

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

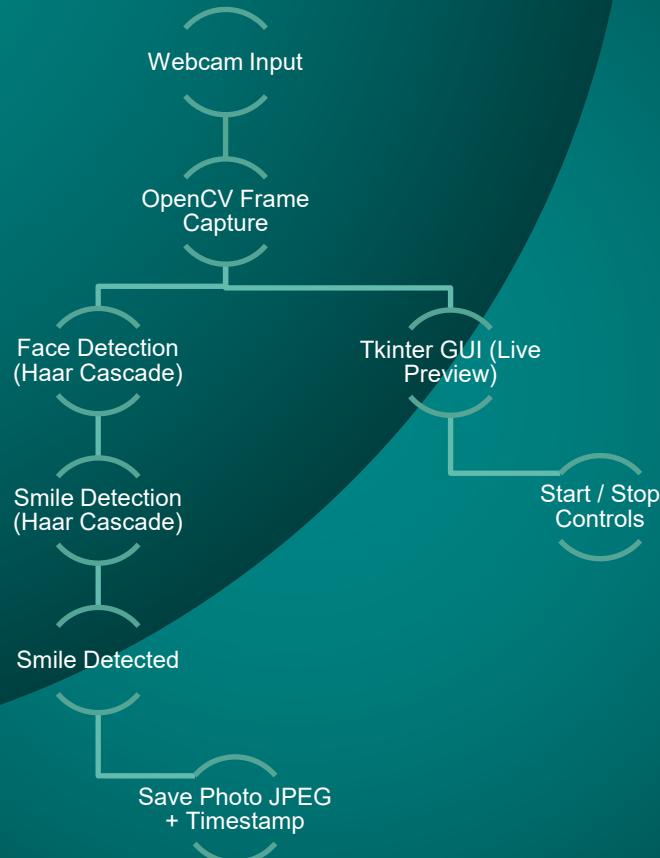
Objective

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

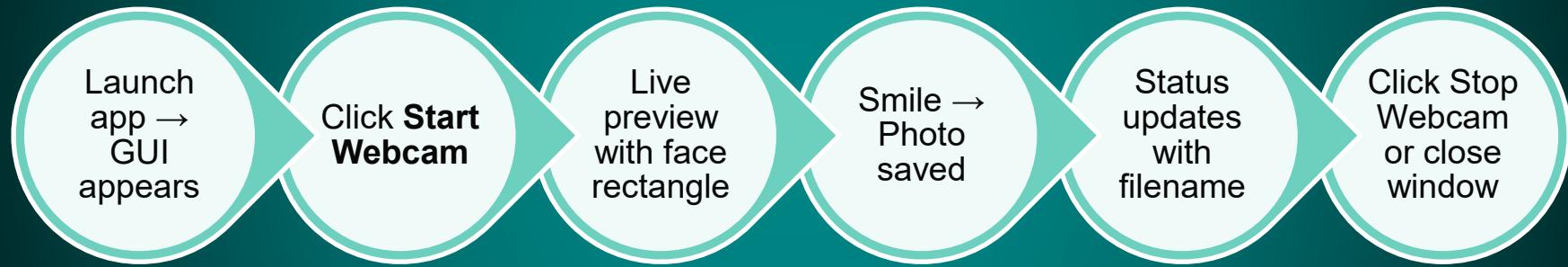


SMILE DETECTION ALGORITHM

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 screenshot shows a Python application window titled "Smile and Capture.py". Inside the window, a video feed from a webcam displays a man with a beard and a black cap. A blue rectangular box highlights his face. Below the video feed are two buttons: "Start Webcam" and "Stop Webcam". A green arrow points to the "Start Webcam" button. To the right of the application window is a file explorer window titled "EXPLORER". The "AI ML" folder is expanded, showing a "captured_photos" folder containing two files: "smile_20251106_112147.jpg" and "smile_20251106_112148.jpg". At the bottom of the file list is the "Smile and Capture.py" file, which is currently selected and highlighted in blue. In the bottom right corner of the slide, there is a vertical teal bar with the number "8" at the bottom.

```
from tkinter import *
from PIL import Image
import numpy as np
import threading
import os
from datetime import datetime
# -----
# 1. Haar cascades (
# -----
```

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

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 SmileyCaptureApp:
2     class SmileCaptureApp:
3         def __init__(self):
4             self.root = tk.Tk()
5             self.root.title("SmileyCapture")
6             self.root.geometry("800x600")
7             self.root.configure(bg="#F0F0F0")
8
9             # ----- UI elements -----
10             self.lbl_video = tk.Label(self.root)
11             self.lbl_video.pack(padx=10, pady=10, expand=True, fill="both")
12
13             # Buttons frame
14             btn_frame = tk.Frame(self.root)
15             btn_frame.pack(pady=5)
16
17             self.btn_start = tk.Button(btn_frame, text="Start Webcam", command=self.start_webcam)
18             self.btn_start.pack(side="left", padx=5)
19
20             self.btn_stop = tk.Button(btn_frame, text="Stop Webcam", command=self.stop_webcam, state="disabled")
21             self.btn_stop.pack(side="left", padx=5)
22
23             self.lbl_status = tk.Label(self.root, text='Press "Start Webcam" to begin.', foreground="blue")
24             self.lbl_status.pack(pady=5)
25
26             # ----- State -----
27             self.cap: cv2.VideoCapture | None = None
28             self.running = False
29             self.photo_dir = "captured_photos"
30             os.makedirs(self.photo_dir, exist_ok=True)
31
32             # start_webcam(self):
33             if self.running:
34                 return
35
36             self.cap = cv2.VideoCapture(0)
37             if not self.cap.isOpened():
38                 messagebox.showerror("Error", "Could not open webcam.")
39             return
40
```

```
Smile and Capture.py
45 class SmileCaptureApp:
46     def __init__(self, frame: np.ndarray):
47         self.frame = frame
48         self.video_loop(self)
49         while self.running:
50             # show in GUI
51             self.lbl_video.config(image=self.frame)
52             self.lbl_video.update()
53             self.lbl_video.pack()
54
55             # capture photo(self, frame: np.ndarray):
56             timestamp = datetime.now().strftime("%Y-%m-%d %H-%M-%S")
57             filename = os.path.join(self.photo_dir, f"smile_{timestamp}.jpg")
58             cv2.imwrite(filename, frame)
59
60             self.lbl_status.config(
61                 text=f"Smile Photo saved [{os.path.basename(filename)}]",
62                 foreground="purple",
63             )
64
65             # ----- Flash effect (optional) -----
66             flash = tk.Toplevel(self.root)
67             flash.configure(bg="white")
68             flash.geometry("200x100")
69             flash.update_idletasks()
70             flash.wm_title("Flash")
71             w = self.root.winfo_screenwidth()
72             h = self.root.winfo_screenheight()
73             x = (w - 200) // 2
74             y = (h - 100) // 2
75             flash.geometry(f"200x100+{x}+{y}")
76             flash.after(200, flash.destroy)
77
78             # ...
79
80             def _clean_up(self):
81                 if self.cap:
82                     self.cap.release()
83                     self.cap = None
84
85                 self.running = False
86                 self.root.after(0, lambda: self.btn_start.config(state="normal"))
87                 self.root.after(0, lambda: self.btn_stop.config(state="disabled"))
88                 self.root.after(0, lambda: self.lbl_status.config(
89                     text="Webcam stopped. Press 'Start Webcam' again.", foreground="blue"
90                 ))
91
92             def on_closing(self):
93                 if self.cap:
94                     self.cap.release()
95                     self.root.destroy()
96
97             # ...
98
99             def _main(self):
100                 if __name__ == "__main__":
101                     root = tk.Tk()
102                     app = SmileCaptureApp(root)
103                     app.protocol("WM_DELETE_WINDOW", app.on_closing)
104                     root.mainloop()
```

```
Smile and Capture.py
105
106         # start video loop in a background thread
107         threading.Thread(target=self._video_loop, daemon=True).start()
108
109         # stop_webcam(self):
110         if self.running:
111             self.running = False
112             # ... this will break the while loop
113
114         # ...
115         def _video_loop(self):
116             smile_detected = False
117             while self.running:
118                 ret, frame = self.cap.read()
119                 if not ret:
120                     break
121
122                 # mirror
123                 frame = cv2.flip(frame, 1)
124
125                 # draw face rectangles
126                 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
127                 faces = FACE.Cascade.detectMultiScale(
128                     gray, scaleFactor=1.1, minNeighbors=5, minSize=(100, 100)
129                 )
130
131                 for (x, y, w, h) in faces:
132                     cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 215, 0), 2)
133
134                 # smile detection + capture
135                 if not smile_detected and detect_smile(frame):
136                     smile_detected = True
137                     self.root.after(0, lambda: self._capture_photo(frame.copy()))
138
139                 elif smile_detected and not detect_smile(frame):
140                     smile_detected = False
141
142             # show in GUI
```

GitHub Link:

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

Thank You!

Open discussion for audience feedback on the presentation.