Polyp-Alert: Near Real-time Feedback during Colonoscopy

Yi Wang^a, Wallapak Tavanapong^{*a}, Johnny Wong^a
JungHwan Oh^b, and Piet C. de Groen^c

^aDepartment of Computer Science, Iowa State University, Ames, IA 50011, USA

^bDepartment of Computer Science and Engineering, University of North Texas, Denton, TX 76203, USA

^cMayo Clinic College of Medicine, Mayo Clinic, Rochester, MN 55905, USA

Abstract

We present a software system called “Polyp-Alert” to assist the endoscopist find polyps by providing visual feedback during colonoscopy. Polyp-Alert employs our previous edge-cross-section visual features and a rule-based classifier to detect a polyp edge---an edge along the contour of a polyp. The technique employs tracking of detected polyp edge(s) to group a sequence of images covering the same polyp(s) as one polyp shot. In our experiments, the software correctly detected 97.7% (42 of 43) of polyp shots on 53 randomly selected video files of entire colonoscopy procedures. However, Polyp-Alert incorrectly marked 4.3% of full-length colonoscopy procedure as showing a polyp when they do not. The test data set consists of about 18 hour worth of video data from Olympus and Fujinon endoscopes. The technique is extensible to other brands of colonoscopes. Furthermore, Polyp-Alert provides as high as ten feedbacks per second for a smooth display of feedback. The performance of our system is by far the most promising to potentially assist the endoscopist find more polyps in clinical practice during a routine screening colonoscopy.

Key Words

Near Real-time, Polyp Detection, Colonoscopy, Medical Imaging/Video

1. Introduction

Colorectal cancer is the second leading cause of cancer-related deaths, claiming close to 50,000 lives annually in the United States [1]. Colonoscopy is an important screening tool that has contributed to a significant decline in colorectal cancer-related deaths. During colonoscopy, a tiny video camera at the tip of the endoscope generates a video signal of the internal mucosa of the human colon. The video data is displayed on a monitor for real-time diagnosis by the endoscopist. Abnormalities like polyps can be found and removed during the colonoscopy. Despite the success of colonoscopy in lowering cancer-related deaths, a significant average miss rate for detection of both large polyps and cancers exists and is estimated to be around 4-12% based on several reports [2 - 4]. In recent years, many computer-aided polyp detection techniques have been developed with the ultimate goal to assist the endoscopist in lowering the polyp miss rate. However, automatic polyp detection in recorded video data during

colonoscopy (or colonoscopy video) is very challenging due to the noisy nature of endoscopic images caused by camera motion, strong light reflections, the wide-angle lens that cannot be automatically focused, and the location and appearance variations of polyps within the colon.

Colon polyps are generally classified based on their appearance as pedunculated, sessile, or flat. Sessile polyps grow on the surface of the colon without a stalk. Pedunculated polyps have short or long stalks. Flat polyps grow along the surface of the colon. In general, sessile polyps, the head of pedunculated polyps, and flat polyps have ellipse shape when small. Some polyps may grow into cancer. Complete removal of polyps during colonoscopy can prevent them from possibly growing into colon cancer. Polyps vary in their appearance, shape, size, amount of protrusion, and location in the colon. The same polyp may appear very differently in different images due to amount of colon insufflation, degree of colon muscular contraction, viewing angle, and distance from the capturing camera (Figure 1, rows 1 and 2). Stool or therapeutic instruments may occlude polyps (Figure 1, row 3).

Figure 1: The first and the second rows: blue rectangles mark polyp regions of polyps seen from different view angles. The third row: polyps occluded by an instrument during polypectomy.

Due to the complexity of appearance of polyps and characteristics of colonoscopy video, existing techniques to detect an image containing a polyp or to mark a polyp region in an image require long processing time. The state-of-the-art technique took about 3 to 4 seconds per image in a recent survey [5 - 20]. Furthermore, a detection rate (recall) of close to 90% generally comes with a high false alarm rate of at least 1 false region per image [19]. High false alarm rates and long analysis times are inadequate for real-time feedback of polyp regions during colonoscopy as part of an automated method to reduce polyp miss rates.

Our contributions in this paper are threefold. First, we present a comprehensive survey of existing literature on polyp detection techniques in colonoscopy. Several polyp detection techniques in colonoscopy were developed in recent years and were nicely summarized in [21]. We added to the previous survey result [21] with information about techniques developed after the survey report along with additional qualitative performance criteria. Most of these techniques have not been quantitatively compared against each other due to lack of common ground truth, common performance metrics, and detailed implementation of the algorithms for reimplementing to conduct a fair comparison. Second, we designed algorithms and developed software modules for fast polyp edge and shot detection. Third, we developed the Polyp-Alert software system that analyzes colonoscopy video for polyps and shows visual feedback of detected polyp edges on screen. We tested Polyp-Alert on a 64-bit Intel (R) Xeon 2.0 GHz duo-core CPU with 6.0 GB RAM running 64-bit Windows 7 Professional and similar systems. Polyp-Alert is able to continuously analyze image streams and identify a detected polyp edge at the rate of 10 frames per second (fps) for a typical NTSC frame rate of 29.97 fps: i.e., the technique analyzes one of every three frames. Existing polyp detection algorithms are too complex to achieve this analysis rate. The fast analysis time means less latency for feedback. Ideally, we want to achieve real-time analysis within one frame time interval to have instant visual feedback. Our technique is by far the most promising given the combination of speed and accuracy. It may help in recognition of polyps for both novice as well as experienced endoscopists, especially at the edges of the image. It also may help when the endoscopist is distracted or fatigued. In addition, real-time or near real-time polyp shot detection will also enable automated real-time documentation for the endoscopist who will simply verify the correctness at the end of the procedure.

The remainder of this paper is organized as follows. We discuss the related work of both polyp image detection and shot segmentation/extraction algorithms in Section 2. We present our polyp image detection technique in Section 3. We present polyp tracking/warning in Section 4. We experimentally evaluate our system in Section 5. We conclude the paper and describe the future work in Section 6.

2. Background and Related work

Our aim is to detect polyps in colonoscopy images from standard endoscopes in routine colonoscopy practice. There are polyp detection techniques proposed for images from magnifying endoscopes or Narrow-Band Imaging (NBI) [22, 23]. NBI or magnifying endoscopes are not typically used for the entire procedure. Note that pit patterns [23] of polyps are not necessarily seen using standard scopes; indeed, detection of pit patterns requires close up inspection which happens after the endoscopist has identified a lesion as a potential polyp. A number of research works have investigated polyp detection for Colon Capsule Endoscopy (CCE) images as summarized in [24 - 30]. In CCE, A small capsule consists of an

imaging device and wireless transmitter which is ingested by the patient. The images are transmitted to the wireless receiver carried by the patient. Images are read by the domain expert after the end of the procedure to look for abnormalities. Treatment cannot be done during CCE. Therefore, anytime a polyp is found the patient needs to undergo a colonoscopy. In addition, the capsule moves according to the organ movement. It cannot be controlled to look at the entire colon wall.

We define some terminologies to be used hereafter. A *polyp image* is an image showing a polyp. A *polyp shot* is a group of consecutive images in the video showing the same polyp(s). Multiple polyps in the same frame are considered in the same shot. Some frames in a polyp shot may not have any polyps in them due to temporary fast camera movement that causes blurry frames or the polyp is briefly out of the field of view of the camera before it reappears in the field of view. We define a *polyp edge* as a binary edge on the contour of a polyp.

2.1. Polyp Image Detection

In recent years, many algorithms for computer-aided automatic polyp detection have been proposed for images from standard endoscopes [5, 6, 8, 11, 21, 31]. Many find a polyp region and polyp images while a few detect polyp shots. The commonality among these techniques is pre-processing that discards noise, feature extraction that extracts discriminative features (e.g., texture, shape, color), and machine-learning classification on the features to identify polyp regions, images, or shots. These techniques take long processing time as a tradeoff to effectiveness, making them unsuitable for an application that assists the endoscopist during an ongoing procedure. Table 1 shows the qualitative performance comparison.

Table 1: Overview of Polyp Detection Techniques in Colonoscopy

Authors, Pub. year	Method/Feature	Feature Type	Classifier	Poly p Shot	Datasets (#images, image resolution)	AT in sec.	Effectiveness
Krishnan et al., 1998 [7]	Curvature analysis	Shape	-	No	(2 normal and 4 abnormal images, -)	NA	NA
Karkanis et al., 2001 [15]	Co-occurrence matrix, Wavelet transform	Texture	NN	No	(8 images, 512 x 512 pixels)	NA	MCE: 4% - 7%, depending on the window size
Wang et al., 2001 [10]	Local Binary Pattern	Texture	NN	No	(3 images, -)	NA	No quantitative measurement
Magoulas et al., 2001 [17]	Energy-Angular Second Moment, Correlation, Inverse Difference Moment, Entropy	Texture, color	NN	No	(4 images, 16 x 16 pixels)	NA	The best performance with unclearly defined metrics as 93%
Kang et al., 2003 [9]	Area, color and shape of segments	Shape, color	-	No	-	1	NA
Karkanis et al., 2003 [11]	Color Wavelet Covariance	Texture, color	LDA, SVM	No	(1,380 images, 1000 x 1000 pixels)	NA	Specificity: 99.3 (± 0.3); Sensitivity 93.6 (± 0.8)%
Tjoa et al., 2003 [14]	Texture spectrum histogram, color histogram	Texture, color	SVM	No	(12 normal and 54 abnormal images, -)	NA	Accuracy=97.72% (texture+color); 96.96% (texture); 90.52% (color)
Magoulas et al., 2004 [16] [18]	Co-occurrence matrix	Texture	NN	No	(4 images, -)	NA	Unclearly defined metrics: 68.88% ~ 94.89%, reported on each image
Li et al., 2005 [13]	Wavelet coefficients, histogram of CIE-Lab color space	Texture, color	Ensemble SVM, LDA	No	(12 normal and 46 abnormal images, 256 x 256 pixels)	NA	AVR = 74.4% (Ensemble SVM); 76.4% (LDC)
Dhanda et al., 2006 [8]	Watershed region segmentation in HIS color plane	Shape, color	-	No	(50 normal and 50 abnormal images, -)	NA	Specificity: 92%; Sensitivity 96%
Iakovidis et al., 2006 [6]	Local Binary Pattern (LBP)	Texture	SVM	No	(15,000 images, 320 x 240 pixels)	NA	OCLBP generally outperforms other tested features with its AUC close to 94%
	Wavelet Energy (WE)	Texture					
	Opponent Color Local Binary Pattern (OCLBP)	Texture, color					
	Color Wavelet Energy (CWE)	Texture, color					
	Wavelet Correlation Signatures (WCS)	Texture, color					
	Color Wavelet Covariance (CWC)	Texture, color					
Hwang et al., 2007 [5]	Curve direction, curvature, Watershed region segmentation, RGB pixel color	Shape, color	-	Yes	(815 polyp frames and 7,806 normal frames, 390 x 370 pixels)	NA	Correctly detected 26 of 27 polyp shots with 5 shots as false alarm
Alexandre et al., 2007 [12]	RGB pixel color	Color	SVM	No	(35 images, 514 x 469 pixels)	NA	AUC = 93.16 (± 0.09)% Sensitivity = 93.69 (± 1.04)% FPR = 23.11 (± 0.93)%

Cheng et al., 2008 [19]	Co-occurrence matrix, color texture	Texture, color	SVM	No	(37 training and 37 test images, 378 x 254 pixels)	13	Sensitivity = 86.2% False positive = 1.26 regions per image
Ameling et al., 2009 [20]	Co-occurrence matrix	Texture	SVM	No	(1,736 images in 4 scenes, 1920 x 1080 pixels)	NA	AUC = 95% for the best testing scene
	Local Binary Pattern	Texture					AUC = 95% for the best testing scene
	Opponent Color – Local Binary Pattern	Texture, color					AUC = 96% for the best testing scene
Park et al., 2012 [32]	Features from Eigen images of sub-images across frames, no. and size of edges, distance of edges from specular spots	Shape, Texture, Temporal information	CRM	No	(35 videos with each of them 20 to 40 min long; 2813 and 7345 polyp frames used, -)	NA	AUC = 89% (white light video) AUC= 88% (narrow- band video)
Bernal et al., 2012 [33]	Watershed region segmentation, RGB pixel color	Shape, color	-	No	(300 images, 500 x 574 pixels)	NA	Precision = 89.59%, Recall = 89%, Accuracy = 97.53%, Specificity = 98.64.
Wang et al., 2013 [34]	Edge cross-section based features	Shape, Color, Texture	DT, SVM	No	(247 training and 1266 test images, 720 x 480 pixels)	7.1	FROC with an example performance of Sensitivity = 81.4%, False positive = 0.32 regions per image
Tajbakhsh et al., 2014 [35]	Edge normal/cross- section based features	Shape, Color, Texture	RF	No	(150 training and 150 testing images, -)	NA	FROC with an example performance of Recall = 80%, Precision = 93%.

AT: Lower bound on the average analysis time in sec. per image reported in the original publication; MCE: Mean Classification Error; AUC: Area under Receiver Operating Curve (ROC); AVR: Average classification rate; CRM: Conditional Random Field; DT = Decision Tree; FPR: False positive rate; FROC: Free Response Operating Curve; LDA = Linear Discriminant Analysis; MSE: Mean Square Errors; NA: Not reported; NN = Neural Network; RF = Random Forest; SVM = Support Vector Machine.

The average analysis times in the AT column are the times reported in the original publications. Since the experimental settings (e.g., image resolution, operating systems, implementation programming language, machine capability) are different, the reported times of these techniques cannot be directly compared with each other. However, the information is useful as a rough guide to indicate the room for improvement in terms of analysis speed for each technique in order to provide feedback during a live procedure. Ideally, the best case is to finish the analysis of the current image frame before the next frame arrives. That is about 1/30 sec. for NTSC and 1/25 sec. for PAL signals. However, Table 1 shows that the current solutions are not able to achieve this stringent time requirement. For some techniques, the times were not reported in the original papers. Of all the existing techniques in Table 1, only the work of Hwang et al., 2007 [5] mentions polyp shot detection. The analysis time was not reported. The technique is inherently slow due to the polyp region detection step that uses a watershed image segmentation

technique.

Our previous work published in 2013 [34] compared our part-based multi-derivative edge cross-section profile (ECSP) features with Local Binary Pattern (LBP) and Opponent Color LBP (OCLBP) [6]. OCLBP was shown to perform best on over 15,000 images [6]. LBP is a popular texture feature for polyp detection [6] [20]. An LBP pattern of a pixel is represented by a binary sequence of length L . The l^{th} position in the binary sequence is assigned a 0 if the pixel value is smaller than the value of its l^{th} neighboring pixel, or assigned a 1 otherwise. An LBP pattern is mapped to a unique LBP value. A histogram of LBP values is used as an LBP feature of an image.

An OCLBP feature [6] [20] is a concatenation of three intra-channel LBP histograms and six inter-channel LBP histograms. Each of the three intra-channel LBP histograms is obtained from pixels in each of the RGB color channels, respectively. Let C_i denote channel i where $i=1, 2$, and 3 . The six inter-channel LBP histograms (C_1-C_2 , C_1-C_3 , C_2-C_1 , C_2-C_3 , C_3-C_1 , and C_3-C_2) are obtained from every combination of two different color channels. For example, the LBP pattern of C_1-C_2 is obtained from comparing the pixel value in the red color channel (C_1) with values of its neighboring pixels in the green color channel (C_2).

Our ECSP features outperform LBP and OCLBP for polyp image detection in both analysis speed and accuracy when tested on 42 distinct polyps. This is the largest number of distinct polyps we have seen in the literature thus far. The free-response receiver operating characteristic curves (FROC) show the comparison of our technique with LBP and OCLBP. Under the same true positive rate, the average false alarm rate of our technique is under 0.5, which is significantly lower than those of LBP and OCLBP. Furthermore, false regions generated by our technique occur within only 20.5% of non-polyp images. However, the false regions generated using LBP and OCLBP are almost equally distributed on different types of non-polyp images. As a result, nearly every non-polyp image (99.8%) has at least one false region. Compared with these sliding-window based methods, our method detects polyp edges, which reflects more accurate locations of the polyps. Thus, the results of our technique are more meaningful in clinical practice. Our technique took 7.1 seconds to process one image implemented using MATLAB. From Table 1, our technique [34] is by far the most promising one to provide feedback in real time. In 2014, Tajbakhsh et al. [35] presented an edge like descriptor based on the similar principle as our work in [34]. The technique detected polyp edges by sliding a square patch on the image, extracting features based on the pixel information surrounding the edge normal directions and classifying features using Random Forest classifier. The technique in [35] implemented a sampler edge descriptor based on our edge normal and cross-section property in [34]. However, the technique does not directly compare with our technique. The technique was tested on only 150 test images and the analysis time was not reported. In this paper,

we focus on improving our technique in [34] in terms of analysis speed while maintaining high recall and low false alarms in order to provide near real-time visual feedback.

2.2. Shot Segmentation for Generic Video

A video shot is a sequence of consecutive frames captured under a single recording operation. Frames within the same shot usually have similar visual properties. Shot segmentation divides a long video file into a number of shots and outputs the start and end frames of these shots. The criterion for grouping frames into a shot is typically based on low-level image features (e.g. color or intensity histograms, significant changes in locations of edges), and temporal information [36 - 38] Sharp changes between shots are easier to detect than gradual changes such as wipes, fade-in/out, and dissolves. Shot segmentation techniques operating on compressed video directly have been proposed [39] to reduce processing time.

As we intend to use the software to process uncompressed video from a capturing device during colonoscopy, we focus on the related work of shot segmentation algorithms on uncompressed video and divide them into three major categories: 1) thresholding-based algorithms, 2) clustering-based algorithms, and 3) model-based algorithms.

- 1) **Thresholding-based algorithms:** A similarity score is calculated between two consecutive frames to measure the similarity of the frames. The similarity score is generally calculated from low-level image features such as pixel intensity values, average intensity values of block-based regions, and color histograms. Pixel comparison methods compute the difference of intensity values between two pixels at the same location in two different images [40]. The average difference of intensity values over all pixels is used to indicate the similarity of the two frames. A cut is detected if the difference is larger than a threshold. Many alternative approaches can be used to compute the similarity score. Block-based comparison methods [41, 42] divide an image into overlapping or non-overlapping blocks and calculate similarity scores among corresponding blocks between two images. The average score of these similarity scores between two images is used to compare with a threshold to find a shot cut. Comparing to the pixel comparison, the block-based comparison considers the overall content between images and it is more robust to small camera or object motion. However, the block-based comparison methods are still sensitive to large motion of camera or objects. Histogram comparison computes the distribution of intensity values represented by histogram bins. The differences of histogram bin values between two frames are used as a similarity score. Histogram comparison is less sensitive to motion of camera or objects. Threshold selection is an important factor that affects effectiveness of these techniques. Thresholds are typically determined empirically based on training

data sets. Hanjalic et al. [37] proposed a statistical method that determines threshold values automatically for different types of videos.

- 2) **Clustering-based algorithms:** In [43] an unsupervised learning algorithm is used to cluster frames into shots. Many features can be used for clustering such as intensity or color histograms. Compared to threshold-based algorithms, clustering-based algorithms do not have shortcomings in selecting an appropriate threshold. However, they do not perform well for detecting gradual transitions [39].
- 3) **Model-based algorithms:** These algorithms either define or train underlying statistical models for classifying consecutive video frames into shot boundary and non-shot boundary frames. Example statistical models are the probabilistic distribution of pixel intensity values [44], and the hidden Markov model for modeling states of frames in shots [45]. Thresholds are not required in the model-based algorithms. However, these models need to be pre-trained. A suitable model may be difficult to find for some types of videos.

Traditional shot segmentation techniques are not suitable for colonoscopy video because of the following reasons. 1) Color information is not as useful since colonoscopy images have limited color ranges of mostly red; with stool, the red color is mixed with green or yellow. 2) There is no significant movement of a specific object of interest in the image, but small movement of the camera can cause rotation, translation, and scaling of the whole image. 3) Strong light reflection, low illumination and blurriness cause significant changes in low-level features, which makes it difficult to group them into one shot. Robust shot segmentation methods that account for the monotone image pixel color property and the effect of camera motion such as strong light reflection and partially blurred images are desirable for segmenting colonoscopy video.

We previously introduced shot segmentation for colonoscopy video in [46, 47]. In [46], we segment a colonoscopy video into shots using the global histogram comparison in the threshold-based approach. In [47], we extract *operation shots*, each showing the cable part of a therapeutic or biopsy instrument in the field of view of the camera. In this paper, as we focus on detecting the appearance of a specific object, a polyp, in a video, we utilize an edge tracking algorithm to track the detected polyp edges for polyp shot extraction.

3. Polyp Image Detection

Our polyp image detection centers on detecting whether an edge is of a polyp. If the image has at least one detected polyp edge, it is considered as having a polyp in it. The polyp edge detection algorithm mainly relies on edge features obtained from our part-based multi-derivative edge cross-section profile

(ECSP) [34]. In this section, we briefly review our part-based multi-derivative ECSP and propose ECSP-based features for near real-time polyp image detection.

3.1. *Edge Cross-section Profile Calculation*

The edge cross-section profile is calculated as a function obtained from the surrounding regions of a detected *Canny* edge in the image [34] [51]. Figure 2 illustrates an example ECSP of a polyp and associated features. We do not detect the stalk of polyps as not all polyps have stalks. For most polyps, we found that the average brightness intensity gradually drops near the edge of the polyp and increases quickly and significantly in the protruding part and levels out. See Figure 2(d). Other objects do not exhibit this property consistently. Besides polyp detection, we have also applied ECSP for appendiceal orifice detection [50, 51] and retroflexion detection [48, 49] in colonoscopy video. The ECSP function of an edge represents the average intensity values of its surrounding regions along the directions perpendicular to the edge pixels. To detect a target object in an image, we first obtain its *Canny* edge, calculate the edge cross-section profile, and finally extract features from its multi-derivative functions: the ECSP function, its 1st order derivative function, and its 2nd order derivative function for classification. To obtain a representative set of features for classification, each multi-derivative function is segmented into parts on some key positions, for example, zero-crossing positions on the function. Features are calculated on the segmented parts by modeling these parts differently using the methods suitable for individual parts. Because this approach allows different parts to be modeled by different methods or have different features, ECSP-based features can effectively describe the complex properties of polyps. The example features for polyp image detection in [34] include the shape of the segmented parts on edge profiles, texture of the polyp region, estimated polyp protrusion, smoothness of polyp surface, etc. The performance using ECSP-based features on over 800 images of 42 distinct polyps outperforms LBP and OCLBP, the best features for this problem in terms of both accuracy and analysis time. Interested readers are referred to the detailed descriptions in [34] and [51].

Figure 2: (a) original gray-scale image of a polyp; (b) edge cross-section profile around the detected edge reshaped into a rectangle with the width twice the ratio of the edge curvature κ to a positive constant λ . The height is the length of the detected edge; (c) corresponding reshaped gray-scale image; (d) intensity profile at the horizontal line in (c); (e) the 1st order derivative of (d); the points C_1 and C_2 in (d) correspond to zero crossing points in (e).

3.2. Feature Extraction

Table 2 shows all the features we use for polyp edge detection. Most of these features are obtained from our part-based multi-derivative ECSP.

Table 2: Features and corresponding Value Ranges used in our Near Real-Time Polyp Image Detection Algorithm

Computation Order	Feature Name	Feature Description	Value Range
1	<i>EdgeLinearity*</i>	MSE (mean squared error) of edge fitting using the least squares linear fitting on edge pixels	≥ 2 pixels
2	$E^*_{ellipse}$	Percentage of edge pixels uncovered by the fitted dilated ideal ellipse with a circular structure element of radius 4 pixels	[0, 0.8]
3	<i>Curvature*</i>	Curvature value range of a polyp edge	[1/400, 1/20]
4	h_2	Amount of intensity change after crossing the edge	≥ 0.02
5	h_2 / h_1	Ratio of amount of intensity return after crossing the edge to the amount of intensity drop before crossing the edge	≥ 0.5

6	h_3	Sharpness of edge before crossing the edge	≤ 1.67
7	h_4	Sharpness of edge after crossing the edge	≥ 0.652
8	$h_4 - h_3$	Difference between the sharpness of edge after crossing the edge and before crossing the edge	≥ 0
9	h_4 / h_3	Ratio of edge sharpness after crossing the edge to the edge sharpness before crossing the edge	≥ 1.0
10	$V_{texture}$	Texture feature on <i>ROI_{concave}</i> ; see Figure 3.	(0.7, 1]

All parameters in this table were obtained based on experiments with the training data set described in Section 5. They were then used for running the experiments on the testing data set. The * indicates parameters that are sensitive to an input image size.

Edges along the polyp are somewhat curved. We use the feature *Curvature* representing the curvature of an edge and the feature *EdgeLinearity* representing the mean squared error of the edge pixel 2D locations fitted by the least square linear fitting method to remove straight edges. The feature $E_{ellipse}$ is calculated as a percentage of edge pixels that are not covered by the dilated ideal ellipse. The dilation is to account for some nearby edge pixels that are not located perfectly on the ideal ellipse. A threshold for this feature is used to remove edges that do not fit well on an ideal ellipse obtained using the least squares ellipse fitting method [52] on pixel locations of all extracted binary edges. The features h_2 , h_2 / h_1 , h_3 , h_4 , $h_4 - h_3$, h_4 / h_3 are derived from the edge cross section profile, capturing various properties of protrusion. They are labeled in Figure 2. The features are pre-selected using J48 Decision Tree. The value range of these features are pre-determined using the value ranges in rules generated by J48 Decision Tree and manually tuned based on the *leave-one-out-cross-validation* method on the training data as described in Section 5.1.

Algorithm: Simple Texture Calculation

```

/* Calculate a simple texture of  $ROI_{concave}$ 
Input:  $ROI_{concave}$  matrix of size  $W \times H$  pixels
Output: Texture value  $V_{texture}$ 
Local variables
    max: Maximum width (in pixels) of a one dimensional sliding window
    min: Minimum width (in pixels) of a one dimensional sliding window
    max > min; max of 17 and min of 2 were determined empirically
    B: Binary matrix of size  $W \times H$  pixels
    w: Width (in pixels) of the sliding window
*/
SIMPLETEXTUREFEATURE ( $ROI_{concave}$ )
1: if  $W > max$  then  $w \leftarrow max$  // choose the width of the sliding window.
2: else  $w \leftarrow min$  end if
3: Set all values in matrix B to 0
4: for (i=0; i < H; i++) //compare pixel values at the two ends
5:     for (j=w/2; j < W-w/2; j++) // of the horizontal sliding window
6:         if  $ROI_{concave}[j+w/2] > ROI_{concave}[j-w/2]$  then  $B[j] \leftarrow 1$ 
7:         else  $B[j] \leftarrow 0$  end if
8:     end for
9: end for
10: return the average of all the values in the matrix B

```

Figure 3: Algorithm for a simple texture calculation

To calculate the texture feature $V_{texture}$, the region of interest $ROI_{concave}$ is a matrix of pixel values in the region area on the concave side of an edge. See Figure 2(b). In [34], we apply a smoothing filter called *Locally Weighted Scatterplot Smoothing (LOWESS)* [53] on each row of $ROI_{concave}$ (the protruding part of the polyp) to remove the effect of small and impulsive speckle noise before calculating texture features on $ROI_{concave}$. A texture feature is computed to capture the tendency of the intensity increase along the protruding direction of a polyp. In most cases, this tendency is not obvious without any smoothing due to the noise. The LOWESS filter is the most time-consuming component of our feature extraction. Hence, we replace this calculation in [34] by a simpler texture calculation method as shown in Figure 3. Clearly, the protruding direction of a polyp is along the row direction of a $ROI_{concave}$. To capture this protruding characteristic, we only need to consider the width of a $ROI_{concave}$ in the calculation. For a $ROI_{concave}$ with a large width, we use a sufficiently large sliding window to handle the noise as shown in Line 1 of Figure 3 and only estimate the intensity increasing trend based on the two end points of the sliding window. For a $ROI_{concave}$ with a narrow width, we use a small window since it generally has smooth surface. This is because the corresponding edge is most likely a short edge usually caused by light reflection. Lines 4-9 in Figure 3 show the calculation for each pixel. If the pixel value on the right end of the sliding window is greater than the pixel value on the left end of the sliding window, we consider that the intensity increases from left to right. The average value in the matrix B representing

the increasing trend of intensity values is returned as the texture feature $V_{texture}$. The boundary values in $ROI_{concave}$ are not taken into account in the calculation since its majority number of pixels is already sufficient to represent the trend of intensity increase along the horizontal directions. Obviously, as the trend of intensity increase mostly occurs from the edge of a polyp to its most protruding center, only the width that represents pixel values along the horizontal directions of a ROI instead of the height needs to be considered to calculate this texture feature.

3.3. Coarse-to-Fine Feature Rejection for Polyp Edge Detection

Algorithm: Polyp Edge Detection

/* Detect whether an edge given by edge *id* is a polyp edge

Input:

edge[]: Array with each element storing information about an edge

id: Edge ID

T_{MSE} : Mean Square Error Threshold

$T_{ELLIPSE}$: Ellipse Fitting Error Threshold

$T_{CURVE\ RANGE}$: Polyp Curvature Range

$T_{FEATURE}[]$: Array of thresholds of corresponding key point features

Threshold values are specified in Table 2.

Output: *true* if the input edge is detected as a polyp edge; *false* otherwise

*/

```

ISPOLYPEDGE (edge[], id,  $T_{MSE}$ ,  $T_{ELLIPSE}$ ,  $T_{CURVE\ RANGE}$ ,  $T_{FEATURE}[]$ )
1: // MSE: Linear line fitting error representing the edge linearity
2:  $MSE \leftarrow \text{MSEOFEDGELINEARITY}(\text{edge}[id])$ 
3: if  $MSE < T_{MSE}$  then return false end if // a linear edge is unlikely a polyp edge
4: if  $\text{ELLIPSEFITTINGSCORE}(\text{edge}[id]) > T_{ELLIPSE}$  then return false end if
5:  $\text{curvature} \leftarrow \text{EDGECURVATURE}(\text{edge}[id])$  // Calculate edge curvature
6: if  $\text{curvature}$  not in  $T_{CURVE\ RANGE}$  then return false end if
7: // Calculate ECSP profiles and features for these profiles
8:  $\text{edge}[id].\text{profiles} \leftarrow \text{ECSPPROFILES}(\text{edge}[id])$ 
9:  $\text{edge}[id].h_1, \text{edge}[id].h_2, \text{edge}[id].h_3, \text{edge}[id].h_4 \leftarrow \text{KEYPOINTFEATURES}(\text{edge}[id])$ 
10: if  $\text{edge}[id].h_2 / \text{edge}[id].h_1 < T_{FEATURE}[0]$ 
11:   or  $\text{edge}[id].h_3 > T_{FEATURE}[1]$ 
12:   or  $\text{edge}[id].h_4 < T_{FEATURE}[2]$ 
13:   or  $\text{edge}[id].h_4 - \text{edge}[id].h_3 < T_{FEATURE}[3]$ 
14:   or  $\text{edge}[id].h_4 / \text{edge}[id].h_3 < T_{FEATURE}[4]$  then
15:     return false
16:   end if
17:  $V_{\text{texture}} \leftarrow \text{SIMPLEEXTUREFEATURE}(\text{edge}[id].ROI_{\text{concave}})$ 
18: if  $V_{\text{texture}} \leq T_{FEATURE}[5]$  then return false end if // no intensity increase from left to right
19: return true

```

Figure 4: Polyp edge detection algorithm

An edge is detected as a polyp edge if all the corresponding features in Table 2 are within their value ranges. Otherwise, it is detected as a non-polyp edge. To minimize the execution time, we use a coarse-to-fine feature rejection strategy. That is, we calculate one feature after another, starting from the calculation of the feature with the fastest computation time. We list the computation orders of these features in Table 2.

If the value of any feature is not in its value range as specified in Table 2, the edge is detected as a non-polyp edge without the calculation of the subsequent features. This coarse-to-fine strategy avoids calculating all features for non-polyp edges, reducing the overall analysis time especially for images with many binary edges. Figure 4 shows the pseudo code of our polyp edge detection. Method names are shown in all capital letters. The SIMPLEEXTUREFEATURE method is shown in Figure 3 whereas the other methods are omitted since the ideas come from our previous paper [34]. Lines 2-6 identify a linear edge or an edge that does not quite fit an ideal ellipse as a non-polyp edge. Lines 8-16 filter out edge profiles

that do not demonstrate important polyp properties. Line 17 filters out edges without the texture feature of the intensity increase trend calculated using our algorithm in Figure 3. We consider an image as a polyp image if any of its edges is detected as a polyp edge.

4. Polyp Shot Extraction and Alert

This step is the key contribution of this paper. It combines edge tracking with our fast polyp image detection described in Section 3. Edge tracking is used to improve the recall of polyp detection and reduce the number of falsely detected edges. We track *Canny* edges obtained from the *Canny* edge detector across multiple frames and detect whether the edges are polyp edges. A real polyp edge should appear in subsequent frames in a short time window. Otherwise, the detected polyp edge is considered a falsely detected edge. We use Pyramid Lucas-Kanade (Pyramid LK) to track the downsampled edge pixels on each *Canny* edge. Each edge pixel is tracked by searching the corresponding edge pixel in a predefined window in the next frame. If at least a predefined number of pixels on an edge are successfully tracked crossing two frames, an edge is successfully tracked. Only the tracked edges are used as the input for the polyp edge detection. A detected polyp edge is highlighted and displayed as visual feedback after it has been detected in a sufficient number of consecutive clear frames. Multiple polyps per image can also be detected and given feedback on if the edges of these polyps satisfy the aforementioned criteria. The details are as follows.

4.1. Edge Tracking

Pyramid LK tracking is a sparse optical flow tracking method which has a reliable performance when given points with good features to track [54, 55], such as corners. The method calculates the tracked feature points from the top layer of the first Gaussian Pyramid. The calculated feature points are used at the next Gaussian Pyramid layer for motion estimation. Generally, Pyramid LK with a larger Gaussian Pyramid level (PL) has a better tracking accuracy but a higher computational time. A larger PL is better for tracking an edge with a large motion because it searches the corresponding tracked point in a finer Pyramid image.

Selection of feature points to track: Many types of local features can be used for tracking. Most popular trackable local features are the pixels with large derivative values in two orthogonal directions [54]. These pixels are considered as interest points, such as corners [56]. Generally, we do not consider pixels on edges good candidates for tracking because many edges of objects have a local linear shape. The pixels on edges with local linear shape have only large derivative values in one direction, but small

derivative values in another orthogonal direction. However, if pixels on edges have large derivative values in both orthogonal directions, these edge pixels can be considered as good points to track, such as pixels on curve edges. Polyp edges are curve edges in most cases. We consider a subset of pixels on polyp edges as good feature points to track.

Track feature points between two edges: We track feature points – pixels on two edges i and $i+1$ in two consecutive frames t and $t+1$ as follows. First, we downsample the pixels on edge i using the sampling factor of L_e , i.e., selecting one pixel out of every L_e pixels on that edge. We use the downsampled edge pixels as feature points. To track feature points, we apply the OpenCV function `cvCalcOpticalFlowPyrLK` [54] which implements the Pyramid LK tracking algorithm. The function calculates the error ε_t for each feature point as the difference of pixel values between the feature point and its tracked point. The tracked point is selected by searching within a small square window W_{ε_t} of the radius size R_w surrounding the location of the feature point. Two feature points are successfully tracked if both of the following conditions are satisfied: 1) the tracking error $\varepsilon_t < Thld_{\varepsilon_t}$; 2) the distance $dist_t < Thld_{dist_t}$, where $dist_t$ is the Euclidean distance between the feature point on edge i in frame t and its tracked point on edge $i+1$ in frame $t+1$.

Track edges between two frames: We consider that the edge i in frame t track the edge $i+1$ in its consecutive frame $t+1$ if at least one feature point on edge i has its tracked point on edge $i+1$. If there is more than one such tracked edge in the previous frame t , we select the edge with the largest number of tracked points as the tracked edge $i+1$ of the edge i .

4.2. Polyp Shot Extraction and Alert Feedback

Besides polyp image detection and edge tracking, we have two additional challenges to overcome. **Challenge 1:** The polyp may temporarily disappear from the FoV but quickly reappears. This could be due to many reasons. For instance, the endoscopist may briefly rotate the camera away to check the surrounding area, move the camera too fast or too close, or inject water into the colon for cleaning, causing blurry frames. We should group these frames together into the same polyp shot to minimize false boundaries. **Challenge 2:** The feedback on the detected polyp should be shown as soon as possible but in the manner that minimizes false feedback. We handle these complexities in our algorithm in Figure 5.

We define *triggering duration* (N_p) as the number of times (clear frames) a tracked edge must be detected as a polyp edge (by the polyp edge detection algorithm in Figure 4) before we give feedback on it. This is to address Challenge 2. For each clear frame, we set N_p to either N_{short} or N_{long} , where $N_{short} < N_{long}$. Initially, we set N_p to the longer duration N_{long} to reduce the chance of recognizing a false start shot boundary (Line 15 in Figure 5). However, once the start boundary of the shot has been identified and

all tracked polyp edges are lost in this frame and either 1) the current frame is still within the wait duration (Line 18 in Figure 5) or 2) there are remaining non-polyp edges tracked in this shot (Line 19 in Figure 5), we set N_p to the shorter duration, N_{short} . This is to show feedback quicker on the polyp that was previously alert, but disappeared and reappeared again in the same shot.

To address Challenge 1, we do not terminate the current shot right way at the first frame without any tracked polyp edge in the frame. We continue to check for a new polyp edge for the *wait duration* (N_{wait}) defined as the number of clear frames since the last frame with at least one tracked polyp edge. We terminate the current shot only if no polyp edges are found during the wait duration and no tracking of any edges in the shot remains. We check the latter condition because if some non-polyp edges are still tracked, the polyp that was previously in FoV is more likely to be close by and may soon reappear within the FoV. N_{wait} should be larger than N_{short} to reduce the chance of missing an alert on a detected polyp edge when the camera moves fast.

In the algorithm in Figure 5, we use five list variables. The *polypShot* is the return variable listing shot information for all the detected shots, including the shot id as well as the starting frame and the ending frame of each shot. The *trackedEdge* is a list of tracked edge information between two clear frames, *prevFrm* and *currFrm*. The *polypEdge* records all the tracked polyp edges within a polyp shot. The *lostPolypEdge* records the tracked polyp edges in the shot which have lost their tracking. Any detected polyp edge finally loses the tracking of it when the camera moves away from the polyp. Therefore, we keep copying pointers to edges from *polypEdge* to *lostPolypEdge* when these edges can no longer be tracked (Lines 51-53) When all detected polyp edges lose their tracking, the two lists contain the same edges. The *edgeSet* records all tracked edges (including polyp edges and non-polyp edges) in frames with polyp edges in the same shot. Obviously, if two tracked frames are in the same shot, the two lists, *trackedEdge* and *edgeSet*, have at least one common edge (either a polyp edge or a non-polyp edge). The lists *edgeSet*, *polypEdge*, and *lostPolypEdge* are reset to empty before searching for the next shot. Lines 18 – 23 apply some of these variables to implement the aforementioned method for setting N_p to either N_{short} or N_{long} . Lines 25-30 handle the situation when the current shot ends. In Lines 25-27, if no end frame of this shot has been set previously, it is set to the last polyp frame found. In many cases, the end frame of the current shot has already been set earlier in Lines 21-23 to be the most recent frame with at least one tracked edge (either a polyp edge or non-polyp edge) in the current polyp shot. Line 29 increments the shot ID and clears the content of the variables for the current shot. At this point, the start frame of the new shot has not been set.

Algorithm: Polyp Shot Extraction

/* Extract polyp shots with the starting and ending frames in shots

Input: *video*: Input video

N_{short}, N_{long} where $N_{long} > N_{short}$

N_{wait} : Wait duration (number of clear frames)

Output:

polypShot[:]: List of detected polyp shots with shot ID, start frame ID, end frame ID

Local variables:

N_p : Number of clear frames to wait before triggering feedback; set to either N_{short} or N_{long}

edgeSet[:]: List of all tracked edges (polyp and non-polyp edges) in the same shot

polypEdge[:]: List of detected polyp edges

lostPolypEdge[:]: List of lost polyp edges

trackedEdge[:]: List of tracked edges between two frames

Other functions:

emptyList() clears the content of each of the input lists

isEmpty() returns true if the list is empty; otherwise, returns false

isOverlap() returns true if the two input parameters have overlapping content; otherwise, false

isEqual() returns true if the two lists have the same content, otherwise, returns false

*/

POLYPSHOT (*video*, N_{short} , N_{long} , N_{wait})

1: // Clear the content of these list variables

2: **emptyList** (*polypShot*, *edgeSet*, *polypEdge*, *lostPolypEdge*, *trackedEdge*)

3: *lastTrackedPolypFrmID* $\leftarrow 0$; // Frame ID of the most recently tracked polyp edge

4: *sID* $\leftarrow 0$; // initialize the polyp shot count to zero

5: **while** (*currFrm* $\leftarrow$ GETFRAME(*video*)) $\neq$ NULL **do** // process every frame in the video

6: **if** ISBLURRYFRAME(*currFrm*) **then**

7: **continue**; // ignore blurry frames

8: **end if**

9: // extract Canny edges using Canny edge detector

10: *currFrm.edge* $\leftarrow$ CANNYEDGES(*currFrm*);

11:

12: // track Canny edges between *prevFrm* and *currFrm* and save them into *trackedEdge*[:]

13: *trackedEdge*[:] $\leftarrow$ TRACKEDGES (*prevFrm.edge*, *currFrm.edge*);

14:

15: $N_p \leftarrow N_{long}$; // Set to a larger number of frames to see before triggering feedback

16: **if not** isEmpty(*lostPolypEdge*) **and** isEqual(*lostPolypEdge*, *polypEdge*) **then**

17: // if all tracked polyp edges are lost, check the wait duration

18: **if** *currFrm.id* - *lastTrackedPolypFrmID* < N_{wait} **or** // if the loss is within the wait period

19: isOverlap(*trackedEdge*, *edgeSet*) **then** // or a non-polyp edge is tracked

20: $N_p \leftarrow N_{short}$;

21: **if** isOverlap(*trackedEdge*, *edgeSet*) **then** // set the shot ending frame

22: *polypShot*[*sID*].*endFrame* $\leftarrow$ *currFrm.id*; // *sID*: a detected polyp shot id

23: **end if**

24: **else**

25: **if** *polypShot*[*sID*].*endFrame* == NULL **then** // if no edge is tracked

26: *polypShot*[*sID*].*endFrame* $\leftarrow$ *lastTrackedPolypFrmID*;

27: **end if**

28: // finish the current polyp shot

29: *sID* $\leftarrow$ *sID* + 1; // increase the number of the polyp shots

30: **emptyList**(*polypEdge*, *lostPolypEdge*, *edgeSet*) // clear the content of the lists

31: **end if**

Algorithm: Polyp Shot Extraction (continued)

```

32:  //process each tracked edge with  $eID$  in the  $trackedEdge[]$ 
33:  foreach  $eID$  in  $trackedEdge[]$ 
34:      // count how many times a tracked edge is detected as a polyp edge
35:      if  $trackedEdge[eID].count < N_p$  and  $IsPOLYPEDGE(trackedEdge[eID]) == \text{true}$  then
36:           $trackedEdge[eID].count \leftarrow trackedEdge[eID].count + 1$ ;
37:      end if
38:      // keep displaying detected polyp edge(s) on a screen to alert the endoscopist
39:      if  $trackedEdge[eID].count == N_p$  then
40:           $DISPLAYALERT(trackedEdge[eID])$ ; // display and alert function
41:           $polypEdge[].INSERT(eID)$ ; // record the detected polyp edge  $eID$ 
42:           $lastTrackedPolypFrmID \leftarrow currFrm.id$ ; //the polyp frame id
43:           $edgeSet[].UNION(trackedEdge[])$ ; // record all tracked edges for this shot
44:          if  $polypShot[sID].startFrame == \text{NULL}$  then
45:               $polypShot[sID].startFrame \leftarrow currFrm.id - N_p + 1$ ; // the shot starting frame id
46:          end if
47:      end if
48:  end foreach
49:
50:  // record the lost polyp edges since it can no longer be found in the last pair of clear frames
51:  foreach  $eID$  in  $polypEdge[]$  but not in  $trackedEdge[]$ 
52:       $lostPolypEdge[].INSERT(eID)$ ;
53:  end foreach
54:
55:  // set the previous frame to the current frame
56:   $prevFrm \leftarrow currFrm$ ;
57: end while
58:
59: return  $polypShot[]$ ;

```

Figure 5: Polyp shot extraction algorithm

The starting and ending frames of a polyp shot: For each tracked edge, the system keeps counting the number of times the edge is detected as a polyp edge. If the current frame $currFrm.id$ has at least one tracked edge with its count value reaching the trigger duration N_p , the visual feedback on this edge is triggered and the system records the frame $currFrm.id - N_p + 1$ as the starting frame of a new polyp shot (Lines 44-46). The system also records the current frame number containing at least one tracked polyp edge in the variable $lastTrackedPolypFrmID$. The $lastTrackedPolypFrmID$ is repeatedly updated until all tracked polyp edges are lost. If after this frame, some non-polyp edges are still tracked for some period of time without any tracked polyp edge, we still consider the frames in the same polyp shot. The system sets the last frame before tracking of all non-polyp edges and polyp edges is lost as the end of the current shot (Lines 21-23 in Figure 5). This condition is checked whether the same edges appear in both $trackedEdge$ and $edgeSet$.

The visual feedback: The $DISPLAYALERT$ method in Line 40 in Figure 5 can be implemented to provide different types of feedback, ranging from a simple text display on the side, an audio cue

indicating that this image shows a polyp, to the precise locations of the detected polyp edges. The latter is more informative but it is more intrusive than the first two types. Once a tracked edge is identified as a polyp edge for N_p times, feedback is given about this edge until tracking on this edge is lost. To determine which type of feedback is most effective in practice requires an extensive study in future clinical trials. We present our implementation that labels the detected polyp edges on screen in Section 4.3.

4.3 Software Implementation and Feedback

We implemented all the algorithms described in this paper in C/C++ using Microsoft Visual Studio 2008 with Open Source Computer Vision library (OpenCV) Version 2.1 [54]. We implemented two new software modules---edge tracking module and polyp shot extraction module written as Windows Dynamic Linked libraries running on our middleware software SAPPHIRE [57].

SAPPHIRE is a novel middleware and software development kit for stream programing on a heterogeneous system of multi-core multi-CPU's with optional hardware accelerators such as Graphics Processing Unit (GPU). A stream program consists of a set of tasks where the same tasks are repeated over multiple iterations of data (e.g., video frames). Examples of such programs are video analysis applications for computer-aided diagnosis and computer-assisted surgeries. Our design goal for SAPPHIRE is to reduce the implementation efforts and ease collaborative software development of stream programs while supporting efficient execution of the programs on the target hardware. SAPPHIRE also comes with several basic modules such as modules for reading frames from MPEG files, grabbing frames from video capturing devices, writing frames to MPEG 2 format, etc. Image frame data can also be processed directly by OpenCV, eliminating the need to create an additional image copy.

A SAPPHIRE program is the middleware with a collection of software modules together implementing desirable tasks. Given a text configuration file listing module names and optional module parameters and values as an input, the SAPPHIRE middleware constructs a parallel task graph of these modules based on the user-defined types of data, prunes unneeded modules from the graph, loads the remaining modules in their own thread to take advantage of multi-core multi-CPU's, and routes the data among the modules according to the task graph. SAPPHIRE provides access to each frame of data in chronological order. The middleware provides an API for modules to communicate with the middleware (as opposed to modules communicating directly with other modules).

Table 3 shows the functional output, the frame analysis rate in frames per second (fps) of each module, and the dependencies among the modules used in this paper. The module for real-time version of the blurry frame detection algorithm presented in [46] was previously implemented. This module outputs a data packet for each image frame it analyzes whether the frame is clear or blurry. It considers a frame as blurry if it does not have a sufficient number of connected edge pixels detected using *Canny* edge detector.

The two new modules are polyp edge detection and polyp shot extraction that take image input from the data source module (from a file or from a capturing device) and data packets output from blurry frame detection. The edge tracking module implemented our edge tracking algorithm which takes *Canny* edges from the clear image detection module as input, and outputs the tracked edge information.

Table 3: Three Modules for Near Real-Time Polyp Detection and Shot Extraction

Module Name	Functional Output	Frame Analysis Rate	Dependency Module Name
Clear image detection	Clear images	30 fps	-
Edge tracking	Tracked edges	30 fps	Clear image detection
Polyp shot extraction and feedback	Alert of detected polyps and extracted polyp shots	10 fps	Edge tracking

Our implementation of feedback marks five red dots spaced at an equal distance along the detected polyp edges in the original image for display. The green circle is displayed as the mean position of the five red dots. It is used to draw the endoscopist attention to the detected polyps. See the two example feedback shown in Figure 6. The visualized feedback with five red dots marks the precise location of edge position of polyp. This design makes it easy for the endoscopist to see. The green circle position indicates the concave side of ROI, which is corresponding to the approximated center location of a detected polyp. The mean position of five red dots is used due to the simplicity of their calculation and the stability of their mean position in terms of camera movement. A formal user study is needed to validate the usability of this design once real-time feedback is achieved.

Figure 6: Example feedback of detected polyp edges

5. Performance Evaluation

We describe our environment setup, performance metrics, and results.

5.1. Environmental Setup

We captured de-identified video files during routine screening colonoscopy at the image resolution of 720×480 pixels at 29.97 fps. Each file contains only one colonoscopy procedure in its entirety about 20 minutes long on average. We randomly selected 61 out of the captured files; 8 video files with polyps were included in the training set, and the remaining 53 files were included in the testing set. Three and five of the training videos were from Olympus and Fujinon endoscopes, respectively. For this study, we only had access to Olympus and Fujinon endoscopes.

Ground truth labeling on training data: Each of these training videos contains at least one polyp. We selected one polyp shot from each video. We manually labeled the starting and ending frames of each shot. This is because our training videos capture the entire colonoscopy procedures, all of which contain polypectomy after a polyp is found. During the removal of a polyp, snares or biopsy forceps appear together with the polyp. These instruments sometimes occlude the polyp. After the polyp removal, a partial polyp or raised scar may remain and its shape may be changed. This is because the polyp is big or it is in a difficult location preventing the endoscopist to remove the entire polyp at once. Multiple attempts for removal may be performed. Therefore, we selected polyp shots before the corresponding polypectomy was performed. We considered frames i and j as starting and ending frames of a polyp shot, respectively if that polyp continuously appears during these two frames, and completely disappears at the frame $j+1$. In summary, we obtained eight polyp shots of eight distinct polyps from eight different videos as our training data. In this way, the polyps used for training are different. The colons are also different, which is useful for testing the effectiveness of our algorithms.

Test Data: Among fifty-three test videos, forty-one of them were from Olympus endoscopes and the rest were from Fujinon endoscopes. Twenty-two videos of these fifty-three test videos contain at least one polyp. The remaining thirty-one videos do not contain any polyps. There are totally forty-three different polyps in these testing set, twenty of them are from Olympus video and the rest twenty-three are from Fujinon video. The colons seen in these videos contained little remaining debris. Table 4 describes the information about these videos.

Software and Parameters Values: All parameter values were determined based on experiments with the training data set of 8 polyp shots from the 8 training videos. We used these parameter values in our implementation with SAPPHIRE discussed in Section 4.3. The parameter values may vary for different image resolutions in other data sets. Table 5 lists the parameters values. For the parameter values in Table 2, we used J48 Decision Tree [58] to find the value ranges using images extracted from the training videos. For parameter values in Table 5, we selected the parameter values using *leave-one-out-cross-validation* on the training video clips. That is, for the training set of N training video clips, we chose parameter values that give the best average performance on the $N-1$ polyp training video clips. We first

fixed the values of 14 parameters and varied values for one parameter from a fixed set to obtain the value for this parameter that gives the best average performance on the $N-1$ training video clips. We used these parameter values to test on the single polyp training video clip left out from the training and recorded the performance. We repeated this step N times. Next, we considered the next uninvestigated parameter and fixed the parameter value that we already studied and repeated the same process. It was a time consuming process to obtain the parameter values presented in Table 5.

Table 4: Description of Videos in the Testing Set

Category	Olympus	Fujinon	Total
Number of videos	41	12	53
Number of polyps	20	23	43
Average video length (hh:mm:ss)	00:18:05	00:28:53	00:20:21
Total video length (hh:mm:ss)	12:11:58	05:46:41	17:58:39

Table 5: System Parameters and Values

No.	Parameter Name	Valid Parameter Value/Range
1	Gaussian filters	Standard deviations of Gaussian filters: $\sigma_1 = 9$ and $\sigma_2 = 3$
2	Canny edge detector	Low threshold of 10; high threshold of 15
3	D_i	Sliding widow size for the calculation of edge normal: 30x30 pixels
4	λ	The parameter in Pyramid LK tracking: $\lambda = 0.25$
5	W_t^*	Size of the window for Pyramid LK tracking: 15x15 pixels
6	$Thld_{\varepsilon_t}^*$	Threshold of error ε_t of tracked edge pixels; $Thld_{\varepsilon_t} = 500$
7	$Thld_{dist_t}^*$	Threshold of the Euclidean distance between an edge pixel in frame t and its tracked pixel in frame $t+1$: $Thld_{dist_t} = 40$ pixels
8	R_w	Size of a window centered at the tracked edge pixel: 3x3 pixels
9	L_e	Downsampling factor of edge pixels on an edge: $L_e = 14$
10	N_{short}	Minimum number of consecutive times on different frames an edge is detected as a polyp edge in order to consider this edge as a warned polyp edge after losing edge tracking: $N_{short} = 2$
11	N_{long}	Minimum number of consecutive times on different frames a tracked edge is detected as a polyp edge for this edge to be considered as a warned polyp edge: $N_{long} = 3$
14	PL	Gaussian Pyramid level of Pyramid LK tracking: $PL = 5$
15	N_{wait}	Wait duration (number of clear frames) to recall the loss of polyps in a shot $N_{wait} = 3$

All parameters in this table were obtained based on experiments with the training data set. They were then used for running the experiments on the testing data set.

* Parameters sensitive to an input image size

5.2. Performance Metrics

Table 6 shows the notations and definitions of our performance metrics to measure the effectiveness. A polyp is correctly alerted if the corresponding polyp shot contains at least one image showing that polyp. Essentially, if we can provide an alert for each polyp even for a brief time period, it may prevent this polyp from not being detected. A high value for $\text{recall}_{\text{alert}}$ is desirable. The average ratio of the duration of all the detected polyp shots to the duration of all the actual polyp shots is another potential performance metric. Although the duration includes frames without any polyps (no feedback) in the polyp shot, the higher ratio means more chance for the endoscopist to observe the feedback if the feedback is easy to miss. Our feedback is quite obvious. In our algorithm, a detected polyp edge has to be tracked in multiple consecutive clear frames before the feedback is displayed on it. The feedback on this polyp edge continues as long as it is tracked. In other words, we do not apply the polyp edge detection when tracking of the alert polyp edge is still successful. Feedback will stop when the tracking of this polyp edge is lost. As a result, flickering of feedback really depends on the amount of camera motion. To understand the flickering behavior in detail, we need to quantify the motion of the camera within a polyp shot. We defer this study to future work when real-time feedback is achieved and a user study of feedback is conducted. We propose three other metrics for false shots as shown in Table 6 as too many false alarms can interfere with the examination. We want low values for these false alarm related metrics. We measured the average execution time for individual modules and the combined execution time per image on an Intel (R) Xeon 2.0 GHz duo-core CPU with 6.0 GB RAM on a 64-bit workstation with Windows 7 Professional operating system.

Table 6: Performance Metrics on Effectiveness

$\text{recall}_{\text{alert}} = \frac{\text{correctly alerted polyps}}{\text{Number of real polyps}}$
$\text{avg\#FL}_{\text{alert}} = \frac{\text{Total number of falsely alerted polyp shots in all tested videos}}{\text{Number of tested videos}}$
$\text{avgLenFL}_{\text{alert}} = \frac{\text{Total duration in seconds of all falsely alerted polyp shots}}{\text{Number of falsely alerted polyp shots in all tested videos}}$
$\text{ratioLenFL}_{\text{alert}} = \frac{\text{avgLenFL}_{\text{alert}}}{\text{Average video length in seconds}}$

5.3. Performance

Table 7 shows the effectiveness of polyp shot extraction. Our system correctly alerted 97.7% (42 of 43) of polyps. We only missed one polyp entirely. The clear image detection module detected all frames of this polyp as blur (out of focus) frames. For the rest of the polyps, we detected at least 3 consecutive clear frames of each polyp. Our technique is robust to different viewing angles and polyp region sizes. It can also detect polyps with little protrusion. Some images of the detected polyps were missed. These misses include 1) a polyp that appears as a tiny region in an image and 2) a polyp with weak illumination. Polyps from these two categories were seen at a far distance from the camera, which makes it difficult to obtain good Canny edges. No polyps that were not identified by the endoscopist were detected by the software.

The total duration of false shots on average is 4.3% of a video duration. That is, our system can correctly alert 97.7% polyps in near real-time and does not generate falsely detected images for the 95.7% duration in an entire colonoscopy procedure. The average number of false shots per video is 36.2 and the average duration of each false shot is only 1.4 seconds. Therefore, our polyp detection module did not analyze these frames. The false shots are from the following categories: 1) protruding colon folds with smooth surface, 2) the protruding appendiceal orifice, 3) the ileocecal valve, and 4) smooth water regions. Out of these four categories, smooth water causes the least false positives. Most of stool and diverticula are correctly rejected by the system.

Table 7: Effectiveness of Polyp Shot/Alert Extraction

Endoscope Type	Polyp Shot Extraction	
	Performance Metrics	Value
Olympus	$\text{recall}_{\text{alert}}$	95%
	$\text{avg\#FL}_{\text{alert}}$	32.8
	$\text{avgLenFL}_{\text{alert}}$	1.3 s.
	$\text{ratioLenFL}_{\text{alert}}$	4.3%
Fujinon	$\text{recall}_{\text{alert}}$	100%
	$\text{avg\#FL}_{\text{alert}}$	48.0
	$\text{avgLenFL}_{\text{alert}}$	1.5 s.
	$\text{ratioLenFL}_{\text{alert}}$	4.4%
Total	$\text{recall}_{\text{alert}}$	97.7%
	$\text{avg\#FL}_{\text{alert}}$	36.2
	$\text{avgLenFL}_{\text{alert}}$	1.4 s.
	$\text{ratioLenFL}_{\text{alert}}$	4.3%

Table 8 lists the execution time of our system to analyze an image for edge tracking and polyp detection modules. They were run at different frame analysis rates – 30 fps for the clear image detection

module, 30 fps for the edge tracking module, and 10 fps for the polyp detection module. The polyp detection module includes both polyp edge detection and shot extraction. We used the clock function in C++/C to record the starting time and ending time of each module running on each frame. We measured the average execution time of two modules running on 1) only clear images and 2) both clear and blurry images. The edge tracking module and the polyp detection module, each took on average less than 30 ms per image to analyze a clear image. When we ran the modules on both clear and blurry frames, it took on average around 20 ms per image. The lower average execution time is because the polyp detection module skips the blurry frames. Based on the output of our clear image detection module running on the test data, on average, 72.4% of images are classified as clear images and the rest of 27.6% of images are classified as blurry images. That is, every classified 7 images contain 5 clear images and 2 blurry images. For a small percentage of clear frames with many edges, the edge tracking and polyp detection modules were as much as 7 times more than the real time requirement of 33 ms. Keep in mind that the execution times do not include the time for multiplexing between different modules and SAPPHERE. Compared to existing work, our technique is by far the most promising for approaching real-time feedback to the endoscopist.

Table 8: Average Analysis Time for Edge Tracking and Polyp Detection tested on the Testing Set (Milliseconds (ms) per image)

Functionality	Clear images		All images			
	mean	median	Mean	median	minimum	maximum
Edge tracking	27 ms	16 ms	19 ms	16 ms	0 ms	203 ms
Polyp detection	28 ms	16 ms	20 ms	16 ms	0 ms	188 ms

The test data in Table 8 consists of frames extracted from close to 18-hour length of colonoscopy videos under the frame analysis rate of 10 fps. Among all extracted frames, 72.4% of them are clear images and the rest of 27.6% are blurry images.

Limitation of our study: The limitation is like most existing vision-based polyp detection methods. First, we did not classify polyps into different types using existing classification schemes such as Paris classification [59]. Second, only polyps that appear within an image can be detected. If polyps are hidden in locations not inspected (i.e., not in the field of view of the camera), our technique cannot find them. Third, we tested on two commonly used endoscope brands. However, our technique is not limited to these two brands as it does not use specific information about the endoscope hardware, just videos from these scopes. The software can be trained with training video data for other endoscope brands. Forth, we did not study the correlation between the polyp detection performance and bowel preparation quality. Unless polyps are covered significantly by stool, we do not expect the performance of our technique to drop

significantly.

6. Conclusion

In this paper, we present real-time feedback for polyp detection and polyp shot extraction. Our algorithms correctly detected 97.7% (42 of 43) of polyps obtained from 53 video files that were randomly selected from routine screening colonoscopy. Our polyp detection/warning and shot extraction system run in near real-time with an average of 36.2 falsely detected polyp shots per video, and the average duration of each of these false shots is less than 1.5 seconds.

To our best knowledge, our system is the first technique that can effectively detect polyps, alert the endoscopist and extract polyp shots in near real-time. Our system is promising for clinical use as it can assist the endoscopist by providing visual feedback of a potential polyp during routine screening colonoscopy. The immediate future work is the following. We will further reduce computational time by optimizing edge tracking and polyp image detection. In the edge tracking module, different edges are tracked independently of each other, but the OpenCV Pyramid Lucas-Kanade (Pyramid LK) tracking we used is a serial implementation. In the polyp image detection module, the computation bottleneck is the extraction of ROI regions. Each ROI region is extracted independently. These two tasks lend themselves to parallelization. We will investigate multithreading and synchronization as this approach does not incur additional hardware cost. If needed, we will investigate LK tracking using Graphics Processing Units (GPUs) such as that implemented in OpenCV 2.4.9 or in [60] that can process high-definition video in real-time. By parallelizing these tasks, we should be able to achieve the real time performance.

Once our technique reaches the real-time performance, we will conduct a user study for suitable feedback during colonoscopy. Our current feedback design makes it easy for the endoscopist to see the detected polyp and it is easy to compute, but it may not be best for the endoscopist. Automated classification of detected polyp types and sizes is another interesting problem as it can be used to generate a detailed objective report that the endoscopist can simply verify at the end of the procedure. Best images showing a polyp for each shot can automatically be included. Checking whether a polypectomy is performed on the detected polyp and provides warning if no polypectomy was performed may be useful. We believe that real-time polyp detection and appropriate feedback has potential to improve quality of care through improving polyp detection rate and documentation. The feedback should assist to improve the polyp detection rate for experienced endoscopists who may be fatigued due to back to back colonoscopies and for inexperienced endoscopists who may not recognize certain polyps. Lastly, our system has the potential to greatly reduce the amount of paperwork or electronic data entry at the end of the procedure, thereby increasing hands-on "scope" time for actual colon inspection.

Acknowledgment and Disclosure

This work was supported in part by the National Science Foundation STTR Grant No. IIP-0956847, EndoMetric Corporation, and Mayo Clinic in Rochester, MN. Findings, opinions, and conclusions expressed in this paper do not necessarily reflect the view of the funding agencies. Johnny Wong, Wallapak Tavanapong, and JungHwan Oh hold positions in EndoMetric Corporation, Ames, IA 50014, U.S.A, a for profit company that markets endoscopy-related software. De Groen is the medical advisor of EndoMetric.

Reference

- [1] "Colorectal cancer facts & figures," American Cancer Society, 2014 - 2015.
- [2] D. Lieberman, "Quality and colonoscopy: a new imperative," *Gastrointestinal Endoscopy*, vol. 61, no. 2, p. 392–394, 2005.
- [3] A. Pabby, R. E. Schoen, J. L. Weissfeld, R. Burt, J. W. Kikendall, P. Lance, E. Lanza, and A. Schatzkin, "Analysis of colorectal cancer occurrence during surveillance colonoscopy in the dietary prevention trial," *Gastrointestinal Endoscopy*, vol. 61, no. 3, pp. 385-391, 2005.
- [4] K. Rex, J. H. Bond, S. Winawer, T. R. Levin, R. W. Burt, and D. A. J. et al, "Quality in the technical performance of colonoscopy and the continuous quality improvement process for colonoscopy: recommendations of the U.S. multi-society task force on colorectal cancer," *American Journal Gastroenterol*, vol. 97, no. 6, pp. 1296-1308, 2002.
- [5] S. Hwang, J. Oh, W. Tavanapong, J. Wong, and P.C. de Groen, "Polyp detection in colonoscopy video using elliptical shape feature," in *Proc. IEEE Int. Conf. Image Processing*, San Antonio, Texas, 2007.
- [6] D. K. Iakovidis, D. E. Maroulis, and S. A. Karkanis, "An intelligent system for automatic detection of gastrointestinal adenomas in video endoscopy," *Computers in Biology and Medicine*, vol. 30, no. 10, p. 1084–1103, October 2006.
- [7] S. M. Krishnan, X. Yang, K. L. Chan, S. Kumar, and P. M. Y. Goh., "Intestinal Abnormality Detection from Endoscopic Images," in *Proc. of the IEEE Engineering in Medicine and Biology Society*, 1998.
- [8] B. V. Dhandra, Ravindra Hegadi, Mallikarjun Hangarge, and V. S. Malemath, "Analysis of Abnormality in Endoscopic Images using Combined HSI Color Space and Watershed Segmentation," in *Proc. of Int'l Conf. on Pattern Recognition*, 2006.
- [9] J. Kang and R. Doraiswami, "Real-Time Image Processing System for Endoscopic Applications," in *Proc. of IEEE Canadian Conference on Electrical and Computer Engineering*, 2003.
- [10] P. Wang, S. M. Krishnan, C. Kugean, and M. P. Tjoa, "Classification of endoscopic images based on texture and neural network," in *Proc. of Int'l Conf. of the IEEE Engineering in Medicine and Biology Society*, 2001.
- [11] S. A. Karkanis, Dimitrios K. Iakovidis, Dimitrios E. Maroulis, Dimitris A. Karras, and M. Tzivras, "Computer-Aided Tumor Detection in Endoscopic Video using Color Wavelet Features," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, no. 3, p. 141–152, 2003.
- [12] L. Alexandre , J. Casteleiro and N. Nobre, "Polyp detection in endoscopic video using SVMs," in *Proc. of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.
- [13] P. Li, K. L. Chan, and S. M. Krishnan, "Learning a multi-size patch-based hybrid kernel machine ensemble for abnormal region detection in colonoscopic images," in *Proc. of IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2005.
- [14] M. Tjoa and S. Krishnan, "Feature Extraction for the Analysis of Colon Status from the Endoscopic Images," *BioMedical Engineering Online*, vol. 2, no. 1, p. 9, 2003.
- [15] S. A. Karkanis, D. K. Iakovidis, D. A. Karras, and D. E. Maroulis, "Detection of lesions in endoscopic video using textural descriptors on wavelet domain supported by artificial neural network architectures," in *Proc. of IEEE Int'l Conf. on Image Processing*, 2001.
- [16] G. D. Magoulas, V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis, "Tumor detection in colonoscopy using the unsupervised k-windows clustering algorithm and neural networks," in *Proc. of European Symposium on Biomedical Engineering*, 2004.
- [17] G. D. Magoulas, V. P. Plagianakos, and M. N. Vrahatis, "Improved neural networkbased interpretation of colonoscopy images through on-line learning and evolution," *European Network of Excellence on Intelligent*

Technologies for Smart Adaptive Systems, pp. 38–43, 2001.

- [18] G. D. Magoulas, V. P. Plagianakos, and M. N. Vrahatis, "Neural network-based colonoscopic diagnosis using on-line learning and differential evolution," *Applied Soft Computing*, vol. 4, no. 4, p. 369–379, 2004.
- [19] D. C. Cheng, W.C. Ting, Y.F. Chen, Q. Pu, X.Y. Jiang, "Colorectal Polyps Detection Using Texture Features and Support Vector Machine," in *Proc. of Int'l Conf. on Advances in Mass Data Analysis of Images and Signals in Medicine, Biotechnology, Chemistry and Food Industry*, 2008.
- [20] S. Ameling, S. Wirth, D. Paulus, G. Lacey, and F. Vilariño, "Texture-based polyp detection in colonoscopy," in *Bildverarbeitung für die Medizin*, 2009.
- [21] S. Ameling, S. Wirth, and D. Paulus, "Methods for polyp detection in colonoscopy video: a review," 14/2008.
- [22] S. Gross, M. Kennel, T. Stehle, J. Wulff, J. Tischendorf, et al., "Polyp segmentation in NBI colonoscopy," *Algorithmen — Systeme — Anwendungen Proceedings des Workshops, Bildverarbeitung für die Medizin, Springer Berlin Heidelberg*, vol. 22, no. 25, p. 252–256, 2009.
- [23] Tanaka S, Oka S, Hirata M, et al., "Pit pattern diagnosis for colorectal neoplasia using narrow band imaging magnification," *Digestive Endoscopy*, vol. 18, no. S1, p. S52–S56, 2006.
- [24] Z. Qian and Max Q.-H. Meng, "Polyp Detection in Wireless Capsule Endoscopy Images Using Novel Color Texture Features," in *Proc. of World Congress on Intelligent Control and Automation*, June 21–25, Taipei, Taiwan, 2011.
- [25] B. Li, Y. Fan, M.Q.-H. Meng, and L. Qi, "Intestinal polyp recognition in capsule endoscopy images using color and shape features," in *Proc. of IEEE Int'l Conf. on Robotics and Biomimetic*, pp. 1490–1494, 2009.
- [26] B. Li, M. Q. -H Meng, L. Xu, "A comparative study of shape features for polyp detection in Wireless Capsule Endoscopy images," in *Proc. of Int'l Conf. of IEEE Engineering in Medicine and Biology Society*, Minneapolis, MN, USA, September 2009, pp. 373–3734.
- [27] B. Li, M. Q. -H Meng, "Small bowel tumor detection for Wireless Capsule Endoscopy images using textural features and support vector machine," in *Proc. of IEEE/RSJ International Conference in Intelligent Robots and Systems*, St. Louis, MO, USA, October 2009, pp. 498–503.
- [28] A. Karargyris, N. Bourbakis, "Identification of polyps in Wireless Capsule Endoscopy videos using Log Gabor Filters," in *Proc. of IEEE/NIH Life Science Systems and Applications Workshop*, Bethesda, Maryland, USA, April 2009, pp. 143–147.
- [29] D. J. C. Barbosa, J. Ramos, J. H. Correia, C.S. Lima, "Automatic detection of small bowel tumors in Capsule Endoscopy based on color curvelet covariance statistical texture descriptors," in *Proc. of IEEE Int'l Conf. of Engineering in Medicine and Biology Society*, Minneapolis, MN, USA, September 2009, pp. 6683–6686.
- [30] S. Hwang, "Bag-of-Visual-Words Approach to Abnormal Image Detection in Wireless Capsule Endoscopy Videos," *Advances in Visual Computing, Lecture Notes in Computer Science* 6939 (2011) 320–327.
- [31] L. Alexandre, N. N. Nobre, and J. C. Casteleiro, "Color and position versus texture features for endoscopic polyp detection," in *Proc. Int'l Conf. Biomedical Engineering and Informatics*, Sanya, China, 2008.
- [32] S. Park, D. Sargent, et al., "A Colon Video Analysis Framework for Polyp Detection," *IEEE T. Bio-Med. Eng.*, vol. 59, no. 5, pp. 1408–18, 2012.
- [33] J. Bernal, J. Sánchez, and F. Vilarino. "Towards automatic polyp detection with a polyp appearance model," *Pattern Recognition*, vol. 45, no. 9, pp. 3166–82, 2012.
- [34] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Part-based Multi-derivative Edge Cross-Section Profiles for Polyp Detection in Colonoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol. PP, no. 99, pp. 1–11, 2013.
- [35] N. Tajbakhsh, C. Chi, Gurudu, R. Suryakanth, J. Liang, "Automatic polyp detection from learned boundaries," in *Proc. of IEEE Int'l Symp. on Biomedical Imaging*, pp.97–100, April 29 2014–May 2 2014.
- [36] R. Lienhart, "Reliable Transition Detection in Videos: A Survey and Practitioner's Guide," *International Journal of Image and Graphics*, vol. 1, no. 3, pp. 469–486, 2001.
- [37] A. Hanjalic, "Shot Boundary Detection: Unraveled and Resolved," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 2, pp. 90–105, February 2002.
- [38] J. S. Boreczky and L. D. Wilcox, "A Hidden Markov Model Frame Work for Video Segmentation Using Audio and Image Features," in *Proc. of IEEE ICASSP*, Seattle, May 1998.
- [39] I. Koprinska and S. Carrato, "Temporal video segmentation: a survey," *Signal Process.: Image Commun.*, vol. 16, no. 5, pp. 477 - 500, 2001.
- [40] T. Kikukawa, S. Kawafuchi, "Development of an automatic summary editing system for the audio-visual resources," *Trans. Electron. Inform.*, Vols. J75-A, pp. 204–212, 1992.
- [41] B. Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proc. of IS&T/SPIE*, 1995.

- [42] W. Xiong, J.C.-M. Lee, M.C. Ip, "Net comparison: a fast and elective method for classifying image sequences," in *Proc. of SPIE Conference on Storage and Retrieval for Image and Video Databases III*, San Jose, CA, 1995.
- [43] T. N. Pappas, "An adaptive clustering algorithm for image segmentation," *IEEE Trans. Signal Process*, vol. 40, p. 901 – 914, 1992.
- [44] P. Aigrain, P. Joly, "The automatic real-time analysis of film editing and transition elects and its applications," *Comput. Graphics*, vol. 18, no. 1, p. 93 – 103, 1994.
- [45] J. S. Boreczky, L. D. Wilcox, "A hidden Markov model framework for video segmentation using audio and image features," in *Proc. of Int'l Conf. on Acoustics, Speech, and Signal Processing*, Seattle, 1998.
- [46] J. Oh, S. Hwang, W. Tavanapong, P. C. de Groen, and J. Wong, "Blurry frame detection and shot segmentation in colonoscopy videos," in *Proc. of IS&T/SPIE Symp. Electron. Imag.*, San Jose, CA, 2004.
- [47] Y. Cao, D. Liu, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen. Computer-aided Detection of Diagnostic and Therapeutic Operations in Colonoscopy Videos. *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 7, p. 1268-1279, July 2007.
- [48] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Near Real-Time Retroflexion Detection in Colonoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol.17, no 1, pp. 143-152, 2012
- [49] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Computer-Aided Detection of Retroflexion in Colonoscopy," in *Proc. of IEEE Int'l Symp. on Computer-Based Medical Systems*, pp. 1-6, Bristol, U.K., 2011.
- [50] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Detection of Quality Visualization of Appendiceal Orifices using Local Edge Cross-Section Profile Features and Near Pause Detection", *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 3. p. 689-695, 2010.
- [51] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Edge Cross-Section Features for Detection of Appendiceal Orifice Appearance in Colonoscopy Videos", in *Proc. of IEEE Engineering in Medicine and Biology Society*, pp.3000-3003, Vancouver, BC, Canada. Aug.20-24 2008.
- [52] H. Trevor, T. Robert and F. Jerome, *The elements of statistical learning: data mining, inference, and prediction*, 2nd Edition ed., Springer, 2009.
- [53] "Polynomial curve fitting," MathWorks, [Online]. Available: <http://www.mathworks.com/help/techdoc/ref/polyfit.html>.
- [54] A. Zelinsky, G. R. Bradski, *Learning OpenCV-- Computer Vision with the OpenCV Library*, 2008.
- [55] J. Shi and C. Tomasi, "Good Features to Track," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1994.
- [56] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the 4th Alvey Vision Conference*, 1988.
- [57] S. Stanek, W. Tavanapong, J. Wong, J. Oh, R. Nawarathna, J. Muthukudage, P. C. de Groen., "SAPPHIRE middleware and software development kit for medical video analysis," in *Proc. of Int'l Symp. on Computer-Based Medical Systems*, Bristol, U.K., 2011.
- [58] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
- [59] Participants in the Paris Workshop. The Paris endoscopic classification of superficial neoplastic lesions: esophagus, stomach, and colon. *Gastrointestinal Endoscopy* vol. 58, no. 6 (Suppl.), 2003.
- [60] S. A. Mahmoudi, M. Kierzyńska, P. Manneback, and K. Kurowski. Real-time motion tracking using optical flow on multiple GPUs. *Bulletin of the Polish Academy of Sciences, Technical Sciences*, vol. 62, no. 1, 2014.