Evaluation of optimization components of a 3D to 2D landmark fitting algorithm for head pose estimation

Tim de Haan
11029668

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie
University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors
dr. Sezer Karaoglu
MSc. Minh Ngo
Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1090 GH Amsterdam

June 29th, 2018
Contents

1 Introduction 4

2 Related work 6

3 Methodology 7
   3.1 The projection matrix defined as a pinhole camera 7
   3.2 Point projection 9
   3.3 Optimization 9

4 Experiments 11
   4.1 Data description 11
   4.2 Data preprocessing 11
   4.3 Functor definitions 11
      4.3.1 Orthogonal projection 12
      4.3.2 Constant focal length or \( t_z \) 12
      4.3.3 3D rotation 13
      4.3.4 Ceres optimization settings 13
   4.4 Evaluation of the projection experiments 13
   4.5 Pose evaluation 13

5 Results 14
   5.1 Performance of different projection definitions 14
      5.1.1 Perspective vs. orthogonal projection 14
      5.1.2 Optimized FOV and \( t_z \) vs. constant FOV vs. constant \( t_z \) 15
      5.1.3 Euler angles vs. Quaternions 16
   5.2 Performance of Ceres Solver settings 16
   5.3 Dlib landmark detection 17
   5.4 Pose estimation 19

6 Discussion 20
   6.1 Future work 21

7 Conclusion 21
List of Figures

1. An overview of the proposed system. The input consists of a 2D image, the 3D landmarks from the 3DMM are projected onto the 2D landmarks from the input image. The projection is then optimized using an iterative approach. The projection parameters are used for pose estimation. 5

2. Facial landmarks. 7

3. The pinhole camera geometry. 8

4. Projection at the indicated iteration, the green points are the extracted landmarks, the blue points are the projected points of the 3DMM. 10

5. Landmark points comparison. 15

6. Landmark points comparison. 18

7. Example of unsuccessful landmark detection, resulting in an inaccurate projection. 18

8. Example of an unsuccessful rotation estimation (absolute angle difference = 44.6°), with good landmark fitting (norm. RMSE = 0.0361779). 20

List of Tables

1. Ceres linear solver types. 13

2. Performance difference in projection type, measured by the total optimization time (in seconds), the mean normalized RMSE, total of projections that have not converged and the linear solver type that Ceres Solver used. 15

3. Performance difference in FOV/tz parameter, measured by the total optimization time (in seconds), the mean normalized RMSE, total of projections that have not converged and the linear solver type that Ceres Solver used. 15

4. Performance of different constant FOV values. Expressed as the mean normalized RMSE and the total optimization time (in seconds). 16

5. Performance of different constant tz values using a perspective projection. Expressed as the mean normalized RMSE and the total optimization time (in seconds). 17

6. Performance difference for rotation types. Expressed as the total optimization time (in seconds), the mean normalized RMSE and the number of images that have not converged. 17

7. Performance of different linear solver types. Expressed as the total optimization time (in seconds), the mean normalized RMSE and the number of images that have not converged. 18

8. Comparison of landmark detection algorithms. Expressed as the percentage of images with a fitting error less than the specified values. Results are taken from [1]. 19

9. Mean and standard deviation of each rotation, splitted into different rotation ranges. The rotation is expressed as the absolute difference between the estimated rotation and the ground truth. 20
Abstract

Many applications would benefit from head orientation. Examples are improvements in the human-computer interaction, where the user would be capable of using the computer with head gestures, or automatic detection of driver drowsiness. Currently many head pose detection algorithms are implemented with the use of neural networks. However, these solutions are not preferable for devices that require a low computational cost, e.g. mobile devices. This thesis proposes a solution for head pose estimation by means of 3D to 2D landmark fitting, using an optimized and efficient non linear least squares approach.

1 Introduction

6 degree-of-freedom (DOF) pose estimation is the challenge of estimating the 3D pose of a 2D object from an input 2D image or video, defined as an orientation (i.e. roll, yaw, pitch) and position (i.e. translation \(t_x, t_y, t_z\)) relative to a coordinate system, given a set of point correspondences. Pose estimation, which is also known as perspective-n-point problem, is a broad subject in computer vision that has been extensively researched, examples are 3D body pose estimation [2], [3] and rigid body pose estimation [4]. This thesis focuses specifically on human head pose estimation analysis; something humans effortlessly can determine, however, for computers it remains a challenging task.

The head orientation contains crucial information for human nonverbal communication, for instance, it is an essential component for the expression of visual focus of attention. Moreover, when the orientation is tracked, it is possible to read gestures such as a nod or a head shake, altogether making head pose estimation useful for various purposes. Examples of applications that could make use of head pose estimation are the improvement of the human-computer interaction, the user would be able to communicate with the computer via gestures using his head. Games could use the orientation of the head as an extra control parameter, in addition the output of the display can be adapted accordingly to the head orientation, resulting in an extra dimension of realism. As a final example, as mentioned in [5] and [6] the head orientation can be useful for driver assistance, by alerting the driver when he or she shows signs of distraction. Likewise, the head orientation can be used in the field of self-driving cars, where the orientation is e.g. used to estimate whether pedestrians crossing the road have noticed the car approaching.

This thesis proposes to solve head pose estimation as an optimization problem, where the point correspondence distance is minimized between a 3D face projection and the 2D input face using a non-linear least square approximation. The point correspondences are defined by mapping a fixed set of points between 3D face model and 2D face images, this set consists of facial landmarks, which is a predefined set of characteristic facial points, examples are the tip of the nose or the contour of the eyebrows, figure 2 displays the full set of landmarks. We propose to use an off-the-shelf (widely accepted by the literature [7]) algorithm for detecting landmarks on 2D images. Furthermore, both the image plane and the world object are known beforehand the optimization, the image plane being the input image and the world object being a 3D face model. Therefore, the objective is to find a projection matrix that minimizes the projection error of the 3D face landmarks and the 2D image landmarks; figure 1 shows an overview of
the system. However, multiple optimization approaches exist for the projection of a 3D point, for example, there are various types of projections and multiple forms to define a rotation, which might result in performance difference. Thus, in this thesis we examine the influence of different settings which form a projection. Ultimately leading to the following research question: *Is it possible to estimate the head pose by means of 3D to 2D landmark fitting using a low-cost solution?*

In addition, head pose estimation can be divided into multiple components, which leads to the following sub-questions:

1. Which optimization settings yield better performance for fitting 3D to 2D facial landmarks?
2. Which definition of 3D projection has the best cost-performance ratio?
3. What is the importance of landmark detection on pose estimation?

![Figure 1: An overview of the proposed system. The input consists of a 2D image, the 3D landmarks from the 3DMM are projected onto the 2D landmarks from the input image. The projection is then optimized using an iterative approach. The projection parameters are used for pose estimation.](image)
2 Related work

Head Pose Estimation: While various systems of head pose estimation have been implemented using multiple images or complex 3D scans as input [8], this system uses a single 2D image to determine the pose. Furthermore, existing systems have also used one-time user specific setups [9], which significantly decrease the flexibility of the system. Therefore, one of the strengths of this system is its capability of pose estimation using a single 2D image with no additional information.

The following survey [10] provides a comprehensive overview of different approaches that have been used for head pose estimation. The most basic method that is discussed, is the appearance template method, where the pose of the input image is set equal to the image from a labeled database with the highest similarity. However, the main disadvantage is the erroneous assumption that image similarity equals similar pose. The next discussed method is the detector array, which consists of a set of detectors, each specific to a certain pose, the main disadvantage of this method is the increase in complexity that comes with the increase in performance, in addition, each detector has to be trained on a specific pose, which requires a large set of training data. The next method that is discussed, is nonlinear regression. According to the author [10], this method is able to achieve the highest performance in practice, which makes it particularly interesting to research. Neural networks are the most widely used nonlinear regression methods for pose estimation. Especially in recent years, where they have proven to be very capable for this purpose. However, the primary trade-off for their capabilities is the expensive computation, something that makes them less convenient for mobile applications, which have limited processing power and memory resources. With the increasing popularity of mobile devices, fast and resource efficient methods for pose estimation become an interesting research topic. Another disadvantage of neural networks are their need for a large training dataset in order to achieve good performance. Unfortunately, there aren’t many annotated datasets available and annotation is a costly process, which makes neural networks less optimal for pose estimation, when a training set is not available or cannot be constructed.

A good candidate that meets the above requirements is least squares regression, a simple, fast and applicable iterative method for data fitting problems, such as pose estimation. This method is widely used for its low computational demands [11], [12], additionally, this method requires a smaller training dataset, compared to neural networks. The wide applicability of least squares has led to the development of libraries that concentrate specifically on the optimization of problems using least squares regression. This thesis uses Ceres Solver [13], a fast, robust and extensively tested library, that has been extensively used [14], [15], [16]. Ceres Solver also runs natively on both desktop and mobile environments, which makes this system very portable.

Landmark Localization: Landmark localization has been extensively researched [1], [17], [18], which has led to the development of Dlib [7], a machine learning toolkit that has its usage in many complex applications, such as image processing. Dlib is capable of rapidly detecting the full set of landmarks, this process can be divided into two components, the face and facial landmarks detection. The face is detected with a pre-trained shape predictor based on a
histogram of oriented gradients combined with a linear classifier \cite{7}, trained on
the iBUG 300-W dataset from \cite{1}, the landmark detection is an implementation
of \cite{19}. In this thesis, \textit{dlib} is used for the landmark localization.

\textbf{3D Morphable Models:} The 3D Morphable Model (3DMM) \cite{20} has offered
many possibilities for computer vision \cite{21} \cite{22}. This model consists of a set of
vertices that jointly form a mean 3D face model of 3D face scans, additionally,
its morphable capabilities allow for a 3D face model that can recreate shape
and texture. Although these capabilities can be used to further extend this
system to 3D face recreation, these features will not be used. Instead, only the
coordinates of the vertices will be used for pose estimation, the mean 3D face
model will be used to exclusively research the influence of different optimization
components.

The 3D model that is used for this thesis, is the Surrey Face Model \cite{23},
which has the same capabilities as the 3DMM that are needed for landmark
fitting. This model is freely available on GitHub\footnote{https://github.com/patrikhuber/eos}.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{facial_landmarks.png}
\caption{Facial landmarks}
\end{figure}

\section{Methodology}

\subsection{The projection matrix defined as a pinhole camera}

The objective is to estimate parameters that describe the head pose of the 3D
face by using an optimized least squares approximation. These parameters are
values of a projection matrix that project the points from a 3D space onto the
2D image plane. This projection matrix is described by the pinhole camera
model, which is used for its useful approximation that defines the mathematical
geometry of the projection. Figure 3 briefly describes the model, each point in
the 3D space is projected straight through the pinhole onto the image plane,
where point \((x_0, y_0, z_0)\) is projected onto the center of the image plane. Un-
derstanding the principle of this model is necessary to correctly interpret each
value of the projection matrix.

\footnote{https://github.com/patrikhuber/eos}
The camera matrix $P$ can be divided into a matrix $K$ and $[R|t]$:

$$P = K[R|t]$$

$K$ represents the intrinsic parameters of the camera, these are internal camera-specific values that consist of a focal length in the x and y-direction ($f_x$ and $f_y$), the principal point ($c_x$ and $c_y$) and a skew coefficient ($s$). Throughout this thesis it is assumed that the input image is not skewed, and therefore the value $s$ will be set to 0, furthermore, principal point ($c_x, c_y$) is assumed to be the image center on all input images, resulting in $c_x$ being half of the image width and $c_y$ half of the image height. Lastly, it is assumed that pixels are square, thus, $f_x = f_y$.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f & 0 & \frac{\text{image width}}{2} \\ 0 & f & \frac{\text{image height}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$[R|t]$ is a matrix that represents the extrinsic parameters of the camera, these are dependent on external conditions, that is rotation and translation of the face relative to the camera. Therefore the extrinsic matrix is composed of a 3D rotation and translation matrix.

$$[R|t] = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{bmatrix}$$

Matrix $R$ can be divided into three rotation matrices with angles $\gamma$, $\beta$ and $\alpha$, which represent rotations around the z, y and x-axes respectively. These rotations are also known as the pitch, yaw and roll.

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

$$= \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$
3.2 Point projection

In order to estimate the projection parameters, a set of 2D image plane to 3D world point correspondences is needed. As discussed, facial landmarks are used as the image plane points, which are extracted with Dlib, beforehand the projection matrix estimation. The 3D world coordinates are extracted from the 3DMM. For each 2D landmark, the corresponding vertex index is defined in a simple text file that serves as a mapper (see example below). Each vertex has a 3D point, which is used for the projection estimation.

The mapper file is predefined by [23] and omits the jawline. Therefore, the points that define this line will not be used.

59 = 264 # lower lip middle bottom right
60 = 431 # lower lip right bottom outer
62 = 416 # upper lip right bottom outer
63 = 423 # upper lip middle bottom
64 = 828 # upper lip left bottom outer
66 = 817 # lower lip left top outer

As stated, the projection parameters are estimated by reducing the distance between each landmark point and the 3D projection of the corresponding 3DMM point. The projected point is obtained by multiplying the 3DMM point vector with camera matrix $P$. Obviously, the object is projected in homogeneous coordinates, this is necessary to allow vector operations such as rotation and scaling, which are necessary for pose estimation. Therefore each projected point has to be divided by the homogeneous coordinate. As a result, the projection for each point $i$ can be written as:

$$\begin{bmatrix}
u_i / \lambda_i \\
v_i / \lambda_i \\
\lambda_i / \lambda_i \\
\lambda_i / \lambda_i \\
\end{bmatrix} = P \begin{bmatrix} x_i \\
y_i \\
z_i \\
1 \\
\end{bmatrix}$$

3.3 Optimization

The next procedure, after the projection definition, is the optimization of its parameters. As discussed, this is solved using the non-linear least squares optimization library Ceres Solver. This library solves problems in the following form.

$$\min_x \frac{1}{2} \sum_i \rho_i \left( \| f_i (x_{i1}, ..., x_{ik}) \|^2 \right)$$

The expression in the summation is known as a residual block, $f$ is known as the cost functor, which defines the projection error distance between the landmarks and the projection, that depends on the parameter blocks $[x_{i1}, ..., x_{ik}]$. $\rho_i$ is a scalar function used to reduce the influence of outliers, but since there are none in this problem definition, $\rho_i$ is unused. The functor that Ceres Solver uses to compute the residuals, the parameter bounds and residual blocks are
undefined. Therefore it is the user’s task to properly integrate Ceres Solver into the problem definition, in order to achieve the best possible performance.

In this task of point fitting, and as is commonly done for this type of tasks \cite{13}, a residual block is added to the equation for each point correspondence. Subsequently, for each residual block, two parameter blocks are defined, one for rotation and one for translation and the intrinsic camera parameter (the focal length). Below is an example of the resulting objective function, if the pinhole camera model is followed. \( N \) represents the total amount of landmarks and cost functor \( f \) is the projection error of the given parameter blocks.

\[
\min_{x} \quad \frac{1}{2} \sum_{i=0}^{N} \left( \|f_i (x_{i1}, x_{i2})\|^2 \right)
\]

\[
s.t. \quad x_{i1} = (\alpha, \beta, \gamma), \quad x_{i2} = (t_1, t_2, t_3, f)
\]

Ceres Solver uses the above formula as a basis to automatically minimize the given parameters, by calculating the Jacobian matrices using Jets \cite{13}, which are expanded dual numbers.

As a starting point the default Ceres Solver settings are used and cost functor \( f \) defines the projection as defined by the pinhole camera model. That is, Euler angles describe the rotation and the projection is of perspective type, where both the focal length and translation in the z-axis are optimized. Other definitions for optimization are described in the experiments section. Figure 4 shows the resulting projection of four iterations, with the default settings as described

Figure 4: Projection at the indicated iteration, the green points are the extracted landmarks, the blue points are the projected points of the 3DMM
above. Note that projection points never fully fit the landmarks, this due to the fact that the 3DMM does not take facial shape into account since it is the model of a mean face.

4 Experiments

Through different experiments the performance of each functor implementation will be evaluated on quality and efficiency, the first is defined by the projection error distance and the latter by computation speed.

4.1 Data description

To evaluate each optimization approach, a dataset with face photographs is needed. This thesis uses the new 300-W dataset [1, 17, 18], this set consists of 600 photographs that can be used to project the 3DMM onto the landmarks detected by dlib. In addition, this thesis focuses on the optimization of one face per image, therefore, the cropped version of the set is used; this set contains only one face per image.

While a projection might seem accurate, the detected landmarks by Dlib are not guaranteed to be correct, which would result in an overall inaccurate projection. Therefore, to take this into consideration, the projected landmarks on each face will be measured against ground truth, which are the landmark annotations that the dataset provides. In addition, these projected landmarks are fitted onto landmarks that were detected by Dlib, therefore, projected landmarks fitted onto ground truth will be measured with ground truth. The difference between these two will present the error that is caused by Dlib.

To evaluate the head pose the Biwi Kinect dataset is used [8], [24], [25]. This set contains over 15 thousand images of 20 different people, with different head poses. Each image is annotated with its corresponding rotation and translation matrix.

4.2 Data preprocessing

The 300-W dataset is annotated in the same style as the set dlib was trained on. Therefore, the evaluation with ground truth consists of a one-to-one comparison of the dlib and annotated landmarks. Hence, no data preprocessing is needed.

The Biwi Kinect dataset also does not need any preprocessing, since all images contain one face per image.

4.3 Functor definitions

There exist multiple forms of projection, this thesis examines the impact on quality and efficiency of perspective and orthogonal projections. Note that a different projection also implies a difference in translation parameters, since orthogonal projections do not have a translation in the z-axis. Furthermore, a perspective camera matrix usually has a focal length and a translation in the z-axis for making projection bigger or smaller, where both parameters can be separately set to a constant value. The effect on the performance of these two possibilities will be evaluated, additionally, the difference in performance
for different fixed focal length values will also be evaluated. With regards to
3D rotation, there exist multiple definitions, two popular variants in computer
vision will be tested on performance, that is Euler angles and quaternions.

4.3.1 Orthogonal projection

Traditionally the projection that is used in the pinhole camera is of perspective
type, objects look smaller the farther away they are. However, it is also possible
to orthogonally project a 3D object, this type of projection does not take the
perspective effect into account. In order to produce an orthogonal projection,
the translation in the third dimension is removed from the object. Thus, the
extrinsic matrix is of the following form.

\[
[R|t] = \begin{bmatrix}
    r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
    r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
    r_{3,1} & r_{3,2} & r_{3,3} & 0
\end{bmatrix}
\]

Additionally, instead of a translation in the z-axis or a focal length param-
eter, the orthographic projection has a view frustum scale parameter. This
represents the region of the world space that is projected onto the image plane.
It is defined as a rectangle of the following size.

\[
2(\text{image aspect ratio}) \cdot \text{frustum scale} \times 2\text{frustum scale}
\]

To summarize, a perspective projection typically has seven parameters, which
are pitch, yaw, roll, 3D translation and a focal length. An orthographic projec-
tion typically has one parameter less: pitch, yaw, roll, 2D translation and the
view frustum scale.

The hypothesis is that an orthogonal projection produces less accurate pro-
jections due to the loss of the perspective effect, which produces less realistic
projections. In contrast, while having one parameter less, it will presumably
optimize faster.

4.3.2 Constant focal length or \( t_z \)

There are two parameters that can make an object appear farther or closer
to the camera, using a perspective projection. That is the focal length and the
distance between the object and the camera, thus \( t_z \). Although both parameters
cause this visual effect, they are not equivalent. Increasing \( t_z \) produces the same
effect when an object is placed farther away. In contrast, when the focal length
is decreased, in other words the object appears to be closer, the field of view is
also decreased.

The hypothesis is that either a constant focal length or \( t_z \) parameter results
in a faster optimization, due to the loss of one parameter that does not have to
be optimized.

Note that the perspective projection function used takes an angle defining
the field of view (FOV), instead of a focal length value. However, one defines
the other, so a short focal length equals a large angle for the field of view and
vice versa. As a result, these two terms will be used interchangeably.
Table 1: Ceres linear solver types

<table>
<thead>
<tr>
<th>Solver type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense normal Cholesky (DNS)</td>
</tr>
<tr>
<td>Dense QR-decomposition (DENSE QR)</td>
</tr>
<tr>
<td>Sparse normal Cholesky (SNC)</td>
</tr>
<tr>
<td>Dense Schur (D. Schur)</td>
</tr>
<tr>
<td>Sparse Schur (S. Schur)</td>
</tr>
<tr>
<td>Iterative Schur (I. Schur)</td>
</tr>
<tr>
<td>Conjugate gradient (CGNR)</td>
</tr>
</tbody>
</table>

4.3.3 3D rotation

A 3D rotation can be expressed in multiple forms, the effect of using a quaternion and Euler angles is examined. Euler angles are expressed in three values, where each rotates around an axis, equivalent to the pinhole camera model. However, quaternions are popular in computer vision due to their calculation speed.

The hypothesis is that quaternions outperform Euler angles in both aspects since the computation of a quaternion rotation is less expensive than using Euler angles.

4.3.4 Ceres optimization settings

Apart from the content of a functor, Ceres-solver also has settings for the optimization solver. These consist of different solver types. See table 1 for all the different types, the performance of each will be tested and elaborated.

4.4 Evaluation of the projection experiments

The evaluation will be based on two performance aspects of the functor implementation, quality and efficiency. Quality will be evaluated by calculating the normalized root mean squared projection error (RMSE) of the projection and landmarks. The normalization is defined by the interocular distance $d$ of the 2D landmarks.

$$d = \sqrt{(x_{20} - x_{29})^2 + (y_{20} - y_{29})^2}$$

Efficiency will be evaluated by timing the mean optimization process of each projection of the images in the dataset, expressed in seconds. The specifications of the setup that was used for these experiments are an Intel Core i5-5200U CPU and 8GB of RAM, running in a Linux environment.

As discussed, a projection might fit the Dlib landmarks, while not fitting the face due to incorrect landmark detection. The measure that will be used for this evaluation is, again, the mean normalized RMSE.

4.5 Pose evaluation

The original translation parameters cannot be recovered from only a 2D image, the translation parameters are only an approximation. Therefore, the pose is
described as a set of angles, namely pitch, yaw and roll. The Biwi Kinect dataset provides a rotation matrix, this rotation will be used as ground truth. In order to compare the estimated rotation with the ground truth, the rotation matrix has to be converted to the corresponding pitch, yaw and roll parameters. This is done using the conversion described using pseudo-code in Algorithm 1.

Pose estimation will be evaluated using the mean and deviation of the absolute difference of the estimated angles and the ground truth.

Algorithm 1 Conversion from rotation matrix R to Euler angles

1: procedure RotationMatrixToAngles
2: \( s \leftarrow \sqrt{R_{11} \cdot R_{11} + R_{21} \cdot R_{21}} \)
3: \( \text{singular} \leftarrow s < 1 \times 10^{-6} \)
4: if \( \sim \text{singular} \) then
5: \( x = \text{atan2}(R_{32}, R_{33}) \)
6: \( y = \text{atan2}(R_{31}, s) \)
7: \( z = \text{atan2}(R_{21}, R_{11}) \)
8: if \( \text{singular} \) then
9: \( x = \text{atan2}(R_{23}, R_{22}) \)
10: \( y = \text{atan2}(R_{31}, s) \)
11: \( z = 0 \)

5 Results

The results are divided into three sections, that is, the performance of different projection definitions and Ceres-solver settings and the accuracy of the Dlib landmark detection.

5.1 Performance of different projection definitions

This section only reports the results of different projection definitions. The starting point of the projection is as the pinhole camera model defines it. Meaning, the rotation is composed of three Euler angles that form a rotation matrix, the projection is a perspective type and both the focal length and translation on the z-axis are optimized. The Ceres-solver settings are set to default.

5.1.1 Perspective vs. orthogonal projection

When looking at table 2, it is visible that the optimization is much faster for the orthogonal projection, but as a trade-off it has a slightly higher error, which confirms the hypothesis. The loss of one parameter that has to be optimized has resulted in a faster and smoother optimization, which can be observed when looking at the total optimization time and number of projections that were not able to converge.

Figure 5 shows an example of both projection types, the blue dots indicate the landmarks of the projected 3D model, the green dots indicate the extracted landmarks. The closer the blue points are to the green ones, the better the
model has fitted. The differences are very minor, which the mean error also indicates. Only a minor difference can be seen when looking at the projection around the eyebrows and nose.

Table 2: Performance difference in projection type, measured by the total optimization time (in seconds), the mean normalized RMSE, total of projections that have not converged and the linear solver type that Ceres Solver used

<table>
<thead>
<tr>
<th>Projection</th>
<th>Opt. t</th>
<th>Error</th>
<th>Not converged</th>
<th>LST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perspective</td>
<td>22.0018</td>
<td>0.0519471</td>
<td>200</td>
<td>SNC</td>
</tr>
<tr>
<td>Orthogonal</td>
<td>3.11411</td>
<td>0.0526716</td>
<td>0</td>
<td>SNC</td>
</tr>
</tbody>
</table>

(a) Orthographic projection

(b) Perspective projection

Figure 5: Landmark points comparison

5.1.2 Optimized FOV and \( t_z \) vs. constant FOV vs. constant \( t_z \)

Since the orthographic projection does not have a \( t_z \) nor focal length/FOV parameter, this experiment can only be executed on the perspective projection. Therefore the landmarks will be fitted with a perspective projection, where both the focal length/FOV and translation in the \( z \)-axis are optimized, the focal length is set to a constant value and \( t_z \) is set to a constant value.

Table 3: Performance difference in FOV/\( t_z \) parameter, measured by the total optimization time (in seconds), the mean normalized RMSE, total of projections that have not converged and the linear solver type that Ceres Solver used

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Opt. t</th>
<th>Error</th>
<th>Not converged</th>
<th>LST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized ( f ) and ( t_z )</td>
<td>22.0018</td>
<td>0.0519471</td>
<td>200</td>
<td>SNC</td>
</tr>
<tr>
<td>Constant ( t_z ) (400)</td>
<td>3.10773</td>
<td>0.0542348</td>
<td>0</td>
<td>SNC</td>
</tr>
<tr>
<td>Constant ( f ) (45°)</td>
<td>3.12073</td>
<td>0.0547229</td>
<td>0</td>
<td>SNC</td>
</tr>
</tbody>
</table>

Table 3 shows the results of this experiment. Optimizing both the focal length and \( t_z \) results in a 7x optimization time increase, compared to setting
Table 4: Performance of different constant FOV values. Expressed as the mean normalized RMSE and the total optimization time (in seconds).

<table>
<thead>
<tr>
<th>FOV (in degrees)</th>
<th>Error</th>
<th>Opt. t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.0526714</td>
<td>6.1054</td>
</tr>
<tr>
<td>1</td>
<td>0.0526716</td>
<td>5.70861</td>
</tr>
<tr>
<td>2</td>
<td>0.0526737</td>
<td>5.29393</td>
</tr>
<tr>
<td>5</td>
<td>0.0526931</td>
<td>4.64955</td>
</tr>
<tr>
<td>10</td>
<td>0.0527714</td>
<td>4.00066</td>
</tr>
<tr>
<td>30</td>
<td>0.0535789</td>
<td>3.42484</td>
</tr>
<tr>
<td>45</td>
<td>0.0547229</td>
<td>3.00464</td>
</tr>
<tr>
<td>60</td>
<td>0.0563969</td>
<td>3.58352</td>
</tr>
<tr>
<td>90</td>
<td>0.0618723</td>
<td>6.2371</td>
</tr>
</tbody>
</table>

one of the two parameters constant. Again, the loss of one parameter results in a much faster optimization, which is visible when looking at both the total time and the number of projections that have not been converged. These results are very similar to the results of the orthogonal projection. However, there is a difference when looking at the mean normalized RMSE. The perspective projection still remains the most accurate projection, which is probably due to the higher precision of projection that is provided when optimizing both $f$ and $t_z$. Then, by a very minor error difference, the orthographic projection outperforms the perspective projection with a constant $f$ or $t_z$. However, the chosen constant value for $f$ or $t_z$ may have an effect on the optimization. Therefore, the error is calculated for different values of $f$ and $t_z$.

Table 4 shows the mean normalized RMSE for different constant values for the field of view. What immediately stands out is that a smaller field of view yields a smaller error. A small FOV has to be compensated with a larger $t_z$ value. This might indicate that Ceres Solver is better at finding more accurate approximations when working with large numbers, although it goes at the expense of the optimization time. Table 5 also confirms this observation. Furthermore, a field of view that is equal to 1°, yields the same mean normalized RMSE as when using an orthographic projection, however, the latter option has the advantage of having a faster optimization. Both projections have the same amount of parameters that have to be optimized since the perspective projection loses one parameter when the FOV is set to a constant value. Hence, the difference in speed might be due to the computational difference of both projection functions.

5.1.3 Euler angles vs. Quaternions

Both rotation types practically have the same error. However, the use of quaternions causes less images to converge, which might also explain the longer optimization time.

5.2 Performance of Ceres Solver settings

Table 7 shows the performance of each Ceres solver type, 541 images from the set have been optimized, Dlib was not able to detect the face on the remaining
Table 5: Performance of different constant $t_z$ values using a perspective projection. Expressed as the mean normalized RMSE and the total optimization time (in seconds)

<table>
<thead>
<tr>
<th>$t_z$</th>
<th>Mean norm. RMSE</th>
<th>Opt. t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.325497</td>
<td>7.25683</td>
</tr>
<tr>
<td>10</td>
<td>0.181766</td>
<td>4.87958</td>
</tr>
<tr>
<td>50</td>
<td>0.0827481</td>
<td>3.74846</td>
</tr>
<tr>
<td>100</td>
<td>0.0662451</td>
<td>3.59133</td>
</tr>
<tr>
<td>500</td>
<td>0.0537119</td>
<td>3.16727</td>
</tr>
<tr>
<td>1000</td>
<td>0.0529484</td>
<td>3.8423</td>
</tr>
<tr>
<td>2000</td>
<td>0.0527411</td>
<td>4.69052</td>
</tr>
<tr>
<td>4000</td>
<td>0.0526877</td>
<td>5.40498</td>
</tr>
<tr>
<td>8000</td>
<td>0.0526745</td>
<td>6.12975</td>
</tr>
</tbody>
</table>

Table 6: Performance difference for rotation types. Expressed as the total optimization time (in seconds), the mean normalized RMSE and the number of images that have not converged.

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Total optimization time</th>
<th>Total error</th>
<th>Not converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>21.8003</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>Quaternion</td>
<td>23.1668</td>
<td>0.0519472</td>
<td>208</td>
</tr>
</tbody>
</table>

59 images. The amount of maximum iterations is set to its standard value of 50, with a function tolerance of $10^{-6}$ (function tolerance = $|\Delta \text{cost}|/\text{cost})$. The total optimization time is the sum of the total time Ceres takes to optimize the parameters, the total error is the normalized mean RMSE and the last column represents the number of projections that did not fully converge.

The task of landmark fitting requires, relative to what Ceres Solver is designed for, a small set of parameters. This might explain the overall little difference between the first five linear solver types of table 7. However, iterative Schur and CGNR stand out because of their higher cost, this might be due to further optimization of the preconditioner that these linear solver types need [13].

5.3 Dlib landmark detection

If the error of the estimated projection is low, it does not necessarily mean that it is an accurate fit on the input image. This is due to Dlib, which does not guarantee to always find the correct landmarks. To visualize the inaccuracy, the model will also fit directly onto the annotated landmarks. Next, the error will be compared between the projection onto the dlib landmarks and the projection onto the annotated landmarks. Figure 7 shows an example of this error.

For this experiment, the projection is defined as the pinhole camera model since it produces the most accurate projections; which is more critical for this experiment than a fast projection.

When the 3DMM projection is fitted onto the landmarks detected by Dlib, it has a mean normalized RMSE of 0.0752541, when measured against ground
Table 7: Performance of different linear solver types. Expressed as the total optimization time (in seconds), the mean normalized RMSE and the number of images that have not converged.

<table>
<thead>
<tr>
<th>Solver type</th>
<th>Optimization time</th>
<th>Error</th>
<th>Not converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNC</td>
<td>21.4175</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>DENSE QR</td>
<td>21.5201</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>SNC</td>
<td>22.2703</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>D. SCHUR</td>
<td>21.3281</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>S. SCHUR</td>
<td>22.2113</td>
<td>0.0519471</td>
<td>200</td>
</tr>
<tr>
<td>I. SCHUR</td>
<td>22.4607</td>
<td>0.0519494</td>
<td>198</td>
</tr>
<tr>
<td>CGNR</td>
<td>25.6177</td>
<td>0.051988</td>
<td>242</td>
</tr>
</tbody>
</table>

Figure 6: Landmark points comparison

(a) Annotated landmarks  (b) Dlib landmarks

Figure 7: Example of unsuccessful landmark detection, resulting in an inaccurate projection

(a) Projection fitted on Dlib landmarks  (b) Projection fitted on ground truth

When the projection is directly fitted onto the ground truth, a mean
normalized RMSE of 0.061564 is obtained. Which means that, on average, Dlib causes an additional error of 0.0136901. Table 8 shows how Dlib compares with other landmarks detection algorithms, which are the winners of the two contests that were organized by [1]. Additionally, the Oracle curve is added, which is the minimum error that can be achieved [1].

Compared to the Oracle curve there is much room for improvement. However, the results suggest that Dlib can be regarded as a competitive and accurate algorithm for landmark detection when compared to current solutions.

Table 8: Comparison of landmark detection algorithms. Expressed as the percentage of images with a fitting error less than the specified values. Results are taken from [1]

<table>
<thead>
<tr>
<th>Method</th>
<th>&lt;0.2</th>
<th>&lt;0.3</th>
<th>&lt;0.4</th>
<th>&lt;0.5</th>
<th>&lt;0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dlib</td>
<td>0.50%</td>
<td>15.5%</td>
<td>38.0%</td>
<td>55.7%</td>
<td>70.0%</td>
</tr>
<tr>
<td>Yan et al. [26]</td>
<td>0.17%</td>
<td>4.17%</td>
<td>25.8%</td>
<td>54.0%</td>
<td>71.0%</td>
</tr>
<tr>
<td>Zhou et al. [27]</td>
<td>0%</td>
<td>2.50%</td>
<td>20.7%</td>
<td>47.7%</td>
<td>69.2%</td>
</tr>
<tr>
<td>Deng et al. [28]</td>
<td>0.17%</td>
<td>4.33%</td>
<td>26.8%</td>
<td>55.5%</td>
<td>74.3%</td>
</tr>
<tr>
<td>Fan et al. [29]</td>
<td>0.33%</td>
<td>14.3%</td>
<td>38.2%</td>
<td>62.0%</td>
<td>75.2%</td>
</tr>
<tr>
<td>Oracle</td>
<td>72.8%</td>
<td>97.2%</td>
<td>99.7%</td>
<td>99.8%</td>
<td>100%</td>
</tr>
</tbody>
</table>

5.4 Pose estimation

Until now the landmark fitting capabilities have been tested. This section shows the results of pose estimation by using the extrinsic projection parameters that are a result of landmark fitting. As discussed, this task will be evaluated using the Biwi Kinect dataset.

For this experiment accuracy is more important than time, therefore, a perspective projection is used.

From the 15678 images that the set provides, 10736 images were used. Dlib was not able to detect the landmarks on the remaining images. The total run time of the system was 2469 seconds, that is including landmark detection and optimization. Thus, the average run time of one image is 0.23 seconds. The total optimization time was 333 seconds, meaning that the average optimization time per image equals 0.03 seconds. Although Dlib shows its limitations regarding landmark detection, Ceres Solver (using the default linear solver type) proves to be a very fast and accurate optimization library, with an average normalized RMSE of 0.0424667.

Table 9 shows the mean and standard deviation of each rotation, expressed as the absolute difference between the estimated angle and the ground truth. From the table can be concluded that a larger rotation seems harder to estimate since for all rotations it is true that both the mean and standard deviation of the absolute difference increase. The system also seems to be much more accurate in estimating the yaw, whereas the pitch has the least accurate results, with a relatively high $\sigma$ when estimating small angles.

It must be noted that a larger error is not caused by a bad projection fit. Figure 8 shows an example, where the estimated yaw has an absolute angle difference of 44.6°, while the fitted projection is accurate. A possible explanation
Table 9: Mean and standard deviation of each rotation, splitted into different rotation ranges. The rotation is expressed as the absolute difference between the estimated rotation and the ground truth

<table>
<thead>
<tr>
<th>Range</th>
<th>Pitch</th>
<th>Yaw</th>
<th>Roll</th>
</tr>
</thead>
<tbody>
<tr>
<td>±(x ≤ 30°)</td>
<td>29.6° 16.9°</td>
<td>3.85° 3.50°</td>
<td>6.28° 6.45°</td>
</tr>
<tr>
<td>±(30° &lt; x ≤ 60°)</td>
<td>54.8° 14.5°</td>
<td>7.28° 7.00°</td>
<td>22.0° 16.4°</td>
</tr>
<tr>
<td>±(60° &lt; x ≤ 90°)</td>
<td>- -</td>
<td>26.6° 11.4°</td>
<td>- -</td>
</tr>
</tbody>
</table>

is the property that one rotation can be described by various combinations of different angles. Which would mean that Ceres Solver has found a different combination of rotations than the ground truth. However, this is not confirmed and should be investigated with further research.

Figure 8: Example of an unsuccessful rotation estimation (absolute angle difference = 44.6°), with good landmark fitting (norm. RMSE= 0.0361779)

6 Discussion

From the results it can be concluded that, with respect to landmark fitting, different types of projection result in performance difference. The perspective projection with an optimized focal length (or field of view) and translation parameter for the z-axis has the lowest error. This is probably due to the extra parameters, which allow for a more precise parameterization, thus resulting in a better-fitted projection. When fast optimization is a priority, the orthogonal projection has proven to be a better option. It has been tried to improve the optimization speed of the perspective projection, by eliminating either the parameter of the focal length or the translation in the z-axis, since accurate projections still can be obtained when only using one of the two parameters. When either the focal length/FOV or $t_z$ is removed, both the perspective and the orthogonal projection have the same number of input parameters. Nevertheless, the latter projection still offers the best ratio for quality and efficiency. Therefore, it can be concluded that the orthogonal projection, in combination with its parameters, 2D translation and frustum scale, offers a better parameterization
than a perspective projection, with 2D translation and FOV or $t_3$.

In conclusion, the perspective projection is recommended when quality is crucial. For a balance of quality and efficiency, it is best to use an orthogonal projection.

With regards to pose estimation, as can be seen from the results, landmark fitting is a competitive option. It is capable of estimating an overall accurate orientation. Although it has proven to not always have a reliable estimation. As was discussed, this might be due to incorrect angle translation from the rotation matrix. However, further research should confirm this.

Coming back to the research question and sub-questions, it is possible to estimate the head orientation by means of 3D to 2D landmark fitting using a low-cost solution. Different projections yield different performances, generally, the orthogonal type should be used for fast optimization. The perspective type should be used for a higher accuracy. Overall the orthogonal projection type offers the best cost-performance ration. The importance of landmark fitting on pose estimation is that the optimized parameters for the projection can be used to estimate the head pose.

6.1 Future work

One considerable disadvantage of the proposed solution is the restriction of having to use all landmarks. This restriction is caused by Dlib, which is not able to return any landmarks when no frontal facing face is detected. As a result, the system is not capable of optimizing the pose for faces that are partially visible, due to large rotations. Ideally, Dlib would return all possible landmarks it can detect. Then, it would be possible for Ceres Solver to optimize the projection for the given landmarks.

Furthermore, the used set of landmarks are defined by the mapping from 2D landmark points to the 3DMM vertex indices. It would be interesting to research the effect on performance when a smaller set of landmarks is used. Since it is feasible that a smaller set of landmarks yields a faster optimization, due to fewer optimization steps that have to be computed. Subsequently, research could demonstrate if there exists a set of landmarks that delivers the best quality and efficiency ratio.

Finally, this method for pose estimation can be used for the process of face alignment that is needed for 3D face reconstruction. This can be achieved by optimizing the parameters of the 3DMM that morph the model accordingly to the input image.

7 Conclusion

The proposed algorithm in this thesis contributes to the field of efficient 3D to 2D landmark fitting algorithms used for head pose estimation. The performance of different projection types has been examined, delivering an indication of which projection should be used to either gain efficiency or quality. To summarize, an orthographic projection should be used when efficiency is preferable, whereas a perspective projection should be used for its quality. The projection has proven to be accurate and the pose estimation from the projection parameters
achieves, on average, good pose estimations, however, it has also shown to give less accurate and inexplicable estimations.

The projection optimization achieves positive results for both quality and efficiency, thanks to its use of the nonlinear least squares regression library Ceres Solver. However, Dlib shows limitations when the face on the input image is partially visible, due to its incapability to detect partially visible faces. Despite this limitation, Dlib has proven to be fast, which all in all result in an algorithm with low computational cost, that is highly portable to e.g. mobile devices.

References


