Liveness Detection with OpenCV

C.Anuradha, R.Kavitha, A.V.Allen Geo, S.R.Srividhya

Abstract: Biometric system is widely used to recognize the authorized person based on either behavioural characteristics or physical. But this can be spoofed using various traits. Spoofing attack is nothing but attacking or harming biometric recognition system using security features to use system without permission of authorized user. Images or video of person can be easily available from social media or can be easily captured from some distance. Consider what would happen if a nefarious user tried to purposely circumvent your face recognition system. Such a user could try to hold up a photo of another person. Maybe they even have a photo or video on their smart phone that they could hold up to the camera responsible for performing face recognition. In the project, we have tried spotting these “fake” versus “real/legitimate” faces and how we could apply anti-face spoofing algorithms into our facial recognition applications by applying live ness detection[1] with Open CV.

Keywords: Biometric System, Optical Flow Algorithms, Local Binary Patterns(LBP)

I. INTRODUCTION

In order to make face recognition systems more secure, we need to be able to detect such fake/non-real faces — liveness detection is the term used to refer to such algorithms.

There are a number of approaches to liveness detection, including:

- Texture analysis, including computing Local Binary Patterns (LBPs) over face regions and using an SVM to classify the faces as real or spoofed.
- Frequency analysis, such as examining the Fourier domain of the face.
- Variable focusing analysis, such as examining the variation of pixel values between two consecutive frames.
- Heuristic-based algorithms, including eye movement, lip movement, and blink detection. These set of algorithms attempt to track eye movement and blinks to ensure the user is not holding up a photo of another person (since a photo will not blink or move its lips).
- Optical Flow algorithms[2], namely examining the differences and properties of optical flow generated from 3D objects and 2D planes.
- 3D face shape, similar to what is used on Apple’s iPhone face recognition system, enabling the face recognition system to distinguish between real faces and printouts/photos/images of another person.

II. OBJECTIVES

The aim of the dissertation “Live ness Detection using OpenCV”[3] is to discuss liveness detection, including what it is and why we need it to improve our face recognition systems.

From there we’ll review the dataset we’ll be using to perform liveness detection, including:

- How to build to a dataset for liveness detection
- Our example real versus fake face images

We’ll also review our project structure for the liveness detector project as well[3]. In order to create the liveness detector, we’ll be training a deep neural network capable of distinguishing between real versus fake faces[4].

III. PROBLEM STATEMENT

In this project we will focus on How would we go about spotting these “fake” versus “real/legitimate” faces? How could we apply anti-face spoofing algorithms into your facial recognition applications? [5]The answer is to apply liveness detection with Open CV. [6]We’ll be treating liveness detection as a binary classification problem.[7]

IV. EXISTING SYSTEM

The currently existing Face recognition technology generally involves[8]

- Image capture
  The first step is to acquire the facial image of user from the camera.
- Face detection
  At the second step, face is detected from the acquired image. It can also be normalized or enhanced for further processing[9].
- Feature Extraction
  At the third step, face recognition process takes place in which the desired facial features are extracted.[11]
- Matching
  These extracted features are matched against the features stored in the database.
- Determine identity
  Finally, the output of face recognition process is used(if there is a match or not) to determine the identity of the person.[12]

A. Disadvantages Of Existing System

The problem with current system is, A face recognition system is also prone to the
spoofing attacks[13]. Our biometric facial data can be easily stolen from social sites and other personal websites. Most common attack on face recognition system is the photograph attack i.e. placing photographs[14] in front of camera. Other facial spoofing attacks are playing video of genuine user in front of camera and using 3D dummy faces or mask.[15]

V. PROPOSED SYSTEM

In order to minimize such problems, liveness detection is integrated within the system. Method of liveness detection detect physiological signs of life from face ensuring that only live face samples are stored for enrolment or authentication[16]. For the PROJECT, we’ll be treating liveness detection as a binary classification problem[17]. Given an input image, we’ll train a Convolutional Neural Network capable of distinguishing real faces from fake/spoofed faces.[18]

A. Architectural Design Specification

In order to build the liveness detection dataset, I:

- Took my Phone and put it in portrait/selfie mode.[29]
- Recorded a ~25-second video of myself walking around my office.[21]
- Replayed the same 25-second video, this time facing my iPhone towards my desktop where I recorded the video replaying.
- This resulted in two example videos, one for “real” faces and another for “fake/spoofed” faces[20].

Finally, I applied face detection to both sets of videos to extract individual face ROIs for both classes Fig:1.

VI. FLOWCHART DIAGRAM

![Flowchart Diagram](Image)

Fig:1 Architecture Diagram

VI. ALGORITHM SPECIFICATION

In gather_examples.py:

Lines 2-5 import our required packages. This script only requires Open CV and Num Py in addition to built-in Python modules.[23]

From there Lines 8-19 parse our command line arguments:

- `--input`: The path to our input video file.[24]
- `--output`: The path to the output directory where each of the cropped faces will be stored.

--detector: The path to the face detector. We’ll be using Open CV’s deep learning face detector.[25]
--confidence: The minimum probability to filter weak face detections. By default, this value is 50%.
--skip: We don’t need to detect and store every image because adjacent frames will be similar. Instead, we’ll skip N frames between detections. We can alter the default of 16 using this argument[26].

Now load the face detector and initialize our video stream[28]. Lines 23-26 load Open CV’s deep learning face detector.[30]

From there we open our video stream on Line 30. We also initialize two variables for the number of frames read as well as the number of frames saved while our loop executes (Lines 31 and 32). Now create a loop to process the frames: Our while loop begins on Lines 35.

VIII. CONCLUSION

Using this liveness detector we can now spot fake fakes and perform anti-face spoofing in your own face recognition systems.

To create our liveness detector we utilized Open CV, Deep Learning, and Python.

The first step was to gather our real vs. fake dataset. To accomplish this task, for this:
1) First recorded a video of ourselves using our smartphone (i.e., “real” faces).
2) Held the smartphone up to our laptop/desktop, replayed the same video, and then recorded the replaying using our webcam (i.e., “fake” faces).
3) Applied face detection to both sets of videos to form our final liveness detection dataset.

After building our dataset we implemented, “LivenessNet”, a Keras + Deep Learning CNN.

This network is purposely shallow, ensuring that:
1) We reduce the chances of over fitting on our small dataset.
2) The model itself is capable of running in real-time (including on the Raspberry Pi).

Overall, our liveness detector was able to obtain 99% accuracy on our validation set.

To demonstrate the full liveness detection pipeline in action a Python + OpenCV[9] was created, script that loaded our liveness detector and applied it to real-time video streams. As demo showed, the liveness detector was capable of distinguishing between real and fake faces.[41][42]

REFERENCES

International Journal of Innovative Technology and Exploring Engineering (IJIITEE)  
ISSN: 2278-3075, Volume-8, Issue-953, July 2019


[36] Sundarraj, B., Kalyanamurthie, K.P, Social network analysis for decisive the ultimate classification from the ensemble to boost accuracy rates 2016 International Journal of Pharmacy and Technology


[38] Sundarraj, B., Kalyanamurthie, K.P, Remote sensing imaging for satellite image segmentation 2016 International Journal of Pharmacy and Technology


AUTHORS PROFILE

C.Anuradha, Assistant Professor, Department of Computer Science & Engineering, Bharath Institute of Higher Education and Research, Chennai, India

R.Kavitha, Associate Professor, Department of Computer Science & Engineering, Bharath Institute of Higher Education and Research, Chennai, India

A.V.Allin Geo, Assistant Professor, Department of Computer Science & Engineering, Bharath Institute of Higher Education and Research, Chennai, India

S.R.Srividhya, Assistant Professor, Department of Computer Science & Engineering, Bharath Institute of Higher Education and Research, Chennai, India

Published By:  Blue Eyes Intelligence Engineering & Sciences Publication