Todo list

BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DIPLOMA THESIS

TITLUL LUCRĂRII DE DIPLOMĂ

Supervisor: Author:

titlu oficial Mircea Ioan-Gabriel   **Sebastian Ștefan**

2018

Contents

rewrite section

Chapter 1

Introduction

1.1 Introduction

Skin detection means identifying pixels and regions in images which correspond to human skin. Starting from these regions we can build models that detect people or certain body parts.

There are many challenges in creating a good skin detector. One of the most challenging aspects is handling illumination. While the human eye can easily distinguish the same color in varying degrees of brightness this is not a trivial task for computers. The second aspect we must consider are the variety of skin tones. A good model should be able to handle people of all races. Another thing to note is the fact that there are a lot of objects and materials that resemble skin such as leather, wood, sand etc. Consequently, most models have a rather large false positive rate (around 10%).

The purpose of this paper is to present the best models currently employed in skin detection and to propose a new model that combines some state of the art detectors. To the best of my knowledge no method that combines image segmentation, color and texture analysis has been published in this area of research. For every approach in skin detection I included a theoretical introduction followed by some examples of its practical use in skin detection.

1.2 Applications

1.2.1 Pornography filtering

It is a known fact that internet has become a commodity of modern life, with a word average global penetration rate of 54.4%[1]. This means that it's inevitable for a large number of children to have unrestricted access to the internet and to be exposed to harmful content. A study conducted in England[2] on over a thousand people aged 11-16 has revealed that over half of them have been exposed to pornographic content.

To avoid such scenarios various pornography filters have been created. Skin detection is a

very useful step in implementing such an application because most pornographic images contain large areas of skin.

One approach is described by Fleck et al. [3]. The model proposed in that paper scans the images for large areas of skin and then applies some geometrical analysis looking for elongated shapes. This model was tested using 138 images of naked people and 1401 assorted images, some containing people but not naked. The system obtained a 52% recall and 60% precision against a large control set (1401 images).

Another similar approach was presented by Chan et al.[4]. The first step was skin extraction based on color which was followed by computing some high level features such as: the ratio of skin area to image area, the ratio of the largest skin segment to the image area and the number of skin segments in the image.

1.2.2   Anchorperson detection

Another application that was created in practice is anchorperson detection in talk shows [5] for automatic annotation and storage. The data can be used for quickly finding videos containing specific persons, calculating how much time each anchorman was on stage or other data mining tasks. This is an ideal application for skin detection because news videos are usually shot indoors in a controlled environment. Consequently, the background can be set up so it won't contain objects with skin color tones.

1.2.3   Preprocessing for complex systems

While skin detection does not have many direct applications, it is usually an intermediary step for detecting more concrete body parts, so I will also present some applications of those methods.

Face and hands detection is used to facilitate human computer interaction. By scanning gestures and reactions a software system can respond in a more convenient way to the user. For example the Kinect camera, which connects to the Xbox console, allows the user to navigate menus using hand swipes [6].

Aside from convenience gesture recognition systems can be very helpful for people that are aging or recovering after an accident or disease. Applications such as the one presented by Pedraza-Hueso et al. [7], are designed to help these types of people get into a better physical shape. Typically they run on a console like Kinect or Wii and take the form of games. These consoles make the users stand-up and perform different types of motions while playing. The main advantages of using such an application are: availability (users can do their exercises at any time in their own home), price (typically much lower than hiring a therapist) and entertainment (applications usually take the form of games which stimulate the user).

Face recognition, the next step after detection, means automatic identification of people in images or videos[8]. This has applications in security, such as unlocking your phone only when it

scans your face, and monitoring certain areas, like the systems used by the Chinese government to fine jaywalkers caught on camera without human intervention [9]. A trivial yet very popular use of face recognition is automatically tagging your friends in photos on social networks.

Chapter 2

Theoretical background

2.1 Related work

The subject of skin detection has been of interest for researchers due to its many applications. I could not include all the important papers in this domain due to the sheer number but I tried to cover the most relevant ones, in terms of popularity and results, for each aspect of skin detection.

2.1.1 Models using color analysis

Skin Probability Maps

One of the most comprehensive experiments in skin detection was conducted by MJ Jones and JM Rehg[10] who created the Compaq skin dataset, which became a standard for evaluating results of research in this domain. Their main approach was building a Skin Probability Map based on Bayes' Theorem. The resulting classifier, published in 1998, has obtained results comparable to modern approaches.

Skin probability maps are histogram based models, meaning they set a number of bins for each color channel where they keep track of how many times each color pixel appears in the training set. Although 256^3 would be an obvious choice for the number of bins (one for each pixel value) they determined that 32^3 is the optimal value.

Jones and Rehg[10] made the observation that given even a large training set most pixels are never seen. They explored their dataset of approximatively 2 billion pixels and came to the conclusion that around 77% of the RGB space is empty. This comes in support of their decision to use larger bins, like 32^3.

For this model they used a 6822 photo training set, where 4483 photos contained no skin and 2336 had portions of skin. The testing set had similar dimensions: 6818 total photos (4482 non-skin and 2336 skin). Using these methods they obtained a 80% true positive rate and 8.5% false positive rate or, with a different threshold, a 90% TP rate and 14.2% FP rate.

Gaussian Models

This approach consists of modeling human skin distribution as a Gaussian probability density function or a combination of Gaussian functions.

The previous paper[10] by Jones and Rehg also proposes a combination of two mixture models for skin and non-skin classes. Each model is composed of 16 gaussians and was trained on approximatively 74% of the histogram data, because only that data was available at the time. The results were similar to those of the previous model: 80%/9.5% and 90%/15.5%.

Lee and Yoo[11] implemented a Single Gaussian in CbCr and a Gaussian Mixture in IQ, both trained and tested on the Compaq dataset with the results of 90%/33.3% and 90%/30%, respectively.

Yang and Ahuja[12] propose a finite Gaussian mixture model. They theorize that this offers an improvement over a single Gaussian function and justify this hypothesis using statistical tests on normality and number of components, namely the Hawkins' Test on Normality and Homoscedasticity and McLachlan's Bootstrap Test. The selected methods of determining the model's input parameters is the Expectation Maximization (EM) technique. This experiment was performed on the Michigan face database that contains around 9.5 million skin pixels. The pixels were transformed from RGB to CIE LUV color space in order to separate and then remove the illumination component. The next step was applying a segmentation algorithm on the image and the analyzing each pixel's color. Every region that had more than 70% of its pixels classified as skin was labeled as a skin region. However, they do not offer and performance metrics on their experiment.

Elliptical boundary model

This is a statistical model proposed by Lee and Yoo[11] as an alternative to Gaussian models. The limitations that this model tries to overcome are the single Gaussian's luminance sensitivity and the mixture's slow training speed.

This model relies on the observation[12] that in most color spaces skin pixels tend to form an ellipsoidal cluster. The training algorithm has two steps: preprocessing and parameter estimation. The first step removes outliers in order to focus on the underlying density of the data set. This consists in removing the first k% samples with the lowest frequency. k is usually chosen in the 0 to 5 range.

The elliptical boundary model is defined as $\Phi(X) = [X - \Psi]^T \cdot \Lambda^{-1} \cdot [X - \Psi]$ where X is the chrominance vector of the considere pixel. Ψ is the average value of skin pixels in the training set. Λ is calculated as $\frac{1}{N} \cdot \sum_{i=1}^{n} f_i(X_i - \mu)(X_i - \mu)^T$ where μ is the mean of chrominance vectors and f_i their number of apparitions in training. T represents the model's threshold.

This model obtains an average of 90% TP rate and 23.3% FP rate on the Compaq dataset.

Explicit thresholding

Another popular approach is explicit thresholding. This domain consists of finding simple rules that apply to each pixel's color features and determine if it's part of the skin class. Some examples include [13], [14], [15], [16] for face detection systems in different color spaces with varying results.

A set of thresholds for YCrCb space where proposed by Chai and Ngan[14]. They set the ranges for Cb from 77 to 127 and for Cr from 133 to 173 and worked with the ECU database.

Dai and Nakano[15] created a model for YIQ, an orthogonal color space, which only used the I component (which stands for in-phase). The range they provided was [0, 50], however most of the images in their databases where of people with yellow skin.

An interesting approach, proposed by Gomez and Morales[17], is having a learner find these rules. They use RCA, a constructive induction algorithm, to build rules expressed with simple arithmetic operations in the rgb space. Their method achieves better results than the Bayesian SPM on their dataset, however it is computationally slower. RCA stands for Restricted Covering Algorithm which resembles a general covering algorithm with the restriction of trying to build a single rule for each class (in this case, a rule for skin detection). The strategy implemented for RCA was finding attributes which cover either a large number of true positives or a few false positives. The starting attributes where r, g, b and the constant 1/3, which would generate new attributes using the operators : +, *, - and squaring. One of their best and simplest generated models looks like this:

$$\frac{r}{g} > 1.185 \quad \text{and}$$
$$\frac{r * b}{(r + g + b)^2} > 0.107 \quad \text{and} \tag{2.1}$$
$$\frac{r * g}{(r + g + b)^2} > 0.112$$

In comparison with the C4.5 decision tree algorithm, the RCA method obtained slightly worse results but with much simpler rules. They used a custom dataset containing images of more than 2000 people and obtained around 90% both in precision and recall.

Brand and Mason [18] applied the technique from [16], which uses the YI'Q' space with the following threshold $14 < I' < 40$, on the Compaq dataset and obtained 94.7% TP rate and 30.2% FP rate.

Hidden Markov Models

Sigal et al.[19] implemented a method for real time skin detection in videos. Their model predicts the evolution of the skin color histogram using a second order Markov model, with an initial prediction from a Bayes classifier over the Compaq dataset. It uses the EM algorithm which consists of two steps: E(frame segmentation based on histogram) and M(histogram adaptation based on feedback from the current frame).

2.1.2 Models using segmentation

Frerk and Al-Hamadi[20] proposed a method that applies image segmentation combined with a Bayesian SPM. The first step is calculating the probability for each pixel in an image and creating P_I, the pixel probability image. The color and position of each pixel is used as input for a SLIC algorithm, an adaptation of k-means, that calculates the superpixels. The parameter k for segmentation is set to be the number of pixels in the image divided by 30^2, so that all regions have a size of approximatively 30x30. A probability is then computed for each superpixel and compared to a threshold.

Poudel et al.[21] took a similar approach to Frerk and Al-Hamadi's[20] work. Firstly, a segmentation is performed (using the Quick shift algorithm) and superpixels are extracted. A probability is calculated for each superpixel as the average of its component pixels probabilities. Then, a CRF(Conditional Random Field) method is used in order to obtain smoother skin regions. A CRF is a probabilistic method that works with graph based structures and models a distribution of type $P(y|x)$. The pair considered in this article is color difference and boundary length between neighboring superpixels. This model was tested on the Compaq database and obtained a 91.17% TP rate and 13.12% FP rate.

2.1.3 Models using textural features

Haralick features

El Abbadi et al.[22] presented a model based on Artificial Neural Networks that analyzes both color and textural features. The color features used are the mean color, the standard deviation and the skewness. These are calculated for each channel (R, G, B). The inputs for the ANN also include the following textural information: Entropy, Energy, Contrast and Homogeneity, which are computed from the Gray-Level Co-Occurrence Matrix. The network has three layers: an input layer, a hidden layer (with 50 neurons) and an output layer. The model was trained on 300 images (80x80 px) of skin and non-skin textures and tested on 100, obtaining a 96% accuracy in classifying whether an image is a skin patch or not.

Medjram et al.[23] proposed another method that combines color and texture information. The first step is converting the initial image to the YCbCr color space. Skin regions are identified using thresholding like the following: $77 <= Cb <= 127, 133 <= Cr <= 173$. Secondly, the image is sharpened in order to enhance its texture. The features considered from the GLCM are Contrast, Homogeneity and Energy, calculated over a 5x5 matrix. The last step is applying a Support Vector Machine classifier on the initial image to classify texture patches. They do not provide any performance metrics only a remark that the model performs well after being tested on hundreds of internet images.

Gabor wavelet transforms

Jian, Yao et. al. [24] use Gabor wavelet transforms to compute textural attributes. This step comes after an initial skin probability map calculation with the aim of reducing its false acceptance rate.

Gabor transforms are filters that analyze orientation and frequency. They use Gaussian functions modulated by complex sinusoids. Before applying this filter the image is converted from RGB to grayscale using the following equation $I(x, y) = 0.3R(x, y) + 0.59G(x, y) + 0.11B(x, y)$.

After the application of the texture filter some pixels that are actually skin are also eliminated which causes the True Acceptance Rate to drop. In order to increase this rate a region growing technique called watershed segmentation is used. This method is applied in order to grow the discovered skin regions based on similarity measures.

To evaluate this model they used their own manually labeled data set of 600 images. A simple SPM approach got 92.7%/20.1% TPR/FPR while the proposed method obtained 94.8%/4.2%.

Contourlet-Based Analysis

Fotouhi et al. [25] proposed a model that uses texture segmentation based on nonsubsampled contourlet coefficients.

A first step is using boosting to predict the class of each pixel using its color features. For this approach several Gaussian mixture models are chosen for the YC_rC_b, normalized rgb and HSV color spaces. This obtains a 93.1% TPR and a 22.8% FPR.

Next, a contourlet transform is applied over the whole image. For each pixel that was classified as skin by the previous method a 8x8 patch is selected in each subimage. These vectors are concatenated with the Principal Component Analysis technique and then fed into a Multilayer Perceptron for classification. The final model obtains a 82.8%/7.6% TPR/FPR rate.

2.1.4   Models using shape information

This approach is usually applied for the detection of specific skin regions such as faces[26] or hands[27] which have predefined and easy to identify shapes. However, there are some general skin detectors that consider shape features that are based on the assumption that most skin patches, like faces, hands or legs, have an elliptical shape.

Such a model was proposed by Kruppa et al.[28]. Their approach works in two steps. Similar to other FAR reducing techniques, the first step is applying a skin color detector as it provides a good starting point for further refinement. After identifying skin color regions a skin patch filter

is applied that finds elliptical shapes in the image.

$$S(x_c, y_c, w, h) = \begin{cases} \frac{1}{1+exp^{-a}} & if \quad \frac{(x-x_c)^2}{w^2} \pm \frac{(y-y_c)^2}{h^2} \leq 1 \\ 0 & else \end{cases} \tag{2.2}$$

The shape model can be seen in figure 2.2. In this equation (x_c, y_c) denote the center and w, h the dimensions of the ellipse and a is a parameter used to smooth out probabilities. These parameters are derived from the skin color model using a technique to maximize mutual information between two probabilities.

This model was tested on a data set of 653 images resized to 150x100 resolution. The authors report an improvement over the classical color model by up to 25% in the error rate value.

2.2   Image segmentation

Image segmentation is a technique for dividing an image into several regions that contain similar pixels. These partitions are often called super-pixels and represent an abstraction layer over the initial image. They can be characterized by color, border or shape(circle, ellipse, polygon, etc.)[29]. The main purpose of segmentation is to represent areas of interest in an image such as faces, fields, buildings, etc, and usually serves as a preparation step for a more complex detection algorithm.

Ideally, the resulting regions should have the following characteristics: uniformity according to the selected feature, such as color or texture, a small number of holes(subregions that differ considerably from the container region), a notable difference from the neighboring areas and smooth borders.[29].

This problem has been researched extensively over the years however there is not a single best solution available. I will present some of the algorithms presented in literature.

2.2.1   Thresholding

Thresholding is one of the simplest approaches to image segmentation. It consists of finding a threshold T for the gray level of pixels in the image. Therefore, we can classify every pixel by comparing its brightness to T. This technique is a perfect fit for separating objects from a darker background but has severe limitations in other tasks[30].

One of the problems with global thresholding, choosing a single value for T over the whole image, is dealing with different levels of illumination. For that reason the local thresholding approach was developed. This expands the previous method by using multiple thresholds for different parts of the image. Chow and Kaneko [31] applied a similar method for detecting the left ventricle of the heart in x-ray pictures. They divide the original image in multiple blocks that do not overlap and calculate a threshold for each of them.

Threshold selection can be done manually or automatically and usually involves analysis of a gray level histogram where the size of each bar is proportional to the number of pixels with that brightness[32]. For example, in an image that contains some objects in front of a darker background the histogram is likely to contain two peaks separated by a valley and we can choose T somewhere in between those peaks.

Automatic threshold detection can be done by modeling object and background populations with a normal distribution like the experiments presented in [33]. The method proposed calculates a least-squares fit of a function f(i), where i is the gray level, to the histogram, using a hill climbing algorithm. The function takes into account the mean and standard deviation of the histogram. Lastly, the best fitting f(i) is the used for classification.

However simple, all these methods have limitations, such as not considering spatial factors, and having no control over border smoothness and holes inside detected regions.

2.2.2 Edge detection

These methods aim to solve image segmentation problems by detecting all edges in images. An edge is characterized by a sudden change in pixel intensity[30]. The result of edge detection is an image that represents to classes of pixels: part of an edge or not. I will present some of the most popular methods.

The Roberts Detection[34] is one of the fastest methods due to its simplicity. It uses the following convolution masks:

$$Gx = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} Gy = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}$$

Each of theses calculates a gradient value for an orientation and their values can be combined to determine an absolute magnitude of the gradient using: $|G| = \sqrt{G_x^2 + G_y^2}$. Lastly, we can apply thresholding for the resulting value to determine if it is part of an edge or not.

Other similar detectors are the Prewitt and the Sobel detectors[34]. The difference is that they use a larger convolutional mask, 3x3, and in case of the former, more orientations.

Some soft computing techniques such as: neural networks, genetic algorithms and fuzzy logic[34], have also been applied in edge detection problems. However they don't offer significant improvements in performance over the simpler and faster methods.

2.2.3 Clustering

We can view the image segmentation process as a clustering problem. All pixels are associated with a region, which corresponds to a cluster.

K-means

One popular approach to segmentation problems is the K-means algorithm, which can divide n data points in k clusters, where k is known in advance. This can be applied to image segmentation if we consider pixel intensities as the values on which we are doing clustering. The algorithm is simple and efficient on large volumes of data[30], however the number of clusters must be known in advance, which can be problematic as the number of distinct regions can vary from image to image.

Mode seeking. The Quick shift algorithm

Mode seeking algorithms are methods for clustering without needing to know the number of clusters in advance. The first step of these algorithms, as presented in [35], is calculating a Parzen density estimate over all given points: $P(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} \cdot k(x - x_i)$. The next step is moving each point towards a mode of P(x) on a trajectory determined by the gradient of P. Clusters are formed by the points converging on the same modes.

The Quick shift algorithm[36] is one of the most efficient in this family. It starts by calculating a Parzen density estimate, most often using an isotropic Gaussian window:

$$P(x) = \frac{1}{2\pi\sigma^2 N} \cdot \sum_{i=1}^{N} e^{\frac{-\|x-x_i\|^2}{2\sigma^2}} \tag{2.3}$$

Every point is then linked to the nearest one with a higher density and the resulting structure is a tree. In order to get our regions we can either cut branches with a size larger than τ or analyzing data points less then τ distance away on the linking step. If we choose the latter the result will be a collection of trees, each representing a cluster.

In the application of Quick shift in image segmentation a relevant feature space must be chosen. An option is using the RGB components and position(x, y coordinates) of each pixel. As the position does not always has fixed bounds (image size can vary) we can scale the position components depending on our input data, such that the importance of color and position remains similar[36].

An example pseudocode implementation as presented in [36] looks like 1. When calculating the density we limit our search to a 3σ window because the contributions for pixels further away should be small[36].

Some advantages of Quick shift are simplicity of implementation, speed $(O(N^2))$, ability to work on any type of data and the control of fragmentation with the given parameters[35].

13

```
# density computation;
for x in all pixels do
    P[x] = 0;
    for n in all pixels less than 3 * σ away do
    |   P[x] += exp(-(f[x] - f[n])^2 / (2 * σ * σ))
    end
end
# neighbor linking;
for x in all pixels do
    for n in all pixels less than τ away do
        if P[n] > P[x] and distance(x, n) is smaller than to previous parent then
            d[x] = distance(x, n);
            parent[x] = n;
        end
    end
end
```

Algorithm 1: The Quick shift segmentation algorithm

2.3   Skin detection by color

Skin pixel detection by color means classifying a pixel while considering only its color features. A first step in applying this approach is selecting a color space.

2.3.1   Color spaces

A color space, also called a gamut, represents a set of colors in a way that is independent of the medium in which they are represented(computer screens, cameras, magazines, etc)[37]. The L*a*b* color space contains all colors that can be seen by the human eye, however most color spaces are smaller due to technical limitations. I will present some of the color spaces which have been used successfully to classify skin pixels.

RGB

To start with, RGB is one of the most popular color spaces for working with image data. It matches the color sensitive receptors of the human eye(red, green, blue) and started as a convenient way to represent the colored rays used by CRT screens[38]. While this model is simple to use it has the disadvantage of mixing chrominance and luminance features[38].

Normalized RGB

Normalized RGB is a color space with a lighter memory consumption than RGB and its components are calculated as follows[38]:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}. \tag{2.4}$$

The third value can be determined from the other 2 so we can avoid storing it. Other advantages according to [39] include reduced differences caused by illumination and ethnicity, and lower variance of skin color clusters than in the normal RGB space.

HSI, HSV, HSL

HSI, HSV, HSL represent perceptual color spaces and they describe the hue, saturation and intensity (or value, lightness). These color spaces are used because they provide invariance to ambient lighting and surface orientation relative to the source of light[39]. We can convert to HSV from RGB using the following formulas[38]:

$$H = arccos\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2 + (R-B)(G-B))}} \tag{2.5}$$

$$S = 1 - 3\frac{min(R,G,B)}{R+G+B} \tag{2.6}$$

$$V = \frac{1}{3}(R+G+B) \tag{2.7}$$

YCbCr

Orthogonal color spaces, which YCbCr is a member of, provide chrominance and luminance separation as they represent colors with statistically independent components. YCbCr is mostly used by European television studios and in image compression[38]. Y represents luma (or luminance) and Cb, Cr are the blue and red difference chroma components and they can be computed as follows:

$$Y = 0.299R + 0.587G + 0.114Bs \tag{2.8}$$

$$Cb = B - Y \tag{2.9}$$

$$Cr = R - Y \tag{2.10}$$

Having such a simple transformation and a clear separation of the luminance component makes the YCbCr a popular choice for skin detection models.

2.3.2   Explicit thresholding

This is one of the simplest skin-color models that can be built. The method aims to define, through the use of simple rules and thresholds, the boundaries of skin clusters in a specific color space. It has been observed in [40] that the colors of human skin tend to cluster in small regions of the color space and human skin pixels differ more in intensity than in color.

An example using the RGB space, from Peer et al.[13] which has been integrated into a face detection system consists of the rules below:

$$
\begin{aligned}
R > 95 \quad \text{and} \quad G > 40 \quad \text{and} \quad B > 20 \quad \text{and} \\
max\{R,G,B\} - min\{R,G,B\} > 15 \quad \text{and} \\
|R - G| > 15 \quad \text{and} \\
R > G \quad \text{and} \quad R > B
\end{aligned}
\tag{2.11}
$$

These thresholds and rules can also be generated using an artificial intelligence algorithm. For example, we could apply genetic programming in a fashion similar to that described in Segaran's book [41]. Rules can be represented as trees whose leafs are color features or constants and inner nodes are simple operators.

The main advantage of using these methods is their speed. For each pixel we apply only a few operations, with a computational time of O(1). For this reason, thresholding techniques tend to be used in real time systems, where fast feedback is more important than overall precision.

2.3.3   Skin Probability Map

A SPM represents a histogram with multiple bins. Each bin stands for a color or a subset of colors and has a value equal to the number of times that values has appeared in the training dataset. When building a SPM you must choose the color space and the number of bins per color channel.

In order to determine whether a given pixel is a skin colored pixel we apply Bayes' theorem. This theorem, that in the general form looks like this $P(H|E) = (P(E|H) \cdot P(H))/P(E)$, is attributed to 18-th century mathematician Thomas Bayes. It allows us to update probabilities for hypotheses(H) given evidence(E)[42]. In simpler terms, the theorem calculates a more accurate probability of an event if we know its relation to other available information. For example, given that there is a correlation between liver diseases and alcoholism you can calculate a more accurate probability of a person having such an illness if you take into account their alcohol consumption.

In order to apply this theorem to our problem we need to model our domain hypothesis(the probability of observing a skin pixel) and evidence(the number of skin pixels with certain values observed in the training data). Here is the form used by Jones and Rehg[10] in one of the most

popular papers on statistical skin detection:

$$P(skin|p) = \frac{P(p|skin)P(skin)}{P(p|skin)P(skin) + P(p|\neg skin)P(\neg skin)} \tag{2.12}$$

In this equation "p" is the notation for the occurrence of the given pixel. Therefore $P(skin|p)$ means the probability of observing skin given our pixel. To determine whether we classify the pixel as skin we compare our probability with the threshold value, β.

$$P(skin|p) > \beta \tag{2.13}$$

The probability of observing skin can be computed as the ratio of the number of skin pixels to the total number of pixels observed in training.

$$P(skin) = \frac{T_S}{T_S + T_N} \tag{2.14}$$

It has been observed that even when using a large data set most pixel values are never seen[10]. This suggests that we might get better results by reducing the number of bins per channel. We can further observe that a small perturbation in the RGB values of a pixel results in a very similar color. Consequently if we classify p as a skin colored pixel then there is a high probability that its neighbors in the color space are also skin colored pixels. This observation is in support of a smaller number of bins for the SPM histogram.

2.3.4   Gaussian classifiers

Gaussian classifiers are parametric skin distribution models with the advantages of being compact, therefore using less memory than SPMs, and able to generalize better using less training data[39].

Single Gaussian

Considering the observations from the thresholding chapter that skin color pixels tend to cluster in a region of the color space we can model that distribution using an elliptical Gaussian joint probability density function, an example of which was provided by [39]:

$$p(c) = \frac{1}{2\pi^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(c-\mu)^T\Sigma^{-1}(c-\mu)} \tag{2.15}$$

In this equation, c is the color vector, μ the mean vector and Σ the covariance matrix. These can be calculated from the training data as follows:

$$\mu = \frac{1}{n} \cdot \sum_{j=1}^{n} c_j$$

$$\Sigma = \frac{1}{n-1} \cdot \sum_{j=1}^{n} (c_j - \mu)(c_j - \mu)^T \tag{2.16}$$

Here c_j are all the color samples used in training the model. In order to establish that the given color describes skin we can compare p(c) with a threshold that can be determined experimentally for a given training set.

Gaussian Mixture Models

A Gaussian mixture model is a form of unsupervised learning used to identify subpopulations within an overall population, provided they are normally distributed[43]. It has been observed in [12] that a mixture of Gaussians is better suited for skin detection than a single distribution, especially in datasets with multiple illumination conditions.

They represent a generalization of the single Gaussian. A mixture's density function can be calculated as the sum of individual Gaussians[39]:

$$p(c) = \sum_{i=1}^{N} w_i \cdot \frac{1}{(2\pi)^{1/2} \cdot |\Sigma_i|^{1/2}} \cdot e^{-\frac{1}{2} \cdot (c - \mu_i)^T \Sigma_i^{-1} (c - \mu_i)} \tag{2.17}$$

In equation 2.17 c, μ_i and Σ_i are the color vector, mean vector and covariance matrix for the ith Gaussian. Also, each of the N models has a weight, w_i, representing its contribution to the mixture.

Determining the unknown parameters can be done with the Expectation Maximization (EM) technique[12]. This is an algorithm of maximum likelihood estimation, or simply speaking finding the parameters that best describe some given data.

The number of Gaussians, N, is also an important aspect. The most common values used in research fall into the 2 to 16 range[39], the idea behind choosing a larger number is to account for various conditions of illumination.

2.4   Texture analysis

Texture is a propriety of all existing surfaces and contains information about their structural composition. While easily identifiable by a human observer, it does not have a precise mathematical definition[29]. Consequently, there are many choices for building a texture classification model depending on the author's interpretation. A good example that illustrates the complexity of defining texture is the surface of wood, which contains intensity variations that despite being nonuniform can compose certain repeated patterns. Texture provides information about the spatial arrangement of intensities in an image[29].

The main approaches in studying image texture are: structural, statistical, model based and transform methods[44].

2.4.1 Structural methods

Structural methods texture as a composition of primitives called texels(microtexture) arranged in a regular or repeated manner(macrotexture). Take a chess board for example - all squares are repeated in a predictable way. This approach is more useful for generating textures than analyzing them because in nature the difference between micro and macro textures is not very clear and the their variability is high.

Voronoi tesselation

The first step is extracting the texels from the image. The implementation of this part can vary depending on the types of textures that are analyzed. An example would be to use thresholding to isolate image regions corresponding to texels[29].

In order to describe the relationships between the extracted regions we compute the Vornoi tesselation. This means finding all points in the image that are closer to the center of a texel than to the center of any other texels. For example, if S is the set of texel centers and P is a value in that set, then the Voronoi polygon of the point P is a polygonal region that covers all points closer to P than to any other point in S[29].

The final step is extracting shape features from the Voronoi polygons and using them to group similar polygons into clusters. Each of these cluster defines a distinct textural region[29].

2.4.2 Statistical methods

In practice this approach is the most common because of its generality and performance. The main concept that these methods have in common is the computation of statistics on the relationships between pixel intensities. These have also been shown to perform better than the other approaches, mainly through the use of second order statistics[44].

Local Binary Partition

This is one of the simplest texture description methods. A simple algorithm is given in [29]. To extract the features of a window you have to iterate all pixels and using the current pixel as the center compare it with eight neighbors (north, north-west, west, etc.) chosen on a circular path. If the compared pixel's value is greater than the center's append a "1", else append a "0" to a string. These string form binary numbers for which a histogram is created. The histogram counts the number of apparitions of each distinct number formed in the previous step. The histogram represents the textural features of the window. Multiple histograms can be compared using the L_1 distance(figure 2.18).

$$L_1(H1, H2) = \sum_{i=1}^{n} |H1(i) - H2(i)| \qquad (2.18)$$

First-order histogram based features

This is a simple statistical approach that relies on first-order histograms. These histograms show, for each gray-level, the number of pixels with that intensity value in the image. This can be computed as in figure 2.19, where I is an image of size(N, M), I(x, y) the pixel located at x, y coordinates, i is a grayscale value, and $\delta(x, y)$ is the Kronecker delta function which returns 1 if x = y and 0 otherwise[44].

$$h(i) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \delta(I(x, y), i) \tag{2.19}$$

To extract features from this histogram we need to compute some metrics on it. Some of these have been exemplified in figure 2.20 but there are many more that can be applied.

$$Mean: \quad \mu = \sum_{i=0}^{G-1} i \cdot p(i)$$

$$Variance: \quad \sigma^2 = \sum_{i=0}^{G-1} (i - \mu)^2 \cdot p(i)$$

$$Skewness: \quad \mu_3 = \sigma^{-3} \cdot \sum_{i=0}^{G-1} (i - \mu)^3 \cdot p(i) \tag{2.20}$$

$$Kurtosis: \quad \mu_4 = \sigma^{-4} \cdot \sum_{i=0}^{G-1} (i - \mu)^4 \cdot p(i) - 3$$

G represents the maximum value for the pixel intensity and p(i) the probability of finding that intensity in the image. Skewness is an indication of symmetry, kurtosis measures the flatness of the histogram and the mean and variance provide information about the average lighting conditions[44].

Haralick's features

This is a statistical method of texture analysis named after Robert Haralick, the author of the paper that first presented them[45]. In a nutshell, they can be described as a series of statistics calculated over the Gray Level Co-occurrence Matrix(GLCM). The author relies on the assumptions that there is a relationship between the texture and the extraction of discrete textural features is a fuzzy task. Based on this, he proposes an analysis of the spatial relationships between gray tones in images.

The basic descriptor for gray tone dependencies is the GLCM. Simply put, this is a matrix that tracks how many times a gray level (a pixel color converted to grayscale) appears in relation to another. This relationship is given by adjacency with an offset. By far, the most common type of GLCM is the one used for second order calculations, meaning that it takes into consideration groups of two pixels from the original image. A third order and further matrix is theoretically possible but computationally restrictive[46].

The distance between the considered pixels can be written as an offset (dx, dy), meaning that the second pixel's position is dx right and dy down in relation to the first. The Gray Level Co-Occurrence Matrix, C_d of image I, whose indexing is done by pixel intensities (i, j) can be computed as seen in figure 2.21, which was presented in [29].

$$C_d(i,j) = |\{(r,c)|I(r,c) = i\,and\,I(r+dc,c+dc) = j\}| \qquad (2.21)$$

There are some variations on the classical GLCM structure such as: the normalized gray level co-occurrence matrix(figure 2.22), that normalizes all values to the (0,1) interval allowing them to be seen as probabilities, and the symmetric gray level co-occurrence matrix(figure 2.23) which groups symmetric adjacencies.

$$N_d(i,j) = \frac{C_d(i,j)}{\sum_i \sum_j C_d(i,j)} \qquad (2.22)$$

$$S_d(i,j) = C_d(i,j) + C_{-d}(i,j) \qquad (2.23)$$

Relying on the assumption that gray-tone arrangement can describe texture, Haralick et al. defined a set of 14 measures of textural features as functions over the normalized, symmetric form of the GLCM. These are the: Angular Second Moment, Contrast, Correlation, Sum of Squares: Variance, Inverse Difference Moment, Sum Average, Sum Variance, Sum Entropy, Entropy, Difference Variance, Difference Entropy, Information Measures of Correlation(two of them) and the Maximal Correlation Coefficient. Their method of calculation and significance are explained in-depth in Haralick's paper[45].

In a typical texture detection problem these statistics are calculated on a window of the given image. This allows the algorithm to identify several textures in the same image. The GLCM features are calculated on each window and then fed into a machine learning classifier which can then distinguish between differently textured areas.

2.4.3   Model based methods

Model based approaches try to represent texture through the use fractal and stochastic models. These have been shown to perform well on some natural textures, however they imply a high computation complexity and lack orientation invariance[44].

Fractal models

A fractal can be defined as a surface that repeats itself when observed at different scales. This is similar to natural occurring shapes such as clouds, trees, rivers, etc. Consequently, this model relies on the assumption that most textures have this property of self-similarity[47].

The main feature extracted by this model is the Fractal Dimension, which is used to measure the degree of roughness of a surface. The concept of a fractional dimension was defined by

Hausdorff and Bescovitch in the following way: $D = log(N_r)/log(1/r)$, and represents an indication of how detail changes in a pattern according to the scale of measurement. Here N_r is the number of copies of the seed and r is the scaling factor. Starting from this equation, the FD can be estimated using various methods, such as: The Blanket Method or the Differential Box Counting Method[47].

2.4.4 Transform methods

Transform methods include Fourier, Gabor and wavelet transforms. These approaches transform the image into a space of features relevant to texture, such as frequency or size[44].

Gabor features

These features, named after Dennis Gabor, are extracted using Gabor filters and have applications in texture segmentation, edge detection, document analysi, etc. Many vision scientists claim that these filters conform well to the mammalian visual system[48].

Image analysis employs the two-dimensional kind of Gabor filters. These are formed as a product between a sinusoidal function and a Gaussian function: $h(x, y) = s(x, y) \cdot g(x', y')$. The complex sinusoid, also known as carrier can be written as equation 2.24 where U, V give the frequency of the function.

$$c(x, y) = e^{j \cdot 2\pi(Ux+Vy)} \tag{2.24}$$

The Gaussian function, also known as the envelope looks like equation 2.25. The values of $(x', y') = (xcos\theta + ysin\theta, -xsin\theta + ycos\theta)$ are used to control rotation.

$$g(x, y) = \frac{1}{2\pi \cdot \theta_x \theta_y} \cdot e^{-0.5 \cdot (\frac{x^2}{\theta_x^2} + \frac{y^2}{\theta_y^2})} \tag{2.25}$$

The previously exemplified form of a 2D Gabor filter is a simple yet efficient one, using elementary functions, and was introduced by Dunn et al. [49].

The combination of this functions allows Gabor filters to be invariant of rotation, scale and translation, a much desired property in image feature analysis. In general multiple Gabor filters forming a "filter bank" are applied to the same image to get a more complete representation of the desired features. These filters are obtained by tweaking the general formula's parameters, the most important of which are frequency and orientation. The values obtained from the concatenation of the filter bank's results are used as features and fed into a machine learning algorithm[50].

2.4.5 Use in skin detection

The main use of texture analysis in skin detection is to reduce the false positive rate. I haven't found any source that uses a texture model as a sole detector, possibly due to the fact that many

surfaces with smooth texture can be misclassified as skin.

Chapter 3

Application development

3.1 Proposed model

3.1.1 Overview

The model proposed by this paper is made up of three steps and was designed to combine the most efficient detectors in all aspects of skin detection. There is a preprocessing step that is applied to all images, namely a resize operation to the (200, 200) dimension, with the purpose of reducing computation time.

Segmentation

The first step is image segmentation, which consists of finding uniformly colored regions, with smooth borders and regular shapes. These attributes fit the description of skin regions well. By passing these patches to the color model we can ensure the results will not contain holes caused by some small variations in a skin region and their borders will follow the skin contours more naturally.

I chose the Quick shift algorithm for this task due to its simplicity and ability to perform clustering without needing to know the number of clusters in advance. Furthermore, this approach allows for future performance optimization as it has been shown that a GPU implementation is possible[36].

I chose the classic input feature for segmentation, the 5-dimensional tuple $(f1, f2, f3, s \cdot x, s \cdot y)$ where $f1, f2, f3$ represent the color features in a certain color space (e.g. RGB) and x, y the coordinates of the pixel. The scaling factor, s, is a constant that can be used to adjust the importance of the spatial features. This was necessary because the pixel position value range scales with image dimension while the color features remain in the same range(0 - 255).

This step extracts a list of superpixels containing the most dominant pixel in the region and all pixels linked to it, which is then fed into the color analysis module.

(a) Without segmentation   (b) With segmentation
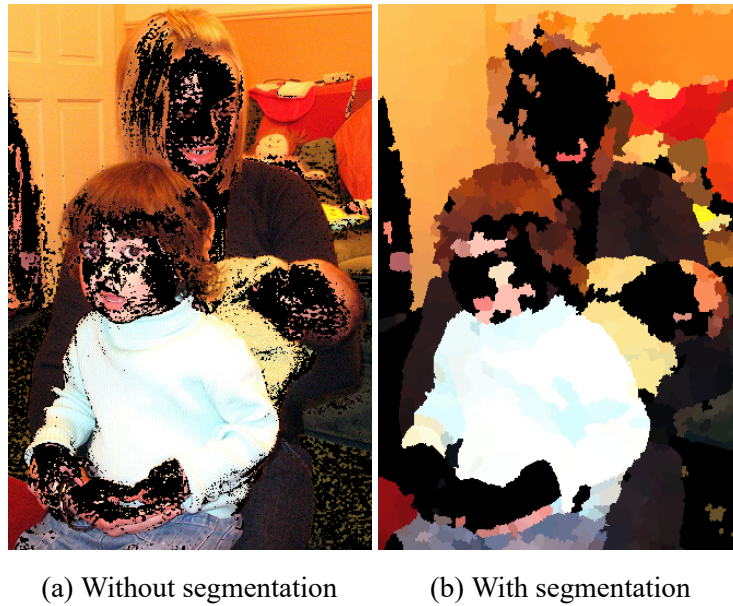
Figure 3.1: Color analysis with and without segmentation

Color analysis

This is the core model used in almost every skin detection approach seen in research. By identifying skin colors you can obtain a high true positive rate and a good starting point for reducing the false acceptance rate.

This module uses Skin Probability Maps to analyze the color feature of each region identified by the segmentation module in the image. The main reasons for selecting this approach are its performance, being marked second in Kakumanu's survey[39], and speed.

My color model is similar to the well known SPM method proposed by Jones and Rehg[10] with a notable difference. In the classical approach the histogram bin size is determined before the training phase. This means the training must be redone if we wish to experiment with other sizes.

My approach is to count the number of pixels for each value and take into account a pixel's neighbors only during the evaluation phase. This means that whenever a probability is calculated the algorithm also computes the neighbors probabilities in the color space and choses the maximum value. This allows for the training to be done only once, however it also means that the model will have a slightly larger size.

Furthermore, when calculating the Bayesian probability the algorithm builds a "bin" around each pixel, meaning that the currently analyzed pixel is always at the center of its bin. Simply put, each probability computation takes into account the pixel and its closest neighbors. However, I found that, in practice, placing the analyzed pixel at the center of the bin does not affect the algorithm's performance noticeably.

A Naive Bayesian probability is calculated for every pixel in each region and the average of these is compared to a threshold to determine if the superpixel's color is likely to be that of skin.

Finally, a skin mask of black color is applied to all regions identified as skin and the image is saved to be used in the final step.

Texture analysis

The main purpose of this step is to reduce the false positive rate of the color analysis module, by identifying objects with skin color but different texture such as wood, leather, etc.

The texture model employs Haralick's features as a descriptor. The main advantage of this approach is that important feature descriptors such as coarseness, roughness and directionality are all taken into account by these metrics[45]. Consequently the textural features of an image patch are the Haralick features computed from the Gray Level Co-Occurrence Matrix with a distance of 1 in each of the four directions (horizontal, vertical, nw-se, ne-sw). The 13 features presented in the initial paper[45] are calculated for each direction and the final feature vector is made up out of the mean value of each feature, having the size of (13, 1).

These features are fed into a Support Vector Machine classifier that tries to learn how to distinguish skin and non-skin textures. A SVM is a supervised machine learning algorithm used, in this case, for binary classification. This is an efficient model if we have the means to extract features about the desired concept (Haralick's texture descriptors) and tends to generalize well[51].

In order to extract only the skin-textured blocks from an image, the model performs texture analysis by building a window around each pixel. The textural features are extracted only from that window and if the SVM detects skin then the center pixel is classified as skin. The model uses a small window because skin-textured areas can have various shapes and sizes, and a large window would have a small acceptance rate for images with small patches of skin.

Finally, a black mask is applied on all pixels identified as centers of skin texture regions and the image is saved to be processed in the next step.

(a) Without texture analysis   (b) With texture analysis

Figure 3.2: Detection with and without texture analysis

The final image

The final image built by the model is a combination of the two images obtained from color and texture analysis. The new skin mask is build as the intersection of the masks generated by the previous two steps. This means, that a pixel is classified as a skin pixel only if both the color and the texture models agree. This relies on the assumption that color and texture are complementary skin descriptors and their combination can reduce the false positive rate.
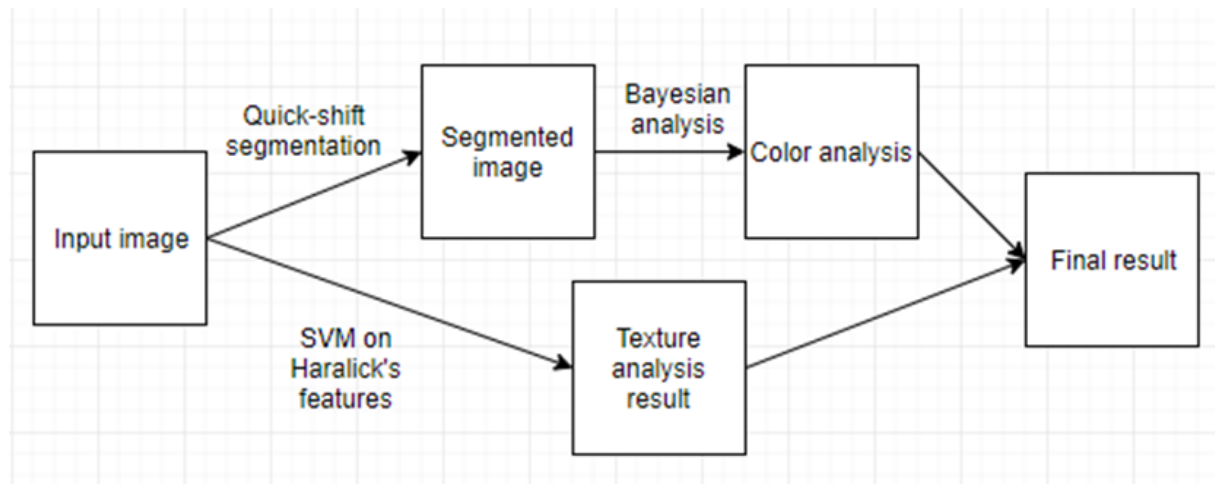

Figure 3.3: Graphical overview of the model

3.1.2 Datasets

For training and evaluating the model I have used two databases. First, the Compaq dataset [10], which is one of the most cited in literature, contains 13.000 images, out of which 4700 contain skin, and was used for the skin color model. The images were downloaded using a web crawler then divided into skin and non-skin images. For every skin image a black and white mask was created by hand to mark the regions of interest. An interesting observation is that even though this dataset is the largest one concerning skin detection, 76.6% of possible RGB values are not encountered at all. I obtained this database directly from its author, however a few hundred of images were corrupted, possibly due to working in a Windows environment, and I had to write a script to filter them out.

The second dataset, SFA [52], was created as a combination of FERET(876 images) and AR(242 images) datasets. It contains skin and non-skin patches of dimension from 1x1 to 35x35, which makes it ideal for the texture model. For each dimension, SFA has 3354 skin images and 5590 non-skin images.

In order to get visual feedback from the model, I used the Pascal 2007 database[53]. This dataset was built for an object detection competition and is quite varied. I selected a subset of images that contained people to use for testing my model. The reason for using this data set for tests is that the images are of a better quality than those present in Compaq or SFA.

27

3.1.3 Parameters

Image segmentation

For image segmentation I tried to find values for the sigma and tau parameters that produce regions as large as possible without distorting the contours. I chose the (3, 5) pair, which can be seen in comparison with other pairs in figure 3.4.

The scaling factor for position features was set experimentally depending on my chosen image size (200, 200). I found the value of 0.5 to produce regions with the most accurate borders.
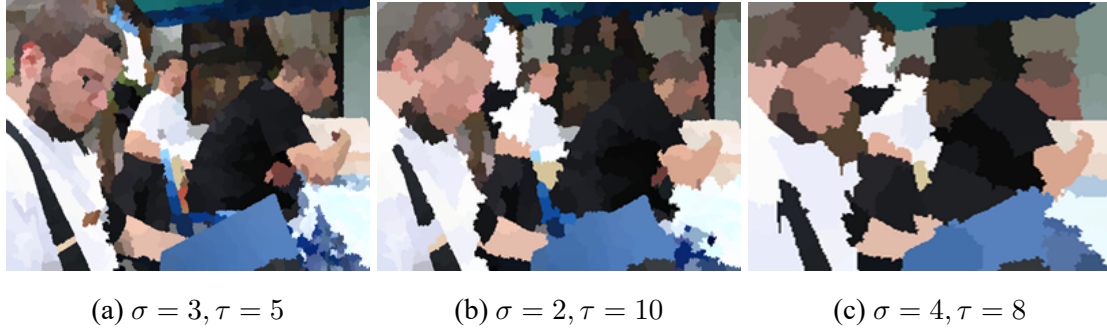

<div align="center">

(a) $\sigma = 3, \tau = 5$ (b) $\sigma = 2, \tau = 10$ (c) $\sigma = 4, \tau = 8$

</div>

Figure 3.4: Segmenting an image with different parameters

Texture analysis

The most suitable area for texture analysis was also determined experimentally. As can be observed in figure 3.5 the 5x5 window offers a good detection of skin areas while the true positive rate goes down drastically for larger ones. This can be explained by the fact that large patches can cover other surfaces bordering skin such as clothes, eyes, lips, etc.

<div align="center">

(a) 5x5   (b) 7x7   (c) 35x35

</div>
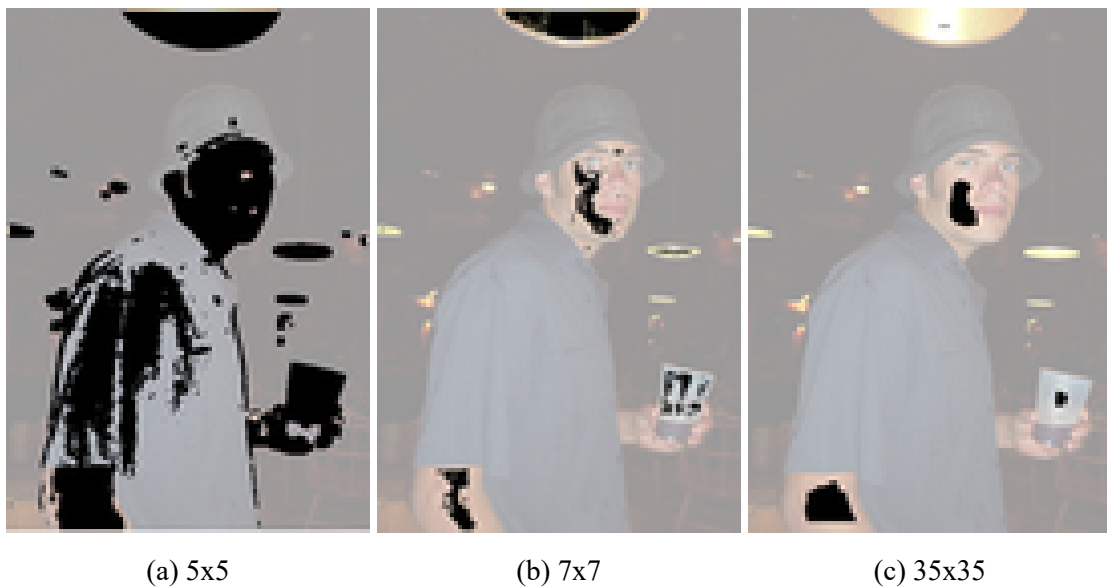
Figure 3.5: Texture detection with different windows

Color analysis

I tried building color models in RGB, HSV and YCrCb and concluded that Bayesian color analysis obtains roughly the same results regardless of the color space chosen(see table 3.1).

For every pixel, I chose the maximum probability on a distance of 4 (similar to the bin size used in [10]) on every feature. The threshold value can be used to control the TPR/FPR ratio.

Table 3.1: The best performing models for each tested color space

Metrics	RGB	YCrCb	HSV
TPR	0.828	0.817	0.815
FPR	0.227	0.216	0.209
Accuracy	0.933	0.927	0.929
Precision	0.506	0.515	0.522

3.1.4   Training

Training is done separately for each of the main modules: color and texture.

For the training of the color model 2000 images that contain skin and their masks are used, alongside 4000 images that don't contain skin, selected from the Compaq[10] dataset.

Texture uses as training data patches from the SFA dataset[52]: 3354 skin images of size 5x5 and 5590 non-skin blocks of size 5x5. From these images a total of 8944 textural features are extracted and used to train the SVM.

3.1.5   Evaluation

Dataset

The model was evaluated on the Compaq skin database in order to compare it with other approaches from literature. There is a difference however, in the number of images chosen for evaluation. While most references use a 6000 image set for testing I used only 50 images due to time constraints.

Performance metrics

Evaluation is done at the level of each image by comparing every classified pixel with the manually drawn skin map. The results for each image are then averaged to get the final metrics for the data set. The model was evaluated by the following performance indicators: True Positive Rate, False Positive Rate, Accuracy and Precision.

In order to compute those metrics we need to calculate the following values for each image:

- True Positives (TP) - the number of pixels correctly classified as skin

- False Positives (FP) - the number of non-skin pixels misclassified as skin

- True Negatives (TN) - the number of pixels correctly classified as non-skin

- False Negatives (FN) - the number of skin pixels misclassified as non-skin

The performance metrics can be described as follows:

- True Positive Rate $TPR = TP/(TP + FN)$ - also called Recall, represents the number of skin pixels detected out of the total number of skin pixels in the image

- False Positive Rate $FPR = FP/(FP + TN)$ - also called Fall-out, represents the probability of getting a false detection

- Accuracy $ACC = (TP + TN)/(TP + TN + FP + FN)$ - calculates the probability of the correct classification of a pixel

- Precision $PPV = TP/(TP + FP)$ represents the number of actual skin pixels among all pixels classified as skin

Results

Some results of evaluating the proposed model on the Compaq dataset with the parameters set in the previous section and with different threshold values can be seen in table 3.2.

Table 3.2: Evaluation of the proposed model on the Compaq dataset

Threshold	TPR	FPR	ACC	PPV
0.05	0.896	0.364	0.943	0.405
0.1	0.865	0.284	0.940	0.461
0.16	0.816	0.212	0.928	0.523
0.2	0.794	0.187	0.923	0.556
0.4	0.671	0.086	0.882	0.683

As seen in the table all runs have a rather low precision, with an average of 50%, and high false positive rate, around 20%. This is an unresolved problem in skin detection, caused by a large number of skin-like surfaces. This problem can be avoided if detection is done in a controlled environment, like the anchorperson annotation application, but this is not always possible in real world problems. Most models seen in research have a false positive rate of over 10%. Papers that I have studied, claiming a smaller false acceptance are evaluated on either unavailable or very small datasets.

Detection examples

Figure 3.6: Some detection examples. Skin is marked with a black mask.

Comparison with related work

In table 3.3 there are the results of models found in literature that were evaluated on the same dataset(Compaq). The TPR and FPR are the most common metrics used for skin model evalu-

ation.

Table 3.3: Evaluation of the proposed model on the Compaq dataset

Model	TPR	FPR
My model	81.6%	21.2%
SPM model (Jones and Rehg[10])	90%	14.2%
Mixture of Gaussians (Jones and Rehg[10])	80%	9.5%
Single Gaussian CbCr (Lee and Yoo[11])	90%	33.3%
Gaussian Mixture IQ (Lee and Yoo[11])	90%	30%
YI'Q' thresholding (Brand and Mason[18])	94.7%	30.2%
Elliptical boundary model (Lee and Yoo[11])	90%	23.3%
Segmentation and SPM (Poudel and Zhang[21])	91.17%	13.12%

My model obtains comparable results to state of the art approaches although a bit worse. Some factors which I think contributed to this difference are image resizing and using a smaller subset for evaluation.

My algorithm resizes every image to increase performance. However, this means that it must use a smaller window for texture analysis which might not be enough to get relevant textural features.

I have only used a subset of 50 images for evaluation due to its computation time, while most models use from 2000 to 6000 images. The images selected for my model could have been not very representative for the dataset.

3.2   Implementation

The application is made up out of five main modules, pictured in figure 3.7: segmentation, $color_analysis$, $texture_analysis$, evaluator and $application_gui$. The first three provide the basic image processing operations, the evaluator provides an interface for detection and performance measurements while the top module offers a visual interface for user convenience. I will present them in detail in separate sections.

Some design decisions that influenced all modules are representing images as numpy arrays (a python library for array processing) and using named tuples as keys for all maps.

Named tuples are a feature implemented in the 'namedtuple' library and are an efficient way of representing keys. The two tuples used in this application are pixel position (x,y components) and color features.

Most of the trivial image operations such as reading, writing and converting color space have been done using the 'opencv' library.

All progress messages generated by the application are handled to a logger. This can be injected at startup to display images on console, file or pass them to another process to be displayed on a GUI.
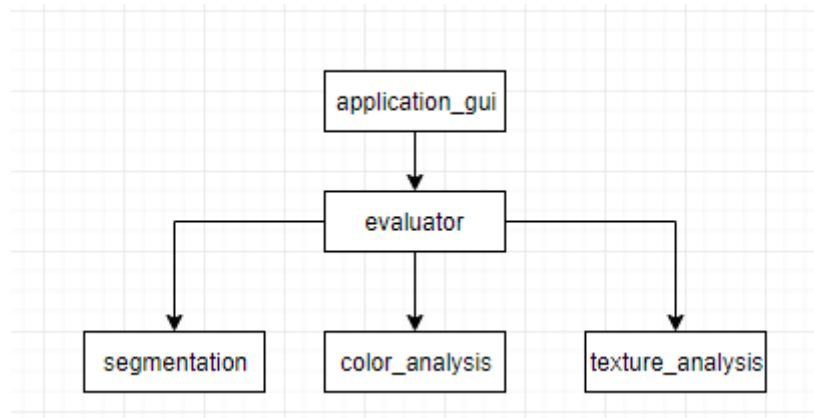
Figure 3.7: Main modules

3.2.1 The image segmentation module

An overview of this module can be seen in figure 3.8. The segmentation functionality is provided by the QuickshiftSegmentation class, which exposes two public methods, apply(image) and $get_s uperpixels(image)$, and takes the sigma and tau parameters in the constructor. The first method applies Quick shift segmentation on the image and returns an image where regions are visible, being colored with the same value as the root pixel. The second method returns a map containing all identified regions. The keys are roots pixels and the values their corresponding superpixels.

Regions are represented by the Superpixel class whose fields are: the initial image, position of the root pixel and a list of positions of all corresponding pixels.

The Distances class encapsulates some methods used by the segmentation algorithm to calculate pixel and feature distances.

3.2.2 The color detection module

This module can be divided into two submodules: training, figure 3.9, and detection, figure 3.10.

In order to train a color detection model you must call the $train\_spm\_model$ static method which takes a SPMTrainConfiguration object. The configurable fields are the database (SFA or Compaq), the color space (RGB, HSV, YCrCb) and paths for resources.

Depending on the value of the database parameter the trainer creates either a SfaComponentExtractor or a CompaqComponentExtractor. Both of these classes provide a method $extract\_components$ that returns a BayesSPMComponents object, which encapsulates: the

33
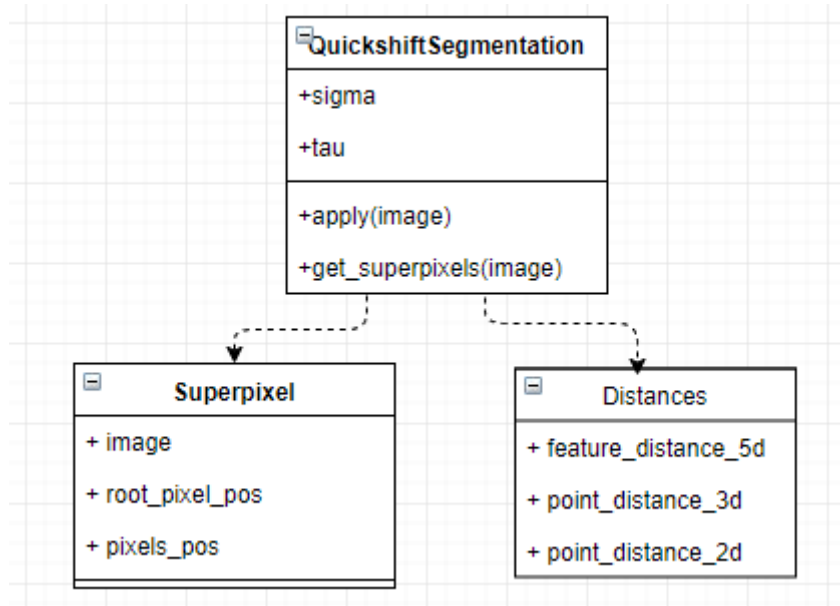
Figure 3.8: Image segmentation module

number of skin and non-skin pixels, a map with the number of apparitions for each pixel and a map with the number of apparitions as skin.

The resulting model, a SPMModel object, is serialized into a file. It contains the selected color space and the components extracted on the previous step.
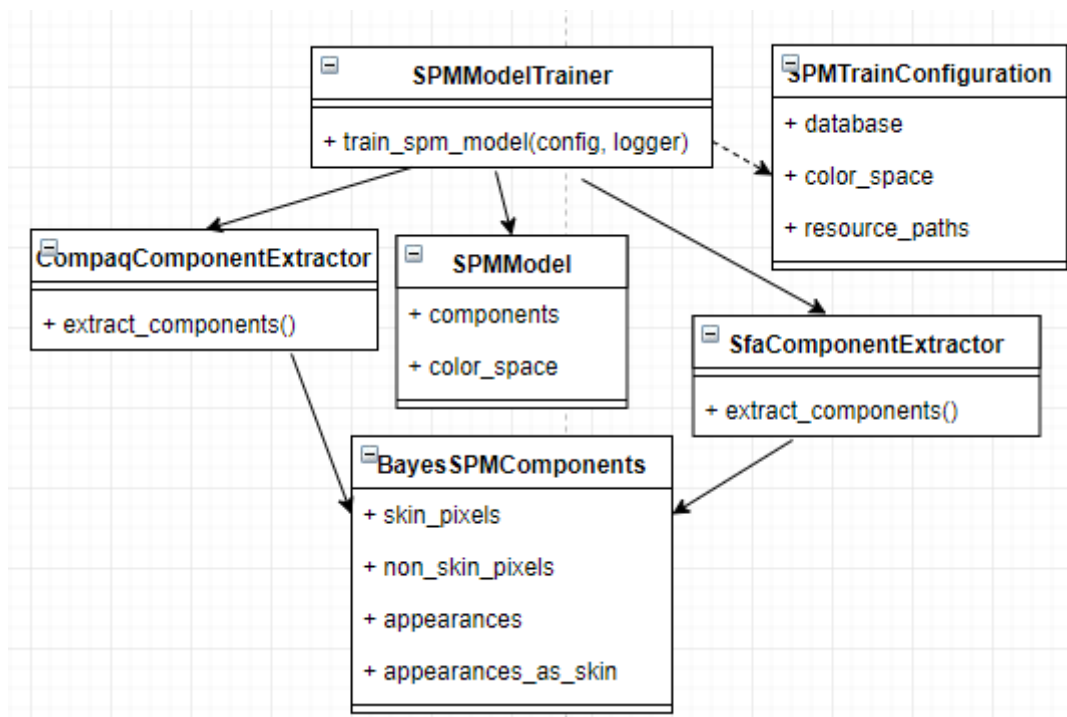
Figure 3.9: Color detector training submodule

Depending on the input parameters the SPMDetectorFactory provides one of the following detectors: SimpleDetector, NeighborDetector or AverageOnSuperpixelDetector. The first one

34

is the simplest approach as it calculates a probability for each pixel independently. The second, takes into account an area around each pixel. This simulates the histogram based approach with the difference that the considered pixel is always at the center of the area. The last detector calculates the average probability of a given superpixel.

All detectors use the methods defined in ProbabilityCalculator through an intermediary layer. This proxy, CachedProbabilityCalculator, was created to speedup operations by caching probabilities. The need for memorizing these values arose from the fact that images usually contain a small subset of pixel values duplicated many times.
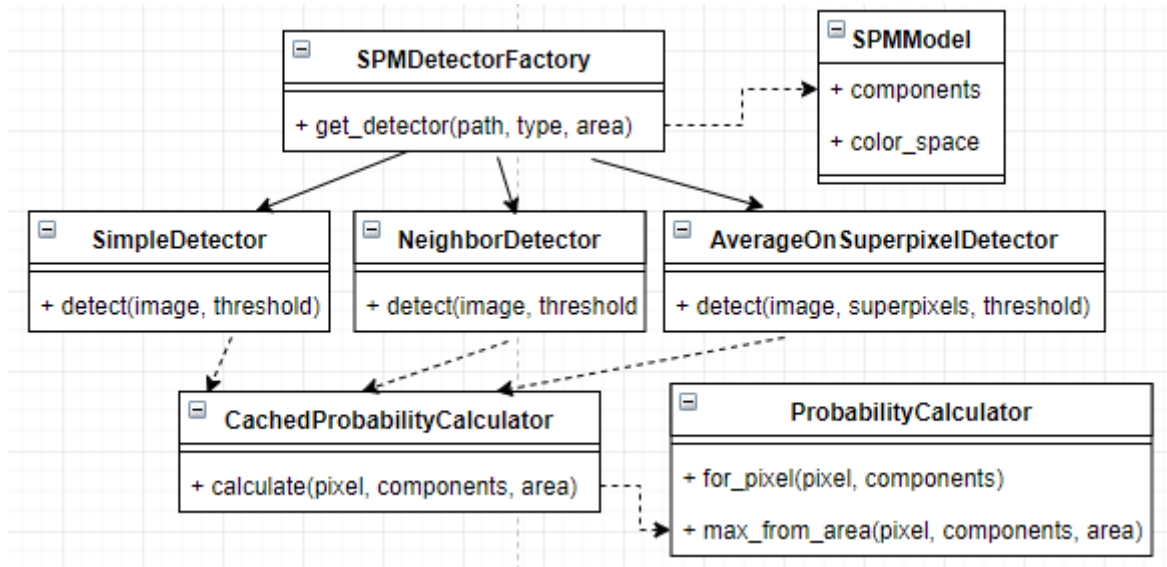
Figure 3.10: Color detection submodule

3.2.3 The texture analysis module

Similarly to the previous module, texture analysis can be divided into two submodules, one for training, figure 3.11, and one for detection, figure 3.12.

The model used in skin texture detection, TextureModel, has an attribute for extracting textural features from images (feature_extractor), one for classifying those values (classifier), and label for interpreting the classifier's results (skin_label). This model is trained by a HaralickModelTrainer which its parameters from a TextureTrainConfig object.

In this project I only implemented a model based on Haralick's features but the design allows incorporating other methods. The HaralickModel extends the basic TextureModel and overrides the feature extraction method. It is trained by a HaralickModelTrainer which injects a HaralickFeatureExtractor, which encapsulates the feature extraction method taken from the mahotas library, and SvmClassifier, which trains a Support Vector Machine classifier using the sk_learn library. The model trainer is also responsible for selecting a method of preparing the input data, in this case using the TrainDataPreparer class, which computes Haralick's features and appends the corresponding label.

35
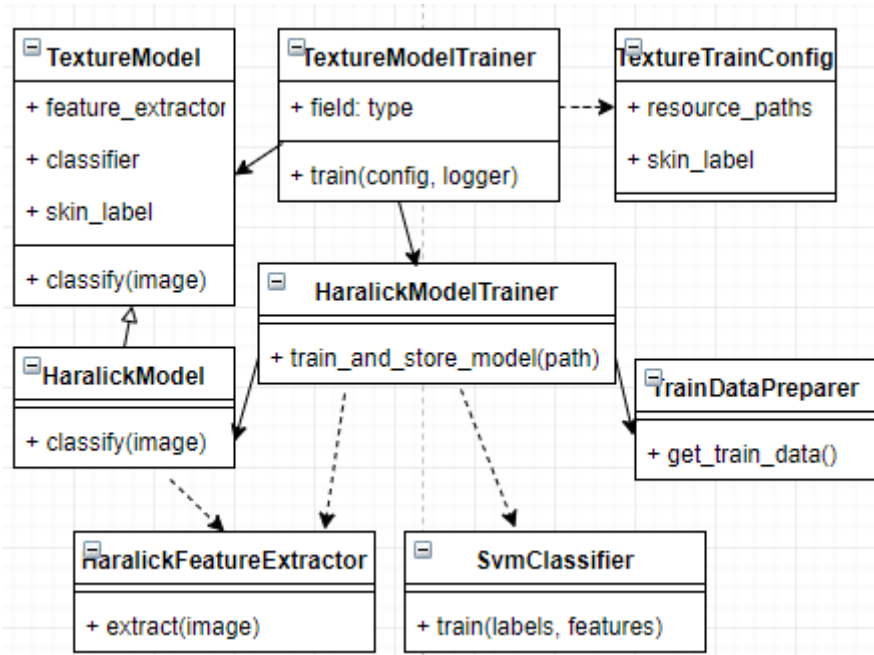
Figure 3.11: Texture detector trainer submodule

There are two types of detectors available. The GridDetector slides a window over the image and classifies each of those blocks. The PerPixelDetector is a more precise approach because it builds a window around each pixel in the image and only marks the center pixel. Although the second approach gains a noticeable increase in precision, it is much slower than the first.
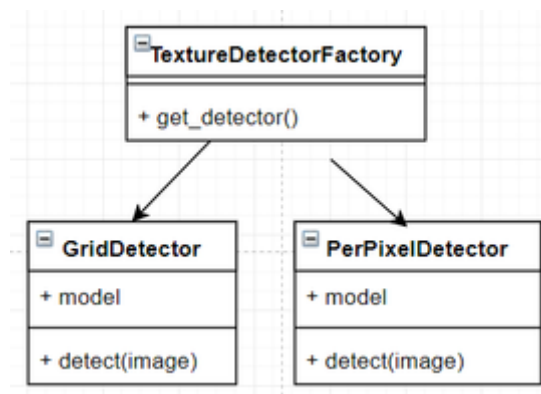

Figure 3.12: Texture analysis submodule

3.2.4   The evaluator module

The evaluator module has a simple structure, as seen in figure 3.13. The Evaluator class contains methods for evaluating a model(run_simulation) and for applying it to an image(run_detection). It takes as input a RunConfiguration object which contains informations about the desired skin and texture models and segmentation parameters.

In order to evaluate the model some quality measures are applied for each analyzed picture.

These methods include the false positive rate, true positive rate, precision, accuracy and recall and are implemented by the Stats class.
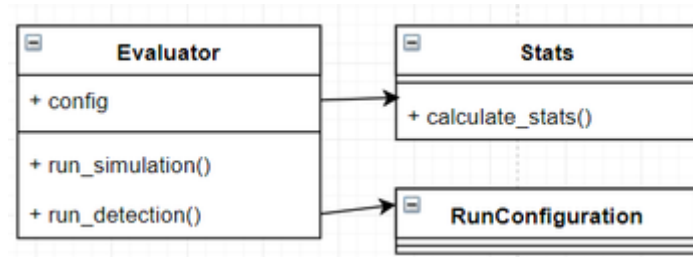
Figure 3.13: The evaluator module

3.2.5 User interface

The user interface was build using the tkinter library. This framework was designed to create modular applications. Consequently every window can be composed out of multiple frames, which are reusable components. The main views used in the application are visible in figure 3.14. The various configuration frames were left out of the diagram to avoid clutter.

MainView is a tab-view component which is provides links to the application's main windows: color model training, texture model training, model evaluation, detection of a picture.

Every component has a ProcessControlFrame which contains a FeedbackFrame and some buttons for starting and stopping operations. When the user presses the 'Start' button a new process is launched which performs the request and logs results into a queue. A monitoring thread reads that from the queue and passes it to the FeedbackFrame for displaying it on the screen. This approach allows the user to follow the application's progress while keeping the user interface decoupled from the algorithms.

The detection view also contains an ImageDisplayFrame, which displays the input and the resulting image.
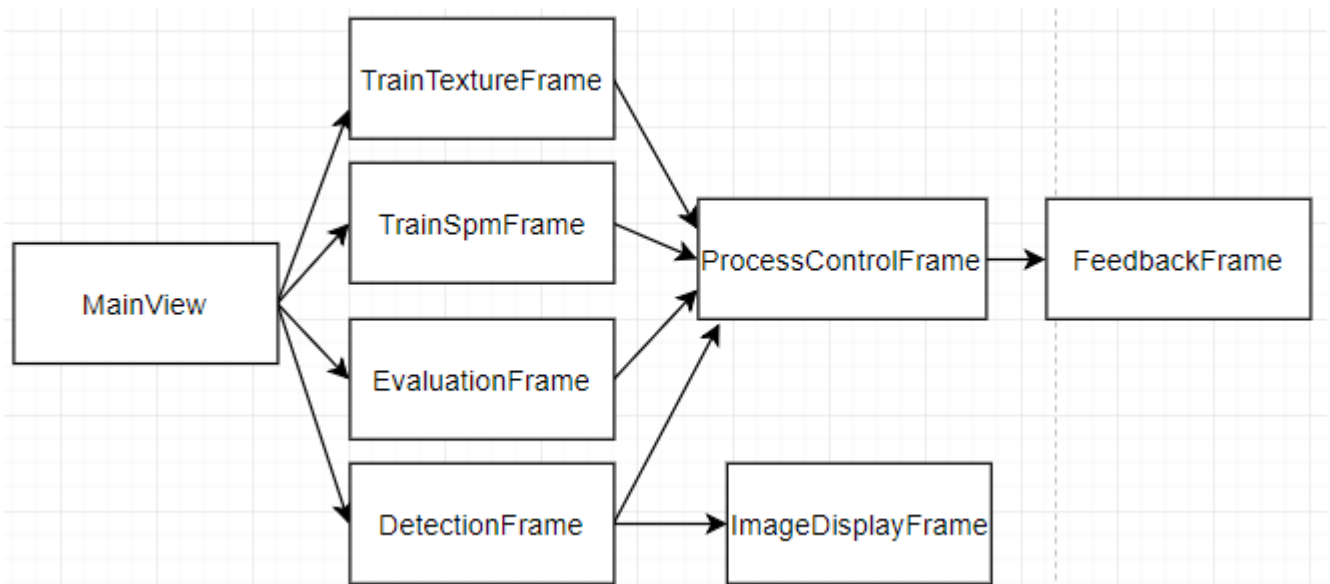
Figure 3.14: The user interface module

Chapter 4

User manual

4.1 Introduction

The goal of this application is to provide an easy-to-use and highly-configurable interface for training and evaluating skin detection models that share a core structure of image segmentation, color and texture analysis. The user can also test the created models on individual images. As the number of parameters for each task can be overwhelming the user can opt to use the default values that have already been set in each field.

The application does all computing intensive tasks in separate processes. This means the user can perform multiple operations, e.g. evaluate a model on the validation dataset and perform some single image detections on the same model, at the same time.

The application has 4 main windows accessible through a tab-view set in the top-left corner. The windows, detailed in the following sections, are the Evaluator, Train SPM, Train texture and Detection windows.
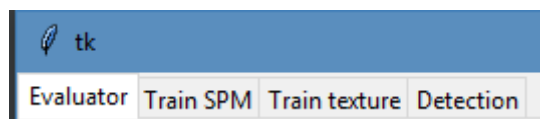
Figure 4.1: The tab-view main menu

4.2 Detection view

This view allows the user to test the model on individual images and get visual feedback. The resulting image will be displayed on the screen with a black mask that marks the identified pixels. This window provides a quick way to evaluate a model or to make a demonstration.

If you want to apply the model to a single image you must specify the detection parameters, or go with the default ones, provide a path to the input image (bottom frame on the left side of the screen) and click 'Start experiment'. The initial image and the result will be displayed on

the frames on the right side of the screen. The other configuration frames are explained in detail in section 4.6.

On the middle of the screen, below the buttons for controlling the experiment, you will see updates of the detection process. A progress bar will follow the time consuming operations: segmentation, color and texture analysis.

Figure 4.2: Detection view

4.3 Evaluator view

This screen allows the user to evaluate the performance of the final model and follow the progress in real time. The selected model can be configured in various ways which will be presented in detail in the common configurations section 4.6.

The application applies the selected model to images from the provided folders. Input images can have any size and name. Masks should have the same size and name as their corresponding input and should mark skin pixels in black (0, 0, 0). The paths where execution logs and run results are stored are also customizable.

To begin evaluation you must press 'Start experiment'. The frame below the button displays information about the currently processed picture, the operation in progress and evaluation results.

A progress bar will indicate the status of each action applied to the image. The main actions are, in order: image segmentation (consisting of density computation, neighbor linking and superpixel extraction), color detection (Bayes SPM), texture analysis (Haralick feature, combination of results and metrics calculation. Images obtained for each step (color, texture and their

Figure 4.3: Evaluator view

combination) are all saved in the results folder, to facilitate the identification of cases for which the model does not perform as expected.

After each picture is analyzed by the model, the performance metrics(true positive rate, false positive rate, true negative rate, false negative rate, accuracy, precision and recall) are displayed on the screen and saved to a file. The application builds a csv file containing metrics from each processed picture, that can be used to obtain overall results. A dump of the initial configuration is done in the file $initial_config.txt$ for documentation purposes.
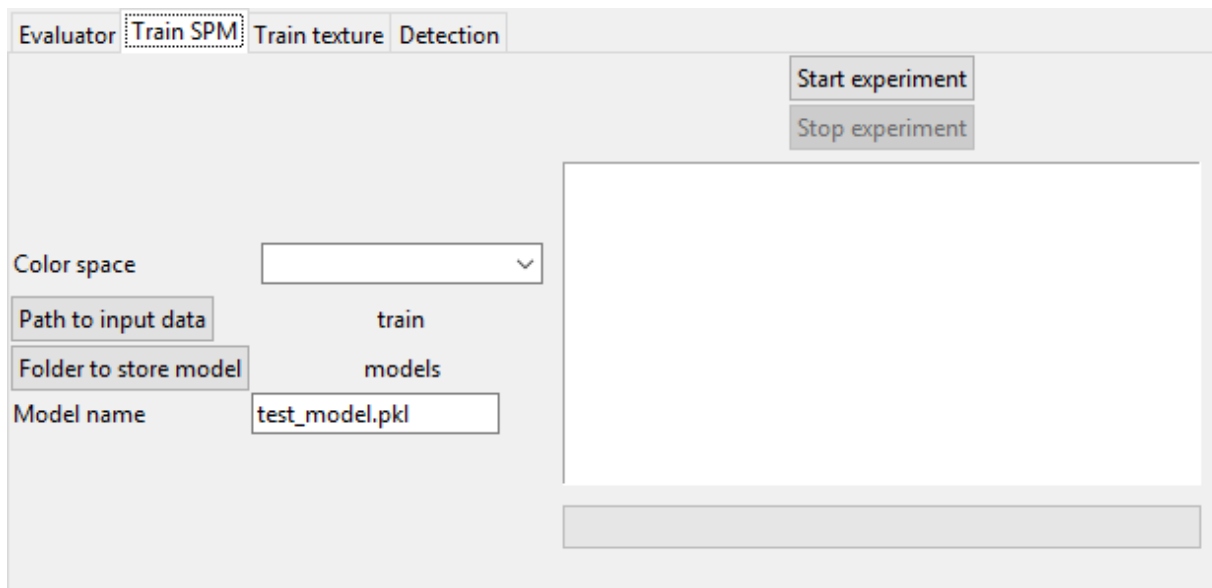
4.4   Color model training view

This window is used to train a Bayesian color detection model. You can select a color space, the path to input data (a folder which should contain the following subfolders: train_images – images containing skin, train_images_ns – images not containing skin, train_masks – masks for images containing skin), the name of the model and the path to the folder where it will be stored.

The model, consisting of a map of pixel values and their count, is stored alongside color space meta data in a .pkl file, which can be later imported by the application for evaluation.

Similarly to the other views, to start training the model you must press the 'Start experiment' button. In the frame below you will see status updates for loading images and extracting features

41

from them as well as a progress bar for each operation.

Figure 4.4: Color model training view

4.5 Texture model training view

The process of training a texture detector is simple. You must select folders with images of skin, images not containing skin and a path to store the resulting model.
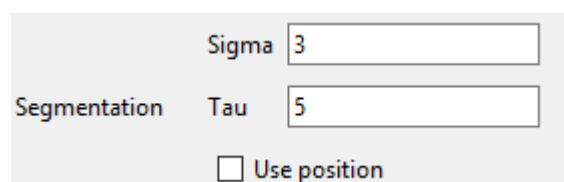
After pressing 'Start experiment' the training begins and you will see updates for feature extraction and machine learning analysis. First, the process builds a list of features and tags (skin/non-skin) from positive and negative images. Then, those features are analyzed by a SVM algorithm and the resulting model is stored in a .pkl file. That file contains the SVM's result, the serialized function for feature extraction and the tag used to identify skin texture.

4.6 Common configuration frames

Some or all of these model configuration frames are present on multiple views. For this reason they are presented in a single section.

4.6.1 Segmentation

The segmentation frame is used for tuning the parameters of the quickshift image segmentation algorithm.
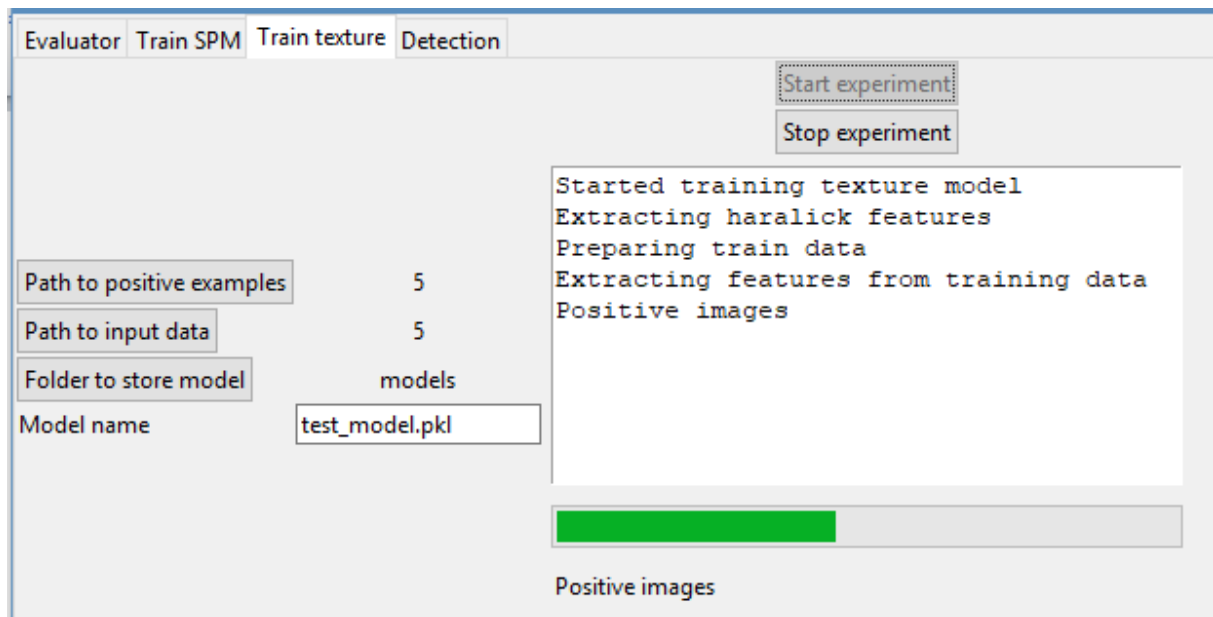

42

Figure 4.6: Segmentation

Figure 4.5: Texture model training view

Sigma controls the shape of the superpixels and Tau the size, and you can set them by typing in their adjacent textboxes.

The 'Use position' checkbox indicates whether the algorithm should consider pixel position in image as a feature for density calculation. If this box remains unchecked the density function will consider only the color space information.

4.6.2 Color detection

This frame provides configurations for the color detection module.

The SPM model is a pickle (.pkl) file generated from the Train SPM view. This indicates which color model should be selected for Bayesian probability calculations.

The detection threshold is a value between 0 and 1 which indicates at what probability to start con-

Figure 4.7: Segmentation

sidering pixels as skin. Consequently all pixels or clusters for which the model calculates a probability higher than the threshold are classified as skin pixels.

The type of the SPM classifier can be selected from the 'Type' dropdown widget. There are
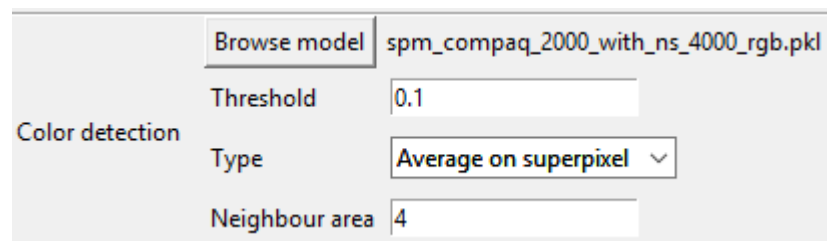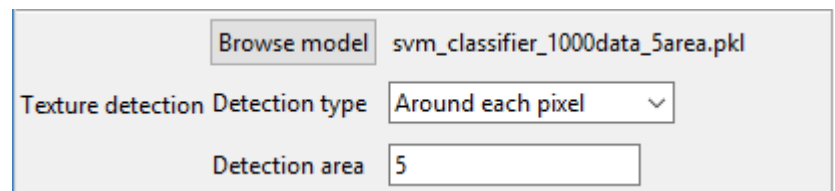
43

two types: average probability over superpixel and probability for the root of the superpixel. This affects how the probabilities are calculated for the regions identified during segmentation.

The neighbor area represents the considered neighbors during the computation of a pixel's skin probability. The final probability is taken as the maximum value obtained by exploring the color space in every direction on a distance limited by this parameter.

4.6.3 Texture detection

By clicking on the 'Browse model' button you can select the texture analysis model, which is a SVM classifier stored as a pickle file. This model is generated from the 'Train texture' window.

From the 'Detection type' dropdown you can select the texture detection strategy:

Figure 4.8: Segmentation

- Window scan, which slides a window over the image, is fast but produces jagged edges

- Around each pixel, which builds a window around each pixel, is slow but obtains smoother results

The size of these windows is set from the 'Detection area' text field.

4.6.4 Resize options

If you wish to resize the image you must check the 'Resize image' box and then specify the given parameters. This is usually done for increasing computation speed.

If you do not check the box then the image will be processed in its original size.
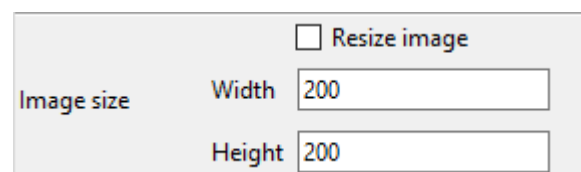
Figure 4.9: Segmentation

4.7 User Feedback

The feedback view can be seen in figure 4.10 and is present on each window. It is a text box that displays feedback from the application. It also contains the 'Start experiment' and 'Stop experiment' button which control the process associated with the view: detection, evaluation or training.

On the bottom of the frame there is a progress bar that keeps track of the current operation, for example: density calculation, superpixel extraction, SPM detection etc. That operation is also displayed in the label underneath the progress bar.
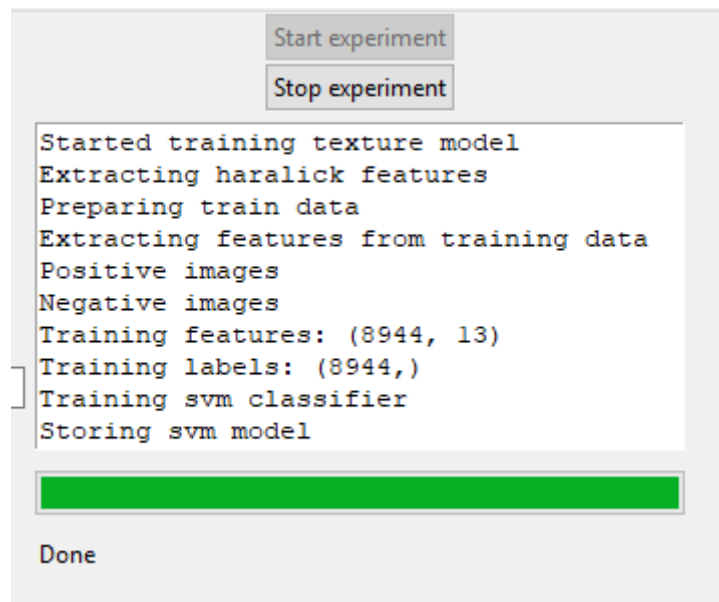

Figure 4.10: Feedback view

Chapter 5

Conclusions

texture methods are few and not well documented smth smth most use their own data sets , hard to compare and evaluate

write section

Bibliography

[1] "Internet world penetration rates by geographic regions - december 31, 2017." `https://www.internetworldstats.com/stats.htm`. Accessed: 2018-06-16.

[2] E. Martellozzo, A. Monaghan, J. Adler, J. Davidson, R. Leyva, and M. Horvath, "I wasn't sure it was normal to watch it," 01 2016.

[3] M. M. Fleck, D. A. Forsyth, and C. Bregler, "Finding naked people," in *Computer Vision — ECCV '96* (B. Buxton and R. Cipolla, eds.), (Berlin, Heidelberg), pp. 593–602, Springer Berlin Heidelberg, 1996.

[4] Y. Chan, R. Harvey, and D. Smith, "Building systems to block pornography," in *Proceedings of the 1999 International Conference on Challenge of Image Retrieval*, IM'99, (Swindon, UK), pp. 8–8, BCS Learning & Development Ltd., 1999.

[5] B. M. and R. Amsaveni, "Anchor person detection using haar-like feature extraction from news videos," *International Journal of Computer Applications*, vol. 153, no. 9, pp. 23–27, 2016.

[6] "The complete list of kinect gesture and voice commands for your referencing pleasure." `https://news.xbox.com/en-us/2013/11/26/xbox-one-kinect-gesture-and-voice-guide/`. Accessed: 2018-05-03.

[7] M. Pedraza-Hueso, S. Martín-Calzón, F. J. Díaz-Pernas, and M. Martínez-Zarzuela, "Rehabilitation using kinect-based games and virtual reality," *Procedia Computer Science*, vol. 75, pp. 161 – 168, 2015. 2015 International Conference Virtual and Augmented Reality in Education.

[8] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 24, no. 1, pp. 34–58, 2002.

[9] C. Geib, "If you jaywalk in china, facial recognition means you'll walk away with a fine." `https://futurism.com/facial-recognition-china-social-credit/`. Accessed: 2018-05-03.

[10] M. J. Jones and J. M. Rehg, "Statistical color models with applications to skin detection," *International Journal of Computer Vision*, vol. 46, pp. 81—96, 2002.

[11] J. Y. Lee and S. I. Yoo, "An elliptical boundary model for skin color detection," in *In Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology*, 2002.

[12] M. hsuan Yang and N. Ahuja, "Gaussian mixture model for human skin color and its applications in image and video databases," in *Proceedings of SPIE '99 (San Jose CA)*, pp. 458–466, 1999.

[13] J. Kovac, P. Peer, and F. Solina, "Human skin colour clustering for face detection," *EUROCON*, 2003.

[14] D. Chai and K. N. Ngan, "Face segmentation using skin-color map in videophone applications," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 9, no. 4, pp. 551—564, 1999.

[15] Y. Dai and Y. Nakano, "Face-texture model based on sgld and its application in face detection in a color scene," *Pattern Recognition*, vol. 29, no. 6, pp. 1007—1017, 1996.

[16] C.Wang and M.Brandstein, "Multi-source face tracking with audio and visual data," *IEEE MMSP*, p. 168, 1999.

[17] G. Gomez and E. F. Morales, "Automatic feature construction and a simple rule induction algorithm for skin detection," *Proceedings of Workshop on Machine Learning in Computer Vision*, pp. 31—38, 2002.

[18] Brand, J., and Mason, "A comparative assessment of three approaches to pixellevel human skin-detection," *In Proc. of the International Conference on Pattern Recognition*, vol. 1, pp. 1056–1059, 2000.

[19] L. Sigal, S. Sclaroff, and V. Athitsos, "Skin color-based video segmentation under time-varying illumination," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 26, no. 7, 2004.

[20] F. Saxen and A. Al-Hamadi, "Superpixels for skin segmentation," *Workshop Farbbildverarbeitung, At Wuppertal*, vol. 20, 2014.

[21] R. P. Poudel, J. J. Zhang, D. Liu, and H. Nait-Charif, "Skin color detection using region-based approach," *International Journal of Image Processing (IJIP)*, vol. 7, no. 4, 2013.

[22] N. K. E. Abbadi, N. Dahir, and Z. A. Alkareem, "Skin texture recognition using neural networks," *CoRR*, vol. abs/1311.6049, 2013.

[23] M. Sofiane, B. M. Chaouki, and M. B. Yamina, "Improved skin detection using colour space and texture," *International Journal of Computer and Information Engineering*, vol. 8, no. 12, 2014.

[24] Z. Jiang, M. Yao, and W. Jiang, "Skin detection using color, texture and space information," *Fuzzy Systems and Knowledge Discovery*, 2007.

[25] M. Fotouhi, M. H. Rohban, and S. Kasaei, "Skin detection using contourlet-based texture analysis," in *2009 Fourth International Conference on Digital Telecommunications*, pp. 59–64, July 2009.

[26] J. Wang and T. Tan, "A new face detection method based on shape information," *Pattern Recognition Letters*, vol. 21, no. 6, pp. 463 – 471, 2000.

[27] M. Panwar, "Hand gesture recognition based on shape parameters," in *2012 International Conference on Computing, Communication and Applications*, pp. 1–6, Feb 2012.

[28] H. Kruppa, M. A. Bauer, and B. Schiele, "Skin patch detection in real-world images," in *Pattern Recognition* (L. Van Gool, ed.), (Berlin, Heidelberg), pp. 109–116, Springer Berlin Heidelberg, 2002.

[29] L. G. Shapiro and G. C. Stockman, *Computer Vision*. New Jersey, Prentice-Hall, 2001.

[30] R. Dass, Priyanka, and S. Devi, "Image segmentation techniques," *IJECT*, vol. 3, no. 1, 2012.

[31] C. K. Chow and T. Kaneko, "Automatic boundary detection of the left ventricle from cineangiograms," *Computers and Biomedical Research*, vol. 5, no. 4, pp. 388–410, 1972.

[32] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, 1993.

[33] Y. Nakagawa and A. Rosenfeld, "Some experiments on variable thresholding," *Pattern Recognition*, vol. 11, pp. 191–204, 1979.

[34] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation – a survey of soft computing approaches," *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, 2009.

[35] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," *Computer Vision – ECCV 2008*, vol. 5305, 2008.

[36] B. Fulkerson and S. Soatto, "Really quick shift: Image segmentation on a gpu," in *In Proceedings of the Workshop on Computer Vision using GPUs, held with the European Conference on Computer Vision*, 2010.

[37] A. Frich, "Color management guide." `https://www.color-management-guide.com/color-spaces.html/`. Accessed: 2018-04-07.

[38] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," *GRAPHICON03*, pp. 85–92, 2003.

[39] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, pp. 1106—1122, 2007.

[40] J. Yang, W. Lu, and A. Waibel, "Skin-color modeling and adaptation," *ACCV98*.

[41] T. Segaran, *Programming Collective Intelligence*. O'Reilly, first ed., 2007.

[42] A. S. Andrew Ellinor, Christopher Williams, "Bayes' theorem and conditional probability." `https://brilliant.org/wiki/bayes-theorem/`. Accessed: 2018-06-11.

[43] J. McGonagle, V. Tembo, and A. Chumbley, "Gaussian mixture model." `https://brilliant.org/wiki/gaussian-mixture-model/`. Accessed: 2018-04-09.

[44] A. Materka and M. Strzelecki, "Texture analysis methods - a review," 01 1998.

[45] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610–621, Nov 1973.

[46] M. Hall-Beyer, "Glcm texture: A tutorial." https://prism.ucalgary.ca/handle/1880/51900, March 2017.

[47] W. Navas and R. Vasquez Espinosa, "Analysis of texture using the fractal model," 03 1997.

[48] S. Marĉelja, "Mathematical description of the responses of simple cortical cells∗," *J. Opt. Soc. Am.*, vol. 70, pp. 1297–1300, Nov 1980.

[49] D. Dunn, W. E. Higgins, and J. Wakeley, "Texture segmentation using 2-d gabor elementary functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 130–149, Feb 1994.

[50] S. Berisha, "Image classification using gabor filters and machine learning," 2009.

[51] S. Karamizadeh, S. M. Abdullah, M. Halimi, J. Shayan, and M. j. Rajabi, "Advantage and drawback of support vector machine functionality," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pp. 63–65, Sept 2014.

[52] J. P. B. Casati, D. R. Moraes, and E. L. L. Rodrigues, "Sfa: A human skin image database based on feret and ar facial images," *IX Workshop de Visão Computacional, Rio de Janeiro*, 2013.

[53] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PAS-CAL Visual Object Classes Challenge 2007 (VOC2007) Results." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.