ECPS205

Final Project - Face Swap

Team : CHENG-CHIH LEE, Jonathan Kim, Sadiq Haruna, Abhijit Rao

2025/11/26

UCI MECPS

Introduction

● Purpose

This project is about utilizing Raspberry Pi and a camera module to perform face detection and swapping the detected faces with cat emojis.

● Background

This project will be utilizing:

Hardware : Raspberry Pi 5 (ARM Quadcore Cortex-A76 CPU), p5v04a camera

Software :

Raspberry Pi OS (Debian 13),
Picamera2 for frame acquisition ,
OpenCV for face detection & overlay,
NumPy

Git Repo: https://github.com/codersrule/REPLACE-HUMAN-HEADS-WITH-CAT-HEADS.git

● Functionality

1. Utilize Camera to get images (video in array)

2. Utilize face detection (derive coordinate and size)

3. Utilize image cropping from the emoji and remove background

4. Resize and paste cat faces to the coordinate

5. Consider when there are more than one faces

● System Architecture

The overall system is designed as a modular real-time image processing pipeline running on a Raspberry Pi 5. Each component performs a specific function, enabling independent testing, replacement, and debugging.

The main processing flow for a closed-loop real-time vision-processing system.:

Camera → Frame Capture → Face Detection → Cat Head Overlay → Rendering → Display Output

The following components interact to complete the head-swapping task:

• Raspberry Pi 5: Executes Python code and OpenCV operations

• Picamera2 API: Captures live frames through the CSI camera module

• FaceDetector: Identifies human faces using Haar Cascades

• Cat Sprite Processor: Extracts cat faces, removes white background

• HeadSwapper: Scales and overlays cat heads on detected faces

• OpenCV Renderer: Displays processed frames

● Module Organization

+cam.py – CameraController

• Wraps Picamera2 into a clean class interface.

• Handles camera initialization, start/stop preview, and frame capture.

• Provides capture_frame() returning NumPy arrays.

• Supports optional shot-taking and storing multiple frames.

• Acts as the main camera abstraction for the whole system.

+cat.py – Cat Sprite Loader & Alpha Blending

• Loads cat head sprites from a sprite sheet.

• Slices 16 cat images using grid-based coordinates.

• Removes white backgrounds using HSV thresholding.

• Provides cat_paste() for alpha blending onto frames.

• Returns fully processed transparent cat PNGs for overlay.

+face_detection.py – FaceDetector

• Loads Haar Cascade classifier from multiple fallback paths.

• detect_faces(): returns (x, y, w, h) for each detected face.

• draw_face_boxes(): draws detection rectangles.

• get_largest_face(): identifies biggest bounding box.

• Acts as the system's face detection module.

+fp2_headswap.py – GUI + Alternate Head Swap Mode

• Tkinter GUI providing buttons for camera operations.

• Supports Start Preview, Stop, Take Picture, Recording, Head Swap, etc.

• Uses Picamera2 or cv2 fallback depending on availability.

• Displays live stream inside Tkinter using PIL/Pillow.

• Runs threaded recording and ffmpeg MP4 conversion.

• Contains simple face-detection-based overlay preview.

+head_swap.py – HeadSwapper (Overlay Logic)

• Handles resizing and placing cat heads over detected faces.

• swap_heads(): automatic overlay with scaling + centering math.

• swap_heads_custom(): user-selected cat indices.

• change_cat_set(): cycles through cat sprite sets.

• set_scale_factor(): changes relative cat size.

• Uses cat_paste() from cat.py for final alpha blending.

+main.py – Real-Time Pipeline Orchestrator

• Initializes camera, face detector, head swapper, and output folder.

• Captures frames in a continuous loop.

• Detects faces, overlays cat heads, draws information panel.

• Supports debug bounding-box mode.

• Handles key inputs:

  • q = quit

  - s = save screenshot
  - c = change cat set
  - + / - = adjust size
  - d = debug mode toggle
- Displays final real-time cat head–swapped output with OpenCV.

● Communication Protocol

No external communication is used (no UDP/TCP/Bluetooth).

The system relies exclusively on internal memory-based exchanges.

● DATA PATH

Picamera2 → CameraController → FaceDetector → HeadSwapper → Display

● FRAME DATA STRUCTURE

- shape = (1080, 1920, 3)
- dtype = uint8 (standard RGB pixel values)

- ## KEYBOARD CONROLS

Keys are handled inside the real-time loop with cv.waitKey(1).
All controls update immediately without restarting the program.

Q — Quit Program
Immediately exits the face-swap system and closes all windows.

S — Save Screenshot
Captures the current processed frame (with cat overlays) and saves it as an image file.

C — Change Cat Set
Cycles through the available cat face styles (sprite variations).

+ — Increase Cat Size
Scales the cat faces larger relative to detected face bounding boxes.

- — Decrease Cat Size
Scales the cat faces smaller for tighter overlays.

D — Detection Debug Mode
Toggles on/off the red face bounding boxes to visualize Haar Cascade detection.