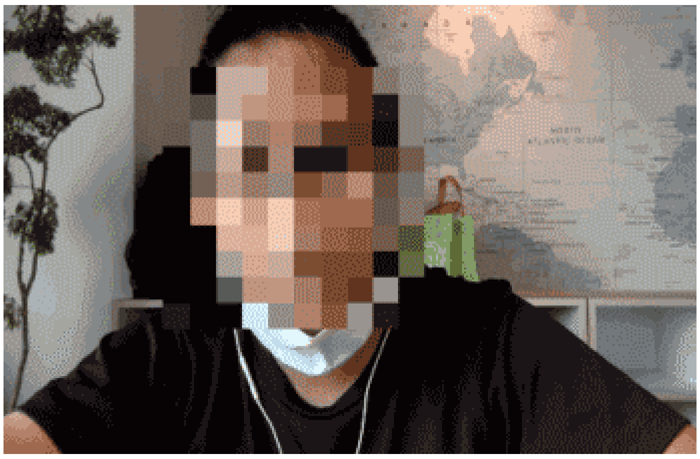
Real-Time Privacy Protection Tool

> **Team Size**: 1
>
> **Q&A**: Zhong Wanli 12332469@mail.sustech.edu.cn

Project Overview

With the rapid development of technology and the spread of camera devices, video surveillance and facial recognition technology have been widely used in many areas. However, the widespread use of these technologies has also led to privacy risks. A face, as a unique biometric feature, contains a lot of personal information. If obtained by bad person, it could result in serious privacy and security problems. Therefore, protecting facial privacy has become increasingly important.

This project aims to develop an efficient real-time privacy protection tool using C++ and OpenCV. By capturing video streams through a camera and detecting faces in real-time, this tool provides multiple privacy protection modes, including Blur, Pixelation, and Mask. Users can choose a suitable protection method according to their needs. Additionally, users can dynamically adjust processing parameters and upload custom mask images for personalized privacy protection.



Project Requirements

Technical Requirements

1. **Libraries**:
   - OpenCV (version 4.10 or higher recommended)
2. **Face Detection Model**:
   - Recommended: YuNet from OpenCV Zoo.
   - **CascadeClassifier is not allowed**: With the development of deep learning, modern methods like YuNet outperform traditional techniques in facial recognition. Reference: OpenCV Face Detection: Cascade Classifier vs. YuNet.

Program Features and Implementation Details

Feature 1: Real-Time Face Detection

- Load the YuNet model to detect faces in each video frame and return bounding boxes.
- Set default thresholds (e.g., NMS, confidence) to detect as many faces as possible.
- Refer to the C++ implementation: face_detecttion_yunet.

Feature 2: Privacy Protection Modes

Mode 1: Blur

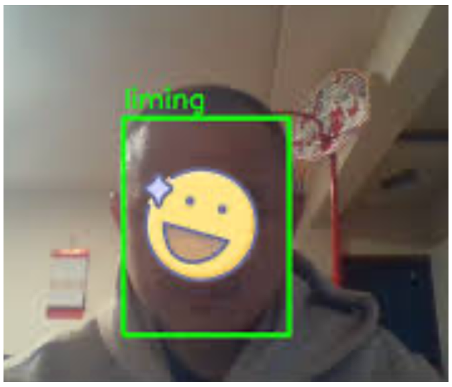
- Apply Gaussian Blur (`cv::GaussianBlur`) to blur facial regions.
- Allow dynamic adjustment of the blur kernel size.

Mode 2: Pixelation

- Use `cv::resize` to scale down and up facial regions, creating a pixelation effect.
- Allow dynamic adjustment of pixel block size.

Mode 3: Mask

- Cover facial regions with a user-uploaded or default mask image.
- Allow users to replace the mask image at runtime and automatically resize it to match the facial region.



Feature 3: Parameter Adjustment

- Dynamically adjust the following parameters during runtime:
    - **Blur kernel size**: Adjustable via the `[`/`]` keys.
    - **Pixel block size**: Adjustable via the `[`/`]` keys.
- Display the current mode and parameter values in the top-left corner of the video window.

Feature 4: Mode Switching

- Switch between different privacy protection modes using keyboard shortcuts:
    - Press `1` to switch to Blur mode.
    - Press `2` to switch to Pixelation mode.
    - Press `3` to switch to Mask mode.

Feature 5: Mask Image Upload

- Press `U` to prompt a command-line input for a new image path or upload a new mask image.
- Automatically load and resize the image to fit the facial region.

Usage Instructions

Command-Line Arguments

The program supports the following command-line arguments:

- `-mode`: Set the initial mode (`blur`, `pixel`, `mask`). Default is `blur`.
- `-blur_size`: Effective only in Blur mode. Sets the initial blur kernel size.
- `-pixel_size`: Effective only in Pixelation mode. Sets the initial pixel block size.
- `-mask_image`: Effective only in Mask mode. Specifies the path to the mask image. Default uses a predefined image.
- `-device`: Specifies the camera device index. Default is 0.

Run Examples

1. **Default Run**:

```
./privacy_protector
```

2. **Custom Parameters**:

```
./privacy_protector -mode mask -mask_image /path/to/image.png
```

Evaluation Criteria

Feature	Evaluation Criteria	Points
Face Detection	Accurate detection using a deep learning model	20
Blur Mode	Accurate blur effect with adjustable parameters	20
Pixelation Mode	Accurate pixelation effect with adjustable parameters	20
Mask Mode	Accurate mask coverage with image upload support	20
Interaction & Adjustments	Complete functionality for mode switching and parameter tuning	10
Code Quality & Documentation	Clear structure and documentation	10

Submission Requirements

Source Code: Submit all relevant source files for the project.

Project Documentation: Write a **brief** doc that explains your design choices, features, and usage instructions.