Journal of Biomedical Informatics 46 (2013) 721–733

Contents lists available at SciVerse ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Privacy-preserving screen capture: Towards closing the loop for health IT usability ☆

Joseph Cooley [1], Sean Smith *

Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA

ARTICLE INFO

Article history:
Received 11 July 2012
Accepted 27 May 2013
Available online 13 June 2013

Keywords:
Health IT
Security
Privacy
Usability
Redaction

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 health IT systems are typically replete with privacy-sensitive data regarding patients, diagnoses, clinicians, and EMR user interface details; instrumentation for screen capture (capturing and recording the scenario depicted on the screen) needs to respect these privacy constraints. Furthermore, real-world health IT systems are typically composed of modules from many sources, mission-critical and often closed-source; any instrumentation for screen capture can rely neither on access to structured output nor access to software internals.

In this paper, we present a tool to help solve this problem: 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-independent and OS-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 health IT deployments.

1. Introduction

Medical enterprises large and small are replacing 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"; but to do so by understanding and tuning a system requires measuring it, in order for this tuning to be a data-driven process.

However, when it comes to health IT systems, even the process of just taking such measurements raises a combination of challenges:

Privacy Preservation In a human-facing IT system, *screenshots* comprise the natural domain for measurement. However, in health IT, 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 health IT user interfaces 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 mechanisms in place to ensure that private data is never leaked throughout the analysis process, or by redacting it 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 health IT 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 of workflow process context. Our redaction system needs to be effective 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.

☆ 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. Tel.: +1 603 646 1618.
E-mail addresses: joe.cooley@gmail.com (J. Cooley), sws@cs.dartmouth.edu (S. Smith).
URL: http://www.cs.dartmouth.edu/~sws/ (S. Smith).
[1] Current address: MIT Lincoln Laboratory, Lexington, MA 02421, USA.

1532-0464/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jbi.2013.05.007

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 (depending on the system) invalidate necessary certification.

Workflow Impact For clinicians using health 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 and delay experienced by these users: it should work automatically and quickly; users should be able to quickly log some issue and move on with their real mission.

This Paper. This paper reports on our research addressing this aspect of "closing the loop": designing, prototyping, and evaluating technology to enable redaction of privacy-sensitive elements in medical IT (privacy preservation), while preserving usage context and permitting interactively selected unredaction (context preservation), passively acting on raw screen data alone (system impact), and operating automatically in real time (workflow impact). Drawing on algorithmic techniques that have been used for various aspects of image processing, our work is, to the best of our knowledge, the first to explore these techniques in the context of automatic redaction of privacy information in health IT (or other domains).

Section 2 motivates this work in the context of closing the loop for health IT usability research and development. Section 3 provides an overview of our prototype system. Section 4 describes our methodologies for text redaction. Section 5 describes the broader system we built around these techniques. Section 6 evaluates the effectiveness of our approaches. Section 7 presents how this work can impact real-world health IT systems. Section 8 reviews related work, and Section 9 concludes.

2. Health IT motivation

In concurrent work [1], our lab has been cataloging many ways in which clinicians report that health IT systems lead to usability frustration. We list just a few examples:

- The age field for a patient does not allow fine enough units to correctly determine medication for newborns—or allow a way to indicate age of patients still in utero.
- A health IT screen with field-defined data did not allow an experienced clinician to record "smell of breath."
- A drop-down menu did not permit easy discovery of the proper diagnosis, leading the clinician to pick a wrong one that was at least "close."
- The interface to refer a patient to a stomach cancer specialist requires the non-specialist clinician to first identify which of 52 varieties of stomach cancer the patient has.
- A health IT system gives each pending lab result an identical title, "Lab Result."
- A medication administration system does not recognize that an order for "10mgs" of a medication can be fulfilled by two 5 mg tablets.
- An EMR screen gives three ways to exit—whose consequences differ substantially but which appear the same to the user.

Each such scenario demonstrates the need for "closing the loop." The very existence of this litany of problems indicates that the learned experts who built these health IT systems, even with the best of foresight and stakeholder input, fail to anticipate usability problems that arise in the field. Furthermore, each such problem is likely solvable. For some; small software changes likely suffice; others may require experimental evaluation of proposed solutions or research into new techniques.

However, before developers can try these small changes or external researchers can explore new techniques, they need to know about these problems. Unfortunately, current state of the art does not permit the frustrated health IT user to easily capture and document these problems. Photographing or printing the screens violates privacy (both of patients and of providers). Instrumenting the internals of the health IT software requires access to code that is usually proprietary—and then requires modifying mission-critical systems. Often, researchers (such as the first author) cannot even look at usage logs of the system, due to restrictions of the university IRB. Just recording oral complaints does not permit easy reproduction of scenarios—and (as the second author found) other clinicians at the same institution may even disbelieve the scenario ever even existed.

A system such as ours—which empowers end-users to record and annotate screenshots automatically redacted of sensitive information without changing the internals of the health IT system being passively monitored—would provide a key component in a solution to systematically alleviate these usability frustrations. We revisit this vision in Section 7.2.

3. System overview

Again, in this work we explore a core problem in closing the loop: screencapture of health IT meeting the privacy and logistical requirements outlined in Section 1. Our prototype system applies text and image redaction to KVM feeds from health IT systems—see Fig. 1.

Our system includes functionality essential to implementing screen capture for sensitive health IT 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 [6–8]. 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 nutshell, 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

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

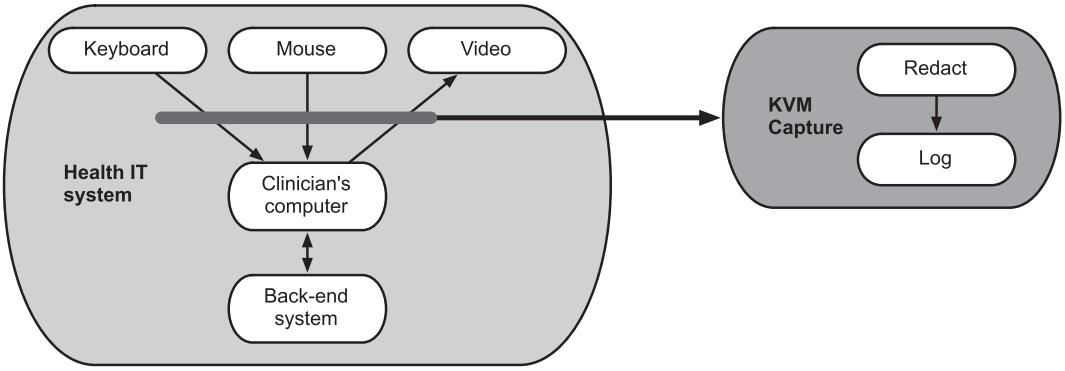

Fig. 1. Our measurement module passively listens to keyboard and mouse input and video output—and consequently remains system-independent and accommodates closed, mission-critical, and/or certified health IT systems.

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/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.

4. 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 overall 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: (a) finding text in an image (also known as text *segmentation*), and then (b) recoloring segmented image regions to "remove" text. (We note that such segmentation is also the first step of *optical character recognition—OCR*.) 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 filtering [13], which aims to classify individual pixels as "text" or "non-text." For Gabor, we looked at both unsupervised classification and supervised classification [14]. (Table 2, at the end of our discussion, summarizes these techniques.)

4.1. 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 3×3 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, we apply the redaction mask to the original image to produce a redacted image.

Fig. 2–4 show examples from our prototype. Unfortunately, standard practice in commercial health IT prevents customers from disclosing user interface details (e.g., [16]), so we cannot show the original screenshots used in our experiments, but rather use a representative open-source one. In terms of complexity, this sample would fall at the minimum of our test corpus (Section 6)—e.g., at the bottom left of Fig. 9a.

4.2. 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). For orientation, we followed Jain and Bhattarchee and chose

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

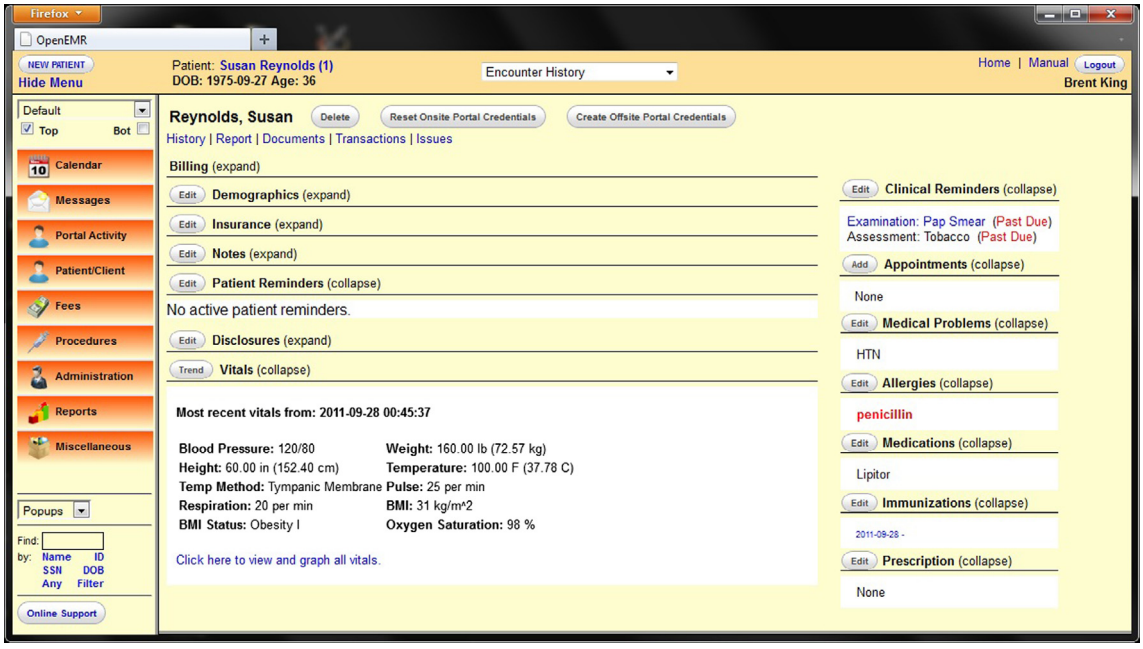

Fig. 2. In this paper, we use this sample screenshot from http://www.open-emr.org to demonstrate our techniques in the context of a realistic health IT system—as we are prevented from disclosing the proprietary screenshots from our test corpus from real EMRs. In terms of complexity, this sample would fall at the absolute minimum of our corpus—e.g., at the bottom left of the top Fig. 9a.

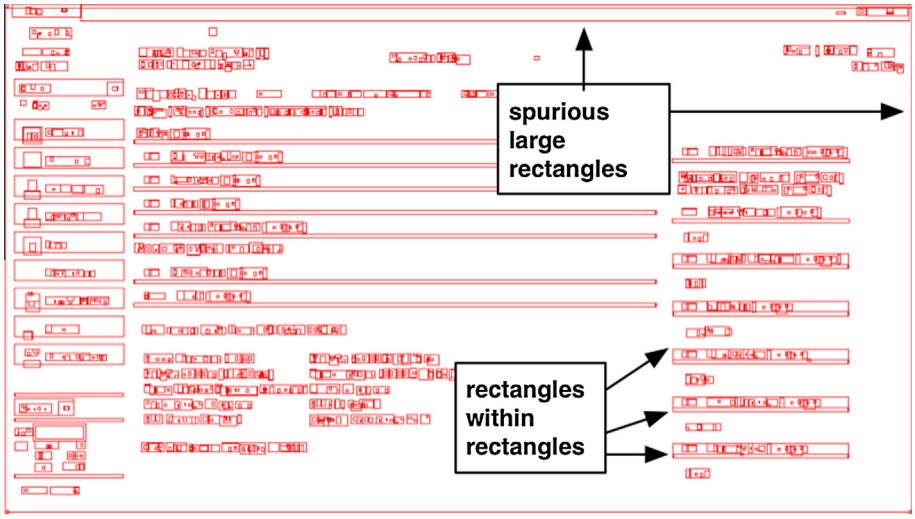

Fig. 3. This image depicts the rectangles that result from processing Fig. 2 with Canny edge detection, polygon detection, and polygon bounding with rectangles. Note tiny rectangles enclosed within larger ones—as well as spuriously large rectangles containing the entire screen.

to detect signals oriented in a uniform variety of positions. For wavelength, we chose powers of two

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

in order to span a collection of feature sizes. When we ran this on a collection of our screenshot, we found it was effective at detecting the relevant features.

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.

4.3. Classification

Once we have 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 [17] 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. Jain and Bhatterjee [14] clustered into three classes for text analysis; we started with that, but found that some screenshots clustered better visually into $k = 2$ classes.

During k-means clustering, the system relied on stopping conditions of the first of 10,000 iterations or an error rate of .0001. We chose the initial cluster centers using a more recent technique

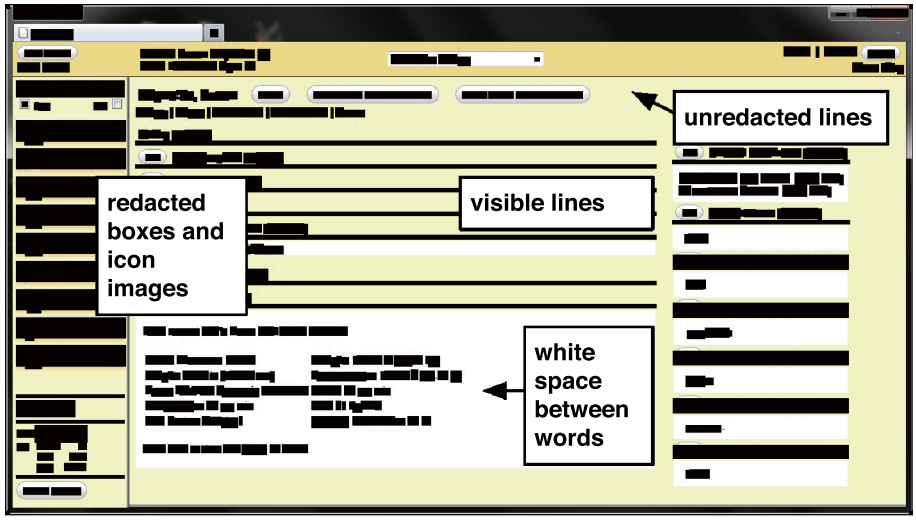

Fig. 4. This image derives using the filled Canny-detected rectangles (Fig. 3) as a redaction mask to the original image (Fig. 2). Canny missed some true edges throughout the image (false negatives for edge detection) and added edges where text does not exist near the icons on the left. Canny also added whitespace between words. (We first eliminated the all-enclosing rectangles from Fig. 3; otherwise, the entire image would have been redacted!)

[18] 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 multi-fold: 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.

5. Experimental tools

Section 4 above described our approaches to automatic text redaction. However, for both clinicians as well as system experimenters, it is important to keep users in the loop. This present section describes two tools we built for this purpose.

5.1. Tool: scrubs

Our *scrubs* tool (Fig. 5) captures and redacts screenshot images dynamically, in real time, using Canny. Our prototype uses x11vnc [10], pthreads [19] and the RFB protocol [9].

When a health IT user decides some sequence of activity should be logged for later analysis, it is 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 (Fig. 6). While paused for user edits, the system continually processes and maintains received screen updates in the background, and upon returning to record mode, the system displays a compilation of all updates processed during pause.

5.2. Tool: *five_in_one*

The Canny Edge approach to text redaction (Section 4.1) 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" (Fig. 7). This tool permits a wide range of interactive operations, including 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. Fig. 8 shows one example.

6. Evaluation

To evaluate our system, we looked at the relative effectiveness of the two approaches to automatic redaction (Sections 6.1 and

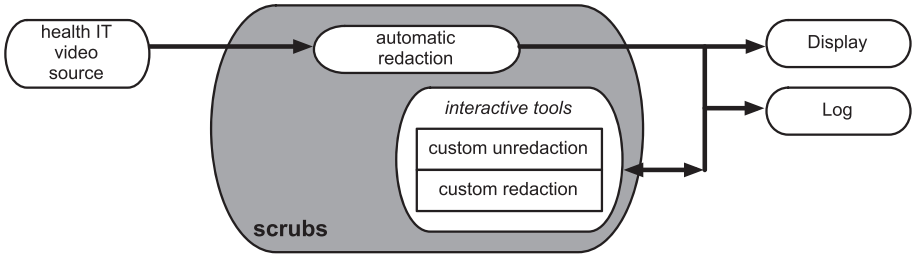

Fig. 5. Our *scrubs* tool enables automatic redaction, interactively tunable, in real time.

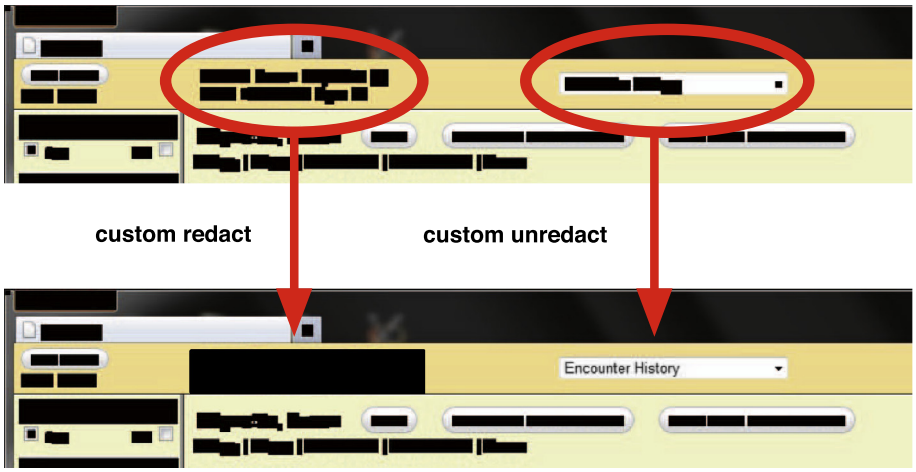

Fig. 6. Our scrubs tool automatically applies redaction, and then permits custom redaction and unredaction.

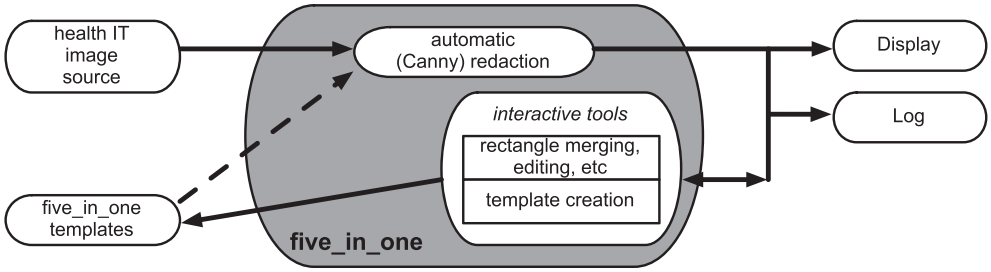

Fig. 7. Our *five_in_one* tool enables experiments and end users to analyze and edit the rectangular regions of potential text flagged by Canny redaction, and to generate and apply templates.

6.2) and at non-text aspects of privacy preservation (Section 6.3). We also looked at effectiveness of context preservation (Section 6.4), as well as basic computational costs (Section 6.5). Section 6.6 summarizes.

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 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.

Fig. 9 illustrates the complexity of the datasets according to the number of redaction rectangles generated for each screenshot.

6.1. Canny

Our testing showed that Canny-based text redaction requires improvements before the system can apply it meaningfully to health IT datasets, such as the EMR screenshots we used.

The Canny approach had several problems. It generated redaction rectangles that cover large parts of the screen, thus reducing 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). For example, Fig. 3 shows false negative and spuriously large rectangle issues; Fig. 10 shows false negative

Fig. 8. As one example of *five_in_one* editing, a user can clean up the redaction rectangles produced by Canny by first thinning out superfluous rectangles and then enlarging the remaining rectangles one pixel at a time.

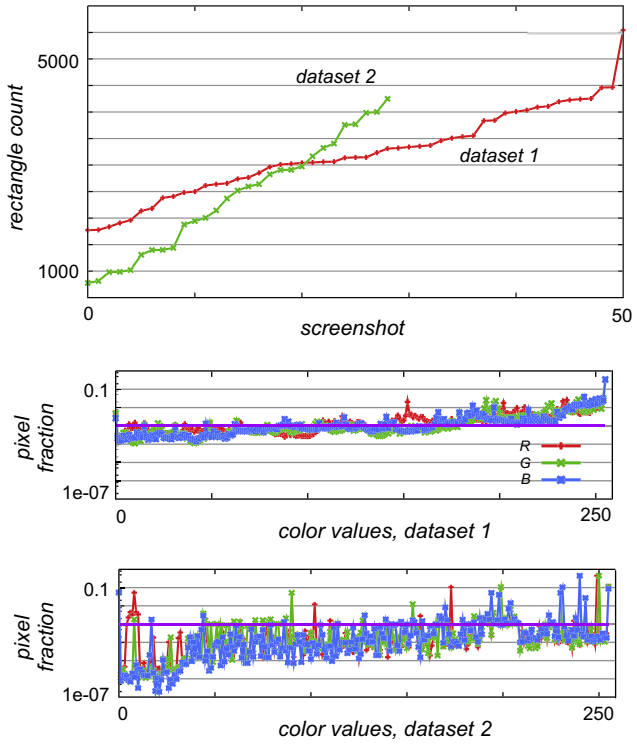

Fig. 9. Measures of the visual complexity of our EMR datasets: (a, top): Canny redaction rectangles, sorted; (b, bottom): normalized color variety.

issues; Fig. 4 shows examples of whitespace and false positive issues.

Our analysis did suggest ways that Canny redaction could be tuned to be usable for this application. As Fig. 11 shows, We can eliminate rectangles that cover all or most of the screen, and rectangles that contain a large number of the other rectangles. As Fig. 12 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.

Manual editing with our five_in_one tool can reduce the rectangle count by 75%, in samples from our corpus. 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.

6.2. 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 icons and logos) and did not erroneously redact large rectangles from the screenshot, as Canny-based redaction did.

Qualitative Analysis. Fig. 13 revisits Fig. 2 using Gabor-based redaction where $k = 2$ and $i = 0$. In Fig. 13, note how Gabor-based redaction fills whitespace between words in sentences but does not redact objects such as the full icons. It does not redact large rectangles from the screen as Canny-based redaction. However, as Fig. 14 shows, Gabor will occasionally fail to redact text with certain font scales and textures.

Quantitative Analysis for Unsupervised. To evaluate unsupervised Gabor redaction, we chose a few representative but dissimilar (using the metric we present below, in Section 6.4) 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

Fig. 10. Canny redaction on a gmail inbox shows missed text.

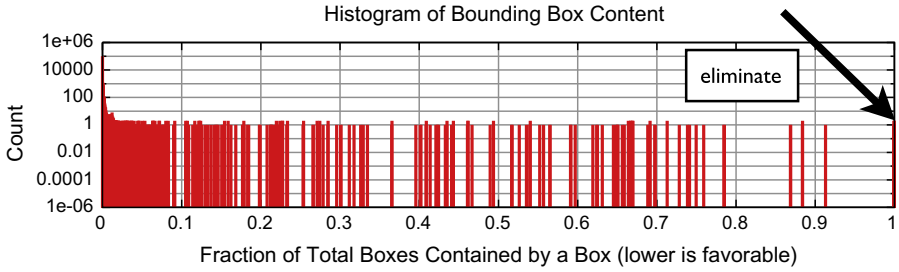

Fig. 11. Histogram of the fraction of total redaction rectangles in a corpus screenshot contained wholly within a given redaction rectangle.

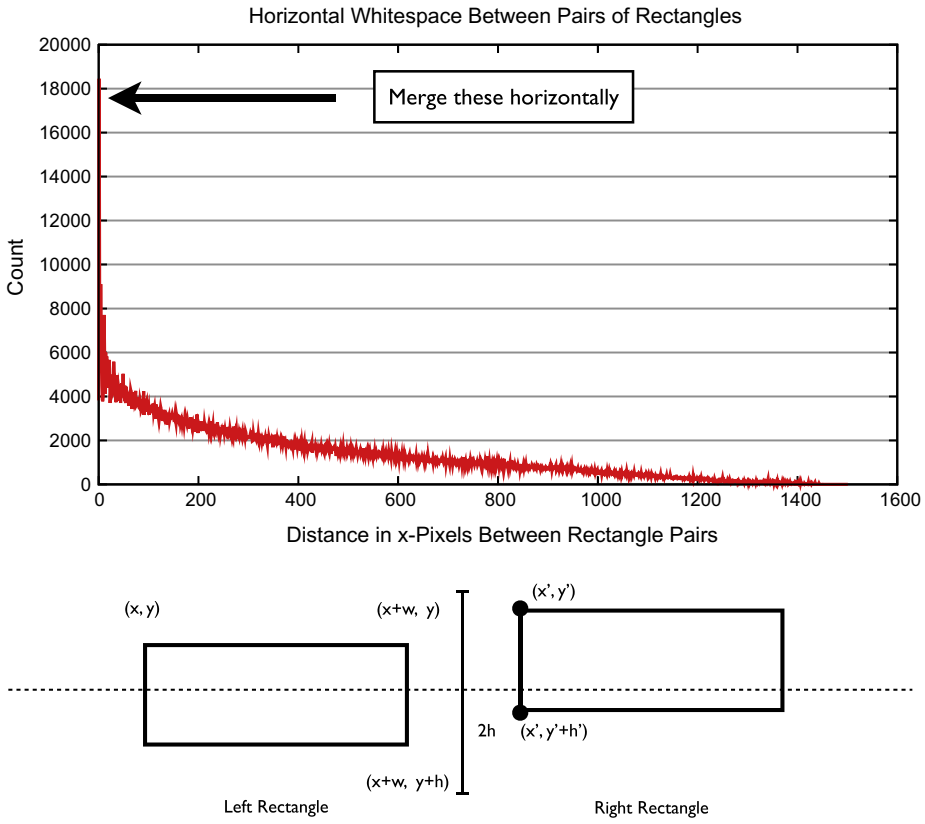

Fig. 12. Histogram of horizontal distances between pairs of rectangles heuristically on the same text line. If $(x+w) \leqslant x'$, $(y-\frac{h}{2}) \leqslant y' \leqslant (y+\frac{h}{2})$, and $y+\frac{h}{2} \leqslant (y'+h') \leqslant y+\frac{3}{2}h$, then we includes the value $x'-(x+w)$ in the histogram.

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.

Quantitative Analysis for Supervised. As Section 4.3 above discussed, we started by running unsupervised Gabor on dataset 1 and, for each screenshot, choosing the "best" screenshots manually, considering the various output classes of the redaction process, including partial false negatives. (Section 7.2.2 and Fig. 7.5 in the first author's thesis [20] contain more information on this process.) We then 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 manually choosing the best one (See Fig. 15.).

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).

6.3. Non-text information leakage

Our experiments also revealed ways in which redacted health IT 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 spe-

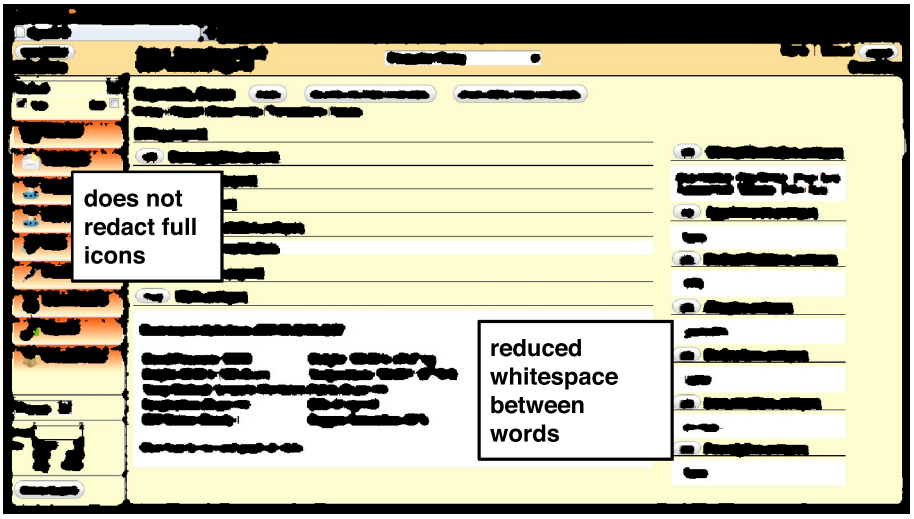

Fig. 13. This image shows Gabor-based redaction on the OpenEMR screenshot from Fig. 2. Gabor preserves more structure of objects such as the icons and horizontal lines, connects whitespace between words in sentences, and (although not shown here) redacts fractional characters at the edge of a screenshot.

Fig. 14. Gabor-based redaction can leave text partially unredacted, and miss text altogether, as this example from a Wikimedia Commons EMR image shows.

cific colors (e.g., red). (Again, the first author's thesis [20] has more information.)

6.4. Context preservation

As the introduction noted, traditional approaches to privacy and confidentiality seek to hide information. However, for our health IT screenshot capture tool to be useful, we also need to preserve information: the context of the health IT 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 (using unsupervised Gabor).

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 our health IT 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). Fig. 16 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 health IT screenshots.

6.5. Latency

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

To measure latency of text redaction, we used 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 programmatically printed timing information. All file loads were measured using a cold file cache. Table 1 shows the results.

6.6. Summary

Table 2 summarizes the two techniques we evaluated for automatic redaction in health IT screenshots. Overall, supervised Gabor seems the best immediate candidate. However, in future work, it would be interesting to explore a hybrid approach: first, applying fast Canny-based reduction; then using the merging and filtering heuristics from Section 6.1 to reduce the number of superfluous rectangles; then applying supervised Gabor redaction over a random sampling of pixels from each redaction rectangle to eliminate false-positives—as a lack of Gabor-detected text would let us automatically discard the redaction rectangle. After this, we would merge whitespace. As a result, we may be able to increase Canny's

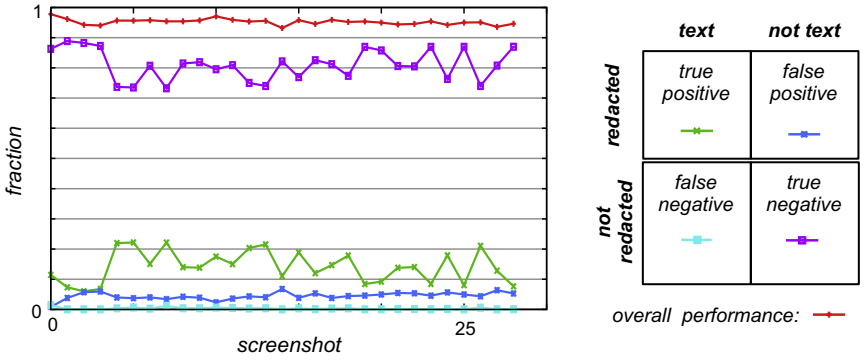

Fig. 15. Effectiveness of Gabor-based text redaction. We trained a liblinear L1-regularized logistic regression classifier on dataset 1 and applied it to dataset 2; in both cases, we used the qualitatively best unsupervised Gabor redaction as "ground truth" labeling.

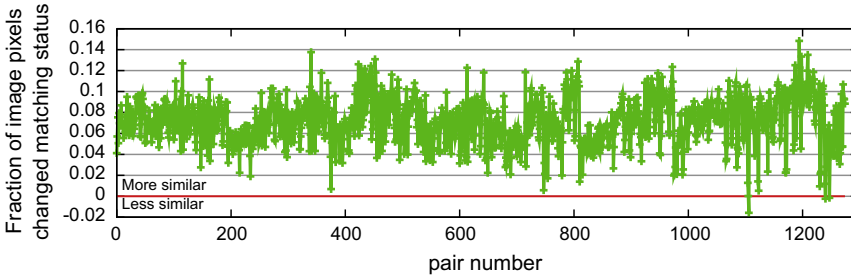

Fig. 16. Effect of text redaction on similarity of pairs of distinct screenshots in dataset 1. Fractions that fall above the horizontal line correspond to screenshots that are more similar after redaction.

sensitivity threshold to redact lower-contrast text while maintaining a low false positive rate.

7. Improving real-world health IT systems

We designed our system within the context of a larger vision: a privacy-protected "Redaction Service" that interacts with a monitored health IT system, a "Sharing Service" that functions as a repository for application stakeholders to share redacted screenshots, the health IT system itself with developers and maintainers, and an end-user who desires their system to be monitored and a web-browser through which the end-user can interact with redaction and sharing services. Fig. 17 illustrates this vision.

7.1. The pieces

Our current prototype implements screen capture and text redaction. For roll-out in medical enterprises, one would need to add the system components to support sharing of data—and to support end-user annotation (both graphical—circles and arrows—and text). One would also need to expand the currently limited UI with more easy-to-use affordances such as drop-down menus.

On a lower level, potential future work could also explore trying ways to improve on the core Gabor and Canny techniques.

For example, currently, we follow the standard practice of tuning Gabor parameters heuristically by starting with a parameter set that worked for others and hand-tuning variables until the system provides "useful" results. Dunn and Higgins propose a systematic method to select the parameters based on decision theory and assuming the images have only two textures of interest [21]; Weldon proposes a systematic approach based on minimizing predicted error [22]. Adapting one these of approaches might reduce the need for heuristics while still providing good results.

We could perhaps improve computational efficiency of Gabor by applying the concept of *steerable filters* developed by Freeman and Adelson [23]; two steerable filters could replace four Gabor orientations and therefore reduce the overall filterbank size by one half.

In our experiments, we used Canny to detect the edges of the text regions. Researchers such as Atae-Allah et al. [24] and Barranco et al. [25] propose using *Jensen-Shannon divergence* as an alternative technique to detect edges; it would be interesting to apply that technique in our setting.

(Sections 9.2 and 9.3 in the first author's thesis [20] contain longer discussion of technical extensions to this work.)

Table 1
Our measured costs of redaction.

	Mean	Stddev
Latency to classify pixels (s)		
Canny	0.096	0.009
Unsup. Gabor	2.077	0.722
Sup. Gabor	2.077	0.017
Latency to generate elements for 28 Gabor filters (s)		
Set up	0.365	0.004
Build	13.12	0.181
Normalize	4.569	0.033
Latency to load Gabor elements from file on disk (s)		
Features	5.987	0.407
Labels	0.226	0.069

7.2. Health IT usability

In Section 2, we discuss a few of the many real-world scenarios we have identified where health IT users experience frustration.

Table 2
Summary of redaction techniques.

Canny	Finds edges	Fast	Many errors; requires tuning
Gabor Unsupervised: requires human judgment Supervised: requires training set	Finds texture	Slow	More accurate

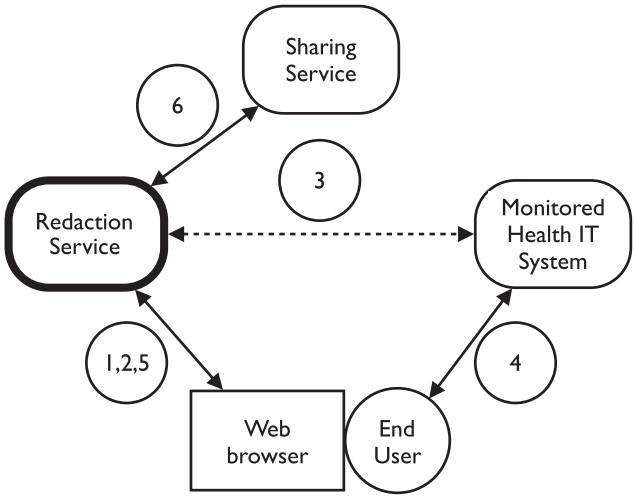

Fig. 17. Our system within the context of a larger vision. The figure includes a "Redaction Service" that interacts with the monitored heath IT system, a "Sharing Service" that functions as a repository for application stakeholders to share redacted screenshots, a monitored health IT system, an end-user who wish their system to be monitored and a web-browser through which the end-user can interact with redaction and sharing services. Within this big-picture context, internal components of the bold box labeled "Redaction Service" define the contribution of this work. A sample usage scenario follows the numerical labels in the following steps. (1) An administrator enables the "Redaction Service." (2) An end-user triggers the "Redaction Service" to log a monitored host. (3) The "Redaction Service" connects to the monitored host and begins logging, with automatic redaction. (4) The end-user interacts with monitored health IT system. (5) The end-user reviews and possibly edits logged data. (6) The end-user publishes redacted screenshots.

With our system in place, when the health IT user experiences frustration, she simply presses a button to generate a redacted screenshot of the scenario in question. Either at that point, or later in the day, she can then use our tools to touch up the image (e.g., by unredacting text that needs to be visible) and annotate it, to explain the issue:

- "See, these lab results all have the same title—useless!"
- "This initial dropdown menu does not contain the right diagnosis."
- "The system should recognize that two 5 mg pills equals one 10 mg pill, in this case!"
- "I don't know which type of stomach cancer; that's why I'm referring the patient to a specialist."

We now have a record—already scrubbed of IRB-sensitive material—which can propagate onto developers, IT managers, and usability researchers.

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.

8. 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 explore complementary tasks; Fig. 18 sketches this space.

GUI tools. The MIT Sikuli research project combines computer vision and programming to enable users to create machine-independent, visually-programmed and actuated programs [26]. A commercial product call eggPlant also allows developers to test GUIs with machine-independent, automation scripts [27]. Many screen capture applications such as Snipping Tool [28], Snapz Pro X [29], and xwd [30] exist. Some programs capture still screenshots, others capture both stills and video, and some allow end-users to annotate captures. Google's in-house UseTube [31] supports employees who wish to perform user studies of any network-connected computer; it simplifies the act of performing, archiving, and accessing user studies.

Interactive graphics and image tools. Many commercial and free-software tools such as Gimp [32], Photoshop [33], Aperture [34], Final Cut Studio [35], Pixelmator [36], and Imagemagick [37] 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. Google Goggles can extract and recognize text from natural scenery for purposes such as language translation among many others [38]. 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.

Data de-identification. 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 [39,40]. 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 [41] and *l*-diversity [42].

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. Document redaction products such as Rapid Redact [43] and brava! [44] 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 (rather than relying on textual metadata).

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 [45]. A 2003 competition sponsored by ICDAR has datasets available for optical character recognition (OCR), word

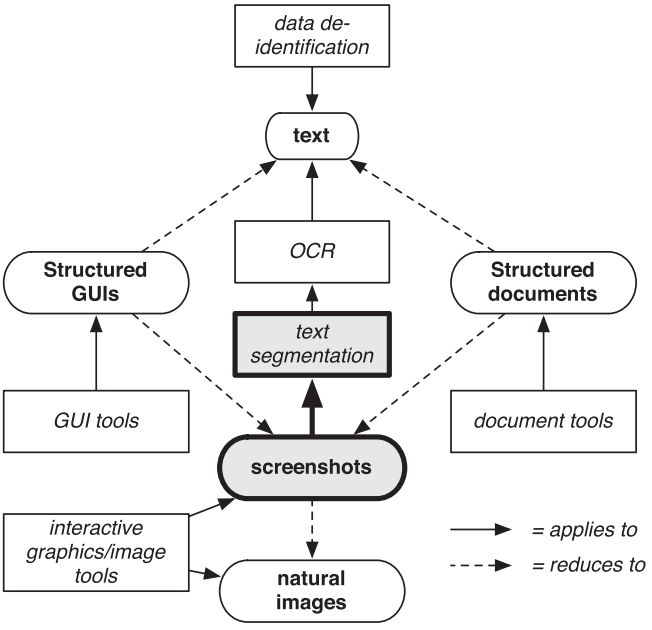

Fig. 18. Sketch of classes of data and tools which act on them. Our work explores automatic discovery and then redaction of text on otherwise flat screenshots.

recognition, text locating, and other purposes [46]. 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.

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

9. 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 health IT, privacy concerns require that any such screenshots have sensitive data redacted and the logistics of real-world health IT require that any solution not touch the internals of the software.

To address these concerns, 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.

Now that we have established that this core redaction step is feasible, the next step would be incorporating it in the larger vision of Fig. 17 and deploying the system in a real user environment. Potential areas for other future work include improving Canny for general-purpose use, implementing predicate matching to process screenshots according to logical conditions, and building a larger ground-truth data corpus.

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 health IT systems in practice.

Acknowledgments

This work is based on the first author's MS thesis work [20].

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

References

[1] S.W. Smith, R. Koppel, Healthcare information technology's relativity problems: a typology of how patients' physical reality, clinicians' mental models, and healthcare information technology differ, Journal of the American Medical Informatics Association, 2013, submitted for publication.
[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] Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. LIBLINEAR: a library for large linear classification. Journal of Machine Learning Research 2008;9:1871–4.
[6] CGAL, Computational Geometry Algorithms Library <http://www.cgal.org>.
[7] Zomorodian A, Edelsbrunner H. Fast software for box intersections. In: Proceedings of the Sixteenth Annual Symposium on Computational Geometry, SCG '00. New York, NY, USA: ACM; 2000. p. 129–38.
[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] Canny J. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI 1986;8(6):679–98.
[13] Gabor D. Theory of communication. Part 1: the analysis of information. Journal of the Institution of Electrical Engineers – Part III: Radio and Communication Engineering 1946;93(26):429–41.
[14] Jain A, Bhattacharjee S. Text segmentation using Gabor filters for automatic document processing. Machine Vision and Applications 1992;5:169–84.
[15] Suzuki S, Abe K. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 1985;30(1):32–46.
[16] Koppel R, Kreda D. Health care information technology vendors' "Hold Harmless" clause: implications for patients and clinicians. Journal of the American Medical Association 2009;301(12):1276–9.
[17] Lloyd S. Least squares quantization in PCM. IEEE Transactions on Information Theory 1982;28(2):129–37.
[18] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Technical Report 2006-13, Stanford InfoLab, June 2006.
[19] 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.
[20] J.A. Cooley, Screen Capture for Sensitive Systems, Computer Science Technical Report (M.S. Thesis) 2011-690, Dartmouth College, May 2011.
[21] Dunn D, Higgins W. Optimal Gabor filters for texture segmentation. IEEE Transactions on Image Processing 1995;4(7):947–64.
[22] T.P. Weldon, The Design of Multiple Gabor Filters for Segmenting Multiple Textures, Tech. rep., University of North Carolina at Charlotte, 2007.
[23] Freeman WT, Adelson EH. The design and use of steerable filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 1991;13:891–906.
[24] C. Atae-Allah, J. Gomez-Lopera, P. Luque-Escamilla, J. Martinez-Aroza, R. Roman-Roldan, Image segmentation by Jensen–Shannon divergence. Application to measurement of interfacial tension, in: Proceedings of the 15th International Conference on Pattern Recognition, vol. 3, 2000, pp. 379–382.
[25] Barranco-López V, Luque-Escamilla P, Martínez-Aroza J, Román-Roldán R. Entropic texture-edge detection for image segmentation. Electronics Letters 1995;31(11):867–9.
[26] Yeh T, Chang T-H, Miller RC. Sikuli: using GUI screenshots for search and automation. In: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, UIST '09. New York, NY, USA: ACM; 2009. p. 183–92.
[27] TestPlant, eggPlant <http://www.testplant.com/> (January 2011).
[28] Microsoft, Snipping Tool <http://windows.microsoft.com> (January 2011).
[29] Ambrosia Software Inc., Snapz Pro X <http://www.ambrosiasw.com/utilities/snapzprox/> (January 2011).
[30] Community Supported, xwd <http://www.xfree86.org/> (January 2011).
[31] LaRosa M, Poole D, Schusteritsch R. 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. New York, NY, USA: ACM; 2009. p. 2971–86.
[32] The GNU Project, GIMP: The GNU Image Manipulation Program <http://www.gimp.org/> (January 2011).
[33] Adobe, Photoshop <http://www.adobe.com/products/photoshop/photoshop/whatisphotoshop/> (January 2011).
[34] Apple, Aperture <http://www.apple.com/aperture/> (January 2011).
[35] Apple, Final Cut Studio <http://www.apple.com/finalcutstudio/> (January 2011).

[36] Pixelmator, Pixelmator <http://www.pixelmator.com/> (January 2011).
[37] Community Supported, ImageMagick <http://www.imagemagick.org/script/index.php> (January 2011).
[38] Google, Google Goggles <http://www.google.com/mobile/goggles/> (January 2011).
[39] Meystre S, Friedlin F, South B, Shen S, Samore M. Automatic de-identification of textual documents in the electronic health record: a review of recent research. BMC Medical Research Methodology 2010;10(1):70.
[40] Aberdeen J, Bayer S, Yeniterzi R, Wellner B, Clark C, Hanauer D, et al. The MITRE identification scrubber toolkit: design, training, and assessment. International Journal of Medical Informatics 2010;79(12):849–59.
[41] Sweeney L. k-Anonymity: a model for protecting privacy. International Journal of Uncertainty Fuzziness and Knowledge Based Systems 2002;10(5):557–70.
[42] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: privacy beyond k-anonymity, ACM Transactions on Knowledge Discovery Data. 1, 1, Article 3 (March 2007). http://doi.acm.org/10.1145/1217299.1217302.
[43] RapidRedact, RapidRedact <http://www.rapidredact.com/> (January 2011).
[44] Informative Graphics Corporation, brava! <http://www.infograph.com/company.asp> (January 2011).
[45] ICDAR <http://www.icdar2011.org/EN/volumn/home.shtml> (April 2011).
[46] ICDAR Dataset <http://algoval.essex.ac.uk/icdar/Datasets.html> (April 2003).