursat, R. (1992), “Neural Networks and the Bias/Variance Dilemma”, Neural Computation, 4, pp. 1-58.

Gonzalez, R. C and Woods, R. E. (2002). ”Digital Image Processing”. New Jersey: Prentice Hall.

Govindaraju, V. (1996). “Locating Human Faces in Photographs,” Int’l J. Computer Vision, vol. 19, no. 2, pp. 129-146.

Hanbury, A and Serra, J. (2003). A 3D-polar Coordinate Colour Representation Suitable for Image Analysis. Vienna University of Technology.

ITU (2002). “Recommendation BT.709. Basic parameter values for the HDTV standard for the studio and for international programme exchange”.

Jones, M. J., Rehg, J. M., (1999). “Statistical color models with application to skin detection”, CVPR99.

Kanade, T. (1973). “Picture Processing by Computer Complex and Recognition of Human Faces,” PhD thesis, Kyoto University.

Kotropoulos, C and Pitas, I. (1997). “Rule-Based Face Detection in Frontal Views,” Proc. Int’l Conf. Acoustics, Speech and Signal Processing, vol. 4, pp. 2537-2540.

Lawrence, S., Giles, C.L., and Tsoi, A.C. (1997), “Lessons in Neural Network Training: Overfitting May be Harder than Expected,” Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97, AAAI Press, Menlo Park, California, pp. 540-545, http://www.neci.nj.nec.com/homepages/lawrence/papers/overfitting-aaai97/overfitting-aaai97-bib.html. Accessed 2008-06-20.

Lawrence, S., Giles, C.L., and Tsoi, A.C. (1996). “What size neural network gives optimal generalization? Convergence properties of backpropagation”. University of Maryland http://clgiles.ist.psu.edu/papers/UMD-CS-TR-3617.what.size.neural.net.to.use.pdf. Accessed 2008-06-28.

Lee, Jae Y., Yoo, Suk I. (2002). “An Elliptical Boundary Model for Skin Color Detection”. In Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology.

Lightness. http://www.poynton.com/PDFs/GammaFAQ.pdf. Accessed 2008-04-15.

Luv. http://dba.med.sc.edu/price/irf/Adobe_tg/models/cieluv.html. Accessed 2008-05-02.

Mardia, K. V., Kent, J.T. and Bibby, J.M. (1979). “Multivariate Analysis”. 2nd ed. London: Academic Press Inc, pp. 96-97.

Moody, J.E. (1992), “The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems”, in Moody, J.E., Hanson, S.J., and Lippmann, R.P., Advances in Neural Information Processing Systems 4, pp. 847-854.

Neal, R. M. (1996) “Bayesian Learning for Neural Networks”, New York: Springer-Verlag.

Norvig, P., Russell, S. (2003). “Artificial Intelligence - A Modern Approach”. 2nd ed. New Jersey: Prentice Hall, pp. 724-727.

Osuna, E. Freund, R. and Girosi, F. (1997). “Training Support Vector Machines: An Application to Face Detection,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 130-136.

Ripley, B.D. (1996) “Pattern Recognition and Neural Networks”. Cambridge: Cambridge University Press.

Rowley, H., Baluja, S. and Kanade, T. (1998). “Neural Network-Based Face Detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, Jan. 1998, pp. 23-38.

Sakai, T., Nagao, M. and Fujibayashi, S. (1969). “Line Extraction and Pattern Detection in a Photograph,” Pattern Recognition, vol. 1, pp. 233-248.

Sarle, W.S. (1995), “Stopped Training and Other Remedies for Overfitting,” Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, pp. 352-360, ftp://ftp.sas.com/pub/neural/inter95.ps.Z

Sirohey, S.A. (1993). “Human Face Segmentation and Identification” Technical Report CS-TR-3176, University of Maryland.

Sonka, M, Hlavac, V and Boyle, R (2008). “Image Processing, Analysis and Machine Vision”. 3rd ed. USA: Thomson Learning. 526-528. Chapter 7.2,-7.4, pp. 265-291.

Swingler, K. (1996). “Applying Neural Networks: A Practical Guide”. London: Academic Press.

Tetko, I.V., Livingstone, D.J., and Luik, A.I. (1995), “Neural Network Studies. 1. Comparison of Overfitting and Overtraining,” J. Chem. Info. Comp. Sci., 35, pp. 826-833.

Viola and Jones, (2004). “Robust Real-Time Face Detection”. International Journal of Computer Vision 57(2), pp. 137–154.

Wahde, M. (2006). “Introduction to Neural Networks”. Department of Applied Mechanics, Chalmers University of Technology. Göteborg, Sweden.

Weigend, A. (1994), “On overfitting and the effective number of hidden units,” Proceedings of the 1993 Connectionist Models Summer School, pp. 335-342.

Yang, G. and Huang, T. S. “Human Face Detection in Complex Background,” Pattern Recognition, vol. 27, no. 1, pp. 53-63, 1994.

Zheng, H., Daoudi, M. and Jedynak, B. (2004). “Adult Image Detection Using Statistical Model and Neural Network”. www-rech.enic.fr/coresa2004/articles/p144-zheng.pdf. Accessed 2008-05-22.