Pixel based skin detection: A survey and suggestions for implementation

G. Monaci
Philips Research Europe
Abstract:
Skin detection is one of the building blocks of many image processing and computer vision algorithms. Within the Video Processing group at Philips Research we are investigating the use of skin detection algorithms for a number of core applications. In this report we focus on skin color based methods. These methods are extremely popular in the computer vision and image processing fields because they are simple and fast, they are robust to geometric variations of the skin patterns, to partial occlusions, scale or rotation changes. The aim of this report is to discuss the different aspects involved in the implementation of a pixel-based skin detection algorithm. These questions will drive the practical realization of the system, which will be discussed and evaluated at the end of the manuscript.

Conclusions:
Color-based skin detection can be a valuable feature in a number of computer vision application, as skin can be detected reliably, simply and quickly using a simple Bayesian classifier or RGB color thresholding. The detection is however not accurate enough to be used alone for most of the applications. Skin color features are indeed used in combination with texture, shape, face, motion information etc. to track and detect human body parts and then reason about human gestures, human activity recognition, sign language recognition etc. In our setting, we prefer to use the RGB color thresholding method by Kovac et al. or 2D UV histograms with 32 or 64 bins for their simplicity and flexibility, the goodness of the detection and the consistency for what concern the automatic setting of a threshold. Setting the threshold for skin detection is a delicate and non-trivial task, which deserves more attention than the one devoted to it in skin detection literature. ROC curves and AUC can sure give an idea of the theoretical capabilities of a classifier, but how to set the proper thresholds to achieve these theoretical performances is a challenging, still open issue.
# Contents

1. Introduction ............................................................................................................................................. 6
   1.1. Objective and organization of the report ....................................................................................... 6

2. Questions about skin color based methods ......................................................................................... 8
   2.1. Which color space should be used? ................................................................................................. 8
       2.1.1. (Some of the) Color spaces used for skin detection ................................................................. 8
       2.1.2. Evaluation of color spaces ...................................................................................................... 10
   2.2. How color distributions should be modeled? .................................................................................. 10
       2.2.1. Discussion of skin distribution models ..................................................................................... 11
   2.3. Which method should be used for classification? .......................................................................... 11
       2.3.1. Discussion of classification methods ......................................................................................... 12

3. Design of our skin detection system ............................................................................................... 13
   3.1. Database ........................................................................................................................................ 13
   3.2. Basic skin detection system ........................................................................................................... 14

4. Experimental results .......................................................................................................................... 16

5. Discussion and guidelines for the implementation of a skin detector .................................. 22
   5.1. Real-time skin detector at work .................................................................................................. 22

Bibliography .......................................................................................................................................... 24

Appendix A ............................................................................................................................................ 26
1. Introduction

Skin detection is one of the building blocks of many image processing and computer vision algorithms. Within the Video & Image Processing group at Philips Research we are investigating the use of skin detection algorithms for a number of core applications including:

- Interest region detection for unobtrusive sensing algorithms;
- People detection and localization;
- Activity recognition;
- Face detection and tracking;
- Hand tracking for
  - Gesture Recognition;
  - Robotic Control;
  - Other Human Computer Interaction.

We can classify skin detection methods into two main categories: Pixel-based methods and region-based methods. Pixel-based methods classify each pixel in an image as skin or non-skin individually, independently from its neighbors. Methods that detect skin-colored pixels individually (i.e. pixel-based skin color methods) fall in this category. Region-based methods take the spatial arrangement of skin pixels into account during the detection stage to enhance the methods’ performance. These methods typically combine pixel-based skin detection with some additional knowledge, like texture or contour shape (1; 2).

In this report we focus on pixel-based skin color methods. These methods are extremely popular in the computer vision and image processing fields because they have a series of appealing properties. First of all they are simple to implement, allowing fast processing. Skin color detection algorithms are robust to geometric variations of the skin patterns, to partial occlusions, scale or rotation changes. However, skin color detection in real-world images and videos can be a challenging task as the skin color is sensitive to illumination changes, skin tone variations, camera response, shadows and motion blur. Many papers investigate how to overcome these limitations by optimizing the various aspects of the color-based skin detection procedure (3; 4; 5; 6; 7; 8; 9; 10). These, and many other papers, investigate issues around the three main steps of color based skin detection systems:

- Color conversion to a color space eventually different from RGB;
- Modeling of skin, and eventually non-skin, pixel distributions in this color space;
- Classification of image pixels as skin and non skin.

Studies on skin color detection propose options for these three steps, claiming advantages for the choice of a particular color space, of a skin model, of a classifier, or of a combination of these. However, there is no definitive solution to the skin detection problem and no clear guideline for the implementation of a skin detection system.

1.1. Objective and organization of the report

The aim of this report is to discuss the different questions arising when implementing a pixel-based skin detection algorithm. These questions will drive the practical realization of the system, which will be discussed and evaluated at the end of the manuscript.

The target skin detection system should be as simple as possible, allowing it to run in real-time on a simple processing unit. We target indoor applications (e.g. videoconferencing and human-computer interaction for smart home appliances), where we assume to have full control of the acquiring camera device. This is an important assumption, since one of the most challenging issues in skin detection is the variation in illumination and capturing conditions. This is obviously a serious issue when dealing with images downloaded from internet, which have all type of content, illumination conditions. An excellent survey on illumination adaptation techniques in the skin detection pipeline can be found in (7). Since we control the camera acquiring the images, we can exploit the algorithms for white-balancing and color compensation that are nowadays more
and more effective and ubiquitously present in off-the-shelf cameras, and thus we can reasonably assume that the images we process do not have extreme color aberrations. Besides, the focus in the indoor environment also constraints the possible lighting levels that we will encounter: very low and very high light levels, as well as colored light sources, will be extremely unlikely. A series of studies focus on skin color detection under changing illumination conditions, the interested readers are referred to (11; 12).

Along this paper we will review some of the literature appeared in recent years in the field. However, this will be a limited survey of some selected, more influential papers. We invite the interested readers to refer to some excellent surveys on color spaces for skin color detection (10) and skin color modelling and detection methods (7).

The rest of the report is structured as follows: in Section 2 we discuss the most frequent question arising when implementing a skin detection system. In Section 3 we discuss the choices and the tests that we made to implement our own pixel-based skin color detection method. Section 4 shows some experimental results of the implemented system. Section 5 concludes the report discussing the results and giving some suggestions for the implementation of a skin detection system.
2. Questions about skin color based methods

As mentioned above, skin color detection is typically performed in three successive steps:

1. **Color conversion** of image pixels to a color space where skin modeling and separation between skin and non-skin pixels is supposed to improve;
2. **Modelling** of skin and eventually non-skin pixel distributions in this new color space;
3. **Classification** of image pixels into skin or non-skin categories using the models.

If one wants to implement a skin detection method following this scheme, three questions naturally arise:

- Which color space should be used?
- How skin and non-skin color distributions should be modelled?
- Which method should be used for skin classification?

In the next paragraphs we will tackle these three questions. We will review the answers to these questions found in literature, discuss them and finally evaluate, based on this evidence, which choices to make for the design of our own skin color detection system.

2.1. Which color space should be used?

In skin detection literature there is a long lasting discussion about which color space is more suitable to represent and then separate skin from non-skin color clusters. Another criterion to choose a working color space is the cost of conversion, which is critical for real-time applications. Many different color spaces have been used for skin representation: RGB, normalized RGB, YUV (or YCbCr), HIS, HSV, HSL, perceptually uniform color spaces like TSL, CIELab or CIELUV, and many others. A detailed discussion of color space selection for skin classification is out of the scope of this report; the interested reader is referred to the excellent survey (10). Here we will briefly present the principal color spaces used and discuss their advantages and disadvantages.

2.1.1. (Some of the) Color spaces used for skin detection

RGB is a natural choice to represent color images. RGB is indeed widely used in several systems and it was used in one of the first and most significant work in the field, the 2002 paper of Jones and Rehg (6). They used the Compaq dataset, 18,696 labeled skin and non-skin RGB images, to build tridimensional RGB histograms skin and non-skin color pixels. While the non-skin color distribution is concentrated around the gray axis, with sharp peaks at white and black, the skin distribution is markedly skewed toward the red side of the space, as shown in Figure 1.

![Figure 1: Contour plots for marginalizations of the skin and non-skin color models. The skin model is on the left, while the non-skin distribution is on the right (from (6)).](image-url)
Along the years researchers investigated different color spaces with the goal of finding transformations that could ensure some type of invariance (to illumination changes, shadows, highlights etc.) or a better separation between skin and non-skin color clusters. A straightforward alternative to RGB is normalized RGB (nRGB). The component of nRGB, \( r, g \) and \( b \), are obtained dividing each RGB color component by the sum of the three, i.e. \( c = C/(R + G + B) \), with \( C = \{r,g,b\} \). nRGB is a 2D color space as one component is the linear combination of the other two (e.g. \( b = I – g – r \)). nRGB is (roughly) invariant to changes of surface orientation relatively to the light source and it is used and tested in several literature works (4; 5; 8; 10).

A natural option for video processing researchers is to use the YUV or YCbCr color spaces, since these are commonly used for video compression. Thus, for real-time video applications, no color transformation is required. The transformation between YUV and RGB is linear and is reported in Appendix A. Besides, YUV separates luminance (Y) from chrominances (UV); it is thus easy to interpret the meaning of the values and to create a 2D color space removing the Y component. Performances of YUV (or YCbCr or similarly defined color spaces) are analyzed in (4; 5; 8; 3; 10; 9). Using the Matlab code reported in Appendix A, it takes about 0.37 seconds on our 2.33GHz processor to convert a 1200x1800 pixels RGB image into the YUV color space.

HSI (Hue, Saturation, Intensity), HSV (Hue, Saturation, Value), HSL (Hue, Saturation, Lightness), and the similarly defined HSB (Hue, Saturation, Brightness) are color spaces that attempt to describe perceptual color relationships more accurately than RGB. These color spaces require a non-linear transformation from RGB (reported in Appendix A) that might present problems for real-time implementations. Besides, extreme intensity values have to be discarded as HS values are instable. The main advantages of these spaces are the intuitive interpretation of the quantities used, the approximate invariance to highlights and shadows under white light sources and the possibility to simply make these spaces 2D keeping only HS components. HSV is evaluated for skin classification in (4; 5; 8; 3; 10; 9). Using the Matlab code reported in Appendix A, it takes about 0.54 seconds on our 2.33GHz processor to convert a 1200x1800 pixels RGB image into the HSI color space.

In academic research, perceptually uniform color spaces like CIELab or CIELUV are taken into account. The conversion from YUV or RGB values to these color spaces is highly non-linear (see Appendix A) and requires a complex transformation, demanding far more computation than previously discussed color spaces. In (4; 5; 8; 10; 9) the usage of perceptual color spaces for skin detection is analyzed. Using the Matlab code reported in Appendix A, it takes about 2.8 seconds on our 2.33GHz processor to convert a 1200x1800 pixels RGB image into the CIELab color space.

Table 1 summarizes the most representative color spaces commonly used for skin color detection and some of their most relevant characteristics.

<table>
<thead>
<tr>
<th>Type of color space</th>
<th>Color space (examples)</th>
<th>Transformation from RGB</th>
<th>Type of invariance</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized RGB</td>
<td>nRGB</td>
<td>Linear</td>
<td>Surface orientation</td>
<td>0.3 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Less sensible to light variations</td>
<td></td>
</tr>
<tr>
<td>Video coding color spaces</td>
<td>YUV, YCbCr</td>
<td>Linear</td>
<td>UV less sensible to light variations</td>
<td>0.37 sec</td>
</tr>
<tr>
<td>Perceived color attributes</td>
<td>HIS, HSV, HSL, HSB</td>
<td>Non-linear</td>
<td>HS roughly invariant to highlights and shadows</td>
<td>0.54 sec</td>
</tr>
<tr>
<td>Perceptually uniform color spaces</td>
<td>CIELab, CIELuv, TSL</td>
<td>Highly non-linear</td>
<td>ab less sensible to light variations</td>
<td>2.8 sec</td>
</tr>
</tbody>
</table>

Table 1: Summary of representative color spaces commonly used for skin color detection.

© Koninklijke Philips Electronics N.V. 2009
2.1.2. Evaluation of color spaces

It is relatively easy to evaluate the complexity and stability of a color conversion, being these implicit properties of the color spaces. It is more difficult to evaluate how well different color spaces work in a skin color classification task, since the final classification will depend not only on the color space itself, but also on how the color distributions are modeled and how pixels are classified. Actually, as we will see along this section, skin detection performances depend on the color space, model and classification method chosen. However, some observations can already be made.

From surveys (7; 10) and research papers it emerges that the choice of the color space has little influence in the skin detection process. In (5) the authors analyze the separability of skin and non-skin color distributions in 8 different color spaces using 4 different separability metrics. Tests were carried out using labeled images from the AR face dataset (13) and the UOPB dataset (14). The researchers conclude that color space transformations do not really help for skin classification. The same research group repeated a more complete study (8) using the same dataset where different modeling and classification schemes were included, reporting inconsistent differences between color spaces. A similar study with several color spaces, modeling and classification choices was conducted in (9) using the ECU database, leading to the conclusion that pixel wise skin segmentation is largely unaffected by the choice of color space. In (3) the authors even prove theoretically that the separability of the skin and non-skin classes is independent of the color space chosen, provided that a linear transformation between color spaces exists.

In the next sections we will see that when a modeling step is introduced, different color spaces can behave slightly differently. In particular, when parametric representations of the distributions are used or when skin is detected by thresholding color components, the choice of color space can play a role in the final results.

2.2. How color distributions should be modeled?

No matter which color space is used, two main modeling choices for the color distributions need to be considered:

1. Keep or not the illumination component: 2D vs. 3D color space;

2D vs. 3D color space: Dropping the illumination component can be beneficial for two main reasons: i) the memory required to store color distributions and the computation complexity are lower; ii) skin segmentation may become more robust to lighting variations. While the first reason appears obvious, the second argument is still debated.

Some researchers consider perfectly logical to remove the luminance component (10), while some recent works shade doubts over this common practice. In (5) the authors have shown that luminance removal decreases the separability of skin and non-skin clusters. This is obviously true because the projection of 3D data on a plane can only smear skin and non-skin classes together. However this result does not tell how well the final skin classifier will generalize in case of changing illumination. In subsequent studies (8; 9) which consider different complete skin detection systems, the removal of the luminance component reduced the performances of skin detectors. In (8) the database used is made of images from the AR and UOPB data sets, which have limited controlled lighting conditions. In this paper test and training sets are taken from the same database using cross-validation: test and training might end up having very similar illumination conditions, limiting the generalization advantages of luminance removal. In (9) the ECU database is used: this corpus is reported to consist mainly of web images with different lighting conditions, although we could not find a detailed description of the dataset content. We believe that these observations shade some doubts on the conclusions drawn by existing studies on the impact of choosing a 2D color space instead of a 3D one. We will carry a series of experiments using the database in (8) and images that we collected to verify this point.
**Parametric vs. Non-Parametric representation:** Skin and eventually non-skin color distributions can be represented either in a non-parametric form (i.e. histograms) or in a parametric form, e.g. using a Gaussian model or a Gaussian Mixture Model (GMM).

Non-parametric models using histograms are typically faster to train and to use for classification. Besides, their performances have been repeatedly shown to be independent to the color space selection (4; 5; 8; 3; 10; 9). As a drawback, they require larger storage space and they have lower generalization power, requiring thus a larger, representative training dataset. Parametric models (usually Gaussian, but cfr. (7; 10) for excellent reviews) have the useful ability to generalize incomplete training data. Besides they are represented by a small number of parameters, requiring thus little storage space. However, they can be slow both in training and for classification, and their performance depends on the shape of the skin distribution, thus on the color space and training set chosen. This is particularly annoying, because as shown by the incongruent results of literature works, this dependency is rather unpredictable (4; 5; 8; 3; 10; 9).

2.2.1. Discussion of skin distribution models

Keeping in mind that we wish to implement a skin detector that runs in real-time on a simple processing unit, we prefer to use non-parametric models. Also, they have the very useful property of being independent of the color space chosen, which translates also into higher robustness. Since we use histograms, in order to reduce the storage space required we would prefer to use 2D histograms. However from the literature it is not clear weather this choice might significantly degrade skin detection performances. In Section 3 we will compare 2D vs. 3D models in our complete skin detection system.

2.3. Which method should be used for classification?

Once the skin and eventually non-skin models are available, they can be used to classify image pixels into skin and non-skin classes.

1. **Skin color thresholding:** in these methods the skin model is explicitly defined by confining skin clusters in the chosen color space.
2. **Look-up Table:** 2D or 3D histogram skin models are used as look-up tables (LUT) to assess the probability of one pixel to be skin or not.
3. **Bayesian Classifier:** histogram models of skin and non-skin colors are used in a simple Bayesian classifier to decide whether a pixel is more likely to be skin or not.
4. **Gaussian Classifiers:** used with Gaussian models of skin and eventually non-skin color distributions.

**Skin color thresholding** is a popular method to detect skin pixels in images (9; 7; 15). Despite their simplicity, thresholding methods exhibit good skin detection rates (9). The obvious advantage of such methods is that the classification is simple and fast. However, the fixed thresholds differ for different color spaces and also for different illumination conditions. This makes difficult to find a range of threshold values that covers all skin color types and illuminations.

**Look-up Table** models are used for example in (4). A 2D or 3D histogram model of skin color, $p(c|\text{skin})$, is built storing in each color bin the number of times a particular color occurred in the training skin images. The histogram is then normalized to obtain a discrete probability distribution that represents the likelihood of any given color to be skin.

**Bayesian Classifier** is a widely used technique for skin classification because of its simplicity and speed (4; 6; 8; 9). These methods build not only a likelihood for the skin, but they also learn a likelihood for non-skin pixels. These distributions are used to build a simple skin classifier using a Bayes maximum likelihood approach (6). Using training skin and non-skin samples the likelihood distributions $p(c|\text{skin})$ and $p(c|\text{non-skin})$ are built. Given these quantities, a given pixel can be classified as skin if:

$$\frac{p(c|\text{skin})}{p(c|\text{non-skin})} > \theta,$$

(1)
where $\theta$ is a threshold.

Gaussian Classifiers are also widely used for skin classification. They provide a compact representation of skin and non-skin models, requiring thus less memory than Bayesian classifiers, and they can generalize better in case of smaller training sets. Gaussian classifiers are used when $p(c|\text{skin})$ and eventually $p(c|\text{non-skin})$ are modelled with a single Gaussian or a GMM. The likelihood $p(c|\text{skin})$ can be used alone as a measure of how likely the color $c$ is to be skin (16; 17; 8), or the classification of skin pixels can be done computing Eq. (1), as in (6).

Other classification methods have been also proposed in recent years. Because of their relative complexity, we did not consider them as candidate for the implementation of our skin detection system. The interested reader is referred to (7; 10) for excellent reviews of more advanced skin classification methods.

2.3.1. Discussion of classification methods

Skin color thresholding is very simple and fast, and it can achieve good detection rates, at the price of high false positive rates (9). However, it has limited generalization capabilities. Since we target an application scenario with controlled lighting conditions and camera settings, we do not need the flexibility required for other applications like skin detection in web images. In Section 3 we will thus test a skin detection method based on thresholding. LUT methods appear to perform poorly from existing literature (4), and Bayesian and Gaussian methods tend to be clearly preferred (7; 10). Among these, non-parametric Bayesian methods exhibit better performances in the main datasets, i.e. Compaq (6), ECU (9) and AR and UOPB (8). As already mentioned, non-parametric methods are also less sensible to color space selection and are faster to train and use. The main drawback is that they require more memory to store histograms than parametric methods. To give an idea, a 2D histogram of 32 bins per dimension requires only $2^{12} = 4k$ bytes of storage, assuming one 4 byte float per bin, but a 3D histogram of 256 bins per dimension requires $2^{26} = 67M$ bytes. However, it has to be kept in mind that these values are theoretical, since skin and non-skin histograms are typically very sparse (6), and thus storage requirements can be largely optimized. In Section 3 we will test non-parametric methods with different histogram bin sizes and considering both 2 and 3 color components for the skin and non-skin models.
3. Design of our skin detection system

Our target skin detection system should be as simple as possible in order to be implemented in real-time on a simple processing unit. Keeping these characteristics in mind, we test a skin detection architecture where different variable conditions are tested. For each step of the skin detector we test:

1. **Color conversion**: YUV, RGB and HSI.
2. **Modelling**: 2D vs. 3D histograms and histograms with 32, 64, 128 bins per dimension.
3. **Classification**: Thresholding vs. Bayesian classifier.

Concerning the **color space selection**, we tend to prefer YUV, which is the native video compression format and thus does not require any color conversion. We will compare its performance to those of RGB and HSI.

The **modeling** step appears much more delicate. As already underlined, in literature there is little agreement on how 2D vs. 3D representations perform. In the same way, few, inconclusive observations emerge from comparative studies on how to set the number of histogram bins. In (6) the authors note that a 32-bin histogram performs better than a 256-bin RGB histogram on the Compaq dataset. In (9) the authors report instead that on ECU database, finer color quantization gives slightly better performance, with 64, 128 and 256-bin histograms giving almost similar results. Jayaram et al. (8) superficially study the influence of the number of histogram bins on the final detection results. There is no clear trend in the presented data, however it appears clear that 64, 128 and 256-bin histograms perform better in the 2D case, while 32, 64 and 128-bin histograms perform better in the 3D case.

Since we want to keep the system as simple as possible, we decided to test 2D and 3D histograms with 32, 64 and 128 bins. In the RGB case, the component that we drop in the 2D case is the G component, while for YUV and HSI, we drop the “luminance” component, i.e. Y and I respectively.

Concerning the **classification** techniques, we compare the popular thresholding method proposed in (15) with the widely used Bayesian classifier. In the thresholding method, each frame is converted into RGB and single pixels are classified as skin if they satisfy the following rules:

```plaintext
% The skin color at uniform daylight illumination
R > 95 AND G > 40 AND B > 20 AND
max{R, G, B} - min{R, G, B} > 15 AND
|R - G| > 15 AND
R > G AND R > B

OR

% The skin color under flashlight or lateral illumination
R > 220 AND G > 210 AND B > 170 AND
|R - G| ≤ 15 AND
R > B AND G > B.
```

3.1. Database

The modeling step requires the learning of skin and non-skin likelihood distribution from training data. For the skin model, \( p(c|\text{skin}) \), we used the labeled data from the AR and UOPB databases collected in (5; 8). The images feature all skin color types with very diverse light conditions. Some of the images downloaded were broken, thus we ended up with about 500 face images containing about 3 millions skin pixels. Images from the data set are shown in Figure 2 (a) and (b), together with a sample ground truth image in Figure 2 (c).
To build the non-skin model, $p(c|\text{non-skin})$, we built a database made of pictures showing indoor rooms, since our target application focuses on indoor scenes. We collected 350 indoor images with all sort of content and illumination for a total of 73 million non-skin pixels. Examples of the non-skin images are shown in Figure 3.

3.2. Basic skin detection system

The basic skin detection system that we use follows the scheme shown in Figure 4. For each pixel of the input image, we compute its probability of being skin, $s = p(c|\text{skin})$, and its probability of being non-skin, $p(c|\text{non-skin})$. The ratio $r = p(c|\text{skin}) / p(c|\text{non-skin})$, together with the skin map $s$, are used to detect skin pixels: a pixel is labelled as skin if $r > \theta$ AND $s > \delta$. The first inequality expresses the Bayesian classifier, as in Eq. (1). The second inequality is added to make sure that we do not classify as skin pixels that have very low probability of being skin and even lower probability of being non-skin. The threshold $\delta$ is chosen empirically, and it is set for all the experiments to $10^{-4}$. The selection of threshold $\delta$ is delicate, as we will discuss in the following. In the experiments we measure the skin classifier performances using different thresholds $\theta$ and building the ROC curves. ROC curves and the area under curve (AUC) give an idea of the theoretical performances of a classifier, but they are not always helpful when it comes to select the threshold $\theta$ to achieve these performances. Thus, we measure the skin classification performance at one single point of the ROC curve, which is given by the value of $\theta$ automatically.
selected using Otsu’s thresholding method (18). The algorithm assumes that the image to be thresholded has two classes of pixels and computes the optimum threshold separating those two classes so that their intra-class variance is minimal. This method is one of the most popular global thresholding algorithms and it is used for thresholding skin map images in (19). Otsu’s method was chosen because from preliminary tests it exhibited better performances than two other popular thresholding methods, the entropic thresholding algorithm by Kapur et al. (20) and the unimodal thresholding algorithm (21).

![Diagram of skin detection scheme]

Figure 4: Proposed skin detection scheme.
4. Experimental results

For each tested configuration, we measured the average area under the ROC curve (AUC), and the True Positive Rate (TPR) and False Positive Rate (FPR) at one threshold, which is automatically selected using Otsu’s method. To summarize the skin classifier performance at one point, we build the quantity $\rho = \sqrt{\text{TPR}^2 + (1 - \text{FPR})^2} / \sqrt{2}$. $\rho$ is equal 1 when $\text{TPR}=1$ and $\text{FPR}=0$, and it is 0 when $\text{TPR}=0$ and $\text{FPR}=1$. All the measurements are averaged over 10 trials of 10-fold cross validation. The values of AUC for the 18 different tested conditions (3 color spaces and 6 histogram types) are shown in Figure 5 (a). The values of $\rho$ are sketched in Figure 5 (b). The complete results are summarized in Table 2.

![Figure 5](image1.png)

**Figure 5**: The values of AUC (a) and $\rho$ (b) for the 18 different tested conditions, i.e. 3 color spaces (RGB, HSI, YUV) and 6 histogram sizes (2D and 3D histograms with 32, 64 and 128 bins per dimension). Results for classifiers based on 2D histograms are on the left of the figures, highlighted in blue. The black line in (b) indicates the performance of the RGB thresholding method.

From these results, it can be observed that in general the YUV space appears to perform better. Both YUV and RGB color spaces perform better in their 2D version than in 3D, while the opposite holds for HSI. For all tested conditions, 32 and 64 histogram bins seem to work as well or even better than 128-bins histograms. Interestingly, the RGB thresholding method proposed in (15) does not lag far behind the best Bayesian classifiers. Concerning the performance at the
automatically selected threshold, we can observe that performance become less predictable, and not always the value of $\rho$ reflects the corresponding theoretical capabilities of the classifier, expressed by the AUC. This seems to confirm that the choice of the threshold is a critical one.

<table>
<thead>
<tr>
<th>Color space</th>
<th>Dimension</th>
<th>Hist. bins</th>
<th>AUC</th>
<th>True Positive</th>
<th>False Positive</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>2D</td>
<td>32x32</td>
<td>0.864309</td>
<td>0.716323</td>
<td>0.147317</td>
<td>0.787460</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td>0.843206</td>
<td>0.944674</td>
<td>0.211840</td>
<td>0.869944</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.794434</td>
<td>0.560772</td>
<td>0.077455</td>
<td>0.763398</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.884849</td>
<td>0.827143</td>
<td>0.155169</td>
<td>0.836034</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.818874</td>
<td>0.448915</td>
<td>0.016484</td>
<td>0.764470</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.653134</td>
<td>0.483577</td>
<td>0.062093</td>
<td>0.746162</td>
</tr>
<tr>
<td>HSI</td>
<td>2D</td>
<td>32x32</td>
<td>0.662365</td>
<td>0.439049</td>
<td>0.200148</td>
<td>0.645185</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td>0.673956</td>
<td>0.948758</td>
<td>0.213052</td>
<td><strong>0.871616</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.679351</td>
<td>0.423568</td>
<td>0.122868</td>
<td>0.688756</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.863009</td>
<td>0.832899</td>
<td>0.125808</td>
<td>0.853795</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.806192</td>
<td>0.422485</td>
<td>0.043997</td>
<td>0.739065</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.672452</td>
<td>0.522555</td>
<td>0.044042</td>
<td>0.770363</td>
</tr>
<tr>
<td>YUV</td>
<td>2D</td>
<td>32x32</td>
<td>0.898657</td>
<td>0.889357</td>
<td>0.229416</td>
<td>0.832093</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td><strong>0.900841</strong></td>
<td>0.954869</td>
<td>0.233334</td>
<td>0.865896</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.897723</td>
<td>0.878183</td>
<td>0.216490</td>
<td>0.832194</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.866076</td>
<td>0.928109</td>
<td>0.194047</td>
<td>0.869180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.853580</td>
<td>0.813270</td>
<td>0.185460</td>
<td>0.813905</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.801876</td>
<td>0.701293</td>
<td>0.114393</td>
<td>0.798784</td>
</tr>
<tr>
<td>Thresholding</td>
<td></td>
<td></td>
<td>0.795782</td>
<td>0.167214</td>
<td>0.814494</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: The performances of the tested 18 combinations are summarized by AUC (area under curve) and the quantity $\rho$. Higher AUC and $\rho$ values indicate better performance. For AUC and $\rho$ the highest values are highlighted in bold. The performance of the RGB thresholding method is reported in the last row.

To summarize, given these results, we tend to prefer the YUV color space since it does not require any color transformation. 2D histograms exhibit better performances than 3D ones in our setting. Considering that 2D histograms require lower storage space and computation, we definitely prefer to use 2D histograms. Our preferred skin classifiers are thus Bayesian classifiers using 32x32 and 64x64 YUV histograms.

In order to test the performance of a real skin detection system, we also carried out another series of experiments where we have learned the probabilities $p(c|\text{skin})$ and $p(c|\text{non-skin})$ using the whole database, and then we have tested the skin classifiers on some realistic images that we captured with different cameras in different indoor lighting conditions. We have used seven images where skin pixels have been manually labelled. Three of these images and the corresponding ground truth are shown in Figure 6. This test should also give indications about
the capability of generalization of the different methods, since now training and testing conditions are really different.

![Figure 6: Three test images (left) and corresponding ground truth images (right). Black pixels in the ground truth denote non skin regions.](image)

As in the previous series of experiments, we learned skin and non skin models for the 18 tested configurations. The level curves of the 2D 128-bins skin and non skin histograms in the RB, HS and UV color planes are shown in Figure 7. As expected, in all cases non skin pixel values are concentrated around achromatic plane regions, i.e. in the black and white corners in the RB plane, at $H=S=0$ in the HS plane and in the middle of the UV plane where $U=V=128$. The skin pixels instead are spread in the direction of the reddish part of the spaces, typically grouped in two or three clusters representing different skin tones and illuminations. Interestingly, the best performing color space, UV, is the one where skin and non skin clusters are more overlapping. However, it has to be noted that the main distribution peaks are well separated. This confirms the observations reported in several literature works according to which the separation of color clusters does not directly translate into skin detection performance (5; 8). Another observation that we can draw from these plots is that the single Gaussian models (8) is clearly unable to properly model the multi-modal distributions of skin colors: a GMM model is needed.
**Figure 7**: Skin and non skin distributions in the RB (left), HS (center) and UV (right) color planes. The skin distribution is drawn in red, the non skin one in black.

Figure 8 shows the values of AUC (a) and $\rho$ (b) for the 18 different tested conditions and the RGB thresholding method. The complete results are summarized in Table 3. All the values are averaged over our seven test images (Figure 6).

**Figure 8**: The values of AUC (a) and $\rho$ (b) for the 18 different tested conditions. Results based on 2D histograms are on the left of the figures, highlighted in blue. The black line in (b) indicates the performance of the RGB thresholding method.
Despite the fact that the number of test images is limited, some interesting observations can be drawn. First, the simple thresholding method performs well. This could be due to the fact that the test images are acquired in lighting conditions that are not extremely challenging; white balance and all other automatic capturing settings ensure that the image has reasonable color quality. The goodness of the thresholding method is also confirmed by qualitative observations on a number of different test images.

The second observation that we can make is that detection performances significantly degrade, both theoretically (i.e. in terms of AUC) and practically (i.e. the value of $\rho$). The methods based on 2D histograms on the UV plane are the only ones whose performance is similar to the one predicted using cross-validation (see Table 2). Also, the setting of an automatic threshold here appears more problematic than in the previous series of experiments. These behaviours have been confirmed also by qualitative observation on a number of different images. The orthogonal decoupling between the Y and UV channels seems to allow for better generalization capabilities of the skin classifiers when the test material is different from the training data.

<table>
<thead>
<tr>
<th>Color space</th>
<th>Dimension</th>
<th>Hist. bins</th>
<th>AUC</th>
<th>True Positive</th>
<th>False Positive</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>2D</td>
<td>32x32</td>
<td>0.674085</td>
<td>0.097993</td>
<td>0.033881</td>
<td>0.686654</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td>0.601205</td>
<td>0.539999</td>
<td>0.050703</td>
<td>0.772254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.507924</td>
<td>0.006702</td>
<td>0.004388</td>
<td>0.704020</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.752453</td>
<td>0.239576</td>
<td>0.027074</td>
<td>0.708513</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.583716</td>
<td>0.000102</td>
<td>0.000123</td>
<td>0.707020</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.482753</td>
<td>0.003104</td>
<td>0.001267</td>
<td>0.706215</td>
</tr>
<tr>
<td>HSI</td>
<td>2D</td>
<td>32x32</td>
<td>0.419036</td>
<td>0.259762</td>
<td>0.421690</td>
<td>0.448285</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td>0.414158</td>
<td>0.527667</td>
<td>0.043816</td>
<td>0.772244</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.421559</td>
<td>0.132264</td>
<td>0.273170</td>
<td>0.522387</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.755264</td>
<td>0.160784</td>
<td>0.019828</td>
<td>0.702349</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.555513</td>
<td>0.018710</td>
<td>0.079600</td>
<td>0.650956</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.485502</td>
<td>0.002373</td>
<td>0.021259</td>
<td>0.692077</td>
</tr>
<tr>
<td>YUV</td>
<td>2D</td>
<td>32x32</td>
<td>0.888283</td>
<td>0.867485</td>
<td>0.102336</td>
<td>0.882704</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64</td>
<td>0.865756</td>
<td>0.855974</td>
<td>0.098890</td>
<td>0.878831</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128</td>
<td>0.862515</td>
<td>0.833597</td>
<td>0.094141</td>
<td>0.870478</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>32x32x32</td>
<td>0.831378</td>
<td>0.454032</td>
<td>0.043082</td>
<td>0.748945</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>0.700900</td>
<td>0.817125</td>
<td>0.090011</td>
<td>0.864804</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128x128x128</td>
<td>0.553742</td>
<td>0.127114</td>
<td>0.014213</td>
<td>0.702828</td>
</tr>
<tr>
<td>Thresholding</td>
<td></td>
<td></td>
<td>0.747284</td>
<td>0.086033</td>
<td>0.834796</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: The performances of the tested 18 combinations are summarized by AUC (area under curve) and the value $\rho$. Higher AUC and $\rho$ values indicate better performance. The performance of the RGB thresholding method is reported in the last row of the table.

An example of the obtained skin segmentation for the test image in Figure 6 (c) is shown in Figure 9. In (a) the skin tone detection result for the RGB thresholding method is shown. Skin
pixels are reliably detected at the cost of a large number of false positive detections. In (b) the Bayesian classifier output with 32x32 UV histograms and automatic detection threshold inferred with Otsu's method is shown. In this case slightly less false detections occur, but the false detection rate is still high. This can be drastically reduced by setting the detection threshold manually, as shown in Figure 9 (c). While the skin probability map is typically accurate, finding a general, automatic rule to threshold the Bayesian classifier is far from obvious. This was confirmed also by empirical observations: we could not find any generic thresholding rule that consistently provided good results in all type of images.

Figure 9: Skin detection on the test image in Figure 6 (c). Skin pixels detected using RGB thresholding (a), Bayesian classifier with 32x32 UV histograms and automatic detection threshold (b) and manually thresholded Bayesian classifier to minimize false positive rate.
5. Discussion and guidelines for the implementation of a skin detector

To summarize, color-based skin detection can be a valuable tool in a number of computer vision applications, as skin can be detected simply and quickly using a simple Bayesian classifier or even RGB color thresholding. Our impression is that performances might vary a lot depending on the system setting (thresholds etc.), training set used and application. Based on literature works and this empirical study, we could not find a general framework applicable in all situations. However, constraining the application scenario, the skin color detection system can be made accurate and robust.

In our setting, we prefer to use the RGB thresholding method in (15) or 2D histograms with 32 or 64 bins for their simplicity and flexibility, the goodness of the detection and the consistency for what concerns the automatic setting of a threshold. Concerning this aspect, setting the threshold for skin detection is a delicate and non-trivial task, which deserves more attention than the one devoted to it in skin detection literature. ROC curves and AUC can sure give an idea of the theoretical capabilities of a classifier, but how to set the proper thresholds to achieve these theoretical performances is a challenging, still open issue. Our suggestion here is to analyze carefully the behaviour of the complete system on the target application and tune the threshold or the thresholding method to the application. An alternative option, which we implemented in our system, is to set the skin color detection threshold to a reasonable value and then let the value be adjustable at run-time by the user.

For many applications, skin color detection alone does not bring enough information. Skin color is used in combination with texture, shape, face, motion information etc. to track and detect human body parts and reason about human gestures (22; 23; 24) human activity recognition (25), sign language recognition (26) and many more.

5.1. Real-time skin detector at work

Based on these considerations we implemented a real-time skin detector with two pixel-based skin color detection methods: one is based on thresholding of RGB pixel values (15), the other is a Bayesian classifier of skin pixels that uses learned 32x32 histograms of skin and non-skin color on the UV plane. Both methods are available in the implemented system: the most appropriate one can be chosen at run-time. The thresholding method is faster and simpler, but it has more false positive detections, in particular in challenging lighting conditions. The Bayesian method requires the setting of the classification threshold, which is delicate. We decided thus to set the threshold to the standard value of 1, and then let the user interactively adjust it if needed. In order to obtain a smooth skin mask and eliminate small isolated detections, skin blobs are morphologically closed and the largest connected components of skin are detected.

Some snapshots of the system running in real-time are shown in Figure 10. Skin pixels are labelled in white and the largest skin connected components are surrounded by yellow boxes. In (a) and (c) skin pixels are detected using the RGB thresholding method, while in (b) and (d) skin is detected using the Bayesian classifier with 32x32 UV histograms and a fixed detection threshold of 1. In all cases, skin tone is reliably detected for different skin types. We empirically observed that the system works well for all type of skin and under most lighting conditions, thanks to the automatic color and white balance adjustments made by the camera. This is clearly a sub-optimal solution, as we do not have direct access to the processing carried-out inside the camera. The ideal solution would be to include an illumination compensation algorithm before the skin detection pipeline, as suggested by several works in the literature (7).

As expected, the RGB thresholding method has more true positive detections, but also false positive detections (upper left corner of Figure 10 (a)). The Bayesian classifier is more selective: it has less true positive but no false positive detections.
Figure 10: Snapshots of the skin detection system. Skin pixels are in white and the largest skin connected components are surrounded by yellow boxes. In (a) and (c) skin pixels are detected using the RGB thresholding method, in (b) and (d) using the Bayesian classifier.
Bibliography


Appendix A

In the following we report the Matlab code to convert RGB images into HSI, YUV and CIELab color spaces.

From RGB to HSI

```matlab
function hsi = rgb2hsi(rgb)
%RGB2HSI Converts an RGB image to HSI.
% hsi = RGB2HSI(RGB) converts an RGB image to HSI. The input image
% is assumed to be of size M-by-N-by-3, where the third dimension
% accounts for three image planes: red, green, and blue, in that
% order. If all RGB component images are equal, the HSI conversion
% is undefined. The input image can be of class double (with values
% in the range [0, 1]), uint8, or uint16.
%
% The output image, HSI, is of class double, where:
% hsi(:, :, 1) = hue image normalized to the range [0, 1] by
% dividing all angle values by 2*pi.
% hsi(:, :, 2) = saturation image, in the range [0, 1].
% hsi(:, :, 3) = intensity image, in the range [0, 1].
% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins
% Digital Image Processing Using MATLAB, Prentice-Hall, 2004
% $Revision: 1.4 $  $Date: 2003/09/29 15:21:54 $

% Extract the individual component images.
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);

% Implement the conversion equations.
num = 0.5*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps));
H = theta;
H(b > g) = 2*pi - H(b > g);
H = H/(2*pi);

num = min(min(r, g), b);
den = r + g + b;
en(den == 0) = eps;
S = 1 - 3.* num./den;

H(S == 0) = 0;
I = (r + g + b)/3;

% Combine all three results into an hsi image.
hsi = cat(3, H, S, I);
```
From RGB to YUV

function dst = rgb2yuv(src)

% RGB2YUV Converts an RGB image to YUV.
% This file is an adaptation of the function rgb2yuv.m available with
% the FireWire Vision Tools by Frank Wornle, available at:
% http://www.mathworks.com/matlabcentral/fileexchange/20033-firewire-
% vision-tools
% The original matrix transformation was slightly modified to match
% more closely the transformation at

% ensure this runs with rgb images as well as rgb triples
if (length(size(src)) > 2)
    % rgb image ([r] [g] [b])
    r = double(src(:,:,1));
    g = double(src(:,:,2));
    b = double(src(:,:,3));
elseif (length(src) == 3)
    % rgb triplet ([r, g, b])
    r = double(src(1));
    g = double(src(2));
    b = double(src(3));
else
    % unknown input format
    error('rgb2yuv: unknown input format');
end

% convert using the conversion matrix found at:
y = floor(0.2988*r + 0.5869*g + 0.1143*b);
u = floor(-0.168*r - 0.332*g + 0.5*b + 128);
v = floor(0.5*r - 0.4189*g - 0.0811*b + 128);

% ensure valid range for uint8 values
y(y > 255) = 255;
y(y < 0) = 0;
u(u > 255) = 255;
u(u < 0) = 0;
v(v > 255) = 255;
v(v < 0) = 0;

% generate output
if (length(size(src)) > 2)
    % yuv image ([y] [u] [v])
    dst(:,:,1) = uint8(y);
    dst(:,:,2) = uint8(u);
    dst(:,:,3) = uint8(v);
else
    % yuv triplet ([y, u, v])
    dst = uint8([y, u, v]);
end
From RGB to CIELab

```matlab
function [L, a, b] = RGB2Lab(R,G,B)
%RGB2LAB Convert an image from RGB to CIELAB
%
% function [L, a, b] = RGB2Lab(R, G, B)
% function [L, a, b] = RGB2Lab(I)
% function I = RGB2Lab(...)
%
% RGB2Lab takes red, green, and blue matrices, or a single M x N x 3
% image, and returns an image in the CIELAB color space. RGB values
% can be either between 0 and 1 or between 0 and 255. Values for L are
% in the range [0,100] while a and b are roughly in the range
% [-110,110]. The output is of type double.
%
% This transform is based on ITU-R Recommendation BT.709 using the D65
% white point reference. The error in transforming RGB -> Lab -> RGB is
% approximately 10^-5.
%
% By Mark Ruzon from C code by Yossi Rubner, 23 September 1997.
% Updated for MATLAB 7 30 March 2009.
% Available at
% http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsdscode/Util

if nargin == 1
    B = double(R(:,:,3));
    G = double(R(:,:,2));
    R = double(R(:,:,1));
end

if max(max(R)) > 1.0 || max(max(G)) > 1.0 || max(max(B)) > 1.0
    R = double(R) / 255;
    G = double(G) / 255;
    B = double(B) / 255;
end

% Set a threshold
T = 0.008856;

[M, N] = size(R);
s = M * N;
RGB = [reshape(R,1,s); reshape(G,1,s); reshape(B,1,s)];

% RGB to XYZ
MAT = [0.412453 0.357580 0.180423;
       0.212671 0.715160 0.072169;
       0.019334 0.119193 0.950227];
XYZ = MAT * RGB;

% Normalize for D65 white point
X = XYZ(1,:) / 0.950456;
Y = XYZ(2,:);
Z = XYZ(3,:) / 1.088754;

XT = X > T;
```
YT = Y > T;
ZT = Z > T;

Y3 = Y.^((1/3));

FX = XT .* X.^((1/3)) + (-XT) .* (7.787 .* X + 16/116);
FY = YT .* Y3 + (-YT) .* (7.787 .* Y + 16/116);
FZ = ZT .* Z.^((1/3)) + (-ZT) .* (7.787 .* Z + 16/116);

L = reshape(YT .* (116 * Y3 - 16.0) + (~YT) .* (903.3 * Y), M, N);
a = reshape(500 * (fX - fY), M, N);
b = reshape(200 * (fY - fZ), M, N);

if nargout < 2
   L = cat(3,L,a,b);
end