rijksuniversiteit groningen

Center Position Normalization in Endoscopy Video

Kevin Gevers
2017

Bachelor Thesis
Scientific Visualization and Computer Graphics
University of Groningen, The Netherlands

Primary supervisor
PhD A. Sobiecki

Secondary supervisor
Prof. Dr. A.C. Telea

In collaboration with ZiuZ Visual Intelligence

Contents

1 Introduction **2**

2 Related works **4**
2.1 Pill-camera endoscopy 4
2.2 Improving pill-camera results 4
2.2.1 Pill-camera related methods 5
2.2.2 Generic image processing methods 6
2.3 Conclusion 7

3 Method **8**
3.1 Image pre-processing 9
3.2 Segmentation 10
3.2.1 Histogram 11
3.2.2 Blob detection 12
3.2.3 Combining results 13
3.3 Outliers 14
3.3.1 Linear interpolation 14
3.3.2 No center detection possible 15
3.4 Time smoothing 16
3.5 Translation 16
3.6 Rotation 16
3.6.1 Local binary pattern brute force 16
3.6.2 Bounding boxes 17
3.6.3 Adjusted binary image brute force 18

4 Results **19**
4.1 Time Smoothing 23
4.2 Rotation 24
4.3 User interaction 26

5 Conclusions **27**
5.1 Discussion 27
5.2 Future works 27

1 Introduction

A capsule endoscopy camera is a device a patient needs to swallow and it will make a recording of it's insides. This reaches the entire intestine, where as colonoscopy does not. The capsule is pill shaped and contains a transmitter, LED lights and a camera on one side. The advantage of pill is that it's less intrusive for the patient and that the doctor doesn't have to prepare a lengthy and costly classical endoscopy session[12]. But because these devices are quite new and only provide the raw video images it's a very lengthy job for a medical doctor to inspect it all. On top of that the raw video feed can be hard on the eyes because the camera can make very sudden moves and twist around a lot. So the time saved before is now lost. Which raises the question of how to decrease the amount of time needed for a doctor to process the endoscopy video?

There are several different things that can be done to achieve this goal. You can try to remove the frames where the camera moved back, so you don't look at the same part of the intestines twice. You can try to lower the contrast in the images in such a way that the darker parts are better visible. You can try to summarize the video automatically so you don't have to watch everything. You can try to have the moving parts in the video stabilized so the image is easier on the eyes. Because this is way to much work for a single bachelor project we will only work on the last one. We will try to detect the center of the image and place that in the center of the frame, thus normalizing the center. Which gives the following research questions: "How to detect the centre of the direction of the image in a capsule video?"

Figure 1: An example of a capsule endoscopy camera

Now this comes with requirements that have to be met in order to complete the project successfully.
The center position should be normalized: This means that the center needs to be detected and the image should be adjusted so that the center is in the middle of the frame.
The video should run smoothly: So that the images don't jump around more because of the normalization than they did before. Otherwise the video would only become harder to watch.
The video should run (nearly) automatic: The doctor should not have to do a lot of work to make the program work. This means that the video should run automatically with the exception of changing some options when preferred.
The program should be computationally efficient enough to run real time: This just means that the program should not be slower than the video, if that were the case the doctor would have to wait for the video and thus not saving time.
The program needs to be able to process the given input video correctly: The input video is in the MPEG-4 codec and is 320 by 320 pixels in size. The video runs at 15 frames per second, but was filmed at 3 frames per second. Video's vary from around 8 hours to 12 hours in real time. So the program needs to be able to open this video, process the images and output a video with the center position normalized.

In this thesis we will discuss the work related to this project in the next chapter. After that we will outline the method proposed to answer the research question and fulfill the requirements, followed by the results of testing and a conclusion.

2 Related works

Because there is a lot of related research already done we will discuss it here. Starting out by what kind of research has been done for the pill camera in general. This is all done to ether help the doctors in viewing the video or to even automatically analyze the video so no doctor has to look at it. Next we will move on to research that has been done to improving the resulting video and because there are also more general image processing techniques that might be useful we will discuss those here as well. This should thus give us a good idea of what techniques have been used for similar projects and what might be useful for us.

2.1 Pill-camera endoscopy

All pill-camera's used for endoscopy have similar components: A plastic disposable capsule, a white light emitting diode (LED), a small camera and a battery source. A doctor can chose to use a pill-camera for patients that he expects to have one of the following conditions: Bleeding in the general intestinal track, Crohn's disease, small intestine tumors and certain syndromes that relate to the intestinal track. When the patient uses the pill he is not allowed to eat and can only drink clear liquids for 10 to 12 hours to make sure the images aren't getting too distorted by it. The patient can go about it's day normally after ingestion but has to watch for the pill in their excrement's [12].

Research has been done using the pill camera for detecting various health issues. One of these is bleeding in the general intestinal track. Because bleeding often occurs in the small bowel, which is much harder to inspect with push enteroscopy and bowel barium radiography. One way to detect bleeding is by using first order histogram probability on the three color planes in RGB (Red green blue color space) images. Using the mean, standard deviation, skew and energy from the image they calculate the histogram probability. After that is done they reduce the amount of different colors in the image to from 255 for each red, green and blue down to 24. They then cut the image into regions and use feature selection in the form of exhaustive search on each of those. From their tests they got a 89% accuracy in detecting bleeding [6].

Another health issue that has been researched is lesion detection. These lesions can be polyps or tumors. The research here uses various techniques to classify these lesions. First they use video segmentation, this means they can split up the image into multiple regions to better process the meaningful sections. After that they use object tracking to track any feature they is meaningful. The use edge detection to find the shapes inside the colon. This means that you look for the section where there is a big difference in contrast that represents an edge. Doing so shows all the shapes in the colon, which could definitely be used for locating the sides of the intestines. But in this case they use it to detect the small lesions. [2]

It's save to say that there is a lot of potential in the pill camera endoscopy, but with the current success rates being under the 99% there is a lot of work to be done before these automatic detection techniques will be so good that doctors trust them enough to no longer look at the video's themselves.

2.2 Improving pill-camera results

The raw images from the pill camera aren't that good and hard to use as is. There are multiple reasons for this: First because the pill is making a lot of movements. Both rotating and turning from side to side cause the video to be hard to follow. The LED inside the capsule can only light the intestines that are close by, so the tissue close by is over-exposed and the tissue farther away is too dark. The pill also goes back and forth due to the bowel contractions, thus recording the same part of the intestinal track multiple times. In most cases improving the results has to be done in the form of post-processing the raw images from the pill camera.

2.2.1 Pill-camera related methods

One way to improve the raw images is to normalize the center position based on lumen. But before we can discuss how to detect the center we need to define what the center actually is. Because the intestines are only lit by the capsule's LED, the dark parts showing up in the video are farther away from the camera than the brightly lit parts. This means that in most cases the dark part will be where the capsule will go next (when moving forward). But because the colon contracts, there are images with multiple smaller dark spots. In most cases the camera is moving towards the largest dark spot because that is where the most space is available for it. That means that the larger dark blobs that are closer to the center are more likely to be the path for the pill. So for the purpose of this paper the center will be bary-center of the large dark blob that is close to the center of the image.

There is little research done for this, since the most research has been done in automatically detecting various things as described above. There is however related work in the form of object tracking for endoscopy video. The goal here has been done for various objects, for instance following medical tools in order to tell a doctor which tool is which when the image is unclear. But also to track the lesions we detected earlier. We want to detect the large dark blob in the images and thus track it. The tracking of objects also relies heavily on segmenting the images. For tracking medical tools a RGB histogram is made for the various regions. Because they color labeled the various medical instruments these will give spikes in the histogram when these tools are present [14]. Even though this is very specific for medical instruments the techniques are very useful for us, because we want to detect dark sections in the images.

Another technique that has been used for improving the raw images is using the mean shift segmentation algorithm. This is a more advanced and complicated segmentation algorithm. It places a certain amount of kernels, which are circles with a certain starting radius representing a region of interest. Next it determines the centroid of the data and moves the kernel so that it aligns with it. This keeps on going until it convergences and high intensity sections are found. While this means that there are possibly multiple points in the image where this occurs the kernels can also go to the same region. The fourth image of the figure below show how it can look like.[15][4][16].

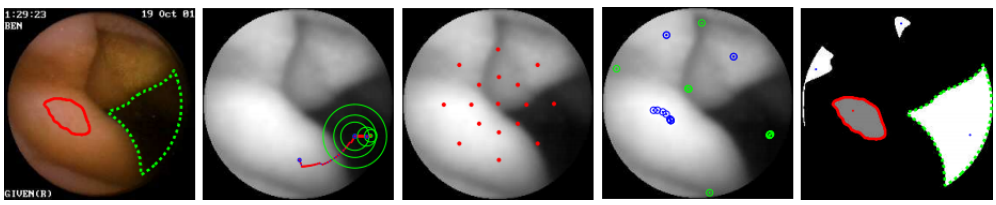


Figure 2: Overview of the steps of method in the "Lumen detection for capsule endoscopy" paper. "Left to right: (a) Original image with superimposed result; solid line outlines highlight and dashed line the lumen. (b) Behavior of the proposed variant of the MS algorithm. Figure shows the trajectory of K and the progressive reduction of its radius. Notice that before converging to the intensity extremum, the trajectory of the kernel is parallel to the boundary of the image. (c) Initialization seeds for multiple runs of the MS algorithm. (d) The resulting MSRs for the dark (bright dots) and bright (dark dots) areas. (e) Extracted regions and their boundary approximation; region representatives indicated with dots."[4]

It is clear from the above figure that it is very accurate in finding both the dark and overexposed sections in the images. This makes it a very robust method for detecting the center position in the frames. However it is also a computationally expensive algorithm, it takes 3 seconds to calculate a single frame [4]. With a frame rate of 3 frames per second at filming and normally playing that at 15 frames per second this is much too slow for our implementation.

2.2.2 Generic image processing methods

Since improving the raw video of the pill camera is done with image processing techniques we also looked into that. Trying to find the best rotation for the image to counteract the rotation of the pill camera also needs image processing techniques which we have not yet encountered.

While we already named edge detection before this was not the main focus. Edge detection is however a very useful technique when looking for section that is much darker than the others. There is a wide variety of edge detection techniques that can be used, some of the simpler ones are: Roberts, Prewitt, Sobel and the Robinson compass. All of these work very similar, all having pros and cons. To explain the basic idea we will discuss Prewitt. These use a mask of which the size can vary, but the simplest one is 3 by 3 pixels. The middle pixel is set to 0 and the top 3 pixels are set to -1, while the bottom 3 are set to 1. This means that when applying the mask you look for an edge where it goes from dark to light in a horizontal line segment. Applying this mask to the entire image results in only horizontal lines, so you also need to do the same again with the mask rotated 90 degrees clockwise. Now this is a simple edge detection algorithm, but if you use more complex ones you get better results.[10]

Figure 3: Left: The original image for which we want to find the edges. Right: The image after running the algorithm, thus only showing the edges. (In this image Canny was used, which is a slightly more complex edge detector.)

While this technique is great for finding edges it does no selecting what so ever and does not keep color in account. That means this is useful for intermediate calculations but does not give us the actual center.

For the rotation of the image it's very important to be able to compare the images properly. By converting images to local binary patterns this is much easier. Local binary pattern again looks at every pixel individually, but now instead of applying a mask it looks if the surrounding pixels are higher or lower than the middle pixel (so a 3 by 3 set). Every pixel that is higher is marked with a 1 and every pixel that is lower is marked by a 0. Now this gives a set of 0's and 1's which can be read as a binary number. This binary number is what is put in the spot of the middle pixel in the resulting image. After this the images are simpler to compare without losing important data.

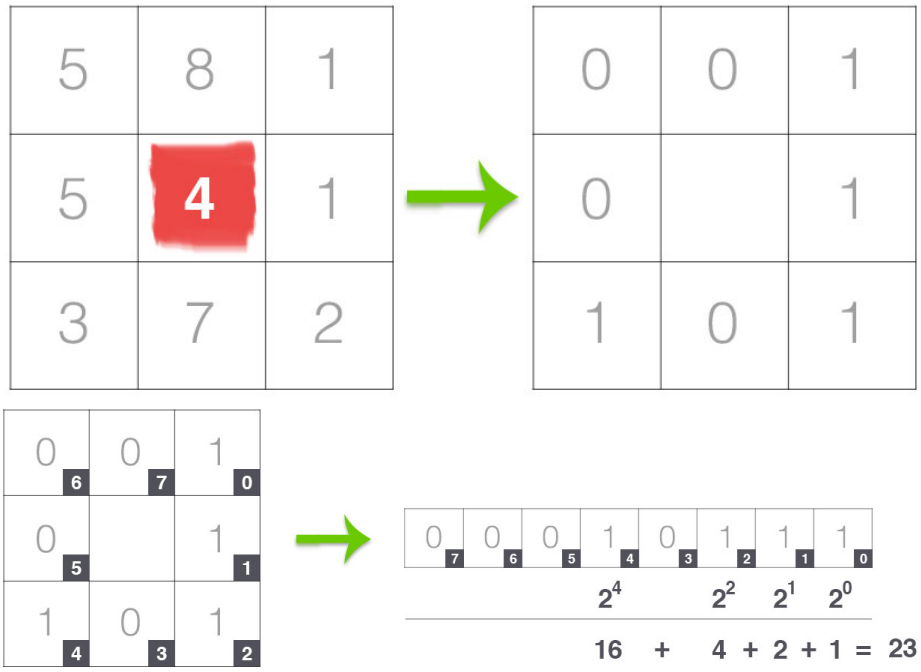


Figure 4: Top: A section of an image before and after the lbp thresholding is done. The middle pixel is the threshold and on the right you can see the 0's and 1's that result from this. Bottom: Now that the binary numbers have been determined they have to be put in sequence to get the actual number. This can be seen here, and in this case 23 will be placed in the resulting image.

2.3 Conclusion

Now that we've gone over the different related works, we can conclude that some of the methods are too specialized, some are too computationally expensive and some have potential. Now one thing that kept coming up was segmentation, with the most basic aspect of it being a histogram. But because this might not be precise enough we will also try a blob detector. A blob detector is a specialized version of edge detection that tries to find blobs, instead of just edges in general. On top of this we found a method to compare images better, which we will use for the rotation aspect of the process.

3 Method

Now that we know what related research has already been done we can continue to set up a method that will answer the research question. In the figure below you will see the design space which will be converted into the pipeline of the actual program. To answer the research question using lumen detection, while making sure all requirements are met, both the detection and the alignment are needed. Detection is where we try to find the center of the image by the means of segmentation, which is again split up in a blob detector and a histogram. The actual classification happens by combining the results from the segmentation and seeing if linear interpolation can be used. This results in a successfully detected center, a center found with linear interpolation or a frame where no center detection was possible. When that is done the alignment has to be done, the subsections there can all be used or just a subgroup of them. If there are outliers in the results linear interpolation can be used, but there are also cases where there is simply no center detection possible. Time smoothing can be done to try and stabilize. Both linear interpolation and time smoothing have to be done (if wanted) before the transformation. Transformation is where the image is translated and/or rotated. For the rotation there are three different methods that can be used, but since you can only use one at a time there is just one node. This gives us the pipeline below, in which all these sections are included at their right time.

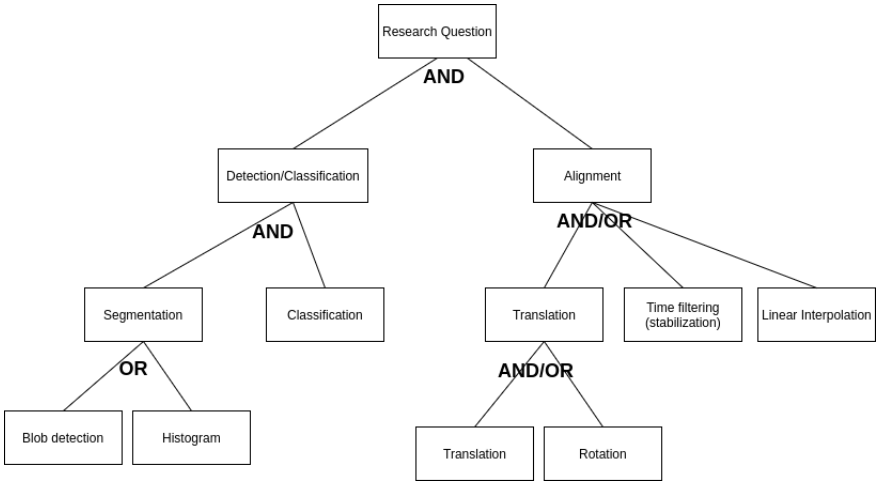


Figure 5: The outline of the proposed solution

In the above diagram it shows what the proposed solution is. Converting this to an actual implementation gives the following steps:

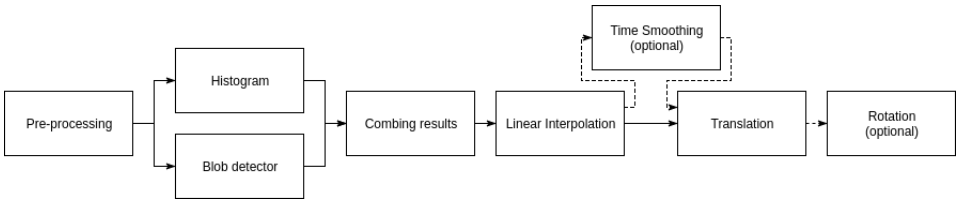


Figure 6: The pipeline for the program

With as input the raw endoscopy video and as output the center normalized video.

3.1 Image pre-processing

The video's are all 320 by 320 pixels, of the MPEG-4 codec and all have the same circular video with writings (as can be seen in the image below). In order to be able to process these images correctly they first need some altering to allow the center to be detected later on. First we need to get rid of the the black corner pieces and writing on it. These are the sections in each corner that don't have actual video on it but are there to make the image square. We do this by making a mask that cuts of almost all the extra pixels, but it leaves a small amount of pixels at the edges because the circle can be 1 or 2 pixels off. These pixels will be less than 1 percent of the image and can thus be ignored.

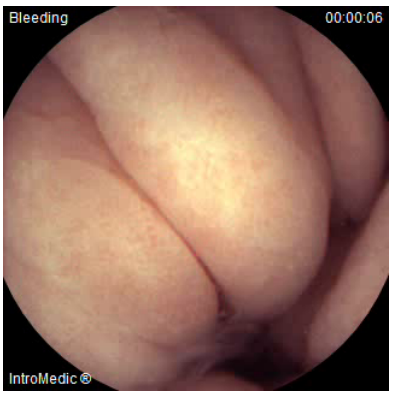


Figure 7: A frame of one of the raw videos

To get the image ready for the segmentation process we use morphological closing. Morphological closing is used for removal of details. It uses a structuring element, which is in this case a circle. The structuring element can be of various shapes, but in this application it would make sense to use a circle or ellipse because the shapes in the images are like that too.

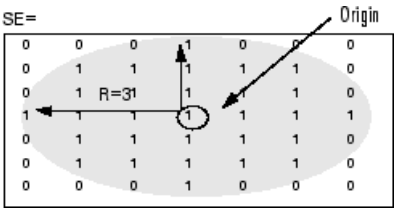


Figure 8: An example of a circular shaped structuring element, as you can see it looks like an ellipse but is actually has the same width and height, and is thus a circle when applied.

The structuring element (see image above) is applied to each pixel, to see if it fits for the other pixels it covers. For dilation only the center has to fit and the image is made bigger by turning the pixels that it covers into 1's. For erosion the whole structuring element has to fit and the edges are trimmed off, by turning the outer rim of the pixels into 0's. So dilation makes a shape bigger while erosion makes it smaller. Combining these in the order dilation, erosion, erosion and dilation gives the morphological closing. Because this process is hard to imagine the images below will clarify what actually happens.

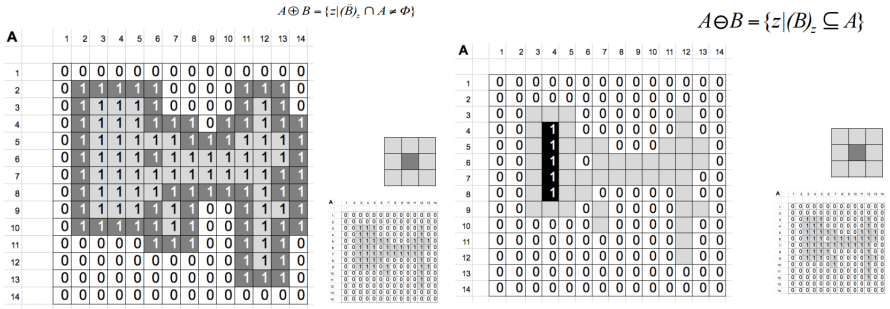


Figure 9: Left: An example of how dilation works. The structuring element is placed on top of every pixel and for every pixel that is marked with a 1 the pixels around it are also changed to 1s. Doing this for every pixel gives the resulting dilated image. Right: An example of how erosion works. This works similar to dilation, but if now the whole structuring element needs to fit, so if you use a square you also need a square of 1's in the image. If this fits the pixel stays a 1, if it doesn't fit for that pixel it is changed to a 0.[9]

Using this method a raw images takes out all the details you don't want to have during the segmentation process that is up next.

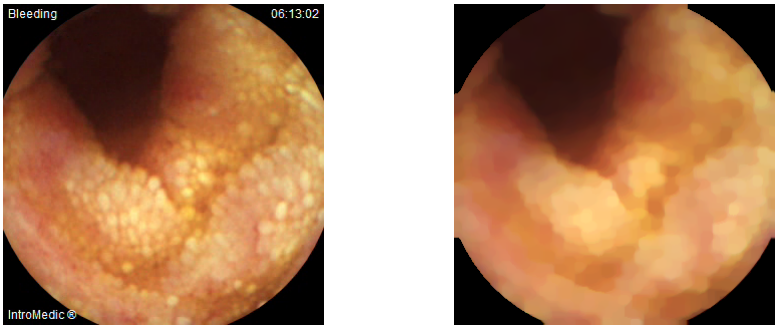


Figure 10: Left: A raw frame from one of the videos. Right: The same frame after the closing operation has been applied. You can see that this image has lost most of the details, but still has the dark blob fairly unaltered. This makes processing this image a lot more reliable than the raw image.

3.2 Segmentation

Now that the image is pre-processed we can start looking for the actual center. This is needed so that we can normalize the center afterward. Trying to find the center is done by image segmentation, which is the process of partitioning a digital image into multiple segments and is used to locate objects or areas in images. A wide variety of methods exist for this, but the easiest is the histogram method. Because this is the easiest (and thus computationally cheap) method we will be trying this. Another method for segmentation is edge detection, where you try to detect the edges in an image by finding places with high contrast shifts. The method we will be using is the "Blob detector", this uses edge detection to look for blobs in an image. Because the dark area's we're looking for are not of any particular shape, a blob is the choice that makes most sense. Because we found limitations in both methods, but these limitations don't overlap we combined the results to get the most accurate final result.

3.2.1 Histogram

The histogram will allow us to see how the intensities are spread out for each frame. Using this we can set a threshold which in turn will allow us to determine which pixels are dark enough to be the center of the direction the camera is going. In order to make a histogram of the intensity of the pixels the image first has to be converted from RGB to greyscale. In the resulting image the pixels have a range of [0..255], 0 being the darkest and 255 being the brightest. Counting the number of pixels for each of these intensities gives me the histogram. Looking for the first peak that is larger than $x\%$ of the largest peak in the histogram, we can determine which peak to use as a threshold. Preferably this peak is narrow because that means it has good contrast to the other intensities. But by also using the peaks that are not narrow it is still possible to find their center. The threshold represents the darkest color of which enough pixels are present in the current frame.

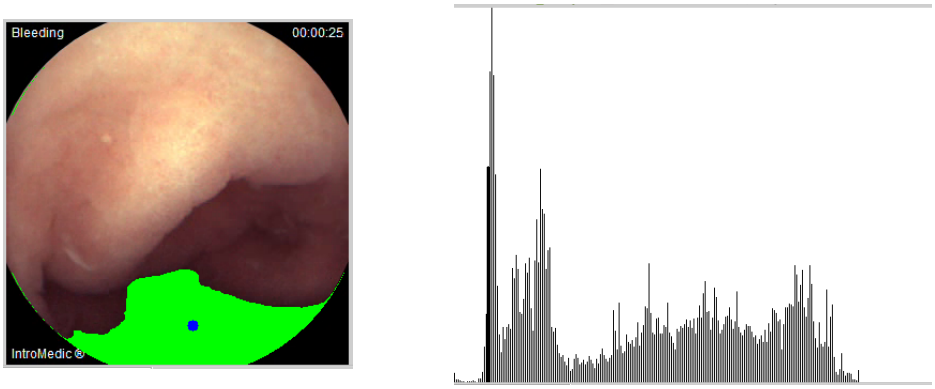


Figure 11: Left: A frame with the pixels below the threshold colored in green and their center marked with a blue dot (the size of the dot is arbitrary). Right: The corresponding histogram, with a narrow peak. The first spike, which is in bold, is used as the threshold.

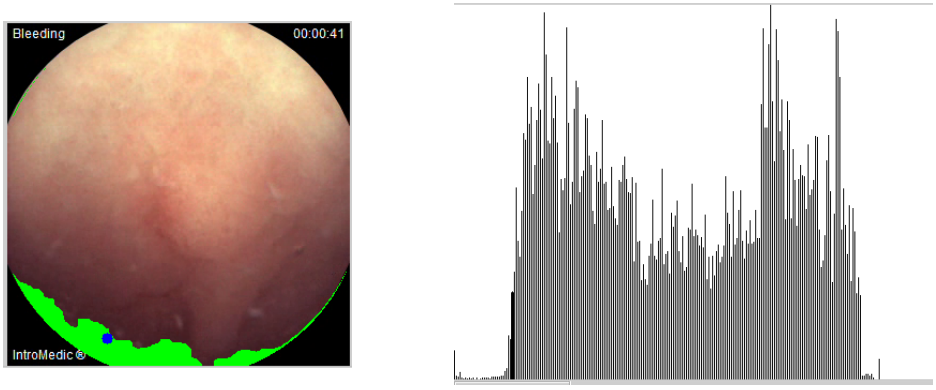


Figure 12: Left: A frame with the pixels below the threshold colored in green and their center marked with a blue dot (the size of the dot is arbitrary). Right: The corresponding histogram, which does not have a narrow peak but still gives the wanted result. The first intensity that is high enough is in bold, but it is clearly not a real spike.

Going over the image again using the found threshold and marking any pixel below it as white and every other pixel as black gives a binary image as the result of the segmentation. Here it is important to ignore any frame that has less than 1% of the pixels colored white because that is caused by the little edges the mask does not cut off.

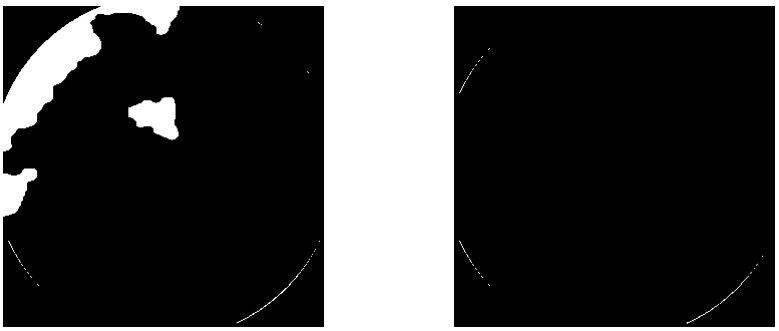


Figure 13: Left: An example of a binary image, where the area's that are darker than the threshold are white and everything else is black. Right: An example of a binary frame that does not meet the 1% cutoff mark. You can see that there are some lines in white where the mask fails. This is because in these cases the pixels that did not get cut off are black already and thus meet the threshold. Since this is clearly not the center we want to ignore frames like these.

Because there is a good possibility we get multiple blobs we have to select the right one, which is most likely the biggest one. This is because the smaller blobs are likely to be noise. Using the largest connected component algorithm we can determine which blob is the biggest and take the center of it. Since this only returns the largest blob found, it might not be correct so we need the blob detection to get a more accurate result.

3.2.2 Blob detection

Here we will use a blob detector, which is a more advanced segmentation algorithm than a simple histogram, but is not too advanced that it starts getting too slow. The "OpenCV" [11] library offers this function, so we did not have to write it ourselves. The algorithm does need a set of parameters in order to run. And since the program should run automatically these have to be determined once, so playing with these parameters will be needed.

The in order to find the dark blobs we want, the blob detector follows a simple algorithm. It starts off by converting the frame to multiple binary images by applying different thresholds. Much like the histogram method, but now using a set of different thresholds. This is done by starting at the minimal threshold and then incrementing with a step until the maximum threshold is reached. From these binary images it extracts the connected components and calculates their center. Then it groups the centers from the different images by their coordinates and combines the ones that are close by. From which it takes the final center and radius for every blob.

Now in order to make this all work as described the algorithm needs a lot of input. Because one of the requirements the project has is that it should run (nearly) automatically, we had to play around with these and find the best ones. The minimum and maximum threshold can be set to quite a big range, since some images are very bright and others are quite dark. The minimum threshold is set to 10 and the maximum threshold is set to 200. Using a thresholding step of 10 this results in 19 different thresholds that are applied. This means that there are 19 binary images and each one of them has more pixels colored white than the one before because the threshold is higher. The minimum area/size of a blob is also important to be set reasonably high (we set this to 10, with the image width being 320), this is because too small blobs are very likely to be cluttering or other noise. The last important parameter is the color, this is set to look for dark blobs instead of light ones.

However there is a downside to the blob detector, it looks for blobs inside the image. So anything that is at the side of the image will not be seen as a blob. Because the video is square but the actual image is round we use a mask, making that the edge of the image. In the videos the center is often to the side of the image due to the pill being shifted to the side of the intestine. This means that this method will fail for

quite a lot of frames, but will give accurate results for the frames where the center is not at the edge of the image.

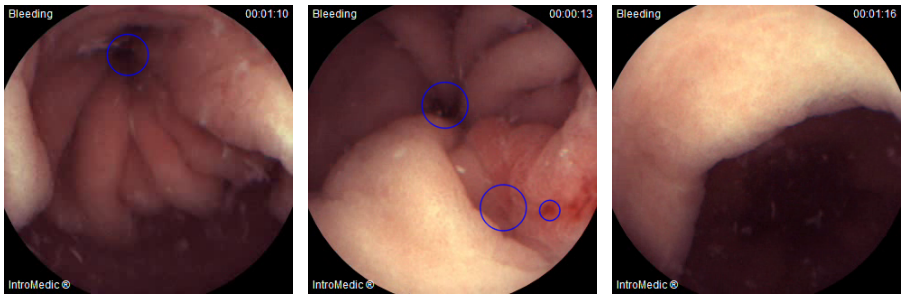


Figure 14: Left: The blob detector found a single blob, marked with a blue circle. Middle: The blob detector found multiple blobs of different sizes, all marked with blue circles. Right: The blob detector failed to detect any blobs because the (big) blob is touching the side of the mask/image.

3.2.3 Combining results

Because the blob detector can't detect blobs if the blob is touching the side of the mask and the histogram doesn't know which blob is actually the best one, we combined their results. This means that if both the segmentation techniques give out positive results we check if the point found by the blob detector is in one of the areas marked by the histogram technique. If the blob is larger than 1% of the image and they match up, the blob is far more likely to be the correct one than simply the largest one.

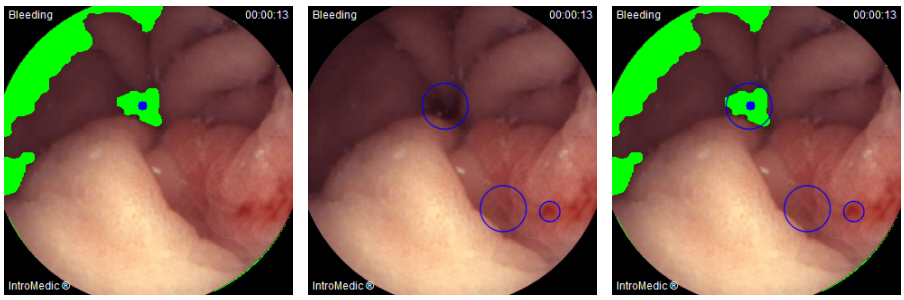


Figure 15: Left: The histogram method was used to make the sections that meet the threshold in green. Middle: The blob detector found multiple blobs, which are marked with blue circles. Right: Combining those results with the histogram results shows that the blob in the top middle was found by both segmentation algorithms. If this is the case this is very likely to be the center and is thus used instead of the biggest blob. The blue dot is the location of the middle of the blob (the size of the dot is arbitrary).

In the above frame the blob detector found multiple blobs, as did the histogram. By checking those results against each other the best blob can be selected. Because the blob detector had multiple blobs detected it would be hard to decide which one would be best without this.

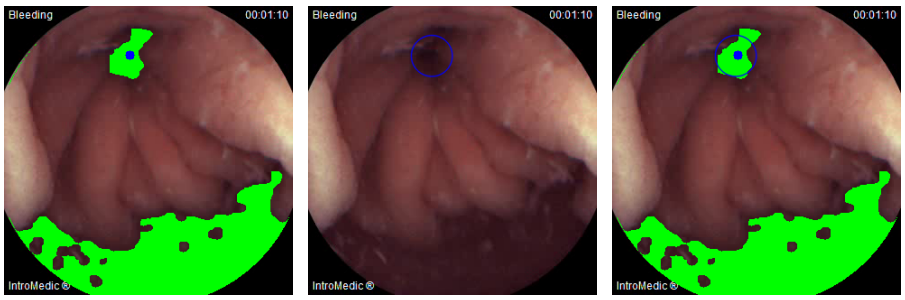


Figure 16: Left: The histogram method was used to make the sections that meet the threshold in green. Middle: The blob detector found a single blob, marked with a blue circle. Right: Combining these results with the histogram results shows that the top middle blob was found by both segmentation algorithms. If this is the case this is very likely to be the center and is thus used instead of the biggest blob. The blue dot is the location of the middle of the blob (the size of the dot is arbitrary).

The histogram will take the largest blob and would thus have selected the wrong one in this case, if it hadn't been for the blob detector. Because it's hard to tell if it's just noise in the histogram results, this makes the combination of the two much stronger. The histogram finds the big black blobs at the edges and the blob detector the smaller ones more to the middle. This means that we get the blob more to the middle as a result whenever both methods found one. It also means that if that doesn't happen but there is a big dark blob to the edge the histogram will still take that as the center. This is good because in those cases it's likely that the camera is slightly turned and the exact center is towards the edge of the frame.

3.3 Outliers

Outliers are the frames that for which the center is not detected by using the above method. There are various reasons for why this can happen. The first is that the frame is not very clear, which can be due to cluttering, contractions or because the pill is positioned in an angle that only the side of the intestine is visible. In these cases the method simply fails to find the center with the established parameters. In this case we can use linear interpolation to estimate where the center should be. But there are other cases that simply can't be classified at all too.

3.3.1 Linear interpolation

In the case that no center was detected but the frame before and after did successfully get a center detected you can use linear interpolation in order to estimate where the center is. This works really well if there is just 1 frame for which no center was detected but get's less and less accurate if more frames are missing. For this reason we use an offset parameter which can be set to how many frames can be missing before stating it's no longer reasonable to estimate the center. For the cases in which the results are within this offset interpolation is used. This is done by taking the two detected centers and measuring the distance between them. Dividing this by the amount of frames for which the center wasn't found gives the step each frame will make from the last detected center before the unsuccessful frames until the first successful frame after. This means you add the step to the center of the frame before.

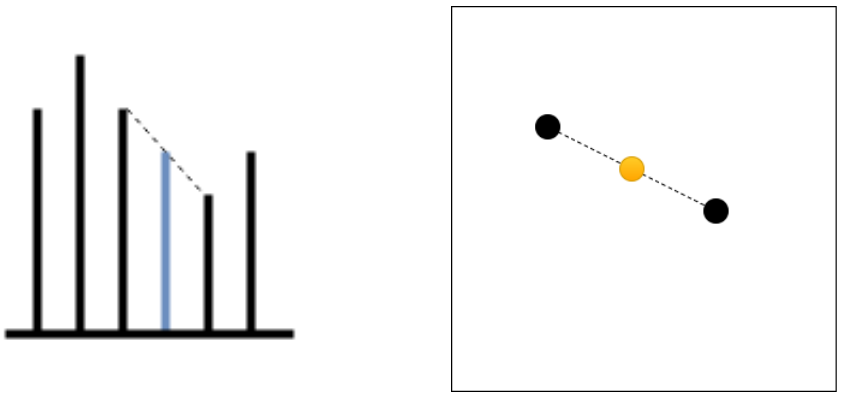


Figure 17: Left: An abstract figure to show how linear interpolation works. In this example there is a bar missing in the row of bars. By drawing a line between the two bars that are next to it you can determine what the bar should have been. Right: This works the same with the centers in images, you draw a line between the two points that are found and simply set a point in the middle of that line.

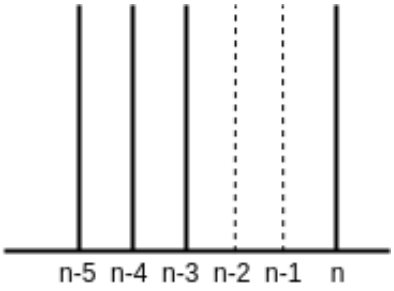


Figure 18: A sketch of the time line for linear interpolation. The solid lines represent frames that had their center successfully detected and the dashed line are the frames that needed linear interpolation. N is the frame that was just calculated and n-5 (depending on the offset 5 can be different) is the frame that is currently shown. Because frame n is the first successfully frame after frame n-3, linear interpolation is used at this point to estimate the centers for n-2 and n-1.

3.3.2 No center detection possible

There are still a lot of frames for which no center detection is possible. In most cases this is due to the camera being pressed against the side of the intestine only showing organic tissue or it floating around in too much liquid in order to see organic tissue at all. For those frames it is simply impossible to say where the center would be, no matter how you look at it. This renders the frames useless and we don't translate them.

But there are also frames for which no center can be detected because there are too many little bits and pieces floating around and obstructing the view or because the intestines are contracting in such a way there is no clear center. These frames are not necessarily useless, because you can still see useful information in them. But because there is no center for these frames, we will not be translating these ether.

3.4 Time smoothing

Because the center positions can jump around quite a bit when the camera is moving heavily, the image would jump around a lot too. In order to make the resulting image move more slowly we apply a time filter to the center positions. This is a triangular symmetric filter that that convolves the detected position signal (the positions as a function of time). So we use a tent shaped filter to determine the weights and find the new center. To adjust the n^{th} frame, we use the $n^{th} - i$ to $n^{th} + i$ frames. Giving the $n^{th} + 1$ and $n^{th} - 1$ frames a weight of 5, the $n^{th} + 2$ and $n^{th} - 2$ frames a weight of 4 and so on until you get a weight of 1 for the $n^{th} + 5$ and $n^{th} - 5$ frames. Then giving the n^{th} frame a weight of 10 and taking the average of all these centers. Using this adjusted center makes the image move much more smoothly, but at the same time causes the center to be slightly off.

3.5 Translation

Using the center positions we found we can now translate the image in such a way that the center is in the actual center of the window. In order to clearly show what happened to the frame, we draw the original location of the frame over it. We also draw a blue dot to show where the center is and a red dot to show where the center was. To make it easier to spot there is also a blue line connecting the dots, showing a find of translation vector. This is also where time smoothing makes it much more watchable and not jump around as much. If wanted the frame can also be clipped to the original size, but that causes parts of the frame to be lost.

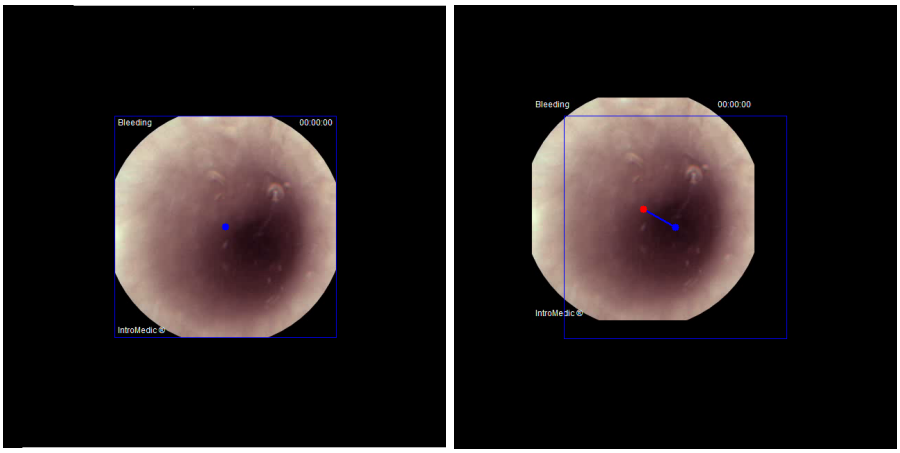


Figure 19: Left: The resulting frame if no translation would be applied and the frame is just placed in the middle. Right: The resulting frame with the translation applied, the center is now in the center of the screen.

3.6 Rotation

Rotation is wanted to show the image in such a way that it looks like the camera is not turning. So we try to counteract the rotation of the pill itself, by rotating the resulting image. In order to achieve this we tried three methods:

3.6.1 Local binary pattern brute force

A simple way to try and see what the rotation is, is to simply try them and see what works. Taking the last frame as reference and the current frame to be rotated. In order to compare them we turn them into local binary patterns. First the frames are converted to greyscale and then the local binary pattern algorithm is applied. This algorithm looks at every pixel and compares the surrounding pixels to it. For every pixel that has a higher value than the center pixel the counter goes up by 1. When all the pixels surrounding the center pixel are checked you get a value between 0 and 15 that goes into the new image. After doing this for each pixel you get a resulting local binary pattern image. This image has all the details stripped away, making it easy to compare. For the current frame you turn it by a certain number of degrees (the fewer

you turn the more precise, but also the slower) and then do the exact same. After doing this you have 2 local binary images you want to compare. Another suggested comparing these frames using the Peak signal-to-noise ratio, which is a function that tries to filter out noise match images that look almost the same. This function is defined below, but it needs "mean squared error" or MSE as well.

$$
\begin{aligned}
MSE &= \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \big[I(i,j) - K(i,j)\big]^2 \\
PSNR &= 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\
&= 20 \cdot log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\
&= 20 \cdot log_{10}\big(MAX_I\big) - 10 \cdot log_{10}\big(MSE\big)
\end{aligned}
$$

where:
m and n = dimensions of the images
I = image that is being compared, this is the un-rotated one here
K = rotated image that is being compared
i and j = iterators to go over every pixel of the images
MAX = maximal possible pixel value (this is 255 here)

3.6.2 Bounding boxes

This approach is build around the "minAreaRect" function of OpenCV. To make it work correctly I follow these steps for both images:

- find contours on binary image
- take biggest one
- get the convexHull
- use minAreaRect

And then take the difference between the 2 rotation angles.

Now what this actually means is that we first find contours (or edges) in the binary image, which is simply all the places where the color changes from black to white. After that we determine which contour has the biggest contour area and take the convex hull of it. The convex hull is basically taking all the points and putting a rubber band around it. The rubber band is where the edges go and that way you get a proper blob even when the contours don't close off a part completely. Using the "minAreaRect" function on this convex hull determines the minimum bounding rectangle for it along with the rotation if there is any. "minAreaRect" returns a "RotatedRect" object, which is just a rectangle with a rotation angle. Taking the difference of the two rotations gives the rotation that is needed for this frame.

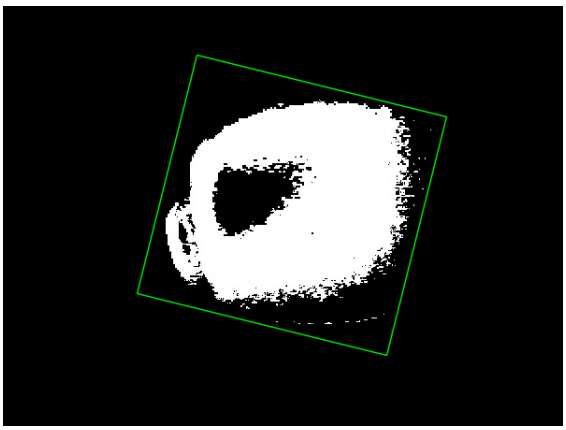


Figure 20: An example of how the bounding boxes would work. It first draws a rectangle around the object and than calculates the rotation it needs to make the bounding box horizontal.

3.6.3 Adjusted binary image brute force

This approach is very similar to the "Local binary pattern brute force" approach, but because turning the image might not get the center positions aligned and thus get a bad PSNR, we adjusted for it. Starting by translating the images, so the centers line up, which is possible because the center is already known at this point. After this we rotate the second image one degree at a time, like we did before. But because the translated images can be moved to the one of the sides of the frame completely, the image is twice as big. To compensate for this we clip both images back to the original image size (320 by 320). Now using the PSNR formula again we take the image that fits best. The angle that goes along with that image is the rotation angle that should be compensated for.

4 Results

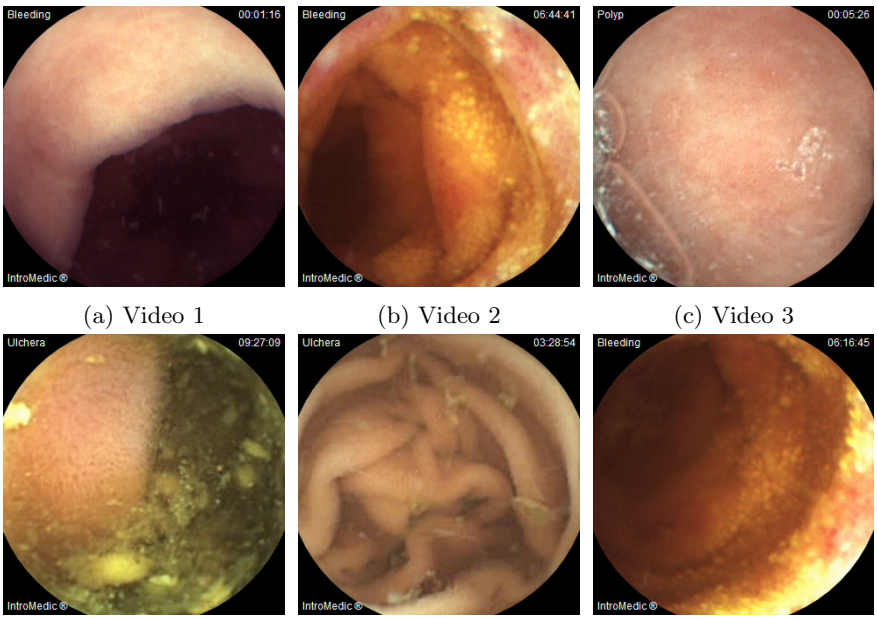
In order to test the program we made 6 video's of 5 minutes (4500 frames) each. These video's were taken from different parts of the provided video's and show different types of frames.

- Video 1: 00:00:00 - 00:05:00 of Bleeding.avi
  This video shows a wide variety of frames, with a lot of contractions. Very little cluttering of floating bits is going on, but the camera is pressed up against the sides of the intestine from time to time.
- Video 2: 00:51:00 - 00:56:00 of Bleeding.avi
  This video also shows a wide variety of frames, but has more clutter and less frames where the camera is pressed against the side of the intestine.
- Video 3: 00:00:00 - 00:05:00 of polyp.avi
  This video contains a lot of frames where the camera is pressed against the side of the intestine.
- Video 4: 01:50:00 - 01:55:00 of Ulchera.avi
  In this video there is a massive amount of little bits and pieces floating around, making it hard to see where the center is.
- Video 5: 00:40:00 - 00:45:00 of Ulchera.avi
  The pill is moving around a lot in this video, so the frames differ greatly. It's also of a part of the intestine where the sides have a lot of wrinkles, which messes with the lighting
- Video 6: 00:45:00 - 00:50:00 of Bleeding.avi
  This video has little contractions and almost no clutter, making it very easy to see where the center is with the naked eye.

Because these video's contain a certain type of frames the results are going to be related to that as well. First we ran the program on them to see how well the detection worked with the parameters we found working best during the process of writing it. This means that the histogram threshold needs a peak of 20% or higher of the maximal peak and can't be higher than the intensity 50 (out of 255). These were found by trying different values and checking by hand which gave the best results. If we turned up the max intensity possible you would get a lot more false positives (detecting a center where there is none). Turning it down would mean false negatives (not detecting a center where there actually is one). For the height of the peak we tried different values as well, if we turned it up more it would get a more robust result, but a lot more false negatives. Turning it down did the exact opposite, giving a lot of false positives. So 20% of the peak and a max intensity of 50 seems to be working the best. Running it with these parameters gave the following results:

Name	Successful center detection	no center detection	average percentage colored
Video 1	2310/4500 (51.3%)	2190/4500 (48.7%)	4.46%
Video 2	3742/4500 (83.2%)	758/4500 (16.8%)	9.00%
Video 3	335/4500 (7.4%)	4165/4500 (92.6%)	0.62%
Video 4	1183/4500 (26.3%)	3317/4500 (73.7%)	3.10%
Video 5	1348/4500 (30.0%)	3152/4500 (70.0%)	3.81%
Video 6	4440/4500 (98.7%	60/4500 (1.3%)	10.12%

What pops out right away is the percentage of successful center detection, because they differ greatly. Now this can be explained by the type of frames each video has. Video 1 has a lot of different frames, so a 50/50 result is not surprising. Video 3 has really low results, but this is easily explained by the fact that the camera is pressed against the side of the intestine, making detection impossible (See figure 14c). The fact that the cutoff for a frame is 1% colored and that the average of video 3 is below that tells me that this is a video that we simply can not classify. Video 6 however has a really high detection rate, which is due to the frames being considerably clear and with little cluttering going on.



(a) Video 1 (b) Video 2 (c) Video 3

(d) Video 4 (e) Video 5 (f) Video 6

Figure 21: An example frame from each video to give an idea of what they look like.

Looking through the first 50 frames that have no detected center of video 1, we can see some center in 4 out of 50 by hand. Now these 4 frames are of a fully contracted intestine making the black blob extremely small. Which means they probably didn't meet the 1% of the pixels colored threshold. The 4 frames do follow each other and the frames that don't have a center are grouped in sets of 3-14. This does mean that linear interpolation would solve the 4 frames that do have a center, but will likely also give some false positives.

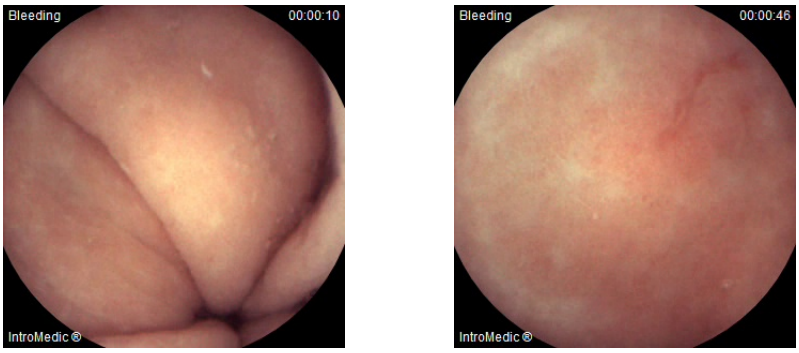


Figure 22: Left: (false-negative) A frame that does have a center, but did not have it detected. Right: (true-negative) A frame that does not have a center and also didn't get one detected

The frames that have no center detected for video 6 show very cluttered images, but we can still see a center in it. What really stands out is that there are at most 3 frames that follow each other before there being a successful frame. This means that the interpolation offset can be as low as 5 and still get a center for 100% of the frames.

Because video 3 had such bad results, we checked the first 50 frames without a center detected for it as well. Of those frames we could detect a center by hand for 7 frames all of which where of contracted intestines. While these frames were in slightly bigger groups than they where in video 6, they mostly don't need more than 5 frames. Where the frames that simply have no center to see mostly come in even bigger groups than in video 6, while there do remain some sets of frames that are 5 or lower..

This would suggest that the ideal offset for the linear interpolation is around 5, so we tested the different offsets:

With the offset set to 20:

Name	Successful frames	interpolated frames	failed frames
Video 1	2310/4500 (51.3%)	1096/4500 (24.4%)	1093/4500 (24.3%)
Video 2	3742/4500 (83.2%)	649/4500 (14.4%)	109/4500 (2.4%)
Video 3	335/4500 (7.4%)	499/4500 (11.1%)	3666/4500 (81.5%)
Video 4	1183/4500 (26.3%)	1913/4500 (42.5%)	1405/4500 (31.2%)
Video 5	1348/4500 (30.0%)	1299/4500 (28.9%)	1853/4500 (41.2%)
Video 6	4440/4500 (98.7%)	60/4500 (1.3%)	0/4500 (0%)

With the offset set to 10:

Name	Successful frames	interpolated frames	failed frames
Video 1	2310/4500 (51.3%)	709/4500 (15.8%)	1480/4500 (32.9%)
Video 2	3742/4500 (83.2%)	570/4500 (12.7%)	188/4500 (4.2%)
Video 3	335/4500 (7.4%)	280/4500 (6.2%)	3885/4500 (86.3%)
Video 4	1183/4500 (26.3%)	1102/4500 (24.5%)	2216/4500 (49.2%)
Video 5	1348/4500 (30.0%)	623/4500 (13.8%)	2529/4500 (56.2%)
Video 6	4440/4500 (98.7%)	60/4500 (1.3%)	0/4500 (0.0%)

With the offset set to 5:

Name	Successful frames	interpolated frames	failed frames
Video 1	2310/4500 (51.3%)	351/4500 (7.8%)	1838/4500 (40.8%)
Video 2	3742/4500 (83.2%)	414/4500 (9.2%)	344/4500 (7.6%)
Video 3	335/4500 (7.4%)	133/4500 (3.0%)	4032/4500 (89.6%)
Video 4	1183/4500 (26.3%)	481/4500 (10.7%)	2837/4500 (63.0%)
Video 5	1348/4500 (30.0%))	258/4500 (5.7%)	2894/4500 (64.3%)
Video 6	4440/4500 (98.7%)	60/4500 (1.3%)	0/4500 (0.0%)

In these results it's very clear that with a smaller offset the amount of frames that don't have a center goes up. Which is very logical because a lot fewer frames can have no center detected. But the real question is what these interpolated frames look like, so we looked at the last successful frame before a (set of) interpolated frame(s), the interpolated frame(s) and the first successful frame afterwards, in order to see what happens.

Doing so for video 1 with offset 5 shows a lot of interpolated frames that have their center correctly located. There are only a few frames that have a center located where there is none and also a few where the center moves around so much that the location is off.

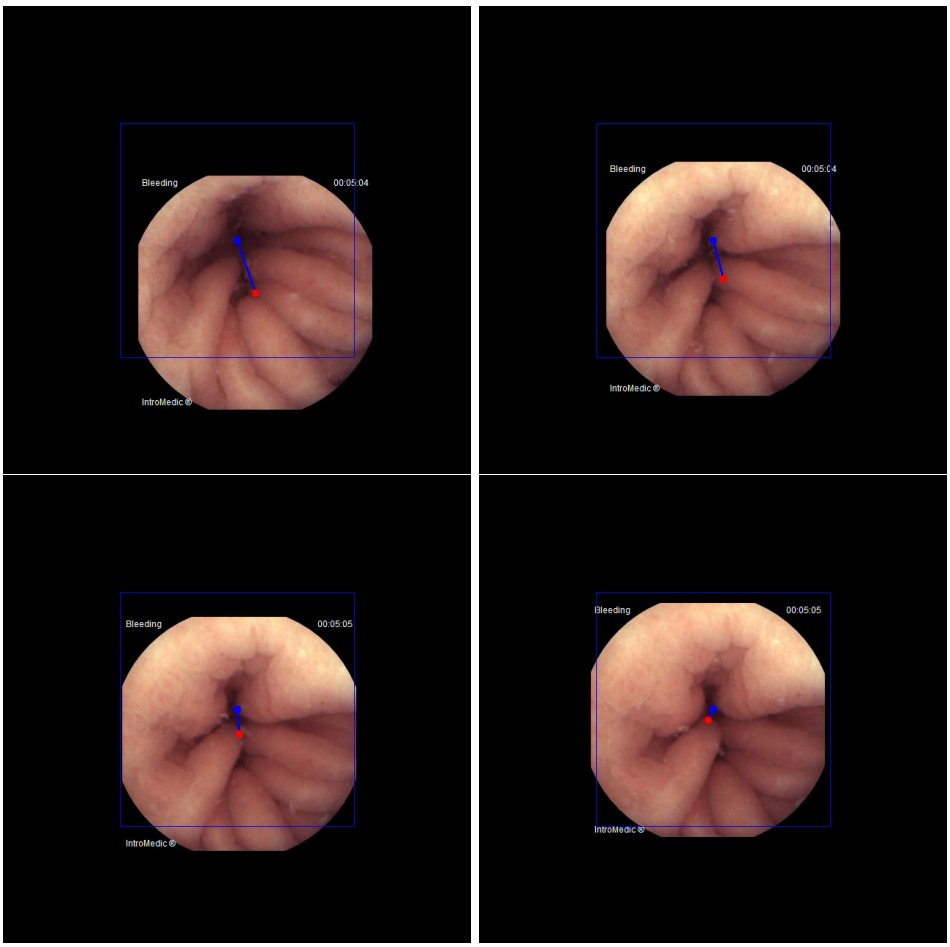


Figure 23: In the above frames you can see successful detection in the top right one and the bottom left one, the other 2 are interpolated but still show the center at the right position

.

Even if we look at the the interpolated frames with the last successful frame before and the first successful frame afterwards for video 3 (the one with bad results due to a lot of frames with just organic tissue), the results are good. Most of the interpolated frames are right on the dot. Only a few frames have no center and are interpolated over. This further shows that the offset of 5 is very good in fixing frames where there are just a few missing, without getting too many false positives.

4.1 Time Smoothing

Time smoothing is working very well and removes the frames that jump all over the screen when the camera is moving heavily. Now this does raise a new problem, because the center that was found is adjusted, causing the actual center to no longer be in the center.

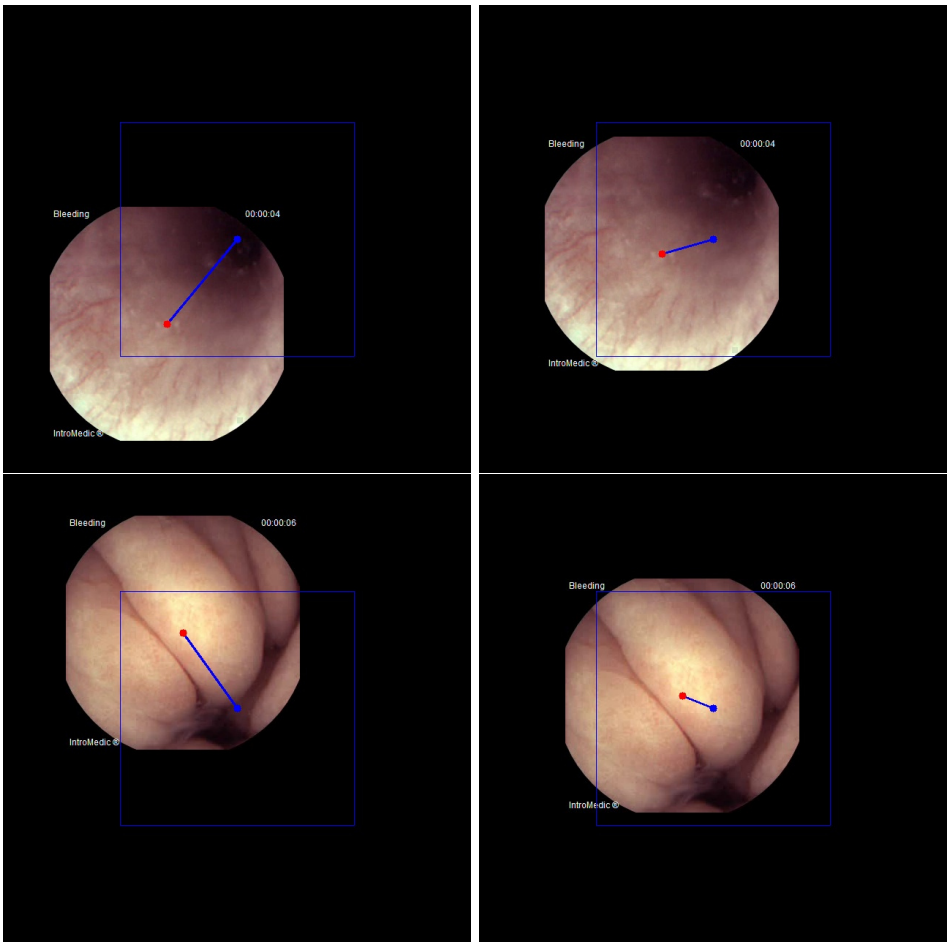


Figure 24: Left frames: Frames without time smoothing. Right frames: The same frames but with time smoothing turned on (these are extreme cases). The center is clearly off, but the images don't go that far from their original location, making the video move smoother.

Even though the images above are of extreme cases and in most cases the center isn't moved all that much it does show that this really is a matter of what is more important. Because the medical doctor will be using this it's really up to their preference and this has become an option you can turn on or off. If you want the frames to have their center position normalized as much as possible you clearly should turn it off, but running the video at higher speed makes it much easier on the eyes. Because you can turn this function off and go back in the frames it's easy to look at a still frame with the center in the right place, while having the time smoothing on when just watching it as a video.

4.2 Rotation

In order to check if the rotation is working properly we had to look at the frames by hand and judge if the frames are rotated correctly. Starting off with the "Local binary pattern brute force" method. This method doesn't give the wanted results at all, all rotations applied are very minor (less than 10 degrees) and don't really seem to help making the video easier to watch. The method is also very slow due to the brute forcing and thus checking 360 degrees. Which breaks the requirement of it being able to compute the frames in real time. Changing the method so it does not check every degree but checks every x degrees, makes it faster the bigger the x. Because the rotation is not giving the wanted result with the most accurate method, this doesn't make the results any better, but slightly worse. Now on to the second method, the "Bounding boxes". This method easily fast enough, but gives really random result. The frames do rotate now, but they are jumping around a lot and most of the time it's not the rotation that should be happening.

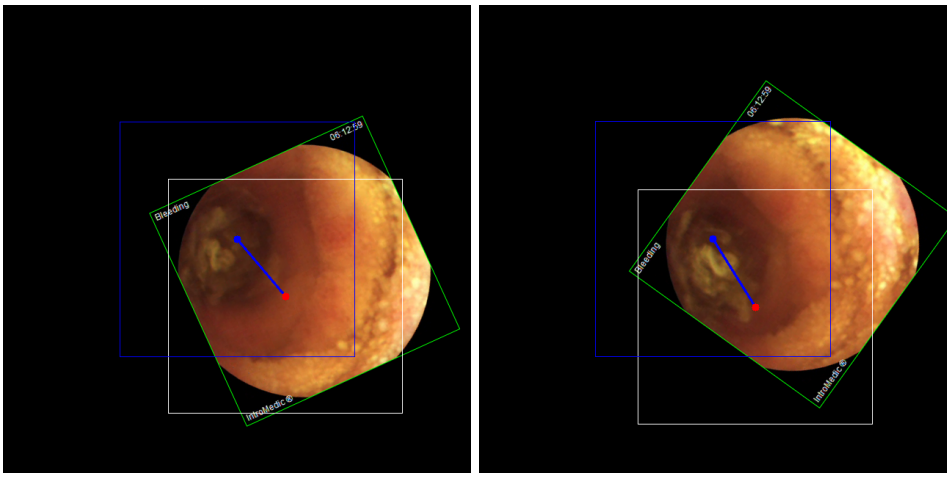


Figure 25: An example of 2 frames, there is quite a bit of rotation visible, but it is clearly not necessary in this case. The blue square is the original location of the frame, the white one is the location it would have without the rotation but after translating the image. The green square is to clarify the location it has now.

At this point we realized our mistake with the first method, if the centers are not aligned the rotation will always fit best when the center is at the place it was originally. This means that the rotation will almost always be 0 or very near to it. Using the "Adjusted binary image brute force" method I do take this into account and that shows in the results. This method gives much better results and is much more stable, but still has frames that jump around due to weird rotations. The problem here is that it's still brute forcing, so it's really slow. Changing the degrees turned for each step up makes it possible to run it at nearly real time, but also gives more frames that rotate incorrectly.

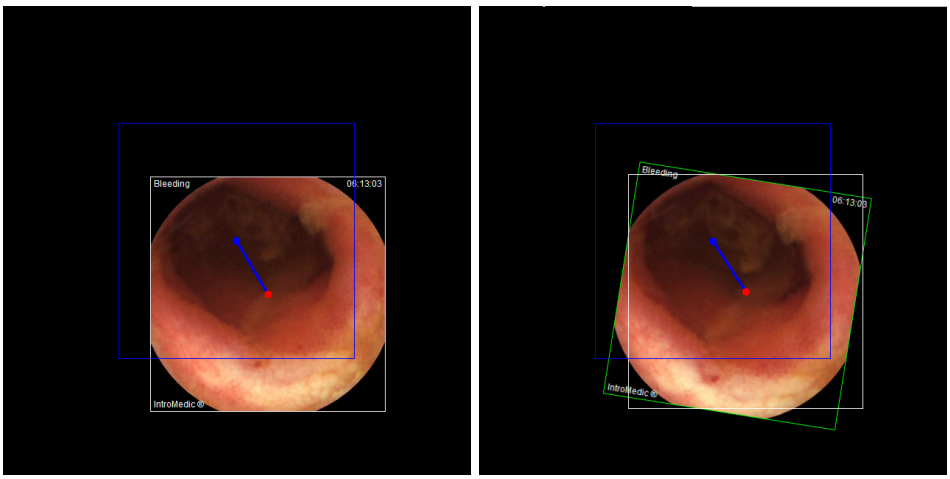


Figure 26: An example of 2 frames where the rotation is working correctly. The downside is that when no rotation is applied it isn't looking that much worse. The blue square is the original location of the frame, the white one is the location it would have without the rotation but after translating the image. The green square is to clarify the location it has now

4.3 User interaction

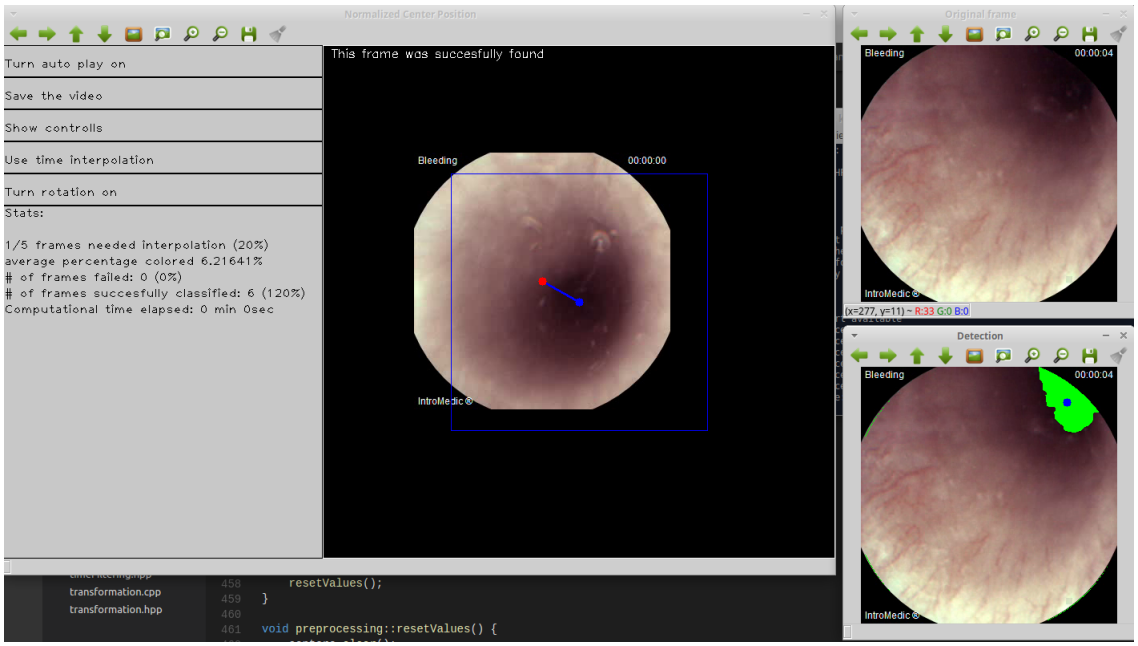


Figure 27: What the program looks like at runtime. The big window on the left is the main window for the program. On the top left half it has buttons with various options that can be set. Below it are the statistics, but the controls could also be shown there. To the right is the resulting image after processing the raw image with the options set at the time. The two windows on the right can show different intermediate results. In this case the top one is the raw image and the bottom one is the raw image with a green section colored, this is the section the histogram segmentation method has found. The blue dot represents the center of this section (the size of the dot is arbitrary).

Because there are so many different things that the medical doctor might want to use or not use there was a need for a user interface. In this interface the user can set the following options by pressing one of the buttons on the upper left half of the screen:

- Turing auto play on or off
- Saving the video or not
- Show the controls or the statistics
- Turing time smoothing on or off
- Turing rotation on or off

All of these feature can be used while running the program, but there are more functions that are not in this menu, that can be changed. To name a couple that a user might want to use: which of the three rotation methods should be used, what the offset should be or which frames should be shown next to the main window. Most of the other features are aimed at debugging and testing.

5 Conclusions

5.1 Discussion

The goal of this bachelor project is to make the raw video produced by endoscopy pill camera's easier to watch for a medical doctor. This meant that the center position had to be normalized and the video should be able to run smoothly.

The images in the thesis have shown that center position normalization is very possible for certain frames. This took both segmentation methods to be combined, but can still run in real time. But there are also frames that are simply impossible to normalize because there is no definitive center in the image. These frames are hard to classify because there are also frames for which the method fails. For most of the frames where the method fails linear interpolation can estimate their center.The biggest problem here is that you need a human to classify each frame in ether a frame with a center or one that does not have one. Only that way you can have really exact statistics about the success rate of the method.

The video does run smoothly when the time smoothing is turned on, but this also means that the center position normalization can be off. This is not what the aim was, but since the image center does jump around quite a lot there is no way to meet both requirements simultaneous perfectly. Because of this this is a feature that can be turned on or off, depending on what the user wants the focus to be on.

Another requirement was that the program should run (nearly) automatic, which has successfully been accomplished. The nearly part being that the user does have some choices to make about which features he want's to use. The important features can be changed at run time using the user interface, but some of them have to be changed in the code.

The program can run efficiently enough to be able to run real time, but can also output a video if wanted. If one of the two brute force rotation methods is turned on however it can't run real time anymore.

5.2 Future works

Benchmark for which images can be classified: The results in the different video's we made are very different, which is caused by the type of images. Now to be able to tell how well my method works we would need to have a human look at every frame and determine if the frame actually has a center to be found or not. Using a benchmark like that would enable me to tweak the program in such a way that the real amount of frames it fails to find a center in goes down.

Optimization: There are two segmentation methods used to determine the center in an image, along with a lot of intermediate steps that makes a lot of passes over each image. The two rotation methods that use brute force slow the program down immensely. You could look into sub-sampling the image by making it smaller. But the way it is right now is of course far from optimal and even without using a completely different segmentation method there is room for optimization.

Rotation: The rotation is working for some cases with the "adjusted binary image brute force" method. The other two methods have more wrong rotations that correct ones. This means that there is still a lot of work to be done to get the correct rotation. There are more algorithms available that could be tested for this purpose. It's also a good thing to hold a survey under the users of the program to see if the rotation is actually making the video easier to watch. Because the rotation is making the images move around more the image isn't as steady as without the rotation. When inspecting all the sides of the intestines the rotation might not be as important as how easy it is to watch the video.

Advanced image segmentation: As discussed in the related works section there are a lot of advanced image segmentation algorithms that could be used. The problem being that they are much more computationally costly. In order to know if that would be worth it a whole set of them should be implemented and tested for time efficiency compared to accuracy. This would require a lot of work and is more complicated and is therefore out of the scope of this project, but these algorithms might give better results.

User interface: Right now only the most important functions are available in the user interface and the others have to be changed in the code. A proper user interface having all these functions would be really helpful. This user interface was written in OpenCV, which is an image processing library, so adding a GUI library and making the user interface with that would make it much easier to use for medical doctors.

References

[1] Melissa F Hale, Reena Sidhu, Mark E McAlindon, Capsule endoscopy: Current practice and future directions, World J Gastroenterol 20(24): 7752-7759, 28 June 2014

[2] Zeno Albisser, Computer-Aided Screening of Capsule Endoscopy Videos, University of Oslo, Master Thesis, Autum 2015

[3] Anastasios Koulaouzidis, Dimitris K Iakovidis, Alexandros Karargyris & John N Plevris, Optimizing lesion detection in small-bowel capsule endoscopy: from present problems to future solutions, ISSN: 1747-4124 (Print) 1747-4132 (Online) Journal homepage: http://www.tandfonline.com/loi/ierh20, 28 August 2014

[4] Xenophon Zabulis, Antonis A. Argyros and Dimitris P. Tsakiris, Lumen detection for capsule endoscopy, International Conference on Intelligent Robots and Systems, 22-26 September 2008

[5] J. Bulat, K. Duda, M. Duplaga, R. Fraczek, A. Skalski, M. Socha, P. Turcza, T. P. Zielinski, Member, IEEE, Data Processing Tasks in Wireless GI Endoscopy: Image-Based Capsule Localization & Navigation and Video Compression, Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference 2007; 2007: 2815-8, August 23-26, 2007.

[6] Sonu Sainju, Francis M. Bui, Khan Wahid, Bleeding detection in wireless capsule endoscopy based on color features from histogram probability, Canadian Conference on Electrical and Computer Engineering, 2 August 2013

[7] Tonmoy Ghosh, Shaikh Anowarul Fattah and Khan Arif Wahid, Automatic Bleeding Detection in Wireless Capsule Endoscopy Based on RGB Pixel Intensity Ratio, 2014 International Conference onElectrical Engineering and Information & Communication Technology (ICEEICT), 9 October 2014

[8] Satya Mallick, Blob Detection Using OpenCV (Python, C++), learnOpenCV.com, Februari 17, 2015

[9] Enrique Alegre, Introduction to Intelligent Systems, Morphological Image Processing, Rijksuniversiteit Groningen, 2016

[10] Enrique Alegre, Introduction to Intelligent Systems, Edge Detection, Rijksuniversiteit Groningen, 2016

[11] http://opencv.org/

[12] ASGE Technology Committee, Wireless capsule endoscopy, Elsevier Volume 78, Issue 6, December 2013, Pages 805-815.

[13] Cheng, Yizong , Mean Shift, Mode Seeking, and Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE. 17 (8): 790–799. August 1995.

[14] Loubna Bouarfa, Oytun Akman, Armin Schneider, Pieter P. Jonker and Jenny Dankelman, In-vivo real-time tracking of surgical instruments in endoscopic video, Minimally Invasive Therapy and Allied Technologies, 21:3, 129-134 (2012).

[15] Daniel DeMenthon, Spatio-temporal segmentation of video by hierarchical mean shift analysis,

[16] Jan Kybic, Jiří Matas, Lecture for Digital image processing / Medical Imaging Systems 1, Department of Cybernetics, Faculty of Electrical Engineering of Czech Technical University of Prague, 2007.