UNIVERSIDAD CARLOS III DE MADRID

Fluorescence Optic Endoscopy Images Quantification

Bachelor's Degree in Biomedical Engineering

Author: MIGUEL RODRÍGUEZ FERNÁNDEZ

Tutor: JORGE RIPOLL LORENZO

ACKNOWLEDGEMENT

Me gustaría dar las gracias a aquellas personas que han hecho posible la realización de este proyecto.

Primero, a Jorge Ripoll Lorenzo, mi tutor, por darme la oportunidad de llevar a cabo este proyecto, por su interés, dedicación, disponibilidad y por su trato siempre amable.

A mis amigos, que han sido un constante apoyo, no solo durante el proyecto, sino a lo largo de toda la carrera, y que siempre han estado ahí cuando han hecho falta.

A mi familia, que ha hecho lo imposible por ayudarme a lo largo de todo este tiempo, y que con su cariño y apoyo me han hecho llegar donde estoy.

También quisiera dar las gracias a mis profesores y compañeros, y al personal del laboratorio que siempre me han prestado ayuda cuando la he necesitado.

Finalmente, quisiera agradecerle a la universidad, los medios y la oportunidad que me han brindado.

ABSTRACT

The endoscopy is a medical procedure that allows physicians to explore human body cavities without having to use invasive surgery and that provides real-time images, not as external techniques.

In this field of study, the development of the optic fiber endoscopes and fluorescence microscopy have been huge advances, as they have enormously increased the quality of the resulting images, helping to make more effective diagnostics.

However, the images obtained with these techniques have two problems, their small field of view and the fact that are discontinuous images. The first problem difficults making good diagnostics and the second one, difficult working with them.

The program developed during this project is aimed tp solve both problems and so, ,offer better image for diagnostic and also analyze images to automatically detect sickness.

INDEX:

FIGURE LIST

1. INTRODUCTION:

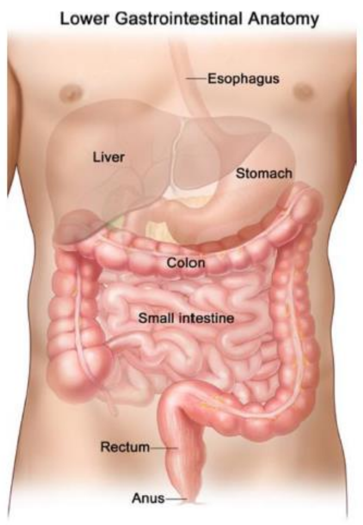
1.1 MOTIVATION

An important field of study in medical science is the research related to gastrointestinal track, An example of diseases related to this system is colorectal cancer, which appears in colon or rectum and is a common cause of death. When leading with this kind of early detection is crucial, in order to achieve it, regular screening methods are used to look for cancer and pre-cancer indicators. One of the most important techniques for prevention and early diagnosis is endoscopy, as it is not harmful and provides information that can not be obtained with other techniques. Because of this, a new fluorescence fiber optic endoscope has been developed. However, the images obtained with this new endoscope present some deficiencies.

The aim of this project is to develop a computer program that solves some of those problems automatically, so the diagnose can be achieved much easily.

1.2 GASTROINTESTINAL TRACK

The gastrointestinal track, also called alimentary canal, is a long tube inside bilaterian animals that is in charge of ingestion, mixing, propulsion and digestion of food, absorption of the resultant substances and, secretion and defecation.



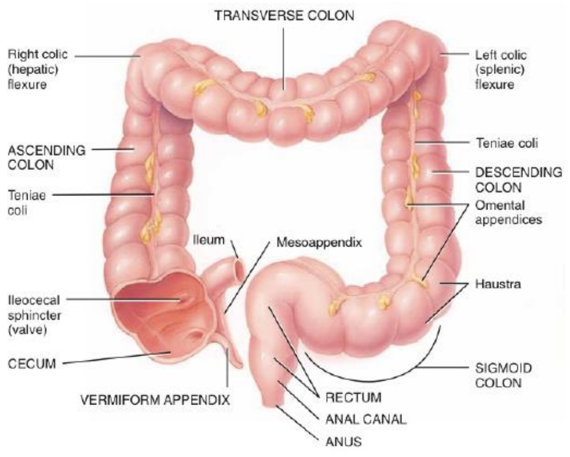
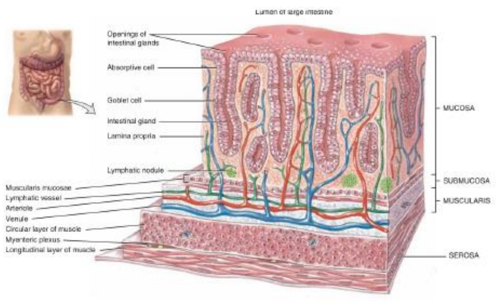


Figure 1. Left: Lower gastrointestinal tract. Right: Large Intestine [1]

The last section of the digestive system is the large intestine, which drives contents of the colon into rectum, converts proteins to vitamin B and K, that then are absorbed as well as water and ions, forms feces and defecates. It is formed by four tissue layers: mucosa, submucosa, muscularis and serosa. For this project, the most interesting of them is the mucosa, that is the outer layer, formed by simple columnar epithelium lamina propia and muscularis mucosae. The epithelium contains absorptive and globet cells, which are both located in straight tubular intestinal glands (crypts of Lieberkühn) that extend to the full thickness of the mucosa.[1]



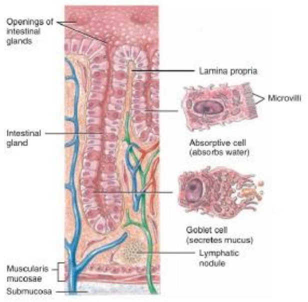

Openings of intestinal glands
Lamina propria
Microvilli
Intestinal gland
Absorptive cell (absorbs water)
Goblet cell (secretes mucus)
Lymphatic nodule
Muscularis mucosae
Submucosa

Figure 2. Detail of large intestine wall. [1]

The rodent's digestive system is very similar to the human's, reason why is very used in research for in vivo tests, but has two main differences, the lack of gallbladder and the longer cecum (section of large intestine), these differences are due to the different alimentary habits of the rodents.

The gastrointestinal track has a significant advantage, that it allows easy direct access, this means the possibility of using noninvasive techniques. Using endoscopy, it is possible to study the texture of a mouse large intestine, including crypts structure.

1.3 COLORECTAL CANCER

It is a type of cancer that starts in the large intestine, in the colon or rectum. It is an adenocarcinoma, that is, a cancer that begins in the cells that release any fluid, and it begins as a growth called polyp, which form on the inner wall of the colon or rectum, evolving some of them into cancer over time. Removing these polyps can also allow preventing this cancer.

To detect indicators of this cancer, several techniques are used: fecal occult blood test, sigmoidoscopy, barium enema and colonoscopy. Virtual colonoscopy and DNA stool tests are new techniques that could be applied also.

It is the third most common type of cancer in USA, but deaths due to it can be reduced with the already mentioned techniques.[2]

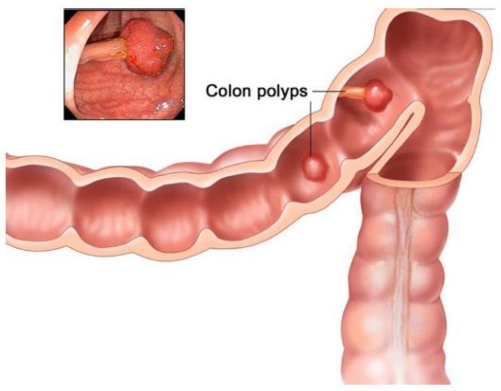


Figure 3. Detail of intestine with colon polips. [2]

1.4 ENDOSCOPY

1.4.1 WHAT IS ENDOSCOPY?

It is a medical procedure that allows looking inside body cavities, by inserting directly the device, called endoscope. The advantage of this technique is that, while other techniques only allow obtaining static information, the endoscopy provides real time information.

The endoscope is composed of four main parts:

- A tube that can be rigid or flexible, which contains fiber bundles and video signal wires and that is inserted inside the cavity to examine. The size of this tube may vary depending on the use that endoscope is going to have.
- A light delivery system.
- A set of lenses and a camera to acquire the image and transmit it.
- An additional channel to allow entry of instruments and manipulators.

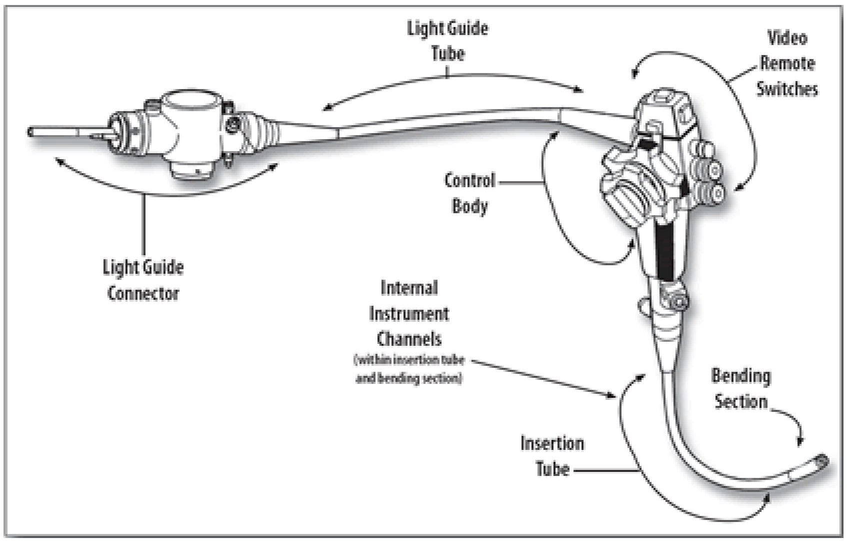


Figure 4. Parts of an endoscope. [3]

1.4.2 A BRIEF HISTORY:

Until the XVIII century, the only body cavities that could be examined were those more external, like mouth, external audition canal, vagina, rectum… It was Philip Bozzini who developed the first endoscope in 1806. However, it was Antonin Jean Desormeaux who achieved an endoscopy, following the actual criteria.

During the XIX century, research continued in endoscopy, and different devices were developed, including in new models, the scientific advances that were being produced, the most important of this advances, was the discovery of electricity, that allowed to use electric light to illuminate the cavities, until that moment the light was originated by a candle.

With the beginning of the XX century, the endoscopy had already become a well-established discipline. In 1928, Schindler made another great advance and created the first flexible tub endoscope, until that moment they were rigid although not straight. Later, in 1945 Karl Storz introduced the use of bright cold light, simultaneously to the image transmission.

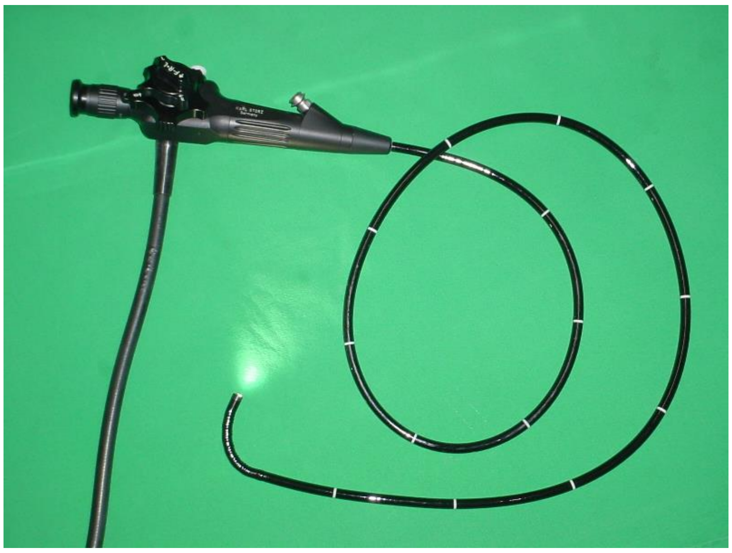


Figure 5. Image of a tube flexible endoscope. [4]

1.4.3 FIBER OPTIC ENDOSCOPY:

The first fiber optic endoscope was invented by Basil Hirschowitz and Larry Curtiss in1957.In this endoscopes, a set of glass fibers transmitted light from one side to the other. This allowed the coherent transmission of an image through the endoscope, and, as it demonstrated being very useful, research was developed and fibers to transmit light from a external source to the patient were added.

These endoscope can be inserted in small cavities that could not be accessed otherwise, and give high quality imaging. They mean new options for high resolution fluorescence imaging, and during this project, it is going to be developed the program for a fluorescence fiber-optic endoscope that uses a single bundle that transmit light and captures image simultaneously.

1.4.4. FLUORESCENCE MICROSCOPY:

Basic optical microscopy is based on the use of visible light and a set of lenses that, combined, magnify images. Fluorescence microscopy, a new and more advanced type of microscopy, allows to characterize certain substances based on the fluorescence. In this microscopy, it is used a detector that measures the radiation emitted by those substances once they have absorbed fluorescent radiation. This capability is not common to all substances, and those which have it are called fluorophores, and in order to use this technique they have to be in the sample, naturally (genetic modification) or having been added (immunofluorescence).

This technique has meant a huge advance in medical research, as it allows to access organs that could only be reached with surgery, and also gives the capability to process certain signals that could not be seen before, leading to the study of already known physiological processes from a new point of view.

2. OBJECTIVES

The objective of the project is to develop a computer program that takes the images from the endoscope and process them, spending the minimum time possible, and covering these three points:

2.1 OBTENTION OF CONTINUOUS IMAGES:

The endoscope is composed of a bundle of fibers, instead of a single one. Because of this, the images obtained are not continuous, but formed by a set of little circles, covering each one of them, part of the field of view of the device. Thus, the resultant images are similar to honeycomb, presenting black spaces, which lack any information, between the circles.

The program should eliminate the black spaces and obtain a continuous image, as close as possible to the original one. In order to do that, it is needed a program that uses the information present in each circle, to interpolate the one that is missing in black spaces.

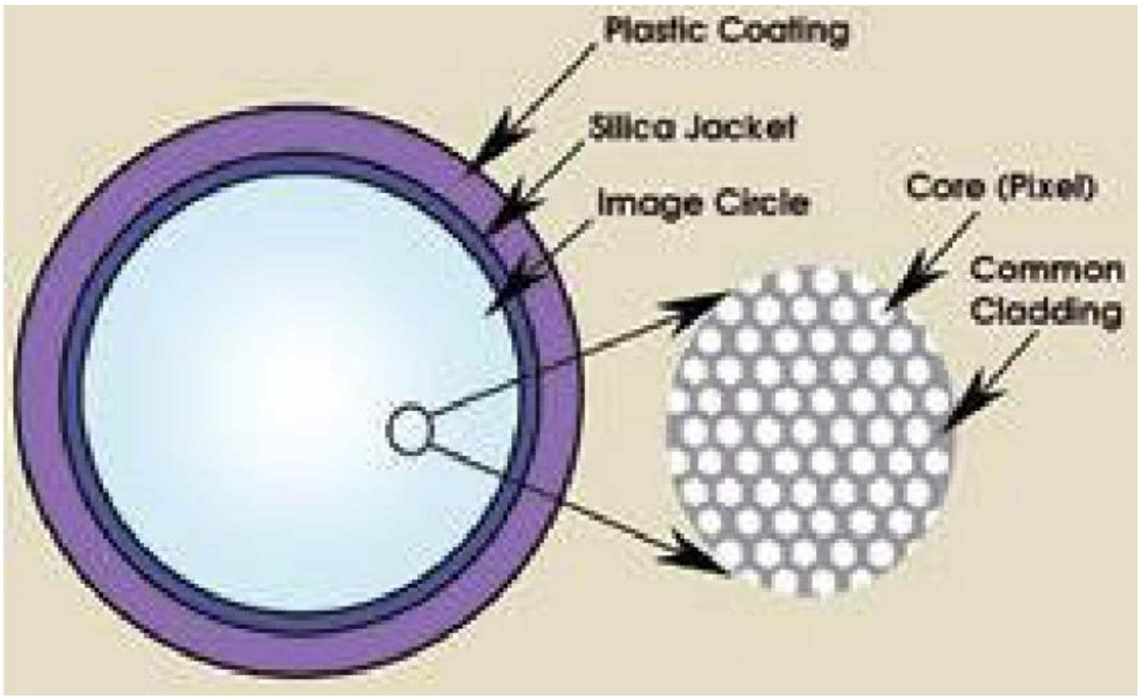


Figure 6. Drawing of the fiber bundle endosocpe detector. [5]

2.2 ANALYSIS OF THE IMAGES LOOKING FOR INFLAMATION INDICATORS:

The intestine wall is not a flat surface, it presents crypts that appear in the endoscopy images as dark areas. In healthy patients, those cells are similar and, thus, the dark areas in the images obtained show all a similar size and shape. However, in case of inflammation, they are irregular and the area covered by them is greater.

So, the program should be able to process the images of the endoscope video in order to measure the area of those dark regions. Having measured already known healthy and sick patients endoscopy images, it should be possible for the program to determine, by comparison, the presence or lack of inflammation.

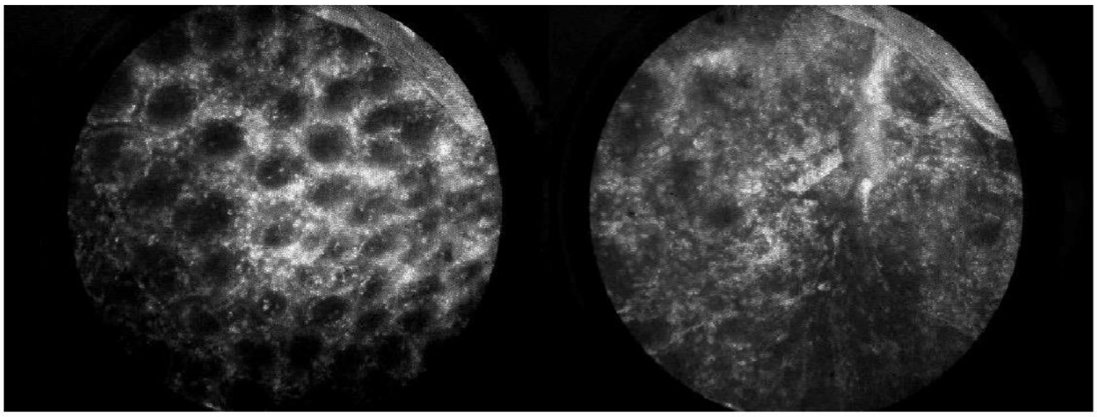


Figure 7. Fluorescence fiber bundle endoscope images. [6]

2.3 EXTRACTION OF A SINGLE IMAGE OF THE WHOLE ENDOSCOPY:

The field of view of the endoscope is very small, this is due to its slight size, essential for the use of the endoscope, and also because of the fact that the endoscope is too close to the intestine wall. This difficults the examination of the intestine, as the area covered at the same time is really small.

In order to solve this problem, the program should combine all the images obtained by the endoscope in a single one covering the whole examined area using a technique called mosaicking. The mosaicking technique consists, as it name allows to imagine, on the creation of a "mosaic" using the video slices.

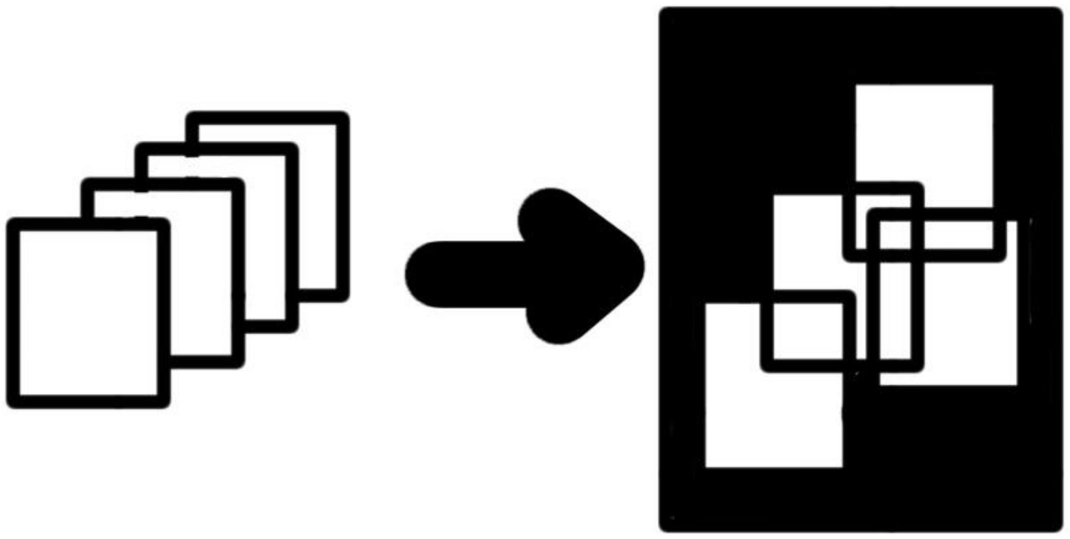


Figure 8. Drawing of the mosaicing technique.

2.4 OTHER OBJECTIVES

Although they are not the objectives that motivated this project, there are several other functions that the program should accomplish. In some cases their purpose will be to increase the quality of the resulting images and the precision of the information extracted. In other cases their accomplishing will be motivated by the necessity of those enhancements for the correct working of the program.

These functions include:

- To eliminate the background light present in all the images.
- To reduce noise due to the device and the environment.
- To increase the contrast of the image.
- To reduce light artifacts due to lack of illumination in border areas when compared to the central ones.

To conclude, it is necessary to mention the necessity of a program as fast as possible and with the simpler use that can be got. Because of that, in some cases it will be necessary to choose between an image with more details and higher quality or fidelity, or a more easily obtained one.

3. MATERIALS AND METHODS:

3.1 SOFTWARE

3.1.1 MATLAB:

Matlab is the programming language that has been chosen to develop this program. There are several languages that could have been used, but matlab showed some advantages that could be useful for this project.

Firstly, it is a program that is updated bi-annually, what means lots of updates and performance enhancement with a short interval and it has technical support from a professional organization. Also, it counts with a very complete and clear written documentation, that includes not only the basic information about commands, but examples and tutorials on how to solve certain problems, and a large user community that leads to lots of free code and information sharing.

More centered on the programing itself, matlab allows to test the code immediately and without recompilation, what is always convenient when developing a new program, as it is required to carry out many tests, but especially useful when working with images and videos, as their processing is very slow. The matlab Desktop environment is also very helpful as allows a comfortable interaction with data and files and simplifies programing.

Specifically talking about image processing, matlab offers advantages over other systems. It allows to work with both, images and video, and it can work with many different image formats, what is very important when working in medical image area. Another advantage is the fact that there are a lot of commands, functions and algorithms to process image in matlab database. Also, programming in matlab with image processing purposes is not very different of programming for other intentions, what means that it is not necessary to learn a whole new way of working with it.

Finally, it is possible to use matlab code to generate C code, giving the possibility to whom continues with this project, to use it in other languages.

In addition to these advantages, matlab is a program very well known in engineering environment, and is the one that has been more deeply studied during the degree. It is also available in all the computers of the university, and, since a short time, the license is available for all the university students.

3.1.2 IMAGEJ

ImageJ is the program that has been used to pre-process the images to be used by the program. It has mainly been used to create the phantom and to eliminate noise in the images, out of the area corresponding to the detector. It has been used also to select a portion of the whole recorded video and to convert it to tiff format.

The reasons why ImageJ has been the chosen option for achieving this, is that it is a free and open program, whose use is very easy, intuitive and fast.

3.2 ENDOSCCOPE

The fluorescence high resolution fiber bundle endoscope used to obtain the images is the one developed by Blanca Zufiria as her degree project. [6]

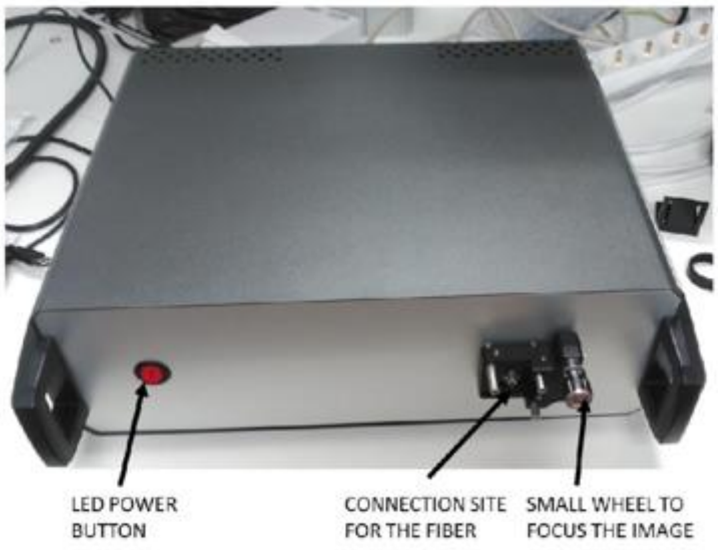


Figure 9. Fiber bundle endoscope[6]

3.3 USAF TEST TARGET

It has been used to generate a video with easily identifiable figures to test the program:

01” Positive USAF1951 Test Target.

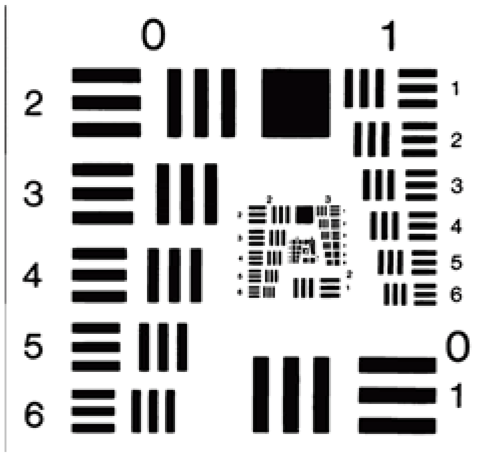


Figure 10. Detail of the target. [7]

4. SOFTWARE DEVELOPMENT

The program developed needs to accomplish three functions, in order to fulfill the three objectives of this project. That is why the program is divided in three parts, corresponding each one of them to a different function.

Part 1: discrete to continuous conversion. This section of the program is in charge of interpolating the image divided into different fibers images, and releasing a continuous unique image.

Part 2: measurement of crypts area. This piece of program develops a segmentation of the image to isolate the dark regions of the images and measure its size to determine if there is cancer existence.

Part 3: mosaicking. The last section of the program will combine several images (video frames), to create a single one that covers the whole examined area.

In order to check and explain more easily the two first functions of the program, it has been developed initially to analyze one single image. Having been used only one frame to test it.

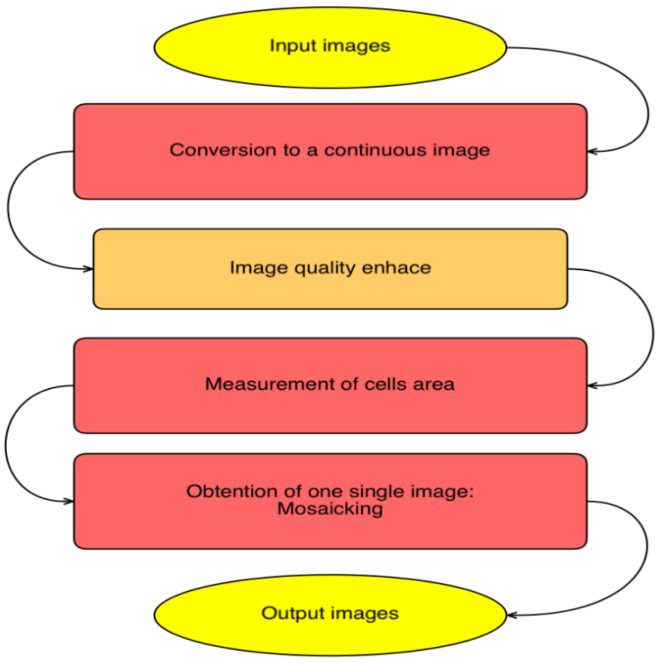

Input images
Conversion to a continuous image
Image quality enhace
Measurement of cells area
Obtention of one single image: Mosaicking
Output images

Figure 11. Diagram of the program parts.

4.1 INPUT:

In order to carry out these functions, the program requires the following inputs:

- A phantom.

-The video to analyze.

- A reference image.

4.1.1 PHANTOM

The phantom is a black and white reference image that will be used to identify which pixels of the image correspond to those recorded by each fiber of the endoscope.

This can be done because the phantom is composed of white circles corresponding to the light detected by the fibers, and black background, corresponding to those regions of the image that don't have any information, as there is no detector there.

So, the main purpose of the phantom is to locate the fibers and their limits, identifying which pixels of the obtained image correspond to fiber detected light. This can't be done just using an ordinary endoscopy image because of the non-uniformity of the "fiber corresponding regions". Endoscopy images are usually dark, with a lot of pixels having values close to the background, this makes very difficult to distinguish the circles and makes the interpolation inefficient.

Another function of the phantom is to localize damaged fibers and don't use their pixels. The fibers used to develop the endoscope are quite fragile, and damage along one of them makes it impossible for that fiber to transmit light. It is not infrequent that the fibers are damaged as they are very fragile and the endoscope use makes it susceptible of receiving hits. Also, it is necessary to take into account that this endoscope is still being developed and that it is being constantly mounted and dismounted being that a risk for the fibers.

In the endoscopy image it is difficult to determine if a black space is due to a fiber lacking or is just what is being detected by the fiber. However, in the phantom, it is easy to see where those damaged fibers are, as, if they are not damaged, they must appear.

Phantom obtaining and requirements:

In order to obtain the phantom, it is necessary to point the endoscope to a source of light, using, if possible, white light, this is done in order to obtain a perfectly white image, without any detail or illumination difference.

The phantom was obtained initially using a white surface and applying the endoscope over it. However, the differences in the illumination of the center of the detector and the extremes made impossible to detect, using a single image, all the fibers, what lead to a huge loss of information and limited enormously the field of view (as all the information in the limits of the endoscope was lost).

Once the image has been taken, it is necessary to process it to eliminate noise and to separate fiber. The lack of noise is important, as the program could interpret some noise as fibers and obtain weird values for them. The importance of processing the image to obtain one in which the circles corresponding to the fibers doesn't touch is huge, as the program detects independent elements, and, if two or more circles are in contact, they will be taken as a single object.

Finally, it is necessary to convert it to a black and white image, as the function is charge of detecting objects needs black and white images.

In our case, the phantom was obtained pointing the fibers toward a white light and then was processed using ImageJ. The background outside the detector was eliminated by cutting that region of the image and the circles were defined using contrast tool in ImageJ. Although pointing the detector to the light source reduces the illumination differences through the image, there is still some artifact because of the non-perfectly perpendicular position, being necessary to apply more than one different contrast enhance in the picture. Finally the image was converted to a black and white one using a threshold value.

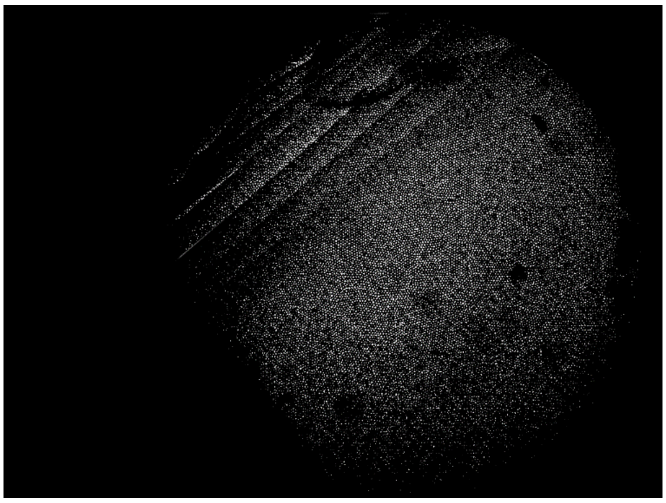


Figure 12. Generated phantom.

It is important to take into account that each time the fiber is put in the machine, the position of the fibers change. Because of that, it is necessary to obtain a new phantom each time the endoscope is used after having removed the detector.

4.1.2 ENDOSCOPY IMAGES

This is the image (or images in the case of a video) from the endoscope that is going to be analyzed by the program:

Obtaining of the regular images:

The images used in most of the project have been obtained using the endoscope over the patron image. The reason behind this is the necessity of image easy to recognize to test the quality of the results obtained. However, also images obtained in vivo will be used.

4.1.3 REFERENCE IMAGE:

The reference image will be used to reduce the illumination artifacts that appear because of differences in the light received by centered and non-centered fibers.

It is similar to the phantom, as it is an image that has not details on it, but is different as it present differences in illumination between regions.

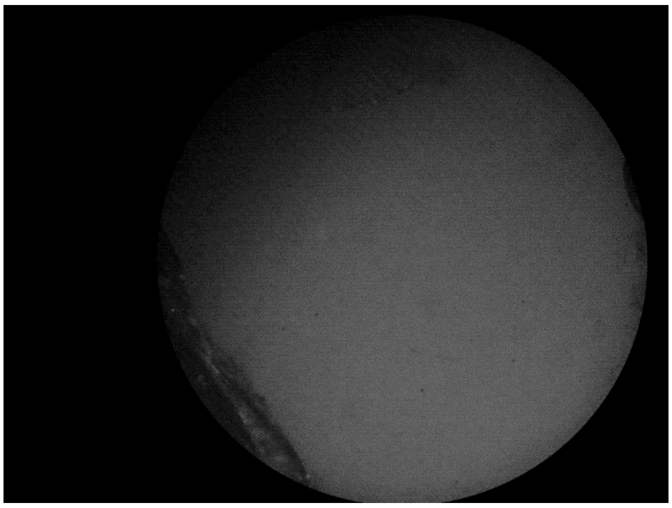


Figure 13. Reference image.

This image will allow seeing where in the image the illumination is better and take that into account to correct the analyzed images.

<u>Obtaining of the reference image:</u>

In order to extract this image, the endoscope was used over a uniformly white not rough surface. So, details are not obtained, but illumination artifacts still appear.

4.2 DISCRETE TO CONTINUOUS

The main purpose of this section of the program is to convert images in which regions corresponding to spaces in the detector, and so, lacking of any useful information, are present, to continuous images where those regions don’t appear.

In order to do that, the program combines several functions:

- A function that locates fibers and determines the pixels corresponding to them. Giving then, a single value to each one of the fibers, whose location is the center of the fiber, generating a grid.
- A function that uses the values obtained with the previous function application to obtain the values of all the pixels of the image by interpolation.
- Filters to make image smother.

After reading the images, the program process to the obtaining of the continuous image, to achieve that is the function *image_processing*.

To obtain a continuous image it is only necessary to apply this process to the image obtained from the endoscope. However, in order to reduce the illumination artifact, a reference image will be required, in order to work with the continuous images, it is necessary to first convert the reference image too. That is why the whole process is applied to both images.[7]

4.2.1 IMAGE_PROCESSING:

image_processing is the function in charge of converting the image from a discontinuous one to a continuous one.

It takes as inputs the image needed to convert and the phantom, and follows this path:

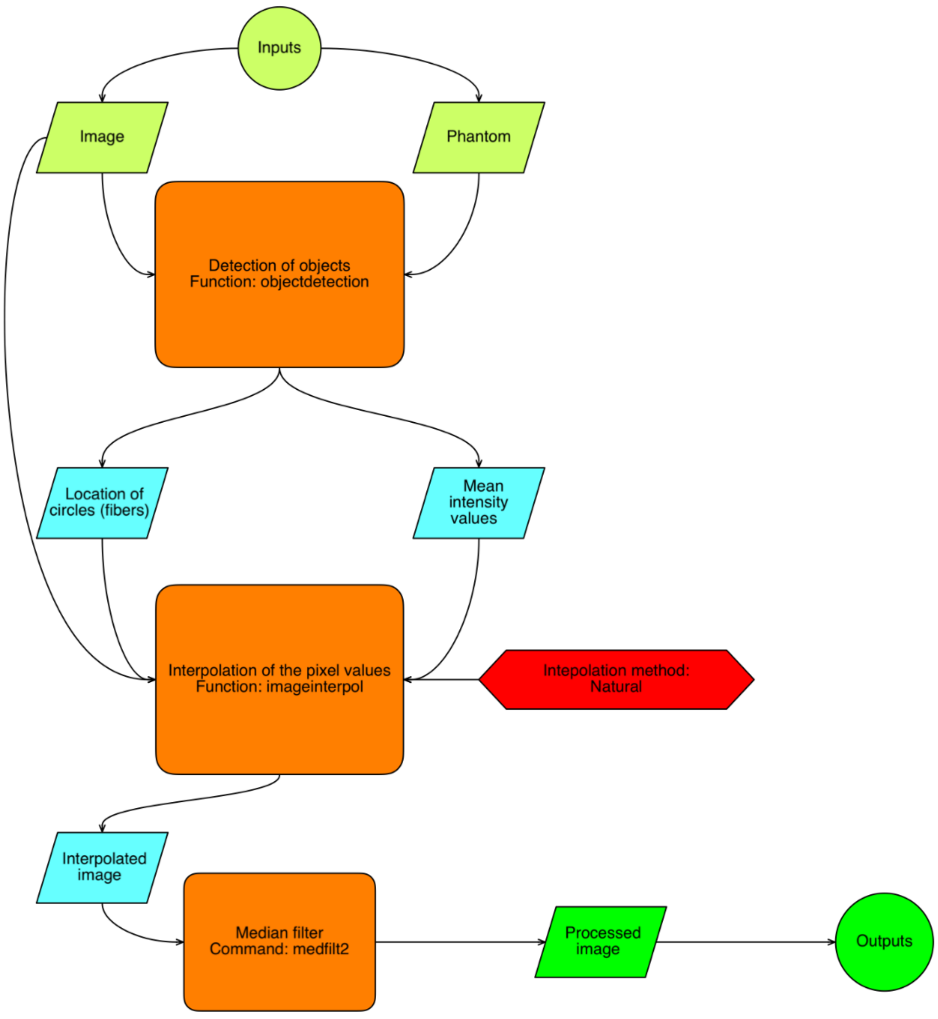


Figure 14. Diagram of image procesing, function in charge of image conversion.

4.2.1.1 OBJECTDETECTION:

The function in charge of distinguishing what is detected by each fiber.

The aim of this function is to use the phantom as a template to identify the pixels corresponding to the image obtained by each one of the fibers of the detector, and, so, being able to establish the mean value of each fiber pixels, giving a single value to each one of them in order to create a grid that can be used later to interpolate a continuous image. It may seem that simplifying the image to one pixel per fiber will reduce the information too much, but the number of fibers is so high, and the number of pixels corresponding to each one is so relatively low, that the information lost is minimum.

The function takes as inputs, the image that is being processed by the program and the phantom. It needs the phantom as a model to identify the fibers, and the image to extract the values corresponding to those pixels, in order to obtain the values for the interpolation

In *objectdetector*, there are two functions: *bwconncomp* and *labelmatrix*.

Bwconncomp is a matlab function that finds the connected elements in a binary image and releases a structure with the information about those elements. Most of this information is not needed; we are interested mainly in the location of these elements.[10]

Labelmatrix is another matlab function that generates a labeled matrix in which elements corresponding to background have values equal to 0, and pixels corresponding to an object, a numerical value depending on that object. Thus, pixels belonging to object 1 will have value 1, those corresponding to object 2 will have valu2, etc.[10]

After these two functions, a structure containing the values of the pixels of each object in the endoscopy image and a matrix containing the positions of those pixels have been obtained.

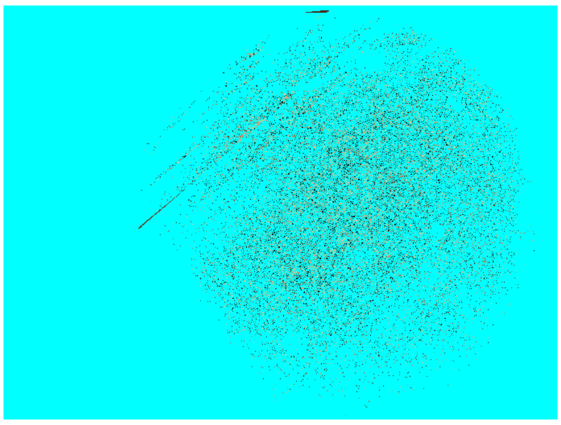


Figure 15. Display of the labeled matrix.

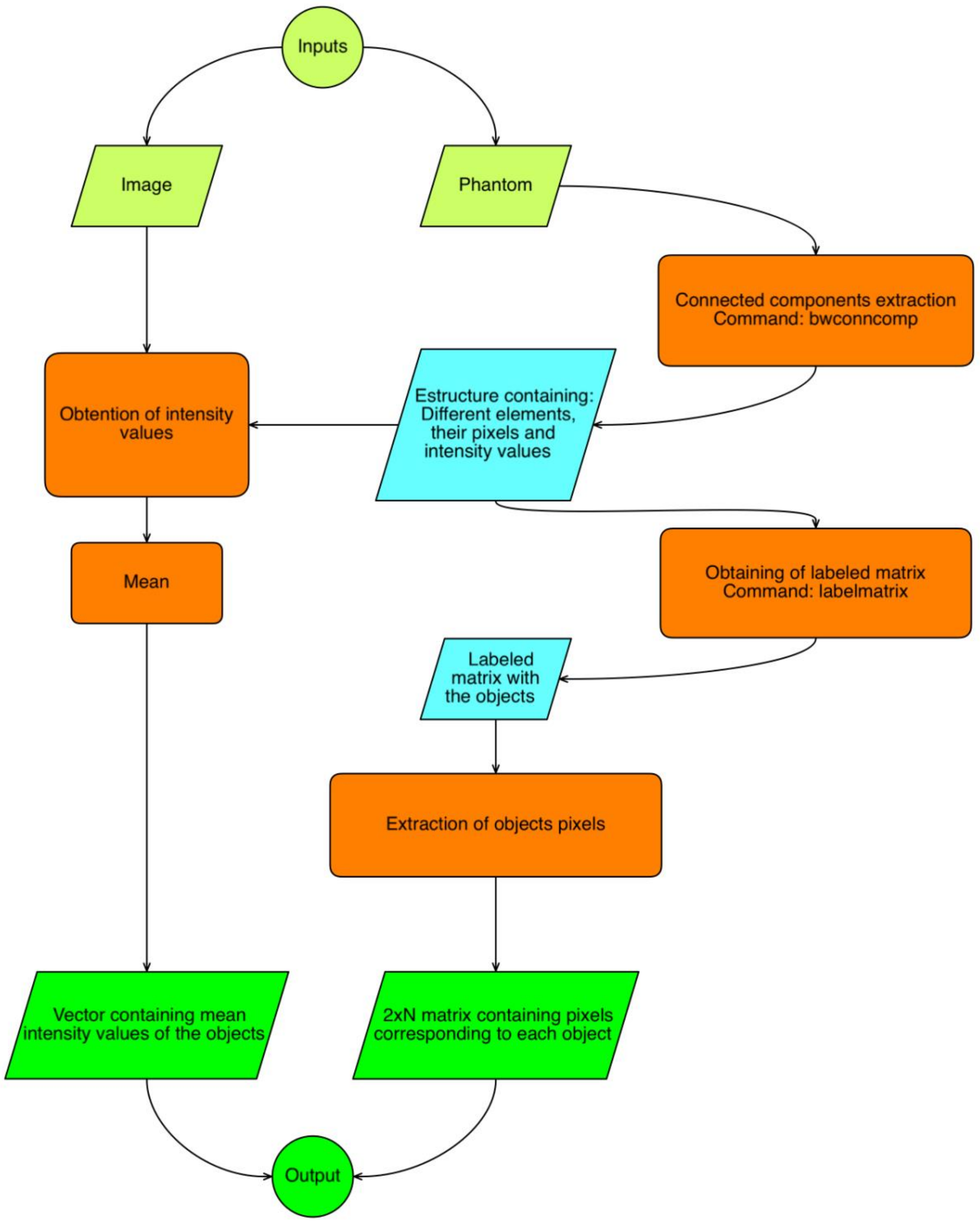
Now, for each object, a *for cycle*:

-Extracts the intensity values of the object pixels from the structure and computes the mean value.

- Extracts the position of those pixels and computes the weighted center

-The values of the objects location are recorded in a 2xN matrix, where N is the number of objects detected, and the mean intensity values are recorded in a vector of length N.

The outputs of this function are the matrix and the vector containing the information about the objects.



Inputs
Image
Phantom
Connected components extraction Command: bwconncomp
Obtention of intensity values
Estructure containing: Different elements, their pixels and intensity values
Mean
Obtaining of labeled matrix Command: labelmatrix
Labeled matrix with the objects
Extraction of objects pixels
Vector containing mean intensity values of the objects
2xN matrix containing pixels corresponding to each object
Output

Figure 16. Flow diagram of the function objectdetector.

4.2.1.2 IMAGEINTERPOL:

Image Interpol is a function that takes as inputs the image, the matrix and the vector generated by *objectdetector* and an strain that determines the method of interpolation.

The image id required to extract its size, and so, use the meshgrid command to generate a grid of points of its size. Once this is done, the command griddata, uses those points in the grid, as well as the other inputs of the function, to develop an interpolation following the method specified.[10]

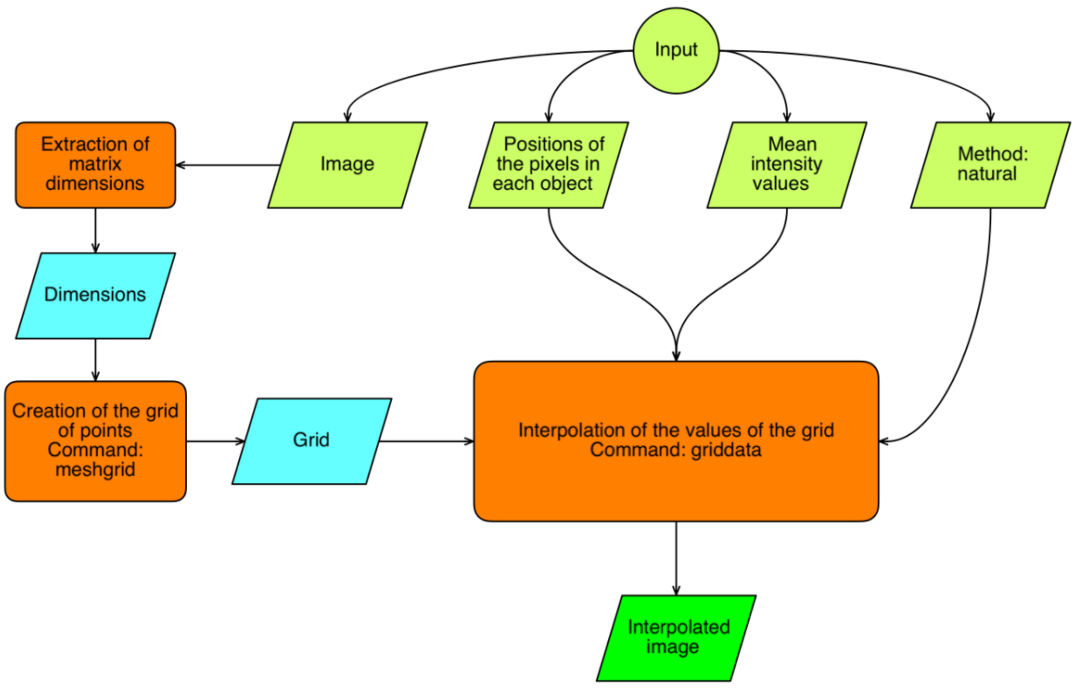

Input
Extraction of matrix dimensions
Image
Positions of the pixels in each object
Mean intensity values
Method: natural
Dimensions
Creation of the grid of points Command: meshgrid
Grid
Interpolation of the values of the grid Command: griddata
Interpolated image

Figure 17. Flow diagram of the function imageinterpol.

<u>Interpolation method selection:</u>

Command griddata offers five methods of interpolation depending on the strain specified:

- ‘nearest’: nearest neighbor interpolation.
- ‘linear’: linear interpolation.
- ‘natural’: natural neighbor interpolation.
- ‘cubic’: cubic interpolation.

- ‘v4’: biharmonic spline interpolation.

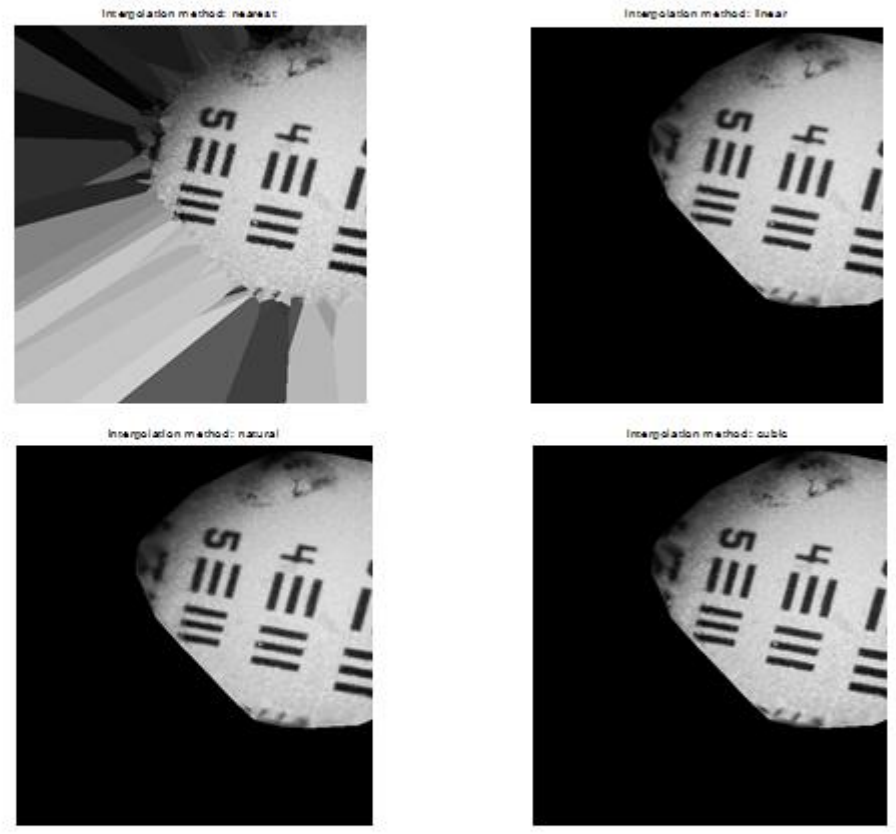

Interpolation method: nearest
Interpolation method: linear
Interpolation method: natural
Interpolation method: cubic

Figure 18. Comparison between different interpolation methods.

In order to decide which method should be used, all of them were checked in order to decide which one of them were more efficient and had the best equilibrium between quality and velocity for this experiment:

Finally, the method chosen was the natural one, that is kind of a middle point between linear and nearest neighbors and had better quality than both of them without spending more time.[10]

The cubic interpolation was discarded because it took too much time to carry it out, and the biharmonic spline interpolation was too complex to be achieved by the computer without crashing.

4.2.1.3 FILTERING STEP

After obtaining the continuous image, it is filtered in order to eliminate noise generated by the interpolation using medfilt2 that applies the median filter to the whole image.

After checking the effect of different filters with a wider or thinner surface, the selected was the median filter applied over a 5x5 matrix.[10]

4.2 MEASUREMENT OF THE CRYPTSAREA

The second part of the program is aimed to measure the size of the intestine crypts, that appear in the obtained images as dark areas.

In order to do this, this section requires as inputs the image obtained from the previous step, already processed, and a reference image, that should have been processed too.

The process includes four steps:

Firstly, the image is compared with the reference image, this is due to determine where the illumination is higher and where lower, avoiding so to wrongly consider dark regions of the picture due to the endoscope, as signals of the presence of crypts.

Secondly, the image is processed using LBP applied to analyze the rotation invariant local variance, and so, classify textures and obtain the edges of the cells. Once this is done, the gaps in the edges are completed.[9][10][11]

Then, the areas contained in closed lines are filled, so a black background with several white areas is obtained, this areas are the dark regions corresponding to cells.

Finally, as the reduction of the illumination artifact generates a lot of noise, what leads to the appearance of lots of little white objects on the borders of the image. In order to eliminate them, a function based on the one that detected the objects in the first part is applied.

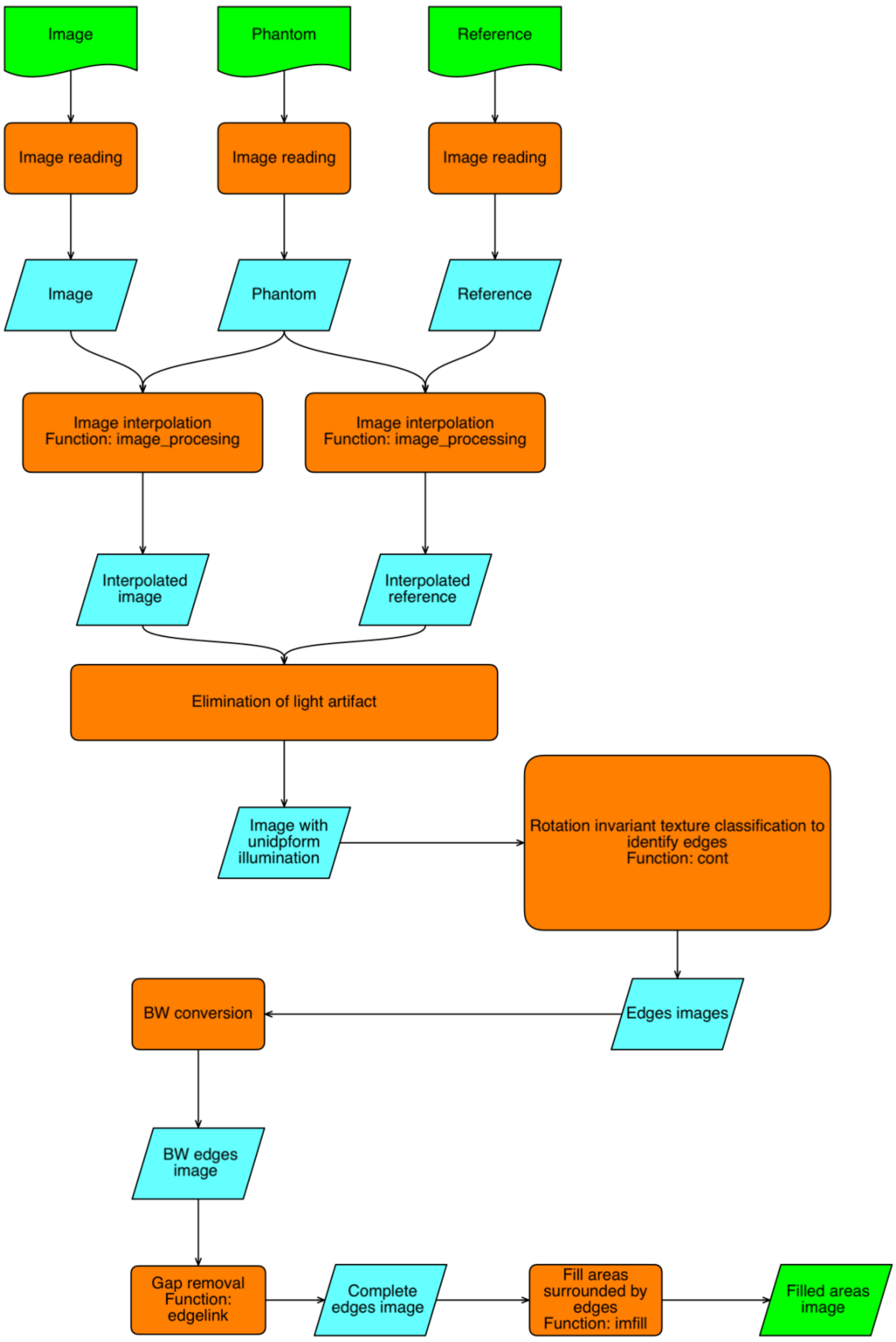


Figure 19. Flow diagram of the second part of the program, aimed to identify the areas corresponding to intestine cells.

4.2.1 ILLUMINATION ARTIFACT ELIMINATION

During this piece of the code, the images are compared with the reference, dividing each pixel value between the corresponding pixel value in the reference image. As the image are in the format double, the regions that have low illumination in the reference will lead to an increase in the values of the endoscopy mage, while regions well illuminated in the reference won't experiment a big change. However, it is important that the reference suffers the same processing that the image that is going to be analyzed.

Although this works very well with the central regions, in the periferical pixels can appear a lot of noise, due to the presence of values equal or very close to zero, that, when dividing the image values, lead to very high or even infinite result.

4.2.2 DETECTION OF THE LIMITS OF THE CryptS

Once the illumination artifact has been eliminated, it is possible to start with the detection of the crypts surface.

In order to do that *cont* function is used. This function carry out a rotation invariant texture classification based on local binary pattern, to obtain the edges of the crypts shapes.

When an image containing only black background and the limits of the areas has been obtained, another function is used in order to complete any gap that those edges could have. This is important because then, the edges are going to be filled with a program that covers the surface inside closed curves.

4.2.3 FILLING THE AREAS

The function *imfill* is used to convert pixels contained in closed curves to 1. It is necessary to have a black and white image, that is why all the pixels with a value higher than background have been set to 1, and the background to 0 (logical image).

The resulting image will contain several white areas corresponding to crypts surfaces and noise.

4.2.4 NOISE ELIMINATION

In order to eliminate the noise that will appear as several small areas in the border of the detector image the function *objectselection* will be used. This function uses the same commands as *objectdetection* to detect the objects, but then, it eliminates all those objects with an area below a certain value. Using this, the small objects corresponding to noise, will be eliminated.

4.4 MOSAICING

In this section of the program, the images obtained with the endoscope are combined to create a bigger image, which includes the whole area covered by the video.

The mosaicing is mainly based on image registration. Image registration consists on the transformation of an image, so it is correctly positioned respect to another one. In this case, the image registration will lead to moving one video frame respect to another one so that the common field of view coincides. When applying this to all the video slices, a great image will be obtained.

The image registration can be based on intensity values or in feature recognition.

In the case of intensity based image registration, one image is moved respect to the other one and the cost function is evaluated. This cost function can vary, and the most convenient cost function will depend on the application.

In this project, the image registration that is going to be applied is a simple intensity based image registration. The program takes two images and computes the mutual information of the images applying several rigid body transformations to one of the images. The reason why we consider this case, a rigid body one (only rotation and translation), is that the endoscope is always going to be in contact with the intestine wall, so there won't be any change in the scaling of the image. Although rigid body can include both translation and rotation, we will only apply translation transformation to the image, as the endoscope is not expected to rotate during the recoding of the images.[8]

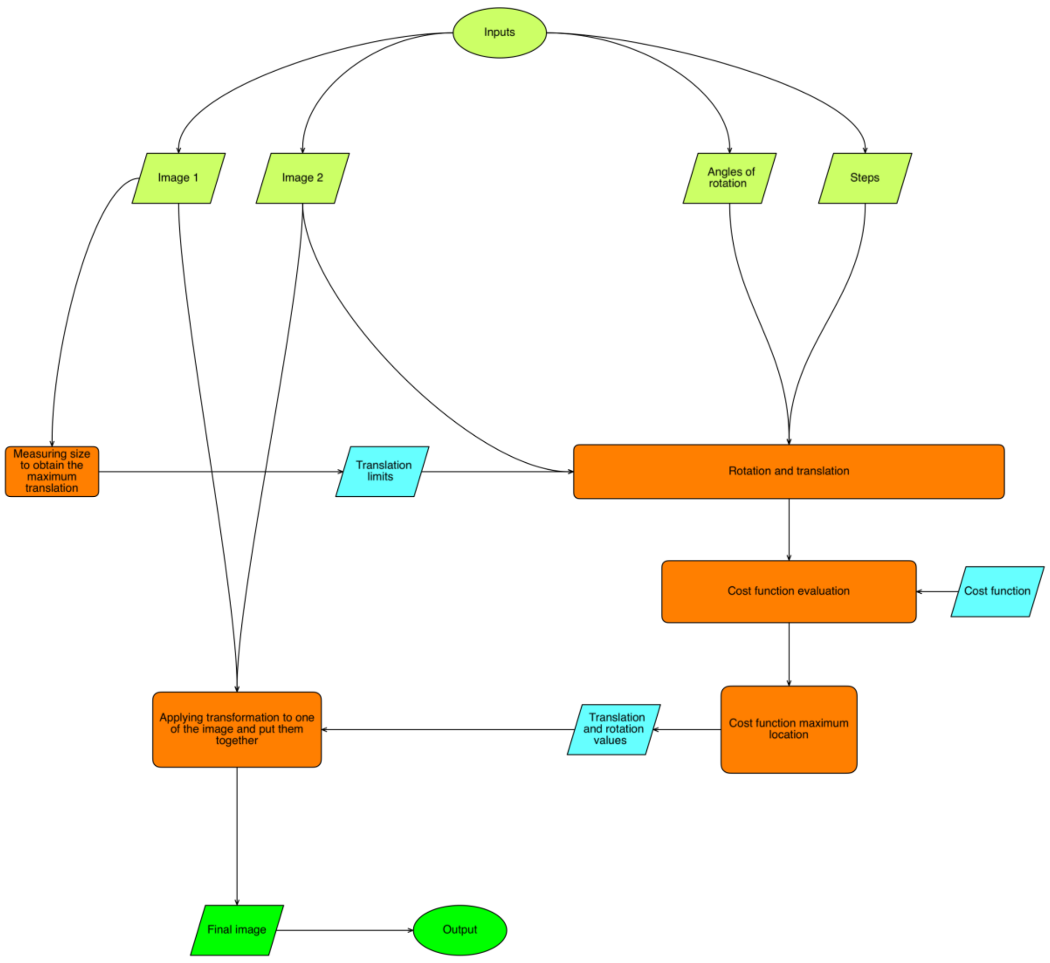

Inputs
Image 1
Image 2
Angles of rotation
Steps
Measuring size to obtain the maximum translation
Translation limits
Rotation and translation
Cost function evaluation
Cost function
Applying transformation to one of the image and put them together
Translation and rotation values
Cost function maximum location
Final image
Output

Figure 20. Flow diagram of the registration function.

This is not the complete mosaicing technique, just the registration. It is still necessary to solve the problem of non-coincidence between values of the intersection pixels in both images. In order to solve it, it could be used the average value of the two images.

5. OBTAINED IMAGES AND DISCUSSION

5.1 OBTENTION OF THE CONTINUOUS IMAGES

The first part of the program was the one in charge of identifying continuous elements corresponding to different fibers, and interpolating them to obtain a continuous image.

In order to test if it works, two images are used.

- An image obtained by the endoscope when using a template.
- An image obtained when using the endoscope in a rat intestine.

5.1.1 TEMPLATE IMAGE

Here, it can be seen the image obtained with the template, before being processed:

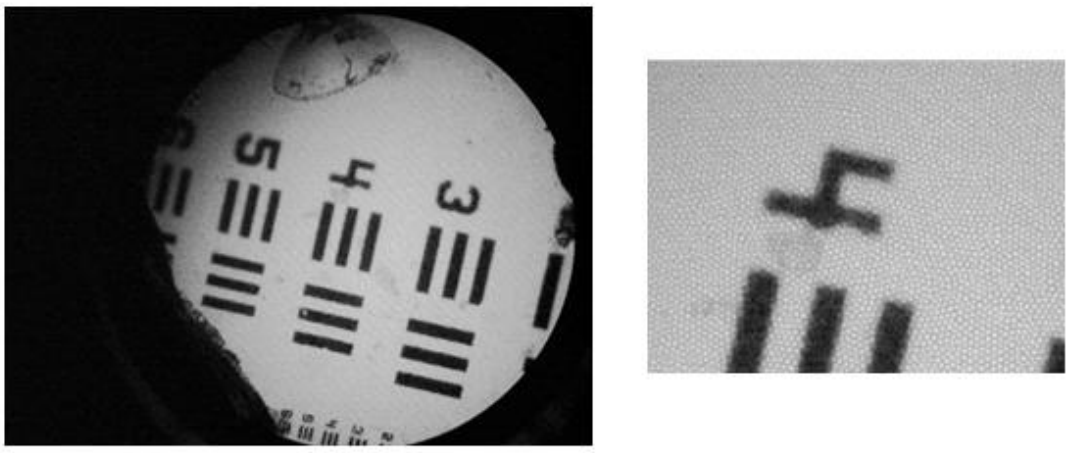


Figure 21. Target image acquired with the endoscope and detail.

This image is just a photogram of the video recorded by the endoscope, and, unlike the phantom, it is not precise to use imageJ to pre-process it before opening it with matlab. The image outside the limits of the big circle doesn't correspond to anything recorded by the detector. The white circumferences concentric to the detector shape are due to light that scape from the fiber. In the detail, it can be appreciated the discontinuity due to the fiber bundle.

In the following figure, the images obtained after running the function *image_processing*, can be seen:

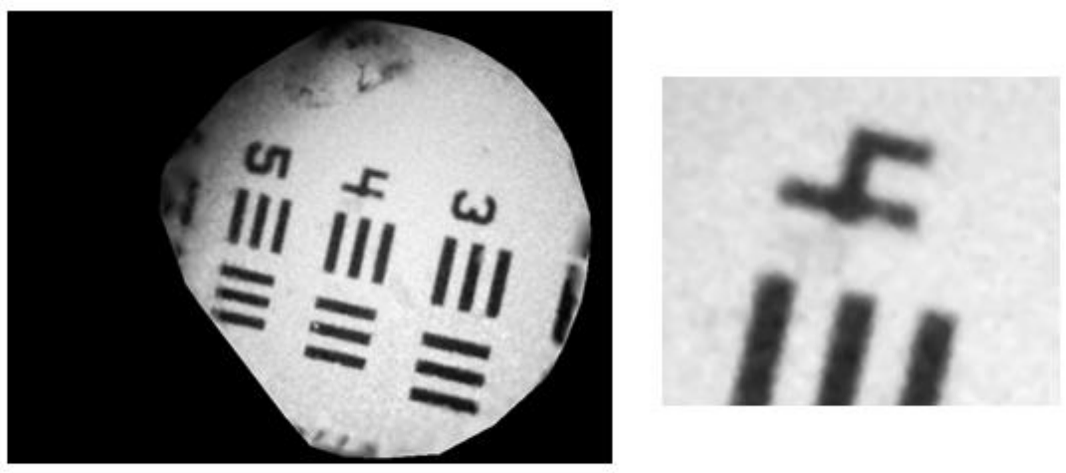


Figure 22. Interpolated image and detail.

The detail shows how the image obtained is continuous. There is noise, but the spaces between each fiber image have disappeared. This, will allow to work with the images, better, as, if spaces remained, trying to analyze the intestine cells, would have been more difficult.

5.1.2 IN VIVO ENDOSCOPY IMAGE

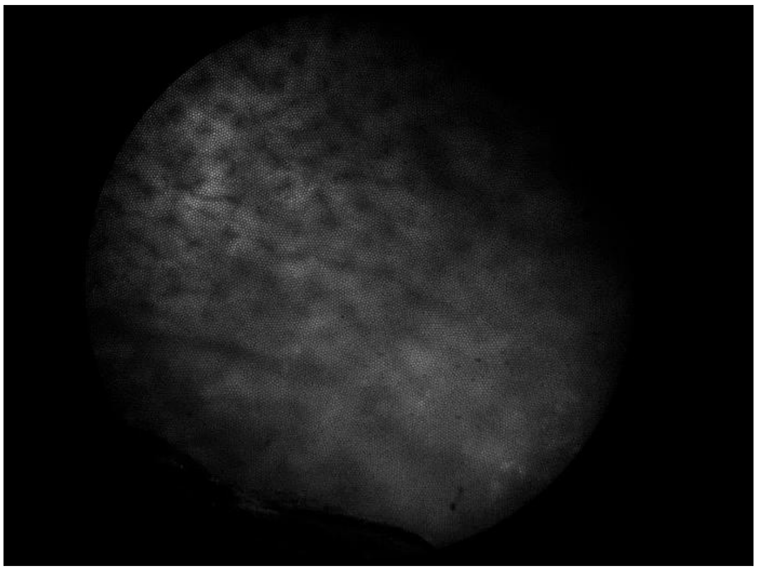


Figure 23. In vivo endoscopy obtained by Blanca Zufiria

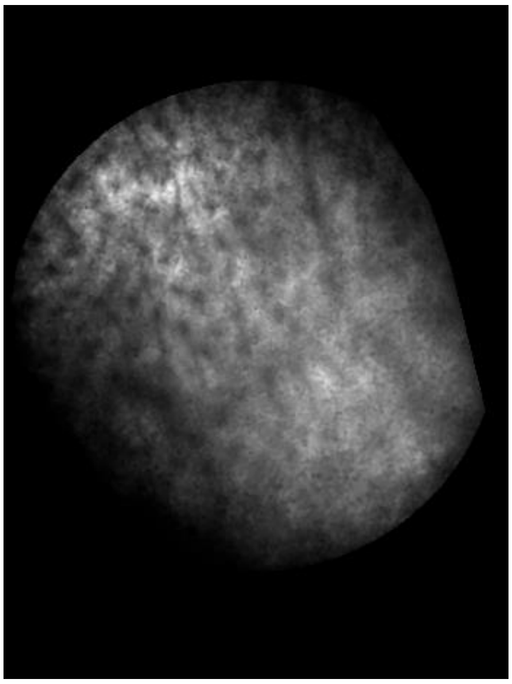


Figure 24. Interpolated in vivo image.

5.2 IDENTIFICATION OF DARK AREAS

As it was mentioned before, in order to correctly identify the areas corresponding to crypts, it is necessary to eliminate the illumination differences in the images. In order to do that, the reference image is taken and processed using exactly the same steps as the endoscopy images.

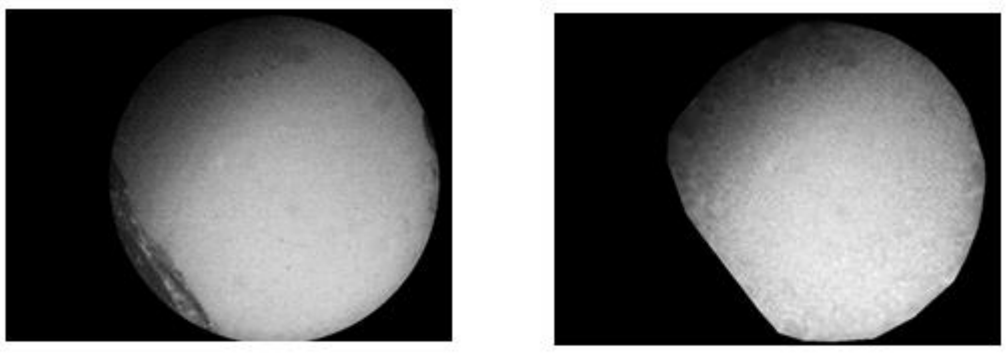


Figure 25. Reference image before and after the interpolation.

Now that there are no gaps in any of the images, the reference image can be used to homogenize the illumination of the image to analyze. The resulting image won't have illumination differences most of the image although it will have a lot of noise in the periferic area, This is because in the center regions of the phantom, most of the fibers appear and have several pixels, but in the side, some of those fibers only have a pixel or even disappear, what leads to more variation in the surrounding regions pixels from one image to another due to noise or slight differences. Because of this, there are pixels there where the values between the two images are very high, what leads to a lot of noise.

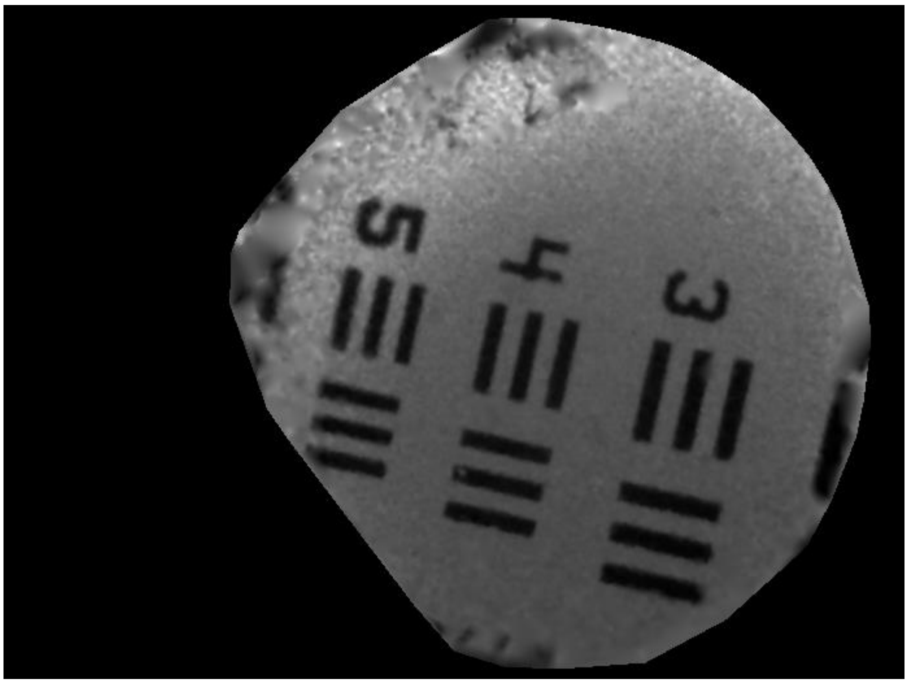


Figure 26. Illumination artifact eliminated using reference image.

Once, this previous steps have been done, the image is processed to identify darker regions.

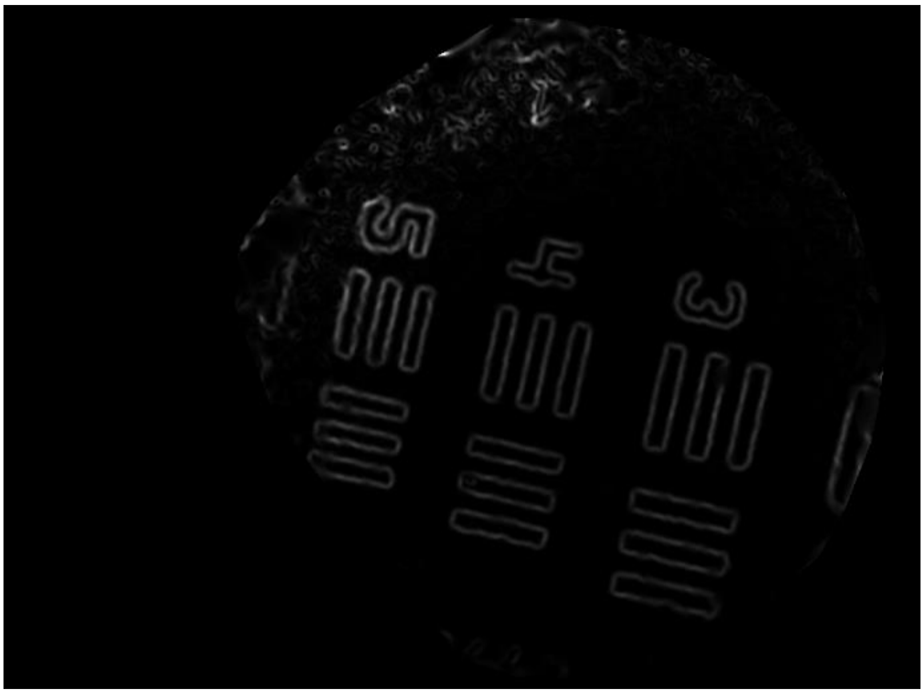


Figure 27. Edges of the regions with differenc texture.

Then, the edges are completed to ensure that all the areas appear in the final image, and the image is converted into a black and white one, being then filled:

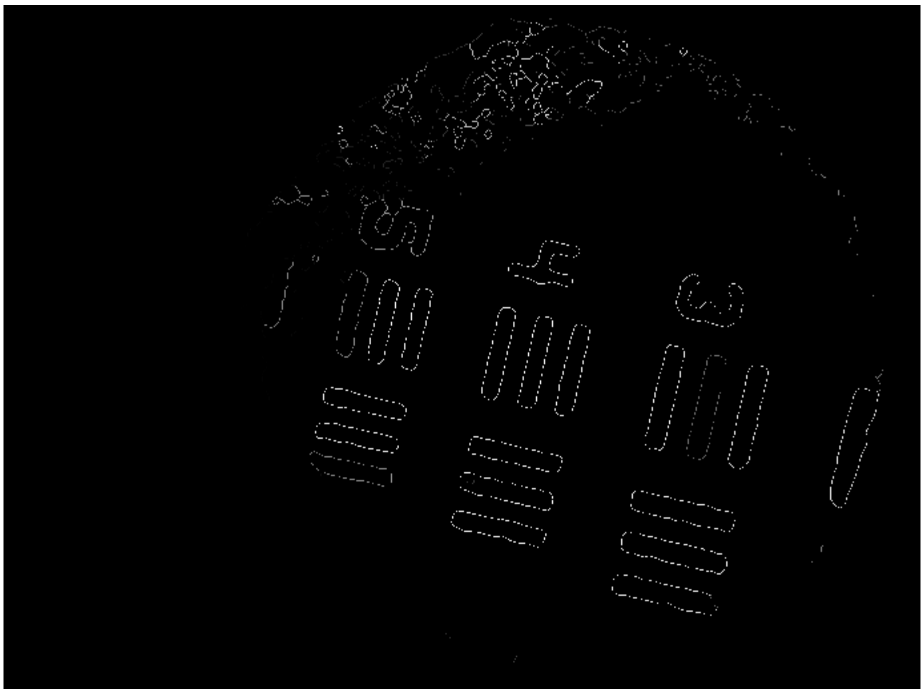


Figure 28. Gaps in the edges completed.

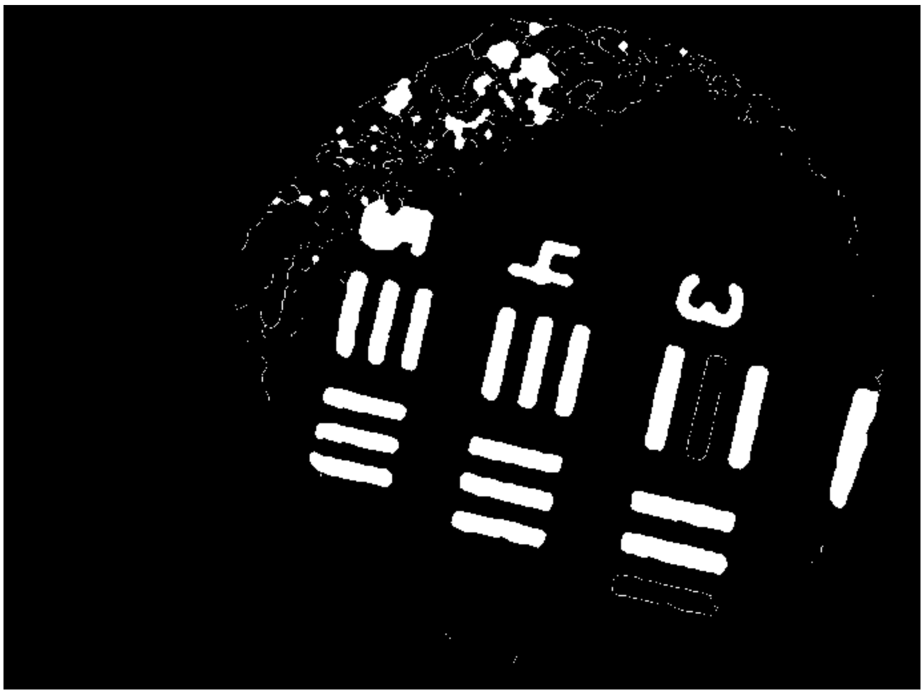


Figure 29. Edges filled.

The problem with this image is that the noise in the outer zone of the detector appears. In order to make it disappear, the function that eliminates small connected components is used. However, in order to avoid the program to consider all the noise as a single objects, it is necessary to filter the image and so, eliminate the lines between noise elements.

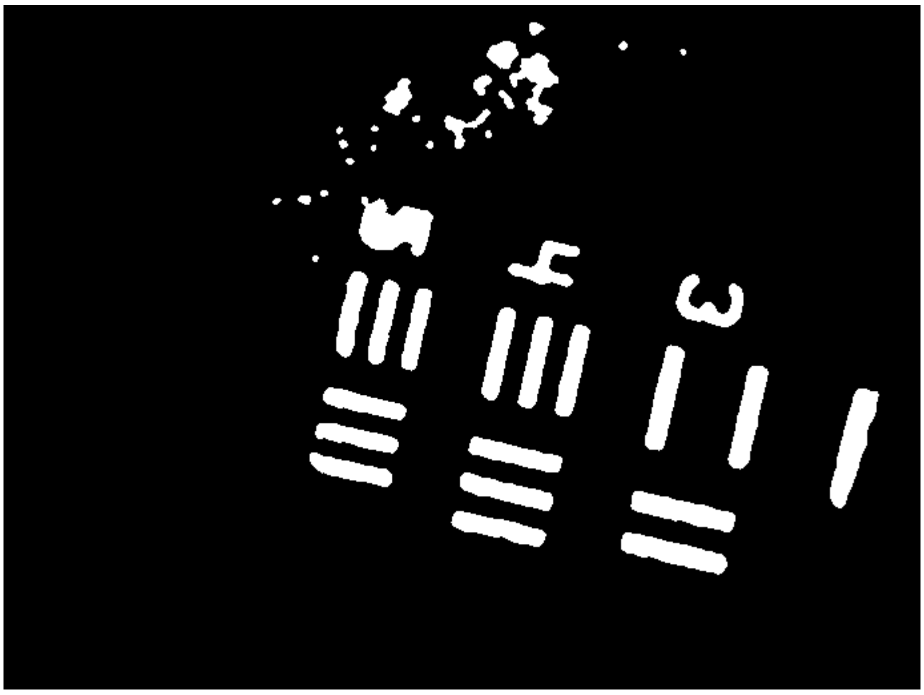


Figure 30. Filtered image of filled edges.

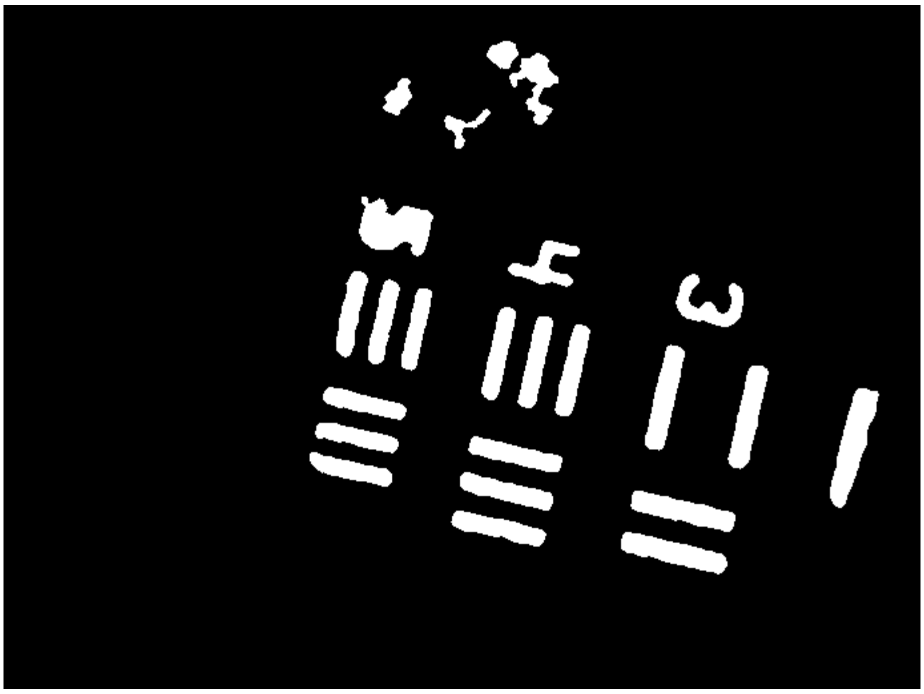


Figure 31. Small object eliminated.

There is still some noise in the image, but it is just a small part of it. Now, it is possible to sum all the white pixels in the image and obtain a single value per image. Computing the mean of all the images in the video, it is possible to obtain a single numerical number that gives an idea of the surface of the crypts in the intestine. If compared with values of sick and healthy intestines, it would be possible for the program to determine the presence or not of sickness.

5.3 MOSAICING

In this last part of the program, the images from the video are joined together to form a bigger one. In order to test this piece of the code, a histology image has been used to simulate a video:

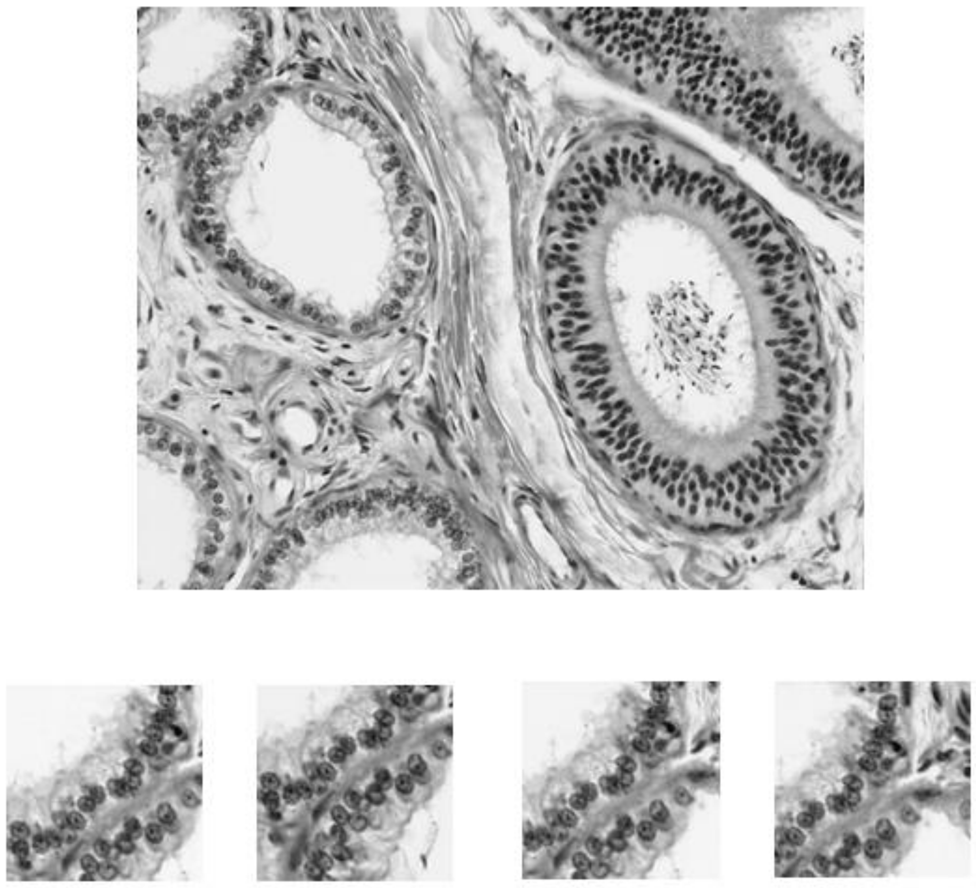


Figure 32. Video simulation using a histology image.

Using this simulated video, the program works, although is necessary to take into account that this “video” fits perfectly because, as it is just asset of cut images, the pixels are exactly the same. Below, it can be seen the cost function represented, what allows to see this perfect match.

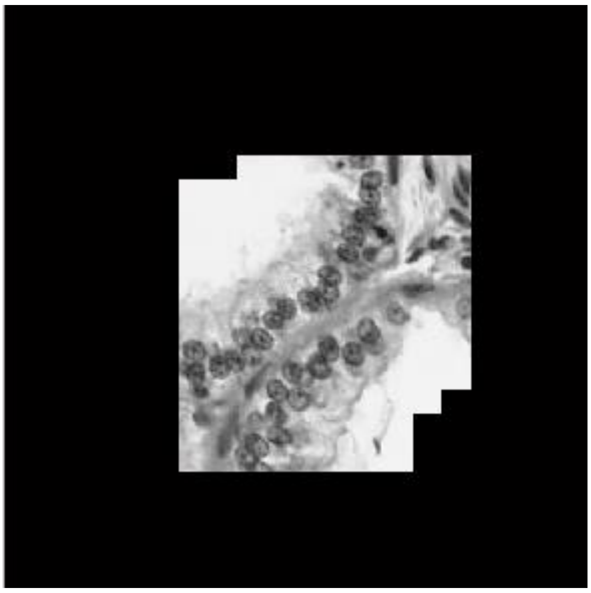


Figure 33. Resulting image after the registration.

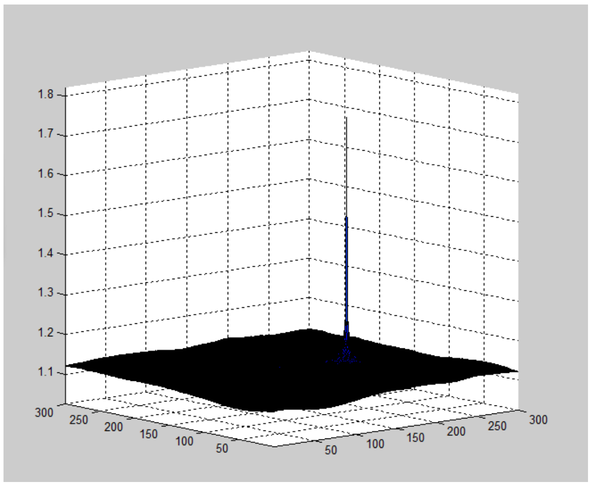


Figure 34. Cost function of two images when being moved to find its position.

6. FUTURE WORK

This project didn’t expected to produce a complete program that could be used directly with the endoscope, and, even in the objectives already achieved, some improvements could be made.

One possible improvement could be the design of a program that automatically, prepares the phantom to be used. It could still use ImageJ, but controlled with matlab, so it doesn’t require the intervention of the user.

Images of healthy and sick subjects should be used to train the program and so, allow it to automatically distinguish between healthy intestines and sick ones.

The mosaicing section could be improved, enhancing the function to be more accurate and faster.

In general, the program should be much faster, as it is thought to be used while obtaining the images.

7. SOCIOECONOMIC IMPACT

Since its appearance, the endoscopy has improved medical care allowing to access non-invasively to areas of the body that could only be reached with surgery. It has given also the possibility of obtaining real-time images, what has made possible diagnostics that were not.

The use of fluorescent light has also leaded to an improvement in the diagnostic respect to the visible light endoscopy.

Finally, the digital image processing developed during this project, should lead to a software that allows to obtain a diagnostic for cancer without the intervention of a physician. And, the development of the mosaicing technique will allow having a clearer view of the patient intestine.

8. BUDGET

In these tables, costs of the project necessary equipment and personnel are presented. It is not a very expensive project, as the only equipment needed is a computer and a matlab license, and not many personnel has been involved.

Technical equipment	Quantity	Cost/Units	Depreciation/Month	Months Employed	Total Cost
personal computer	1	700 €	10 €	6	60 €
university computer	1		10 €	6	60 €
matlab (software)	1	6.000 €	100 €	6	600 €
					720 €

Human resources	Months	Cost/Month	Total
Student	6	750 €	4.500 €
Tutor	6	1.200 €	7.200 €
			11.700 €

9. ANNEX

9.1 PROGRAM TO OBTAIN A CONTINUOUS IMAGE AND IDENTIFY AREAS

```
for i = 1:10
video(:,:,i) = imread('USAF target.tif',i);
end;

phantom = imread('phantom.tif');
reference = imread('reference.tif');

reference_pro = image_processing(reference,phantom);
reference_pro = medfilt2(reference_pro,[5 5]);

for j = 1:10
image_pro = image_processing(video(:,:,j),phantom);
image_pro = medfilt2(image_pro,[5 5]);
image_final = (image_pro)./(reference_pro+1);

c = cont(image_final,5,20);
[h,x] = hist(c(c>0));
cn = c>min(x);
[edgelist, cp, etype] = edgelink(cn);

cf = imfill(cp>0,'holes');
cfilt = medfilt2(cf,[5 5]);
bigareas = objectselection(cfilt,1000);

figure;
imshow(bigareas,[])

end;
```

9.1.1 IMAGE_PROCESSING

```
function  [image_processed] = image_processing(image,phantom)



[wc,meanint] = objectdetection(phantom,image);

imagefilt = medfilt2(image, [3 3]);

method = 'natural'

[image_int] = imageinterpol(imagefilt,wc,meanint,method);

image_processed = medfilt2(image_int, [3 3]);
```

9.1.1.1 OBJECTDETECTION

```
function  [weighted_centre,mean_int] = objectdetection(phantom,imagen)

phantom = double(phantom);
imagen = double(imagen);

BW = phantom;
cc = bwconncomp(BW);
 %Connectivity
 %ImageSize
 %NumObjects
 %PixelIdxList

labeled = labelmatrix(cc);

RGB_label = label2rgb(labeled,@copper,'c','shuffle');
```

```
cc.NumObjects

for i = 1:cc.NumObjects

 int = imagen(cc.PixelIdxList{i});
 [x,y] = find(labeled==i);

 mean_int(i) = mean(int);
 if mean_int(i) == 0

     weighted_centre(1,i) = mean(x);
     weighted_centre(2,i) = mean(y);



 else

     weighted_centre(1,i) = sum(int.*x)/sum(int);
     weighted_centre(2,i) = sum(int.*y)/sum(int);



 end;

end;
```

9.1.1.2 IMAGE INTERPOL

```
function                                [imagen_interpolada]              =
imageinterpol(imagen,wc,meanint,method)

tam = size(imagen);
[yq,xq] = meshgrid(1:1:tam(2), 1:1:tam(1));
imagen_interpolada   =   griddata(wc(1,:),   wc(2,:),   meanint,   xq,   yq,
method);
```

9.1.2 CONT

```
%C computes the VAR descriptor.
% J = CONT(I,R,N,LIMS,MODE) returns either a rotation invariant local
% variance (VAR) image or a VAR histogram of the image I. The VAR
values
% are determined for all pixels having neighborhood defined by the
input
% arguments. The VAR operator calculates variance on a circumference
of
% R radius circle. The circumference is discretized into N equally
spaced
% sample points. Function returns descriptor values in a continuous
form or
% in a discrete from if the quantization limits are defined in the
argument
% LIMS.
%
% Examples
% ---------
%
%       im = imread('rice.png');
%       c  = cont(im,4,16);
%       d  = cont(im,4,16,1:500:2000);
%
%       figure
%       subplot(121),imshow(c,[]), title('VAR image')
%       subplot(122),imshow(d,[]), title('Quantized VAR image')




function result = cont(varargin)
% Version: 0.1.0



% Check number of input arguments.
error(nargchk(1,5,nargin));

image=varargin{1};
```

```
d_image=double(image);

if nargin==1
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
    neighbors=8;
    lims=0;
    mode='i';
end



if (nargin > 2) && (length(varargin{2}) == 1)
    radius=varargin{2};
    neighbors=varargin{3};
    spoints=zeros(neighbors,2);
    lims=0;
    mode='i';
    % Angle step.
    a = 2*pi/neighbors;

    for i = 1:neighbors
        spoints(i,1) = -radius*sin((i-1)*a);
        spoints(i,2) = radius*cos((i-1)*a);
    end

    if(nargin >= 4 && ~ischar(varargin{4}))
        lims=varargin{4};
    end

    if(nargin >= 4 && ischar(varargin{4}))
        mode=varargin{4};
    end

    if(nargin == 5)
        mode=varargin{5};
    end
end

if (nargin == 2) && ischar(varargin{2})
    mode=varargin{2};
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
    neighbors=8;
    lims=0;
end





% Determine the dimensions of the input image.
[ysize xsize] = size(image);

miny=min(spoints(:,1));
maxy=max(spoints(:,1));
minx=min(spoints(:,2));
maxx=max(spoints(:,2));

% Block size, each LBP code is computed within a block of size
bsizey*bsizex
bsizey=ceil(max(maxy,0))-floor(min(miny,0))+1;
bsizex=ceil(max(maxx,0))-floor(min(minx,0))+1;

% Coordinates of origin (0,0) in the block
origy=1-floor(min(miny,0));
origx=1-floor(min(minx,0));

% Minimum allowed size for the input image depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
    error('Too small input image. Should be at least (2*radius+1) x
(2*radius+1)');
end

% Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;

%Compute the local contrast

for i = 1:neighbors
```

```
    y = spoints(i,1)+origy;
    x = spoints(i,2)+origx;
    % Calculate floors and ceils for the x and y.
    fy = floor(y); cy = ceil(y);
    fx = floor(x); cx = ceil(x);

    % Use double type images
    ty = y - fy;
    tx = x - fx;

    % Calculate the interpolation weights.
    w1 = (1 - tx) * (1 - ty);
    w2 =      tx  * (1 - ty);
    w3 = (1 - tx) *      ty ;
    w4 =      tx  *      ty ;
    % Compute interpolated pixel values
    N  = w1*d_image(fy:fy+dy,fx:fx+dx)  + w2*d_image(fy:fy+dy,cx:cx+dx)
+ ...
        w3*d_image(cy:cy+dy,fx:fx+dx) + w4*d_image(cy:cy+dy,cx:cx+dx);
    % Compute the variance using on-line algorithm
    %                                                                   (
http://en.wikipedia.org/wiki/Algorithms_for_calculating_variance#On-
line_algorithm ).
    if i == 1
        MEAN=zeros(size(N));
        DELTA=zeros(size(N));
        M2=zeros(size(N));
    end
    DELTA=N-MEAN;
    MEAN=MEAN+DELTA/i;
    M2=M2+DELTA.*(N-MEAN);

end

% Compute the variance matrix.
% Optional estimate for variance:
% VARIANCE_n=M2/neighbors;
result=M2/(neighbors-1);
```

```
% Quantize if LIMS is given
if lims
    [q r s]=size(result);
    quant_vector=q_(result(:),lims);
    result=reshape(quant_vector,q,r,s);
    if strcmp(mode,'h')
        % Return histogram
        result=hist(result, length(lims)-1);
    end

end

if strcmp(mode,'h') && ~lims
    % Return histogram
    %epoint = round(max(result(:)));
    result=hist(result(:),0:1:1e4);
end

end

function indx = q_(sig,partition)

[nRows, nCols] = size(sig);
indx = zeros(nRows, nCols);

for i = 1 : length(partition)
    indx = indx + (sig > partition(i));
end

end
```

9.1.3. EDGELINK

```
% EDGELINK - Link edge points in an image into lists
%
% Usage: [edgelist edgeim, etypr] = edgelink(im, minlength, location)
```

```
%
%       **Warning**  'minlength'  is  ignored  at  the  moment  because
'cleanedgelist'
%                 has some bugs and can be memory hungry
%
% Arguments:  im         - Binary edge image, it is assumed that edges
%                          have been thinned (or are nearly thin).
%               minlength  - Optional minimum edge length of interest,
defaults
%                           to 1 if omitted or specified as []. Ignored
at the
%                          moment.
%                  location    - Optional  complex valued  image  holding
subpixel
%                          locations of edge points. For any pixel the
%                           real part holds the subpixel row coordinate
of
%                                that edge point and the imaginary part
holds
%                          the column coordinate.  See NONMAXSUP.  If
%                               this argument is supplied the edgelists
will
%                          be formed from the subpixel coordinates,
%                           otherwise the the integer pixel coordinates
of
%                          points in 'im' are used.
%
% Returns:  edgelist - a cell array of edge lists in row,column coords
in
%                     the form
%                    { [r1 c1   [r1 c1   etc }
%                       r2 c2    ...
%                       ...
%                       rN cN]   ....]
%
%            edgeim   - Image with pixels labeled with edge number.
%                       Note that junctions in the labeled edge image
will be
%                       labeled with the edge number of the last edge
that was
```

```
%                        tracked through it.  Note that this image also
includes
%                            edges that do not meet the minimum length
specification.
%                       If you want to see just the edges that meet the
%                       specification you should pass the edgelist to
%                       DRAWEDGELIST.
%
%                etype    - Array of values, one for each edge segment
indicating
%                       its type
%                       0  - Start free, end free
%                       1  - Start free, end junction
%                            2  - Start junction, end free (should not
happen)
%                       3  - Start junction, end junction
%                       4  - Loop
%
% This function links edge points together into lists of coordinate
pairs.
% Where an edge junction is encountered the list is terminated and a
separate
% list is generated for each of the branches.
%
% Note I am not sure if using bwmorph's 'thin' or 'skel' is best for
% preprocessing the edge image prior to edgelinking.  The main issue
is the
% treatment of junctions.  Skel can result in an image where multiple
adjacent
% junctions are produced (maybe this is more a problem with my
junction
% detection code).  Thin, on the other hand, can produce different
output when
% you rotate an image by 90 degrees. On balance I think using 'thin'
is better.
% Note, however, the input image should be 'nearly thin' otherwise the
thinning
% operation could shorten the ends of structures. Skeletonisation and
thinning
% is surprisingly awkward.
%
```

```
% See also:  DRAWEDGELIST, LINESEG, MAXLINEDEV, CLEANEDGELIST,
%            FINDENDSJUNCTIONS, FILLEDGEGAPS

% Copyright (c) 1996-2013 Peter Kovesi
% Centre for Exploration Targeting
% The University of Western Australia
% peter.kovesi at uwa edu au
%
% Permission is hereby granted, free of charge, to any person obtaining a copy
% of this software and associated documentation files (the "Software"), to deal
% in the Software without restriction, subject to the following conditions:
%
% The above copyright notice and this permission notice shall be included in
% all copies or substantial portions of the Software.
%
% The Software is provided "as is", without warranty of any kind.

% February  2001 - Original version
% September 2004 - Revised to allow subpixel edge data to be used
% November  2006 - Changed so that edgelists start and stop at every junction
% January   2007 - Trackedge modified to discard isolated pixels and the
%                  problems they cause (thanks to Jeff Copeland)
% January   2007 - Fixed so that closed loops are closed!
% May       2013 - Completely redesigned with a new linking strategy that
%                  hopefully handles adjacent junctions correctly. It runs
%                  about twice as fast too.

function [edgelist, edgeim, etype] = edgelink(im, minlength, location)

    % Set up some global variables to avoid passing (and copying) of arguments,
    % this improves speed.
    global EDGEIM;
    global ROWS;
    global COLS;
    global JUNCT;

    if ~exist('minlength','var') || isempty(minlength), minlength = 0;
end

    EDGEIM = im ~= 0;                         % Make sure image is binary.
    EDGEIM = bwmorph(EDGEIM,'clean');         % Remove isolated pixels

    % Make sure edges are thinned.  Use 'thin' rather than 'skel', see
    % comments in header.
    EDGEIM = bwmorph(EDGEIM,'thin',Inf);
    [ROWS, COLS] = size(EDGEIM);

    % Find endings and junctions in edge data
    [RJ, CJ, re, ce] = findendsjunctions(EDGEIM);
    Njunct = length(RJ);
    Nends = length(re);

    % Create a sparse matrix to mark junction locations. This makes
junction
    % testing much faster.  A value of 1 indicates a junction, a value
of 2
    % indicates we have visited the junction.
    JUNCT = spalloc(ROWS,COLS, Njunct);
    for n = 1:Njunct
        JUNCT(RJ(n),CJ(n)) = 1;
    end

    % ? Think about using labels >= 2 so that EDGEIM can be uint16,
say. %
    EDGEIM = double(EDGEIM);    % Cast to double to allow the use of -
ve labelings
    edgeNo = 0;

    % Summary of strategy:
    % 1) From every end point track until we encounter an end point or
    % junction.  As we track points along an edge image pixels are
labeled with
    % the -ve of their edge No.
    % 2) From every junction track out on any edges that have not been
    % labeled yet.
    % 3) Scan through the image looking for any unlabeled pixels.
These
    % correspond to isolated loops that have no junctions.


    %% 1) Form tracks from each unlabeled endpoint until we encounter
another
    % endpoint or junction.
    for n = 1:Nends
        if EDGEIM(re(n),ce(n)) == 1  % Endpoint is unlabeled
            edgeNo = edgeNo + 1;
            [edgelist{edgeNo}  endType]  =  trackedge(re(n),  ce(n),
edgeNo);
            etype(edgeNo) = endType;
        end
    end

    %% 2) Handle junctions.
    % Junctions are awkward when they are adjacent to other junctions.
We
    % start by looking at all the neighbours of a junction.
    % If there is an adjacent junction we first create a 2-element
edgetrack
    % that links the two junctions together.  We then look to see if
there are
    % any non-junction edge pixels that are adjacent to both
junctions. We then
    % test to see which of the two junctions is closest to this common
pixel and
    % initiate an edge track from the closest of the two junctions
through this
    % pixel.  When we do this we set the 'avoidJunction' flag in the
call to
    % trackedge so that the edge track does not immediately loop back
and
```

```
    % terminate on the other adjacent junction.
    % Having checked all the common neighbours of both junctions we
then
    % track out on any remaining untracked neighbours of the junction

    for j = 1:Njunct
        if  JUNCT(RJ(j),CJ(j))  ~=  2;    %  We  have  not  visited  this
junction
            JUNCT(RJ(j),CJ(j)) = 2;

            % Call availablepixels with edgeNo = 0 so that we get a
list of
            % available neighbouring pixels that can be linked to and
a list of
            % all neighbouring pixels that are also junctions.
            [ra, ca, rj, cj] =  availablepixels(RJ(j), CJ(j), 0);

            for  k  =  1:length(rj)               %  For  all  adjacent
junctions...
                %  Create  a  2-element  edgetrack  to  each  adjacent
junction
                edgeNo = edgeNo + 1;
                edgelist{edgeNo} = [RJ(j) CJ(j); rj(k) cj(k)];
                etype(edgeNo)  =  3;    %  Edge  segment  is  junction-
junction
                EDGEIM(RJ(j), CJ(j)) = -edgeNo;
                EDGEIM(rj(k), cj(k)) = -edgeNo;

                %  Check  if  the  adjacent  junction  has  some  untracked
pixels that
                % are also adjacent to the initial junction.  Thus we
need to
                %  get  available  pixels  adjacent  to  junction  (rj(k)
cj(k))
                [rak, cak] = availablepixels(rj(k), cj(k));

                %  If  both  junctions  have  untracked  neighbours  that
need checking...
                if ~isempty(ra) && ~isempty(rak)
```

```
                    %  Find  untracked  neighbours  common  to  both
junctions.
                    commonrc =  intersect([ra ca], [rak cak], 'rows');

                    for n = 1:size(commonrc, 1);
                        % If one of the junctions j or k is closer to
this common
                        % neighbour use that as the start of the edge
track and the
                        % common neighbour as the 2nd element. When we
call
                        % trackedge we set the avoidJunction flag to
prevent the
                        %  track  immediately  connecting  back  to  the
other junction.
                        distj = norm(commonrc(n,:) - [RJ(j) CJ(j)]);
                        distk = norm(commonrc(n,:) - [rj(k) cj(k)]);
                        edgeNo = edgeNo + 1;
                        if distj < distk
                            edgelist{edgeNo} = trackedge(RJ(j), CJ(j),
edgeNo, ...
                                                       commonrc(n,1),
commonrc(n,2), 1);
                        else
                            edgelist{edgeNo} = trackedge(rj(k), cj(k),
edgeNo, ...
                                                       commonrc(n,1),
commonrc(n,2), 1);
                        end
                        etype(edgeNo)   =   3;     %   Edge   segment   is
junction-junction
                    end
                end

                %  Track  any  remaining  unlabeled  pixels  adjacent  to
this junction k
                for m = 1:length(rak)
                    if EDGEIM(rak(m), cak(m)) == 1
                        edgeNo = edgeNo + 1;
                        edgelist{edgeNo}   =   trackedge(rj(k),   cj(k),
edgeNo, rak(m), cak(m));
```

```
                        etype(edgeNo)  =  3;    %  Edge  segment  is
junction-junction
                    end
                end

                % Mark that we have visited junction (rj(k) cj(k))
                JUNCT(rj(k), cj(k)) = 2;

            end % for all adjacent junctions

            % Finally track any remaining unlabeled pixels adjacent to
original junction j
            for m = 1:length(ra)
                if EDGEIM(ra(m), ca(m)) == 1
                    edgeNo = edgeNo + 1;
                    edgelist{edgeNo} = trackedge(RJ(j), CJ(j), edgeNo,
ra(m), ca(m));
                    etype(edgeNo) = 3;   % Edge segment is junction-
junction
                end
            end

        end  % If we have not visited this junction
    end   % For each junction

    %% 3)  Scan  through  the  image  looking  for  any  unlabeled  pixels.
These
    %  should  correspond  to  isolated  loops  that  have  no  junctions  or
endpoints.
    for ru = 1:ROWS
        for cu = 1:COLS
            if EDGEIM(ru,cu) == 1  % We have an unlabeled edge
                edgeNo = edgeNo + 1;
                [edgelist{edgeNo}   endType]   =   trackedge(ru,   cu,
edgeNo);
                etype(edgeNo) = endType;
            end
        end
    end
    edgeim = -EDGEIM;   % Finally negate image to make edge encodings
+ve.

    % Eliminate isolated edges and spurs that are below the minimum
length
    % ** DISABLED for the time being **
%    if nargin >= 2 && ~isempty(minlength)
%   edgelist = cleanedgelist2(edgelist, minlength);
%    end

    % If subpixel edge locations are supplied upgrade the integer
precision
    % edgelists that were constructed with data from 'location'.
    if nargin == 3
    for I = 1:length(edgelist)
        ind = sub2ind(size(im),edgelist{I}(:,1),edgelist{I}(:,2));
        edgelist{I}(:,1) = real(location(ind))';
        edgelist{I}(:,2) = imag(location(ind))';
    end
    end

    clear global EDGEIM;
    clear global ROWS;
    clear global COLS;
    clear global JUNCT;

%----------------------------------------------------------------------
-
% TRACKEDGE
%
% Function to track all the edge points starting from an end point or
junction.
% As it tracks it stores the coords of the edge points in an array and
labels the
% pixels in the edge image with the -ve of their edge number. This
continues
% until no more connected points are found, or a junction point is
encountered.
%
```

```
%  Usage:     edgepoints  =  trackedge(rstart,  cstart,  edgeNo,  r2,  c2,
avoidJunction)
%
% Arguments:   rstart, cstart   - Row and column No of starting point.
%              edgeNo           - The current edge number.
%                r2, c2             - Optional row and column coords of
2nd point.
%                     avoidJunction      - Optional flag indicating that
(r2,c2)
%                                    should not be immediately connected
to a
%                                  junction (if possible).
%
% Returns:     edgepoints       - Nx2 array of row and col values for
%                               each edge point.
%              endType          - 0 for a free end
%                               1 for a junction
%                               5 for a loop

function [edgepoints endType] = trackedge(rstart, cstart, edgeNo, r2,
c2, avoidJunction)

    global EDGEIM;
    global JUNCT;

    if ~exist('avoidJunction', 'var'), avoidJunction = 0; end

    edgepoints = [rstart cstart];        % Start a new list for this
edge.
    EDGEIM(rstart,cstart) = -edgeNo;    % Edge points in the image are
                                   % encoded by -ve of their edgeNo.

    preferredDirection = 0;             % Flag indicating we have/not a
                                        % preferred direction.

    % If the second point has been supplied add it to the track and
set the
    % path direction
    if exist('r2', 'var') && exist('c2', 'var')
        edgepoints = [edgepoints
```

```
                  r2    c2 ];
        EDGEIM(r2, c2) = -edgeNo;
        %  Initialise  direction  vector  of  path  and  set  the  current
point on
        % the path
        dirn = unitvector([r2-rstart c2-cstart]);
        r = r2;
        c = c2;
        preferredDirection = 1;
    else
        dirn = [0 0];
        r = rstart;
        c = cstart;
    end

    % Find all the pixels we could link to
    [ra, ca, rj, cj] = availablepixels(r, c, edgeNo);

    while ~isempty(ra) || ~isempty(rj)

        % First see if we can link to a junction. Choose the junction
that
        % results in a move that is as close as possible to dirn. If
we have no
        %  preferred  direction,  and  there  is  a  choice,  link  to  the
closest
        % junction
        % We enter this block:
        % IF there are junction points and we are not trying to avoid
a junction
        % OR there are junction points and no non-junction points, ie
we have
        % to enter it even if we are trying to avoid a junction
        if  (~isempty(rj)  &&  ~avoidJunction)   ||  (~isempty(rj)  &&
isempty(ra))

            % If we have a prefered direction choose the junction that
results
            % in a move that is as close as possible to dirn.
            if preferredDirection
```

```
                dotp = -inf;
                for n = 1:length(rj)
                    dirna = unitvector([rj(n)-r  cj(n)-c]);
                    dp = dirn*dirna';
                    if dp > dotp
                        dotp = dp;
                        rbest = rj(n); cbest = cj(n);
                        dirnbest = dirna;
                    end
                end

            % Otherwise if we have no established direction, we should
pick a
            % 4-connected junction if possible as it will be closest.
This only
            %  affects  tracks  of  length  1  (Why  do  I  worry  about
this...?!).
            else
                distbest = inf;
                for n = 1:length(rj)
                    dist = sum([rj(n)-r;  cj(n)-c]);
                    if dist < distbest
                        rbest = rj(n); cbest = cj(n);
                        distbest = dist;
                        dirnbest = unitvector([rj(n)-r  cj(n)-c]);
                    end
                end
                preferredDirection = 1;
            end

        % If there were no junctions to link to choose the available
        % non-junction pixel that results in a move that is as close
as possible
        % to dirn
        else
            dotp = -inf;
            for n = 1:length(ra)
                dirna = unitvector([ra(n)-r  ca(n)-c]);
                dp = dirn*dirna';
                if dp > dotp
                    dotp = dp;
```

```
                  rbest = ra(n); cbest = ca(n);
                  dirnbest = dirna;
              end
          end

          avoidJunction = 0; % Clear the avoidJunction flag if it
had been set
        end

        % Append the best pixel to the edgelist and update the
direction and EDGEIM
        r = rbest; c = cbest;
        edgepoints = [edgepoints
                        r    c  ];
        dirn = dirnbest;
        EDGEIM(r, c) = -edgeNo;

        % If this point is a junction exit here
        if JUNCT(r, c);
            endType = 1;  % Mark end as being a junction
            return;
        else
            % Get the next set of available pixels to link.
            [ra, ca, rj, cj] = availablepixels(r, c, edgeNo);
        end
    end

    % If we get here we are at an endpoint or our sequence of pixels
form a
    % loop.  If it is a loop the edgelist should have start and end
points
    % matched to form a loop.  If the number of points in the list is
four or
    % more (the minimum number that could form a loop), and the
endpoints are
    % within a pixel of each other, append a copy of the first point
to the end
    % to complete the loop

    endType = 0;  % Mark end as being free, unless it is reset below
    if length(edgepoints) >= 4
    if abs(edgepoints(1,1) - edgepoints(end,1)) <= 1  &&  ...
           abs(edgepoints(1,2) - edgepoints(end,2)) <= 1
        edgepoints = [edgepoints
              edgepoints(1,:)];
            endType = 5; % Mark end as being a loop
        end
    end

%----------------------------------------------------------------------
-
% AVAILABLEPIXELS
%
% Find all the pixels that could be linked to point r, c
%
% Arguments:  rp, cp - Row, col coordinates of pixel of interest.
%             edgeNo - The edge number of the edge we are seeking to
%                      track. If not supplied its value defaults to 0
%                            resulting in all adjacent junctions being
returned,
%                      (see note below)
%
% Returns:     ra, ca - Row and column coordinates of available non-
junction
%                      pixels.
%                  rj, cj  - Row and column coordinates of available
junction
%                      pixels.
%
% A pixel is avalable for linking if it is:
% 1) Adjacent, that is it is 8-connected.
% 2) Its value is 1 indicating it has not already been assigned to an
edge
% 3) or it is a junction that has not been labeled -edgeNo indicating
we have
%     not already assigned it to the current edge being tracked.  If
edgeNo is
%    0 all adjacent junctions will be returned
```

```
function  [ra, ca, rj, cj] = availablepixels(rp, cp, edgeNo)

    global EDGEIM;
    global JUNCT;
    global ROWS;
    global COLS;

    % If edgeNo not supplied set to 0 to allow all adjacent junctions
to be returned
    if ~exist('edgeNo', 'var'), edgeNo = 0; end

    ra = []; ca = [];
    rj = []; cj = [];

    % row and column offsets for the eight neighbours of a point
    roff = [-1  0  1  1  1  0 -1 -1];
    coff = [-1 -1 -1  0  1  1  1  0];

    r = rp+roff;
    c = cp+coff;

    % Find indices of arrays of r and c that are within the image
bounds
    ind = find((r>=1 & r<=ROWS) & (c>=1 & c<=COLS));

    % A pixel is avalable for linking if its value is 1 or it is a
junction
    % that has not been labeled -edgeNo
    for i = ind
        if EDGEIM(r(i),c(i)) == 1 && ~JUNCT(r(i), c(i));
            ra = [ra; r(i)];
            ca = [ca; c(i)];
        elseif (EDGEIM(r(i),c(i)) ~= -edgeNo) && JUNCT(r(i), c(i));
            rj = [rj; r(i)];
            cj = [cj; c(i)];
        end
    end
```

```
%-----------------------------------------------------------------------
% UNITVECTOR Normalises a vector to unit magnitude
%

function nv = unitvector(v)

    nv = v./sqrt(v(:)'*v(:));
```

9.2 MOSAICING

```
im = imread('Testis_histology_014.jpg');
im = rgb2gray(im);
im = double(im);

video(:,:,1) = im(50:250,50:250);
video(:,:,2) = im(100:300,50:250);
video(:,:,3) = im(50:250,75:275);
video(:,:,4) = im(30:230,100:300);

for j = 1:4

if j == 1
    si = size(video(:,:,j));
    im2 = zeros(si(1)+300,si(2)+300);
    im2(151:si(1)+150,151:si(2)+150) = video(:,:,j);
disp('primer ciclo')
end;
j
im1 = video(:,:,j);
[h,im_final, theta,I,J]=imregistMI(im1, im2, 0, 1);

 im2 = im_final;

end;
```

9.2.1 IMREGISTERMI

```
srfunction  [h,im_final,  theta,I,J]=imregistMI(image1,  image2,  angle,
step);

% Kateryna Artyushkova
% Postdoctoral Scientist
% Department of Chemical and Nuclear Engineering
% The University of New Mexico
% (505) 277-0750
% kartyush@unm.edu

a=isa(image1,'uint16');
if a==1
     image1=double(image1)/65535*255;
 else
     image1=double(image1);
end

a=isa(image2,'uint16');
if a==1
    image2=double(image2)/65535*255;
else
    image2=double(image2);
end

[m,n]=size(image1);
[p,q]=size(image2);
[a,b]=size(angle);
im1=round(image1);
method = 'Normalized';




for k=1:b
    J = imrotate(image2, angle(k),'bilinear'); %rotated cropped IMAGE2
    image21=round(J);
    [m1,n1]=size(image21);
    for i=1:step:(m1-m)
        for j=1:step:(n1-n)
                im2=image21(i:(i+m-1),j:(j+n-1));  % selecting part of
IMAGE2 matching the size of IMAGE1
                im2=round(im2);
                h(k,i,j)=MI2(im1,im2,method); % calculating MI
            end
        end
    end

[a, b] = max(h(:));% finding the max of MI and indecises
[K,I,J] = ind2sub(size(h),b);




theta=angle(K);
im_rot = imrotate(image2, theta,'bilinear');
im_matched=im_rot(I:(I+m-1),J:(J+n-1));
im_final = im_rot;
im_final(I:(I+m-1),J:(J+n-1)) = im1;
H.Position=[232 258 259 402];
figure(H)
subplot(1,3,1),imagesc(im1)
subplot(1,3,2),imagesc(im2)
subplot(1,3,3),imagesc(im_final)
```

9.2.1.1 COST_F

```
function h=MI2(image_1,image_2,method)
% function h=MI2(image_1,image_2,method)
%
% Takes a pair of images and returns the mutual information Ixy using
joint entropy function JOINT_H.m
%
% written by http://www.flash.net/~strider2/matlab.htm
```

```
a=joint_h(image_1,image_2);  %  calculating  joint  histogram  for  two
images
[r,c] = size(a);
b= a./(r*c); % normalized joint histogram
y_marg=sum(b); %sum of the rows of normalized joint histogram
x_marg=sum(b');%sum of columns of normalized joint histogran

Hy=0;
for i=1:c;    %  col
      if( y_marg(i)==0 )
         %do nothing
      else
         Hy  =  Hy  +  -(y_marg(i)*(log2(y_marg(i)))); %marginal  entropy
for image 1
      end
   end

Hx=0;
for i=1:r;    %rows
   if( x_marg(i)==0 )
         %do nothing
      else
         Hx  =  Hx  +  -(x_marg(i)*(log2(x_marg(i)))); %marginal  entropy
for image 2
      end
   end
h_xy = -sum(sum(b.*(log2(b+(b==0))))); % joint entropy

if method=='Normalized';
h = (Hx + Hy)/h_xy;% Mutual information
else
h = Hx + Hy - h_xy;% Mutual information
end
```

10. REFERENCES

[1] Gerard J. Tortora & Bryan Derrickson. “Principles of Anatomy and Physiology”. *John Wiley & Sons*. 13th edition.

[2] National Cancer Institute Web Page. Colorectal Cancer – Patient Version

Link: https://www.cancer.gov/types/colorectal

[3] Care and Handling of the Flexible Endoscope. Anatomy and Physiology of a Flexible Scope: Anatomical Structure.

[4] Wikipedia Contributors. “Endoscopy”. *Wikipedia, The Free Encyclopedia.* Wikipedia, the Free Encyclopedia 30 Ag 2016

[5] Walker R and Beasant N., Customization of optical fiber leads to new applications in monitoring, manufacturing and research. Photonics Spectra.

[6] Blanca Zufiria Gerbolés. Fluorescence Endoscopy In-vivo based on Fiber-bundle Measurements.

[7] Mark Nixon & Alberto Aguado. “Feature Extraction and Image Processing”.*Elsevier*. 2nd Edition

[8] Stafford michahial, Latha M, Akshatha S, Juslin F, Ms Manasa B, Shivani U.Automatic Image Mosaicing Using Sift, Ransac and Homography. IJEIT, vol 3, Issue 10, 2014 April 2

[9] Ruben Tous, Jaime Delgado, Thomas Zinkl, Pere Toran and Gabriela Alcalde.The Anatomy of an Optical Biopsy Semantic Retrieval System.

[10] MathWorks – Documentation. Link: http://es.mathworks.com/

[11] Center for Machine Vision and Signal Analysis – Downloads – LBP Matlab.

Link: http://www.cse.oulu.fi/wsgi/CMV/Downloads/LBPMatlab

[12] Asociación Mexicana de Endoscopia Gastrointestinal – Historia de la Endoscopia.

Link:http://www.amegendoscopia.org.mx/index.php/ameg/historia/145-historia-de-la-endoscopia