International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 10 (2018) pp. 331-341

Received 4 May, 2018; Accept 6 Nov, 2018; Publish 15 Nov, 2018

Distributed Multi-Agent Implementation of Text Line Segmentation Using Parallel Seam Carving

Mehdi Daldali[1] and Abdelghani Souhar[2]

LaRIT, Faculty of Sciences, Ibn Tofaïl University,
B.P 133, Kénitra 14000, Morocco
[1] *mehdi.daldali@outlook.com*
[2] *houssouhar@gmail.com*

***Abstract*: Most Optical Character Recognition systems (OCR) and Handwritten Recognition systems (HWR) are based on sequential processing of document images, while some others use parallelism to alleviate the computational load on single processors by balancing the tasks on parallel hardware using "Single Instruction, Multiple Data" (SIMD) architectures. This does not completely correspond to how humans read, where reading a line rely on parallel decision making based on the central and peripheral vision. Thus, it makes these OCR systems limited in face of highly complex OCR problems, unable to synchronize their multiple tasks in a participative way. This paper proposes a parallel text line segmentation approach based on distributed multi-agent systems using a parallel implementation of seam carving. The seam carving text segmentation technique is a language-independent approach capable of processing grayscale images, and showed promising results at extracting handwritten text lines. Additionally, its "top-down" approach to text segmentation offers an opportunity for building efficient parallel OCR systems, using multiple specialized system components built around a multi-agent architecture. This can be further exploited as a base to build more collaborative neuromorphic systems. Tests on different handwritten documents showed an important runtime speedup with parallelization using MPI as a communication interface, without compromising the efficiency.**

***Keywords*:** document image analysis, handwriting recognition, text line segmentation, parallel computing, multi-agent systems.

I. Introduction

Despite showing impressive cognitive prowess, the common human can't focus on more than one task at a time due to human context switching and lack of attention. Although, defining text lines on a piece of paper remains a simple task for most human readers, even if the human eyes can only focus on a small area of the document read.

This is possible thanks to our brain's visual cortex which is capable of processing multiple levels of visual focus simultaneously, where even if our focus is mainly put on the central visual field (foveal vision), we are still able to notice events happening on the peripheral. This phenomenon can be observed while reading full paragraphs of printed or handwritten text, when it is easy to recognize each text line from its neighbors by using the peripheral vision as a guide to the eye movements while reading the text line.

The last affirmations are backed by studies concerning the role of eye movements in reading and information processing [1], where it was shown that the peripheral field of vision is contributing heavily in the reading process. By using the neighboring elements of the central visual field, the reader can efficiently read the text without having to think about where the next spot to look at is.

This central/peripheral analysis could also be useful while reading clumped handwritten text, where it's easier to make decision on uncertain words or characters and touching text components, by analyzing the text relatively to its neighboring elements.

This phenomenon can bring some improvements to the OCR/HWR techniques, when dealing with complex handwritten text, such in the case of Arabic manuscripts. Implementing such a collaborative process implies the use of parallel and distributed computing architectures.

With the democratization of cloud computing, big data systems, and high performance computing, it's becoming more and more appealing to take advantage of these technologies for runtime acceleration or building advanced parallel methods and multi-agent systems. With a proper approach, a machine can adopt the collaborative aspect of human vision by using a parallel OCR model, and will permit the use of more elaborated and intelligent methods, with a side benefit of accelerating the processing time of document images.

In the next sections, a brief look will be taken into some of the related works, and a description of the seam carving technique will be presented as well as its OCR/HWR implementation for text line segmentation [2], followed by a presentation of its parallel variation using distributed multi-agent architecture, tests results of efficiency and acceleration rates achieved, and finally, a brief conclusion and perspective for future works.

II. Related works

OCR/HWR approaches fall into three main schools of thought: "bottom-up" analysis, "top-down" analysis, and hybrid analysis based on both approaches. The "bottom-up" analysis relies on pixel-level features and is used to detect elemental components that forms letters, then words, text lines, and so on [3, 4]. This vision can face some problems when dealing

with low quality document images or historical manuscripts where it's very difficult to extract text components by binarization. This early segmentation by binarization can be completely avoided when following "top-down" analysis, where global image features are extracted directly without any significant pre-processing, such as in the case of the Seam Carving segmentation [2]. The hybrid methods are by far the most used in the literature, where "top-down" and "bottom-up" analysis are used cooperatively to achieve better segmentation efficiency.

The Seam Carving segmentation follows a "top-down" analysis, making it one of the most fit approaches to hand written text segmentation. Its ease of implementation, simplicity, and ability to process grayscale images are also points that contributed in our choice for using it as a base for the proposed collaborative multi-agent OCR/HWR system.

Many OCR systems have been developed over the years, dealing with different aspects of document image analysis. Constrained by the technology of the time, most of these systems were designed to process data sequentially until the apparition of new hardware technology capable of efficient parallel processing. As a result, many efforts were put into building new parallel OCR systems, in order to either develop new original systems or to try theoretical approaches that could not be implemented before, due to their massive computational load.

Parallel programing made it possible to build faster OCR systems by relaying computational loads to parallel hardware like in the case of GPGPU. In [5], a parallel implementation was proposed for runtime acceleration based on simple and already well known sequential segmentation approaches. Sadly, the GPGPU use in similar cases, is done for the sole purpose of runtime speedup as the complexity of computations possible on GPUs is limited due to the SIMD (Single Instruction Multiple Data) nature of these hardware architectures. This is also the case for systems based on statistical inference or using machine learning like neural networks, as the use of parallelization benefits are limited to the runtime speedup during the data analysis. Such is the case in [6] where a parallel implementation of a statistical inference based OCR software was proposed and deployed on Intel's processor Xeon Phi.

Other methods use parallelism more sophistically. In [7], an Arabic OCR system was proposed based on the Dynamic Time Warping algorithm which is considered as one among the strongest approaches in the literature, by being able to achieve high recognition rates. The problem faced by this method is the huge runtime required during the recognition process. This was addressed by using intelligent load balancing techniques to efficiently exploit heterogeneous computing clusters composed of general purpose computers, such as grid computing. Such parallel approaches focus more on the runtime acceleration of the parallel software instead of the system's efficiency regarding parallel processing and decision making during the segmentation and recognition phases. Other OCR approaches proposed by several researchers never addresses the possible parallel implementations of their systems, even though some of them show a huge potential for building efficient parallel systems. In [8], the problem of OCR system errors was addressed. Based on different methods, different OCR systems can deal better with problems faced by others. And as such, two OCR system were used in a complementary implementation, where each system tries to correct the other's recognition errors. This type of systems can be implemented in a parallel architecture where each subsystem will be running independently and be capable of an active and intelligent collaboration instead of a turn-based computing paradigm.

As for the text line segmentation methods, most are based on connected components analysis which require data locality, making an efficient parallelization of such systems hard to achieve. In case of [9], the lines detection relies on processing all of the text components present in the document, where connected components are clustered in several aligned clusters representing text lines based on a Hidden Markov Model requiring an important amount of processing power to deliver results in acceptable time. The same can be said for [10], where a multi-agent approach was proposed. In this case, the agents' results are based on other agents' decisions which in turn, need to process the total number of the extracted text components, making it hard to take advantage of any intelligent parallelism, even though, a decent runtime acceleration could be achieved by using GPGPU for faster calculations.

In this paper, the proposed system takes inspiration mainly from the human cognitive systems, such as the case in [11], where the authors developed an HPC-based context-aware intelligent text recognition system (ITRS) adopting a parallel computing architecture that incorporates the HPC technologies with advances in neuromorphic computing models. Basically, the proposed system learns from what has been already read to form predictions of the words and sentences during later recognition process. Such approaches provide robust performance in case of noisy or damaged document images. This idea can be further applied to other aspects of OCR/HWR, like the lines segmentation phase.

The seam carving approach presents itself as a simple method capable of combining local and global vision of the problem in an efficient way, making it a promising candidate for parallelization to build efficient and collaborative neuromorphic parallel systems. Thus, the proposed system is a parallel implementation of the seam carving line segmentation method, following a distributed multi-agent approach. This architecture choice can be beneficial for building a multi component system capable of collaborating in both decision making and computation load balancing, and can be easily adapted to be used as a base for future works.

III. Text lines segmentation by Seam Carving

A. Seam Carving

The seam carving was first proposed in 2007 as a new content-aware approach to image resizing using dynamic programming and the gray scale representation of the image [12]. This method consists of detecting horizontal or vertical seams with the least amount of details in order to either remove or duplicate them, which preserve the important areas dimensions and ratios.

The cost map representing the cost to reach each pixel from the side is calculated using a dynamic programming technique from an energy representation of the image, usually a gradient magnitude that represents edge position as in Fig. (Fig. 1).

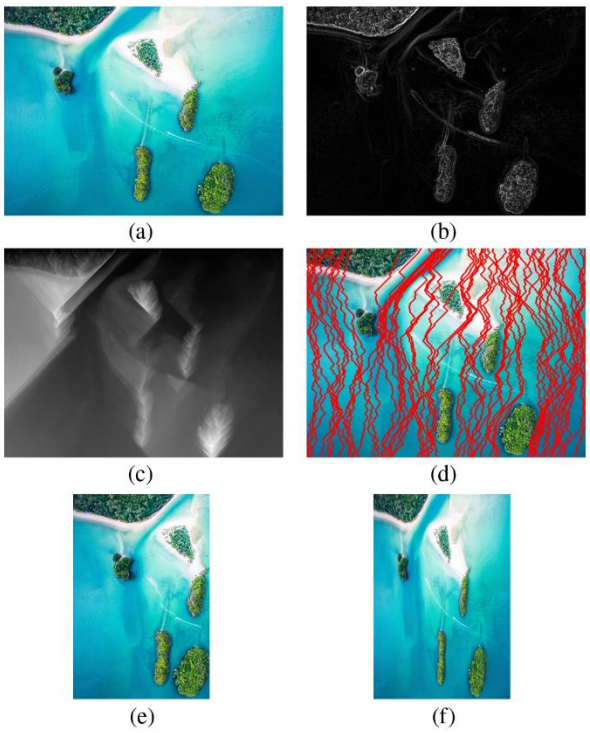

Figure 1. Example of the seam carving technique. (a) Original image. (b) The energy map / gradient magnitude (Sobel filter). (c) The energy cost map. (d) Some of the carved seams. (e) Resized image using seam carving, and (f) interpolation for comparison.

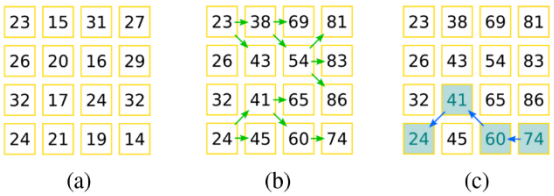

Figure 2. Example of the dynamic programming used in the seam carving technique. (a) The cost map extracted from the image energy representation (Sobel operator). (b) The cost calculation process. (c) The least cost seam, retrieved by backtracking from the last least cumulative cost position.

Each position on the cost map equivalent to the pixel position on the image have a value equal to the sum of its own energy, and the least cost value of the 3 neighboring pixel positions from the last column processed:

$$C(I_{i,j}) = e(I_{i,j}) + \min_{k\in\{-1,0,1\}} \left(C(I_{i-1,j+k})\right) \qquad (1)$$

where $I_{i,j}$ designate the image value at the i-th row and the j-th column, with the origin at the upper left corner of the image, and $e(I_{i,j})$ the energy at the same location, while $C(I_{i,j})$ is the cost value of the position (i,j). To find the least-cost seam, the algorithm backtracks starting from the position with the minimum cost value on the last column of the cost map as in Fig. 2.

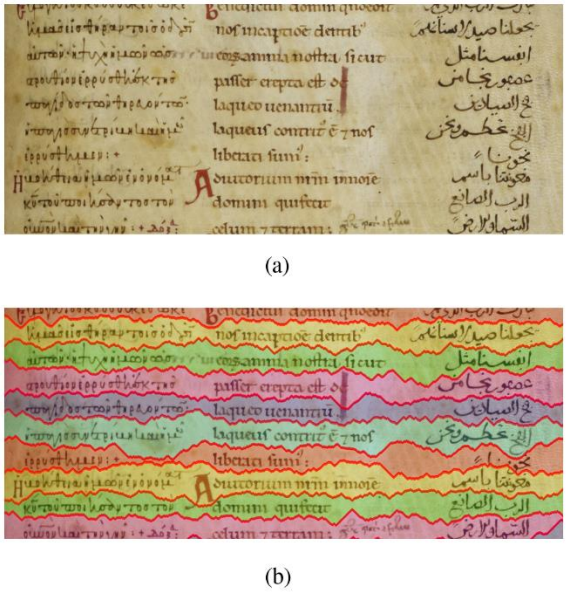

Figure 3. Demonstration of the seam carving text segmentation [2]: (a) original image from the Harley Trilingual Psalter, and (b) the segmentation result.

B. Text lines segmentation

Seam carving can also be used in order to search for least cost paths from side to side of a cost map. This feature was exploited in some text line segmentation approaches by finding separating seams laying between each two consecutives text lines [13, 14]. In a prior work [2], this seam carving method was also used to find the least cost seam which separate each two adjacent text lines, Fig. 3 shows an example of handwritten text segmentation.

First, medial seams are identified by approximating the text baseline using a projection profile method. This is achieved by slicing the document image into a number of vertical slices of equal width. A horizontal histogram is then extracted for each slice in order to detect local maxima representing an approximation of the text baselines. To form fully connected baselines, each local maximum detected on a vertical slice are then matched with the extracted maxima from the adjacent slice to remove false detections. The resulting medial seams are then used as guides when searching for the least cost separating seams (Fig. 4). Each two adjacent baselines approximations are used to define a constrained image area where the seam carving technique makes it possible to find a separating seam crossing the interline area without crossing over text components.

Test results on a dataset [15] made of 123 handwritten documents, showed an f1-score of more than 97% using a matching threshold of 90% [2]. The nature of this approach makes it possible to build an efficient parallel implementation where each separating seam can be calculated independently from other seams. This can not only speedup the processing, but also offer a new ground for building complex cooperative OCR/HWR systems.

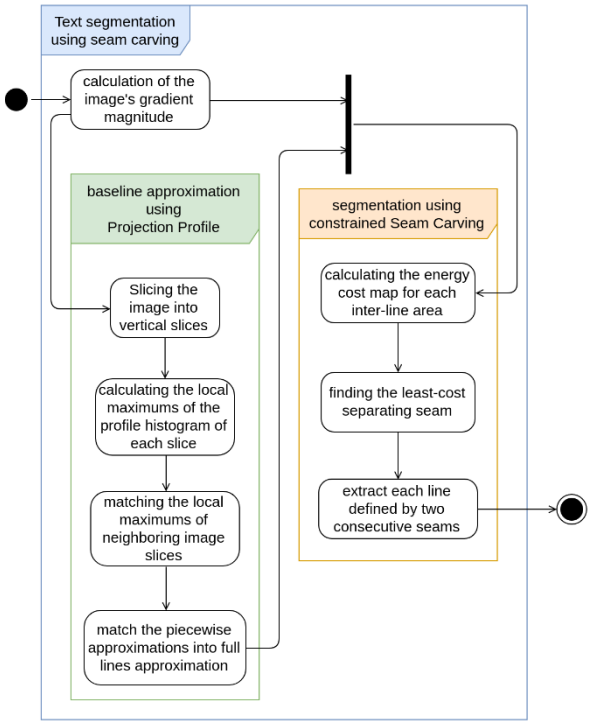

Figure 4. Schema describing text line segmentation using seam carving.

IV. Parallel text line segmentation

A. *Parallel Seam Carving*

The seam carving is a recursive process where each next computation is dependent to the last one. Many efforts were made to gain more time efficiency by implementing parallelization techniques.

In the work discussed in [16] and [17], a brief study was conducted on the ability to parallelize the seam carving operator. Two distinct approaches were proposed. The first one [16] is based on row calculations. The image is sliced into horizontal slices, to compute each slices cost map individually by communicating the cost value to the adjacent cost map calculator when the edge cost value is needed on the other side (Fig. 5).

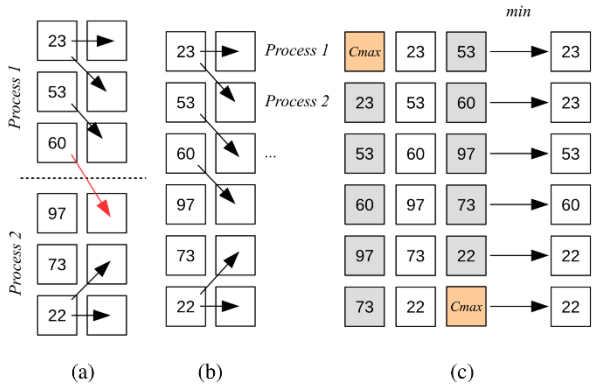

Figure 5. Different approaches of the cost map parallel computation: (a) the method proposed in [16] using message passing to communicate the edge values, (b) the method proposed in [17] using shared memory, and (c) the proposed method based on a vector implementation of (b).

This approach showed to be communication intensive making it inefficient. The second approach [17] is more granular. To parallelize the computation, each parallel process will be in charge of calculating the cost of one pixel, filling the cost map column by column.

In the proposed system, a vector implementation of the latter approach is used for the cost map calculation. For each column of the energy image, 2 views are generated by rolling the elements of the column up and down by 1 step, a stack is then compiled from the original column data and the two new columns, filling the blank values with a maximum cost value C_{max}, in case of gray scale images of m columns:

$$C_{max} > 255 \times imageWidth \tag{2}$$

For each row of this new matrix of width 3, the minimum value is then calculated and added to the same row on the next cost map column in addition of the energy cost (Fig. 5). The algorithm used is presented on (Alg. 1).

Data: $eMap$: image's energy map
Result: $cMap$: cost map
$cMap \leftarrow eMap$
for $column_{i+1} \in eMap$ **do**
  $c[0][:-1] \leftarrow column_i[1:]$
  $c[0][-1] \leftarrow \infty$
  $c[1] \leftarrow column_i$
  $c[2][1:] \leftarrow column_i[:-1]$
  $c[2][0] \leftarrow \infty$
  $minCost \leftarrow min(c, axis = 1)$
  $cMap[:, i+1] \leftarrow cMap[:, i+1] + minCost$
end

Algorithm 1. Vectorized computation of the energy cost map.

This method showed to be more parallel-friendly, and the vectorization could offer enough performance without the need of a GPGPU acceleration due to the overhead introduced by context switching and data transfers (Tab. 1).

The runtime tests showed that the speedup gained from the parallel computation is considerable. Still, there's some more speedup that can be achieved by parallelizing the text segmentation process.

Table 1. Results of the runtime measures using sequential and parallel cost map computation for the seam carving segmentation (document image with a resolution of 5100x6600).

Method	Runtime (s)
Sequential	126.32
Parallel	31.94

B. *Multi-agent architecture*

The seam carving could be implemented as a multi-agent system capable of distributing the computation load efficiently, by building specialized agents, each of them responsible of a special task. As such the Seam Carving Text Segmentation proposed in [2] can be described by the Diagram (Fig. 6), showing great potential for a distributed and parallel implementation.

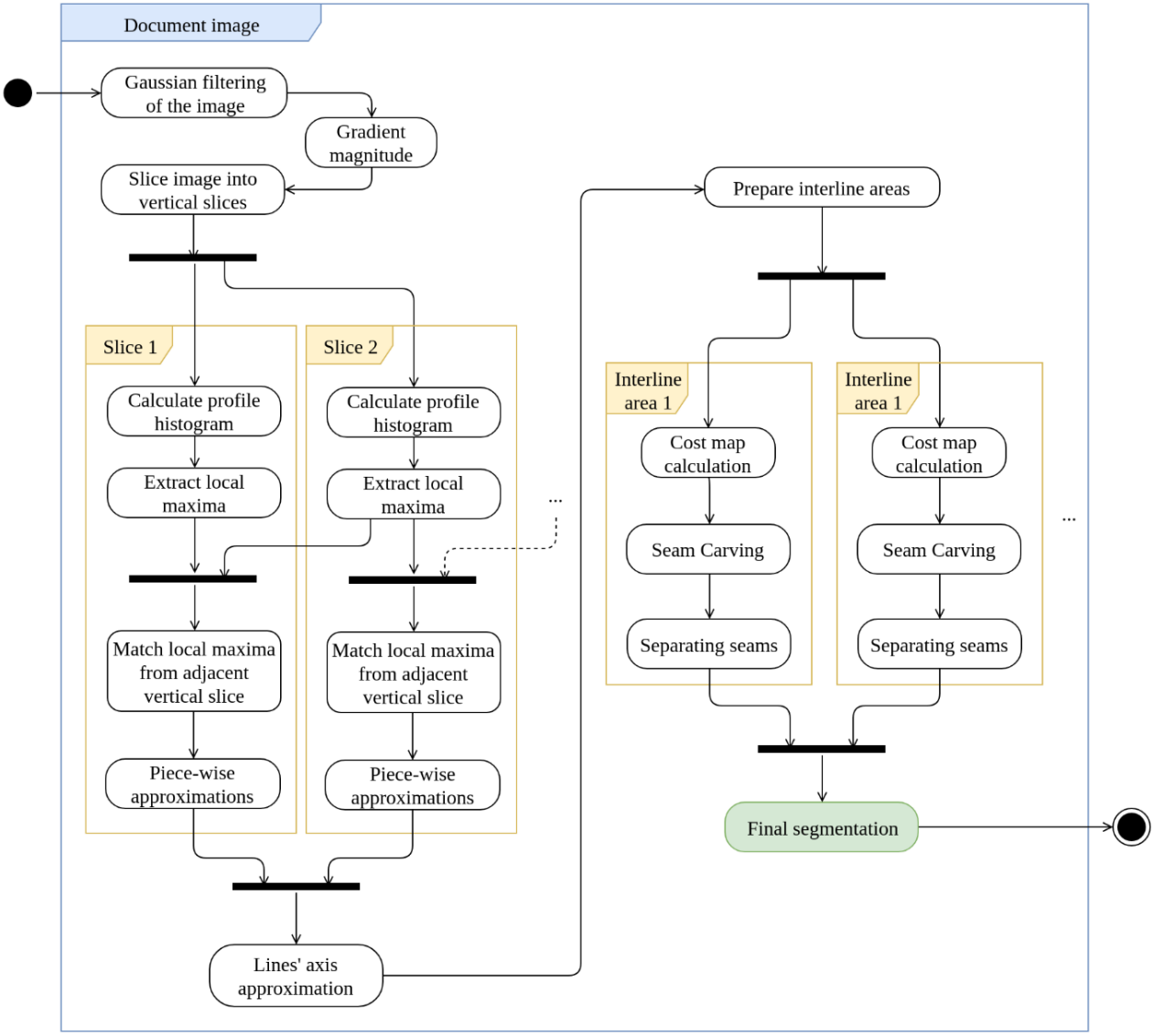

Figure 6. Diagram depicting the distributed Seam Carving Text Segmentation proposed in [2].

The literature contains a multitude of definitions regarding agents and multi agent systems. These definitions are biased by the background field of research. The most general and widely accepted definition of an agent in computer science, is that an agent is any system that can perceive its environment through sensors and act upon that same environment through effectors.

And as such, the widely accepted definition of multi agent systems is defined by being a coupled network of problem solving agents that work together to find answers to problems that are beyond the individual capabilities or knowledge of each agent. This means that the core concept of any MAS is cooperation between those intelligent entities.

In addition to runtime acceleration, parallel computation can bring some improvements to Multi-agent systems (MAS) where the agents can simultaneously participate in problem solving more efficiently. For text lines detection using seam carving, the problem can be split into a number of sub-problems of low complexity, that could be addressed with specialized sub-systems.

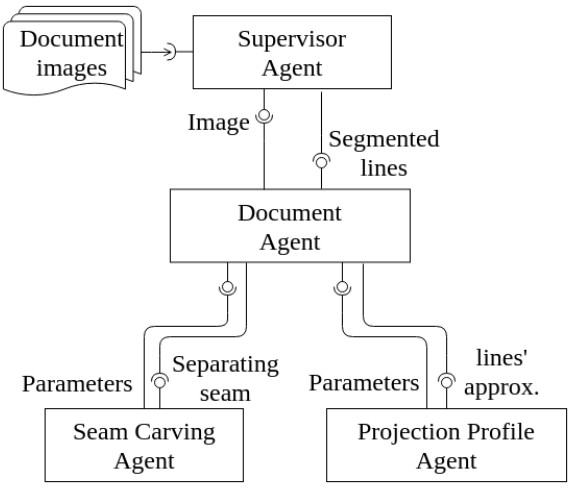

Figure 7. The proposed multi-agent system components.

Thus, the proposed implementation is a distributed multi agent system (Fig. 7) composed of a supervisor agent (SA) in charge of the over-all process, document agents (DA) in charge of each document image, and two types of worker agents: projection profile agents (PPA) in charge of processing each document images vertical slices, and seam carving agents (SCA) in charge of finding the separating seam. In addition to improving runtime acceleration, this architecture makes it possible to tackle more complex problems faced by other OCR systems, using collaborative processing.

1) The Supervisor Agent (SA)

In order to keep a global awareness of the state of the text recognition process, a shared interface should be deployed and made accessible for the agents in order to interact efficiently. This interface could take many forms such as a simple shared data storage system or a more intelligent solution such an interactive intelligent agent able to play a supervising role. And as such, the supervisor agent can be modeled as a simple reflex agent [18] in charge of the over-all process. Its environment can be described as a number of document images to process: for each document image, a new document agent is spawn to process it. After the separating seams are computed, the supervisor is then capable of handling the extracted lines from the document agent output (Fig. 8). Thus, the SA purpose is to serve as a synchronizing hub for the other document agents by managing tasks and extracted text lines. This interactive communication could also be used in order to synchronize the document agents to solve potential problems that can arise from the inherent concurrency of multi-agent systems.

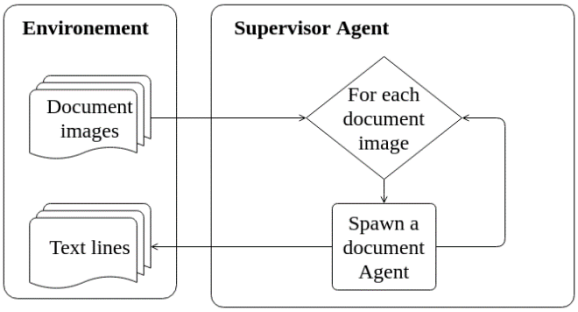

Figure 8. Interaction of the Supervisor Agent and its environment.

Load balancing is a very important aspect of performance optimization in any parallel or distributed system. According to the problem's nature, the load balancing methods used can take many forms. For instance, the Round Robin algorithm rely on sheer availability to avoid "Deny of Service" situations, where multiple instances of servers are used simultaneously providing the same service.

Other methods use efficient algorithms to save computational power available, such as the Random Polling algorithm which relay workloads to available computational units. Following the same idea as the Random Polling algorithm, the SA in our implementation spawn a fixed number of document agents (DA) and then create a shared message queue used to dispatch each document image segmentation task to the first available DA (Fig. 9) while running the DA sub processes (PPAs and SCAs) locally on a single node to avoid any communication overhead caused by the network low performance.

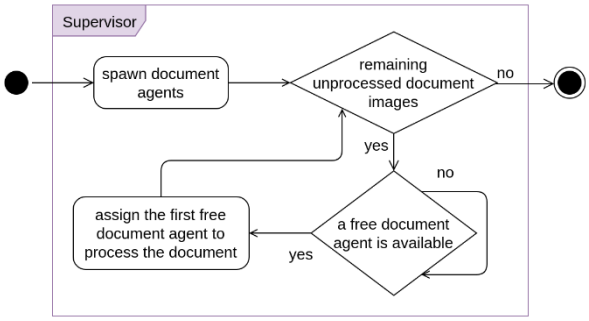

Figure 9. The taskdispatchingprocessused by thesupervisor agent based on message queues.

2) The Document Agent (DA)

This agent is the principal component of the system. Each document agent is a model-based reflex agent [18] in charge of finding the separating seams by reflex decisions made of the interaction resulting from the underlying projection profile agents (PPAs) and the seam carving agents (SCAs). This agent interacts with the document image indirectly through the PPAs and SCAs, by taking decisions based on the PPAs results, that are then used to constrain the image areas where the SCAs are deployed to define seams that separates each two adjacent text lines on the image (Fig. 10).

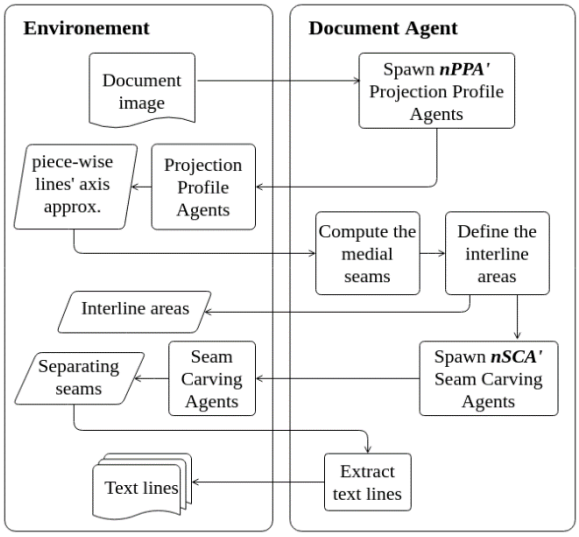

Figure 10. Interaction of the Document Agent and its environment.

The Document Agent get its parameters from the Supervisor Agent, which are set manually: the assigned document image to process, the σ value of the Gaussian filter used for noise reduction, nPPA, the maximum number of Projection Profile Agents that can be deployed, and nSCA, the maximum number of Seam Carving Agents that can be deployed.

After getting the parameters from the supervisor agent, the optimal number of worker agents is calculated. Then, a set of vertical slices are assigned to each PPA spawned in order to compute the lines axis approximations. Using the resulting medial seams approximations, the interline spaces are in turn divided into sets and then assigned to the seam carving agents in order to compute the separating seams.

Every document agent is in charge of preparing data to dispatch across PPAs and SCAs. Two new parameters are introduced to limit the number of worker agents spawn dynamically, nPPA the maximum number of the projection profile agents, and nSCA the maximum number of seam carving agents. These parameters are used to optimize the load balance according to the available computational resources and are then optimized automatically at runtime when the data is prepared before the job planning process on each document agent begins assuring an optimal load balance across the deployed agents:

$$(nPPA', nSCA') = \left(\left\lceil \frac{s-1}{\left\lceil \frac{s-1}{nPPA} \right\rceil} \right\rceil, \left\lceil \frac{l-1}{\left\lceil \frac{l-1}{nSCA} \right\rceil} \right\rceil \right) \quad (3)$$

where l is the final number of lines axis approximations and s the number of vertical slices, while *nSCA'* and *nPPA'* are the new optimized parameters. As an example, in case of a document segmentation using 5 slices and resulting in 9 interline areas to process, using 3PPAs and 4SCAs, we can see that the optimal number of PPAs stays at 3 while the optimal number of SCAs becomes 3 instead of 4, ensuring a more uniform load across the agents. This minimize the number of agents used without exceeding the minimal runtime possible as shown in (Fig. 11).

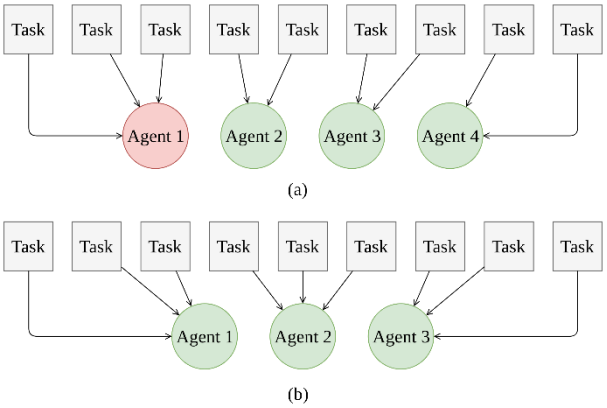

Figure 11. Example of optimization concerning the number of agents. (a) the uneven workload distribution across the agents, and (b) the workload balance after optimizing the number of agents used.

3) The Projection Profile (PPA) and the Seam Carving (SCA) agents

These agents are simple reflex agents [18], where their processing is only bound to the parameters passed by the supervising document agent, acting in partially observable environments defined by the image areas assigned by the document agent.

The PPA gets its parameters from the DA: the width of the vertical slice, the position of the first column of the assigned vertical slice relative to the complete document image, and the smoothing factor used during the local maxima extraction from the horizontal histogram of the vertical slice. As a result, the PPA have a limited vision of the environment (document image), defined by the assigned vertical slice. This does not limit the PPA from interacting with its neighbors by sharing its results.

Each PPA is in charge of detecting lines axis approximate positions by detecting the local maxima from the horizontal histogram of the processed vertical slice of the image. Then, from the agent in charge of the adjacent slice, it recovers the found positions in order to search for matching positions and then passing them as piece-wise approximations to the document agent (Fig. 12).

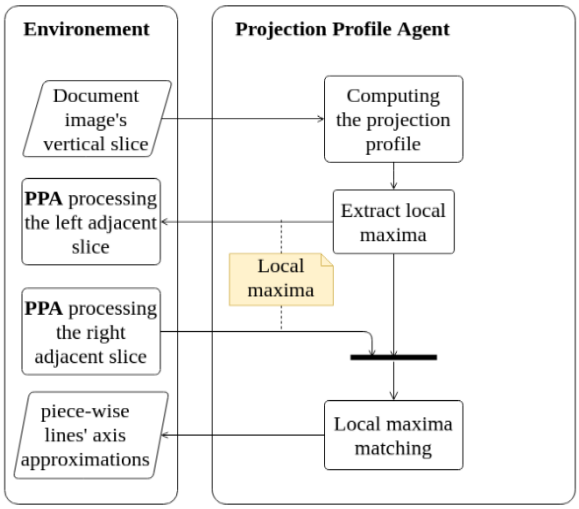

Figure 12. Interaction of the Projection Profile Agent and its environment.

The DA use the gathered piece-wise approximations to build a sparse matrix representing the detected text lines' axis. After post-processing the sparse matrix, full text lines' axis approximations are then defined by interpolating the sparse data from the matrix creating a continuous lines running horizontally from one side to the other.

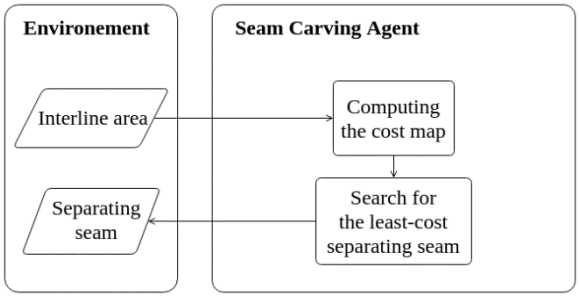

Figure 13. Interaction of the Seam Carving Agent and its environment.

Next, the seam carving agents SCAs take charge of finding the least-cost separating seam from each interline space extracted by the document agent from the PPAs resulting text lines' axis approximations (Fig. 13). The DA prepare pairs of adjacent lines approximations defining the specific area of the document image between the actual text lines where a separating seam should be defined. The pairs are then assigned to the seam carving agents in order to constrain the search space. To be memory efficient, the cost map is computed for each interline area separating each two consecutive medial seams S_i and S_{i+1} using a cost map of a width equal to the width of the original image, and a height H_i (Fig. 14):

$$H_i = \max S_{i+1} - \min S_i \quad (4)$$

The cost is only calculated for pixels in between the two medial seams while any other pixel position outside of the interline space are assigned a maximum cost value C_{max}.

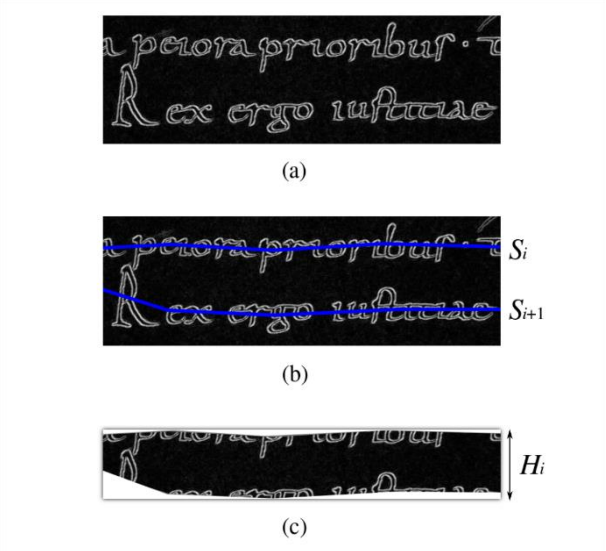

Figure 14. The cropped energy map used to calculate the cost map between each two adjacent text lines. (a) the original energy map, (b) the detected text lines, and (c) the new cropped energy map.

V. Performance tests

A. Infrastructure

For the tests, we used a Linux powered Beowulf cluster made of two nodes, each machine is a simple commodity computer powered by Intel dual-core processors i5 4th generation 4200U (2.60GHz, 3MB of cache) using 4GB DDR3 RAM. As for the network used for communication, the nodes are deployed through a local 10/100 Ethernet.

In parallel computation, one of the main problems of performance optimization is managing data, dealing with several aspects, such as data access, data locality, and data communication. For computational heavy methods, data managing can be relatively ignored as most of the load will be generated by computation instead of data access or manipulation. Other methods rely on massive data manipulation to operate, as it is the case of image analysis. The data managing represents a massive portion of the process, where transmitting big amounts of data through the parallel network can be costly, and negatively affect the performance in case of sub-par data management.

In our deployment, the document images dataset is accessed by the agents through an NFS server which simplify the data access, this is more efficient than sending parts of the image over the network to each worker agent especially when the images are high resolution and the network speed is noticeably slower than what a direct file system access can deliver. This is made possible by using the local cache system which is a part of NFS, making document access faster than direct requests over the network.

B. Environment

Practical tests of this multi-agent system were deployed on a Linux operated cluster using Pythons MPI4Py, a wrapper library for MPI routines based on C/C++ implementations of the standard. The main reason for choosing MPI as communication platform, was its ease of use to implement inter agent communication routines.

C. Efficiency

Based on the prior work [2], we searched for a way to improve the Seam Carving approach by addressing some of its weaknesses. Unidirectional seam carving is by far the most known variant used in image processing. Unfortunately, the unidirectional carved seams can fail to find the best visually satisfying seam as in many cases, the backtracking algorithm have to decide what position to choose from two to three next positions having the same energy cost. In our case of text segmentation, we can safely assume that the text lines are somewhat horizontal and as such, we can choose the next horizontal position on the same line as the standard next position.

Another problem arises when dealing with extending characters and touching text components, where the right to left carved seam can be different from the left to right one.
As a result, we developed a new method based to exploit two carved seams of different directions, that we can merge into one, by only keeping the farthest position from the medial seams for each column (Alg. 2).

Data:
$imageWidth$: image width in pixels
$medSeam_{i(+1)}$: medial seams
$carvSeam_{RTL}$: carved seam from right to left
$carvSeam_{LTR}$: carved seam from left to right
Result: $finSeam$: final carved seam
for $k \leftarrow 0$ **to** $imageWidth$ **do**
  $mid \leftarrow (medSeam_i[k] + medSeam_{i+1}[k])/2$
  $distR \leftarrow |mid - carvSeam_{RTL}[k]|$
  $distL \leftarrow |mid - carvSeam_{LTR}[k]|$
  if $distR >= distL$ **then**
    $finSeam[k] \leftarrow carvSeam_{LTR}[k]$
  else
    $finSeam[k] \leftarrow carvSeam_{RTL}[k]$
  end
end

Algorithm 2. Algorithm used for bidirectional seam carving.

The new approach seemed to be promising, unfortunately it had little to no effect on the final results. The segmentation of some of the document images, showed a huge improvement, especially for the matching score threshold of 95%, while other images had lower F1 scores than using the original method (Fig. 15). For MS = 90%, the result was the same in case of Fig. 15(a), with an improvement for MS = 95%, from 91.42% to 100% (Fig. 15(b)). The same can't be said for Fig. 15(c), as the segmentation deteriorated for MS = 95%, from 100% to 90.90% (Fig. 15(d)).

Unfortunately, even if the score stayed relatively similar to the original method, the execution time had almost doubled as the SCAs had to compute the cost map twice, from left to right and right to left, making this bidirectional approach very

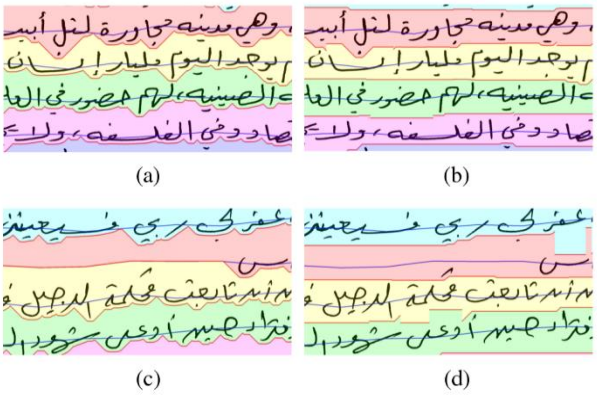

Figure 15. Segmentation result of the Bidirectional Seam Carving approach.

Table 2. Results of the 123 document images segmentation with the associated runtime measures.

slices	sigma	smooth	F1 Score (%)		Runtime (min)
			MS=90%	MS=95%	
8	10	0.01	88.87	80.59	30.34
8	10	0.02	95.37	88.53	27.32
8	10	0.03	96.3	89.55	27.88
8	15	0.01	95.5	90.34	27.73
8	15	0.02	96.25	91.06	27.47
8	15	0.03	95.82	90.81	26.74
8	20	0.01	95.16	90.21	30.79
8	20	0.02	96.72	91.96	31.89
8	20	0.03	95.9	91.05	33.15
12	10	0.01	46.33	37.34	37.61
12	10	0.02	93.13	86.26	28.19
12	10	0.03	97.89	91.4	29.71
12	15	0.01	75.4	68.35	34.07
12	15	0.02	98.44	95.06	33.15
12	15	0.03	98.53	95.05	30.77
12	20	0.01	71.07	64.98	39
12	20	0.02	98.51	95.77	34.11
12	20	0.03	98.64	95.83	34.77
16	10	0.01	19.27	13.53	55.42
16	10	0.02	67.6	56.56	32.26
16	10	0.03	93.35	86.18	27.85
16	15	0.01	24.37	17.77	47.6
16	15	0.02	92.25	88.07	33.33
16	15	0.03	98.54	94.67	31.53
16	20	0.01	13.62	10.79	54.27
16	20	0.02	94.44	91.26	32.39
16	20	0.03	98.75	96.18	30.81

inefficient. This pushed us to focus more on the parametrization of the original algorithm.

The main parameters of the Seam Carving text segmentation described in our prior work [2] are:

- *slices*: the number of vertical slices used in the projection profile.
- *sigma*: the standard deviation of the Gaussian filter used for noise reduction.
- *smooth*: the smoothing factor of the smoothing spline used when extracting the local maxima during the projection profile phase.

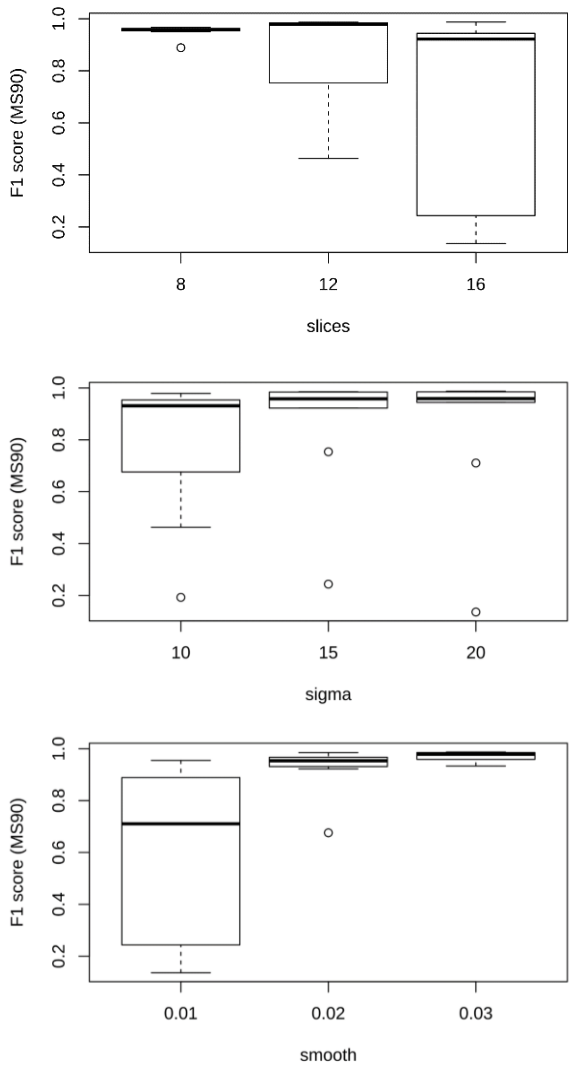

Figure 16. Boxplot showing the distribution of the segmentation's F1 Score according to the parameters values.

slices remain the most important parameter, as it's used to define the document image sampling: a low number of slices can result in a low text lines detection rate, when an excessive number can make the algorithm oversensitive. On the other hand, *sigma* and *smooth* are complementary in order to reduce the noise respectively from the document image and the horizontal histograms of the projection profile.

In addition to the segmentation performance, these parameters values may also influence the runtime speed depending on the number of potential text lines detected which means having more interline areas to segment for the SCAs.

D. segmentation results

The first test was conducted on a simple sequential implementation of the multi agent system (1xDA, 1xPPA, 1xSCA) locally on a single machine (Tab. 2). A number of parameters combinations were used to segment the total number of the images available on the dataset [15].

As mentioned earlier, those results prove that the parameters have an impact on both aspects of performance, the F1 Score and the runtime (Fig. 16). According to those results, it's always better to have a high number of vertical slices to better sample the document image irregularities, the only problem

Table 3. Comparison of results.

(slices, sigma, smooth)	F1 Score (%)	DA × (PPA,SCA) (min)								
	(MS=90%)	1×(1,1)	1×(2,2)	1×(3,3)	2×(1,1)	2×(2,2)	3×(1,1)	3×(2,2)	4×(1,1)	4×(2,2)
(16, 20, 0.03)	98.75	30.81	30.67	45.42	28.72	29.66	30.03	142.62	27.87	170.62
(12, 20, 0.03)	98.64	34.77	34.61	46.08	30.15	30.83	30.16	138.27	29.53	194.33
(16, 15, 0.03)	98.54	31.53	31.7	40.36	28.15	28.78	26.42	118.27	28.99	175.97

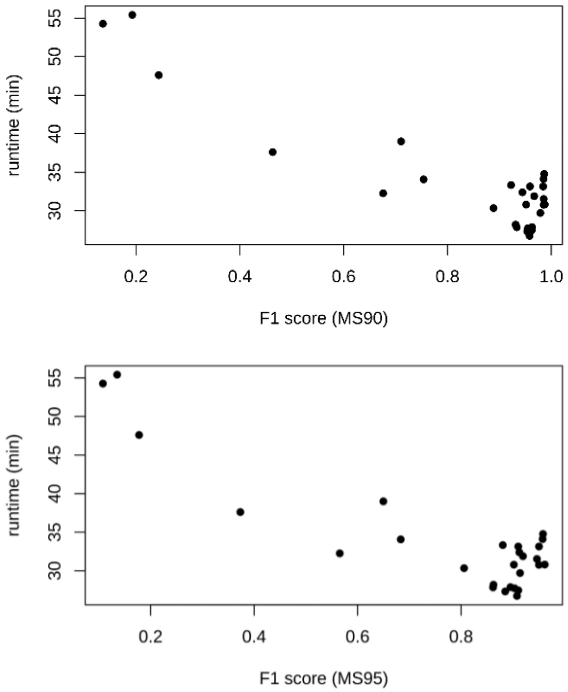

Figure 17. Correlation of the segmentation's F1 Score with the segmentation runtime.

arising with high-sampling is the induced noise that must be dealt with by finely tuning the sigma and smooth parameters.

E. Acceleration

It was also shown in the last results that the F1 Score of the segmentation is inversely correlated with the runtime. This can be explained by the high number of false positive detections of text lines when using low value for *sigma* and *smooth* (Fig. 17). In Addition to the segmentation parameters, the number of the deployed agents plays also an important role in the runtime speedup. To test the parallel and distributed segmentation, a number of agents' combinations were tested on the three top parameters sets from the previous segmentation tests (Tab. 3). The results show that the Projection Profile and Seam Carving phases do not draw huge benefits from parallelism as they are very throughput and communication intensive processes, where the fastest runtimes were accomplished using a minimum of PPAs and SCAs for as much of DAs as the system can handle.

VI. Conclusion and Future Work

The classical sequential approach to the OCR problem, sets unnecessary limitations while building new systems. Recognition problems are dealt with according to a sequential processing where each step rely heavily on the last steps results, and as a result, no efficient and complex collaboration of the system components are thought of during the design process. Using a parallel architecture can free the system components making it possible to design new heavily collaborative systems that are able to solve complex problems by combining efforts of multiple kinds of specialized agents. The system discussed in this article shows an interesting ground for developing these new intelligent parallel and distributed OCR systems. The low complexity and high adaptability of bottom-up approaches like the seam carving technique are key elements for developing highly collaborative multi-agent systems.

In future works, we intend to improve on several points, notably the parallel calculations, and more cooperative processing of data between the multiple agents adding an additional efficiency to the method. Furthermore, the parallelism opens new doors for building more complex and intelligent collaborative systems, in effort to solve more complex problems faced by common sequential OCR software.

As a side note, the proposed parallel system in this article is mainly using the NFS cache redundancy to avoid communication overhead. Other solutions could be more efficient in case of processing massive numbers of document images, such as using a distributed file system like the Hadoop toolkit which offers a distributed file system HDFS and a programming model MapReduce capable of taking advantage of the data locality in an efficient workload balancing architecture. Other computing frameworks were also built using HDFS and the YARN resource manager, components of Hadoop that grant a possibility of taking advantage of data locality. These solutions are diverse, and deserve to be further explored for their ability to deploy efficient multi-agent systems on large scales.

References

[1] Rayner, K., 1998. "Eye movements in reading and information processing: 20 years of research". *Psychological bulletin*, 124(3), p.372.

[2] M. Daldali, A. Souhar., 2018. "Handwritten Arabic Documents Segmentation into Text Lines using Seam Carving". *International Journal of Interactive Multimedia and Artificial Intelligence*. http://dx.doi.org/10.9781/ijimai.2018.06.002

[3] Amer, I.M., Hamdy, S. and Mostafa, M.G., 2017. "Deep Arabic document layout analysis". In *Intelligent Computing and Information Systems (ICICIS), 2017 Eighth International Conference* (pp. 224-231). IEEE.

[4] Cruz, F. and Terrades, O.R., 2018. "A probabilistic framework for handwritten text line segmentation". arXiv preprint arXiv:1805.02536.

[5] Singh, B., Gupta, N., Tyagi, R., Mittal, A. and Ghosh, D., 2011. "Parallel implementation of Devanagari text line and word segmentation approach on GPU". *International Journal of Computer Applications*, 24(9), pp.7-14.

[6] Ahmed, K., Qiu, Q., Malani, P. and Tamhankar, M., 2014. "Accelerating pattern matching in neuromorphic text recognition system using intel Xeon phi coprocessor". In

Neural Networks (IJCNN), 2014 International Joint Conference (pp. 4272-4279). IEEE.

[7] Khemakhem, M. and Belghith, A., 2009. "Towards A Distributed Arabic OCR Based on the DTW Algorithm: Performance Analysis". *International Arab Journal of Information Technology (IAJIT)*, 6(2).

[8] Volk, M., Marek, T. and Sennrich, R., 2010. "Reducing OCR errors by combining two OCR systems". In *ECAI-2010 workshop on language technology for cultural heritage, social sciences, and humanities* (pp. 6165).

[9] Boulid, Y., Souhar, A. and El Kettani, M.E.Y., 2016. "Detection of Text Lines of Handwritten Arabic Manuscripts using Markov Decision Processes". *IJIMAI*, 4(1), pp.31-36.

[10] Boulid, Y., Souhar, A. and El Kettani,M.Y.,2015. "Arabic handwritten text line extraction using connected component analysis from a multi agent perspective". In *Intelligent Systems Design and Applications (ISDA), 2015 15th International Conference* (pp. 80-87). IEEE.

[11] Qiu, Q., Wu, Q., Bishop, M., Pino, R. and Linderman, R., 2012. "A parallel neuromorphic text recognition system and its implementation on a heterogeneous high performance computing cluster". *IEEE Transactions on Computers*.

[12] Avidan, S. and Shamir, A., 2007. "Seam carving for content-aware image resizing". In *ACM Transactions on graphics (TOG)* (Vol. 26, No. 3, p. 10). ACM.

[13] Saabni, R. and El-Sana, J., 2011. "Language independent text lines extraction using seam carving". In *Document Analysis and Recognition (ICDAR), 2011 International Conference* (pp. 563-568). IEEE.

[14] Arvanitopoulos, N. and Susstrunk, S., 2014. "Seam carving for text line extraction on color and grayscale historical manuscripts". In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference* (pp. 726-731). IEEE.

[15] Handwritten Arabic Proximity Datasets. Language and Media Processing Laboratory. Web: http://lampsrv02.umiacs.umd.edu/projdb/project.php?id=65

[16] Duarte, R. and Sendag, R., 2012. "Accelerating and characterizing seam carving using a heterogeneous cpu-gpu system". In *PDPTA*.

[17] Stultz, J. and Edelman, A., 2008. "Seam Carving: Parallelizing a novel new image resizing algorithm". Project Report (Online).

[18] Russell, S.J. and Norvig, P., 2016. "Artificial intelligence: a modern approach". Malaysia; Pearson Education Limited.

[19] Fischer, A., Frinken, V., Forns, A. and Bunke, H., 2011.TranscriptionalignmentofLatinmanuscriptsusing hidden Markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing* (pp. 29-36). ACM.

Author Biographies

Mehdi Daldali received the Master's degree in Big Data and Cloud Computing in 2018 from University Ibn Tofaïl, Faculty of Science, Kénitra, Morocco. In 2017, he was certified as Big Data Developer and Business Intelligence Analyst by the IBM MEA Academy. His research interests include high performance computing and pattern recognition.

Abdelghani Souhar is a full Professor of computer science at the University of Ibn Tofaïl, Faculty of science Kénitra Morocco. He received the M.S. degree in applied Mathematics in 1992, PhD degree in computer science in 1997 from the University of Mohammed 5 in Rabat-Morocco. His habilitation thesis concerned Modeling complex systems as Arabic handwritten recognition and an intelligent system for generating mesh. His research interests include automatic processing of Arabic language, Modeling complex systems, CAE/CAD, and Artificial Intelligence.