PixLabeler: User Interface for Pixel-Level Labeling of Elements in Document Images

Eric Saund
Jing Lin
Prateek Sarkar
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA, USA
{saund, jlin, psarkar}@parc.com

Abstract

We present a user interface design for labeling elements in document images at a pixel level. Labels are represented by overlay color, which might map to such terms as "handwriting", "machine print", "graphics", etc. The primary purpose is to streamline processes for manual production of groundtruth data, which is necessary for training algorithms and evaluating performance. Unlike general paint-type programs, the UI design is targeted specifically toward selection of collections of foreground pixels that are likely to be meaningful elements in a document image analysis context. Our implementation, called PixLabeler*, is available for download and allows customized plug-ins for bootstrapping according to the labeling task.*

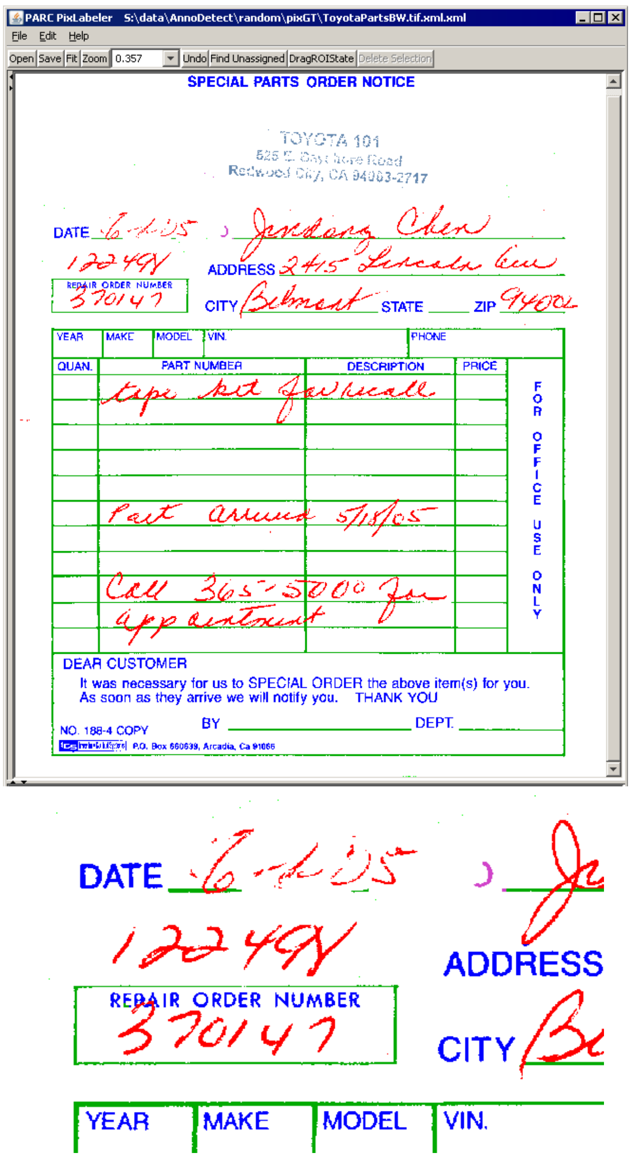
1. Introduction

The need for groundtruth is pervasive in document image analysis. Groundtruth is needed to train machine learning based algorithms, and it is needed to evaluate algorithm performance. But groundtruth data can be difficult to come by. It is tedious and time-consuming to produce, and it relies on formats and standards that may differ from research group to research group. The lack of abundant and high quality groundtruth is consistently mentioned in workshops as an impediment to the field. Commercial OCR companies view their large databases of groundtruthed documents as competitive assets.

Largely as a matter of expediency, most groundtruthing focuses on labeling of rectangular regions, according to type, [3, 6, 15]. Polygonal regions are a recent development [24]. Some investigators employ groundtruth at the level of connected components [10], stroke fragments [8], groupings of connected components [18, 22], or entire text lines [11, 13]. These region definitions are perhaps suitable for many page layout analysis applications, but rectangles are too crude a granularity for segmenting handwritten words, for distinguishing noise scattered among text, or for labeling graphics as distinct from text. For these purposes, a preferred approach is pixel level labeling of image material [1, 4, 9, 19, 17]. Pixel labels can form the basis for a more universal measure for assessing labeling quality, either on a pixel count basis or in terms of groupings of labeled pixels into connected components or larger structures.

Our own interest in this problem originates from the need for groundtruth in order to train and evaluate algorithms for labeling marking type. We wish to distinguish machine print text, machine print graphics, handwritten text, handwritten graphics, stamps, salt-and-pepper noise, and other noise, as shown in Figure 1. Most previous work on this problem has relied on groundtruth at level of connected components and word boxes. This is not suitable for groundtruthing images where different marking types touch or overlap. We were able to take advantage of a relatively small number of groundtruthed pages shared by Zheng et al., but required a larger amount of pixel level groundtruth. General paint-style tools such as Photoshop are unsuitable for this task, which can rapidly become very labor-intensive [1].

For this reason, we designed and implemented a customized pixel-level labeling tool, for document images. Documents images are characterized by certain kinds of structure that lend themselves to the use of accelerated user interface commands and techniques, so that large collections of like-label pixels can be tagged together, with few user operations. Previous document image groundtruthing tools commonly take advantage of the layout and visual structure in document images [7, 20, 21]. The work most


PARC PixLabeler S:\data\AnnoDetect\random\pixGT\ToyotaPartsBW.tif.xml.xml

File Edit Help

Open Save Fit Zoom 0.357 Undo Find Unassigned DragROIState Delete Selection

SPECIAL PARTS ORDER NOTICE

TOYOTA 101
525 E. Bayshore Road
Redwood City, CA 94063-2717

DATE 6-1-05 Jindong Chen

12249N ADDRESS 2415 Lincoln Ave

REPAIR ORDER NUMBER 370147 CITY Belmont STATE ZIP 94002

YEAR	MAKE	MODEL	VIN.	PHONE

QUAN.	PART NUMBER	DESCRIPTION	PRICE	FOR OFFICE USE ONLY
	tape kit for recall			
	Part arrived 5/18/05			
	Call 365-5000 for appointment			

DEAR CUSTOMER
It was necessary for us to SPECIAL ORDER the above item(s) for you.
As soon as they arrive we will notify you. THANK YOU

NO. 188-4 COPY BY ______ DEPT. ______

P.O. Box 660639, Arcadia, Ca 91066

Figure 1. Screen shot of PixLabeler after labeling a binary image according to pixel type. red:handwritten; blue:machine-print: green:machine graphics; grey: stamp

close in spirit to our own is the chart groundtruthing system of Yang, et al. [21]. Here, we focus expressly on pixel-level groundtruthing and user interface design for rapid selection and labeling of meaningful groups of pixels. The remainder of the paper describes the major UI features of our program, and the devices behind them. These include dual enclosure/brush selection modes, a "tentative labeling" state, grouping of pixels through layers, automatic grouping by structure recognition, bootstrapping with task-specific pro-

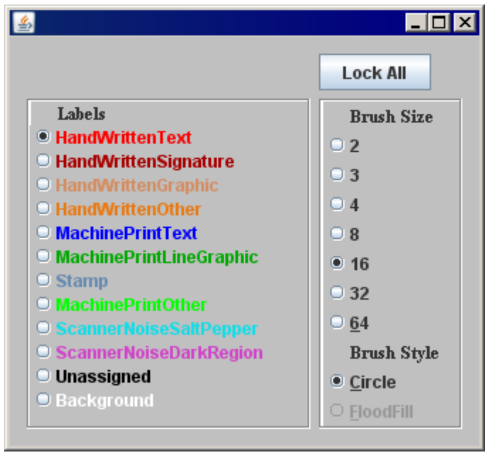


Figure 2. PixLabeler label menu. Rectangle indicates that label is locked.

grams, and tools for comparison of groundtruth produced by multiple users.

2. User Interface Design

REGION and BRUSH Modes

Unlike photographic images, documents are dominated by foreground markings set against a relatively uniform white or other color background. In contrast to Baird et al. [1], we elect to label only foreground pixels, not background. Thus the first step is to separate color, greyscale, or bitonal-black foreground markings from background, using well established image processing techniques. Background pixels are treated as immutable "white".

The right mouse button is used to pop up a menu selecting the active label and its associated color that will overlay foreground pixels selected to have that label (Figure 2). Label descriptions and their colors are set by a user-editable XML configuration file.

The user interface exists in one of two primary modes. In REGION mode, the user drags the mouse to select regions of the image by enclosure. A technique called *overloaded loop selection*, first introduced by Saund et al. [16], permits selection by rectangle or lasso without prior selection of which tool is desired. Note in Figure 1 the deliberate absence of a tool palette. While dragging the mouse with the left button pressed, either a selection rectangle or lasso is dynamically indicated depending on the shape of the user's drag as shown in Figure 3. At the point of mouse release, if the selection path is sufficiently closed in shape a lasso

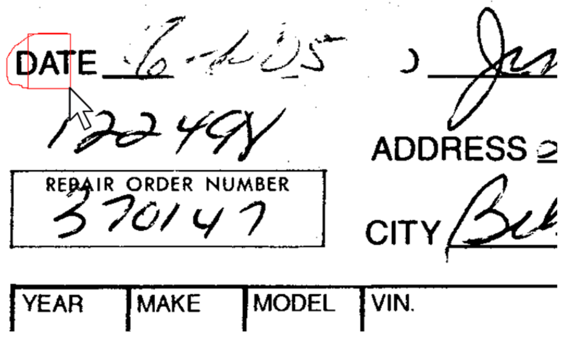


Figure 3. Overloading selection by rectangle drag and lasso.

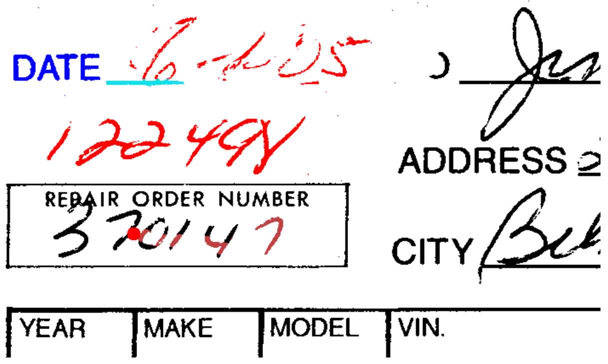


Figure 4. Brush labeling a swath of pixels. Note the desaturated red color where the brush has passed, indicating that these pixels are labeled only tentatively.

enclosure is selected; otherwise, the selection rectangle will be in display and pixels in its region will be selected. Further, double-clicking the mouse initiates a polygon selection state which enables region selection by dropping the vertices of a polygon using mouse clicks. Other capabilities of REGION mode are discussed below.

The second primary mode is BRUSH mode. The user transitions between REGION mode and BRUSH mode by pressing the space bar or clicking a button on the toolbar. In BRUSH mode, the cursor is replaced by a round or otherwise shaped "brush". Foreground pixels are selected when the brush is dragged over them (with the left mouse button held down), as illustrated in Figure 4.

TENTATIVE **State**

In both REGION mode and BRUSH mode, after a user has selected a set of pixels, those pixels are displayed in the color indicating the selected label class. For example, the user may define red to mean "handwriting" and blue to mean "machine print text". Immediately after any selection, though, the interface first enters a TENTATIVE state and the selected pixels are indicated by a desaturated version of the label color. TENTATIVE state permits the user to easily change the label (and associated color) of the selected pixels, before committing to it. (However, it is always possible to undo any action, and no commitment of pixel labels is irreversible.) This would be done by clicking the right button and selecting a different label (and associated color). Also, in TENTATIVE state, the user can click the Undo button on the toolbar to reset the selected pixels to their prior labels, or unassigned status. Unlike the standard *select*-then-*command* paradigm of image editing tools, the labeling task is facilitated by combining both of these steps into a single operation. In user interface design terms, TENTATIVE state is a quasi-mode that gives the user the opportunity to modify the label immediately after selecting material, but the state exits automatically and does not get in the way of rapid serial coloring commands.

Pixel Groups

REGION mode also accesses several advanced features of the interface design based on automatic and manual establishment of groups of foreground pixels. Every time the user selects and labels a set of pixels in either REGION or BRUSH mode, those pixels are remembered as a group that can be selected later with a single mouse click. Specifically, in our design, if the mouse is positioned over a pixel that belongs to a group, a Shift-Left-Click command selects a group of foreground pixels that contains that pixel. This makes it easy for a user to do some initial labeling work, then go back and change these labels without tediously reselecting the pixels involved.

It is possible for a pixel to belong to multiple groups. These can be accessed in turn by successive Shift-left-mouse click commands. The set of groups are cycled through one at a time, each displayed in the current tentative label color.

Automatic Detection of Group Structures

Pixel groups can be created not only by manual region- or brush-based selection, but also through automatic processes. In our design, the user may invoke a "Recognize Structure" operation from the Edit menu. Then, automatic image processing is performed to form groups of pixels that correspond to perceptually meaningful elements of the document. In our current implementation, image processing

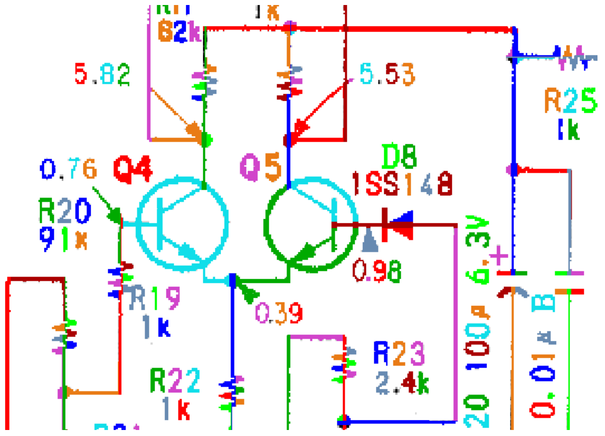


Figure 5. Simple image segmentation forms pixel groups of straight lines and remaining fragmented connected components. Groups are arbitrary colored in this example. In combination with region and brush selection, this simplifies selection of pixels for labeling.

techniques are used to detect horizontal and vertical lines and form these as groups. In a scratch image, horizontal and vertical lines are subtracted from the full image, and remaining connected components collected. Each connected component forms another pixel group that can be selected by point-and-click. Figure 5 illustrates. The program architecture is easily extensible to other forms of structure recognition, including text lines and layout, to the extent automatic programs for detecting these groupings are available.

Bootstrapping

Pixel labels may be initialized by performing a bootstrapping process. This runs an automatic recognition algorithm on the initial image (or any user-selected region of it) and colors foreground pixels with labels obtained by this algorithm. In our implementation which focuses on UI design and not recognition algorithms themselves, users may supply their own custom pixel label bootstrapping programs in the form of a Java JAR file that includes a class that implements a simple interface.

Unassigned Pixels

After a user has labeled nearly all of the foreground pixels in an image, it can be difficult for them to detect pixels they have not yet labeled, such as specks of background noise. The interface design includes a “Find Unassigned Pixels” button on the toolbar. This displays overlay ellipses over all connected components formed by all yet unlabeled pixels, making it easy for the user to detect these.

Layer Locking

PixLabeler includes a locking feature. Users can “lock” any label (color) so that pixels already given this label cannot be changed. This makes it possible to label some pixels in a region, lock the label, then label the remaining pixels in the region all at once through a large region selection operation (rectangle drag, lasso, or polygon). In general, we have observed that the suite of tools provided by PixLabeler invites users to evolve their own strategies for labeling images of various types.

Labeling Comparison

In order to ensure accuracy of groundtruth, it is desirable to have several users label a given image, then compare their output. PixLabeler includes a groundtruth comparison feature. Users can load multiple labelings of a given image, typically obtained by groundtruth efforts of other users or automated processes. They can then invoke a “Compare” function which visually indicates where the groundtruth labelings agree or disagree.

3 Implementation

PixLabeler is implemented in Java/Swing, in about 40 custom classes on top of a larger image analysis toolkit. The state machine diagram for the program is presented in Figure 6. Each selection/labeling operation creates a bitmap layer containing only foreground pixels selected and colored at that operation. The collection of labeling operations forms a stack which is rendered onto a single-layer canvas that the user sees.

The output of a labeling is quite simple, consisting of an XML file listing the mapping from integer indices to label colors (integers) and their textual descriptions, and containing a CDATA block encoding the label-colored image itself in PNG format. For convenience of viewing, a separate standalone png image of the labeling is output as well.

At present the program supports only one label per pixel. In many document image applications it is desirable for image elements to adopt several labels representing, for example, a handwritten signature that overlaps a machine-printed baseline, or the word, line, and paragraph membership of a glyph. This capability is architecturally compatible with PixLabeler, but lies in the realm of future work from an implementation and UI design standpoint.

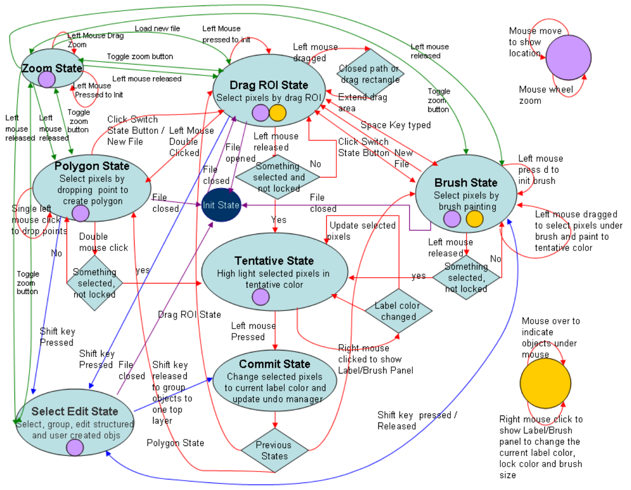

Zoom State
Left Mouse Drag Zoom
Load new file
Left Mouse pressed to init
Toggle zoom button
Left mouse released
Left Mouse Pressed to Init
Drag ROI State
Select pixels by drag ROI
Left mouse dragged
Closed path or drag rectangle
Extend drag area
Left mouse released
Toggle zoom button
Space Key typed
Click Switch State Button New File
Mouse move to show location
Mouse wheel zoom
Left mouse released
Left mouse released
Toggle zoom button
Click Switch State Button / New File
Left Mouse Double Clicked
File opened
File closed
Left mouse released
Something selected and not locked
No
Polygon State
Select pixels by dropping point to create polygon
File closed
Init State
File closed
Yes
Brush State
Select pixels by brush painting
Left mouse press d to init brush
Left mouse dragged to select pixels under brush and paint to tentative color
Single left mouse click to drop points
Double mouse click
No
Update selected pixels
Left mouse released
No
Toggle zoom button
Something selected, not locked
yes
Tentative State
High light selected pixels in tentative color
yes
Something selected, not locked
Label color changed
Drag ROI State
Left mouse Pressed
Mouse over to indicate objects under mouse
Shift key Pressed
Shift key Pressed
File closed
Shift key released to group objects to one top layer
Right mouse clicked to show Label/Brush Panel
Commit State
Change selected pixels to current label color and update undo manager
Shift key pressed / Released
Right mouse click to show Label/Brush panel to change the current label color, lock color and brush size
Select Edit State
Select, group, edit structured and user created objs
Polygon State
Previous States

Figure 6. State diagram detailing the user interaction design of the PixLabeler program.

4 Usability and Release

At this writing, the PixLabeler program is new and has been used to label about 100 images. Our experience with novice users is limited. Anecdotally, one of the authors labeled the image of Figure 1 in ten minutes, using 107 labeling operations, without using any bootstrapping program. Of course this time can be greatly diminished if most of the pixels are already correctly labeled by bootstrapping. Then, PixLabeler is needed only for touch-up and correction.

The program is released for academic use at our web site, http://www.parc.com/PixLabeler. A video of the program in operation is also available at that site. It is our hope that the PixLabeler program will serve the document recognition community as an effective tool that will simplify and streamline the production of much-needed groundtruth for training and evaluation of document recognition algorithms.

References

[1] H. Baird, M. Mill, and C. An. Truthing for pixel-accurate segmentation. *DAS 2008*, 2008.
[2] G. Bal, G. Agam, O. Frieder, and G. Frieder. Interactive degraded document enhancement and ground truth generation. *DAS*, 2008.
[3] G. Ford and G. R. Thoma. Ground truth data for document image analysis. *Symposium on Document Image Understanding and Technology*, 2003.
[4] B. Gatos, A. Antonacopoulos, and N. Stamatopoulos. Icdar2007 handwriting segmentation contest. *ICDAR*, pages 1284–1288, 2007.
[5] J. K. Guo and M. Y. Ma. Separating handwritten material from machine printed text using hidden markov models. *Proc. ICDAR*, pages 439 – 443, 2001.
[6] I. Guyon, R. M. Haralick, J. J. Hull, and I. T. Phillips. Data sets for ocr and document image understanding research. *Proc. SPIE - Document Recognition IV*, 1997.
[7] L. C. Ha and T. Kanungo. The architecture of trueviz: A groundtruth/metadata editing and visualizing toolkit. *Pattern Recognition*, 36(3):811–825, 2003.
[8] G. F. Houle, K. Blinova, and M. Shridhar. Handwriting stroke extraction using a new xytc transform. *Proc. ICDAR*, pages 91 – 95, 2001.
[9] Y. Huang and M. S. Brown. User-assisted ink-bleed correction for handwritten documents. *IEEE and ACM Joint Conf. Digital Libraries*, 2008.
[10] E. Kavallieratou, M. Balcan D.C and, Popa, and N. Fakotakis. Handwritten text localization in skewed documents. *ICIP*, pages 1102–1105, 2001.
[11] E. Kavallieratou, S. Stamatatos, and H. Antonopoulou. Machine-printed from handwritten text discrimination. *IWFHR-9*, pages 312 – 316, 2004.
[12] K. Kuhnke, L. Simoncini, and Z. Kovacs-V. A system for machine-written and hand-written character distinction. *ICDAR*, pages 811–814, 1995.
[13] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger. A new algorithm for detecting text line in handwritten documents. *IWFHR*, pages 35–40, 2006.
[14] R. Manmatha and J. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE. TPAMI*, 27(8):1212 – 1225, August 2005.
[15] O. Okun and M. Pietikainen. Automatic ground-truth generation for skew-tolerance evaluation of document layout analysis methods. *ICPR*, pages 376–379, 2000.
[16] E. Saund, D. Fleet, D. Larner, and J. Mahoney. Perceptually-supported image editing of text and graphics. *ACM UIST*, pages 183–192, 2003.
[17] F. Shafait, D. Keysers, and T. M. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. *ICPR*, pages 872–875, 2006.
[18] S. Shetty, H. Srinivasan, M. Beal, and S. Srihari. Segmentation and labeling of documents using conditional random fields. *Proc. SPIE*, 6500, 2007.
[19] L. Wenyin and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications*, 9:240 – 250, 1997.
[20] S. Yacoub, S. Vinay, and S. Sami. Perfectdoc: A ground truthing environment for complex documents. *DAS*, pages 452 – 456, 2005.
[21] L. Yang, W. Huang, and C. Tan. Semi-automatic ground truth generation for chart image recognition. *DAS*, pages 324–335, 2006.
[22] Y. Zheng, H. Li, and D. Doermann. Machine printed text and handwriting identification in noisy document images. *IEEE TPAMI*, 26(3):337–353, March 2004.
[23] G. Zi and D. Doermann. Document image ground truth generation from electronic text. *ICPR*, pages 663–666, 2004.
[24] E. Zotkina, H. Suri, and D. Doermann. Gedi: Groundtruthing environment for document images. *http://lampsrv01.umiacs.umd.edu/projdb/project.php?id=53*.