Parallel Processing Considerations for Image Recognition Tasks

Steven J. Simske

HP Laboratories
HPL-2011-20

Keyword(s):

OCR, Document Classification, Image Authentication, Image Forensics, Meta-Algorithmics, Workflow, Predictive Selection, Precision, Document Analysis

Abstract:

Many image recognition tasks are well-suited to parallel processing. The most obvious example is that many imaging tasks require the analysis of multiple images. From this standpoint, then, parallel processing need be no more complicated than assigning individual images to individual processors. However, there are three less trivial categories of parallel processing that will be considered in this paper: parallel processing (1) by task; (2) by image region; and (3) by meta-algorithm. Parallel processing by task allows the assignment of multiple workflows -- as diverse as optical character recognition [OCR], document classification and barcode reading -- to parallel pipelines. This can substantially decrease time to completion for the document tasks. For this approach, each parallel pipeline is generally performing a different task. Parallel processing by image region allows a larger imaging task to be sub-divided into a set of parallel pipelines, each performing the same task but on a different data set. This type of image analysis is readily addressed by a map-reduce approach. Examples include document skew detection and multiple face detection and tracking. Finally, parallel processing by meta-algorithm allows different algorithms to be deployed on the same image simultaneously. This approach may result in improved accuracy.

External Posting Date: February 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: February 21, 2011 [Fulltext]
Published and presented at Electronic Imaging 2011, San Francisco CA, 23-27 January 2011

Parallel Processing Considerations for Image Recognition Tasks

Steven J Simske
Hewlett-Packard Labs, 3404 E. Harmony Rd., MS 36, Fort Collins, CO, USA 80528

ABSTRACT

Many image recognition tasks are well-suited to parallel processing. The most obvious example is that many imaging tasks require the analysis of multiple images. From this standpoint, then, parallel processing need be no more complicated than assigning individual images to individual processors. However, there are three less trivial categories of parallel processing that will be considered in this paper: parallel processing (1) by task; (2) by image region; and (3) by meta-algorithm. Parallel processing by task allows the assignment of multiple workflows—as diverse as optical character recognition [OCR], document classification and barcode reading—to parallel pipelines. This can substantially decrease time to completion for the document tasks. For this approach, each parallel pipeline is generally performing a different task. Parallel processing by image region allows a larger imaging task to be sub-divided into a set of parallel pipelines, each performing the same task but on a different data set. This type of image analysis is readily addressed by a map-reduce approach. Examples include document skew detection and multiple face detection and tracking. Finally, parallel processing by meta-algorithm allows different algorithms to be deployed on the same image simultaneously. This approach may result in improved accuracy.

Keywords: OCR, Document Classification, Image Authentication, Image Forensics, Meta-Algorithmics, Workflow, Predictive Selection, Precision, Document Analysis

1. INTRODUCTION

Image recognition tasks running the gamut from handwriting and character recognition to automatic image analysis are well-suited to parallel processing. Obviously, many imaging tasks—ranging from video analysis and multi-face recognition to parallel processing of printed information in multiple regions of interest in an image—can benefit from the simultaneous analysis of multiple images. From this standpoint, then, parallel processing need be no more complicated than assigning individual images to individual processors. However, parallel processing of the same image by multiple, parallel pipelines can also be advantageous. With this in mind, three categories of parallel processing will be considered in more depth in this paper: parallel processing (1) by task; (2) by image region; and (3) by meta-algorithm.

Parallel processing by task, described in Section 2, allows the assignment of multiple workflows to parallel pipelines. This can substantially decrease time to completion for the document tasks. For this approach, each parallel pipeline is generally performing a different task. Examples of this type of parallel processing are optical character recognition (OCR), handwriting recognition, template matching, document classification, text order determination, form information extraction, language detection, image/document inspection, image forensics and/or authentication, barcode or other data-containing mark reading, etc. Parallel processing by task often benefits from a hybrid approach: serial steps are performed first to determine the image segmentation into sub-segments; then parallel processing of the image sub-sections occurs; and finally serial analysis of the results of the parallel processing is used to complete the image recognition workflow.

Parallel processing by image region, described in Section 3, allows a larger imaging task to be sub-divided into a set of parallel pipelines, each performing the same task but on a different data set. This type of image analysis is readily addressed by a map-reduce approach. Examples include document skew detection and multiple face detection and tracking. Other multiple imaging segmentation systems include slide and negative scanning and image quality assurance/inspection.

Parallel processing by meta-algorithm, described in Section 4, allows different algorithms to be deployed on the same image simultaneously. This approach may result in improved accuracy. Patterns to be described in the paper include voting, predictive selection, tessellation and recombination and speculative parallelism. We will show how these techniques are robust to sparse or non-Gaussian training data, and relatively inexpensive from a throughput standpoint,

especially for multi-core systems. Moreover, meta-algorithmic approaches to image recognition tasks are highly advantageous for accuracy and functional imaging purposes. This section also overviews the application of meta-algorithms to document segmentation and image classification.

Finally, the paper concludes with a discussion of how the right parallel processing architecture (serial-parallel, parallel-serial and other hybrids or pure parallel), can be chosen for the image recognition application, based on the system needs—throughput and accuracy in particular.

2. PARALLEL PROCESSING BY TASK

I use the term "parallel processing" in a very general sense throughout this paper. So, when I describe "parallel processing by task" I simply mean taking a task that is, on a rudimentary computing device like a laptop, single-core workstation, etc., performed serially by default. By this definition, then, intelligent multi-core systems may already perform parallel processing by task, although usually this requires multi-threaded coding on the part of the image processing application developer.

So, in general, "by task" will mean here the assignment of multiple workflows to parallel pipelines that would otherwise be performed sequentially by the same processor. With this wide net, I can catch a large number of image recognition workflows. With the pipelines synchronizing their tasks, this can substantially reduce the time-to-completion of an image recognition task. Effectively, each parallel pipeline is performing a different task.

In the following sub-sections, I will describe image recognition tasks associated with security printing, scene understanding, optical character recognition (OCR) and handwriting recognition, template matching, document classification, text order determination, form information extraction, language detection and image/document inspection. While not an exhaustive list by any means, I chose these to try to cover a large gamut of applications.

2.1 Security Printing

Security printing is concerned with the printing of information that can be read—and acted upon—at a later time, either by a person or a machine. Security printing marks include barcodes, steganographic marks (including digital watermarks), microtext and guilloche patterns.

A combination of security-related printed marks is shown in a security medallion (Figure 1). When the medallion is printed and then later scanned, it is segmented[1,2] into eight regions

A familiar 2D barcode (a 2D DataMatrix[3] barcode) is located in the lower right, above a set of color bars. Each uses approximately 25% error-correcting code (ECC), and so contains more than 200 bits of data each. The software used to the read the 2D barcode is run in a separate thread from the software to interpret the color bars. In the example of Figure 1, the 2D DataMatrix barcode is used to make the barcode compliant with standards (such as GS1), so that it can be used for point-of-sale, as part of the track-and-trace/supply chain visibility for the product, and increasingly for consumer/brand interaction[4]. The color bars themselves comprise 16 different colors and are used, additionally, to determine the color gamut of the printer used on the label. This can be used for quality control, as well as replicating the data (including the ECC) in the barcode.

To the left of the 2D barcode is a 3D (color) barcode, consisting of color tiles. There are 14 data tiles of 2 bits each (cyan, magenta, yellow and black colors, or CMYK) along with a color checkbit (in green). An additional 28 bits are stored in the "hybridized" data carrier in the upper left portion of the medallion. The colors {CMYK} are automatically calibrated by the four large tiles in the lower left, which are readily imaged by any mobile camera. The color tiles themselves are readily imaged by most mobile imagers—note that these tiles are roughly 2 x 2 mm in size, allowing a more reliable "read" than for the 2D DataMatrix tiles. By "hybridization" is meant a direct relationship between the two coded marks—examples include replication, scrambling, and digital signing.

Above the 2D barcode is a product-specific image. This continuous tone image is suitable for forensic analysis[5], and can also be used to convey product branding information or carry a digital watermark payload. Finally, in the upper right portion of the medallion is a set of microtext and brand-specific logos and photos.

Combined, the medallion contains eight different imaging segments, or regions, which can be processed in parallel. The first stage is to segment the medallion image (e.g. captured by a scanner or mobile camera) into these eight segments—(1) barcode, (2) color bars, (3) color tiles, (4) color tile calibrating colors, (5) hybridized color tiles, (6) continuous tone

image, (7) microtext, and (8) logos/branded images—which can generally be accomplished using a lower-resolution, or down-sampled, representation of the image. Next, the analyses of these eight regions—as described above—can be performed in parallel.

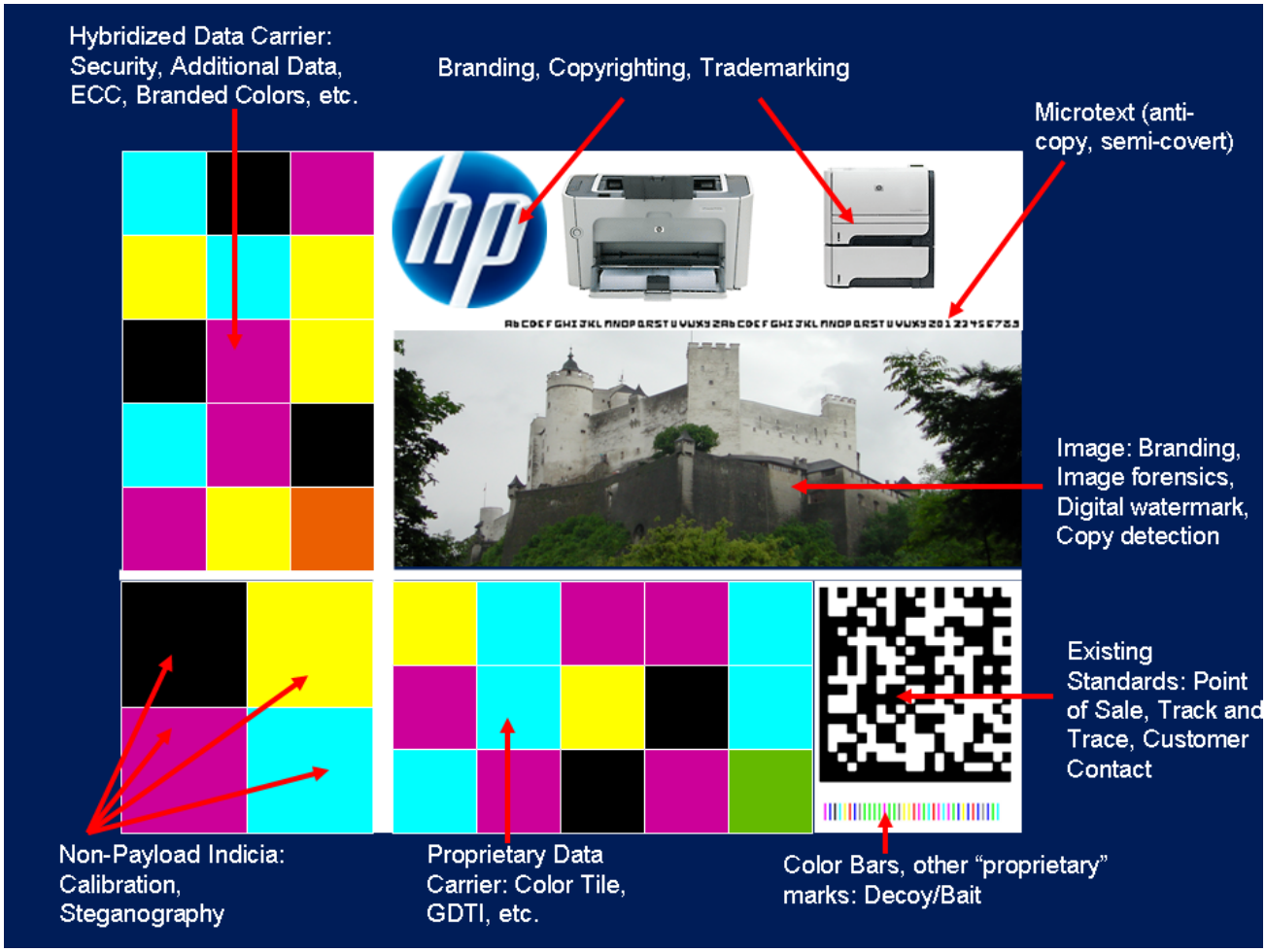

Figure 1. Multi-purpose "printed information" seal. Original seal is roughly 1.5 cm in height and 2.5 cm in width. Eight distinct pipelines are enabled from this single seal, as described in the text.

2.2 Scene Understanding

Having seen how image segmentation and subsequent assignment of each image segment to a separate, parallel pipeline can be used in security printing, it is easy to see the applicability of the same approach to other image understanding tasks. Collectively, we call these tasks "scene understanding" tasks. Facial recognition is an example. An image can be segmented at low resolution into likely facial regions, using for example hue-based segmentation. Next, each of the potential face regions can be processed in parallel by a face recognition engine.

Other examples of scene understanding include landmark recognition, surveillance and other biometric and medical imaging application. For example, a retinal scan is in many ways similar to extracting rivers from an image.

2.3 Optical Character Recognition (OCR) and Handwriting Recognition

Optical character recognition (OCR) is the analysis of text images to convert them into machine-usable (e.g. ASCII, Unicode) data. OCR is well-suited to parallel processing since different regions of a scanned image can be analyzed

simultaneously by multiple OCR engines (analysis routines). In an earlier book-digitization project, we assigned multiple pages, or multiple documents, to their own OCR pipelines[6]. Since OCR was the time-limiting factor, the operation scaled more or less linearly with the number of processors we could attach the high-speed interconnected system (OCR-ready regions are typically quite small in size, and region/file transmission time is typically less than 5% of the processing time). Handwriting recognition has essentially the same approach, except that handwriting, not text, is analyzed.

OCR can also be used in combination with scene recognition (Section 2.2). We previously[7] built a system to extract all likely text regions from a natural scene (captured by a mobile camera). Each likely text region can then be processed in parallel by one or more OCR engines and then translated. In our case, we performed OCR with three OCR engines, combined the results and then translated the result into French. Parallel processing would have sped up the OCR phase by 50-60% in most cases, and further processing improvement would be gained in any scene with more than one candidate text region.

2.4 Template matching

With template matching, we are concerned with comparing the information in an image to the information in the template(s). Depending on the type of image—security printing (Figure 1), document, natural scene, etc.—a wide variety of imaging tasks may be required, including the aforementioned OCR and scene understanding. Nevertheless, template matching is readily suited for parallel processing, since it usually involves comparing several elements to the templated elements.

2.5 Document Classification

Document classification can be considered, at minimum, to be one of two types. The first, and perhaps less interesting, type of document classification is finding out what type of document it is—e.g. loan application form, utilities bill, personal health record, etc. This type of document classification is roughly equivalent to template matching. I term this document classification by style.

The other, perhaps more interesting, type of document classification is when the content of the document determines its class[8]. Here, the document must be analyzed for all salient data—linguistic and/or image-related. Mixed-type documents (containing text, image, graphics, etc.) can be segmented along the lines of the security printing approach described in Section 2.1 above, and each region assigned to the appropriate classifier(s) for data recognition and extraction.

2.6 Text Order Determination

Determining the text order, or flow of the text, in a document is an important problem in article extraction from a larger document, such as a magazine, blog, or other digest/mash-up. Effectively, text ordering comprises extracting the key words and/or semantics from each text region (which can include figure headings, tables, etc.) and comparing them to the other text regions that may precede or follow them in sequence. Parallel processing can provide an order of N, or $O(N)$—where N=the number of text regions—speed up for the initial processing. This may include OCR in addition to word/semantic extraction. Parallel processing may provide an $O(N^2)$ improvement in subsequent processing, since for N regions there are N(N-1)/2 comparisons for sequencing.

2.7 Form Information Extraction

Forms are specialized documents which require both template matching (see Section 2.4 above, please) and then salient information extraction. The latter is termed *document indexing*. Extracting the document indices is important for determining that all of the parts of the form have been filled correctly. As such, a parallel operation can be performed on each field in the form.

2.8 Language Detection

Language detection is an interesting problem. Different sections of a document may contain different languages—instruction sets, for example, often include several languages, each nominally representing the same information. Also, foreign words can be included in the text—*faux pas* and *Gesundheit* being two obvious examples. Nevertheless, designation of the main language is a logical task—the process by which a language is determined using parallel processing is effectively the same as for OCR.

2.9 Image/Document Inspection

Image and document inspection are related to scene interpretation. Instead of tagging a scene or element in a scene—e.g. with a label such as “face” or “face of Mahatma Gandhi” of “Mahatma Gandhi’s left eye”—an inspection process determines salient features about an image, such as SSIM and VIF[9], that provide an indication of printing success. Inspection is readily coupled onto other, specific, data-reading tasks, such as reading color barcodes during manufacturing/production, and entering the sequences read into the production database. In this case, inspection seems very much like a security printing application (see Section 2.1, please).

3. PARALLEL PROCESSING BY IMAGE REGION

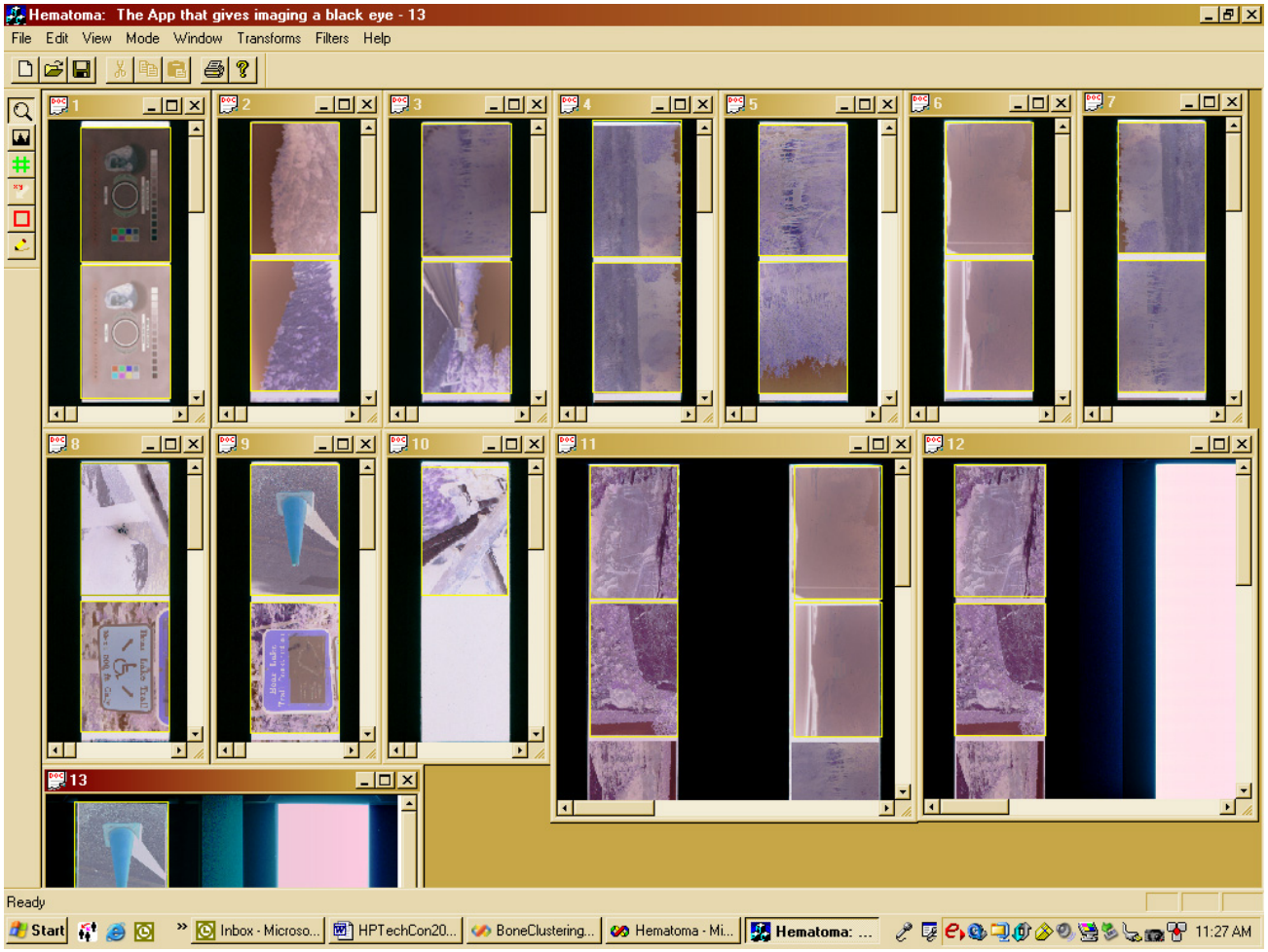

Figure 2. Example of a parallel processing by image region implementation. In this example, the slides captured from 13 different slide adapter-equipped scanners are shown. 13 parallel processors can be used for analysis of the slides (and parallel processing by image). In addition, the individual images can be parallel processed by task, as described in Section 2.

Parallel processing by image region allows a larger imaging task to be assigned to a set of parallel pipelines, each performing the same task but on a different data set. Image analysis of this type is readily amenable to a familiar cloud computing approach, map-reduce, when for example a large set of images is being analyzed to find a single item. Examples of parallel processing by image region include (multi-page) document skew detection and multiple face detection and tracking. The difference between this approach and parallel processing by task (Section 2) may be only

semantic in certain cases—OCR and language detection, for example—but in this section I define "by image region" to be concerned with tasks that comprise multiple images **before** segmentation.

From that perspective, simultaneous multiple image scanning—such as used for slide and negative scanning and image quality assurance/inspection—is readily addressed through this type of parallel processing. Figure 2 shows this for simultaneously slide extraction from many scanned images.

Another type of parallel processing by image region is when the same image region is analyzed in parallel by two entirely different algorithms/engines/services. Figure 3 illustrates this for a simple scanned document page. On the left, the document is being inspected for print defects[9] and template matching; on the right, it is being segmented—in this case, using a ground truthing application[10]—prior to performing OCR and document analysis.

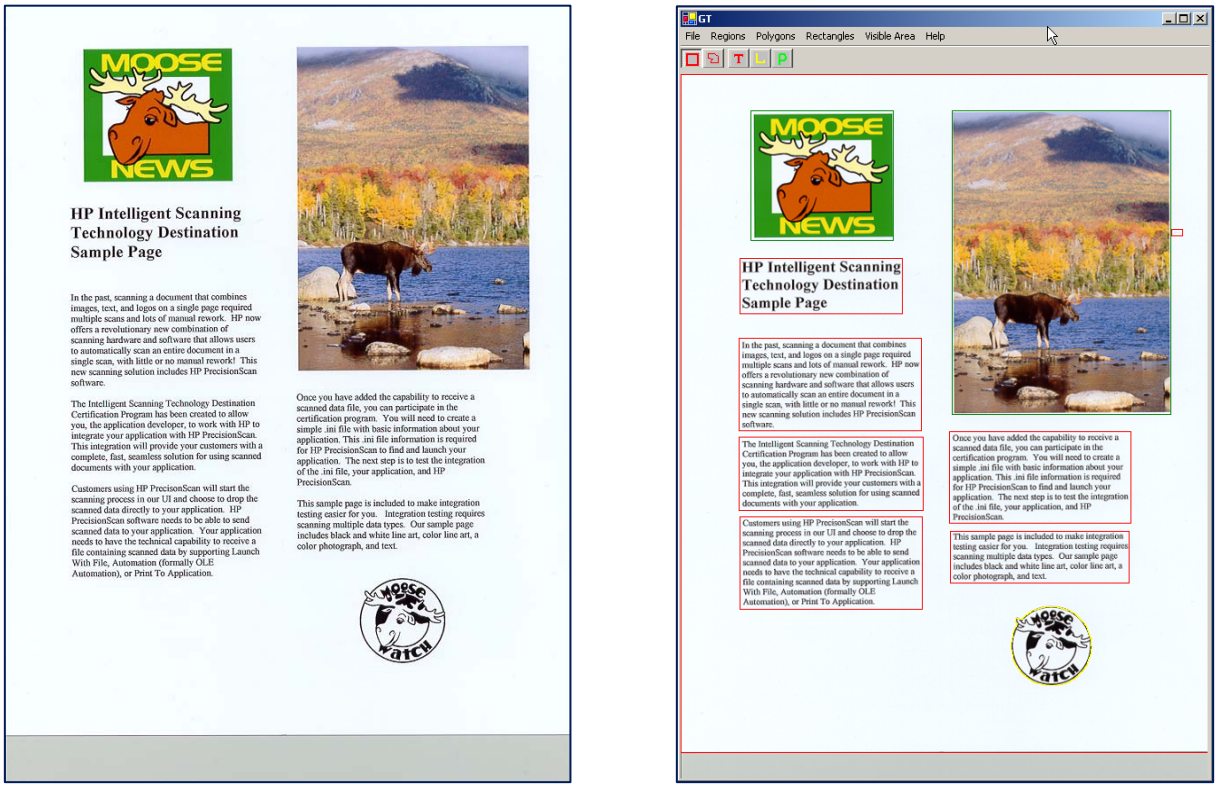

Figure 3. Parallel processing by image region for a mixed-content document page. On the left, the document has been scanned and is being analyzed by an image inspection algorithm to determine scanned image quality. On the right, the document is being ground truthed[10] for document digitization purposes. The regions shown are analyzed by an OCR and document archiving/digitization process.

In Figure 4, the concept of parallel processing of a single image by multiple analysis engines is illustrated. In this system, we used three different page zoning (which perform segmentation into regions; classification of regions; and clustering of like regions as appropriate) systems, followed by three different OCR engines, and then broke up all of the regions into their smallest possible representation such that no region created by any engine, however small, was combined into another region at the end of the zoning stage. This process is called tessellation, and is an important part of a key meta-algorithmic pattern (See Section 4.2, below, please). For the purposes of this Section, it is important to note that the tessellation comes from a parallel-series design. Next, multiple region (connected component) classifiers are used in parallel. The output from these classifiers is intelligently combined to create the final document analysis output. Thus, the system of Figure 4 uses a parallel-series-parallel-series architecture. Importantly, the parallel steps are the ones which take (by far) the most processing time. The overall design thus utilizes parallel processing where it is most important—where it can most ameliorate throughput performance. The "series" portions are required since they

involve decisions based on the whole set of data output by the parallel portions. Because of non-uniform performance among the zoning, OCR and classification engines, the overall system performance is improved less than the "theoretical" maximum of 66.7%, but by a still-impressive 50-55% depending on the document type.

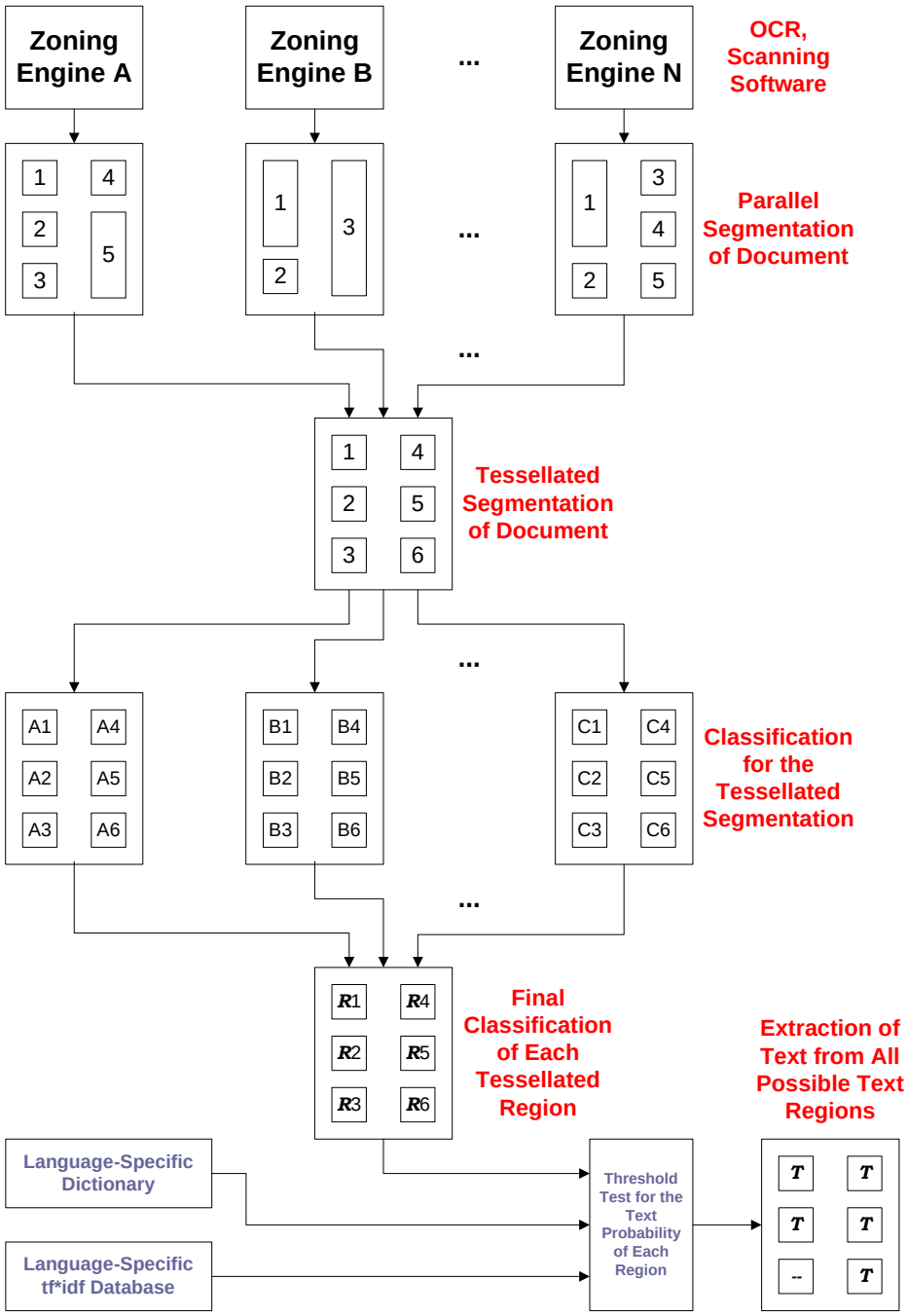

Figure 4. Parallel processing of a single document by multiple zoning and OCR engines. See text for description.

4. PARALLEL PROCESSING BY META-ALGORITHM

The word “algorithm” can take on a lot of different meanings, depending on context. It can be a simple algorithm, like a mathematical equation. It can be a coded algorithm, translating one or mathematical equations into executable machine code. Or it can be a large system for analysis, such as an OCR engine, an ASR (automatic speech recognition) engine, or a classifier. This is the interpretation of “algorithm” assumed when I introduce the concept of a meta-algorithm. A meta-algorithm is, according to the etymology of the Greek prefix “meta”, an abstraction of an algorithm to elaborate it. As such, we define a meta-algorithm as the means of using two or more algorithms in synchrony to provide a better overall result than any of the single algorithms. Meta-algorithms are ideal for parallel processing, and for this reason they are a primary focus of this paper.

Parallel processing by meta-algorithm allows different algorithms to be deployed on the same image simultaneously. So far, this is in agreement with the parallel processing by image region diagrammed in Figure 4. Meta-algorithms, however, must have a prescribed pattern for combining the output of these multiple algorithms, and also must have a training set to optimize the pattern chosen for a given problem. This approach may result in improved accuracy.

Patterns to be described in this Section are voting, tessellation and recombination predictive selection and constrained substitution. These techniques are robust to sparse or non-Gaussian training data, and relatively inexpensive from a throughput standpoint, especially for multi-core systems. Moreover, meta-algorithmic approaches to image recognition tasks are highly advantageous for accuracy and functional imaging purposes.

4.1 Voting

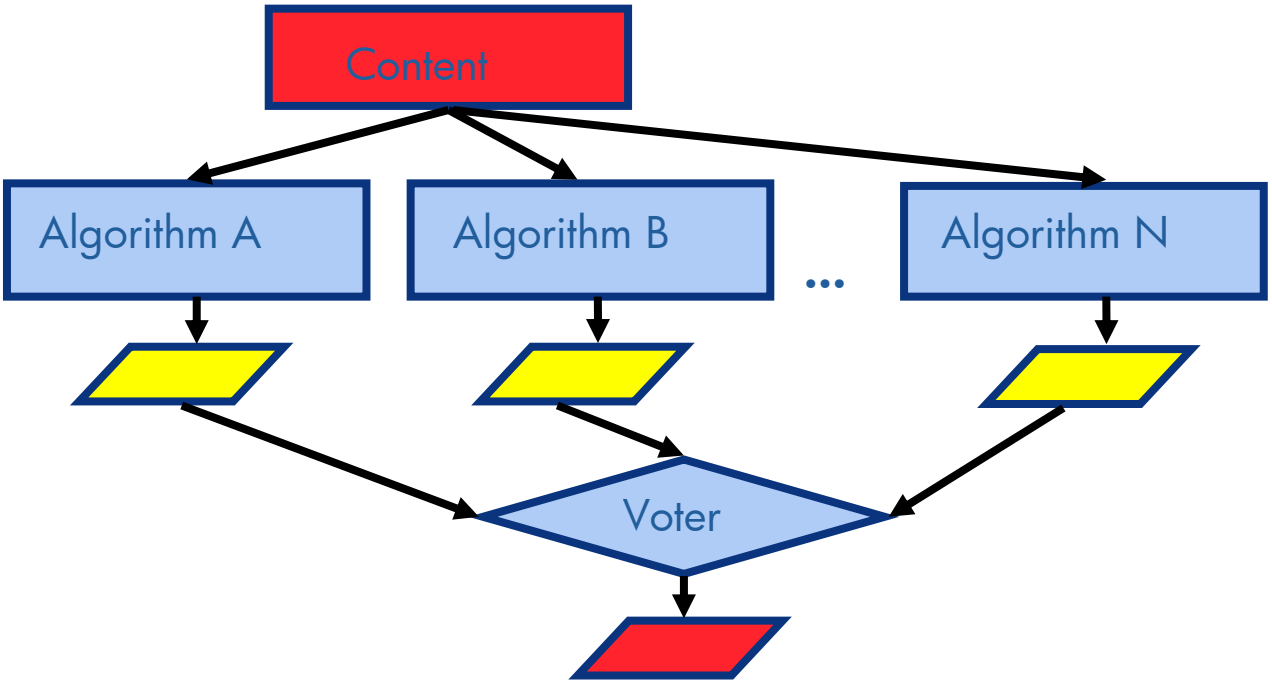

Figure 5. Voting meta-algorithm. Each of the algorithms in the set are run against the content. Both majority and plurality voting can be addressed, and voting can be weighted by the accuracy of each individual algorithm[11].

Figure 5 illustrates the voting pattern, perhaps the simplest form of meta-algorithmic pattern. Using a voting pattern, the output of two or more algorithms—which could be complex analytical engines such as ASR engines—is simply

combined, with each individual algorithm voting on the final output. Plurality or majority voting is readily accommodated, and binary (discrete) voting and weighted (continuous) voting are readily deployed. We have shown that, in general, an odd number of individual algorithms provides the best combination of accuracy and throughput[11].

4.2 Tessellation and Recombination

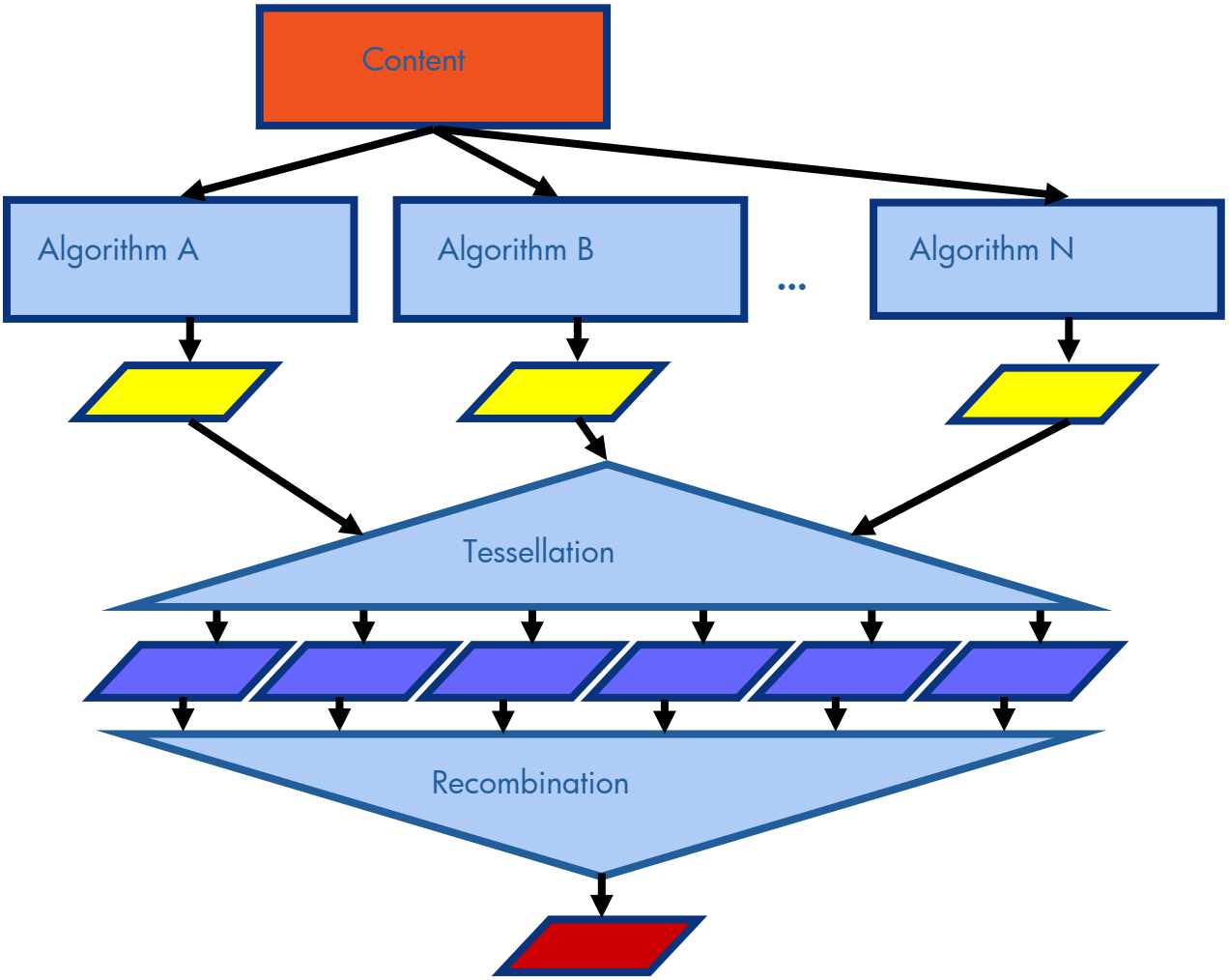

Figure 6. Tessellation and recombination pattern. Each algorithm is run in parallel against the content. The recombination step can involve alignment when algorithms product different output (deletion, insertion, substitution).

Figure 6 provides the design pattern for tessellation and recombination. As described in brief above (Section 3), tessellation is the process by which an analysis task—usually a classification stage—is divided into a set of primitive elements. When reduced to a tessellated set, the individual elements can be voted on or otherwise classified, then recombined into logical elements comprised of one or more tessellated elements.

This pattern can be explained by illustration, using document image segmentation and classification as an example. In Figure 7, left, there are 7 regions formed by segmentation. In the right image, there are 15 regions segmented. The regions formed on the left side image are not simply combinations of the more numerous regions on the right side image—instead, the tessellation produces a larger set (a total of 19 regions) than either of the two segmentation engines

produces by itself. The results of the tessellation are shown in Figure 8, in which these 18 regions and their classification are indicated. The regions classified as text are shown outlined in green, the regions classified as photos are shown outlined in yellow, and the regions classified as graphics—including business graphics, logos and non-font text—are shown outlined in magenta.

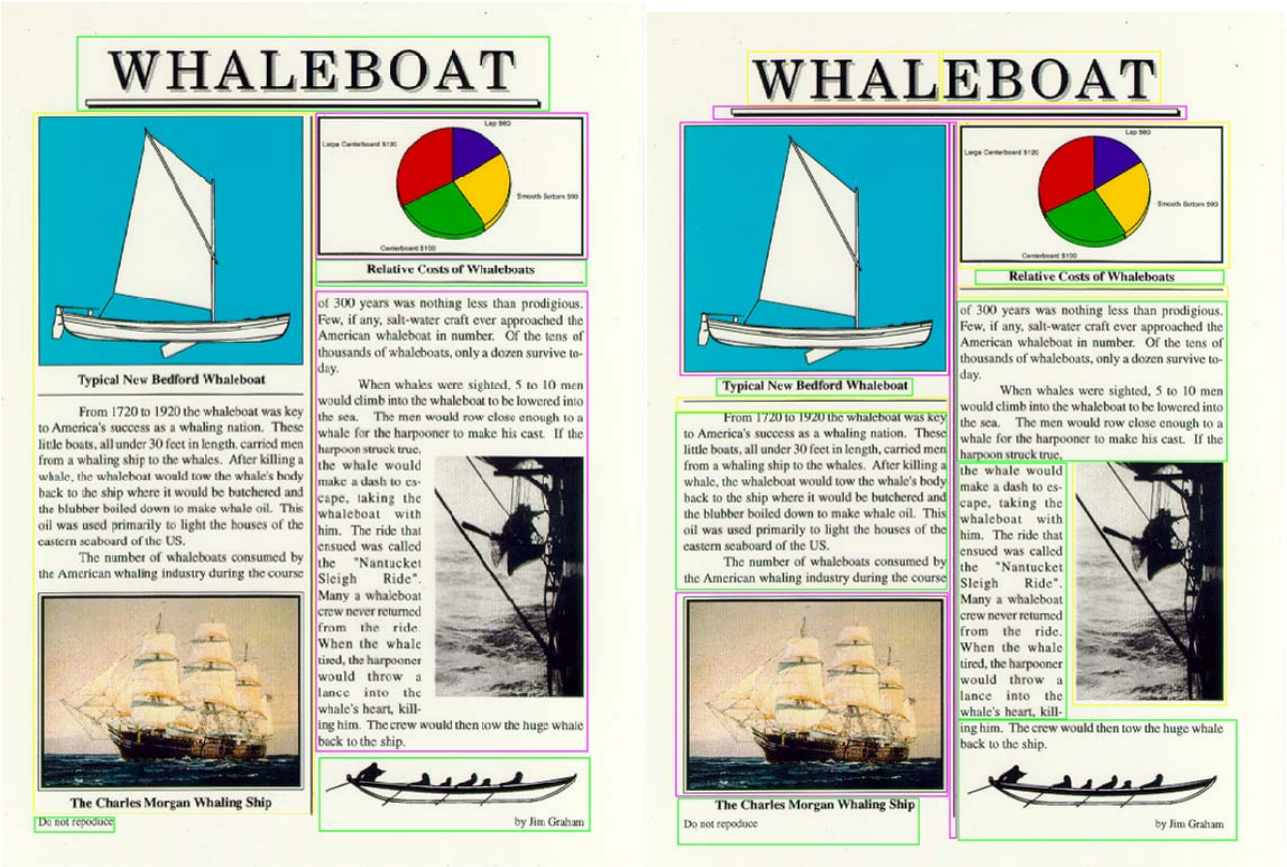

Figure 7. Output of two different segmentation/classification algorithms. Left: 80% of the region areas are mistyped, and only 7 of the expected 15 regions are segmented. Right: 32% of the region areas are mistyped, but there are 15 regions, as expected.

In Figure 8, the "recombination" process has not yet occurred. However, it is clear that the two sections of the "WHALEBOAT" title at the top can be recombined. Additionally, the three sections of text in the right column can be recombined into a single region, reducing the number of regions to 16. Importantly, however, the classification accuracy has gone from 20% (Figure 7, left) or 68% (Figure 7, right) to 100% accuracy in Figure 8. The analysis takes no longer than the slower of the two segmentation/classification engines, since they are processed in parallel.

4.3 Predictive Selection

The third type of meta-algorithmic pattern considered is the predictive selection pattern (Figure 9). With predictive selection, the decision on the final analysis is made by selecting the output of one of the plurality of algorithms. Predictive selection relies on training data, and in particular on the classification confusion matrix. Suppose two classification engines (E1 and E2) are to assign the output to one of N classes. For a particular instance, suppose E1 assigns the object to class A and E2 assigns the object to class B. Then, we simply consider the precision associated with assignment to class A for E1, denoted P(E1→A) and the precision associated with assignment to class B for E2, denoted P(E2→B). If P(E1→A) > P(E2→B), then we assign the object to class A, and if (E1→A) < P(E2→B), then we assign the object to class B.

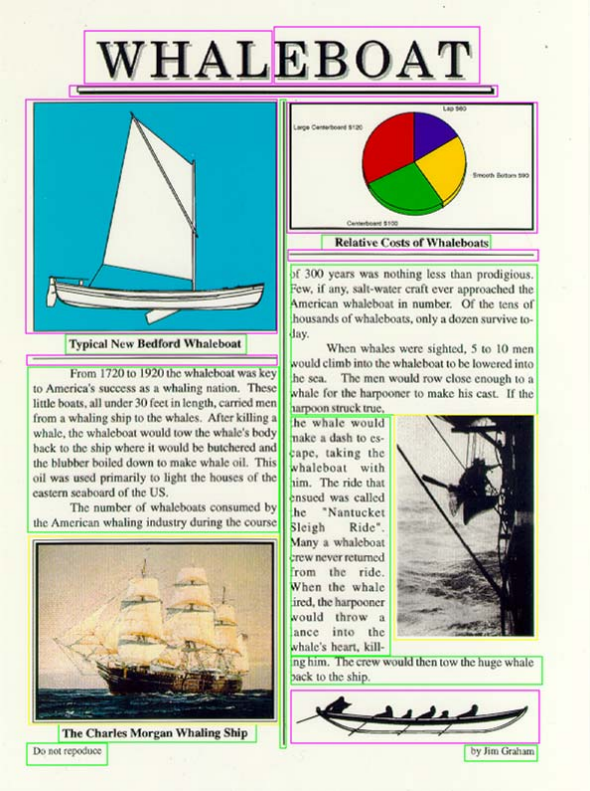

Figure 8. Meta-algorithmic combination of the two segmentation engines in Figure 7.

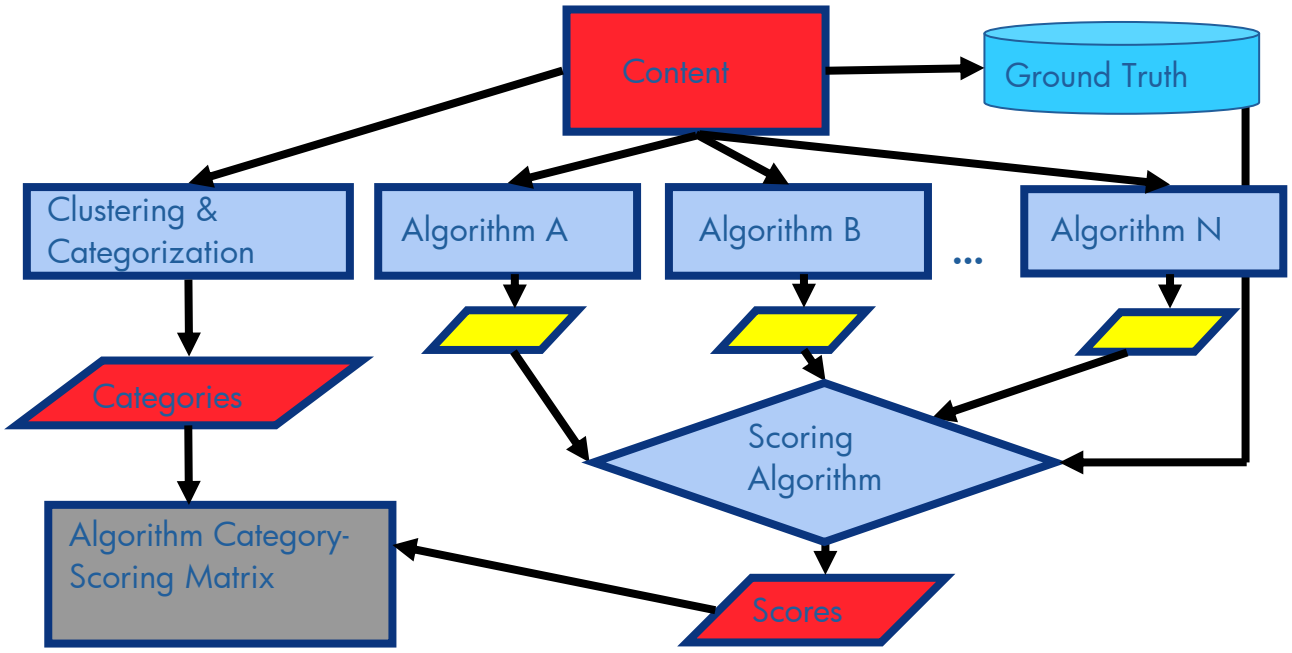

Figure 9. Predictive selection. Usually based on precision. Ground truthing must be sufficient to allow a later prediction on which algorithm to choose from a (usually) simple categorization.

Predictive selection can be used in place of voting in any situation in which reliable ground truthing has been created. We have shown the utility of predictive selection in several classification domains, including document classification[8]. Predictive selection is effectively a "pure" parallel processing pattern, inasmuch as all of the algorithms are run in parallel by definition during the run-time phase.

4.4 Constrained Substitute

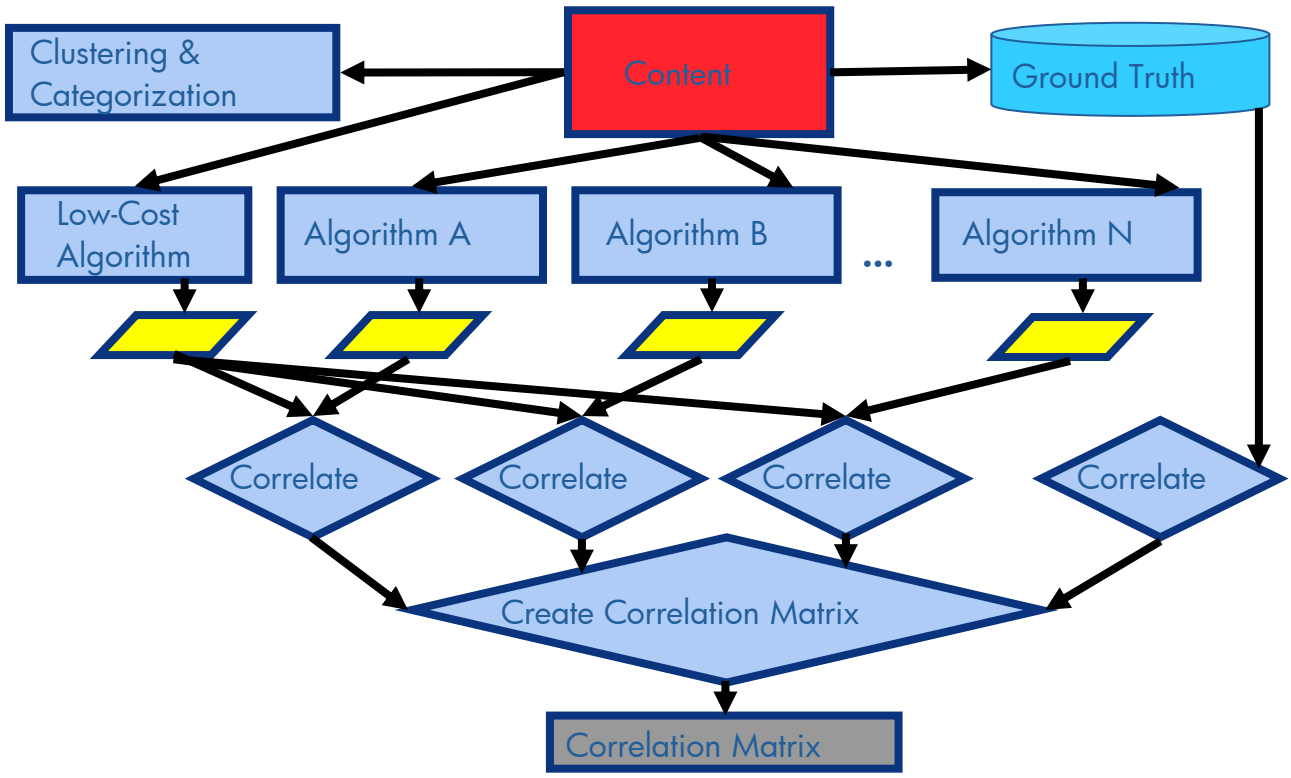

Figure 10. Constrained substitute, training (correlation) phase. Find out the higher cost algorithm with which the low cost algorithm correlates most highly.

The last meta-algorithmic pattern discussed in this paper is the Constrained Substitute pattern, shown in Figure 10 (training phase) and Figure 11 (run-time). The Constrained Substitute was created to allow a lightweight algorithm to perform a function similar to a heavyweight algorithm. The lightweight algorithm consumes less system resources (memory or execution cycles) or provides a simplified analysis for a different purpose. At runtime, if the system is not constrained, then the full algorithm is used. However, if one or more of the constraints hold true, then the output of the lightweight algorithm is used instead of any of the heavyweight algorithms—if and only if the output of the lightweight algorithm has historically proven to be correlated with any of the heavyweight algorithms. The lightweight algorithm then acts as a *substitute* for the original algorithm.

This meta-algorithmic pattern makes use of parallel processing primarily during the training phase. During run-time, the Constrained Substitute pattern effectively replaces the parallel processing of other meta-algorithmic patterns with the highest-throughput of the parallel paths. Thus, the Constrained Substitute pattern uses neither serial nor parallel processing during run-time.

4.5 Other Analyses Benefitting from Meta-Algorithmics

In addition to the examples shown in this paper, we have also successfully applied meta-algorithmics to part-of-speech tagging, OCR, document summarization, web and other database searches, journal splitting (defining all the fields in bibliographical entries), keyword generation, document clustering, and speech analysis (gender, accent, emotion).

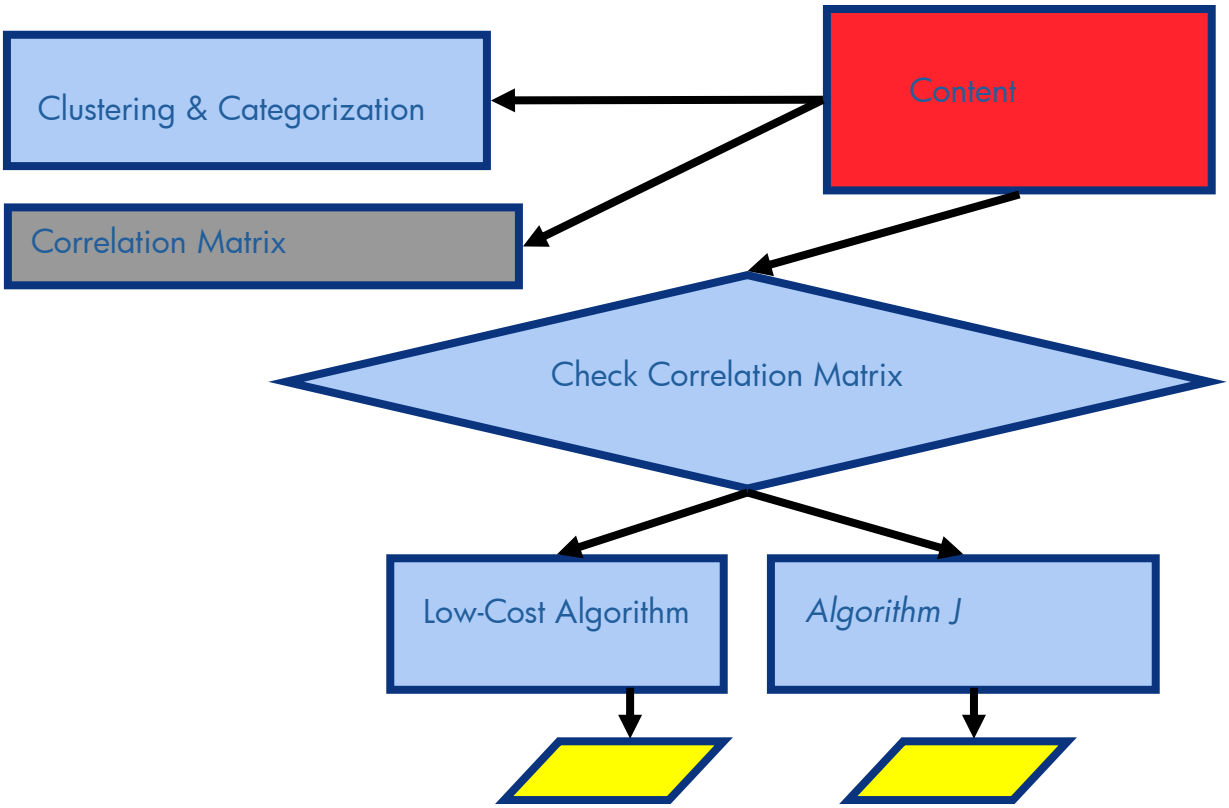

Figure 11. Constrained Substitute, run-time. If correlation is high enough, only the low-cost algorithm is performed. Generally, the "low-cost" is an advantage monetarily (licensing costs) and in performance (throughput)—and perhaps simplicity of integration—but at the expense of lower accuracy. This pattern is geared to uncover where its accuracy will match that of the "high cost" algorithms—e.g. its substitution is constrained to certain correlation (r^2) values.

5. DISCUSSION AND CONCLUSIONS

5.1 Security Printing

The security printing example (Figure 1) uses a low-resolution (down-sampled) version of the medallion image for segmentation. This segmentation took 10 msec using an HP ELiteBook 6930p laptop. Next, each of the analysis tasks were performed. These took from 30-400 msec in time (the sum of all 8 tasks was 1760 msec), resulting in a parallel processing time of 400 msec (time for the longest process), for a parallel processing time of 410 msec instead of 1770 msec for serially processing. This is a 76.8% reduction in processing time.

Parallel processing has particular benefits for improved throughput in the case of identifying non-authentic multi-component security images. Since each of the authentication tasks (e.g. eight separate pipelines for the medallion in Figure 1) is run in parallel, as soon as one fails, all the processes can be interrupted at once. This is an important point due to the broader implications for parallel design, in general. When a task is likely to fail on one or more threads, parallel processing can offer tremendous performance advantages—above even those exhibited for the "normal" use of parallel processing. Using the example above, suppose that the failure process takes as long as the approval process for each task. Then, parallel processing will show failure after the first failed process is completed. This will be in the range of 30-400 msec, meaning that the improvement in performance (or response time) is from 76.8% to 98.3% compared to the serial approval (1760 msec). This matters in a real-world application, such as a wide variety of mobile commerce[4] or mobile authentication applications, where any delay in the response can lead to frustration, confusion, and/or non-compliance by the consumer, retailer, distributor or other agent.

I used security printing in the initial discussion since it is a very broad field, involving as it does the printing of intentional information with the intent of reading (extracting and validating) at least some of this information at a later

point. The fact that security printing benefits from parallel processing, therefore, is an exemplar for the broader set of technologies accompanying the entire image recognition field.

5.2 Architectural Considerations

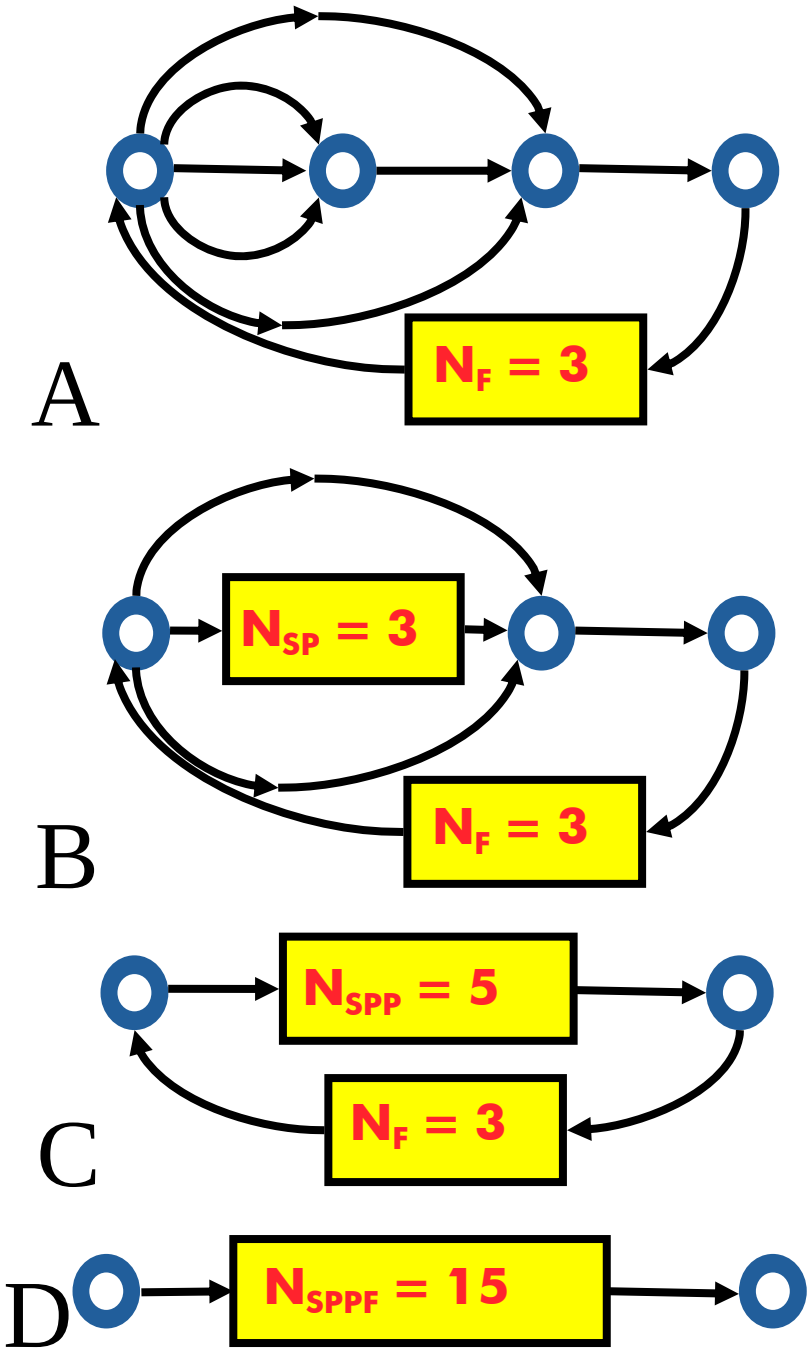

Figure 12. Simple system complexity number (N) computed from the feedback, series and parallel processing occurring within a system. In this example, the output is fed back three stages giving a feedback complexity number, N_F, of 3 (top diagram, A). Next the 3 parallel branches combined give N_P of 3, and combined with series, an N_{SP} of 3 (diagram B). Now the two additional parallel branches augment these to create a N_{SPP} of 5 (diagram C), which when multiplied by the feedback give N_{SPPF} of 15. This branching complexity can be compared among different system architectures.

Throughout the paper, I have tried to highlight how parallel processing can be used to perform the slower tasks—OCR, image segmentation, etc.—in parallel. Faster tasks, such as voting or meta-algorithmic pattern application, can be readily run in serial with little or no effect on performance. In fact, turning this around, it is possible to consider how to architect image recognition systems to optimize their performance. Figure 12 shows how a consideration of the system architecture complexity can be estimated based on how data flows in series, in parallel and through feedback loops in the architecture. The architecture in diagram A collapses to a complexity rating of 15, which can be compared against other architectures. In general, higher complexities will be more highly trainable and robust for a given task. However, this comes with a price (performance, complexity of system architecture, the need for more training data, etc.).

Beyond the scope of this paper, but certainly worth considering, is the argument that a suitable parallel processing architecture (serial-parallel, parallel-serial, parallel-series-parallel-feedback as in Figure 12, etc.), can be chosen for the image recognition application, based on the system needs. Among those needs are throughput (performance), and accuracy, robustness, trainability, access to high-quality training data, and system maintenance.

6. ACKNOWLEDGMENTS

Thanks to Giordano Berretta and SPIE for inviting me to write this paper. I am honored to have it included in the proceedings of the first SPIE Conference on Parallel Processing for Imaging Applications. I also wish to thank many incredible colleagues who have worked with me on one or more of the projects and/or approaches described herein. A non-exhaustive list of colleagues is Jason Aronoff, Igor Boyko, John Burns, Dalong Li, Xiaofan Lin, Margaret Sturgill, and Sherif Yacoub. I would be remiss not to mention that there is a much larger body of parallel processing research ongoing in the larger imaging community. This paper was not meant to be a review paper or an exhaustive overview; rather, it is meant to show from my own experiences the tremendous opportunities parallel processing offers for the overall imaging research field. Were I to include even a modest portion of that great body of work, I would not have been able to keep to length limit—and so this paper merely scratches the surface of the overall work in this field. Fortunately, however, a glance at the conference program has assured me that this larger body of work will be addressed by the excellent papers in the remainder of this Conference.

REFERENCES

[1] Wahl, F.M., Wong, K.Y., and Casey R.G., "Block Segmentation and Text Extraction in Mixed/image Documents," Computer Vision Graphics and Image Processing, vol. 2, 1982, pp. 375-390.

[2] Shi, J. and Malik, J., "Normalized Cuts and Image Segmentation," *IEEE Trans Pattern Analysis Machine Intelligence*, vol. 22, no. 8, 2000, pp. 888-905.

[3] International Standard ISO/IEC 16022, "Information Technology—Automatic Identification and Data Capture Techniques—Data Matrix Bar Code Symbology Specification," 2nd ed., 15 Sept. (2006).

[4] Mobile Commerce: Opportunities and Challenges, a GS1 Mobile Com White Paper, web site last accessed 11 November 2010, www.gs1.org/docs/mobile/GS1_Mobile_Com_Whitepaper.pdf.

[5] Simske, S., Sturgill, M., Everest, P., and Guillory, G., "IBFS: A System for Forensic Analysis of Large Image Sets," *WIFS 2009: 1st IEEE International Workshop on Information Forensics and Security*, 16-20, 2010.

[6] Simske, S.J., and Lin, X., "Creating Digital Libraries: Content Generation and Re-Mastering," DIAL 2004, 33-45.

[7] Berclaz, J., Bhatti, N., Simske, S., and Schettino, J., "Image-based mobile service: automatic text extraction and translation," Multimedia on Mobile Devices 2010 (EI 2010: IS&T/SPIE Electronic Imaging Symposium), Proceedings of the SPIE, Vol. 7542, 754204-754204-12, 2010.

[8] Simske, S.J., Wright, D.W., and Sturgill, M., "Meta-Algorithmic Systems for Document Classification," *ACM DocEng 2006*, 98-106, 2006.

[9] Vans, M., Schein, S., Staelin, C., Kisilev, P., Simske, S., Dagan, R., and Harush, S. "Automatic visual inspection and defect detection on Variable Data Prints," HP Labs Technical Report HPL-2008-163(R.1), http://www.hpl.hp.com/techreports/2008/HPL-2008-163R1.html?jumpid=reg_R1002_USEN, 42 pp., 2008.

[10] Simske, S. and Sturgill, M., "A Ground Truthing Engine for Proofsetting, Publishing, Re-purposing and Quality Assurance," *DocEng 2003, ACM Symposium on Document Engineering*, Grenoble France, Nov. 20-22, 2003.

[11] Lin, X., Yacoub, S., Burns, J., and Simske, S., "Performance Analysis of Pattern Classifier Combination by Plurality Voting," Pattern Recog. Letters, 24(12), 1959-1969, 2003.