# Missing Surface Detection Pipeline

Yubo Zhang, Ruibin Ma, Sarah K. McGill, Julian Rosenman, Stephen Pizer

July 2020

## 1 Motivation

Colonoscopy is an effective examination for large intestine lesion (colon) screening and preventing colon cancer. Doctors carry out the examination by watching the live video sent from a camera that is controlled moving inside the patient's colon. Due to time limitation, colonoscopy is a one-pass exam, and due to human error, some part of the colon surface may be left out from the field of the camera and thus be unsurveyed, potentially causing cancerous or pre-cancerous lesions to be undetected. Therefore, it is important to realize the existence of unsurveyed colon surface regions and to warn the doctor during the colonoscopy process. In order for the warning to be useful, the detection must be communicated within a few seconds.

Our group's previous work [Ma et al., 2019] has focused on reconstructing colon surfaces from video input, which produces chunks of 3D reconstructed surface. Based on these reconstructed chunks in this work we develop a missing surface detection pipeline to find holes in the 3D surface. The reconstructed chunks can be seen as generalized cylinders. We project these surfaces onto the 2D plane, simplifying the geometry of the 3D objects and thus enabling the detection of holes. Mathematical morphology approaches are then applied to 2D images, denoising the reconstruction result and finding the missing patches on the surface. Together with the reconstruction process, our pipeline has great potential to be applied in real-time colonoscopy, alerting doctors to the unsurveyed surface regions and decreasing the chance of missing lesions.

## 2 Methods

To detect holes on reconstruction surfaces, our method can be seen as a two-step pipeline. The reconstructed chunks are point clouds, consisting of discrete points floating in 3D space according to their coordinates. Our first step is to flatten the 3D chunks and project the points onto the 2D plane, by assigning one coordinate along the cylinder and the other around the cylinder. To do this, we compute a centerline for the generalized cylindrical chunk. The second step is denoising and hole finding using mathematical morphology.

### 2.1 Centerline fitting

In order to flatten a 3D chunk surface, a centerline is needed which will serve as an axis of the flattened 2D plane. The reconstructed colon chunk can be seen as a cylinder with some noise and local deformations. The centerline of the chunk, like the centerline of a right circular cylinder, should be a line that travels inside the chunk, from one end to the other, without piercing through the surface. Once we have the centerline, cross sections orthogonal to the centerline can provide the around-cylinder coordinate.

To get the centerline from this relatively noisy cylinder, we will utilize the geometry of the reconstructed surface. Take the right circular cylinder as an example, if we compute the principal curvatures and principal directions of each point on the surface, the centerline will be paralleled to each point's principal direction of the smallest absolute curvature. Considering the colon chunk as a generalized cylinder, it is desirable to compute the principal direction of each data point in the point cloud and to generate a summary direction from these directions.

According to [Pauly, 2003], we use principal component analysis (PCA) to obtain the principal directions of the points on the surface. For each sample point $\boldsymbol{p}$, we compute the centroid $\boldsymbol{c}_p$ of its neighbors $\{\boldsymbol{q}_i\}$

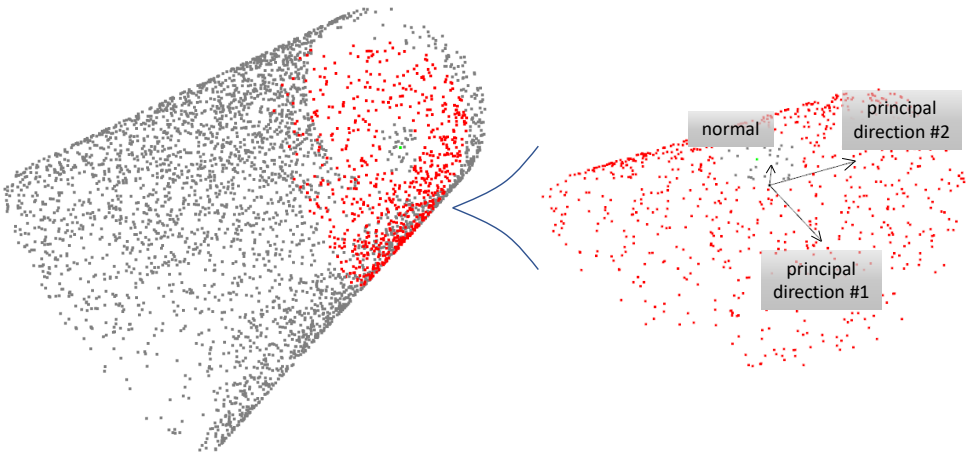


Figure 1: Principal directions of a sample point on a right circular cylinder

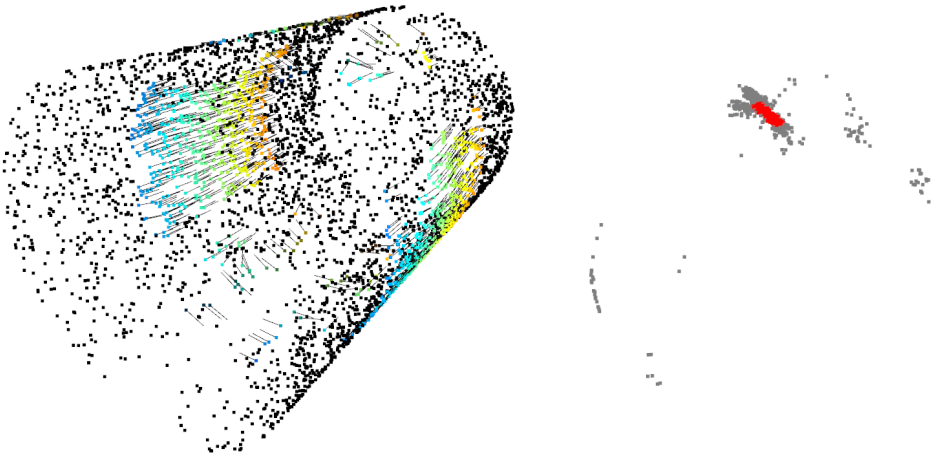

Figure 2: Candidates of cylinder centerline direction

within a certain range, and their covariance matrix $\boldsymbol{C}_p$, where $\gamma$ selects the neighbors:

$$\boldsymbol{c}_p = \frac{1}{N} \sum_{\boldsymbol{q}_i} \boldsymbol{q}_i \cdot \gamma(\boldsymbol{p}, \boldsymbol{q}_i) \tag{1}$$

$$\boldsymbol{C}_p = \sum_{\boldsymbol{q}_i} (\boldsymbol{q}_i - \boldsymbol{c}_p) \cdot (\boldsymbol{q}_i - \boldsymbol{c}_p)^{\mathrm{T}} \cdot \gamma(\boldsymbol{p}, \boldsymbol{q}_i) \tag{2}$$

The covariance matrix $\boldsymbol{C}_p$ reflects the spatial distribution of the sample point and its neighbors. Its eigenvector of the smallest eigenvalue can be used as the surface normal at $\boldsymbol{p}$, while the other two eigenvectors can be used as the principal directions. In our case, we use the eigenvector of the largest eigenvalue at $\boldsymbol{p}$, which corresponds to the direction of the smallest curvature, as a candidate direction of the cylinder centerline. For example, for the green vertex on the cylinder in Fig. 1, the eigenvectors of $\boldsymbol{C}_p$ are corresponding to the normal and principal directions of that point, and the principal direction #2, which is corresponding to the largest eigenvalue, points to the same direction as cylinder centerline.

To deal with the noise on the reconstructed surface, the range of neighborhood points is set to be related

to the estimated cylinder radius $r$:

$$\gamma(\boldsymbol{p}, \boldsymbol{q}_i) = \begin{cases} 1 & r/5 < d(\boldsymbol{p}, \boldsymbol{q}_i) < r \\ 0 & otherwise \end{cases} \tag{3}$$

where $d(\boldsymbol{p}, \boldsymbol{q}_i)$ is the Euclidean distance between center point $\boldsymbol{p}$ and its neighbor $\boldsymbol{q}_i$. We set criteria for whether the eigenvector associated with the largest eigenvalue at $\boldsymbol{p}$ is chosen as a centerline direction candidate $\boldsymbol{v}_p$. The criteria are that the two larger eigenvalues are larger than selected respective thresholds. Otherwise, the point set $\{\boldsymbol{q}_i\}$ may be too noisy or curve too much on principal directions to be fitted into a cylinder surface. The candidate directions of a right circular cylinder are shown in the left figure of Fig. 2.

We further remove the outliers in centerline direction candidates. After obtaining the direction candidates, the z-score of each candidate is computed as

$$\boldsymbol{z}_p = (\boldsymbol{v}_p - \bar{\boldsymbol{v}}_p)/\boldsymbol{S}_p \tag{4}$$

where $\bar{\boldsymbol{v}}_p$ is the average direction of all candidates and $\boldsymbol{S}_p$ is their standard deviation. The direction with an absolute z-score larger than 2 in any dimension of $\boldsymbol{z}_p$ is considered an outlier and will be removed. This candidate clearing process (i.e., computing z-score and outlier removing) is been executed for 5 iterations. Then, for the remaining candidates, their average is been computed as the direction $\boldsymbol{v}_c$ of the chunk centerline.

After the direction has been set, the centerline is anchored to a center point of the chunk. Because ourliers of the chunk surface have been removed, the center point is defined from the extremal values of $x$, $y$, $z$ in the chunk as

$$\boldsymbol{c} = \{(x_{max} + x_{min})/2, (y_{max} + y_{min})/2, (z_{max} + z_{min})/2\} \tag{5}$$

## 2.2 Surface flattening

After the centerline is found, every point on the 3D chunk surface can be projected onto the 2D plane according to its relative position to the centerline. The $\{x, y, z\}$ coordinate of each vertex is transformed into 2D coordinate $\{d, r\theta\}$. Considering each vertex is on a cylinder cross section orthogonal to the centerline, $d$ is defined as the position of the cross section-centerline intersection relative to the center point $\boldsymbol{c}$, and $r\theta$ is the position of the vertex on the cross section plane, where $r$ is the distance from the vertex to the intersection and we define $\theta$ as its angel around the intersection.

$d$ can be computed by projecting the vertex $\boldsymbol{p}$ onto the centerline, where $\boldsymbol{v}_c$ is the normalized vector of centerline direction:

$$d = (\boldsymbol{p} - \boldsymbol{c}) \bullet \boldsymbol{v}_c \tag{6}$$

To compute $\theta$, we pick a direction $\boldsymbol{v}_{cs}$ orthogonal to $\boldsymbol{v}_c$ as the reference vector of every cross section, and $\theta$ is defined as the angle between $\boldsymbol{v}_{cs}$ and the vector from cross section-centerline intersection $\boldsymbol{p}_{cs}$ to $\boldsymbol{p}$:

$$\boldsymbol{p}_{cs} = \boldsymbol{c} + d \cdot \boldsymbol{v}_c \tag{7}$$

$$\theta = (\boldsymbol{p} - \boldsymbol{p}_{cs}) \bullet \boldsymbol{v}_{cs}/|\boldsymbol{p} - \boldsymbol{p}_{cs}| \tag{8}$$

To stabilize the problem, we set $r$ in the second term of 2D coordinates as the average of $|\boldsymbol{p} - \boldsymbol{p}_{cs}|$.

An example is shown in Fig. 3. For the chunk on the left, when flattened around the centerline shown as the black line traveling through the chunk, the 3D surface is been transformed to the figure on the right. The holes in the middle and the edges of the end of the chunk are well preserved.

## 2.3 Denoising and hole finding

After projecting 3D point cloud onto the 2D plane, we discretize the points' coordinates and convert the surfaces into 2D images. Then mathematical morphology methods can be used to clean the noisy point surface. We first apply closing operation with a diamond kernel to remove the small holes caused by artifact, then opening operation with the diamond kernel of the same size to open up the remaining holes. To identify the holes on the surface, the connected components of the image background are computed, which correspond to the missing surface on in middle and the possible holes at the end of the chunk.

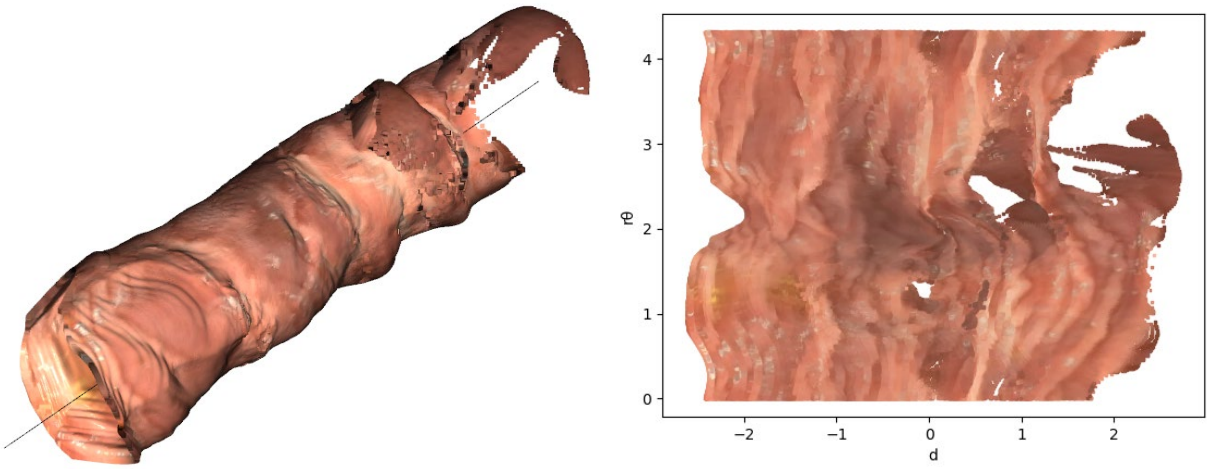


Figure 3: An example of a 3D chunk been projected onto the 2D plane.

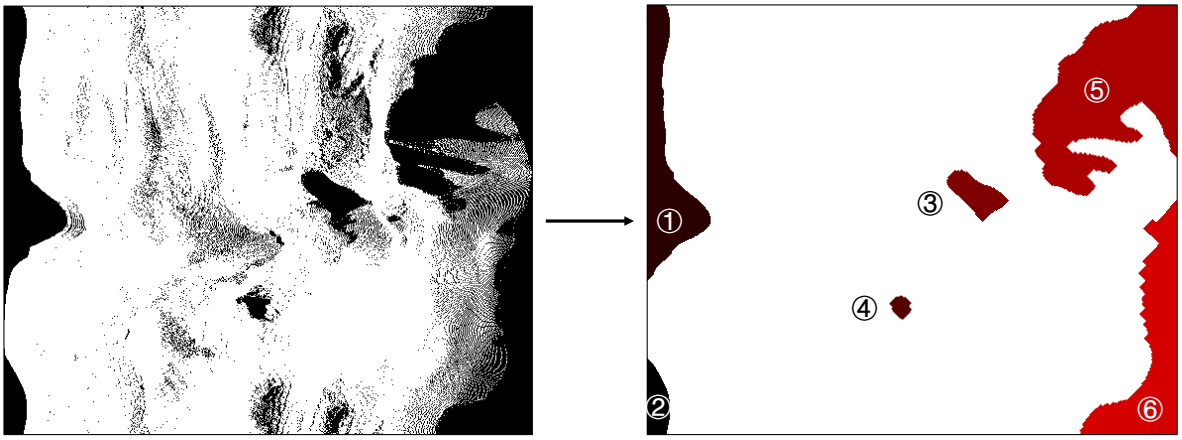


Figure 4: After applying mathematical morphology methods, there are 6 connected components in the background, which are the possible missing surface pieces.

For the chunk in Fig. 3, after applying closing and opening for denoising, there are 6 connected components in the background, which are marked in different colors in Fig. 4. Among them, #3 and #4 are the holes on the reconstructed chunk, and #5 is a possible hole appearing at the end of the chunk.

# 3 Results

**Results on more chunk samples** Beside the chunk sample in Fig. 3 and 4, we have tested our pipeline on other six chunk samples, shown in Fig. 5. The centerlines are visualized as black lines going through the chunk. The 2D images show the hole finding results. White areas are the flattened chunk surfaces, and different holes are marked in black and red in different darkness. Although some of the surfaces are nosier comparing to the example in Fig. 3, their centerlines are not distracted by the noise and locate in reasonable positions, efficient for surface flattening. For all six samples, the kernels we use for mathematical morphology operations are in the same size. Most of the small holes caused by noise are effectively removed, and the large areas of missing surface are clearly revealed. Note that the kernel size can be manually picked, depends on how large the hole is defined as significant.

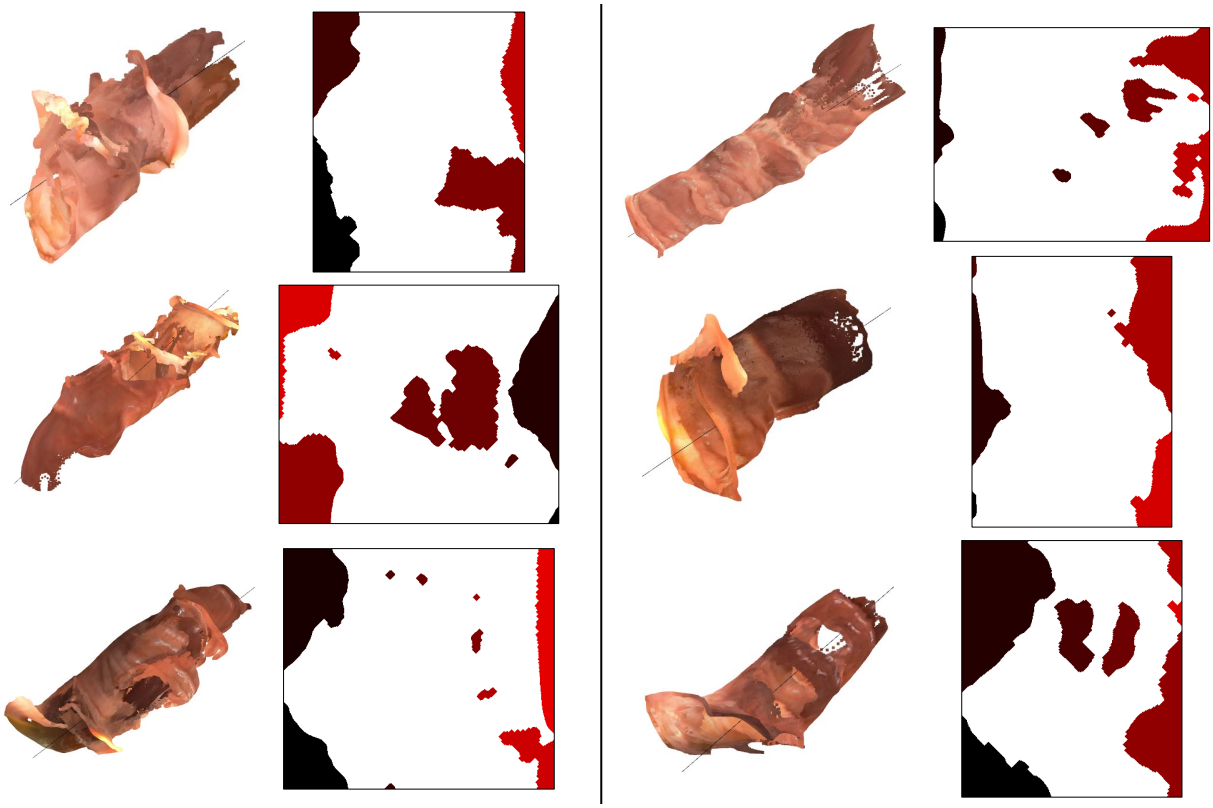

Figure 5: Results on six chunk sample. The white areas are the flattened surface. The holes found by our pipeline are marked in different colors.

**Time efficiency** To be applied in colonoscopy, the program must run in real-time. For our pipeline, the average running time of the above seven examples is about 3.4 seconds on a CPU. The most time-consuming step is to fit the centerline, where eigenvectors are computed for every point, which takes about 2.6 seconds. To speed up the process, multi-thread programming can be used; we leave it to the future work.

# 4 Conclusion and future work

To automatically detect missing surface on the 3D reconstructed colon chunk, we develop a two-step pipeline. Considering the chunk as a generalized cylinder, after fitting a centerline using cylinder geometry, we project the 3D point cloud onto the 2D plane. Then the mathematical morphology operations are applied to denoise the surface and identify the holes. Our method produces reliable results on our testing cases, clearly revealing the missing surface.

There are several directions that are worth exploring for future work: (1) As we consider the reconstructed chunk as a cylinder, there are cases that the chunks have large curves and a simple centerline cannot be fitted. In this situation, preprocessing should be conducted to break those chunks that each of them can be seen as a cylinder. (2) To be applicable in clinical usage, the program must run in real-time. Multi-thread programming or faster deep-learning approaches can be developed to speed up the process. (3) Although missing surface is a crucial problem in colonoscopy, the statistics of what is the exact proportion of the surface that is unsurveyed have not been made. Our pipeline can be further applied to provide these statistics.

# References

[Ma et al., 2019] Ma, R., Wang, R., Pizer, S., Rosenman, J., McGill, S. K., and Frahm, J.-M. (2019). Real-time 3d reconstruction of colonoscopic surfaces for determining missing regions. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 573–582. Springer.

[Pauly, 2003] Pauly, M. (2003). *Point primitives for interactive modeling and processing of 3D geometry*. Hartung-Gorre.