Universal Skin Detection Without Color Information

Jia Khot
Computer Science and Engineering
Shiv Nadar Institution of Eminence
Delhi NCR, India
jk637@snu.edu.in

Lalit Maurya
Computer Science and Engineering
Shiv Nadar Institution of Eminence
Delhi NCR, India
lm742@snu.edu.in

Rachit Kumar
Computer Science and Engineering
Shiv Nadar Institution of Eminence
Delhi NCR, India
rk551@snu.edu.in

***Abstract*—This project implements the methodology proposed in [1] Universal Skin Detection Without Color Information, aiming to detect human skin regions from gray-scale images without relying on color cues. Due to the absence of a suitable pre-existing dataset aligned with the paper's requirements, a custom dataset was constructed, encompassing a range of illumination conditions, skin tones, and image resolutions. The system leverages face detection as a spatial prior, local texture features (such as Local Binary Patterns and lacunarity), and gray-scale statistics to perform adaptive region growing for skin segmentation. Extensive hyper-parameter tuning was conducted to optimize detection precision and recall under varying.**
Github Repository: https://github.com/icingtea/skin_detection

I. Introduction

Skin detection is a foundational task in many computer vision applications, including face detection, gesture recognition, surveillance, and content filtering. Traditional methods rely heavily on color information, exploiting the relatively distinct distribution of skin tones in various color spaces. However, color-based techniques struggle under conditions such as poor illumination, legacy gray-scale imagery, or near-infrared imaging, where color cues are either unreliable or completely absent.

The work by Sarkar et al. [1] addresses this challenge by proposing a novel framework for detecting human skin without relying on color information. Their method leverages geometric priors from face detection to identify initial skin regions and employs local texture descriptors—such as Local Binary Patterns (LBP) and lacunarity—as well as gray-scale intensity statistics to expand the skin region through adaptive region growing.

This project implements the methodology proposed in [1] in Python. Due to the lack of a suitable existing dataset tailored to the paper's requirements, we curated a custom dataset capturing diverse skin tones, illumination conditions, and image qualities. Our implementation involved extensive hyper-parameter tuning to achieve robust performance across different scenarios.

II. Dataset Acquisition

A. Relabeled HELEN Dataset

The Relabeled HELEN Dataset, curated by JPlin [2], is an enhanced version of the original HELEN facial image dataset. It comprises 2,330 high-resolution facial images, each accompanied by meticulously annotated segmentation masks. These masks delineate various facial components, including skin, eyes, eyebrows, nose, lips, and jawline. The relabeling effort aimed to improve the accuracy and consistency of the annotations, making the dataset more suitable for tasks such as facial mask generation.

B. Skin Mask Extraction Process

To utilize the Relabeled HELEN Dataset for skin detection, we implemented the following processing pipeline:

1) **Annotation Parsing**: The dataset provides segmentation masks where each pixel is labeled according to its corresponding facial component. We parsed these masks to identify and isolate the skin regions.
2) **Ensuring Solo Subjects**: Since the dataset contains images of multiple people, we used a prebuilt face detector to discard any images with multiple people.
3) **Mask Generation**: Using the parsed annotations, we generated binary skin masks for each facial image. Pixels labeled as skin were assigned a value of 1, while all other pixels were set to 0.
4) **Validation**: The generated skin masks were visually inspected to ensure correctness. Any discrepancies identified during this process were corrected manually to maintain the quality of the dataset.

III. Skin Detection Using Contextual Information

A. Intensity based skin pixel prediction

We use a face detector provided in [3] to extract landmarks which are used to create a face mask. This mask is carefully selected to ensure minimal overlap with non-skin features (e.g, nostrils, facial hair, accessories). We use a different skin mask than what was suggested in [1]. We saw better and more consistent results with the modified skin mask.

The positioning of this skin mask gives us insight into skin pixel intensities due to the symmetry of a human face. These intensities are used to construct a histogram.

$$P(I \mid s) = \frac{n(I)}{N_s} \tag{1}$$

$P(I \mid s)$ represents the conditional probability of occurrence of intensity value $I \in \{0, 1, 2, ..., 255\}$ given that it is a skin pixel. $n(I)$ is the number of pixels of intensity I, and N_s represents the total number of pixels in the mask.

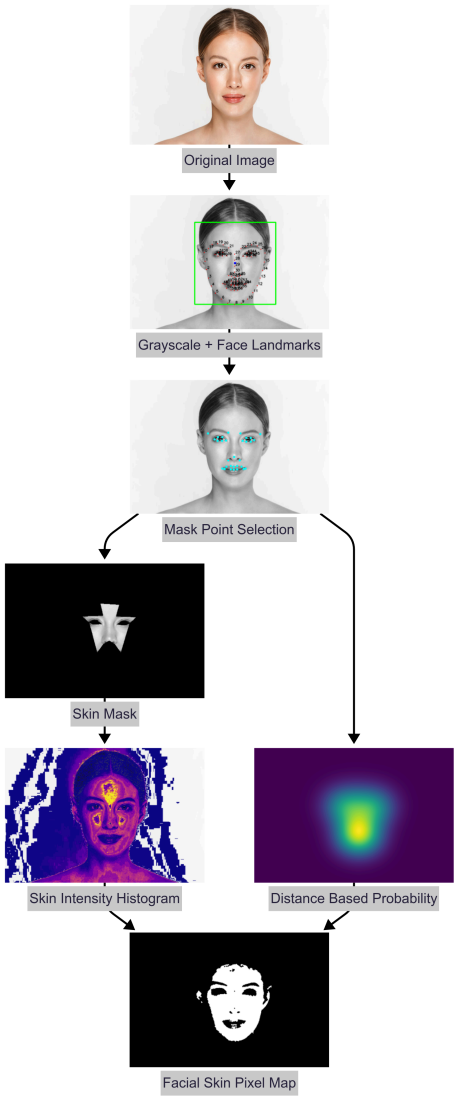

Fig. 1: Usage of contextual information to construct a face mask

B. Distance Based Skin Pixel Prediction

We also introduce an empirically defined *distance-based prior probability* $P_{\text{face}}(s)$ to approximate the expected locations of the skin pixels with respect to facial landmarks detected. This term is defined such that it is high for any pixels near the center of the detected face, and gradually decreases as distances increase.

[1] did not mention how they arrived at the prior face shown in their paper. We tried our best to recreate a prior estimate by creating a gaussian distribution on every relevant landmark. If two Gaussians overlapped, we added up their values. All values were then scaled to lie between 0 and 1 using the maximum value of the prior estimate.

C. Combining Intensity and Distance

Now, we can combine the previous two steps to acquire a conservative mask of the skin pixels in each face. A set of skin pixels can be detected using

$$P(s \mid I) \propto P(I \mid s)[P_{\text{face}}(s)]^{\alpha} > \lambda \tag{2}$$

where λ is an empirically chosen threshold and α is a relative weighting, also selected empirically.

α represents how much importance we provide to the distance estimate, higher values of α will result in a skin mask that is tightly bound to the center of the previously detected mask.

IV. Grayscale Feature Extraction

Skin pixels tend to cluster in distinct patterns in color spaces, but we do not have the advantage of color information within the scope of [1]. Instead, we use certain grayscale statistics within restricted sections of the image matrix, with the assumption of [1] that these sections (superpixels, in the case of this implementation) contain relatively uniform features. Superpixels are clusters of pixels within a region that show similar properties. We have used the SLIC algorithm (available in `cv.ximgproc`) to find these, as prescribed in [1]

A. Basic Grayscale Intensity Features

The first superpixel features we tackle are mean intensity (for overall brightness information), aggregate local standard deviation (to find the intensity variation in neighborhoods from their respective means), and aggregate local neighborhood entropy (for a measure of unpredictability and complexity in neighborhoods). Our approach to calculating the latter two is sliding a general kernel of neighborhood size N that executes the respective computation, and then calculating the mean per superpixel. Entropy and standard deviation over a neighborhood are defined as follows:

$$STD(\mathbb{N}i) = \sqrt{\frac{1}{r^2}\sum_{j\in\mathbb{N}}\left(x_j - \bar{x}_i\right)^2} \tag{3}$$

$$ENT(\mathbb{N}i) = -\sum_{i} P(x_i)\log(P(x_i)) \tag{4}$$

B. Local Binary Patterns

Local Binary Patterns are effective at learning local textural patterns in grayscale images. The two numeric arguments required for $LBP_{P,R}$ are P: the number of selected sample points around a pixel p_i, and R: the radius of the symmetric, circular neighborhood from where the points are chosen. A thresholding function $s(x)$ is then applied to compare the intensity of $p_i = I_p$ to the intensities of the sample points. Standard LBP for a pixel can be represented as the following:

$$LBP(P,R) = \sum_{p=0}^{P-1} s\left(I_p - I_{p_i}\right)2^p \tag{5}$$

$$s(x) = \begin{cases} 1 \text{ for } x > 0 \\ 0 \text{ for } x < 0 \end{cases} \tag{6}$$

[1] calls for a uniform rotation invariant operator $\text{LBP}^{\text{riu2}}_{P,R}$. We have implemented our understanding of this. The high-level steps are as below:

1) For a given image, calculate $LBP_{P,R}$ using the standard method.

2) Initialize an empty lookup table with size 2^P (the number of binary patterns possible pertaining to a pixel I_p).
3) Fill the lookup table:
   - For each possible binary pattern from 0 to $2^P - 1$, count the number of bit transitions in its binary representation.
   - If the number of transitions < 2, store the pattern in the lookup table.
   - Else, store the value P + 1 as a placeholder to reduce the number of unique binary patterns appearing in a superpixel, retaining rotation invariance.
4) Use the default LBP map to find the corresponding lookup table values for all its individual binary patterns.
5) Return the rotation invariant binary pattern map.

We have used the selection of 12 $[P, R]$ values referred to in [1]:

$$\begin{aligned}&\{[4,1],[4,3],[8,1],[8,3],[16,2],[16,5],\\&[4,2],[4,4],[8,2],[8,4],[16,3],[16,7]\}\end{aligned} \tag{7}$$

Next, we find lacunarity maps for each of the 12 LBP maps, which gives us an idea of the "gappiness" or irregularities in binary images. We use a non-traditional LBP-based lacunarity as prescribed in [1] for its high performance. The first step is to transform the LBP map into binary maps of 1s and 0s. For a pixel p_i in the LBP map, a binary image can be created as such:

$$J^K_{[P,R]}(p_i) = \begin{cases} 1 \text{ if } I_{[p,r]}(p_i) = k \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

where

$$k = \begin{cases} \{1,2,4\} \text{ if } P = 4 \\ \{1,4,7,8\} \text{ if } P = 8 \\ \{1,8,15,16\} \text{ if } P = 16 \end{cases} \tag{9}$$

This gives us a total of 44 binary maps, of which we can get 44 scalar lacunarity values calculated as given below:

$$Lac^k_{[P,R]}(sp_i) = \frac{1}{N_i} \sum_{p_i \in sp_i} I(p_i = 0) \tag{10}$$

where N_i is the number of individual pixels in the superpixel sp_i. This can be explained as the number of 0s in the binary map (given P, R, k) of the superpixel. The k values seem to have been picked by [1] based off of the desired textural features represented by the value, which they found useful for detecting skin. The resulting lacunarities of the above process can be consolidated into a 44 dimension feature vector corresponding to its respective superpixel.

V. Feature Based Probabilities & Non-Face Regions

We work with the assumption that skin pixels in the same image do not vary too significantly, meaning superpixels containing skin would have a similar feature set.

The feature set of a superpixel is defined as shown below, as the attributes of a Python class containing scalar values obtained via feature extraction:

```
class Feature:
  label: int
  mean_intensity: np.float64 | None
  std_intensity: np.float64 | None
  entropy: np.float64 | None
  lacunarity_vector: np.ndarray | None
```

We will now use the skin patch (facial skin pixel map) evaluated in section III as a representative of texture properties for general skin-like regions. Upon learning textural features from this patch (called Ω_s), seed points (superpixels) are selected that are statistically similar enough to Ω_s, signalling skin regions with higher probabilities, then utilized in a region growing algorithm to capture the rest of the skin.

For a superpixel i, the probability of it being skin given a set of n features it represents are as follows:

$$P(s \mid f^i_1, f^i_2, f^i_3, ..., f^i_n) = \Pi^n_{k=1} P_i(s \mid f^i_k) \tag{11}$$

$P_i(s \mid f^i_k)$ can be referred to as the learned probability for a feature f_k:

$$P_i(s \mid f^i_k) = \Phi_k\left(d^{f^i_k}_{i,\Omega_s}\right) \tag{12}$$

where

$$\Phi_k(x) = \frac{1}{1 + e^{-\frac{1}{x}}} \tag{13}$$

and

$$d^{f^i_k}_{i,\Omega_s} = \begin{cases} KL\left(f^{\Omega_s}_k \parallel f^x_k\right) \text{ if } f_k \in \mathbb{P} \\ \parallel f^{\Omega_s}_k - f^i_k \parallel_2 \text{ if } f_k \in \mathbb{R} \end{cases} \tag{14}$$

additionally, where *KL* is the Kullback-Leibler divergence formula. Somewhat disappointingly, we do not have any features belonging to the probability space $\mathbb{P}$, so we did not look into or implement Kullback-Leibler divergence. We did, however, compute Euclidean distances for all our features, which are scalar quantities.

An adjustment we made for accuracy was calculating superpixel feature divergences (again, Euclidean distance) with the superpixel belonging to Ω_s that showcases the highest $P(s \mid F^i)$ with the specific former superpixel. So:

$$d^{f^i_k}_{i,\Omega_s} = \parallel f^j_k - f^i_k \parallel_2 \text{ if } f_k \in \mathbb{R} \tag{15}$$

where *j* is the ideal, best fit superpixel pair for *i*.

Also, the above formula (13) for $\Phi_{k(x)}$ seems practically/ experimentally arrived at in [1], considering the lack of explanation provided. We surmise that the choice of equation is for the purpose of punishing larger distance matches, and rewarding closer ones.

VI. Region Growing Algorithm

This algorithm is the main algorithm [1] uses to detect skin superpixels outside the established Ω_s. After a probability map of P_i is calculated for all superpixels $\in \overline{\Omega_s}$, We apply a

threshold (τ_F) to select the top m ratio of these superpixels WRT P_i. These are the seed superpixels.

For each seed superpixel sp_s, we select valid neighbors N_{sp}. For each superpixel $\mathrm{sp}_i \in N_{\mathrm{sp}}$, we recalculate class probabilities P_i^{new}. Here:

$$P_i^{\mathrm{new}}(s \mid F^i) = P_i^{\mathrm{node}}(s \mid F^i) P_i^{\mathrm{edge}}(s \mid F^i) \tag{16}$$

where:

$$P_i^{\mathrm{node}}(s \mid F^i) = P_i(s \mid F^i) \tag{17}$$

$$P_i^{\mathrm{edge}}(s \mid F^i) = \Pi_{k=1}^{n} \Phi_k \left(\frac{1}{N_i^{\mathrm{skin}}} \sum_{j \in N_i^{\mathrm{skin}} i} d_{i,j}^{f_k} \right) \tag{18}$$

τ_F determines the candidacy of any superpixel sp_i, and the algorithm stops if:

- There are no further candidate superpixels around the seeds.
- If it reaches a defined maximum number of iterations.
- Maximum probability value deviation from the original seed points.

$$\tau_F = \kappa \frac{1}{N_s} \sum_{j \in \mathrm{sp}_s} P(s \mid F^i) \tag{19}$$

where N_s is the total number of seed superpixels and κ is a hyperparameter.

Algorithm 1: Region growing algorithm

Initially compute class probability of each superpixel and select set of seed superpixels $\{sp_s\}$; from contextual information, define set of non-skin superpixels $\{sp_{ns}\}$;
For $l = 1$ to max iterations **do**
  For $k = 1$ to number of seeds sp_i **do**
    Select candidate neighboring superixels, N_{sp}
    For $i = 1$ to number of superpixels in N_{sp} **do**
      Calculate $P_i^{new}\left(s|F^i\right)$
      If $P_i^{new}\left(s|F^i\right) > \tau_{F^i}$ then
        Update $\{sp_s\} = \{sp_s\} \cup \{sp_i\}$
      else
        Update $\{sp_{ns}\} = \{sp_{ns}\} \cup \{sp_i\}$
      end If
    end For
  end For
  Check stopping criteria
end For

Fig. 2: Sample selected seed superpixels

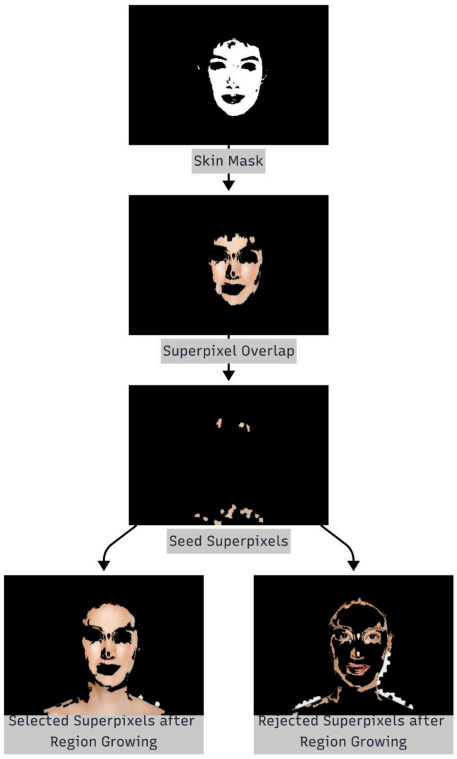

Fig. 3: The skin detection process

VII. Experimental Results

The algorithm seems to have relative success in detecting large amounts of ground-truth skin area in images where the skin pixels do not vary too much from each other due to factors of harsh/low lighting, irregular perspective, extreme shadows, or low contrast. Some examples of image results are given below, featuring people of various races in various environments.

Fig. 4: 1: Busy environment

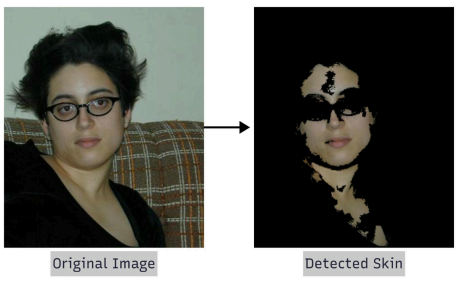

Fig. 5: 2: Glasses as obstruction

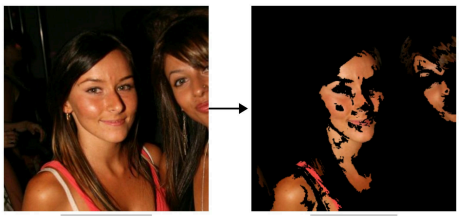

Fig. 6: 3: Two people

A. *Limitations*

Unfortunately, A ground-truth loss validation loop could not be carried out, due to the fact that the reference paper's dataset was not publicly accessible, as were most datasets we could find. As for algorithmic results:

- Hyperparameters require tweaking for unique skintones and environments, meaning a scope for improvement could include the evaluation of intensity and noise features in the entire image to functionally tune parameters.
- Additionally, results sometimes fare better with the exclusion/inclusion of subsets of features rather than all of them. A few interesting examples of varying performances for all subsets of the feature powerset in considerations are included below. A major scope for improvement is using the above mentioned approach of overall image feature/statistics evaluation to find the ideal subset of features to take into consideration for seed picking and region growing.
- Noisier images fare much better in the segmentation results when the number of iterations are reduced compared to images with less unpredictability or texture variations. Intuitively, we feel this may be due to a limitation of the number of features we include to consider and differentiate skin pixel trends from non-skin pixel trends. A mix of additional feature extraction and image analysis as mention above to tune iterations seems to be a good avenue for improvement.
- Additional, well thought stopping criteria for the prevention of selection leaking can be looked into.
- Of course, access to a skin segmentation dataset will provide significant insight into algorithm performance. We are considering reaching out to private dataset owners for permission to use their ground truths.
- Currently, calculation of most favorable divergence vector is a brute force list search. This can lead to a slow runtime and needs to be optimized

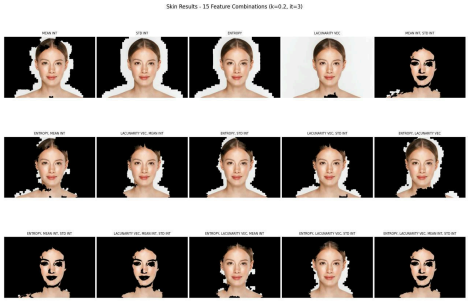

Fig. 7: All combinations of features: 1

Fig. 8: All combinations of features: 2

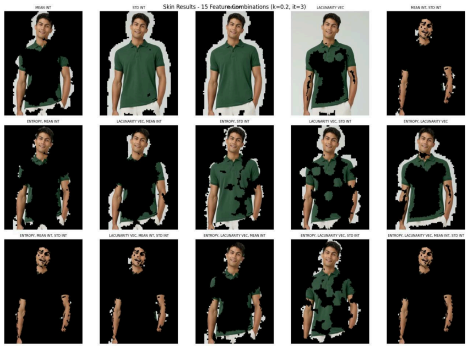

Fig. 9: All combinations of features: 3

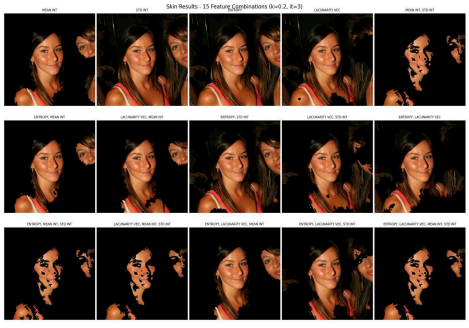

Fig. 10: All combinations of features: 4

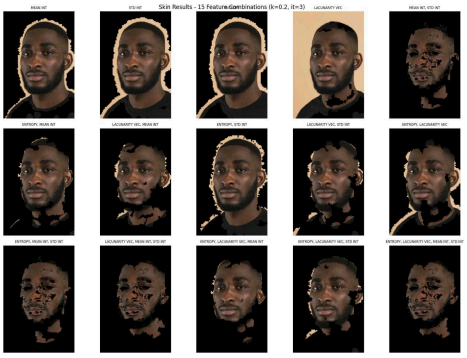

Fig. 11: All combinations of features: 5

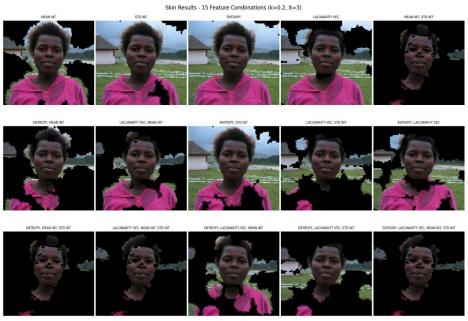

Fig. 12: All combinations of features: 6

VIII. Appendix

A. SLIC (Simple Linear Iterative Clustering)

[4] proposes a method to segment images into superpixels, which are meaningful clusters of spatially connected pixels that can be used in place of the granular, rigid grid layout of individual pixels.

There are multiple reasons we may want to work with superpixels rather than the existing atomic unit of pixels.

1) **Efficiency**: Superpixels increase computation speed and efficiency of graph-based, neighborhood-based or region-based algorithms.
2) **Meaning**: Superpixels can capture spatial and textural features better than individual pixels, which cannot capture characteristics with just intensity.
3) **Noise Handling**: Unlike individual pixels, superpixels offer less significance to rogue anomalies.
4) **High Level Algorithms**: Pixel clusters provided by superpixels are significantly better input to segmentation and boundary-mapping algorithms than individual pixels.

SLIC improves over traditional clustering algorithms like k-means by operating in a 5D space—three dimensions for color (typically CIELAB) and two for pixel coordinates. It restricts each cluster's search space to a local square region proportional to the expected superpixel size, significantly reducing computational cost. The clustering is guided by a distance metric:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 \cdot m^2} \tag{20}$$

where d_c is Euclidean distance in color space, d_s is spatial Euclidean Distance, S is the spacing between cluster centers, and m is a compactness/uniformity factor.

In grayscale, d_c is just a simple intensity difference.

B. Haar-Cascades Facial Recognition

In, [5], Haar-Cascades utilizes **Haar features**, essentially feature rectangles, to detect faces. Haar features represent local patterns of intensity contrast, where adjacent rectangular regions have their intensities summed up.

Here are the Haar features commonly seen in faces:

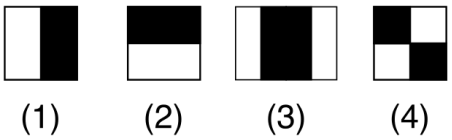

Fig. 13: Haar features

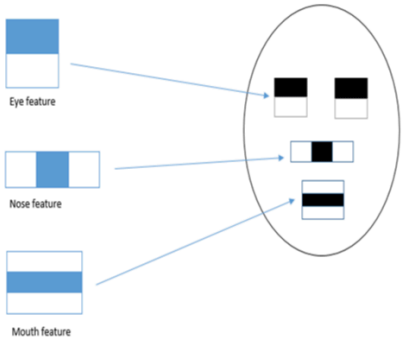

Fig. 14: Typical Haar feature placement

To compute these features efficiently at various scales and positions, the algorithm uses integral images (a.k.a. summed-area tables). An integral image is a preprocessed version of the input image where each pixel at position (x, y) stores the sum of all pixel intensities above and to the left of (x, y), inclusive. This enables constant-time computation *O(1))* of any rectangular region's sum, regardless of its size—drastically speeding up feature evaluation across sliding windows.

During training, a massive number of Haar features (on the order of tens or hundreds of thousands per window) are considered. AdaBoost is then employed to select a small number of highly discriminative features and combine them into a strong classifier. Each weak classifier corresponds to a single Haar feature. AdaBoost iteratively selects weak classifiers that minimize the weighted error over the training samples, updating sample weights to emphasize those examples that are misclassified. The final output is a weighted sum of weak classifiers—forming a strong classifier capable of distinguishing faces from non-faces.

References

[1] A. Sarkar, A. L. Abbott, and Z. Doerzaph, "Universal Skin Detection Without Color Information," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 20–28. doi: 10.1109/WACV.2017.10.

[2] J. Plin, "Relabeled HELEN Dataset." 2020.

[3] S. Mallick, “FacialLandmarkDetection.”
[4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods,” vol. 34, no. 11. pp. 2274–2282, 2012. doi: 10.1109/TPAMI.2012.120.
[5] P. Viola and M. Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features.” IEEE, pp. 511–518, 2001. doi: 10.1109/CVPR.2001.990517.