JOHANNES KEPLER
UNIVERSITY LINZ

Philipp Hofer
Institute of
Networks and Security

@ philipp.hofer@ins.jku.at
https://www.digidow.eu/

July 2021

Face recognition: Increase accuracy by filtering images with heuristics

Technical Report

Christian Doppler Laboratory for
Private Digital Authentication in the Physical World

This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World, funded by the Christian Doppler Forschungsgesellschaft, 3 Banken IT GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Contents

1. Motivation

This document tries to find simple heuristics of images of faces to differentiate between successful and unsuccessful face recognition. Intuitively, the camera-face angle might play an important role: In full-frontal images a lot of information is contained, in contrast to full-profile images where at least half of the face is hidden. Therefore, as a proxy for this angle we will focus on these metrics:

1. Distance between the eyes, relative to the face-width
2. Distance between the center of the eye to the mouth, relative to the face-height
3. Face size

```
2021-09-24 09:17:28.528260: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object file: No such file or directory
2021-09-24 09:17:28.528318: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-09-24 09:17:32.234276: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or directory
2021-09-24 09:17:32.234359: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2021-09-24 09:17:32.234402: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (ph-work): /proc/driver/nvidia/version does not exist
2021-09-24 09:17:32.278150: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

For evaluation, we created a dataset by mounting a camera in our printer room and collecting images whenever a person is detected in front of the camera.

- There are 13 different people in the dataset.
- There are between 5 and 210 images of every person, with an average of 79.3 images/person.

Every one of the sum(1 for x in person_folders()) people has a template. First, we calculate the embedding of this template image for every person:

```
def get_template_embs(path):
    templates = dict()
    for person in os.listdir(template_path):
        person_path = os.path.join(template_path, person)
        person_name = os.path.splitext(person)[0]
        templates[person_name] = rec.get_template_embedding(person_path)
    return templates
templates = get_template_embs(template_path)
```

```
threshold = 1.2
```

Next, we go through the dataset, perform facerecognition and split the result in correct and wrong identifications with respect to a pre-defined threshold of threshold .

```
correct = []
wrong = []
for person_folder in person_folders():
    template = templates[os.path.basename(person_folder)]
    for img in os.listdir(person_folder):
        img_path = os.path.join(person_folder, img)
        emb = rec.get_emb(img_path)[0]
        if rec.get_score(template, emb) < threshold:
            correct.append(img_path)
        else:
            wrong.append(img_path)
```

```
    13it [01:59,  9.22s/it]
```

There are 261 correct and 770 wrong identifications (accuracy of 25.3%).

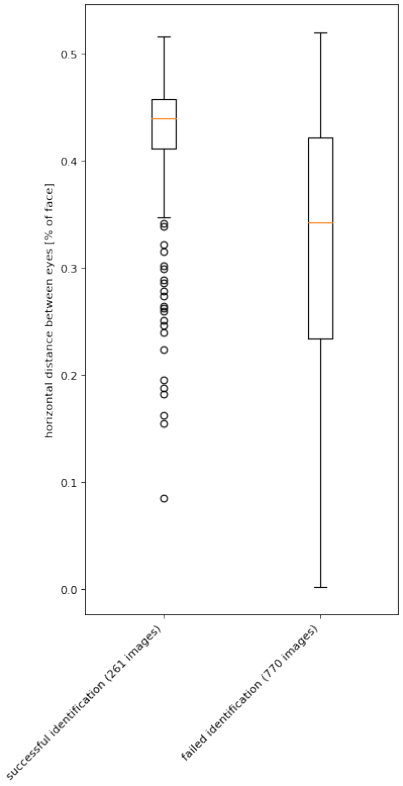
2. Eye distance

Next, for each image, we calculate the distance between the eyes, and scale by face width. We save the distances where face recognition failed in *failures_eye_distance...*

```
failures_eye_distance = []
for img_path in wrong:
    failures_eye_distance.append(rec.get_eye_distance(img_path))
```

...and similarly, we save distances where face recognition succeeded in *success_eye_distance*:

```
success_eye_distance = []
for img_path in correct:
    success_eye_distance.append(rec.get_eye_distance(img_path))
```




```
amount_images = len(np.where(np.array(failures_eye_distance) < min(
    success_eye_distance))[0])
```

We can clearly see, that there seems to be a certain threshold. If the distance is short of this threshold, face recognition does not work anymore. The smallest distance where face recognition succeeded is 8.5% of the face width. There are 51(/1031) images with a smaller eye distance, where face recognition failed.

2.1 Learning

Discarding all faces where the eye distance is smaller than 8% of the total face width reduces the amount of false-positives without having any negative effect. In order to generalize well to new datasets, you might want to reduce this threshold a little.

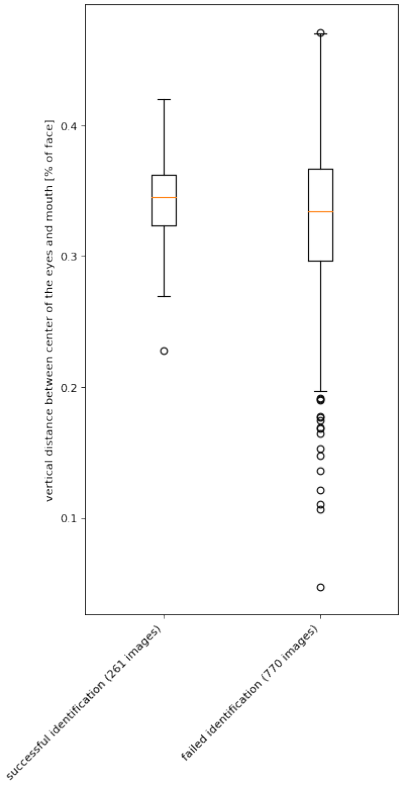
3. Eye-Mouth distance

Similar to the *eye distance*, we perform a similar analysis with respect to the distance between the center of the eyes and the mouth. We calculate it's distance in *[failures|sucess]_eye_mouth_distance*:

```
failures_eye_mouth_distance = []
for img_path in wrong:
    failures_eye_mouth_distance.append(rec.get_eyes_mouth_distance(img_path))

success_eye_mouth_distance = []
for img_path in correct:
```

```
    success_eye_mouth_distance.append(rec.get_eyes_mouth_distance(img_path))
```

Next, we plot the resulting data:




```
amount_images = len(np.where(np.array(failures_eye_mouth_distance) < min(
    success_eye_mouth_distance))[0])
```

Again, we can clearly see that there is a certain threshold (0.2% of the face height), where face recognition fails if the eye-mouth distance is short of this threshold. There are 51(/1031) images below this threshold.

3.1 Learning

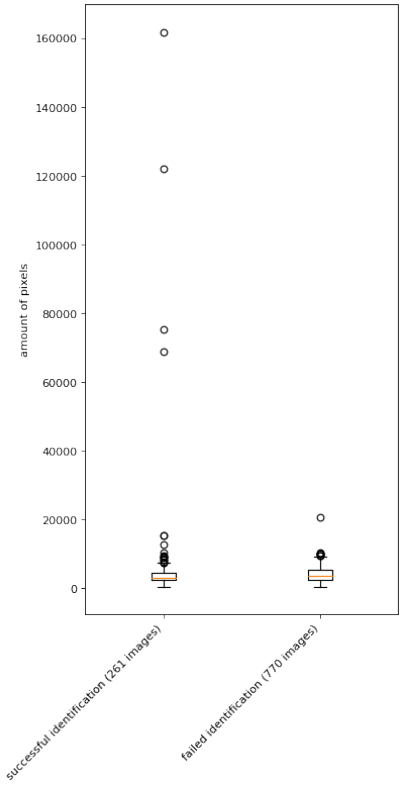
Discarding all faces where the eye-mouth distance is smaller than 22% of the total face height reduces the amount of false-positives without having any negative effect. In order to generalize well to new datasets, you might want to reduce this threshold a little.

4. Face area

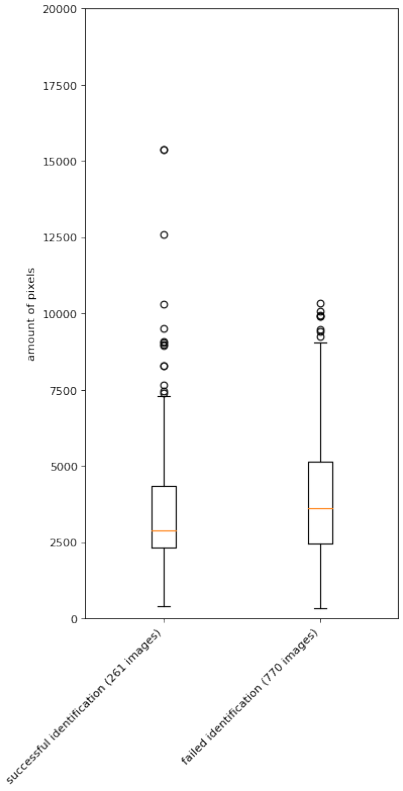
Intuitively, a larger face should contain more information than a smaller one, and thus achieve superior accuracy. To test this hypothesis, we calculated the amount of pixels for every face, grouped by whether face recognition worked.

```
def get_face_area(img):
    faces, img_width, img_height = rec.extractor.get_retinaface_output(img)
```

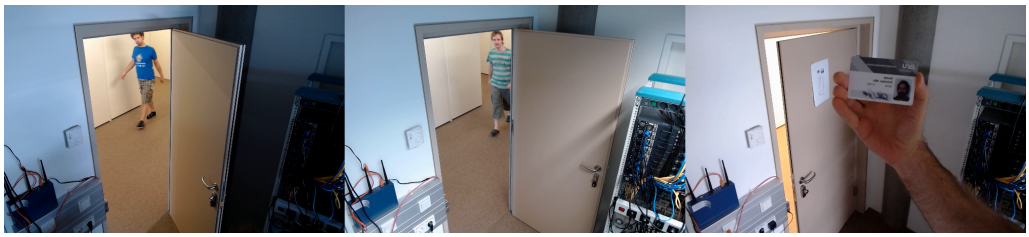
```
    face = faces[0] # we expect only a single person in the image
    face_width = abs(face[0]-face[2])*img_height
    face_height = abs(face[1]-face[3])*img_width
    face_area = face_width*face_height
    return int(face_area)
failures_facearea = []
for img_path in wrong:
    failures_facearea.append(get_face_area(img_path))

success_facearea = []
for img_path in correct:
    success_facearea.append(get_face_area(img_path))
```



There are a few outliers, which makes it hard to draw conclusions. If we remove this outliers, we get the following plot:



The image with the lowest amount of pixels where face recognition succeeded contained 405 pixels; the image with the lowest amount of pixels where face recognition failed contained 341 pixels. The 3 images with the lowest amount of pixels where face recognition succeeded contain [405, 407, 465] pixels:



4.1 Learning

In our dataset, there does not seem to be a clear threshold with respect to the amount of pixels. One possible explanation is that all images in our dataset are large enough for face recognition to work well. Further studies could study a similar setting, where images are (artificially) smaller than the images in this setting.