SLAM in Deformable Environment **2021 Winter School**

Project: Aortic 3D Deformation Reconstruction

Total Marks: 13 marks (+2.5 bonus marks)

University of Technology Sydney (CAS) Due date: **Day 5 (9 Jul. 2021)**

1 Introduction

Endovascular interventions provide a lower risk alternative to open surgery for cardiovascular diseases. The minimal incision benefits patients and broadens the options for age groups to receive the interventional treatment [1]. However, this therapy is challenging due to the requirement of a precise catheter manipulation in a highly dynamic surgical environment while limited visualisation is available [2]. Hence, the intra-operative recovery of vascular structure is needed for providing a 3D guidance on catheter surroundings and assisting catheter navigation. Although a 3D vessel structure is available pre-operatively by computed tomography (CT) scans, the vessel is soft and deforms during the procedure. This means the pre-operative data cannot be used directly to perfectly reflect aortic current shape, and therefore, the deformation should be updated intra-operatively [3].

The purpose of this project is to inform you a recent framework [4] that recovers aortic 3D deformation using a pre-operative model and two intra-operative X-ray images from different viewing direction. The pixels which present the aortic wall contours are selected as the observed features. A 2D-3D registration approach is applied to calculate correspondence according to these features. The vessel's reconstruction is formulated as a nonlinear optimization problem based on the embedded deformation graph approach [5].

In this project, you will need to implement the framework using Matlab. To make this project easy, we provide a pre-operative 3D mesh model. Some data and code are also provided.

2 Methodology

2.1 Feature Selection

In this project, the contour pixels of each X-ray images are provided (Fig. 1).

Data:

- Contour pixels: *obs_p* in *Observation.mat.*
- Contour tangent: *obs_t* in *Observation.mat*

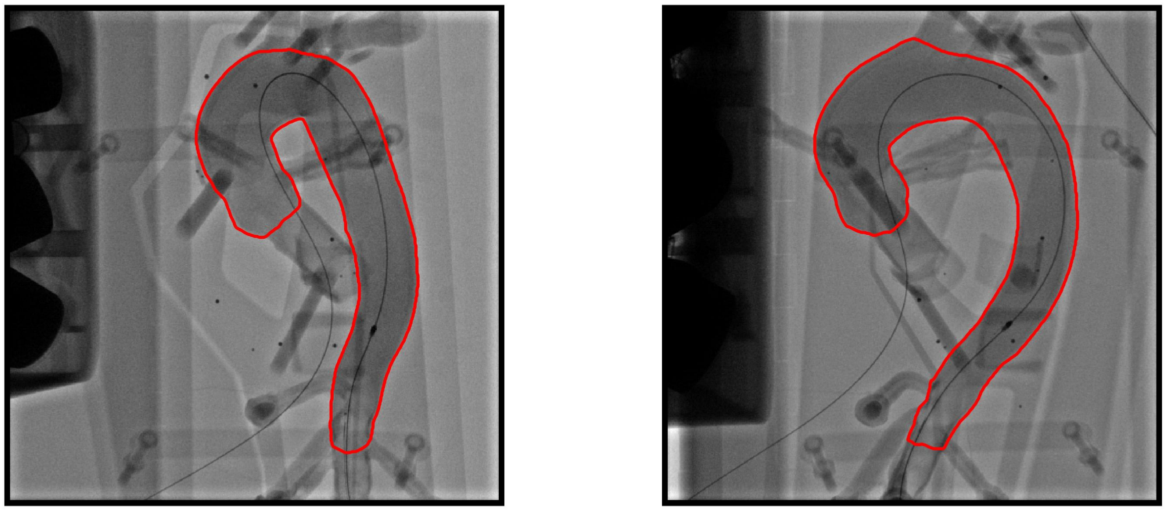


Figure 1: X-ray images and contour pixels (red).

- Images: *Images.mat*

2.2 Camera Projection

Having the selected contour pixels, the deformation is calculated by minimising the misalignment between the observed contour and the contour of model projection. Here, we ignore the camera distortion and using a simple orthographic camera projection to approximate the camera model. Suppose $\mathbf{p} \in \mathbb{R}^3$ denoting a model vertex, $\{\mathbf{R} \in \mathrm{SO}(3), \mathbf{T} \in \mathbb{R}^3, s \in \mathbb{R}\}$ denoting the relative transformation and the scale factor, the camera projection is formulated as:

$$g(\mathbf{p}) = s\mathbf{U}\mathbf{R}(\mathbf{p} + \mathbf{T}), \tag{1}$$

where $\mathbf{U} = \left[\mathbf{I}_2, \mathbf{0}_{2\times1}\right]$ gets the two upper rows of a full rotation matrix $\mathbf{R}$.

You need to write the projection function according to the given data (**0.5 marks**):

- Rotation, translation and scale: R, t, s in *Camera.mat.*

2.3 Pixel-vertex Correspondence

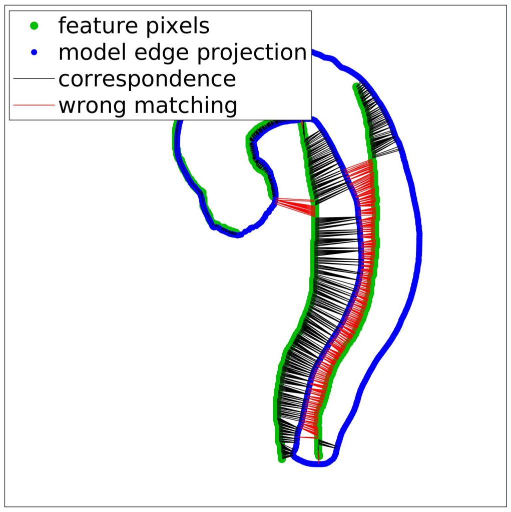
We find the correspondence between the contour of model projection and contour pixels from image. The edge of model projection is obtained by first project all vertices into image frame using (1), then calculate the edge of the projected point cloud using the alpha-shape method [6].

You need to calculate the edge of model projection (**0.5 marks**):

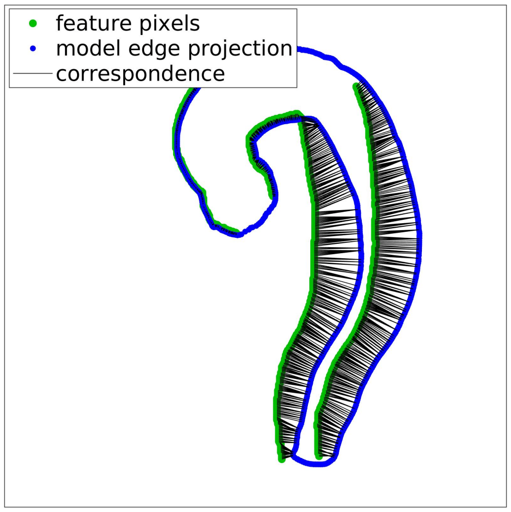
- Please refer to `https://au.mathworks.com/help/matlab/ref/alphashape.html` for 2D alpha shape.

- Please tun and choose appropriate alpha radius.

In this way, we simplify the 2D-3D registration problem to a 2D-2D registration problem. Here, we consider the normal vector as well as the distance to perform the registration, as only use the distance increases the chance of wrong matching (Fig. 2).



(a) Without geometric information



(b) With geometric information

Figure 2: An illustration of improvement on matching result without/with normal vectors. Green points: feature pixels, blue points: edge points of model projection, black lines: correct matching result, red lines: wrong matching using pure Euclidean distance.

Suppose $\mathcal{P} = \{\cdots, g(\mathbf{p}_i), \cdots\}$ denoting a set of edge points from model projection, $\mathcal{Q} = \{\cdots, \mathbf{q}_j \cdots\}$ denoting a contour pixels from a image. For each $\mathbf{q}_j$, we first calculate the set of potential corresponding points $\tilde{\mathcal{P}}_i \subset \mathcal{P}$ according to following rule:

$$\tilde{\mathcal{P}}_j = \{g(\mathbf{p}_i) \; | g(\mathbf{p}_i) \in \mathcal{P}, d(g(\mathbf{p}_i), \mathbf{q}_j) < d_{\text{th}}, \; \theta(\mathbf{n}(g(\mathbf{p}_i)), \mathbf{n}(\mathbf{q}_j) < \theta_{\text{th}}\}, \tag{2}$$

where $\mathbf{n}(\cdot) \in \mathbb{R}^2$ denotes the 2D normal vector in image frame, $d(\cdot)$ represents Euclidean distance between two points, $\theta(\cdot)$ represents the angle between two normal vectors. d_{th} and θ_{th} are distance and angle threshold. If $\mathcal{P}_i = \emptyset$, we skip this feature, otherwise, the 2D corresponding points w.r.t. $\mathbf{q}_j$ is calculated by simply using the minimum distance:

$$\tilde{\mathbf{p}}_j = \underset{g(\mathbf{p}_i) \in \mathcal{P}_j}{\arg\min}\, d(g(\mathbf{p}_i), \mathbf{q}_j). \tag{3}$$

You need to write this correspondence (**2 marks**):

- The function for 2D normal vector is provided *LineNormals2D.m*. Please refer to `https://au.mathworks.com/matlabcentral/fileexchange/32696-2d-line-curvature-and-normals` for more details.

2.4 Deformation reconstruction

The deformation is reconstructed based on embedded deformation graph (ED graph) [5]. Its main idea is to use the weighted average of local affine transformations to represents the whole deformation. The deformation graph consists of a set of uniformly scattered sparse graph nodes whose positions $\mathbf{g}_j \in \mathbb{R}^3$ are usually calculated by down-sampling the model vertices [7]. The local transformation around each ED node can be calculated using an affine matrix $\mathbf{A}_j \in \mathbb{R}^{3\times 3}$ and a translation vector $\mathbf{t}_j \in \mathbb{R}^3$. For any vertex $\mathbf{p}_i$ from a 3D model, its new position after deformation is calculated using its K nearest ED nodes:

$$\hat{\mathbf{p}}_i = \sum_{j=1}^{K} w_j(\mathbf{P}_i)\left[\mathbf{A}_j\left(\mathbf{p}_i - \mathbf{g}_j\right) + \mathbf{g}_j + \mathbf{t}_j\right], \quad w_j(\mathbf{p}_i) = \left(1 - \|\mathbf{p}_i - \mathbf{g}_j\|/d_j\right)/n_j, \tag{4}$$

where weight $w_j(\mathbf{p}_i)$ is used for quantifying the influence of ED node j to vertex i, d_j is the distance to the $K+1$ nearest ED node, and n_j is the normalization factor.

Given the control points, ED graph estimates the deformation parameters (affine matrices and translation vectors) by minimizing the energy function:

$$E = w_{\text{rot}}E_{\text{rot}} + w_{\text{reg}}E_{\text{reg}} + w_{\text{ob}}E_{\text{ob}}. \tag{5}$$

More details about conventional ED graph can be found from [5]. We also provide our tutorial slides about ED graph.

For aortic deformation reconstruction, the observation term is modified such that ED graph deforms the blood vessel by penalising the misalignment between the reconstruction projection and the observation. This is easy when we have the vertex-pixel correspondence, which is used as the control points here. Assuming for image I_l, a feature and its corresponding point from the 3D model are $\{\mathbf{q}_{l_i}, \tilde{\mathbf{p}}_{l_i}\}$, then the observation term is the sum of squared distance between all corresponding points

$$E_{\text{ob}} = \sum_{l=1}^{L} \sum_{i\in\mathbb{N}(l)} \|g(\hat{\mathbf{p}}_{l_i}) - \mathbf{q}_{l_i}\|^2, \tag{6}$$

where $\hat{\mathbf{p}}_{li}$ is calculated by $\tilde{\mathbf{p}}_{l_i}$ using (4), $\mathbb{N}(l)$ is the index set of pair-wise points with correspondence in this image, $L = 2$ is the total number of images.

The estimation of (5) can be processed using an iterative approach e.g. Gauss-Newton algorithm.

You need to modify given implementation of the conventional ED graph to recover aortic 3D deformation. (**8 marks**):

- For conventional ED graph (which use 3D control points), please refer to our tutorial slides.
- You need to initialise the ED nodes (e.g., calculate the node position) using aortic pre-operative model. (**0.5 marks**).

- You need to calculate the control points using vertex-pixel correspondence. (**1 marks**).
- You need to rewrite the residual function *calculate_Error.m* according to new observation term (6) (**1.5 marks**).
- You need to calculate the Jacobian formulation according to the residual function (**2 marks**) and rewrite the Jacobian function *calculate_Jacobian.m* (**2 marks**).
- You need to update the model vertices using the calculated deformation parameters $\mathbf{A}_j, \mathbf{t}_j$ and plot your result with the ground truth (**1 marks**). We also provide a function to calculate the 3D reconstruction error.
- You can also try to tun the weights and some other parameters to improve your result.

3 Additional Improvements

Congratulations, now you know the basic idea on how to recover the aortic 3D deformation based on ED graph. We now introduce additional parts to improve our current basic framework. You will get some bonus marks here.

3.1 Correspondence Improvement

Previously, we calculate the correspondence for control points once before iteratively estimating the deformation parameters using Gauss-Newton method. You can consider update the correspondence and model vertices within each Gauss-Newton iteration. **1 bonus marks** will be given if you finish this part (**0.5 for correspondence and 0.5 for updating model vertex**) .

3.2 Pre-operative Model Segmentation

We also provide the corresponding CT data, with which you can segment your own 3D pre-operative model using ITK-Snap and Meshlab. Please refer to our tutorial on CT segmentation using ITK-snap. **1.5 bonus marks** will be given for those students who finish this part (**1 for CT segmentation and 0.5 for getting exterior surface using Meshlab**).

4 Result

To get full marks, you need to provide your code (**2 marks**). A simple report describing your project are also encouraged. On the report, feel free to say any of your understanding/suggestions on this aortic deformation reconstruction framework.

References

[1] M. Mirabel, B. Iung, G. Baron, D. Messika-Zeitoun, D. Détaint, J.-L. Vanoverschelde, E. G. Butchart, P. Ravaud, and A. Vahanian, "What are the characteristics of patients with severe, symptomatic, mitral regurgitation who are denied surgery?" *European heart journal*, vol. 28, no. 11, pp. 1358–1365, 2007.

[2] T. Kono, H. Kitahara, M. Sakaguchi, and J. Amano, "Cardiac rupture after catheter ablation procedure," *The Annals of thoracic surgery*, vol. 80, no. 1, pp. 326–327, 2005.

[3] L. Zhao, S. Giannarou, S.-L. Lee, and G.-Z. Yang, "Registration-free simultaneous catheter and environment modelling," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2016, pp. 525–533.

[4] Y. Zhang, L. Zhao, and S. Huang, "Aortic 3d deformation reconstruction using 2d x-ray fluoroscopy and 3d pre-operative data for endovascular interventions," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 2393–2399.

[5] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 80, 2007.

[6] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on information theory*, vol. 29, no. 4, pp. 551–559, 1983.

[7] J. Song, J. Wang, L. Zhao, S. Huang, and G. Dissanayake, "Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4068–4075, 2018.