Supplementary Materials

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

Chengwei Ye[1], Kaiwei Che[2,3], Yibing Yao[4], Nachuan Ma[5], Ruo Zhang[6], Yangxin Xu[1], Jiankun Wang[6], Max Q.-H. Meng[1,6,7]

[1]Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong 999077, China.
[2]School of Electronic and Computer Engineering, Peking University, Beijing 100871, China.
[3]Peng Cheng Laboratory, Shenzhen 518000, Guangdong, China.
[4]Department of Oncology, Air Force Medical Center, PLA, Beijing 100142, China.
[5]College of Electronic & Information Engineering, Tongji University, Shanghai 201804, China.
[6]Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, Guang dong, China.
[7]Shenzhen 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}

Function `Result_cleaning`(y_{pred})**:**
  while ***True*** **do**
    $y_{final} \longleftarrow empty\ list$;
    `Check_ends`($y_{pred}[:10]$);
    for $i = 4 \rightarrow$ `len`(y_{pred})-4 **do**
      $check\_list \longleftarrow y_{pred}[i-4 : i+4+1]$;
      if `Sum`($check\_list[0:4, 5:8] > 3$) **then**
        $y_{final}.append(1)$;
      else
        $y_{final}.append(0)$;
    `Check_ends`($y_{pred}[-10:]$);
    if `Check_step`(y_{pred}) $= 1$ **then**
      break;
    else
      $y_{pred} \longleftarrow y_{final}$;
  return y_{final};

Algorithm 2: Check_ends

Input: $input\_list$

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

Algorithm 3: Check_step

Input: $input\_list$
Output: Number of step: $step$

Function `Check_step`($input\_list$)**:**
  $step \longleftarrow 0$;
  for $i = 0 \rightarrow$ `len(input_list)`-1 **do**
    $step += |input\_list[i] - input\_list[i+1]|$;
  return $step$;