Contactless palm print recognition: Novel design and palm openness classification

Leo Li & Simon Mahdavi

Master's thesis in Mathematics

Supervisors
Johan Windmark
Axel Kärrholm
Filip Winzell
Anders Heyden

LUND UNIVERSITY

LTH

FACULTY OF ENGINEERING

Department of Mathematics

Contents

Abstract

Biometric technologies, such as facial and fingerprint recognition, have become widely adopted for identity verification in various applications. Palm biometrics, utilizing the unique patterns of the human palm, has gained significant attention for its accuracy and security. This thesis aims to investigate and propose a comprehensive design for certain aspects of a contactless palm print recognition system, taking into account the issue of usability and palm openness. Contactless palm print recognition offers several advantages over other commonly adopted biometric systems. Firstly, it can operate effectively with low-resolution images and inexpensive cameras, making it more cost-efficient. However, the most significant advantage, particularly in the present circumstances, is its hygienic nature. But the technology also comes with new challenges: distance to the camera and the changes in palm print due to palm openness to name a few. Through the employment of transfer learning and landmark-based methods, the classification of palm openness from input images achieved an average accuracy of 0.90. In conclusion, despite their perceived visual quality, openness creates slight variations in width, depth, and crease distances on the palm print, which in turn affects the matching between images. Our results indicate that palm images are best matched to ones of the same openness. These findings validate the importance of palm openness and its impact on system performance, as well as the ability of the classifier to correctly classify the openness. By addressing these challenges, we can improve the accuracy and applicability of contactless palm print recognition technology.

Acknowledgements

This thesis project has been conducted at Precise Biometrics in Lund. We would like to thank our supervisors at LTH, Anders Heyden and Filip Winzell, for always providing good input and making sure we stayed on the right track. Another big thank you goes to our supervisors at Precise Biometrics, Johan Windmark and Axel Kärrholm, for answering both well-formulated and less well-formulated questions throughout the project. Finally, we would like to express our gratitude to all Precise Biometrics employees for making us feel a part of the team and frequently allowing us to capture photos of their hands.

Notations and Abbreviations

ROI - Region of Interest
FAR - False Acceptance Rate
FRR - False Rejection Rate
EL - Enrolled images taken with additional light source
ENL - Enrolled images taken with no additional light source
VL - Verification images taken with additional light source
VNL - Verification images taken with no additional light source
DPI - Dots per inch
ML - Machine learning
XGBoost - Extreme Gradient Boosting
KNN - K-Nearest Neighbour
NN - Neural Network
CNN - Convolutional Neural Network
RF - Random Forest

Verify - Saved ROI templates in the system to be matched against
Enroll - Input image asking for access to the system
Spoof - Fabricated biometric information
Handedness - Right or left hand
Genuines - Images that should be matched
Imposters - Images that should not be matched

1 Introduction

Biometric technologies, such as facial and fingerprint recognition have become commonplace in our daily lives, serving as reliable means of verifying our identity when using our devices or accessing various services. Palm biometrics uses the patterns of the human palm for identification and authentication of individuals. The field has gained attention in recent years for being highly accurate and secure. This technology utilizes unique biometric features of the palm, and the information can be extracted through different techniques such as contactless palm print, palm vein, and contact-based palm print. Contactless palm print recognition, although less common than palm vein and contact-based palm recognition, has gained traction among researchers and industry experts in recent years.

Contactless palm print recognition involves capturing an image of the palm surface and analyzing the pattern of lines and ridges to create a biometric template. The pattern is formed during fetal development and remains the same throughout a person's life, providing a highly reliable means of identification [1].

The use of biometrics has proven to be particularly useful in scenarios where traditional identification methods, such as passwords or PINs, may not provide adequate security [2]. Compared to more commonly used biometric solutions like face and fingerprint recognition, contactless palm print recognition offers several advantages. It serves as a less intrusive and invasive alternative to face recognition as it can't be used for identification by public surveillance cameras. Additionally, many consider facial information to be more delicate information than hand-related information. There are also security concerns, as faces often are publicly available on social media, and can be used for spoofs. Palm prints, with their larger surface area as well as distinctive lines in comparison to fingerprints, offer a viable option for contactless recognition with low-resolution cameras, which has risen in demand in recent years,

due to infectious diseases giving cause to hygienic solutions [3].

A major challenge for contactless palm print recognition is to reliably extract the region of interest (ROI), particularly in a mobile setting where background, lighting, and hand posture can vary greatly. In this context, the ROI refers to an area of the hand from which the information to identify an individual is extracted. The quality and choice of ROI is critical to the accuracy of identification, and much research has been conducted on developing methods for extracting the ROI [4]. Despite these efforts, challenges still exist in achieving accurate and reliable ROI extractions.

Contactless palm print shows many benefits in comparison to other more established biometrics. But the technology also encounters a new set of challenges.

The purpose of this thesis aims to investigate and propose a good design for some parts of a biometric palm print system, employing a holistic approach. Among the various challenges, we have chosen to focus specifically on addressing the issue of palm openness.

It is crucial to clarify that our thesis is not solely focused on design aspects, but rather incorporates machine learning and image analysis. While the primary objective lies in developing these technical components, we recognize the significance of creating practical tools that extend beyond theoretical concepts. Thus, it becomes essential to consider the ultimate application and end-use of the technology we are developing. By aligning our research with practical limitations and intended use, we aim to bridge the gap between theoretical advancements and real-world usability.

From the outset, our intention was to develop a minimal yet functional system that would allow us to explore the primary challenges involved. Consequently, this report is structured to reflect our work process, encompassing various design implementations and providing an overview perspective before delving into the specific problem of openness classification.

2 Background

2.1 User design

Before delving into the different aspects of the system, it is essential to establish a clear understanding of what constitutes a "good" design within the context of a biometric palm print system. Within the scope of this thesis, the definition of a "good" system, as established by the authors, includes two fundamental aspects: user experience and accuracy, both explained below.

User experience

- *Affordance* - From the interface the user understands how the system should be used and interacted with.
- *Feedback* - Information to the user is necessary to ensure that everything is functioning correctly, make adjustments if needed, or provide reassurance that their action has triggered a response. Instructions may be given to guide the user in modifying their action if necessary.
- *Mapping* - The relationship between control and effect. The effect of action needs to be in relation, in terms of magnitude and similarity, to the effect it generates.
- The system needs to be designed with a visually pleasing aesthetic, good responsiveness, and fast processing.

Accuracy

- The information the system requires needs to be extracted in a consistent and reliable way.
- The information needs to be of high quality or otherwise rejected.

- The correct person needs to be identified and authorized, and if not, rejected.
- *Liveness & spoof* - Synthetic, altered, or fake information needs to be rejected.

2.2 Palm print characteristics

The surface of the palm contains three principal lines, also called flexion creases. Then there are secondary creases, more commonly called wrinkles, and lastly the ridges. The flexion and major secondary creases are formed between the third and fifth months of pregnancy and the rest are created after we are born. The three principal lines are genetically dependent, for instance, principal lines look different for individuals of some genetic diseases such as Down syndrome, for whom the distal transverse and proximal transverse are replaced by a singular transverse crease. [2].

However, besides the principal lines, the remaining creases are not genetically dependent, and even identical twins differ in these patterns [5].

These principal lines together with secondary creases are what allow contactless palm print recognition using low-resolution cameras, as ridges are too small to be captured reliably contactless, see Figure 2.1 for the creases on the palm.

2.3 Machine Learning

Machine learning is a field of study in computer science that deals with the development of algorithms and models that can learn from data to make predictions or decisions. The term "learning" refers to the process by which a machine learning algorithm adjusts its parameters to minimize a given loss function, based on the data it has been trained on [6]. Furthermore, it can be said that the final objective of machine learning is to construct a model that can accurately predict outcomes for new data points drawn from an unknown probability distribution [7].

Machine learning models can be classified into two categories based on

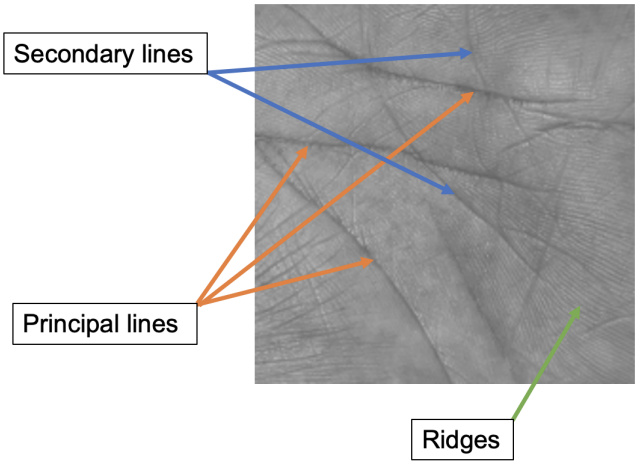


Figure 2.1: Principal lines, secondary lines, and ridges of the palm print

the structure of the input data and the desired output: supervised learning, and unsupervised learning [8]. Supervised learning uses data that is labeled, where the inputs are paired with their corresponding desired outputs. The goal of the model is to create an algorithm that understands how the mapping between the inputs and the outputs is conducted. Unsupervised learning, on the other hand, includes learning patterns in the data without the use of labeled examples.

Modern machine learning techniques, such as deep neural networks, extreme gradient boosting and transfer learning, have pushed the boundaries of what is possible with machine learning algorithms. These techniques have enabled the development of models that can perform complex tasks such as image recognition, natural language processing, and playing games at a superhuman level [9]. However, despite these advances, machine learning algorithms are still far from possessing human-like intelligence, and the development of such algorithms remains an active area of research [10].

2.3.1 Machine learning classifiers

Support vector machines

A support vector machine (SVM) is a robust and non-probabilistic classifier that separates classes by using a hyperplane in the feature space. The main goal of SVM is to find the hyperplane that maximizes the distance, known as the margin, between the hyperplane and the data points of different classes. While this concept is easily visualized in two dimensions, it becomes more challenging as the number of classes increases [11].

k-nearest neighbor

The k-nearest neighbors (k-NN) algorithm is a simple yet effective classification method. It works by measuring the distances between a given point and all labeled data points in the feature space. The algorithm then identifies the k nearest neighbors to that point and assigns it the label that is most frequent amongst those neighbors. The process of training involves mapping the data to the feature space, while the classification step involves counting the frequency of labels among the k-nearest neighbors [12].

Let d be some distance metric (e.g., Euclidian) and k a defined positive integer. The algorithm of classifying a data point, x, with k-NN, can then be summarized as follows:

1. $D \leftarrow d(x, x_i)$ for $i = 1, ..., n$
2. Sort D in increasing order.
3. $D \leftarrow D(1, ..., k)$
4. Let K_i define the number of data-points belonging to class i among D.
5. Assign x to class $max(K_i)$

Random forest

Random forest is an ensemble machine learning algorithm that combines predictions from multiple decision trees to make accurate predictions. The algorithm constructs a collection of decision trees by using bagging, a technique that creates multiple subsets of the original training dataset [13]. Each subset is used to train a separate decision tree. The trees are constructed independently, allowing them to capture different aspects of the data [14].

To make a prediction using random forest, the algorithm clusters the predictions of all the decision trees. For classification tasks, it takes a majority decision among the trees, while for regression tasks, it computes the average prediction. This ensemble approach helps to reduce the variance and increase the overall accuracy of the model [15].

Extreme gradient boosting

Extreme gradient boosting (XGBoost) is a machine learning algorithm that has gained popularity due to its impressive performance in various competitions and real-world applications. The algorithm is an extension of the traditional gradient boosting method and incorporates several enhancements to improve efficiency and accuracy [16].

The main idea behind XGBoost is to create a strong ensemble model by iteratively adding weak learners, typically decision trees, to the ensemble. Each weak learner is trained to correct the mistakes made by the previous ones. The final prediction is obtained by combining the predictions of all the weak learners.

XGBoost also includes additional features to improve its performance. It employs a technique called tree pruning to remove unnecessary branches during the construction process, reducing model complexity and preventing overfitting. Moreover, it utilizes parallel processing and distributed computing frameworks to accelerate training and prediction times for large-scale datasets [17].

2.3.2 Transfer learning

Training machine learning models often requires a substantial amount of data for effective training. Transfer learning offers a solution by leveraging knowledge gained from a previous classification task and applying it to a new task.

One commonly used form of transfer learning is inductive transfer learning. In this approach, a model is initially trained on a large labeled dataset, such as ImageNet [18], and subsequently fine-tuned to classify new data. The early layers of a network captures general information about low-level structures like edges and corners, which often remain relevant to the new classification task. On the other hand, the dense or fully connected layers at the network's end contain specific classification information, making them better suited for replacement when training on new datasets with different classes [19].

2.3.3 Neural networks

Neural networks, first proposed by Donald Hebb in 1949, are a type of machine learning model that use interconnected nodes to transmit signals [20]. In a typical neural network, artificial neurons are arranged in layers with connected nodes that apply a linear transformation to the input through a connective edge called a synapse.

Nodes in a neural network apply an activation function to calculate their output. By sequentially transforming the input through multiple layers, the network produces an output, which could be a classification decision or another desired outcome. Training the neural network involves adjusting the weights and biases of the network to ensure that a given input yields the desired output when processed through the network. This parameter-tuning process is achieved using a technique called back-propagation. Back-propagation calculates gradients to optimize weights, turning them in the opposite direction of the gradient [21]. The optimizer algorithm determines the update rule and learning rate for weight adjustments, aiming to minimize the loss and improve network performance [13].

The architecture of neural networks varies greatly depending on the specific purpose of the model. For image classification tasks, commonly used and effective architectures are convolutional neural networks or residual neural networks.

Convolutional neural networks

One popular type of neural network is the convolutional neural network (CNN), commonly used for image analysis tasks. CNNs utilize convolutional layers to apply learnable filters to input images, identifying different features and creating feature maps. These feature maps are then passed through activation functions to produce nonlinear transformations of the data. The convolutional layer performs convolutions by sliding filters over the image, extracting features at different locations. This allows CNNs to automatically learn and extract meaningful features from images, making them effective in tasks like image classification and object detection. [9].

CNNs also use pooling layers, which downsample the feature maps to reduce the number of parameters in the network and increase its robustness to variations in the input. The final layers of a CNN are fully connected layers, which perform classification based on the features identified in the earlier layers [22].

Residual neural networks

Residual Networks, also known as ResNets, are a type of deep neural network architecture that were introduced by He et al. in 2015 [23]. The objective was to address the issue of degradation or vanishing gradients in very deep neural networks, where the performance of the network degrades as the depth increases.

The key idea behind ResNets is the use of residual blocks, which introduce shortcut connections or skip connections, that allow the network to bypass one or more layers. This is achieved by adding the input of a layer to the output of a subsequent layer, effectively creating the residual mapping. By doing so, the network can learn residual functions, which capture the difference between the input and the desired output.

ResNets have demonstrated superior performance in various computer vision tasks, such as image classification, object detection, and image segmentation, surpassing the performance of previous network architectures [24].

2.3.4 Evaluation metrics

In this section, we will summarize the commonly used metrics for evaluating palm print recognition algorithms and machine learning algorithms in general. Once a model is trained, it is typically assessed using a test dataset. To facilitate the discussion, we introduce the following terms:

- P: The number of positive samples or examples corresponding to a match.
- N: The number of negative examples.
- TP: The number of true positives predicted by the model.
- TN: The number of true negatives predicted by the model.
- FP: The number of false positives.
- FN: The number of false negatives.

With these terms, four metrics can be described using the following equations:

$$Recall = \frac{TP}{P}$$

$$Accuracy = \frac{TP+TN}{P+N}$$

$$F1\ Score = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

$$False\ Acceptance\ Rate\ (FAR) = \frac{FP}{N}$$

$$False\ Rejection\ Rate\ (FRR) = \frac{FN}{P}$$

2.3.5 Precise Biometrics matcher

The core part of a biometric palm print system is needless to say the matching algorithm. However, as this has not been the focal point of this project and is also the intellectual property of *Precise Biometrics* this report will not explain in detail on how it works, or has been further developed. What can be said is that the algorithm is based on the same technology as *Precise Biometric's* fingerprint matcher, and is a pattern-based algorithm that analyzes image texture. The algorithm itself is general and can give a matching score to any two images, not only palm- or fingerprints. As mentioned, the basis of the algorithm is the same as their fingerprint technology but the algorithm has then been retrained on the *Precise Biometrics dataset* (see Chapter 3.2.3) to be specifically adapted to palm prints.

When two images are sent to the matching algorithm it gives a score on how "similar" the images are, the score itself is based on a multitude of proprietary algorithms. However, the matching score is closely related to the False Acceptance Rate, FAR, as well as the False Rejection Rate, FRR, and for the purpose of this report, the matching score can be considered a representation of FAR.

In biometrics, the concept FAR is the ratio of *imposters* that are incorrectly accepted. *Imposters* are invalid inputs or in our case, palm prints that should not be accepted. A FAR value of 1/50K means that theoretically one out of 50,000 palm prints from the dataset would be falsely accepted. FRR is instead the number of *genuines* that are falsely rejected, genuines are palm prints that should be accepted. For a biometric system in commercial use, FAR is often prioritized as a false accept would incur a security risk, whilst a false reject could be solved by another attempt. A common minimum requirement is a FAR value of 1/50K.

As the matching score itself is an arbitrary value, its sentiment is decided by what FAR values it corresponds to. For instance, if the matching algorithm would be used to determine whether a pattern is a right or left hand, a score of 2,500 would be enough to correspond to a FAR of $1E^{-}5$. Whilst in our case of matching different palm prints a score of 7,223 is necessary to correspond to a FAR of $1E^{-}5$. For the matching algorithm

Score at 1/FAR								
1/FAR	**100k**	**50k**	**10k**	**5k**	**1k**	**500**	**100**	**50**
Score	7,223	5,357	4,685	4,391	3,802	3,552	2,980	2,731

Table 2.1: FAR to score values for the palm matcher

that we've utilized we get the following score to FAR-value relationship, see Table 2.1.

A higher matching score is always preferable. What score to choose as a threshold depends on what FAR it corresponds to, and in turn what FAR to use is a result of the security requirement of the task. Values larger than the threshold will be accepted as matches, while smaller numbers will be rejected.

Note that the FAR is logarithmic in relation to the linearly increasing matching scores. Meaning that low scores represent very poor FAR and that the score of 2,000 should *not* be considered half as "good" as a score of 4000.

2.4 MediaPipe

MediaPipe is an open-source project by Google, currently at the alpha stage. MediaPipe consists of the Framework and the Solutions which are built upon the Framework. The multiple solutions solve different types of computer vision tasks as well as tools to customize their machine learning solutions and retraining for specialized tasks. The **Solutions**; Hand Landmark Detection, Face Detection, Image Segmentation, Pose, Object Detection to name a few, are "examples" or complete **Solutions** that solve a specific machine vision task and that are based on pre-trained TensorFlow models [25].

The **Framework** consists of multiple components, with the main ones being the Packets, Graphs and Calculators. A **Graph** is the complete processing pipeline where data in the form of **Packets**, C++ type payload, are transported throughout. In the Graph there are Nodes or **Calculators** that produce/consume packets. The calculators can be for example the TensorFlow model, Image to Tensor transformers or Ren-

derer component. These Calculators or Nodes are written in C++, and handle specific computational tasks [26].

The Framework is written in C++. However, the solutions are also available in Python, JS, Coral, iOS, and Android. The MediaPipe Framework handles still images, decoded video frames as well as live video feeds, and can be used with embedded devices.

MediaPipe has been an internal tool at Google since 2012, since then it has been integrated into many of the services Google provides, including YouTube, Google Photos, Gmail, and Augmented Reality Ads to name a few. One of the core strengths of MediaPipe is how efficiently it is written, as it can be run even on IoT devices. As of 2019, MediaPipe was publicly released and available to all, as well as being continuously updated and having new solutions and features added to this day [25].

3 Methods

3.1 System and devices

The setup presented in Figure 3.1 shows how the data set described in the following section was collected, as well as the intended setup for verification images. Users are instructed to position their hand in front of the iPad's camera, which captures an image for identification. To optimize image quality and enhance the palm's biometric characteristics, a Ledgo LG-268c light source was employed. The image acquisition process utilized the front-facing camera of a 9th-generation iPad running iPadOS version 16.3. Linux-based computers served as the processing units in our setup, facilitating the connection of the iPad camera via the Iriun Webcam application.

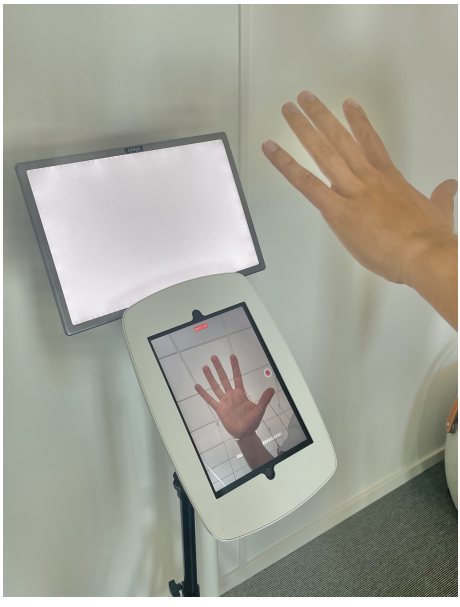


Figure 3.1: The intended setup for the contactless palm print recognition system.

3.1.1 Enroll & Verify

In biometric systems, there are two types of images that the system interacts with. There are the *enrolled* templates that are stored by the system. These enrolled templates often consist of multiple ROI templates and connected IDs, and are considered by the system as "ground truth" and grant access to the system. Then there are the *verification* images. These images are sent to the system and are requesting access. Verification images are pre-processed to be matched against all the enrolled templates in the database, and if the matching score is above the set threshold the system grants access. If a verification image is granted access, they are often stored to update the enrolled template in the case of minor changes in the biometric measurement.

In the scope of this thesis, remote enrollment is intended in the end system. In this context, remote enrollment means that a user can send an image or video of the palm biometric from a mobile phone to be saved as templates in the access system.

3.2 Datasets and acquisition

To be able to train, evaluate and develop algorithms we needed data to work with. From an earlier data collection, we had the Precise Biometrics dataset to work with, collected on behalf of Precise Biometrics in 2020. Besides the Precise Biometrics dataset, we also collected two smaller sets, one being the Openness collection, to develop and train our openness classifiers, using an iPad front-facing camera. The other was the *Smartphone enroll images*. These images were captured using various models of smartphones to develop and evaluate different algorithms that we have developed, as well as to assess the performance of the matcher and openness classifiers.

3.2.1 Openness collection

The *"Openness collection"* was collected by the authors of this paper and contains palm videos of 19 employees at Precise Biometrics as well as 33 students at the Faculty of Engineering, Lund University. Of the 52 people

in the collection 10 (19%) were women and the remaining 42 (81%) were men. The videos were converted to images skipping every other frame, totaling 9213 images and taking up 3,2GB. Worth noting that the videos are not of the same length, as it was dependent on the speed at which the subjects closed their palm. Meaning that the dataset is unbalanced, where some individuals are represented with more images than others. This report will only showcase example images of the authors and not that of the other subjects in the data collection.

The purpose of the Openness collection was to develop methods of evaluating the level of openness of the palm, each level of openness was divided into different classes to be classified by different models. This is described more in detail in later sections.

Verification

The videos were collected using the front-facing camera of a 9th-generation iPad, with iPadOS version 16.3. The iPad was mounted and a Ledgo LG-268c was used as a light source to minimize differences between images and control affecting parameters. See Figure 3.1 for the setup. In each video the subject was asked to perform a controlled motion, going from a maximally stretched hand to a closed fist, see Figure 3.5.

Data split

When training classifiers standard practice is to split the data into training and testing sets. However, as we converted videos of subjects closing their palms into multiple still frames, many images belonging to one subject are very similar. This means that if we were to split the whole dataset randomly the algorithms would train on data that is almost identical to the data it later is tested on. To avoid this we split the data based on subjects, meaning that we excluded all images from certain subjects when splitting the data. In total two train-test splits were made.

However, some considerations were lacking in the forming of the split. When we split the data we did it by subject ID, with 10 test subjects and 42 training subjects, and mistakenly failed to consider that the subjects had uneven number of images as well as being unbalanced in terms of the

different classes, which resulted in the two splits having different sized train and test sets, see Table 3.1. To compensate for this we weighed the evaluation metrics in relation to the size of the validation sets. The reasoning for the two splits was to mimic a small-scale k-fold operation, with k=2.

Test data split							
Split	IDs	Class 1	Class 2	Class 3	Class 4	No. images	Ratio
1	43-52	627	131	107	309	1174	0.376
2	31-40	823	313	299	509	1945	0.624

Table 3.1: Test data split, number of samples in each class and split

3.2.2 Smartphone enroll images

As the intended system uses remote enrollment with smartphones, it was also necessary to collect images and videos shot with different smartphones. The following smartphones were included in the study:

- iPhone 12 mini
- iPhone 7

3.2.3 Precise Biometrics dataset

One of the databases we utilized was a database collected by a data collection center on behalf of *Precise Biometrics* in August 2020. The database consists of 223 subjects and 2533 images of left and right hands, totaling 1,7GB of data. However, in the collection a number of images lacked hands, the posture was too poor to extract a ROI from, and some images were too blurry to use. From the original 2533 images, 429 were removed due to lacking hands or poor posture. Blurry images were kept. No images from the collection will be shown in this report.

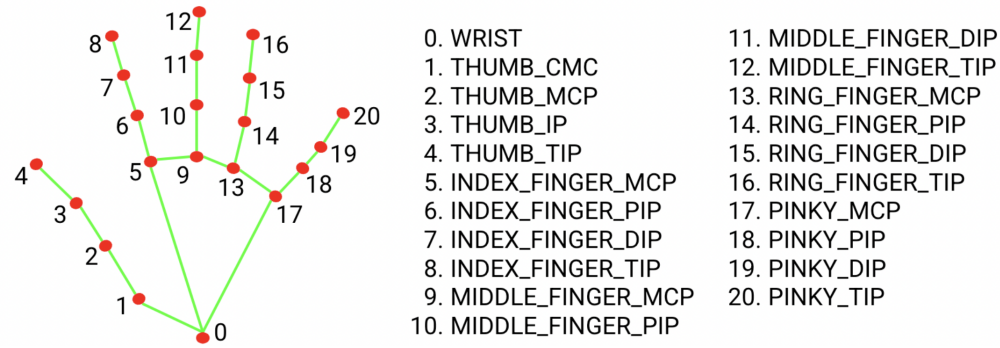


Figure 3.2: MediaPipe annotation of 21 hand landmarks, (figure from MediaPipe documentations [27]

3.3 Hand landmarks

The Solution that we've based large parts of our work on is the MediaPipe "Hand Landmarker task". From an input image, the **Task** gives a confidence score on whether there is a hand in the image. On the premise that there is a hand within the frame, the "hand landmark model bundle", a TensorFlow model, locates 21 hand-knuckle landmarks, and returns xyz-coordinates for them.

The TensorFlow model was trained on 30,000 real-world images as well as synthetically rendered hand models imposed over different kinds of backgrounds.

Each of the 21 hand landmarks consists of x,y, and z coordinates. The x and y coordinates are normalized to the width and length of the frame respectively (in pixels) whilst the z coordinate is estimated using the wrist or "0" landmark as the origin. The z coordinate represents the landmark depth, with the depth at the wrist ("0" landmark) being the origin. The smaller the value, the closer the landmark is to the camera. The magnitude of z uses approximately the same scale as x. The MediaPipe annotation of the hand landmarks can be seen in Figure 3.2, observe that the hand shown in the Figure is a left hand and that the annotation is the same for a right hand, with "1" always being THUMB_CMC.

MediaPipe parameters

Parameters	Value
running_mode	Image
static_image_mode	True
model_complexity	0
max_num_hands	1
min_detection_confidence	0.5
min_tracking_confidence	N/A

Table 3.2: Overview of MediaPipe settings employed in our thesis

MediaPipe settings

MediaPipe offers several adjustable parameters. The primary parameter is the **running_mode** setting, which determines whether the input data is treated as a single-shot frame or as a continuous video or live-stream feed. Depending on the chosen mode, MediaPipe either extracts landmarks from a "clean" slate or utilizes historical landmark positions from previous frames to detect new landmarks. While using the system we observed significant differences in landmark coordinates based on the running_mode set, static image setting and video returned different coordinates for the landmarks. Apart from the running_mode, there are other parameters that can be adjusted. For a complete list, please see the MediaPipe Hands documentation [27]. Table 3.2 displays the settings used in our application.

3.4 Region of Interest segmentation

Extracting a Region of Interest, or ROI, is one of the first hurdles when developing a biometric palm print matcher. The ROI of a palm print is often defined as a square at the center of the palm. Most approaches are similar in the way that they start with some type of hand segmentation, for example color thresholding. Then the methods differ, some documented methods are; using a CNN to segment, alternatively using a mass center to find valley points, or maximizing a circle within the hand area and extracting a square from that [28].

We have decided to approach this problem using MediaPipe Hand Land-

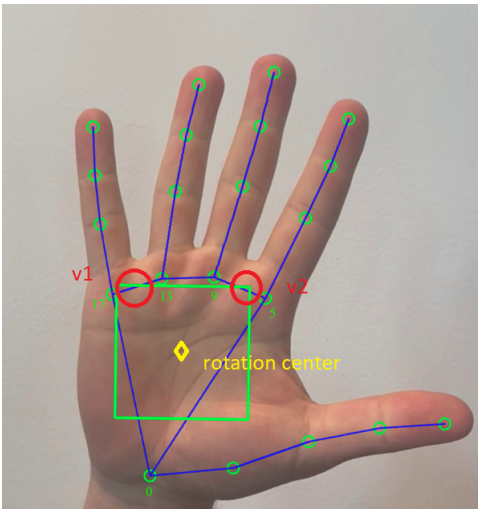


Figure 3.3: Illustration of the segmentation logic used to extract the ROI. V1 and v2 indicates the valley points and landmarks 0, 5, 9, 13, and 17 are also annotated.

marks and solve this using the 21 hand landmarks and a logical approach to finding valley points and center.

3.4.1 Segmentation logic

The general idea of the approach is to use the valley between (see Figure 3.2 for numeric annotation) the index (5) and middle (9) finger as well as the valley between the ring (13) and little (17) finger to create a plane, see Figure 3.3. This plane will be the upper border of the square that will become the ROI. From the plane, a rotation angle is computed and later used to rotate the whole image accordingly, see Algorithm 1 for the rotation algorithm. The rotation center (also the center of the ROI) is determined by the mean of the wrist (0), index (5), and little (17) finger landmarks. The size of the ROI is determined by the distance between the index (5) and little (17) finger, multiplied by a size factor which is 1.3 and centered around the rotation center.

From this approach, we get a consistent ROI that is invariant to rotation and insensitive to lighting differences. However, it is dependent on MediaPipe Landmark detection.

Algorithm 1 Rotation algorithm
Used in ROI segmentation, landmark normalization, and posture method

$v_1(x, y) = mean(L5(x, y), L9(x, y))$ ▷ Index-Middle valley
$v_2(x, y) = mean(L13(x, y), L17(x, y))$ ▷ Ring-little valley
$V = v1(x, y) - v2(x, y)$

$$\theta = \tan^{-1}\left(\frac{v_{2y} - v_{1y}}{v_{2x} - v_{1x}}\right) * \frac{180}{\pi} \tag{3.1}$$

if *Left hand is True* **then**
  $V(x, y) = (-V_x, -V_y)$ ▷ Invert the vector
end
if $V_y < 0$ **then**
  $\theta = \theta + 180°$ ▷ If the hand is upside down
end

rotation_center = mean(L0,L5,L17)
M = getRotationMatrix2D(rotation_center, θ, 1)
lengthX,lengthY = image.shape()
s = abs(M(0,0))
c = abs(M(0,1))
newX = lengthY*s+lengthX*c
newY = lengthY*c+lengthX*s

rotated = warpAffine(image, M, (newX,newY) ▷ Rotated image or landmars
return rotated

3.5 Tools

During the project, there were multiple tools developed for different purposes. Some of these purposes are ROI segmentation, data collection, database filtering, and more. This is a description of those methods.

3.5.1 Blur detection

Blur detection is a classical computer vision task to assess the sharpness of an image. The perception of blur is closely related to the level of contrast, detail, and high-frequency presence in an image.

In our work, we've had two reasons for blur detection. In the proposed system a ROI with high blur content will yield poor information for the matcher, and motion blur can easily emerge from quick movements when the palm is recorded. Therefore, a need to filter out blurry images is necessary to know if an image is rejected because of no match, or poor image quality. Secondly, when training our algorithms we utilized data collections, one collected by us the authors, and one at a data collection center on behalf of Precise Biometric. The data collection from the data collection center was performed in a much larger scope, but also in a less controlled way. Therefore a challenge with using the collection was the number of blurry images, mislabeled images, and images without palms. For our algorithms to be trained on "good" data, we had to create tools to filter out images of poor quality.

In the scope of this thesis, two different methods of detecting blur in images were implemented, a Fast Fourier Transform (FFT) method and a Laplacian method.

Fast Fourier transform

When constructing a Fast Fourier Transform (FFT) blur detector, the frequency domain representation of an image is computed by using the FFT algorithm [29]. The algorithm converts the image from the spatial domain to the frequency domain, by applying two one-dimensional Fourier Transforms, see Equation 3.2, which also reduces the number of computations [30].

$$F(k,l) = \frac{1}{N}\sum_{b=0}^{N-1} P(k,b)\cdot \exp\left(-2\pi i \frac{lb}{N}\right), \quad \text{where} \tag{3.2}$$

$$P(k,b) = \frac{1}{N}\sum_{a=0}^{N-1} f(a,b)\cdot \exp\left(-2\pi i \frac{ka}{N}\right)$$

The low frequencies in the obtained image, which correspond to the smooth regions of the image are then removed by applying a high-pass filter. This leaves only the high-frequency components, which correspond to the edges and details of the image that are often lost in blurry im-

ages. Lastly, by computing the inverse FFT, a filtered image is obtained. By taking the mean of the magnitude values in the filtered image and comparing it to a set threshold the image could be classified as blurry or not.

Laplacian variance

Laplacian blur detection is an image processing technique used to detect edges and sharp transitions in an image. The image is first grayscaled using a single channel. Then, a Laplacian filter is applied to the image, convolving it with a Laplacian kernel (see Figure 3.4) to obtain the second derivative of the image intensity. This results in a new image that enhances the edges and features in the original image. Edges and sharp features are defined by rapid changes in intensity in a grayscale image, which the Laplacian filter can effectively highlight [31].

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 3.4: Laplacian kernel

After obtaining the Laplacian image, the variance of the pixel values can be used as a blur metric. A higher variance indicates that the image has more edges and features and therefore less blur. In contrast, an image with a lower variance is considered to be blurrier. Lastly, the images are classified as either blurry or not blurry based on a pre-defined threshold that is set after inspecting the variance of multiple palm images.

3.5.2 Hand distance - area

The logic behind the ROI segmentation ensures that the ROI is chosen at the same location invariant to change in distance. However, the resolution or quality of the ROI is dependent on the distance between the camera and the palm.

To ensure the distance between the palm and the sensor, an area-based method was developed to guide the placement of the hand. A bounding

box is set around the hand, and the area of the bounding box is continuously measured, and by setting a threshold value for the bounding box, real-time feedback can be provided to the user for adjusting the distance from the camera. This approach aids in acquiring an optimal distance for accurate ROI segmentation, thereby enhancing the reliability and effectiveness of the system.

3.5.3 Hand posture

Just as information about whether a hand is shown to the camera is relevant, the information about whether the hand is in a position to show usable features is also important. To capture images more reliably, where the palm is shown the "hand posture" method was developed.

The idea of the hand posture method is to make sure that the palm is visible. It does this with rather simple logic. First, the image is rotated (see Algorithm 1), then the algorithm can be divided into three conditions, and if all three are *True* the posture is considered good. Rotation of the landmarks is necessary for the hand posture method to function properly.

1. Fingertips are the highest-located joint of each finger
2. Thumb is not occluding the palm
3. Hand is perpendicular to the camera

The conditions explained in more detail:
1) If the y coordinate of each fingertip is greater than the respective DIP joints (distal interphalangeal) the first condition is true.
For example: $L_y12 > L_y11$, the y coordinate of the middle fingertip is greater than the y coordinate of the distal interphalangeal of the middle finger.

2) If the x coordinate of the thumb tip is greater (the opposite is true for a left hand) than the metacarpophalangeal joint of the index finger, the condition is considered satisfied.
Right hand: $L_x4 > L_x5$, **Left hand:** $L_x4 < L_x5$.

3) If a) and b) below are within the threshold, the third condition is true. The thresholds are experimentally set values, and specific to each setup.

a) if the z coordinate of landmark 9 is within threshold values the vertical angle is considered acceptable. This works as the z coordinates use the wrist landmark as origin, meaning that all z coordinates are given in relation to the wrist.
$t_{low} < L_z 9 > t_{high}$

b) If the ratio between the width of the palm to the length of the palm is within threshold, the horizontal angle is considered acceptable. See Equations in 3.3.

$$\begin{aligned} \text{width\_palm} &= \sqrt{(L_{x5} - L_{x17})^2 + (L_{y5} - L_{y17})^2} \\ \text{mid\_point}(x, y) &= \left(\left\| \frac{L_{x5} - L_{x17}}{2} \right\|, \left\| \frac{L_{y5} - L_{y17}}{2} \right\| \right) \\ \text{length\_palm} &= \sqrt{(L_{x0} - \text{mid\_point}_x)^2 + (L_{y0} - \text{mid\_point}_y)^2} \\ \text{width\_palm} &> \frac{\text{length\_palm}}{\text{threshold}} \end{aligned} \tag{3.3}$$

3.6 Openness classification

While testing the matching algorithm we learned that the openness of the palm is one of the key factors for a good matching score. As flexion (closing of the hand) alters the principal and secondary creases of the palm, from which the pattern matching is done. It also affects the shading of the palm. Therefore, investigating how the openness of the palm can be classified and integrated into the system design is of great importance.

Two main approaches were considered for this task. The first one was openness classification based on landmarks extracted from Mediapipe, and the second one involved using transfer learning on images from the *Openness collection* dataset. The second approach was tested on both

(a) Class 1 **(b)** Class 2 **(c)** Class 3 **(d)** Class 4

Figure 3.5: A selection of the *Openness collection* dataset with their respective class.

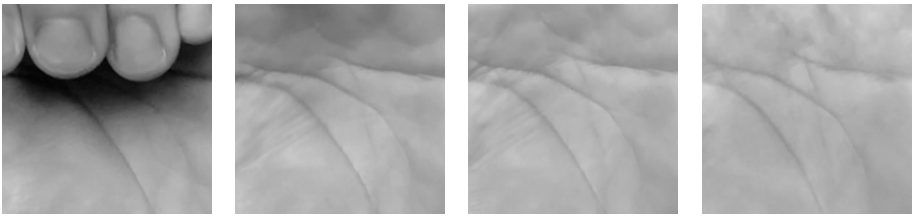
input images of original data in the dataset and of its extracted ROIs.

3.6.1 Data labeling

The dataset used for the openness classification task was the *Openness collection* dataset. This dataset was annotated manually, where each image was assigned a label (1-4) indicating the level of openness exhibited by the hand. Figure 3.5 illustrates the four different classes of openness defined and Figure 3.6 showcases the extracted ROI from these images.

Class 1 spans everything from a closed fist to the point where the nails of the little, ring, middle, and index finger are parallel to the camera (see Figure 3.5a). This means that class 1 images, as can also be seen in the ROI image (Figure 3.6a), are images where the principal lines are covered to a degree where relevant information is not visible. Therefore, these images need not be considered when matching as they are of too "poor" quality.

Class 2 is defined as when all the fingers (excluding the thumb) are perpendicular to the camera, or alternatively when the fingertips are pointing at the camera, see Figure 3.5b. These images will often yield an ROI with some shadow at the top. The creases are also deeper and more compact (see Figure 3.6b).



(a) Class 1 **(b)** Class 2 **(c)** Class 3 **(d)** Class 4

Figure 3.6: A selection of the extracted ROIs from the *Openness collection* dataset with their respective class.

Class 3 images are defined by the angle of the little to index fingers. The fingers are at a roughly 45-degree angle to the camera, being in between classes 2 and 4, see Figure 3.5c. The charachteristics of class 3 ROI images are in between those of class 2 and class 4.

Class 4 is defined by a completely flexed palm. The angle of the fingers is at a $\geq$ 90-degree angle and, see Figure 3.5d, the resulting ROI has creases that are stretched and flat (see Figure 3.6d).

3.6.2 Landmark based methods

In the case of the landmark-based approach, the entire dataset was processed through a landmark extraction script. This script used the MediaPipe hand landmarks segmenter to extract landmark coordinates, along with a blur detector to filter out images where landmarks could not be extracted or the image quality was insufficient. Additionally, the dataset was divided into a training set and a validation set, see Table 3.1.

Normalization methods

After extracting landmarks using MediaPipe Hand Landmarks, it's important to consider how to pre-process the data and what information to include for optimal results.

As mentioned in Chapter 3.3 the native format of the hand landmarks are normalized values with respect to the frame of the image (width and height), when it comes to x and y coordinates. The z coordinate is not

Landmark	(x,y,z)
1	(0, 0, 0.01)
2	(0.21, 0.02, 0.43)
.	
.	
.	
21	(0.43, 0.81, 0.02)

Table 3.3: Example of MediaPipe landmark coordinates

normalized but shares a magnitude scale similar to the x coordinates, and has its origin in the wrist landmark, see Table 3.3 for an example.

When feeding the classifiers the landmarks there is no obvious way to pre-process or normalize the data. Therefore, we chose to evaluate six different ways of pre-processing and normalizing the data to see which had optimal outcomes. The following six methods were evaluated.

1. "Native" untouched xyz data as given by MediaPipe.
2. "Native" xy data as given by MediaPipe excluding the z coordinates.
3. The x and y coordinates are expressed using the wrist landmark as origin, then normalized using max(x) and max(y) respectively. Excluding the z coordinate.
4. The x and y coordinates are expressed using the wrist landmark as origin, then normalized using max(x) and max(y) respectively. Including the z coordinate.
5. The x and y coordinates are expressed using the wrist landmark as the origin. However, normalized using max(x) and max(y) on only the palm, i.e. excluding fingers. Rotating the landmarks using algorithm 1 and including z coordinate.
6. The x and y coordinates are expressed using the wrist landmark as origin, then normalized using max(x) and max(y) respectively. The x and y landmarks are then rotated using algorithm 1 also including the z coordinate.

Method 1 serves as the reference, assessing the performance of classifiers on the unaltered data. In normalization method 2, the investigation focuses on the relevance of the z coordinate. Initially, when examining the data, uncertainties arose regarding the validity and significance of the z coordinates.

Starting from normalization method 3, all x and y coordinates have origin in "0"-wrist landmark and are normalized to max(x) and max(y), where the motivation is to emphasize the relationships between landmarks rather than their exact positions in the frame, considering the landmarks as a cluster instead. Method 3 evaluates the data without considering the z coordinate, while method 4 includes the z coordinate.

For method 5, the data is normalized by utilizing the maximum values on the palm, excluding the fingers. Rather than normalizing the coordinates including the fingers, this approach tightens the cluster and lessens the effect of fingers or an odd finger positioning. The landmarks are also rotated to be invariant to rotational differences in the input data. Method 6 closely resembles method 5, with the distinction that the landmarks are normalized to the absolute maximum and minimum coordinates, thereby incorporating the fingers again.

Machine learning classifiers

Five classifiers, namely SVM (RBF-kernel), k-NN (n=5), random forest, XGBoost, and a neural network were trained using a training dataset and a separate validation dataset, following the aforementioned split approach (see Table 3.1). Notably, an assessment of the *Openness collection* dataset revealed an inherent imbalance in class distribution. To make up for the unbalanced dataset the training was performed using class weights, which causes the model to pay more attention to the underrepresented classes. All classifiers except the k-NN were trained with class weights since the k-NN algorithm determines the class of a sample based on the majority class among its k-nearest neighbors. It does not explicitly consider class weights or adjust its decision-making process based on the distribution of the classes.

The architecture of the neural network classifier was a sequential model

Sequential Neural network architecture

Layer	Layer size
Input layer	(21*3)
Dropout (rate = 0.2)	
Dense layer, ReLu	20
Dense layer, Sigmoid	20
Dropout (rate = 0.4)	
Dense layer, ReLu	10
Output layer, softmax	4

Table 3.4: Neural network architecture

with an input layer of either size 63 or 42 depending on the inclusion of the z coordinate. As the openness classes are to be viewed as a decision based on the general formation of the landmarks and not any specific one, dropout layers were introduced to minimize over-fitting and the classifier to be too dependent on any one landmark. See Table 3.4 for the whole architecture. Furthermore, the F1-score and accuracy metrics were systematically computed to evaluate the classification performance of the trained models.

3.6.3 Transfer learning

As an alternative approach to the openness classification of the hand, transfer learning was used for classifying the data in the *Openness collection*. The pre-trained network used for this task was the residual network, ResNet50. The model consists of 50 layers and uses residual blocks that contain skip connections that allow the information to transfer from one layer to another, skipping intermediate layers. The model uses a building block that consists of two 3x3 convolutional layers, followed by batch normalization and a ReLU activation function. The skip connection adds the input to the output of the second convolutional layer. In addition to the residual blocks, ResNet50 also uses average pooling instead of fully connected layers at the end of the network [23]. The network is pre-trained on ImageNet, a large-scale visual database used to train and evaluate computer vision models. It contains millions of labeled images covering a wide range of objects and scenes. The dataset is organized into classes, with the commonly used subset consisting of

around 1.2 million images from 1,000 classes.

To adapt ResNet50 for the openness classification task, a top layer was added. This top layer, often called the classifier, is responsible for learning task-specific patterns and making predictions. By freezing the layers of the ResNet50 model, their weights are not updated during training, ensuring that the learned features are preserved. The trainable top layer is added on top of the pre-trained layers and trained to adapt the model's learned features specifically for the openness classification task. Leveraging the learned features from the pre-trained model through transfer learning significantly reduces the time and computational resources required to train the model from scratch.

Training of the models

The models were at first trained on labeled data from the *Openness collection* using the labels presented in 3.5. Later on the *Openness collection* was further processed where the input data was images of ROIs only (see Figure 3.6. The same labels were used for this training as well.

For images where the entire hand was visible, the input shape was (224, 224, 3), while for ROI data, the input size was (200, 200, 3). Both models utilized sparse categorical cross-entropy as the loss function and the ADAM optimizer.

To prevent over-fitting, an adaptive learning rate strategy was implemented. The initial learning rate was set to 0.0001 and was gradually reduced by a factor of 0.2 whenever the learning process plateaued. The minimum learning rate was set to $1e^6$.

Data augmentation was applied to the input data during training. The dataset was augmented with random transformations, such as rotation, zooming, and flipping, using parameters randomly generated with a factor of 0.1. These augmented images were combined with the original dataset to increase diversity. A simplified workflow of the training process is shown in Figure 3.7.

In addition to the mentioned techniques, class weights were used during training to account for class imbalance in the data and ensure balanced

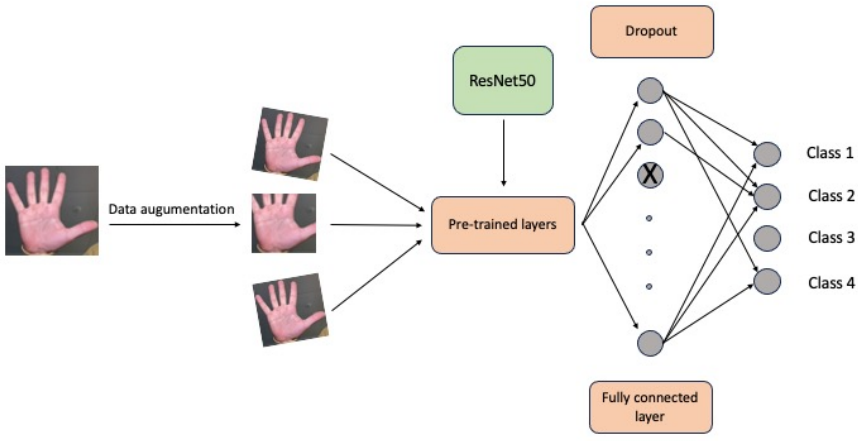


Figure 3.7: Overview of the training workflow.

learning between the classes. By assigning higher weights to the less-represented classes, the model was encouraged to give it more attention during the training process.

3.6.4 Lighting

When evaluating the system, we noticed that one of the critical factors was lighting on the palm. Consequently, we chose to set up tests to see how much and in what way lighting affected the matching algorithm. The main question was whether the verification images, taken with an iPad front-facing camera, and enroll images taken, taken with the back-facing camera of a smartphone, should be taken with extra lighting/flash or no extra light. To test this we set up two experiments with different enrollment cameras and subjects. Both subjects had 5 enroll images taken with flash and 5 images without flash. They were then matched against 165 verification images of the same subject without extra lighting and 165 with extra lighting, see Figure 3.1 for setup. The average scores of the matches were then computed for each case, see Chapter 2.3.5.

Subject 1: Iphone 7, single camera system 12MP.
Subject 2: Iphone 12 mini, double camera system 12MP.

3.6.5 Evaluating Openness by matching score

In the previous subsections, we describe the different methods for classifying openness. However, the usefulness of the openness metric is only hypothesized. Therefore, our last implemented test is intended to evaluate the relevance of an openness metric as well as the performance of the best classifier, that being the XGBoost classifier trained on data normalized using method 6, see Chapter 3.6.2.

Using an iPad verification video, see Chapter 3.2.1, 108 images of the palm in different states of openness were stored as templates. Similarly, using the back-facing camera of an iPhone 12 mini an enrollment video was recorded of the palm closing to a fist from a flexed state, see Figure 3.5d. From this video, 105 frames were extracted.

Both enrollment and verification frames were then classified into classes 1-4 using the XGBoost classifier, with the data pre-processed using method 6, see Chapter 3.6.2. See Table 4.5 for the number of samples in each class. Using the ROI segmentation described in Chapter 3.4.1, all images were then segmented and gray-scaled, extracting the ROI. All the ROIs were then matched against each other, totaling 11,340 matches. From these matches, the average score of each state was computed.

4 Results

In this section, the results of the essential tests for evaluating a contactless palm print recognition system are presented. Additionally, the evaluation of palm openness using various approaches is included, along with the corresponding results.

4.1 System design

A pipeline describing the functionality of a proposed design for a biometric palm print system is presented in Figure 4.1. Implementing an end-to-end solution allowing the enrollment and verification of palm prints.

The reasoning and details of the proposed system will be described more thoroughly in the subsections below and in the discussion.
But to gain an initial overview:

Enrollment

Users begin enrolling their hands by taking a photo or short video sequence, using the back camera, of the whole face of their hand. The image or video should be taken without flash or any other light besides natural occurring light in the room or space. This image or video is then segmented to an ROI template and labeled with an openness class as well as a left/right-hand label.

Verification

When the user then arrives at the access system an iPad is setup with a light source at the top part of the iPad, with the iPad showing the screen outwards. The user places the hand in front of the front camera, receiving feedback through the screen on the hand distance to the camera, hand posture, and palm openness and visibility. When the palm is positioned at an optimal distance, posture, and openness, the iPad records a short

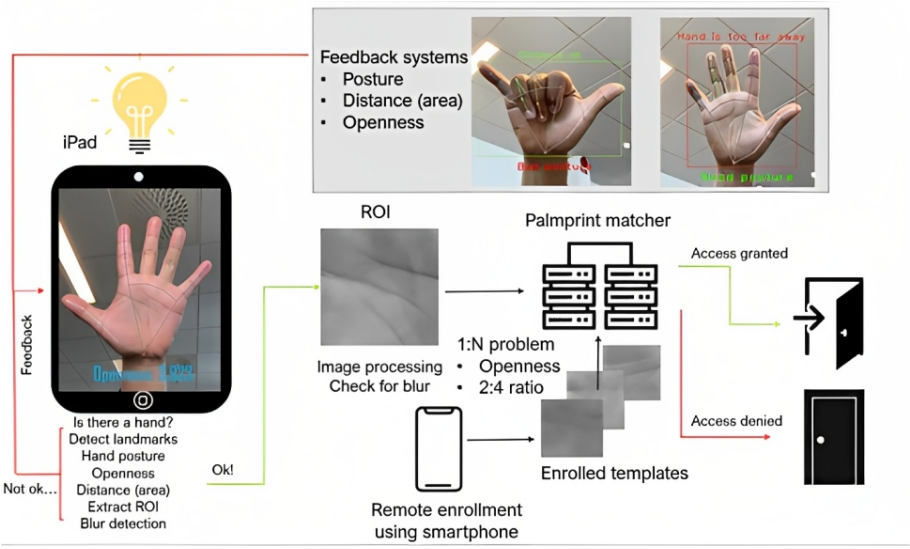


Figure 4.1: A simplified overview of the proposed system design.

video sequence of the palm. The recorded video captures the openness labels and landmarks, and the resulting image is then sent for processing.

Matching

The image is segmented using the saved landmarks. The ROIs are checked for blur and blurry ones are discarded. Then the ROI is sent for pre-processing and finally to the matcher, to be matched against enrolled templates. When matching, the algorithm can use the handedness labels as well as the openness class to optimize the number of enrolled templates it needs to go through (1:N-problem). If the verification ROI receives an acceptable score from any of the enrolled templates, the user is granted access.

4.1.1 Lighting

The enrolled images were taken using 12MP iPhone cameras (7 and 12 mini) whilst the verification images were taken using an iPad front-facing 12MP ultra-wide camera, see Figure 4.2. From Table 4.1, it can be seen that the enrolled images with no extra lighting in combination with the verification images with extra lighting yield significantly higher scores. After that, the light-to-light combination has the highest average score,

and verification images without extra light combined with the enrolled images with extra light produce the lowest average. Keeping in mind from Table 2.1 that scores above 7,200 are considered matches.

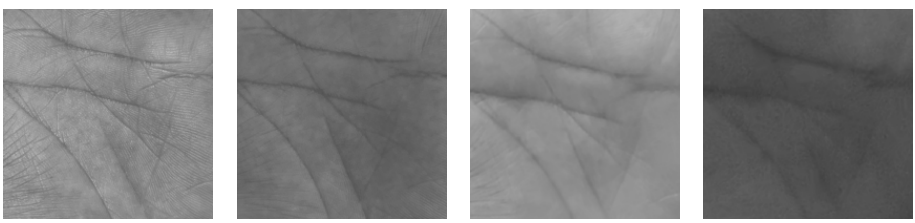
The following annotations are used for the four image types:

- Enroll light = EL
- Enroll no additional light = ENL
- Verify light = VL
- Verify no additional light = VNL

They can be seen in Figure 4.2 in the same order from left to right.

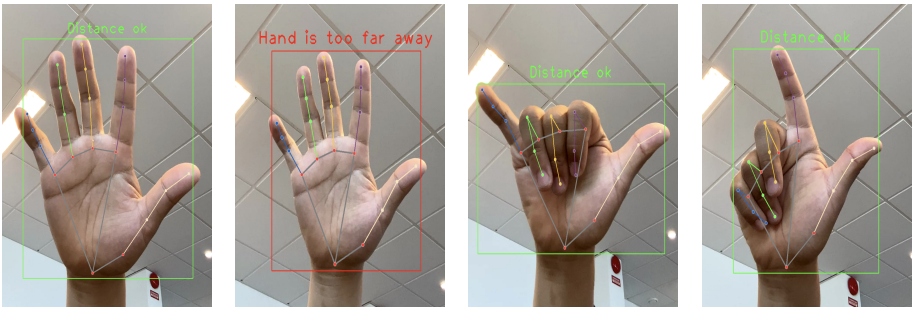
		Verify	
		Light	**No Light**
Enroll	**Light**	5628	3129
	No Light	8395	5157

Table 4.1: Average matching scores of verification and enrollment images taken with and without extra lighting



(a) EL (iPhone 12 mini) **(b)** ENL (iPhone 12 mini) **(c)** VL (iPad 9th Gen) **(d)** VNL (iPad 9th Gen)

Figure 4.2: Example images of the four light-to-no light cases of the same palm



(a) (b) (c) (d)

Figure 4.3: Four different user cases showcasing the functionality and limitations of the hand distance tool.

4.1.2 User feedback

There are three main systems to guide the user into the correct position, the distance method, the hand posture method, and finally the openness method. The aforementioned two are presented in this section.

Hand distance - area

The results from the implementation of the hand area tool are presented in Figure 4.3. The implementation functioned as expected, as can be seen in Figure 4.3a and 4.3b. However, the tool alone has limitations as it relies solely on the area of the bounding box for feedback. In the user cases demonstrated in Figures 4.3c and 4.3d, the area of the bounding box remains above the set threshold, even though the palm is not visible and the user's hand posture is poor.

Hand posture

Some examples from the implementation of the hand posture tool are presented in Figure 4.4. The method was tested thoroughly but only some images from the collection are presented to demonstrate the benefit and issues with the method. The three conditions perform as expected and function as an extension of the hand distance tool, see the examples below.

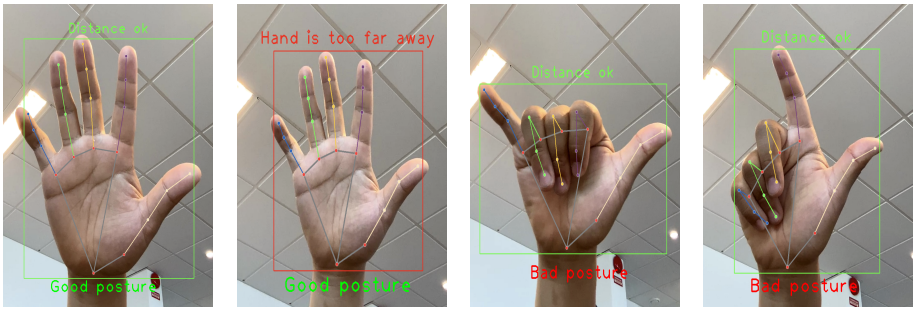


Figure 4.4: Example output from hand posture and hand distance methods

4.2 Openness classification

In this section, we present the results obtained from the classification of the openness of the hand, employing two distinct approaches. The first approach involves the utilization of landmarks extracted through MediaPipe, while the second approach relies on transfer learning.

4.2.1 MediaPipe landmarks approach

The extracted landmarks require addressing two crucial decisions: **1)** determining the appropriate normalization technique, if necessary, for the data, and **2)** selecting an optimal classifier to achieve the most favorable outcomes. To comprehensively evaluate these factors, we implemented various normalization methods in combination with multiple machine-learning classifiers. The subsequent sections present and discuss the results of these experiments.

Landmark normalization methods

When evaluating each normalization method and machine learning algorithm we chose to split the data by whole subjects, instead of doing it randomly by image from the whole set. Meaning that we left out certain people when training the algorithms that we later could use for testing. The reasoning behind the decision was that we didn't want the algorithm to have trained on the same individuals it later was evaluated on, simulating a more realistic situation.

Two data splits were made, see Table 3.1, and then the evaluation metrics were averaged according to the size of each testing set. In total six different normalization methods were tested, see Chapter 3.6.2 for the comprehensive list. In combination, five classifiers were trained and tested on each normalization method. The classifiers were initially implemented with minimal parameter tweaking, to be later improved.

From Table 4.2, we could see which normalization method performed best, and we also had an understanding of which machine learning methods showed the most promise.

From the table, it can be seen that the native format of the data has decent results across the board. However, normalization methods 5 & 6 have the highest average F1 scores. A big discrepancy can be seen between the data splits, where split 2 consistently gives a lower F1 score.

Regarding the classifiers, the SVM and XGBoost classifiers yield the best results with the highest F1 scores.

F1-score matrix: ML-Algorithms to Normalization Methods

	SVM			k-NN			XGBoost			NN			RF			
	1	2	w	1	2	w	1	2	w	1	2	w	1	2	w	Average
Norm 1	0.92	0.87	0.89	0.85	0.80	0.82	0.86	0.87	0.87	0.83	0.73	0.77	0.85	0.86	0.79	0.82707
Norm 2	0.91	0.86	0.88	0.84	0.77	0.80	0.84	0.87	0.85	0.80	0.69	0.73	0.81	0.89	0.86	0.82355
Norm 3	0.92	0.88	0.90	0.85	0.85	**0.85**	0.88	0.89	0.89	0.85	0.76	0.79	0.88	0.87	0.88	0.86028
Norm 4	0.93	0.88	**0.90**	0.89	0.81	0.84	0.90	0.87	0.88	0.85	0.77	0.80	0.88	0.88	0.88	0.86006
Norm 5	0.90	0.88	0.89	0.86	0.81	0.83	0.91	0.90	**0.90**	0.86	0.78	**0.81**	0.90	0.90	**0.90**	0.86525
Norm 6	0.90	0.87	0.88	0.87	0.82	0.84	0.92	0.89	**0.90**	0.86	0.78	**0.81**	0.91	0.90	**0.90**	**0.86701**
Average			0.88			0.83			0.88			0.78			0.86	

Table 4.2: The accuracy for the five different machine learning classifiers on both data splits (see Table 3.1) and the weighted average of the two.

Machine learning classifiers

In the following section, the result of the machine learning classifiers is presented.

The confusion matrices for the five different classifiers are presented in Figure 4.5. Following the evaluation of various normalization methods presented earlier in Table 4.2, it was determined that normalization method 6 yielded the best results. Consequently, we've chosen to highlight the machine learning classifiers trained using said data. The data split used during training was exclusively based on split 1 (see Table

3.1), due to time constraints. Furthermore, to account for the imbalanced nature of the data, class weights were employed during the training of all classifiers to try to improve the results further. To evaluate the performance of each classifier, key metrics such as accuracy and F1-score were assessed and the outcomes are presented in Table 4.3.

Classifier	**Accuracy**	**F1 Score**
XGBoost	0.92	0.92
SVM	0.92	0.92
k-NN	0.87	0.86
Random Forest	0.91	0.91
Neural Network	0.87	0.83

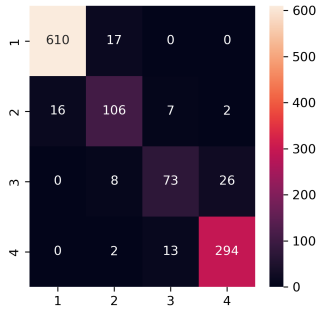
Table 4.3: Performance metrics for the five classifiers on data split 1.

A general pattern across all classifiers is that classes 2 and 3 are the classes with most misclassified samples. When comparing the confusion matrices of the XGBoost and SVM classifiers it can be seen that the SVM classifier has a slightly higher number of samples misclassified two classes away. The XGBoost classifier has in total 4 such samples that have been misclassified two classes away compared to 13 such samples for the SVM classifier.

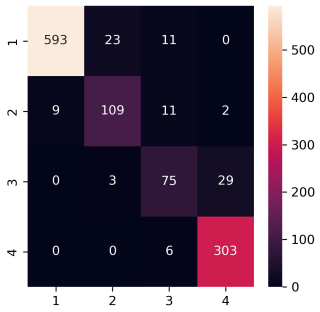
4.2.2 Transfer learning

The results from training the models using transfer learning are presented in this section. For both the original data and the ROI extracted data from the *Openness collection*, the training was conducted using the ResNet50 model with a modified top layer. Many different architectures of the top layer together with a variation of hyper-parameters were tested with the objective to optimize the models fully.

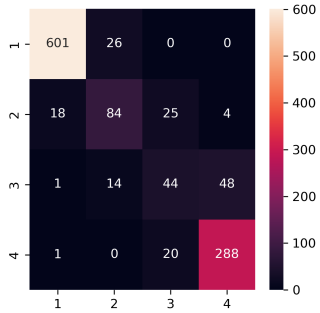
To ensure a robust evaluation of the model's performance on unseen hands, the validation set comprised hands from specific individuals who were not included in the training set, according to the splits in Table 3.1. This separation allowed for a rigorous assessment of the model's generalization ability to hands it had not encountered during the training.



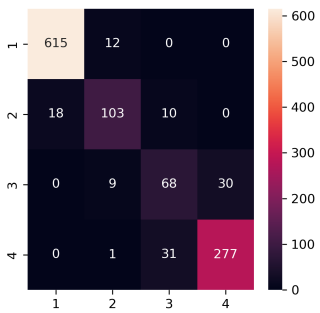
(a) Confusion matrix for the XGBoost classifier.



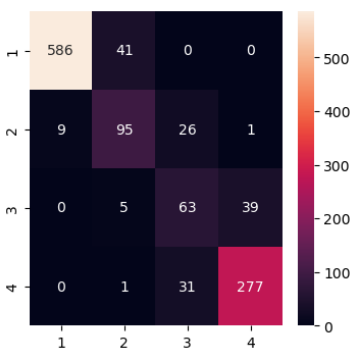
(b) Confusion matrix for the SVM classifier.



(c) Confusion matrix for the k-NN classifier.



(d) Confusion matrix for the random forest classifier.



(e) Confusion matrix for the neural network classifier.

Figure 4.5: Confusion matrices for five different machine learning classifiers using data split 1. The y-axis shows the true label while the x-axis shows the predicted label.

	Validation accuracy			Validation loss	
	Split 1	**Split 2**	**Weighted average**	**Split 1**	**Split 2**
PALM	0.9130	0.9002	0.9050	0.3122	0.3971
ROI	0.8309	0.8480	0.8416	0.6093	0.8005

Table 4.4: Comparison of the validation accuracy and validation loss for the different datasets and data splits respectively.

ResNet50

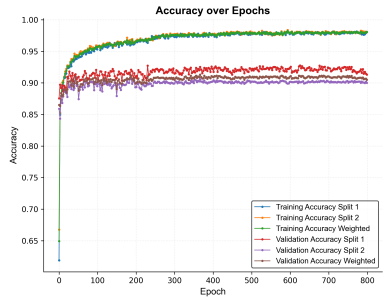
In this section, we present the results obtained from the ResNet50 model on both the PALM dataset (see Figure 3.5) and the ROI dataset (see Figure 3.6). Beginning with the PALM dataset, the top layer of the model consisted of a fully connected layer with 512 nodes, utilizing the ReLU activation function. To prevent over-fitting, a dropout layer with a dropout rate of 0.4 was incorporated. The training process spanned 800 epochs, considering both split 1 and split 2 (refer to Table 3.1). The combined weighted average of the two splits, along with the training and validation accuracies, is illustrated in Figure 4.6a. The training and validation losses for the two splits are presented in Figure 4.6b.

Regarding the ROI dataset, the top layer also consisted of a fully connected layer with a 512 number of nodes and used the ReLU activation function. Similarly, a dropout layer with a dropout rate of 0.4 was utilized. The training process spanned 800 epochs, and the evaluation metrics, analogous to those for the PALM dataset, are shown in Figure 4.7.

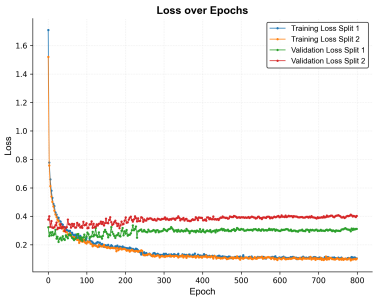
The final values for the evaluation metrics are summarized in Table 4.4 and Figure 4.8 demonstrates the confusion matrices for both datasets and splits respectively.

4.2.3 Matching score openness

The landmark-based classifier with the highest performance, XGBoost, underwent a final evaluation. Using XGBoost, the verification and enroll images were sorted into their respective classes, as presented in Table 4.5. Subsequently, all images were cross-matched against each other, using the *Precise Biometrics matcher*, and the average score was computed to see

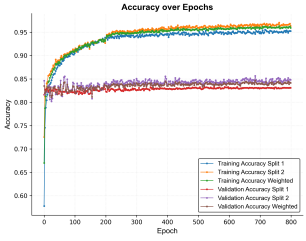


(a)

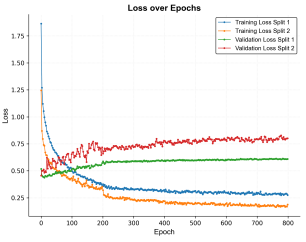


(b)

Figure 4.6: PALM dataset. a) Accuracy over epochs curve for the training and validation data for both data splits, together with the weighted average. b) Loss over epochs curve for the training and validation data for both data splits.

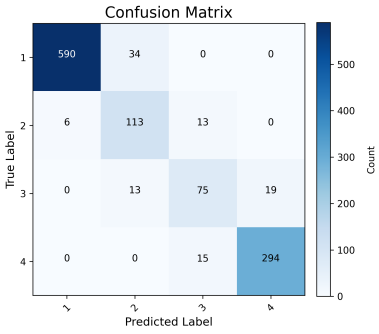


(a)

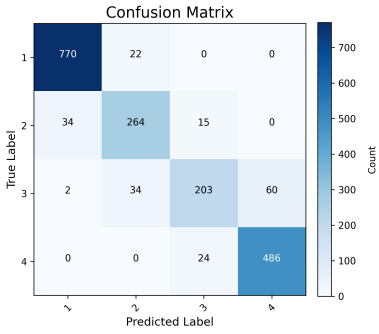


(b)

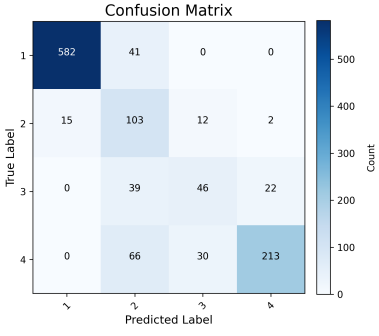
Figure 4.7: ROI dataset. a) Accuracy over epochs curve for the training and validation data for both data splits, together with the weighted average. b) Loss over epochs curve for the training and validation data for both data splits.



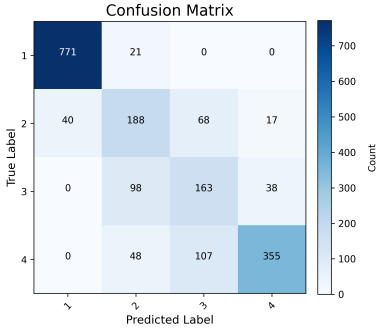
(a) PALM dataset split 1



(b) PALM dataset split 2



(c) ROI dataset split 1



(d) ROI dataset split 2

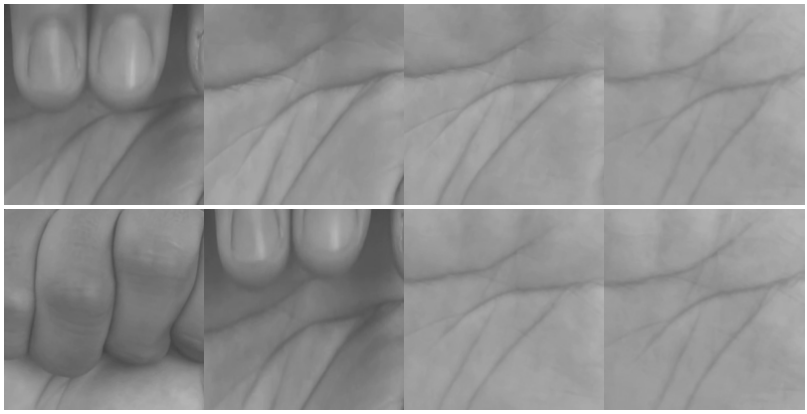
Figure 4.8: Confusion matrices for the PALM and ROI dataset for data split 1 and 2.

the relevance of the different classes, see Table 4.6 for the result. For a comprehensive explanation of the methodology, see Chapter 3.6.5.

In Figures 4.9 and 4.10 some representative examples have been chosen to show how the XGBoost classifier has sorted the data. From the figure, it can be seen that class 2 has both images with and without fingers included, see Figure 4.10b. And that the creases of class 2 images are much deeper, but also more distinct that the class 3 and 4 images. Class 4 images are correctly so flatter and more spread out in comparison to classes 2 and 3.

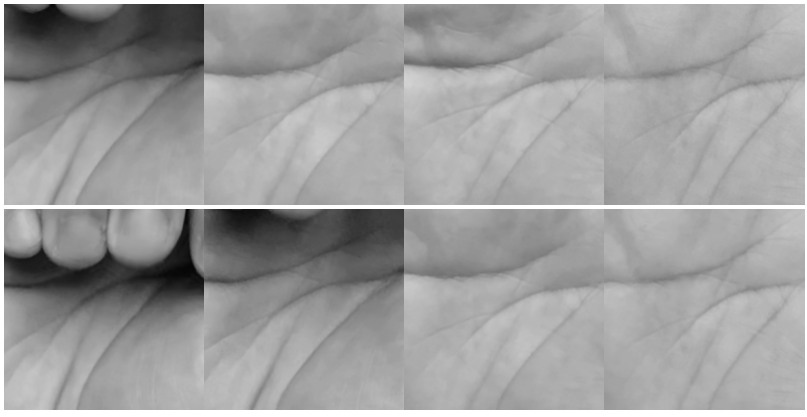
	Classes			
	1	2	3	4
Enroll	25	10	11	59
Verify	51	15	11	31

Table 4.5: Number of samples in each openness class, as classified by the XGBoost classifier using the normalization method 6.



(a) Openness class 1 **(b)** Openness class 2 **(c)** Openness class 3 **(d)** Openness class 4

Figure 4.9: Selected examples from *Enrolled* images classified by XGBoost.



(a) Openness class 1 **(b)** Openness class 2 **(c)** Openness class 3 **(d)** Openness class 4

Figure 4.10: Selected examples of *Verification* images classified by XGBoost.

From Figures 4.9 and 4.10 it can be seen that some of the images have been misclassified. The bottom ROI of Figure 4.9b should be a class 1 image. And the bottom ROI of Figure 4.10a should be of class 2.

		Verify			
		1	2	3	4
Enroll	1	3857	1473	1099	935
	2	751	5291	**7157**	4758
	3	599	3086	8809	7874
	4	544	2898	8652	9011

Table 4.6: Average matching scores between verification (VL) and enroll (ENL) images for each class

From Table 4.6, the highest average score can be observed along the diagonal, except for class 2. Classes 1, 3, and 4 exhibit the highest average scores when matched against their respective classes. The enroll 2 class instead produces the highest average score against the verify 3 class. Notably, five average scores closely approximate the 1/100,000 FAR threshold of 7,200, all within verification classes 3 and 4.

5 Discussion

5.1 System design

In this section, the various tests that prompted different system design decisions are discussed and explained in further detail. The main focus will be on lighting setup for enrollment and verification, and the result and relevance of openness classification, as these have quantifiable results presented.

5.1.1 Lighting

From Table 4.1 it is apparent that the combination of verification images taken with extra lighting (VL), and enrollment with no flash (ENL), produces the highest average score. This came as somewhat of a surprise as we assumed that images of higher quality would be the ideal match. Assuming images are taken with more light to give more contrast from the reflected surface, resulting in a higher resolution image.

But when examining the images more closely, see Figure 4.2, the quality of the smartphone images is definitely better than that of the iPad front-facing camera. When comparing the EL (see Figure 4.2a) and VL (see Figure 4.2c) images we can see a big discrepancy in terms of visible lines. We believe that this causes the EL images to have more lines extractable for the matcher than its VL counterpart, resulting in penalties for all patterns that are not matched, yielding a lower average score for EL to VL matches.

From this, the conclusion is drawn that the matcher yields better results when matching images of a similar level of detail, which is why the ENL to VL produces the highest average scores, see Figures 4.2b and 4.2c.

Following the same logic the matches between VNL (see Figure 4.2d) to EL images should result in the lowest average score. As the significantly

higher resolute enrolled images create too much of a discrepancy with the lower resolute VNL images, which is the case.

There is the possibility that the scores are a result of palm images taken differently, giving lower or higher scores due to differently positioned ROIs, meaning that the scores are not a reflection of the presence of light. But under visual inspection, the ROI templates in 4.2 are positioned almost identically on the palm as well as the openness is similar. Considering this we believe the discrepancy in scores to be too significant for such small differences.

In conclusion, this is the reason for having no flash for the enrollment images whilst the verification images should be lit up using an additional light source in the proposed system. An alternative or addition to the system would be to down-sample enrollment images of too high quality to gain more similar-quality images.

5.1.2 Hand posture & distance

One of the challenges that arise with contactless palm print recognition is to ensure the distance between the camera and the palm. If the palm is too close to the camera, an out-of-focus image will be obtained. However, if there is too much distance between the palm and the camera the dots per inch (DPI) will be too low, and not enough valuable information extracted.

As the iPad camera has no time-of-flight sensor, distance had to be controlled in another manner. To solve this, the area-based method was developed, see Chapter 3.5.2 for implementation. This method proved to have two main drawbacks. Firstly, as the area is calculated using a bounding box, by extending the fingers in certain manners the bounding box is falsely enlarged, see Figures 4.3c and 4.3d. Secondly, very small or large hands can have a bounding box area that is not representative of the distance to the camera. Resulting in out-of-focus or low-resolute images if the set distance interval is not wide enough.

To solve the first problem as well as other posture challenges, the hand posture method was developed. The hand posture method guarantees

that the fingers are not occluding the palm, by not overlapping any landmarks. It also checks the angle horizontally and vertically. By ensuring that the posture is good the distance method also works correctly, resulting in a system that requires both conditions to pass, see Figure 4.4. The feedback in our early implementation is only a text box. In an optimal system, this should be replaced by a combination of graphic and textual feedback, that guides the user.

In retrospect, an alternative solution to the distance hand area method would have been to make a bounding box of the palm landmarks instead, i.e. excluding the fingers. In this way, the bounding box is not affected by finger placement that can falsely enlarge the bounding box.

In conclusion, the proposed distance and posture methods function as intended. They solve the challenge of controlling the distance to the camera in most cases and ensure that the posture of the user's hand is open and ready for the openness classifier. With that said, the hand area-based distance controller leaves a lot of room for improvement for future work.

5.1.3 (1:N problem)

As the 1:N problem is not one of the challenges we have chosen to focus this thesis on, it will not be discussed to any length. With that said some insights have been gained regarding the subject that we would like to note for future work. A common challenge for biometric systems used in large-scale commercial settings is the number of matches that need to be processed. To improve on this we have some proposals that could optimize the number of samples looped through:

- By labeling ROI templates with handedness, i.e. right or left hand, N can be halved.
- The Openness classifier we have developed can be used as a metric to prioritize templates saved with the same openness class.
- The 2:4, or second-to-fourth digit ratio, is one that divides the world population physiologically. What is meant is whether the

index or ring finger is longer giving a larger or smaller than one ratio. Using MediaPipe Landmarks this metric could be stored and used for minimizing the number of templates matched against.

5.2 Openness classification

As mentioned two main methods of classifying openness were pursued. Landmark-based classifiers and transfer learning. The transfer learning approach was investigated to see how well openness could be predicted without the use of MediaPipe Landmarks, and in addition if it was possible to predict openness solely from the ROI. The results of the approaches are discussed in the sections below.

Data

It is important to note that openness classes are not natural phenomena that can be measured or defined exactly, but definitions that the authors have created, and should not be considered as an exact class but instead a scale. Furthermore, as the *Openness collection* was manually annotated by the authors, and the existence of mislabeled images is apparent when going through the collection, there are traces of human errors. Therefore, it would be impossible for the classifiers to reach 100 % accuracy, and if we were to say that 5% of images are mislabeled then the theoretical maximal accuracy possible would be 95%, in practice this number would most likely be lower.

5.2.1 Landmark based method

Analyzing Table 4.3, it becomes apparent that the XGBoost and SVM classifiers performed best, achieving a weighted average accuracy and F1-score of 0.90, using normalization method 6. Overall, all five classifiers demonstrated good performance.

Normalization methods

From Table 4.3 we see that all normalization methods function reasonably well with the lowest F1 score being 0.69 for normalization method 2 combined with the neural network classifier. With that said the highest

normalization method to classifier scores are definitely produced by normalization methods 5 and 6. The normalization, rotation, and new origin were implemented with the intention to express the landmarks as a cluster, invariant to hand size, absolute positioning on the frame, and rotation. And as a result, more clearly states the relation between landmarks for the classifiers. Our hypothesis was that normalization method 5 would be the most promising method as it expresses the landmarks in relation to the palm and not the fingers. And given more or different splits this might be the case as the average scores of normalization methods 5 and 6 are very close.

From the significantly lower split 2 scores, there are still uncertainties regarding the outcome, if there were more splits and more balanced ones in terms of class representation the result might have shown a different outcome. In conclusion, normalization method 6 was the ideal method according to our tests, and is as a result the one incorporated into the proposed system design.

Machine learning classifiers

While the random forest classifier showed comparable results to both SVM and XGBoost, the preference for XGBoost stems from a detailed examination of the confusion matrices. Notably, XGBoost and SVM exhibited a more balanced distribution of accurately predicted images across all four classes, where XGBoost performs slightly better when looking at samples misclassified more than one class away.

Furthermore, evaluation of the misclassified images and the confusion matrices illustrated in Figure 4.5, it is apparent that a majority of the incorrect classifications originated from class 3. The images of class 3 generally had the largest variations as different people's movements when closing a fist varies. Another reason behind the poor classification of class 3 could be the imbalance in the dataset. Applying class weights help to better capture and learn from the minority class that was tested. However, the effectiveness of class weights may vary depending on the dataset and problem at hand. By testing the SVM classifier without using class weights for instance showed that it had minimal to no impact as the accuracy and F1 score showed no improvement for split 1. However,

class 3 was most commonly misclassified to class 4 which isn't a huge problem for the end system, as they have high matching scores against each other, see Table 4.5.

5.2.2 Transfer learning

The results presented in Table 4.4 demonstrate that the ROI data was more challenging to classify compared to the other dataset, which was expected. Upon analyzing the dataset, it became evident that the images within different classes exhibited significant similarities, making classification more difficult. This similarity can lead to confusion for the model as it tries to distinguish between similar patterns or features. This argument is strengthen by studying the confusion matrices in Figure 4.8, where it is obvious that the model used for classifying the ROI dataset struggled heavily with class 3. Similar observations are found for the PALM dataset, but the results there are slightly better.

Examining the loss per epoch curve in Figure 4.6b, we observe a noticeable gap between the training and validation loss curves. Such a discrepancy suggests that the training dataset is unrepresentative of the entire data distribution, as it fails to fully capture the essential features necessary to accurately represent a new palm image. Moreover, the validation loss in Figure 4.6b appears to be noisy and stochastic, indicating that the validation data is limited and does not adequately represent the training data. This further supports the notion of an unrepresentative validation dataset.

Similar behavior can be observed for the ROI dataset, as depicted in Figure 4.7b. However, the deviations in this case are even higher and increase with time. This suggests that the limited amount of data available in the training dataset is a significant factor contributing to the observed issues. The model struggles to generalize due to the insufficient representation of variations and patterns in the training data, leading to an increasing deviation between the training and validation loss curves.

While employing cross-validation could potentially address the issue of unrepresentative training data, it may introduce additional uncertainty in this specific case due to the high similarity of input images from a

subject. Therefore, a more sustainable solution would involve collecting more data and applying further pre-processing techniques to enhance the PALM dataset. By collecting a larger and more diverse set of images, the training data can better capture the essential features and variations necessary for accurate palm classification. Additionally, applying pre-processing techniques that focus on reducing unnecessary image information and highlighting key features relevant to palm classification can further enhance the dataset's quality and improve the model's performance.

In summary, the observed challenges in classifying the ROI data and the unrepresentative nature of the training dataset indicate the need for more diverse and informative data, as well as appropriate pre-processing techniques, to overcome these limitations and improve the model's classification performance. The model can also be optimized further to better classify the ROI data. This task which was something we decided to add to the project a bit late, and its clear insufficient time and research has been conducted to obtain desired results.

5.2.3 Relevance of the openness classifiers

At the beginning of the project, when evaluating the scores of different matches we could see significantly lower scores for some ROI images that under visual inspection were assessed as good-quality images. After further inspection, subtle differences in width, depth, and distance between creases could be seen between the images, it was from this observation that the hypothesis that the openness made a big impact. In comparison, contact-based palm print or fingerprint systems don't face this problem as the hand or finger is pressed against a surface.

From the scores presented in Table 4.5, it is evident that higher scores are obtained when matching the class to itself, resulting in the best scores for classes 3 and 4. Additionally, our assessment was that class 1 images had too few visible lines to yield satisfactory matching results, while class 2 was to be considered borderline in terms of suitability. When inspecting Figures 3.5a and 3.5b the lines are almost completely blocked for class 1 images. For class 2 there is also the case of some images

having fingers inside the frame, blocking the principal lines. Classes 3 and 4 were intended to result in ideal ROI templates and matching scores. When inspecting Figures 3.5c and 3.5d the ROI images are of the highest quality. Despite the small-scale nature of the test, we assert that the results strongly support the validity of our hypotheses, thereby confirming and validating our selection of the main topic: palm openness, for this thesis.

5.3 Conclusion and future work

In the scope of this thesis, we have shown a novel design of a palm print recognition system that is fully functional. With the hardware setup, feedback systems, as well as ROI segmentation method, described, a ROI can be segmented to a quality that reaches the necessary FAR values of a commercial biometric system. The relevance of openness as a metric has been shown through the correlation it has with matching scores, with images of the same level of openness having the highest scores. Subsequently, we have shown that both landmark-based methods using MediaPipe, and transfer learning with ResNet and the whole palm can be implemented to classify openness with good accuracy. Using transfer learning solely on the ROI to classify openness resulted in an accuracy of 0.84 which is lower than the other methods but a method that could be considered with further work. For the landmark-based approach the XGBoost classifier in combination with normalization method 6, showed the most promise, with the highest average F1-score of 0.90. The distance and posture methods are functional, but we strongly believe that there is a better way of ensuring the distance between the palm and the camera.

For future work, we would be interested in seeing if down-sampling high-quality images to match lower-quality images could work to improve matching scores. It would also be interesting to see if a CNN could be trained to geometrically reconstruct a palm ROI from one openness class to another, for instance from class 3 to 4.

Bibliography

[1] Anil K Jain, Patrick Flynn and Arun A Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007.

[2] Adams Kong, David Zhang and Mohamed Kamel. "A survey of palmprint recognition". In: *pattern recognition* (2009). DOI: https://doi.org/10.1016/j.patcog.2009.01.018.

[3] Kah Ong Michael Goh, Tee Connie and Andrew Teoh. "Touch-Less Palm Print Biometric System." In: vol. 2. Jan. 2008, pp. 423–430.

[4] Goh Kah Ong Michael, Tee Connie and Andrew Beng Jin Teoh. "Touch-less palm print biometrics: Novel design and implementation". In: *Image and vision computing* (2008). DOI: https://doi.org/10.1016/j.imavis.2008.06.010.

[5] Schmidt R. Dar H. and H.M Nitowsky. "Palmar crease variants and their clinical significance: A study of newborns at risk". In: *Pediatric Research 11 (2), pp. 103-108* (1977). DOI: doi:10.1203/00006450-197702000-00004.

[6] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* (2015). DOI: https://doi.org/10.1126/science.aaa8415.

[7] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[8] Trevor Hastie, Robert Tibshirani, Jerome H Friedman and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[9] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. "Deep learning". In: *Nature* (2015). DOI: https://doi.org/10.1038/nature14539.

[10] Gary Marcus. "Deep learning: A critical appraisal". In: *arXiv preprint arXiv:1801.00631* (2018).

[11] Theodoros Evgeniou and Massimiliano Pontil. *Support vector machines: Theory and applications*. Springer, 2001.

[12] Kashvi Taunk, Sanjukta De, Srishti Verma and Aleena Swetapadma. "A brief review of nearest neighbor algorithm for learning and classification". In: *IEEE* (2019). DOI: https://doi.org/10.1109/ICCS45141.2019.9065747.

[13] M Ohlsson and P Edén. "Introduction to artificial neural networks and deep learning". In: *Lund: Computational Biology, Biological Physics, Department of Astronomy, and Theoretical Physics, Lund University* (2021).

[14] Leo Breiman. "Random forests". In: *Machine learning* (2001). DOI: https://doi.org/10.1023/A:1010933404324.

[15] Andy Liaw, Matthew Wiener et al. "Classification and regression by randomForest". In: *R news* (2002).

[16] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001). DOI: https://doi.org/10.1214/aos/1013203451..

[17] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: (2016).

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: 2009. DOI: https://doi.org/10.1109/CVPR.2009.5206848.

[19] Karl Weiss, Taghi M Khoshgoftaar and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* (2016). DOI: https://doi.org/10.1186/s40537-016-0043-6.

[20] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Science editions, 1949.

[21] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* (1986). DOI: https://doi.org/10.1038/323533a0.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* (1998). DOI: https://doi.org/10.1109/5.726791.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. “Deep residual learning for image recognition”. In: *arXiv preprint arXiv:1512.03385* ().

[24] Sasha Targ, Diogo Almeida and Kevin Lyman. “Resnet in resnet: Generalizing residual architectures”. In: *arXiv preprint arXiv:1603.08029* (2016). DOI: https://doi.org/10.48550/arXiv.1603.08029.

[25] Kukil. *Introduction to MediaPipe*. 2022. URL: https://learnopencv.com/introduction-to-mediapipe/ (visited on 03/05/2023).

[26] Google. *Framework conecpts*. 2023. URL: https://developers.google.com/mediapipe/framework/framework_concepts/overview (visited on 03/05/2023).

[27] Google. *Hand Landmarks Detection Guide*. 2023. URL: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (visited on 31/05/2023).

[28] Dane Brown and Karen Bradshaw. “Deep palmprint recognition with alignment and augmentation of limited training samples”. In: *SN Computer Science* (2022). DOI: https://doi.org/10.1007/s42979-021-00859-3.

[29] Paul Heckbert. “Fourier transforms and the fast Fourier transform (FFT) algorithm”. In: *Computer Graphics* (1995).

[30] R. Fisher and T. Drummond. “Fourier Transform”. In: (2003).

[31] Raghav Bansal, Gaurav Raj and Tanupriya Choudhury. “Blur image detection using Laplacian operator and Open-CV”. In: *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*. 2016. DOI: https://doi.org/10.1109/SYSMART.2016.7894491.