Smartscan

Giuseppe Carere[1], Bernard Geurts[2], Bas van 't Hof[3], Fred Holtkamp[4], Erwin Luesink[5], Matthias Schlottbom[6], Julian Suk[7], Michelle Sweering[8] and Jelmer Wolterink[9]

Abstract

This report describes and develops different methods for converting 3D time series data to surface representations. The considered methods contain public domain mesh generation software, as well as linear regression models and surface representations using signed distance functions. We provide a simple code base for the latter two methods such that they can be used for further research in a simple manner. We apply and test the algorithms on a point cloud for a foot model. All methods yield good representations of the underlying foot geometry. Such algorithms can therefore have a big impact on handling foot problems.

[1]TU Eindhoven, The Netherlands
[2]University of Twente, The Netherlands
[3]Vortech BV, The Netherlands
[4]Fontys Paramedische Hogeschool, The Netherlands
[5]University of Twente, The Netherlands
[6]University of Twente, The Netherlands
[7]University of Twente, The Netherlands
[8]CWI, The Netherlands
[9]University of Twente, The Netherlands

KEYWORDS: meshing tools, signed distance function, implicit neural representation, linear regression

3.1 Introduction

More than half (57%) of the Dutch population experiences foot problems that can be treated by a pedorthist, i.e., an orthopedic shoe engineer. Approximately 20% of the foot problems can be treated with orthoses that support the natural shape of the foot. An orthosis is an externally applied device used to straighten or align parts of the neuromuscular system or the skeleton. A smaller group of people with foot problems require complex orthopedic aids, such as ankle-foot orthoses or custom-made shoes. Especially in this group, a careful fitting process is required to avoid injuries, which may lead to complications and in the worst case to amputation.

The main goal of the Smartscan project is to develop a high-tech instrument to renew the fitting/casting process to overcome shortcomings of the current method, such as risk of injuries or fittings that depend on the medicating person. With this new method, the plaster cast of the foot is replaced by a digital representation of the geometry of the foot. Based on this digital twin of the foot, it becomes possible to perform peer consultation. In addition, it provides the additional option to compare foot models digitally and remotely.

To create such a digital twin of the foot, an instrumented glove equipped with a position sensor on each fingertip and pressure sensors in the palm of the hand will be used. The benefit of an instrumented glove is that it is a minor change from the current manual approach, where latex gloves are used by the orthopaedic expert to diagnose the foot problem. The pressure sensors in the palm area of the glove measure the pressure applied by the orthopaedic expert during the casting and correction phase. The acquired pressures will give insight in the amount of occurring pressure points that can influence the desired corrections. The position and pressure data can subsequently be used to develop 3D digital computer-aided design (CAD) models that can be further edited for manufacturing. A digital representation of the geometry of the foot together with the applied pressures will assist the orthopaedic

expert in finding the optimal solution.

The main research question is how to clean and control the flow of data from sensors to CAD in real time in an efficient and effective manner, as real time processing would provide the operator of the glove direct feedback.

In the following we report on several approaches to obtain a digital foot model from sensor data. The methods range from using available meshing tools, such as meshlab, a signed distance function approach, and a linear regression model. We show that all approaches yield usable models, while the latter two models can be easily used for further development.

3.2 Procedure

According to NDI[10], i.e., the manufacturer of the sensors, electromagnetic tracking works along the following steps.

1. The transmitter emits a low-intensity, varying electromagnetic field that establishes the measurement volume.

2. Small currents are induced inside the sensors when they enter the EM field.

3. These currents are relayed to the sensor interface unit, where they are amplified and digitized as signals.

4. The signals are transmitted to the system control unit, which calculates each sensor's position and orientation as a transformation.

The result of this procedure is a data set of the form as illustrated on the left in Figure 3.1. This particular dataset constructed contains $n = 3\,963$ samples in time, measured at a frequency of 50Hz. Each sample consists of 9 observed parameters for the three sensors attached to three fingers. The first parameter is time. Next, there are the recorded x, y and z coordinates of each of the sensors, and the orientation of the sensors measured with respect to the reference sensor placed on the foot,

[10] `https://www.ndigital.com/technology/em-overview/`

in the Euler angles α, β, γ. A scalar quality measure is included that indicates the amount of metal interference present in the observations for each finger. Finally, a label is included that should indicate the state of the examination: 'off foot' (0), 'on foot' (1), or 'manipulation' (2). The latter has not been used since only 0-values are reported. The data provide a rough outline of a foot but also contain the movement of the glove towards and away from the foot. To obtain a better estimate of the foot's surface at subsequent steps, we filter this data by removing the initial and final movement of the glove. On the right in Figure 3.1 a filtering procedure has been applied, which has removed the data points indicated in red. The filtering procedure is explained in more detail in Cazacu et al. (2021, Section 2).

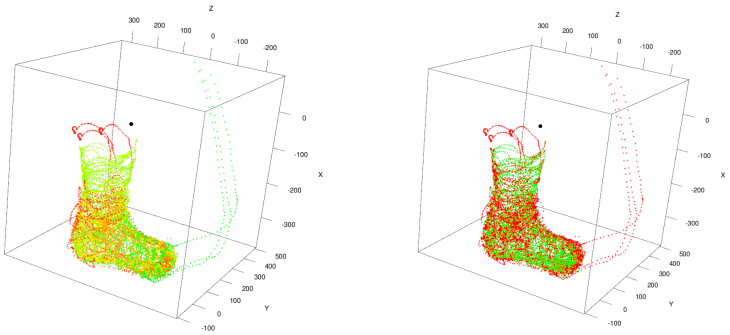

Figure 3.1: Time series position data of a calibration phase of the glove with three location sensors. The black dot indicates the origin. *Left:* time is indicated by color, from green at the initial time to red at the terminal time. *Right:* removed point in pre-applied filtering procedure are indicated in red, retained points in green.

After having established a reference data set by means of the calibration phase, the practitioner performs the diagnostics required to gain the knowledge required to construct a correcting cast. This would be for instance pressure data that would then be projected on a surface created out of the point cloud date from the calibration phase. The calibration data together with the adjustment data should be used to

generate a CAD model for a correcting cast.

3.3 Methods and results

The filtered point cloud can be associated with a surface in a number of meaningful ways.

3.3.1 Meshlab

Meshlab[11] is open source software that takes as input a point cloud and outputs a meshed surface. A guideline how to create a surface mesh from point cloud data can be found here[12]. The meshed surface produced in this way by MeshLab associated to the point cloud data of the foot is shown in Figure 3.2.

Since MeshLab is a ready-to-use package, one can quickly produce surface meshes from the given point cloud data. MeshLab can export such meshes in different formats, such as STL, which can then be used for 3D printing. While the availability of different meshing techniques is quite powerful, a drawback is that several parameters can be tuned, which also needs manual interaction by an operator. Moreover, post-processing the obtained mesh needs further tools. Also, the exploitation of the available time-data seems not possible.

[11] `https://www.meshlab.net`

[12] `http://fabacademy.org/2019/docs/FabAcademy-Tutorials/week05_3dscanning_and_printing/point_cloud_mesh.html`

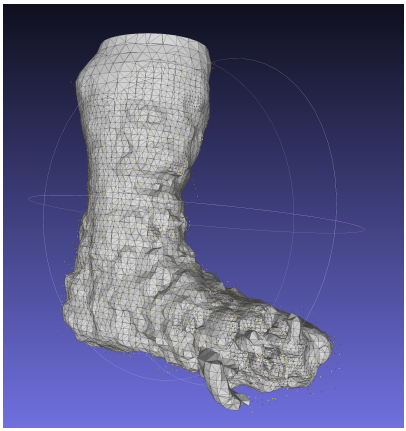

Figure 3.2: A MeshLab mesh from pre-filtered point cloud data.

3.3.2 Signed distance function

As uniform function approximators, deep neural networks can learn a wide class of functions up to arbitrary accuracy under some smoothness conditions. We propose to represent the volume V bounded by the manifold Ω by the zero level-set of its signed distance function (SDF) with arbitrary distance measure d

$$\phi_t(p)\colon p \mapsto \begin{cases} -d(\Omega(t), p) & p \in V(t), \\ d(\Omega(t), p) & p \notin V(t), \end{cases}$$

where $\Omega(t) = \{p \mid \phi_t(p) = 0\}$. We optimize a multi-layer perceptron neural network that takes as input the x, y and z coordinates of a point p, and as output provides its SDF value $d(\Omega, p)$. In machine learning research, using a neural network to represent a shape or function implicitly is known as an implicit neural representation (INR). For downstream applications, ϕ_t can be queried at any point p in V to find the surface representing the polygonal mesh using the marching cubes algorithm (see Figure 3.3).

Any suitable deep neural network can be trained to become such a signed distance function by minimising a certain objective over the

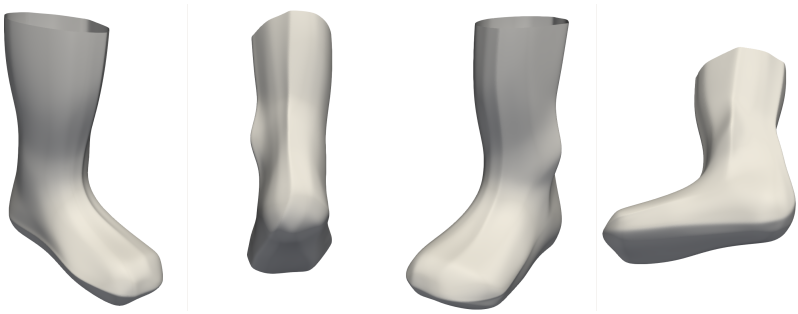

Figure 3.3: Once a SDF is available it can be queried on a voxel grid to recover an arbitrary amount of points on the manifold $p \in \Omega(t)$ at $\phi_t(p) = 0$. Marching cubes algorithm then yields a polygonal surface mesh which is depicted here.

space W of its trainable parameters. To construct a suitable optimisation objective, we sample points $q \in \mathrm{B}_r(p)$ which lie beyond the manifold $p \in \Omega$:

$$\min_{w \in W} \quad \underbrace{\log(1 + \phi_{t,w}(p))^2}_{\text{manifold loss}} + \lambda \underbrace{(\|\nabla_q \phi_{t,w}(q)\| - 1)^2}_{\text{eikonal term}} \qquad p \in \Omega(t), \quad q \in \mathrm{B}_r(p)$$

The manifold loss (compare Figure 3.4) acts as regularisation to deal with noisy measurement data by penalising negative SDF values, following the rationale that during tracing of the foot no measurement can lie inside it. This approach is similar to a (deep-learning-free) regression model without shape prior which fits the surface implicitly and by a sophisticated optimisation objective. It was inspired by Gropp et al. (2020) and uses their open-source code.

Once a SDF $\phi_t(p)$ is computed, normal vectors on the manifold are trivially available as gradient $\nabla_p \phi_t(p)$ for $p \in \Omega(t)$ on the surface. Our implementation is available on GitHub[13].

[13] `https://github.com/sukjulian/swi-fontys-smartscan`

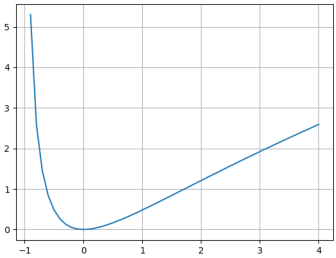

Figure 3.4: Manifold loss $f(x) = \log(1 + x)^2$ to deal with noisy point cloud data conditions the SDF to regard only the innermost points.

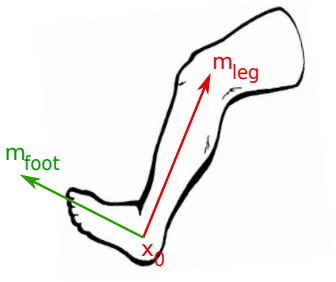

Figure 3.5: A center-line is drawn through the lower leg, and another one through the foot.

3.3.3 Linear regression model

A simple model of a person's lower leg and foot

Figure 3.5 shows a picture of a person's lower leg and foot. We will use this to make a parametric description of the leg in the case where the leg does not move. We shall discuss the situation of a moving foot later.

Center line

A center line can be computed directly from the data set to provide a basis for a linear regression model. This can be achieved by means of univariate splines for each coordinate direction. This method produces a single center line through the point cloud. The spline approach can be applied to any point cloud data set, but is sensitive to the quality of the data. It can be seen in Figure 3.6, in the picture on the right, that one side of the toe area has many more data entries than the other side. This causes the algorithm to produce a center line that is biased towards the number of measurements in a certain area. This issue can be addressed in several ways. A first option is to make sure that the data is evenly distributed, which could be difficult in practice. A second option is to increase the weight of undersampled areas, but this also amplifies outliers. A third option is to apply an additional filter to make the data set more suitable for spline methods.

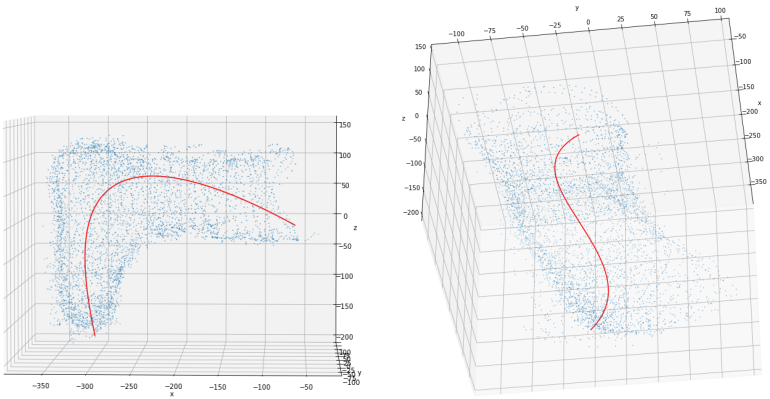

Figure 3.6: Spline fitted through the point cloud data set. The side view in the left figure shows a good fit to the data set, but the figure on the right indicates that the point cloud is undersampled on the left-hand side of the foot.

Alternatively, one can use principal component analysis to find center lines. In Figure 3.1 we can see that the orthopedic expert traces

out the foot as time progresses. The idea is to regard the point cloud as time series data, and consider time windows in which the fingers of the expert have traced out a part of the foot. We choose a window of 100 data points, or two seconds, and consider only the filtered data set. If we then consider the average position of the data points in each window, we obtain a cloud of averages which lies inside the foot (see Figure 3.7 *top left*). This moving average lies somewhat around a center line through the foot, which is revealed by a principle component analysis. In particular, the first principle component contains the direction of the most amount of variance of the moving average, and is shown as the yellow line in Figure 3.7 *top right*. This trend allows us to split the leg from the foot, by considering the plane orthogonal to this line at the center. We then repeat the procedure of averaging time windows and principle component analysis for the foot part and leg part separately, to get two candidate center lines (Figure 3.7 *bottom left*). In order to have them intersect, these can be slightly adjusted, producing the desired center lines (Figure 3.7 *bottom right*). This method is fully data driven, since the time window (which we took as 100 data points) can be chosen depending on the number of data points available.

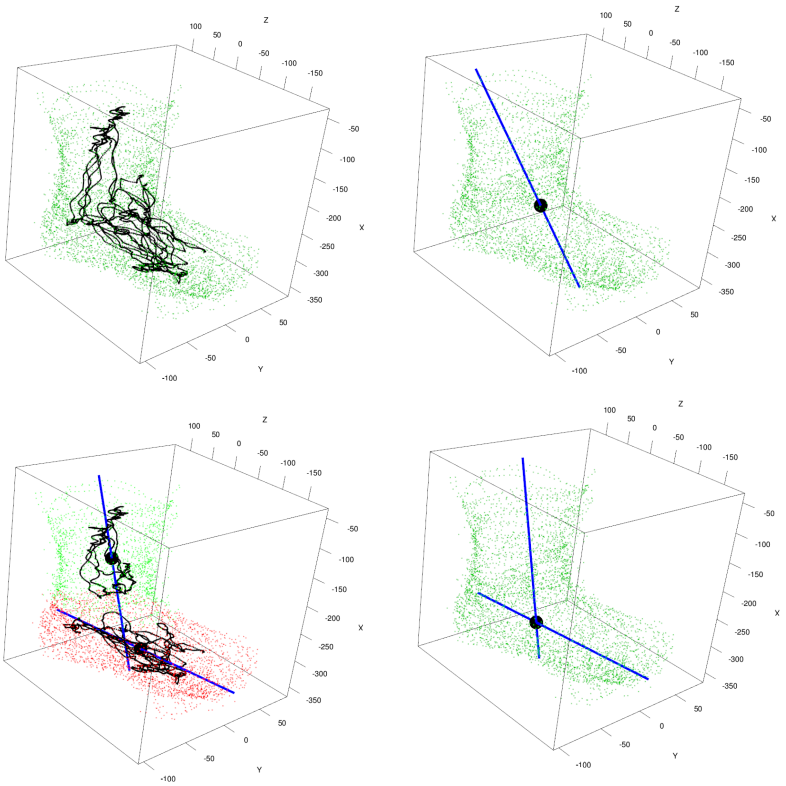

Figure 3.7: Point cloud of filtered data set. *Top left:* moving average in the inside of the foot is shown. *Top right:* line along the first principle component of the moving average is shown, with its center as black dot. *Bottom left:* point cloud is split into foot part and leg part and for each part the line along the first principle component of the corresponding moving average is shown, with centers as black dots. *Bottom right:* The obtained center lines, which intersect at the black dot.

To overcome potential errors in the data-driven centerline extraction approach, a center line can also be introduced manually. The linear regression model that we will now describe is based on two center lines.

The manual procedure uses the point $\vec{x}_0$, which is the location of the ankle joint, and the unit vectors $\vec{m}_{leg}$ and $\vec{m}_{foot}$, which indicate the direction of the leg and foot, respectively. This point and these vectors are used to define center lines, one through the leg, and one through the foot. This is illustrated in Figure 3.5 below.

We introduce the additional unit vectors $\vec{a}_{leg}$ and $\vec{b}_{leg}$, which are both perpendicular to each other and to $\vec{m}_{leg}$ for the leg. Similarly, we introduce $\vec{a}_{foot}$ and $\vec{b}_{foot}$ for the foot. Using this data, we can describe the foot and the leg as two separate shapes, each with the following parametrisation:

$$\vec{x}(l,\phi) = \vec{x}_0 + l\vec{m} + (\cos(\phi)\vec{a} + \sin(\phi)\vec{b})r(l,\phi), \tag{3.1}$$

where the symbols have the following meaning

- The origin $\vec{x}_0$,
- The leg's orientation: $\vec{m}_{leg}$, $\vec{a}_{leg}$, $(\vec{b}_{leg} = \vec{a}_{leg} \times \vec{m}_{leg})$,
- The leg's radius function: $r_{leg}(l,\phi)$
- The foot's orientation: $\vec{m}_{foot}$, $\vec{a}_{foot}$, $(\vec{b}_{foot} = \vec{a}_{foot} \times \vec{m}_{foot})$,
- The foot's radius function: $r_{foot}(l,\phi)$

Now we have two separate problems, which can be coupled because they share the point $\vec{x}_0$.

Finding the radius function

For a given orientation, the radius function can be determined. The samples are converted from $\vec{x} = (x,y,z)$ to the parametrization (l,ϕ,r):

$$\begin{aligned} l &= (\vec{x} - \vec{x}_0) \cdot \vec{m}, \\ \phi &= \text{atan2}((\vec{x} - \vec{x}_0)\vec{b}, (\vec{x} - \vec{x}_0)\vec{a}), \\ r &= |\vec{x} - \vec{x}_0 - l\vec{m}|. \end{aligned} \tag{3.2}$$

For the measurement $\vec{x}_1, ..., \vec{x}_N$ this will give parameter pairs $(l_1,\phi_1), ..., (l_N,\phi_N)$ and measured radii $(r_1, ..., r_N)$.

Next, we introduce a sampling grid for r:

$$\mathtt{r}_{i,j} \approx r(i * \delta l, j * \delta\phi) \tag{3.3}$$

And we determine interpolation weights for all available (l, ϕ) couples:

$$\begin{aligned} r(l,\phi) &= r_{\mathtt{floor}(l/\delta l),\mathtt{floor}(\phi/\delta\phi)}(1-\mathtt{mod}(l,\delta l))(1-\mathtt{mod}(\phi,\delta\phi)) \\ &+ r_{\mathtt{floor}(l/\delta l),\mathtt{ceil}(\phi/\delta\phi)}(1-\mathtt{mod}(l,\delta l))\mathtt{mod}(\phi,\delta\phi) \\ &+ r_{\mathtt{ceil}(l/\delta l),\mathtt{floor}(\phi/\delta\phi)}\mathtt{mod}(l,\delta l)(1-\mathtt{mod}(\phi,\delta\phi)) \\ &+ r_{\mathtt{ceil}(l/\delta l),\mathtt{ceil}(\phi/\delta\phi)}\mathtt{mod}(l,\delta l)\mathtt{mod}(\phi,\delta\phi) \end{aligned} \tag{3.4}$$

Doing this for the available measurements (l_n, ϕ_n), we get an overdetermined system

$$\begin{pmatrix} r_1 \\ \vdots \\ r_N \end{pmatrix} \approx \mathsf{Interp}\ \mathtt{r}, \tag{3.5}$$

for which $\mathtt{r}$ is the least squares solution. We get the result shown in Figure 3.8.

3.4 Filtered inputs

As mentioned before, the input was given to us by Fontys in two forms: there were *raw* and *filtered* data. The reconstruction process itself, however, also goes through some iterative filtering: points which are too far from the reconstructed surface are discarded.

The results are shown in Figure 3.9. When using the filtered data, the algorithm's own filtering has to be disabled, to avoid removing essential parts of the input. With these modifications in the treatment, the results based on the filtered data seem to be slightly smoother. This means that the filtering technique that was employed by Fontys did a slightly better job of removing unreliable data.

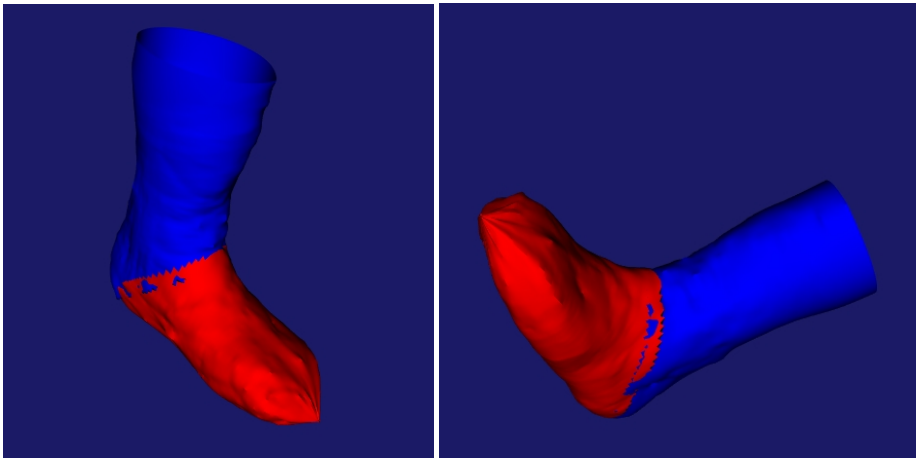

Figure 3.8: 3D pictures of the reconstructed foot using the regression approach.

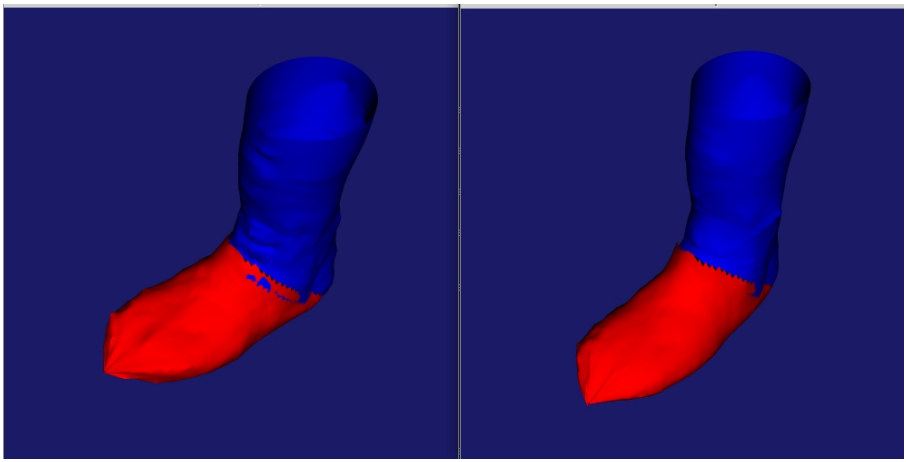

Figure 3.9: Left: reconstructed foot made from *raw* data. Right: reconstructed foot from filtered data

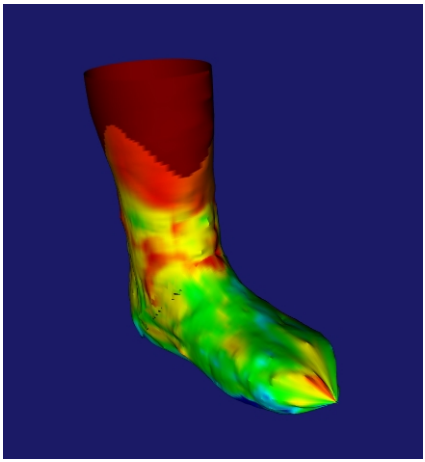

Figure 3.10: A scalar field was mapped onto the surface, and is displayed here in false-color.

3.5 Mapping a scalar field on the surface

Eventually, a goal will be to display pressure fields on the 3D image of the foot. We do not have pressure data, but for now we can use the 'quality' data that were supplied in the input.

These values (like the pressure values) are time-dependent, and we visualize only the contributions of a chosen time interval.

To map the scalar field from the data points to the surface, we first create an array that contains the scalar values in the selected data points (the ones in the chosen time frame), and zeros elsewhere. Next, we create a similar array, containing ones in the selected time frame and zero outside.

Next, we apply the same interpolation to the two arrays as we did to the input radii r. Finally, the interpolated scalar values are scaled with the interpolated ones to get the values which we display, like in Figure 3.10.

3.6 Conclusion and outlook

Figure 3.11: **Side-by-side comparison** of reconstructions from the linear regression model **(left)** and the implicit neural representation **(right)**. The resolution of the implicit neural representation reconstruction is chosen extremely fine at the cost of considerable computational overhead.

Figure 3.11 shows a comparison of the reconstructed foot model using the linear regression model and the neural field representation, respectively. We observe that both models represent the overall foot quite well. Both models depend on certain parameters, which influence the smoothness of the reconstructed foot, and balance smoothness against data noise. Both computational models are available in GitHub[14] and can be used for further developments.

A suggested procedure for producing more reliable point clouds using the glove is by introducing a number of calibration stages. The calibration stages depend on the glove, for instance, whether pressure sensors are included or not. If the glove has a combination of location and pressure sensors, the practitioner should touch the pressure sensor with each of the location sensors to determine the distance between the location and the pressure sensors. If the glove touches the foot, a certain minimal pressure should be applied to distinguish sensor data *touching* the foot and *outliers*.

[14] `https://github.com/sukjulian/swi-fontys-smartscan`

In case the glove only consists of location sensors, the practitioner shall first create a virtual cast of the foot in its current position. For that a prescribed number of steps may be performed to allow for simple preprocessing of the data, i.e., filtering outliers and creating easily center lines for the linear regression model.

References

Cazacu, Eduard et al. (2021). "A Position Sensing Glove to Aid Ankle-Foot Orthosis Diagnosis and Treatment". In: *Sensors* 21.19, p. 6631.

Gropp, Amos et al. (2020). "Implicit Geometric Regularization for Learning Shapes". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event.* Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 3789–3799.