User-Guided Semi-Automatic Segmentation of Medical 3D Images

Sander Vanheste

Supervised by A. Vaxman

A thesis presented for the degree of
Master of Science

Universiteit Utrecht

Department of Information and Computing Sciences
Utrecht University
The Netherlands
2019

1 Abstract

In cooperation with the Utrecht Medical Center (UMC), I built a 3D browser-based MRI/CT e-learning system. I propose an improved way of defining correct target volumes in 3D MRI and CT scans: instead of user-defined, handcrafted target volumes in which the target contour of a specific anatomical structure has to be traced by hand for each 2D image slice, this method introduces automatic interpolation between user-provided contours. This is achieved with a new variant of the active contour method. As a result, only a subset of the slices have to be segmented manually. This improves on similar contour-based user-guided segmentation methods.

2 Introduction

The Utrecht Medical Center (UMC) currently uses an e-learning and examination system called VQuest. This software runs on their local computers, and provides radiology teachers the ability to define anatomical questions (like 'point to the liver in the following 3D CT') based on 3D CT and MRI scans. The system allows for the creation of questions that ask the student to put a marker on certain desired anatomical target structures (eg. the liver or the stomach). Teachers often manually draw the correct target region on each MRI, for each slice in which the structure is present, even though they have an arsenal of semi-automatic segmentation tools available that are supposed to ease this process (The UMC uses MITK segmentation tooling).
In light of this, and to increase the accessibility and portability of their e-learning, the UMC wanted to introduce a web-based, new system that is more accessible and offers teachers a new user-guided segmentation tool that is very easy to use, provide decent results and runs fast enough in a distributed client-server architecture. A big part of the problem with their current system is the complexity of the segmentation tooling: teachers get overwhelmed by the amount of options, they have a number of different algorithms to choose from, and the results are often undesirable. My effort is to create a single, coherent segmentation tool that uses a new segmentation algorithm, allowing the process of segmentation to become very easy. I will implement it in the context of the novel e-learning environment which I develop in parallel. The method must run relatively fast on modern systems. After the teacher provides 2D contour masks on a number of slices of his/her own choice, my novel variant of active contours provides a full segmentation of the dataset, using the masks the teacher provided as constraints. This makes sure that each slice conforms to the local image edges that are present in that slice. After the algorithm runs, the teacher can look at the result and optionally decide to add a few defined 2D masks to increase the accuracy of the segmentation. Part of the segmentation then gets updated accordingly. The power of this new method lies in its simplicity: all parameters and steps are pre-determined. The teacher only has to provide a number of slice contours. The use of user-defined masks on an arbitrary amount

of slices, and using this as input to an active contours method, is a novel way of approaching the segmentation problem, so a close relation with the associated teachers at the UMC is of vital importance. I will be gathering feedback and conducting a user study to discover whether the new method is causing teachers to more often use automatic segmentation, and how well they think the new system functions. In short, my contributions are:

- A novel, web-based e-learning system to be used by the UMC
- A novel variant of a user-guided segmentation algorithm, developed specifically for use in the new system
- A new segmentation tool to be used by radiology teachers
- A critical reflection and analysis of performance, mainly based on input from the department of Radiology at the UMC

The segmentation tool must contain as little parameters and options to tweak as possible. The tool must only be usable within the context of the e-learning system. This way, it distinguishes itself from existing segmentation tooling, such as Medical Imaging Toolkit (MITK), which is currently being used by the UMC radiology department and has an overwhelming amount of options. The tool must implement a novel user-guided segmentation algorithm, that is based on existing work on 2D/3D segmentation.

3 Literature Review

3.1 (3D) segmentation of datasets

A well-established user-guided segmentation technique is called Active Contours. Grounded in a 1988 study called 'Snakes: Active Contour Models' [Kass et al., 1988], the core idea of this method is that an initial (user-provided) contour estimate is evolved until a certain *energy* is minimized. It allows rough, user-drawn curves to automatically conform to the gradient of an image in a natural, smooth way. An explicit, parametric representation is used to model the curve. There are two types of energies considered: intrinsic and extrinsic. Intrinsic energy is represented using the differentiation of the curve, and penalizes sharp corners and non-compact shapes. Extrinsic energy is derived using information from the dataset (typically: the gradient), and awards parts of the curve that follow edges in the data. From these energies, an ideal energy equation is derived, which, in practical implementations, must be discretized and the curve updated iteratively to approximate the ideal energy. The idea of evolving contours using energy minimization is very powerful, and has given rise to a lot of different variants. In its most basic form, the algorithm is applicable to 2D images and runs very fast, but different extensions have widely varying complexity and applicability. An interesting extension first proposed in [Caselles

et al., 1997] used mathematical concepts from geodesics to change the active contours model. Using geodesics greatly increases the mathematical complexity of the model, and decreases the running speed somewhat, but has an advantage: it allows for changes in topology, meaning that contours can naturally split and merge. This property is not required for the intended segmentation tool however: a single 3D contour that masks an anatomical structure is enough. The idea of using geodesics is still interesting, and does not have to be combined with an active contours model. For example, [Criminisi et al., 2008] used geodesics to create a distance transformation of the image, and a probabilistic model to segment the foreground from the background.
Because the only intended use is e-learning, the professors at the UMC explicitly stated that a high segmentation accuracy is less important then the speed of the tool. It needs to be as fast as possible: deviations from a perfect segmentation in the form of noise is permitted. Considering an extension like geodesics must be justified: it must increase the general segmentation quality while keeping the running time relatively low. The e-learning system uses 3D datasets, so an absolute requirement is that the method segments 3D contours. Analyzing studies that aim to extend regular active contours to three dimensions is therefore justified. There have been a number of attempts to extend active contours to three dimensions, with varying results. [Caselles et al., 1996, Way et al., 2006, Yushkevich et al., 2006a]. [Way et al., 2006] is very promising: it combines active contours with an automatic diagnosis system to classify the segmentations, with very good results. The main idea is that the user provided a Volume Of Interest (VOI), on which their variant of Active Contours (AC) is run. In the energy equation of the original AC, the gradient of the image is used as an energy, as well as the 2D curvature of the curve. Here, the energy equation is extended to three dimensions by introducing the 3D gradient of the image, and the 3D curvature. Now, each point (voxel) has an energy that is related to all three dimensions, creating a continuity in the z-direction that 2D AC cannot achieve. Furthermore, the system uses static parameters that are pre-set to their most optimal values, creating a method that is very user-friendly, making it interesting in context of this research. And, the method is generally applicable, as it does not use shape priors or training based on a supervised set (this way, any kind of pathology can be segmented). Letting the user only specify a Volume of Interest (VOI) is not desired though: in the intended segmentation tool, the expert must be able to provide exact slice masks on a number of slices, letting the system finish the remaining segmentation work automatically. An interesting idea is to introduce additional energy terms which are derived from the expert-drawn masks, so the 3D contour tends to move towards these contours in the corresponding slices.

3.2 Implicit vs. explicit representation

In the landscape of AC methods, the different variants can be classified into two different groups: explicit, parametric methods and implicit, level-set based methods. Level-set based methods like [Osher and Sethian, 1988] [Malladi

et al., 1995] [Chan and Vese, 2001], and the geodesic method described earlier [Caselles et al., 1997] are generally more computationally involved, because a single implicit function most be optimized over the entire image domain, while explicit methods can iteratively update each discrete curve point. The main benefit level-set methods have, namely the allowance of changing topology (merging/splitting) does not justify their use in context of this research, because changing topologies are not required. So, the question remains whether level-set methods should be researched and considered at all in this context. The original AC algorithm has some big limitations, mainly the need for the initial contour to be very close to the final segmentation (because of the local image gradient energy term). In my system, the expert will only provide a number of slice masks, so a general Volume of Interest (VOI), or a coarse interpolation method can be used as input to the main segmentation algorithm. [Guo et al., 1995] [Vizireanu, 2005] are interesting morphology-based interpolation schemes that could be helpful, but even then, there is no guarantee that the initial rough estimated contour on each slice will be close enough to the final desired result. Because of this fact, investigating implicit level-set methods as well as different explicit extensions on AC can be fruitful: they each have different ways of solving the 'local convergence' problem.
Level-set methods are generally slower than explicit representations. However, there have been a few studies that spend a lot of resources on optimizing these implicit solvers, with positive results. Sometimes, the speed of a level-set algorithm even manages to come very close to algorithms like [Way et al., 2006]. A recurring problem of level-set solvers is the deviation from a perfect distance function caused by discretization. The solvers only approximate an ideal level-set evolution, thereby introducing an increasing error. An ideal implicit function behaves like a signed distance function, meaning its gradient satisfies the eikonal equation (the length of the gradient is 1, everywhere). To fix the error, a number of studies perform 'reinitialization', which basically forces the implicit function to reset in such a way that it satisfies the signed distance constraint. [Peng et al., 1999] [Fedkiw and Osher, 2002] This approach increases the chances of the zero level-set to shift while it evolves, introducing a bias in the segmentation result. It also increases the computational complexity, because of the expensive reinitialization steps. [Li et al., 2005] and [Li et al., 2010] (continued work) solve this problem elegantly by enforcing the level-set to always behave like a signed distance function instead of periodically resetting it. A metric that measures how much the function is like a signed distance function is embedded into the energy functional and derives the entire algorithm. Because no re-initialization is required, larger timesteps can be used, and a relatively (computationally) simple solver using discrete differences. Extending the formulation to three dimensions is an interesting premise that would make this approach applicable for use in context of the new segmentation tool. The amount of iterations the solver requires to converge the contours is very impressive here (typically only a few hundred). However, an extension to three dimensions is not straightforward, requires the gradient flow and every energy metric to be converted to a higher dimensional space, and would significantly increase computation times.

Active contours using explicit representations, like in [Way et al., 2006], is probably a more suitable method in light of the limited computational flexibility of the environment in which the new method is to be implemented. 3D level-set methods can be implemented in such a way that speed is acceptable, such as in [Droske et al., 2001, Street et al., 2007]. The 2001 study [Droske et al., 2001] presents a highly optimized algorithm that evolves a 3D segmentation from a set of user-provided 'seed points' (points within the desired structure). It uses a volumetric, hierarchical grid to simplify the 3D space, and expands the seed points using a technique they call 'adaptive fast marching'. It is a variant of regular fast marching, which is a widely used numerical algorithm to solve level-set problems. While the idea of letting the user only specify a number of points is interesting, and very easy from a user perspective, it has some severe limitations. The user must be able to refine the segmentation, and in a system where only individual points can be provided as input, this is very hard to do. Allowing the specification of an arbitrary amount of correct slice masks is far more convenient. Another limitation of 3D level set methods like this one is their crude approximations (a coarse grid in this case), which sacrifice general segmentation quality. And, compared to other 3D algorithms like [Way et al., 2006], it still is computationally more expensive. Limiting the use of image information is an approach some studies have used to increase the speed of 3D level-set methods. For example, [Heckel et al., 2011] let the user specify desired slice masks in an arbitrary amount of slices, and from different perspectives. This is a very powerful premise in context of the new segmentation tool. The different masks are converted to a point cloud representation, from which a smooth surface is generated using variational interpolation and Radial Basis Functions (RBSF's). While the results are decent (they showed that their method produces segmentations which are, on average, 76 percent similar to fully manual segmentations), not using gradient information from the image dataset causes a lot of fine details to be lost in the final segmentation. The user must provide a lot of slice masks in order to generate a decent segmentation, and the large amount of seed points cause the method to be computationally expensive.
My conclusion about level-set methods in general is that their main advantages are of minor importance in context of the intended new tool (asking users to provide each topological change by hand increases the accuracy of the segmentation and is not too demanding for the user). Their ability to allow for changes in topology is not of vital importance, while their computational demands, even with the many optimizations different authors proposed, still are relatively high compared to similar explicit contour methods. Proposing a novel take on (3D) explicit contour evolution seems like a more viable approach.

3.3 Region-based methods

There are multiple ways to improve AC models by increasing the capture range of image edges, so that the evolving contour is attracted to edges from further away. The simplest idea is to use a Gaussian kernel to smooth the image, thereby stretching the edges. Of course, beyond a certain point, the information

about fine edges is lost, and the segmentation fails. Region-based methods are an interesting approach [Chan and Vese, 2001, Wang et al., 2009, Nguyen et al., 2012] that largely solve this problem, with some (varying) drawbacks unfortunately. Similar to basic AC, an energy minimization is still being performed, but the edge-based energy terms are dropped and replaced by region-based ones. The essence of this approach is: the desired Region of Interest (ROI), or in the case of 3D segmentation Volume of Interest (VOI), is divided into 'outside' and 'inside' points. On both sets, differences in intensity are penalized with large energies. Algorithms that use this idea are not significantly slower then regular AC, and can still provide decent segmentations on all types of images. The approach allows for some convenient user-interaction to make the process more interactive. For example in [Nguyen et al., 2012], the user can provide some 'inside' and 'outside' strokes to refine the final segmentation. This type of interactivity is desirable: there will undeniably be cases where the segmentation result is not good enough, and in those cases, letting the user refine the result without requiring the use of vague parameters or settings is very powerful and something the UMC is definitely interested in. Region-based methods have a general limitation though: because local edge information is dropped and replaced by an aggregate (region), edges tend to follow the image gradient less precisely. However, this effect can be very minimal: [Wang et al., 2009] for example, uses a well-balanced mix of the aggregate image intensity information and local image features, generating segmentations that increase the capture range of structure edges while still maintaining acceptable accuracy. However, they require a level-set representation to model the different energies correctly. As discussed, this representation has a larger computational complexity compared to explicit representations. The ITK-SNAP software (described in [Yushkevich et al., 2006a]), which implements a large amount of cutting-edge (at the time) segmentation algorithms, proves that level-set methods can be entirely justified and optimized to the point of acceptable performance. Using their implementations is very fast on modern machines, even though their solvers use expensive implicit curve representation. Their tools require the user to tune a (relatively) large amount of parameters, and some of their algorithms need a lot of refinement and guidance (by users). Another way of increasing the capture range of edges is by using a technique called Vector Field Convolution [Xu and Prince, 1997, Yu and Bajaj, 2002, Li and Acton, 2007]. First introduced in [Xu and Prince, 1997], this technique drops local edge energy terms and replaces them by a global vector energy field that is pre-computed from the image. The vector field is calculated using the gradient of the image and derivative computations that make sure a vector direction is assigned to each point in the image (even in homogeneous image regions with no gradient whatsoever). Calculating the vector field slows the algorithm down, compared to regular AC, and a possible extension to three dimensions (which would be required if the variant were to be used in context of the new segmentation tool) would be even slower, because of the volumetric differentiation calculations. But in its 2D implementation, it can be used in context of this research as an external force in the AC model that can increase the capture range of the new method.

3.4 Model-based algorithms

Up to this point, only studies that can segment any kind of shape have been presented. This is a desirable quality, because the segmentation tool will mainly be used to segment pathological biological tissue. This kind of tissue does not have a distinct shape, and can instead have any kind of form. However, studying some works that use shape priors, training of a neural network or any other kind of knowledge may provide some valuable insights, even though they cannot be used directly to to the limitations they enforce on the kind of shapes that can be segmented. A lot of studies present variations of segmentation algorithms that first learn some statistical/geometrical data. Some notable ones are [Cootes et al., 1995, 2001, Awad et al., 2007] [Heimann and Meinzer, 2009]. Most of these methods are not user-guided, and provide a full segmentation of the entire 2D/3D image. [Awad et al., 2007] for example segments the entire image into gradient-based segments using an artificial neural net (SOM type) and a genetic algorithm. The neural net is used to extract color features (grouped), and on these features, a selection operator is applied and a population is mutated through different generations using a genetic algorithm. Finally, a segmentation of the input image is obtained. The method is applicable in a context where full segmentations are required, but its large amount of tweaking, dependency on the shape training database, and high computational complexity make it not suitable for application in this research. Analogously, similar methods that use a training database have severe limitations. Sometimes, a statistical model is learned from the training examples that captures recurring shape information and guides the segmentation process. But again, pathological anatomical stuctures do not conform to known shapes. Studies like [Heimann and Meinzer, 2009] do however allow for the specification of very general libraries of shapes. By using basic shapes like 'oval' or 'rectangular', and combining them in a Principle Component Analysis (PCA), a wide range of segmented shapes can be generated. Segmentations of common medical structures, like the heart or the liver, can benefit greatly from such an approach. Often even pathological tissue can be segmented correctly, because of its tendency to be somewhat circular. That said, active contours still is a more usable approach: no shape information is assumed, only the information the user provided, and the information from the dataset itself is used to steer the segmentation in the correct direction. So, the applicability of AC is much more general, allowing teachers to have good results in a more wider spectrum of anatomical scans. General applicability is of vital importance for the intended algorithm.

3.5 Existing segmentation tools

The most modern, up-to-date and widely used existing system that focuses on user-guided active contour segmentation is from 2006 and is called ITK-SNAP Yushkevich et al. [2006b]. My research emphasizes the need for a practical, easy to use implementation that teachers and doctors around the world can use without having to go through extensive configuration and understanding of the

underlying algorithms. A reflective validation study Yushkevich et al. [2006b] comparing hand-provided slice contours and the SNAP guided method shows that SNAP provides great accuracy, comparable even to expert-drawn 3D contours. Their method sometimes even outperforms fully manual 3D contours. This is because when providing a manual contour, the user typically draws the contours in a single perspective, progressive going through each slice. From another perspective, the contour then appears jagged (staircase-artifact), which is not desirable. With a semi-automatic method like SNAP, the generated contour typically looks a lot smoother, with some loss of accuracy. Still, ITK-SNAP is not being considered for application in medical e-learning. Its main problem is that ITK-SNAP is difficult to understand, and requires a number of parameters and settings to be changed in order to have good results (it is open-source, so anyone can download the package at itksnap.org). Furthermore, most of the tooling is based on the user providing a target region of interest, and then grows this region to conform to local edges. For the intended segmentation tool, the amount of parameters need to be very small, and the input needs to be a number of fully hand-drawn image slice contours. There should be no learning curve: teachers should immediately grasp the idea and start making segmentations.

MITK is another modern segmentation tool that has implementations of a number of state-of-the-art segmentation algorithms and is widely used. Its interface is relatively difficult to understand, again because of an extensive amount of options and parameter settings. It is often used in situations where there is (a lot of) domain knowledge from users. The e-learning environment I'm working on is targeted at radiology teachers who generally do not have much knowledge about segmentation algorithms, so the tooling can be as simple and straightforward as possible. The new segmentation method only needs to have a single use-case: a user provides a number of slices and the system fills in the rest, so this allows for much simplicity in its interface. MITK, just like ITK-SNAP, is a bit too general and provides too many options, so the UMC stated that a tool that is as simple as possible is very desired.

These are the two largest image segmentation tools in circulation right now, with a lot of small tools that are not used in e-learning environments existing as well, IntSeg 3D for example, a surface-based segmentation tool. But an e-learning system that only provides a single, very simple segmentation tool does not publicly exist. The UMC uses MITK in conjunction with their local e-learning, causing teachers to most often just specify each slice by hand, overwhelmed as they are by the different options and parameters.

3.6 Conclusions

A new segmentation tool that is tightly integrated with a new e-learning system, using a new variant of active contours, is a powerful combination. No segmentation tool has been created with this specific purpose in mind, and no algorithm allows for the specification of a number of slice masks, filling in the rest of the

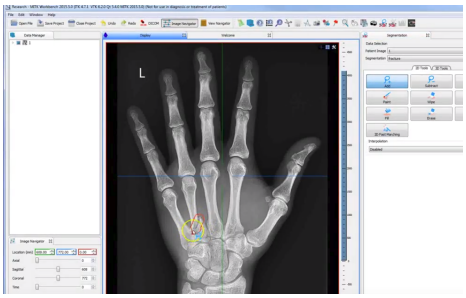

Figure 1: MITK has a lot of options. Currently being used for VQuest at the UMC

segmentation automatically and without setting parameters. By adapting 2D active contours to include forces derived from user-provided masks, running AC for each slice that needs to be generated, and incorporating the method into a very simple, easy to use segmentation tool, I will have a concrete contribution that has immediate practical applications.

4 Background

4.1 Active Contours: The basics

Active Contour Models were introduced in 1988 by Kass et al. [1988]. In essence, a so-called *snake* is represented explicitly using a parametric formulation (there are, as explored in the literature review, other representations, but the explicit representation is used here). We consider the 2D version first. Its basic definition is:

$$v(s) = (x(s), y(s)), s \in [0, 1] \tag{1}$$

An energy functional assigns a certain energy value to the curve function. The core idea is then to try to minimize this curve energy by morphing the parameterized curve into different shapes. This is where the curve gets its conventional name: a *snake*, because it 'slithers' into different configurations while the solver tries to find an optimal solution. A general snake energy functional has the following form:

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))\ ds \tag{2}$$

In this equation, three types of energy are represented: internal energy, image-derived energy and external constraint energy.

- The internal energy is derived from the shape of the curve itself, and causes the energy minimization procedure to favor shapes that are more compact

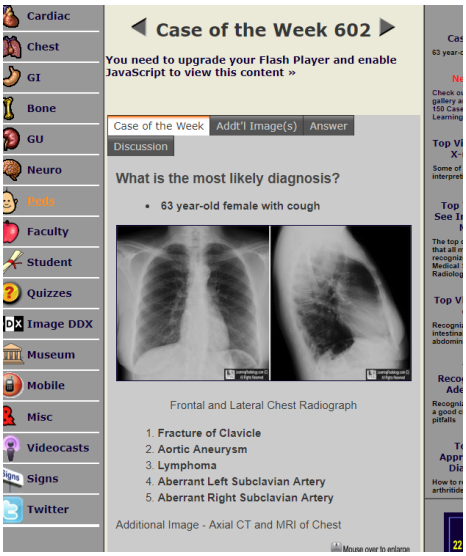

Figure 2: A lot of 2D-only e-learning is still being used in web-based situations

and round. This is achieved by taking derivatives of the curve function: the first derivative 'awards' compact shapes without many twists/turns and the second derivative enforces the curve to behave like a *thin plate*, meaning it does not contain sharp corners. Most often, a weighted average is used between these two curve derivatives:

$$E_{int} = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)/2 \tag{3}$$

Suitable values for α and β can be discovered empirically, and depend on the context of application. This research will be using very low values for them, around 0.05 (see results section).

- There are various ways to represent the second term in equation (2): the image energy. Typically, the most important information that is extracted from the image (which is represented as a grid of color or intensity values, depending on the context) is the *gradient*, defined by $\nabla I(x, y)$, with I being the image represented as an intensity map. By letting the snake be attracted to large gradient value, it is attracted to level sets in the image. The original Kass et al. [1988] study uses three different image terms: the line, edge and termination functionals. Then, by using a weighted combination of the three terms, the final image energy can be represented:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{4}$$

Again, the weights in this formula need to be set to fixed values using an empirical examination (The algorithmic details and parameters are

devised in such a way that the resulting converged curves have general acceptable quality). The definitions for the different subterms in the above equation are:

- The line functional (E_{line}), this can simply be the intensity of the image: $I(x, y)$. Remember that in equation (2), the integral of the energy over the entire curve domain is used. The solver will try to minimize this integrated energy, meaning that for the intensity term, the curve will favor darker parts of the image. That is, unless a negative weight is used in equation (4), in that case the lighter parts of the image will be favored.
- The edge functional (E_{edge}), this can simply be set to the negative gradient of the image like this: $E_{edge} = -|\nabla I(x, y)|^2$. As discussed in the literature review, this has some limitations, mainly the local convergence problem: the gradient will have a large spike at image edges, so this functional will only have an effect when the curve is already close to an edge. Different extensions and improvements on the different energy functionals will be discussed in the next subsection, here we stick to the basics.
- The last part of equation (4) (E_{term}) is the termination functional. This one is slightly more involved, and will cause the curve to be attracted to terminations of line segments and (sharp) corners. The idea is to first smooth the image somewhat using a gaussian kernel: $C(x, y) = G_\sigma(x, y) \cdot I(x, y)$, and then calculate the gradient angle of the image, $\theta = tan^{-1}(C_y/C_x)$, and the normal vector that is perpendicular to the gradient direction: $n_\perp = (-\sin\theta, \cos\theta)$. Then, by taking the derivative of the θ function and dividing it by the derivative of the perpendicular normal function ($\frac{\partial\theta}{\partial n_\perp}$), an energy term is created that minimizes at sharp corners and line terminations in the image.
  This concludes the basic mathematical formulation of active contours, that models a curve which follows lines and edges, is drawn to terminations, and provides decent results for basic images (see 4.1 for an example of this basic model in action).

With the modeling of these three types of energy, a snake is created that tries to maintain its general shape, and conforms to areas in the image, and even terminations like lines. See figure 3 for an example of the original snake model: the curve conforms nicely to lines and blobs in the image.

4.2 Constraints and extensions

There are many ways of extending the standard active contours model to improve its performance. The original model has general applicability (on all sorts of image data), just like the method for this research needs to have, but some extensions have been created for a specific domain and increase performance

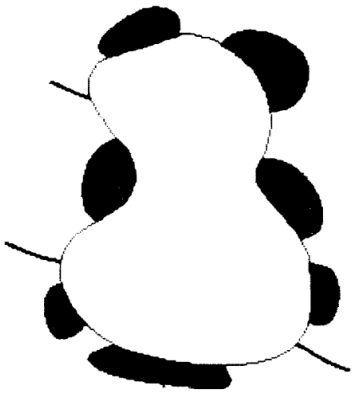

Figure 3: The original Active Contours model in action [Kass et al., 1988]. The black blobs and lines are image parts, the thin black line is the converged contour curve. The contour gets attracted to edges and termination points, as can be seen here.

only for specific types of shapes/images. [Chen et al., 2002] presents such an extension, that embeds the use of shape priors into the energy equation.
An extension that has general applicability and is used in this research, was first introduced in [Li and Acton, 2007]. They call it *Vector Field Convolution*, and it solves the local convergence problem for a large part. Furthermore, it improves the sensitivity of the contours to concavities, making them better conform to rough, concave shapes in the image. The *Vector Field Convolution* technique is used as a so-called *static* energy, meaning it does not dynamically depend on the position of the curve but is a single, static vector field that is generated purely using the image data once. This static field is calculated by using discrete convolution: the edge map of the image is convolved with the *vector field kernel*, which is defined like this: $k(x, y) = [uk(x, y), vk(x, y)]$. In this kernel, all vectors point towards the kernel origin. This way, a free particle that is placed in the field will move toward the origin of the kernel.

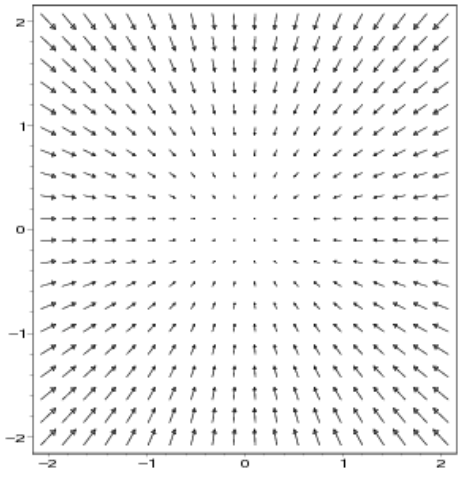

Figure 4: A Vector Field Kernel with all vectors pointing towards the origin (Source: Cambridge University Press)

And, when the kernel is convoluted over the edge map of the image, the vectors will point towards nearby edges, since the contribution of the edge pixels to the field will be significantly higher (edge pixels have larger image values). To enforce a gradient that gets weaker as the distance to strong edges increases (declining gradient), a decreasing positive function is multiplied with the vector kernel: $k(x, y) = m(x, y)n(x, y)$, in which $m(x, y)$ is a linear function, which can be $m(x, y) = (r + \epsilon)^{-\gamma}$ (the ϵ and γ are determined empirically). This increases the capture range of edges, while still keeping their area of influence limited.

4.3 Discretization

The equations of active contours map curve functions to concrete energy values. Aim is to find curve functions that have minimal energy values (by morphing the curve shape), and to do this, a mathematical concept called the *Euler-Langrange* formulation can be used. As an example, the Euler-Langrange formulation of equation (2) and (3) has the following form:

$$0 = \alpha \cdot v'' - \beta \cdot v'''' + \nabla E_{image} \tag{5}$$

This basically states that the image forces balance the internal, derivative based forces. In order to solve such formulations in practice, a mapping to a discrete representation must take place, after which an approximate gradient solver will iteratively optimize the target contour until an equilibrium is reached, and the Euler-Langrange equation is satisfied. The most convenient way to represent a 2D contour is by a polygon represented as a vector, in which each individual

element represents a 2D point on the curve, and the vector as a whole is a route through all successive curve points. In the 3D case, the contour can be stored in a matrix in which each row represents a vertical slice. Once this discrete representation is obtained, numerical methods are used to approximately calculate the various forces (internal/image) at each iteration, and update the curve points accordingly. When the α and β parameters are allowed to be different for each curve point, as is typically the case, the discrete Euler formulation of equation (6) has the following form (here, v is a finite vector of curve points):

$$\begin{aligned} 0 =& \alpha_i(v_i - v_{i-1}) - \alpha_{i+1}(v_{i+1} - v_i) + \beta_{i-1}(v_{i-2} - 2v_{i-1} + v_i) \\ & - 2\beta_i(v_{i-1} - 2v_i + v_{i+1}) + \beta_{i+1}(v_i - 2v_{i+1} + v_{i+2}) \\ & + (E_{imagex}(i), E_{imagey}(i)) \end{aligned} \tag{6}$$

Note that we use a closed curve here, so if index $i - 1$ is negative, the index wraps to the other end of the curve vector. In the 3D case, instead of differences between neighboring curve points, a volumetric neighborhood can be used to approximate the derivatives for each curve point. Of course, the image forces will then need to be defined in three dimensions as well. To solve equation (7), all these values can be put in matrix form:

$$Ax + E_{imagex}(x_i, y_i) = 0 \tag{7}$$

$$Ay + E_{imagey}(x_i, y_i) = 0 \tag{8}$$

These equations can be solved by matrix inversion like this (with introduction of timestep t and timestep size γ):

$$x_t = (A + \gamma I)^{-1}(x_i t - 1 - E_{imagex}(x_i t - 1, y_i t - 1)) \tag{9}$$

$$y_t = (A + \gamma I)^{-1}(y_i t - 1 - E_{imagey}(x_i t - 1, y_i t - 1)) \tag{10}$$

The vector of curve x positions is calculated separately from the curve y positions each iteration. When the curve reaches a minimum (equilibrium), the time derivative vanishes as the updated positions converge. During the exploration of the method in the next chapter, an explicit Euler solver will be used.

5 The Method

In this chapter, we provide a thorough description of the segmentation method, breaking down its individual components and how they work together to generate the final segmentations. The method operates and is developed in parallel with the creation of a new e-learning system for the Utrecht Medical Center. First, a global overview of the structure of this system is provided, which is the

context in which the method is developed, refined and tested. After this, the Active Contours model that lies at the heart of the method is described. This AC model consists of a number of relatively straightforward forces, and a few more intricate ones, namely the Gradient Vector Field (GVF) force, and the 2.5D Shape Morphing force. They are described in their respective subsections.

5.1 System Infrastructure

The development of the new e-learning software and the specification and refinement of the new method have been tightly intertwined from the start. The software provides a convenient framework, and a niche context against which the method is specialized and adjusted. The idea of the software pilot is that for the first time, teachers and students (of the UMC) can create and access a library of radiology cases from anywhere, scroll through 3D datasets that are highly optimized for use in browsers, and answer medical questions about them. Teachers can create full segmentations, use the auto-complete function that lies at the heart of this thesis, and upload 3D scans, all within a normal browser. This means that the method operates in a client-server architecture. An abstract flow of segmentation operations is provided in figure 5. Generally, a user draws an incomplete segmentation on the client-side, and sends this data over to the server. The server considers all user-specified segmentation slices fixed, and generates the remaining inbetween slices, which are send back to the client. After some potential refinement, the original user-specified slices along with the refined generated ones are again send to the server, which repeats the process of filling in missing slices. This iteration continues until the user is satisfied with the result. This system overview is all contextual information that provides an explanation and justification for some of the design decisions made throughout the project, but keep in mind that the segmentation method can potentially be decoupled from this context, should this be desired, and implemented for use in another application.

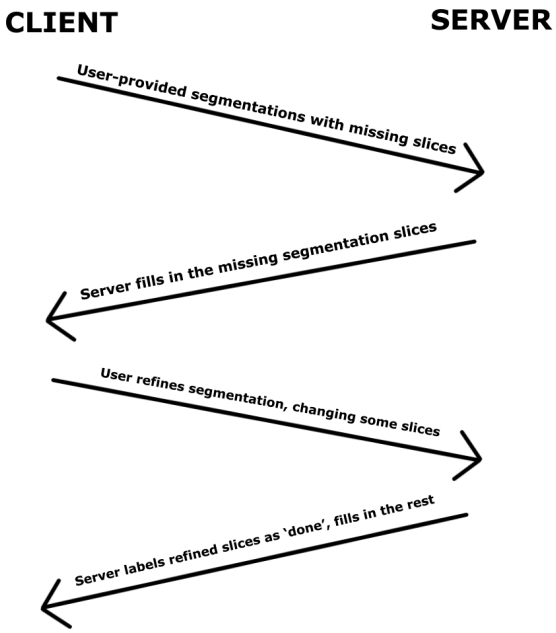

Figure 5: Implementation context of the method: Client/Server architecture

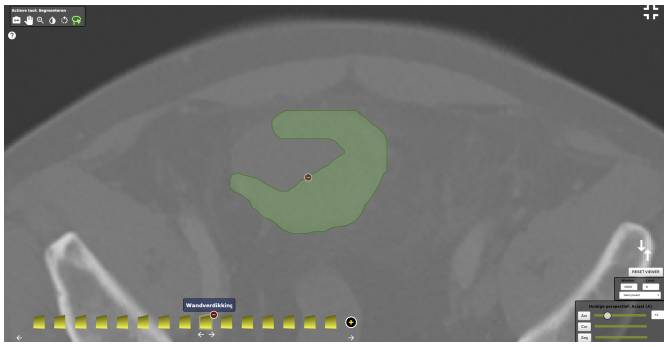

Figure 6: Interface in which users create their segmentations

All scan data is stored on the server, and send to clients in compressed form to optimize loading speeds. Segmentations are send back and forward between client and server, and are annotated with essential metadata. Each segmentation slice can contain one or more contours. Each contour is represented as a list of 2D points in the coordinate system of the scan. Each segmentation slice is annotated with the name of the scan, the perspective (axial, coronal, sagittal), and the slice index. This data representation of a segmentation has a very small

footprint (typically only a few KB), making sure that transmission times are minimal, even in low-bandwidth situations. In the remaining of the chapter, a few terms will be used that require some explanation. When a 'contour' is mentioned, this means a single, closed 2D curve. When a 'slice segmentation' is mentioned, this means one or more of these contours presented in a single 2D coordinate system (typically, an image slice). When a 'segmentation' is mentioned, this means on or more slice segmentations, which can be spread across multiple image slices or even scan perspectives.

5.2 Active Contours Model

The core of the method is based on the classic Active Contours model by Kass et al. [1988]. As described in the background chapter, this model defines an energy equation that consists of a number of terms. The model operates on a single, closed, two-dimensional contour, which is presented parametrically by $v(t) = (x(t), y(t))$, where $t \in [0, 1]$. The energy is integrated over the entire contour:

$$E_{snake} = \int_0^1 E_{int}(v(t)) + E_{ext}(v(t))\ ds \tag{11}$$

E_{int} and E_{ext} stand for the internal and external energy components respectively. The internal and external energies are derived by forces that are defined for each point on the contour. The solution state of Active Contours is a curve configuration for which the total integrated energy over the full curve is minimized. In this method, there are four different internal forces: **Elasticity**, **Resistance to curvature**, **Expansion**, and **Stiffness**:

$$E_{int} = (p_1 \cdot v_t(t)^2 + p_2 \cdot v_{tt}(t)^2 + p_3 * E_{exp} + p_4 * E_{stiff})/4 \tag{12}$$

These are included in almost every variant of the Active Contours model. They are as follows:

- Elasticity provides a force that resists stretching and is defined by the first derivative of the curve ($v_t(t)$).
- Resistance to curvature provides a force that resists bending (sudden changes in curvature), and is defined by the second derivative of the curve ($v_{tt}(t)$).
- Expansion provides a force that grows the entire contour. It is defined on each point on the curve, by using the normal vector as a force. It prevents the curve from imploding too much.
- Stiffness provides a 'return home' force: before starting the Active Contours evolution process, a snapshot of the initial curve is recorded, so that on each iteration and for each point on the curve, a force vector can be

calculated that points towards the initial position of that point on the curve.

The four parameters (p_1 to p_4) determine the relative strength of these energies, and will be determined empirically (see the result chapter for the analysis) In addition to these internal forces, two external forces are defined, which are refined and implemented in a specific and new way. The first one is a **Gradient Vector Field force** (GVF), which is derived from the scan dataset. It is a vector field that guides the curve towards scan areas of high contrast (edges), and gradually increases in strength as the curve gets closer to edges in the image. The scan, perspective and slice to load from disk is provided in the metadata that is send along with each slice segmentation. These vector fields cannot be cached for each dataset (to reduce calculation times), because they are dependent on the window/level settings of the user (the user can adjust scan window/level to increase contrast in desired anatomical shapes). Therefore, this calculation can quickly become a bottleneck of the entire system. To optimize it, an implementation is chosen that is very fast and applicable for any 2D image, based on a study by [Han et al., 2007]. A subsection about the calculation of this vector field will follow. The other external force is a **2D shape morphing force**. An important constraint of the entire method is that any missing segmentation slice (on which Active Contours needs to run), has two nearest user-created segmentation slices, one in both directions (on a higher and a lower slice number). This constraint is enforced by only letting the system generate slices between user-provided slices (as opposed to all the slices in the target perspective). The idea is to match the closest contours on these two slices, and morph them into each other. This morphing is done parametrically ($t \in [0, 1]$) and linearly: 0.5 represent a shape that is halfway inbetween the two boundary shapes:

$$S = u_1 \cdot (1 - t) + u_2 \cdot t \tag{13}$$

S is the generated contour, u_1 is the user-provided contour in one direction, u_2 the contour in the other direction. Using this morpher, reference shapes can be generated that are used by Active Contours as an external attraction force (by taking the normalized distance between the contour and the reference shape, on each curve point). Details about the generation of these morphed shapes are also presented in a subsection later this chapter.
To discretize the Active Contours model, each input curve is represented as an array of curve points, for which the different forces are calculated. During a few hundred iterations, the positions of these curve points are updated, after which new forces are calculated. Depending on how many evolution iterations are used, and the different parameters that derive the relative power of different forces, the system reaches an equilibrium or not. In many cases cases it won't, but this is not a problem: reaching an equilibrium is not a constraint for a good looking result. So, to provide an overview of a single Active Contours run, here is some pseudo-code:

```
Data: List of 2D points that form a closed contour; Morphed mask
      target (list of 2D points)
Result: Discretely iterated, evolved new contour; a list of 2D points
        with the same length
Calculate Gradient Vector Field;
for Iteration count (constant) do
    for Each point in the input list of points) do
        Calculate Curvature Force ;
        Calculate Elasticity Force ;
        Calculate Expansion Force ;
        Calculate Stiffness Force ;
        Calculate Image Force (using the GVF) ;
        Calculate Morphed Mask Force (using the morphed mask
         target);
    end
    for Each point in the input list of points) do
        Update position of point by applying all the forces, multiplied by
         their force multipliers (constant parameters);
    end
end
```

Algorithm 1: A single run of the Active Contours implementation

The algorithm will be repeatedly executed for each slice that needs to be generated. Each slice that needs to be generated is guaranteed to lie between two user-provided slices (see Section 5.3 for further details). The input of the algorithm is an initial contour guess, and a morphed mask target, which is generated from the shape morpher, that blends these two user-provided slices into each other.

First, the algorithm calculates the Gradient Vector Field, using the correct image data. This step is placed outside the loops, because this vector field is static: it does not change as the contour evolves. Next, a loop is started that repeats the inner steps 'iteration count' times: this is a parametric constant. Then, for each point on the current contour, all the different forces are calculated, after which the positions of the contour vertices are updated. Notice that the forces are calculated in a separate loop: this is because the Curvature, Elasticity and Expansion forces use information about neighboring vertices, so they cannot be updated together with the force calculation. After the algorithm is finished, it returns back to the main segmentation algorithm that will be described later.

5.2.1 Fast Gradient Vector Field Calculation

This implementation is based on this study: Han et al. [2007]. The input of the algorithm is the image from which the GVF needs to be calculated. Here, it is pre-processed with the user-specified scan window and level. To optimize running speed, the bounding rectangle around the contour that Active Contours uses as input is used to cut out a portion of the scan image, with a 100 pixel margin to increase convergence accuracy and capture range of the GVF calcu-

lator. First, an edge map is generated from this pre-processed image rectangle. Here, this is done using the well-known Canny Edge detector. For images that can differ largely in contrast, details and scale, this detector provides the most consistent and reliable results. From the resulting edge map, a horizontal and vertical derivative map is extracted using convolution with Sobel kernels (x and y). This is done because the GVF algorithm generates the X-component of the output GVF separately from Y-component. The dimensions of the output GVF is the same as the original image region ($width \cdot height$)
The calculation of the gradient vector X and Y maps is done using a multigrid solver. This entire pipeline is presented visually in figure 7.

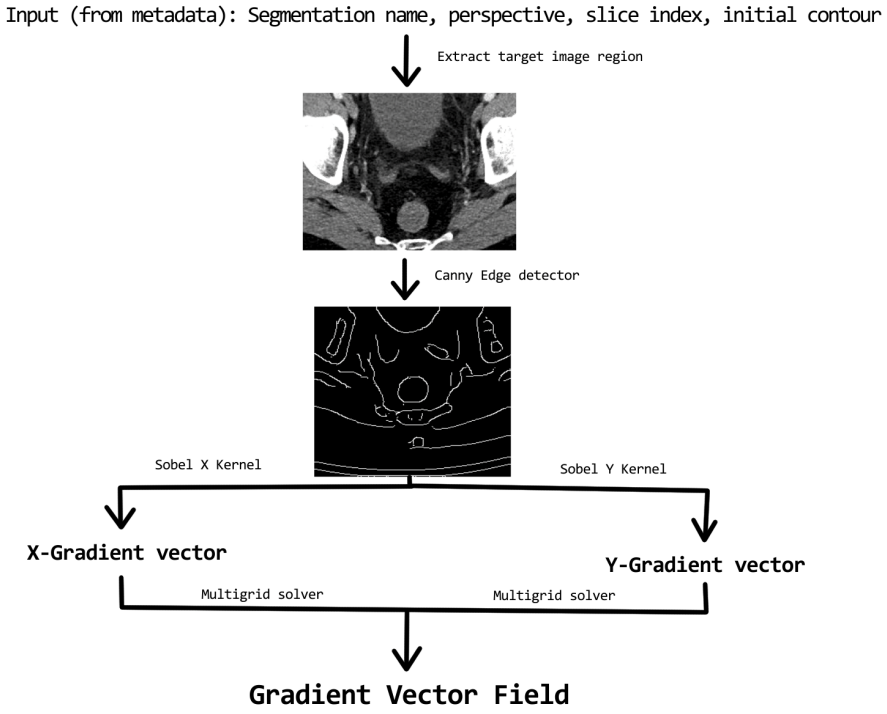

Figure 7: The pipeline of the GVF algorithm, presented visually

During the extraction of the image region, the 16 bit scan is processed using the Window/Level settings specified by the user, and normalized. During the segmentation process, the user can impact the resulting generated segmentation by actively Window/Leveling the dataset. The effect this can have on the final Gradient Vector Field is illustrated in figure 8. Note that the active contour is influenced by this GVF, so changing the Window/Level can have a visible effect on the final segmentation.

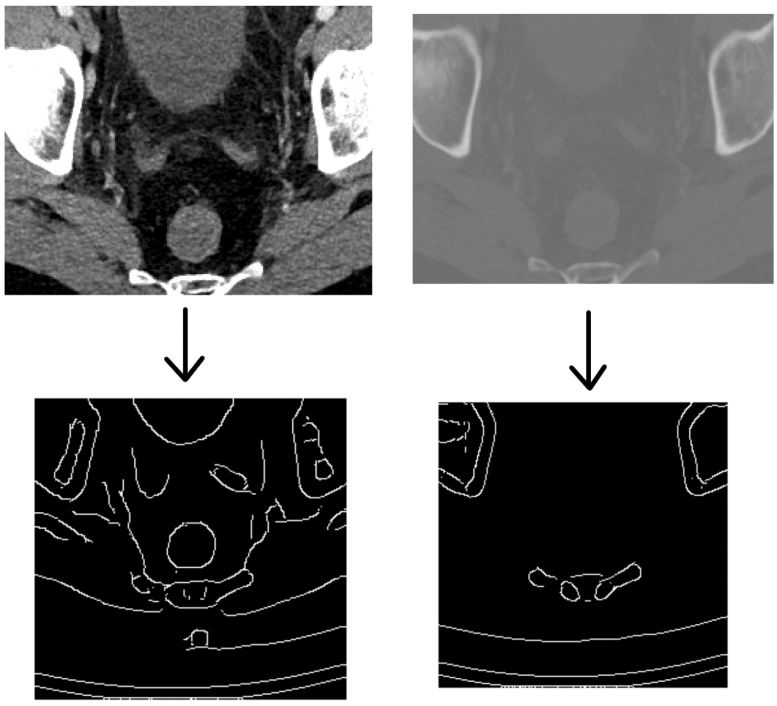

Figure 8: The impact using different Window/Level settings can have on the resulting edge map, and therefore on the final GVF.

The difficult part of this algorithm lies in the implementation of the Multigrid solver, that takes the image gradient and outputs the converged GVF. As described in the study by X. Han et al., this solver finds an equilibrium solution to the following equation:

$$0 = \bigtriangledown(g(F) \bigtriangledown V) - p(F)(V - F) \tag{14}$$

In this equation, F is the input edge map (x and y gradients combined into vectors), V is the desired output (the Gradient Vector Field), and p and g are two weighting functions that determine the region of influence of edges, and how close the GVF is to the input F:

$$g(F) = e^{-(F/K)^2} \tag{15}$$

(K is a smoothing constant that determines the falloff of the edges in the GVF. For this research, K is set to 4 as a result of empirical experimentation)

$$p(F) = 1 - g(F) \tag{16}$$

As mentioned earlier, the GVF is solved in the X and Y dimension separately. This is done by forming two PDEs (partial differential equations) from equation 14 of the following form:

$$0 = \bigtriangledown(g(x,y) \bigtriangledown u(x,y)) - p(x,y)u(x,y) - r(x,y) \tag{17}$$

Here, u is the component of the vector field that needs to be solved (x or y). This equation can now be solved for the X and Y dimension. This is done by using different so-called 'grid levels' of the input image. At the finest grid level, its dimensions are the same as the input image region. Both dimensions are divided by 2 to obtain a smaller grid level, and the image values are restricted to the new grid using a restriction operator (average of neighbors). This is repeated until, at the coarsest level, the grid is only 2x2. The multigrid method repeatedly 'relaxes' the error of the target PDE on each level of the grid, by averaging with neighboring grid elements. To transfer the values back to a finer grid level, a prolongation operator is used (simply copies each value to neighboring cells to obtain the finer grid). By relaxing the error on multiple grid levels, finer parts as well as more global (smooth) parts of the error function are minimized. The algorithm starts with an initial guess, which is set to be the zero vector, and a desired result, which is set to be the input gradient vector. The target PDE is a discretization of the differentiable part of (17), and looks like this:

$$\begin{aligned}
&\bigtriangledown\,(g(x,y) \bigtriangledown u(x,y) = \\
&\frac{(g[i+0.5,j] \cdot (u[i+1,j] - u[i,j])) - (-g[i-0.5,j] \cdot (u[i,j] - u[i-1,j]))}{s} \\
&+ \frac{(g[i,j+0.5] \cdot (u[i,j+1] - u[i,j])) - (-g[i,j-0.5] \cdot (u[i,j] - u[i,j-1]))}{s}
\end{aligned} \tag{18}$$

s is the spacing of the current grid. To improve upon the current guess u, an correction vector term is added to the current guess. This is set to be the difference between the current guess, and the relaxed, discretized PDE: $diff = u - PDE$.
These corrections are done repeatedly, on all grid levels, as to correct for finer and more coarse errors in the resulting function. For more (implementation) details about multigrid solvers, see [Han et al., 2007].

Because the amount of grid levels is logarithmic in the input image size, and the algorithm only needs to relax the error a few times on each level, this algorithm significantly increases the speed compared to a simple, iterative GVF solver that simply repeatedly convolves the edge image with a 2D kernel until a GVF is obtained. Speed that is very desired for this application: the user needs to be able to make relatively quick edits to a segmentation and see the results as responsively as possible.

Before the calculated GVF is sent back to the Active contours model, it is processed in the following way: each vector (corresponding to a pixel in the input) that has a magnitude greater than one, is normalized. This makes sure that 1) the field does not contain outliers that are too big, thereby having too much effect on the contour, and 2) the effect of edges on the field degrades as the distance to the edges increases. Both properties are desirable for the model this research aims to create.

5.2.2 2.5D Shape Morphing Force

The algorithm that generates a morphing sequence between two user-provided, closed 2D contours, represented as a list of points, is based on a study Yang and Feng [2009]. This study described a generally applicable method to morph two arbitrary shapes into each other based on matching features and an as-rigid-as-possible interpolation. After experimentation and implementation of the system as the study describes, it became clear that it presented some problems in the context of this research, the main problem being that the matching algorithm that links parts of the two respective contours to each other, often generated crossing lines and invalid links. The way in which the user-provided contours are drawn helps to shed light on this issue: about every 5th slice is user-provided, and because anatomical structures are fixed (don't move as the slice number increases), the correct matching of the contour vertices is local instead of based on shape features. With this in mind, and the observation that using an as-rigid-as-possible interpolation often favors the rigidity constraint over an evenly spread interpolation (creating unwanted growing and shrinking of the interpolated shape), we created the following shape morphing algorithm:

- Because both contours are densely sampled, for each contour, take every 15th vertex and label every group of inbetween vertices as 'features' of the shape.
- Use a recursive, dynamic programming algorithm to create a desired sequential, one-to-one matching between the features of both contours. If one contour contains more features, features will be grouped in order to create a one-to-one correspondence. The dynamic algorithm assigns a distance cost to each matching between the two contours. It sequentially goes through all the feature points of the contour with the smallest amount of feature points, and tries linking them with the feature points of the other contour, always taking the minimum cost of using versus not using each particular link option. By caching correspondence costs, the algorithm becomes dynamic and provides a matching very fast (typically in around half a second, on an Intel I7-4790K). The algorithm is very similar to, and inspired by, existing minimal cost dynamic algorithms.
- Every pair of matched features is resampled so they contain the same number of vertices (by linear interpolation). Now, the morphing sequence

can be generated. Given an interpolation factor ($0 <= t <= 1$), the two contours are morphed into each other by linear interpolation: every matched feature now contains the same number of vertices whose locations can be interpolated.

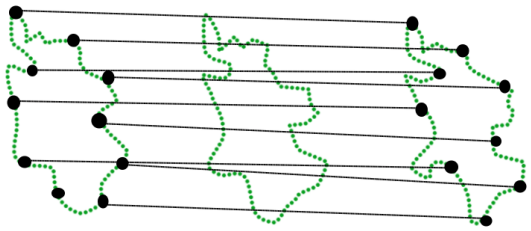

Figure 9: The two contours are split into features (black dots), and a best match is generated using dynamic programming. Then, an inbetween morphing sequence is created (here t=0.5)

The matching between the two contours is cached, so that a morphed shape can quickly be calculated for each interpolation factor t.

5.3 Running in 2.5D

The system runs for each missing slice. Between every two user-provided slices, a morphing sequence is generated. The resulting morphed masks are send along with each active contours run. The first user-provided contour is copied to the next slice, on which AC runs (using the morphed mask as input parameter). The result of this run is then copied over to the next slice, after which the process repeats until the next user-provided slice is reached. If there are still slices missing, a new morphing sequence is generated, and the entire process is repeated. See figure 10 for an overview of this pipeline.

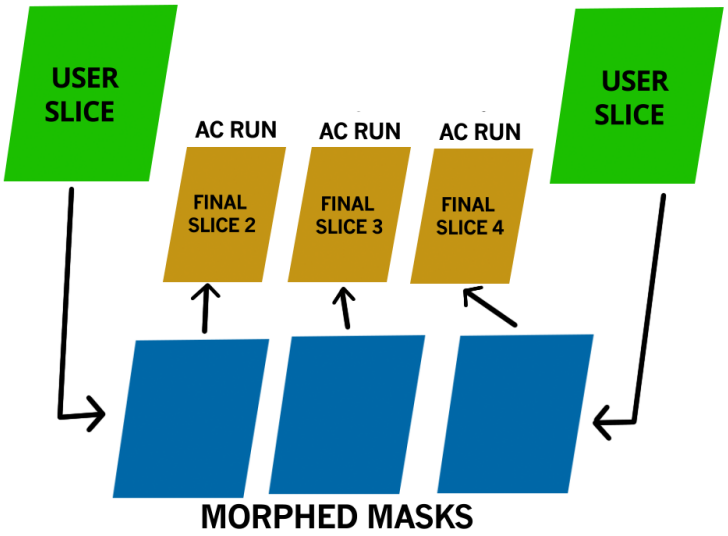

Figure 10: The pipeline of the segmentation system. Between every two user-provided slices, a morphing sequence is generated. A number of discrete interpolated shapes is fed into each corresponding inbetween scan slice, on which, together with the calculated Gradient Vector Field for that slice, and the other, internal forces and parameters, Active Contours runs to provide the final contour to return to the user.

6 Results

The system relies heavily on an acceptable tuning of the different parameters, where 'acceptable' means a configuration that generally provides the best results across the domain of 3D medical scans. This tuning process is of vital importance, mainly because all of the parameters are hidden from the user in the final product. This section describes the structured approach that led to a fixed choice for the parameters, with a side note: this final choice cannot be the absolute best one. There is a huge parameter space to explore, because each variable is unrestricted and may contain an arbitrary amount of decimal digits, and also because there is an inherent semantic aspect to the segmentation of scans. The active contours model does not have this semantic domain knowledge, and bases its guess entirely on a number of concrete forces. Therefore, expert knowledge has to be incorporated into the reflection process. This is done objectively and subjectively. A radiology expert provides a number of hand-drawn segmentations which are compared (using a distance measure) to an auto-completed version of the same segmentation. This provides an objective accuracy measure. The system is also evaluated subjectively, within the e-learning context it is designed to work in. A few expert radiologists from the Utrecht Medical Center judge the accuracy, ease of use and overall usability of the system, and their findings are included later in this section. The performance and applicability of the method are also examined in different subsections.

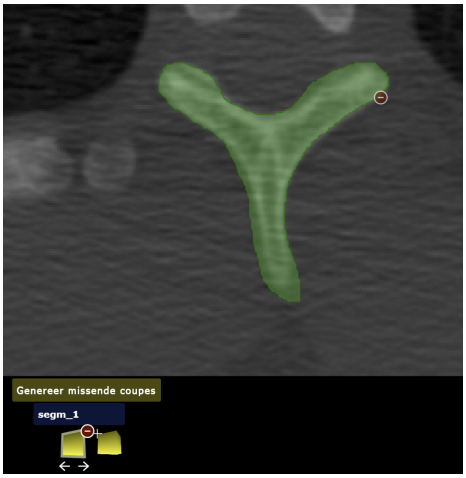

Figure 11: The software interface (the large yellowish button generates the missing slices)

6.1 Parameter exploration

To streamline the process of choosing parameter values, a development build of the online environment is used that allows for the manual tweaking of each parameter, and visualization of the forces that lead to the final shape of a segmentation.

In Section 5, the steps, forces and formulations of the algorithm are described in detail. It is there for a better understanding of the parameters and how they drive and prioritize the different subsystems. Here, the parameters are simply presented one by one, after which they will be tweaked individually and together. The parameters of the active contours model are the following:

- **Elastic force multiplier:** this sets the strength of the resistance to stretch force, meaning the larger this value is, the less the 2D contour is able to stretch in different directions.
- **Curvature force multiplier:** this sets the strength of the resistance to bending force, meaning the larger this value is, the smaller the integration of the curvature over the curve will become.
- **Expansion force multiplier:** this sets the strength of the tendency of the curve to expand. It is essentially the opposite of the elastic force, a correct trade-off between these two forces is important.
- **Image force multiplier:** this controls the strength of the force derived from the gradient vector field of the frame image (the parameters internal to the GVF subsystem have been set to their optimal values for this application in an earlier stage. For details, see the 'Method' section.
- **Morphed mask force multiplier:** for each missing slice that lies in between two user-provided slices, a morphed shape is generated using the algorithm described in the 'Method' section. This parameter controls the tendency of the active contour to move towards this morphed shape.
- **Stiffness force multiplier:** this determines the resistance to any change from the mask of the previous slice. (Recall that active contours successively runs on each missing slice in ascending order).
- **Number of iterations:** this is simply the number of position updates / force recalculations each run of active contours consists of. The theoretical ideal situation is to let the curve iterate until the different forces converge to an equilibrium, but for this practical exploration, smaller iteration counts are also considered (which generate a pre-converge state as the final curve).
- **Post-smooth:** A parameter that sets the amount of smoothing (laplacian) that needs to take place, after active contours runs.

The different parameters can influence each other in a lot of different ways. For example, if the image parameter is set so that it has an ever so slightly stronger effect than the morphed mask strength on a specific slice, this can potentially cascade to a larger domination of the image force over the course of many iterations. The two forces that have the largest contribution to the overall shape of the iterated contour are the image force and the morphed mask force. These forces steer the contour toward the desired configuration, while the other forces only refine it by smoothing and making small local adjustments. Therefore, the strength of these shape-defining forces should be relatively higher compared to the other three. In order to test the objective accuracy of each choice of parameters, a number of different reference segmentations are used. These are provided by radiology professors at the UMC, and represent the 'ground truth' the algorithm needs to approach as closely as possible. This is compared against a version where part of the slices are left out, and the rest automatically generated. The distance of the generated segmentation to the reference one, is used as an objective accuracy measure. By experimenting with different combinations of parameters, the overall average accuracy is minimized. This distance value is calculated in the following way: for each slice in the segmentation, the union of the generated mask and the reference (ground-truth) mask is taken, and divided by the intersection of these same two masks. These values are summed over all slices of the segmentation, normalized, and the resulting value should be as close to 1 as possible (ideally, it should be exactly 1, when the generated segmentation and reference segmentation are exactly the same). So, the accuracy value then is the absolute difference between 1 and the calculated distance value.
Before the parameters can be optimized, a suitable starting configuration needs to be chosen. Recall that all force vectors are normalized during the active contours process, so the parameters do not have a unit of measurement and their relative relations are important. The starting configuration for the system is based on some trial and error: the parameters are quickly tweaked until the generated segmentations look somewhat acceptable, after which the fine-tuning can begin. The initial configuration of parameters is the following:

- Elastic force: 0.15
- Curvature force: 0.2
- Expansion force: 0.25
- Image force: 0.2
- Morphed mask force: 0.2
- Number of iterations: 250
- Stiffness force: 0.1
- Smooth strength: 0.1

6.1.1 Parameter: Curvature force

Using the analysis method described above, an objective accuracy measure can be done on any of the segmentations provided by the hospital, using any configuration of parameters. The first parameter that will be tweaked is the curvature force multiplier. This parameter is not expected to have a large impact on the shape and therefore the accuracy of generated segmentations, and is there to limit large spikes / changes in the overall shape of a segmentation. By zeroing out all the other forces of the active contours model, the contribution of this force can be seen visually. Figure 12 shows this contribution (recall that generated segmentation slices always lie inbetween user-provided segmentation slices, which are green in this image):

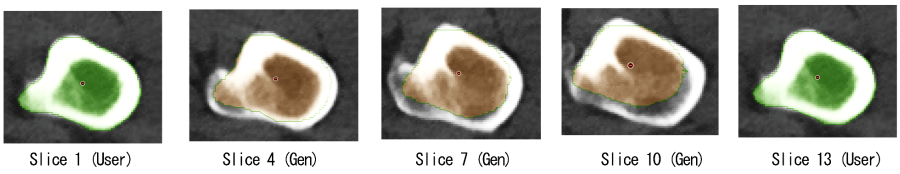

Figure 12: The effect of using solely the curvature force on the shape of generated masks (Using a CT abdomen scan)

To tweak this parameter, it is initialized at 0, and then gradually increased while the resulting accuracy is tracked. It is then set to the optimum of the tracked accuracy curve. The other parameters of the model are set to their starting values, as provided earlier this section. The accuracies have to be calculated using a segmentation that can represent a 'general' anatomical structure. The segmentation used for this analysis is a segmentation of an anatomical structure called the 'Flexura Hepatica', which contains some fast structural changes and irregular shapes, as well as some more stable, larger shape areas. A problem is that with other segmentations, the results may vary. For example, in the case of a structure that grows a lot, setting the expansion force to a large value makes sense and will increase the accuracy of the result. But, again, all parameters should be hidden from the user and set only once, so per-segmentation tweaking of parameters is not allowed.

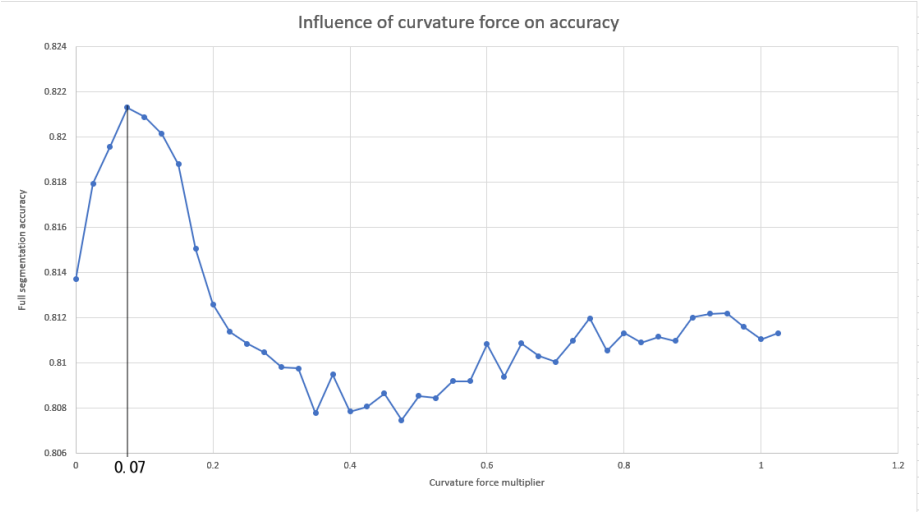

Figure 13: The influence of the curvature multiplier parameter on the accuracy of the segmentation (scan: CT Abdomen, segmentation: Flexura Hepatica. Every 4th slice is user-provided, the rest is generated.)

After running this analysis, it can be observed from figure 13 that **0.07** is the optimal value for the curvature force.

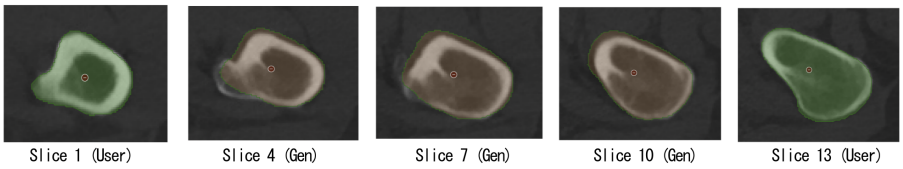

Figure 14: The effect of using a low (0.01) curvature force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

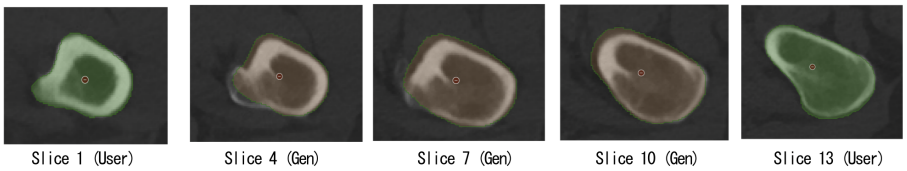

Figure 15: The effect of using the found ideal (0.07) curvature force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

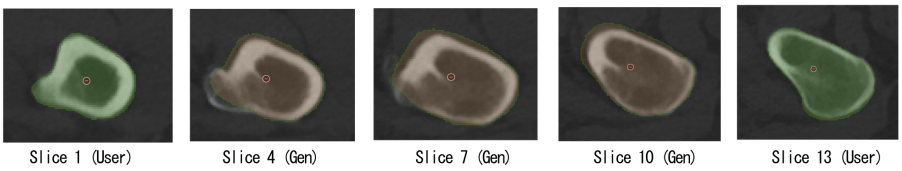

Figure 16: The effect of using a high (0.2) curvature force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

When looking at some visual results (see above figures), it can be observed that the influence of the curvature force on the shape of the segmentation is very minimal. When using irregular shapes with spikes/high curvature areas, the effect will be more visible, but in this example, other forces dominate, even when the curvature force is relatively high. When zooming in, a small difference can be observed: with a high curvature force, the contour becomes slightly more rounded (and therefore compact). Notice that this result corresponds with the created accuracy graph: the interval of the y-axis is very small, indicating a small difference. Also, the segmentation used for the calculation of the accuracy contains a lot more slices and some areas of high curvature change, so when observing that entire segmentation, some noticeable differences in shape can be observed on certain coupes.

6.1.2 Parameter: Elasticity force

The next parameter that needs to be tuned is the elasticity force multiplier. This force also does not have a lot of influence on the overall shape of the generated segmentations, but the elastic property it adds tends to make generated curves a bit more streamlined and compact (prevents curves from growing too much, providing a counterpart for the expansion force which roughly does the opposite). The same analysis method is used here, with the same segmentation ('Flexura Hepatica') from which the total accuracy is calculated. The sole influence of the elasticity force can be observed visually in figure 17. As can be seen, the force tends to make the shape more compact and small by pulling neighboring shape vertices to each other (watch the orange, generated contours).

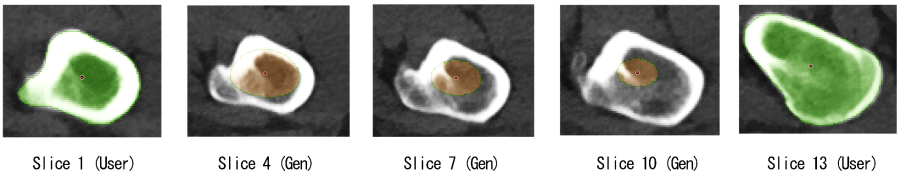

Figure 17: The effect of using solely the elasticity force on the shape of generated masks (Using a CT abdomen scan)

Now there are two different options for choosing parameter values. The curvature force has already been tuned, so the choice is using its initial value, or its tuned value. The initial value is chosen here, and to compensate for the influence changing one parameter can have on the rest of the system, all parameters will be manually tweaked a bit after they have all been tuned and saved.

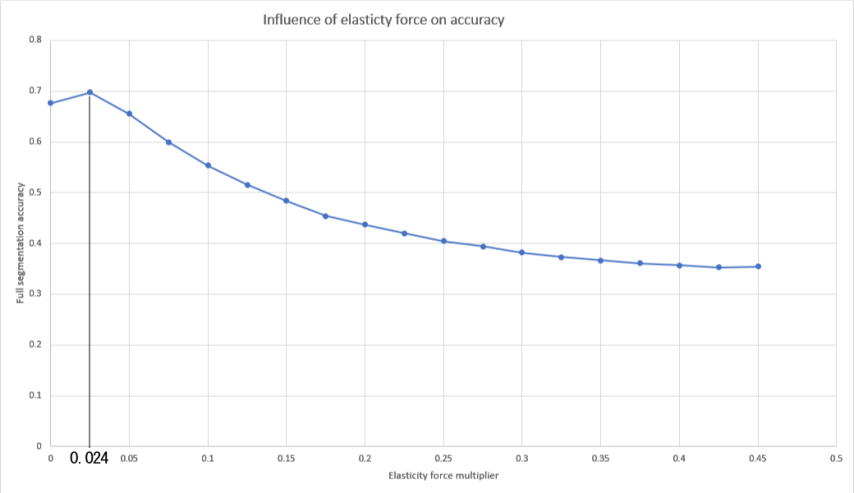

Figure 18: The influence of the elasticity multiplier parameter on the accuracy of the segmentation (scan: CT Abdomen, segmentation: Flexura Hepatica. Every 4th slice is user-provided, the rest is generated.)

After running this analysis, it can be observed from figure 18 that **0.024** is the optimal value for the elasticity force.

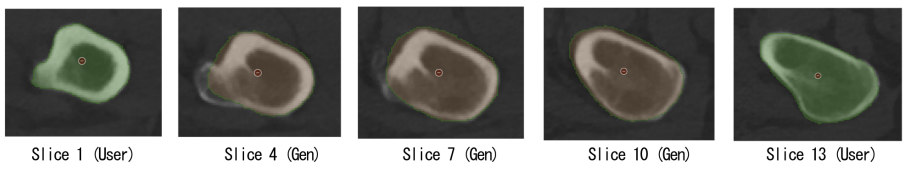

Figure 19: The effect of using a low (0.01) elastic force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

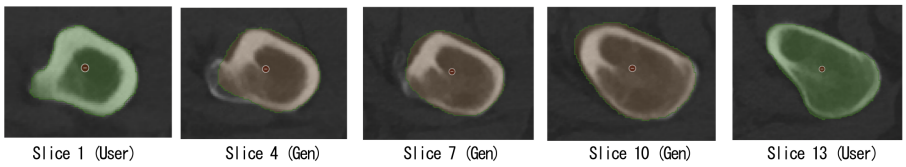

Figure 20: The effect of using the found ideal (0.024) elasticity force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

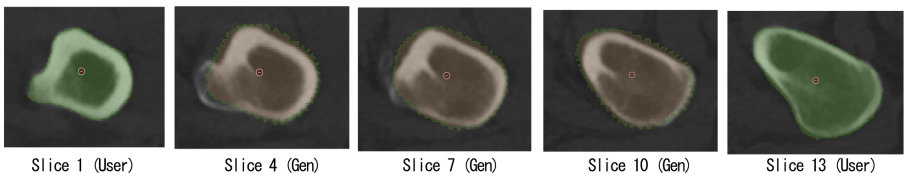

Figure 21: The effect of using a high (0.35) elasticity force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

The visual differences between using a low, medium and high value for elasticity (with the rest of the parameters set to their defaults) can be observed in the above images. Just like the curvature force, the elasticity force doesn't play a crucial role in determining the shape of the final contour on each generated slice. It ensures a regular spacing between contour vertices, but when it is too high, it can create irregularities in the contour border (see the high elasticity image). This force can be particularly helpful when successive slices of a segmentation contain high-detail areas on different parts of the contour: the elasticity force ensures that the vertices of the contour spread out better to cover these changes.

6.1.3 Parameter: Expansion force

Next up is the expansion force. It is another minor force that prevents the contours from getting too small and can help to push the edges towards the desired shape. The effect it has on generated segmentation can be observed in figure 22.

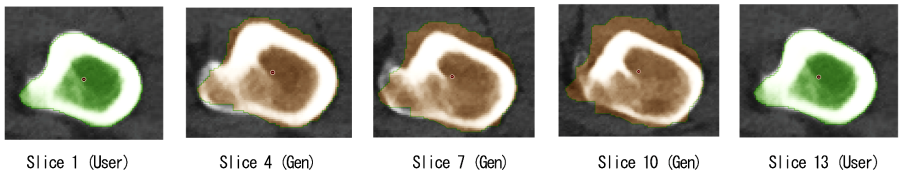

Figure 22: The effect of using solely the expansion force on the shape of generated masks (Using a CT abdomen scan)

Again, the initial parameter configuration is used, and the expansion force multiplier is increased from zero to 1 while the resulting generated segmentation accuracy is tracked, using the Flexura Hepatica segmentation.

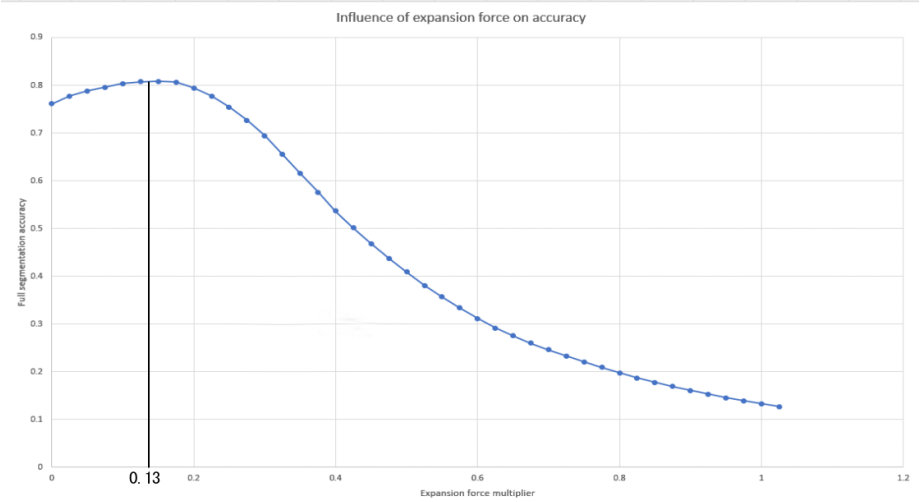

Figure 23: The influence of the expansion multiplier parameter on the accuracy of the segmentation (scan: CT Abdomen, segmentation: Flexura Hepatica. Every 4th slice is user-provided, the rest is generated).

The resulting optimal value is **0.13**, after which the accuracy drops as a result of over-extending the shape past the boundaries of the anatomical structure.

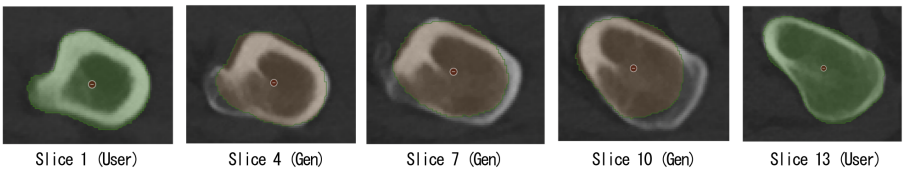

Figure 24: The effect of using a low (0.02) expansion force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

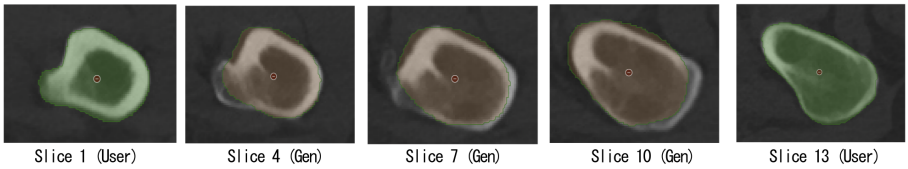

Figure 25: The effect of using the found ideal (0.13) expansion force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

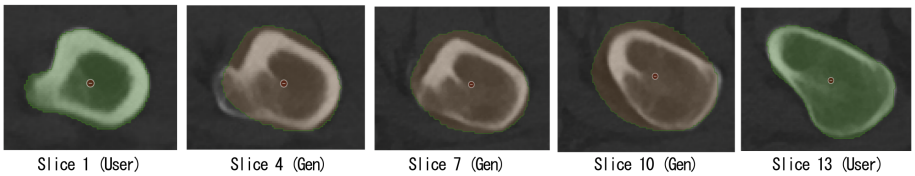

Figure 26: The effect of using a high (0.35) expansion force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

As can be seen in the above images, the expansion force plays an important role in guiding the evolution of the contour towards edges of the image by expanding it. The segmentation that uses a low expansion force becomes increasingly too small, partly because the curvature force tries to make each slice more compact. The expansion force can be seen as a helper of the image force: it push the contour toward edges, where the image force is stronger and takes over. It can be set too high, as is the case in figure 26. Here, the expansion force dominates the other forces, even when the image force is at its maximum (on an edge), creating contours that are too large.

6.1.4 Parameter: Image force

The image force plays a big role in the morphing of the shape to conform to desired edges. Recall the way the entire method functions: it generates the slices inbetween two user-provided slices, and uses the result of active contours on the previous (neighboring) slice as an initial contour for each slice that needs to be generated. So, adding a force that attracts the shape to nearby edges can have a cascading (large) effect on the entire segmentation, over the course of multiple slices. Because of this, the force falls in a different category compared to the three already tuned ones (the curvature, elasticity and expansion forces). Therefore, the already tuned values are used as parameter configuration during the accuracy analysis of this force. The remaining parameters (including the morphed mask force, which is not tuned yet) are set to their initial values. This is done to partly compensate for the influence different parameters can have on each other.
In figure 27, the sole influence of the image force can be observed visually, analogous to the previously tuned forces.

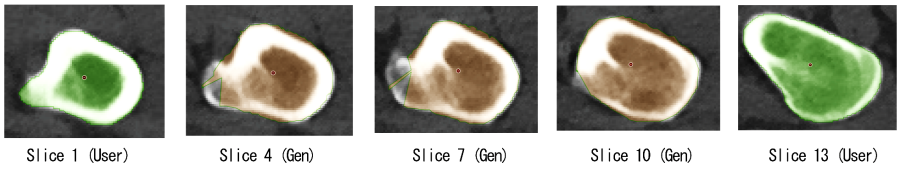

Figure 27: The effect of using solely the image force on the shape of generated masks (Using a CT abdomen scan)

Observe that this image force on its own does a decent job of following the desired edges of the anatomical contour, but has trouble conforming to larger changes in curvature (like the left part of the anatomical structure in figure 27). This force is tuned in the same manner as the other forces, starting from 0 while tracking the resulting accuracy of the generated segmentations (again, using the Flexura Hepatica segmentation). See figure 28 for this graph.

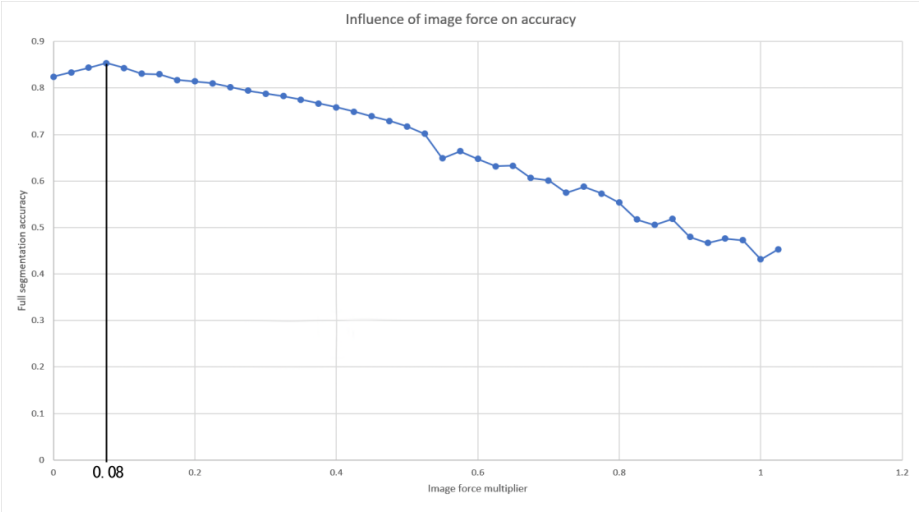

Figure 28: The influence of the image multiplier parameter on the accuracy of the segmentation (scan: CT Abdomen, segmentation: Flexura Hepatica. Every 4th slice is user-provided, the rest is generated.)

What is surprising about this result, is that the overall positive influence of the image force is not very high: the accuracy of the segmentation is 0.82 without using the image force, and 0.85 when it is used (at the optimum, which is at **0.08**). But, note that for other segmentations the influence can be stronger, and the fact that users can window/level the scan, thereby increasing edge strength in the desired structure, can also increase the image force influence.

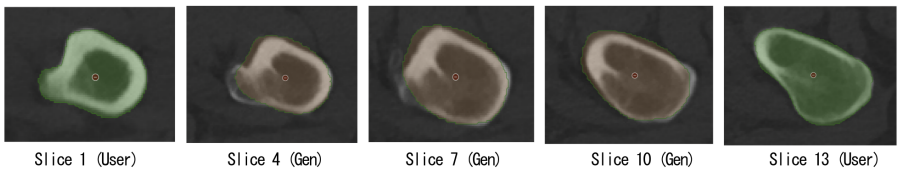

Figure 29: The effect of using a low (0.01) image force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

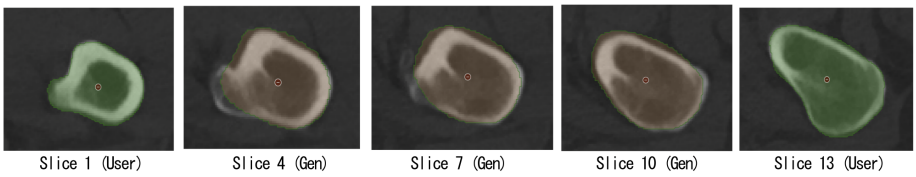

Figure 30: The effect of using the found ideal (0.08) image force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

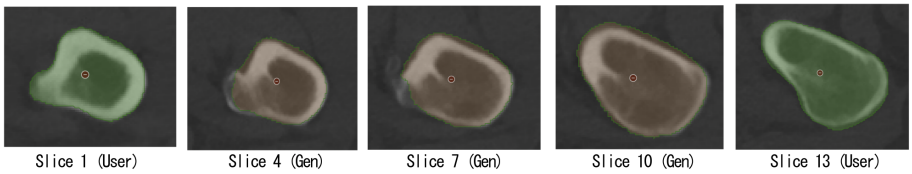

Figure 31: The effect of using a high (0.4) image force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

The image force plays an important role in guiding the contour towards structural (anatomical) edges. In the example segmentation that is used as a visual demonstration here (see the above images), this effect can be observed. The other forces, mainly the morphed mask force that steers the contour towards slice 13, cause the contour to drift away from the anatomical bulge on the left side of the image too fast. But, when using a large image force value, the contour conforms to this visual structure, creating a better overall segmentation in this case. In this visual example, it can even be argued that using a higher image force (higher than the found ideal value) is better, but keep in mind that this is only one example. In the larger segmentation that is used to determine the ideal image force value, more and different kinds of structural changes are present. When the image force is too strong, the contour can be drawn towards wrong edges, or towards noise or irregularities in the image.

6.1.5 Parameter: Morphed mask force

The last vital parameter to tune is the morphed mask force multiplier. This force determines how strongly the contour is pulled towards the morphed shape trajectory generated inbetween user-provided slices. Its effect can be observed visually in figure 32.

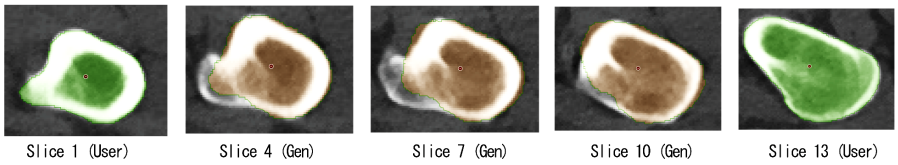

Figure 32: The effect of using solely the morphed mask force on the shape of generated masks (Using a CT abdomen scan)

The force plays a very important role in guiding the contour shape towards the next user-provided goal slice. The morphed mask multiplier is an overall constant that is applied to all generated slices, but the strength of this force also linearly decreases as the distance to a user-provided slice increases. This makes sense, because the further a provided slice is, the less reliable using the morphed mask becomes. For the tuning of this parameter, it again starts at zero and gradually increases while the resulting accuracies are tracked (with the Flexura Hepatica segmentation). The result is presented in figure 33. Be aware of the scale of the y-axis of this graph: the increase in accuracy that using the morphed mask adds is about 0.1. As can be seen, the optimum lies around **0.23**, beyond that the curve roughly stays flat. This has to do with the attractive strength of the morphed mask decreasing as the contour moves closer to it (the strength of the morphed mask is also inversely proportional to the distance to it, as described in the Method section), and the strength also decreasing as the distance to user-provided slices increases.

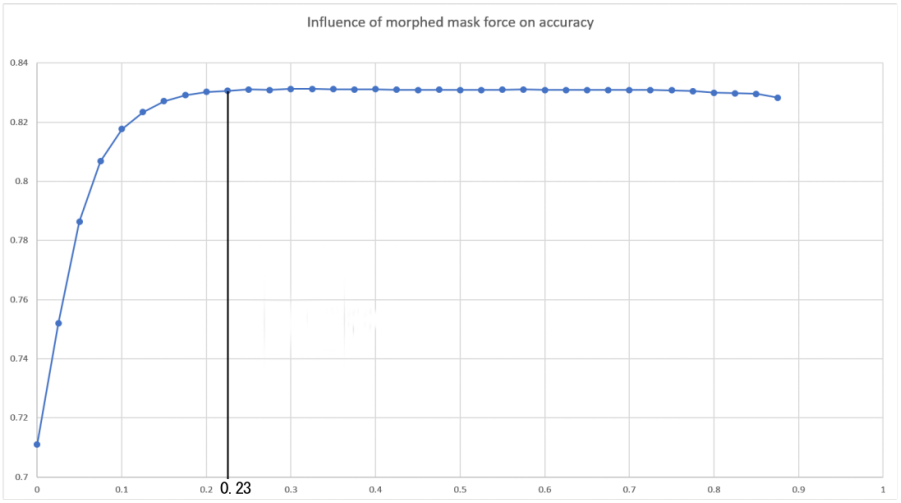

Figure 33: The influence of the morphed mask multiplier parameter on the accuracy of the segmentation (scan: CT Abdomen, segmentation: Flexura Hepatica. Every 4th slice is user-provided, the rest is generated.)

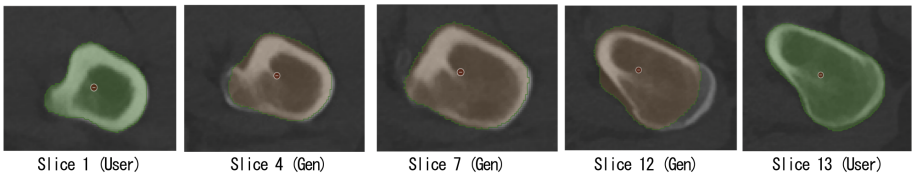

Figure 34: The effect of using a low (0.01) morphed mask force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

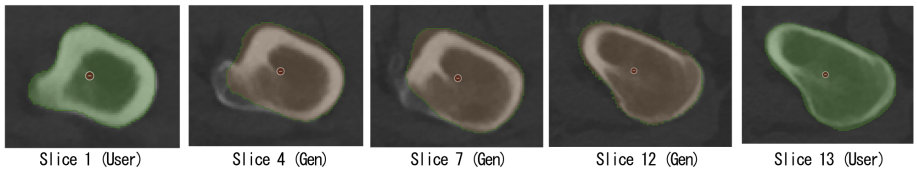

Figure 35: The effect of using the found ideal (0.23) morphed mask force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

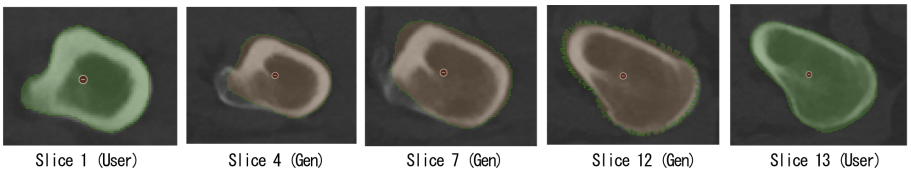

Figure 36: The effect of using a high (0.4) morphed mask force on the shape of generated masks (Using a CT abdomen scan). The rest of the parameters are the defaults specified earlier.

The main role of the morphed mask force is to ensure the evolution of the contour is evenly spread across all the inbetween slices that need to be generated. When it is set too low, like in image 29, the contour can fall increasingly behind, creating a large, undesirable jump (see difference between slice 12 and 13) in the contour shape. On the other hand, when this force is set too high, it can create irregularities in the contour boundary due to overshooting, and it can overrule the image force, causing the contour to ignore small structural changes in the image dataset.

6.1.6 Parameter: Remaining parameters

The remaining mentioned parameters of the active contours model, namely the stiffness force, the smoothing strength and the number of iterations, can be tuned by hand, without extensive analysis. These parameters provide some small improvements that slightly prettify the overall look of resulting segmentations, while playing almost no role in deriving shape and structure. The number of contour iterations was initially set at 250, and because all other values are tuned based on this value, it does not need to change much. While using the tuned parameters and observing results, increasing the curve smoothing and slightly decreasing the contour stiffness improved the result visually, so the values for these parameters finally became **0.02** for the stiffness force and **0.2** for the smoothing strength. Also, giving the algorithm a little more contour iterations seemed to slightly improve the results too, so the contour iterations were tuned to **270**. A more extensive subjective analysis of the system is provided later this section, because the subjective judgment (the 'niceness' of the system) is essentially worth more than an objective analysis: radiology teachers are the users of the new method and will judge the usability of the results purely visually.

6.1.7 Combining parameters and measuring accuracy

So, after this tweaking and optimizing, and adjusting for the influences the parameters have on each other (mainly the image and morphed mask parameters need to be tuned optimally, because they mainly define the shape of the final segmentation), the final values for the different parameters became the following:

- Elastic force: 0.03
- Curvature force: 0.068
- Expansion force: 0.035
- Image force: 0.13
- Morphed mask force: 0.14
- Number of iterations: 275
- Stiffness force: 0.02
- Smooth strength: 0.2

The objective accuracy of the reproduced segmentations (using the distance measure to the reference segmentation described earlier) using these parameters is generally good: most often, it lies around 80 percent when letting the user provide about every 4th slice. As expected, this accuracy decreases as the distance between user-provided slices increases, because of the system having to generate more slices based on less data. See figure 37 for a plot of this effect.

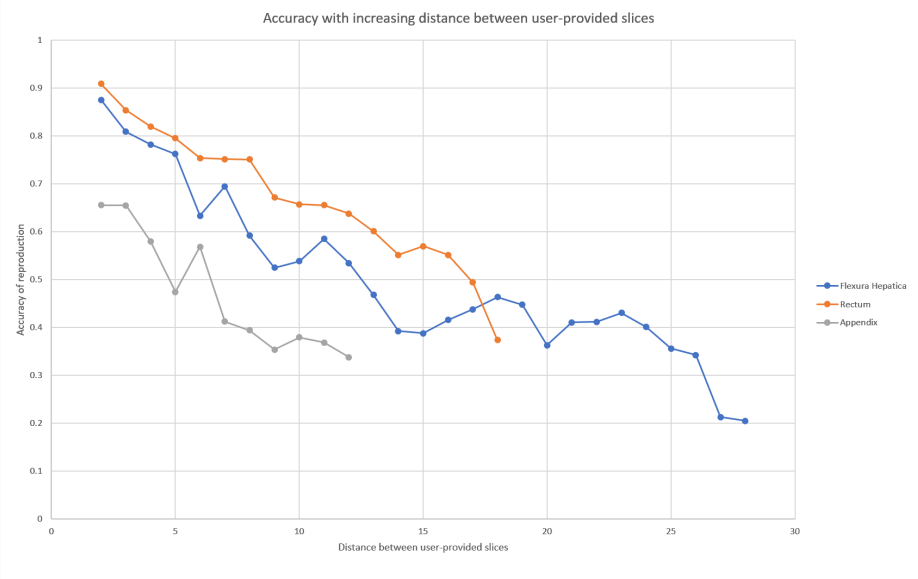

Figure 37: As the distance between user-provided slices increases, the accuracy of the generated reproduction decreases (scan: CT Abdomen, data for three different segmentations provided by the UMC)

The values chosen here are as general as possible, yielding acceptable results across the entire domain of anatomical segmentations, but by not allowing any tweaking for different types of scenarios, a significant amount of accuracy is sacrificed. For example, here are some comparisons of accuracy results of three different UMC-provided segmentations, using the fixed, tuned parameters versus optimized parameters for that specific segmentation (each parameter tweaked until a local maximum is reached).

- Colon descendens, every 5th slice is user-provided, fixed parameters accuracy: **0.771**, optimized parameters accuracy: **0.845**
- Flexura hepatica, every 5th slice is user-provided, fixed parameters accuracy: **0.791**, optimized parameters accuracy: **0.852**
- Rectum, every 4th slice is user-provided, fixed parameters accuracy: **0.912**, optimized parameters accuracy: **0.934**

In conclusion: to optimize the different active contours parameters as much as possible, as much domain knowledge as possible should be taken into account. For the application this thesis is aimed at, the chosen fixed values generally provide acceptable results, but could potentially be improved for individual cases.
Some visual results of using the optimized parameters can help clarify the strong and weak points of the method. Below are some examples that help illustrate

the observation that sometimes, the generated segmentation can look very nice and accurate, and sometimes less so. An automatic parameter tuning system could help alleviate this variance in usability.

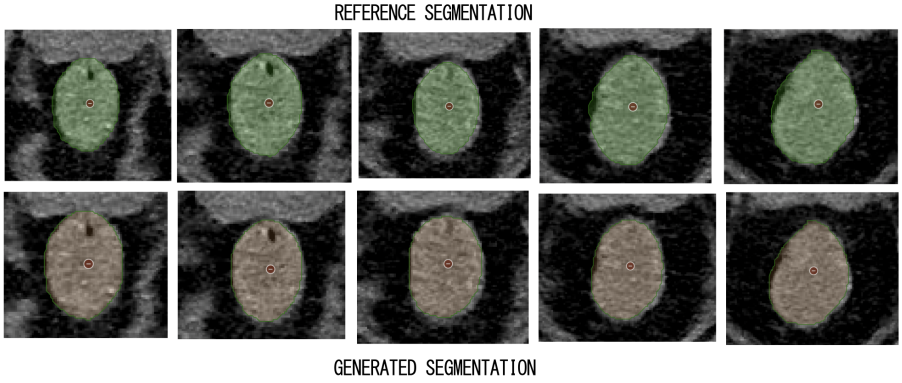

Figure 38: Five successive slices of the reference segmentation 'Rectum' isolated and compared with an auto-generated version of the same slices, using the tuned, fixed parameters.

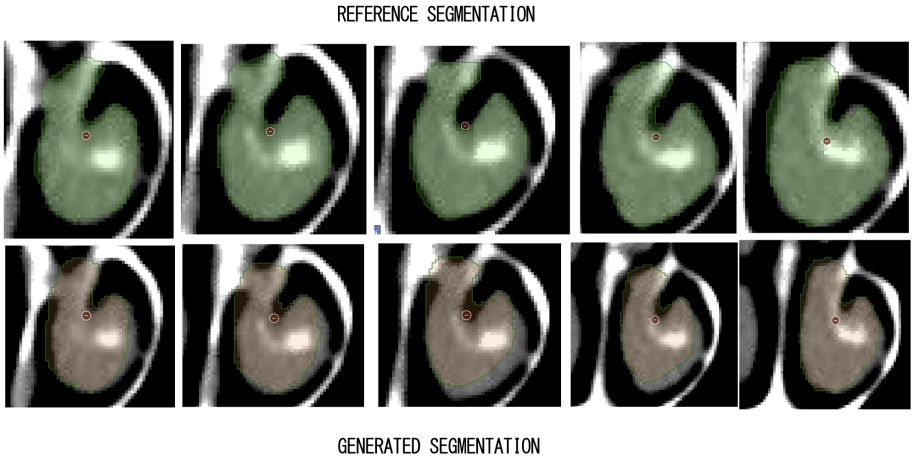

Figure 39: Five successive slices of the reference segmentation 'Concha Nasalis' isolated and compared with an auto-generated version of the same slices, using the tuned, fixed parameters.

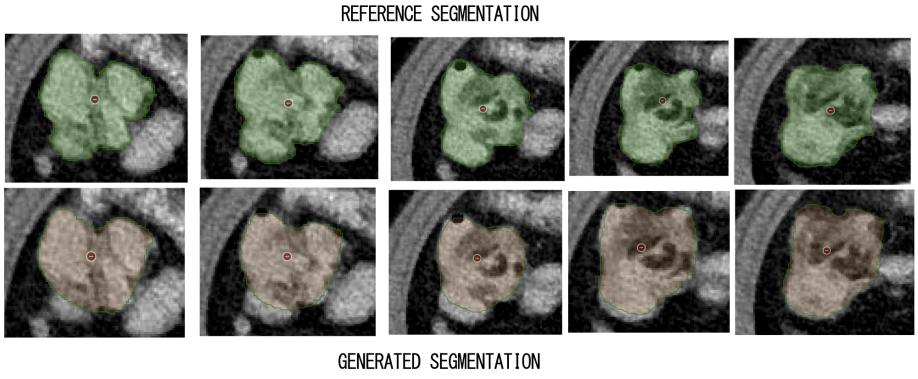

Figure 40: Five successive slices of the reference segmentation 'Coecum' isolated and compared with an auto-generated version of the same slices, using the tuned, fixed parameters.

As can be deduced from these visual examples, the method performs best when no large per-slice differences in shape are present (as expected). The Rectum segmentation even shows improvement compared to the user-provided reference version: the contour fits more tightly around the anatomical structure. A side-effect of using fixed parameters can be observed in the Concha Nasalis segmentation: the image force is set too low in that case. A larger value would improve the accuracy. The Coecum segmentation is relatively good, but would need one or two additional edits to let the contour conform to the lump at the bottom part of the structure.

6.2 Performance overview

The system consists of a number of separate components. The time-complexity of these can be analyzed independently. The core system, namely the active contours model, has linear complexity in the number of contour vertices (the number of contour iterations is constant). There are three main variables that are subject to change from use case to use case: the number of vertices in the segmentation, the number of slices that need to be generated, and the 2D resolution of the 3D dataset (this is relevant for the GVF algorithm that is run on each slice). The shape morphing algorithm only scales with the amount of vertices of the source and target masks, which does not provide a bottleneck in all practical cases. This is because segmentation slices typically only contain a few hundred vertices, for which generating a morphing sequence is relatively fast (typically under half a second, tested on an I7-4790K). The Gradient Vector Field algorithm runs in $O(n \cdot log(n))$, where n is the resolution of the input image. Most medical scans are around 512x512, and because of the added region optimization (only the area including and around the segmentation contour is used as input to the algorithm), a single run of GVF typically takes about 0.4

seconds). Within the active contours model, the elasticy, expansion and curvature forces are calculated in constant time, so the bottleneck of the system is expected to be the repeated execution of the GVF algorithm on each separate inbetween slice. As is apparent from figure 41 and 42, the running time of the entire system grows linearly in the number of slices that are generated, and this is expected: it consists of successive repeats of the same sub-algorithms with the same complexity, because the result of the previous slice is copied to the next slice for use as initial guess (so the number of vertices stays the same across slices). When the number of vertices in the generated inbetween slices, and thereby also the the size of the contour (the contour is pre-processed so that roughly is uniformly sampled), is plotted against the running time (on a single slice) of the algorithm, the resulting curve is a little less linear. This is mainly because the input size of the GVF algorithm grows with the contour, introducing a growing component with non-linear complexity that is influencing the running time of the whole algorithm to some extend. This effect can be seen in figure 43

Analysis of the total running time of the algorithm:

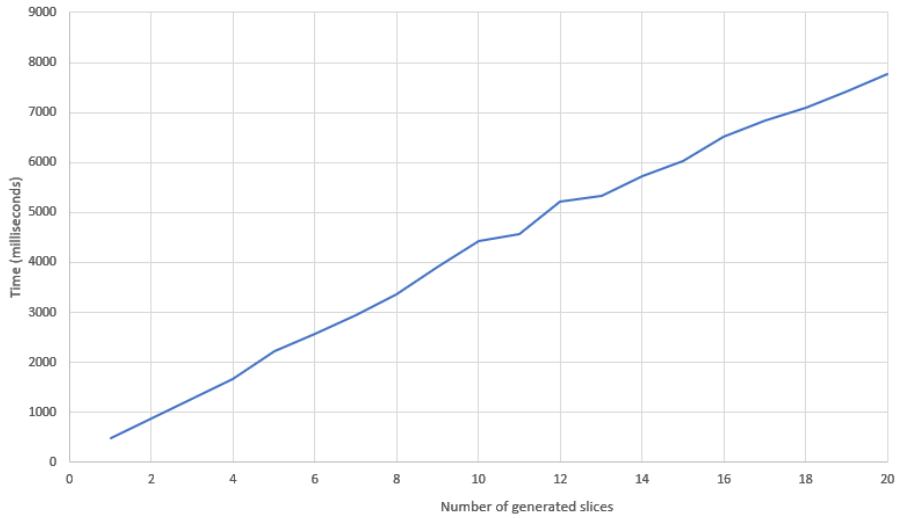

Figure 41: The relation between the number of generated (inbetween) slices, and the running time, for a contour with 184 vertices

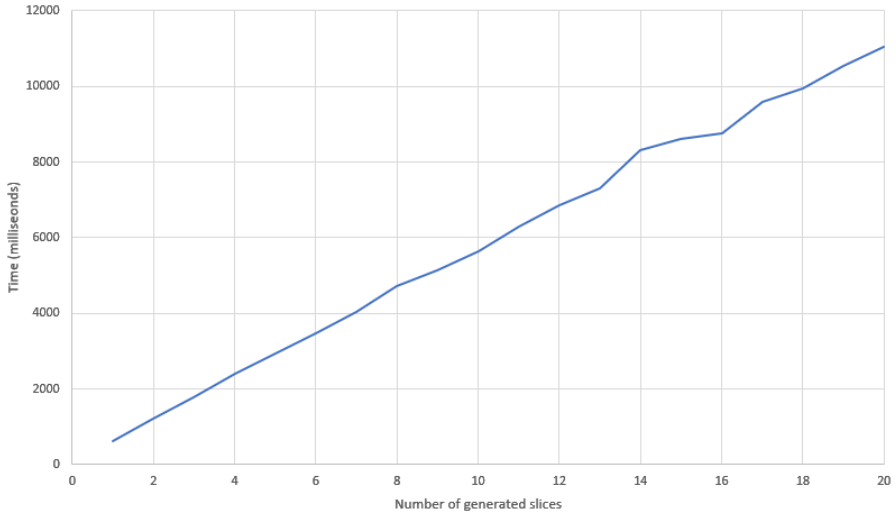

Figure 42: The relation between the number of generated (inbetween) slices, and the running time, for a contour with 309 vertices

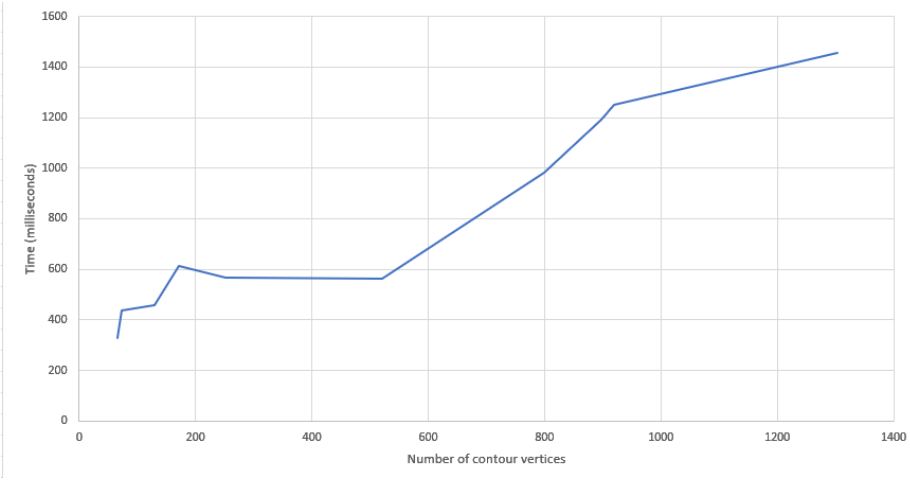

Figure 43: The relation between the number of vertices in the active contour, and the running time (single slice)

Providing some tables with running times of the core parts of the system can expose the bottleneck of the system as a whole. The following three tables are recorded using three different segmentations provided by the UMC, and contain the running times of the active contours iterations, the GVF calculation and the

morphed mask generation. Note that the running time of the morphed mask generation does not change when the distance between user-provided slices increases. This is because it does not matter how large this distance is: there is always one morph sequence between two user-provided shapes. There are a few minor parts of the system that are not included in this table, like the laplacian curve smoothing and the morphed mask generation (because the morphed mask generation does not scale with the amount of generated slices: there only is one morphing sequence between any two user-provided slices). Their contribution to the total running time is negligible and can be left out.

No. of slices	**Active Contours time**	**GVF time**	**Total time**
1	0.15s	0.45	0.848
2	0.27s	1.01s	1.531s
3	0.41s	2.04s	2.71s
4	0.64s	2.74s	3.66s
5	0.88s	3.6s	4.781s
6	1.08s	4.43s	5.82s
7	1.14s	5.01s	6.5s
8	1.39s	5.62s	7.42s
9	1.69s	6.07s	8.19s
10	1.88s	6.39s	8.71s
11	2.11s	7.1s	9.67s
12	2.45s	7.78s	10.74s

Table 1: Running times on segmentation Flexidura Hepites (on an I7-4790K). The number of generated slices is the same as the distance between user-provided slices: basically all the slices that are left open and therefore need to be generated.

No. of slices	Active Contours time	GVF time	Total time
1	0.32s	1.05s	1.752s
2	0.61s	2.01s	3.011s
3	0.91s	3.14s	4.454s
4	1.13s	4.14s	5.679s
5	1.47s	5.04s	6.928s
6	1.81s	6.11s	8.355s
7	2.12s	7.19s	9.79s
8	2.47s	7.98s	10.944s
9	2.71s	8.95s	12.16s
10	3.01s	9.43s	12.952s
11	3.29s	10.11s	13.925s
12	4.02s	10.86s	15.421s

Table 2: Running times on segmentation Rectum (on an I7-4790K). The number of generated slices is the same as the distance between user-provided slices: basically all the slices that are left open and therefore need to be generated.

No. of slices	Active Contours time	GVF time	Total time
1	0.08s	0.35s	0.55s
2	0.17s	0.6s	0.903s
3	0.28s	1.01s	1.431s
4	0.35s	1.41s	1.916s
5	0.46s	1.71s	2.334s
6	0.56s	2.05s	2.788s
7	0.61s	2.46s	3.27s
8	0.62s	3.14s	3.99s
9	0.73s	3.34s	4.314s
10	0.8s	4.01s	5.059s
11	0.95s	4.31s	5.52s
12	1.01s	4.86s	6.136s

Table 3: Running times on segmentation Iliocaapklep (on an I7-4790K). The number of generated slices is the same as the distance between user-provided slices: basically all the slices that are left open and therefore need to be generated.

Note that, as expected, the repeated GVF calculations are the bottleneck of the entire system, with some distance. These vector fields cannot be cached and retrieved, because they are different for each separate window/level setting. Earlier tests, with a version that does not only calculate the GVF for a given target region but instead for the entire image frame, were even more extreme; now, the GVF calculations at least are within the same rough magnitude as the other parts of the system. There definitely still is room for improvement: al-

lowing quick edits of segmentations with quick feedback from the system largely contributes to user satisfaction and usability. Currently, the system has a big memory overhead: it needs to load each image into memory, every time a specific slice from a specific dataset is requested. A memory caching system that smartly preloads different images may improve the speed of the GVF system.

6.3 Applicability

The segmentation tool, presented as part of the new e-learning system and 3D scan viewer developed for the UMC, is evaluated by a number of radiologists and developers currently working at the UMC. They are used to working with their old environment (MITK for segmentations, an offline software package for e-learning and examination), so they can provide a fair comparison.

6.3.1 Short description of the interface

In order to better understand the way people use the system and interact with it, a description of the interface that wraps the segmentation method and presents it to the user is in place.

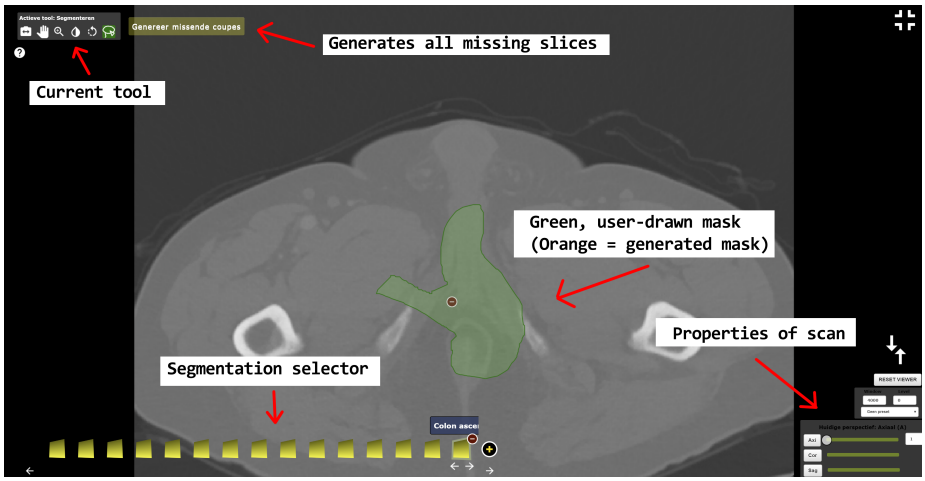

Figure 44: The interface the users use to interact with the segmentation method

This interface lets the users create masks (contours) on slices, correct them by adding/subtracting parts, and generate inbetween missing slices by using the large yellow button. The algorithm is then executed on the server, after which the segmentation gets updated: generated slices are labeled as orange. When an edit is made on a generated mask, it switches its label from 'generated' to 'user', so that if the user runs the algorithm again, the new correction is taken into account (when running the algorithm, only user-generated slices are kept). This way, the user can continually update the segmentation, editing the inbetween

slices where necessary, until satisfied.
Now, more details about how users interacted with the system and their insights can be discussed.

6.3.2 General remarks

The users (testing group of two radiologists and two software developers currently at work at the UMC) agreed that the interface is easy to understand and clear. The creation and editing of segmentations is simple, and can be done in three different perspectives (axial, coronal, sagittal). It is clear how the segmentation method is intended to be used, and the 'one button design philosophy' is appealing. While segmenting, iteratively editing slices and auto-generating again is convenient, but could be more responsive: when generating more slices, the waiting time can become frustrating. The users emphasized the possibility of applying the segmentation method to other e-learning and examination systems: its portability and ease of use can be exploited in similar medical applications. The fact that information from multiple perspective cannot be combined (segmentation reconstructions and use of that data during automatic completion) is a limitation. Automatically handling changes in topology would improve the usability of the method as well.

6.3.3 Use cases

Here, some use cases and observations are discussed that describe how the system (and the segmentation method in particular) is received within the context of application.

- **User 1:** The first user that tested the system is a software developer currently working for the radiology department at the UMC. He had a basic understanding of the interface and the intended use of the auto-complete feature, and started by choosing an arbitrary anatomical structure to segment. After drawing six slices by hand (single contours, no multiple topology areas), leaving a space of around five slices between every two drawn slices, he used the auto-generate button. He was satisfied with the result. With another segmentation (a more irregular shape), he mentioned that the generated contours did not exactly follow the image boundary. Most likely, this is due to the image force being too low for that particular case. He suggested that a simple slider that lets the user balance the morphed mask force against the image force might improve the results. Even though this goes against the design intentions, using a single simple slider to balance the two most important forces of the model might provide benefits that outweigh the disadvantage of the added complexity. The user also noticed that the system only generating slices *between* user-provided slices, and not running active contours on the user-provided slices themselves, can be inconvenient. He wanted the possibility to roughly draw the contours, refine them using active contours with a high image force (and no morphed mask force), after which the rest of the slices get generated.

The workflow can be quicker this way: users can provide estimates instead of pixel-perfect segmentation slices. But overall, while testing a number of segmentations, this user noticed the auto-complete feature can save a significant amount of time, mainly when segmenting parts that are not too irregular and have a constant topology (no splits / joins). When segmenting a structure that changes in topology, the user remarked that the center of gravity coupling technique (contours on different slices are matched and morphed into each other by minimizing the distances between their center of gravities) worked correctly. The requirement that any two successive slices containing a change in topology must both be user-provided, could be enforced by providing the user with an error (currently, the algorithm simply does not run). The user considered not being able to utilize segmentation data from multiple perspectives and not allowing the generation of changing topologies two limitations of the system.

- **User 2:** The second user was a professional radiologist. She had a different way of using the system and different requirements. In general, she needed more precision: almost pixel-perfect segmentations. The initial observation that, because the main application for the method is e-learning, very precise segmentation would not be necessary because over-estimations are acceptable, does not entirely hold: she mentioned the desire and requirement of a radiologist to be precise, and iteratively edited generated slices until the segmentations were nearly pixel-perfect. She also used the Window/Leveling tools a lot more than the first user, to improve the usability of the image force by increasing the contrast on the desired anatomical structure. With simple structures, generated segmentation were accurate (using a spacing of around five slices), only requiring a few small edits. With longer, smaller structures that are more irregular, she edited bigger parts of the boundary before being satisfied. In general, she remarked that structures that have a direction perpendicular to the viewing angle (axial, coronal, sagittal) seem to get segmented best, probably because of the increased gradient clarity. In cases where a lot of edges lie very close to each other in the dataset, she had to make more adjustment as parts of the contour drifted away from the correct edge. Another added feature she would have liked to see is the allowance of holes in segmentations (double boundaries): this currently is not possible, as each contour is a single list of 2D points. Another case that does not work ideally is one in which an anatomical boundary is not clearly visible in the dataset (absorbs the radiation at the same wavelength as surrounding tissue). On automatically generates slices, this means that the contour edge on such cases often is not entirely correct. Expert knowledge is required to specify the exact tissue boundary, for each slice in which the gradient is not clear. This problem is only solvable with models that have knowledge about common anatomical structures (which introduce a lot of added complexity). But, in a lot of cases, the method performed very well. She was surprised by the performance when segmenting the pancreas, as this is an irregular

contour that quickly changes shape across slices. Only a few small edits were necessary here. Overall, the method is usable in a lot of cases, and increases the speed in which segmentations can be made, but would be even more usable if changes in topology are allowed, and reconstructions in multiple perspectives.

- **User 3:** The third user is another developer at the UMC. He did not use the Window/Level feature as much as the previous user, and mainly focused on creating segmentations of large structures. For example, he segmented the liver, drawing every tenth slice, and found the resulting generated segmentation to be quite accurate (only a few edits were necessary), despite of the large distance between user-provided slices. He remarked that the viewer and segmentation tool felt responsive, the generation tool finished fast enough, but suggested the user of a parameter balancing slider (just like user 1). In a number of cases, the image force or morphed mask force seemed too strong. Letting the user slide a single slider to tune the relative strength of those two main forces could improve the resulting segmentation significantly: the correct balance is dependent on the specific case. For example, in a structure that is very rough (a lot happens between user-provided slices), the image force needs to be very strong to compensate for this.

When the software is more widely used by teachers (at the UMC), for which there are concrete plans, more insights can be gathered. For now, it is enough to recognize the potential of the segmentation method and point out its strong and weak points.

6.4 Fail cases

As mentioned earlier, the algorithm runs in a single perspective, and cannot handle topological changes. This means that on every slice where a change in topology is present, the teacher is required to fill in the segmentation manually. This implies that if a slice has multiple segmentation parts, a one-to-one correspondence with the next user-provided slice is required. The parts that correspond are matched using their center of gravity as a distance measure.
The algorithm works well if successive image slices don't have large structural changes (rapid changes in shape). If, on the other hand, large per-slice changes in the image are present, the stiffness and shape morphing forces, as well as noise in the gradient vector field of the image, cause the contour to fall behind the rapidly changing anatomical structure. See figure 45 for some examples of this artifact.

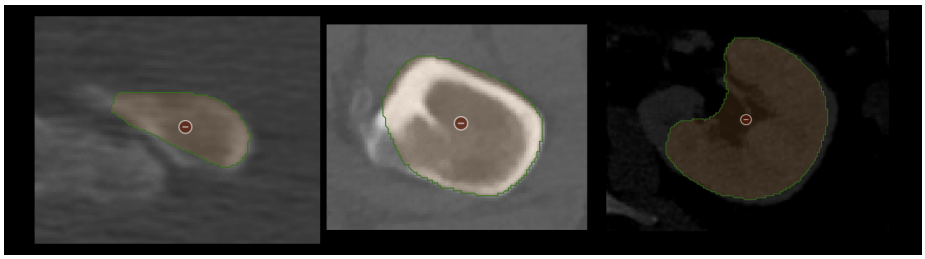

Figure 45: When the dataset contains rapidly changing shapes, generated slices can fall behind. Here are a few examples of such artifacts.

Another situation that may generate undesirable segmentations, is one in which multiple anatomical structures with defined edges lie very close to each other. The resulting gradient vector field will be influenced by these structures, and steer (parts of) the active contour in the wrong direction. This effect is demonstrated in figure 46 (here, an anatomical structure that is very close attracts the left side of the active contour in the wrong direction).

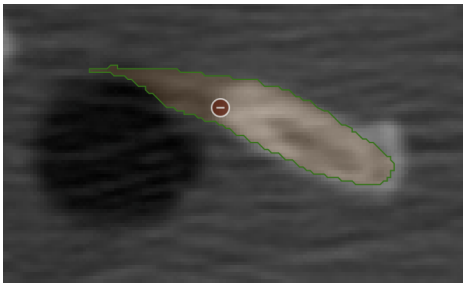

Figure 46: When the dataset contains multiple contrast-rich edges, parts of the contour may be drawn to the wrong shape

As demonstrated during the exploration of the parameters, the accuracy of the provided reconstruction drops rapidly as the distance between user-provided slices increases. This makes the system unsuitable for cases in which a larger amount of inbetween slices need to be accurately generated (more then about 4 slices). The use of domain knowledge about likely evolution patterns of anatomical structures could improve upon the system. For example, an extra force could be added called 'anatomy force', that is derived from a database of likely progressions of the target shape. Alternatively, these likely progressions could be generated using a neural network. In short, the fact that the system does not use any medical domain knowledge, is a limiting factor. The choice to use fixed parameters can also be limiting. The requirement of not providing any parameter settings to the user as to increase usability still leaves room for experimentation. For example, the parameters could be set automatically for each segmentation by analyzing the user-provided slices and feeding them to a neural

net that outputs parameter settings, and is trained on a database of reference segmentations. This may decrease the running speed of the algorithm, but the added accuracy can make a big difference.

7 Discussion

From the start of the project, the method has been designed for use within the context of the new UMC e-learning environment. Its main contribution is the the 'one button design philosophy': as opposed to software the UMC currently uses for the creation of segmentations for educational use, the new system only has a single button that automatically generates missing slices. The parameters and different steps of the algorithm are all hidden from the user, making the method very easy to use. The segmentation system is tightly integrated in the created online 3D viewer. Because the web-based viewer and the segmentation system have been developed in parallel, they are loaded as a single module that is available to all teachers. Even though the method (or ideas from it) could potentially be used in another context, its niche application is the use in a medical e-learning application that requires teachers to create structural segmentations. The main two reasons for this are:

- It has a focus on ease-of-use, thereby sacrificing the added accuracy per-case that parameter tweaking can provide. This is justified by the application context and the people who will use the new system: the context is a learning application that students can use to improve their radiology skills, implicating that exact, pixel-perfect boundaries of segmentations are not required, and the radiologists who create the questions want to be able to do this as quickly and efficiently as possible, without any knowledge of the meaning of subsystems and parameters

- It has a specific way of use that is required to get optimal results. Teachers are asked to choose a scan viewing perspective, and then draw about every fifth slice (or less/more depending on the morphology of the target structure). All missing slices are automatically generated using the system, after which teachers can correct/redraw/remove the segmentation at will. As described, the method only operates in a single perspective, so it cannot use mask information from multiple views to generate segmentations.

During the development of the algorithm, the department of Radiology at the UMC Utrecht remained involved. The software they currently use for the creation of segmentations is MITK, which is a program with many options, parameters, types of algorithms and segmentation types. The main advantage of this new system is the ease of use: the segmentation system is now tightly integrated into the e-learning environment and the algorithm runs with a single button, without any tweaking of parameters. Currently, all segmentations are fully hand-drawn by radiology experts at the UMC, mainly because of the absence of

an easy-to-use auto-complete feature that doesn't require any knowledge about parameters and other settings. With the new system, using auto-complete has become very easy. The developed software and the new method will, in their current form, not be used or further developed by the UMC, but ideas from the research and the software pilot done in January will be adopted by the VQuest development team at the UMC (they are creating a new software package for medical e-learning). The typical way in which the developed system can be used is the following:

- The viewer programmed for this research is used to scroll through the 3D scan, localizing the target anatomical structure.
- A preferred perspective is chosen, in which a number of slices on which the structure is visible is segmented.
- The 'generate' button is clicked, so that the system generates all the missing slices inbetween provided ones
- The user notices that the segmentation is not ideal, and provides some additional corrections to slices that are incorrect
- The edited slices get labeled as user-provided, after which the user clicks the button again and the system generated the missing slices in the same manner
- The user continues this correcting process until satisfied.

The involved radiologists recognized the potential of the system, and specified a number of use cases in which the method already provides a benefit over their current situation, but emphasized that additional improvements and additions are necessary to truly provide a convenient experience that substantially improves upon drawing segmentations by hand or using available software like MITK. The precision with which they like to segment demands accurate results: they usually are perfectionists and like very accurate segmentations, even for e-learning purposes. The accuracy of the system as it is, often is too low: around 80 percent on substantially large segmentations, with every 5th slice user-provided. To improve the usability, this value needs to go up. A possible way to achieve this, is by creating a model-driven automatic parameter tuning system. But overall, the research has been a success: the UMC will continue to develop the e-learning software created for it, mainly the 3D viewer that allows for the viewing of 3D scans and creation of segmentations in-browser. The segmentation method will be subject to continued development within this context.

The segmentation method developed during this research can also be used as a base to expand upon. There are a few areas where improvements are possible. The first is the use of fixed parameters in the Active Contours model. This greatly improves the ease of use and speed of the method, and allows for the 'one-button design philosophy' established earlier, but compromises somewhat

on the accuracy of the resulting segmentations. By using a neural net or some other model that, given a specific image dataset and the input user segmentation slices, outputs the desired parameters that are as optimal as possible for the specific case. Another weak point of the algorithm is the speed of the Gradient Vector Field sub-algorithm: the reading from memory and performing the multigrid solving method make up a substantial part of the total running time of the method. This can be optimized by creating a memory caching system, or improving upon the speed of the multigrid method.
Another way of improving the experience for users is to incorporate segmentation data from multiple perspectives into the generation process. This increases the flexibility and the 3D accuracy of the segmentations: because the current implementation only uses 2.5D (depth) data, reconstructions to other perspectives can introduce inaccuracies and staircasing (sudden jumps in the contours) artifacts. This can be done by incorporating a 'perspective force' into the active contours model, that is generated from contour points specified in other perspectives that are transformed into forces.

References

Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.

Antonio Criminisi, Toby Sharp, and Andrew Blake. Geos: Geodesic image segmentation. In *European Conference on Computer Vision*, pages 99–112. Springer, 2008.

V Caselles, R Kimmel, G Sapiro, and C Sbert. 3d active contours. In *ICAOS'96*, pages 43–49. Springer, 1996.

Ted W Way, Lubomir M Hadjiiski, Berkman Sahiner, Heang-Ping Chan, Philip N Cascade, Ella A Kazerooni, Naama Bogot, and Chuan Zhou. Computer-aided diagnosis of pulmonary nodules on ct scans: Segmentation and classification using 3d active contours. *Medical physics*, 33(7Part1):2323–2337, 2006.

Paul A Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C Gee, and Guido Gerig. User-guided 3d active contour segmentation of anatomical structures: significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006a.

Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995.

Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

Jun-Feng Guo, Yuan-Long Cai, and Yu-Ping Wang. Morphology-based interpolation for 3D medical image reconstruction. *Computerized Medical Imaging and Graphics*, 19(3):267–279, 1995. ISSN 08956111.

Dragos Nicolae Vizireanu. Generalized Morphological 3D Shape Decomposition Grayscale Interframe Interpolation Method. *Engineering and Technology*, 1 (December):84–87, 2005.

Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A pde-based fast local level set method. *Journal of computational physics*, 155(2):410–438, 1999.

Stanley Osher Ronald Fedkiw and Stanley Osher. Level set methods and dynamic implicit surfaces. *Surfaces*, 44:77, 2002.

Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D Fox. Level set evolution without re-initialization: a new variational formulation. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 430–436. IEEE, 2005.

Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE transactions on image processing*, 19(12):3243, 2010.

Marc Droske, Bernhard Meyer, Martin Rumpf, and Carlo Schaller. An adaptive level set method for medical image segmentation. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 416–422. Springer, 2001.

Ethan Street, Lubomir Hadjiiski, Berkman Sahiner, Sachin Gujar, Mohannad Ibrahim, Suresh K Mukherji, and Heang-Ping Chan. Automated volume analysis of head and neck lesions on ct scans using 3d level set segmentation. *Medical physics*, 34(11):4399–4408, 2007.

Frank Heckel, Olaf Konrad, Horst Karl Hahn, and Heinz-Otto Peitgen. Interactive 3d medical image segmentation with energy-minimizing implicit functions. *Computers & Graphics*, 35(2):275–287, 2011.

Li Wang, Chunming Li, Quansen Sun, Deshen Xia, and Chiu-Yen Kao. Active contours driven by local and global intensity fitting energy with application to brain mr image segmentation. *Computerized medical imaging and graphics*, 33(7):520–531, 2009.

Thi Nhat Anh Nguyen, Jianfei Cai, Juyong Zhang, and Jianmin Zheng. Robust interactive image segmentation using convex active contours. *IEEE Transactions on Image Processing*, 21(8):3734–3743, 2012.

Chenyang Xu and Jerry L Prince. Gradient vector flow: A new external force for snakes. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 66–71. IEEE, 1997.

Zeyun Yu and Chandrajit Bajaj. Image segmentation using gradient vector diffusion and region merging. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 941–944. IEEE, 2002.

Bing Li and Scott T Acton. Active contour external force using vector field convolution for image segmentation. *IEEE transactions on image processing*, 16(8):2096–2106, 2007.

Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):681–685, 2001.

Mohamad Awad, Kacem Chehdi, and Ahmad Nasri. Multicomponent image segmentation using a genetic algorithm and artificial neural network. *IEEE Geoscience and remote sensing letters*, 4(4):571–575, 2007.

Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3d medical image segmentation: a review. *Medical image analysis*, 13(4):543–563, 2009.

Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage*, 31(3):1116–1128, 2006b.

Yunmei Chen, Hemant D Tagare, Sheshadri Thiruvenkadam, Feng Huang, David Wilson, Kaundinya S Gopinath, Richard W Briggs, and Edward A Geiser. Using prior shapes in geometric active contours in a variational framework. *International Journal of Computer Vision*, 50(3):315–328, 2002.

X Han, C Xu, and JL Prince. Fast numerical scheme for gradient vector flow computation using a multigrid method. *IET Image Processing*, 1(1):48–55, 2007.

Wenwu Yang and Jieqing Feng. 2d shape morphing via automatic feature matching and hierarchical interpolation. *Computers & Graphics*, 33(3):414–423, 2009.

Jörgen Ahlberg. Active contours in three dimensions, 1996.

Bram van Ginneken, Wouter Baggerman, and Eva M van Rikxoort. Robust Segmentation and Anatomical Labeling of the Airway Tree from Thoracic CT Scans. In Dimitris Metaxas, Leon Axel, Gabor Fichtinger, and Gábor Székely, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008*, pages 219–226, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-85988-8.

Yoshitaka Masutani, Thomas Schiemann, and Karl-Heinz Höhne. Vascular shape segmentation and structure extraction using a shape-based region-growing model. In William M Wells, Alan Colchester, and Scott Delp, editors, *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, pages 1242–1249, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49563-5.

Zahra Najafi, Arash Taki, S K Setarehdan, R A Zoroofi, Andreas Konig, and Nassir Navab. A new method for automatic border detection in IVUS images and 3D visualization of segmented frames. *Conference on Visual Information Engineering*, pages 1–4, 2007.

TF F Cootes, A Hill, CJ J Taylor, and J Haslam. Use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12: 355–365, 1994. ISSN 02628856.

Jan Ehrhardt, René Werner, Dennis Säring, Thorsten Frenzel, Wei Lu, Daniel Low, and Heinz Handels. An optical flow based method for improved reconstruction of 4D CT data sets acquired during free breathing. *Medical Physics*, 34(2):711–721, 2007. ISSN 00942405.

Ji\v Rí Hlad and Eduard Gröller. Direction-Driven Shape-Based Interpolation of Volume Data. *Proceedings of Vision, Modeling, and Visualization (VMV)*, D(January):113–120,521, 2001.