Real-Time Texture Analysis for Esophagogastroduodenoscopic Exams

Stefano Donne, *MA1 Student, Ecole Polytechnique de Bruxelles (ULB)*

Abstract—This project has been realized in the context of the PROJ-H402 course. Its goal is to answer to the problematic raised by the service of *Hépato-Gastro-Entérologie* of CHU St-Pierre in the scope of a clinical trial. The study consist in the evaluation of a protocol to improve patient preparation for gastroscopic exam ; more precisely to reduce the amount of visual pollution induced by foam and saliva in the patients upper gastrointestinal (GI) tract. The clinicians need is to have a tool analysing automatically and in real-time the video flux of the exam, outputting a relevant score concerning the level of visual pollution. An empirical approach has been taken with several iterations, tests and retries. A Python script was developed, using conventional Image Processing techniques but also a Deep Neural Network (DNN) classification model. The outcome of this work is a prototype embedded on a Raspberry Pi 4 which is currently tested, alongside solutions for further improvement.

1 Introduction

The randomised double blind medical study led at CHU St-Pierre aims to evaluate the effect of a premedication with Simethicone versus placebo. The study evaluates the quality of visualisation of the gastric mucosa in each Esophagogastroduodenoscopy (EGD). The Simethicone is supposed to improve [1] the exam quality by reducing the amount of pollution induced by foam or saliva.
The EGD visual quality is assessed by a qualitative score resulting from the physician observation. The purpose of this project is to assist the physician in this task, by developing a solution to automatically detect in real-time the pollution in the EGD video and indicate its quality to the physician with a quantitative score. The project is realised inside the scope of this study but could be developed beyond if the results are encouraging.
The organisation of this paper follows the steps taken in the project realization. The current paper describes the methodology, starting with the development of a Python script to detect pollution on individual images with Image Processing methods. Then find a solution to output a score for each anatomical sequence of the EGD. The script performances were optimized for real-time video input and a prototype has been realized as 'proof of concept' with some simplifications.

2 Image Treatment

The processing of the frames follows several steps in order to extract relevant informations from them. The first step is to determine if a frame is acceptable or not. A non accepted frame can be : a blurred frame, a frame obstructed by liquid or a frame too close to the wall.
If the frame is acceptable, it will be processed to detect the presence of pollution (segmentation). The output is a binary image (mask) where each white pixel corresponds to pollution.
A score can be extracted from the acceptable segmented image based on the detected pixels density.

2.1 Uniformity

The solution for frame filtering follows this intuition : an unacceptable frame will look blurry or even uniform in the case of obstruction. In opposition, an accepted frame will possess more variations and details.
This can be translated as this : the spectrum of an accepted frame will lie in higher frequencies because the rate at which intensity is changing in the spatial domain is higher.
The explored solutions to extract this feature are the FFT (Fast Fourier Transform) [2], Laplacian Filter + Variance [3], Entropy and finally the Difference of Gaussian which is the selected approach.
The method consists in substracting one Gaussian blurred version of the image with a less blurred one. This acts similarly to a high-pass filter. A single score is extracted by taking the sum of the square difference for every pixel, then dividing it by the dimensions of the image to obtain an intensive value. The higher the score, the higher the difference between the two blurred images, meaning that the image possesses a higher level of details that are disappearing with the blur. A threshold has been arbitrary determined through observation and set at *20*. A frame where the measured uniformity score is above this threshold is considered as acceptable for treatment.
A comparison between a frame affected by motion blur and an acceptable frame is displayed in Fig. 1. (Obstruction example in *Appendix A*)

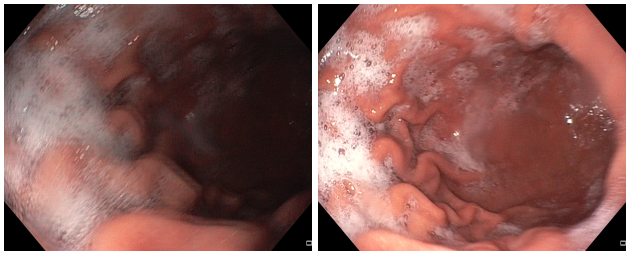

Fig. 1: Uniformity Comparison (left : 15.67 — right : 40.47)

2.2 HSV Segmentation

The segmentation is a process consisting in isolating regions of the image, based on the properties of these areas. The pollution (foam) possesses some characteristics which makes it easily segmentable in the HSV color-space (Hue, Saturation, Value).
The most important criterion is the saturation. Because it mostly reflects light, the pollution is in average way less saturated than its surrounding. Fig. 2 is an illustration of this observation. The foam saturation values are around the little spike at sat=50 in the histogram. (More illustrations in *Appendix B*).
The Hue and Value criteria are mainly used to avoid miss-detection of flares, liquid and during blue light imaging.

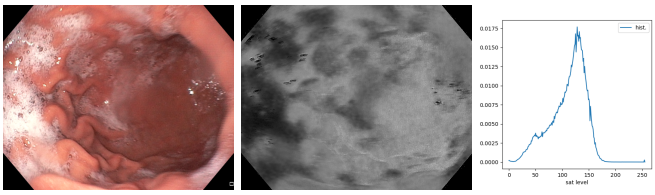

Fig. 2: Foam Saturation

The nature of the pollution is summarized by :

1) under direct light : great hue-range, low-medium saturation, high value
2) low light : hue range around red values, low saturation, low-medium value

Examples of HSV values for various images areas are given in *Appendix C*.
Two segmentation masks are created based on those criteria, which added together result in a single output as in Fig. 3. This is a binary mask, white pixel = pollution.

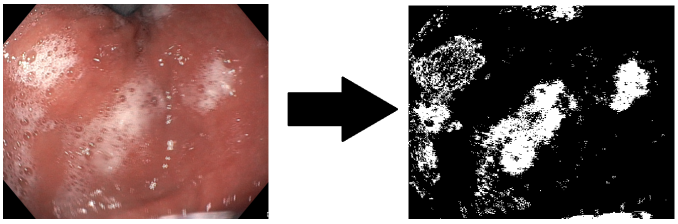

Fig. 3: HSV Segmentation Mask

2.3 Morphological Transforms

The result of the HSV segmentation isn't perfect. The polluted areas aren't continuous and some false positive pixels caused by flares appear.
The solution to refine the segmentation is to use Morphological Transforms [4]. The operations used to enhance the detection are *CLOSE* and *OPEN*. The *CLOSE* operation is used to fill the gaps inside polluted areas. The *OPEN* operation 'de-noise' the result by removing isolated and false-positive pixels. Fig. 4 illustrate the capacity of the transforms to improve the result. More examples in *Appendix D*.

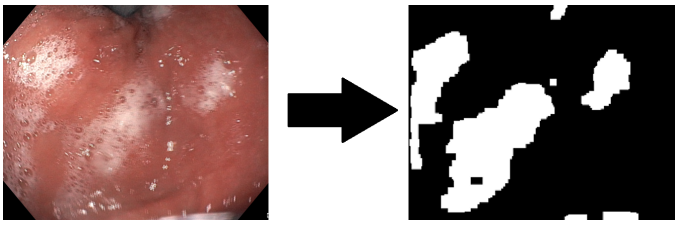

Fig. 4: Enhanced Detection

2.4 Scoring

The resulting enhanced masks were translated into a quantitative score. The chosen metric is the *pollution density*. Its determination only consists in counting the number of white pixel detected and to divide this amount by the dimension of the image. The measured densities can be averaged on a succession of frames and used to describe the pollution level of a certain sequence. The sequencing of the score is detailed in sections *3* and *4*.

3 Esophagogastroduodenoscopy (EGD)

3.1 Exam Conduct

The EGD is the most common practiced exam to diagnose upper GI tract pathologies. It is done by inserting the endoscope into the patient's mouth and allows the physician to have visual information about the visited areas: esophagus, stomach and duodenum. It usually lasts a few minutes and can be done in ambulatory.

3.2 Scoring Standard

The EGD visual quality assessment done in the study is based on the Total Mucosal Visibility Score [5] and is applied on five anatomical sequences : oesogastric junction, gastric body, antrum, angle and fundus (see Fig. 5). This score can be Excellent (1), Adequate (2) and Inadequate (3). The addition of these metrics gives a total score on 15 points for the whole video. The video is considered as Adequate if the added score is strictly inferior to 7, meaning that at most one sequence is polluted. A translation from the quantitative script output (*pollution density*) to this standard can be done with 2 thresholds to partition the densities level into the three required classes.

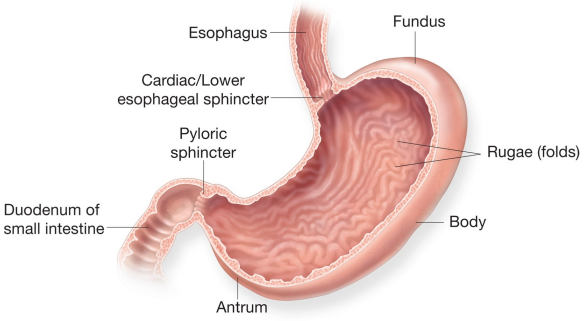

Fig. 5: Stomach anatomy [6]

3.3 Sequencing Issue

In order to translate the quantitative score into a meaningful output for the physician it is first needed to split it into sequences. It is indeed possible to average the score of following frames and take this metric to characterize the pollution in this whole set of frames.
But the 5 sequences evaluated in each EDG correspond to some precise anatomical sites. Separate them is not a trivial task. The first (naive) approach is based on the fact that pictures are taken on those sites. It is possible to detect when a picture is taken and therefore switch to the next score sequence at this moment. But this method didn't pass the trial of experimentation, the pictures taken aren't always on the same spot and sometimes pictures are taken inside a sequence because of the physician needs (e.g. to document anomalies).

4 DNN Solution

A more robust and smart approach is needed to obtain a proper anatomical sequencing of the score. It was decided to implement a DNN solution and train by transfer-learning a model to classify pictures of the upper GI tract. The data-set fell short of a few classes in order to be able to fully solve the score sequencing problem. However, training on this data-set and evaluate on the CHU St-Pierre study images will be relevant to know if this solution is feasible.

4.1 Data-Set

The data used for training have been gathered at Bærum Hospital in Norway and published as the *Hyper Kvasir* data-set [8]. This data-set is a collection of 10,662 labeled and 99,417 unlabeled images from the GI tract. Of course, only the labelled image from the upper GI tract are relevant to solve the sequencing problem. It represents in total 2764 images, split in 3 anatomical landmark classes : *z-line* (oesogastric junction), *retroflex-stomach* (fundus) and *pylorus*. In Fig. 6 is displayed an example for each class.
From the study of CHU St-Pierre, 525 images have been retrieved from the EDGs. Those images have been classified from the anatomical landmarks they represent, 291 images correspond to the 3 classes labelled in *Hyper Kvasir*.

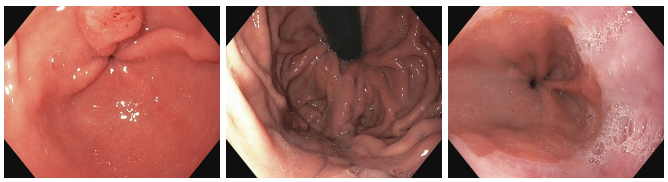

Fig. 6: pylorus, retroflex and z-line images

4.2 Model

The chosen model is *MobileNetV2*. [9] [10] It is designed to perform on visual recognition problems, including classification. One of its main features is that this model is lightweight and relatively not too resources-consuming, having 2,261,827 parameters. [11] It is optimized for mobile applications and is released alongside Tensorflow-Slim library. For this reason, this model seems fit for fast development and embedded integration.
The model used is pre-trained on the ImageNet data-set (14,197,122 images), achieving a Top-1 Accuracy of 74.7% on this set for classification. [11]
The input layer takes batches of 32 RGB images of size 160x160. The output layer is of size [3x1] and consists of 3 neurons, 1 for each class considered. Their levels of activation determine which class is predicted by the model, the highest one is selected. The used Loss function is the *Sparse Categorical Crossentropy*, which is the recommendation for multi-label classification.

4.3 Training and Accuracy

For training purpose, a data-augmentation layer is added on top of the model with the following transformations: random rotation, horizontal random-flip and vertical random-flip. This technique is used to artificially increase the diversity of the input data.
For the transfer-learning of this model [12], the training set is the 2764 labelled images from *Hyper Kvasir*, minus 10% of the set which is kept for evaluation purpose. For each training epoch, this evaluation set allows to track the evolution of the model accuracy.
In parallel of the training, after each epoch the model is also evaluated on the 291 images of the CHU St-Pierre study set. The idea is to keep these images away from the training to avoid any bias at this point. Indeed, if without training on this set the model shows good prediction results on them, it is a proof of a certain robustness from this DNN solution.
Both set were evaluated before training. The accuracy values obtained are 0.3079 and 0.3397 respectively for *Hyper Kvasir* and the study set, or about $1/3$ which is the expected value for the untrained model.

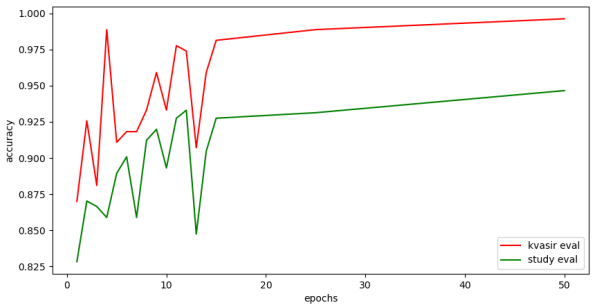

Fig. 7: Model learning curve

On Fig. 7 is displayed the accuracy evolution for both set, for the 15 first epochs then the 25th and 50th.
However, it was decided to stop the training at 10 epochs in order to avoid over-fitting the model. A confusion matrix is used to assess the performances of the trained model on the study set (*Table 1*).

TABLE 1: Confusion Matrix on the study set

	Real Pylorus	Real R-Flex	Real Z-Line	TOTAL
Pred. Pylorus	32.2%	1.7%	1.4%	35.6%
Pred. R-Flex	0.34%	25%	0%	25.34%
Pred. Z-Line	2.7%	4.8%	31.5%	39%
TOTAL	35.2%	31.5%	32.9%	

From this matrix, multiple observations can be made. The Top-1 accuracy on the study-set is given by the sum of the diagonal of the matrix. Its value is 88.7%. A comparison per input class gives a more in-depth result. When given a Z-Line image as input, the model output the correct prediction 95.7% of the time. The score for the Pylorus is 91.4%, but fall to 79.4% for the Retroflex-stomach images.
Those results are close to what has been obtained on an other similar classification problem [15].
An other insightful metric to assess the model classification performances is the *Cohen's kappa coefficient* (κ). This metric indicates how far the model performs from a random classification (1/3). A value < 0 is indicating no agreement , 0–0.20 as slight, 0.21–0.40 as fair, 0.41–0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1 as almost perfect agreement. [16]
A value of $\kappa = 0.8305$ is obtained from the *Table 1*. These results allow to conclude that this proof of concept of DNN solution can be validated and further improved.

5 Code Overview and Execution Pipeline

Python is the chosen language to develop the solution, the main motivations being the speed of development and the availability of many Image Processing and DNN libraries (e.g. OpenCV, Keras/Tensorflow, ...).
The implementation is presenting itself in 3 Python files : *Main.py* which starts the script and implements the fetching and display of the frames, *Treatment_Process.py* which apply the necessary transformation and detection functions, finally *Score_Wrapper.py* which implements methods for managing and outputting the score and other relevant information. (whole code on GitHub [7])
Parallelism is implemented through multi-threading and multi-processing in order to avoid strong bottleneck during run time, as illustrated in Fig. 8.

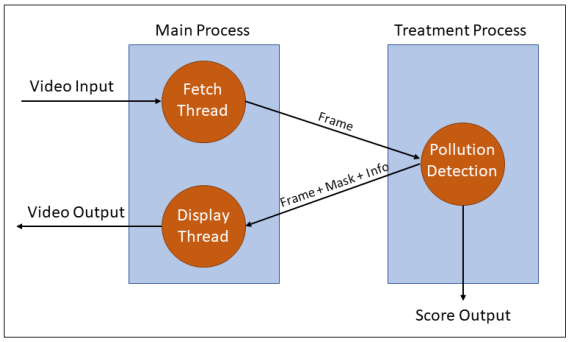

Fig. 8: Pipeline

Threads communicate between themselves through two FIFO Queues : one for fetched frames waiting for treatment and one for treated frames waiting for display. The treatment related tasks are embedded in their own process in order to optimize CPU usage for these high consuming tasks.
Fig. 9 displays a simplified decision tree. It represents the flow of actions taken for each frame in order to have the desired output. What isn't explicit in this tree is how the score is managed.
Each time an acceptable frame is treated, a pollution score is retrieved from. This score is first added to a buffer. This buffer is checked each time a frame is not accepted, the buffered values are discarded if the number of values is small (< 10). It allows to limit the occurrence of false positive accepted frames; a frame is definitely accepted only and only if it is part of a set of other successive accepted frames.
In the other case, those values are added to the list of current sequence score. When a sequence change occurs, the average value of this list is outputted. This output characterize the average level of pollution measured in the set of acceptable frames taken during this sequence.

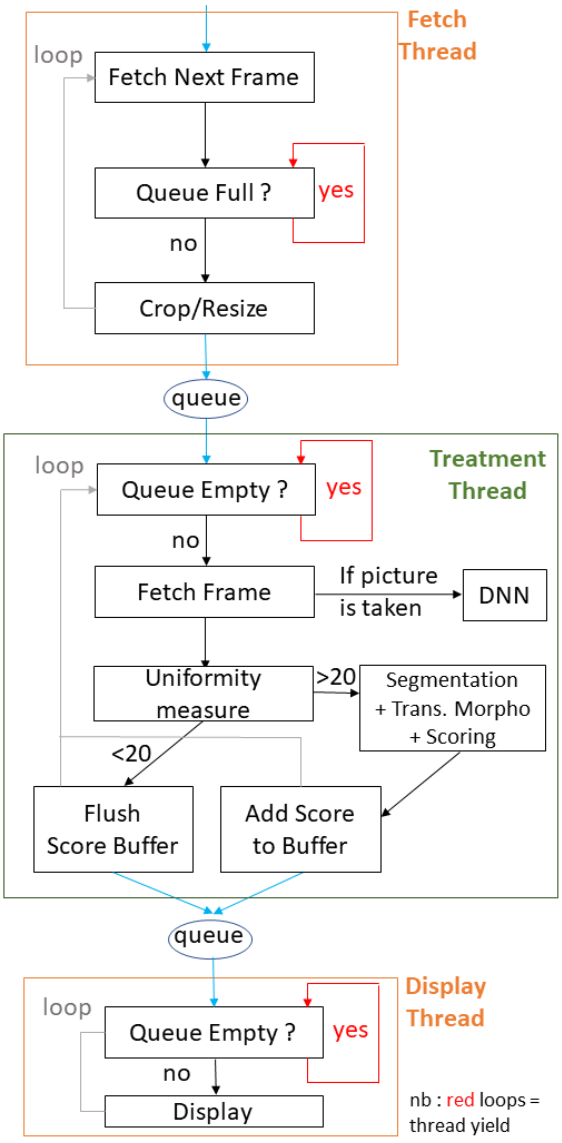

Fig. 9: Decision Tree

6 Hardware Implementation

It was decided to integrate the solution into an embedded system in order to have a prototype that can be used seamlessly without modifying the EDG acquisition software. A benchmark of the Python Script on the RPI-4 is done to assess the embedded solution performances.

6.1 Set-Up

The embedded solution is added onto the existing hardware. It samples the video flux from the SDI cable, without modifying it.

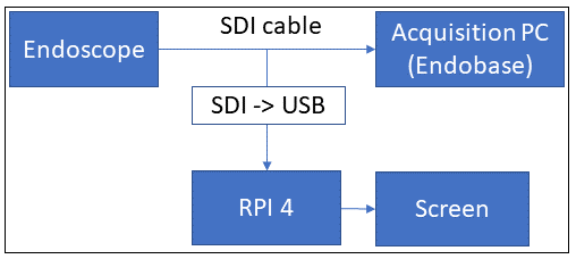

Fig. 10: Set-Up

The RPI-4 is bundled with a touchscreen that serves for basic interaction with the operator (start/stop).

6.2 Performances

The Raspberry Pi 4 (RPI-4) is equipped with a Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz and 4GB of RAM (DDR4). [17] Three performances tests are executed to detail the performances of the implementation on the RPI-4 versus laptop (machine of reference). The laptop is equiped with an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz (8 CPUs) and 8GB of RAM (DDR4).
The first benchmark consists in measuring the average time taken to only fetch, crop/resize and display frames from a video input. The second benchmark measures the average time taken to apply treatment to video input frames (only libraries function calls). A comparison is done on the second benchmark between the performance on full input size (1080x1080) and resized (216x216). The third benchmark is done on the whole python script, giving the average frame per second count on a run. The fps count must be above 24 fps for real-time purpose, because the video input is at this rate.
For each benchmark the result is taken from the average of 3 runs. The *Table 2* gives the benchmark results.

TABLE 2: Benchmarks results

	Benchmark 1	Benchmark 2 (1080)	Benchmark 2 (216)	Benchmark 3 (216)
Laptop	16.64 ms	3.2 ms	0.16 ms	60.17 fps
RPI-4	33.2 ms	14.4 ms	2.76 ms	26.8 fps

The Fig. 11 gives a more detailed insight of the third benchmark. It is visible that the frame count is uneven with an upper bound around 50+ fps and a lower bound around 16 fps which cause an unstable throughput, which isn't desired for real-time purpose. A solution to improve the frame-rate on the RPI-4 without reducing even more the resolution consist in skipping a certain amount of frames.

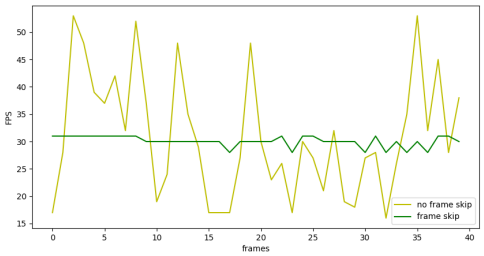

Fig. 11: Frame-Rate variations

The result is displayed in the same figure and shows a more stable frame-rate. One frame out of two is discarded. However, the impact of this frame skipping on the scoring hasn't been assessed yet.

7 Improvements

The work described in the previous sections is subject to improvements. Several propositions are made in order to move from a proof of concept to a fully functional implementation in the future.

7.1 Classification

The current classification model based on *MobileNetV2* showed good results. However, it is not yet able to fully solve the sequencing problem.
The data-set needs two more anatomical sites : Gastric Body and Angle, plus a collection of various EDG images to detect when a picture is taken off site.
The performances of the model itself can be improved with some changes. First, move to the newer *MobileNetV3* model, which showed better Top-1 accuracy on *ImageNet* data-set, but also a lower latency. [19] Then, its training can be enhanced following a few modifications :

1) More variation in the data augmentation layer
2) Increase image input size
3) Implements a cyclic learning rate for faster training and potential higher accuracy cap [13]

7.2 Pollution Detection

The pollution detection based on image colorimetry, as described in *section 2*, works most of the time but isn't robust enough. It is very sensible to hue and saturation levels because of the pre-determined thresholds. It is giving less accurate results in some cases, in the oesophagus section for example (see *Appendix E*).
Basic texture segmentation methods have been tried to increase the algorithm accuracy. But the measured average treatment time per frame (resized) is about 300 ms on laptop, almost 100 time slower than the HSV method, making it unsuitable for real-time treatment.
But a more ambitious approach based on DNN can be tried. First, extracting from the study EDG videos a data-set formed from valid frames and their corresponding segmentation mask outputted by the HSV method. Then, training a segmentation DNN model on this set.

Data augmentation is the key idea to obtain better result than the current method. It is possible to modify the colorimetry of the input images in order to simulate the cases where the HSV method doesn't perform as well. The aimed goal is to obtain a model that perform as close as possible as the HSV method on standard colorimetry and way better in other cases. Different real-time implementations using *MobileNetV3* or other models as *U-NET* already exist and serve as proof of feasibility for this method. [20] [21]

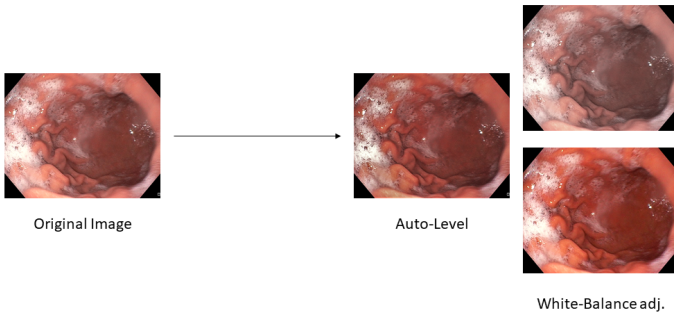

Fig. 12: Example of data-augmentation

8 Conclusion

The purpose of this project was to develop a tool to automatically detect in real-time the pollution in the EGD video and indicate its quality to the physician with a quantitative score. This goal has been partially met.

An image treatment algorithm has been developed. Based on classic image processing methods, giving relatively accurate results. A DNN serving as proof-of-concept has been implemented to separate the outputted score into anatomical sections. This work has been embedded into a RPI-4, which is the tool prototype.

Finally, some improvements have been discussed. They are backed by already existing works and might drastically improve the solution effectiveness.

Appendix A

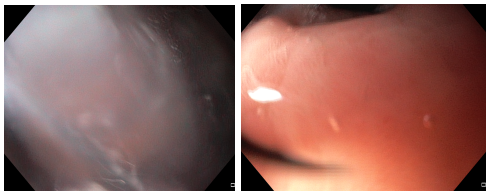

Fig. 13: Uniformity Comparison (left : 10.79 — right : 12.2)

Appendix B

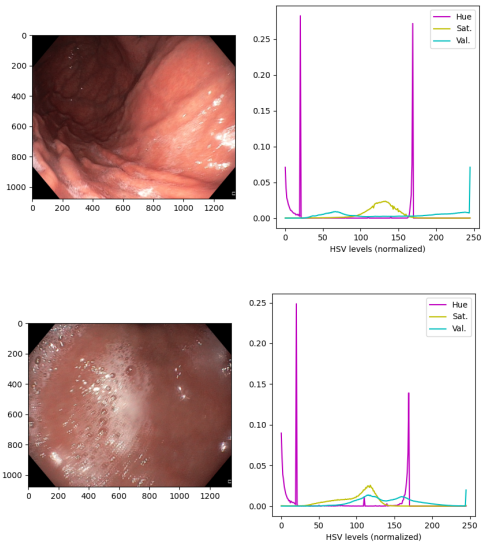

Fig. 14: Two frames alongside their HSV histogram

APPENDIX C

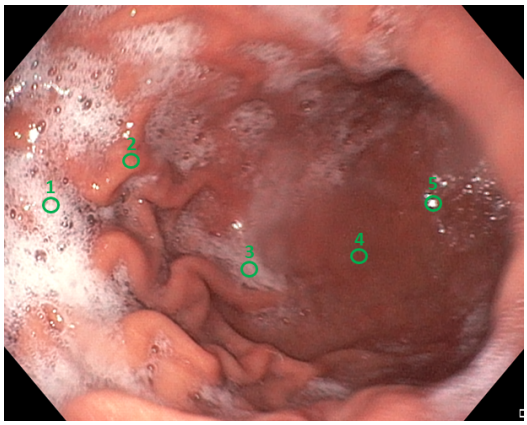

Fig. 15: Example areas for HSV values

TABLE 3: HSV Values for given areas (range 0 : 255)

	Hue	Saturation	Value
1	6.9	38.25	250
2	6.8	135.15	214
3	0.2	71.5	163
4	3.6	153	120
5	42.5	0	255

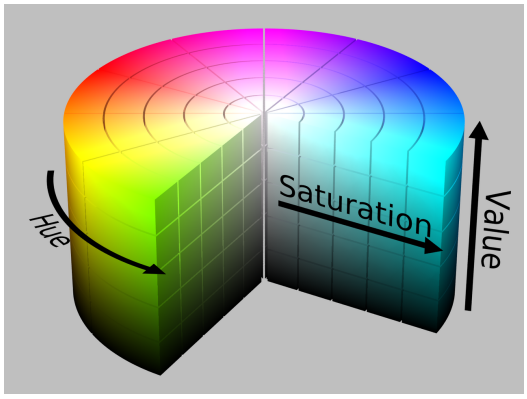

Fig. 16: HSV Space representation (context) [14]

APPENDIX D

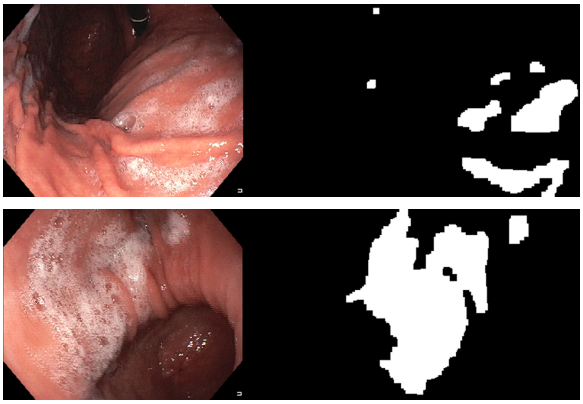

Fig. 17: Illustrations of pollution detection (A)

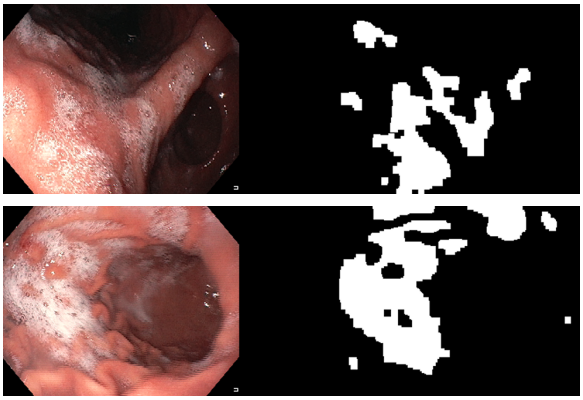

Fig. 18: Illustrations of pollution detection (B)

APPENDIX E

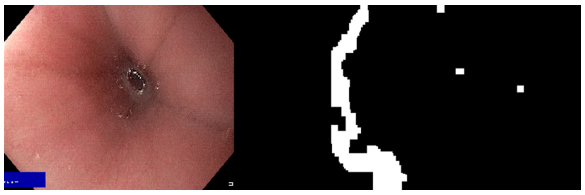

Fig. 19: Wrong pollution detection at oesogastric jonction

REFERENCES

[1] Y. Li, F. Du, D. Fu *The effect of using simethicone with or without N-acetylcysteine before gastroscopy: A meta-analysis and systemic review.* Saudi J Gastroenterol 2019;25:218-28.

[2] A. Rosebrock, 2020, *OpenCV Fast Fourier Transform (FFT) for blur detection in images and video streams*, 17-03-2021 https://www.pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/

[3] A. Usman and M. T. Mahmood, *Analysis of Blur Measure Operators for Single Image Blur Segmentation* MDPI, 2018.

[4] *OpenCV documentation : Morphological Transforms*, 17-03-2021 https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

[5] L. Elvas, M. Areia, D. Brito, et al. *Premedication with simethicone and N-acetylcysteine in improving visibility during upper endoscopy: a double-blind randomized trial.* Endoscopy 2017;49:139-45

[6] *Anatomy of the stomach*, 18-03-2021 https://anatomy-medicine.com/digestive-system/29-the-stomach.html

[7] S. Donne, 2021 *Real time texture analysis for gastroscopic-exam*, 02-04-2021 https://github.com/khookh/Real-time-texture-analysis-for-gastroscopic-exam

[8] Borgli, H., Thambawita, V., Smedsrud, P.H. et al. *HyperKvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy.* Sci Data 7, 283 (2020).

[9] M. Sandler, A. Howard, M. Zhu,. et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* CoRR, 2018.

[10] M. Sandler, A. Howard, 2018, *MobileNetV2: The Next Generation of On-Device Computer Vision Networks*, 18-03-2021 https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

[11] S. Tsang, 2019 *Review: MobileNetV2 — Light Weight Model (Image Classification)* , 18-03-2021 https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

[12] S. Donne, 2021 *Gastroscopic Classification notebook*, 20-03-2021 https://www.kaggle.com/stefanodonne/gastroscopic-classification

[13] L.S. Smith, *Cyclical Learning Rates for Training Neural Networks* CoRR, 2017.

[14] *Ilustration-of-the-HSV-Color-Space*, 20-03-2021 https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_saturation_gray.png

[15] Q. He, S. Bano, O.F. Ahmad et al. *Deep learning-based anatomical site classification for upper gastrointestinal endoscopy.* Int J CARS 15, 1085–1094 (2020).

[16] J.R. Landis, G.G. Koch."*The measurement of observer agreement for categorical data*". Biometrics 33 (1): 159–174, (1977)

[17] *Raspberry Pi 4 Tech Specs*, 19-03-2020 https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[18] *Benchmarking TensorFlow Lite on the New Raspberry Pi 4, Model B*, 19-03-2020 https://www.hackster.io/news/benchmarking-tensorflow-lite-on-the-new-raspberry-pi-4-model-b-3fd859d05b98/

[19] A. Howard, S. Gupta 2019, *Introducing the Next Generation of On-Device Vision Models: MobileNetV3 and MobileNetEdgeTPU*, 25-03-2021 https://ai.googleblog.com/2019/11/introducing-next-generation-on-device.html

[20] *A PyTorch implementation of MobileNetV3 for real-time semantic segmentation, with state-of-the-art performance*, 25-03-2021 https://reposhub.com/python/deep-learning/ekzhang-fastseg.htmlcontributions

[21] M. Siam, M. Gamal, A-R. Moemen et al. *A Comparative Study of Real-time Semantic Segmentation for Autonomous Driving* CVPR, 2019.