Project Title: Real-Time Virtual Makeup Style Switching for Indian Skin Tones

CS24M1005 - SINDHIYA R

---

1. BASE PAPER

Title: A Comprehensive Review of Virtual Makeup Transfer: Current Techniques, Challenges, and Future Directions

This research paper presents a real-time virtual makeup application method by detecting facial features and applying makeup colors after face segmentation. The technique uses face landmarks and divides the face into sections like lips, eyes, and eyebrows to apply makeup digitally.

Key Concepts Taken for the Project:
- Using face landmarks for identifying key facial features such as lips and eyes
- Segmentation of the face into regions to apply makeup only to relevant areas
- Real-time makeup application using OpenCV

---

2. CODE EXPLANATION

The system uses **OpenCV** for image processing and **Mediapipe** for detecting facial landmarks in real-time. Different makeup styles can be applied, and users can switch between them with keyboard inputs.

Modules Used:
- cv2 – for capturing video from the camera and displaying results
- mediapipe – to detect 468 facial landmarks
- numpy – for handling numerical operations and creating masks

Main Components:
1. **Face Detection:** Mediapipe detects 468 key points on the face.
2. **Makeup Region Extraction:** The face is divided into regions like the lips, eyes, and eyebrows using specific landmarks.
3. **Makeup Application:** Color is applied to the identified regions using transparent blending techniques.
4. **Style Switching:** Users can switch between different makeup styles instantly using keypresses.

image.py
1. **get_skin_tone(image, landmarks):**
    a. This function detects the skin tone of a person based on a sample point from their face.
    b. It takes an image and face landmarks as input. It extracts the color of a pixel near a specific facial landmark and converts it to HSV color space.

    c. Based on the value (V) component of the HSV color, it classifies the skin tone as "fair", "medium", or "deep" by checking the pixel brightness.

   d. If the brightness value (V) is higher than 200, it returns "fair", between 120 and 200 returns "medium", and below 120 returns "deep".

2. **adjust_makeup_colors(skin_tone):**
   a. This function adjusts the makeup colors based on the detected skin tone.
   b. It updates the colors_map for the lip colors. For "fair" skin, it applies a soft pink color; for "medium" skin, it uses red; and for "deep" skin, it applies a deep wine color for the lips.
   c. This allows the makeup colors to adapt dynamically based on the user's skin tone.

3. **main(image_path):**
   a. This is the core function that processes an image to apply makeup based on skin tone.
   b. It first reads the input image and detects the face landmarks using the read_landmarks() function.
   c. The get_skin_tone() function is then used to determine the skin tone, and adjust_makeup_colors() adjusts the makeup colors accordingly.
   d. The function creates a mask with the adjusted makeup colors using the add_mask() function, which is applied to the original image.
   e. The final image is created by blending the original image with the mask, and optional skin tone information is overlaid on the output.
   f. Finally, the processed image is displayed using show_image().

4. **__main__ block:**
   a. This part of the script initializes the program by parsing command-line arguments for the image path.
   b. The main() function is called with the provided image path, processing the image as described above.

<u>**utils.py**</u>

1. **show_image(image, msg)**
   a. Displays the input image using OpenCV.
   b. Used mainly for debugging or showing the final output with an optional title.

2. **read_landmarks(image)**
   a. Detects facial landmarks using Mediapipe FaceMesh.
   b. Converts normalized landmark coordinates to pixel values and returns them as a dictionary.

3. **add_mask(mask, idx_to_coordinates, face_connections, colors)**
   a. Draws filled colored polygons over the facial features (like lips, eyes, eyebrows) using landmark points.
   b. Applies Gaussian blur to the mask for smooth blending.

4. **face_points**
   a. A dictionary mapping specific facial features (e.g., LIP_UPPER, EYEBROW_LEFT) to their respective Mediapipe landmark indices.
   b. Used to define regions for applying makeup.

camera.py

1. **get_skin_tone(image, landmarks):**
    a. Detects the user's skin tone (fair, medium, or deep) by sampling a pixel near a specific facial landmark.
    b. Converts the pixel color to the HSV color space and uses the value (V) to categorize the skin tone based on brightness levels.

2. **adjust_makeup_colors(style, skin_tone):**
    a. Adjusts the makeup colors (specifically lips) based on the detected skin tone.
    b. For "fair" skin, it applies a bright lipstick color, while for "deep" skin, it uses darker lipstick shades. It returns the updated style.

3. **draw_style_buttons(image, selected_idx):**
    a. Draws buttons on the screen that allow the user to choose between different makeup styles.
    b. The selected style is highlighted, and the button labels are updated based on the style index.

4. **handle_mouse_event(event, x, y, flags, param):**
    a. Handles mouse click events to detect when the user clicks on one of the style buttons.
    b. Changes the current makeup style based on the user's selection by checking the coordinates of the click.

5. **main loop:**
    a. Captures video frames from the webcam in real time.
    b. Detects face landmarks and applies makeup using the selected style and adjusted colors.
    c. Displays the result on the screen, including a label showing the detected skin tone.
    d. Allows style selection via mouse click or keyboard input, and updates the video display accordingly.
    e. Exits the loop when the "q" key is pressed.

---

3. NOVELTY

- **Indian Skin Tone Focus:** The makeup color palettes are specifically chosen to complement Indian skin tones, ensuring they look realistic and vibrant.
- **Real-Time Switching:** 7 Makeup styles has been added in the output interface.
- **Distinct Makeup Styles:** Each style is designed to offer a visually noticeable difference, making it easy for users to try different looks quickly.