Supervised Learning Methods for Skin Segmentation Based on Pixel Color Classification

TAAN Ahmad, FAROU Zakarya

Abstract. Over the last few years, skin segmentation has been widely applied in diverse aspects of computer vision and biometric applications including face detection, face tracking, and face/hand-gesture recognition systems. Due to its importance, we observed a reawakened interest in developing skin segmentation approaches. In this paper, we offer a comparison between five major supervised learning algorithms for skin segmentation based on the color of individual pixels in images. The algorithms involved in this comparison are: Support Vector Machines (SVM), K-Nearest-Neighbors (KNN), Naive Bayes (NB), Decision Tree (DT), and Logistic Regression (LR). Various scenarios of data pre-processing are proposed including a conversion from RGB into YCbCr color space as well as duplicated records dismissal. Using YCbCr representation gave a better performance in skin/non-skin classification. Despite the settled comparison criteria, KNN was found to be the most desirable model that provides a stable performance overall the several experiments conducted.

Keywords: Supervised learning methods, Pixel-based skin segmentation, RGB color space, YCbCr color space.

1. Introduction

Skin segmentation is an essential topic that aims at segmenting regions in an image which refer to skin for further processing. It is broadly used in diverse aspects of computer vision and biometric applications, including face detection, face tracking, and face/hand-gesture recognition systems. Segmentation methods can be pixel-based where only individual pixels are examined without considering neighborhood pixels or region-based. Various image attributes can be used for skin segmentation, one of these to consider is color. Indeed, by investigating the skin dataset which is collected by a random sampling of RGB values of face images of various age groups (young, middle, and old), community groups (white, black, and Asian), and genders, we found that skin color can be solely applied to identify skin regions from its neighborhoods for a given image. Among the total amount of available examples, roughly 0.02% were perceived common amid skin and non-skin pixels. That means that color is a distinctive characteristic of skin, which will result in very high segmentation accuracy if applied judiciously.

To address this issue, we can adopt different approaches. For instance, due to the small region of the intersection between skin and non-skin in color space, it is conceivable to think of deriving mathematical formulas for both regions, then substituting RGB values with formal equations. Thereby, the categorization of skin and non-skin pixels would be simple, however, due to the complex relationship between RGB values, it might be impossible to represent skin and non-skin pixels by explicit formulas. Even though mathematical relations sound to be simpler, we observe a similar situation when transforming RGB values into YCbCr color space. Considering a novel transformation for this situation is worth to try. However, in this paper, we consider another approach where we employ machine learning (ML) or more precisely supervised learning methods (SL). As each SL algorithm possesses pros and cons and that the performance of SL algorithms depends on the concerned problem, we aim to examine the performance of five major SL algorithms for skin segmentation.

The rest of the paper is structured as follows. In Section 2, a decent review of five major SL methods, color spaces as well as related works are presented. Afterward, the dataset is explored extensively via illustrative graphics in the first part of Section 3. Then training the models for various cases are handled in the second part of Section 3 alongside the specification of the hardware and runtime environment used for the conducted experiments. Evaluation metrics and achieved results are discussed with graphics and comparative tables in Section 4. Last, but not least, in Section 5 we conclude the paper by suggesting possible future works.

2. Background

Machine learning (ML) [1] is a form of artificial intelligence that enables a system to learn from data rather than through explicit programming. As described in Figure 1, ML can be divided into three subgroups: supervised learning (SL), unsupervised learning (UL), and reinforcement learning (RL). In SL methods, both data and its desired label are used as input during the training process, which will allow the machine to learn from the data and produce a model. However, in UL methods where no labels are provided, an algorithm is applied to extract to the similarity among input data, regroup them inside a group and extract the clusters to efficiently classify new data. Other possible applications of UL would be to study the density estimation of input data which is diffused in latent space. The last type of ML is reinforcement learning, RL is a more advanced ML that allows an agent to learn in interactive conditions by trial and failure using feedback of its actions and experiences.

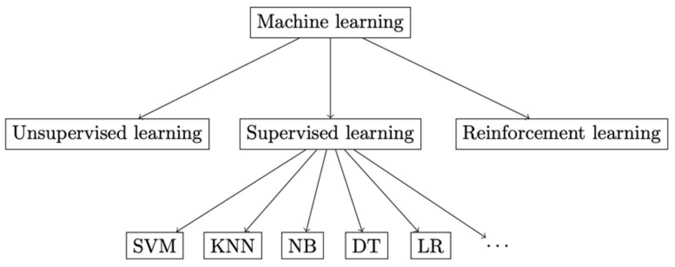

Figure 1: Types of ML

In this paper, we are considering only a subset of SL methods. The subset contains five classification techniques: Support Vector Machines, K-Nearest-Neighbors, Naive Bayes, Decision Tree, and Logistic Regression. As it is important to understand the methods that we will use during our experiments, a brief review of the methods is given in the next subsection.

2.1. Supervised Learning Methods

2.1.1. Support Vector Machine

Support Vector Machine (SVM) [2][3] is a SL algorithm proposed in the 1990s and used mainly for pattern recognition, and more often for classification problems. As described in Figure 2, a linear SVM attempts to divide a given dataset into two separate classes by finding the best hyper-plane. A hyperplane could be interpreted as a line that linearly separates and classifies a set of data in case, we are dealing with a dataset with only two features. The further away the data points are from the hyper-plane, the more reliable and accurate the model will be. We therefore want our data points to be as far away from the hyper-plane as possible, but still on the correct side of the hyper-plane.

Support vectors (instances with green color in Figure 2) are the closest elements to the hyper-plane and the most important data points of a dataset. Excluding support vectors would change the position of the dividing hyper-plane and reduce the chance of correctly classifying new data. If no data points are linearly separated, they are projected into a higher dimension. In 2D the hyper-plane was a line, therefore in 3D the hyper-plane becomes a surface. The mapping of data into a higher dimension is known as *kernelling*. [5]

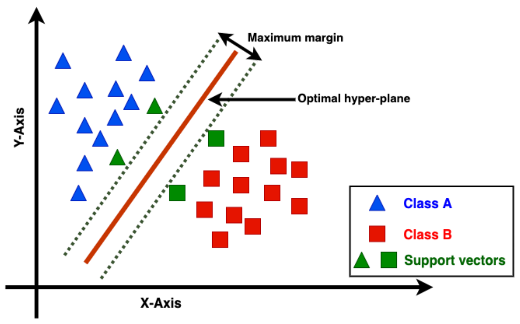

Figure 2: Linear SVM classifier with a hyper-plane

2.1.2. K-nearest Neighbors

K-nearest neighbors (KNN) is a lazy learning algorithm [3], which is often used for classification tasks. It is one of the easiest classification algorithms to understand. Despite its simplicity, it can still deliver very competitive results. KNN is based on similarity measurements between dataset instances, where *K* is a critical hyper-parameter for this algorithm, indicating how many nearest neighbors we should consider for the classification of a new instance.

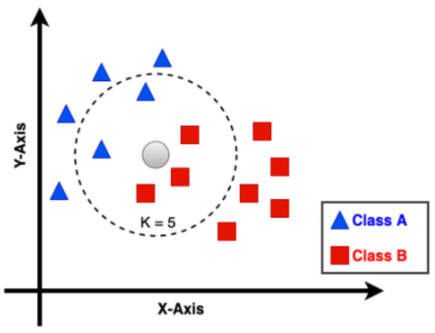

Figure 3: An Example of KNN Algorithm where *K=5*

For example, if we have two classes: squares and triangles (as described in Figure 3) and we fix *K*=5. With a new instance (circle) and based on the *K* parameter, the circle will be classified as a square, because the majority of its neighbors (in this case 3 of 5) belong to the same class. This means that the new instance is more likely to be a square than a triangle.

2.1.3. Naive Bayes

Naive Bayes (NB) [3][4] is a classification algorithm for binary (two-class) and multi-class classification problems that uses the Bayes' theorem. Bayes' theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

$$P(y|X) = \frac{P(X|y) \times P(y)}{P(X)} \tag{1}$$

Where $P(y)$ is the prior probability of y, $P(X)$ is the evidence of X, $P(y|X)$ is the probability of a class given the provided data, y is the class label, and X is a dependent feature vector of size N where:

$$X = (x_1, x_2, x_3, \dots, x_N) \tag{2}$$

The technique is the easiest to understand when described using binary or categorical input values. It is called *Naive Bayes* because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value $P(x_1, x_2, x_3|y)$, they are assumed to be conditionally independent given the target value and calculated as $P(x_1|y) \times P(x_2|y)$, and so on.

2.1.4. Decision Tree

A decision tree (DT) [3] has a structure that looks like a flowchart graph where each inner node has a threshold-based test for a specific feature, each branch represents the outcome of the test, while leaf nodes represent class labels. The entire path starting from root until reaching the leaf represents classification rules. Tree-based learning algorithms are considered to be one of the best and mostly used supervised learning methods as they offer models that are highly accurate, stable and easily interpreted compared with black-box models such as neural nets as they are very close to human logic.

2.1.5. Logistic Regression

Logistic Regression (LR) [6] is mainly used for binary classification problems, LR is a predictive analysis algorithm based on a linear regression model which uses a complex cost function. The cost function used for LR is called Sigmoid function σ, which is known as the logistic function, hence the name of logistic regression. σ is used to map predicted values (class labels) to probabilities, it means that given any real number, applying the sigmoid function on it will give a new value between [0...1]. The sigmoid function is defined as the following:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Such a classifier uses a decision boundary. A decision boundary in case of LR for binary classification problems is a fixed threshold value. Based on the threshold value, LR algorithm gives a class label to the probabilities calculated with σ. For Example, if we have 2 classes that has 0 and 1 as labels, a threshold φ, and a new instance d, we give a class label y to d by using the following rules:

$$y_d = \begin{cases} 0, & \sigma(d) < \varphi \\ 1, & \sigma(d) \geq \varphi \end{cases} \tag{4}$$

2.2. Related Works

Many researchers have addressed the problem of skin segmentation in their studies. Each providing different algorithms and aspects to deal with the problem. Some of them focused on color models to improve skin detection accuracy, whereas others argued it is not related to color model selection. For example, in [7][8][9][10] they adopted the RGB or normalized RGB color models for skin color modeling. Whereas YUV was selected in [11]. YCbCr which is adopted in our study was used also used in [12] and [13]. In [14], they conducted a comparative study between the color spaces (Normalized RGB, YCbCr, Fleck HS, CIE Lab, and HSV) for skin detection. In [15], the authors argued that color space does not contribute to skin segmentation accuracy. They tried to prove this fact in their comparison between RGB, YCbCr, and HSV color spaces. They explained that an optimal skin model can be achieved for each color space.

Apart from color spaces, diverse methodologies are present in literature for skin segmentation. Some methods rely on statistics and probability concepts, such as probability distribution functions, distance-based methods, and look-up tables. However, an attempt to build a skin Gaussian model with CgCr color space was studied in [16]. In [17], the authors used look-up tables of skin color probabilities, where mathematical calculations do not take place in the skin detection process, so they perform much faster than other methods. On the other hand, distance-based approaches need to calculate distance equations for each pixel in an image to classify pixels into skin and non-skin. This makes execution time drastically higher. Such a method was adopted by [18].

Other methods involve ML (similar to our paper), where the model is trained on the skin segmentation dataset and then tries to classify new pixels into skin and non-skin. For instance, SVM and convolutional neural networks (CNN) were adopted to detect skin regions in [20] and [19], respectively.

As a result, in this paper, we are considering the usage of various color spaces (RGB, YCbCr, and CbCr) alongside SL methods (previously described in Section 2.2.) for the classification of skin segmentation dataset [21].

2.3. Color Spaces for Skin Segmentation

Color is not an intrinsic property of objects, but a perception of another property which is an object's interaction with light photons. An object with red color emits only red wavelengths with different quantities, whether it is self-emitting or reflecting light photons. The perception of the emitted photons from objects is what is defined as color. The receptors in the eye send signals to the brain that are respective to the received wavelengths (chrominance) and their quantities (intensity of color).

A good analogy would be the perception of an object mass by its weight. An object with a mass of 1 Kilogram (real property) weighing 10 Newton (perception) when situated in the earth's gravitational field *G*, weighs less in the moon's gravitational field instead. Again, this is analogous to exposing an object to different lighting conditions which are perceived as having different colors. Consequently, careful consideration should be accounted for when dealing with color. As various color models were invented to quantize color with numerical values, each of them has different ways of describing colors. The following subsections are dedicated to the color spaces used in our paper.

2.3.1. RGB

In RGB, color is defined by three primary colors (three values) Red, Green, and Blue. Each combination of these values gives a different color. The RGB model might be the simplest to represent color, thus it is suitable for technical applications, however, it has a major drawback that its *R*, *G*, and *B* values are highly correlated. They collectively carry the intensity (luminance) and color (chrominance) information. This behavior is not perceptual hence not preferred in image classification applications.

2.3.2. YCbCr

YCbCr is one of the models that separates luminance and chrominance information. The *Y* component carries the luminance information, whereas the *Cb* and *Cr* together carry the chrominance information. The conversion from RGB to YCbCr is done by Equations 5, 6, and 7 described below.

$$Y = R\frac{77}{256} + G\frac{150}{256} + B\frac{29}{256} \tag{5}$$

$$Cb = B - Y \tag{6}$$

$$Cr = R - Y \tag{7}$$

Many skin segmentation articles used this color space, due to the separation between color and intensity information as well as its simple transformation from RGB.

3. Experiments

The goal of this study is to compare the performance of five major SL models for pixel-based skin segmentation, evaluate the effect of using different color spaces on the performance of each model and assess the effect of duplicated records on the results. To achieve this, RGB and YCbCr color spaces were adopted and different thresholding and ranking evaluation metrics were used to settle the comparison.

3.1. Dataset Specifications

The dataset is taken from the UCI machine learning repository. Skin segmentation dataset [21] has a sample size of 245057 out of which 50859 (21%) are skin samples, and 194198 (79%) are non-skin samples. Each sample comprises RGB values of a pixel taken randomly from a human face image associated with a binary label referring to either skin (1) or non-skin (0). Face images of various age groups (young, middle, and old), race groups (white, black, and Asian), and genders obtained from FERET and PAL databases were involved in forming the dataset. The dataset includes duplicated records with 51444 unique ones (28% skin and 72% non-skin). Only 11 distinct RGB tuples out of 51444 refer to both skin and non-skin at the same time.

3.2. Data Preprocessing

Having a small region of overlap between skin and non-skin implies the dismissal of redone records outwardly influencing the classification performance. First, the RGB values were turned into YCbCr, then both were visualized for two cases, with and without duplicates. Figure 4 shows the consequence of converting RGB values into YCbCr for both skin and non-skin pixels, on the left with duplicates, and on the right without duplicates. We can see that for *Cb*, and *Cr* values are focused on narrower bands than the other attributes. Additionally, removing duplicated records increases the data spread over a wider range which might be significant for some ML algorithms.

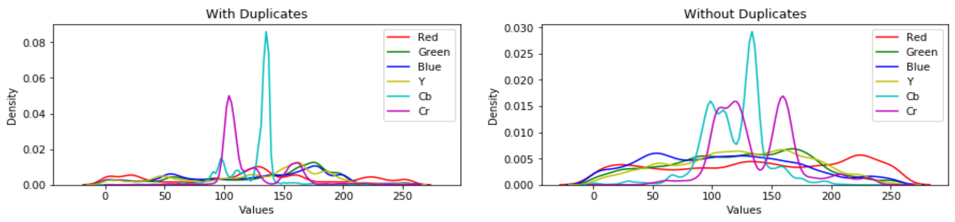

Figure 4: Density Plots of RGB And YCbCr Values of Skin and Non-skin Pixels

From Figure 5 and Figure 6, we can perceive that in YCbCr skin and non-skin values are less overlapping concerning *Cb* and *Cr* values that hold the chrominance feature (color) while in *R*, *G*, *B*, and *Y* there is no clear separation. However, considering only *Cb* and *Cr*, the overlapping happens at 2430 distinct tuples (instead of 11). So, disregarding the *Y* component will certainly increase the prediction error. This proves the fact that luminance (intensity) is also an important feature of skin and should not be disregarded if optimal performance is required. To confirm this, the experiment was repeated on different combinations of features including RGB, YCbCr, and CbCr color spaces. Also, the effect of eliminating duplicates was examined by experimenting on classifiers' performances with and without duplicates.

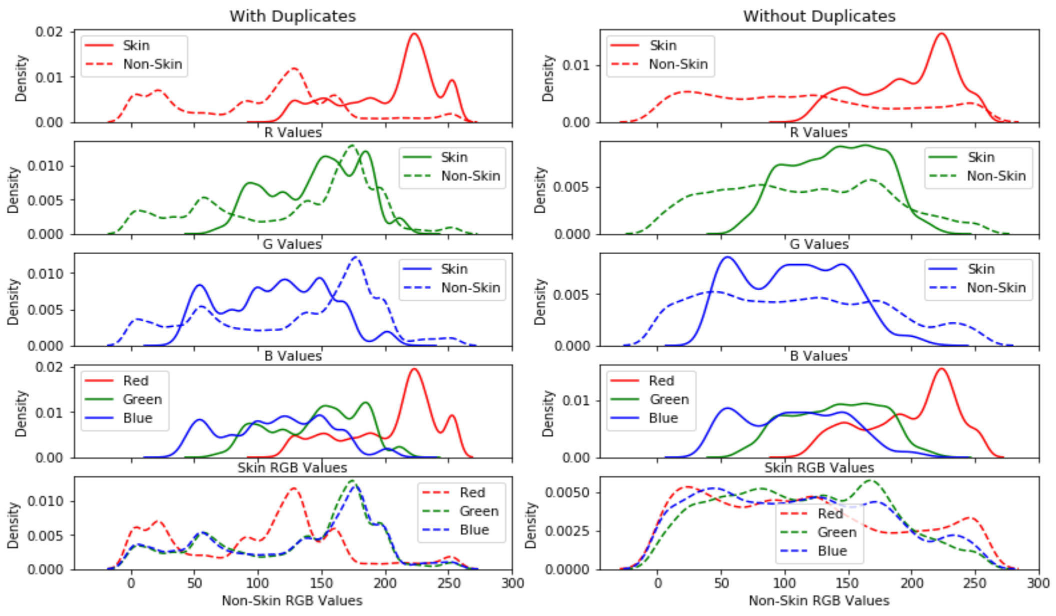

Figure 5: Density Plots of RGB Values of Skin and Non-skin Pixels

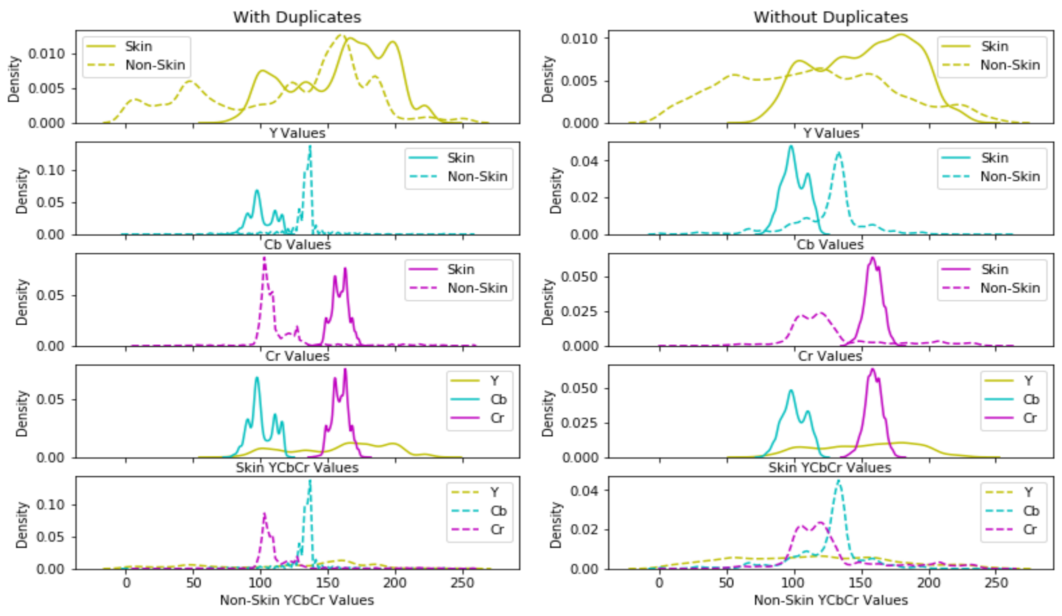

Figure 6: Density Plots of YCbCr Values of Skin and Non-skin Pixels

Figure 7 and Figure 8 show pair-plots of six attributes *R*, *G*, *B, Y*, *Cb*, and *Cr* with and without duplicated samples, respectively. The upper right triangle exhibits scatter plots of the actual data, while the lower triangle has 2D probability density plots. Again, the separation of *Cb* and *Cr* is more obvious in these figures.

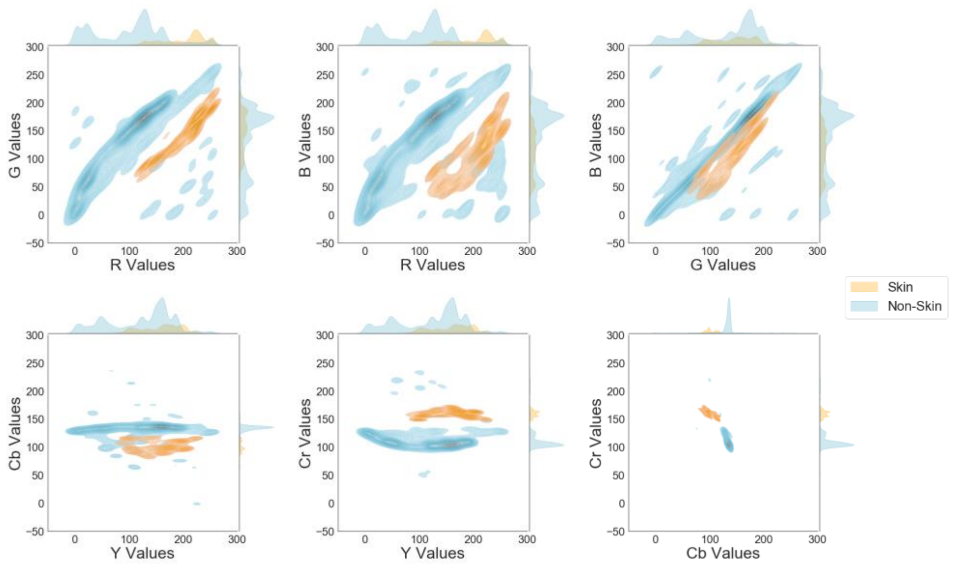

Figure 7: Pair-plots of RGB and YCbCr Values of Skin (in orange) and Non-skin (in green) Pixels Including Duplicates

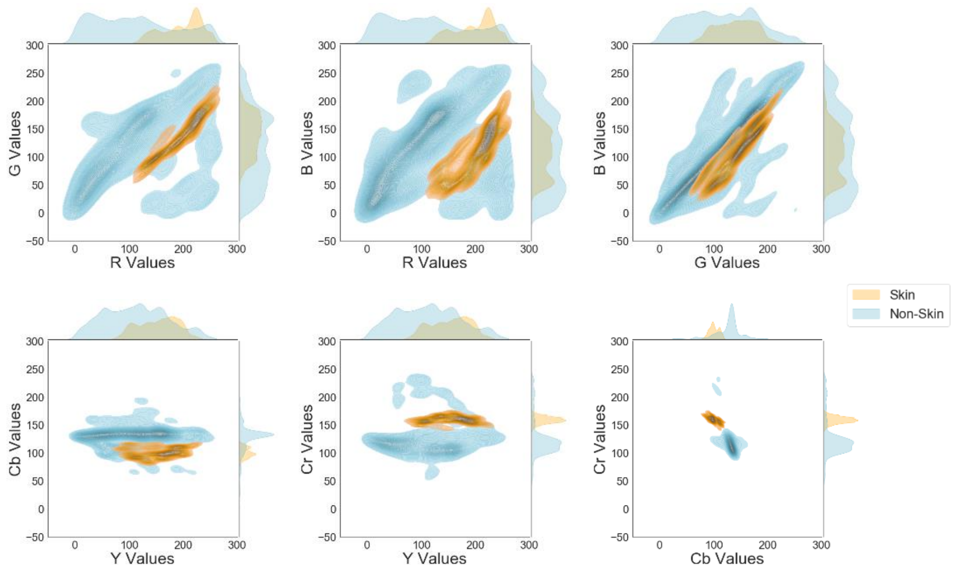

Figure 8: Pair-plots of RGB and YCbCr Values of Skin (in orange) and Non-skin (in green) Pixels Without Including Duplicates

3.3. Hardware and Runtime Environment

The experiments were conducted using HP Pavilion 14-ce1003ne Laptop under the hardware and runtime environment (RTE) described in Table 1.

Specification	Description
Processor	Intel® Core™ i5-8265U (1.6 GHz base frequency, up to 3.9 GHz with Intel® Turbo Boost Technology, 6 MB cache, 4 cores)
Memory	16 GB DDR4-2400 SDRAM (1 x 16 GB)
Storage	Crucial MX500 1TB 3D NAND SATA 2.5-inch 7mm SSD
Video Graphics	NVIDIA® GeForce® MX150 (2 GB GDDR5 dedicated)
Operating System	Windows 10 Home 64
RTE	Spyder IDE (Anaconda3) - Python 3.7.6

Table 1: Hardware and Software Environment Specifications

3.4. ML Training Configuration

For our comparison, we used known SL methods (SVM, KNN, NB, DT, and LR). The implementation of these SL methods is publicly available and can be directly imported from the *sklearn* library [22]. The details and parameter setting of each classifier is given in Table 2.

Classifier	**Algorithm used**	**Parameters**	
SVM	sklearn.svm.LinearSVC	loss=squared_hinge	*C*=1
KNN	sklearn.neighbors.KNeighborsClassifier	n_neighbors=5	*p*=2
NB	sklearn.naive_bayes.GaussianNB	var_smoothing=1e-9	
DT	sklearn.tree.DecisonTreeClassifier	criterion=gini	
LR	sklearn.linear_model.LogisticRegression	penalty=l2	

Table 2: Parameters Settings of The Classifiers Used for The Experiments

For SVM, and more specifically Linear SVC (Support Vector Classifier), *C* is used for regularization, which describes a squared *l2* penalty, while the squared hinge is used as a loss function. In KNN, *K*=5 with equal weights for neighbors and the Euclidean norm to measure the distance (*p*=2). For DT, the Gini impurity criterion was selected to measure the quality of a split. To train and evaluate SL models, the K-fold cross-validation technique is used. It is commonly applied when the sample size of the dataset is small, and to give more realistic results overall. It involves randomly splitting the dataset into *K* groups (*K*=5 in our case), keeping the same data distribution among classes inside each group. Training and evaluating the model happens *K* times by taking each group as testing data at a time while preserving all the others as training data. Eventually, scores obtained from the *K* trials are used to evaluate the model skill.

4. Experimental Results

4.1. Evaluation Metrics

After training the models, evaluation metrics are employed to compare SL methods. As our dataset is imbalanced, the use of common metrics can lead to a sub-optimal classification and might produce misleading conclusions, since these measures are insensitive to skewed domains [23]. Due to this fact, various metrics were selected to ensure a more realistic comparison. The confusion matrix is first presented as many other metrics can be extracted from it.

	Predicted as Positive	**Predicted as Negative**
Actually Positive	True Positives (TP)	False Negatives (FN)
Actually Negative	False Positives (FP)	True Negatives (TN)

Table 3: Confusion Matrix

For a binary classification problem, the confusion matrix has 4 entries (as described in Table 3). The diagonal holds the correct predictions, namely true positives (TP) and true negatives (TN), whereas false positives (FP) and false negatives (FN) are off the diagonal, they refer to the wrong predictions made by a classifier. From the confusion matrix, we can extract the classification accuracy, precision, recall, and F1-score which are illustrated in Table 4.

Metric	Definition
Accuracy (ACC)	$(TP + TN) / (TP + FP + FN + TN)$
Precision (*P*)	$TP / (TP + FP)$
Recall (*R*)	$TP / (TP + FN)$
F1-Score (F1)	$P \times R / (P + R)$

Table 4: Binary classification metrics extracted from the confusion matrix

The metrics mentioned so far are known as *threshold* metrics; they are concerned about quantifying the error in prediction. However, the ones discussed now are relevant to class separation; they are called *ranking* metrics. ROC-AUC (Receiver Operating Characteristic – Area Under Curve) is the most used metric in this regard, it is defined as the area under the curve having False Positive Rate (FPR) on the x-axis, and True Positive Rate (TPR, also known as recall) on the y-axis. TPR and FPR are defined by Equations 8 and 9, respectively.

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

The ROC curve is generated by diversifying the discrimination threshold of the model, calculating TPR and FPR for each case, and connecting the resulting points. The closer the model to the upper-left corner, the larger ROC-AUC, and the better classification will be. ROC-AUC can give confident results, in general. Though, for a severely imbalanced dataset it might be optimistic, especially if the minority class records are few. PR-AUC (Precision-Recall – Area Under Curve) focuses on the minority class, hence gives a better evaluation in our case. Like ROC-AUC, the PR curve is formed by connecting the points having precision on the y-axis versus recall on the x-axis for different thresholds. The perfect classifier resides at the upper-right corner (Figure 9).

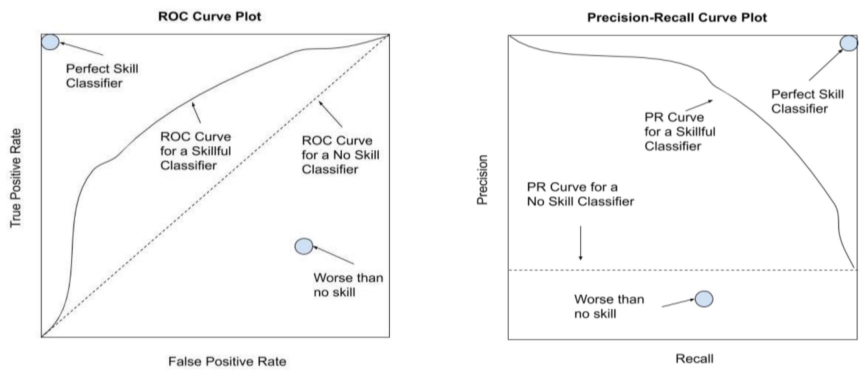
ROC Curve Plot
Perfect Skill Classifier
ROC Curve for a Skillful Classifier
ROC Curve for a No Skill Classifier
Worse than no skill
True Positive Rate
False Positive Rate
Precision-Recall Curve Plot
Perfect Skill Classifier
PR Curve for a Skillful Classifier
PR Curve for a No Skill Classifier
Worse than no skill
Precision
Recall

Figure 9: ROC and PR curves [24]

4.2. Results

Experimental results on skin segmentation dataset revealed that in most of the cases and despite the parameters and the metrics used, KNN outperformed other SL methods.

Table 5 shows the confusion matrix entries of the results for all the experiments performed. From these parameters, two threshold metrics (Accuracy and F1-score) were derived (and, if needed, others can be calculated for further comparison). Two ranking metrics (ROC-AUC and PR-AUC) were generated by diversifying the discrimination threshold. These four metrics are presented in Table 6 and depicted graphically in Figure 10. The best values (higher true predictions marked with "+" and lower false predictions marked with "-" for the confusion matrix entries) of each metric in both tables (Table 5 and Table 6) are highlighted in bold for the six cases of different color spaces and with/without duplicates.

The confusion matrix entries in Table 5 suggest that KNN and DT are the best in classifying skin and non-skin pixels. However, a closer look tells that KNN focuses on the minority class (i.e., TP and FN are actual skin pixels) whereas DT on the majority (i.e., TN and FP are actual non-skin pixels). In other words, KNN is more useful in identifying skin pixels while DT is better in detecting non-skin pixels. This is precisely the definition of the recall metric for each class. Therefore, Table 6 reveals that DT only has one occasion where it outperforms the other SL methods with an F1-score of 95.94% in the case of CbCr color space with duplicates. Furthermore, KNN still has the best overall F1-score of 98.49% concerning the case of YCbCr with duplicates. We conclude that the direct usage of confusion matrix entries is misleading in finding the best classifier.

Model	**With duplicates**				**Without duplicates**				**Color space**
	TP	**FP**	**TN**	**FN**	**TP**	**FP**	**TN**	**FN**	
SVC	16292	7422	186776	34567	751	1623	35167	13903	
KNN	**49749**$^{+}$	908	193290	**1110**$^{-}$	**14362**$^{+}$	721	36069	**292**$^{-}$	
NB	37115	5202	188996	13744	10789	3608	33182	3865	**RGB**
DT	45864	**901**$^{-}$	**193297**$^{+}$	4995	13338	**467**$^{-}$	**36323**$^{+}$	1316	
LR	40236	11585	182613	10623	8720	5333	31457	5934	
SVC	31163	11555	182643	19696	8055	6370	30420	6599	
KNN	**49999**$^{+}$	659	193539	**860**$^{-}$	**14366**$^{+}$	537	36253	**288**$^{-}$	
NB	46163	1669	192529	4696	13408	321	36469	1246	**YCbCr**
DT	46779	**220**$^{-}$	**193978**$^{+}$	4080	13673	**188**$^{-}$	**36602**$^{+}$	981	
LR	40199	11470	182728	10660	8707	5342	31448	5947	
SVC	30362	19114	175084	20497	2931	7310	29480	11723	
KNN	**47519**$^{+}$	355	193843	**3340**$^{-}$	**14010**$^{+}$	283	36507	**644**$^{-}$	
NB	46172	2194	192004	4687	13275	327	36463	1379	**CbCr**
DT	47334	**296**$^{-}$	**193902**$^{+}$	3525	13341	**239**$^{-}$	**36551**$^{+}$	1313	
LR	37600	12618	181580	13259	4959	5738	31052	9695	

Table 5: Confusion Matrix Entries for Various Cases of Color Space and Duplicate Inclusion

From Table 6, we observe that the two SL models that share the best values for all the metrics are KNN and NB (except for one occasion for DT). Hence, we will use them first to evaluate the effect of color spaces and duplicates on the skin segmentation problem. For KNN, converting from RGB to YCbCr enhances the results for all the metrics, but going from YCbCr to CbCr makes them all worse. For example, moving from RGB to YCbCr in case of including duplicates, changes the accuracy from 99.18% to 99.38%, F1-score from 97.98% to 98.49%, ROC-AUC from 99.18% to 99.40%, and PR-AUC from 98.03% to 98.39%. However, moving from YCbCr to CbCr color space changes the values back down to 98.49%, 95.85%, 97.37%, and 95.46%, respectively, which are even worse than the case of RGB color space. This suggests that intensity (*Y*) is an important feature for skin segmentation and that chrominance information (*Cb* and *Cr*) is not enough to discriminate skin. Surprisingly enough though, NB shows the same behavior when moving from RGB to YCbCr but neglecting the *Y* feature does not affect the ranking metrics, while slightly degrading performance in terms of threshold metrics. From this, we can conclude that intensity importance is relevant to the algorithm used. Nonetheless, we can say that generally YCbCr is the best color space among the three for skin segmentation.

Considering only the YCbCr now, the dismissal of duplicate records gives worse performance for KNN despite the metric that we apply. For NB, there is a slight change in values up and down for both metric types. Hence, again we can say that the importance of information held by duplicate records is relevant to the algorithm used.

Now to compare SL algorithms, by considering only the threshold metrics, KNN dominates in five cases out of six. Consequently, this leads to considering KNN as the best classifier regarding the prediction error with a maximum accuracy of 99.38% and F1-score of 98.49% both in YCbCr color space with duplicates. If ranking metrics are considered however, NB gives the best results in four cases out of six (all in YCbCr and CbCr) and maximum overall values of 99.75% and 98.51% for ROC-AUC and PR-AUC, respectively. This concludes that NB could separate class instances better than KNN did when YCbCr and CbCr color spaces were adopted. However, if we consider the case of YCbCr including duplicates, KNN gives 99.40% and 98.39% for ROC-AUC and PR-AUC, respectively, which are very close to the overall maximum values of NB. As KNN is continually leading in terms of threshold metrics, we can assume that KNN is better than NB for skin segmentation classification, and the best among all SL models considered in our paper.

Model	**With duplicates**				**Without duplicates**				**Color space**
	ACC	**F1**	**ROC-AUC**	**PR-AUC**	**ACC**	**F1**	**ROC-AUC**	**PR-AUC**	
SVC	81.95	45.91	93.34	69.07	65.14	21.59	81.84	55.07	**RGB**
KNN	**99.18**	**97.98**	**99.18**	**98.03**	**98.03**	**96.59**	**98.93**	**96.64**	
NB	92.27	77.49	93.61	84.45	85.47	73.27	93.13	85.9	
DT	97.59	93.19	94.85	90.54	96.56	93.57	94.96	90.6	
LR	90.94	77.63	94.59	70.51	78.1	59.28	85.79	63.7	
SVC	92.03	81.95	94.26	68.96	71.63	61.91	83.15	58.06	**YCbCr**
KNN	**99.38**	**98.49**	99.4	98.39	**98.4**	**97.2**	98.97	96.81	
NB	97.4	92.71	**99.75**	**98.51**	96.95	94.24	**99.6**	**98.57**	
DT	98.17	95.07	95.77	92.97	97.76	95.89	96.51	94.13	
LR	90.97	77.66	94.58	70.44	78.06	59.15	85.79	63.68	
SVC	86.67	72.45	94.48	67.18	67.17	39.98	83.72	59.14	**CbCr**
KNN	**98.49**	**95.85**	97.37	95.46	**98.2**	**96.78**	98.46	96.71	
NB	97.19	92.21	**99.75**	**98.55**	96.68	93.8	**99.62**	**98.47**	
DT	98.44	95.94	96.7	94.4	96.98	94.46	95.69	92.79	
LR	89.44	73.88	94.05	67.89	70	33.78	84.74	62.02	

Table 6: Summary of Experimental Results for Skin Segmentation Dataset

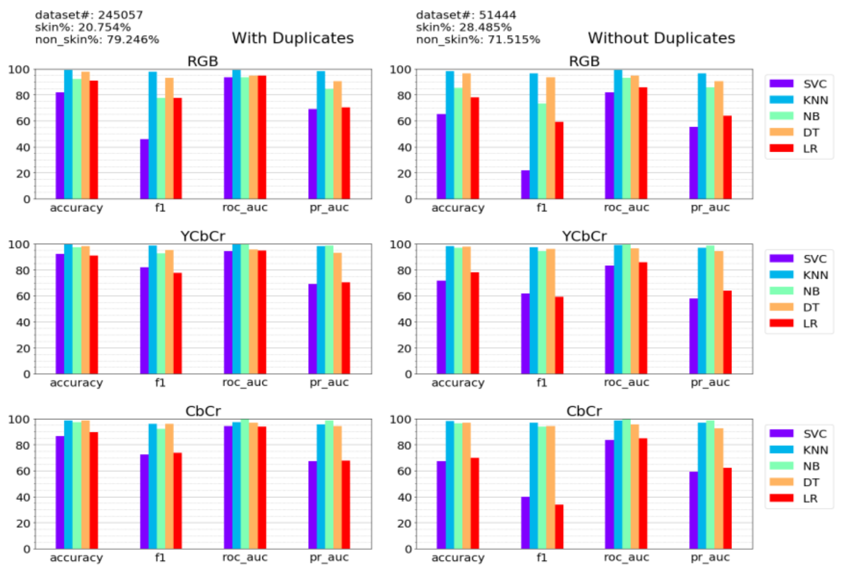

Figure 10: Comparative Graphical Representation of The Achieved Results by SL Methods for Skin Segmentation Dataset Classification

Nonetheless, if execution time is crucial, which is the case in real-time applications, then opting for NB is more beneficial. Indeed, based on Table 7 and Figure 11 which indicate the execution time for each classifier (best results are marked with bold), we can quickly discern that KNN takes much more time than any other SL method. The reason of this phenom is clear, KNN algorithm measures the distance between a new instance and all existing instances, stores the indexes and the distances in a collection, sorts the collection in ascending order, selects the class label of the *K* first records from the collection, and finally applies the mod of the *K* class labels as a label for the new instance. Those steps are repeated for each instance belonging to the test set, which makes the execution time of KNN very long as opposed to other SL methods.

With duplicates					Without duplicates					Color space
SVC	**KNN**	**NB**	**DT**	**LR**	**SVC**	**KNN**	**NB**	**DT**	**LR**	
0.239	2.562	0.223	**0.211**	0.218	0.048	0.424	0.047	**0.036**	0.039	**RGB**
0.226	2.534	0.188	**0.167**	0.179	0.045	0.401	0.045	0.042	**0.041**	**YCbCr**
0.197	4.235	0.188	0.196	**0.178**	0.048	0.361	0.048	0.037	**0.035**	**CbCr**

Table 7: Summary of Experimental Results in Terms of Execution Time (sec)

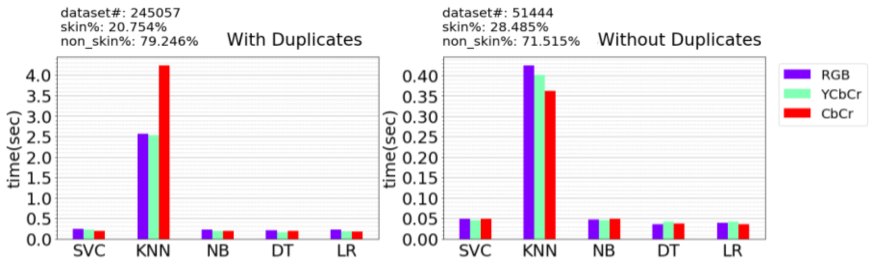

Figure 11: Comparative Graphical Representation of The Achieved Results by SL Methods for Skin Segmentation Dataset in Terms of Execution Time

5. Conclusion

In this study, we addressed the skin segmentation problem based on pixel color using SL methods. We aimed to find the best color space among RGB and YCbCr for this purpose, we investigated the effect of the intensity (*Y*) on skin segmentation classification, we also evaluated the effect of duplicate records dismissal on the classification results, and eventually, we found the best SL model for skin segmentation classification considering all the aspects mentioned above.

In all the cases, YCbCr was perceived to be better than RGB. The intensity (*Y*) importance was found to be relevant to the algorithm used, for instance, KNN performed better considering the intensity (*Y*), but NB was almost not influenced by it. Duplicates' influence was also found to be relevant to the used algorithm. KNN performed better with duplicates, and NB almost gave even results. The best SL model that outperformed all the other SL models in almost all the cases considered is KNN. Despite this, evaluation metrics were found crucial for finding the superior SL model. In the case of threshold metrics, KNN outperformed all the other classifiers, still, NB achieved commending results while considering the ranking metrics. As a result, metrics should be chosen carefully for each specific application. Execution time is also an important factor in the decision of the best model, the KNN algorithm was the worst in this regard.

For future works, we suggest focusing on a specific skin segmentation application and trying to find the best evaluation metric that suits the application in concern then making the comparison again. Adopting other SL methods such as artificial neural networks (ANN), convolutional neural networks (CNN), or deep neural networks (DNN) will be examined. Also, the inclusion of other color spaces that separate intensity and color information is of great interest. Finally, experimenting on a different dataset will guarantee more robust results. For example, the ECU dataset [25] which consists of 4,000 color images with ground-truth prepared manually for skin segmentation and face detection. It is the largest dataset available with 209 million skin pixels and 902 million non-skin pixels.

Acknowledgement

This project has been supported by Telekom Innovation Laboratories (T-Labs), the Research and Development unit of Deutsche Telekom and the European Union, co-financed by the Social Fund. EFOP-3.6.3-VEKOP-16-2017-00001

Bibliography

1. Alpaydin, E. (2020). *Introduction to machine learning.* MIT press.
2. Zhang, D. (2019). Support vector machine. In *Fundamentals of Image Data Mining* (pp. 179-205). Springer, Cham. DOI: 10.1007/978-3-030-17989-2_8
3. Sen P.C., Hajra M., Ghosh M. (2020) Supervised Classification Algorithms in Machine Learning: A Survey and Review. In: *Mandal J., Bhattacharya D. (eds) Emerging Technology in Modelling and Graphics.* Advances in Intelligent Systems and Computing, vol 937. Springer, Singapore. DOI: 10.1007/978-981-13-7403-6_11

4. Brownlee, J., 2020. *Naive Bayes Classifier From Scratch In Python*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python [Accessed 8 April 2020].

5. Hofmann, M. (2006). Support vector machines-kernels and the kernel trick. *Notes*, *26*(3).

6. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media. DOI: 10.1007/978-0-387-21606-5

7. Bergasa, L. M., Mazo, M., Gardel, A., Sotelo, M. A., & Boquete, L. (2000). Unsupervised and adaptive Gaussian skin-color model. *Image and Vision Computing*, *18*(12), 987-1003, ISSN 0262-8856. DOI: 10.1016/S0262-8856(00)00042-1

8. Siddiqui, K. T. A., & Wasif, A. (2015). Skin detection of animation characters. *arXiv preprint arXiv:1503.06275*.

9. Hajraoui, A., & Sabri, M. (2014). Face detection algorithm based on skin detection, watershed method and gabor filters. *International Journal of Computer Applications*, *94*(6), 33-39. DOI: 10.5120/16349-5695

10. Chen, W. C., & Wang, M. S. (2007). Region-based and content adaptive skin detection in color images. *International journal of pattern recognition and artificial intelligence*, *21*(05), 831-853. DOI: 10.1142/S0218001407005715

11. Vadakkepat, P., Lim, P., De Silva, L.C., Jing, L., Ling, L.L. (2008, March). Multimodal approach to human-face detection and tracking. *IEEE Transactions on Industrial Electronics* 55 (3): 1385-1393. ScholarBank@NUS Repository. DOI: 10.1109/TIE.2007.903993

12. N. Brancati, G. De Pietro, M. Frucci, L. Gallo (2017). Human skin detection through correlation rules between the YCb and YCr subspaces based on dynamic color clustering. *Computer Vision and Image Understanding*, vol. 155, pp. 33-42. DOI: 10.1016/j.cviu.2016.12.001

13. Kumar, C. R., & Bindu, A. (2006, November). An efficient skin illumination compensation model for efficient face detection. In *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics* (pp. 3444-3449). IEEE. DOI: 10.1109/IECON.2006.348133

14. Zarit, B. D., Super, B. J., & Quek, F. K. (1999, September). Comparison of five color models in skin pixel classification. In *Proceedings International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems. In Conjunction with ICCV'99 (Cat. No. PR00378)* (pp. 58-63). IEEE. DOI: 10.1109/RATFG.1999.799224

15. Albiol, A., Torres, L., Bouman, C. A., & Delp, E. (2000, September). A simple and efficient face detection algorithm for video database applications. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)* (Vol. 2, pp. 239-242). IEEE.

16. Ghazali, K. , Ma, J. , Xiao, R. , & lubis, S. (2012). An Innovative Face Detection Based on YCgCr Color Space. *Physics Procedia*, 25. DOI: 10.1016/j.phpro.2012.03.358

17. Nadian-Ghomsheh, A. (2016). Pixel-based skin detection based on statistical models. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *8*(5), 7-14. ISSN: 2180–1843, e-ISSN: 2289-8131.

18. Störring, M., Kočka, T., Andersen, H. J., & Granum, E. (2003). Tracking regions of human skin through illumination changes. *Pattern Recogn. Lett.* 24, 11 (July 2003), 1715–1723. DOI: 10.1016/S0167-8655(02)00327-6

19. Y. Kim, I. Hwang and N. I. Cho (2017). Convolutional neural networks and training strategies for skin detection, *IEEE International Conference on Image Processing (ICIP)*, Beijing, 2017, pp. 3919-3923, DOI: 10.1109/ICIP.2017.8297017

20. J. Han, G. Awad and A. Sutherland (2009, March). Automatic skin segmentation and tracking in sign language recognition. in *IET Computer Vision*, vol. 3, no. 1, pp. 24-35. DOI: 10.1049/iet-cvi:20080006

21. Bhatt, R., &amp; Dhall, A. (2012, July 7). Skin Segmentation Dataset, UCI Machine Learning Repository Machine Learning by Andrew Ng at Coursera. Retrieved April 1, 2020, from https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation.

22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

23. Luis Torgo, P. B., & Ribeiro, R. (2016). A survey of predictive modeling under imbalanced distributions. *ACM Comput. Surv*, *49*(2), 1-31. DOI: 10.1145/2907070

24. Brownlee, J. (2020, January 14). Tour of Evaluation Metrics for Imbalanced Classification. Retrieved April 26, 2020, from https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification.

25. Son Lam Phung, Bouzerdoum, A., & Chai, D. (2005). Skin segmentation using color pixel classification: Analysis and comparison, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (pp. 148-154). IEEE. DOI: 10.1109/TPAMI.2005.17

Authors

TAAN Ahmad

Eötvös Loránd University, Faculty of Informatics, Department of Data Science and Engineering, Hungary,
e-mail: ahmadtaan91@gmail.com.

FAROU Zakarya

Eötvös Loránd University, Faculty of Informatics, Department of Data Science and Engineering, Telekom Innovation Laboratories (T-labs), Hungary,
e-mail: zakaryafarou@inf.elte.hu.

About this document

Published in:

CENTRAL-EUROPEAN JOURNAL OF NEW TECHNOLOGIES IN RESEARCH, EDUCATION AND PRACTICE

Volume 3, Number 1. 2021

ISSN: 2676-9425 (online)

DOI:

10.36427/CEJNTREP.3.1.779

License