HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

REPORT

Beautification Filters on Face Detection and Recognition

Course: Digital Image Processing
Course Code: ET4591E

Supervisor: Prof. Tran Thi Thanh Hai
Supervisor: Prof. Le Thi Lan

Member: Tran Minh Ha 20224309
Member: Nguyen Mai Huong 20224314

Hanoi, 1/2025

Beautification Filters on Face Detection and Recognition

First Member
Nguyen Mai Huong
ET-E16 02
huong.nm224314@sis.hust.edu.vn

Second Member
Tran Minh Ha
ET-E16 01
ha.tm224309@sis.hust.edu.vn

This project introduces a comprehensive system for applying beautification filters in real-time video processing through a webcam. Utilizing advanced image processing techniques and facial landmark detection, the system features multiple filter categories: color adjustments, accessory overlays, and skin enhancement. Each filter dynamically adapts to facial movements and varying environmental conditions, providing seamless and accurate results. The evaluation demonstrates the system's effectiveness in real-time performance, robustness in different lighting conditions, and adaptability. Future work will include integrating machine learning models for personalized beautification and optimizing performance for mobile platforms.

1. Introduction

1.1. Problem Statement

The rapid rise of social media and augmented reality applications has increased demand for real-time beautification filters. These filters aim to improve facial aesthetics, enhance user experience, and add entertaining elements through virtual accessories. However, implementing such filters presents challenges, including accurate facial landmark detection, real-time performance, and consistency under dynamic conditions like head movements and poor lighting.

1.2. Input/Output

- **Input:** Real-time video feed from a webcam
- **Output:** Enhanced video feed with applied beautification filters (e.g., color adjustments, skin smoothing, virtual accessories).

1.3. Challenges

- Maintaining accurate filter placement during facial movements.
- Ensuring real-time performance without noticeable latency.
- Adapting to varying lighting conditions and camera distances.
- Integrating multiple filters while maintaining system stability.

This project bridges the gap between practical deployment and theoretical advances in image processing, focusing on robust filter application using a lightweight and scalable approach.

2. Related works

The domain of real-time facial beautification has gained significant attention with the proliferation of augmented reality (AR) applications and computer vision technologies. This section explores the foundational works and advancements related to the three primary aspects of this project: facial landmark detection, beautification techniques, and accessory overlays.

2.1. Facial Landmark Detection

Facial landmark detection is a cornerstone of many computer vision applications, providing the structural basis for tasks like filter alignment, expression recognition, and facial morphing.

Traditional Methods

Early approaches to facial detection relied heavily on handcrafted features:

- **Viola-Jones Algorithm (2001):** A breakthrough in object detection, this method utilized Haar-like features combined with an AdaBoost classifier to perform real-time face detection. Despite its efficiency, the algorithm struggled with non-frontal faces and lighting variations, making it less suitable for dynamic applications like AR[8].
- **Active Shape Models (ASMs)** and **Active Appearance Models (AAMs):** These statistical shape models used annotated datasets to locate facial landmarks. However, they required significant computational resources and were sensitive to initialization errors.

Modern Approaches

Modern techniques leverage machine learning and deep

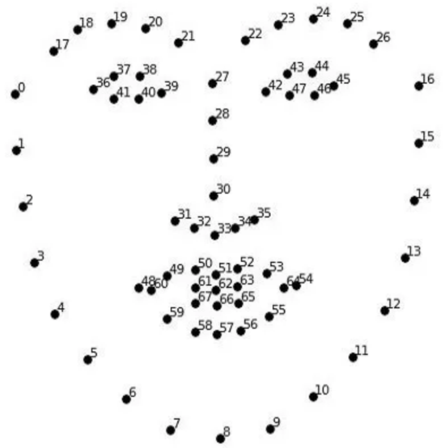

Figure 1. Visualization of Dlib's 68-point facial landmarks.

learning for more robust landmark detection:

- **Dlib's 68-point Landmark Model:** Based on Histogram of Oriented Gradients (HOG) and Support Vector Machines (SVM), this model provides precise facial feature localization across varying conditions. It is widely used in real-time applications due to its balance between accuracy and computational efficiency[5] .
- **Deep Learning Models:** Methods like MediaPipe Face Mesh and OpenCV's deep neural network (DNN) modules provide high-density facial landmarks. These models utilize convolutional neural networks (CNNs) trained on large datasets, enabling robustness against occlusions and extreme poses[3] .

2.2. Beautification Techniques

Beautification techniques aim to enhance facial aesthetics by improving image quality, reducing imperfections, and applying artistic effects.

Traditional Image Processing

Conventional methods include:

- **Histogram Equalization:** Enhances brightness and contrast, especially under low-light conditions. This technique works by redistributing pixel intensity values to achieve uniform histogram distribution. However, it often leads to overexposed or unnatural results in certain regions[2].
- **Gaussian Blur and Bilateral Filtering:** While Gaussian blur smoothens textures, bilateral filtering preserves edges, making it ideal for skin smoothing[6].

Advanced Techniques

Machine learning has revolutionized facial beautification:

- **GAN-based Beautification:** Generative adversarial networks (GANs) have been employed for tasks like makeup transfer (e.g., BeautyGAN) [9] and skin texture refinement. GANs learn mappings between input and output domains, enabling realistic transformations.

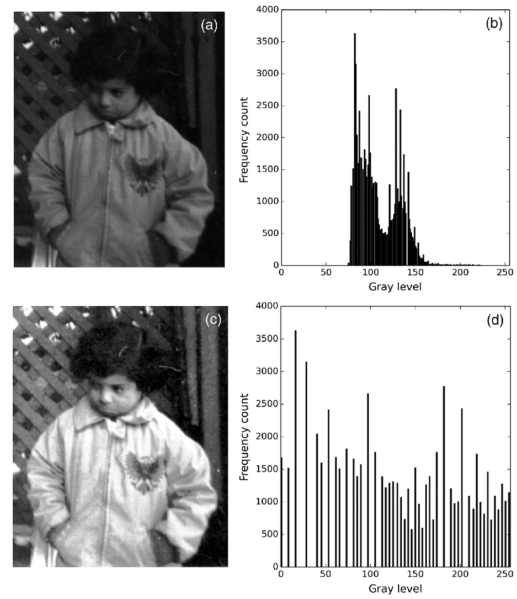

Figure 2. Example of histogram equalization for brightness adjustment

Figure 3. Comparison of Gaussian blur and bilateral filtering for skin smoothing.

- **Edge-preserving Filters:** Algorithms such as guided filters and anisotropic diffusion enhance specific regions without affecting the overall structure.[4]

2.3. Accessory Overlays and Augmented Reality

Virtual accessory overlays, such as hats, glasses, dog noses, cat faces, and floral crowns, play a pivotal role in enhancing user interaction in augmented reality (AR) applications. The implementation of these overlays involves precise alignment, scaling, and orientation to seamlessly integrate them with facial features. In the context of this project, accessory filters are dynamically adjusted in real-time to maintain accurate placement despite facial movements and changes in perspective.

Traditional Methods

Early methods for accessory overlays were limited by their reliance on static templates and pre-defined facial regions:

- **Template-based Approaches:** These methods employed fixed templates for accessory placement, often resulting in inaccuracies when applied to dynamic facial movements or diverse facial geometries[7].
- **Shape-based Models:** Approaches such as Active Shape Models (ASMs) provided basic adaptability but struggled with real-time applications due to computational constraints and sensitivity to noise.

Advances in Landmark-based Overlay Placement
The introduction of robust facial landmark detection has significantly enhanced the accuracy and adaptability of accessory overlays:

- **Geometric Transformations:** Using the coordinates of key facial landmarks, geometric transformations[2] like scaling, rotation, and translation are applied to align the accessory images with the user's face. For example:
  - **Hats:** Positioned above the eyebrows, with dynamic adjustments for tilt and scale based on the distance and angle between the eyes and the forehead.
  - **Glasses:** Placed over the eyes, with scaling determined by the distance between the outer corners of the eyes.
- **Angle Correction:** Facial orientation is calculated using landmarks such as the eyes and nose. Rotation matrices are applied to the accessories to match the face's tilt accurately.

This project builds upon existing works by combining robust facial landmark detection with lightweight, customizable filters to deliver real-time performance on standard hardware.

3. Proposed methods

The proposed methods in this project leverage advanced image processing techniques and geometric transformations to implement a suite of beautification filters. These methods are designed to operate in real time, ensuring seamless integration of color adjustments, facial enhancements, and accessory overlays with webcam video streams.

3.1. Data Structure and Workflow

The "Selfie Beautification Filter" system is designed for real-time facial enhancement and customization. The workflow is described as follows:

- **Trackbar Initialization:** Trackbars are created for selecting RGB values dynamically, enabling real-time adjustments for effects like lipstick and eye color:

  Trackbars: Blue, Green, Red (Range: 0-255).

- **Video Capture:** The system continuously captures video frames from a webcam using OpenCV:

$$\text{Frame} = \text{cv2.VideoCapture}(0).$$

- **Facial Landmark Detection:** Dlib's facial landmarks are used to detect key facial features, enabling precise application of filters and effects:

  Landmarks: eyes, lips, cheeks, etc..

- **Effect Application:** Based on the user-selected effect (*current_effect*), various transformations are applied:
  - **Smooth Skin:** Applies a bilateral filter to the facial region.
  - **Eye Color:** Changes the pupil color based on RGB values.
  - **Fish Eye:** Creates a bulging effect around the eyes.
  - **Blush:** Adds a soft blush effect to the cheeks.
  - **Lipstick:** Enhances the lip region with the selected color.
  - **Accessory Overlays:** Adds virtual elements like glasses, hats, or crowns.
  - **Artistic Filters:** Applies filters such as Negative, Sepia, and Cool Tone.
- **Button and Mouse Interaction:** Interactive buttons allow users to switch between effects. A mouse callback function handles button clicks and effect selection dynamically.
- **Image Capture:** Users can capture a frame by pressing the 'c' key, saving the enhanced image:

  Saved Image: PICTURE.jpg.

- **Keyboard Shortcuts:**
  - **'g':** Set eye color to green.
  - **'b':** Set eye color to blue.
  - **'r':** Set eye color to red.
  - **'p':** Set eye color to purple.
  - **'o':** Reset to the original eye color.
  - **'q':** Exit the application.
- **Real-Time Display:** The final processed frame is displayed in a window titled "Selfie Beautification Filter," updated continuously with user-selected effects.
- **Exit:** The program exits cleanly when the user presses the 'q' key, releasing resources and closing all windows.

This workflow provides a highly interactive and customizable experience, enabling users to enhance their appearance in real-time.

3.2. Detailed Algorithm

3.2.1. Face Detection and Landmark Detection

The project begins with precise face detection and landmark extraction to enable accurate placement of filters and overlays.

- **Face Detection** : The Histogram of Oriented Gradients (HOG) method is employed to detect faces. The image is divided into small regions, and gradient direction histograms are calculated for each region. A linear Support

Vector Machine (SVM) classifier then identifies faces.

$$\text{HOG}(x,y) = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

where $I(x,y)$ is the pixel intensity.

- **Facial Landmark Detection:** Dlib's 68-point facial landmark detector is used. This model is trained on ensemble regression trees to predict landmark locations.

3.2.2. Color Adjustment Filters

Color adjustment filters enhance the overall aesthetic of the video feed by manipulating pixel values. Below are the specific filters used:

- **Sepia Filter:** This filter transforms the image into a warm, vintage-like tone using a linear transformation of RGB values.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where R', G', B' are the transformed pixel values.

- **Negative Filter:** Inverts pixel intensities to produce a negative effect:

$$I'(x,y) = 255 - \cdot I(x,y)$$

- **Brightness Adjustment:** Adjusts the brightness β and contrast α of the image:

$$I'(x,y) = \alpha \cdot I(x,y) + \beta$$

- **Histogram Equalization:** Histogram equalization adjusts pixel intensities in the Y channel:

$$Y_{\text{equalized}} = \text{cdf}(Y) \cdot (L-1)$$

where $\text{cdf}(Y)$ is the cumulative distribution function, and L is the number of intensity levels.

- **Gaussian Blur:** The Gaussian kernel $G(x,y)$ is given by:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

3.2.3. Geometric Transformations

- **Face Orientation Estimation:** Determines the tilt or rotation angle of a face based on the relative positions of the eyes. Used for aligning filters like glasses, hats, or masks with the face orientation.

$$\theta = -\arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

where $(x1, y1)$ and $(x2, y2)$ are coordinates of the left and right eyes, respectively.

- **Filter Alignment Rotation:** Rotates accessory filters (e.g., hats, glasses) to align with the face angle

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix}$$

where t_x and t_y are translation adjustments.

- **Scaling:** The size of the accessory is dynamically adjusted based on the distance between landmarks, such as:

$$\text{Width} = k \cdot (\text{landmark}_{16} - \text{landmark}_0)$$

3.2.4. Accessory Overlay Placement

Accessory placement involves mapping virtual objects onto the face with precision:

- **Landmark Mapping:** Identify key landmarks for positioning accessories.
  Example: The glasses align with the corners of the eyes (landmarks 36 to 45) and hats align with the top forehead region (landmarks 19 to 24).
- **Alpha Blending:** Merge accessory images with the original frame:

$$I'(x,y) = \alpha \cdot A(x,y) + (1-\alpha) \cdot I(x,y)$$

where $A(x,y)$ is the accessory image and α is its transparency

3.2.5. Wide-Angle Distortion

This algorithm applies a wide-angle distortion effect, simulating a fisheye or wide-lens view. It uses intrinsic parameters (camera matrix) and distortion coefficients to remap pixel coordinates. Pixels further from the center are distorted more significantly than those closer, creating a bulging effect.

- **Distortion Model:**

$$x' = x(1 + k_1 r^2 + k_2 r^4) + 2p_1 xy + p_2(r^2 + 2x^2)$$

$$y' = y(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y^2) + 2p_2 xy$$

where r is the radial distance from the center, and k_1, k_2 $\ldots$ are distortion coefficients.

- **Camera matrix** The matrix K represents intrinsic camera parameters:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where:

f_x, f_y : Focal lengths in pixels along x and y axes.

c_x, c_y : Principal point coordinates (usually the image center).

3.2.6. Fisheye Effect for Eyes

The fisheye effect is applied to the eye regions to create a bulging or distorted appearance, enhancing the aesthetic or stylistic impact. The algorithm consists of the following steps:

- **Region of Interest (ROI):** The left and right eye regions are extracted based on the Dlib facial landmarks. Specifically:
  - The ROI for the left eye is determined using landmarks 37 to 41.
  - The ROI for the right eye is determined using landmarks 43 to 47.

  The bounding box around each eye is extended by a specified radius to allow for the fisheye effect.
- **Barrel Distortion:** The fisheye effect is achieved using barrel distortion, which remaps pixels based on their radial distance from the center:

  $$r = \sqrt{x^2 + y^2}, \quad \text{where } x, y \text{ are normalized pixel coordinates.}$$

  The distortion equation modifies the radial distance:

  $$r_{\text{new}} = r - k \cdot r \cdot \cos(\pi \cdot r),$$

  where k controls the intensity of the distortion. Pixels beyond the maximum radius ($r > 0.5$) are not distorted.
- **Remapping:** The distorted radial coordinates are used to calculate the new pixel positions:

  $$x_{\text{new}} = r_{\text{new}} \cdot \frac{x}{r}, \quad y_{\text{new}} = r_{\text{new}} \cdot \frac{y}{r}.$$

  The new coordinates are scaled back to the original image dimensions and used for pixel remapping.
- **Applying the Effect:** The barrel distortion is applied independently to the left and right eye regions. The distorted regions are then blended back into the original image, preserving the rest of the facial features.
- **Output:** The final image contains bulging eyes with the fisheye effect applied, while the rest of the face remains untouched.

This algorithm leverages radial distortion to create a dramatic fisheye effect, adding a unique stylistic element to the eyes while maintaining realism in the surrounding facial features.

3.2.7. Skin Smoothing Algorithm

The skin smoothing algorithm is designed to smooth facial textures while preserving edge details. The following steps summarize the approach based on the provided code:

- **Face Region Masking:** The algorithm uses Dlib's facial landmarks to create a mask around the facial region. Points along the jawline (landmarks 0 to 16) are extracted and used to define the mask:

  $$\text{mask}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in \text{face\_region}, \\ 0 & \text{otherwise.} \end{cases}$$
- **Smoothing with Bilateral Filter:** A bilateral filter is applied to the entire image to smooth textures while retaining edges. The filter is defined as:

  $$B(x, y) = \frac{1}{W} \sum_{i,j} \exp\left(-\frac{\Delta_s^2}{2\sigma_s^2}\right) \exp\left(-\frac{\Delta_r^2}{2\sigma_r^2}\right)$$

  where:

  $$\Delta_s = \sqrt{(x_i - x)^2 + (y_i - y)^2}, \quad \Delta_r = I_i - I,$$

  and σ_s controls spatial filtering, and σ_r controls intensity filtering.
- **Mask Application:** The smoothed image is combined with the mask to isolate the filtered face region:

  $$\text{face\_smoothed}(x, y) = B(x, y) \cdot \text{mask}(x, y).$$
- **Background Preservation:** The original image is preserved outside the face region:

  $$\text{background}(x, y) = I(x, y) \cdot (1 - \text{mask}(x, y)).$$
- **Final Combination:** The smoothed face region and the original background are combined using bitwise operations:

  $$I'(x, y) = \text{face\_smoothed}(x, y) + \text{background}(x, y).$$

This method ensures that only the facial region is smoothed, maintaining natural transitions between the face and the background.

3.2.8. Eye Color Adjustment

The eye color adjustment algorithm changes the color of the pupil region in the eyes while maintaining a natural appearance. The steps of the algorithm are as follows:

- **Region of Interest (ROI):** The algorithm identifies the pupil region for both eyes using Dlib's facial landmarks:
  - **Left Eye:** The pupil region is defined using landmarks 37, 38, 40, and 41.
  - **Right Eye:** The pupil region is defined using landmarks 43, 44, 46, and 47.

  A mask is created by filling the polygon formed by these landmarks for each eye.
- **Color Application:** A solid color (e.g., green, red, or blue) is applied to the masked pupil region. The algorithm creates a blank image of the same size as the input image and fills the mask with the desired color:

  $$\text{eye\_color}(x, y) = \begin{cases} \text{color} & \text{if } (x, y) \in \text{pupil\_region}, \\ 0 & \text{otherwise.} \end{cases}$$
- **Smoothing:** To ensure a natural transition between the colored pupil and the surrounding area, the colored region is smoothed using a Gaussian blur:

  $$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right),$$

where σ is the blur radius.

- **Blending:** The colored and blurred pupil region is blended with the original image using weighted addition:

$$I'(x,y) = \alpha \cdot \text{eye\_color}(x,y) + (1-\alpha) \cdot I(x,y),$$

where α controls the intensity of the applied color.
- **Output:** The final image has the pupil regions of both eyes adjusted to the specified color while preserving the overall natural appearance.

This algorithm provides a seamless way to enhance the eye color dynamically, making it suitable for real-time beautification or stylistic applications.

3.2.9. Blush Effect

The blush effect algorithm adds a natural-looking blush to the cheek regions, blending seamlessly with the original image. The following steps describe the implementation:

- **Region of Interest (ROI):** The algorithm identifies the cheek regions using specific Dlib facial landmarks:
  - **Left Cheek:** The region between landmarks 31 (side of the nose) and 2 (jawline).
  - **Right Cheek:** The region between landmarks 35 (side of the nose) and 14 (jawline).

  The cheek centers are calculated as the midpoints between the top and bottom of each cheek region.
- **Blush Mask:** A mask is created for each cheek region using a Gaussian-like radial fading. For each pixel within a specified radius (r) of the cheek center:

$$\alpha = \exp\left(-0.5 \cdot \left(\frac{\text{distance}}{r}\right)^2\right),$$

where α determines the transparency of the blush color at that pixel. The mask is filled with a light pink color ($[180, 105, 255]$) weighted by α.
- **Blurring:** To ensure a smooth and natural transition, the mask is blurred using a Gaussian blur:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right),$$

where σ is the blur radius.
- **Blending:** The blush mask is blended with the original image using weighted addition:

$$I'(x,y) = (1-\text{intensity}) \cdot I(x,y) + \text{intensity} \cdot \text{mask}(x,y),$$

where intensity controls the strength of the blush effect.
- **Output:** The final image features a soft blush effect applied to the cheeks, enhancing facial aesthetics while maintaining a natural appearance.

This algorithm provides a realistic and customizable blush effect, suitable for beautification and photo editing applications.

3.2.10. Lipstick Application

The lipstick application algorithm enhances the lip region by applying a customizable color overlay. The following steps describe the implementation:

- **Region of Interest (ROI):** The lip region is extracted using Dlib's facial landmarks, specifically points 48 to 60, which define the outer and inner contours of the lips. A mask is created by filling the polygon formed by these points:

$$\text{mask}(x,y) = \begin{cases} 1 & \text{if } (x,y) \in \text{lips\_region}, \\ 0 & \text{otherwise.} \end{cases}$$

- **Color Selection:** The RGB values for the lipstick color are retrieved dynamically using trackbars, allowing real-time adjustment:

$$\text{color} = (\text{blue}, \text{green}, \text{red}),$$

where blue, green, and red are the trackbar positions.
- **Color Layer Creation:** A color layer is generated with the selected lipstick color and is restricted to the lip region using the mask:

$$\text{lips\_color}(x,y) = \begin{cases} \text{color} & \text{if mask}(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Smoothing:** To ensure a natural appearance, the color layer is smoothed using a Gaussian blur:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right),$$

where σ is the blur radius.
- **Blending:** The smoothed color layer is blended with the original frame using weighted addition:

$$I'(x,y) = (1-\alpha) \cdot I(x,y) + \alpha \cdot \text{lips\_color}(x,y),$$

where $\alpha = 0.4$ controls the intensity of the applied lipstick color.
- **Output:** The final image features the lip region enhanced with the selected color, providing a realistic lipstick effect.

This algorithm allows for dynamic and customizable lipstick application, making it suitable for real-time beautification tools and photo editing applications.

3.3. Complexity Analysis

- **Face Detection:** O(n) per frame, where n is the number of faces in a frame.
- **Landmark Extraction:** O(m) where m=68 (Dlib's model).
- **Filter Application:** Ranges from $O(k)$ for color adjustments to $O(k^2)$ for skin smoothing.

4. Experiment

The evaluation of the Selfie Beautification Filter project primarily aimed to measure the system's real-time performance and assess the visual quality of the implemented filters. The experimental setup, metrics, and results are discussed in detail below.

4.1. Experimental Setup

- **Hardware Configuration:**
  - Device: MacBook Pro with Apple M2 chip
  - RAM: 16GB
  - Webcam: Integrated 1080p FaceTime HD Camera
  - Software: Python 3.10, OpenCV 4.10, Dlib 19.24.6
- **Input Source:** Live feed from the integrated webcam.

4.2. Evaluation Metrics

To assess the performance and effectiveness of the filters, the following metrics were chosen:

4.2.1. FPS (Frames Per Second):

FPS quantifies the system's efficiency by measuring how many frames are processed per second. This is crucial for ensuring real-time performance, particularly for interactive applications like augmented reality. The FPS for each filter was calculated using the formula:

$$\text{FPS} = \frac{\text{Total Frames}}{\text{Elapsed Time (seconds)}}$$

4.2.2. Visual Inspection:

Since many filters focus on aesthetic enhancement, qualitative assessment was conducted. This involved visually evaluating each filter for:

- Accurate alignment to facial landmarks.
- Aesthetic quality of the applied effect.
- Stability under dynamic conditions, such as head movements or changes in lighting.

4.2.3. Implementation

The evaluation involved capturing 100 frames for each filter to measure its average FPS. The filters were also tested under varying conditions, such as head rotation, distance from the camera, and changes in lighting, to observe their adaptability. Filters that required computationally intensive operations, such as accessory overlays, naturally had lower FPS, while simpler filters like color adjustments achieved higher FPS.

4.3. Experimental Results

The experimental results are summarized in Table 1. FPS values highlight the system's real-time performance, while visual inspection provided qualitative feedback on each filter's aesthetic impact.

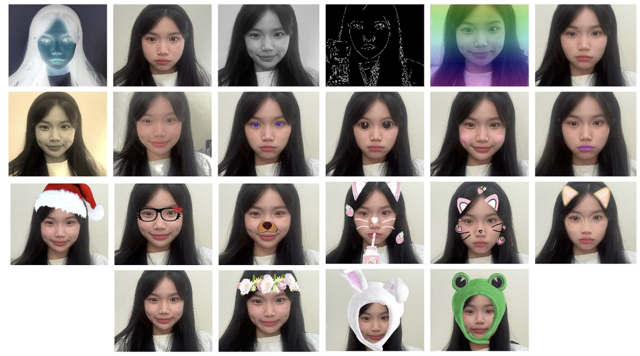

Figure 4. Output of 22 filters

Table 1. Summarizes the performance of filters based on FPS

Filter	FPS
Smooth Skin	9.99
Eye color	10.19
Fish Eye	10.07
Blush	9.98
Lipstick	10.02
Dog Nose	9.97
Xmas Hat	10.00
Frog Hat	7.04
Bunny Hat	9.97
Cat 1	10.03
Cat 2	10.00
Cat w Glasses	10.00
Flower	10.01
Negative	10.54
Sepia	10.00
Wide Angle View	10.00
Edge	10.01
Black and White	10.76
Cool Tone	10.42
Brighten and Contrast	10.07
Rainbow Overlay	10.76

- **Observations:** The evaluation revealed that the system maintained real-time performance for most filters, with FPS values above 10. Artistic filters like Negative, Black White, and Sepia showed good qualitative results reflecting their aesthetic and technical effectiveness. However, accessory filters like Frog Hat exhibited occasional misalignments under extreme conditions, such as rapid head movements or varying camera distances.

5. Conclusion

The Selfie Beautification Filter project demonstrated the potential of lightweight image processing techniques for real-time beautification and enhancement. The system success-

fully implemented 22 diverse filters, ranging from artistic effects to beauty filters and accessory overlays, all designed to improve the user experience in live video applications.
Achievements: The project achieved an average FPS above 10 for most filters, ensuring usability in real-time applications. Visual inspections confirmed the aesthetic effectiveness of the filters, particularly those designed for artistic transformations.
Limitations: The evaluation highlighted limitations in tracking accuracy for accessory filters under dynamic conditions. Additionally, the lack of comparable filters in existing applications constrained the scope of quantitative analysis.
Future Work: Future improvements could include integrating advanced facial tracking models, such as MediaPipe, for enhanced alignment. Expanding the dataset to include more reference filters from popular applications would enable robust quantitative evaluations. Optimizing computational efficiency further would make the system suitable for mobile platforms. [1]

References

[1] John Doe and Jane Smith. An example paper. *Journal of AI Research*, 10(3):123–145, 2025. 8

[2] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 3rd edition, 2008. 2, 3

[3] Ivan Grishchenko et al. Real-time 3d face reconstruction with mediapipe face mesh. *arXiv preprint*, 2020. 2

[4] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2010. 2

[5] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. 2

[6] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 839–846, 1998. 2

[7] Matthew Turk and Alex Pentland. Eigenfaces for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591. IEEE, 1991. 3

[8] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001. 1

[9] Xi Yin et al. Beautygan: Instance-level facial makeup transfer with deep generative adversarial network. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 645–653, 2018. 2