Interpolation of 3D slice volume data for 3D printing

Samuel Littley and Irina Voiculescu
Department of Computer Science, University of Oxford

Abstract

Medical imaging from CT and MRI scans has become essential to clinicians for diagnosis, treatment planning and even prevention of a wide array of conditions. The presentation of image data volumes as 2D slice series provides some challenges with visualising internal structures. 3D reconstructions of organs and other tissue samples from data with low scan resolution leads to a 'stepped' appearance. This paper demonstrates how to improve 3D visualisation of features and automated preparation for 3D printing from such low resolution data, using novel techniques for morphing from one slice to the next. The boundary of the starting contour is grown until it matches the boundary of the ending contour by adapting a variant of the Fast Marching Method. Our spoke based approach generates scalar speed field for FMM by estimating distances to boundaries with line segments connecting the two boundaries. These can be regularly spaced radial spokes or spokes at radial extrema. We introduce clamped FMM by running the algorithm outwards from the smaller boundary and inwards from the larger boundary and combining the two runs to achieve FMM growth stability near the two region boundaries. Our method inserts a series of uniformly distributed intermediate contours between each pair of consecutive slices from the scan volume thus creating smoother feature boundaries. Whilst hard to quantify, our overall results give clinicians an evidently improved tangible and tactile representation of the tissues, that they can examine more easily and even handle.

Introduction

The performance of any segmentation tool is limited by the quality of the data on which it has to work. Our own segmentations [4] often deal with data sets with inter–slice spacing of $2.5mm$ or higher. Even when the segmentation is proven to be close to a ground truth, any 3D reconstruction of that data [1] will lack smoothness due to the limitations of the original scan. In order to mitigate such limitations, we add interpolated slices in order to smooth the contours of the segmented shape. 3D–printed models of reconstructed organs look and feel more realistic, which lead to improved pre–op preparation.

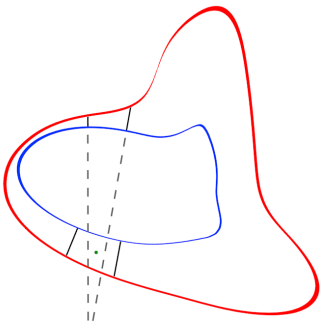
The interpolation is carried out by morphing two 2D contours in consecutive slices. The boundary of the starting contour is grown until it matches the boundary of the ending contour. This is achieved adapting a variant of the Fast Marching Method (FMM), a special case of level set methods [2, 3]. The intermediate contours are then distributed uniformly across the third dimension of the inter–slice gap, thus re–creating a 3D shape.

FMM

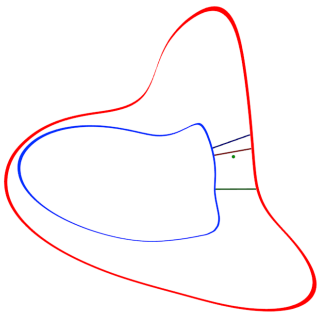
FMM numerically approximates the development of fronts propagating through $\mathbb{R}^n$ space ($\mathbb{R}^2$ in this case) by tracking the arrival time of the front at each point of a lattice. When running the algorithm, each point in the 2D domain is marked as to whether it has been accepted or not. We seed the algorithm with some starting points or a starting region; the seed(s) are accepted and marked to have arrival time 0; every other point can be considered to have infinity for arrival time $u(x)$. The final stage of initialisation is to compute an updated arrival time for each x adjacent to some accepted x'. The updated $u(x)$ is computed using a discrete numerical approximation to the Eikonal equation of FMM [2]; the resultant value depends on the arrival times of the points around it which have been accepted. The updated value is only kept if it is less than the previous value. On each step, the algorithm takes the (non-accepted) point with the lowest arrival time, marks it as accepted and updates the $u(x)$ values for its neighbours. The algorithm terminates when all the points have been accepted.

To use FMM for morphing, we use a starting region in, say, the horizontal plane as the seed, and run the algorithm allowing the inner region to grow into the outer region. To create the snapshots, we take the greatest arrival time seen t_{max} (inside the outer region), divide it by the number of layers we want to add, and compute level sets at multiples of this time. This gives the first level set at $t = 0$ as the starting region, the last level set at $t = t_{max}$ as the ending region, and the regions in between as morphed steps in between the two. The results given by this process contain vertical discontinuities where the growing level set reaches the boundary of the outer region at some $t < t_{max}$. An alternative approach here would be to take t_{max} as the average (mean or median) of the arrival times of the level set at each point along the boundary of the outer region, and clamp arrival times greater than t_{max} to be equal to t_{max}. This would make the vertical discontinuities smaller, at the expense of introducing horizontal discontinuities.

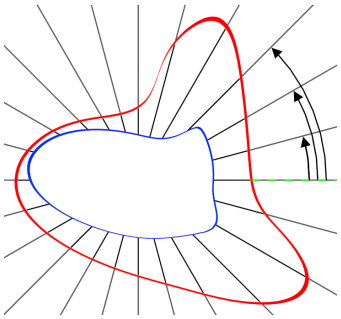
Generating speeds for FMM The main body of the work in using FMM is in generating the scalar speed field used to compute the arrival times at the points. We generate this field in such a way that the growing level set arrives at each point on



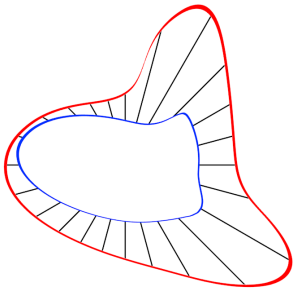
(a) If distance to the infinite line defined by the spokes were used here, instead of distance to the line segment, the point marked by a green dot would be closer to the lines formed by the spokes on the opposing side of the region, giving an incorrect speed.



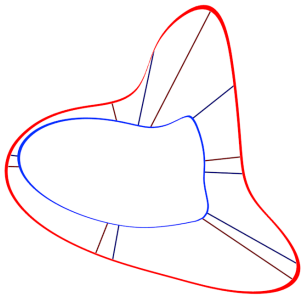
(b) Here, interpolating the red and blue spokes would give the wrong result, despite them being the nearest spokes to the green point. The red and green spokes should be used instead, because the point lies between them.



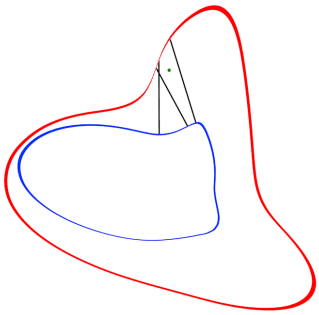
(c) Radial spokes are stored ordered by the angle between the spoke and some reference line (shown as a green dotted line) to allow for easy searching and simple interpolation.



(d) Radial spokes created at regular angles around the centroid (here, every $15°$).



(e) Radial spokes created at extrema of radius from the centroid (dark red spokes are spawned from the outer region boundary, dark blue from the inner).



(f) Non-radial spokes (generated perpendicular to the region boundary) can intersect. The point marked in green here is (approximately) equidistant from all three spokes; by moving in different directions, different pairs of spokes will be used for interpolation, leading to discontinuities in the speed field.

Figure 1: How spokes work

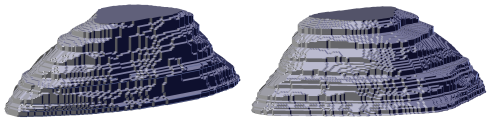
the boundary of the outer region at almost exactly the same time. To do this, we must have the level set grow faster where the distance between the inner and outer region boundary is larger; this means that the speed should be roughly proportional to the distances between the boundaries.

Clearly, this requirement is difficult to specify in concrete terms — whilst in the continuous case the growth of the level set occurs along the normal to the boundary of the level set (and this is approximated in the discrete FMM algorithm), we cannot say in advance what the boundary will be at any given iteration. As such, we do not know in what direction the level set will grow, and thus in which direction to compute the distance between the boundaries.

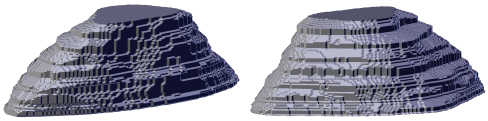
Spoke based approach

We address this unknown by generating "spokes" between points on the inner and outer boundaries (Figure 1). These spokes have a fixed length (i.e. the distance between the endpoints) from which we generate a speed for the spoke (we directly use the length of the spoke, although other schemes could feasibly be devised). We then interpolate between the spokes to compute speeds for all the discrete points in the scalar field (specifically, by interpolating between the nearest two spokes to either side of each point). By increasing the number of spokes generated, in theory the accuracy of the generated speed field can be increased, although given that this scheme can only approximate an appropriate speed field, there is little benefit to creating overly large numbers of spokes. Systematic experiments (Figure 2) determined that 64 spokes gave reasonable accuracy.

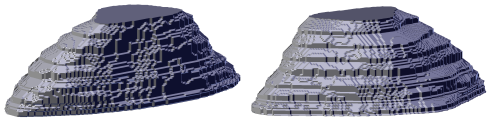
It is important that the distance between spokes is computed as the distance to the line segment formed by the spoke, rather than the distance to the infinite line passing through the two endpoints of the spoke. Otherwise, the speed at a given point



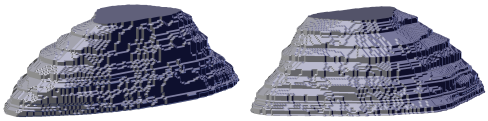
(a) 8 spokes



(b) 16 spokes

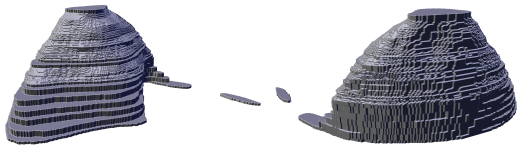


(c) 32 spokes

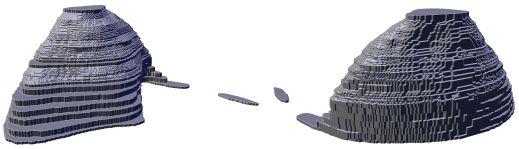


(d) 64 spokes

Figure 2: Renderings of part–liver models, each with 7 interpolated layers, produced using the FMM algorithm with speeds generated from varying numbers of regularly spaced radial spokes. For the models produced with fewer spokes, there are more areas with vertical drops, where some parts of the growing region meet the boundary of the outer region much sooner than others.



(a) This model was produced using the clamped FMM algorithm, using regular spokes. There are almost no vertical discontinuities (the vertical faces visible on the back of the model are due to the lack of x-y resolution in the original images).



(b) This model, produced with clamped FMM using radial extrema spokes, is almost identical to that in Figure 3(a). This suggests that the clamping algorithm does not necessarily require as much detail in the speed field as we currently give it.

Figure 3: Smoother FMM–generated models (front and back views)

may be based on a spoke from the opposing side of the regions, simply because the infinite line formed by that spoke passes very close to the point in question (see Figure 1(a)). Similarly, we must use spokes from either side of the point rather than simply the nearest two spokes (such that the point falls within the region bounded by the two spokes and the two sections of the region boundaries between the endpoints of the spokes, Figure 1(b)). Otherwise, as the point moves, it would switch to a different pair of spokes partway through the interpolation (leading to discontinuities in the speed field). When implemented correctly, the pair of spokes providing the speed at a point only changes when the point falls almost exactly on one of the two spokes. The point then takes the value of that spoke, with no influence from the other spoke in the pair which changes.

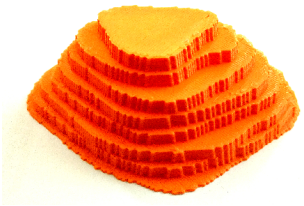
The full paper will discuss and compare how spokes can be generated radially to some central point (Figure 1(d)), or by starting at radial extrema (Figure 1(e)), or non radially (Figure 1(f)). In our implementation, a central point is either the computed centroid of one of the regions, or the average of both. The use of radial spokes has the advantage of making the finding of the nearest spokes and the respective interpolation much easier, as the spokes are indexed and sorted by the angle between the spoke and some reference line, and so can be iterated in order or looked up using binary search (Figure 1(c)).

Clamping FMM to endpoints

Using the algorithm detailed above we get a reasonable yet hard to quantify result (Figure 2), with smooth gradients leading away from the smaller contour. This approach does not have C0 continuity where the slope meets the next layer. This is caused by the growth of the level set reaching some parts of the boundary of the outer region before other parts. Since the extra interpolated layers are based on the arrival time at a point as a fraction of the largest such arrival time, certain points along the boundary are arrived at "too early", leading to discontinuities and steps. This section deals with the issue of making the level set *reach every point on the boundary at the same time*.

Knowing that the level set "leaves" the boundary of the starting region at the same time, we also run FMM inwards, starting at the outer boundary. This thus gives us a set of arrival times where the times at the boundary of the outer region are all zero. We can then invert the arrival times (by subtracting each time from the maximum time) to have the arrival times in the same order as the outward growing iteration.

The problem then becomes how to combine these two FMM runs such that we use the outward growing iteration when at/near the inner region boundary, the inward growing iteration for the outer region boundary, and some combination of the two in between. It suffices here to use the distance from each boundary to define the ratios of the two iterations we apply at each point. To approximate these distances, we run two more FMM iterations (one in each direction), both with a constant



(a) 3D printed section of part–liver model, generated with no interpolation and no smoothing.

(b) 3D print of Figure 4(a) with interpolated layers and *no smoothing*.

(c) 3D print of Figure 4(a) improved with interpolation, decimation and Laplacian smoothing.

Figure 4: Photographs of 3D printed models

speed field. The arrival times can be considered to be representative of the distance from the boundary.

In order to simplify the logic, we scale/normalise the arrival times for each iteration to the range $[0..1]$ by dividing each time by the maximum time for the iteration. The result of this algorithm is a set of arrival times which are all 0 at the inner region boundary, and all 1 at the outer region boundary, and smoothly changing in between.

Results and limitations

The clamped FMM algorithm results in C1 continuity through each set of interpolated layers, and C0 continuity where the sets of interpolated layers meet, as shown in Figure 3. The full paper will also discuss how to finish the extremities of a reconstructed feature to create a smooth appearance (rather than a flat region at the top). The method works best when the region on one of the layers fits inside the region on the other layer. If neither region fits inside the other, we need to take a slightly different approach of scaling one of the layers so that the region on one layer fits inside the scaled region on the other. For each region, we grow a containing box inside it with the aspect ratio of the bounding box of the other region until it meets the boundary of the region on two opposing sides or corners.

Our algorithm is most successful with components which appear as single regions on each layer. Unsurprisingly, it fails to connect regions which identify the same feature but are disjoint between layers (e.g. ribs which are long and thin, and almost perpendicular to the z axis). It also fails in the case of a branching feature (e.g. a network of blood vessels); it is possible for two regions on one layer join into a single region on another layer (but all 3 regions have the same component identifier).

Photographs of the physical difference the algorithm makes are illustrated in Figure 4. These took approximately 1hr 30 mins to print on a home–assembled RepRap Mendel™, and used approximately $12.5m$ of $1.75mm$ diameter filament, at a cost of approximately $£1$. These are extreme cases simplified for the space requirements of this abstract, although intermediate models with a variety of different properties can be generated. On the basis of Figure 4(c) we recommend Laplacian smoothing followed by decimation for all such models in order to maintain the final 3D printed model within manageable parameters.

Conclusions and future work

The methods detailed in this paper specifically attempt to have the generated surface pass through the exact boundaries of the feature selections on each slice. These selections do not truly represent the boundary of the feature itself due to the fact that each slice is an average over a thickness rather than an actual planar snapshot. A good next step in the process would be to relax this requirement in some way, and generate surfaces which are close to, but do not necessarily align exactly with these boundaries. Future developments will also focus on improved means of matching feature selections where this is not the case, or where one feature selection forms a contiguous region in 3D, but appears as multiple 2D regions on slices.

In the long term, we aim to set up a systematic study of the benefits of reducing radiation in knee and shoulder joint scans whilst still allowing surgeons to handle 3D printed models of the joints on which they are about to operate.

References

[1] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[2] James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[3] Varduhi Yeghiazaryan and Irina Voiculescu. The use of fast marching methods in medical image segmentation. Technical Report CS-RR-15-07, Department of Computer Science, Oxford, UK, 2015.

[4] Varduhi Yeghiazaryan and Irina D. Voiculescu. Automated 3D renal segmentation based on image partitioning. In *Proc. SPIE 9784, Medical Imaging 2016: Image Processing*, page 97842E. SPIE International Society for Optics and Photonics, 2016.