Feature Detector and Descriptor for Medical Images

Dusty Sargent[*a],Chao-I Chen[b], Chang-Ming Tsai[b], Yuan-Fang Wang[b], Dan Koppel[a]
[a] STI Medical Systems, 733 Bishop Street, Honolulu, HI, USA 96813
[b] Dept. of Computer Science, University of California, Santa Barbara, CA, USA 93106

ABSTRACT

The ability to detect and match features across multiple views of a scene is a crucial first step in many computer vision algorithms for dynamic scene analysis. State-of-the-art methods such as SIFT and SURF perform successfully when applied to typical images taken by a digital camera or camcorder. However, these methods often fail to generate an acceptable number of features when applied to medical images, because such images usually contain large homogeneous regions with little color and intensity variation. As a result, tasks like image registration and 3D structure recovery become difficult or impossible in the medical domain.

This paper presents a scale, rotation and color/illumination invariant feature detector and descriptor for medical applications. The method incorporates elements of SIFT and SURF while optimizing their performance on medical data. Based on experiments with various types of medical images, we combined, adjusted, and built on methods and parameter settings employed in both algorithms. An approximate Hessian based detector is used to locate scale invariant keypoints and a dominant orientation is assigned to each keypoint using a gradient orientation histogram, providing rotation invariance. Finally, keypoints are described with an orientation-normalized distribution of gradient responses at the assigned scale, and the feature vector is normalized for contrast invariance. Experiments show that the algorithm detects and matches far more features than SIFT and SURF on medical images, with similar error levels.

Keywords: feature, keypoint, detector, descriptor

1. INTRODUCTION

Feature detection and description are possibly the most important steps in many computer vision algorithms. Distinctive image features can be used to establish matches across multiple images in a video sequence. These matches can be used to carry out many computer vision tasks, such as the estimation of camera motion parameters. In the medical domain, one use of such information is in the creation of a 3D model[6] of internal organs from endoscopic video. In contrast to models produced from CT or MRI data, such as Virtual Colonoscopy[9], a 3D model created from video allows complete visualization of internal organs, containing not only structure, but also color and texture information. In order to build an accurate 3D model, it is necessary to detect and match a large number of reliable features across all video frames.

Our method is based on two popular state-of-the-art discrete feature detection algorithms, SIFT[3,7] and SURF[2]. Both of these methods extract scale, rotation, and contrast invariant features which are described by 128 and 64 element vectors respectively. David Lowe's SIFT(Scale-Invariant Feature Transform) is the most popular feature detector/descriptor currently in use in various computer vision applications. SIFT keypoints are extrema in a difference-of-Gaussian(DoG) scale pyramid. A dominant orientation is determined for each keypoint, which is then described by a distribution of gradient responses in an orientation-aligned window surrounding the keypoint. Keypoint locations are interpolated in scale space to provide continuous scale invariance, and the descriptor is normalized for invariance to contrast, which is a scale factor. An extension to SIFT, called gradient location and orientation histogram[10], changes the descriptor computation and incorporates principal component analysis to increase the distinctiveness of the descriptor at the cost of greater computational complexity.

[*] dustysargent@gmail.com; phone 805-284-8326

SURF, or Speeded-Up Robust Features, includes both a detector and a descriptor. The methods are based on SIFT, with many approximations made in order to increase efficiency. The DoG scale pyramid is replaced with a determinant of Hessian pyramid. The Hessian computation is accelerated using box filter approximations to the second derivatives of a Gaussian. Box filters of any size are evaluated in constant time through the use of integral images. The descriptor is based on the SIFT descriptor, but once again integral images are used to speed up the computation. A distribution of Haar wavelet responses is used to approximate the gradient distribution of Gaussian smoothed images used by SIFT. The Haar wavelet responses are also calculated in constant time through approximation using box filters. These approximations allow SURF to outperform SIFT in terms of efficiency while retaining a distinctive descriptor that provides reliable matches.

Although SIFT and SURF perform well on indoor and outdoor scenes, they fail to produce an acceptable number of features and matches when applied to medical data. Our method combines elements of SIFT and SURF and is specifically tuned for performance on medical data. The following sections describe our methods and show that our algorithm can outperform the current state-of-the-art on medical images.

2. METHOD

2.1 Keypoint Detection

The goal of keypoint detection is to find image points that can be detected reliably in different images of the same scene. Our method uses a version of the determinant of Hessian pyramid for scale-invariant keypoint detection.

2.1.1 Scale Invariance

Scale invariance allows features to be matched across images of a scene taken at different zoom levels or distance from the scene. In order to achieve this, an image pyramid is constructed and features are detected at multiple scales. The first step in the creation of the image pyramid is converting color images into grayscale. We use a standard formula of $(0.3R+0.59G+0.11B)/255$ to convert a color image into an intensity matrix with elements in the range of 0 to 1. As an additional pre-processing step that greatly improves performance on medical data, we also double the image size using bilinear interpolation. We then compute the integral image which allows for fast evaluation of box filters. The integral image representation J of an image I is defined in the following way:

$$J(x,y)=\sum_{i=0}^{x}\sum_{j=0}^{y}I(i,j) \tag{1}$$

Some methods for increasing the efficiency of integral image computation are given in [11], which describes a GPU accelerated SURF implementation. Given this image representation, the sum of image intensities in any axis-aligned rectangle can be computed with only two subtractions and one addition. Thus, instead of repeatedly smoothing successive scale levels to produce new levels, filters of any size can be applied directly to the base image.

Our algorithm uses a modified version of the determinant of Hessian scale pyramid used in SURF. Box filters are used to approximate the second derivatives of a Gaussian, giving the following formula for the determinant approximation at each pixel:

$$\det(H)=D_{xx}D_{yy}-(0.9D_{xy})^2 \tag{2}$$

In equation 2, D_{xx}, D_{yy} and D_{xy} are the box filter approximations to the Gaussian second derivatives. As noted in [11], the original SURF paper leaves many details ambiguous and provides the exact form of the filters for only the first level of the scale pyramid. Therefore, we have made some modifications to suit the needs of medical images

The SURF algorithm uses a filter of size 9 as the base scale level, with the size of following levels in the first octave growing in increments of 6, so the first octave contains filters of size 9, 15, 21, and 27. Due to the shape of the filters, the size must be a multiple of 3 for all levels. For subsequent octaves, the size increment between adjacent filters doubles to 12, then 24, and then 48. This means that the filters at the highest scale level have size 363, corresponding to σ=32.4. Additionally, there are large gaps between successive levels. Noting that medical images are unlikely to experience this kind of extreme zooming motion and that features are harder to detect in general, we chose not to implement the octave system. Instead, we keep the size increment at 6 and allow any number of scales to be used in detection. This approach maintains a smaller jump between scale levels, allows for greater adaptability to specific data, and gives us the opportunity to detect more features since no scale levels are skipped.

The original SURF paper omits many details about the shape of the filters, providing only the exact form of the base level filters. In order to generalize this filter to higher scale levels, we simply scale up the size 9 filters, shown in figure 1, maintaining as closely as possible the ratios of the sizes of the sub-boxes in each filter. This method provides a good discretized approximation to the Gaussian second derivatives. For an *xx* filter of size *S*, the vertical band of nonzero elements has height equal to floor(*S*/2)+1, and the width of the middle box is *S*/3. For the *xy* filter, the size of all sub-boxes is *S*/3 and they remain separated by the middle row and column of the filter as in the base case. The *yy* filter has the same proportions as the *xx* filter. The *xy* filter requires the evaluation of 4 box filters for a total of 12 operations and 16 lookups, while the *xx* and *yy* filters require the evaluation of only two of the boxes, since we can subtract 3 times the response of the middle box from the response of the entire middle band. In order to maintain consistent response magnitude in the filter responses across different scale levels, all filter responses are normalized by the Frobenius norm of the filter. Further optimizations of the filter computations are explored in [11].

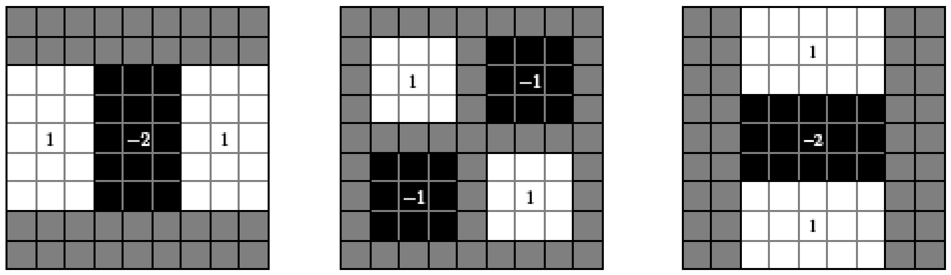

Figure 1. D_{xx}, D_{xy} and D_{yy} second derivative filters.

Keypoints are detected as local maxima or minima in scale space. This means that the value of the approximated Hessian determinant at a candidate keypoint must be greater or less than its 9 neighbors in the scales above and below, and its 8 neighbors at the same scale. Since the value at any pixel and any scale in the pyramid can be computed from the base integral image in constant time, it is possible to avoid storing the image pyramid completely, and simply compute these pyramid values as needed when testing for potential keypoints. This allows for large savings in memory over SIFT, in which the entire image pyramid is usually stored. However, the approach of not storing any of the image pyramid leads to many repeated computations. As a compromise, we store only the 3 levels needed at a time: the current level in which keypoints are being detected, and those below and above it. These 3 levels are labeled current, top, and bottom. When moving to a higher level for keypoint detection, the previous bottom level can be replaced by the new top level, and the previous current and top levels become the new bottom and current levels. As the pixels at the lowest and highest scale levels and those on the edges of the image do not have all 26 neighbors for extremum testing, we skip these points during keypoint detection.

Once a candidate keypoint is detected, it must pass several tests before being accepted as a feature. We use Lowe's method[7] to interpolate the keypoint location to sub-pixel precision. This is a Newton's method[4] based approach, modeling the local scale-space function(the Hessian determinant) with a quadratic and using an iterative process to find the extreme point. The following is a quadratic approximation to the scale-space function *D* about *x*:

$$D(x+\delta) = D(x) + (\nabla D(x))\delta + (\tfrac{1}{2})\delta^{T}(\nabla^{2} D(x))\delta \tag{3}$$

In equation 3, *x* is a 3-vector with the pixel location and scale as components. In order to minimize this model, we can set the left side equal to zero and take the derivative with respect to *δ*. Solving the resulting linear system gives the step *δ* to the minimum of the quadratic model. In Lowe's implementation, if the step is larger than 0.5 in *x* or *y*, the keypoint is determined to be closer to a different pixel, so the process is repeated about that point instead. Due to the inexact nature of computing first and second derivatives from pixel differences, there are cases in which this process will continue to iterate indefinitely. Some versions of SIFT set a limit to the number of iterations, and declare the keypoint unstable if the limit is reached. In our method, we accept the keypoint if the initial step is less than 0.6 in *x* and *y*, otherwise the point is rejected without any iteration. In practice, this method retains the vast majority of kepyoints that are retained by the iterative process, while increasing the efficiency of keypoint localization. Finally, we reject keypoints with low contrast and eliminate points with strong edge response, as they likely lie at a depth discontinuity and thus their 3D location is not well defined. We follow the method in [7] for these two tests, leaving the contrast threshold as a user-set parameter.

2.1.2 Orientation Assignment

In order to achieve invariance to rotation, we must determine a dominant orientation for each keypoint, so that the descriptor can be computed with respect to this orientation. Rotation invariance in this case refers to a 2D rotation of the image which would correspond to rotation the camera about its viewing direction. The SURF paper says that their descriptor is also invariant to a moderate amount of out-of-plane rotation, but there is nothing explicitly encoded in the function to ensure this. In practice, no current local feature descriptor can handle any significant out-of-plane rotation of the camera between image frames, as such motions can significantly change the appearance of local image regions in the two images. Some current feature detectors, such as those discussed in [1] and [8], incorporate affine invariance. However, we follow Lowe's recommendation that the potential gains of supporting affine invariance are outweighed by the added complexity it introduces.

Our orientation assignment method combines elements of the methods used by SIFT and SURF. We compute approximate gradient orientations in a circular window of radius 6*s* surrounding the keypoint, where *s* is the scale at which the keypoint was detected. This is done by computing *x* and *y* Haar wavelet responses, as used by SURF, in a square window with side length 12*s*, ignoring any points that lie farther than 6*s* from the center of the window. The size of the Haar wavelets is 4*s* and the responses are weighted by a keypoint-centered Gaussian with σ=2.5*s*. We compute approximate gradient orientations as follows, where the derivates are estimated by the Haar wavelet responses:

$$\theta(x,y) = \tan^{-1}\left(\frac{\delta I(x,y)}{\delta x} \bigg/ \frac{\delta I(x,y)}{\delta y}\right) \tag{4}$$

We group the gradient orientations into a histogram with 12 bins each covering 30 degrees. This number of bins was determined through experimentation on medical data. The high number of bins(36) used by SIFT was found to be sensitive to noise and outliers, as a single response could dominate the entire histogram, making the orientation unreliable. The exact orientation calculation of SURF is unclear from the paper, but they use a sliding window of size 60 degrees. We found that using only 6 bins smoothed the histogram too much, leading to weak dominant orientations. To determine the final orientation, we use the interpolation method suggested by Lowe. We fit a quadratic to the value of the maximum bin and the bins on its right and left. The maximum point of this quadratic is then taken as the orientation of the keypoint. We generate additional features at the same location and scale for any orientation bin whose value is at least 80% of the maximum value, although this rarely occurs in practice.

An important point to note is that, since keypoints are located to sub-pixel precision, the sampling points for orientation computation are not at integer coordinates. Therefore, we use bilinear interpolation for image sampling:

$$I(x+\Delta x, y+\Delta y) = (1-\Delta x)(1-\Delta y)I(x,y) + \Delta x(1-\Delta y)I(x+1,y) + (1-\Delta x)\Delta y I(x,y+1) + \Delta x \Delta y I(x+1,y+1) \tag{5}$$

So, evaluating box filters becomes much more expensive when they are centered at non-integer coordinates. In fact, through profiling we found that the bilinear interpolation function comprises a large percentage of the total running time of our algorithm. This also affects descriptor computation, which evaluates a large number of Haar filter responses at sub-pixel locations. Methods for reducing the cost of bilinear interpolation are explored in [11].

2.2 Descriptor Computation

Our descriptor is based on the length 64 SURF descriptor. with some modifications. The descriptor is a distribution of *x* and *y* Haar wavelet responses computed in an orientation-aligned box centered at the keypoint. The box is divided into 16 sub-boxes, each of which is sampled on a 5x5 grid. So, computing the descriptor for a single keypoint requires the evaluation of 2*5*5*16=800 Haar wavelet responses. These Haar wavelets are not axis-aligned, since they are computed with respect to the dominant orientation of the keypoint, and they are not centered at integer locations. Since we can only evaluate axis-aligned box filters, our approach is to evaluate axis-aligned Haar filters at each sample point, combine the *x* and *y* responses into a vector, and rotate the vector by the keypoint's orientation to get the orientation-aligned responses.

The descriptor window has size 20*s* and the size of the Haar filters is 2*s*. Filter responses are weighted by a Gaussian with σ=3.3*s*. The 64 elements of the descriptor vector are formed by first summing the *x* and *y* responses in each of the 16 sub-boxes, along with their absolute values. Thus, each sub-box contributes the following 4-vector:

$$v = (\sum \delta_x, \sum \delta_y, \sum |\delta_x|, \sum |\delta_y|) \quad (6)$$

Thus we get a descriptor of length 64. We experimented with reducing the dimensionality of the descriptor to 32 in order to reduce the number of necessary Haar wavelet computations. However, this produced unacceptable degradation in feature matching results. In any case, Haar wavelet responses can be computed with only 6 calls to bilinear interpolation due to the fact that two of the corners are shared by both boxes of the filter. After the wavelets are computed for all sub-boxes, the feature vector is normalized, providing invariance to contrast, which is a scale factor.

2.3 Feature Matching

We use a brute-force nearest-neighbor algorithm to match features between two images. The similarity score between two feature vectors is the magnitude of the difference of their descriptors, so a lower score indicates a closer match. For each feature *p* in image 1, we compute the difference between *p* and every feature *p'* in image 2, keeping track of the best and second-best matches. We accept a match between *p* and *p'* if the difference between *p* and *p'* is less than *t* times the difference between *p* and its second-best match from image 2. Additionally, to prevent points in image 2 from being matched to more than one feature in image 1, we output only the best match for each feature in image 2.

3. EXPERIMENTAL RESULTS

3.1 Comparison of SIFT and SURF

Our decision to base our feature detector on SURF's feature detector was motivated by experimentation on medical data. Testing was done using images from colonoscopy video. Often in these videos, the endoscope's light will produce many glare regions which are often chosen as features by feature detectors. Since the glare regions move when the endoscope moves, these regions are not suitable as features for computing camera motion. Although it is possible to apply glare removal before running feature detection, glare detection must fill in glare regions by interpolation, which often produces artifacts. So, a better method is to use glare detection to located glare regions, and then remove features detected in the glare regions as a post-processing step. An example comparison between SIFT and SURF on adjacent frames showing a polyp is given in figure 2. The same brute-force matching program described above is used in both cases. In this figure, the lines going from left to right connect the matched features between image 1 and 2. As can be seen, nearly all of the features matched by SIFT are located in glare regions. Conversely, SURF detects many features on the polyp outside of the glare region.

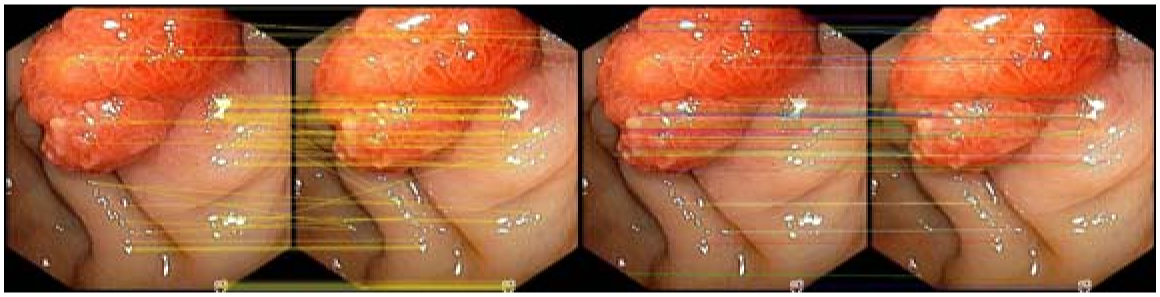
Figure 2. Feature matching comparison using SIFT(left) and SURF(right).

3.2 Comparison between SURF and our Algorithm

We have performed extensive comparisons between our algorithm and the SURF executable provided by the authors. For our experiments, cervical and colonoscopic test images were used. We use the author's recommended parameter settings for SURF. After experimenting with the tunable parameters of SURF, we found that varying things like the step size and number of octaves produced little change in the final results when compared with the recommended parameters. In this comparison, we once again use the same brute-force matching program for SURF and our algorithm, setting the matching threshold to 0.7.

First we show some examples of testing on cervical images. The original size of these images is 600x900. The image transformation in these cases is performed artificially using Matlab. Therefore, we know exactly how to map every pixel in image 1 to its position in image 2, allowing us to calculate the pixel error in the matches returned by both programs. In all cases, we removed outliers before computing average matching errors, although neither algorithm produced very many erroneous matches.

For the first example, we test rotation invariance. Figure 3 shows a feature matching comparison in which the left image has been rotated by 90 degrees to create the right image.

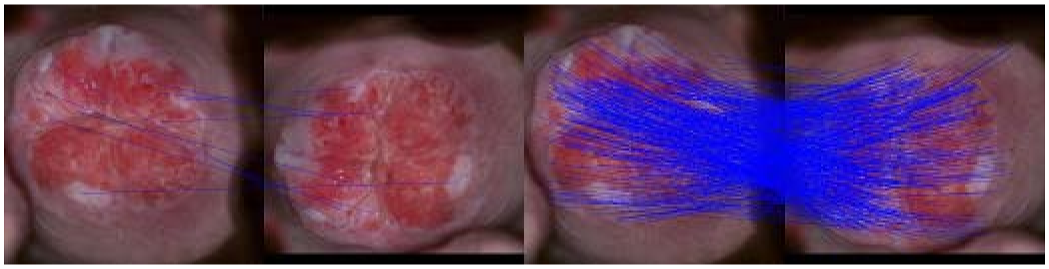
Figure 3. Rotation invariance test. SURF matching(left) and our result(right).

In this case, SURF detected only 18 features in image 1 and 15 features in image 2, and found only 9 correct matches. The average pixel error of these 9 matches is 0.3508. Conversely, our method was able to detect and match 557 features, with a similar average pixel error of 0.3033.

For the second example, we test scale invariance. In this case, image 1 is the same as before. The second image is created by extracting the middle part of image 1 and scaling by a factor of 2 in both dimensions. The matching comparison is shown in figure 4.

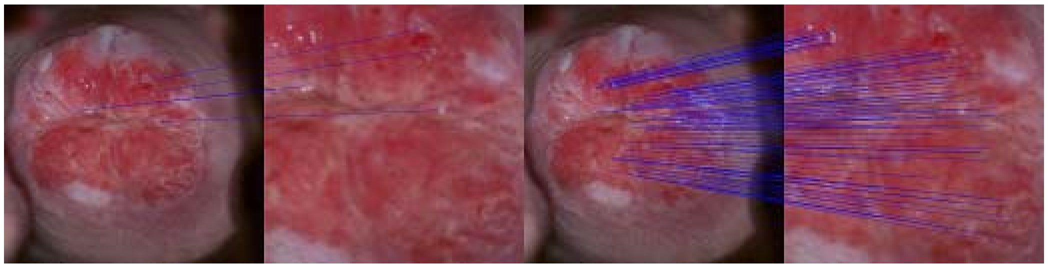

Figure 3. Scale invariance test. SURF matching(left) and our result(right).

In this case, it is not possible to find as many matches as the previous test, because image 2 contains only a small patch form image 1. However, in this case our algorithm was able to successfully match 98 features with an average pixel error of 0.4818. SURF's average matching error was slightly lower at 0.3855, but only 5 matches were detected, below the minimum number of matches needed, for example, to compute camera motion using the 8-point algorithm[5].

Finally, we show an example of feature matching in which the input images are adjacent frames from a colonoscopy video, showing a polyp. In this case we do not know the ground truth transformation, so we cannot calculate the average pixel error of both algorithms. However, since the images are taken from adjacent video frames, the motion is very small and appears to be horizontal. So, any matching in which the *y*-coordinates are nearly the same has a high likelihood of being correct. This means that the lines connecting matched features should be roughly horizontal. Figure 4 shows the final matching comparison.

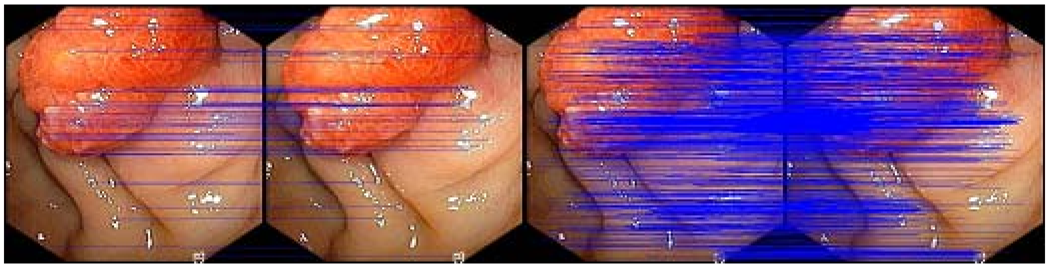

Figure 4. Colon video test. SURF matching(left) and our result(right).

As can be seen, our method was able to detect and match many more features(822) than SURF(62). In addition, our algorithm was able to locate many non-glare features, providing a further improvement in this respect to that obtained by switching from SIFT to SURF.

4. CONCLUSIONS

This paper presents a successful application of discrete scale, rotation and illumination invariant features to medical images, making use of computer vision algorithms possible in the medical domain. Our algorithm incorporates and expands on elements of SIFT and SURF, making changes and tuning parameters to better suit the difficult type of data targeted by the algorithm. We have performed extensive testing of SIFT, SURF, and our method on medical data, of which several examples have been presented here. The goal of future work on this algorithm is to improve the running time. The memory use of our program is quite low, but the running time of our algorithm, while faster than SIFT, is not as fast as the original SURF algorithm. This is in part due to the number of features detected; however, GPU acceleration of our method as explored in previous work would be very helpful in improving the efficiency.

5. REFERENCES

1. A. Baumberg. "Reliable feature matching across widely separated views", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 774-781, 2000.
2. H. Bay, T. Tuytelaars, L.V. Gool. "SURF: Speeded Up Robust Features", *Proceedings of the 9th European Conference on Computer Vision,* Vol. 3951 Part 1.**,** pp. 404-417, 2006.
3. M. Brown and D. Lowe. "Invariant features from interest point groups", *British Machine Vision Conference*, Cardiff, Walves, pp. 656-665.
4. J. Dennis and R. Schnabel. "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", *SIAM,* Philadelphia, PA. 1996.
5. R. Hartley. "In Defense of the Eight-Point Algorithm", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 6, pp. 580-593, June 1997.
6. R. Hartley and A. Zisserman. "Multiple View Geometry in Computer Vision", *Cambridge University Press,* Cambridge, MA, 2003.
7. D. Lowe. "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, Vol. 2, No. 60, pp. 91-110, 2004.
8. J. Matas , O. Chum, M. Urba, and T. Pajdla. "Robust wide baseline stereo from maximally stable extremal regions.", *British Machine Vision Conference*, pp. 384-396, 2002.
9. H. Messman. "Atlas of Colonoscopy", *Thieme*, Stuttgart, Germany, 2007.
10. K. Mikolajczyk and C. Schmid. "A Performance Evaluation of Local Descriptors", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 10, pp. 1615-1630, October 2005.
11. T. Terriberry, L. French and J. Helmsen. "GPU Accelerating Speeded-Up Robust Features", *3D Data Processing, Visualization and Transmission*, Georgia Tech, Atlanta, June 2008.