Progress Report: Handling Structural Inconsistencies in CT Image Reconstructions

Cameron Stevenson

Supervisor: Ramakrishnan Mukundan

June 3, 2021

1 Overview

This is a progress report for a Computer Science Honours project related to mesh reconstructions from CT image stacks. I will recap the most relevant content of my project proposal and literature survey, and describe my progress in investigating improvements on prior methods. I will also take this opportunity to discuss techniques yet to be investigated and my rationale for them. Finally I will outline my goals and schedule for the remainder of the year.

2 Introduction

Medical imaging modalities such as CT scans and MRI produce stacks of images depicting regions of different tissue. These image stacks can be used to reconstruct the 3D structure of internal organs. The renders or models produced are useful in diagnosis, treatment, education, surgical simulation, and robot assisted surgery [1, 2]. The process of producing renders or reconstructions can be improved by reducing manual user input, improving reconstruction accuracy, or improving computational efficiency.

Branching structures can be problematic to reconstruct and are a common issue in prior methods. My project aims to improve the accuracy of reconstructions involving branches and other problematic structures. In particular the lungs are used as an organ of focus.

3 Background

Most of my time in this project so far has gone towards a literature survey, and so I have included the essential details of common methods below, with more elaboration given to the approach relevant to this project. If more detail is necessary refer to my literature survey [1].

[1] `https://github.com/cstevenson3/cosc470writing/blob/main/survey.pdf`

3.1 Generic Methods

The simplest form of rendering in medical imaging is direct volume rendering. The pixels in the image stacks are treated as voxels, forming a 3D grid of intensity data. Rasterisation or raycasting methods can be applied to these voxels to give a render. These renders are easy to understand and detailed, however they are computationally expensive.

Mesh rendering is preferred for real-time applications, but this requires a mesh reconstruction first. The mesh represents where boundaries of tissue lie in 3D space. To begin with segmentation of images must be done to find boundaries in 2D space. Segmentation methods are usually specific to an organ of interest and are tuned as such. A segmentation gives contours of points representing the boundaries between tissue. These points are then used for mesh reconstructions.

Point cloud methods for mesh reconstruction are adopted from applications like laser scanning, where a lot of points are dumped in as input and the algorithm has to infer which are connected. The contour points from segmentation can be used as input in a point cloud method to produce a reconstruction. Point cloud methods typically interpolate from points and don't use them directly in the output mesh.

Marching cubes is another method for mesh reconstruction. Boundary voxels are found, and their neighbouring voxels are considered in a lookup table to decide what local triangulation should be made for that voxel. All the triangles are then connected for the whole mesh. Marching cubes produces jagged results and ambiguity can lead to defects.

3.2 Correspondence Methods

A more direct approach to mesh reconstruction is to decide which contours should be connected, then triangulate between them, with an optional point correspondence step in the middle. Such an approach produces output meshes which contain the original points. The result is a method which gives computationally faster renders than direct volume rendering, potentially more accurate meshes than point cloud methods, and a smoother mesh than marching cubes. Mackay [1] presents one such paper to use this approach, and my project focusses on extending his method.

3.2.1 Contour Correspondence

Contour correspondence takes as input all the contours in a segmentation of an image stack, and produces as output matchings of contours which should be connected. In a simple case a matching would be one contour in one slice and a similarly positioned contour in an adjacent slice. In more complicated cases such as branching there may be one-to-two matchings. Metrics for matching include centroid position closeness, size and shape similarity etc.

Correspondence algorithms can operate on different levels:

- local: looking only at adjacent slices each iteration
- global: looking for the best matching across all slices each iteration
- growing: keeping a hierarchy of connected components, and adding unmatched contours onto these components each iteration

3.2.2 Point Correspondence and Triangulation

With contour correspondences established, points in matched contours must themselves be matched to create a triangulation for a mesh. Points can be matched by Euclidean distance, and by their index within a contour.

Mackay [1] proposes Dynamic Time Warping (DTW) as a method of point correspondence. DTW matches features on the same structure which has been slightly warped, making it suitable for contours on adjacent slices which should differ only slightly. DTW generates a warping path going around both contours, producing point matches as it goes. Euclidean distance is used as the cost function, and some constraints are put on the warping path to keep it progressing on both contours in case Euclidean distance is not suitable for a section.

Triangulation is relatively simple, taking point correspondences as edges and filling gaps where necessary. Ideally the two contours have similar sizes otherwise many triangle fans will be created, which gives a rougher mesh.

3.2.3 Branching Problem

Branching structures are problematic to reconstruct. When scanned they show up as one contour in one slice and two contours in the next slice. There is ambiguity in contour correspondence, as this could represent many possible structures (a branch, a bend, the end of one tube near another tube). Mackay [**?**] looks for lung branches as one contour approximately splitting in half into two nearby contours.

When it comes to point correspondence, branches affect the position of points drastically and so correspondence metrics are skewed. In Mackay's [1] method DTW requires two ordered contours as input, so the two-contour side of the branch must be merged into a single contour first. DTW does not handle all branching cases well, with some twisting and too many points matched to a single point. Fixing this is left as further research.

4 Progress

Following the literature survey, I found a week to begin trialling new techniques to improve correspondence methods, with the intent of handling branching structures better. I am using Mackay's method and C++ code as a base to work from, adding modifications as I go. Fortunately Mackay's code is well structured and easy to drop in modifications to test. I took one day to set up my environment and run the original code successfully. I then took four days to try my first idea in python, port it over to C++, and substitute it into Mackay's code.

4.1 Point Angle Rationale

The first idea I tried began with me observing twisting produced by DTW in branching cases. I hypothesised two reasons for this. Firstly the choice of starting point in each contour was the leftmost point (lowest x value). For similarly shaped contours this should give roughly the same point in each contour (as a person would understand it). However in branching cases where the two-contour merged contour is oriented mostly in the z-direction, the leftmost point could be either on the top or bottom contour, whereas on the single contour it would be towards the middle. This

would cause triangle fanning at the start and end of DTW. Secondly, a structure progressing at an angle through the CT scan planes would have contours shifted away from each other. Since DTW relies on Euclidean distance as a cost function, I hypothesise this shift would bias which points are matched.

For the above two reasons my first trialled idea was to calculate the centroids of matched contours, and translate them to the origin from the perspective of point correspondence. This allows for choosing of a starting point near the x-axis (the rightmost point), and so starting points are better aligned between contours. With this translation made, it opened up the opportunity to try an alternative pathing algorithm. My idea was to use point angle from the now shared centroid to match points across contours. In ideal conditions each contour would have one point per angle, but this is not the case, particularly in merged contours. For the pathing algorithm I required monotonicity, so for each point if a prior point on the ordered contour had a greater angle I used this as the angle for the current point. This produces segments of points with the same angle, and so I added in a second metric to keep the path moving. This metric was progression along the contour, measured simply as how many points had been passed out of the total number of points. A weighted sum of the angle and progression metrics is used in the pathing algorithm. A current pair of points is kept, and points on each contour are leapfrogged between such that whichever point is behind on a metric is advanced to the next point. All pairs are recorded to give the point correspondence. I refer to this method as point-angle.

4.2 Point Angle Testing

5 Future

Future...

References

[1] D. Mackay, "Robust contour based surface reconstruction algorithms for applications in medical imaging," 2019.

[2] R. Mukundan, "Reconstruction of high resolution 3d meshes of lung geometry from hrct contours," in *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 247–252.