

# GPU-accelerated Polyp Detection in Virtual Colonoscopy

**María Poó, Eduardo Buysee, Rhadamés Carmona,  
Ernesto Coto and Héctor Navarro**

Centro de Computación Gráfica, Universidad Central de Venezuela  
Escuela de Computación, Caracas, Venezuela, 1041-A  
*malenapoo@hotmail.com, elchuck81@gmail.com,*  
*{rhadamés.carmona, ernesto.coto, hector.navarro}@ciens.ucv.ve*

## Abstract

Nowadays, Virtual Colonoscopy (VC) is an important non-invasive alternative for the study of the colon. Substantial research efforts have been dedicated to this method, and one of the major challenges has always been producing accurate results in a short period of time. One of the most crucial phases of VC is the detection of polyp candidates, where possible lesions on the colon walls are automatically detected. Frequently, this stage requires intensive computations and therefore it is important to develop new techniques for reducing its execution time. This paper presents a technique for automatic detection of polyp candidates based on curvature analysis that reduces the execution time using parallel programming in CUDA. Additionally, we introduce a novel technique for discarding false positive detections based on the shape of a candidate in a planar cut. The obtained results show a remarkable reduction in execution time with respect to a CPU implementation as well as a low rate of false positives.

**Keywords:** Virtual Colonoscopy, Polyp Detection, Curvature Analysis, CUDA.

## 1 Introduction

Cancer is the generic name for a group of diseases consisting on the division and abnormal reproduction of body cells. Cancers are named after the part of the body where they start. When the cancer starts at the colon or rectum is called *colon cancer* or *colorectal cancer*. Nowadays, hundreds of persons in Venezuela [1] and millions worldwide [2] die of colon cancer. This type of cancer is related to the abnormal growth of tissue at the mucous that covers the colon wall, called a *colorectal polyp*. The most common type of polyp is a *metaplastic polyp*, in which cells change from one normal type to another. They have almost no risk of becoming malignant. *Adenomatous polyps* can become malignant. If these polyps are not detected and extracted on an early stage they can degenerate and transform into cancer.

An optical colonoscopy is the gold standard examination for searching polyps in a patient's colon. However, this procedure is very invasive and uncomfortable for the patient. *Virtual Colonoscopy* is an excellent non-invasive alternative for colon polyp detection. In Virtual Colonoscopy, images from the patient's colon are captured with a *Computer Tomography (CT) Colonography* and these images are then processed and virtually browsed in a computer, searching for polyp candidates.

Typically, VC medical workstations provide an automatic virtual flythrough inside the colon, which the doctor can stop at any moment and move the virtual colonoscope freely inside the colon. This searching procedure has been proved effective but it is well known that it can consume a considerable amount of time to examine the complete extent of the colon. Therefore, automatic polyp detection has been suggested as a method for rapidly pointing out relevant areas of the colon to the medical doctor. Several polyp detection methods have been proposed in the literature. Among others we can mention methods based on curvature analysis [3][4][5], curvature lines [6][4], surface adjustment [7], heat diffusion fields [8] and water-planes [9]. All these techniques require a lot of computations, producing long response times.

In this work, we present the parallelization of the algorithm for automatic polyp detection method based on curvature analysis proposed by Yoshida and Nappi [3]. In this method, a shape index is computed for each voxel on the colon wall, which requires the approximation of first and second order partial derivatives, where intensive computations are needed. These derivatives are implemented in a parallel algorithm using CUDA [10], taking advantage of current NVidia's *Graphics Processing Units (GPUs)*. Since curvature analysis can generate numerous false positives, in this work we also propose a strategy for discarding false positives by studying the shape of the polyp candidate in a longitudinal cut. Several tests were performed on CT Colonography scans showing an excellent

response time reduction, as well as an effective decrease of false positives.

The following section describes previous works on automatic polyp detection and the parallelization of VC algorithms with CUDA, and then we describe our new variation of Yoshida and Nappi's method [3] and its CUDA implementation on Section 3. Following this, we describe tests and results of our method over six CT Colonography datasets in Section 4. Finally, conclusions and future work are described in Section 5.

## 2 Previous Works

There are several methods for polyp detection on volumetric data. These methods usually implement two phases. The first phase detects polyp candidates, where structures that might be polyps are detected. On the second phase false positives are discarded.

In order to detect polyp candidates, Yoshida and Nappi [3] proposed in 2001 a method based on curvatures that extracts a thick region of the colon wall to eliminate the largest possible amount of false positives. This method extracts 3D geometric attributes of the polyps using a shape index and a curvature index, grouping polyp candidates and then determining other attributes that might help reducing false positives. That same year, Gokturk and Tomasi [11] used a *Support Vector Machine* (SVM) to eliminate false positives. They used orthogonal planes in the polyp candidate, and extract geometric features that feed the SVM in the training. They also consider voxel intensity to feed up the classifier.

In 2004, Paik et al. [12] introduced an algorithm called *Surface Normal Overlap* (SNO), which is used for colonic polyp detection as well as for lung nodule detection in CT images. Their approach computes for each voxel a score proportional to the number of surface normals that pass through or near it. They noticed that polyp voxels tend to have a higher score than other voxels, due to the presence of convex regions in the surface of polyps.

In 2005, Kitasaka et al. [7] proposed a polyp detection method computing the curvature by means of surface fitting. At each voxel of the colon wall, a second degree polynomial  $f(x,y,z)$  is fitted to the voxel's neighborhood, using minimum squares. Changing the size of the neighborhood allows the detection of polyps with different sizes. The authors report they found less false positives than applying the curvature based method. That same year, Bitter et al. [9] proposed the *water-plane* method. This method works over a triangulated surface mesh of the colon, extracted from the CT data. The method simulates that the colon wall is being filled up with water from a vertex on the mesh, from the outside of the colon, until the water starts to spill out. The idea is finding the set of vertexes that form an almost convex volume. According to their tests, good candidates contain at least 20 vertexes.

In 2006 Zhao et al. [6] introduced a technique for characterizing and visualizing curvature lines on the colon surface in order to detect polyps. The basic idea is to show the colon surface using a local illumination model, and enhance it by displaying the curvature of these surfaces. This method works on the surface mesh of a colon extracted with the *marching cubes* technique [13], and with implicit iso-surfaces. The main advantage of this technique is that it allows enhancing virtual colonoscopy images by adding the curvature lines which allows the radiologist to easily characterize polyps. However, this method has not been adapted to automatic polyp detection.

Later in 2007, Konukoglu et al. [8] proposed to use *Heat Diffusion Fields* (HDFs) for polyp detection. The main idea is using the heat diffusion process to generate a vector field which is correlated with the shape of the colon wall. They found that the heat diffusion process generates a diffusion pattern singularity near the centers of protruding structures, like polyps. They suppose an improvement of Yoshida's method, but a direct comparison between both methods is not included. The main drawback of this method is its high computational requirements.

In 2008, Huang et al. [5] researched on the use of the *Pareto front* for evaluating and enhancing region growing algorithms based on curvature, commonly used for detecting polyp candidates. They evaluated and proposed an improvement of Yoshida and Nappi's work [3] and also reported a reduction of 92% in the number of false positives in comparison with the method of Paik et al. [12].

The works previously described show that it is possible to detect polyps using the explicit representation (surface or mesh), or an implicit representation (a set of grayscale values) of the colon wall. Van Ravesteijn et al. [4] presented in 2009 a pattern for recognizing polyps, which combines the two types of representations and reveals that they are somehow complementary. Each of these approaches characterizes different aspects of the candidates, which are detected by means of the deformation of an explicit representation of the colon surface, or by the modification of the intensity data that contains an implicit representation.

Qui et al. [14] implemented an algorithm of dissection and flattening of the colon (*Virtual Colon Flattening*) using CUDA. In our case, we use CUDA for accelerating the computation of the first and second derivatives, which are needed for computing the curvatures on each voxel of the colon walls. Our work is based on the parallelization of the curvature analysis, whose sequential version is presented in Yoshida and Nappi [3]. The technique is complemented with a heuristic for reducing false positives.

## 3 Method

In this work we parallelized the work of [3] to determine polyp candidates. Initially, the patient's colon is segmented

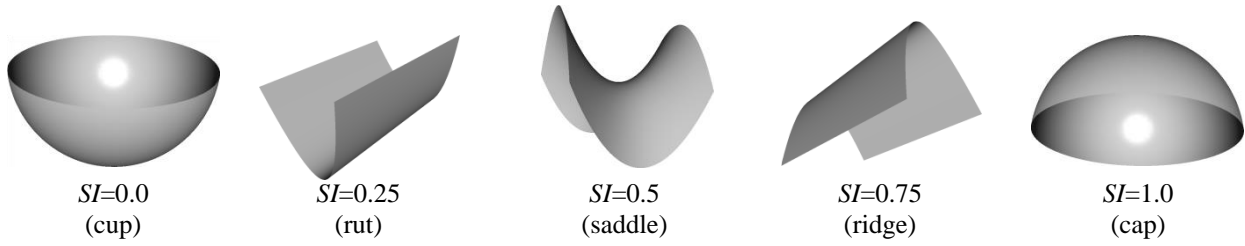
using a region-growing algorithm. After this, the voxels on the surface of the colon wall are obtained. These voxels are classified according to their curvature, and grouped together to generate an initial list of polyp candidates. This list is then refined, removing false positives by studying the shape of the polyp candidate on a longitudinal cut. The elements remaining in the list correspond to the polyp candidates detected by the method. In the following, we describe in detail each of the phases of the method.

### 3.1 Volume Segmentation

In order to segment the volume, an intensity-based region-growing algorithm was used. The algorithm starts growing from a seed voxel inside the colon lumen with an intensity corresponding to air, and then it starts growing through the neighboring air voxels. The growing process stops when it hits the voxels of the colon wall. This process is described in more detail in [15]. Once the air voxels are grouped, the resulting subvolume is dilated until the voxels of the colon walls are obtained.

### 3.2 Voxel Classification

The voxels on the colon wall are classified according to their Shape Index ( $SI$ ) as proposed by Yoshida and Nappi [3]. In their work, they proposed that voxels with a  $SI$  close to 1.0 could be part of a polyp, since this value is associated with the shape of a cap and this shape is similar to the shape of a polyp, see Figure 1.



**Figure 1:** Relationship of the classic surfaces with the Shape Index ( $SI$ ) values

The Shape Index is computed as follows:

$$SI(p) = \frac{1}{2} - \frac{1}{\pi} \arctan \frac{k_1(p) + k_2(p)}{k_1(p) - k_2(p)}$$

where  $k_1(p)$  and  $k_2(p)$  are the main curvatures at position  $p$ . The main curvatures are the curvatures of maximum and minimum value, and to compute them we use the formula proposed by Monga and Benayoun [16]:

$$\frac{h^T H h + u^T H u \pm \sqrt{(h^T H h - u^T H u)^2 + 4(h^T H u)^2}}{2}$$

where

$$\vec{h} = \left( \frac{f_y}{\sqrt{f_x^2 + f_y^2}}, -\frac{f_x}{\sqrt{f_x^2 + f_y^2}}, 0 \right), \quad H = \begin{pmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{pmatrix},$$

$$\vec{u} = \left( \frac{f_z \cdot f_x}{\sqrt{f_x^2 + f_y^2} \cdot \sqrt{f_x^2 + f_y^2 + f_z^2}}, \frac{f_y \cdot f_z}{\sqrt{f_x^2 + f_y^2} \cdot \sqrt{f_x^2 + f_y^2 + f_z^2}}, \frac{\sqrt{f_x^2 + f_y^2}}{\sqrt{f_x^2 + f_y^2 + f_z^2}} \right)$$

with  $f_a = \frac{\partial f}{\partial a}$ ,  $f_a^2 = \frac{\partial^2 f}{\partial a^2}$  and  $f_{ab} = \frac{\partial^2 f}{\partial a \partial b}$ , being  $f$  the function that represents the volume. In order to obtain the approximate value of the first and second degree partial derivatives at a specific point, we need to perform a convolution in that point with an appropriate convolution kernel. That kernel is built based on the method proposed by Möller et al. [17], which can be summarized in 3 steps: Build the 1D interpolation filters, discretize the filters to some number of samples, and build the 3D filters.

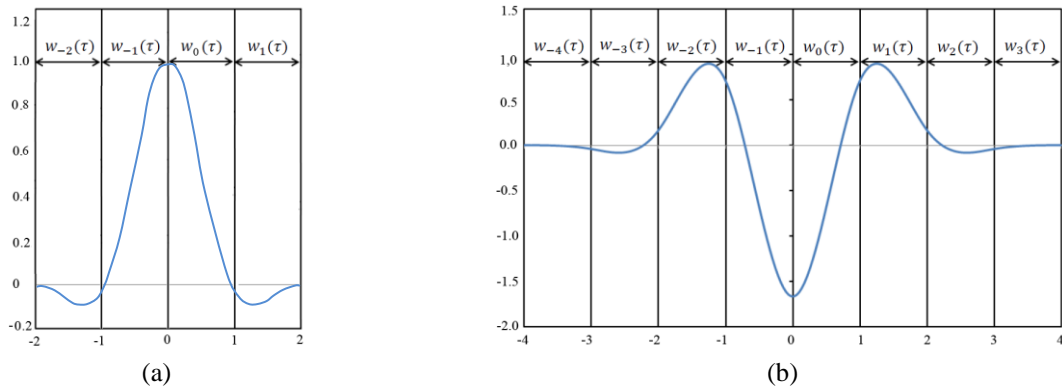
#### 3.2.1 Build 1D interpolation filters

Möller et al. [17] describe how to design interpolation filters and derivatives with a minimum numerical error. In

order to do this, they use Taylor expansions for the sums of convolutions. They also established that in order to build a 1D interpolation filter, first we have to define what derivative we want to reconstruct, what precision we need in the reconstruction process, and what kind of continuity  $C^n$  we need for the reconstructed function.

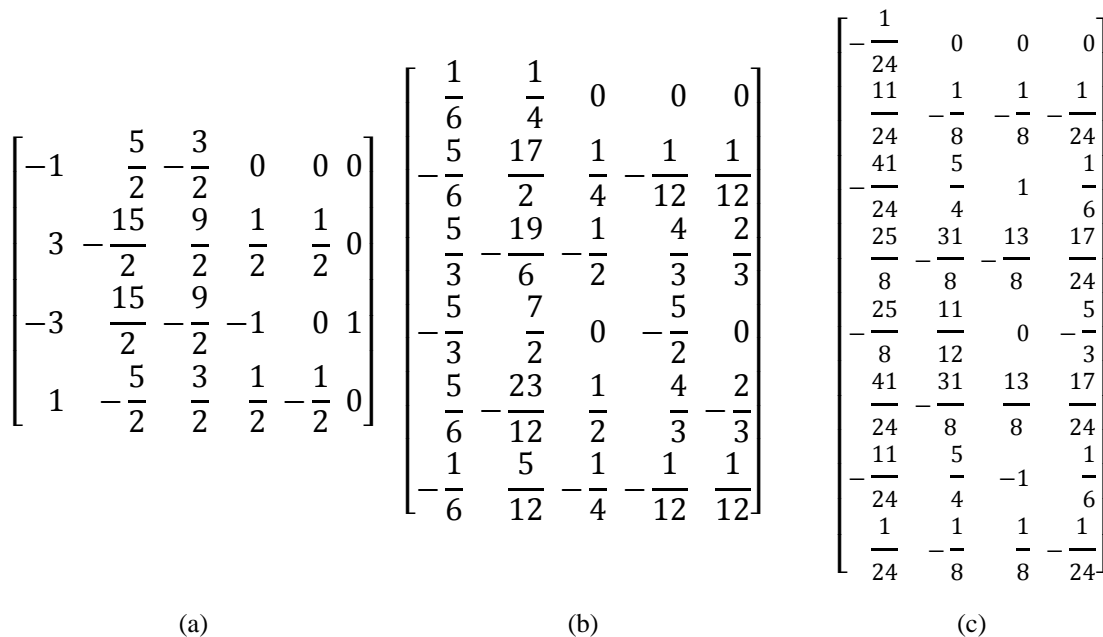
In our case, we need the following filters: a derivative of degree 0 (also known as an interpolation filter), a derivative of degree 1, and a derivative of degree 2. Regarding the precision, we tolerate an error of order 3 and a continuity of  $C^2$ . An equation system is obtained from the Taylor expansion series and the restrictions given by the degree of the derivative, the degree of the error, and the desired continuity. Solving this equation system produces the formula for filter  $w(\tau)$ , in several pieces  $w_k(\tau)$ . With an error of degree 3 and continuity  $C^2$ , the interpolation filter is defined by 4 pieces, the derivative of degree 1 is defined by 6 pieces, and the filter for derivative 2 by 8 pieces. Figure 2 shows two of the obtained filters.

We used the 1D interpolation filter and the 1D derivative of degree 1 from Möller et al. [17]. Using a similar procedure, we obtained the 1D filter for the derivative of degree 2.



**Figure 2:** Filter  $w(\tau)$  defined by 4 pieces for derivative 0 (a), and by 8 pieces for derivative 2 (b)

Let  $[w_{-4}, w_{-3}, w_{-2}, w_{-1}, w_0, w_1, w_2, w_3]^T = M \cdot [t^3, t^2, t, 1]^T$  be the general form of our 1D filters. The coefficient matrix  $M$  for each filter is shown in Figure 3.



**Figure 3:** Coefficient matrix for each filter. (a) Interpolation filter. (b) First derivative filter. (c) Second derivative filter

### 3.2.2 Filter Discretization

To define the filter  $w(\tau)$  from its pieces  $w_k(\tau)$ , we have to consider that the support of each piece  $w_k(\tau)$  is  $[0,1]$ , but it

represents the domain  $[-k, -k+1]$  in  $w(\tau)$  according to Figure 2. Hence, we define  $w_k(\tau) = w(\tau+k)$ . From the obtained filters we create the 1D kernels. For this we take equally separated samples on the domain of the filter. The samples are not taken on the borders of the filter since their value is always zero. Therefore, if we need to obtain  $\eta$  samples, we must divide the filter domain in  $\eta+1$  intervals of the same size, and the samples will be taken in the points that are between two intervals. It should be noted that  $\eta$  must be an odd number. In the tests that we performed we used several filters of different sizes, varying  $\eta$  between 11 and 29.

### 3.2.3 3D Filter Construction

Each of the 3D convolution kernels  $W$  is obtained from the following formula:

$$W[i, j, k] = w_x[i] \cdot w_y[j] \cdot w_z[k]$$

where  $w_x$ ,  $w_y$  and  $w_z$  are the 1D kernels of the derivatives that we want to obtain in the  $x$ ,  $y$  and  $z$ , respectively. For example, to build the kernel that approximates the value  $\partial^2 f / \partial x^2$ ,  $w_x$  would have the values obtained for the 1D kernel of the second degree derivative, while  $w_y$  and  $w_z$  would have the values obtained by the 1D interpolation kernel (derivative of degree 0). In order to approximate all the required partial derivatives, 9 convolution kernels are built, 3 for the first degree partial derivatives:  $\partial f / \partial x$ ,  $\partial f / \partial y$ ,  $\partial f / \partial z$ ; and other 6 for the second order partial derivatives:  $\partial^2 f / \partial x^2$ ,  $\partial^2 f / \partial y^2$ ,  $\partial^2 f / \partial z^2$ ,  $\partial^2 f / (\partial x \partial y)$ ,  $\partial^2 f / (\partial x \partial z)$ ,  $\partial^2 f / (\partial y \partial z)$ .

Once the convolution kernels are obtained, the values of the first and second order partial derivatives in the points on the surface of the colon wall are approximated. Let  $C$  be an array storing that set of points, and assuming that the central position of a filter  $W$  is point  $(0,0,0)$ , the value of the convolution in the point  $p$  in  $C$  is given by the formula:

$$f^{(W)}[p] = \sum_{q \in D(W)} f[p+q] \cdot W[q]$$

where  $f^{(W)}[p]$  is the value of the partial derivative in  $p$  approximated by the convolution kernel  $W$ . Those  $p+q$  values out of the volume are clamped to the volume border. This is done in order to avoid any abrupt cut, because this could produce incorrect polyp detections.

Computing the convolution for all points  $p$  in  $C$  is a process that requires intensive computing. For this reason, this part of the method is implemented using parallel programming in CUDA, whereby multiple threads can be executed in multiple CUDA processing cores. The general idea is to have multiple threads execute the same instructions for different data. If the execution time of threads was unlimited, each thread could compute different convolutions of a  $p$  in  $C$ . However, there are restrictions on the executing time of threads, so the computation assigned to each thread is just a term on the sum of the convolution for a  $p$  in  $C$ . On each thread activation, a term is computed, and the partial sum with that term is updated. In order to avoid consistency problems, the threads must be synchronized before computing each term of the sum. This process is repeated until all the terms of the convolution are computed.

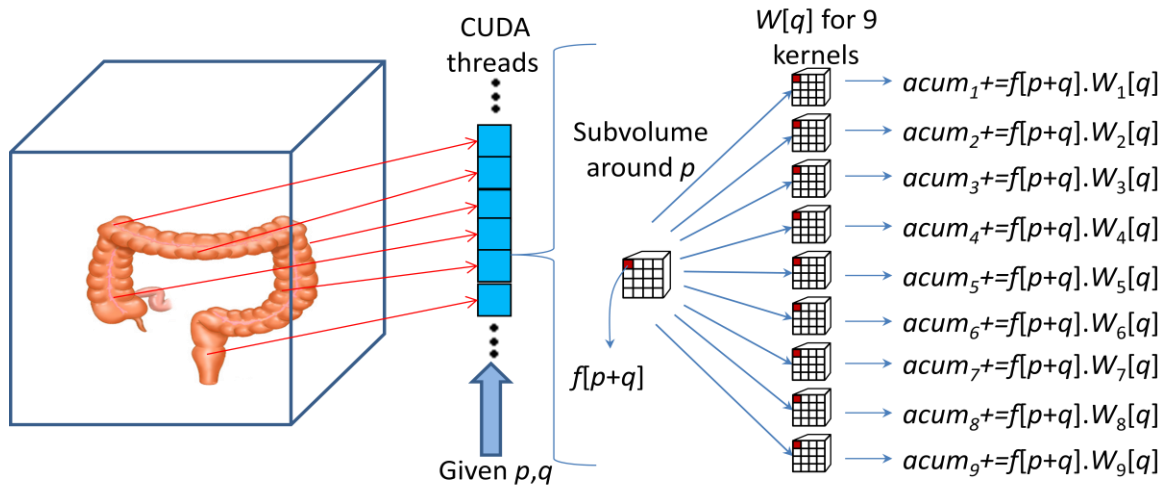
In order to reduce the amount of synchronizations, each thread computes one term of each of the 9 convolutions which must be done for each point. This way, each thread activation performs 9 computations of the form:

$$acum = acum + f[p+q].W[q]$$

where  $p$  represents the position  $(x,y,z)$  of a voxel of the colon wall in the volume, and  $q$  represents a translation  $(dx,dy,dz)$  necessary to compute a term of the convolution between the voxel  $p$  and the filter  $W$  centered on  $p$  (see Figure 4). Each CUDA thread receives a position  $p$  in  $C$  and a translation  $q$  in the convolution filters. A thread computes the convolution term  $f[p+q].W[q]$  for 9 filters  $W$ , so it performs 9 multiplications. On every execution of a thread, it accumulates that term in its accumulator ( $acum$ ), which is equivalent to performing 9 sums. When all the threads that compute a term of the convolution end their execution, the execution of the threads that will compute the next term starts. When all terms have been computed and accumulated on the threads, another group of positions  $p$  in  $C$  is processed, until all the entries in  $C$  have been processed.

### 3.3 Candidate Selection and Grouping

Usually, the top part of a polyp looks like a cup. Therefore we need to select voxels whose neighborhood has a likely shape. As can be seen in Figure 1, if the value of  $SI$  is close to 1.0, the surface near this point presents a shape similar to the desired shape. Empirically, we set that a voxel can belong to a polyp if  $SI > 0.85$ . Voxels  $v$  satisfying this condition are stored in an array we call  $S$ . Based on tests done by Yoshida and Nappi [3], voxel candidates



**Figure 4:** Parallel execution of the algorithm in CUDA

usually gather on top of polyps. Hence, we can group voxel candidates by their proximity, considering that they can be part of the same polyp or structure. In order to accomplish the grouping, a 3D boolean array  $B$  with the same dimensions of the original volume was created, and the positions of the selected points are marked. The algorithm for grouping voxels is the following:

1. Create an empty group  $G$  and an empty auxiliary queue  $Q$ .
2. Move any given voxel from  $S$  to  $Q$ , and unmark it as selected in  $B$ .
3. Extract a voxel  $v$  from  $Q$ , and add it to group  $G$ .
4. Check in a cubic region around  $v$  for other nearby voxel which are selected as well as marked in  $B$ .
5. If such voxels are found, insert them in  $Q$ , delete them from  $S$ , and unmarked them in  $B$ .
6. If there are voxels still left in  $Q$ , return to step 3.
7. If no voxel remains in  $Q$ , group  $G$  is considered a polyp candidate, and therefore it is stored in a list of groups  $LG$ .
8. If there are voxel remaining in  $S$ , return to step 1.

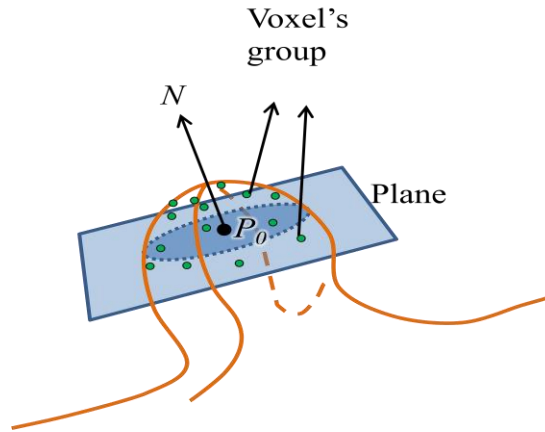
Each of the obtained groups in  $LG$  will be considered a polyp candidate. However, it is necessary to employ an additional technique to verify that in fact the voxels associated to the group are part of a structure with the shape of a polyp. For this, we defined a heuristic for determining whether the group is a good polyp candidate or not, by studying the shape of a planar cut of the candidate.

### 3.4 Planar Cut Extraction

Each polyp candidate has a group of voxels associated to it, which we suppose are located in the top of the polyp. Assuming so, then the volumetric center  $P_0$  of these voxels must be located inside the polyp. Additionally, if we compute a normalized vector  $N$  corresponding to the average normal to the voxels in the same group, and we locate its origin at  $P_0$ , then we can suppose that  $N$  points towards the top of the polyp. Using  $N$  and  $P_0$  we can construct the equation of a plane  $(P-P_0) \cdot N=0$ , which then allows us to capture an image corresponding to a planar cut of the polyp candidate (see Figure 5), so that we can study its shape later on.

### 3.5 Selective Analysis

After obtaining the image of the planar cut we analyze it for selecting strong polyp candidates and discard possible false positives. It is expected that the image would contain an isolated figure, that is, a region with grey intensities (corresponding to tissue) completely surrounded by a region with dark intensities (corresponding to air). If this does not occur, most likely the grey region would intersect the border of the image, indicating that is unlikely that the candidate is a polyp. In this case, the candidate is discarded (see Figure 6). For detecting isolated figures, we use a region-growing algorithm seeded in the center of the image  $P_0$ , with an appropriate threshold.



**Figure 5:** Planar cut of the polyp candidate

Even if an isolated figure is found, if the shape of the grey region is too elongated then it is unlikely to correspond to a good candidate (see Figure 6b). For discarding such candidates we consider the relation  $R$  between the region's perimeter  $p$  and area  $A$ :  $R(p, A) = p^2/A$ , which indicates how elongated is the region. For instance, for a circle  $R(p, A) \approx 12,56$ ; for a square  $R(p, A) = 16$ ; and for a rectangle with longest side  $L=9$  and shortest side  $l=1$ ,  $R(p, A) = 40$ . Note that the higher the value of  $R$  the longest is the figure. Therefore, we could use this criterion for discarding candidates with elongated shapes.



**Figure 6:** Analysis of the planar cut. (a) Strong polyp candidate. (b) The region presents an elongated shape, so the candidate is discarded. In (c) and (d), the candidate is discarded since the grey region intersects the image border

For computing the perimeter of the region found in the planar cut, we simply count the number of pixels in the border of the region. For computing the area, we count the number of pixels inside the region. Following this, those candidates with a value of  $R$  higher than a certain threshold, are discarded. After the analysis, the remaining candidates are considered strong polyp candidates.

## 4 Test and Results

The main goal of the tests was counting the number of false positives removed by the selective analysis heuristic, and comparing the response times between a sequential algorithm for CPU and the parallel algorithm for GPU presented in this paper. Six CT Colonography datasets were employed for the tests. Table 1 shows the characteristics of the test datasets along with the results of the first test for counting the number of detected candidates and the number of false positives removed.

Several key parameters were set for the first test. For the region-growing algorithm which separates the tissue from the air during the planar-cut analysis, we set a threshold of 45 for processing datasets 2, 3, 4, 5 and 6. For dataset 1 we set the threshold to 950 since it has a wider dynamic range and higher intensity values than the other datasets. For studying the shape of the tissue region, a threshold of  $R_{(p,A)} > 30$  was set.

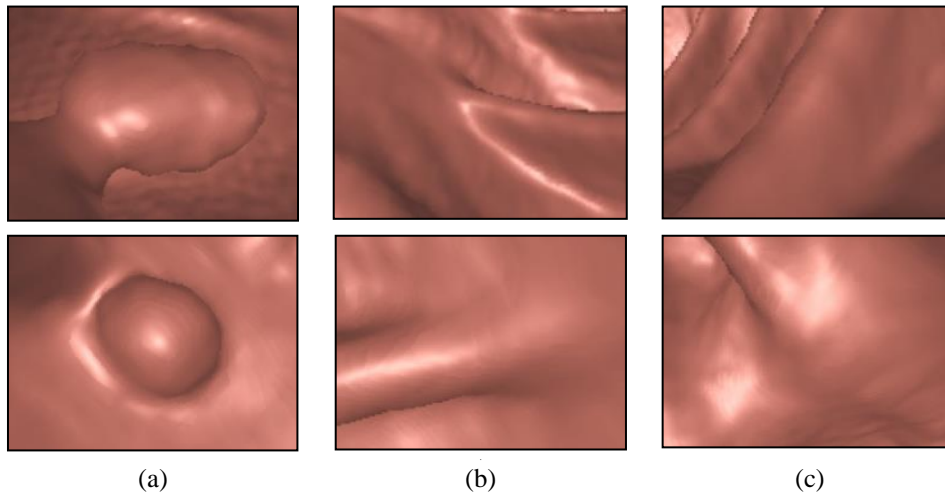
Initially, we varied the size of the convolution filters for the derivative approximation (the  $\eta$  parameter) between  $11^3$  and  $29^3$  samples, in order to find the most appropriate size for the tests datasets, and then we found that the size of the kernels affected considerably the size of the detected candidates. Moreover, we noted that setting the kernel size to  $13^3$  and  $21^3$  was enough to detect polyp candidates of different sizes. Therefore, the kernel size was set to these two values for the first test, see column *kernel size* in Table 1. Nevertheless, it was expected that with these settings, in many cases we were going to detect the same region with both kernel sizes. Therefore, we also computed the number of detected candidates combining both kernel sizes, but eliminating repeated detections, see the last column of Table 1. This table also shows the amount of detected candidates (see column *detections*) which ranges

from 249 to 760, although the majority of them are later on discarded by the planar cut analysis (see column *false positives*).

**Table 1:** Test datasets and results for the first test

| Dataset | Dimensions<br>( $X \times Y \times Z$ ) | Size<br>(MB) | Kernel<br>Size | Detections | False<br>positives | Candidates<br>per kernel | Candidates<br>(combined) |
|---------|---|--------------|----------------|------------|--------------------|--------------------------|--------------------------|
| 1       | 512x512x442                             | 221          | 13             | 266        | 266                | 0                        | 0                        |
|         |   |              | 21             | 268        | 268                | 0                        |                          |
| 2       | 512x512x544                             | 272          | 13             | 334        | 334                | 0                        | 2                        |
|         |   |              | 21             | 296        | 294                | 2                        |                          |
| 3       | 512x512x548                             | 274          | 13             | 760        | 748                | 12                       | 18                       |
|         |   |              | 21             | 734        | 722                | 12                       |                          |
| 4       | 512x512x469                             | 234          | 13             | 302        | 300                | 2                        | 2                        |
|         |   |              | 21             | 249        | 249                | 0                        |                          |
| 5       | 512x512x463                             | 231          | 13             | 388        | 387                | 1                        | 5                        |
|         |   |              | 21             | 387        | 382                | 5                        |                          |
| 6       | 512x512x426                             | 213          | 13             | 404        | 402                | 2                        | 3                        |
|         |   |              | 21             | 383        | 380                | 3                        |                          |

From all datasets, we only had a medical report for datasets 5 and 6. These two datasets were acquired from the same patient. Dataset 5 was acquired in prone position and dataset 6 in supine position. They were both obtained from [18] along with their medical reports. For these two datasets, all the polyps mentioned in the medical reports were detected by our approach. Figure 7 shows screenshots of the detected polyps, along with a few false positives.



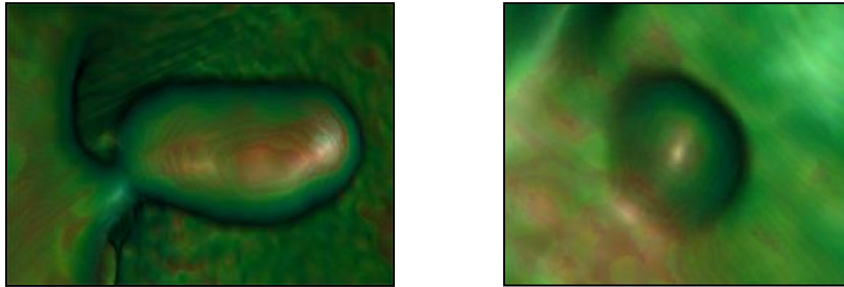
**Figure 7:** Polyp candidates obtained from datasets 5 and 6. (a) Real polyps diagnosed by a medical doctor. (b,c) False positives

We also performed an *electronic biopsy* [19] on the two diagnosed polyps in datasets 5 and 6. The technique consists of producing a translucent version of the VC images by mapping the intensity values in the original data to different colors and opacities, in such a way that the interior structure of the polyps can be visible. In the translucent image the interior of metaplastic polyps looks different than the interior of adenomatous polyps. Figure 8 shows the electronic biopsy for the two diagnosed polyps in Figure 7a.

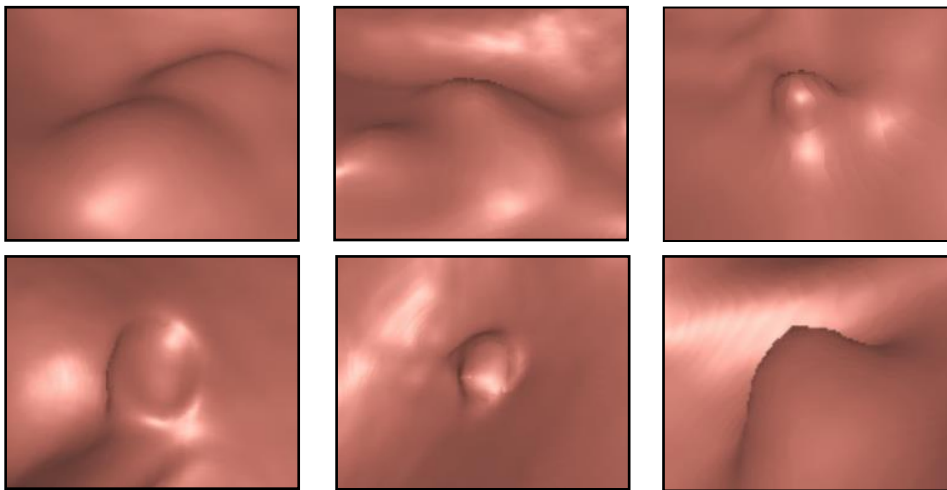
Figure 9 shows screenshots for some of the detections in datasets 1, 2, 3 and 4. Notice that all of them have the typical shape of a polyp. However, we did not have a medical report for these datasets. Therefore, we could only employ the electronic biopsy technique to rule out false detections in these results. Figure 10 shows the electronic biopsy technique being applied on the three polyps at the bottom of Figure 9. Notice that Figure 10b looks different



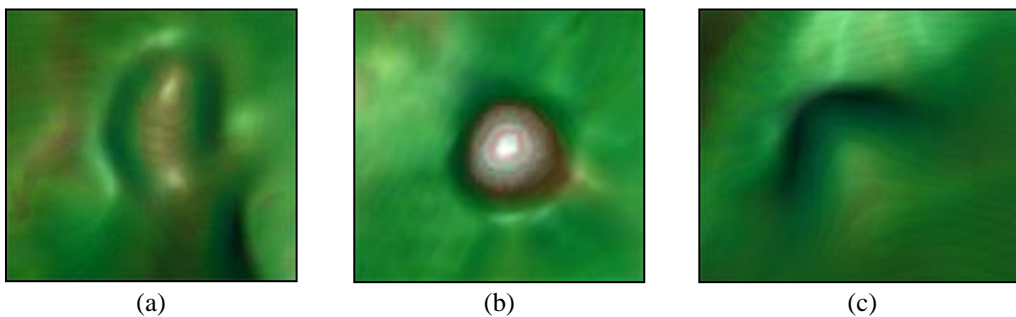
than the others, because this detection actually corresponds to a residual stool ball, not to a polyp. Figures 10a y 10c look similar to the images in Figure 8, but we did not have any medical report to prove that these are real polyps. Therefore, it was impossible for us to calculate the actual accuracy of our method.



**Figure 8:** Electronic biopsy of two polyps diagnosed by a medical doctor in datasets 5 and 6. Both polyps appear to be adenomatous polyps



**Figure 9:** Polyp candidate examples obtained from datasets 1, 2, 3 and 4



**Figure 10:** Polyp candidate examples obtained from datasets 1, 2, 3 and 4; (a) and (c) appear to be adenomatous polyps, while (b) turns out to be a residual stool ball

We also implemented a sequential version of our algorithm for CPU, in order to compare its execution time with the GPU version proposed in this work. Both versions were executed in the two Personal Computers (PCs) shown in Table 2. The results of the tests for each dataset, each test PC and each kernel size are shown in Table 3. In the first PC, we achieved a reduction between 60% and 74.67% in execution time, meaning our proposed algorithm is between 2.5 and 3.94 times faster than the CPU version. In the second PC we achieved a reduction in execution time of between 94.73% and 98.14%, meaning our proposed algorithm is between 19 and 53 times faster.

It is worth mentioning that the CUDA thread hierarchy groups the threads in blocks, where each block has a

shared memory space of fast access for the threads belonging to the same block, therefore allowing the collaboration between such threads for increasing the response time. Given that CUDA allows the developer to decide the number of threads per block, we performed several tests changing the number of threads per block. However, no significant change in the response times was observed, with respect to the times reported in Table 3.

**Table 2:** Characteristics of test computers

| # PC | CPU                           | Mem. | Operating System   | Graphics Card (GC) | # Cores GC | GC Bandwidth | GC Memory |
|------|-------------------------------|------|--------------------|--------------------|------------|--------------|-----------|
| 1    | Intel Core 2 Quad<br>2.40 Ghz | 3 GB | Windows XP 32 bits | GeForce 8800 GTS   | 96         | 64 GB/s      | 640 MB    |
| 2    | Intel Core i3<br>3.07 Ghz     | 4 GB | Windows 7 64 bits  | GeForce GTX 470    | 448        | 133.9 GB/s   | 1280 MB   |

**Table 3:** Results for the second test: Execution times for the sequential algorithm (CPU) and the parallel algorithm in CUDA (GPU) per dataset, test PC and kernel size

| # Dataset | # PC | Kernel size | Time (seconds) |     |
|-----------|------|-------------|----------------|-----|
|           |      |             | CPU            | GPU |
| 1         | 1    | 13          | 110            | 34  |
|           |      | 21          | 537            | 136 |
|           | 2    | 13          | 89             | 3   |
|           |      | 21          | 484            | 10  |
| 2         | 1    | 13          | 102            | 37  |
|           |      | 21          | 470            | 123 |
|           | 2    | 13          | 78             | 4   |
|           |      | 21          | 401            | 10  |
| 3         | 1    | 13          | 247            | 93  |
|           |      | 21          | 1195           | 372 |
|           | 2    | 13          | 200            | 7   |
|           |      | 21          | 1077           | 20  |
| 4         | 1    | 13          | 85             | 34  |
|           |      | 21          | 409            | 126 |
|           | 2    | 13          | 69             | 3   |
|           |      | 21          | 369            | 10  |
| 5         | 1    | 13          | 94             | 39  |
|           |      | 21          | 484            | 169 |
|           | 2    | 13          | 76             | 4   |
|           |      | 21          | 436            | 11  |
| 6         | 1    | 13          | 89             | 36  |
|           |      | 21          | 436            | 140 |
|           | 2    | 13          | 72             | 4   |
|           |      | 21          | 393            | 10  |

## 5 Conclusion and Future Works

In this work we have described the acceleration of an algorithm for the automatic detection of polyps, taking advantage of the parallelism provided by current graphics cards. Specifically, the detection was performed by

curvature analysis, which requires the computation of the first and second order derivatives of the voxels in the colon wall. This computation was implemented in CUDA. The results show a reduction in execution time of up to 98% when comparing the CUDA version of the algorithm with a sequential version for CPU. Several tests were also performed changing the amount of CUDA threads per block, but no significant change in the response time of the algorithm was observed between the different tests.

A new heuristic was implemented for removing polyp candidates which do not have the shape of a polyp. This heuristic consisted on analyzing the 2D shape of the candidate on a planar cut of the polyp, involving its perimeter and area. The amount of removed false positives with this heuristic is significant. Furthermore, the proposed that planar cut analysis could be easily integrated with other detection methods, such as the one developed by Kitasaka et al. [7].

The final polyp candidates obtained by our proposed algorithm are strong polyp candidates according to their shape. However, we could not compute the accuracy of our method because we do not have medical reports for all test datasets.

As future work we propose performing more tests and analyzing the results with medical doctors for determining a concrete percentage of sensitivity and false positives rate. It would also be useful performing tests involving plain polyps, for determining whether our propose method can detect them or not. Additionally, the planar cut analysis could be used to segment the polyps out of the dataset. The idea would be moving the center of the cut in the direction of the normal forward and backwards, so as to obtain several cuts which comply with the conditions of the selective analysis and then select the voxels corresponding to the tissue region on every cut.

### Acknowledgements

This work has been financed by the “Centro de Desarrollo Científico y Humanístico de la Universidad Central de Venezuela (CDCH-UCV)”, under project PG 03-00-6515-2006. The screenshots of the detected polyp candidates were captured using the CT Colonography module of Biotronics3D’s 3Dnet Suite (<http://www.3dnet suite.com/>).

### References

- [1] *Anuario de Mortalidad 2008*. Ministerio del Poder Popular para la Salud. República Bolivariana de Venezuela. May 2010. [http://www.bvs.org.ve/anuario/anuario\\_2008.pdf](http://www.bvs.org.ve/anuario/anuario_2008.pdf).
- [2] “Cancer”, *Fact Sheet No. 297*, Media Centre, World Health Organization, Feb. 2012. <http://www.who.int/mediacentre/factsheets/fs297/en/>
- [3] H. Yoshida and J. Nappi, “Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps”, *IEEE Transactions on Medical Imaging*, vol. 20, no. 12, pp. 1261–1274. Dec. 2001.
- [4] V.F. van Ravesteijn, L. Zhao, C.P. Botha, F.H. Post, F.M. Vos and L.J. van Vliet. “Combining Mesh, Volume, and Streamline Representations for Polyp Detection in CT Colonography”, in *Proceedings of the 6<sup>th</sup> IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 907-910. Jul. 2009.
- [5] A. Huang, J. Li, R. Summers, N. Petrick and A.K. Hara, “Improving polyp detection algorithms for CT Colonography: Pareto front approach”, *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1461–1469. Aug. 2010.
- [6] L. Zhao, C. Botha, J. Bescos, R. Truyen, F. Vos and F. Post, “Lines of curvature for polyp detection in virtual colonoscopy”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 885–892. Sep. 2006.
- [7] T. Kitasaka, Y. Hayashi, T. Kimura, K. Mori, Y. Suenagac and J. Toriwaki, “Detection of colonic polyps from 3D abdominal CT images by surface fitting”, in *CARS 2005: Computer Assisted Radiology and Surgery*, vol. 1281, pp. 1151–1156. May 2005.
- [8] E. Konukoglu and B. Acar, “HDF: Heat diffusion fields for polyp detection in CT Colonography”, *Signal Processing*, vol. 87, no. 10, pp. 2407–2416. Oct. 2007.
- [9] I. Bitter, B. Aslam, A. Huang and R. Summers, “Candidate determination for computer aided detection of colon polyps”, in *Medical Imaging 2005: Physiology, Function, and Structure from Medical Images*. A. Amini and A. Manduca (Eds.), pp. 804–809. Apr. 2005.
- [10] *CUDA Home*. NVIDIA Corporation. 2011. [http://www.nvidia.com/object/cuda\\_home\\_new.html](http://www.nvidia.com/object/cuda_home_new.html)

- [11] S. Gokturk and C. Tomasi, “A New 3-D Pattern Recognition Technique with Application to Computer Aided Colonoscopy”, in *Proceedings of the 2001 IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 93-100. Dec. 2001.
- [12] D. Paik, C. Beaulieu, G. Rubin, B. Acar, R. Jeffrey Jr., J. Yee, J. Dey and S. Napel. “Surface normal overlap: a Computer-Aided Detection Algorithm with application to Colonic Polyps and Lung Nodules in Helical CT”, *IEEE Transactions On Medical Imaging*, vol. 23, no. 6, pp. 661–675. Jun. 2004.
- [13] W. Lorensen and H. Cline, “Marching Cubes: A high resolution 3D surface construction algorithm”, *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169. Jul. 1987.
- [14] F. Qiu, Z. Fan, X. Yin, A. Kaufman and X.D. Gu, “Colon flattening with Discrete Ricci Flow”, in *Proceedings of the MICCAI 2008 Workshop: Computational and Visualization Challenges in the New Era of Virtual Colonoscopy*, pp. 97–102. Sep. 2008.
- [15] E. Coto and S. Grimm, “Improved data processing for virtual colonoscopy”, in *Proceedings of the IV Iberoamerican Symposium in Computer Graphics (SIACG 2009)*, O. Rodriguez et al. (Eds.), pp. 171–179. Jun. 2009.
- [16] O. Monga and S. Benayoun, “Using partial derivatives of 3D images to extract typical surface features”, in *Proceedings of the Third Annual Conference of AI, Simulation and Planning in High Autonomy Systems: Integrating Perception, Planning and Action*, pp. 225-236. Jul. 1992.
- [17] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju and R. Yagel, “Design of accurate and smooth filters for function and derivative reconstruction”, in *Proceedings of the IEEE Symposium on Volume Visualization 1998*, pp. 143–151. Oct. 1998.
- [18] *VolVis.org*. <http://www.volvis.org>.
- [19] M. Wan, F. Dachille, K. Kreeger, S. Lakare, M. Sato, A. Kaufman, M. Wax and Z. Liang, “Interactive electronic biopsy for 3D virtual colonoscopy”, in *Physiology and Function from Multidimensional Images*, ser. Proceedings of SPIE, C. Chen and A. Clough (Eds.), vol. 4321, pp. 483–488. May 2001.