UPPSALA
UNIVERSITET

F15003
Examensarbete 30 hp
Januari 2015

Kidney Dynamic Model Enrichment

Nils Olofsson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Kidney Dynamic Model Enrichment

Nils Olofsson

This thesis explores and explains a method using discrete curvature as a feature to find regions of vertices that can be classified as being likely to indicate the presence of an underlying tumor on a kidney surface mesh. Vertices are tagged based on curvature type and mathematical morphology is used to form regions on the mesh. The size and location of the tumor is approximated by fitting a sphere to this region. The method is intended to be employed in noninvasive radiotherapy with a dynamic soft tissue model. It could also provide an alternative to volumetric methods used to segment tumors. A validation is made using the images from which the kidney mesh was constructed, the tumor is visible as a comparison to the method result.

The dynamic kidney model is validated using the Hausdorff distance and it is explained how this can be computed in an effective way using bounding volume hierarchies.

Both the tumor finding method and the dynamic model show promising results since they lie within the limit used by practitioners during therapy.

Handledare: Marc Daniel
Ämnesgranskare: Anders Hast
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F15003

Sammanfattning

Njurtumörer som växer på eller nära ytan hos organet kommer orsaka en bula. Genom att analysera ytan hos njuren kan en sådan bula hittas och tumörens storlek och position kan sedan approximeras med en sfär. Detta kan användas vid strålbehandling av sådana tumörer.

I det här fallet är njuren en datormodel som ursprungligen skapats från skiktröntgenbilder. Datormodellens yta består består av ett triangulärt nät som analyseras genom att beräkna kurvaturen för varje knutpunkt i nätet. Kurvaturen hos en grupp av knutpunkter kan sedan avgöra om gruppen indikerar en underliggande tumör eller ej. När en sådan grupp har funnits kan en sfär som minimierar avståndet till dessa knutpunkter hittas.

För att validera datormodellen behövs ett slags mått på hur fel den är jämfört med valideringsdata. Här valideras modellen med det *s.k.* Hausdorff avståndet som säger att det felet är det största av alla lägsta avstånd från ena ytan till den andra. Modellen valideras mot två stycken 3D ytor skapade från skiktröntgenbilder.

Både metoden för att hitta tumörer med hjälp av kurvaturdata hos modellen samt modellen själv är lovande då båda har uppskattade fel som ligger innanför gränsen på 5mm som används vid behandling. Dock skulle det vara önskvärt att validera både metoden för att lokalisera tumörer och modellen med mer jämförelsedata.

Résumé

Des tumeurs rénale qui grandissent sur, ou à côté de la surface de l'organe vont faire une bosse. En analysant la surface du rein on peut trouver une bosse comme ça et la taille et le position de la tumeur peut être approximé avec une sphère. On peut utiliser ça dans le traitement de rayons.

Dans ce cas, le rein est un modèle d'ordinateur qui est originairement créé par des images de scanographie. La surface du modèle est un maillage de triangles que l'on analyse pour, dans chaque nœud du meillage, calculer la courbure. Avec la courbure d'un group de nœuds on peut déterminer si le group indique une tumeur ou pas sous la surface. Si on trouve un groupe comme ça, une sphère qui minimise la distance à des nœuds du group peut être trouvé.

Pour valider le modèle on a besoin de quelque measure d'erreur à comparer contre des informations de validation. Le modèle est validé avec la, soi-disant, distance de Hausdorff qui dit que l'erreur est la plus grande distance, dans tous les moins des distances entre des deux surfaces. Le modèle est valideé avec deux surfaces 3D qui sont créé par des images de la scanographie.

Aussi la méthode pour trouver des tumeur par utilisation des courbures du modèle, et même le modèle sont prometteurs car les erreurs approximatifs sont à moins de la limite de 5mm qui est utilisé par des praticiens dans de traitement. Pourtant, ça serait à valider aussie bien la methode pour trouver des tumeurs que le modèle avec plusieurs des informations de validations.

Contents

1 Introduction

1.1 Introduction

Tumors growing on or near the surface of a kidney will cause a small bump. This thesis explains how we given a surface reconstruction of the organ can retrieve the part of the surface that is indicative of a tumor. The tumor is presumed to be spherical in shape and so the output of the presented method is a sphere that gives the approximate location and size of the tumor. This project is an extension of a dynamic kidney model presented in [1] that simulates the organ deformation as a function of time during the breathing cycle using a mesh morphing approach. In the model a tumor such as the one mentioned is present. The end goal is to use the dynamic model and the tumor finding method in conjunction during radiotherapy in order to minimize damage to tissue that surrounds the tumor. The dynamic model is validated in this thesis using the so called Hausdorff distance as an error measurement between meshes obtained from the simulation and meshes reconstructed from images. The Hausdorff distance computation has no real time constraints but was still optimized using a space partitioning technique.

The proposed method is promising but needs to be consolidated and fine tuned. More data, artificial or preferably real, to test the method on would be preferred before any strong conclusions can be made. The same can be said about the validation of the dynamic model, which also shows promising results. The Hausdorff distance computation optimization gave a speed up of between 5 and 50 times compared to a brute force method depending on the mesh size.

The method for finding the tumor is based on computing the principal curvatures κ_1 and κ_2 of the surface which provides significant information about its local shape. This is exploited as a basic feature for finding regions of interest (ROI) that present a certain curvature characteristic. We consider a ROI as a bump with high positive Gaussian curvature that indicates the presence of an outwards growing tumor. The ROI is found by a morphological dilatation of a curvature type followed by a recursive tagging algorithm that divides the mesh into different regions. The regions are then labeled in order to determine the ROI. Finally, a Best Fit Sphere (BFS) algorithm allows to approximate the tumor. The novelty of this approach lies in the use of tagging based on the discrete curvature. In our context, it is used together with a dynamic model in non-invasive radiotherapy of tumors. It also provides a computationally cheap and noise insensitive alternative to tumor segmentation.

1.2 Goals of this Thesis

This thesis has two main goals. The first goal is to find an approximate position and size of a sphere shaped tumor. This should be done by creating a method for finding a highly convex area on an existing kidney triangular surface mesh using a discrete curvature based approach. Further a sphere should be adjusted to this area in a best fit sense. The result should in some way be validated.

The second goal of the thesis is to investigate the possibility of using an existing or original approach for assessing an error in the dynamic kidney model that this thesis expands on. This error measurement should then be used to validate the dynamic model.

This is achieved through literature studies in mesh processing, 3D programming techniques, 3D rendering, nonlinear optimization and medical imaging and through implementation in some programming language.

1.3 Thesis Outline

The thesis contains a survey related to tumor detection in Section 2.1, a short presentation of the associated dynamic kidney model in Section 2.2 and an introduction to triangle meshes, curvatures and discrete curvatures and how these may be computed on a triangle mesh in Sections 2.3 and 2.4. A discussion on error computation in the context of 3D meshes and organ simulation can be found in Section 2.5 and a related discussion on how space partitioning can be used to optimize distance calculations lie in Section 2.6.

An explanation of the tumor obtaining method resides in Section 3.2 followed by a description of the Hausdorff-distance and how it may accurately and efficiently be computed between two meshes in Section 3.3.

In Section 4, results and validation are presented and briefly discussed. Finally a richer discussion of the results, possible improvements and future possibilities reside in Section 5.

2 Background and Related work

2.1 Lesion Detection

In tumor segmentation and lesion detection, the two main families of methods are image-based (2D) and shape-based (3D) approaches.

Image-based methods operate directly on the 2D medical acquisitions and are associated with image segmentation. Thus, methods based on classification [2], texture analysis [3] or region growing [4] can all be found to be used. These methods cover lesion detection for several organs, *e.g.* liver [5] or kidneys [6]. Moreover, modalities in image segmentation to enhance contrast between different tissues can help detect the structure of interest. The work described in [7] is based on images coming from multi-phases acquisitions. Microwave tomography is a new modality that is efficient when applied to tumor detection, especially for breast cancer [8].

The second family is based on 3D information and is related to shape analysis. Approaches using discrete curvature [9] or spherical harmonics [10] can be found. Shape indices also represent a local measurement of the surface and this information is used to detect colon [11] or breast [8] tumors. Noise in medical images can lead to a noisy reconstruction which reduces the robustness of 3D-based methods. A very efficient smoothing is based on the heat kernel diffusion [12] since it allows the extraction of geometrical features of surface models [13].

A method based on image analysis would be justified if the contrast of the images is high enough. *E.g.* by using microwave tomography or multi-phases acquisitions. The images from which the kidney is reconstructed come from a standard CT-scan acquisition. The method we present here deals with 3D information and is based on discrete curvature characteristics. The use of 3D data is the logical continuity of [14] since its output is a 3D surface model. Unlike [9], this method is entirely automatic and does not need a multi-phases CT-scan acquisition.

2.2 Context of the Dynamic Model

The method proposed in this thesis is developed as a complement to the dynamic model described in [1]. This previous work simulates the motion and the deformation of the kidney during the breathing cycle. It is created from three medical imaging acquisitions taken at three different breathing phases: the exhale phase, the inhale phase and the middle phase between the two previous ones. From these acquisitions, three surface meshes of the kidney, $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$, are constructed [14]. The motion and the deformations are then calculated by a mesh morphing from $\mathcal{M}_1$ to $\mathcal{M}_2$ and then from $\mathcal{M}_2$ to $\mathcal{M}_3$. As mesh morphing approaches are reversible, the full breathing cycle can be covered without any additional computation.

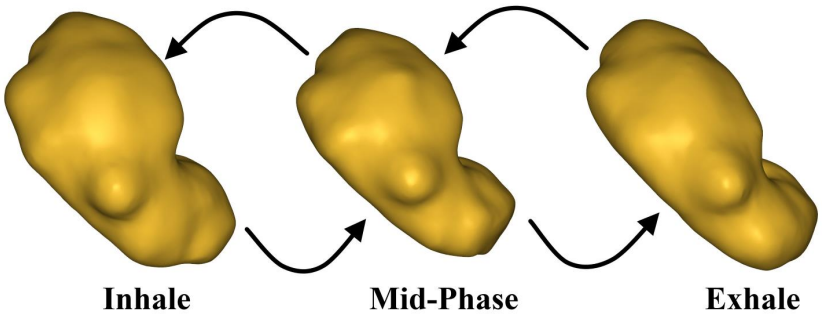


Figure 1: Deformation of kidney during the breathing cycle by mesh morphing.

2.3 Triangle Meshes

The dynamic model is a discrete surface in the form of a triangle mesh. Triangles are together with quadrilaterals the most common fundamental polygon for creating piecewise linear discrete surfaces. This since all other polygons of higher degree can be subdivided into a set of triangles.

A triangular mesh $\mathcal{M}$ carries geometric and topological information. The geometric information consists of a set of vertices $\mathcal{V}$ that store discrete positions in space [15].

$$\mathcal{V} = \{v_1, ..., v_n\}, \quad v_i = (x_i, y_i, z_i) \in \mathbb{R}^3, \quad 1 \leq i \leq n$$

The topological information is how these vertices are connected to each other on a graph. The vertices are connected by a set of edges $\mathcal{E}$, each edge is the connection of two vertices, and a set of faces $\mathcal{F}$, each face is a collection of three edges that together form a triangle.

$$\mathcal{E} = \{e_1, ..., e_m\}, \quad e_j = (v_t, v_h), \quad 1 \leq j \leq m$$

$$\mathcal{F} = \{f_1, ..., f_p\}, \quad f_k = (e_{j1}, e_{j2}, e_{j3}), \quad 1 \leq k \leq p$$

An important topological property of $\mathcal{M}$ is that it has to be so called $2-manifold$. This means that it cannot contain any edges that has more than two adjacent faces and no vertices that are incident to more than one fan of triangles. This ensures that the surface of $\mathcal{M}$ has no self-intersections and that local neighborhoods can be well defined so that certain surface properties can be calculated.

We now introduce the notion of a topological neighborhood on $\mathcal{M}$. A pair of vertices v_1 and v_2 are neighbors if they have a common edge. The ring neighborhood or simply just the neighborhood $\mathcal{N}(v)$ of a vertex v is the set of all vertices that share an edge with v, see Figure 2.

An important property of a $\mathcal{M}$ is its surface normal $\mathbf{n}$ which indicates its orientation. On the face of a triangle the normal is well defined and can be calculated by a cross product between two vectors that lie along two of the edges, though care must taken into which edges are used so that the surface orientation becomes consistent over all of $\mathcal{M}$. On the edges and on the vertices however it is in general not well defined and is instead calculated as a weighted average of the normals of its adjacent faces.

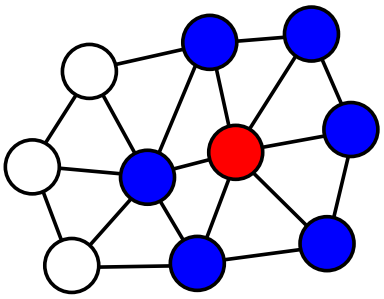


Figure 2: A small set of vertices depicted as circles. They are connected by edges, black lines. They in turn form faces, the white spaces on the interior. One vertex is colored red and its neighborhood vertices has been colored blue.

2.4 Differential Geometry and Discrete Curvatures

2.4.1 1D Curves and Curvature in 2D

This section is dedicated to introducing the concept of curvature and how it can be calculated. It also serves as a very brief introduction to parts of the field of differential geometry [15]. We begin by introducing the notion of curves $\mathbf{x}(s)$ and their curvature on 1 dimensional parametric functions embedded in $\mathbb{R}^2$. $\mathbf{x}(s)$ is a differentiable vector valued function and s is its natural parameter.

$$\mathbf{x}(s) = x(s)\hat{\mathbf{i}} + y(s)\hat{\mathbf{j}}, \qquad s \in [a, b] \in \mathbb{R} \tag{1}$$

$$\mathbf{t}(s) = \mathbf{x}'(s) = x'(s)\hat{\mathbf{i}} + y'(s)\hat{\mathbf{j}} \neq 0, \quad \forall s \in [a, b] \tag{2}$$

$\mathbf{t}(s)$ is the tangent vector of the curve $\mathbf{x}(s)$ and we also calculate a normal $\mathbf{n}(s)$ to the curve by rotating $\mathbf{t}(s)$ by 90° and normalizing it.

$$\mathbf{n}(s) = \frac{\mathbf{t}(s)^{\perp}}{||\mathbf{t}(s)^{\perp}||} \tag{3}$$

The orientation of the curve and its normal is important because it also determines the sign of the curvature. The curvature measures how much a curve deviates from a straight line and can be defined and calculated in one of two ways. Either as the second derivative of $\mathbf{x}(s)$ or by its oscillating circle, which also provides an intuitive image of the curvature property, Figure 3. The oscillating circle is the circle that best approximates the curve locally at some point and the inverse of its radius give the curvature.

$$\kappa(s) = ||\mathbf{x}''(s)|| = \frac{1}{||r(s)||} \tag{4}$$

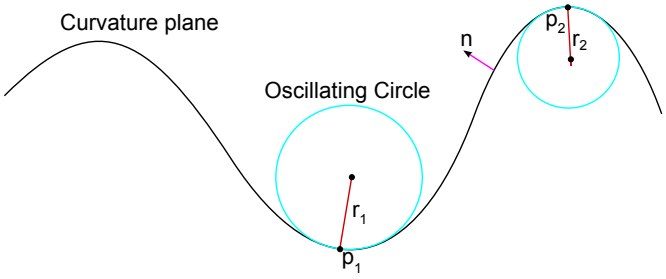


Figure 3: The curvature at point p_1 is defined as the inverse of radius r_1 of the oscillating circle at that point. Note that the curvatures at the two points p_1 and p_2 have different signs since their respective circle of curvature lie on different sides of the surface, whose orientation is indicated by the surface normal n.

2.4.2 2D Surfaces and Curvature in 3D

We further generalize the notion of curvature on smooth surfaces $\mathbf{S}$ embedded in $\mathbb{R}^3$ with the following parametrization:

$$\mathbf{S}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \qquad (u,v) \in \Omega \in \mathbb{R}^2 \tag{5}$$

and partial derivatives:

$$\begin{cases} \mathbf{x}_u(u,v) = \frac{\partial \mathbf{S}}{\partial u}(u,v) \\ \mathbf{x}_v(u,v) = \frac{\partial \mathbf{S}}{\partial v}(u,v) \end{cases} \tag{6}$$

Assuming a regular parametrization, $\mathbf{x}_u \times \mathbf{x}_v \neq 0$, the tangent plane is spanned by two tangent vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ so that the surface normal is given by:

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{||\mathbf{x}_u \times \mathbf{x}_v||} \tag{7}$$

We further introduce the Jacobian matrix $\mathbf{J}$.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_u & \mathbf{x}_v \end{bmatrix} \tag{8}$$

$\mathbf{J}$ corresponds to the linear map that transforms a vector $\bar{\mathbf{w}}$ in parameter space into a tangent vector $\mathbf{t}$ on the surface. From $\mathbf{J}$ we introduce the so called first fundamental form:

$$\mathbf{I} = \mathbf{J}^T\mathbf{J} = \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} \mathbf{x}_u^T\mathbf{x}_u & \mathbf{x}_u^T\mathbf{x}_v \\ \mathbf{x}_u^T\mathbf{x}_v & \mathbf{x}_v^T\mathbf{x}_v \end{bmatrix} \tag{9}$$

It defines an inner product on the tangent space of $\mathbf{S}$. *E.g.* is the cosine of the angle between these two unit direction vectors given by the scalar product $\mathbf{w}_1\mathbf{I}\mathbf{w}_2$.

We now extend the notion of curvatures on the surface $\mathbf{S}$. We let $\mathbf{t} = (u_t, v_t)^T$ be a tangent vector in parameter space at a surface point $\mathbf{p} \in \mathbf{S}$. The normal curvature $\kappa_n(\mathbf{t})$ at $\mathbf{p}$ is the curvature of the planar curve created by intersecting

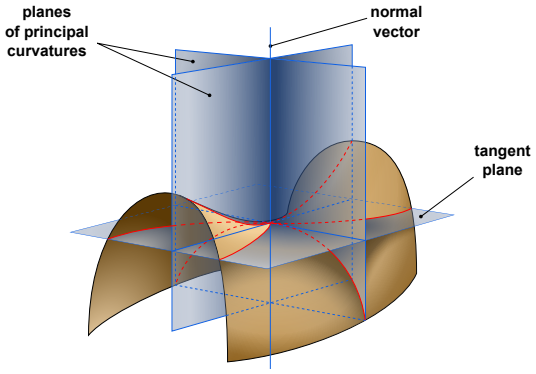


Figure 4: Tangent plane, principal direction planes and surface normal on a saddle shaped surface. The principal curvatures are visible as the red curves that don't lie in the tangent plane. [16]

the surface at $\mathbf{p}$ with the plane spanned by $\mathbf{t}$ and the surface normal $\mathbf{n}$. We can express normal curvature in the direction $\mathbf{t}$ as:

$$\kappa_n(\mathbf{t}) = \frac{\mathbf{t}^T \mathbf{\Pi} \mathbf{t}}{\mathbf{t}^T \mathbf{I} \mathbf{t}} = \frac{e u_t^2 + 2 f u_t v_t + g v_t^2}{E u_t^2 + 2 F u_t v_t + G v_t^2} = \mathbf{t}^T \mathbf{W} \mathbf{t} \tag{10}$$

Where $\mathbf{\Pi}$ is the second fundamental form.

$$\mathbf{\Pi} = \mathbf{J}^T \mathbf{J} = \begin{bmatrix} e & f \\ f & g \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{uu}^T \mathbf{n} & \mathbf{x}_{uv}^T \mathbf{n} \\ \mathbf{x}_{uv}^T \mathbf{n} & \mathbf{x}_{vv}^T \mathbf{n} \end{bmatrix} \tag{11}$$

The partial derivatives in Equation 11 are calculated as:

$$\begin{cases} \mathbf{x}_{uu} = \frac{\partial^2 \mathbf{S}}{\partial^2 u} \\ \mathbf{x}_{vv} = \frac{\partial^2 \mathbf{S}}{\partial^2 v} \\ \mathbf{x}_{uv} = \frac{\partial^2 \mathbf{S}}{\partial u \partial v} \end{cases} \tag{12}$$

Rotating the tangent vector $\mathbf{t}$ around the normal and assuming that $\kappa_n(\mathbf{t})$ varies with $\mathbf{t}$ it can be shown that the curvature has two distinct extreme values, called the principal curvatures. We denote these with κ_1 for the maximum curvature and κ_2 for the minimum. If $\kappa_1 \neq \kappa_2$ we can also denote two unique associated tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ that we call the principal directions. It can also be shown that $\mathbf{t}_1$, $\mathbf{t}_2$ and $\mathbf{n}$ are orthogonal to each other, Figure 4.

The local properties of a surface can be compactly described in parameter space using the Weingarten matrix $\mathbf{W}$ seen in Equation 10. It is a 2×2 symmetric matrix whose eigenvectors are the principal directions $\mathbf{t}_1$ and $\mathbf{t}_2$ and its eigenvalues are the principal curvatures κ_1 and κ_2.

$$\mathbf{W} = \begin{bmatrix} A & B \\ B & C \end{bmatrix} \tag{13}$$

Now we also introduce the Mean and Gaussian curvatures, denoted by H and K. These metrics are very important in shape analysis and also prove the foundation for the proposed tumor finding method.

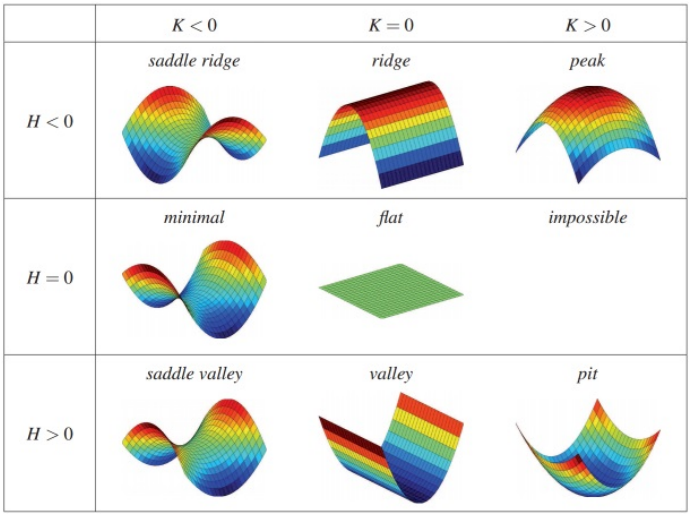
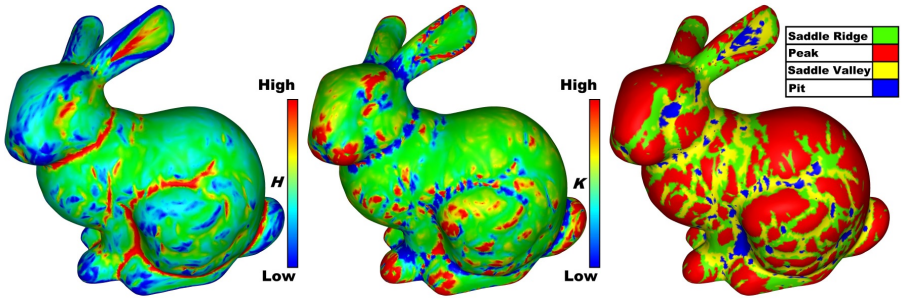


Figure 5: Curvature types by sign in Gaussian and Mean curvatures [17].

$$H = \frac{\kappa_1 + \kappa_1}{2} \tag{14}$$

$$K = \kappa_1 \cdot \kappa_2 \tag{15}$$

Further we also introduce nine curvature types defined by the sign changes in H and K, Figure 5. It should be noted that out of these nine types only four, *saddle ridge*, *peak*, *saddle valley* and *pit* occur in the context of general meshes since *ridges* and *valleys* effectively require that κ_1 or κ_2 is 0, or in the discrete case within some threshold value and the *minimal* type also requires the special case where κ_1 and κ_2 is of the same absolute value but of different sign. This can be seen in Figure 6c where only the four remaining curvature types are present.



(a) Mean curvature. (b) Gaussian curvature. (c) Curvature types.

Figure 6: Mean, H, and Gaussian, K, curvatures together with the curvature types formed by their signs.

2.4.3 Discrete Curvatures

When computing an approximation to the principal directions and curvatures on a triangular mesh we make use of the local properties of neighboring vertices, edges and faces. In this section four such methods are mentioned and briefly compared to the method of choice, which is further described in Section 3.2.2.

One way of doing this is by using covariance matrices as approximations to the fundamental forms $\mathbf{I}$ and $\mathbf{\Pi}$ in Equations 9 and 11 [18]. For a vertex v and its n neighbors $\mathcal{V}_N$ on a mesh the first 3×3 covariance matrix at v is calculated as:

$$\mathbf{C}_I = \frac{1}{n}\sum_{i=1}^{n}(v_i - \boldsymbol{\mu}_v)(v_i - \boldsymbol{\mu}_v)^T, \qquad \boldsymbol{\mu}_v = \frac{1}{n}\sum_{i=1}^{n} v_i$$

Two of its eigenvectors, $\mathbf{t}_1$ and $\mathbf{t}_2$ form an approximation to the tangent plane at v while the third is the normal $\mathbf{n}$. A second 2×2 matrix is calculated as:

$$\mathbf{C}_\Pi = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{\mu}_y)(y_i - \boldsymbol{\mu}_y)^T, \qquad \boldsymbol{\mu}_y = \frac{1}{n}\sum_{i=1}^{n} y_i$$

with

$$y_i = \begin{bmatrix}(v_i - v)\mathbf{n}\end{bmatrix}\begin{bmatrix}(v_i - v)\mathbf{t}_1 \\ (v_i - v)\mathbf{t}_2\end{bmatrix}$$

The eigenvectors to $\mathbf{C}_\Pi$ is then considered as approximations to the principal directions at v. This method is simple but uses only the position of the neighboring vertices.

Three other methods are described in [19] along with the one used. Like in the previous method we consider a vertex v and its n neighbors $v_i \in \mathcal{V}_N$. We also consider the normal vector to v, $\mathbf{n}$ and the vector $\mathbf{y}_i$ from v to v_i projected on the local tangent plane. There is a so called normal curvature approximation method in which the normal curvature $\kappa_i = \mathbf{y}_i^T\mathbf{W}\mathbf{y}_i$ is approximated by:

$$\kappa_i = 2\frac{(v - v_i)\cdot\mathbf{n}}{(v - v_i)\cdot(v - v_i)}$$

Which produces a system of equations:

$$\mathbf{y}_i^T\mathbf{W}\mathbf{y}_i = \kappa_i, \qquad 1 \leq i \leq n$$

That we want to solve for $\mathbf{W}$. The system can be written as:

$$\mathbf{y}_i^T\mathbf{W}\mathbf{y}_i = \begin{bmatrix}u_i & v_i\end{bmatrix}\begin{bmatrix}A & B \\ B & C\end{bmatrix}\begin{bmatrix}u_i \\ v_i\end{bmatrix} = \begin{bmatrix}u_i^2 & 2u_iv_i & v_i^2\end{bmatrix}\begin{bmatrix}A \\ B \\ C\end{bmatrix}$$

Letting $\mathbf{U}$ be the $n \times 3$ matrix with rows $(u_i^2, 2u_iv_i, v_i^2)$, $\mathbf{x} = (A, B, C)^T$ and $\mathbf{d}$ be the n-vector containing the curvatures κ_i the system can be written as:

$$\mathbf{U}\mathbf{x} = \mathbf{d}$$

This system will be overdetermined and needs to be solved in a least squared fashion.

In the quadric surface fitting method [19] a quadratic surface patch $\mathbf{S}_p$ is fitted locally to the vertex v.

$$\mathbf{S}_p = \frac{A}{2}u^2 + Buv + \frac{C}{2}v^2$$

Where A, B, C are the same entries in $\mathbf{W}$ as in Equation 13. Just as in the normal curvature method this will form a systems of equations. It can be shown that these two methods are identical in nature, just that the former will approximate the curvature with a circle and the latter with a parabola.

A more novel method described in [20] calculates an estimate for the curvature tensor for each edge, then an average may be calculated for the vertex. This method may be much more accurate and robust than the ones mentioned, especially on highly irregular meshes.

The first three methods mentioned are at most second order and so the third order method chosen, in which a cubic surface patch is used, supposedly will be more accurate. The last method should be compared to this one in practice but was never implemented. However the meshes used are highly regular and smooth and so a more robust method may in practice make very little difference.

2.5 Dynamic Model Error Approximation

In order to validate the dynamic models accuracy an error measurement is required. No one standard error measurement exists for this kind of medical soft tissue mesh simulations and many methods are validated simply by visual inspection, *e.g.* a comparison against some image data or proposed reference models [21]. However for general meshes a standard difference measurement is the Hausdorff-distance [22]. It is commonly used to measure the difference in meshes that has been smoothed, subdivided or re-sampled to a lower resolution [23]. This in order to make sure that the overall shape of the mesh is preserved during the operations and that the mesh remains of good quality. Other methods are also used to validate mesh quality but is inapplicable in this context [24]. Hausdorff-distance is also used for precise collision detection between complicated meshes [25]. The Hausdorff-distance is a measurement of the global worst error, and as a complement a root mean squared distance (RMSD) is also calculated.

The error is computed for two meshes. They correspond to the dynamic model between the inhale and mid-phase and between the mid-phase and exhale. The two meshes from the dynamic model is then compared to meshes constructed from image acquisitions at the corresponding in between breathing phases.

2.6 Space Partitioning and Bounding Volume Hierarchies

Even though the error estimation is performed off-line an optimization was made to speed up the computation.

Space partitioning is a family of commonly used optimization techniques that works by dividing one problem, where many objects interact with each other based on their relative distance, that is of some complexity into two steps of lower complexity. Thereby reducing the overall complexity and the required runtime. However there is a trade off in memory consumption since new data structures are needed. Two essential kinds exist, Bounding Volume Hierarchies (BVH) and Quad-Tree(2D)/Octree(3D)/KD-trees. The former mainly used on static object components and the later more common in problems with dynamic objects where the partitioning data structure has to be re-created often. They are also used in conjunction with each other. A famous example of Octree/Quad-Tree partitioning is the N-body problem in which celestial bodies or particles interact with each other through gravity or other forces in 2 or 3 dimensions [26]. Space partitioning is also commonly used in ray trace rendering and in the video games industry where fast real time calculations are necessary. There it is used for rigid body physics simulations where the model meshes usually are much more complex than the bounding volumes through which they are tested for collision with each other. Game objects are then divided and encapsulated by BVHs while the game world is divided using an Octree or Quad-tree. The method implemented here uses BVHs of sphere swept rectangles (SSR) to encapsulate the 3D models.

3 Method

3.1 Software and Work Process

The complete methods for both the tumor detection and Hausdorff distance calculations are implemented in a user interactive application made in C++. The 3D rendering is made using OpenGL with the color maps and model shading implemented in shader programs. Some code for the distance calculations between geometric primitives and their representation was taken from or inspired by the library Geometric Tools [27]. Some code for 3D mathematics including mesh processing and curvature computation was provided. The user interface is constructed using AntTweakBar [28], a portable UI library that works with different versions of both OpenGL and DirectX. The window and OpenGL environments are provided by GLFW [29] and GLEW [30] respectively.

The problems was solved using an iterative development process with some prototypes first being implemented and tested in MATLAB. Additional third party software used was Meshlab [31], which was used for constructing high resolution meshes as well as importing and exporting meshes into different file formats.

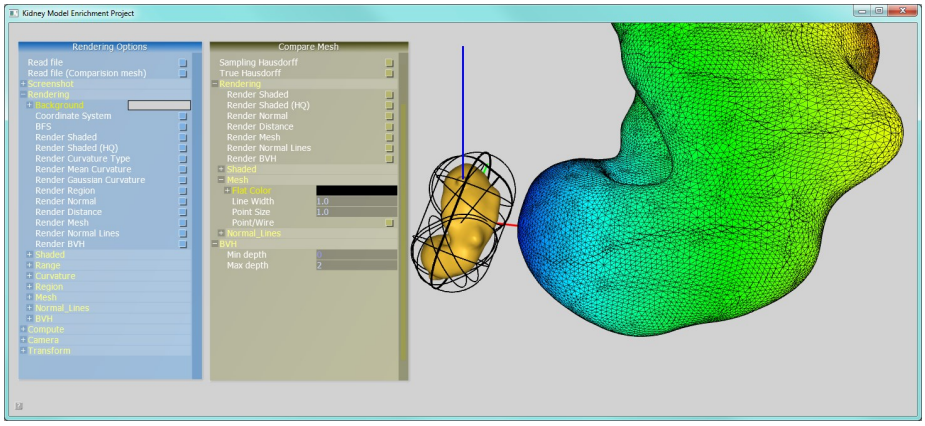


Figure 7: Screenshot of the application developed and used within the project. It supports rendering of most mesh properties and concepts introduced in the past and following sections. It also has support for some mesh processing and some functionality for user friendliness such as color map manipulation and a screen capture function that removes the GUI and applies anti-aliasing to the image by multi sampling.

3.2 Tumor Detection

3.2.1 Method Overview

The input to the method is a triangular mesh $\mathcal{M}$ and the output one or more spheres that represents tumors. However in the real life case only one tumor is expected and so in the following sections this may also be assumed. The different steps of the complete method are mentioned below and are described more precisely in the following subsections. They are carried out in this order.

1. Compute curvatures and curvature types for all vertices on the mesh.
2. Dilation on the mesh to enclose the vertices of the tumor indicative area.
3. Region forming by recursive tagging.
4. Labeling of all regions as either tumor or non-tumor.
5. Best Fit Spheres (BFS) positioning of the tumor region.

3.2.2 Curvature Computation

The principal curvatures on the surface mesh are calculated for each vertex using a local surface approximation method as proposed in [20]. The principal curvatures, κ_1 and κ_2, and their corresponding principal directions, $\mathbf{t}_1$ and $\mathbf{t}_2$, denote the maximum and minimum curvatures at a point on the surface. From the principal curvatures the mean, H, and Gaussian, K, curvatures can be calculated using Equations 14 and 15. Their sign changes can be used to construct the table of curvature types seen in Figure 5.

The normal for one vertex is calculated by taking the average of the normals of all adjacent faces, where each contribution is weighted by the angle of the face at the vertex [32]. Angle weighted normal computations are in general more accurate then using constant or area weights [15].

The method used extends the methods mentioned in section 2.4.3 by not only accounting for the position of the neighboring vertices but also for their surface normals. This allows for a third-order approximation method [19]. As in these previously mentioned methods the goal is to calculate an approximation for the Weingarten matrix $\mathbf{W}$ and then eigendecompose it in order to find our approximated principal directions and curvatures.

In the following we consider the calculations for one vertex $v \in \mathcal{M}$ that is surrounded by n directly connected neighbors $v_i \in \mathcal{N}(v)$, $1 < i < n$. The calculations are done in a local parametric surface space, just as in section 2.4.2. The local space is created by considering the normal $\mathbf{n}$ at v as the z-direction and then constructing two arbitrary perpendicular vectors that we denote $\mathbf{u}$ and $\mathbf{v}$ in the tangent plane at v. The set of vectors $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ will then be our local coordinate system where the computations are made.

Just as the quadric method in Section 2.4.3 we now also attempt to fit a surface to the neighboring vertex data. We let this be a cubic surface patch given by:

$$f(x, y) = z(x, y) = \frac{A}{2}x^2 + Bxy + \frac{C}{2}y^2 + Dx^3 + Ex^2y + Fxy^2 + Gy^3 \quad (16)$$

Where x, y and z are the parameters in $\mathbf{u}, \mathbf{v}$ and $\mathbf{n}$ respectively. We now remind ourselves of the appearance of $\mathbf{W}$, which is the same as when it was introduced in section 2.4.2, Equations 10 and 13. However we are now dealing with an approximation where the surface factors A, B and C in Equation 16 correspond to the entries in $\mathbf{W}$.

$$\mathbf{W} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

The normal to the surface in Equation 16 is given by:

$$\bar{\mathbf{n}}(x,y) = \begin{pmatrix} f_x(x,y) \\ f_y(x,y) \\ -1 \end{pmatrix}^T = \begin{pmatrix} Ax + By + 3Dx^2 + 2Exy + Fy^2 \\ Bx + Cy + Ex^2 + 2Fxy + 3Gy^2 \\ -1 \end{pmatrix}^T \tag{17}$$

We let $\bar{\mathbf{n}}_i = (a_i, b_i, c_i)$ and $\bar{\mathbf{p}}_i = (x_i, y_i, z_i)$ denote the normal and position of the neighboring vertex v_i. Both the position and normal need to be expressed in the local coordinate system $(\mathbf{u}, \mathbf{v}, \mathbf{n})$. The transform is done from $\mathbf{p}_i$ and $\mathbf{n}_i$ by first subtracting the position of the current vertex v from $\mathbf{p}_i$, then both it and $\mathbf{n}_i$ are multiplied with the matrix $[\mathbf{u}, \mathbf{v}, \mathbf{n}]^T$ from the left.

$$\begin{aligned} \bar{\mathbf{p}}_i &= [\mathbf{u}, \mathbf{v}, \mathbf{n}]^T \times (\mathbf{p}_i - v)^T \\ \bar{\mathbf{n}}_i &= [\mathbf{u}, \mathbf{v}, \mathbf{n}]^T \times \mathbf{n}_i^T \end{aligned}$$

We rewrite $\bar{\mathbf{n}}$ in Equation 17 as $(-\frac{a_i}{c_i}, -\frac{b_i}{c_i}, -1)$ and let:

$$\mathbf{x} = (A \quad B \quad C \quad D \quad E \quad F \quad G)^T$$

Then we use Equations 16 and 17 together with $\mathbf{x}$ which gives us three equations for each vertex that we can solve to find the values of the entries in $\mathbf{x}$.

$$\begin{aligned} (\frac{1}{2}x_i^2 \quad x_i y_i \quad \frac{1}{2}y_i^2 \quad x_i^3 \quad x_i^2 y_i \quad x_i y_i^2 \quad y_i^3)\mathbf{x} &= z_i \\ (x_i \quad y_i \quad 0 \quad 3x_i^2 \quad 2x_i y_i \quad y_i^2 \quad 0)\mathbf{x} &= -\frac{a_i}{c_i} \\ (0 \quad x_i \quad y_i \quad 0 \quad x_i^2 \quad 2x_i y_i \quad 3y_i^3)\mathbf{x} &= -\frac{b_i}{c_i} \end{aligned}$$

This system is then written as:

$$\mathbf{U}\mathbf{x} = \mathbf{d}$$

$\mathbf{U}$ is a $3n \times 7$ matrix and $\mathbf{d}$ is a $3n$ vector. It should be noted that the vertex must have at least three neighboring vertices in order for the system to not be underdetermined. However in our case the system will always be overdetermined which means a true solution does not exist and has to be approximated using a least-squares fit. For more information on how such a systems can be solved see section 3.2.6. Since our kidney meshes are closed and cyclic a case where a vertex has less than 3 neighbors is not possible. Should such vertices occur, which is a possibility for a general mesh, the same method is however applied but with an expected accuracy penalty.

Once we have our solution vector $\mathbf{x}$ we can use the values of A, B and C to form $\mathbf{W}$, whose eigenvectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, and associated eigenvalues, λ_1 and λ_2 where $\lambda_1 > \lambda_2$, gives us the principal directions and curvatures. Since the eigenvectors are given in local parameter space they need to be transformed into world space in order to obtain $\mathbf{t}_1$ and $\mathbf{t}_2$.

$$\mathbf{t}_1 = \mathbf{e}_1^1 \times \mathbf{u} + \mathbf{e}_1^2 \times \mathbf{v}$$
$$\mathbf{t}_2 = \mathbf{e}_2^1 \times \mathbf{u} + \mathbf{e}_2^2 \times \mathbf{v}$$

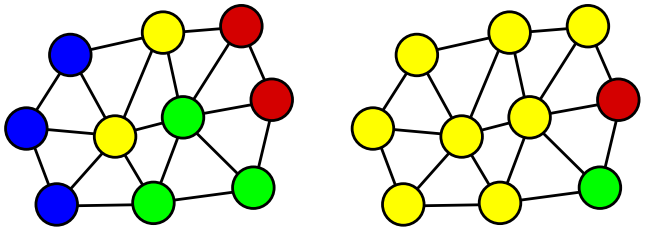
Where the upper indices indicate the index in the vector. The transform contains no scaling so the obtained eigenvalues are equal to the principal curvatures κ_1 and κ_2. The Mean and Gaussian curvatures are calculated using Equations 14 and 15 and the vertex is assigned a curvature type from Figure 5.

3.2.3 Morphological Dilatation of the Regions

All vertices are assigned a binary tag based on their curvature types. Vertices of *saddle valley* type (Figure 5) are tagged as 1 and all others as 0. Then, the dilatation is carried out as follows: each vertex tagged as 0 is re-tagged as 1 if any of its neighbors is tagged as 1, see Algorithm 1. The dilatation is repeated until a ROI has been found or every vertex has the same tag, which means that the method has failed.

In practice this is done by simply changing the curvature type in all neighbors of a vertex that is of *saddle valley* type to that type. This can be seen in Figure 8 where vertices of all present curvature types exists. All neighbors of a yellow vertex is then changed to yellow in the dilatation process.

The purpose of the dilatation is to create a closed region of vertices which surrounds the convex tumor area. The result can be seen in Figures 15a and 15b where the yellow region has grown to completely enclose a set of vertices that are indicative of the tumor.



(a) Mesh with 0 dilatations. (b) Mesh with 1 dilatation.

Figure 8: A mesh where each vertex has a color coded curvature type. Yellow vertices are tagged as 1 and so is expanded so that every neighbor of a tagged vertex also becomes tagged.

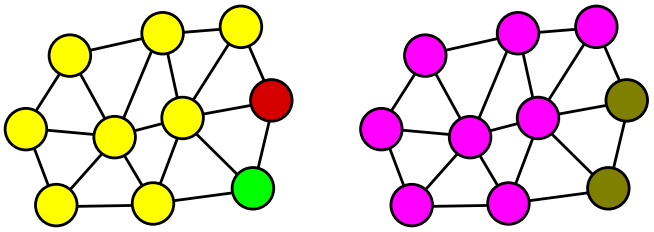
Algorithm 1 Dilation. All neighbors of a vertex with curvature type *saddlevalley* is re-tagged as 1.

Data:
List of ingoing vertices with an associated tag, $\mathcal{V}_{in} = \{v_1, ..., v_i, ..., v_n\}$
List of outgoing vertices $\mathcal{V}_o$ that will replace $\mathcal{V}_{in}$
Begin:
Copy all vertices from $\mathcal{V}_{in} \rightarrow \mathcal{V}_o$
for all $v_i \in \mathcal{V}_{in}$ **do**
$\tilde{v}_i$ is the vertex at the current index i in $\mathcal{V}_o$
if v_i is tagged as 1 **then**
Get list of neighboring vertices of $\tilde{v}_i$, $\tilde{v}_n$
Set tag of all vertices in $\tilde{v}_n$ to 1
end if
end for

3.2.4 Recursive Tagging

The recursive tagging is performed by considering the same binary tags of the vertices as in the previous Section, 3.2.3. First all vertices are considered as not belonging to any group and are placed in a list. A vertex that does not yet belong to a group is chosen from the list and added to a new group. From the current vertex all neighbors are expanded, if a neighbor has the same tag as the current vertex, it is added to the group and this vertex is expanded as well. This process continues until all vertices have been assigned to a group. See Algorithm 2.

An example can be seen in Figure 9 where two groups are formed. All yellow vertices form one group and the two remaining vertices form a second group. For larger meshes multiple such groups are generally formed.



(a) Mesh with 1 dilatation. (b) Mesh with 2 regions.

Figure 9: Yellow vertices are tagged as 1 and others as 0. Vertices with the same tag belonging to the same connected region will form a separate group. In this example two groups are formed, all yellow vertices form one group and the green and the red form another.

Algorithm 2 Region forming by recursive tagging.

Data:
List of ingoing vertices with an associated tag, $\mathcal{V}_{in} = \{v_1, ..., v_i, ..., v_n\}$
Return a list of groups of vertices, $R = \{R_1, ..., R_j, ..., R_m\}$

procedure EXPAND(Vertex v)
Get neighboring vertices, $n = \mathcal{N}(v)$
for all $\hat{v} \in n$ **do**
if $\hat{v}$ has same tag as v **then**
Add $\hat{v}$ to same group as v and EXPAND($\hat{v}$)
end if
end for
end procedure

Begin:
for all $v \in \mathcal{V}_{in}$ **do**
if v does not yet belong to group **then**
Add v to a new group and EXPAND(v)
end if
end for

3.2.5 Region Labeling

In order to classify the groups as a tumor several features are calculated for each group. The once used are the most common curvature type, the relative amount of vertices in the group compared to the whole mesh, mean values of the principal curvatures κ_1 and κ_2 and then the groups mean and Gaussian curvatures are calculated from these curvatures. It is possible to *by hand* define some criteria that selects the correct groups but in order to automatize part of the process a Artificial Neural Network (ANN) [33] is used as a classifier. The ANN is trained in MATLAB using the ANN toolbox and then exported to a text file that is later imported by the main application where the feed forward classification procedure is calculated.

The layout of the ANN can be seen in Figure 10, it was found using some experimenting and rules of thumb. It has six inputs, one for each feature, a hidden layer with 10 nodes, a bias and a hyperbolic tangent activation function that outputs values in the range $[-1, 1]$. The output layer has 10 inputs, same as the number of outputs from the hidden layer, a bias, a logarithmic sigmoid as activation function that outputs values in the range of $[0, 1]$ where 1 means the group represents a tumor and 0 means that it is not a tumor. Training data was created by visual inspection and manual tagging of the groups as either being indicative of a tumor or not. One way to automatize this process could be to grow artificial tumor like areas on the mesh but due to time constraints this was not feasible.

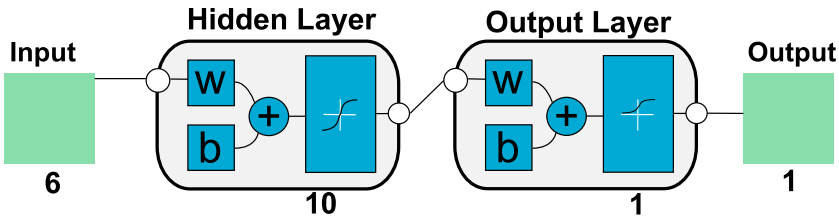


Figure 10: Typical layout of the classifying ANN used.

3.2.6 Best Fit Sphere (BFS) Positioning

Fitting a sphere in a best fit sense to a vertex distribution $\mathcal{V}$ of n vertices can be formulated as a non-linear least squares optimization problem [34]. The squared distance from one vertex $v_i \in \mathcal{V}$ to the sphere is given by Equation 18, where $\mathbf{x}$ is the position and radius of the sphere. The squared sum of Equation 18 over all vertices is written in vector form as $F(\mathbf{x})$ in Equation 19. The problem is solved using an unconstrained Gauss-Newton method [35] with a line search.

From the squared distance from v_i to the sphere in Equation 18 the minimization function is formulated as Equation 19.

$$f_i(\mathbf{x}) = (x_i - a)^2 + (y_i - b)^2 + (z_i - c)^2 - r^2, \quad \mathbf{x} = \{x, y, z, r\} \tag{18}$$

$$min\, f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} f_i(\mathbf{x})^2 = \frac{1}{2}F(\mathbf{x})^T F(\mathbf{x}) \tag{19}$$

The gradient vector and Hessian matrix is calculated for v_i as Equations 20 and 21.

$$\nabla f_i(\mathbf{x}) = \begin{bmatrix} -2(x_i - x) \\ -2(y_i - y) \\ -2(z_i - z) \\ -2r \end{bmatrix}^T \tag{20}$$

$$\nabla^2 f_i(\mathbf{x}) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix} \tag{21}$$

The chain rule gives the gradient and Hessian of the minimization function in Equations 22 and 23. $\nabla F(\mathbf{x})$ is an $n \times 4$ matrix where row i contains the gradient vector $\nabla f_i(\mathbf{x})$.

$$\nabla f(\mathbf{x}) = \nabla F(\mathbf{x}) F(\mathbf{x}) \tag{22}$$

$$\nabla^2 f(\mathbf{x}) = \nabla F(\mathbf{x}) \nabla F(\mathbf{x})^T + \sum_{i=1}^{n} f_i(\mathbf{x}) \nabla^2 f_i(\mathbf{x}) \tag{23}$$

A Gauss-Newton approximation is made to the Hessian so that the summation part is ignored and it gets the final expression as Equation 24. This approximation is made since the summation part causes the matrix to become non-positive semi definite and therefore cause the optimization algorithm to fail.

$$\nabla^2 f(\mathbf{x}) \approx \nabla F(\mathbf{x}) \nabla F(\mathbf{x})^T \tag{24}$$

Equation 25 is solved for $\mathbf{p}$ and $\mathbf{p}$ is added to the solution $\mathbf{x}$. This is done iteratively until the size of the gradient vector, Equation 22, is sufficiently close to zero. In practice this limit was set to about $10\times^{-6}$.

$$\begin{cases} \nabla F(\mathbf{x}) \nabla F(\mathbf{x})^T \mathbf{p} = \nabla F(\mathbf{x}) F(\mathbf{x}) \\ \mathbf{x} = \mathbf{x} - \mathbf{p} \end{cases} \tag{25}$$

In order to make sure that every iteration decreases the value of the error function a bracketing style line search strategy was used in which the length of $\mathbf{p}$ was halved until this criterion was fulfilled.

After a solution has been found, the radius of the sphere is adjusted while its position is not modified. This way, it prevents the sphere from extending through the vertex distribution and guarantees that the approximated tumor lies entirely inside the kidney model.

The complete method can be seen in Algorithm 3.

A 2D equivalent, a Best Fit Circle problem, can be seen in Figure 11. There, a perturbed arc distribution of vertices can be seen as blue $\times$'s. The initial solution is set as the barycenter of the vertex distribution and the initial radius as half the distance to the closest vertex. This is also used to initialize the BFS. The final solution after the radius adjustment can be seen as the red circle.

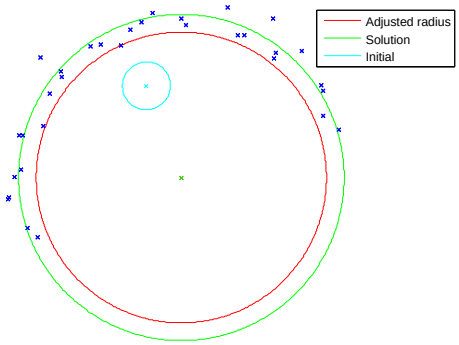


Figure 11: Best Fit Circle, 2-dimensional equivalent of the BFS problem. The light blue circle is the initial position and the red and green circles are the solutions, the red circle having the same center as the green but with an adjusted radius so that it lies completely inside the vertex distribution, seen as blue $\times$'s.

Algorithm 3 Best Fit Spheres method.

Data:
List of ingoing vertices: $\mathcal{V} = \{v_1, ..., v_i, ..., v_n\}$
Returns a vector: $\mathbf{x} = \{x, y, z, r\}$
Begin:
Initialize $\{x, y, z\} \in \mathbf{x}$ to the barycenter of $\mathcal{V}$
Set $r \in \mathbf{x}$ to half the shortest distance from $\{x, y, z\} \in \mathbf{x}$ to $\mathcal{V}$
Calculate initial value of $\nabla f(\mathbf{x})$

while $\nabla f(\mathbf{x}) > \epsilon$ AND max number of iterations NOT achieved **do**
for all $v_i \in \mathcal{V}$ **do**
Add contribution of $f_i(\mathbf{x})$ to $F(\mathbf{x})$
Add contribution of $\nabla f_i(\mathbf{x})$ to $\nabla F(\mathbf{x})$
end for

Calculate function: $f(\mathbf{x}) = \frac{1}{2} F(\mathbf{x})^T F(\mathbf{x})$
Calculate gradient: $\nabla f(\mathbf{x}) = \nabla F(\mathbf{x}) F(\mathbf{x})$
Calculate Hessian: $\nabla^2 f(\mathbf{x}) = \nabla F(\mathbf{x}) \nabla F(\mathbf{x})^T$
Solve the system for $\mathbf{p}$: $\nabla F(\mathbf{x}) \nabla F(\mathbf{x})^T \mathbf{p} = \nabla F(\mathbf{x}) F(\mathbf{x})$

$\alpha = 1$
repeat
$\mathbf{x}_s = \mathbf{x} - \alpha \mathbf{p}$
Calculate a function value $f(\mathbf{x}_s)$ in same way as $f(\mathbf{x})$
$\alpha = \frac{1}{2} \alpha$
until $f(\mathbf{x}) > f(\mathbf{x}_s)$ OR max number of line search iterations achieved
$\mathbf{x} = \mathbf{x}_s$
end while
Adjust $r \in \mathbf{x}$ so that it is the minimum distance from $\{x, y, z\} \in \mathbf{x}$ to $\mathcal{V}$

3.3 Model Error Estimation using the Hausdorff Distance

3.3.1 Hausdorff Distance

Given two surface meshes $\mathcal{M}_A$ and $\mathcal{M}_B$ in $\mathbb{R}^3$ the one-sided Hausdorff distance $h(\mathcal{M}_A, \mathcal{M}_B)$ from $\mathcal{M}_A$ to $\mathcal{M}_B$, $h(\mathcal{M}_A, \mathcal{M}_B)$ is defined as:

$$h(\mathcal{M}_A, \mathcal{M}_B) = max_{a \in \mathcal{M}_A}(min_{b \in \mathcal{M}_B} d(a, b)) \tag{26}$$

where $d(.,.)$ is the euclidean distance between the points a and b. The two sided Hausdorff distance $H(\mathcal{M}_A, \mathcal{M}_B)$ is then defined as:

$$H(\mathcal{M}_A, \mathcal{M}_B) = max(h(\mathcal{M}_A, \mathcal{M}_B), h(\mathcal{M}_B, \mathcal{M}_A)) \tag{27}$$

These definitions are also valid for general surfaces $\mathbf{S}$, not just surface meshes. Exact Hausdorff distances are complicated and time consuming to compute for large general meshes. Therefore an approximate computation of Equation 26 is performed. All triangles in $\mathcal{M}_A$ are sub-sampled at random positions k-times within the triangle and the distance from the sampled points to all triangles in $\mathcal{M}_B$ is calculated. An example using two surfaces consisting of one triangle each can be seen in Figure 12. Sub-sampling of triangles is compared to a uniform triangle subdivision in [23] and is found to produce similar results and convergence. Randomized sub-sampling of surfaces is also used when calculating Hausdorff distances in the context of non-uniform rational basis splines surfaces (NURBS) [36]. For each vertex and for each sampled point on mesh $\mathcal{M}_A$ the shortest distance to the surface of mesh $\mathcal{M}_B$ is computed. The current largest distance is kept and updated once a larger distance is discovered. Once all points have been queried against the other mesh this largest distance is $h(\mathcal{M}_A, \mathcal{M}_B)$. $h(\mathcal{M}_B, \mathcal{M}_A)$ is computed in the analogous way and $H(\mathcal{M}_A, \mathcal{M}_B)$ is computed using Equation 27.

A triangle can be randomly sampled by using Equation 28. v_A, v_B, v_C are the three vertices of the triangle and $\mathbf{s}$ is the sampled point inside the triangle. r_1 and r_2 are random numbers sampled between and including 0 and 1.

$$\begin{cases} \mathbf{s} = (1 - \sqrt{r_1}) \cdot v_A + \sqrt{r_1} \cdot (1 - r_2)) \cdot v_B + \sqrt{r_1} \cdot r_2) \cdot v_C \\ \mathbf{s}, v_A, v_B, v_C = [x, y, z] \in \mathbb{R}^3 \\ r_1, r_2 \in [0, 1] \end{cases} \tag{28}$$

The distance between a triangle and a point in $\mathbb{R}^3$ can be calculated in different ways, *e.g.* by projecting the point onto the surface of the triangle and performing the initial calculation in $\mathbb{R}^2$ and afterwards adding the distance to the plane [37]. The method used is the one implemented in [27] and is based around breaking the problem into parts where distances are calculated to the triangle plane, the line segments and its vertices. The MSD from $\mathcal{M}_A$ to $\mathcal{M}_B$ is calculated as the mean of the square root distance of every sampled point.

$$msd = \sqrt{\frac{1}{N} \sum_{k=1}^{N} d_k^2} \tag{29}$$

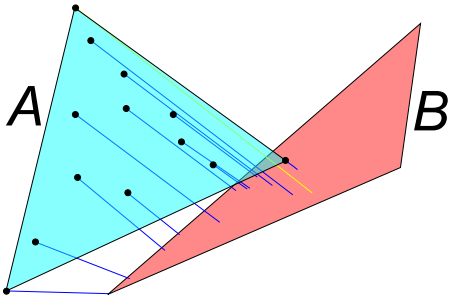


Figure 12: One sided Hausdorff-distance from surface $\mathcal{M}_A$ to $\mathcal{M}_B$, $h(\mathcal{M}_a, \mathcal{M}_B)$, seen as the yellow line. Surface $\mathcal{M}_A$ is sampled 10 times in addition to its vertices and the closest distance for every such point to surface $\mathcal{M}_B$ is calculated. The largest of all those distances is the approximated one sided Hausdorff-distance.

3.3.2 Optimization Using Bounding Volume Hierarchies

The brute force Hausdorff-distance computation method was optimized using a BVH. The approach uses sphere swept rectangles (SSR) that encapsulates different parts of the surface and is stored in a binary tree structure [25]. A SSR is a Minkowsky sum of a rectangle in $\mathbb{R}^3$ and a sphere of some radius at all points on the rectangle.

Oriented BVs are generally preferred over axis aligned bounding boxes (AABB) and bounding spheres (BS) since they can give a tighter fit around the triangles and converge much faster to the underlying geometry, therefore fewer BVs are needed [38]. However when comparing against SSRs, distance queries against BS's is faster and their memory consumption lower. Comparing SSR against the also commonly used oriented bounding box (OOB) the SSR provides simpler and faster distance queries. An approach with mixed BVs was considered but not implemented due to time constraints and already satisfying results using only SSRs.

The SSRs are constructed as follows. Given a list of N triangles, a sub-mesh, $f^i = \{v_A^i, v_B^i, v_C^i\} \in \mathcal{M}_s$ a SSR is created. First a mean $\boldsymbol{\mu}$ is calculated for the triangles and their vertices using Equation 30.

$$\boldsymbol{\mu} = \frac{1}{3N} \sum_{i=0}^{N} (v_A^i + v_B^i + v_C^i), \qquad \boldsymbol{\mu} \in \mathbb{R}^3 \tag{30}$$

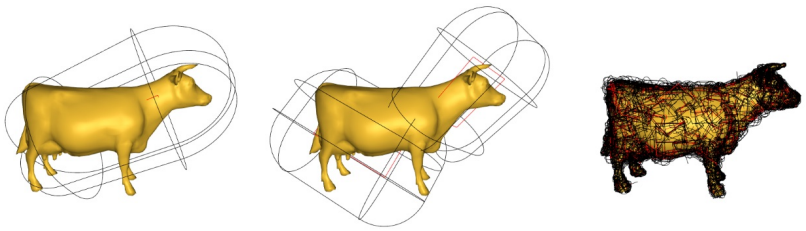
Then a 3×3 covariance matrix C is calculated for $\mathcal{M}_s$ using Equation 31.

$$\begin{cases} C_{jk} & = \frac{1}{3N} \sum_{i=0}^{N} (\hat{\mathbf{p}}_j^i \hat{\mathbf{p}}_k^i + \hat{\mathbf{q}}_j^i \hat{\mathbf{q}}_k^i + \hat{\mathbf{r}}_j^i \hat{\mathbf{r}}_k^i), \qquad 1 \leq j, k \leq 3 \\ \hat{\mathbf{p}}^i & = v_A^i - \boldsymbol{\mu} \\ \hat{\mathbf{q}}^i & = v_B^i - \boldsymbol{\mu} \\ \hat{\mathbf{r}}^i & = v_C^i - \boldsymbol{\mu} \end{cases} \tag{31}$$

The three eigenvectors of C are calculated and normalized to form the basis of the SSR, centered at $\boldsymbol{\mu}$. Along each eigenvector the maximum distance to a vertex in $\mathcal{M}_s$ is calculated and the size of the SSR is adjusted to those distances. Along the eigenvector with the highest distance is the rectangles long side, along the one with next highest distance is the short side and along the shortest is

the normal to the rectangle. Then the size of the rectangle and its sphere are adjusted so that all vertices in $\mathcal{M}_s$ lie inside the bounding volume. This method is somewhat sensitive to the density if the vertex distribution and can be further improved upon by the use of convex hull sampling which will effectively cause the SSR to align better with the shape of the mesh. This improvement was left out because of time constraints and already good performance.

The BVH is built using a top-down approach which means that first, we start with a BV that encloses all triangles in $\mathcal{M}_s$. Then assuming $\mathcal{M}_s$ contains a large number of triangles it will be subdivided into two subsets, $\mathcal{M}_{s1}$ and $\mathcal{M}_{s2}$. These will form the branches of our current node in the BVH. This process of subdividing the triangle sets continues until they contain fewer then some specified amount of triangles. Setting this amount too high will speed up the BVH creation but slow down the distance queries so some hands-on tuning is generally necessary for good results. Through some experimenting it was found that for the models used a limit of about 10 triangles per leaf node gives good performance. The triangles in $\mathcal{M}_s$ are separated into the new subsets by a plane centered at $\boldsymbol{\mu}$ with a normal parallel to the current SSR's long side axis. Then for each triangle in $\mathcal{M}_s$ the barycenter is calculated and depending on which side of the plane the barycenter lies on the triangle is put in either $\mathcal{M}_{s1}$ or $\mathcal{M}_{s2}$. The final result of a complete BVH of a 3D model on different depth levels can be seen in Figure 13.

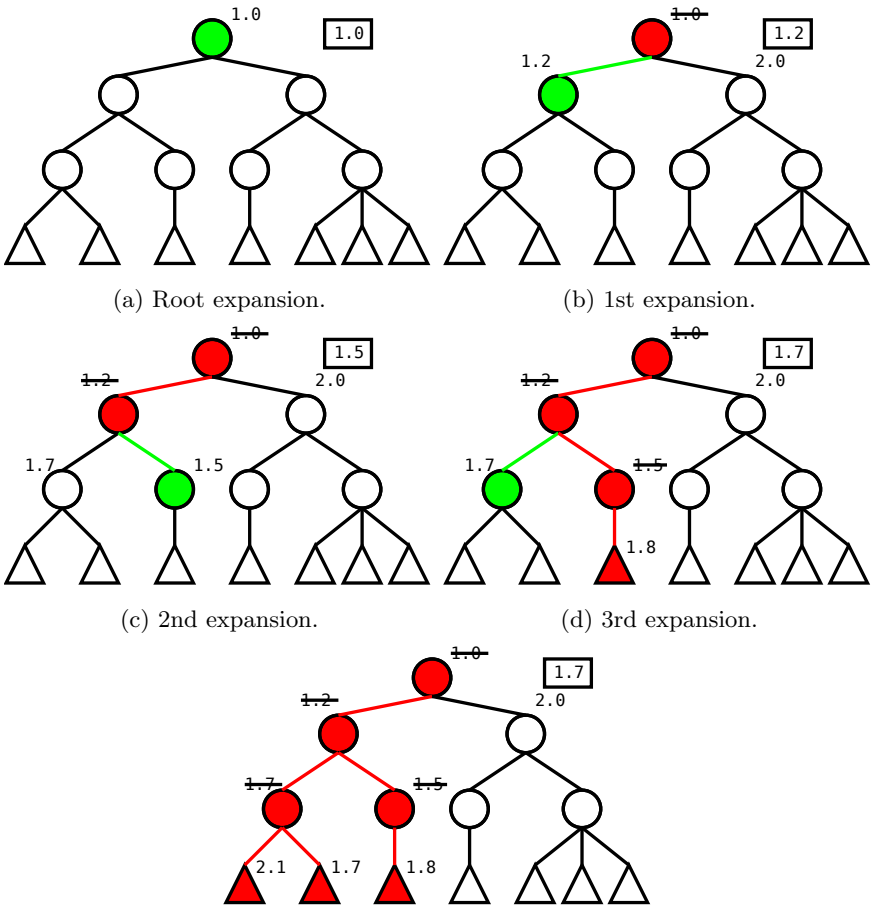


(a) Root depth. (b) Second depth level. (c) Leaf nodes.

Figure 13: Varying levels of depth in BVH enclosing the triangles and their vertices using SSRs on a 3D model.

The distance queries against the BVH is done for vertices. For distance calculations of vertices against rectangles and SSRs in 3D see [27] or [39]. The tree is traversed using an informed depth first search (DFS). Informed in this context means knowledge about the child nodes is taken into account when deciding which node to expand first. The node whose associated BV has the shortest distance to the queried vertex is expanded first. An example of a BVH distance query can be seen in Figure 14. Expanded nodes are visible in red and in green, the next node to be expanded. First the root node is expanded and the first two child nodes are inspected, Figure 14a. The child node with the BV with the shortest distance to the queried vertex is expanded. This distance is visible next to the node or triangle. The current shortest distance to an unexpanded node or triangle is visible in the box next to the tree. The tree is traversed in Figures 14a through 14e and the final distance of 1.7 is returned. In this example only one main branch of the tree is traversed and we do queries against less than half

of the triangles. In this example a brute force approach is probably faster but it demonstrates the power of the approach, should the number of triangles and nodes in the tree be greater.



(a) Root expansion.

(b) 1st expansion.

(c) 2nd expansion.

(d) 3rd expansion.

(e) Final expansion.

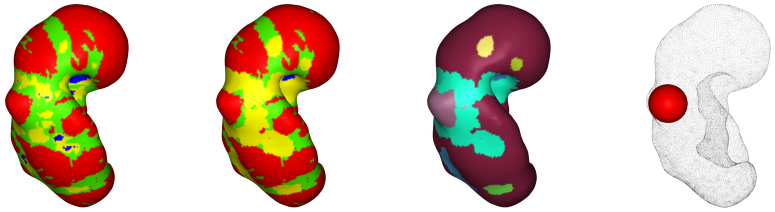
Figure 14: BVH distance query. Red nodes are visited and the green node is the next to be expanded. The number in the box displays the currently known shortest distance to a triangle or unexpanded node. Traversal ends when no unexpanded nodes with a shorter distance than the distance to a currently known triangle exists. Then that distance is returned.

4 Results and Validation

4.1 Sphere Tumor Approximation

4.1.1 Results

The results of a sphere approximation of the tumor location and size can be seen in Figure 15 together with maps of the curvature types and the obtained regions of the mesh.



(a) Curvatures before dilatations.

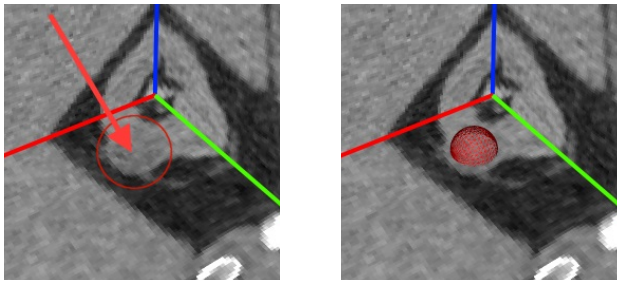
(b) Curvatures after two dilatations.

(c) Obtained regions.

(d) Tumor location and size estimation.

Figure 15: Result of curvature computations and two dilatations, region forming and tumor location.

In Figure 16, the sphere approximating the tumor is compared to the image acquisitions from which the mesh was originally reconstructed. The contrast of the kidney has been augmented with radiopaque agent. Therefore, the tumor can be seen as a slightly darker area within the red circle in Figure 16a. The obtained tumor approximation is shown in Figure 16b. It encloses the darker area while remaining inside the kidney.

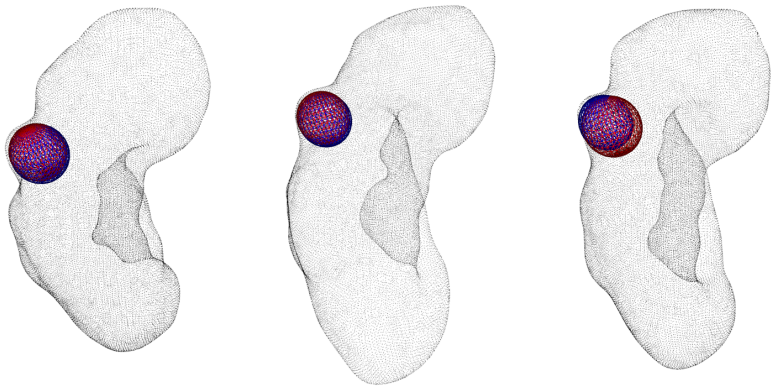


(a) Tumor marked with red circle and its center indicated by the arrow.

(b) BFS estimation of the tumor.

Figure 16: Validation of the best fit sphere on 3D orthogonal slices. The tumor can be seen in (a) as the slightly darker area within the red circle. The corresponding BFS approximation is shown in (b).

The precision of the retrieved tumor is obtained by a comparison with its manual segmentation (see Figure 17 and table 1). Results show satisfying accuracy in a medical context since it is below the 5mm margin of error practitioners use when removing a tumor.



(a) Inhale. (b) Mid-phase. (c) Exhale.

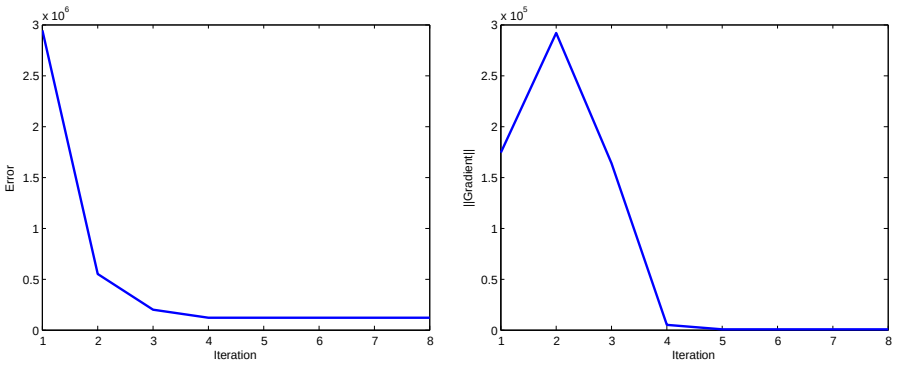
Figure 17: Manually set tumors in blue and the method obtained ones in red.

Mesh	Inhale	Mid-phase	Exhale
Distance between center	0.70	1.57	1.14
Radius difference	0.27	0.37	0.97

Table 1: Error estimation (in millimeters) between manually defined sphere shaped tumors and the BFS approximations. Error is estimated for one kidney corresponding to three breathing phases.

4.1.2 BFS Convergence

A typical convergence behavior of the BFS method can be seen in figure 18. The gradient is quickly reduced to a value in size of about 7×10^{-7} after 8 iterations at which the reduction in gradient size starts to flatten out. For this reason it was set to terminate once the gradient reached a value below 7×10^{-6}. The final value of the error function is in this case about 1.2×10^{-5}, measured in millimeters. The line search makes sure that the value of the error function decrease with each iteration.



(a) Error. (b) Gradient size.

Figure 18: An example of a convergence behavior for the Gauss-Newton method used in the BFS method.

4.1.3 Execution Time

The execution times for the different parts are presented in Table 2. The mesh contains 2.9k vertices and 5.8k triangles. The labeling step is omitted since its computational time is comparatively negligible. Most of the runtime is dedicated to the computation of curvatures and the region forming by recursive tagging. Optimizing these steps could considerably improve the whole execution time, although it is already sufficiently efficient for the considered application. The method was executed on a simultaneous multi-threading capable CPU with 4 cores at 2.1GHz and 8GBs of RAM.

The normal, curvature and BFS implementations are partially parallelized using OpenMP and are expected to scale well on increasingly parallel architecture. The dilatation and tagging methods have serial implementations.

	Norm.	Curv.	Dil.	Tagging	BFS	Tot.
ms	6.40	110	2.45	51.5	1.07	171
%	3.73	64.2	1.43	30.0	0.62	100

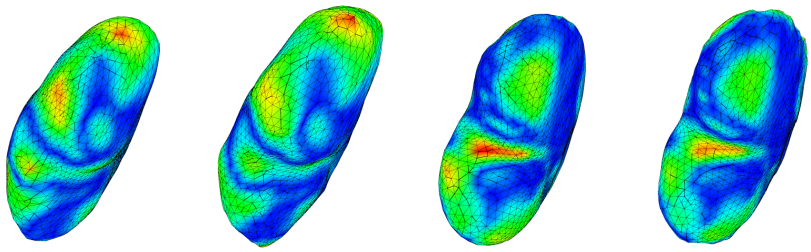
Table 2: Runtime for the different parts in the method, both in *ms* and as percentage of total runtime. The steps are in order, computing normals and curvatures, performing dilatation operations, forming of groups and applying the BFS method

4.2 Validation of the Dynamic Model

4.2.1 Results

The dynamic model is validated using two pairs of meshes. Two are created by the dynamic model and two are constructed from image acquisitions. The meshes correspond to breathing phases in between the inhale and mid-phase, and the mid-phase and exhale. In Figure 19 the results can be seen for the two mesh pairs. *R14 Meta* is compared against *R14 Real* and *R36 Meta* is compared against *R36 Real*.

The distances between the meshes are generally small and the Hausdorff-distance is within or close to the 5mm margin, table 3. This result indicates that the dynamic mesh morphing model is indeed a promising approach.



(a) R14 Meta (b) R14 Real (c) R36 Meta (d) R36 Real

Figure 19: Per vertex distance between meshes. Red is a larger distance and blue is a lower.

	h(A,B)	h(B,A)	H(A,B)	msd(A,B)	msd(B,A)	MSD(A,B)
R14	5.65	5.16	5.65	1.52	1.28	1.40
R36	3.58	3.27	3.58	0.96	0.91	0.94

Table 3: Hasdorff-distances and MSD in millimeters for the in-between meshes.

4.2.2 Hausdorff-distance Bounding Volume Optimization

The BVH Hausdorff-distance calculation is compared to a brute force implementation where distances are calculated between each vertex on one mesh and each triangle on the other. Four sets of two meshes each was used, two with a lower resolution and two with a higher. The meshes are the same as in the previous Section 4.2.1, *R14 Lo* in Table 4 corresponds to a calculation between mesh (a) and (b) in Figure 19. The higher resolution meshes are constructed from the lower resolution ones by subdivision and smoothing. The meshes that are compared are of similar resolution. The low resolution ones have about 2.8k vertices and 5.7k triangles and the high resolution meshes have about 28k vertices and 56k triangles.

In Table 4 it can be seen that for the lower resolution meshes the BVH method is about 5 times faster than the brute force method introduced in Section 3.3.1 and about 50 times faster for the larger meshes.

	Construction	Traversal	Complete BVH	Brute Force
R14 Lo	71.57	32.23	103.8	462.6
	0.69	0.31	1.0	4.46
R36 Lo	65.77	31.10	96.87	482.8
	0.68	0.32	1.0	4.98
R14 Hi	602.7	551.3	1154	60075
	0.52	0.48	1.0	52.1
R36 Hi	590.7	479.3	1070	53459
	0.55	0.45	1.0	50.0

Table 4: Hausdorff-distance calculation and BVH construction execution times in ms in the upper rows and in relation to the BVH query in the lower rows. The times for the complete BVH method include the time for constructing the BVH as well as the time for distance calculations by traversal of the tree structure.

The use of BVHs increases memory consumption. Each node stores two pointers, a rectangle object, a list of pointers to triangles, that is empty if the node is not a leaf node, and a counter of how many vertices belong to the BV. The rectangle is constructed out of three 3D vectors and two real numbers, for a total amount of 11 real numbers. If all these integers, pointers and floating point values are of the same size, as is the case in this implementation, one node contains a total of 15 4-byte values and the total memory consumption can then be calculated as:

$$\text{Memory Consumption} = 4 * (K * 15 + N - M)$$

Where K is the number of BVH nodes, N is the number of triangles and M is the number of leaf nodes. In table 5 it can be seen that the memory consumption has an approximate linear dependence on the number of vertices on the mesh.

Vertices	Triangles	BVH Nodes	Leaf Nodes	Memory (MB)	Byte / Vertex
2286	4568	1407	704	0.10	43.7
8659	17314	5207	2604	0.37	42.9
29194	58384	17487	8744	1.25	42.7

Table 5: Number of BVH nodes and memory consumption for three kidney meshes that represents the same geometry but with different number of vertices.

5 Discussion

5.1 Possible Improvements

During the project several possible improvements to the different parts where thought of or found in literature but never implemented and tested for various reasons. Be it time constraints or satisfying results from the approach already implemented and in use.

Improvements to the tumor finding method could possibly be made to the curvature computations using the method mentioned in Section 2.4.3, introduced in [20]. At least it should be compared to the current method to see if it produces any significant difference in curvature values, though it is not expected since our meshes are relatively smooth, closed, of high resolution and in general well behaved. A histogram equalization method for the curvature values κ_1 and κ_2 or the mean and Gaussian curvatures could possibly improve the initial region forming. One could also imagine using a more advanced region forming method *e.g.* by not considering the curvature types in Figure 5 but instead using a generalized watershed algorithm on the curvature values. Using a machine learning technique for the labeling of the regions has the benefit of automation, however this is lost when data still has to be sorted and marked by hand. In order to fully automatize this process one could grow artificial tumors on the surface mesh by displacing vertices *e.g.* as a bell curve. Then the vertices belonging to the tumor would be known a priori and training data for a supervised method could be created. The BFS method could see an improvement by using a more advanced line search then the implemented backtracking method. Though the accuracy it reaches correspond to nanometer scale which is well below the millimeter scale limit used when treating tumors of this kind and the convergence is already fast. A form of barrier method could also be used instead of adjusting the radius of the result. Should the tumor be known to be of some other geometric shape or be segmented before hand and known to not change its shape during the breathing cycle, a similar optimization problem could be formulated and solved by the same general method. All of these relatively small improvements could possibly result in a more robust and versatile method.

The BVH method used when calculating the Hausdorff distance can be improved by using mixed BVs which could create tighter fits around the triangle data. Also the way the BV is calculated can be improved, one such improvement is mentioned in Section 3.3.2. One can also imagine using an optimization method similar to the BFS to fit geometries in a best possible way to the triangle data.

Many of the algorithms used throughout the project, especially the ones involving computations of vertex properties, are highly parallelizable and could see benefits in speedup by performing the computations on graphics processing units (GPU).

5.2 Conclusions

We showed that discrete curvatures can be a relevant information to detect a bumped region indicating a tumor on a surface model. Once this region is identified, it is possible to retrieve the position and size of the tumor using a BFS algorithm. However, a prerequisite is that the tumor is large enough and lies close enough to the surface so that a significant bump may occur. Another prerequisite is that the mesh resolution has to be high enough to provide good curvature information. A low resolution would aggravate the local approximations which would lead to wrong results. This approach is promising but needs to be consolidated.

Validation of the dynamic model using a Hausdorff-distance and a RMS distance suggests that the mesh morphing is a valid approach in this context. However the dynamic model has only been validated in this way for two meshes made from the same organ. In order to strengthen the result, more such meshes would be needed. Since there seem to be no one gold standard method for validating models of this kind other measurements could also be used in conjunction with the Hausdorff distance. For example it should be excepted for the organ to be nearly incompressible during normal motion and so volume and possibly also its surface area should be preserved.

Space partitioning using BVHs can greatly improve the computational time needed for distance calculations between groups of geometric primitives. The results in this case ranged from a speedup of 5 to 50 times compared to a brute force method. The usage of BVHs gives a memory increase that grows approximately linearly with the mesh size and for larger meshes increased the memory consumption with 1-2 MBs, which for most real time applications of this kind is a reasonable trade off.

5.3 Future Work

In order for the tumor detection method to be useful in practice the dynamic model needs to be continuously validated against image acquisitions where the kidney is tracked. This is supposed to be performed in real time during a radiation treatment session and so a very limited amount of image slices can be produced. The intention is to compare the mesh of the dynamic model against a vertical and a horizontal slice and calculate an error between the actual position and shape of the kidney and the current state of the dynamic model. The model can then be adjusted in order to compensate for this error.

Acknowledgments

I would like to thank my supervisors Valentin Leonardi, Jean-Luc Mari and Marc Daniel of Aix-Marseille Université, CNRS, LSIS UMR for their insights, encouragement and for the opportunity to work with them on this project. I would also like to thank Giang Vo for her previous work and involvement within the project and Anders Hast for his help and assistance.

Finally I would like to thank the people of Marseille in general and of the campus of Luminy in particular for being good friends and making my stay in France both interesting and enjoyable.

List of Figures

References

[1] Marc Daniel Valentin Léonardi, Vincent Vidal and Jean-Luc Mari. Multiple Reconstruction and Dynamic Modeling of 3D Digital Objects Using a Morphing Approach. *The Visual Computer*, 2014.

[2] J.J Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient Multilevel Brain Tumor Segmentation With Integrated Bayesian Model Classification. *IEEE Transactions on Medical Imaging*, 27:629 – 640, 2008.

[3] Anna Jerebko, Ronald Summers, James Malley, Marek Franaszek, and C. Daniel Johnson. Computer-Assisted Detection of Colonic Polyps with CT Colonography Using Neural Networks and Binary Classification Trees. *Medical Physics*, 30:52 – 60, 2003.

[4] Sarang Joshi, Stephen Pizer, Thomas Fletcher, Paul Yushkevich, Andrew Thall, and J. S. Marron. Multiscale Deformable Model Segmentation and Statistical Shape Analysis Using Medial Descriptions. *IEEE Transactions on Medical Imaging*, 21(5), May 2002.

[5] L. Massoptier and S. Casciaro. A new fully automatic and robust algorithm for fast segmentation of liver tissue and tumors from CT scans. *European Radiology*, 18:1658 – 1665, 2008.

[6] DY Kim and JW Park. Computer-Aided Detection of Kidney Tumor on Abdominal Computed Tomography Scans. *Acta Radiologica*, 45:791 – 795, 2004.

[7] Marius Goerge Linguraru, Jianhua Yao, Rabindra Gautam, James Peterson, Zhixi Li, W. Marston Linehan, and Ronald Summers. Renal Tumor Quantification and Classification in Contrast-Enhanced Abdominal CT. *Pattern Recognition*, 42:1149 – 1161, 2009.

[8] Natalia Irishina, Miguel Moscoso, and Oliver Dorn. Microwave Imaging for Early Breast Cancer Detection Using a Shape-based Strategy. *IEEE Transactions on Biomedical Engineering*, 56:1143 – 1153, 2009.

[9] M. Linguraru, S. Wang, F. Shah, R. Gautam, J. Peterson, W. Linehan, and R. Summers. Automated Noninvasive Classification of Renal Cancer on Multiphase CT. *Medical Physics*, 38:5738 – 5746, 2011.

[10] M. El-Shenawee and E.L. Miller. Spherical Harmonics Microwave Algorithm for Shape and Location Reconstruction of Breast Vancer Tumor. *IEEE Transactions on Medical Imaging*, 25:1258 – 1271, 2006.

[11] H. Yoshida, J. Näppi, P. MacEneaney, D.T. Rubin, and A.H. Dachman. Computer-Aided Diagnosis Scheme for Detection of Polyps at CT Colonography. *Radiographics*, 22:963 – 979, 2002.

[12] Seongho Seo, Moo Chung, and Houri Vorperian. Heat Kernel Smoothing Using Laplace-Beltrami Eigenfunctions. *MICCAI*, pages 505 – 512, 2010.

[13] J. Liu, S. WanS. Wang. Yao, M. Linguraru, and R. Summers. Manifold Diffusion for Exophytic Kidney Lesion Detection on Non-contrast CT Images. *MICCAI*, 8149:340 – 347, 2013.

[14] V. Leonardi, J.-L. Mari, V. Vidal, and M. Daniel. 3D reconstruction from CT-scan volume dataset - application to kidney modeling. In *SCCG 2011, 27th Spring conference on Computer Graphics*, pages pp. 141–148, Viničné, Slovakia, 28 April 2011.

[15] Mario Botsch et al. *Polygonal Mesh Processing*. A K Peters, 1 edition, 2010. Chapter 3: Differential Geometry.

[16] Wikipedia. Minimal Surface Curvature Planes. http://en.wikipedia.org/wiki/Principal_curvature. Accessed: 2014-07-14.

[17] Dimitri Kudelski. *Détection automatique d'objets géologiques á partir de donnés numériques d'affleurements 3D*. PhD thesis, Université de Provence, Aix Marseille I, 2011.

[18] Sylvain Petitjean. A Survey of Methods for Recovering Quadrics in Triangle Meshes. *ACM Computing Surveys*, 2(34), July 2002.

[19] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, January 2004.

[20] Harlen Costa Batagelo and Shin-Ting Wu. Estimating curvatures and their derivatives on meshes of arbitrary topology from sampling directions. *Vis. Comput.*, 23(9):803–812, August 2007.

[21] R. Haddad, P. Clarysse, M. Orkisz, P. Croisille, D. Revel, and I E. Magnin. A Realistic Anthropomorphic Dynamic Heart Phantom. In *Computers in Cardiology, 2005*, pages 801–804, Sept 2005.

[22] Mikhail J. Atallah. A Linear Time Algorithm for the Hausdorff Distance Between Convex Polygons. *Information Processing Letters*, 17(4):207 – 209, 1983.

[23] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring Error on Simplified Surfaces. Technical report, Meshlab, Paris, France, France, 1996.

[24] Kai Wang, Fakhri Torkhani, and Annick Montanvert. A Fast Roughness-Based Approach to the Assessment of 3D mesh Visual Quality. *Computers and Graphics*, 36(7):808 – 818, 2012. Augmented Reality Computer Graphics in China.

[25] Min Tang, Minkyoung Lee, and Young J. Kim. Interactive Hausdorff Distance Computation for General Polygonal Models. *ACM Trans. Graph.*, 28(3):74:1–74:9, July 2009.

[26] Josh Barnes and Piet Hut. A Hierarchical O(N log N) Force-Calculation Algorithm. *Nature*, 324:446 – 449, 1986.

[27] LLC Geometric Tools. geometrictools. http://www.geometrictools.com/. Version: Wild Magic 5.11, License: Boost Software License, Accessed: 2014-06-17.

[28] Philippe Decaudin. AntTweakBar GUI. http://anttweakbar.sourceforge.net/. Version: 1.16, License: zlib/libpng, Accessed: 2014-06-17.

[29] The GLFW Development Team. GLFW. http://www.glfw.org/. version: 3.0.4, License: zlib/libpng, Accessed: 2014-06-17.

[30] Milan Ikits and Marcelo Magallon. GLEW - OpenGL Extension Wrangler. http://glew.sourceforge.net/. version: 1.10.0, License: Modified BSD License, Mesa 3-D License and Khronos License, Accessed: 2014-06-17.

[31] ISTI CNR. Meshlab. http://meshlab.sourceforge.net/. License: GPL, Accessed: 2014-06-17.

[32] Grit Thürmer and Charles A. Wüthrich. Computing Vertex Normals from Polygonal Facets. *J. Graph. Tools*, 3(1):43–46, March 1998.

[33] Andries P. Engelbrecht. *Computational Intelligence*. John Wiley & Sons, 2 edition, 2007. Chapters 2 and 3.

[34] D Gatinel, J Malet, T Hoang-Xuan, and DT Azar. Corneal Elevation Topography: Best Fit Sphere, Elevation Distance, Asphericity, Toricity, and Clinical Implications. *Cornea*, 30(5):508–515, May 2011.

[35] Ariela Sofer Stephen G. Nash. *Linear and Nonlinear Programming*. McGraw-Hill, 1 edition, 1995. Appendix D.

[36] Iddo Hanniel, Adarsh Krishnamurthy, and Sara McMains. Computing the Hausdorff Distance Between NURBS Surfaces Using Numerical Iteration on the GPU. *Graphical Models*, 74(4):255 – 264, 2012. GMP2012.

[37] Mark W. Jones. 3D Distance from a Point to a Triangle, CSR-5-95. Technical report, Department of Computer Science, University of Wales Swansea, February 1995.

[38] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

[39] Christer Ericson. *Real-Time Collision Detection*. CRC Press, 1 edition, 2004. Chap. 4-6.