GPU-accelerated Real-time Gastrointestinal Diseases Detection

Konstantin Pogorelov[*,•], Michael Riegler[*,•], Pål Halvorsen[*,•], Peter Thelin Schmidt[‡,○]
Carsten Griwodz[*,•], Dag Johansen[♭], Sigrun Losada Eskeland[♣], Thomas de Lange[†,♣]

[*]Simula Research Laboratory, Norway [†]Cancer Registry of Norway [‡]Department of Medicine, Karolinska Institute, Sweden
[•]University of Oslo, Norway [○]Center for Digestive Diseases, Solna and Karolinska University Hospital, Sweden
[♣]Bærum Hospital, Vestre Viken Health Trust, Norway [♭]The Arctic University of Norway, Norway
Email: konstantin@simula.no

***Abstract*—The process of finding diseases and abnormalities during live medical examinations has for a long time depended mostly on the medical personnel, with a limited amount of computer support. However, computer-based medical systems are currently emerging in domains like endoscopies of the gastrointestinal (GI) tract. In this context, we aim for a system that enables automatic analysis of endoscopy videos, where one use case is live computer-assisted endoscopy that increases disease- and abnormality-detection rates. In this paper, a system that tackles live automatic analysis of endoscopy videos is presented with a particular focus on the system's ability to perform in real time. The presented system utilizes different parts of a heterogeneous architecture and can be used for automatic analysis of high-definition colonoscopy videos (and a fully automated analysis of video from capsular endoscopy devices). We describe our implementation and report the system performance of our GPU-based processing framework. The experimental results show real-time stream processing and low resource consumption, and a detection precision and recall level at least as good as existing related work.**

***Index Terms*—medical; multimedia; information; systems; classification**

I. Introduction

With the rapid developments in technology that allow miniaturization of cameras and sensors for moving them through the human body, there is an increasing need for real-time medical systems. These improvements lead to a lot of advantages for both patients and doctors, but also challenges for the computer science community. A system supports humans in a critical field like medicine has to fulfill several requirements, including fault tolerance, data security and privacy. Additionally, to support real-time detection of diseases in medical images and videos, the system must exhibit high performance and low resource usage.

In this paper, we describe an new version of system called EIR [1] that provides real-time support for medical image and video data analysis, and we enhance the system with GPU acceleration support. Our goal is to provide an efficient, flexible and scalable analysis and support system for endoscopy of GI tract (see figure 1). It should be applicable both for supporting traditional live endoscopies by giving real-time support and for offline processing of videos generated by wireless capsule endoscopes that are used in large-scale screening. At this time, our system detects abnormalities like those shown in figure 2, in videos of the colon. It does this through a combination of filters using machine learning, image recognition and extraction of global and local image features. However, our system is not limited to this use case, but can be extended to cover analysis of the entire GI tract. Therefore, we developed a live system that can be utilized as a computer-aided diagnostic system and a scalable detection system. In the scenario of medical image processing and computer-aided diagnosis, high precision and recall are important and the object of many studies. Our system must therefore both provide an accurate detection and analysis of the data, and address the often ignored processing performance at the same time. This is important for live feedback during examinations.

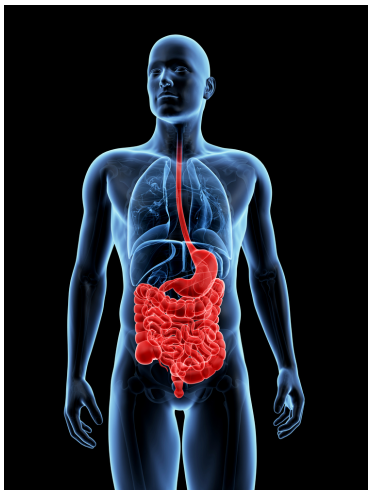
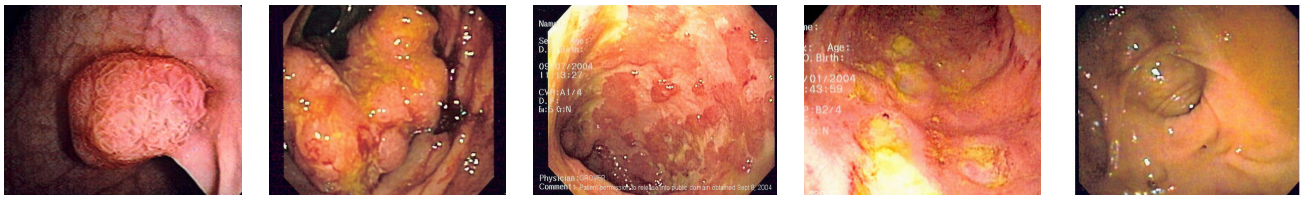


Fig. 1. Our system targets the whole GI tract (Image: kaulitzki/shutterstock.com).

A closer look at the most recent and complete related work, PolypAlert [2], reveals that real-time speeds are not achieved by the current existing systems. To tackle this problem, we have extended and improved the EIR system [3], [4], focusing on the speed of detection. Speedup is gained by applying heterogeneous technologies, in particular graphical processing units (GPUs), where we distribute the workload on a large number of processing cores. The initial results from our experimental evaluation show real-time stream processing and low resource consumption, with a precision and recall of detection at least as good as existing related systems. Compared to existing systems, it is more efficient, scales better with more data at higher resolutions and, it is designed to support different diseases in parallel at run time.



(a) Colon Polyp (b) Colorectal Cancer (c) Ulcerative Colitis (d) Crohn's Disease (e) Diverticulosis

Fig. 2. Some examples of abnormalities that can be found using colonoscopy (images are from Wikimedia Commons).

The rest of the paper is organized as follows. First, we present related work in section II. Then, in section III-A, we briefly describe the base system architecture. This is followed by a presentation of the improved system in section III-B. Next, we present the performance of the system in section IV with polyp detection as a use case. Finally, we draw conclusions in section V.

II. Related Work

Research on automatic detection of abnormalities in the GI tract is usually focused and limited to a very specific disease or abnormality. Most existing work targets detection of polyps in the colon with a specific type of camera, both due to lack of available test data, but also since it is easier to narrow the focus and create more specialized solutions. Systems aimed at polyp detection [5], [6], [7] are promising, but there is a lack of systems that are able to perform their analysis in real-time, which is required to support doctors with computer-aided diagnosis during colonoscopies.

In terms of detection performance, several systems and algorithms have been presented in literature with promising performance. The most recent and also best-performing one is the polyp-detection system of Wang et al. [2]. The presented Polyp-Alert system is able to provide near real-time feedback during colonoscopies. Near real-time in this context is defined as being able to process 10 frames per second. This is done by using visual features and a rule-based classifier to detect the edges of polyps. The system reaches an impressive performance of 97.7% correctly detected polyps. The dataset that has been used for this tests contains 52 videos taken from different colonoscopies. The dataset is not available and a direct comparison is therefore not possible. Polyp-Alert is at the moment limited to polyp detection and does not give real-time feedback for current 25 fps colonoscopy systems.

Nawarathna et al. [8] presented an approach that is not limited to polyp detection in colonoscopy videos. It is also able to detect abnormalities like bleeding. To achieve this, a texton histogram of an image block is used. Nevertheless, this system does not reach real-time performance.

A possible solution to achieve real-time instead of near real-time performance is the SAPPHIRE middleware and software development kit for medical video analysis [9]. The toolkit has been used to built the EM-Automated-RT software [10]. EM-Automated-RT does real-time video analysis to determine the quality of a colonoscopy procedure, and it is able to give visual feedback to the endoscopist performing the procedure. This is done to achieve optimal accuracy of the inspection of the colon during the procedure. Nevertheless, it is limited to the assessment of the endoscopist's quality, and does not automatize disease detection itself.

A dominant trend to speed up processing of CPU-intensive tasks is to offload processing tasks to GPUs. Stanek et al. [9], [10] indicate that utilizing a GPU and program it using either CUDA[1] or OpenCL[2] can be the right way to achieve real-time performance. In other areas this has already been explored to a certain extent. For example, we applied it in sport technology [11], [12], where GPUs were used to improve the video processing performance to achieve live, interactive panning and zooming in panorama video.

In summary, actual computer-aided diagnostic systems for the GI tract do not provide real-time performance in combination with a sufficient detection or localisation accuracy. Therefore, we present a system focusing on both high accuracy detection and real-time performance. Additionally, the aim is to provide flexibility for other diseases that can be detected.

III. System

In our research, we target a general system for automatic analysis of GI tract videos with high detection accuracy, abnormality localisation in the video frames, real-time performance and an architecture that allows easy extensions of the system. In this paper, we focus on achieving real-time performance without sacrificing high detection accuracy.

A. Basic Architecture

Our system consists of three main parts. The first is feature extraction. It is responsible for handling input data such as videos, images and sensor data, and extracting and providing features from it. The most time-consuming aspect here is the extraction of information from the video frames and images.

The second part is the analysis system. Currently, a search-based classifier that is similar to a K-nearest-neighbour approach [13] is implemented. The search-based classifier use more than 20 different global image features and combinations of them for the classification. In our use case of polyp detection, we used an information gain analysis [14] to identify a combination of the features Joint Composite Descriptor (JCD) (which is a combination of Fuzzy Color and Texture

[1]http://www.nvidia.com/object/cuda_home_new.html
[2]http://developer.amd.com/tools-and-sdks/opencl-zone/

Histogram (FCTH) and Color and Edge Directivity Descriptor (CEDD)) and Tamura as the best working ones. The features mainly focus on texture and color, and a detailed description can be found in [15]. Additionally, a localisation algorithm for polyp localisation is supported. The implementation of this part is modular and can be extended with additional diseases, classifiers or algorithms as needed. Of course, adding additional modules will require more computing power to keep the systems real-time ability. We address this by designing a heterogeneous architecture.

The last part is the presentation system. It presents the output of the real-time analysis to the endoscopist. The most challenging aspect here is that the presentation should not introduce any delays, which would make the system unsuitable for live examinations. The presentation of the results is implemented in a light-weight way using web technologies. The advantage is that it does not require additional installations, which sometimes can be problematic in a hospital environment and due to its simplicity it does not consume relevant amounts of resources.

The first version of our system worked on at most two image features at a time, it was restricted to a single computer, and the localisation part did not achieve real-time speed for full high-definition videos. Its performance is given for comparison in section IV-B.

To acheive real-time speed, the architecture had to be improved. We chose to do this by applying heterogeneous processing elements. As discussed in the related work, the most promising approach is the utilization of GPUs.

B. Heterogeneous Architecture Improvement

To improve the performance of our initial basic system architecture, we re-implemented most compute-intensive parts in CUDA. CUDA is a commonly used GPU processing framework for Nvidia graphic cards. We designed an architecture with a heterogeneous processing subsystem as depicted in figure 3.

At the moment, GPU-accelerated processing is implemented for a number of features (JCD, which includes FCTH and CEDD, and Tamura) for the feature descriptor extraction, color space conversion, image resizing and prefiltering.

In our architecture, a main processing application interacts with a modular image-processing subsystem both implemented in Java. The image-processing subsystem uses a multi-threaded architecture to handle multiple image processing and feature extraction requests at the same time. All compute-intensive functions are implementated in Java to be able to compare performance with the heterogeneous implementation, which is transparently accessible from Java code through a GPU CLib wrapper. The JNA API is used to access the GPU CLib API directly from the image processing subsystem. The GPU CLib is implemented in C++ as a Linux shared library that connects to a stand-alone processing server and pipes data streams for handling by CUDA implementations. Shared memory is used to avoid the performance penalty of data copying. Local UNIX sockets are used to send requests and receive status responses

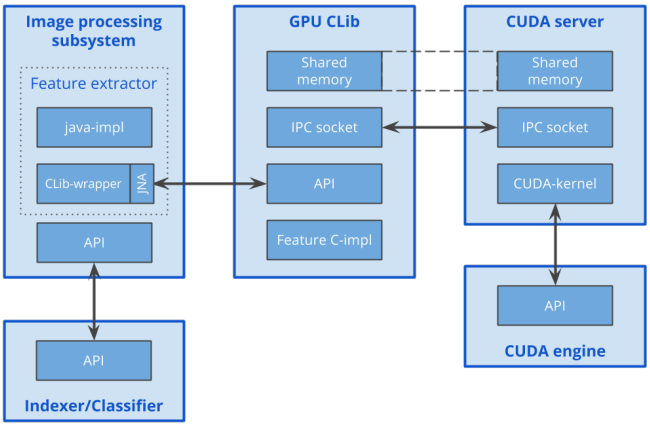


Fig. 3. The main processing application consisting of the indexing and classification parts uses the GPU-accelerated image processing subsystem. This subsystem provides feature extraction and image filtering algorithms. The most compute-intensive procedures are executed on a stand-alone CUDA-enabled processing server. The interaction between application and server is done via a GPU CLib shared library, which is responsible for maintaining connections and streaming data to and from the CUDA-server.

from the CUDA server because they can be integrated more easily with asynchronous on the JNI side then shared-memory semaphores. The CUDA server is implemented in C++ and uses CUDA SDK to perform computations on GPU. The CUDA server and all heterogeneous-support subsystems are built with distributed processing in mind, and can easily be extended with multiple CUDA servers running locally or on several remote servers.

The processing server can be extended with new feature extractors and advanced image processing algorithms. It enables the utilization of multi-core CPU and GPU resources. As an example, the structure of the FCTH feature extractor implementation is depicted in figure 4. It shows that for the image features, all pixel-related calculations are executed on the GPU. In the case of the FCTH feature, this includes also the processing of a multi-threaded shape detector and fuzzy logic algorithms.

To achieve better performance, a heterogeneous processing subsystem provides the transparent caching of input and intermediate data, which reduces the CPU-GPU bandwidth usage and eliminates redundant data copy operations during image processing.

IV. Evaluation

To evaluate our system, we use colorectal polyp detection as a case study. As test data, the ASU-Mayo Clinic polyp database[3] has been used. This dataset is the largest publicly available dataset consisting of 20 videos. We converted the videos from WMV to MPEG-4 for the experiments. The 20 videos have a total number of 18.781 frames with a maximum resolution of 1920×1080 pixels (full high definition) [16]. Further, we concentrate the experiment on the detection part.

[3]http://polyp.grand-challenge.org/site/Polyp/AsuMayo/

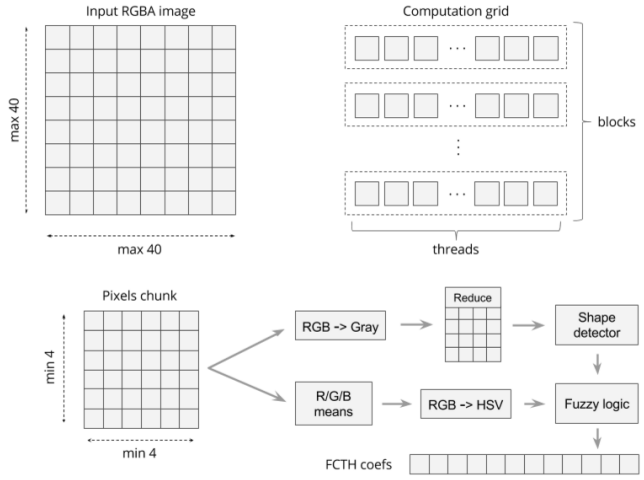


Fig. 4. GPU-acceleration is used to extract various features from input frames. The figure shows an example of our FCTH feature implementation. The input frame is split into a number of non-overlapping blocks. Each of them is processed separately by two GPU-threads. The main processing steps include color space conversion, size reduction, shape detection and fuzzy logic computations.

TABLE I
LEAVE-ONE-OUT CROSS-VALIDATION FOR 20 VIDEOS IN THE USED DATASET. THE TABLE DEPICTS TP (TRUE POSITIVES), TN (TRUE NEGATIVES), FP (FALSE POSITIVES), FN (FALSE NEGATIVES) AND THE METRICS PRECISION, RECALL AND F1 SCORE.

Video	TP	TN	FP	FN	Precision	Recall	F1
np_5	1	680	0	0	1	1	1
np_6	1	836	0	0	1	1	1
np_7	1	767	0	0	1	1	1
np_8	1	710	0	0	1	1	1
np_9	1	1,841	0	0	1	1	1
np_10	1	1,923	0	0	1	1	1
np_11	1	1,548	0	0	1	1	1
np_12	1	1,738	0	0	1	1	1
np_13	1	1,800	0	0	1	1	1
np_14	1	1,637	0	0	1	1	1
wp_2	140	9	20	70	0.875	0.6666	0.7567
wp_4	908	1	0	0	1	1	1
wp_24	310	68	127	12	0.7093	0.9627	0.8168
wp_49	421	12	62	4	0.8716	0.9905	0.9273
wp_52	688	101	284	31	0.7078	0.9568	0.8137
wp_61	162	10	165	0	0.4954	1	0.6625
wp_66	223	12	165	16	0.5747	0.9330	0.7113
wp_68	172	51	20	14	0.8958	0.9247	0.9100
wp_69	265	185	138	26	0.6575	0.9106	0.7636
wp_70	379	1	0	29	1	0.9289	0.9631
Weighted average:					0.9388	0.9850	0.9613

Localisation of the polyp in the frame is also implemented and optimized, but due to space restrictions, it is not included here.

A. Polyp Detection

In terms of detection performance, we reach acceptable results, as illustrated in table I. The actual performance of the system has been assessed using a combination of JCD and Tamura features. For a robust and representative evaluation, we conducted a leave-one-out cross-validation with all available video sequences. The training of the system using 19 videos takes around 2 minutes. Due to the problem that different video sequences contribute values based on different numbers of video frames, we weighted the values contributed by every single video sequence with the overall number of frames in the sequence. This led to an average precision of 0.9388, an average recall of 0.9850, and an average F1 score value of 0.9613. That means that the system can find polyps with a precision of almost 94% and detect almost 99% of all frames that contain a polyp.

These results demonstrate that the system is able to reach high detection accuracy and also, that it can compete with other state-of-the-art systems. For example, Wang et al. [2] reach with their system a recall of 97.70% while our system reaches 98.50%. Hwang et al. [17] report a precision of 83.00% while we achieve 93.88%. In terms of sensitivity, we reach 96.37% compared to Wang et al. [18] with 81.40%, Alexandre et al. [19] with 96.69% and Cheng et al. [20] with 86.20%. Thus, our system performs at the high level of precision compared to the best related systems. However, more important in this paper is the comparison of our own basic architecture with the improve heterogeneous approach in terms of their time-performance.

B. Live Analysis in Real-time

Basic Architecture. The basic multi-core CPU-only architecture performance results are depicted in figure 5. For all the tests, we used 3 videos from 3 different endoscopic devices and different resolutions. The three videos are wp_4 with $1,920\times1,080$, wp_52 with 856×480 and np_9 with 712×480. We chose these videos to show the performance under the different requirements that the system will have to face when in practical use. The computer used was a Linux server with 32 AMD CPUs and 128 GB memory. The figures show, that the basic system was able to reach real-time performance for full HD videos using a minimum of 16 CPU cores and at least 12 GB of memory. This has the huge disadvantage that real-time speed is only achieved on expensive multi-CPU systems. In terms of memory, tests showed that the system has rather small requirement. This is good, since it means that memory consumption is not a bottleneck to scalability, and that we can ignore it for now.

Heterogeneous Architecture. The videos used to evaluate the system performance have different resolutions. The resolutions are full HD (1920×1080), WVGA1 (856×480), WVGA2 (712×480) and CIF (384×288). They are labelled correspondingly in figures 6, 7, 8 and 9. A framerate of 30 frames per second (FPS) was assumed, and consequently, 33.3 milliseconds processing time per frame was considered real-time speed. Our results for the heterogeneous architecture were obtained using a conventional desktop computer with an Intel Core i7 3.20GHz CPU, 8 GB RAM and a GeForce GTX 460 GPU. To be able to compare the basic and improved systems directly, the same Java source code from the basic system was used to collect the evaluation metrics. In the figures, the basic system's results are labelled as Java. The improved

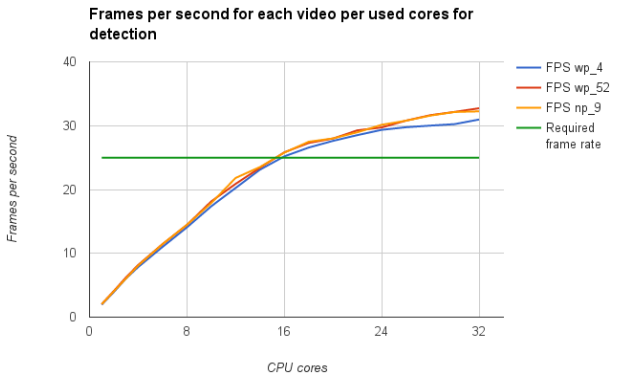


Fig. 5. The detection performs efficiently and the required frame rate is reached with 12 GB of memory and 16 CPU cores used in parallel on cluster-based computation platform without utilizing heterogeneous architecture.

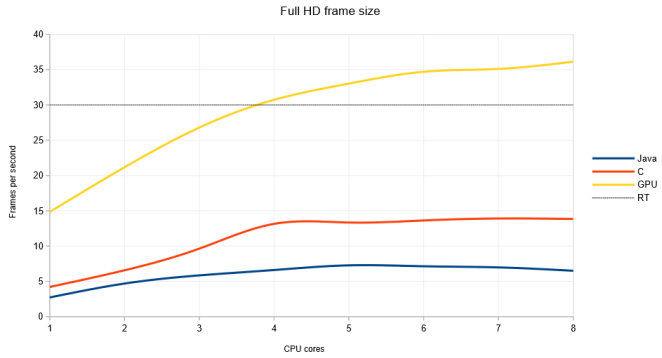


Fig. 6. The improved GPU-enabled heterogeneous algorithm reaches real-time performance (RT line) with 30 frames per second for full HD (1920×1080) videos on a desktop PC using only 4 CPU cores and 5 Gb of memory. The maximum frame rate is around 36 FPS using 8 CPU cores. The Java and C implementations cannot reach real-time performance on the used hardware.

system's results with disabled GPU-acceleration are labelled as C. Finally, the improved system's run in the heterogeneous mode with enabled GPU-acceleration is labelled as GPU.

The performance evaluation shows, that the basic architecture can process full HD frames using all 8 available CPU cores and up to 4 GB of memory at 6.5 FPS for Java and 13.8 FPS for the C implementations (see figure 6) with corresponding frame processing times of 154ms and 72ms, respectively (see figure 8). For the smaller frame sizes, real-time speed was reached at most 4 CPU cores and at most 4 GB of memory. The maximum frame rates that were be reached were 49 FPS, 51 FPS and 66 FPS for WVGA1, WVGA2 and CIF frame sizes, respectively (see figure 7 and figure 9).

The evaluation of the improved heterogeneous system shows that the GPU-enabled architecture can easily process full HD frames using only 4 CPU cores (see figure 6) and up to 5 Gb of memory with a frame processing time of 32.6ms (see figure 8). The maximum frame rate for full HD frames was 36 FPS using all 8 CPU cores. For the smaller frame sizes, the real-time requirements were reached with only 1 CPU core and up to 4.5 GB of memory. The maximum frame rate that we achieved was around 200 FPS (see figure 7 and figure 9).

The results show clearly, that the given hardware system with the basic architecture cannot reach real-time performance for full HD videos even using all available CPU cores, and only for the low-resolution WVGA videos, real-time can be reached. For the improved heterogeneous system, the real-time performance for full HD videos is easily reached using only 4 CPU cores and one outdated GPU. The smaller videos can be processed utilizing only one CPU core plus GPU. Memory size is not a limiting factor and the system can be deployed even on desktop PCs with a general-purpose GPU as an accelerator.

These quantitative results illustrate, that using a heterogeneous architecture is key to real-time performance and parallel analysis of videos with different approaches. Furthermore, the improved heterogeneous system has significant over-performance in terms of real-time video processing. This

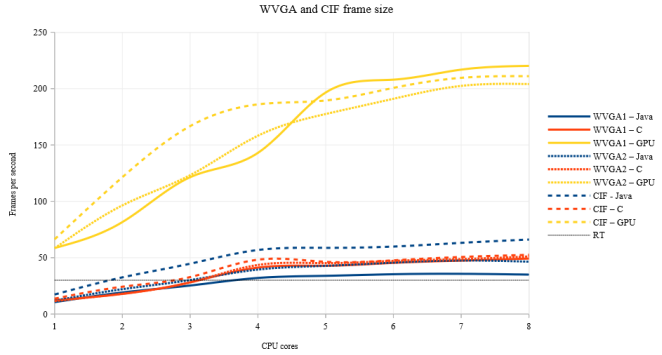


Fig. 7. The smaller WVGA1 (856×480), WVGA2 (712×480) and CIF (384×288) videos can be processed by the improved GPU-enabled heterogeneous algorithm in real-time using only 1 CPU core. The maximum frame processing rate reaches more than 200 FPS. These results can be improved by putting all feature-related computations on the GPU.

makes it possible to implement more feature extractors, classifiers and many other image processing algorithms to increase the number of detectable diseases by our system while keeping the real-time capability.

V. Conclusion

Efficient and fast data analysis of medical video data is important for to several reasons, including real-time feedback and increased system scalability. In this paper, we have presented a computer-based medical systems that tackles live automatic analysis of endoscopy videos. The presented system utilizes different parts of heterogeneous architectures and will soon be tested in a clinical trial with high definition colonoscopy videos. Compared to existing systems, our system provides an abnormality detection precision and recall level at least as good as existing related work. However, with an achieved

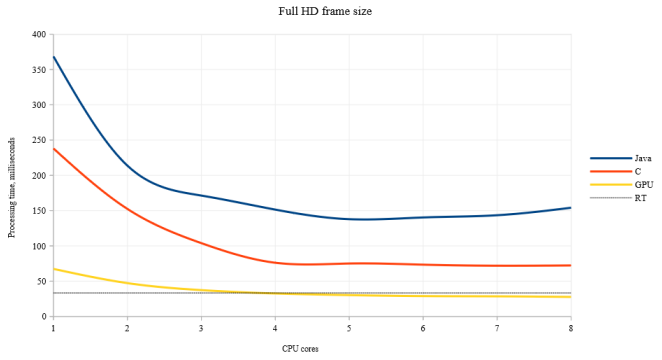


Fig. 8. The processing time for the GPU-accelerated algorithm decreases slightly with increasing number of used CPU cores for a single full HD frame. This happens due to the CPU-parallel implementation of feature comparison and search algorithms which are not as compute intensive as feature extraction. The Java and C implementations reach the minimum frame processing time with 4 used CPU cores. The reason is that the used CPU has 4 real cores with hyper-threading feature enabled and it cannot handle CPU-intensive calculations efficiently for all 8 (real plus virtual) cores.

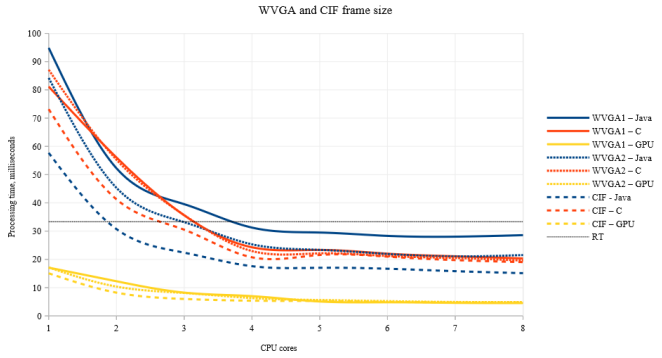


Fig. 9. For the smaller frame sizes the GPU-accelerated algorithm results in a processing time far below the real-time margin. The minimum is reached with 5 milliseconds using 8 CPU cores. This is a prove for the high system performance and ability to be extended by additional features or to process several video streams at the same time on a conventional desktop PC.

performance of 200 frames per seconds, it is superior with respect to video stream processing time and the ability to provide real-time automatic feedback during live endoscopies.

We continue to optimize and improve our implementation of the detection system. Ongoing work includes moving the localisation to the GPU, and we are in the process of extending the number of diseases detected. Our current performance easily allows for this, and our future multi-disease detection system will be distributed on several computers.

Acknowledgment

This work has been founded by the Norwegian Research Council under the FRINATEK program, project "EONS" (#231687).

References

[1] M. Riegler, K. Pogorelov, P. Halvorsen, T. de Lange, C. Griwodz, P. T. Schmidt, S. L. Eskeland, and D. Johansen, "EIR - efficient computer aided diagnosis framework for gastrointestinal endoscopies," in *Proc. of CBMI*, 2016.

[2] Y. Wang, W. Tavanapong, J. Wong, J. H. Oh, and P. C. de Groen, "Polyp-alert: Near real-time feedback during colonoscopy," *Computer methods and programs in biomedicine*, no. 3, 2015.

[3] M. Riegler, K. Pogorelov, J. Markussen, M. Lux, H. K. Stensland, T. de Lange, C. Griwodz, P. Halvorsen, D. Johansen, P. T. Schmidt, and S. L. Eskeland, "Computer aided disease detection system for gastrointestinal examinations," in *Proc. of MMSys*, 2016.

[4] K. Pogorelov, M. Riegler, J. Markussen, H. Kvale Stensland, P. Halvorsen, C. Griwodz, S. L. Eskeland, and T. de Lange, "Efficient processing of videos in a multi-auditory environment using device lending of gpus," in *Proc. of MMSys*, 2016.

[5] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Near real-time retroflexion detection in colonoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 1, pp. 143–152, 2013.

[6] Y. Wang, W. Tavanapong, J. S. Wong, J. Oh, and P. C. de Groen, "Detection of quality visualization of appendiceal orifices using local edge cross-section profile features and near pause detection," *IEEE Biomedical Engineering (BME)*, vol. 57, no. 3, pp. 685–695, 2010.

[7] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Computer-aided detection of retroflexion in colonoscopy," in *Proc. of IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, 2011, pp. 1–6.

[8] R. Nawarathna, J. Oh, J. Muthukudage, W. Tavanapong, J. Wong, P. C. De Groen, and S. J. Tang, "Abnormal image detection in endoscopy videos using a filter bank and local binary patterns," *NC*, 2014.

[9] S. R. Stanek, W. Tavanapong, J. Wong, J. Oh, R. D. Nawarathna, J. Muthukudage, and P. C. De Groen, "Sapphire middleware and software development kit for medical video analysis," in *Proc. of CBMS*, 2011, pp. 1–6.

[10] ——, "Sapphire: A toolkit for building efficient stream programs for medical video analysis," *Computer methods and programs in biomedicine*, vol. 112, no. 3, pp. 407–421, 2013.

[11] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk *et al.*, "Bagadus: An integrated real-time system for soccer analytics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 1s, p. 14, 2014.

[12] R. Langseth, V. R. Gaddam, H. K. Stensland, C. Griwodz, and P. Halvorsen, "An evaluation of debayering algorithms on gpu for real-time panoramic video recording," in *Proc. of ISM*, 2014, pp. 110–115. [Online]. Available: http://dx.doi.org/10.1109/ISM.2014.59

[13] M. Riegler, K. Pogorelov, M. Lux, P. Halvorsen, C. Griwodz, T. de Lange, and S. L. Eskeland, "Explorative hyperbolic-tree-based clustering tool for unsupervised knowledge discovery," in *CBMI*, 2016.

[14] J. T. Kent, "Information gain and a general measure of correlation," *Biometrika*, vol. 70, no. 1, pp. 163–173, 1983.

[15] M. Lux and O. Marques, *Visual Information Retrieval Using Java and LIRE*. Morgan & Claypool, 2013, vol. 25.

[16] N. Tajbakhsh, S. Gurudu, and J. Liang, "Automated polyp detection in colonoscopy videos using shape and context information," *IEEE Transactions on Medical Imaging*, 2015.

[17] S. Hwang, J. Oh, W. Tavanapong, J. Wong, and P. de Groen, "Polyp detection in colonoscopy video using elliptical shape feature," in *Proc. of ICIP*, Sept 2007, pp. 465–468.

[18] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Part-based multiderivative edge cross-sectional profiles for polyp detection in colonoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1379–1389, 2014.

[19] L. A. Alexandre, J. Casteleiro, and N. Nobreinst, "Polyp detection in endoscopic video using svms," in *Proc. of PKDD*, 2007, pp. 358–365.

[20] D.-C. Cheng, W.-C. Ting, Y.-F. Chen, Q. Pu, and X. Jiang, "Colorectal polyps detection using texture features and support vector machine," in *Advances in Mass Data Analysis of Images and Signals in Medicine, Biotechnology, Chemistry and Food Industry*. Springer, 2008, pp. 62–72.