7-1-2012

Privacy-Preserving Screen Capture: Closing the Loop for Medical Informatics Usability

Joseph Cooley
Massachusetts Institute of Technology

Sean Smith
Dartmouth College

Privacy-Preserving Screen Capture: Closing the Loop for Medical Informatics Usability☆

Joseph Cooley[1], Sean Smith*

Department of Computer Science
Dartmouth College
Hanover, NH 03755 USA

Abstract

As information technology permeates healthcare (particularly provider-facing systems), maximizing system effectiveness requires the ability to document and analyze tricky or troublesome usage scenarios. However, real-world medical applications are typically replete with privacy-sensitive data regarding patients, diagnoses, clinicians, and EMR user interface details; any instrumentation for screen capture (capturing and recording the scenario depicted on the screen) needs to respect these privacy constraints. Furthermore, real-world medical informatics systems are typically composed of modules from many sources, mission-critical and often closed-source; any instrumentation for screen capture cannot rely on access to structured output or software

☆This work is supported in part by the US National Science Foundations Trustworthy Computing award #0910842, by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002, and by Google. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government or Google.

*Corresponding author

Email addresses: `joe.cooley@gmail.com` (Joseph Cooley), `sws@cs.dartmouth.edu` (Sean Smith)

URL: `www.cs.dartmouth.edu/~sws/` (Sean Smith)

[1]Current affiliation: MIT Lincoln Laboratory, Lexington MA 02421 USA

internals.

In this paper, we present a solution: a system that combines *keyboard video mouse (KVM)* capture with automatic text redaction (and interactively selectable unredaction) to produce precise technical content that can enrich stakeholder communications and improve end-user influence on system evolution. KVM-based capture makes our system both application and operating-system independent because it eliminates software-interface dependencies on capture targets. Using a corpus of EMR screenshots, we present empirical measurements of redaction effectiveness and processing latency to demonstrate system performances. We discuss how these techniques can translate into instrumentation systems that improve real-world medical informatics deployments.

Keywords: EHR/EMR, security, privacy, usability, redaction

---

1. Introduction

Medical enterprises large and small are supplanting paper-based systems with IT-based ones, and upgrading old, piecemeal IT-based systems with new, federated ones. However, as with any large engineering project, it is unlikely that the first solution produced and deployed is exactly right. Standard engineering tenets teach the importance of "closing the loop"; understanding and tuning a system requires measuring it, in order for this tuning to be a data-driven process.

However, when it comes to taking such measurements, medical informatics systems raise a unique combination of challenges:

Privacy Preservation In a human-facing IT system, *screenshots* comprise

the natural domain for measurement. However, in medical systems, screenshots are full of privacy-sensitive material. First, we have the obvious issues: names and identifying information of patients; images of patients; text regarding diagnoses and medication and other treatments. But there are more subtle issues as well, such as names of providers, details of an EMR user interface protected by vendor agreements, and non-text indicators (such as "warning" icons) that can betray confidential patient details.

A measurement methodology needs to respect these privacy constraints—either by putting cumbersome measurements in place to ensure that private data is never leaked throughout the analysis process, or by automatically redacting it in the first place. However, any such redaction system needs to be effective in two ways: both at removing sensitive information, but also at retaining (in conjunction with end-user feedback) the system behavior information we were trying to measure in the first place.

Context Preservation Traditional work on privacy and confidentiality seeks to hide information. However, to fulfill their purpose of tuning and analysis, redacted medical screenshots still need to contain information—a blacked-out screen would preserve all privacy, but be useless. We need to balance hiding of privacy-protected information with communication workflow process context.

System Impact Medical enterprises deploy IT in order to further their medical mission, within the constraints of various business objectives.

A measurement methodology needs to respect these deployment constraints—it cannot make assumptions about underlying applications, operating systems, access to source code, access to structured protocol communications, even access to documentation. Furthermore, a measurement methodology cannot disrupt the underlying system; besides impeding enterprise mission, changes might also invalidate necessary certification.

Workflow Impact For clinicians using medical IT systems, the primary motivation is helping patients rather than wrestling with computing systems—even to document troublesome scenarios in order to enable these systems to be fixed. Consequently, a measurement methodology needs to minimize the work required by these users: they should be able to quickly log some issue, and move on with their real mission.

This Paper. In this paper, we present our research addressing these needs: instrumentation that captures text-redacted keyboard/video/mouse (KVM) traces—the point where *humanspace and cyberspace* [1] intersect. By capturing data at the KVM interface and text-redacting images, we eliminate software interface dependencies. Section 2 provides an overview of our prototype system. Section 3 describes our methodologies for text redaction. Section 4 describes the broader system we built around these techniques. Section 5 evaluates the effectiveness of our approaches. Section 6 presents how this work can impact real-world medical informatics systems. Section 7 reviews related work, and Section 8 concludes.

2. System Overview

Our prototype system applies text and image redaction to KVM feeds from medical informatics systems—see Figure 1.

Our system includes functionality essential to implementing screen capture for sensitive systems. The basic steps of instrumenting such systems include screen capture, image processing and editing, and data sharing. After capture, the system processes an image to find and redact text.Additionally, the system may search for regions within the image that match a set of image snippets or "templates" and count, redact, or unredact matching regions. Finally, a user may wish to edit the image and further redact or unredact a portion of the processed screenshot.

Implementation. The bulk of our system implementation relies on a mixture of C and C++ code spanning multiple open-source libraries and custom-developed libraries and applications, including boost [2], C++ STL [3], OpenCV [4], liblinear [5], and CGAL [6, 7, 8]. Altogether, we implemented approximately 9000 lines of code.

To remain system-independent, we implemented certain functionality with higher-level APIs; our development environment is a MacBook Pro running OS X 10.5 with 8 GB of memory.[2] Certain, low-level OpenCV routines rely on system libraries, but these are transparent to our code—OpenCV is cross-platform.

Screen Capture. Our system relies on a *virtual network computer (VNC)* arrangement to capture screen material from a remote host [9]. In a nut-

[2]We upgraded to OS X 10.6 midway through development and analysis.

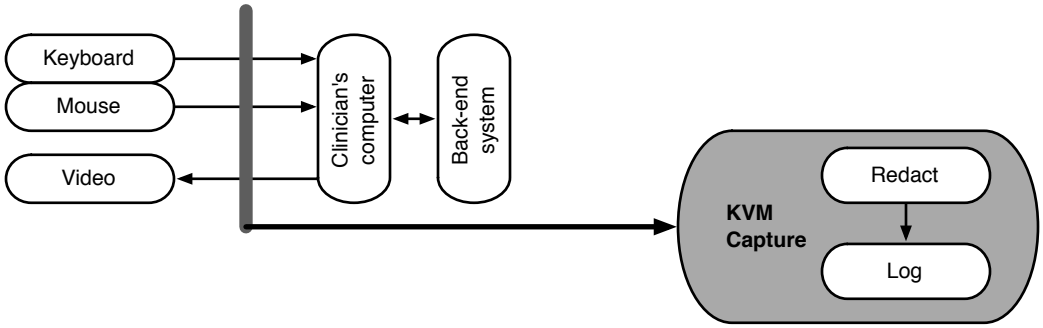

Figure 1: Our measurement module listens to keyboard and mouse input from devices and video output received by the computer—and consequently remains system independent and accommodates closed and certified systems.

shell, VNC defines a protocol for transporting a computer's framebuffer, keyboard, and mouse data over the network. By building a system with this protocol, our system can capture and operate on all KVM events in a system-independent fashion. In our test configuration, Mac OS X 10.6 functions as the "Capture System" and the application x11vnc [10] running on an Ubuntu Linux 9.10 running within a VMware [11] instance serves as the "Capture Target." The client implements read-only functionality and therefore does not pass keyboard or mouse events from the VNC client to the VNC server.

Our client connects to the VNC server using TCP. After connecting, the endpoints proceed through a handshake phase and negotiate the protocol version "RFB 003.008\n" and the "raw" pixel format to transfer screen updates from the server to the client without compression.

3. Text Redaction Approaches

Text redaction is a fundamental aspect of the system because it removes sensitive text from screen capture data, relieving the end-user from manually redacting screen captures before sharing. By default, our approach implemented a "deny-all" policy and thus redacts all text it finds. An end-user can then "unredact" small regions as necessary to facilitate their conversation. Because redaction affects just text and a small number of icons, our intention is that screen context remains despite removal of potentially sensitive data.

In a different approach to redaction, our system could simply redact an entire screen (e.g., turn the entire screen black) and the end-user could unredact whichever small piece supports their needs. We believe this approach provides too little screen context to observers, and would require too much work from end-users. Unredacted, unsensitive screen data provides context to application stakeholders that may help focus their discussion.

Image-based text redaction consists of two principal steps: finding text in an image, also known as text segmentation, and recoloring segmented image regions to "remove" text. (We note that such segmentation is also the first step of optical character recognition.) Redacting images using this approach ensures that no "hidden" text or other data exists within the final redacted product (as often plagues redaction in standard office document formats).

For automatic text redaction, we explored two approaches: Canny Edge Detection [12], which aims to bound text with boxes, and Gabor-wavelet [13] filtering, which aims to classify individual pixels as "text" or "non-text." For Gabor, we looked at both unsupervised classification and supervised classification [14].

3.1. Canny Edge Detection

In order to be legible, screenshot text exists with an intensity contrast in relation to its background and thus creates gradient high points. The Canny approach analyzes an image's intensity gradient and marks edges at gradient high points—thus (in theory) segmenting screenshot text.

First, we convert a color screenshot to 8-bit gray scale. We then apply a Gaussian blur using a $3x3$ window to reduce image noise—Canny output qualitatively contained less noise with this initial blurring step. Next, we executed Canny using low and high threshold values of 100 and 300 respectively to find edges—the values provide qualitatively-reasonable redaction results for a variety of desktop screenshots. Gradient magnitudes greater than the high threshold are considered edges and traced throughout the image. Values above the low threshold denote edges that branch from an existing trace process. Together, these tunable values reduce noise during edge detection.

After executing the Canny algorithm, we find connected components (polygons) using Canny output and an algorithm suitable for doing so [15]. For each polygon discovered, we compute a bounding rectangle and draw a filled version of the rectangle into an image "redaction mask." Finally, the redaction mask is applied to the original image to produce a redacted image.

Figure 2 shows examples from our prototype. Unfortunately, standard practice in commercial EMR prevents customers from disclosing user interface details (e.g., see [16]), so we cannot actually show the screenshots we used in our experiments.

Figure 2: Canny-based text redaction. The top image is a screenshot snippet from the Wikipedia page about Canny edge detection [17] (recall that contractual obligations prevent us from disclosing the EMR screenshots we tested on). The second image depicts the rectangles that result from processing the first image with Canny edge detection, polygon detection, and polygon bounding with rectangles. The third image derives from filling the rectangles in the second image and then applying the second image as a redaction mask to the first. Canny missed some true edges throughout the image (false negatives for edge detection) and added edges where text does not exist near the globe (false positive for text detection). Finally, notice whitespace between words and tiny rectangles enclosed within larger ones.

3.2. Gabor Filters

In general, a *wavelet* is a wave with some orientation and frequency that when convolved with an image, resonates and creates a detectable signal.

Gabor wavelets, which are commonly used in image processing, are comprised of a sine wave modulated by a Gaussian envelope; for our application, they use a two-dimensional envelope. Both real and imaginary components comprise the wavelet, but we follow the model of Jain and Bhattarchee [14] and only use the real, symmetric (cosine) component. When an individual filter is convolved with an image, our system extrapolates border pixels to increase the image size and prevent the filter from "falling off" the image edge (other extrapolation approaches failed in our experiments).

Using a bank of filters enables detection of image features of different frequencies and orientations. In the wavelets we used in our application, we considered five standard deviations of the Gaussian (again following Jain and Bhattarchee). This left us two tunable parameters for wavelet functions: wavelength (λ) and orientation (θ). Through qualitative analysis, we settled on parameters

$$\lambda \in \{.5, 1.0, 2.0, 4.0, 8.0, 16.0, 32.0\}$$

and

$$\theta \in \{0.0, 45.0, 90.0, 135.0\}$$

for a filter-bank size of 28 filters ($|\lambda| \times |\theta|$).

Like Jain and Bhattarchee, we vary the parameter θ to detect signals oriented in a uniform variety of positions. However, unlike them, we chose λ to vary by powers of 2 in order to form a dyadic collection of filters that span a collection of feature sizes. Through qualitative experiments, we found our chosen values to detect features among a collection of screenshots.

Feature Vectors. We apply a Gabor wavelet filter by first convolving the image with this wavelet function.

If we have a bank of n filters, we then have n filtered images, yielding (after thresholding) an n-dimensional vector for each pixel in the image. We then append each pixel's x and y position to each vector, and shift each vector to zero mean and unit standard deviation. Thus, applying a bank of n filters yields an $n + 2$-dimensional *feature vector* for each pixel.

3.3. Classification

Once we've used our bank of Gabor filters to turn each pixel into a feature vector, we then need to determine which vectors represent text pixels and which represent non-text.

Unsupervised Classification. In our first approach, we use the *k-means algorithm* [18] to cluster features into k classes, where $k \in \{2, 3\}$. The algorithm assigns each pixel a class label $i \in [0, k-1]$, where one class may correspond to text if text exists. (In our qualitative experiments, we found that some screenshots clustered better visually into $k = 2$ classes and others into $k = 3$ classes.)

During k-means clustering, the system relied on stopping conditions of the first of 10000 iterations or an error rate of .0001. We chose the initial cluster centers using a more recent technique [19] and ran the algorithm one time to the stopping conditions before assigning labels. After running k-means, the label i corresponding to text must be chosen manually. The designated "text" pixels form a mask that redacts text when combined with the original image.

Supervised Classification. The downside to unsupervised classification is multifold: k and i are chosen manually; the approach classifies pixels into k clusters

whether or not text exists; and k-means clustering can be slow (particularly with a feature count easily surpassing one million with modern screen resolutions).

To address these issues, we also tried *supervised classification*. Instead of using k-means, we feed each feature vector to a trained classifier that labels the pixel as "text" or "not text." All pixels labeled as "text" are converted to the color black; all other pixels maintain their values.

We chose a linear *support vector machine (SVM)* to label pixels as members of classes $\{-1, 1\}$.

We experimented with two classifiers (a) L1-regularized L2-loss support vector classification and (b) L1-regularized logistic regression. We chose these classifiers because after training, they can contain a 0-valued parameter for each feature that remained unused during the training process. Such features can be eliminated from input during future predictions and thus not computed in the first place. Their absence reduces computational overhead in the running system. (Interestingly, in our tests with EMR screenshots, only one feature was not used).

To begin machine learning, we first partition our set of screenshots into a training set and testing set. Then to train the classifier, we generate a set of ground-truth feature vectors and labels from the training set. We generate ground-truth by manually choosing the features and labels associated with "best" redaction results using the unsupervised classification technique described above. This ground-truth is fed into a program we implemented that interfaces with the liblinear library [5] to train and save the resultant classifier. The classifier can then be run on any image using another program

we wrote to classify pixels as $\{-1, 1\}$ and thus redact text.

During the SVM training process, we used default liblinear values for all SVM parameters. We experimented with cross-validation to tune the constant C in the SVM expression (see liblinear for details [5]). However, we experienced minimal performance improvements and therefore relied on default values to train each classifier.

4. Experimental Tools

Section 3 above described our approaches to automatic text redaction. However, for both EMR clinicians as well as system experimenters, it's important to keep users in the loop. This present section describes two tools we built for this purpose.

4.1. Tool: scrubs

Our *scrubs* tool (Figure 3) captures and redacts screenshot images dynamically, in real time, uaing Canny. Our prototype uses x11vnc [10], pthreads [20] and the RFB protocol [9].

When an EMR user decides some sequence of activity should be logged for later analysis, it's possible that automatic redaction may remove too much information (such as non-sensitive text that would help illuminate the issue requiring analysis) or too little (such as a sensitive logo or image). Consequently, our scrubs tool also provides an edit mode, which pauses display of screen updates and allows the user to click and drag the mouse to define custom redaction and/or unredaction rectangles (Figure 4). While paused for user edits, the system continually processes and maintains received screen

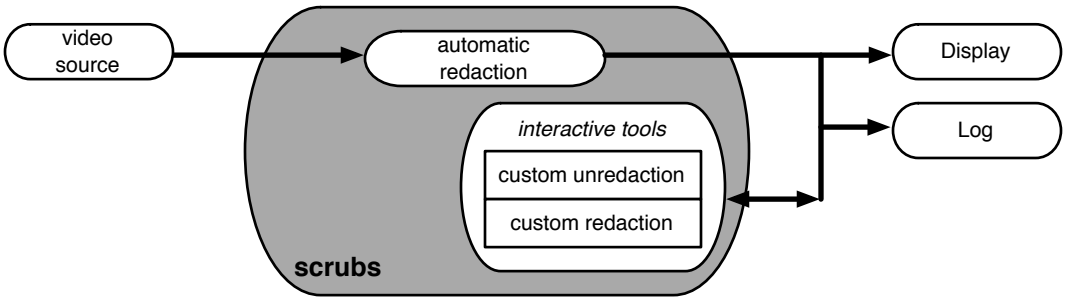

Figure 3: Our *scrubs* tool enables automatic redaction, interactively tunable, in real time.

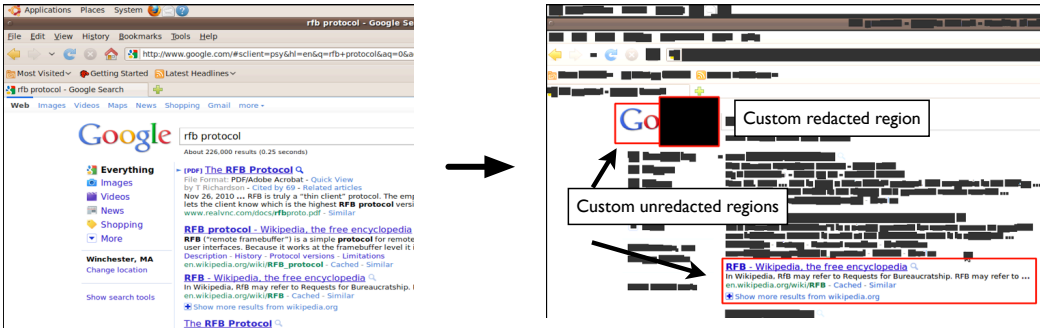

Figure 4: Our scrubs tool automatically applies redaction, and then permits custom redaction and unredaction.

updates in the background, and upon returning to record mode, the system displays a compilation of all updates processed during pause.

4.2. Tool: five_in_one

The Canny Edge approach to text redaction (Section 3.1) overlays a screenshot image with rectangles marking regions of potential text. In our experiments on EMR screenshots, we found that the resulting set of rectangles could often benefit from additional massaging. Thus, for the purpose of exploration and for end-user use, we developed a tool called "five_in_one" (Figure 5). This tool permits a wide range of interactive operations, in-

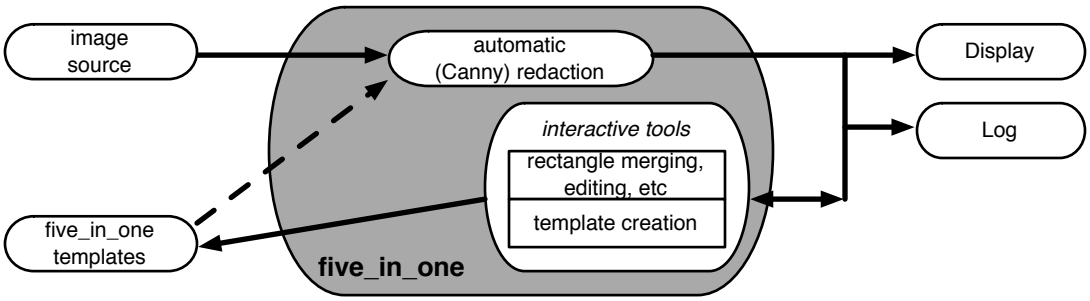

Figure 5: Our *five_in_one* tool enables experiments and end users to analyze and edit the rectangular regions of potential text flagged by Canny redaction, and to generate and apply templates.

cluding merging, copying and deleting rectangles; toggling display between transparent and solid rectangles; generating (and then automatically applying) redaction templates; overlaying with a grid; and thinning out redundant rectangles. Figure 6 shows one example.

5. Evaluation

To evaluate our system, we looked at the relative effectiveness of the two approaches to automatic redaction (Section 5.1 and Section 5.2) and at non-text aspects of privacy preservation (Section 5.3). We also looked at effectiveness of context preservation (Section 5.4), as well as basic computational costs (Section 5.5).

For our empirical analysis, we used a corpus of 80 screenshots from EMR systems at two large healthcare providers. As we noted earlier, although the datasets contain fake patient data, the donor organizations still considered the details sensitive, so we cannot publicly show them. In one dataset, images were in PNG format, RGB color, and were approximately 1500×1900 pixels

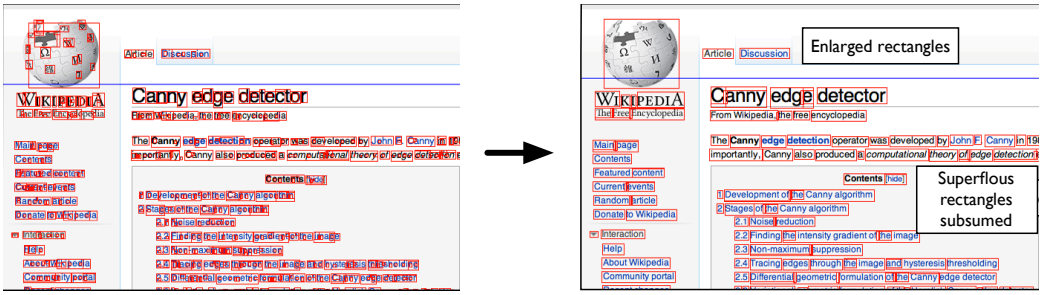

Figure 6: As one example of five_in_one editing, a user can clean up the redaction rectangles produced by Canny (left) by thinning out superfluous rectangles (in this case, reducing the count about 75% from 969 to 237) and then enlarging the remaining rectangles one pixel at a time.

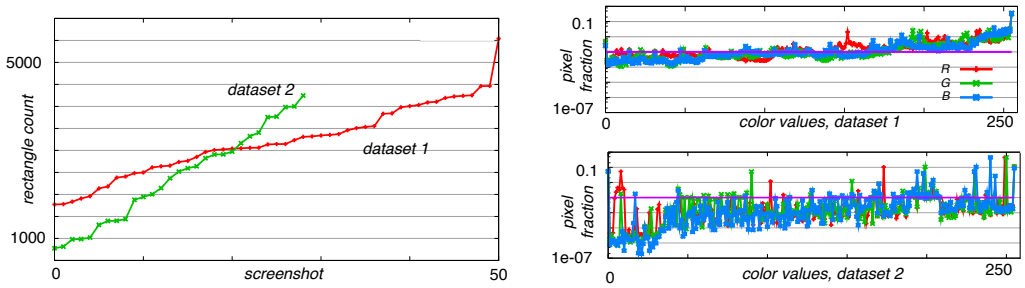

Figure 7: Measures of the visual complexity of our EMR datasets: Canny redaction rectangles, sorted (left); normalized color variety (right).

and 1-1.5 MB each. In another, images were also PNG and RGB, but 1680 × 1080 pixels and 230-390 KB. Due to their sensitive nature, we stored the corpus in an AES-encrypted disk volume.

Figure 7 illustrates the complexity of the datasets according to the number of redaction rectangles generated for each screenshot.

5.1. Canny

Our testing showed that Canny-based text redaction requires improvements before the system can apply it meaningfully to EHR datasets.

The Canny approach had several problems. It generated redaction rectangles that cover large parts of the screen, reduce potentially useful, non-private screenshot context. (Occasionally, Canny even redacted the entire screen!) The Canny approach also sometimes found interior edges of letters such as "p" which produce very small rectangles embedded in larger ones. Canny left whitespace between words, which may enable word-based frequency analysis that reveals redacted text. Canny also tended to miss some text (false negatives) while redacting some non-text (false positives). Figure 2 shows examples of whitespace and false positive issues; Figure 8 shows false negative and spuriously large rectangle issues.

Our analysis did suggest ways that Canny redaction could be tuned to be usable for this application. We can eliminate rectangles that cover all or most of the screen, and (as Figure 9 shows) rectangles that contain a large number of the other rectangles. As Figure 10 shows, we can also identify (and then merge) rectangles that are close enough vertically to be considered on the same "text line," but whose horizontal gap is small enough to be considered whitespace.

As Figure 6 noted, even manual editing with our five_in_one tool could reduce the rectangle count by 75%. Reducing rectangle count can reduce the latency of subsequent processing steps that involve all Canny rectangles, such as rendering rectangles in an image or analyzing and merging adjacent words.

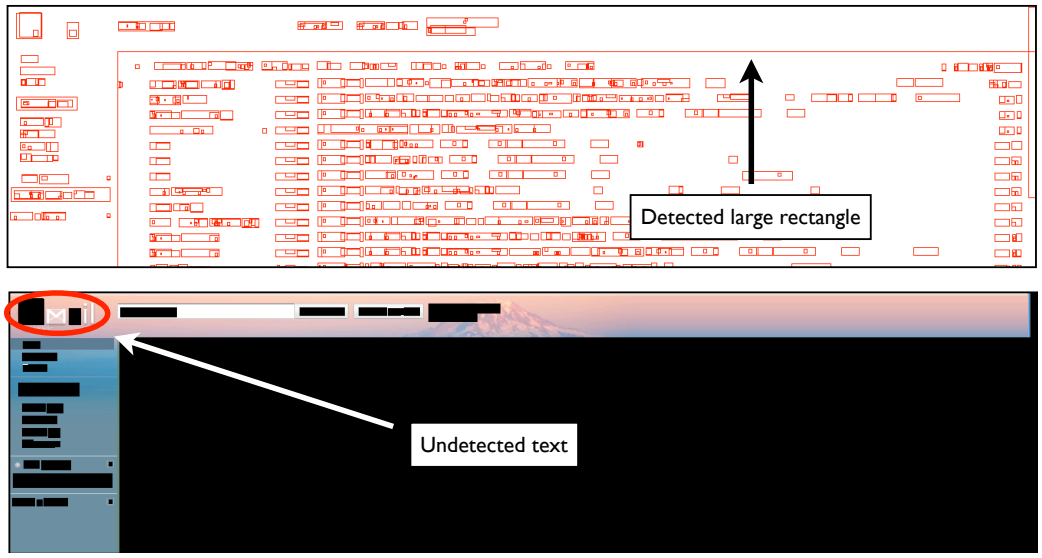

Figure 8: Canny redaction on a gmail inbox shows both spurious large redaction rectangles as well as missed text.

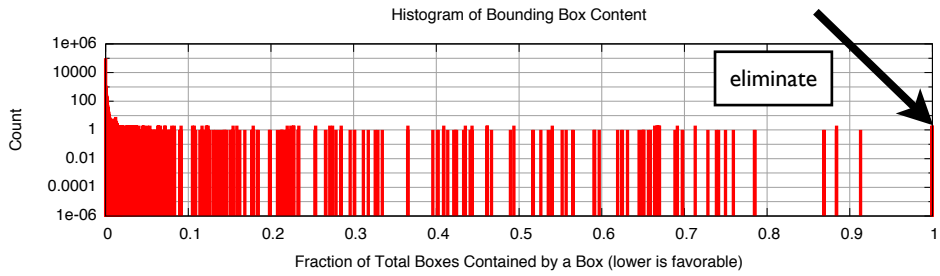

Figure 9: Histogram of the fraction of total redaction rectangles in a screenshot contained wholly within a given redaction rectangle.

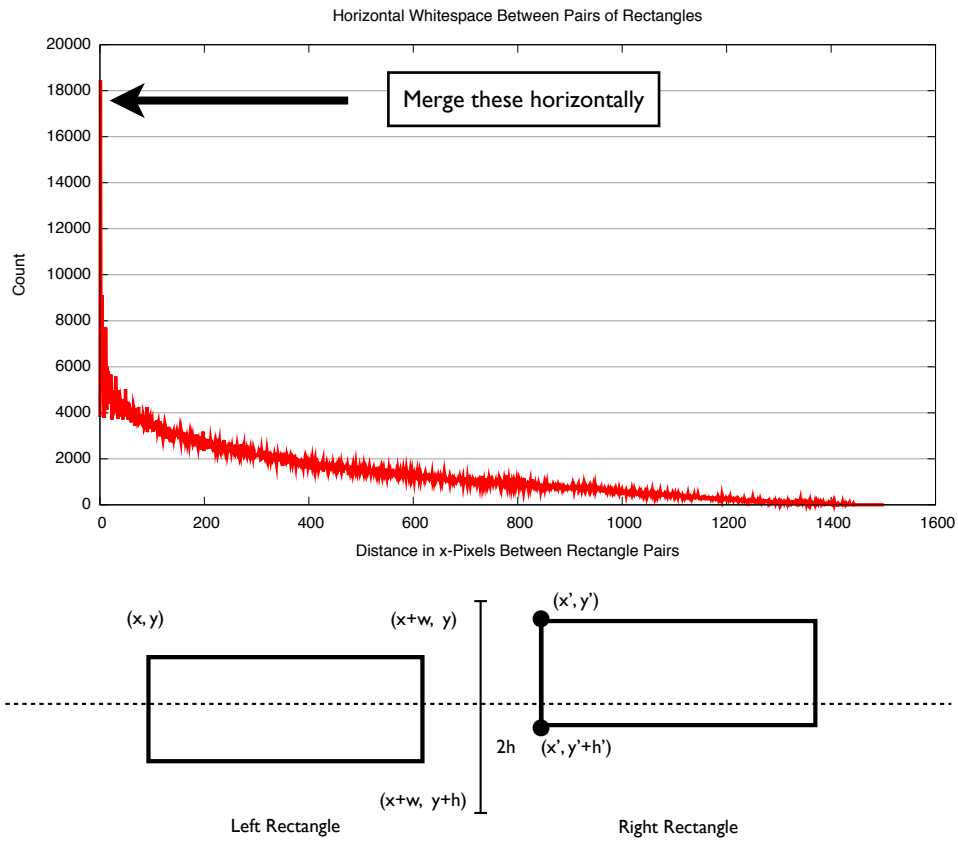

Figure 10: Histogram of horizontal distances between pairs of rectangles heuristically on the same text line. If $(x + w) \leq x'$, $(y - \frac{h}{2}) \leq y' \leq (y + \frac{h}{2})$, and $y + \frac{h}{2} \leq (y' + h') \leq y + \frac{3}{2}h$, then we includes the value $x' - (x + w)$ in the histogram.

5.2. Gabor

Visually, Gabor-filtering redacts more precisely than Canny-based filtering. Unlike the Canny-based approach, Gabor fills whitespace between words and redacts fractional characters. Gabor also redacts fewer non-text objects (such as the Wikipedia globe, in a non-EMR example we showed) and did not erroneously redact large rectangles from the screenshot, as Canny-based redaction did.

Qualitative Analysis. Figure 11 revisits Figure 2 using Gabor-based redaction where $k = 2$ and $i = 0$ and Figure 12 revisits Figure 8 using $k = 2$ and $i = 1$. In Figure 11, note how Gabor-based redaction fills whitespace between words in sentences, does not redact objects such as the globe, and does redact fractional characters. It does not redact large rectangles from the screen as Canny-based redaction. In Figure 12, note how the system failed to redact text with certain font scales and textures in the upper left corner and also throughout lighter message-body in the message lines.

Quantitative Analysis for Unsupervised. To evaluate unsupervised Gabor redaction, we chose a few representative but dissimilar (using the metric we present below, in Section 5.4) screenshots from dataset 1. For each screenshot, we chose the unsupervised redaction that looked best qualitatively, and then manually counted the text characters missed by redaction. In these screenshots, the character counts ranged from 1094 to 2145. False negatives (characters entirely unredacted) ranged from 0.036% to 2.7%; partial false negatives (characters with at least one pixel left unredacted) ranged from 1.7% to 4.3%. We did not count false positives because they represent non-characters, and would require counting pixels to be meaningful.

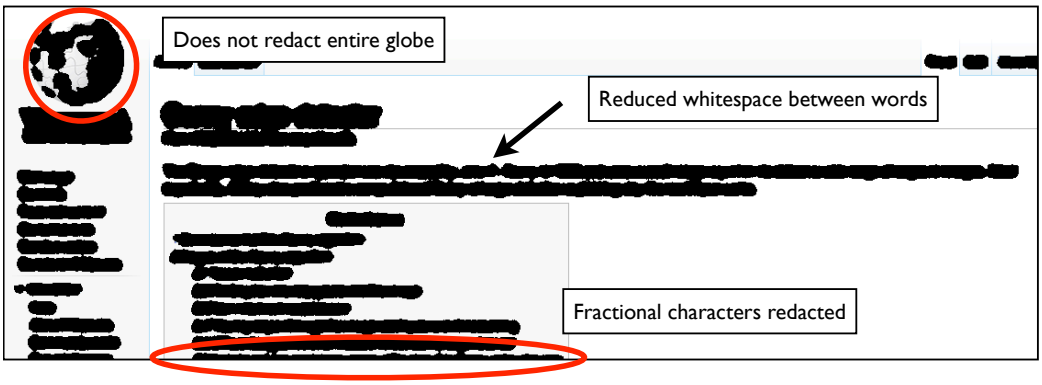

Figure 11: Gabor-based redaction does not redact large objects such as the globe, connects whitespace between words in sentences, and redacts fractional characters found at the edge of the screenshot.

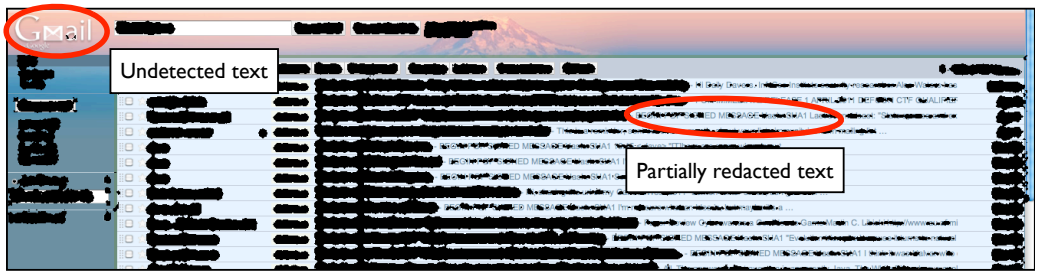

Figure 12: Gabor-based redaction missed the large gmail text and small "by Google" text below the gmail text. It partially redacted lighter message text in the inbox. Note that Gabor-based redaction did not compute a large rectangle of false positives as Canny did on the same image.

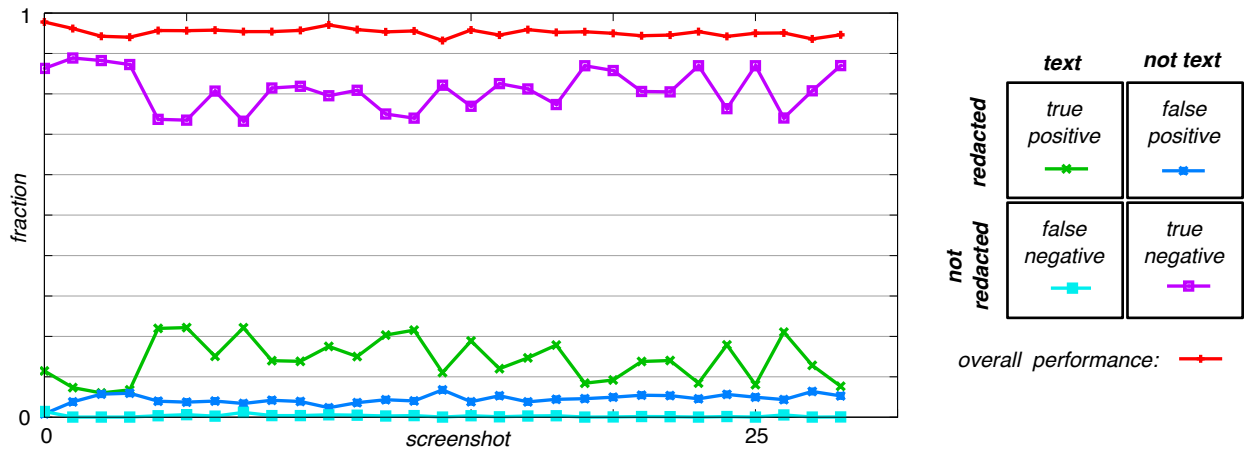

Figure 13: Effectiveness Gabor-based text redaction. We trained a liblinear L1-regularized logistic regression classifier on on dataset 1 and applied it to dataset 2; in both cases, we used the qualitatively best unsupervised Gabor redaction as "ground truth" labeling.

Quantitative Analysis for Supervised. As Section 3.3 above discussed, we started by running unsupervised Gabor on dataset 1 and, for each screenshot, choosing the qualitatively best result as "ground truth," and used these labels to train the L1-regularized logistic regression classifier on dataset 1. To evaluate supervised Gabor classification, we then applied this trained classifier to dataset 2, and compared the results against the "ground truth" obtained by running our unsupervised Gabor variations on each screenshot and qualitatively choosing the best one.

The mean classification performance is 95.2% with a stddev of .953% and a minimum performance value of 93.2%—*larger minima are better than smaller ones.* The mean false-negative rate is .307% with a stddev of .338% and a maximum value of 1.4%—*smaller maxima are better than larger ones.*

5.3. Non-Text Information Leakage

Our experiments also revealed ways in which redacted EMR screenshots still revealed possibly sensitive information. The positioning of redacted text within a page can betray information, as can similarities and differences between successive lines of redacted text. A tick-box with a redacted "checkmark" is still distinguishable from unchecked box; visual "alerts" such as red exclamation points or yellow-highlighted text also convey potentially sensitive information.

To address these concerns, we explored techniques to normalize redaction rectangles against a background grid, to identify and redact specific icon templates, and to identify and redact specific colors (e.g., red). The first author's thesis [21] has more information.

5.4. Context Preservation

As the introduction noted, traditional approaches to privacy and confidentiality seek to hide information. However, for our EMR capture tool to be useful, we also need to preserve information: the context of the EMR screenshot involved.

To quantitatively evaluate how well our techniques work at preserving context, we looked at two ways of measuring differentiating information between pairs of redacted screenshots.

In the first approach, we measured the fraction of overlapping text-redacted pixels in an image pair. With this metric, changes accumulate only when a pair of pixels exist, at least one pixel of the pair begins non-black, and both pixels are redacted. When one or both pixels begin non-black and the pixels correspond to text, redaction removes differentiating information

by converting both values to black. Removing information reduces differentiating screenshot context. Taken to the limit, redaction blackens each screenshot entirely and leaves no differentiating information.

Looking at all pairs of screenshots in dataset 1, the mean fraction of overlapping, redacted text is 9.3% with a standard deviation of 3.5%; for pairs of identical screenshots, 23.7% and 3.6%. Redaction preserves 90% of differentiating information in all pairs and 76% in pairs of identical screenshots—on average, redaction affects no more than 24% of the pixels in any screenshot.

In our second approach, we computed a distance between two screenshots by counting the number of pixels that match within the pair. Because EHR screenshots are nearly identical in size and aligned in content (e.g., items such as menus are not pixel-shifted among screenshots) this measurement gives a notion of similarity that enables useful pairwise-screenshot comparisons (as we qualitatively validated). Figure 14 shows the results of text redaction in similarity of over 1275 screenshot pairs of dataset 1 (we excluded pairs of identical screenshots). Overall, redaction has little impact on pairwise screenshot similarity with changes ranging from 2% to 15%. Text redaction retains potentially important context in the EMR screenshots.

5.5. Latency

The principal computational component of our system consisted of text redaction.

To measure latency of text redaction, we used a a MacBook Pro running Mac OS X 10.7 with 8 GB of memory serves as the experimental platform. An AES-256-encrypted disk image stores image, feature, and label files associated with redaction. To obtain timing information, we used dtrace and

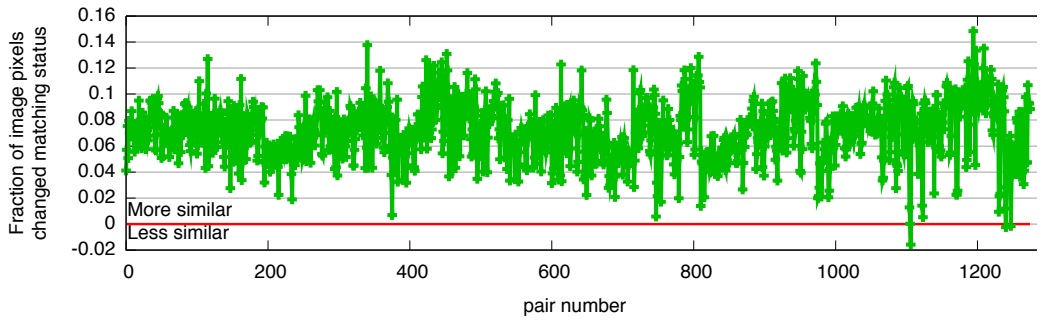

Figure 14: Effect of text redaction on similarity of pairs of distinct screenshots in dataset 1. Fractions that fall above the horizontal line correspond to screenshots that are more similar after redaction.

	mean	stddev
Canny	0.096	0.009
Unsup. Gabor	2.077	0.722
Sup. Gabor	2.077	0.017

latency to classify pixels (seconds)

	mean	stddev
Set up	0.365	0.004
Build	13.12	0.181
Normalize	4.569	0.033

latency to generate elements for 28 Gabor filters (seconds)

	mean	stddev
Features	5.987	0.407
Labels	0.226	0.069

latency to load Gabor elements from file on disk (seconds)

Table 1: Our measured costs of redaction.

programmatically printed timing information. All file loads were measured using a cold file cache. Table 1 shows the results.

6. Improving Real-World Systems

We designed our system within the context of a larger vision: a privacy-protected "logging Service" that interacts with a monitored system, a "sharing Service" that functions as a repository for application stakeholders to share redacted screenshots, a monitored system with developers and maintainers, and an end-user who wishes their system to be monitored and a

web-browser through which the end-user can interact with logging and sharing services. In a sample usage scenario, an end-user triggers the "Logging Service" to log a monitored host; the "Logging Service" connects to the monitored host and begins logging with automatic redaction; when the end-user detects a scenario she wishes to bring to the attention of the system engineers, she reviews, edits, and possibly unredacts the logged screenshot and then shares it with the system engineers.

In our long-term vision, the fruit of this work can enable empirical feedback paths between application stakeholders. Developers, administrators, end-users, organizations that produce or deploy a particular system, and legislators or governance bodies that create rules to govern systems all represent different types of stakeholders. With established feedback paths, such stakeholders can begin to understand empirically the day-to-day, system-effects of their decisions.

A simple capture system can also provide direct value to end-users by endowing them with a larger, empirical role in the software maintenance cycle. They can capture, annotate, and share problems, configurations, ideas, bugs, and other captured scenarios with stakeholders. They can inform existing *ad hoc* stakeholder interactions such as online support forums and help-desk interactions with rich, contextual data. Additionally, end-users can use traces as visual web search keys during their own investigations.

Playing a larger role in the software maintenance cycle can motivate end-users to share their findings continually: if end-users believe and experience that their contributions make a positive difference to their workflow, end-users may be motivated to contribute further. Consequently, organizations

may improve empirical insight into their information security systems and associated risk calculations. When organizations lack the expertise to analyze traces in-house, they could hire third parties to do so.

7. Related Work

Our work combines existing technologies of screen capture and computer vision with a goal of improving the quality of communications among application stakeholders and ultimately, improving our understanding of "usable security." Many research and commercial products implement pieces of our work in isolation and for different purposes.

Screen Capture. The MIT Sikuli research project combines computer vision and programming to enable users to create machine-independent, visually-programmed and actuated programs [22]. A commercial product call eggPlant also allows developers to test GUIs with machine-independent, automation scripts [23]. Many screen capture applications such as Snipping Tool [24], Snapz Pro X [25], and xwd [26] exist. Some programs capture still screenshots, others capture both stills and video, and some allow end-users to annotate captures.

Our system captures data and modifies it with text redaction—we use screen captures for a different purpose than these works.

Segmenting Text. Many commercial and free-software tools such as Gimp [27], Photoshop [28], Aperture [29], Final Cut Studio [30], Pixelmator [31], and Imagemagick [32] allow one to paint, create, touch up, and modify still images and/or video. These applications could be used to manually redact text from a screenshot.

Our work builds on existing text segmentation research [14] to redact text automatically from screenshots.

Google Goggles can extract and recognize text from natural scenery for purposes such as language translation among many others [33]. The scope of our system is limited to computer screenshots. However, screenshots taken with a camera may include angles and lighting similar to the natural scenery submitted to Google Goggles.

User Studies. Google's in-house UseTube [34] supports employees who wish to perform user studies of any network-connected computer; it simplifies the act of performing, archiving, and accessing user studies.

Deidentifying Data. In the medical domain, a large body of work relates to deidentifying protected health information (PHI) in electronic documents once it is already in text format [35, 36]. Our work approaches the deidentification problem from a complementary angle. Our system does not interpret data; rather, it redacts all text within an application screenshot and allows a domain expert to unredact portions relevant to their needs.

Document redaction products such as Rapid Redact [37] and brava! [38] exist in the commercial marketplace. These products parse document structure and can help users achieve WYSIWYG. In contrast, our system redacts material from images directly (there is typically no hidden structure).

Anonymity. Deidentified data records can still contain visual information that reveals sensitive data. Research in the context of databases that contain a mix of sensitive and unsensitive records explored the concepts k-anonymity [39] and l-diversity [40].

In some circumstances, redacted text in our system may suffer from a visual form of the k-anonymity problem; these techniques may apply in our setting.

Document Analysis. The International Conference on Document Analysis and Recognition (ICDAR) has many papers and competitions related to the problem applying machine learning and computer vision to analyze documents analysis [41]. A 2003 competition sponsored by ICDAR has datasets available for optical character recognition (OCR), word recognition, text locating, and other purposes [42]. These datasets do not apply directly to our problem; we segment text, in some cases have a more constrained segmentation problem, and do not apply OCR.

8. Conclusion

Effective usability engineering in any system requires closing the loop, so users can easily identify and communicate troublesome scenarios. In an interactive electronic system, a natural way to do this is via screenshots. In EMR and EHR, privacy concerns require that any such screenshots have sensitive data redacted and the logistics of certified commercial medical IT require that any solution not touch the internals of the software.

To address these concerrns, we have designed, built, described, and empirically analyzed a system that allows end-users to take screen captures on sensitive systems. The system automatically redacts screenshot text and allows end-users to fine-tune redacted results for their needs. The automated redaction process requires no end-user intervention. We evaluated our system using a corpus of screenshots from EMR systems at two large medical

facilities.

Potential areas for future work include improving Canny for general-purpose use, implementing predicate matching to process screenshots according to logical conditions, building a larger ground-truth data corpus, building system components for sharing redacted screenshots, and deploying the system in a real user environment.

Ultimately, our redaction system can facilitate data-driven communications among application stakeholders and guide system evolution to address stakeholder needs. With accurate and timely tuning enabled by our work, stakeholders can achieve and maintain usable and secure systems in practice.

Acknowledgments

This work is based on the first author's MS thesis work [21].

The authors are also grateful to Andrew Gettinger, David Hanauer, Ross Koppel, and Lorenzo Torresani and for their helpful advice and assistance.

References

[1] A. Odlyzko, Providing Security With Insecure Systems, in: Workshop on the Economics of Information Security (WEIS), 2010.

[2] The boost Community, boost C++ Libraries, `http://www.boost.org/` (April 2011).

[3] ISO/IEC 14882:2003: Programming Languages: C++, 2003.

[4] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[6] CGAL, Computational Geometry Algorithms Library, `http://www.cgal.org`.

[7] A. Zomorodian, H. Edelsbrunner, Fast software for box intersections, in: Proceedings of the Sixteenth Annual Symposium on Computational Geometry, SCG '00, ACM, New York, NY, USA, 2000, pp. 129–138.

[8] L. Kettner, A. Meyer, A. Zomorodian, Intersecting Sequences of dD Iso-oriented Boxes, in: CGAL User and Reference Manual, 3.7 Edition, CGAL Editorial Board, 2010.

[9] T. Richardson, The RFB Protocol, RealVNC, Ltd. (3.8).

[10] Karl Runge, x11vnc: a VNC server for real X displays, `http://www.karlrunge.com/x11vnc/` (January 2011).

[11] VMware, VMware, `http://www.vmware.com` (April 2011).

[12] J. Canny, A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI–8 (6) (1986) 679–698.

[13] D. Gabor, Theory of communication. Part 1: The analysis of information, Journal of the Institution of Electrical Engineers – Part III: Radio and Communication Engineering 93 (26) (1946) 429 –441.

[14] A. Jain, S. Bhattacharjee, Text segmentation using gabor filters for automatic document processing, Machine Vision and Applications 5 (1992) 169–184.

[15] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing 30 (1) (1985) 32–46.

[16] R. Koppel, D. Kreda, Health Care Information Technology Vendors' "Hold Harmless" Clause: Implications for Patients and Clinicians, Journal of the American Medical Association 301 (12) (2009) 1276–1279.

[17] Wikipedia, Canny edge detector, `http://en.wikipedia.org/wiki/Canny_edge_detector` (April 2011).

[18] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129 – 137.

[19] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Technical Report 2006-13, Stanford InfoLab (June 2006).

[20] The IEEE and The Open Group, The Open Group Base Specifications Issue 6 – IEEE Std 1003.1, 2004 Edition, IEEE, New York, NY, USA, 2004.

[21] J. A. Cooley, Screen Capture for Sensitive Systems, Computer Science Technical Report (M.S. Thesis) 2011-690, Dartmouth College (May 2011).

[22] T. Yeh, T.-H. Chang, R. C. Miller, Sikuli: using GUI screenshots for search and automation, in: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, UIST '09, ACM, New York, NY, USA, 2009, pp. 183–192.

[23] TestPlant, eggPlant, http://www.testplant.com/ (January 2011).

[24] Microsoft, Snipping Tool, http://windows.microsoft.com (January 2011).

[25] Ambrosia Software Inc., Snapz Pro X, http://www.ambrosiasw.com/utilities/snapzprox/ (January 2011).

[26] Community Supported, xwd, http://www.xfree86.org/ (January 2011).

[27] The GNU Project, GIMP: The GNU Image Manipulation Program, http://www.gimp.org/ (January 2011).

[28] Adobe, Photoshop, http://www.adobe.com/products/photoshop/photoshop/whatisphotoshop/ (January 2011).

[29] Apple, Aperture, http://www.apple.com/aperture/ (January 2011).

[30] Apple, Final Cut Studio, http://www.apple.com/finalcutstudio/ (January 2011).

[31] Pixelmator, Pixelmator, http://www.pixelmator.com/ (January 2011).

[32] Community Supported, ImageMagick, http://www.imagemagick.org/script/index.php (January 2011).

[33] Google, Google Goggles, http://www.google.com/mobile/goggles/ (January 2011).

[34] M. LaRosa, D. Poole, R. Schusteritsch, Designing and Deploying UseTube, Google's Global User Experience Observation and Recording System., in: CHI '09: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, ACM, New York, NY, USA, 2009, pp. 2971–2986.

[35] S. Meystre, F. Friedlin, B. South, S. Shen, M. Samore, Automatic de-identification of textual documents in the electronic health record: a review of recent research, BMC Medical Research Methodology 10 (1) (2010) 70.

[36] J. Aberdeen, S. Bayer, R. Yeniterzi, B. Wellner, C. Clark, D. Hanauer, B. Malin, L. Hirschman, The MITRE Identification Scrubber Toolkit: Design, training, and assessment, International Journal of Medical Informatics 79 (12) (2010) 849–859.

[37] RapidRedact, RapidRedact, http://www.rapidredact.com/ (January 2011).

[38] Informative Graphics Corporation, brava!, http://www.infograph.com/company.asp (January 2011).

[39] L. Sweeney, k-anonymity: A model for protecting privacy, International

Journal of Uncertainty Fuzziness and Knowledge Based Systems 10 (5) (2002) 557–570.

[40] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, ACM Trans. Knowl. Discov. Data 1.

[41] ICDAR, `http://www.icdar2011.org/EN/volumn/home.shtml` (April 2011).

[42] ICDAR Dataset, `http://algoval.essex.ac.uk/icdar/Datasets.html` (April 2003).