# Automatic Measurement of Cartilage Thickness in the Human Knee

Simon Westfechtel, B.Sc.

November 23, 2021

## 1 Introduction

This paper seeks to discuss different methods to automate the measurement of cartilage thickness in the human knee using MRI scans. The methods are based on previous work by Wolfgang Wirth and Felix Eckstein. [WE08]
All computational methods make use of segmented MRI scans of the human knee, from the OAI dataset, and are written in Python. More precisely, there were two distinct sets of segmentations used: One set of manually segmented images, containing 507 samples, and one set of automatically segmented images, containing 24,783 samples. The set of automatic segmentations contained samples from the entire OAI dataset, i.e. over all time periods. Manual segmentations are stored as MHD and automatic segmentations as NIFTI files, respectively, which can be read and converted into Numpy arrays using the SimpleITK library. These arrays map each point in the three-dimensional space of the scan to an integer encoding of the segmentation, meaning for example points belonging to the femoral cartilage are assigned a value of 3. This makes isolating and extracting the cartilage volumes straightforward. Four different methods have been developed to determine mean cartilage thickness, plus other statistical measures. Thickness was measured for different subregions of the cartilage plate, which were determined as referenced in [WE08]. For a more detailed description of the procedure, refer to appendix A.

## 2 Mean cartilage thickness using meshes

This is a three-dimensional approach using meshes and normal vectors to determine the mean thickness of a cartilage volume. The main idea is building an upper and a lower mesh, and calculating the average distance between the two, by ray tracing along the normal vectors of the lower mesh against the upper mesh. Other methods, like a K-D-tree nearest neighbour search, are also possible, but have not been implemented. The mesh building is relatively trivial for the tibial cartilage due to its physical shape. Some formal definitions are necessary:

**Definition 2.1 (Point Cloud)** *Each cartilage is represented by a set of vectors $V$. The vectors $v \in V$ are defined as a triplet $(x, y, z)$, with $x$, $y$ and $z$ denoting a unique position in the three-dimensional Euclidean space, such that $\forall u, v \in V : u \neq v$.*

**Definition 2.2 (Mesh)** *A mesh is a Delaunay-triangulated surface $DT(P) = (V, E, N)$ of a point cloud $P$, such that no point $p \in P$ lies inside the circumcircle of any triangle in $DT(P)$. [Wik21a] $DT(P)$ consists of a set of vertices $V$, a set of edges $E$ and a set of surface normals $N$, where $V \equiv P$, $|V| = |N|$.*

The point cloud representation of the tibial cartilage is obtained by filtering the aforementioned array representation of the MRI scan for the integer encoding specific to the tibial cartilage. The vectors forming the upper and lower surface of the cartilage, respectively, are defined as follows: Let $P$ be the point cloud

**Algorithm 1** Point Cloud Representation of the Cartilage

```
1: procedure CartilageExtraction(A,C)
2:     $A \leftarrow$ image array
3:     $C \leftarrow$ color code $\in \{3, 4\}$
4:     $P \leftarrow \{\emptyset\}$
5:     for each $(x, y, z) \in A$ do
6:         if $A[(x, y, z)] = C$ then
7:             $P \leftarrow P \cup (x, y, z)$
8:
       return $P$
```

representing the tibial cartilage. Furthermore, let $P_u$, $P_l$ be the point clouds representing the upper and lower surface of the tibial cartilage. Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}; X \times Y \mapsto \mathbf{Z}$ be a function where $X$ and $Y$ are scalars representing x and y values in the three-dimensional Euclidean space, and $\mathbf{Z}$ be the vector of all $z$ values in the Euclidean space for the position $(x, y)$. Then $P_u$ and $P_l$ are defined as:

$$P_u := \{ \ (x, y, z)_i \in P \mid z = max(f(x, y)) \ \} \tag{1}$$

$$P_l := \{ \ (x, y, z)_i \in P \mid z = min(f(x, y)) \ \} \tag{2}$$

This essentially means grouping $P$ by $x$ and $y$ coordinates and for each unique coordinate pair $(x_i, y_i)$ in the Euclidean space, choosing the vector with the maximum $z$ coordinate for the upper point cloud $P_u$, and the vector with the minimum $z$ coordinate for the lower point cloud $P_l$. The result is two point clouds consisting of vectors $(x, y, z)$, as seen in figure 1 (red points make up the upper cloud, green the lower). These are then converted into polygon meshes making use of the Delaunay triangulation. For the lower surface mesh, its surface normals are calculated. A surface normal vector is defined as the vector perpendicular to the tangent plane of the surface at a point $P$ [Wik21b]. Finally, mean cartilage thickness, i.e. the mean distance between the two meshes, is calculated by tracing along the normal vectors of the lower mesh against the upper mesh. To this end, each surface normal is iteratively extended until it hits the upper mesh, and the Euclidean norm of the vector, i.e. its length, and therefore the distance between the two meshes at a specific point, is computed. This is done for all surface normals, and the average, standard deviation of the calculated distances as well as the mean of the minimum (maximum) 1% of measurements. Due to the orthogonal nature of the normals, these traces can take into account the topography of the cartilage, allowing for more accurate distance measurements than going in the same static direction for every point $p \in P_l$. For other methods not making use of surface normals, like the previously mentioned nearest neighbour search, the Delaunay triangulation may be omitted.

Mesh building for the femoral cartilage is not as trivial. Due to its convex

**Algorithm 2** Upper and Lower Point Clouds (Tibia)

```
1: procedure PointClouds(P)
2:     $P_u \leftarrow \max_z(P.groupby((x, y)))$
3:     $P_l \leftarrow \min_z(P.groupby((x, y)))$
4:     return $P_u, P_l$
```

**Algorithm 3** Mean Cartilage Thickness (Tibia)

```
1: procedure TibialThickness($P_u$, $P_l$)
2:     $D \leftarrow \{\emptyset\}$
3:     $M_u \leftarrow delaunay(P_u)$
4:     $M_l \leftarrow delaunay(P_l)$
5:     $N_l \leftarrow normals(M_l)$
6:     for each $n \in N_u$ do
7:         $v \leftarrow raytrace(n, M_u)$
8:         $d \leftarrow ||v||$
9:         $D \leftarrow D \cup d$
10:    $\overline{D} \leftarrow \frac{\sum_{i=0} d_i \in D}{|D|}$
11:    return $\overline{D}$
```

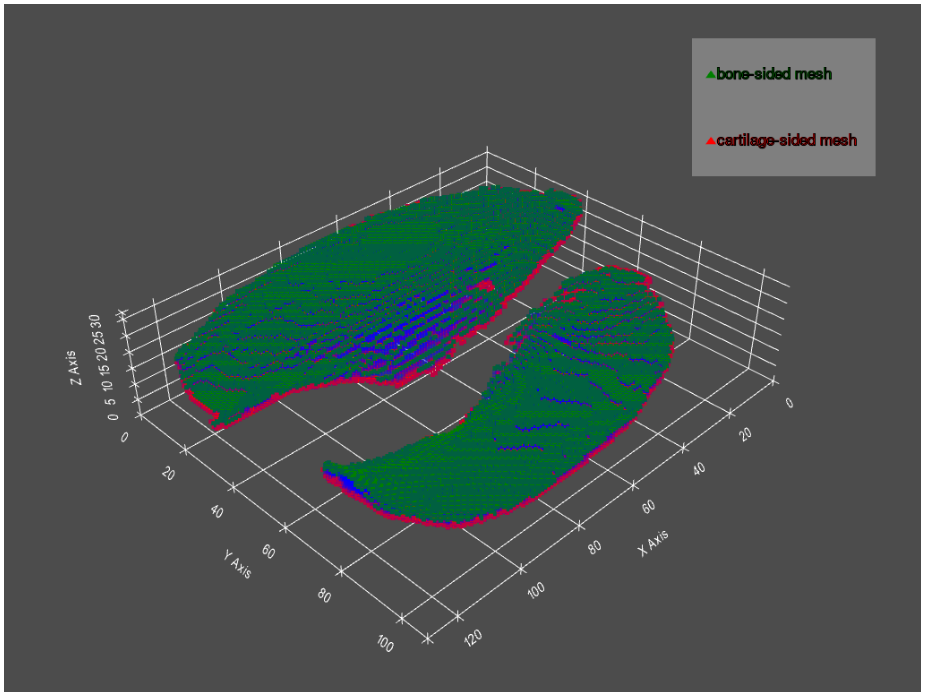

Figure 1: Point clouds of the tibial cartilage. The figure is flipped, i.e. the camera is focused on the bone side of the cartilage.

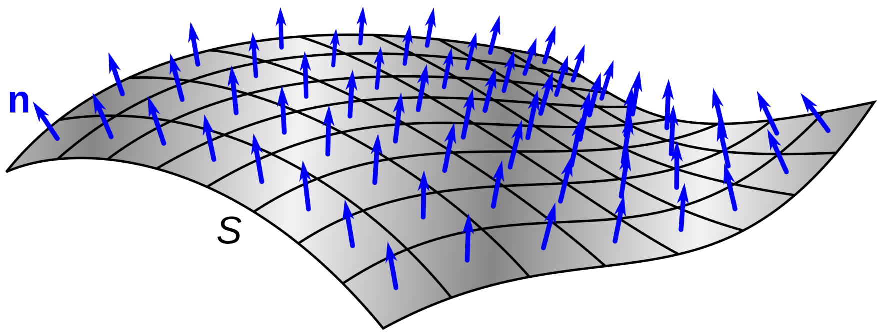

Figure 2: A curved surface showing the unit normal vectors (blue arrows) to the surface. [Che19]

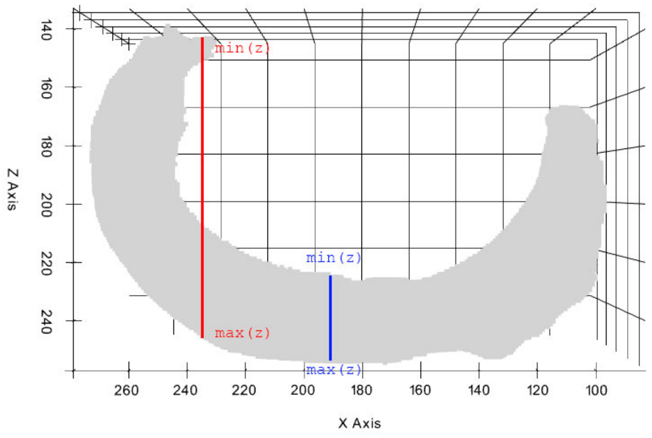

Figure 3: Coronal view on the femoral cartilage. The red bar illustrates the problem of extracting the inner and outer outline by $z$ value: In the posterior part of the cartilage, points that, in reality belong to the outer outline, are assigned to the inner outline, and vice versa.

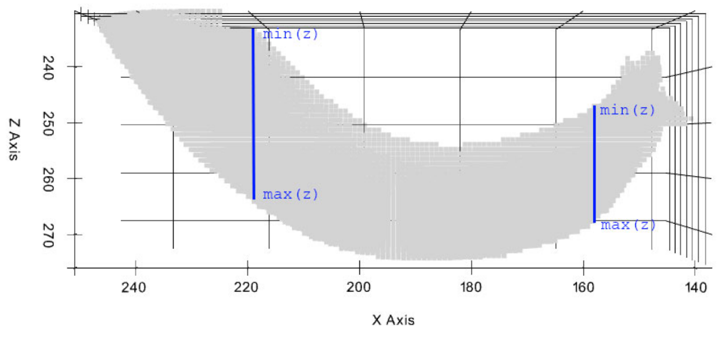

Figure 4: Coronal view on the posterior part of the femoral cartilage. The part has been rotated to allow outline extraction by z value.

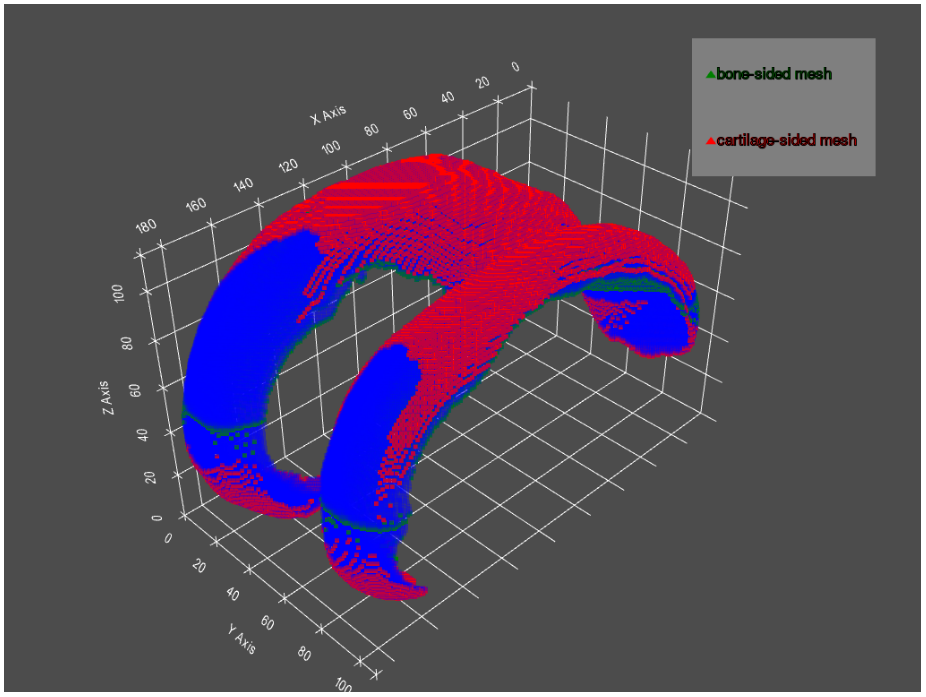

Figure 5: Point clouds of the femoral cartilage.

shape, obtaining the upper and lower point clouds $P_u$ and $P_l$ can not be achieved by simply always taking the maximum (minimum) $z$ value for each coordinate pair $(x, y)$. For an illustration of the problem, refer to figure 3. Using the same approach as for the tibia results in point clouds where some areas are left bare, as illustrated in figure 5. This issue can be circumvented by splitting the cartilage into parts, and rotating problematic sections in such a way that it is once again possible to choose points according to the $z$ coordinate, as illustrated in figure. For a detailed description of how this splitting is achieved, refer to appendix B. For each part, the outlines are extracted and meshes are calculated analogous to the tibia, and the meshes' surface normals are used to calculate mean thickness for each part of the femoral cartilage.

## 3 Mean cartilage thickness using ray tracing from a central point

This is a three-dimensional approach using ray tracing along normal vectors to determine the mean thickness of a cartilage volume. This is another proposed solution to the previously discussed issue with the convex shape of the femoral cartilage. Instead of fitting meshes to the cartilage, a sphere is constructed around a central point above or below the cartilage A sphere is defined as follows:

**Definition 3.1 (sphere)** *A sphere $S = (c, r, V, E, N)$ is a surface mesh, where $c$ is the central point with coordinates $(x, y, z)$, $r$ is the radius, $V$ is a set of vertices, $E$ is a set of edges and $N$ is a set of surface normals, such that $|V| = |N|$.*

Let $P$ be the point cloud representing a cartilage $C$. Then

$$\begin{aligned} c &= (x, y, z); \\ c_x &= \min_x P + (\max_x P - \min_x P) \div 4 \\ c_y &= \min_y P + (\max_y P - \min_y P) \div 2 \\ c_z &= \min_z P + (\max_z P - \min_z P) \div 2 \end{aligned}$$

for the central weight-bearing zones,

$$\begin{aligned} c_x &= \min_x P + (\max_x P - \min_x P) \div 4 \\ c_y &= \min_y P + (\max_y P - \min_y P) \div 2 \\ c_z &= \min_z P + (\max_z P - \min_z P) \div 2 \end{aligned}$$

for the medial and lateral posterior femoral cartilage,

$$\begin{aligned} c_x &= \min_x P + ((\max_x P - \min_x P) \div 4) \cdot 3 \\ c_y &= \min_y P + (\max_y P - \min_y P) \div 2 \\ c_z &= \min_z P + (\max_z P - \min_z P) \div 4 \end{aligned}$$

for the medial and lateral anterior femoral cartilage, and

$$\begin{aligned} c_x &= \min_x P + (\max_x P - \min_x P) \div 2 \\ c_y &= \min_y P + (\max_y P - \min_y P) \div 2 \\ c_z &= \max_z P \ \cdot \ 1.25 \end{aligned}$$

for the lateral and medial tibial cartilage.

$|V|$ corresponds to the resolution of the sphere, which is a tuple $(\theta, \phi)$, where $\theta$ is the number of vertices in the horizontal (longitude) direction, and $\phi$ the number of vertices in the vertical (latitude) direction. $\theta$ should be equal to $\phi$ in order to obtain a real sphere, as opposed to an ellipsoid; in our case, $\theta = \phi = 60$, and therefore $lvertV| = 60 + 59 \cdot 58 = 3,482$. Given a point cloud $P$ and a sphere $S = (c, r, V, E, N)$, the thickness of the corresponding cartilage can be calculated with the following procedure: Each surface normal $n \in N$ is iteratively extended until it hits a point $p \in P$, or a maximum number of iterations is reached, in which case the normal is discarded. Once a point is hit, its position is saved as $p_1$, and $n$ is further extended until no more points are hit. For every iteration of this further extension, the hit point in that iteration is saved as $p_2$. At the end of the iteration, the Euclidean distance between the first hit $p_1$ and the last hit $p_2$ is calculated. This approach is one of the more computationally expensive ones, as for one, generally more than half of the surface normals will never make a hit, and tracing along these is a wasted effort, although this may be remedied somewhat by checking whether the search space, i.e. the point cloud, lies in the direction of the vector a priori. The other major reason is that the nature of ray tracing is inherently expensive.

**Algorithm 4** Ray Tracing from a Sphere

```
1: procedure RayTracing($P, S$)
2:     $D \leftarrow \{\emptyset\}$
3:     for each $n \in N$ do
4:         repeat
5:             $n \leftarrow n \cdot 2$
6:             if $n \in P$ then
7:                 $p_1 \leftarrow n$
8:                 break
9:         until iterations $> 100$
10:        repeat
11:            $n \leftarrow n \cdot 2$
12:            if $n \notin P$ then
13:                break
14:            $p_2 \leftarrow n$
15:        until iterations $> 100$
16:        $dist \leftarrow d(p_1, p_2)$
17:        $D \leftarrow D \cup dist$
18:
19:    $\overline{D} \leftarrow \frac{\sum_{i=0} d_i \in D}{|D|}$
20:    return $\overline{D}$
```

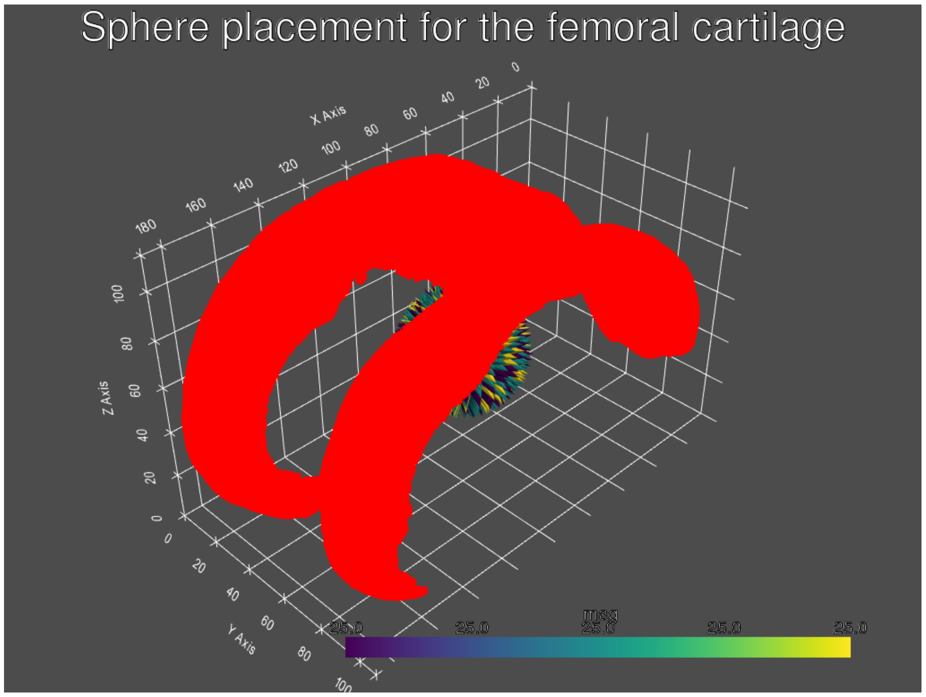

Figure 6: Ray tracing against the femoral cartilage using a central sphere

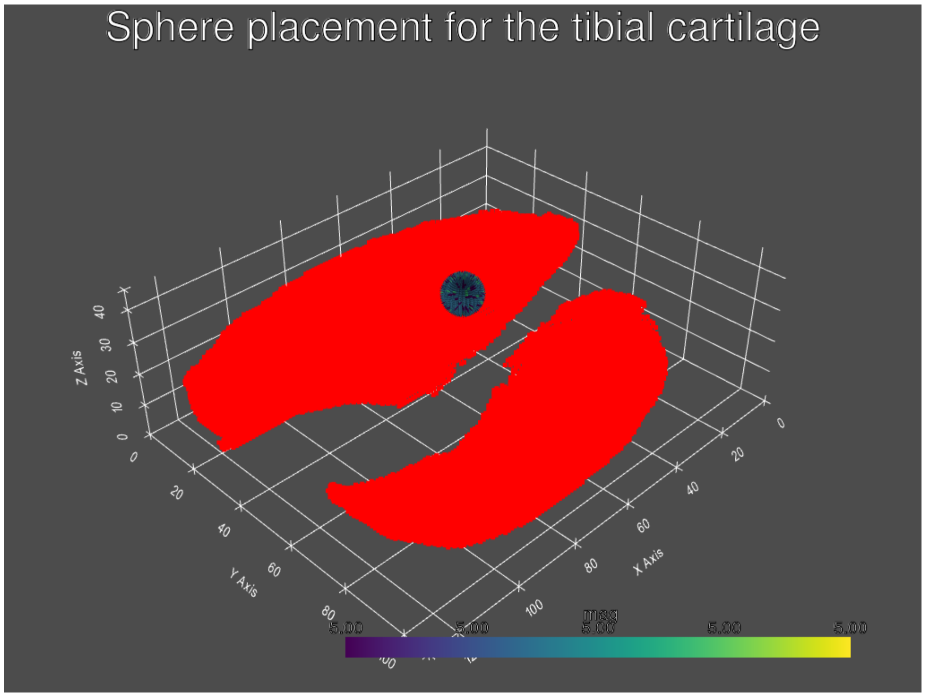

Figure 7: Ray tracing against the tibial cartilage using a central sphere

# 4 Mean cartilage thickness using two-dimensional function fitting

## 4.1 Determine thickness via function normals

This approach is settled in the two-dimensional rather than the three-dimensional Euclidean space. Instead of considering the MRI scan as a whole, and thus the entire cartilage, the idea here is to look at slices instead. A cartilage is divided into layers, and a polynomial function through the centre is obtained by a least-square fit. A number of normals can then be calculated, making use of the derivative of the function. Since the normal of a function $f$ at a point $x$ is perpendicular to the tangent of $f$ at $x$, the thickness of the cartilage at that point can accurately be determined by finding the outermost intersection points of the normal with the cartilage in either direction, and computing the Euclidean distance between the two intersection points. Do this for every slice and average over the individual measurements to obtain the mean thickness of the cartilage. The advantage of this approach is that the polynomial functions are generally easier to fit than the triangulated meshes discussed earlier, bar for some edge cases (e.g. the outermost slices of the cartilage, where data points can be very sparse and thus, getting a good fit is difficult if not impossible). On the other hand, some sections may be omitted from calculations, as can be seen in figure 8.

**Algorithm 5** Normals of a Polynomial Fit

```
procedure FunctionNormals(P)
    $D \leftarrow \{\emptyset\}$
    $L \leftarrow convertIntoLayers(P)$
    for each layer $l \in L$ do
        $f \leftarrow polyfit(l)$
        $\nabla f \leftarrow derivative(f)$
        $N \leftarrow normals(f)$
        for each $n \in N$ do
            $m \leftarrow slope(n)$
            $p_1 \leftarrow lastIntersection(n, m)$
            $p_2 \leftarrow lastIntersection(n, -m)$
            $dist \leftarrow d(p_1, p_2)$
            $D \leftarrow D \cup dist$

    $\overline{D} \leftarrow \frac{\sum_{i=0} d_i \in D}{|D|}$
    return $\overline{D}$
```

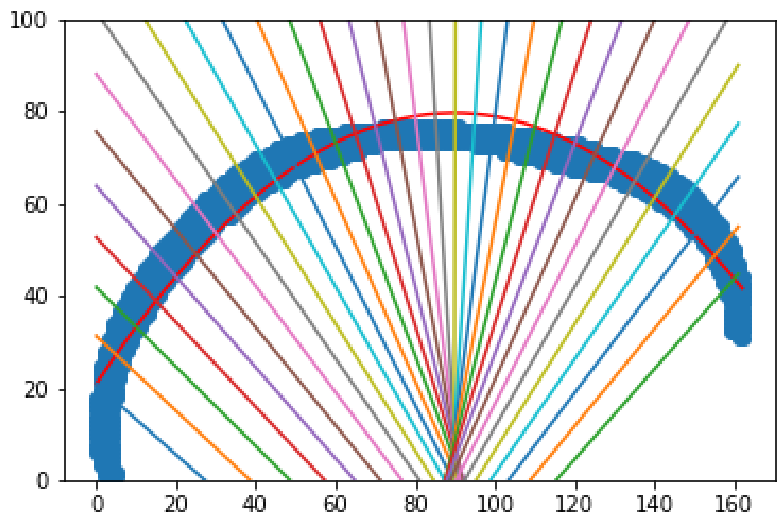

Figure 8: Normals along a two-dimensional least squares fit

## 4.2 Determine thickness via integration/function values

This approach is also settled in the two-dimensional Euclidean space, the difference to the previous being that there are two polynomial functions instead of one, which don't go through the middle of the cartilage but rather are fit along its outlines, as illustrated in figure 9. The distance between the functions, i.e. the thickness of the cartilage, can then easily be calculated, for example by integrating both functions and taking the difference ($\int f(x)dx - \int g(x)dx$), or calculating the difference between the function values for every $x$ value ($[\sum_{x_i=0}^{\max_x} f(x_i) - g(x_i)] \cdot \frac{1}{\max_x}$). Since no normals and therefore no intersection points have to be calculated, this approach is the computationally least expensive. The downside is that it can not take the topography of the cartilage into account, resulting in less accurate measurements especially for the anterior and posterior parts of the femoral cartilage. This can be somewhat remedied by making use of splines, i.e. splitting the cartilage into multiple parts in a similar fashion to the procedure used for the earlier discussed mesh building.

As previously mentioned, one issue with both of these approaches is that the function fitting may not work well for certain shapes, especially for layers where the number of data points is very sparse. Although most of the layers are going to have no more than two inflection points (layers of the femoral cartilage can generally be approximated by a second-order polynom, while the layers of the tibial cartilage tend to follow a third-order form), some edge cases, e.g. the outermost layers, might return a poor fit, or no fit at all. This could possibly be

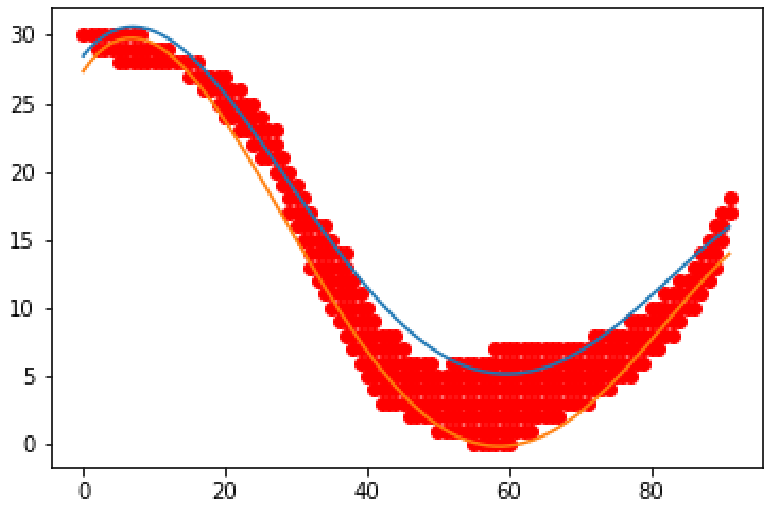

Figure 9: Two functions fit over the outlines of a tibial cartilage slice exposure

remedied by choosing a higher degree for the polynomial fit, but doing so may in turn lead to undesirable effects (overfitting, Runge's phenomenon, etc.).

## A Determination of the cartilage subregions

As proposed by Wirth & Eckstein, the two cartilage plates of the medial / lateral femoral condyle are divided into three subregions each (central, internal and external), while the two cartilage plates of the medial / lateral tibia are divided into five subregions each (central, internal, external, anterior, posterior). For nomenclature, refer to the table of acronyms below.

### A.1 Determination of the femoral subregions

For each plate of the femur, each subregion should encompass approximately 33% of the plate volume. This was achieved in practice by splitting the volume into three parts along the y-axis; two splitting points $x1, x2$ are chosen arbitrarily, such that $y_{min} < y1 < y2 < y_{max}, y1 = \frac{y_{max}-y_{min}}{3}, y2 = 2 \cdot y1$, where $y_{max}, y_{min}$ are the maximum / minimum y values, respectively. $y1$ and $y2$ are then iteratively moved along the y-axis until the constraint is satisfied, i.e. 33% of all data points lie left of $y1$, 33% lie between $y1$ and $y2$ and 33% lie right of $y2$.

### A.2 Determination of the tibial subregions

For each plate, an ellipse around the center of the plate, aka the central region, should encompass approximately 20% of the plate volume, and four triangles surrounding the ellipse should be of variable size. This was achieved in practice by first determining the center of gravity of the plate through K-Means clustering and constructing an ellipse around this point; a radius $r$ is chosen arbitrarily, and is lengthened iteratively until the constraint is satisfied, i.e. 20% of all data points lie within the ellipse. The four triangles surrounding the ellipse are determined by calculating the corners $a, b, c, d$ of the plate, and data points are assigned to subregions according to their position relative to the vectors $\vec{ac}$ and $\vec{db}$, which is calculated via cross product.

```
Procedure to determine femoral subregions
Given:
y_axis := range of x-axis
plate := data points (x, y, z) making up a cartilage
↪  plate

Procedure:
y_min, y_max := min(y_axis), max(y_axis)
y_range := y_max - y_min
y1 := y_range / 3
y2 := 2 * y1
first_third := empty set
second_third := empty set

```

```
while len(first_third) / len(plate) is not .33 do
        first_third := {d ∈ plate | d.y < y1}
        if len(first_third) > .33
                y1 := y1 - 1
        else
                y1 := y1 + 1

while len(second_plate) / len(plate) is not .33 do
        second_plate := {d ∈ plate | y1 < d.y < y2}
        if len(second_plate) > .33
                y2 := y2 - 1
        else
                y2 := y2 + 1
```

```
Procedure to assign a femoral point to a subregion
Given:
plate := data points (x, y, z) making up a cartilage
 ↪  plate
y1, y2 := split points

for point in plate do
        if point.y < y1
                point.region = external/internal #
                 ↪  depending on whether point lies in
                 ↪  left or right plate
        if y1 < point.y < y2
                point.region = central
        else
                point.region = external/internal #
                 ↪  depending on whether point lies in
                 ↪  left or right plate
```

```
Procedure to determine tibial subregions
Given:
plate := data points (x, y, z) making up a cartilage
 ↪  plate

Procedure:
r := 20
c := KMeans(plate)
points_in_ellipse := empty set

while len(points_in_ellipse) / len(plate) is not .2 do
        points_in_ellipse := {d ∈ plate | dist(d, c) < r}
        if len(points_in_ellipse) > .2
                r = r / 2
        else
                r = r + .5

x_min := {min(d.x) | d ∈ plate}
x_max := {max(d.x) | d ∈ plate}
y_min := {min(d.y) | d ∈ plate}
y_max := {max(d.y) | d ∈ plate}

a := (x_min, y_min)
b := (x_max, y_min)
c := (x_max, y_max)
d := (x_min, y_max)
```

```
Procedure to assign a tibial point to a subregion
Given:
plate := data points (x, y, z) making up a cartilage
↪  plate
a, b, c, d := plate corners
points_in_ellipse := set of points lying within the
↪  central ellipse

Procedure:
for point in plate do
        if point is in points_in_ellipse
                point.region := central

        ac := c - a
        db := b - d
        pc := c - point
        pb := b - point

        if ac × pc > 0
                if db × pc > 0
                        point.region := internal
                else
                        point.region := posterior
        else
                if db × pc > 0
                        point.region := anterior
                else
                        point.region := external
```

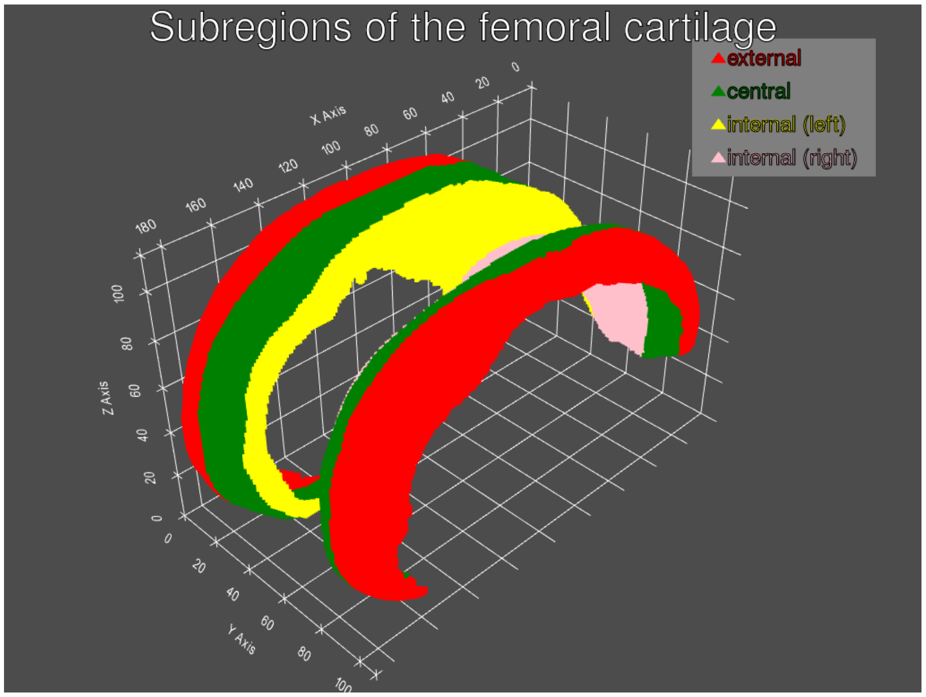

Figure 10: Subregions of the femoral cartilage

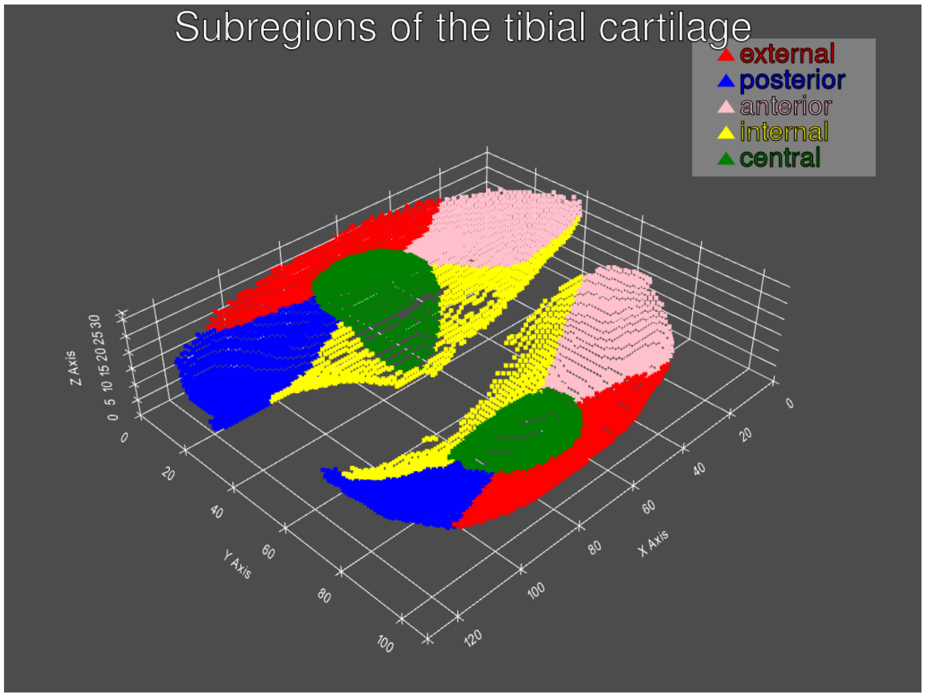

Figure 11: Subregions of the tibial cartilage

## B Splitting of femoral cartilage volume for meshes

As described above, constructing meshes for the femoral cartilage is not possible without some pre-processing. The convex parts of the volume on either side need to be split off and rotated by 90° so that minimum and maximum points can reliably be extracted without missing a significant amount. The sections are isolated as follows:
The cartilage volume is a point cloud made up of vectors $(x, y, z)$. Group the points by $(x, y)$ and for each of these combinations, calculate the range of $z$. Then calculate the mean or median of $z$ and take all combinations $(x, y)$ for which the range of z is less than the previously calculated mean/median, i.e. those points where the minimum and maximum z values are not too far apart. Note that the convex parts of the cartilage are characterized by a large ranges for z. Having now isolated the central part of the volume, it is a trivial task to extract the convex parts: We can calculate the minimum and maximum x value of the central vectors, and everything left of the minimum gets assigned to the left convex part, and everything right of the maximum gets assigned to the right convex part.
With the sections isolated, the problematic convex parts can then be rotated by 90°. Now, again for every combination $(x, y)$, the vector $(x, y, max(z))$ can be added to the upper mesh, and the vector $(x, y, min(z))$ to the lower ($min(z)/max(z)$ may be replaced by distance to a central point, depending on implementation).

```
Procedure to split a femoral cartilage volume into three
↪  parts
Given:
volume := vectors (x, y, z) making up the volume

Procedure:
z_range := volume.group_by((x, y)).z.max() -
↪  volume.group_by((x, y)).z.min()
z_med := median(z_range)
z_index := {d ∈ volume | d.z < z_med}
lower_bound := {min(d.x) | d ∈ z_index}
upper_bound := {max(d.x) | d ∈ z_index}
left_part := {d ∈ volume | d.x < lower_bound}
middle_part := {d ∈ volume | lower_bound < d.x <
↪  upper_bound}
right_part := {d ∈ volume | d.x > upper_bound}
left_part := rotate(90)
right_part := rotate(90)
```

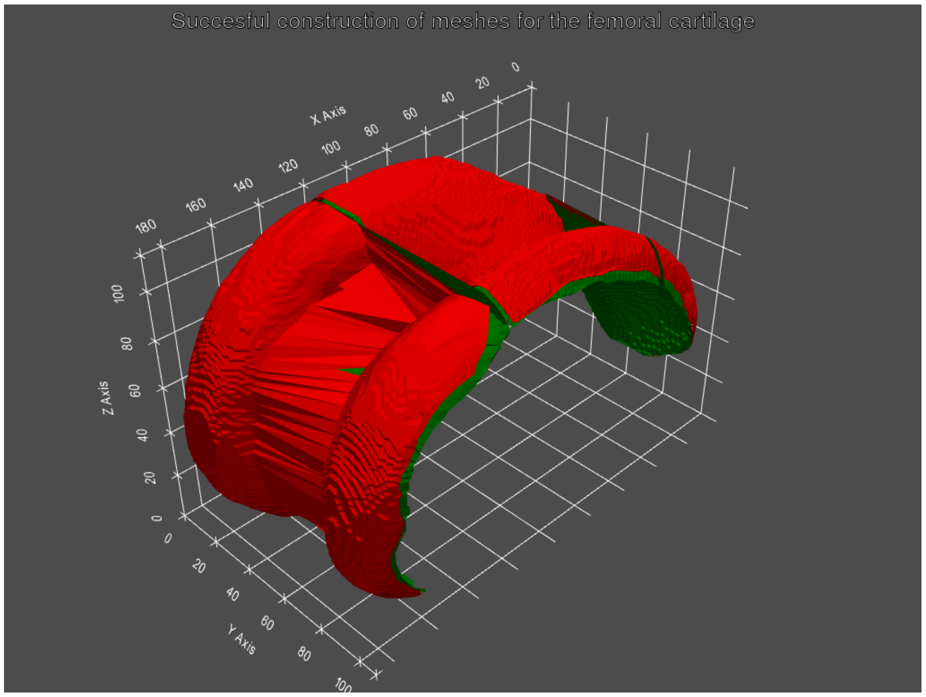

Figure 12: Successful construction of meshes for the femoral cartilage

## C Acronyms

**ecLF** Mean thickness of the external cartilage subregion of the central part of the lateral femoral condyle

**ccLF** Mean thickness of the central cartilage subregion of the central part of the lateral femoral condyle

**icLF** Mean thickness of the internal cartilage subregion of the central part of the lateral femoral condyle

**icMF** Mean thickness of the internal cartilage subregion of the central part of the medial femoral condyle

**ccMF** Mean thickness of the central cartilage subregion of the central part of the medial femoral condyle

**ecMF** Mean thickness of the external cartilage subregion of the central part of the medial femoral condyle

**aLF** Mean thickness of the anterior cartilage subregion of the lateral femoral condyle

**aMF** Mean thickness of the anterior cartilage subregion of the mdeial femoral condyle

**cLT** Mean thickness of the central cartilage subregion of the lateral tibia

**aLT** Mean thickness of the anterior cartilage subregion of the lateral tibia

**eLT** Mean thickness of the external cartilage subregion of the lateral tibia

**pLT** Mean thickness of the posterior cartilage subregion of the lateral tibia

**iLT** Mean thickness of the internal cartilage subregion of the lateral tibia

**cMT** Mean thickness of the central cartilage subregion of the medial tibia

**aMT** Mean thickness of the anterior cartilage subregion of the medial tibia

**eMT** Mean thickness of the external cartilage subregion of the medial tibia

**pMT** Mean thickness of the posterior cartilage subregion of the medial tibia

**iMT** Mean thickness of the internal cartilage subregion of the medial tibia

**x.aSD** Standard deviation of the thickness of the respective (x) cartilage subregion

**x.aMav** Mean value of the maximum 1% measurements of cartilage thickness of the respective (x) cartilage subregion

**x.aMiv** Mean value of the minimum 1% measurements of cartilage thickness of the respective (x) cartilage subregion

## References

[Che19] Chetvorno. Normal vectors on a curved surface. `https://en.wikipedia.org/wiki/File:Normal_vectors_on_a_curved_surface.svg`, 2019. [Online, accessed 14-November-2021].

[WE08] Wolfgang Wirth and Felix Eckstein. A technique for regional analysis of femorotibial cartilage thickness based on quantitative magnetic resonance imaging. *IEEE transactions on medical imaging*, 27(6):737–744, 2008.

[Wik21a] Wikipedia contributors. Delaunay triangulation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Delaunay_triangulation&oldid=1044718653`, 2021. [Online; accessed 14-November-2021].

[Wik21b] Wikipedia contributors. Normal (geometry) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Normal_(geometry)&oldid=1054386602`, 2021. [Online; accessed 15-November-2021].