Facial Liveness Testing For The Web

Ryan Collins
Department of Computer Science, Durham University

What are Facial Liveness Tests?

Liveness tests are a method of testing whether an input image has been spoofed for malicious intent. Methods such as holding a piece of paper in front of a camera, holding a video of a subject in front of a camera, or wearing a mask of the subject are a few ways that existing facial recognition systems can be fooled into unlocking a device/system for a malicious third party. Facial liveness tests are the answer, aimed at detecting these attacks, based on a variety of different methods including analysis of quality, texture, or facial structure.

Why are Facial Liveness Tests important?

One of the common arguments against facial recognition systems is that while a password can be changed, your face cannot. This implies that someone could steal the facial likeness, and use it to break into accounts/systems. Providing a facial liveness system does a good enough job, this argument is massively reduced by preventing spoofing attacks from happening, thus reducing the risk of someone stealing another person's likeness.

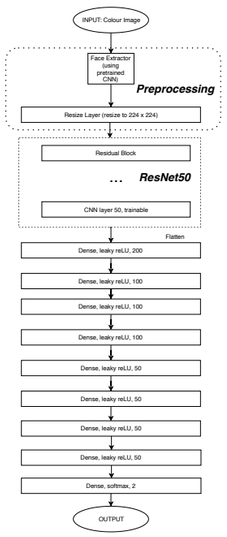
What does facial spoofing look like?

Here is an example. The aim here is to obtain a realness factor from our liveness tests that considers a mixture of the image quality differences between real fake images (specifically in the more detailed high frequencies within an image), analysis of facial stucture differences.

Figure: A real image of a person. Image from the NUAA dataset.

Figure: A spoofed image with a paper based replay attack. Image from the NUAA dataset.

Method 1: CNN based liveness detector



Recently, Convolutional
Neural Networks have played a major part in image classification. Specifically, Residual Networks have performed very well on the ImageNet dataset, classifying types of objects based on a 2D photo. Since facial liveness detection is also an image classification problem, an ImageNet based classifier could be adapted to the role of liveness detection. This model can be used to analyse facial structure and textural information to classify liveness. The architecture of our model can be seen on the left.
Within the preprocessing step, a CNN based face detector
is used to produce the bounding box of the face. This is then cropped into a square image (224x224), which contains just the facial area. This image is what's used to be classified. The ResNet50 model is mostly pretrained, with the last layer being trainable, since image classification features between ImageNet and facial liveness will likely be similar. Then, a standard feed forward network is used to produce the final output of realness (either 0 or 1), using the softmax activation function.

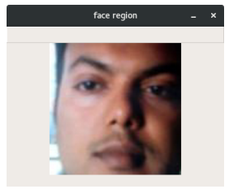
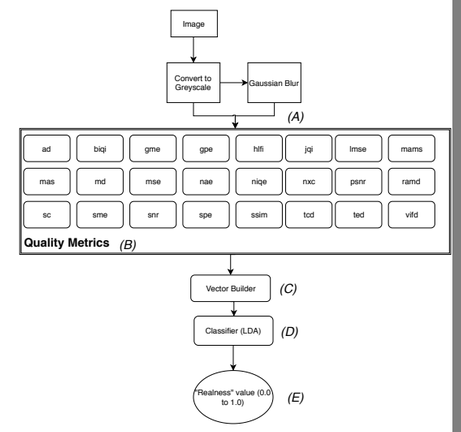


Figure: The result of the preprocessing step, yielding a cropped face.

Accuracy yielded was adequate, with a 71% accuracy
on the Replay Attack test dataset. However, the model had a very low false positive rate, with most errors being caused by false negatives (which is annoying for a user but not a security threat).

Method 2: Whole Image Quality Assessment

This method considers the whole image quality, as a spoofed image is likely to have less details than a non-spoofed image. This liveness test uses 24 different methods of measuring image quality. Some metrics are full reference, which require the use of a Gaussian Blurred version of the image to compare to, which analyse the overall pixel based error, image noise, and frequency ranges. Other quality metrics use pretrained classifiers to analyse quality. The results of these quality metrics are then fed
through a classifier. The classifier used was Linear Discriminant Analysis, with an eigenvector solver. Below, a diagram of how this system works can be seen. Overall, this method led to 87% accuracy
when classifying liveness on the ReplayAttack test dataset. A high number of true negatives and positives were yielded, but when an error occured, the classifier was more likely to class spoofed images as real, compared to the other way around, which isn't ideal for security conscious applications. The current method does have drawbacks, as the training data led to an expected resolution, and with images of different resolutions the prediction is often erroneous. One solution is to include resolution as it's own image quality metric within the classifier. For the demo, the solution was to resize an input image down to (640x480), which solves the problem temporarily, but future improvements should be able to predict liveness without this resize step.
Furthermore, while the time to predict a single image is fairly fast, it could be improved further by migrating to a sklearn based classifier for the BIQI image quality metric. Currently, a libsvm based classifier is used, which is called by accessing the file system. This is very slow.



Real-time Liveness System - a proof of concept

The liveness methods above were brought together into a real-time liveness detection system, utilising an OpenCV based GUI. While simple, this could be furhter improved in the future to allow for liveness as a service platform, allowing developers to calculate liveness on the cloud, rather than relying on built-in features. For the screenshots below, videos from the Replay-Attack dataset were used, but this system also allows for webcam access in real time. This is a proof of concept of a future liveness system that could be put into production.

Figure: A spoofed image being correctly classified as fake by both liveness tests. Frames obtained from the Replay-Attack test dataset.

Figure: A real image being correctly classified as real. Frames obtained from the Replay-Attack test dataset.

One additional finding by testing in real time was that the CNN based liveness test was often correct in it's prediction, but occasionally would switch to a prediction of 'fake' for a single frame. Therefore, the liveness test could be improved by analysing several frames, and determine the modal liveness from that set of frames.
Furthermore, and improvement that can be made to this is to speed the calculation of each liveness test up a bit. Currently, each liveness test is called and executed individually, in series. This could be parallelised, either on one machine or by using distributed computing technologies.