3D Reconstruction of CT Colonography Models for VR/AR Applications using Free Software Tools

Soraia Figueiredo Paulo[1], Nuno Figueiredo[2], Joaquim Armando Jorge[1,3], Daniel Simões Lopes[1]

[1] INESC-ID, Lisboa, Portugal
[2] Colorectal Surgery, Champalimaud Foundation, Lisbon, Portugal
[3] Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

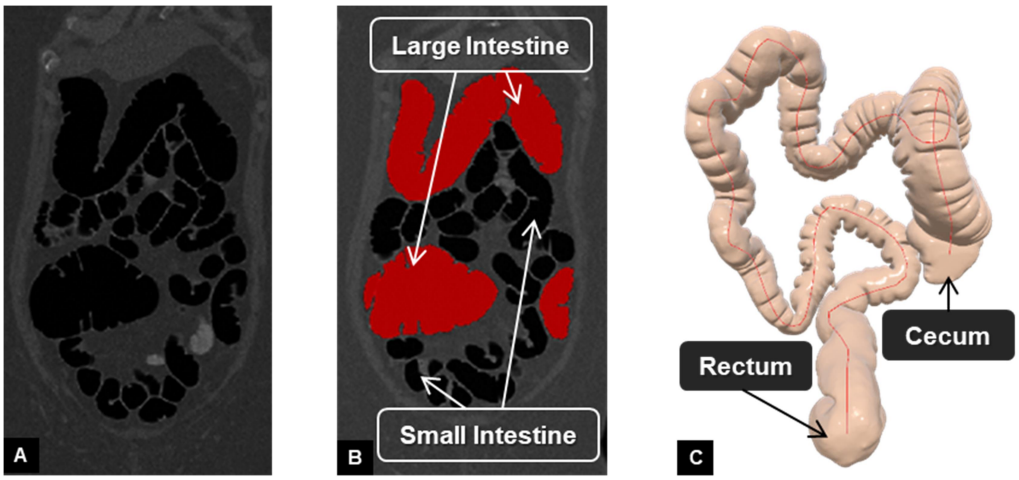

Figure 1. CT Colonography 3D reconstruction: (A) Original CT slice; (B) Segmented colon (red) highlighted in the CT slice; (C) Reconstructed 3D model with centerline (red)

Computed Tomography Colonography (CTC) is an established non-invasive colorectal screening tool that can replace traditional colonoscopy [1, 2]. As an image-based procedure, CTC 3D reconstruction pipeline requires a CT acquisition after intestinal preparation, namely in supine or prone position. Afterwards, a sequence of image processing and mesh adjustment techniques recreate a subject-specific 3D model of a colon from the CT scan images. Similarly to colonoscopy, medical professionals are able to navigate along a planned path between the rectum and the cecum, usually defined by the lumen's centerline. The goal of this tutorial is to show how a well-described geometric modelling pipeline [3] can be used to reconstruct subject-specific 3D models of large intestines from CTC images. The proposed pipeline relies on free and open-source software, which makes it readily accessible to other practitioners, researchers and educators. The reconstructed 3D colon and centerline can then be imported into a game engine, such as Unity 3D (https://unity3d.com/), to be used as digital content for Virtual Reality or Augmented Reality applications, using commercial off-the-shelf apparatus, such as Oculus Rift or HTC Vive. Figure 2 illustrates all pipeline stages that are described in the following sections.

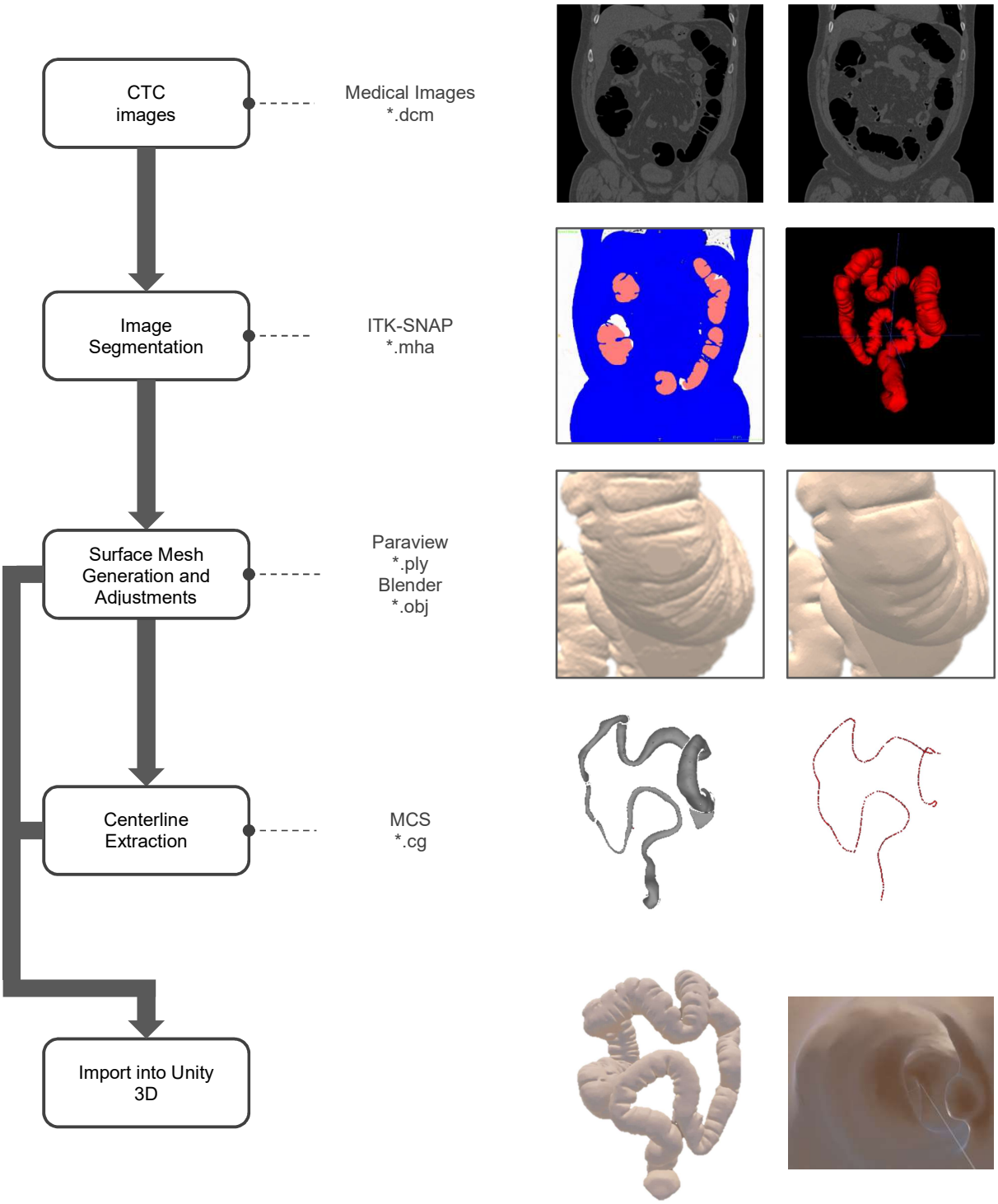

Figure 2. CT Colonography 3D reconstruction pipeline: each stage (box) is associated with a specific software and file extension (indicated by the dashed lines). The final 3D model is a combination of the final mesh and the result of the Mean Curvature Skeletons (MCS) algorithm.

In this tutorial, we used a single CTC dataset available in *The Cancer Imaging Archive* [4] (subject ID: CTC-3105759107), which was acquired in a supine position, had almost no liquid and presented large (>10mm) and quite visible polyps along with several diverticula.

1. Image Segmentation

The first stage consists of identifying the colon inner surface from a set of 2D CTC images using the ITK-SNAP software (www.itksnap.org). CTC images present high contrast between the luminal space (air: black) and luminal surface (wall: light grey), which facilitates image segmentation by global thresholding followed by semi-automatic active contours based on region competition. Both segmentation techniques partition the original grayscale slices into binary images, where the resulting non-null pixel values correspond to the intensities of the luminal space. Note that active contours complement the global thresholding techniques by allowing a set of snakes, which are strategically placed along the region of interest (ROI), to iteratively evolve from a very rough estimate obtained by global thresholding to a very close approximation of the anatomical structure of interest (Figure 3, Figure 4).

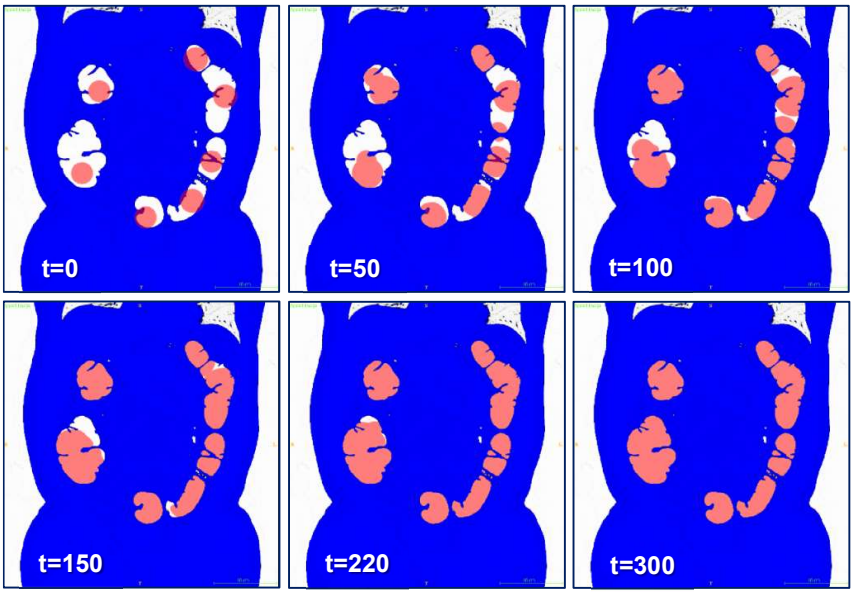

Figure 3. Active contours progression at different time steps (transversal view of the segmented data).

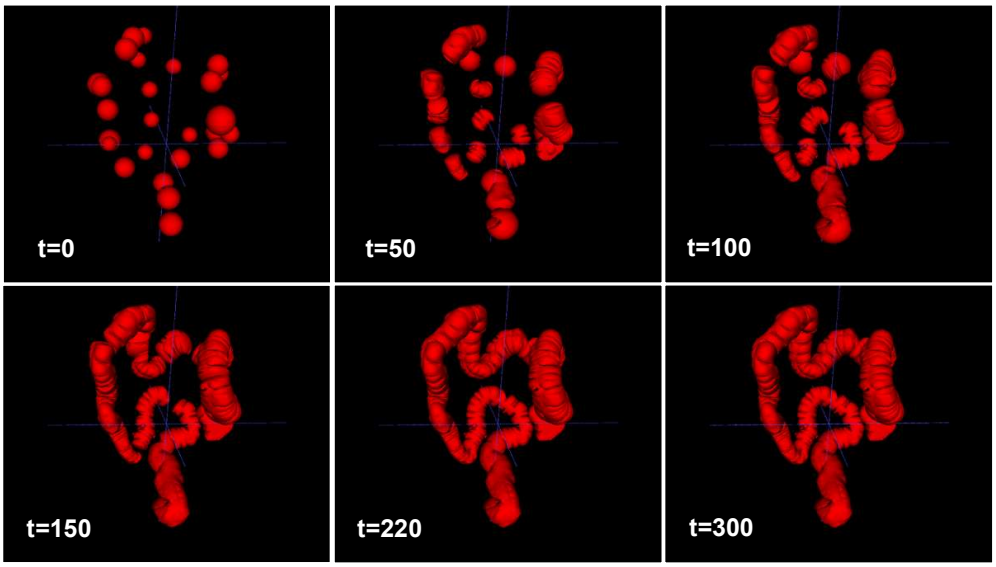

Figure 4. Active contours model progression at different time steps (3D view of evolving contours).

2. Surface Mesh Generation and Adjustments

After segmentation, ITK-SNAP automatically generates a triangular 3D surface mesh using the marching cubes algorithm. Since the segmented images consist of a discrete scalar field, the extracted mesh poses two problems: (i) a stair-step artefact, and (ii) a large number of unnecessary vertices (Figure 5 (A)). Therefore, it is necessary to perform mesh smoothing to attenuate the stair-step artefact, followed by decimation to reduce the number of vertices (or triangles) of the surface mesh. To do so, the user must export the final segmentation metadata file (**.mha*) from ITK-SNAP and import the file into Paraview (www.paraview.org). After applying the smoothing filter, which consists of a low-pass filter carried on for 500 iterations, the resulting mesh enters the decimation process to remove at least 75% of the mesh triangles. Finally, the same smoothing filter is applied to remove eventual decimation mesh artifacts (Figure 5 (B)). The final mesh can then be exported as **.ply (ASCII)* file and converted to an **.obj* file via Blender (www.blender.org).

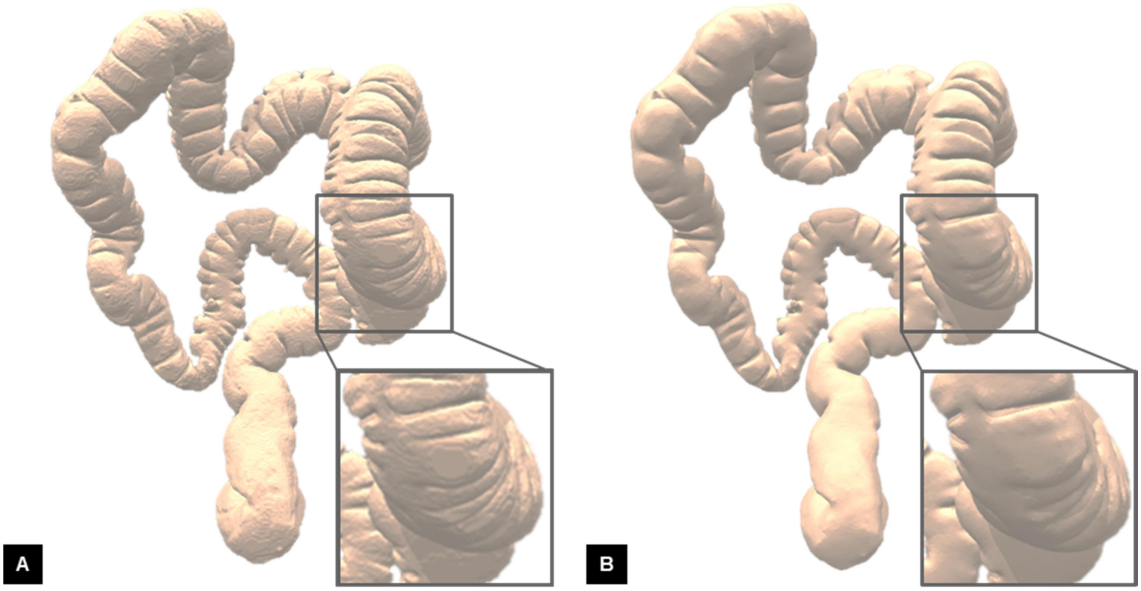

Figure 5. 3D mesh of a colon: (A) before and (B) after Smoothing → Decimation → Smoothing

3. Centerline Extraction

To compute the planned path that guides inner colon navigation, it is necessary to compute the 3D centerline of the colon mesh. Software made available by Tagliasacchi et al. [5] generates a 3D mesh skeletonization using the Mean Curvature Flow (MCF) filter (Figure 6). Once the user imports the **.obj* 3D colon, it is necessary to apply the Voronoi based Medial Axis Transformation (MAT) since it is the input of the MCF filter. For the purpose of our example, we required 26 iterations, using all default values, except for *edge_TH=1,75* . The final centerline model is exported as a *.cg* file.

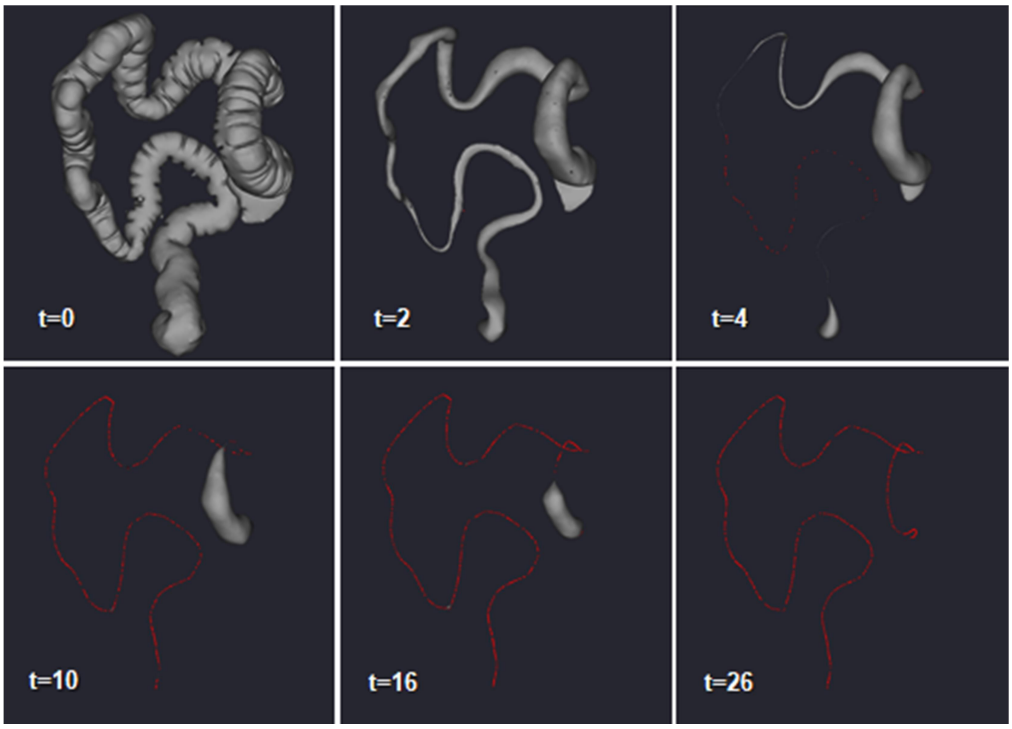

Figure 6. Skeletonization progression at different time instants: MCF iterative evolution.

4. Import 3D colon and centerline into Unity Game Engine

To generate the final model, the *.*obj* file of the 3D mesh of the colon can be imported into Unity 3D game engine (https://unity3d.com/) and dragged into the scene. A GameObject with the same name as the *.*obj* file is automatically created, along with a default GameObject child including the mesh filter, the mesh renderer and the default material components. For the purpose of this tutorial we will refer to the GameObject of the 3D mesh of the colon as *3DColonModel*.

In order to access the interior and exterior of the luminal wall, Inside and Outside meshes are required. To do so, it is necessary to duplicate the default child of the *3DColonModel*, and rename one of the meshes as *Inside* and the other as *Outside.* To display the inner wall, the reverse normals of the Inside mesh must be computed, which requires the user to attach a *ReverseNormals.cs* script to the *Inside* GameObject (http://wiki.unity3d.com/index.php/ReverseNormals), as well as a Mesh Collider component. The color of the rendered *3DColonModel* depends on the material that is associated to the mesh renderer. In our case, we created a material called *ColonColor* and defined the RGBA color as (255,207,171,255). This material is associated both to the Inside and the Outside mesh renderers. Figure 7 illustrates the resulting *3DColonModel*.

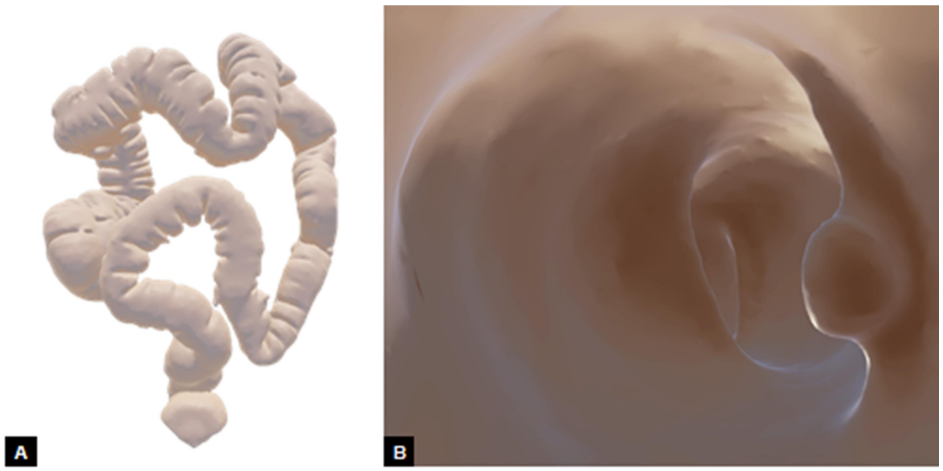

Figure 7. 3D Colon Model views: (A) Outside; (B) Inside.

The planned path is created based on the spline interpolation of the list of points made available through the *centerline.cg* file. To start, the user must create a child GameObject of Outside, called *SplineRoot*. Then, it is necessary to attach a script file to *SplineRoot*, which we called *DrawLines.cs,* responsible for drawing the centerline. This script needs two other scripts: *ControlPointsParser.cs* and *NearestNeighborList.cs* (http://it-medex.inesc-id.pt/project/ctc-models). In order to obtain the list of points, *DrawLines.cs* receives the path for the *centerline.cg* file and sends it to *ControlPointsParser.cs*, that is responsible for parsing the file and returning the corresponding list of points. However, the points in the list are not made available in the correct order. To address this issue, *DrawLines.cs* sends the imported list to *NearestNeighborList.cs*, which calculates the pairwise distance between all points and determines the correct order of points based on the minimum distance value between each pair. Finally, *DrawLines.cs* creates a new GameObject, *myLine*, and attaches a LineRenderer component to it. The centerline is drawn by the LineRenderer using the points from the centerline's ordered list (Figure 8).

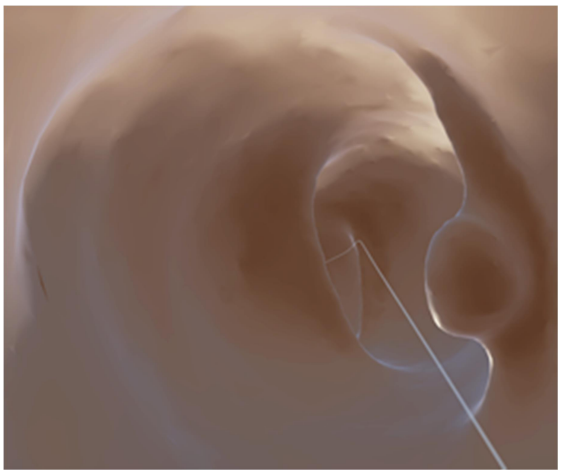

Figure 8. 3D Colon Model's interior displaying the centerline (light grey line).

As a remark, the reconstructed model of this tutorial was used by Lopes et al. [6] to evaluate the potential of a set of interaction techniques to perform CTC reading in an immersive VR environment.

Summary

We provide a short step-by-step description of the stages involved in this tutorial.

1) Import the 2D CTC image dataset into ITK-SNAP
   a) Enter the Active Contour Segmentation Mode
   b) Select Segment 3D
   c) Define a global threshold to identify the luminal surface
   d) Define the seed points
   e) Run the active contour evolution
   f) Save the segmentation image in *.mha* format

2) Import the segmentation image into Paraview
   a) Select Contour
   b) Apply smoothing and decimation filters
   c) Export the 3D mesh in *.ply* format

3) Import the 3D mesh into Blender and convert it into an *.obj* file

4) Obtain the 3D centerline of the colon's mesh using the MCS algorithm
   a) Apply the Voronoi based MAT
   b) Apply the MCF filter
   c) Export the centerline as a *.cg* file

5) Import into Unity
   a) Drag the 3D mesh into the scene
   b) Duplicate the mesh
   c) Rename the GameObjects as *Inside* and *Outside*
   d) Attach the *ReverseNormals.cs* script and add a Mesh Collider component to the *Inside* GameObject
   e) Create a material to color the 3D colon model
   f) Add the material both to the *Inside* and *Outside* mesh renderers
   g) Create a child GameObject of *Outside* called *SplineRoot*
   h) Add the *DrawLines.cs*, *ControlPointsParser.cs* and *NearestNeighborList.cs* scripts to the project's Scripts folder (Assets→Scripts)
   i) Attach the *DrawLines.cs* script to *SplineRoot*
   j) Fill in the path to the centerline's *.cg* file in the *DrawLines.cs* script

6) Run Unity

References

1. de Haan, M. C., van Gelder, R. E., Graser, A., Bipat, S., and Stoker, J. (2011). Diagnostic value of CT-colonography as compared to colonoscopy in an asymptomatic screening population: a meta-analysis. European radiology, 21(8), 1747-1763. https://doi.org/10.1007/s00330-011-2104-8

2. Pickhardt, P. J., Hassan, C., Halligan, S., and Marmo, R. (2011). Colorectal cancer: CT colonography and colonoscopy for detection—systematic review and meta-analysis. Radiology, 259(2), 393-405. https://doi.org/10.1148/radiol.11101887

3. Ribeiro, N. S., Fernandes, P. C., Lopes, D. S., Folgado, J. O., and Fernandes, P. R. (2009). 3-D solid and finite element modeling of biomechanical structures-a software pipeline. Proceedings of the ESMC2009 7th Euromech Solid Mechanics Conference, Lisbon, Portugal, September 2009, 2-17.

4. Smith, K., Clark, K., Bennett, W., Nolan, T., Kirby, J., Wolfsberger, M., Moulton, J., Vendt, B. and Freymann, J. (2015). Data From CT_COLONOGRAPHY. The Cancer Imaging Archive. https://doi.org/10.7937/K9/TCIA.2015.NWTESAY1

5. Tagliasacchi, A., Alhashim, I., Olson, M., and Zhang, H. (2012, August). Mean curvature skeletons. In Computer Graphics Forum, 31 (5), 1735-1744. https://doi.org/10.1111/j.1467-8659.2012.03178.x

6. D.S. Lopes, D. Medeiros, S.F. Paulo, P.B. Borges, V. Nunes, V. Mascarenhas, M. Veiga, J.A. Jorge, Interaction Techniques for Immersive CT Colonography: A Professional Assessment, In: Frangi AF, Schnabel JA, Davatzikos C, Alberola-Lopez C, Fichtinger G. (eds) Medical Image Computing and Computer Assisted Intervention - MICCAI 2018. MICCAI 2018. Lecture Notes in Computer Science, Springer, Cham, 2018 (accepted paper)