

Avatars you can recognize: testing 3D model render likeness with a face recognition based metric

Alexander Porshnev, Kseniia Sizova, Victor Erukhimov

Itseez3D, Inc.

Introduction

Avatar SDK creates recognizable avatars. They are not hyper-realistic, because the majority of the platforms our customers use won't be able to handle the fidelity of hyper-realism in real-time. However, we are aiming to make an avatar such that if you know the person in real life, you would recognize their avatar. In order to achieve that we test every major release on focus groups to measure recognizability. However, this process is relatively expensive and time consuming, so we can run it only on a limited number of avatar versions. This is why we are considering complementary approaches that would allow us to have a less expensive test on a wider variety of avatars. In this paper we want to discuss the application of a facial recognition system for evaluating recognizability of avatars. Given advances in automated facial recognition, we expect that the machine learning system can be effectively used to produce recognizability results consistent with human perception. Below we describe the facial recognition system that we used, the avatars we tested for recognizability, the experimental setup and the results we got on our test dataset.

Facial recognition

Automated face recognition aims to emulate the human brain's ability to detect and recognize faces and have achieved impressive results in discrimination and robustness. In our experiments we used Face Recognition python library (Geitgey, 2017/2023) as it provides the ability to recognize faces on images using a simple and convenient interface. It is based on the Dlib C++ library, which includes a pre-trained deep neural network for the facial recognition task (King, 2009). Here is how it works.

We have two images with one frontal face in each, and we want to understand if the faces from these images belong to the same person or not. At first, the library finds bounding boxes and landmarks for faces in each photo. If faces are found, necessary transformations (scale, rotation, translation) are applied to them. Then these preprocessed faces are used as inputs for the deep neural network.

This deep neural network is the pre-trained model from Dlib. It is based on the ResNet-34 network from the paper "Deep Residual Learning for Image Recognition" by He et al., but with 29 convolutional layers and reduced number of filters by layer (He et al., 2015). According to the creator's information the network was trained from scratch on a dataset of about 3 million faces, which included 7485 individual identities. The goal of the original training process was to project all identities into 128-dimensional feature vectors that lie sufficiently far from each other for different people.

The mapping of an image containing a face to a 128-dimensional feature vector is conveniently done with the function `face_encodings` from Face Recognition library. This function applies a face detection algorithm from the Dlib library to find a location of the face on the image (it works for several faces in

one image either, but it is not our case). Also it is possible to pass the location to the `face_encodings` function directly if the location is known. Then the function uses the `shape_predictor` algorithm from the Dlib library to identify locations of the important facial landmarks. And finally it uses the `face_recognition_model_v1` object from the Dlib, which is exactly the recently described neural network, to run the `compute_face_descriptor` function. This function at first applies necessary transformations to the input face to center, rotate upright and scale it to 150x150 pixels, and then calculates the 128-dimensional output vector for the face.

Creating vector representation for the images makes it possible to calculate the Euclidean distance between them and evaluate how close these vectors are. Smaller distances mean images are of the same person, and larger distances correspond to different people. If the distance between two vectors is lower than 0.6 (the original threshold suggested by the authors of the Face recognition library) the images are likely to belong to one person. If the value of the Euclidean distance between the predicted vectors is higher than this threshold, it means that the original images contain different people.

The described neural network obtains the accuracy of 99.38% on the [Labeled Faces in the Wild](#) benchmark (Kawulok et al., 2016). As a part of testing each face is slightly augmented 100 times, a feature vector is computed on each of the 100 images, and the results are averaged. If there is only one predicted vector for each face, the model obtains 99.13% accuracy on the LFW.

Measuring recognizability with facial recognition

In order to measure recognizability of avatars we created the avatar renderings dataset (ARD) representing versions of avatar modeling pipelines, each containing screenshots of 36 renders (one for each 36 test photos). We compute the 128-dimensional vectors for each image of a rendered avatar and the corresponding input image. The recognition distance (RD) is computed as a euclidean distance between two feature vectors. The smaller the RD is, the more the corresponding images are like each other.

Test images depict individuals belonging to three distinct ethnicities (Asian, African, and European), three different age groups (young adults, middle-aged individuals, and seniors), and both the male and female sexes, with two images for each combination of these categories. Each test image contains a frontal photo of a person's face (see Figure 1, for example).

Avatar renders were made from test images using Avatar SDK's online demo applications, Head 1.2 and Head 2.0 with <https://webdemo.avatarsdk.com>, and MetaPerson 2.0 with <https://metaperson.avatarsdk.com>, both based on Unity WebGL. In order to obtain renders we upload each test image without any preprocessing and make a screenshot of a computed 3D model (Firefox 109.0.1 and Chrome 109.0.5414.120, Windows 10). Screenshots contain only the avatar's face in the frontal position (Head Shape panel option, see Figure 1). The resolution of the test images was 300 x 300 ppi, and of renders was 120 x 120 ppi, all images were approx. same size approx. 1000 x 1000 pixels.

Figure 1. Steps of Avatars renders dataset preparation.



Avatars used for testing

Head 1.2 is the oldest pipeline in Avatar SDK. It generates a head with a hairstyle and a bust. The head and the hairstyle are represented by a single mesh with topology that varies from one avatar to another. The bust is reconstructed with the clothes that are visible in the input image. Head 2.0 is a newer pipeline that represents a generated hairstyle as a separate mesh, and the topology of the rest of the head is the same across different avatars. MetaPerson 2.0 is the newest pipeline that generates full-body avatars with higher resolution facial and hair textures than Head 1.2 and 2.0.

Experimental results


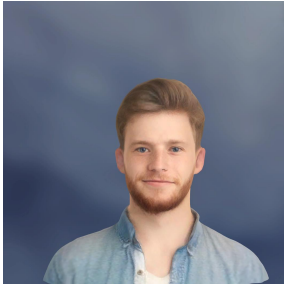
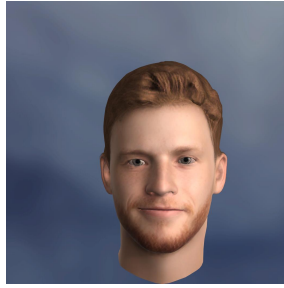





Analysis of distances between 128-dimension vectors for each of the 36 images and each of the three main pipelines with the feature vectors for the corresponding input images (see Table 1 for median, min, and max values for each pipeline) showed that the distribution of distances lies well below the suggested 0.6 threshold (Geitgey, 2017/2023). This means that while humans can distinguish between photos of people and avatar renders given the latter is in sufficiently high resolution, the facial recognition system finds a strong resemblance between an input image and an avatar render.

Table 1. Descriptive statistics for distances between test and avatars images for the main pipelines.

Measures	Head 1.2	Head 2.0	Meta person 2.0
Median	0.325	0.439	0.436
Min	0.263	0.264	0.313
Max	0.546	0.592	0.573




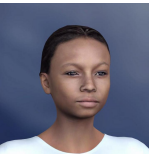
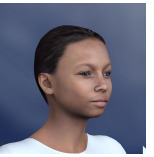
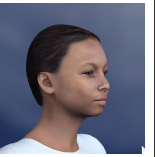
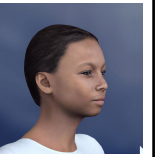
We have tested if the size of the avatar head affects the recognition distances. In order to test this assumption we ran experiments with an adjusted head size for each avatar render. Figure 2 shows example images and the corresponding recognition distances. One can see that the head size has a small influence on the results.

Figure 2. Distance from test image (without and with head size adjustment)

Test Image	Head 1.2	Head 2.0	Meta person 2.0
			
Distance	0.380	0.419	0.399
			
Distance	0.379	0.411	0.394



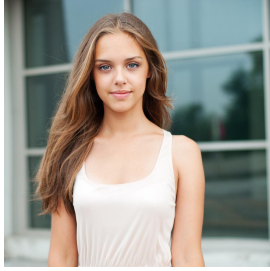

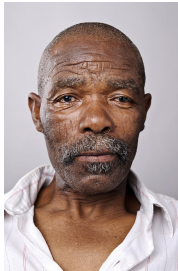



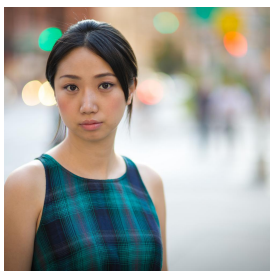
We have also tested the effect of a head position on the recognition distance. Figure 3 shows the example results of a recognition distance for various head rotations. While the score is higher for frontal images, the rotated images score is still below the threshold of 0.6. This means that the avatar will be identified as the person in the input photo by the facial recognition system.

Figure 3. Distance from test image for rotated avatars.

						
Distance	0.383	0.377	0.388	0.423	0.583	0.565

We also wanted to directly test the recognizability of the avatars. In order to do that, for each avatar render we picked the input image with the smallest recognition distance. Figure 4 shows example results for MetaPerson 2.0 avatars. For all 36 MetaPerson 2.0 avatars, the closest image was the test image used to create this avatar (median equal to 0.436). The difference in distances between the test image and the next closest image had a median of 0.194. The distance between the avatar and the second closest image had a median equal to 0.622.

Figure 4. Example of avatar renderings and two closest test images

		
Distance	0.466	0.709
		
Distance	0.313	0.639
		
Distance	0.398	0.525

Discussion

The results show that the facial recognition system provides adequate results for testing the recognizability of avatars. Avatar SDK generates avatars that this system classifies as recognizable by putting them close to the input images in terms of recognition distances.

Table 1 shows that Head 1.2 has a median distance lower than for Head 2.0 and MetaPerson 2.0. It is true that some of the Head 1.2 avatars are more recognizable. It is mostly due to the fact that Head 1.2

avatars are best used statically, without animation and any augmentation. MetaPerson 2.0 avatars, on the other hand, can be easily animated and can be customized. These results are in line with our focus group polls that showed a comparable level of recognizability by humans for images and avatar renderings.

Thus, this machine learning based system can be used to improve the recognizability and monitor the quality of avatars generated from images.

Reference

- Geitgey, A. (2023). *Face Recognition* [Python]. https://github.com/ageitgey/face_recognition (Original work published 2017)
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. <http://arxiv.org/abs/1512.03385>
- Kawulok, M., Celebi, M. E., & Smolka, B. (Eds.). (2016). *Advances in Face Detection and Facial Image Analysis*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-25958-1>
- King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 10, 1755–1758.