IJDAR
DOI 10.1007/s10032-016-0260-8

ORIGINAL PAPER

Detecting figures and part labels in patents: competition-based development of graphics recognition algorithms

Christoph Riedl[1,5] · Richard Zanibbi[2] · Marti A. Hearst[3] · Siyu Zhu[4] · Michael Menietti[5] · Jason Crusan[6] · Ivan Metelsky[7] · Karim R. Lakhani[8]

Received: 22 January 2015 / Revised: 1 January 2016 / Accepted: 1 February 2016

Abstract Most United States Patent and Trademark Office (USPTO) patent documents contain drawing pages which describe inventions graphically. By convention and by rule, these drawings contain figures and parts that are annotated with numbered labels but not with text. As a result, readers must scan the document to find the description of a given part label. To make progress toward automatic creation of 'tool-tips' and hyperlinks from part labels to their associated descriptions, the USPTO hosted a month-long online competition in which participants developed algorithms to detect figures and diagram part labels. The challenge drew 232 teams of two, of which 70 teams (30 %) submitted solutions. An unusual feature was that each patent was represented by a 300-dpi page scan along with an HTML file containing patent text, allowing integration of text processing and graphics recognition in participant algorithms. The design and performance of the top-5 systems are presented along with a system developed after the competition, illustrating that the winning teams produced near state-of-the-art results under strict time and computation constraints. The first place system used the provided HTML text, obtaining a harmonic mean of recall and precision (F-measure) of 88.57 % for figure region detection, 78.81 % for figure regions with correctly recognized figure titles, and 70.98 % for part label detection and recognition. Data and source code for the top-5 systems are available through the online UCI Machine Learning Repository to support follow-on work by others in the document recognition community.

Keywords Graphics recognition · Text detection · Optical character recognition (OCR) · Competitions · Crowdsourcing

✉ Christoph Riedl
c.riedl@neu.edu

Richard Zanibbi
rlaz@cs.rit.edu

Marti A. Hearst
hearst@berkeley.edu

Siyu Zhu
sxz8564@rit.edu

Michael Menietti
mmenietti@fas.harvard.edu

Jason Crusan
jason.crusan@nasa.gov

Ivan Metelsky
imetelsky@topcoder.com

Karim R. Lakhani
klakhani@hbs.edu

1 D'Amore-McKim School of Business, and College of Computer and Information Science, Northeastern University, Boston, MA 02115, USA

2 Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA

3 School of Information, UC Berkeley, Berkeley, CA 94720, USA

4 Center for Imaging Science, Rochester Institute of Technology, Rochester, NY 14623, USA

5 Institute for Quantitative Social Science, Harvard University, Cambridge, MA 02138, USA

6 Advanced Exploration Systems Division, NASA, Washington, DC, USA

7 TopCoder Inc., Glastonbury, CT 06033, USA

8 Department of Technology and Operations Management, Harvard Business School, Boston, MA 02134 USA

1 Introduction

The United States Patent and Trademark Office (USPTO) is in the process of bringing an archive of eight million patents into the digital age by modernizing the representation of these documents in its information technology systems. In their daily work, patent examiners at the USPTO, as well as patent lawyers and inventors throughout the world, rely on this patent archive. Locating existing patents related to new patent application requires significant effort, which has motivated research into automatic retrieval of patents using both text [36] and images [3]. Most USPTO patent documents contain drawing pages which describe the invention graphically. By convention and by rule, these drawings contain figures and parts that are annotated with numbered labels but not with text, and so readers must scan the entire document to find the meaning of a given part label.

One would like to be able to automatically link part labels with their definitions in digital patent documents to save readers this effort. For example, one could create 'tool-tips' for part labels and figures, where hovering the pointer over a part label or figure brings up text describing the part or figure, reducing the need to switch back and forth between diagram and text pages. Unfortunately, robust solutions to this problem are currently unavailable. While document image analysis [38] and optical character recognition [12] have made significant advances, detecting figures and labels scattered within drawings remains a hard problem. More generally, text detection in documents and natural scenes [27,29,50,66] remains a challenging image processing task.

Prize-based competitions have a long history of encouraging innovation and attracting unconventional individuals who can overcome difficult challenges and successfully bridge knowledge domains. This has lead to an emergence of commercial platforms including TopCoder, InnoCentive, and Kaggle that have specialized in executing large-scale competitions around algorithm or software development. In September 2009, President Obama called on all US federal government agencies to increase their use of competitions to address difficult challenges. Following this, the US Congress granted all those agencies authority to conduct prize competitions to spur innovation in the America COMPETES Reauthorization Act of 2010 [25]. These developments helped provide a legal path for government agencies to conduct prize competitions. NASA, which already had prize authority and experience working with the TopCoder software competition community [1], opened a Center of Excellence for Collaborative Innovation to help other US federal agencies run challenges.

These developments together led to the USPTO launching a software challenge to develop image processing algorithms to recognize figure and part labels in patent documents on the TopCoder platform in December 2011 [53]. The goal of the competition was to detect figure locations and labels along with part labels in patent drawings, to enable their use in cross-referencing text and image data.

References to figures and part labels are common throughout a patent's text. Often, many specific references are combined in a single sentence. For illustration, we reproduce a sample sentence from a patent on a sifter apparatus. The quoted text appears on page ten while the referenced figure appears on page two of patent *US6431367* (emphasis in bold is ours).

> **FIG. 1** shows an outside appearance of the prior art sifter in which a plurality of sifter frames **101** is stacked on a sifting machine frame **102**, and is fixed unitarily to the sifting machine frame **102** by bolts **103** and nuts **104**.

Ideally, each of the part references highlighted in bold could be turned into a tool-tip that overlays the text, showing specific parts from Fig. 1 on page two. Tool-tips showing the description of parts from page ten when hovering over part labels on page two would also be useful. Both would facilitate patent examination by avoiding the need to scroll within patents.

The specific goal of the challenge was to extract the following from patent drawing pages: (1) figure locations and titles, and (2) part label locations and text. Each region type was represented by a rectangle (bounding box) with a text label (e.g., as a triple ((20, 20), (100, 100), '9b') representing part label '9b' located in a rectangle with top-left corner (20, 20) and bottom-right corner (100, 100)). Inputs and outputs for competition systems are provided in Table 1.

Participants were provided with images of patent drawing pages, each of which contains one or more figures (see Fig. 1). Each figure has a title and, in most cases, a large number of part numbers affixed to their parts with curved lines and arrows. Most part labels are numerical or alphanumerical. Complicating matters and many drawing pages also include additional numbers and text, such as page numbers, dates, patent numbers, or inventor names.

Each drawing page image is accompanied by the text of the associated patent in HTML format. These are useful because figures are described explicitly in most recent patents, and part labels must be referred to at least once in the text. Participants could use the HTML text to validate and modify character recognition output.

Figure 1 illustrates results from the first place system (leftmost two columns) and second place system (rightmost two columns). Both target ('ground truth') regions and regions detected by the algorithms are shown. The drawing page in the leftmost column contains one figure titled 'FIG. 3' which has 18 part labels. A number of challenges are illustrated in Fig. 1 including differing page orientations (portrait vs. land-

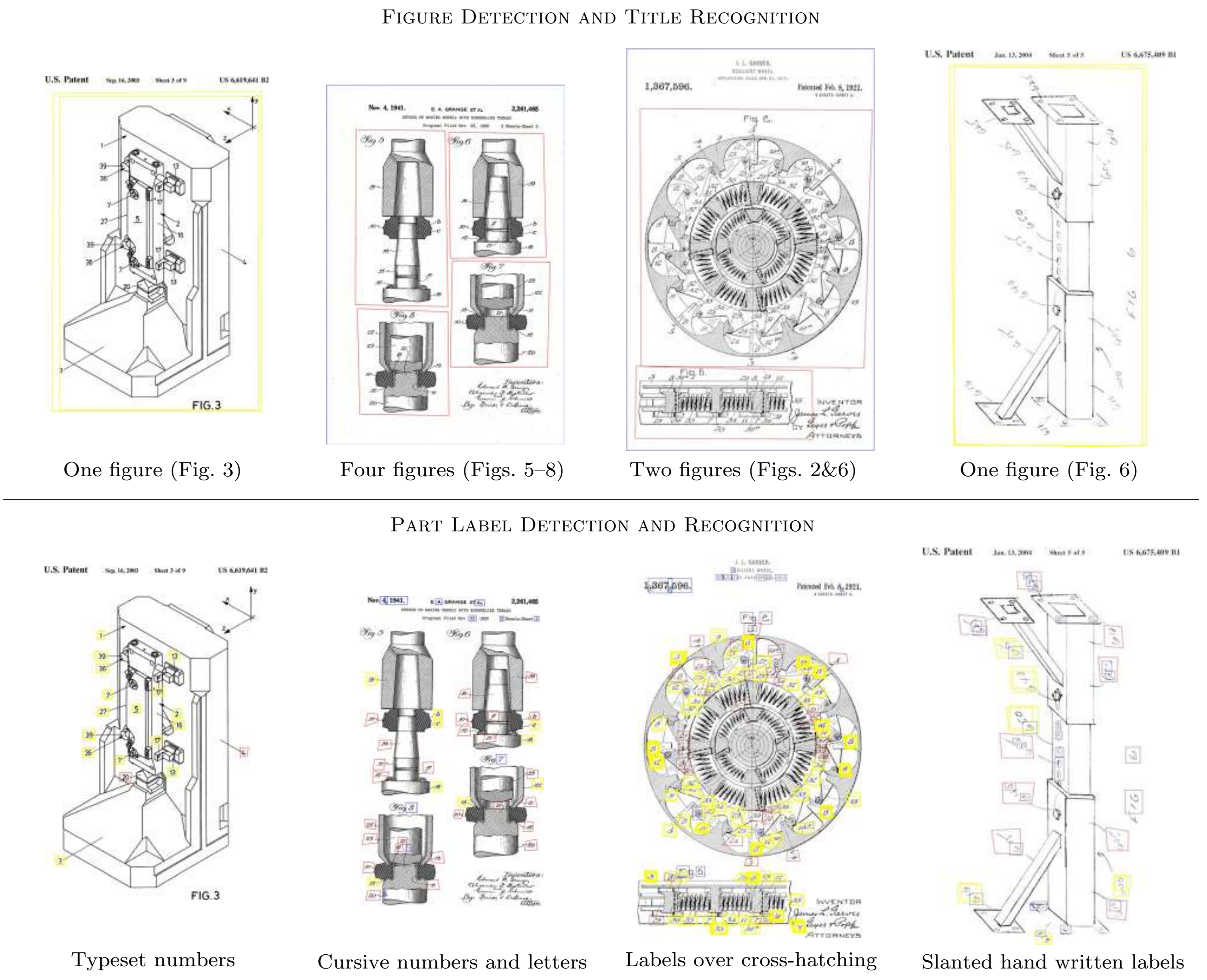
Figure Detection and Title Recognition
One figure (Fig. 3)
Four figures (Figs. 5–8)
Two figures (Figs. 2&6)
One figure (Fig. 6)
Part Label Detection and Recognition
Typeset numbers
Cursive numbers and letters
Labels over cross-hatching
Slanted hand written labels

Fig. 1 Sample results from the first place system (*leftmost two columns*) and second place system (*rightmost two columns*) on the figure detection and title recognition task (*top row*) and the part label detection and text recognition task (*bottom row*). Target regions are shown in *yellow* (located, or *true positive*), *blue* (*false positive*), and *red* (missed, or *false negative*). Figure titles are correctly recognized when they are included in the figure region box, and the text label for the figure region box contains the correct title. On the top row, inner columns show multi-target figures (*red*) which the algorithm mistakenly merges into a single figure (outer *blue* rectangles). Figure regions and titles are recognized correctly in the outer examples. The *bottom row* shows part label detection and recognition results. In the *leftmost* figure, all but one label is detected by the first place algorithm, but in the second column from *left*, it misses several labels because of the font used. The example in the third column has dense line art and hatching. The second place algorithm's character recognizer can better handle the cursive font used, but a number of labels are missed (*red*) including labels touching or on top of lines. In the rightmost example, the page is rotated (in landscape orientation). Here, six labels are detected correctly by the second place algorithm, but a number of false positives (*blue*) are caused by over-segmenting part labels. Additional false positives (*blue*) are produced for holes in the diagram (which have the same shape as 0/O) and the figure title (at *right*)

scape), multiple figures on a page, text that does not belong to figures or part labels, different fonts and font styles, handwritten text, rotated text, drawing elements that look like characters, and part labels that intersect lines in a drawing.

A sample of figure titles from the USPTO competition data are shown in Fig. 2. There is a large variety of fonts, font styles (bold, italic, underline), and formats (e.g., 'Fig. 2,' 'Fig 2,' 'FIG-2,' 'Figure 2'), in either portrait or landscape orientation. For the most part, part labels are typeset numeric ('11') or alphanumeric ('14b') strings in either portrait or landscape orientation. As illustrated in Fig. 1, there are also a number of drawing pages for which part labels are handwritten at an angle (i.e., slanted).

In this paper, we present the protocol and results of this competition. Section 2 describes related work. Section 3 describes the challenge in detail. In Sect. 4, we describe

Table 1 USPTO challenge overview

System input	
	1. 8-bit grayscale drawing page scan (300 dpi)
	2. Associated HTML patent text
System output	
	1. Figure bounding boxes and title text
	2. Part label bounding boxes and text

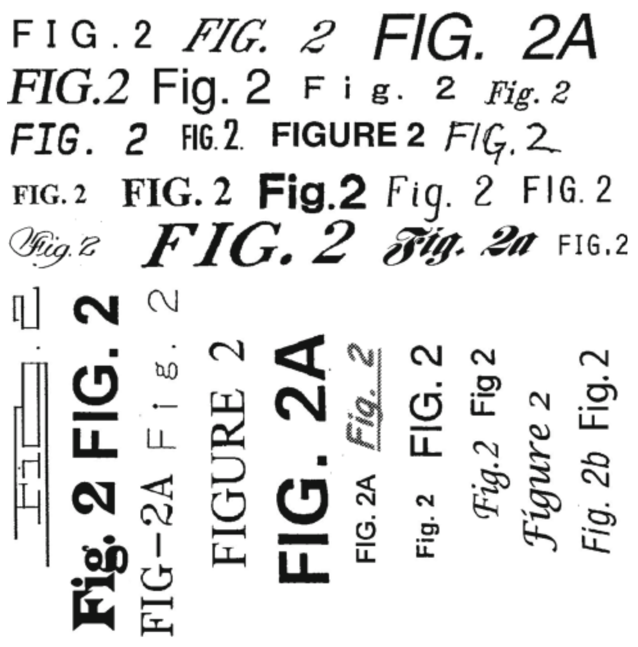

Fig. 2 Figure titles from the USPTO competition dataset

the approaches used in the top-5 algorithms submitted to the competition and a benchmark system we developed after the competition in Sect. 5. The top-5 ranked systems adopted similar strategies, but differed in their approaches to text/graphics separation, page orientation detection, region segmentation, character recognition (OCR), validation, and use of the provided HTML text. We analyze their performance in Sect. 6. We then summarize lessons learned in Sect. 7. In Sect. 8, we discuss the implications of this work and describe our competition design which may provide a template for other competitions aiming to solve document analysis problems using a global talent pool [37].

To encourage others to develop this work further, source code for the top-5 systems along with all labeled training and test data has been published under Apache License 2.0 in the UCI Machine Learning Repository.[1]

[1] http://archive.ics.uci.edu/ml/datasets/USPTO+Algorithm+Challenge\%2C+run+by+NASA-Harvard+Tournament+Lab+and+TopCoder++++Problem\%3A+Pat.

2 Related work

In this section, we present background on competition-based algorithm and software development in general, as well as competitions in document image analysis and information (patent) retrieval in particular. We then provide an overview of graphics recognition and work in text and engineering drawing dimension recognition. Recognizing engineering and architectural drawings is closely related to the USPTO challenge, in the sense that detecting objects in drawings is similar to figure detection, and recognizing dimensions is similar to recognizing part labels in patents.

Competition-based innovation Prize-based competitions have driven innovation throughout history [55]. For example, in the eighteenth century the British government announced a prize of £20,000 for finding a method to determine the longitude of a ship's location. More recently, prize-based competitions have been used to find solutions to hard algorithmic problems in biotech and medical imaging [31,41]. These competitions provide an alternative to approaches requiring an extensive search to identify and contract with potential solvers.

In recent years, prize-based contests have emerged as part of a major trend toward solving industrial R&D, engineering, software development, and scientific problems. In the popular press, such competitions are often referred to as 'crowdsourcing' [23]. In general, crowdsourcing has come to imply a strategy that relies on external, unaffiliated actors to solve a defined problem [31]. Competitions provide an opportunity to expose a problem to a diverse group of individuals with varied skills, experience, and perspectives [6]. Often, these individuals are intrinsically motivated, e.g., by the desire to learn or gain reputation within a community of peers. Competitions also allow rapid exploration of multiple solutions in parallel as multiple competitors attempt to solve a problem simultaneously [40,56].

Academic competitions Competitions at academic document analysis conferences are common. For example, the International Conference on Document Analysis and Recognition (ICDAR), the International Association for Pattern Recognition (IAPR) International Workshop on Graphics Recognition (GREC), and the International Workshop on Document Analysis Systems (DAS) have hosted numerous competitions on a variety of different document image analysis tasks over a period of decades.

More broadly, some of the best-known and most highly regarded academic competitions held within Computer Science are the Text REtrieval Conference (TREC) competitions, held for over two decades to develop and refine algorithms for text and multimedia search [54]. TREC competitions are numerous and focus on a broad variety of information retrieval tasks. In recent years, the TREC tasks span several domains including web search, knowledge base curation, temporal summarization, and information retrieval for medical and legal documents. Consequently, methods submitted to these competition span a wide variety of approaches used in information retrieval.

In the past, TREC has included a competition for text-based chemical patent retrieval, in which participants were given patent documents and asked to locate related patents in the test collection [22], and a task in 2011 involving recognition of chemical diagrams in images (the winning system used a bottom-up, rule-based strategy [49]). Similar competitions have also been held as part of the Cross-Language Evaluation Forum (CLEF), including competitions on recognition of chemical diagrams and flowcharts found in patent images [36,47]. Image-based patent search presents an opportunity for members of the document analysis community, as work is currently in an early stage [3].

Competitions on graphics recognition problems are held regularly, for both lower-level operations such as vectorization and text/graphics separation, recognition of text in diagrams (including rotated and slanted text such as found in the USPTO data; see Figs. 1, 2), and the interpretation of specific graphic types including technical drawings, tables, flowcharts, chemical diagrams, and mathematical notations [2,4,18,42]. These competitions normally consider the recognition of isolated graphics, whereas in the USPTO competition inputs are complete drawing pages, with associated headers, annotations, and text.

While tremendously valuable for discerning and advancing the state of the art, participants in academic competitions tend to belong to the community associated with a particular conference, prize amounts (if any) are small, and often a conference participation fee is required. For the USPTO competition described in this article, crowdsourcing with significant cash prizes for top-placing systems was used to solicit solutions from a global pool of talent reaching beyond the academic image processing community.

2.1 Graphics recognition

Graphics recognition concerns a family of structural pattern recognition problems in which the appearance and content of diagrams, notations (e.g., math, chemistry), plots, tables, figures, and other non-text document regions are recognized automatically. In the following, we summarize aspects of graphics recognition that pertain to the USPTO competition.

Language models Concise, well-fit language models provide beneficial constraints for hypothesis generation, validation, and selection [38,62]. As a simple example, recognizing arbitrary words is much more difficult then recognizing US postal codes, which are five-digit tokens for which the set of valid codes is known. Invalid postal codes created by recognition errors can be easily detected and replaced with similar valid codes. Similar word models are used by USPTO competition systems to identify and correct invalid figure and part labels, along with graphical/visual constraints such as expected positions and sizes for figure and part label text.

Generally speaking, as language model complexity increases, so does the amount of information that may be automatically inferred from similarity to legal hypotheses and/or context. However, detailed models can be hard to define and can also lead to fragility such as when a valid interpretation cannot be found or when a few errors lead to many others due to the propagation of constraints.

There is another, related trade-off in terms of hypothesis generation: to obtain high recall for recognition targets in the presence of noise, alternative interpretations (hypotheses) must be generated. However, additional hypotheses increase execution time and the likelihood of missing valid hypotheses and accepting invalid hypotheses. In the USPTO competition, examples of noise include touching characters in a figure title, or part labels intersecting lines in a drawing.

Architectural and engineering drawings Lu et al. [34] consulted an expert to design a sophisticated knowledge-based system for architectural drawing recognition. They observed that many *implicit* relationships need to be interpreted in architectural drawings, such as symmetry markings indicating inherited properties of objects, and extensive use of reference to indicate correspondences of objects within and between diagrams. An attributed context-free grammar language model is used to formalize the language of recognizable drawings and to coordinate recognition during a top-down parse of the input. The grammar is designed to allow the parser to recognize objects in decreasing order of reliability. As objects are recognized, context is utilized through propagating constraints arising from implicit relationships between recognized objects.

Syntactic pattern recognition techniques such as used by Lu et al. can be brittle, in the sense that inputs not resulting in a valid interpretation produce empty input. Returning partial parses and error-correcting parsing [9] can mitigate this problem, but not entirely solve it. Grammar for syntax-based methods is created by system designers presently, as grammatical inference remains a very difficult machine learning problem [14,19,20].

Recognizing annotations in engineering and architectural drawings, such as object dimensions [26], and identifying part descriptions in diagram legends [58] are closely related to finding part labels in the USPTO challenge. Part labels in patent drawings commonly appear at the end of lines pointing to the corresponding part (see Fig. 1), similar to the appearance of dimensions between or at the end of arrows in engineering drawings. For recognizing dimensions, language constraints are critical; for example, detected arrows are used to locate dimension text more reliably [13,30,60].

Table detection Patent documents often contain tables which can be easily confused with patent diagrams or drawings due to the presence of lines and the two-dimensional arrangement of content in cells. A variety of techniques have been used to detect the location and underlying grid structure of tables in document images including projection profile cutting, detection and analysis of lines (e.g., using the Hough transform), and whitespace gap intersections, along with histogram smoothing and mathematical morphology operations [63]. There is also ongoing work in table structure recognition and table content interpretation and compilation [15,39], but these tasks are not considered in the USPTO competition.

2.2 Text/graphics separation and OCR

For the USPTO competition, participating systems needed to separate graphics from text in page images, in order to locate figure regions and recognize figure titles and part labels using OCR. We summarize work related to these tasks below.

Text/graphics separation A common early processing task is text/graphics separation, in which regions containing text and other page contents are separated into two or more layers. Most text/graphic separators filter large connected components in the early stages, along with long/thin and very small connected components. This tends to filter out characters that are small (e.g., '.', ',') or that touch characters or graphics; attempts are made to recover these lost characters using context during word and text line detection, as described below.

Image features for detecting text have included connected component shape, aspect ratio, density, and spacing [35,65], similar features for skeletonized connected components [13], and textural features that exploit the relatively high visual frequency of text in comparison to graphics (e.g., using Gabor filters [65] or Gaussian second derivatives [61]).

A key issue for text/graphics separation is handling different font sizes. This is dealt with by measuring features at different scales [11,61,67]. Recently, image patches have been used instead of connected components, along with feature learning methods such as k-SVD and sparse representation [11] and convolutional k-means [67] to construct discriminative image patch feature spaces.

Character segmentation Casey and Lecolinet [8] identify three main aspects that character segmentation techniques incorporate to different degrees: (1) dissection (cutting using image features), (2) recognition-based methods (incorporating OCR output as features), and (3) holistic recognition (classifying complete words rather than individual characters). Many sophisticated segmentation methods are recognition-based, with final segmentation maximizing a criterion based on probabilities or costs associated with recognized characters. A common example is using hidden Markov models (HMM) to segment words and characters by maximizing the joint probability of the inferred characters and words.

Optical character recognition (OCR) A very wide array of techniques have been used for OCR, and for space, we provide only a (very) brief summary. An important OCR benchmark was the University of Nevada at Las Vegas competitions, held annually during the early 1990s [44,45]. Since that time, text OCR has become a mature technology, and text recognition research has shifted toward the harder problems of recognizing text in natural scenes, videos, and documents captured by camera [16,24,33,62]. Over the last decade, there have been a number of robust reading competitions on these topics, held as part of the International Conference on Document Analysis and Recognition [27,28,50].

Word and text line segmentation Words and text lines are commonly detected through clustering detected characters [43]. Distances between characters, words or text lines are estimated using the distance, relative orientation, and similarity of connected components. Morphological operations have been used to merge clusters and shrink/tighten boundaries of connected components during clustering [35].

To detect rotated text such as found in engineering drawings and maps, Fletcher and Kasturi [17] make use of the Hough transform to determine text line orientations from connected components. Tombre et al. [57] extend their approach using median and linear regression to produce additional local text orientation estimates when clustering detected characters into strings. To focus search, estimated word or text line end points may be used as the initial cluster centers [46,57]; Roy et al. [46] use the shape of spaces between characters while expanding the ends of text lines, and are able to extract curved text, such as found in document seals.

Bukhari et al. [7] have provided a recent survey of current methods for detecting curved and warped text lines. One strong approach estimates baseline and x-line (the 'middle' line that sits on top of a lower case 'x') locations for individual characters and then places active contours (snakes) at the top and bottom of connected components which are deformed based on an energy model, after which overlapping snakes are combined.

3 The USPTO challenge

This section describes the design of the USPTO algorithm competition, including the recognition tasks, reference image data, ground truth data creation, evaluation and scoring methods, and the competition outcome.

Input and output Table 1 summarizes the inputs and outputs for the competition. For input, systems receive a 300-dpi grayscale patent document image (patents are, by requirement, grayscale) and an HTML file containing the text of the patent. The HTML file does not contain the patent's title page, which has filing meta information such as patent number, filing dates, and a short abstract. Using combined visual and textual information for graphics recognition is infrequent in the literature and an unusual characteristic of the competition.

For output, systems need to identify figure locations and titles, along with part label locations and part label text. Figure and part locations are represented by bounding boxes. Participant systems needed to implement two separate functions to produce these outputs: the first for figure data and the second for part label data.

System constraints For system tests and final scoring, the competition imposed a time limit of 1 min per test case and a memory limit of 1024 MB. There was no explicit code size limit, but a limit of around 1 MB was advised. Furthermore, the binary executable size was limited to 1 MB, and the compilation time limit was 30 s. These are the standard time limits used for competitions on TopCoder. These default values seemed appropriate, so we decided to keep with TopCoder conventions and work within bounds that competitors were familiar with. The programming languages supported by TopCoder and allowable for the competition were Java, C++, C#, or Visual Basic .Net.

Data and tools Three different datasets were created for the competition (see Sect. 3.2 for details). A training dataset was available for download to all participants, which they could use to design and test code on their own machines. During the contest, any system submission by a participant to the contest site would generate a score visible only to the team using a second private dataset (the system test set). To prevent over-fitting of submissions to the training or system test datasets, the last submission of each team was re-scored using a third, private evaluation dataset at the end of the contest.

We also provided an offline tester/visualizer, including Java source code, which allowed participants to visualize their results and check the precise implementation of the scoring calculation. This tool was used to produce the images shown in Fig. 1.

Team composition and ranking Consistent with usual practices in programming contests, participants were able to make repeated code submissions to enable testing of solutions and gather feedback about solution quality. Participants were organized in teams of two, and both members of a team were able to submit program solutions.[2] Submissions were compiled and executed on competition servers, where solutions were tested against a private test case image set to allow objective scoring. The overall team score was given as the maximum score of both team members.

System scores on the final test set were used to rank systems and award prizes. System scoring is described in Sect. 3.3. The execution and prizes for the competition are described next.

3.1 Running the competition

TopCoder The contest was run on the TopCoder.com online programming competition Web site, a commercial platform established in 2001 [1]. Working with TopCoder provides convenient access to a standing community of over 800,000 software developers who regularly participate in crowdsourcing competitions and provides infrastructure for online test and scoring of solutions. (TopCoder also had a working relationship with NASA's Center for Excellence as described above, which allowed the USPTO to pay the cash prizes.) Apart from developing conventional software solutions, competitors on this crowdsourcing platform also regularly compete in contests to solve abstract algorithmic problems that require a mix of logic, mathematical, and programming skills. Since its inception a decade ago, the platform has awarded over $70 million in cash prizes.

Schedule and prizes Given the complexity of the task to be solved, the competition ran for 4 weeks between the end of 2011 and beginning of 2012 (many TopCoder competitions run only for 10 days). To attract participants, we offered a combined prize pool of $50,000 which was split into two overall prizes and 22 smaller prizes for virtual competition rooms. We offered two highly attractive overall prizes of $10,000 and $5000 for the first- and second-placed teams. However, offering large but few overall prizes may not result in the best outcome due to an effort-reducing effect of greater rivalry [5,21]: if everyone competes against everyone else, an individual team's likelihood of winning may be too low to warrant the investment of substantial effort. Therefore, we organized the competition into 22 virtual rooms, each of which offered an additional $1000 and $250 'room prize' for the room winner and runner-up. Furthermore, all active participants also received a limited edition T-shirt to acknowledge their efforts in participation which was paid for by the Harvard-NASA Tournament Lab.

[2] Embedded within this competition was a social science experiment to investigate different team formation mechanisms. Two treatments were implemented. In treatment one, teams were formed through bilateral agreement between participants after communicating through a public forum or private direct messaging (this was termed the 'free-form' treatment). In the second treatment, teams were formed based on a stable-matching algorithm using participants' stated preferences (termed 'algorithm' treatment). We found no significant differences in

Footnote 2 continued
algorithm performance between the two treatments. The exact details of the social science experiment are beyond the scope of this paper. Some preliminary results can be found in this working paper http://goo.gl/NjoWce.

3.2 Patent page image and text datasets

Data selection For the purposes of this online competition, we prepared a representative corpus of 306 patent drawing pages from various different patent classes. For some patents we included one drawing page in the set, and for other patents, we included multiple drawing pages. The whole corpus was divided into three subsets A (train), B (system test), and C (final test) containing 178, 35, and 93 drawing pages, respectively. The division was made randomly but with one restriction: all drawing pages belonging to the same patent were always placed into the same subset. We chose this approach to test generalization toward the full patent archive which contains many patents with more than one drawing page. In addition to image data, participants also had access to the patent text in HTML format which was provided by USPTO. As described earlier, title pages were omitted, which do not contain the main content of the patent such as references to figures and part descriptions.

Ground truth creation To create the ground truth reference standard, we used the image annotation tool LabelMe [48].[3] We used a private instance of LabelMe rather than the open crowdsourcing platform, to prevent leakage of the final scoring images. Two contractors were paid by the USPTO to manually identify and label the ground truth figure and part label regions. The organizers then visually inspected the two contractors' work and used the more accurate ground truth for the competition.

File formats Page images were stored in JPEG format, and patent texts in HTML format. Figure and part label regions and text are represented separately using text files called *answer* files. Answer files begin with the number of detected regions (figures or part labels) on the first line of the file. Each remaining line defines a region, by a polygon represented using a list of vertex coordinates followed by the associated text. Ground truth annotators used polygons with various numbers of vertices, as supported by LabelMe. All top-5 systems represented each region using the four points of a bounding box, as bounding boxes were used to evaluate region detection (see below).

3.3 Evaluation metrics and scoring

Region matching criteria The axis-aligned bounding box for a candidate region B_C matches the axis-aligned ground truth bounding box for a region B_G when the intersection of the two boxes is as large as some percentage α of the larger of the two boxes:

$$\text{area}(B_C \cap B_G) \geq \alpha \max(\text{area}(B_C), \text{area}(B_G)) \tag{1}$$

[3] http://labelme.csail.mit.edu/Release3.0/.

where $\alpha_f = 0.8$ for figures, and $\alpha_p = 0.3$ for part labels. Different α values are used for figures and part labels because of the much smaller size of the part labels.

Text matching criteria Figure titles and part labels are normalized before comparison, as shown below.

- *Figure titles* Figure titles (identifiers) are provided without the 'Fig.', 'Figure' etc. indication, e.g., '1a' is the correct title for 'Figure 1a.' All letters are converted to lower case; characters other than a–z, 0–9, (,), -, ', <, >, . (period), and / are removed. An output string must match the normalized ground truth string for a figure title to be considered correct.
- *Part labels* The same characters are preserved as for figure identifiers. However, there are some special cases, for example where two part labels may be indicated together, e.g., '102 (103)' or '102, 103' indicating parts 102 and 103. Periods/dots must be removed from the end of part labels. Subscripts are indicated using <and> (e.g., A <7> for A_7); superscripts are represented in-line (e.g., 123^b is represented by 123b).

Scoring test files Each input file receives two scores: one for part labels, and one for figures. Files for which no output is produced are scored 0 points, whether due to (1) exceeding the 1-min time limit, (2) exceeding the 1024 MB memory limit, (3) a system crash, or (4) improperly formatted output.

For a figure or part label to be correct, both the region and label must match. Partial credit is given for matching a region correctly but mislabeling it. The match score $match_s$ for partial matches was 0.25 and full matches was 1.0. To compute the accuracy, $match_s$ scores are added and used in weighted recall for ground truth regions (R), precision of output regions (P), and their harmonic mean (F, the F-measure):

$$R = \frac{\sum \text{match}_s}{|\text{Target regions}|} \quad P = \frac{\sum \text{match}_s}{|\text{Output regions}|} \tag{2}$$

$$F = \frac{2RP}{R+P} \tag{3}$$

For a test file, given system F-measure accuracy F and runtime in seconds $T \leq 60$ s, the figure or part label detection score is given by:

$$\text{score} = F \times \left(0.9 + 0.1\left(\frac{1}{\max(T, 1)}\right)^{0.75}\right) \times 10^6 \tag{4}$$

Execution time determines 10 % of the final score: to give a sense of the effect of speed on the score, at or under 1 s incurs no penalty, at 2 s roughly a 4 % penalty, at 5 s 7 %, and at 25 s 9.9 %. This is noteworthy, because including execution time directly in a scoring metric is uncommon in the document image analysis literature. We perform additional analyses to determine the effect of the specific relative weighting on determining contest winners in Sect. 6.

System scoring The final system score was defined by the sum of all figure and part label test file scores. There are many more part labels than figure regions, as most figures contain multiple parts. Using the sum of figure and part label scores insures that for each file the figure and part label results are weighted equally in the final system score.

4 Participant solutions

In this section, we analyze the five systems with the strongest results submitted for the USPTO competition (the 'top-5'). We provide a brief overview of each system, followed by a discussion of their similarities, and then by a discussion of their differences. To support our comparisons between systems, we use Table 2 to summarize early processing and character recognition, and Table 3 to summarize additional steps for locating figure titles, figure regions (which *include* the location of the figure title), and part labels. These tables were created after carefully studying the source code and system descriptions provided by each team.

4.1 Top-5 USPTO system summaries

Below, we briefly summarize the top-5 systems in the competition. We also provide the programming languages used to implement each system (from first to fifth place). Sample results from the top-2 systems are shown in Fig. 1.

1. *JacoCronje* (*JC*, impl. C++). The winning system is the only one to use the provided HTML text to validate detected figure titles and part labels, and uses OCR results for detected page numbers and headers on drawing pages to help locate likely titles and labels in the HTML text.
2. *Protocolon* (*PC*, impl. C++/OpenCV). It uses an MLP character classifier trained using synthetic as well as provided character images to handle the variety of fonts (see Fig. 2). Classification is iterative, re-estimating font parameters while searching for figure titles and part labels, and character candidates are fit into a box containing an ascender and descender region above and below the writing line.
3. *Wleite* (*WL*, impl. Java). It uses varying-width templates for character recognition, defined by the average training sample width for each class. It initially locates figure titles using the shape of 'Fig' and 'Figure' rather than character strings, and segments figure regions using pixel projection profile cutting rather than agglomerative clustering.
4. *GoldenSection* (*GS*, impl. C#/R(MLP training)). It uses two MLP classifiers for character recognition, one for frequent characters, and the other for a larger vocabulary (the maximum-confidence result is used). This system employed the simplest strategy, using no validation of detected figure titles or part labels, and instead applying OCR in three different directions to try and capture portrait and left/right-rotated landscape page orientations.
5. *tangzx* (*TZ*, impl. C++). This system distinguishes connected components for label text from figure text and graphics, instead of just text and graphics. The simplest character features are used in this system (15×15 binary grids). Only this system tries to capture missing dots on 'i' characters prior to recognizing part labels.

4.2 System similarities

Generally speaking, systems are more similar in the early stages of processing than character recognition (see Table 2) or later processing when specific targets are sought after (see Table 3). We summarize the main ways in which the top-5 systems are similar below.

Processing pipeline All top-5 systems use a data-driven, bottom-up pipeline recognition architecture. Connected components of a given minimum and maximum size are treated as character candidates and then used to filter non-figure text (page headers and tables), detect page orientation, and produce word candidates through distance-based clustering, with some maximum distance used to merge CCs into clusters. After this, pattern matching is applied to OCR results for words to locate figure titles and part labels, with most systems employing some validation step that filters and/or corrects detected titles or labels.

Figure regions are identified by clustering CCs with the nearest detected title and then including the title in the final region, with the exception of the third place system (WL; see below), which uses projection profile cutting to locate regions containing a single detected figure label. Other deviations and difference for specific systems are provided in Tables 2 and 3.

Text/graphics separation (Table 2) As seen in Fig. 1, many drawing elements will form connected components that are large in comparison with characters. Very small connected components are likely noise (e.g., specks of dirt on the patent document), although small character components such as the dot in an 'i' can also be small. Text/graphics separation is performed using thresholds to define minimum and maximum sizes for CC character candidates (see Table 2). System TZ (fifth place) is unique in that it defines three rather than two size ranges: regular characters, characters in figure titles, and graphics. This is to take advantage of the fact that figure text is normally larger than graphic CCs, but larger than part labels.

Filtering non-figure contents (Table 2) Figure headers are located near the top of the page in either landscape or portrait orientation and are not considered part of the figure itself.

Table 2 Early processing and character recognition for top-5 systems

	System place and initials				
	1st (JC)	2nd (PC)	3rd (WL)	4th (GS)	5th (TZ)
Text/graphics separation and word detection					
CC classes	(2) TextGraph	(2) Text Graph	(2) Text Graph	(2) Text Graph	(3) Text Graph FText
CC size filter	✓	✓	✓	✓	✓
Word det. after OCR		✓		✓	
CC clusters	Ver. overlap &hor. distance	(2 dirs.) Height ratio &hor. distance; baseline deviation	Ver. overlap &hor. distance	Ver. overlap &hor. distance; remove short chars i, l, 1, or not in 0-9a-zA-Z	Ver. overlap &hor. distance: Text &FText (fig. text) CCs clustered separately
Page orientation and filtering non-figure contents					
Orientation	Text CCs vote by wider/taller &max hor. versus vert. overlap with nearest Text CC	OCR and seg. words (both dirs); max mean word width versus height	Text CCs vote by wider/taller	Consider OCR'd Text CCs in 3 directions: *portrait, landscape-left, landscape-right* (highest conf. results used)	Text CCs vote by wider/taller
Filter header	✓	✓	✓		✓
Table filter	If ruling lines in 'word' perimeter &intersecting 'words'	if ≥ 2 rows &cols, via hor/ver projection &line detection,	If ruling lines in perimeter and not near an incoming line		If lines found at two of four 'word' BB sides
Character recognition					
Character	(14)	(36)	(22)	(31) *Classifier a:*	(56)
Classes	0-9fgac[a]	()0-9a-hj-np-z[a]	0-9a-dA-F[b]	0-9a-giruA-GIRU!	()0-9a-lnr-uxy
				(69) *Classifier b:*	A-TV-Z
				()0-9a-zA-Z-/.,!	
Features	16 × 32 ternary (b/w, hole)	16 × 16 gray values (mean subtracted); De-slanting; empty areas added for missing ascender/descender regions	W × 40 gray values (avg. intensity);W is avg. width for each char. class	6 × 6 density (% black), width, height, aspect ratio, total density	15 × 15 binary
Classifier	*Template* (pixel match %)	*MLP* 256:50:36	*Template* (ssd pixel gray values) resizes input region to match each char. template width	*2 MLPs* max of:	*MLP* 225:225:56
		Iter. refinement		40:30:31 MLP *(frequent chars)*	
		Re-estimate ascent/descent regions, spacing, italic angle		40:25:69 MLP	

[a] Lower case symbol only: no symbol 'i'
[b] No symbol 'g' or 'G'; Symbols '0' and '1' have two subclasses

Table 3 Figures and part labels: top-5 system text recognition, region segmentation and validation techniques

	System place and initials				
	1st (JC)	2nd (PC)	3rd (WL)	4th (GS)	5th (TZ)
Figure detection & title recognition					
T. syntax	Contains ‘f1g’ (fig)	Patterns for ‘fig’ (‘f1g’, ‘fzg’, etc.)	**(Word shape)** FIG + 1–3 digits, FIGURE + 1–2 digits	Starts with ‘fig’	3–9 chars; contains ‘fig’/‘FIG’ legal letters: a–d, A–D
T. detection	Match words containing ‘f1g’	Match ‘fig’ patterns in words; estimate font attributes, repeat OCR &look for ‘fig’ patterns; match words with first three letters with similar ascent, descent, w/h as detected titles; re-estimate spacing for large fonts; repeat OCR on candidates	Match word shape models; find best candidate by location in figure and char confidence, discard candidates highly dissimilar from ‘best’ candidate	Match words starting with ‘fig’	OCR *FText* CC cluster candidates, reject illegal titles
Fig. regions	Merge words with nearest fig. title; random search to min. penalty to obtain one title per figure	Merge words starting with figure titles; iteratively add CCs, updating BB, allowable growing directions. Obtains k regions for k labels (k = 1 if no fig labels detected)	*(Top-down)* Cut page at horizontal or vertical gap until each figure contains one fig. title	Merge CCs by proximity (omitting fig labels); obtain k figure regions for k figure titles; assign titles to regions by minimum total distance	Merge non-graphics CCs with closest graphics CC. Detect figure titles, then assign merged CCs to closest figure title
Validation	Sort extracted figure titles numerically. Match best sequence of figure titles in HTML text using detected page number and header		Reject titles with low character confidence or unexpected position (i.e., not near boundary of figure region)		Penalize matching score by not counting unexpected characters in titles
Part label detection and recognition					
L. syntax	Max 4 chrs, at least 1 digit, no more than 1 alphabetic; reject legal figure labels	**Max 9 chrs**, up to 2 alphabetic; penalize labels starting with alpha. char; cannot contain ‘mm’; reject ‘0’ ‘l’ ‘1’ and isolated alphabetic characters	Max 4 characters, up to 2 alphabetic	Max 4 characters, containing digits, digits followed by letters or letters followed by digits.	Max 4 characters, cannot be an isolated 0 (zero) or 1 (one)
L. detection	Filter large/small words; reject low-confidence OCR, illegal labels	Reject illegal; score candidates by char confidences &unlikely label penalties (e.g., ‘a1’); re-estimate font parameters on most confident candidates; reject characters far from mean-confidence, OCR-modified labels	Reject figure labels; most frequent text height for part labels is estimated as most frequent in Text CCs; reject low-confidence labels not near a connecting line	Reject illegal labels	Find small *Text* CCs (e.g., ‘.’), merge them with adjacent Text CCs; OCR and reject illegal labels; reject if overlapped by fig. title
Validation	For low character confidence labels, assign most similar word found in HTML text to detected label allowing common confusions (e.g., ‘5’ and ‘s’), reject label if more than 1/2 chars replaced	Separate labels joined by parentheses (e.g., ‘103(104)’)	Single digits must have high confidence		

Hence, they need to be removed. They are removed using one—or a combination—of the following simple methods: (a) considering a margin around the entire page image (JC); (b) removing detected text located near the top (WL) and/or left edge of the page (PC); and (c) taking the length of the detected text lines as a cue regarding the presence of a header (TZ).

All systems but GS use simple table detection and filtering, in order to remove words belonging to table regions (which are not considered drawings in USPTO patent documents and would hence not contain part labels). Line detection is done using simple connected component analysis (looking for long/narrow CCs). Only PC (second place) makes use of line projections to try and detect the presence of at least two or more columns and rows to identify a table location. The remaining methods consider whether ruling lines surround a word. The WL system (third place) considers whether the end of a line appears near the word, in which case the word is assumed to be a candidate part label.

Training Aside from the character recognizers, many parameters in these systems are set based on assumptions, or 'empirically' by trying different parameter values and selecting those that perform best. While potentially sub-optimal, this is necessary due to the short duration of the competition.

4.3 System differences

Tables 2 and 3 show a number of differences between the systems, and include deviations from the strategy above. Here we will discuss the most significant differences.

Word detection and page orientation (Table 2) In the USPTO systems, word orientations are often used to detect whether patent drawing pages are vertical or horizontal. Word detection and page orientation are performed in different orders by the top-5 systems. As seen in Table 2, PC and GS (the second and fourth place systems) perform OCR before clustering connected components (CCs) into 'words' and estimating the page orientation, while the remaining systems estimate orientation and word locations before OCR. GS considers three rather than two page orientations; portrait, along with landscape rotated left, and landscape rotated right.

Character recognition (Table 2) While all systems employ binary images for connected component analysis, two of the systems (PC and WL) make use of grayscale values during classification. All systems make some use of a template grid feature, dividing a connected component image into a fixed number of rows and columns (see Table 2).

Template classifiers or neural networks are used for character classification. These are logical given the time constraints of the competition, as both are fast to execute. All neural networks are multi-layer perceptrons with a single hidden layer (although with differing numbers of nodes in each layer).

For the template character classifiers (first and third place, JC and WL), templates are defined using training images: JC uses a small set of characters directly taken from training images for just the characters *0–9, f, g, a* and *c*, while WL uses approximately 5000 images, which are then averaged to produce a template for a much larger character set.

For the multi-layer perceptron (MLP) classifiers, the fourth and fifth place systems (GS and TZ) use characters located in the provided training images, while the second place system (PC) uses training image characters along with synthetic characters generated using fonts with different parameters. This produces a total of 145,000 training samples (95 % of which are used to train the final classifier). Characters from training images are transformed to eight different slant angles, and 89,000 synthetic characters are created using different variations of the 'Hershey' font, along with variations in thickness and slant.

The PC (second place) system is unique in that font metrics used for classification are iteratively re-estimated at run-time. PC performs OCR in stages, adapting font parameters to revise character classification and segmentation results (e.g., when character spacing is re-estimated for figure titles). Some other systems adapt parameters at run-time, but to a lesser degree. A number of the systems discard 'words' whose size differs significantly from the mean size of detected words, for example.

A number of the systems use thresholds on character recognition confidence in ranking and selecting figure title and part label hypotheses (e.g., in JC, to control correction of part labels—see Table 3).

Figure detection & title recognition (Table 3) The most distinctive figure title detection and recognition strategy is that of WL (third place system). Rather than matching patterns in recognized character strings, a word shape model based on the geometry of 'Fig', 'Figure' etc. is used to locate possible titles before applying OCR, after which this word shape model is updated to match the best candidate (determined by location of the title and character confidence). Further, figure regions are obtained using a top-down X–Y cutting approach [38], rather than bottom-up agglomerative clustering of words or CCs.

Characters (Table 2) & title/label syntax (Table 3). A key difference between the top-5 systems are the language models used for character classes and titles/label syntax, as seen in Tables 2 and 3. The JC (first place) system makes use of only 14 character classes, with no class for 'i' or 'I', which class '1' is expected to catch. The reason that this does not lead to terrible performance is the correction mechanism used. For both figure titles and part labels, words extracted from the patent HTML text using pattern matching define a set of possible output strings, and hypothesized figure titles are matched to their most similar sequence in the patent text, and low-confidence part labels are corrected using the most similar part label detected in the HTML text (see Table 3). JC was the only system to use the HTML text to validate and constrain titles and part labels.

Figure title syntax is similar across systems, with some variations of ‘fig’ assumed to be at the beginning of a figure title. Most systems assumed that part labels are four characters long except for the second place system, where up to nine characters are permitted. All systems reject labels inconsistent with their language model in order to avoid false positives, and some also reject titles or labels containing low-confidence characters (see Table 3 for details).

5 Benchmark: part label recognition system

After the competition, we created a system for part label detection, to use as a benchmark for evaluating and contextualizing USPTO participant solutions. The comparison system was developed using state-of-the-art techniques.

System description Like the USPTO solutions, our comparison system uses a data-driven, bottom-up recognition pipeline. Feature learning and sliding windows are used for text detection, along with Tesseract [52] for OCR. Part label text is detected using a boosted ensemble of three binary AdaBoost base classifiers. Image patches at different resolutions are used to accommodate different character sizes. Three template dictionaries are learned using convolutional k-means [10] for (1) text patches, (2) background patches, and (3) combined foreground and background patches. Nearby pixels detected as text within are clustered into candidate words, recognized by Tesseract and then corrected if necessary.

This system improves upon an earlier system [67]. Improvements included: (1) modifying training label regions to more tightly crop around figure and part label text, (2) CC size filters and page header removal, (3) using projection profile cutting (using gaps in pixels projected on the x or y-axis) to detect and separate multiple text lines in detected word regions, and (4) refined part label language model, and improved validation and correction of OCR results.

Language model and validation Character classes were 0-9a-zA-Z. Labels must contain at least one digit and be at most four characters long. At most two characters may be alphabetic. Isolated ‘i’ ‘I’ and ‘0’ characters are rejected, along with words containing ‘Figure,’ ‘Fig,’ etc. Letters ‘O’ and ‘o’ are replaced by ‘0.’

OCR is performed twice for each candidate label, using progressively larger paddings around a detected label. If the results differ, the result producing the minimum Levenshtein (i.e., string edit) distance [59] with a word in the set of words extracted from the HTML patent text is selected for output (similar to the first place USPTO system).

6 Results

6.1 Participants

The challenge drew 232 teams (463 participants), of which 70 teams (30 %) submitted code. The remaining teams were lurkers and did not actively participate in the competition. Twenty-nine countries were represented among the participants who submitted solutions. The group of submitters included 49 % professionals, 39 % students, and the remainder reporting not working or working part-time. The majority of participants were between 18 and 44 years. Seven of the participants were academics (PhD students, professors, or other research positions). Most (80 %) non-student participants were self-described software developers of various kinds.

Collectively, teams submitted 1797 solutions that compiled on the competition servers, averaging to 25.7 submissions per team. The submitted solutions used four programming languages (C#, C++, Java, VB). Participants reported spending an average of 63 h each developing solutions, for a total of 5591 h of development time.

6.2 Overall ranking

All submissions were scored and ranked using the method described in Sect. 3.3.[4] A test set of 93 drawing pages and HTML texts was used for evaluation (see Sect. 3.2). The winning solution (JC) was created by a team of two participants from the USA and South Africa. Figure 3 provides boxplots illustrating variance in performance of the top-5 systems on the test data. All algorithms fail on at least some test cases in both tasks achieving a score of zero. Conversely, all algorithms also achieve a perfect score for at least some test cases. The figure detection task was significantly easier, with most algorithms receiving a perfect score on many test cases.

We find a high correspondence in ranks between the two tasks (Kendall’s tau rank correlation of 0.511; $p < .05$ for the first ten ranks) indicating that teams that did well in one task also did well on the other. No system outside the top-5 scored higher on any individual task (figure or part label detection). Consequently, the discussion of the top-5 solutions covers the best submitted approaches.

We performed additional analyses changing the relative weighting between accuracy and execution time. We find that the relative ranking of the top-5 algorithms is relatively stable to different weights. Decreasing the weight of execution time would not affect the ranking at all: all top-5 systems would be ranked in exactly the same order and the top-ranked algo-

[4] The final ranking of all submissions is publicly available on the TopCoder Web site at https://community.topcoder.com/longcontest/stats/?\&sr=1\&nr=50\&module=ViewOverview\&rd=15027.

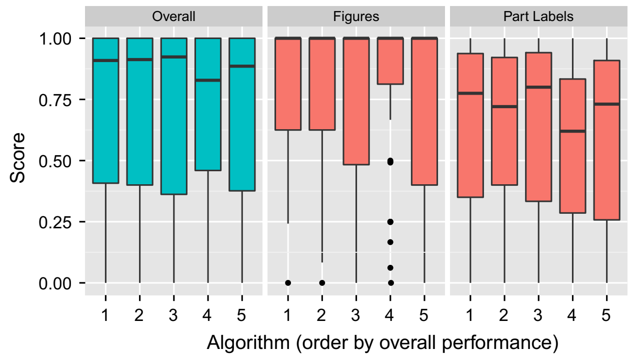

Fig. 3 Boxplot of algorithm performance (achieved competition score which combines F-measure accuracy and run-time as given in Eq. 4) of individual test case score (*dots* show data beyond the 1.5 inter-quartile range; i.e., outliers). At *left* is overall score (including time taken), at *center* is weighted F-measure for figure regions and titles, and at *right* is weighted F-measure for part labels (1.0: perfect score; 0: complete failure)

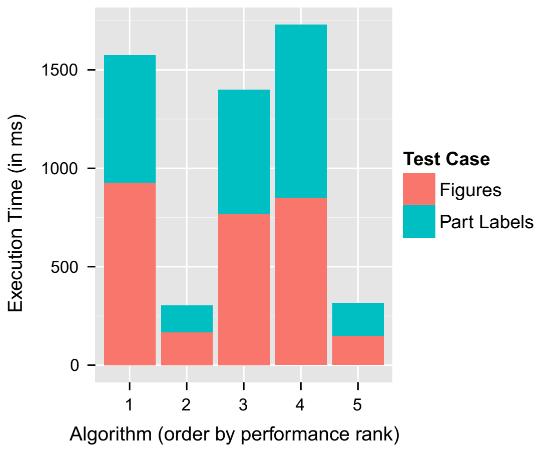

Fig. 4 Average execution time per test case (time given in milliseconds)

rithm would remain the overall winner, even if execution time was not considered in system scoring. Increasing the relative importance of execution time to 50 % would result in only one change: The systems ranked fourth and fifth would switch rank. In summary, while the relative weighting of accuracy and execution time may have guided developers in their algorithm design during the competition, the chosen weighting mattered little in determining contest winners.

6.3 Speed

Average execution speed per test case is shown in Fig. 4. Runtimes are in milliseconds as measured on a quad-core Intel Xeon 3.60 GHz with 4 GB of RAM. In all cases, part label detection is slower than figure detection and labeling. This is because there are many more part labels than figure titles in patent diagrams. All character recognizers used in the top-5 systems have fast execution, as a penalty is incurred when execution take more than 1 s per page (see Sect. 3.3).

The PC and TZ systems (second and fifth place) are much faster than the other systems. They are faster by more than a full second per page, on average. This is despite PC using an iterated font adaptation for its classifier, which is constrained based on character confidences (i.e., if confidences are high, adaptation is not performed). TZ has the simplest design, using simple geometric and visual features and running OCR just once with simple correction (see Table 3).

The slower execution of the remaining systems can be explained as in the following. GS (fourth) runs full OCR and page analysis in three different page orientations, using two different classifiers for every character, making this the slowest system. WL resizes each character image to fit a different template width for every character class, while JC uses an iterated random walk to locate figure regions, and its validation using HTML text performs a linear search over all candidate words.

Benchmark Average execution time of the benchmark part label recognizer was 7.52 s, using a *single process* on a 24-core Intel Xeon 2.93 GHz with 96 GB of RAM. Executing Tesseract takes roughly 1–2 s. This slow execution is due to using Python for programming, and the cost of pixel-level convolutions for the three visual word dictionaries (which could be accelerated significantly using a GPU).

Clearly, the top-5 USPTO competition systems have much faster average execution times, even taking into account the slightly slower processor on which the benchmark was run. The substantially slower execution of the benchmark system emphasizes the high run-time performance for competition systems, particularly when their detection accuracies are taken in account, which we discuss next.

6.4 Recognition accuracy

Figures 1 and 2 illustrate some of the challenges for recognition of USPTO figures titles, figure locations, and part labels. These include the presence of multiple figures on a single page and multiple page orientation, the intersection of part labels with drawing elements, drawing elements similar in appearance to characters, the variety of font faces, the use of handwriting, and slanted text (see Fig. 2).

Table 4 shows the average metrics for figure location and title detection (top panel), and part label locations and text (bottom panel). Part label detection results are also shown for the benchmark algorithm (*Bmk.*). The best result in each panel is that with the highest F-measure ($2RP/(R + P)$), where R is weighed recall and P is weighted precision (see Sect. 3.3). The F-measure penalizes both low recall (i.e., many false negatives) and low precision (i.e., many false positives).

Table 4 Accuracy: average recall, precision and F-measures for localization (BB detection) and localization with correct labels

Rank	Location (BB)			Location + label		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Figures						
1	0.89	0.89	0.89	0.79	0.79	0.79
2	**0.92**	**0.91**	**0.92**	0.79	0.79	0.79
3	0.89	0.89	0.89	0.78	0.78	0.78
4	0.86	0.88	0.87	**0.80**	**0.82**	**0.81**
5	0.87	0.88	0.87	0.76	0.77	0.77
Parts						
1	0.83	0.78	**0.81**	0.72	0.70	**0.71**
2	0.69	**0.89**	0.78	0.60	**0.79**	0.69
3	0.83	0.73	0.77	0.74	0.69	**0.71**
4	0.65	0.85	0.74	0.54	0.76	0.63
5	**0.84**	0.65	0.73	**0.78**	0.62	0.69
Bmk.	0.80	0.84	**0.82**	0.72	0.74	**0.73**

Top figure results, *bottom* part results
Highest score highlighted in bold

Summary Overall, the difference in average figure detection accuracy for the top-5 systems is small. They are within 4–6 % for all recall, precision and F-measures for figure localization, and when also matching title text (labels). For part labels, the difference in F-measures increases to 8 % and the variance in precision and recall measures across top-5 systems increasing substantially. The highest part label F-measures are also 10–11 % lower than the highest figure F-measures, reflecting the greater difficulty of correctly locating part labels.

Figures GS (fourth place) has the strongest average figure localization and title results, with better than 80 % recall and precision. Interestingly, this was not the best system for localizing figures—in fact, GS has a 5 % lower F-measure than PC (second place), but then recognizes a number of titles incorrectly. The GS system employs two different MLPs for character recognition and detects figure titles simply by matching 'fig' at the beginning of a segmented word (see Tables 2, 3).

Part labels JC and WL (first and third place) have the best average part label location and text (label) results. WL has slightly higher precision, and JC slightly higher recall. PC and GS (second and fourth) find more part labels, with roughly 5–10 % higher recall than JC and WL, but also produce more false positives with 10–15 % lower precision reflecting limited or absent validation for part label text (see Table 3).

The increased part label recall for PC and GS may be explained by two things. First, they apply OCR in multiple directions (two and three, respectively), while the other systems first estimate the page orientation and then apply OCR in one direction. Second, they use the most sophisticated character recognizers (see Table 2). GS has the lowest precision, perhaps in part because it does not remove words located in tables. The high precision but low recall of TZ (fifth place) may be explained by a well-trained MLP classifier paired with strict validation rules (see Table 3), leading to low recall for part label detection.

Benchmark The results for the benchmark part label detector are shown in the bottom panel of Table 4. It obtains slightly higher part label localization and localization with text results, by 1–2 %. The final precision and recall measures for part label localization with text are 71.91 and 73.55 % (F: 72.72 %), compared with 72.14 % precision and 69.87 % recall (F: 70.99 %) by JC (first place).

This shows that part label detection accuracy in JC (first place) is very close in performance to a system using convolutional feature learning and a sophisticated OCR engine. Our benchmark benefits from being designed after the competition results had been published, which were used to inform the system design. Given that participant systems needed to operate under strict time and space constraints and had no benefit of hindsight, we argue that our results confirm that the top-performing USPTO systems are of high quality.

7 Lessons learned

We learned several valuable lessons about how to best organize challenges. A first observation is that participants in online competitions will use all available information. While only few competitors leveraged the HTML text we provided, some solutions use it to obtain relevant performance improvements, and this is a technique that may be beneficial in other document recognition applications.

Second, to draw a broad audience, it is important to make start-up costs to participate as low as possible. In our case, we provided clear instructions and a stub implementation of the two functions that needed to be implemented, specification of what the expected input and output is, and training data. We also provided an offline tester/visualizer (including source code), and an online system with a separate, small dataset for participants to automatically test their results during the competition.

Third, for engineering efforts, it is important that the evaluation of systems reflect real-world solution requirements, particularly when a winner is to be selected and given a reward. If a scoring function fails to discriminate among top solutions, e.g., because it reaches a maximum and assigns the same scores to different solutions, this can be catastrophic. Furthermore, the scoring function can be used to place emphasis on aspects of solutions that are particularly relevant. In our case, making execution time part of the scoring function meant that participants had to optimize their solutions for speed at least to some degree to avoid being penalized. This illustrates the use of scoring functions as a general approach to support system engineering to build practical systems. Furthermore, a simple weighting mechanism can be used to adjust the relative importance of different solution aspects. Depending on the application, competition organizers (i.e., firms or other solution seekers) can steer the trade-off between accuracy dimensions (e.g., precision and recall), execution time and space, or other aspects as needed.

Fundamental to innovation is the ability to successfully solve scientific, technical, and design problems. However, often it is not apparent *ex ante* which approaches are most promising and the R&D process is fraught with uncertainty [40]. One leading view casts innovation and problem solving as a process of 'search' over some poorly understood knowledge landscape [51].

A 'parallel search' approach whereby multiple independent solvers (or teams of solvers) compete to solve the same innovation problem in parallel is a widely used approach to address this challenge. Following a parallel-path strategy allows the seeker of a solution to expose a problem to a set of solvers with varying skills and who might employ varying approaches when it is *ex ante* unknown which approach might be successful [40,56]. Our competition shows how this parallel search process was not just successful in exploring multiple solutions, but also how it can be leveraged to explore clever solutions to parts of the problem that can even be combined (e.g., the use of validation using HTML text). In this parallel search, even weaker systems can surprise with interesting solutions to subproblems. It is then possible to break up solutions of interconnected modules based on a decomposition in subprocesses.

An important challenge for researchers in document analysis and pattern recognition more broadly is the construction of frameworks that help identify and formalize modules created for competitions under time constraints, so that they can be later pulled apart and recombined in a repository of modules and then used or refined in future work (e.g., follow-on competitions). There has been some early work along these directions [32,64], but more might be done.

8 Conclusion

In this paper, we present the results of a monthlong algorithm competition to solve a difficult text processing and graphics recognition task for the USPTO. In summary, we show in detail the results of using a prize-based contest to recruit skilled software developers to productively apply their knowledge to a relevant graphics recognition task in a practical setting. The resulting diversity in the submitted solutions has the potential to further improve the solution, for example, by combining the most promising solutions to sub-problems. In the simplest case, performance could be improved by combining the best solution for the figure detection task (fourth place system) with the best solution for the label detection task (first place system). The comparison against the performance of a leading alternative implementation confirms the quality of the top-performing systems.

The emergence of commercial online contest platforms such as TopCoder, InnoCentive, and Kaggle, which offer access to large pools of skilled software developers, has the potential to enable organizations to crowdsource solutions to difficult software and algorithm development tasks which could not have been developed in-house. This approach could be especially useful to address the demand for the many different and highly specialized graphics recognition algorithms which are required as a result of an explosion available imaging data.

The top algorithms presented in this paper used a variety of approaches, were fast, and also accurate. Although the systems perform well, they are not yet accurate enough to be put into every day use. However, the scoring mechanisms that reflect real-world performance considerations, the training data, including ground truth, and the top-performing solutions are openly available. We hope that this will stimulate additional research in patent document analysis. Releasing the source code of the five winning solutions makes a breadth of alternative approaches available and offers the opportunity to study the specific causes of differences in performance. The analyses we presented in this work are a first step in that direction. The results of the winning teams provide a great starting point for future developments, and the implementation of a repository of modules leveraging the diversity of submitted solutions will hopefully lead to more accurate solutions in the future.

Acknowledgments We are grateful for helpful comments provided by Ahmad Ahmad and the anonymous reviewers. This research was supported in part by the NASA Tournament Laboratory and the United States Patent and Trademark Office (USPTO).

References

1. Archak, N.: Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on topcoder.com. In: Proceeding of the International Conference World Wide Web, pp. 21–30 (2010)
2. Barney Smith, E., Belaid, A., Kise, K. (eds.): Proceedings of the International Conference Document Analysis and Recognition. IEEE Computer Society, Washington, DC (2013)
3. Bhatti, N., Hanbury, A.: Image search in patents: a review. Int. J. Doc. Anal. Recognit. **16**(4), 309–329 (2013)
4. Blumenstein, M., Pal, U., Uchida, S. (eds.): Proceedings of the International Work. Document Analysis Systems. IEEE Computer Society, Gold Coast, Australia (2012)
5. Boudreau, K.J., Lacetera, N., Lakhani, K.R.: Incentives and problem uncertainty in innovation contests: an empirical analysis. Manag. Sci. **57**(5), 843–863 (2011)

6. Boudreau, K.J., Lakhani, K.R.: Using the crowd as an innovation partner. Harv. Bus. Rev. **91**(4), 61–69 (2013)
7. Bukhari, S.S., Shafait, F., Breuel, T.M.: Coupled snakelets for curled text-line segmentation from warped document images. Int. J. Doc. Anal. Recognit. **16**(1), 33–53 (2013)
8. Casey, R., Lecolinet, E.: Strategies in character segmentation: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **18**(7), 690–706 (1996)
9. Chan, K.F., Yeung, D.Y.: Error detection, error correction and performance evaluation in on-line mathematical expression recognition. Pattern Recognit. **34**(8), 1671–1684 (2001)
10. Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D., Ng, A.: Text detection and character recognition in scene images with unsupervised feature learning. In: Proceedings of the International Conference Document Analysis and Recognition, pp. 440–445. Beijing, China (2011)
11. Do, T.H., Tabbone, S., Ramos-Terrades, O.: Text/graphic separation using a sparse representation with multi-learned dictionaries. In: Proceedings of the International Conference Pattern Recognition, pp. 689–692. Tsukuba, Japan (2012)
12. Doermann, D., Tombre, K. (eds.): Handbook of Document Image Processing and Recognition, vol. 2. Springer, New York (2014)
13. Dori, D., Wenyin, L.: Automated CAD conversion with the machine drawing understanding system: concepts, algorithms, and performance. IEEE Trans. Syst. Man Cybern. A **29**(4), 411–416 (1999)
14. D'Ulizia, A., Ferri, F., Grifoni, P.: A survey of grammatical inference methods for natural language learning. Artif. Intell. Rev. **36**(1), 1–27 (2011)
15. Embley, D.W., Hurst, M., Lopresti, D.P., Nagy, G.: Table-processing paradigms: a research survey. IJDAR **8**(2–3), 66–86 (2006)
16. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: IEEE Conference Computer Vision and Pattern Recognition, pp. 2963–2970 (2010)
17. Fletcher, L., Kasturi, R.: A robust algorithm for text string separation from mixed text/graphics images. IEEE Trans. Pattern Anal. Mach. Intell. **10**(6), 910–918 (1988)
18. Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.): Information Access Evaluation. Multilinguality, Multimodality, and Visualization—4th International Conference of the CLEF Initiative, Lecture Notes in Computer Science, vol. 8138. Springer, Valencia (2013)
19. Fu, K.S., Booth, T.L.: Grammatical inference: introduction and survey—part I. IEEE Trans. Syst. Man Cybern. **5**(1), 95–111 (1975)
20. Fu, K.S., Booth, T.L.: Grammatical inference: introduction and survey—part II. IEEE Trans. Syst. Man Cybern. **5**(4), 409–423 (1975)
21. Fullerton, R.L., McAfee, R.P.: Auctioning entry into tournaments. J. Polit. Econ. **107**(3), 573–605 (1999)
22. Gobeill, J., Teodoro, D., Pasche, E., Ruch, P.: Report on the TREC 2009 experiments: chemical IR track. In: Text Retrieval Conference (TREC'09) (2009)
23. Howe, J.: Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business. Crown, New York (2008)
24. Jung, K., Kim, K.I., Jain, A.K.: Text information extraction in images and video: a survey. Pattern Recognit. **37**(5), 977–997 (2004)
25. Kalil, T., Sturm, R.: Congress grants broad prize authority to all federal agencies (2010). http://wh.gov/OSw
26. Kanungo, T., Haralick, R., Dori, D.: Understanding engineering drawings: a survey. In: Proceedings of Work. Graphics Recognition, pp. 217–228 (1995)
27. Karatzas, D., Mestre, S.R., Mas, J., Nourbakhsh, F., Roy, P.P.: ICDAR 2011 robust reading competition-challenge 1: reading text in born-digital images (web and email). In: Proceedings of the International Conference Document Analysis and Recognition, pp. 1485–1490 (2011)
28. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Gomez i Bigorda, L., Robles Mestre, S., Mas, J., Fernandez Mota, D., Almazan Almazan, J., de las Heras, L.P.: ICDAR 2013 robust reading competition. In: Proceedings of the International Conference Document Analysis and Recognition, pp. 1484–1493 (2013)
29. Koo, H., Kim, D., et al.: Scene text detection via connected component clustering and non-text filtering. IEEE Transaction Image Processing, pp. 2296–2305 (2013)
30. Lai, C., Kasturi, R.: Detection of dimension sets in engineering drawings. IEEE Trans. Pattern Anal. Mach. Intell. **16**(8), 848–855 (1994)
31. Lakhani, K.R., Boudreau, K.J., Loh, P.R., Backstrom, L., Baldwin, C., Lonstein, E., Lydon, M., MacCormack, A., Arnaout, Ra, Guinan, E.C.: Prize-based contests can provide solutions to computational biology problems. Nat. Biotechnol. **31**(2), 108–111 (2013)
32. Lamiroy, B., Lopresti, D.: An open architecture for end-to-end document analysis benchmarking. In: Proceedings of the International Conference Document Analysis and Recognition, pp. 42–47. Beijing, China (2011)
33. Liang, J., Doermann, D.S., Li, H.: Camera-based analysis of text and documents: a survey. Int. J. Doc. Anal. Recognit. **7**(2–3), 84–104 (2005)
34. Lu, T., Tai, C.L., Yang, H., Cai, S.: A novel knowledge-based system for interpreting complex engineering drawings: theory, representation, and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(8), 1444–1457 (2009)
35. Lu, Z.: Detection of text regions from digital engineering drawings. IEEE Trans. Pattern Anal. Mach. Intell. **20**(4), 431–439 (1998)
36. Lupu, M., Hanbury, A.: Patent retrieval. Found. Trends Inf. Retr. **7**(1), 1–97 (2013)
37. Mervis, J.: Agencies rally to tackle big data. Science **336**(6077), 22 (2012)
38. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 38–62 (2000)
39. Nagy, G., Embley, D.W., Krishnamoorthy, M.S., Seth, S.C.: Clustering header categories extracted from web tables. In: Ringger, E.K., Lamiroy, B. (eds.) Document Recognition and Retrieval XXII, Proceedings of SPIE, vol. 9402, p. 94020M. San Francisco (2015)
40. Nelson, R.R.: Uncertainty, learning, and the economics of parallel research and development efforts. Rev. Econ. Stat. **43**(4), 351–364 (1961)
41. Niemeijer, M., Van Ginneken, B., Cree, M., Mizutani, A., Quellec, G., Sanchez, C., Zhang, B., Hornero, R., Lamard, M., Muramatsu, C.: Others: retinopathy online challenge: automatic detection of microaneurysms in digital color fundus photographs. IEEE Trans. Med. Imaging **29**(1), 185–195 (2010)
42. Ogier, J.M. (ed.): Proceedings of the International Work. Graphics Recognition (GREC 2013), Lecture Notes in Computer Science, vol. 8746. Springer, Bethlehem, PA (2014)
43. O'Gorman, L.: The document spectrum for page layout analysis. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1162–1173 (1993)
44. Rice, S.V., Jenkins, F.R., Nartker, T.A.: The fifth annual test of OCR accuracy. Information Science Research Institute (1996)
45. Rice, S.V., Nagy, G.L., Nartker, T.A.: Optical Character Recognition: An Illustrated Guide to the Frontier. Kluwer, New York (1999)
46. Roy, P.P., Pal, U., Lladós, J.: Text line extraction in graphical documents using background and foreground information. Int. J. Doc. Anal. Recognit. **15**(3), 227–241 (2012)
47. Rusiñol, M., de las Heras, L., Ramos, O.: Flowchart recognition for non-textual information retrieval in patent search. Inf. Retr. **17**(5–6), 545–562 (2014)

48. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: LabelMe: a database and web-based tool for image annotation. Int. J. Comput. Vis. **77**(1–3), 157–173 (2008)
49. Sadawi, N.M., Sexton, A.P., Sorge, V.: Performance of MolRec at TREC 2011—overview and analysis of results. In: The Twentieth Text REtrieval Conference Proceedings (TREC). National Institute of Standards and Technology (NIST), USA (2011)
50. Shahab, A., Shafait, F., Dengel, A.: ICDAR 2011 robust reading competition challenge 2: reading text in scene images. In: Proceedings of International Conference Document Analysis and Recognition, pp. 1491–1496 (2011)
51. Simon, H., Newell, A.: Computer simulation of human thinking and problem solving. Monogr. Soc. Res. Child Behav. **27**, 137–150 (1962)
52. Smith, R.: An overview of the Tesseract OCR engine. In: Proceedings of the International Conference Document Analysis and Recognition, vol. 2, pp. 629–633. Curitiba, Brazil (2007)
53. Sturm, R.: New center for excellence fuels prize to help modernize tools for patent examination (2011). http://wh.gov/DdM
54. Tassey, G., Rowe, B.R., Wood, D.W., Link, A.N., Simoni, D.A.: Economic impact assessment of NIST's text REtrieval conference (TREC) program. National Institute of Standards and Technology (2010)
55. Terwiesch, C., Ulrich, K.T.: Innovation Tournaments: Creating and Selecting Exceptional Opportunities. Harvard Business Press, Boston (2009)
56. Terwiesch, C., Xu, Y.: Innovation contests, open innovation, and multiagent problem solving. Manag. Sci. **54**(9), 1529–1543 (2008)
57. Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B., Dosch, P.: Text/graphics separation revisited. In: Lopresti, D.P., Hu, J., Kashi, R.S. (eds.) Document Analysis Systems, Lecture Notes in Computer Science, vol. 2423, pp. 200–211. Springer, Berlin (2002)
58. Valveny, E., Lamiroy, B.: Scan-to-XML: automatic generation of browsable technical documents. In: Proceedings of the International Conference Pattern Recognition, vol. 3, pp. 188–191. Québec City, Canada (2002)
59. Wagner, R., Fischer, M.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974)
60. Wendling, L., Tabbone, S.: A new way to detect arrows in line drawings. IEEE Trans. Pattern Anal. Mach. Intell. **26**(7), 935–941 (2004)
61. Wu, V., Manmatha, R., Riseman, E.: Textfinder: an automatic system to detect and recognize text in images. IEEE Trans. Pattern Anal. Mach. Intell. **21**(11), 1224–1229 (1999)
62. Ye, Q., Doermann, D.: Text detection and recognition in imagery: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **37**(7), 1480–1500 (2015)
63. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition. Int. J. Doc. Anal. Recognit. **7**(1), 1–16 (2004)
64. Zanibbi, R., Blostein, D., Cordy, J.R.: White-box evaluation of computer vision algorithms through explicit decision-making. Computer Vision Systems. Lecture Notes in Computer Science, vol. 5815, pp. 295–304. Springer, Liège, Belgium (2009)
65. Zheng, Y., Li, H., Doermann, D.: Machine printed text and handwriting identification in noisy document images. IEEE Trans. Pattern Anal. Mach. Intell. **26**(3), 337–353 (2004)
66. Zhou, W., Li, H., Lu, Y., Tian, Q.: Principal visual word discovery for automatic license plate detection. IEEE Trans. Image Process. **21**(9), 4269–4279 (2012)
67. Zhu, S., Zanibbi, R.: Label detection and recognition for USPTO images using convolutional k-means feature quantization and AdaBoost. In: Proceedings of the International Conference Document Analysis and Recognition, pp. 633–637. Washington, DC (2013)