

Supplementary Materials

A deep learning-based system for accurate detection of anatomical landmarks in colon environment

Chengwei Ye¹, Kaiwei Che^{2,3}, Yibing Yao⁴, Nachuan Ma⁵, Ruo Zhang⁶, Yangxin Xu¹, Jiankun Wang⁶, Max Q.-H. Meng^{1,6,7}

¹Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong 999077, China.

²School of Electronic and Computer Engineering, Peking University, Beijing 100871, China.

³Peng Cheng Laboratory, Shenzhen 518000, Guangdong, China.

⁴Department of Oncology, Air Force Medical Center, PLA, Beijing 100142, China.

⁵College of Electronic & Information Engineering, Tongji University, Shanghai 201804, China.

⁶Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, Guangdong, China.

⁷Shenzhen Research Institute of the Chinese University of Hong Kong, Shenzhen 518057, Guangdong, China.

Correspondence to: Prof. Jiankun Wang, Prof. Max Q.-H. Meng, Department of Electronic and Electrical Engineering, Southern University of Science and Technology, 1088 Xueyuan Avenue, Shenzhen 518055, Guangdong, China. E-mail: wangjk@sustech.edu.cn; max.meng@ieee.org

The post-processing algorithm is shown in Algorithms 1, 2, and 3. Algorithm 2 is designed to determine whether the first and last four frames are positive or negative based on the ten neighboring frames. Algorithm 3 is utilized to count the number of changes from positive to negative or from negative to positive. Algorithm 1 combines Algorithms 2 and 3 to update the final detection results until the number of changes is equal to 1. In the algorithms, input y_{pred} denotes the intermediate detection result; output y_{final} denotes the final detection result; variable *input_list* denotes a temporary variable containing a segment of y_{pred} .

Algorithm 1: Result_cleaning

Input: Intermediate detection result: y_{pred} **Output:** Final detection result: y_{final}

```
1 Function Result_cleaning( $y_{pred}$ ):
2   while True do
3      $y_{final} \leftarrow$  empty list;
4     Check_ends( $y_{pred}[:10]$ );
5     for  $i = 4 \rightarrow \text{len}(y_{pred}) - 4$  do
6        $check\_list \leftarrow y_{pred}[i - 4 : i + 4 + 1]$ ;
7       if Sum( $check\_list[0 : 4, 5 : 8] > 3$ ) then
8          $y_{final}.append(1)$ ;
9       else
10         $y_{final}.append(0)$ ;
11     Check_ends( $y_{pred}[-10 :]$ );
12     if Check_step( $y_{pred}$ ) = 1 then
13       break;
14     else
15        $y_{pred} \leftarrow y_{final}$ ;
16   return  $y_{final}$ ;
```

Algorithm 2: Check_ends

Input: $input_list$

```
1 Function Check_ends( $input\_list$ ):
2   if Sum( $input\_list$ ) > 5 then
3      $y_{final}.extend([1, 1, 1, 1])$ ;
4   else
5      $y_{final}.extend([0, 0, 0, 0])$ ;
```

Algorithm 3: Check_step

Input: $input_list$ **Output:** Number of step: $step$

```
1 Function Check_step( $input\_list$ ):
2    $step \leftarrow 0$ ;
3   for  $i = 0 \rightarrow \text{len}(input\_list) - 1$  do
4      $step += |input\_list[i] - input\_list[i + 1]|$ ;
5   return  $step$ ;
```
