ISSN NO: 2770-0003
Date of Publication: 11-11-2023

Model Of Recognition Face Spoofing

Khan Suliman.
Computer science and English as the foreign language Teacher at Hajiabad Public School. Timergara, Lower Dir. KPK. Pakistan. Master Degree Graduator from Jiangsu University of Science and Technology. Zhenjiang. Jiangsu. China.
sulimankhan041993@outlook.com

Abstract: This article argues model of recognition face spoofing by methods of stream demonstration and Haar cascades. These methods created with comfort and rapidity, can be functional to any face spoofing on program Python, which we use with OpenCV to stream “Motion JPEG” through standard output.
Keywords: face spoofing, recognition, application, protection.

Introduction

In recent years, computer vision in the Republic of Uzbekistan has gained popularity and stood out in a separate direction. Developers are creating new applications that are used around the world.

In this regard, the concept of open source is attractive. Even technology giants are ready to share new discoveries and innovations with everyone, so that technology does not remain the privilege of the rich.

One such technology is face recognition. When used correctly and ethically, this technology can be applied in many areas of life.

It all depends on creating an effective face recognition algorithm using open source tools.

Face spoofing in social networks changed tagging images manually to automatically generated tag suggestions for each image uploaded to the platform. Social networks use a simple face recognition algorithm to analyze pixels in an image and compare it with their respective users.

Face spoofing in the field of security is a simple example of using face spoofing technology to protect personal data - unlocking a smartphone “by face”.

This technology can also be implemented in the access system: a person looks into the camera, and it determines whether to enter it or not.

Also, face spoofing technology can be used to count the number of people attending an event (such as a conference or concert). Instead of manually counting participants, we install a camera that can capture images of the faces of participants and give out the total number of visitors. This will help automate the process and save time.

Methods Of Research

We developed a face spoofing system with using two methods camera video stream demonstration and Haar cascades.

In the first example, we will try to create an application that opens a window with a demonstration of the video stream of the camera.

First, we imported the libraries needed for the job.

import(
“log”
“gocv.io/x/gocv”)

After that, we created the Video Capture object using the Video Capture Device function. The latter makes it possible to capture the video stream using the camera. The function uses an integer as a parameter (it represents the device ID).

```
webcam, err := gocv.VideoCaptureDevice(0) if err != nil { log.Fatalf(“error opening web cam: %v”, err) } defer webcam.Close()
```

After that, an n-dimensional matrix was created. It will store images read from the camera.

```
img := gocv.NewMat() defer img.Close()
```

ISSN NO: 2770-0003
Date of Publication: 11-11-2023

Toward display the video stream, you need to create a window - this can be done using the New Window function.

```
window := gocv.NewWindow(“webcamwindow”) defer window.Close()
```

Since the video was a continuous stream of image frames, we created an endless loop for reading the camera’s video stream endlessly. To do this, you need the Read method of type Video Capture. It will wait for the type Mat (the matrix we created above), returning a boolean value indicating whether the frame from Video Capture was read successfully or not.

```
for { if ok := webcam.Read(&img); !ok || img.Empty( { log.Println(“Unable to read from the webcam”) continue } . . . }
```

After the frame was displayed in the created window. Pause to go to the next frame - 50 ms.

window.IMShow(img)
window.WaitKey(50)

After starting the application, a window with a video stream from the camera will open.

```
package main import ( "log" "gocv.io/x/gocv" ) func main() { webcam, err := gocv.VideoCaptureDevice(0) if err != nil { log.Fatalf("error opening device: %v", err) } defer webcam.Close() img := gocv.NewMat() defer img.Close() window := gocv.NewWindow("webcamwindow")
```

According to the second method, a face recognition system was built based on the Haar cascade.

Haar cascades are cascading classifiers that are trained based on the Haar wavelet technique. They analyze the pixels in the image to detect certain signs.

In the current example, cascades will be used to identify a person’s face in face.

```
harrcascade := “opencv_haarcascade_frontalface_default.xml”classifier := gocv.NewCascadeClassifier()classifier.Load(harrcascade) defer classifier.Close()
```

Findings

In Python, we can use with OpenCV to stream “Motion JPEG” through standard output.

“Motion JPEG (M-JPEG)” sounds like something complicated and fantastic. Although in reality it’s just a concatenation of frames into one JPEG with certain use of delimiters. Most of all, it looks like a CSV for video.

Detection faces in the image you need to use the DetectMultiScale method. This function takes a frame (type Mat) and returns an array of type Rectangle. The size of the array represents the number of faces that can be found in the frame. A rectangle on the console that creates a border around the detected rectangle. Rectangle. It will accept a mat that reads the camera, a Rectangle object that was the return method of the Detect Multi Scale, the color and thickness for the border.

In order to do this, you need to create a classifier.

```
for _, r := range rects { fmt.Println(“detected”, r) gocv.Rectangle(&img, r, color, 2) }
```

```
package main import ( "fmt" "image/color" "log" "gocv.io/x/gocv" ) func main() { webcam, err := gocv.VideoCaptureDevice(0) if err != nil { log.Fatalf("error opening web cam: %v", err) } defer webcam.Close() img := gocv.NewMat() defer img.Close() window := gocv.NewWindow("webcamwindow") defer window.Close() harrcascade := "opencv_haarcascade_frontalface_default.xml" classifier := gocv.NewCascadeClassifier() classifier.Load(harrcascade) defer classifier.Close() color := color.RGBA{0, 255, 0, 0} for { if ok := webcam.Read(&img); !ok || img.Empty() { log.Println("Unable to read from the device") continue } rects := classifier.DetectMultiScale(img) for _, r := range rects { fmt.Println("detected", r) gocv.Rectangle(&img, r, color, 3) } window.IMShow(img) window.WaitKey(50) } }
```

ISSN NO: 2770-0003
Date of Publication: 11-11-2023

Conclusions

Thus, our model of recognition face spoofing system, created with ease and speed, can be applied to any face spoofing.

References

1. http://mitc.uz/ru/pages/info_security
2. Decree of the President of the Republic of Uzbekistan dated November 21, 2018 N PP-4024 "On measures to improve the control system for the implementation of information technologies and communications, the organization of their protection"
3. Jasur Gayrat ugli Rashidov" Cryptography as protection web sites from cyber-attack", Study manual. "Publishing house". pp. 289,2017, Tashkent, Uzbekistan.