

Smile-Capture: Real-Time Smile-Triggered Photography Using OpenCV

Submitted to
Dr. Shamim Al Mamun
Professor, IIT, JU

Submitted by
Team B
25102, 25109, 25110, 25111, 25114

MEET THE TEAM



Md. Amanullah Parvez
25102



Md. Mizanur Rahaman
25109



Md. Mamunur Rushid
25110



Foysal Hasib
25111



Md. Shahidul Alom Siddiki
25114

AGENDA

- - 1 Problem statement
Slide 01
 - 2 Objective & Scope
Slide 04
 - 3 Technical Architecture
Slide 05
 - 4 Smile Detection Algorithm
Slide 06
 - 5 Results & Demo
Slide 08
 - 6 Future Enhancements
Slide 10

OBJECTIVE & SCOPE

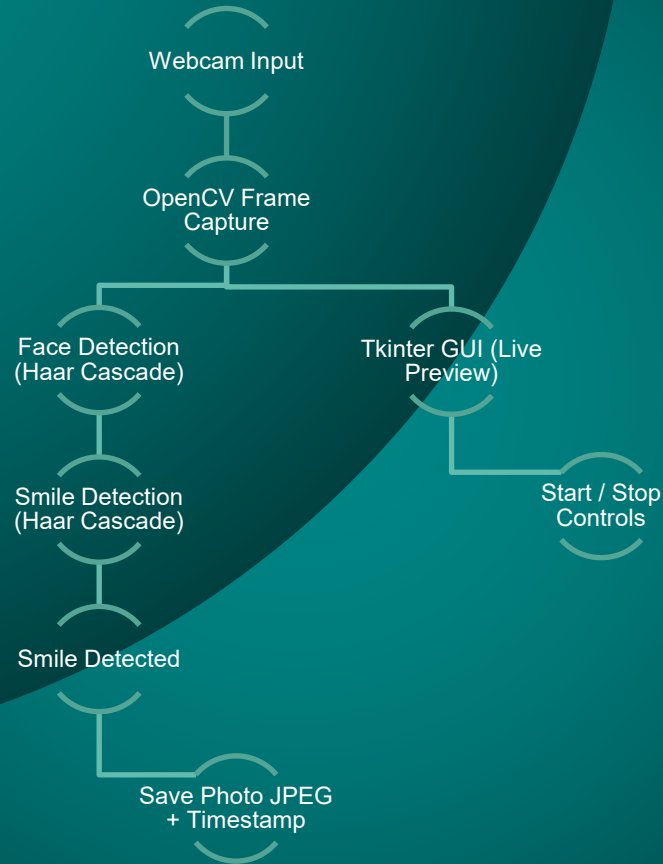
Objective

To develop a real-time smile detection application with an intuitive GUI that captures a photo automatically when a smiling face is detected using a webcam.

Scope

- Local machine (no cloud)
- Python-based
- Open-source libraries
- Single user, frontal face

TECHNICAL ARCHITECTURE

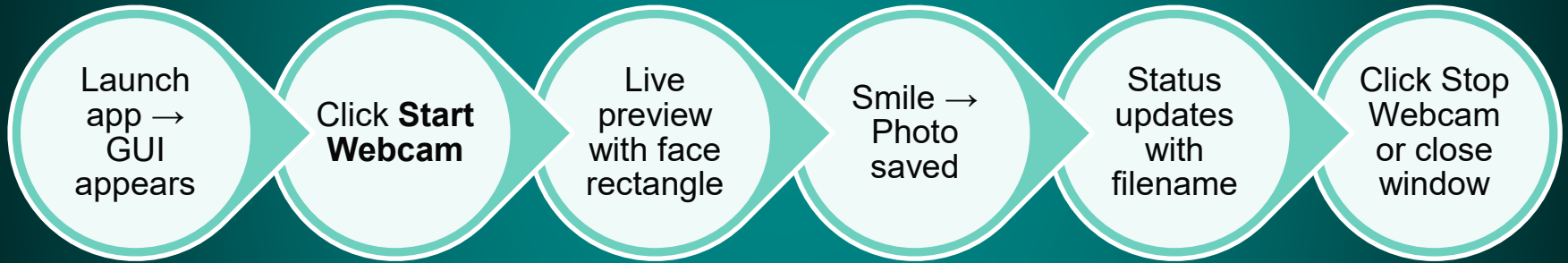


SMILE DETECTION ALGORITHM

Two-Stage Haar Cascade Pipeline

1. Face Detection
 - `haarcascade_frontalface_default.xml`
 - Parameters: `scaleFactor=1.1`, `minNeighbors=5`
2. Smile Detection (ROI)
 - `haarcascade_smile.xml`
 - Applied only within detected face
 - `minNeighbors = 55` → high confidence
 - `scaleFactor=1.7`, `minSize=(25,25)`

USER FLOW



RESULTS & DEMO

The image is a collage illustrating the results and demo of a smile detection application. It features three main components:

- Top Left:** A video frame showing a man with a beard and a black turban. A blue bounding box is drawn around his face, indicating the detected region of interest.
- Top Right:** A larger video frame showing the same man, now smiling. A blue bounding box is also drawn around his face, demonstrating the application's ability to detect smiles.
- Bottom Left:** A screenshot of the application's interface. It shows a "Start Webcam" button and a "Stop Webcam" button. A green arrow points to the "Start Webcam" button, and a text label below it says "Press 'Start Webcam' to begin".
- Bottom Center:** A screenshot of a file explorer window. It shows a folder named "captured_photos" containing two files: "smile_20251106_112147.jpg" and "smile_20251106_112148.jpg". The file "Smile and Capture.py" is also visible in the list.
- Bottom Right:** A snippet of Python code from the application, showing imports for Tkinter, PIL, NumPy, threading, and os, along with a datetime import.

```
3 from tkinter import *
4 from PIL import Image
5 import numpy as np
6 import threading
7 import os
8 from datetime import
```

BENEFITS

- **Educational:** Hands-on OpenCV & Tkinter learning
- **UX Boost:** Hands-free, emotion-triggered photos
- **Efficiency:** Real-time detection, no duplicate shots
- **Accessible:** Motor-friendly; inclusive design
- **Extensible:** Base for AI apps (IoT, mobile)
- **Cost-Free:** Webcam + open-source tools
- **Fun Factor:** Joyful, spontaneous selfies

CHALLENGES & LIMITATIONS

Challenge

Lighting variations

Multiple faces

Head tilt / glasses

Haar cascade accuracy

Webcam quality

Limitation

Requires good ambient light

Only first face processed

May reduce accuracy

Not deep learning level

Depends on hardware

FUTURE ENHANCEMENTS

Feature	Description
Multi-Person Support	Detect & save per face
Mobile App Version	Port to Android/iOS with Kivy or Flutter
Emotion Gallery	Auto-sort by happy/surprise
Cloud Backup	Upload to Google Drive/Dropbox
Voice Feedback	"Smile detected!" audio cue

```
EXPLORER
└─ AI ML
  └─ captured_photos
    └─ smile_20251106_112147.jpg
    └─ smile_20251106_112148.jpg
  └─ Smile and Capture.py

Smile and Capture.py
1 import cv2
2 import tkinter as tk
3 from tkinter import ttk, messagebox
4 from PIL import Image, ImageTk
5 import numpy as np
6 import threading
7 import os
8 from datetime import datetime
9
10 # -----
11 # 1. Haar cascades (global)
12 # -----
13 FACE_CASCADE = cv2.CascadeClassifier(
14     cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
15 )
16 SMILE_CASCADE = cv2.CascadeClassifier(
17     cv2.data.haarcascades + "haarcascade_smile.xml"
18 )
19
20 SMILE_CONFIDENCE = 55 # higher = stricter
21
22
23 def detect_smile(frame: np.ndarray) -> bool:
24     """Return True if a confident smile is found."""
25     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
26     faces = FACE_CASCADE.detectMultiScale(
27         gray, scaleFactor=1.1, minNeighbors=5, minSize=(100, 100)
28     )
29     for (x, y, w, h) in faces:
30         roi_gray = gray[y:y+h, x:x+w]
31         smiles = SMILE_CASCADE.detectMultiScale(
32             roi_gray,
33             scaleFactor=1.7,
34             minNeighbors=SMILE_CONFIDENCE,
35             minSize=(25, 25),
36         )
37         if len(smiles) > 0:
38             return True
39     return False
40
Smile and Capture.py
1
2 # 2. GUI + webcam thread
3 # -----
4
5 class SmileCaptureApp:
6
7     def __init__(self, root: tk.Tk):
8         self.root = root
9         self.root.title("Smile-Capture")
10         self.root.geometry("800x600")
11         self.root.configure(bg="#f0f0f0")
12
13         # --- UI elements ---
14         self.lbl_video = ttk.Label(root)
15         self.lbl_video.pack(padx=10, pady=10, expand=True, fill="both")
16
17         # Buttons frame
18         btn_frame = ttk.Frame(root)
19         btn_frame.pack(pady=5)
20
21         self.btn_start = ttk.Button(btn_frame, text="Start Webcam", command=self.start_webcam)
22         self.btn_start.pack(side="left", padx=5)
23
24         self.btn_stop = ttk.Button(btn_frame, text="Stop Webcam", command=self.stop_webcam, state="disabled")
25         self.btn_stop.pack(side="left", padx=5)
26
27         self.lbl_status = ttk.Label(root, text="Press 'Start Webcam' to begin", foreground="blue")
28         self.lbl_status.pack(pady=5)
29
30         # --- State ---
31         self.cap: cv2.VideoCapture | None = None
32         self.running = False
33         self.photo_dir = "captured_photos"
34         os.makedirs(self.photo_dir, exist_ok=True)
35
36     def start_webcam(self):
37         if self.running:
38             return
39
40         self.cap = cv2.VideoCapture(0)
41         if not self.cap.isOpened():
42             messagebox.showerror("Error", "Could not open webcam.")
43             return
44
Smile and Capture.py
1
2 class SmileCaptureApp:
3
4     def start_webcam(self):
5         if not self.cap.isOpened():
6             messagebox.showerror("Error", "Could not open webcam.")
7             return
8
9         self.running = True
10        self.btn_start.config(state="disabled")
11        self.btn_stop.config(state="normal")
12        self.lbl_status.config(text="Webcam ON | smile to capture!", foreground="green")
13
14        # start video loop in a background thread
15        threading.Thread(target=self._video_loop, daemon=True).start()
16
17    def stop_webcam(self):
18        self.running = False # <- this will break the while loop
19
20    def _video_loop(self):
21        smile_detected = False
22        while self.running:
23            ret, frame = self.cap.read()
24            if not ret:
25                break
26
27            # mirror
28            frame = cv2.flip(frame, 1)
29
30            # draw face rectangles
31            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
32            faces = FACE_CASCADE.detectMultiScale(
33                gray, scaleFactor=1.1, minNeighbors=5, minSize=(100, 100)
34            )
35            for (x, y, w, h) in faces:
36                cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 215, 0), 2)
37
38            # smile detection + capture
39            if not smile_detected and detect_smile(frame):
40                smile_detected = True
41                self.root.after(0, self._capture_photo, frame.copy())
42            elif smile_detected and not detect_smile(frame):
43                smile_detected = False
44
45    def _capture_photo(self, frame: np.ndarray):
46        # save photo to disk
47        flash_update_status()
48        flash_update_status()
49
50        w = self.root.winfo_screenwidth()
51        h = self.root.winfo_screenheight()
52        x = (w - 200) // 2
53        y = (h - 100) // 2
54        flash_geometry(f"200x100x{x}|{y}")
55        flash_after(200, flash_destroy)
56
57    def _cleanup(self):
58        if self.cap:
59            self.cap.release()
60            self.cap = None
61
62        self.running = False
63        self.root.after(0, lambda: self.btn_start.config(state="normal"))
64        self.root.after(0, lambda: self.btn_stop.config(state="disabled"))
65        self.root.after(
66            0,
67            lambda: self.lbl_status.config(
68                text="Webcam stopped. Press 'Start Webcam' again.", foreground="blue"
69            ),
70        )
71
72    def on_closing(self):
73        self.running = False
74        if self.cap:
75            self.cap.release()
76        self.root.destroy()
77
78 # 3. Run the app
79 if __name__ == "__main__":
80     root = tk.Tk()
81     app = SmileCaptureApp(root)
82     root.protocol("WM_DELETE_WINDOW", app.on_closing)
83     root.mainloop()
```

GitHub Link:

<https://github.com/amanullahradiant/assignment-smile-detection-and-capture>

The background is a solid teal color with three large, overlapping circles. One circle is in the top center, another is on the left side, and a third is on the right side. The circles have a slight gradient, being darker at the edges.

Thank You!

Open discussion for audience feedback on the presentation.