TITLE

SMART TARGETING SYSTEM

Submitted to: Mr. Sagar Pandey

Name: Sai Vikram Patnaik

Student ID: 11802375

Roll no.: 58

Section :K18KK

Email Address : saivikram.patnaik@gmail.com

GitHub Link: https://github.com/Vikram126/Smart-Targetting-System

Table of contents

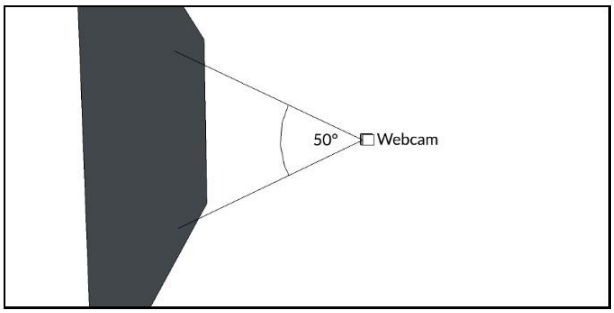
Abstract

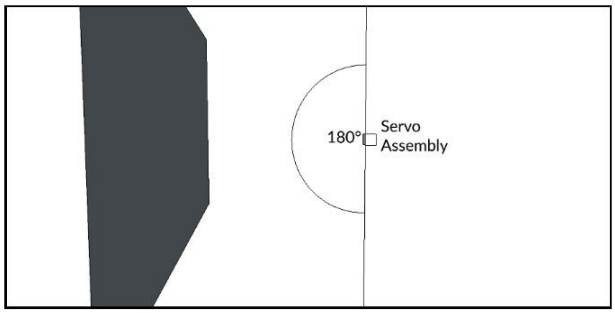
The Smart Targeting System (STS) is a find and point system which could point, or in some cases trigger special actions when a certain criterion is met.

The system I designed is made specifically to point at faces. It is a two-part project.

1. The code: A python file and a C++ file(Arduino wrapper)
2. The model

The model is connected to the computer (wired/wireless). The user is prompted if he/she wants to use an IPcam or laptop camera. The turret should be placed accordingly. The camera streams images to the computer (If IPcam is selected, lower the stream quality to 320p for minimum lag). The python script detects a face and the coordinates are relayed to a special translation logic which converts coordinates on the image to angles. This data is later relayed to the microcontroller using this data structure: "X(x°)Y(y°)". E.g. X45Y120. The Arduino code parses the data structure and relays them to the microcontroller.





❖ Introduction:

WHAT DOES MY PROJECT CONSISTS OF?

The Smart Targeting System (STS) combines the following into a single unit.

- A Facial Recognition Program
- Translation of 2D coordinates to a point in 3D Space
- A turret made using
    - A microcontroller
    - Servo motors (for pan and tilt)
    - A shooting mechanism or a laser
- A User Interface (for real time use)
- A real time video stream

❖ Objective:

The STS's facial recognition processes a video stream from a device (like a smart phone) and searches for the faces of the targets (previously trained from a data set).

When the target is acquired, the STS calculates the coordinates regarding where to shoot and translates them into angular measure.

This data is then put in a special data structure, serialized and sent to the microcontroller.

❖ Motivation:

This project was conceived as a means to aid the military. When employed the STS can detect a specific attribute(Face in my case) about the target and carry out special instruction as per requirement. This will reduce human casualties. The design was inspired by THE TERMINATOR (1984).

The idea was to create something malleable so as to fit any purpose. STS can not only be programmed to be a child's toy but also an advanced form of weaponry.

❖ Implementation of the project

PYTHON:

```
import tkinter as tk
from tkinter import simpledialog as sd
from tkinter import messagebox as mb
import cv2
import PIL.Image, PIL.ImageTk
import time
import serial
import sys
#face detection using haar cascade classifier
faceCascade=cv2.CascadeClassifier(r"Face2.xml")

#arduino Connection
try:
#Change COM port and baud rate as per connections
    ard=serial.Serial('COM6',9600)
#let it connect
    time.sleep(2)
except Exception:
    print("Arduino not connected")
    sys.exit(0)
class Turret:
    def __init__(self, window, window_title, video_source=0):
        self.window = window
        self.window.title(window_title)
        self.video_source = video_source
        # open video source (by default this will try to open the
computer webcam)
        self.vid = MyVideoCapture(self.video_source)
        # Create a canvas that can fit the above video source size
        self.canvas = tk.Canvas(window, width = self.vid.width,
height = self.vid.height)
        self.canvas.pack()
        # Button that lets the user take a snapshot
        self.btn_snapshot=tk.Button(window, text="Take a Picture",
width=50, command=self.snapshot)
        self.btn_snapshot.pack(anchor=tk.CENTER, expand=True)
        # After it is called once, the update method will be
automatically called every delay milliseconds
        self.delay = 15
        self.update()
        self.window.mainloop()
    def snapshot(self):
        # Get a frame from the video source
        ret, frame = self.vid.get_frame()
        if ret:
            cv2.imwrite("frame-" + time.strftime("%d-%m-%Y-%H-%M-
%S") + ".jpg", cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
                        )
    def update(self):
        # Get a frame from the video source
        ret, frame = self.vid.get_frame()
```

```
        if ret:
            self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(frame))
            self.canvas.create_image(0, 0, image = self.photo,
anchor = tk.NW)
            self.window.after(self.delay, self.update)



class MyVideoCapture:
    def __init__(self, video_source=0):
         # Open the video source
        self.vid = cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source",
video_source)
         # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)
        #Scaling constants
        self.xScaleConst = (self.width) / 180;
        self.yScaleConst = (self.height) / 180;

    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                # Return a boolean success flag and the current
frame converted to BGR
                gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
                faces=faceCascade.detectMultiScale(gray,1.3,5)
                for (x,y,w,h) in faces:

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                    a,b=((x+w)/2)/self.xScaleConst,
((y+h)/2)/self.yScaleConst
                    message="X{0}:Y{1}".format(a,b)
                    ard.write(message,"utf8")
                return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            else:
                return (ret, None)
        else:
            return (ret, None)
     # Release the video source when the object is destroyed
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()
#to validate ip addresses
def validate_ip(s):
    a,b = s.split(':')
    a="".join(a).split(".")
    if len(a) != 4:
        return False
    for x in a:
        if not x.isdigit():
            return False
```

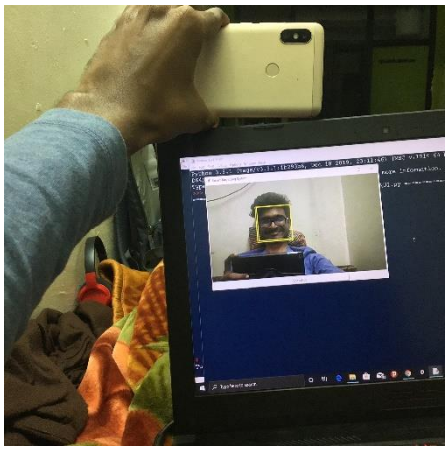
```
        i = int(x)
        if i < 0 or i > 255:
            return False
    return True

root = tk.Tk()   # create a GUI window
root.geometry("300x150") # set the configuration of GUI window
root.title("Select Feed") # set the title of GUI window

def call(f_id):
    if f_id==1:
        root.destroy()
        Turret(tk.Tk(), "Turret")
    elif f_id==2:
        ip=sd.askstring("Enter IP address","https://",parent=root)
        if validate_ip(ip):
            ip="https://{}/video".format(ip)
            root.destroy()
            Turret(tk.Tk(), "Turret",ip)
        else:
            mb.showinfo("Error","Enter valid Ip")
            call(2)

# create a Form label
tk.Label(text="Laptop Cam Or IP Webcam", bg="#90EE90", width="300",
height="2", font=("Forte", 13)).pack()
tk.Label(text="").pack()

p=[tk.PhotoImage(file=r"Icons\laptop.png"),tk.PhotoImage(file=r"Icon
s\cctv.png")]
q=[l.subsample(10,10) for l in p]

b1=tk.Button(root,text="Laptop Cam",command=lambda :
call(1),image=q[0],compound=tk.TOP).pack(padx=50,pady=5,side=tk.LEFT
)
b2=tk.Button(root,text='IP Webcam',command=lambda :
call(2),image=q[1],compound=tk.TOP).pack(padx=2,pady=5,side=tk.LEFT)
root.mainloop() # start the GUI
```

Arduino code:

```
#include<Servo.h>

Servo serX;

Servo serY;

String Data;

void setup() {
```

```
  serX.attach(7);
  serY.attach(10);
  Serial.begin(9600);
  Serial.setTimeout(10);
}
void loop() {
  //unrequired but necessary
}
void serialEvent() {
  Data = Serial.readString();
  serX.write(parseX(Data));
  serY.write(parseY(Data));
}
int parseX(String data){
  data.remove(data.indexOf("Y"));
  data.remove(data.indexOf("X"), 1);
  return data.toInt();
}
int parseY(String data){
  data.remove(0,data.indexOf("Y") + 1);
  return data.toInt();
}
```

❖ Output:



❖ Scope:

The STS is a find-and-point system and can be extended into various real time cases like :

- Threat elimination
- Alarm System
- Games like “Where’s Waldo?”
- Remainder Systems
- Monitoring Systems

❖ Work Distribution:

As this was a one-person project, all of the work was undertaken by me.

That consisted of

- Conception and design
- Creating the UI
- Creating a facial recognition module
- Programming the microcontroller
- Constructing the structure
- Model development and improvement

❖ Libraries Used:

1. Opencv
2. Tkinter
3. Pillow
4. Time
5. PySerial
6. Sys

❖ GitHub link:

https://github.com/Vikram126/Smart-Targetting-System

❖ References:

1. Python Programming for Arduino by Pratik Desai
2. Python notes for beginners (Goalkickers)
3. Michael Reeves (Full Stack Software Developers)