

Whiteboard sharing: capture, process, and print or email

Michael Gormish, Berna Erol, Daniel G. Van Olst, Tim Li, Andrea Mariotti
Ricoh Innovations, Inc.
2882 Sand Hill Rd. Ste 115, Menlo Park, CA USA 94025
<http://beta.rii.ricoh.com>, lastname@rii.ricoh.com, +1.650.496.5700

ABSTRACT

Whiteboards support face to face meetings by facilitating the sharing of ideas, focusing attention, and summarizing. However, at the end of the meeting participants desire some record of the information from the whiteboard. While there are whiteboards with built-in printers, they are expensive and relatively uncommon. We consider the capture of the information on a whiteboard with a mobile phone, improving the image quality with a cloud service, and sharing the results. This paper describes the algorithm for improving whiteboard image quality, the user experience for both a web widget and a smartphone application, and the necessary adaptations for providing this as a web service. The web widget, and mobile apps for both iPhone and Android are currently freely available, and have been used by tens of thousands of people.

Keywords: Image capture, enhancement, mobile phone, web service

1. INTRODUCTION

Whiteboard Usage

Whiteboards that print in black and white, spot color, or full color have existed for many years. These whiteboards typically have a writing rotating surface, a line scanner and built-in printer. Such whiteboards are quite common in Japan and produce quite readable text on A4 or letter sized pages. Their value is well recognized with meetings being preferentially held in rooms with these boards. Obviously, such printers are not available with every whiteboard. Furthermore, even when a whiteboard has a printer it can be out of paper or ink or suffer from paper jams. Whiteboard printers are typically fairly slow, and thus getting everyone in the meeting a copy of the whiteboard can require a trip to the copier.

The information from whiteboards is so valuable there are also whiteboards with dedicated cameras mounted above to capture images. The left side of Figure 1 shows a whiteboard with a built-in printer connected just below the pen tray. This whiteboard also has a camera mounted to the wall to capture images of the whiteboard, only the bottom of the mounting bracket of this camera is visible in the picture. While mounted cameras avoid the problem of running out of supplies, again they are relatively expensive, not available everywhere, and don't work with portable whiteboards. Also, it is not immediately obvious how to get images captured by the mounted camera to the meeting participants.

For these reasons we initially approached the problem of capturing information from a whiteboard by using a portable digital camera, and then as mobile phone cameras improved in quality we made use of these devices. Pictures taken with portable cameras of whiteboards have lots quality problems: reflection from room lights or camera flash, uneven illumination, and sometimes very dark "white" boards. Low quality images do not serve the purpose of capturing the information from a meeting. Thus we provide an algorithm to improve the quality of whiteboard images and compare it to other techniques.

Image Quality Improvements

We take the approach of segmenting the image into foreground and background, removing the background (presenting it as white), and enhancing the foreground (presenting as more saturated color). There are a variety of background subtraction methods for different applications.¹ Many background subtraction algorithms make use of a reference image, e.g. taken before the foreground existed, or a sequence of images, and assume that the background is the portion of the image that does not move. While we could have asked users to take an image of the background before writing on

the whiteboard, people usually don't think about capturing information from a meeting that far in advance. Thus we are interested only in methods requiring a single image.

Binarization algorithms work on single images and many are designed to enhance text, a primary component of whiteboard information. We wanted to maintain color from the multiple markers that might be used on a whiteboard, and we even wanted to maintain some of the difference in intensity of written information. Our algorithm thus consists of background removal and edge enhancement.

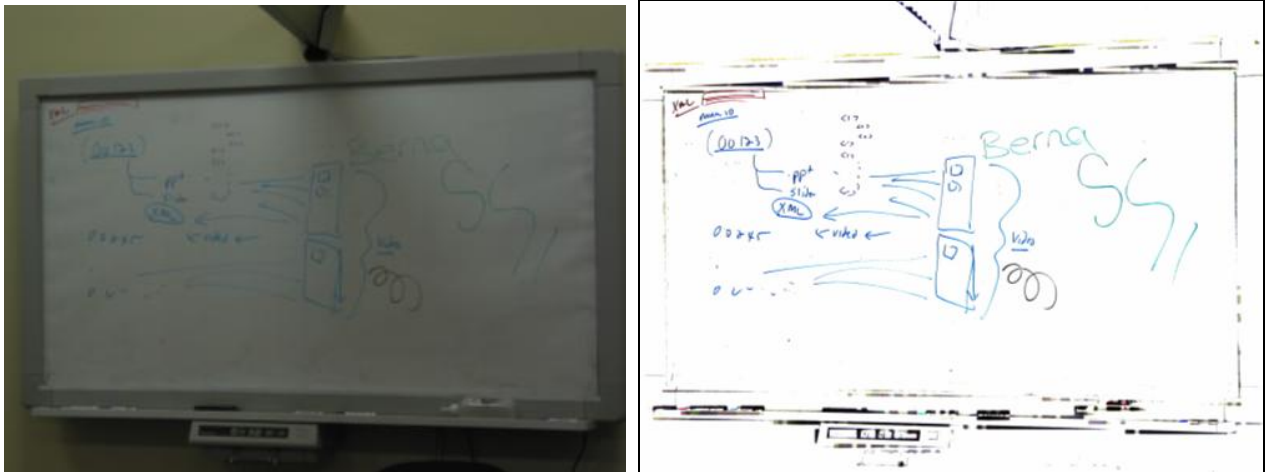


Figure 1. Whiteboard image captured by a digital camera: original (left), enhanced with our method (right)

User Experience

Whiteboard cleanup software has existed for years as a stand alone software package, e.g. by Polyvision.² We wished to avoid install hassles and gain the other benefits of Software as a service (SaaS). Therefore, we deployed the clean-up algorithm as a web widget, with an elegant user interface, and a backend server providing the image processing horse power. Although the widget required only three steps, namely, upload, process, download, to get cleaned-up images, we observed only limited usage. Realizing that users had to get images from cameras to PCs to use the widget and then had to do something with the resulting image after downloading, we implemented applications to capture the whiteboards on smartphones and use the same web service for processing. We also added email capability for the results. Thus the user still had three steps: capture, select destination, and send, but those steps went from whiteboard to inbox, rather than just disk to website to disk.

Many times whiteboard images are printed for ease of sharing or the ability to view many images at once. Not only does the clean up make that operation easier, but the quality background removal uses less toner or ink in the printing process.

For awhile, our iPhone application became the number four free business application in the United States, and number one in Japan. Extensive usage has enabled further improvements because we are able to find out user needs for a very large population.

2. WHITEBOARD ENHANCEMENT ALGORITHM

For whiteboard image enhancement we considered several methods. The first method was to enhance the contrast of the image by assuming the background of the board is white. However this approach did not work well for most images because (1) when the white balance settings in a digital camera is not set correctly, the captured whiteboard image may have variety of background colors besides white, such as red or dark blue, and (2) the illumination of the whiteboard is usually not uniform and the flash reflecting on the whiteboard (usually significantly brighter than the rest of the

whiteboard) reduce the contrast enhancement accuracy significantly. The second method we considered was to binarize the whiteboard image while preserving the colors of the foreground strokes. Because whiteboard images are compressed by the camera using JPEG compression, foreground strokes suffer from ringing and chroma bleeding artifacts caused by the fact that some high frequency data is omitted during compression. This effect is shown in Figure 2. When binarization is directly performed on the images with JPEG artifacts, then the bright rings around the text lines result in additional really thin lines in parallel with the desired text, which becomes almost impossible to read.

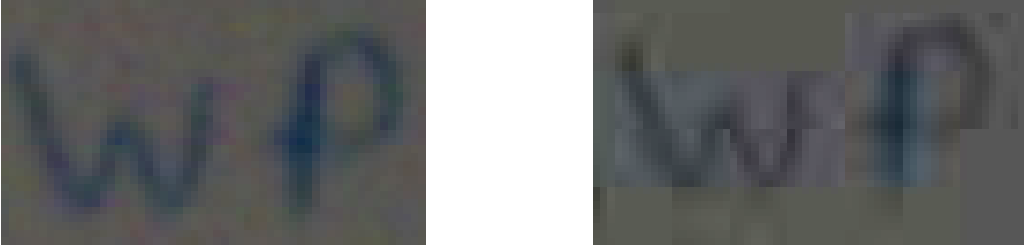


Figure 2. A section from a whiteboard image: uncompressed (left), JPEG compressed (right)

Our method for whiteboard enhancement is based on background identification and subtraction and color enhancement along edges. The algorithm was tuned specifically for whiteboards (asymmetric processing of the background), and the color enhancement was designed to operate best on pen strokes rather than regions. The algorithm needed to work with images as small as 640x480 up to multi megapixel camera images. Because we intended to run on PCs or servers we were not memory constrained, but because we wished to handle lots of images it was necessary that the algorithm be relatively fast.

Figure 3 is a block diagram of the image processing operations. The left half computes a background image and subtracts it from the input image. The right half computes an edge image and uses the presence of an edge to select between the saturated image and the non-saturated image. In this way the whiteboard strokes are enhanced, but slight variations still remaining after background removal are not.

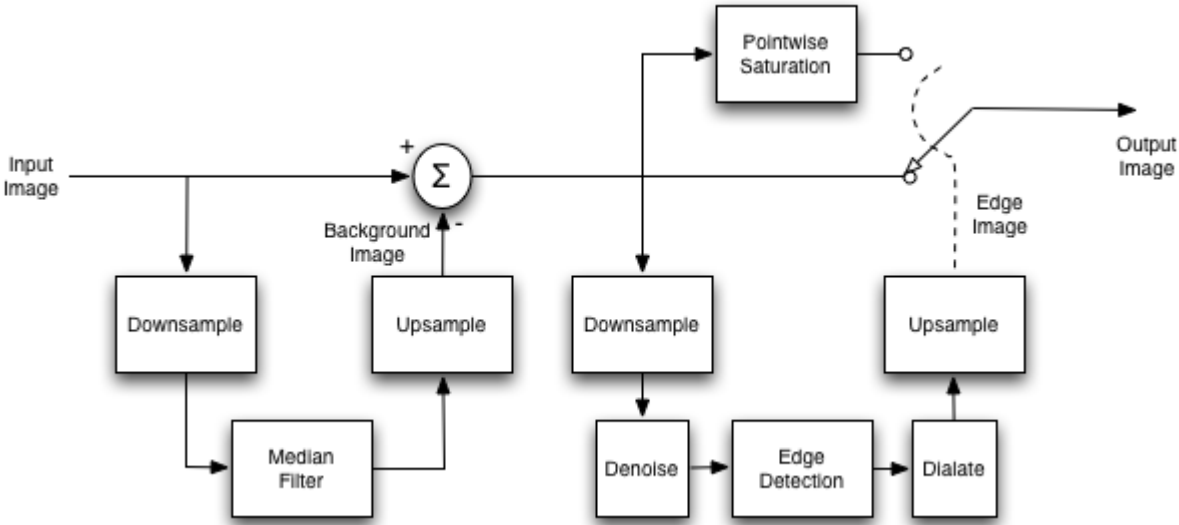


Figure 3. Whiteboard cleanup image processing path

Background Subtraction

Since we do not have multiple images to use to define the background, and we are most interested in whiteboards, we assume the whiteboard is the color of the majority of the pixels. Of course, because of different illumination a solid color whiteboard can vary substantially across the surface, so a local median filter seemed ideal for identifying the background. For very small images, a median filter with a diameter of 17 worked well, but for higher resolution images this was too small. Using larger median filters significantly slowed the computation. Therefore, regardless of the input image size, we resample to an image with roughly 30,000 pixels, and apply the median filter to this image. The median filter thus operates quite quickly, and the background detection is less dependent on the resolution of the input image.

Once computed the background image is subtracted from the original image. Since we assume the background is white, and operating with dark pixels having a value of 0 and white pixels a value of 255, the "subtraction" operator actually moves all pixels with a color close to the background color to white (255). Pixels that are darker than the background maintain the same distance to the new background. Thus if the background was 230, and a pixel was 200 (30 darker than the background), its value after "subtraction" would be 225 (30 darker than a white background). In this sense, the algorithm is definitely optimized for whiteboards rather than blackboards. The left side of Figure 4 shows a whiteboard image with the background subtracted.

Foreground Enhancement

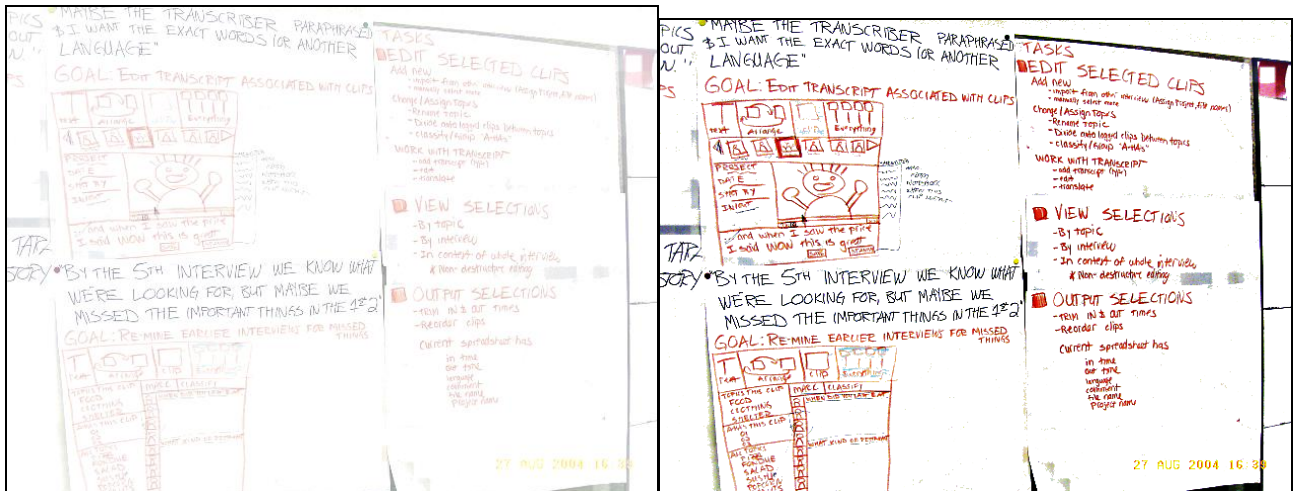


Figure 4. Foreground image: without enhancement(left) with enhancement (right)

Once the background has been subtracted the marks on the whiteboard are quite light and need to be darkened, and maintain their colors. However, this enhancement can bring out artifacts from compression or otherwise if done indiscriminately. Thus first the image is denoised, stroke sized edges are detected, and those regions corresponding to stroke sized edges are enhanced. Once again speed and resolution independence are enhanced by reducing the image before denoising and edge detection. In this case detail is more important than with the background so the image is reduced to roughly 500,000 pixels. Grayscale is sufficient for edge detection, so the reduced image is also converted to grayscale. The grayscale image is smoothed by convolving with a 5x5 box kernel. Then the Canny edge detector is run. Our implementation makes use of OpenCV and we operate the Canny edge detector with an initial edge strength requirement of 10, and a linking value of 0.1, i.e. `cvCanny(image, image, 10, 0.1)`. Since the Canny edge detector produces very thin edges, the edges are dilated by 2 pixels, thus any stroke of up to four pixels wide in the reduced image will be completely selected as an edge. Finally the edge image is returned to the original resolution by linear interpolation. This edge image will then be used to decide if pixels should have saturation adjusted. If pixels are on edges their saturation is increased with the following curve fitting.

The enhanced R' , G' , B' components are computed by

$$R' = \frac{255}{1 + e^{C(k-R/255)}}, G' = \frac{255}{1 + e^{C(k-G/255)}}, \text{ and } B' = \frac{255}{1 + e^{C(k-B/255)}}$$

An example foreground image after background subtraction and after foreground enhancement is shown in Figure . As can be seen from the figure, the S-shaped fitting of colors results in enhancement of primary and secondary colors in the image, for example grey becomes black, red becomes bright red and yellow becomes bright yellow. The parameters C and k affect how much enhancement takes place. Our experiments showed that $C=60$ and $k=0.93$ results in good image quality.

3. IMAGE PROCESSING IN THE CLOUD

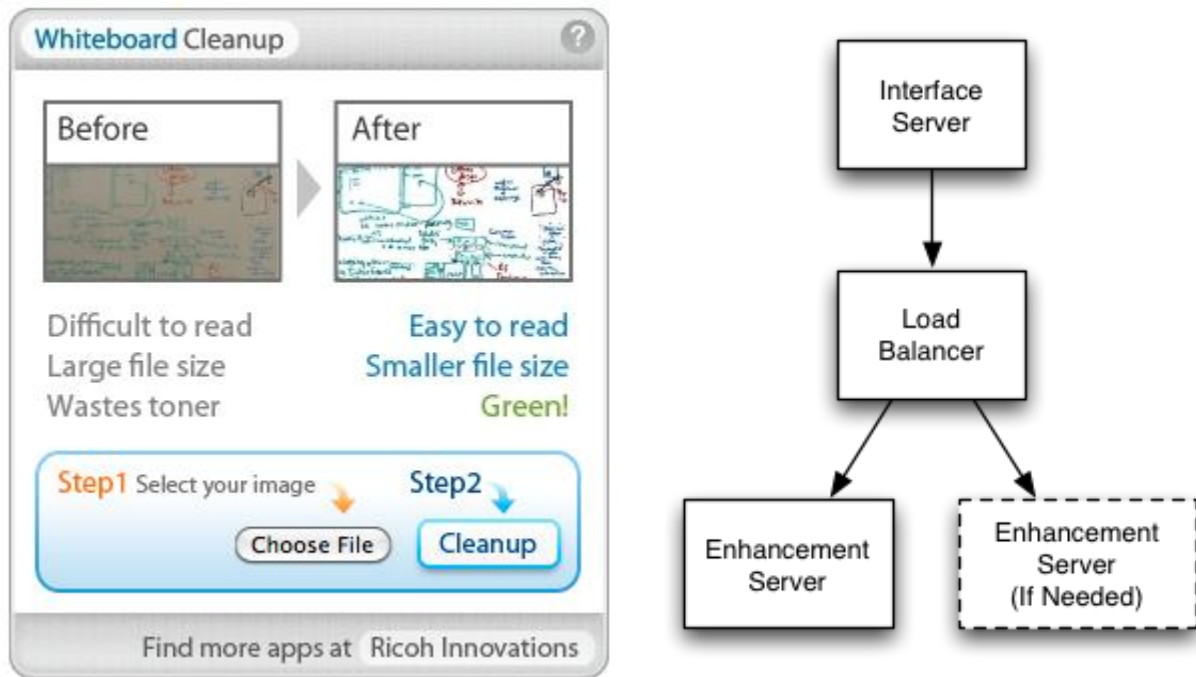


Figure 5. Whiteboard Cleanup Widget: UI (left), Architecture (right)

We initially made the whiteboard cleanup algorithm available via a webservice, the current interface is shown on the left of Figure 5. We provided the interface as simple html that could easily exist as an iframe, so it could be included in other web pages, or even on a user's Google gadget page. There are only three steps to using the widget:

- Choose a file to upload
- Press the Clean-up button
- Click on the preview image to download the cleaned up result

The web architecture is shown on the right of Figure 5. The interface server has a fixed address and provides the html to any client accessing the widget. The interface server also keeps track of usage and in the case of our mobile phone applications sends emails to the destination addresses. However, once images are received they are handled by one or more enhancement servers. The enhancement servers run OpenCV and some custom C code to do the image processing. We initially ran on a dedicated hardware with one interface server and one enhancement server. However, we have moved to Amazon's Elastic Computing environment, and our enhancement servers are part of an auto-scaling group. This means more enhancement servers are allocated (and rented), when we receive a large number of enhancement requests. Because this service is being freely provided we do limit the number of uses.

4. MOBILE PHONE USER EXPERIENCE

Although we had clear needs for whiteboard cleanup even within our office, and we provide a free "easy to use" web interface, usage was much less than we had hoped. Consideration of the real use of the widget quickly reveals additional steps, beyond the "three" we counted. In order to provide whiteboard information to all of the meeting participants the following steps were typical:

- Bring digital camera to meeting
- Take picture
- Transfer picture to laptop
- Bring up the whiteboard clean up widget
- Use the widget
- Launch an email application
- Choose email addresses from address book
- Attach image to email message
- Send

Fortunately, at just the time we were developing this algorithm, camera phones were reaching sufficient quality to capture whiteboard images. Furthermore, smartphones allowed special applications to be used to "grant a wish," and contained email address books. This allowed us to achieve the same result on a smartphone with the following steps:

- Launch our Whiteboard Share application
- Take picture
- Select email addresses from phone address book
- Send

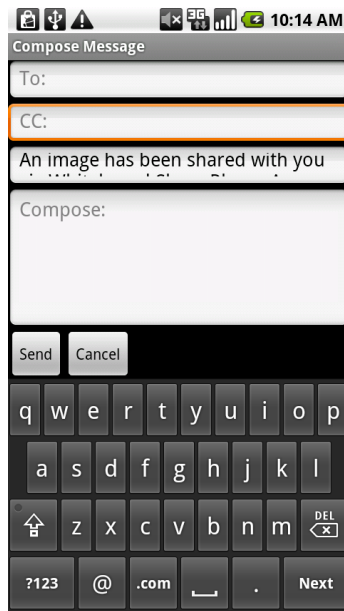
We built applications for the iPhone and phones using the Android operating system. The phone interface is shown in Figure 6, where 6a is from the iPhone and 6b-6d are from the Android (the interfaces identical except for screen size variation and use of operating system components users are used to, e.g. camera, email, and photo selection). When the application is launched the user may immediately choose to capture an image of a whiteboard, or may choose to use a previously taken picture from the phone's gallery of images(6a). A preview screen allows the user to decide if the correct gallery image has been selected or if the captured image is good enough(6b). If a good image is available, the user selects email addresses to send the image to, and then sends the image(6c). The image then goes to the same interface and enhancement servers used by the web widget(6d). In this case the interface server is responsible for emailing the image to the final destination. A settings page in the application allows the user to select default email addresses and messages. The user may also choose to include the original image with the cleaned up image, in case they are concerned that the cleaned up image will lose information.



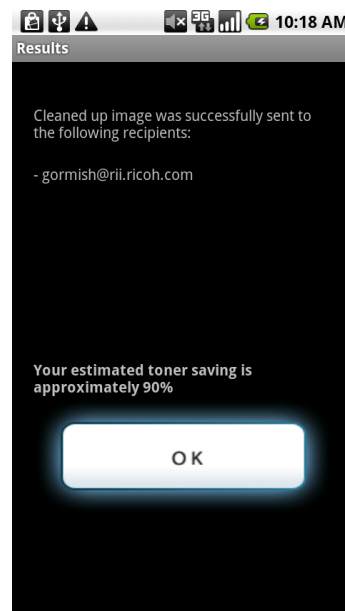
(a)



(b)



(c)



(d)

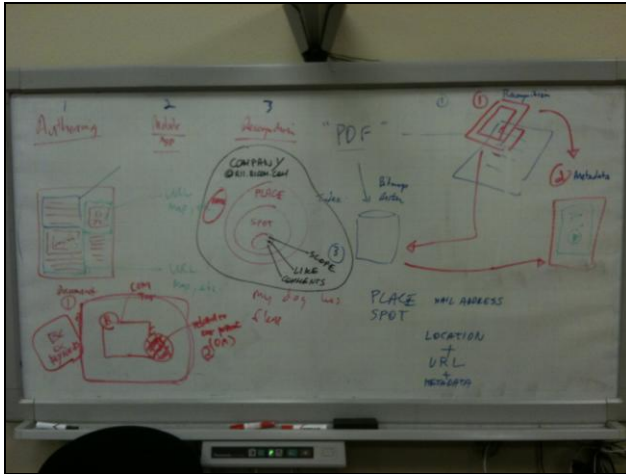
Figure 6. Mobile phone screen shots: initial screen (a), image preview(b), email(c), confirmation(d)

5. COMPARISON WITH OTHER APPLICATIONS

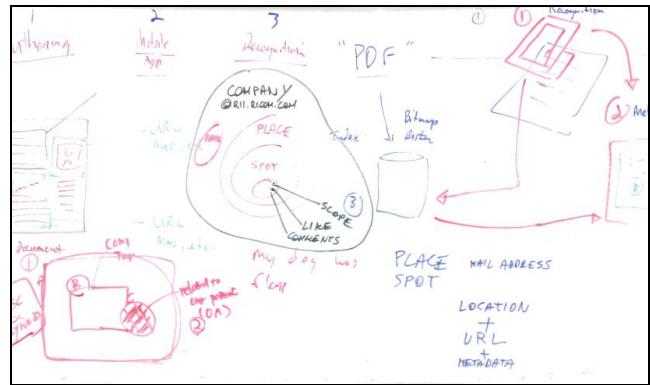
There are quite a few mobile phone applications that cleanup whiteboard images. Some do so by allowing the user to manually adjust brightness and contrast. While this method gives the human the final decision, it is time consuming and does not work with images that have mixed illumination. Thus we do not compare with any of those manual processing methods. Figure 7 shows an original image of a whiteboard, and the processed images provided by Jotnot,³ and Shareyourboard⁴ on the Android phone, and our Whiteboard Share method.⁵ Unfortunately, Shareyourboard does not support processing images already on the phone, instead images must be captured within the application, thus it was

impossible to use the same image for all three applications. Instead we captured one image within Shareyour board, and another image taken from the same place using an iPhone 3GS camera. The Android does have a better camera than the iPhone 3GS, which was used for Jotnot and our method. Jotnot and our camera application and our widget allow processing of any image, so we took the image in the upper left of Figure 7 with the standard camera application, then used the same input for both Jotnot and WhiteboardShare.

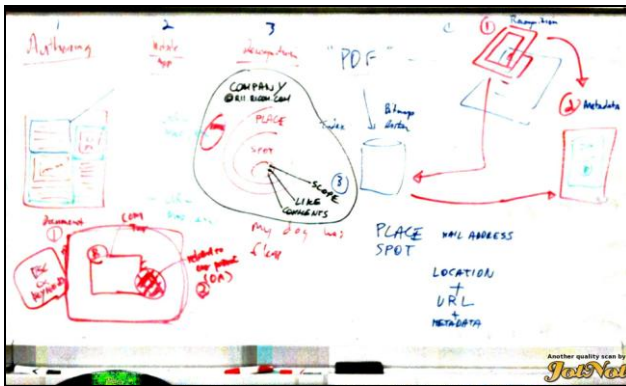
The first item that is immediately obvious is not all applications present the whole image Shareyourboard, and JotNot both do some automatic cropping, and perspective correction. In fact Shareyour board is so aggressive when cropping that one must be careful to include the outer edges of the board in the image. The second obvious difference is the strength of the lines. Our Whiteboard Share generate more readable and more consistent lines because it is doing enhancement of the strokes rather than just whitebalance.



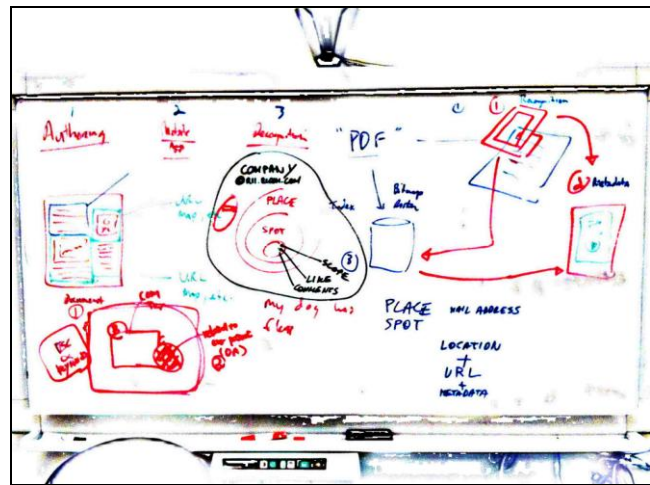
(a)



(b)



(c)



(d)

Figure 7. Good whiteboard image: Original(a), Shareyourboard(b), JotNot(c), Whiteboard Share (d)

Figure 7 was captured under fairly good lighting conditions. Often whiteboards are much darker, and other times there is substantial glare from lights. Figure 8 shows a portion of a whiteboard where there were 3 lights reflected in the board. All three applications lost some of the strokes in the middle of the white spots. Shareyour board, and JotNot kept a little

shadow around some of the spots. Again all the strokes but especially the yellow strokes are most visible with our WhiteBoard Share.

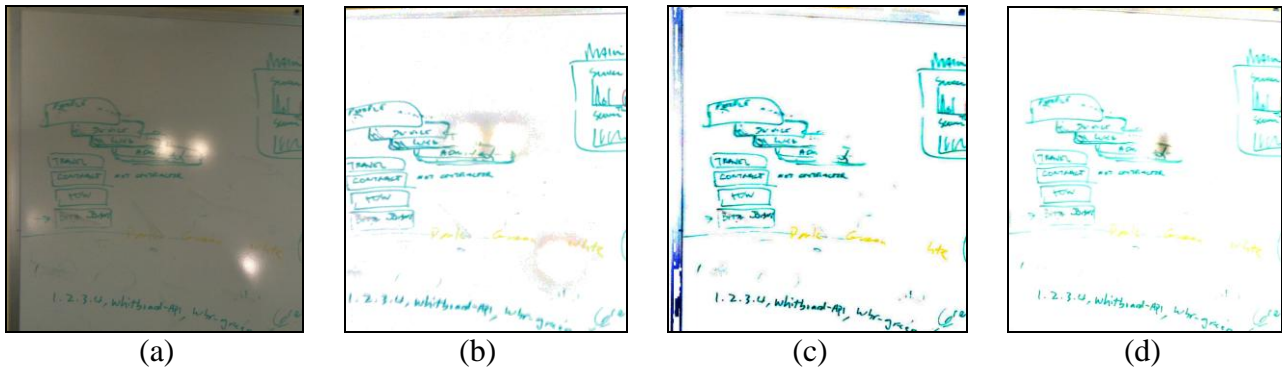


Figure 8. Whiteboard with glare from lights: original(a), Shareyourboard(b), Whiteboard Share(c), JotNot(d)

6. ADDITIONAL ENHANCEMENTS

Many whiteboard cleanup have added perspective correction to the image processing, which we have not yet implemented. Automatic correction risks losing information, so we will probably allow a user to identify a region of interest within an image and crop to that region. After releasing an early version of the application we received surprised feedback that sometimes the 'cleaned up' image was larger than the 'original.' In part this was due to the many different compression settings that could be used when the image was initially captured, but we increased our compression after processing so as not to surprise the user with a larger image. We considered using png files for the output because in many cases they compress better once the background has all been set to 255, and the edges have been enhanced. However, there are some email clients that still read JPEG but not png files, and we did not want to provide an "original" jpeg, that the client could read and an "improved" png that the client could not.

7. CONCLUSION

We have presented an algorithm for enhancing whiteboard images, an architecture for providing an image processing webservice to phone and camera users, and user friendly interfaces. We have had tens of thousands of users for the service. We believe the architecture and user experience pattern are useable in additional image processing applications.

REFERENCES

- [1] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Helene Laurent, and Christophe Rosenberger, Comparative study of background subtraction algorithms, *J. Electron. Imaging* 19, 033003 (2010), DOI:10.1117/1.3456695
- [2] Polyvision Whiteboard Photo Image Capturing Software, <http://www.polyvision.com/>
- [3] MobiTech 3000 LLC, <http://www.mobitech30000.com/>, JotNotScanner Free, version 2.4.0.
- [4] Share Your Board, Android Application, <http://shareyourboard.com/>, up to date on 12/10/2010.
- [5] Ricoh Innovations, Inc., <http://beta.rii.ricoh.com/>, Whiteboard Share, v2.3.
- [6] Z. Zhang and L. He, "Notetaking with a camera: Whiteboard scanning and image enhancement," ICASSP, 2004.
- [7] L. He, Z. Liu, and Z. Zhang, "Why take notes? Use the whiteboard capture system," ICASSP, 2003.