CS231A Final Project - Video to PDF

Yuki Inoue
School of Electrical Engineering
Stanford University
Email: yinoue93@stanford.edu

Jason Liu
Department of Computer Science
Stanford University
Email: liujas00@stanford.edu

Abstract

We introduce a system which takes a video of a document to be scanned as its input and outputs a set of images that correspond to the document pages. These images could then be converted to a single pdf file. Unlike the earlier works, not only does the system recognize each page flips and rectifies the pages, it also applies a series of image enhancements to improve the output image quality. The system also contains a false positive mechanism to remove any duplicate images. As taking a video takes significantly less time than scanning each page at a time using traditional methodologies, this greatly reduces the amount of time and mental pain that a user must go through when scanning documents.

1. Introduction

Scanning pages takes a long time. It would be highly convenient for users to be able to take a video of themselves flipping through a document and have the document automatically converted to a PDF. Such a program will have wide usages. For Teaching Assistants, this could be used to upload paper exams to grading sites. For historians, this could be useful for digitizing primary historical sources. For an average person, perhaps this would be helpful when keeping track of receipts. Simply put, such a program can lessen the time and mental burden put on the users when s/he has to scan a document.

2. Background

2.1. Earlier Works and Our Improvements

As this study has a specific application goal in its mind, it contains many topics that otherwise do not go together. As such, there have been many works that focuses on portions of the process we propose, but only a few tackles the problem as a whole.

One notable example of the few that tackle the problem as a whole is a work done by Lu and Viswanathan [1]. Their work is on an efficient method to digitize the exam papers, which involved taking a video of the exam paper to digitize, and segment/rectify the video. This is very similar to our work, except for a few key differences. First, due to how their application was meant to be used, having duplicates of the same document page is acceptable, though not desired. On the other hand, our application is scanning the documents, so it is unacceptable to have duplicates of the same page. Therefore, after extracting the pages from the video, there needs to be a false positive detection mechanism that can remove the duplicate pages. Another key difference is that in addition to rectifying the images, we also applied a number of image enhancement techniques. Lu and Viswanathan's paper does mention about super resolution, but it also states that the implementation was not practical, as the iterative method used takes a long time to process. We use two image enhancement techniques, and they can both be applied within a practical time period (under half a second).

There have been many papers written on super resolution (SR) [5,6]. Super resolution is a technique to convert a low resolution image to a higher one. However, as low resolution images contain less information than its higher counterpart, this problem is ill-posed and have many solutions. Therefore, SR techniques "recovers" the lacking information by filling in the information that are most likely. Traditional SR techniques use things like notable patterns present in the image, and pixel interpolation to estimate the lost information. Recent papers even introduce machinery such as Convolutional Neural Nets for this purpose [6]. However, most of these SR techniques do not apply to our situation, as normal SR techniques aim to recover images of pictorial scenery, which are much more complex than the images of printed documents. Printed documents tend to have set features such as white background, only use few colors, etc, that makes it easier to estimate how to reconstruct the image. Therefore, SR for printed documents do not need to be able to do nuanced procedures like pattern matching and estimating how much to smooth/blend colors. Rather, it just needs to be able to sharpen the letters and lines in an image,

but at a much greater speed.

Of all the papers regarding SR, a paper by Zheng, et. al. [2] introduces a real time document enhancement technique that we decided to implement as the super resolution step. It blends the edge enhanced image with the original image in a way such that the regions that need edge enhancement receive edge enhancement and vice versa. Though we used different method to determine where the boundaries between the letters and the background are, we followed their lead in implementing SR.

2.2. Contributions

Our contributions towards this problem space involve creating methods that are generalizable to the problem space itself, such as eliminating false positives and hand detection. As stated previously, we also tackled super-resolution in a more focused space of text image enhancement, with which we demonstrate a more reasonable speed of image processing.

3. Technical Aspects

3.1. High Level Description

Video to PDF conversion involves multiple steps. The high level flow chart of the system is outlined in Figure 1. First, given a video input, the system detects which frames contain good images of the document to be scanned. It tries to avoid duplicates of the same page, but it is not absolutely critical to get everything right, as the next step figures out which images are similar to each other, and removes any duplicates. This step also figures out if hands are in part of the image as well. After removing those false positives, the system estimates the bounding box for the document, and rectifies the image using a perspective transformation. We technically have a scanned document at this point, but as video frames tend to be of a low image quality, a number of image enhancements are applied onto the rectified images. First, the brightness of the document is corrected using a method based on least squares. This allows the output image to be of uniform brightness, making it easier to read. Super resolution is then applied to the images to sharpen the boundaries between the letters and the document background. This helps the images to appear to be from a high definition source. After all steps are complete, we are left with a set of images that represent the pages of the document in the video.

3.2. Technical Details

3.2.1 Image Segmentation

The first step in creating a PDF from a video is to automatically detect and extract frames that correspond to document pages. This segmentation problem is mentioned in Lu and Viswanathan's paper as well, and they attempted to solve the problem by taking the affine transform of the consecutive frames and looking at the norm of the affine transformation. We implemented this algorithm, with one extra step. After taking the norms, we found the difference between each norm and the one after, and found out that no matter how much the parameters are tuned, we were always left with many false positives and false negatives.

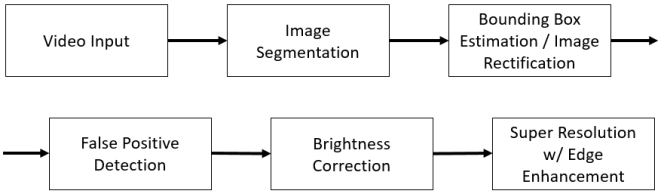

Figure 1: Outline of the proposed system

The problem with the algorithm proposed by Lu and Viswanathan is their underlying assumption: that if two consecutive frames do not contain the image of the same page, then the affine transformation mapping from one frame to another will be large (large here means that the sum of the squared values of the elements of the affine transformation matrix is large). But there is no guarantee that the affine transformation is larger for the frames with different document pages. The only guarantee for such a situation is that there will not be any 'good' affine mapping between the two frames, as two images are not of the same scene. This does tend to mean that the affine transformation matrix is "larger," as it results in the affine transformation trying to unnecessarily morph the frame to make it look like the other, but that is not always the case. For example, it is perfectly possible to have a situation in which the best affine transformation between two frames is the one that does nothing. In this case, this will always produce a false negative, because the affine transformation will be very small.

3.2.2 Using Norm Differences

Lu and Viswanathan decided to threshold the norms, and extract frames determined by this treshold value. We wanted to go for something a bit more robust. And so, after calculating norms, we then went through and found the difference between consecutive norms (by frame order). We then took our array of differences and smoothed it out with a Hanning window. Finally, we extracted frames where the difference changed from negative to positive (indicating a transition from high activity to stillness). The graphs for this process on a sample video can be seen in Figure 2.

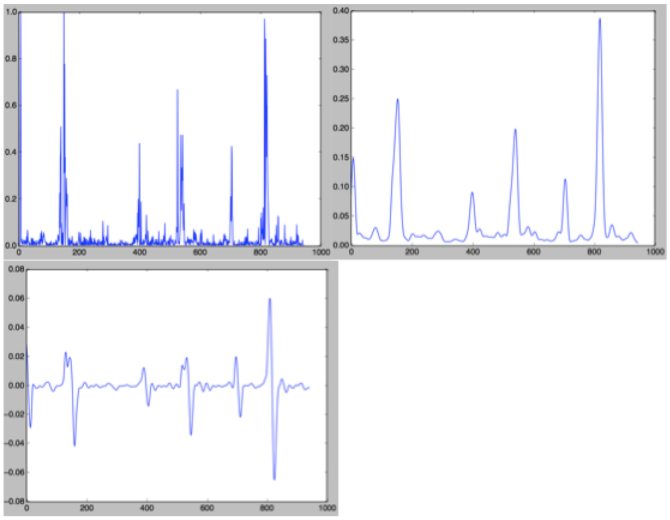

Figure 2: Image segmentation with gradients
(Top-left) Original graph of norms of affine matrix. (Top-right) Graph smoothed with Hanning window. (Bottom) Difference between consecutive points

3.2.3 Hand Recognition

We noticed that a number of images returned from our Image Segmentation Algorithm contained hands. These appeared to arise from two different circumstances. Firstly was a combination of noise, as well as the phenomenon described earlier where obtaining the affine transform. Due to this, we would accidentally extract pictures from scenes where we are actively flipping a page. Second, during our videos, there were moments when the flipper would spend time making small adjustments to a page after it was flipped with their hands. Our segmentation algorithm would determine these frames as being fairly still and extract at least one. Regardless of the situation, the presence of a hand in the outside of the frame was constant. Therefore, we developed a way of detecting hands. This method involves bounding box estimation, which is explained in the next section. We first take the frames extracted from image segmentation, and find the bounding box for each one. Then, we perform the hand-recognition method, which is as follows:
We take the image and consider a 5 pixel border around the pictures that is 10 pixels in from the outside. For each of the three color channels, we subtract consecutive pixels in the direction of each border. For the top and bottom borders, this means subtracting left-right. For the left and right borders, this means subtracting top-bottom. After subtracting, we take the absolute value of all differences. We then collapse each border of depth 5 into depth 1 by adding up each of the five values. Finally, we determine whether there are hands by the presence of more than one high values from any of the borders. Our rationale is that the presence of hands will lead to a high difference in values of at least one of the color channels. We finally take only the images that our algorithm thinks does not contain hands.

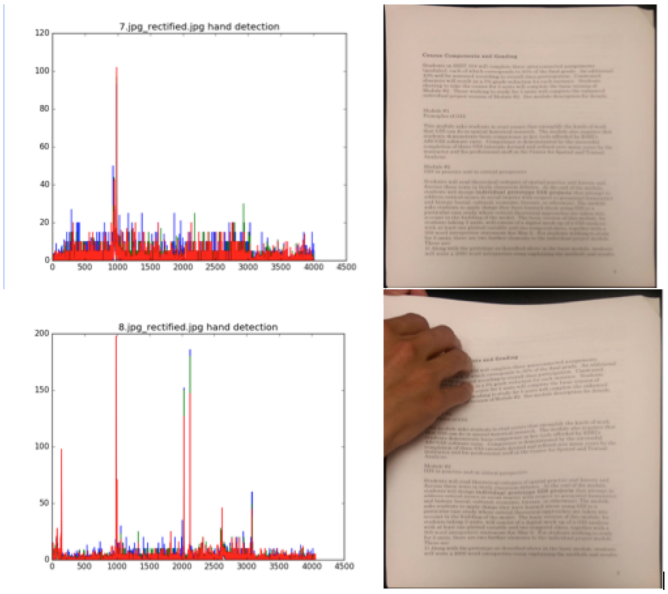

Figure 3: Hand detection comparison
(Top) image without hand. (Bottom) image with hand. Graph: plots of the differences for all three channels for each image

3.2.4 False Positive Detection

After performing hand detection, we moved on to implementing false positive detection using feature point matching. First, feature points for each image are discovered using SURF, and then matched to see how similar they are to each other. We used OpenCV's Fast Library for Approximate Nearest Neighbor (FLANN), which, for each feature point in one image, finds the two closest features points in the other image.

Since feature matching takes a while to run, the size of the images were reduced to a tenth of the original size when extracting and matching features. Example output is shown in Figure 4. The very top image is the feature matching between two different pages, and the middle is for two of the same pages. While the green lines indicating feature matches is few and sparse for the top image, the lines cover the images themselves for the middle image. The difference in feature matches between images of the same page and the images of different pages was around 100 times. The bottom image of Figure 4 shows the feature matching result for tenth of the image size. We see that some of the feature matches survive even after we reduce the size of the image.

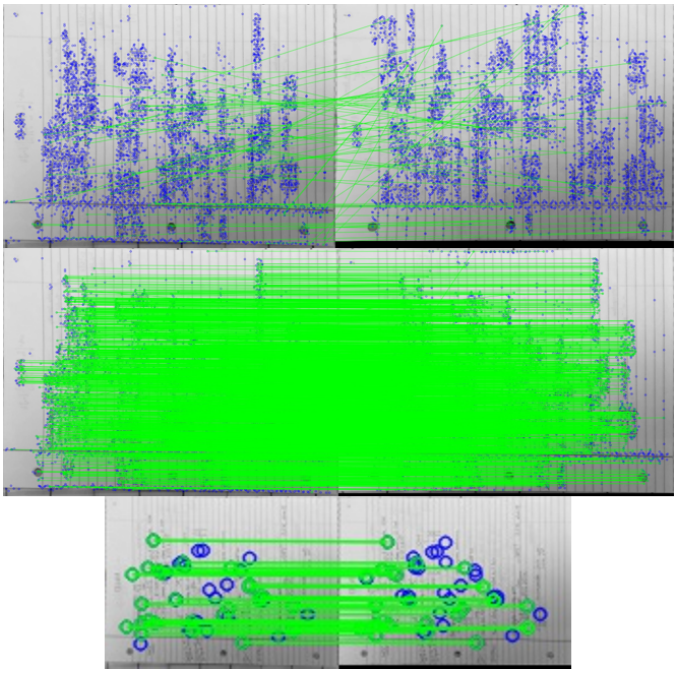

Figure 4: Feature matching between different frames. Full image size, different pages (top), full image size, same page (middle), 1/10 image size, same page (bottom)

We then take the number of the matches found to determine if two frames have changed significantly. Using this, we perform feature matching between consecutive frames and filter our found matches based on a ratio test of its distances between the two closest neighbors. The test is that the distance to the closest feature point must be less than .7 times the distance to the second closest feature point.[7] This is because if FLANN finds a unique feature point match, it is very likely to be closer to our feature point in question than any other potential matches. We normalize by dividing the number of matches filtered by the number of matches found. Then, we threshold the values and extract frames in regions where threshold is low (indicate low number of matches due to a page being flipped).

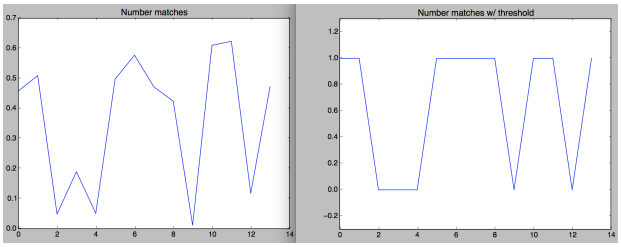

Figure 5: Example of SURF feature matching and with a threshold

3.2.5 Bounding Box Estimation and Image Rectification

Perspective correction is needed to correct the scanned document to be flat. The first step in perspective correction is to figure out the bounding box of a document. Lu and Viswanathans paper addresses this problem by using edge detector and finding the largest bounding box. Unfortunately, as mentioned in the paper, this method is not robust, as it gets easily fooled by nonidealities such as patterned background surface as well as figures within the documents giving false readings.

In order to make the bounding box algorithm robust, it was first noted that the boundary between a document and the background should be defined by difference in pixel values that continues for more than a few pixels. Therefore, edges defined with thin lines should be removed before edge detection. This was realized by using the dilation function of OpenCV, which expands the regions of brighter color. The top right image in Figure 6 shows the dilated result. As one can see, words as well as the figure on the document is nearly removed by the dilation procedure.

After the thin lines are removed, boundary lines are estimated by first applying an edge detection and then applying the Hough Transform. The bottom left image of Figure 6 shows the estimated lines in blue and red. What we noticed while implementing the bounding box algorithm was that certain document edges tend to be more defined than the others, resulting in multiple boundary candidates for one edge all having higher "voting count" than the candidates for other edges. Therefore, lines that are too similar to each other were grouped together and the line with the highest voting count was chosen to represent the group. Two lines are considered too similar, if their slopes are similar, and either the x- or y-intercept is too close to each other. As the output of the Hough Transform in OpenCV returns lines in the form $\rho = xcos(\theta)+ysin(\theta)$, the x- and y-intercepts can be found by $\frac{\rho}{sin\theta}$ and $\frac{\rho}{cos\theta}$ respectively, and θ can be used as the slope of the line. The bottom left image of Figure 6 shows the result, and the red lines are the lines with the highest voting counts for that particular line group.

The last step in finding the bounding box is to choose which four points should be chosen to represent the bounding box. This is done simply by finding the set of line intersections that results in the largest area. Finally, a perspective transformation was applied to rectify the image, as shown in the bottom right of Figure 6.

The case in Figure 6 is ideal, as the background is completely black. How does the algorithm fare with background that are neither black nor monotone? Figure 7 shows the cases in which the background is neither black nor monotone. As one can observe, the algorithm correctly identifies the bounding box even if the background is nonideal.

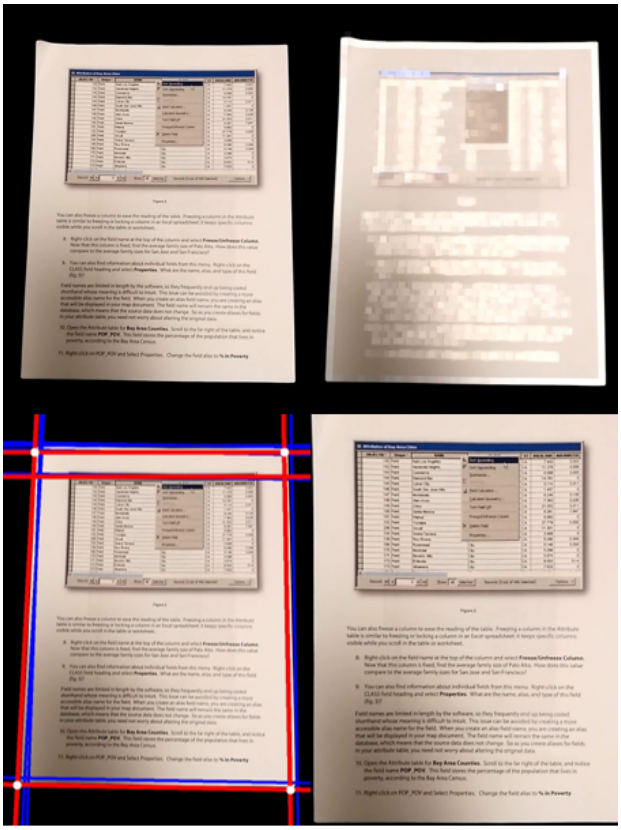

Figure 6: Bounding Box Algorithm in Action - original (top left), dilated (top right), bounding box estimation (bottom left), rectified (bottom right)

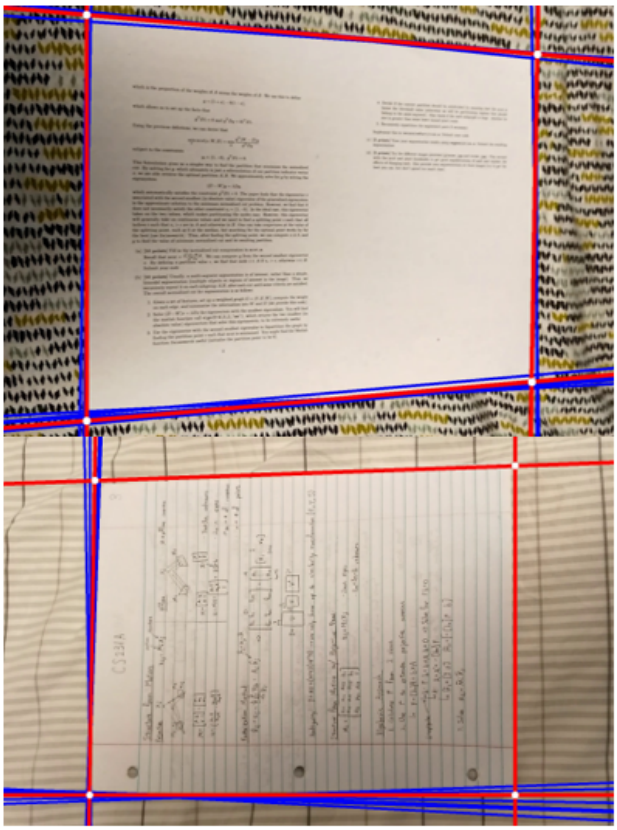

Figure 7: Bounding Box Algorithm for nonideal background

3.2.6 Brightness Correction

One of the problems with using a video to scan a document is that it is very difficult to have a uniform lighting on the document. Also, even if a uniform lighting condition is realized at the beginning, it is hard to maintain such a configuration while flipping through the document and holding a video camera at the same time. So it would be very helpful if the system can correct for an uneven lighting.

One way to tackle this problem is to subdivide the image into smaller chunks and assume that the brightness of a region is roughly the average of the subsection. Though this method is quick and easy, it would perform poorly for the document with many figures and letters, as they would significantly lower the brightness estimate. Also, these subdivisions would have to be big enough to collect enough brightness information for a region, although having too large of a subdivision will end up with ragged transition for neighboring subdivisions. This requires careful parameter tuning, which is not ideal.

So we decided to model the problem differently. Realizing that what needs to be estimated is the gradual change in the brightness value, all we need to reproduce is the low frequency components of the pixel changes. This sheds a light on how we can model the brightness information. Using the Discrete Fourier Decomposition, we can model the brightness profile by combining a handful of low frequency sinusoidal waves as follows:

$$Pixel\,Value = C + \sum_{n=1}^{N} \left(a_{4n} cos\left(\frac{n\pi x}{w}\right) + a_{4n+1} sin\left(\frac{n\pi x}{w}\right) + a_{4n+2} cos\left(\frac{n\pi y}{h}\right) + a_{4n+3} sin\left(\frac{n\pi y}{h}\right) \right)$$

where h and w are the height and the width of the image, and x and y are the coordinates of a point. Now the problem is that of how to set the appropriate coefficient for each sinusoidal components. As it is a linear problem, this can be solved using Least Squares, by setting up the problem in the form $y = Xa$, and solve for a by calculating $a = (X^T X)^{-1} X^T y$. In this context, y is the vector of pixel values at each point, and X is the matrix of sinusoidal values (i.e. $cos\left(\frac{n\pi x}{w}\right)$, $sin\left(\frac{n\pi x}{w}\right)$, etc.) at each sampled point. In other words, the problem can be posed as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} =$$

$$\begin{bmatrix} 1 & cos\left(\frac{\pi x_1}{w}\right) & sin\left(\frac{\pi x_1}{w}\right) & \dots & cos\left(\frac{n\pi y_1}{h}\right) & sin\left(\frac{n\pi y_1}{h}\right) \\ 1 & cos\left(\frac{\pi x_2}{w}\right) & sin\left(\frac{\pi x_2}{w}\right) & \dots & cos\left(\frac{n\pi y_2}{h}\right) & sin\left(\frac{n\pi y_2}{h}\right) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & cos\left(\frac{\pi x_m}{w}\right) & sin\left(\frac{\pi x_m}{w}\right) & \dots & cos\left(\frac{n\pi y_m}{h}\right) & sin\left(\frac{n\pi y_m}{h}\right) \end{bmatrix} \begin{bmatrix} C \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ \vdots \\ a_{4n} \end{bmatrix}$$

One caveat to note is that the sample points for the Least Squares calculation should be representative of the brightness value. Therefore, any pixels corresponding to texts or diagrams on the document should not be used as a sample point. These 'objects' on documents were detected by applying the findContour method in OpenCV, and assuming that anything inside of the detected contours as the unwanted objects. Top right image in Figure 9 shows the result. The bottom left image of the same figure shows the sampled points, and bottom right is the brightness adjusted image.

Figure 8 shows the result for brightness correction. The highest sinusoidal frequency (denoted N in the summation) was chosen to be 4, and the number of sample points was around 800 for the simulation (the number of sample points varies from image to image, as the algorithm avoids sample points that it judges is part of drawings on the document). The top left is the original image, and it gets dilated to remove the unnecessary information such as letters and lines (shown top right). Then the proposed algorithm is used to estimate the brightness mapping (bottom left), and the estimate is used to remove the brightness disparity from the original image (bottom right).

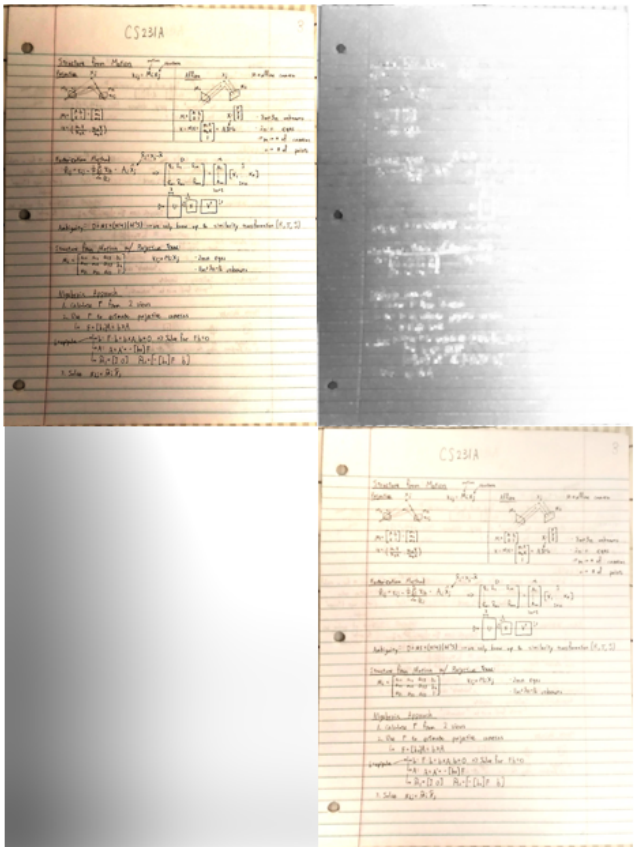

Figure 8: Brightness Adjustment -
original (top left), dilated (top right), brightness estimation (bottom left), brightness corrected (bottom right)

3.2.7 Super Resolution (SR)

Another problem with using videos to scan documents is that videos tend to have low resolution, especially when compared to still images. Therefore, resolution enhancement is needed. In the world of computer vision, such a procedure is called Super Resolution (SR). SR in general is applied for any pictures, to recover details that are not clear or present in a picture.

In the case of document SR, all that is required is to apply edge enhancement to the document, as the difference in the pixel value between the background and the texts should be large, and thus there is no need to try to recreate a gradual change of colors. However, it is not ideal to apply edge detection to the whole picture, as edge detectors will enhance noise in areas of low pixel changes, as can be seen in Figure 10.

Zheng et al. mentions a method that can take the best of both worlds. Revised version of their method is outlined in Figure 11. The essence of the algorithm is to figure out which portions of the image needs edge enhancement (noted α in the diagram), and use that information to blend the original image and the sharpened image. This way, regions that need sharpening will get sharpened, and vice versa. The sharpening region mask was created by finding the difference between the dilated and the eroded versions of the original image. The result is shown in Figure 12. After blending the edge enhanced image and the original images, bi-cubic interpolation is applied to double the size of the image.

We chose to use Teager Filter to sharpen the edges [3]. Teager Filter is a nonlinear filter of the following form:

$$y_{i,j} = 3y_{i,j}^2 - \frac{1}{2}y_{i+1,j+1}y_{i-1,j-1} - y_{i+1,j}y_{i-1,j} - \frac{1}{2}y_{i+1,j-1}y_{i-1,j+1} - y_{i,j+1}y_{i,j-1}$$

Compared to edge enhancement via linear Laplacian kernel, we noted that Teager Filter performed better at enhanc-

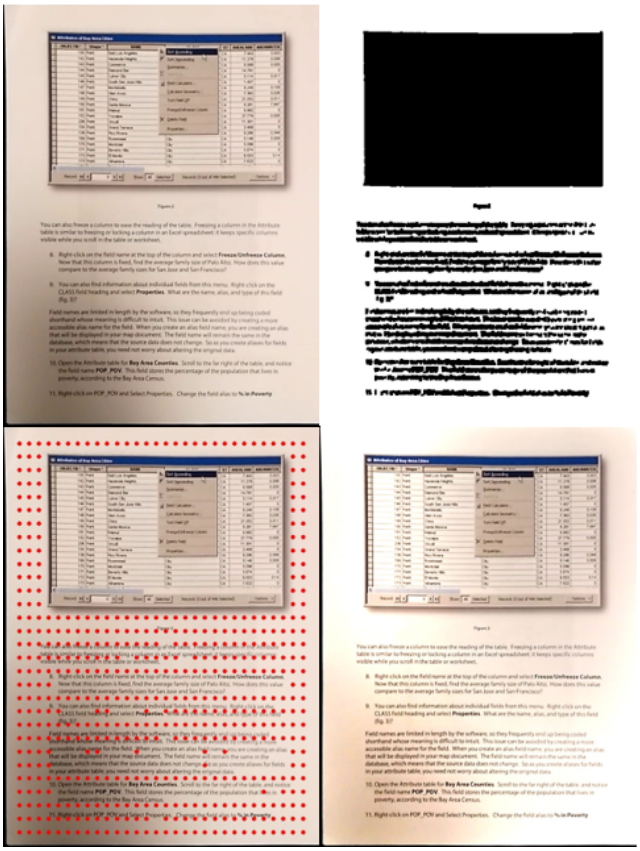

Figure 9: Brightness Adjustment for documents with figures - original (top left), object mask (top right), sampled points (bottom left), brightness corrected (bottom right)

Figure 10: Edge detection applied to uniform region. Before (left), After (right)

ing very large pixel changes. As the pixel difference between the letters and the document background tend to be large, Teager Filter was the filter of choice.

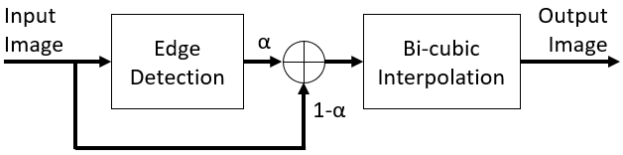

Figure 11: Super Resolution Flow Chart

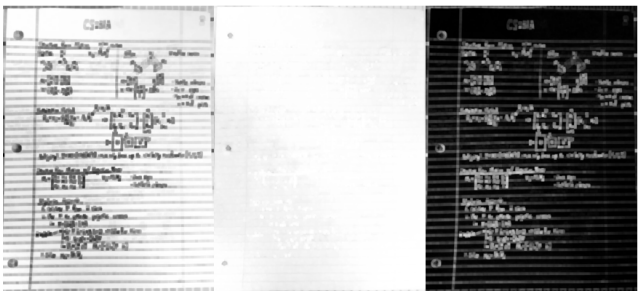

Figure 12: Boundary Finding Algorithm. Eroded image (left) minus Dilated image (middle) yields the boundary mask (right)

4. Experimental Result

Our overall project essentially breaks down into two parts: image segmentation and text resolution. As such, we decided to evaluate them independently of one another.

4.1. Image Segmentation

We recorded six videos of flipping through pages of various documents for a total of 44 pages. After just segmentation, we ended up with 172 frames, representing 41 pages of the 44. Hand removal was applied next, and 83 frames were removed. Finally, false positive elimination was then applied to the 89 frames left, and we ended up with 44 frames (thus having 3 false positives).

4.2. Text Resolution

Figures 13, 14, and 15 show example results obtained from sample images. For each figures, the images from left to right are original, brightness adjusted, and Super Resolution-ed. Lu and Viswanathan also mentioned that their method takes 20 seconds to run their iterative super-resolution algorithm on multiple small images of size 200 x 300 pixels. Our algorithm can process a single image of around 1000 x 900 pixels in less than half a second.

It is hard to create a ”fair” metric on text readability, as it is a vague concept to begin with. In order to define the term more concretely, we defined readability as the number of letters that a black box OCR algorithm can pick up. Therefore, a metric of redability improvement is defined by how many more letters the OCR is able to decode after

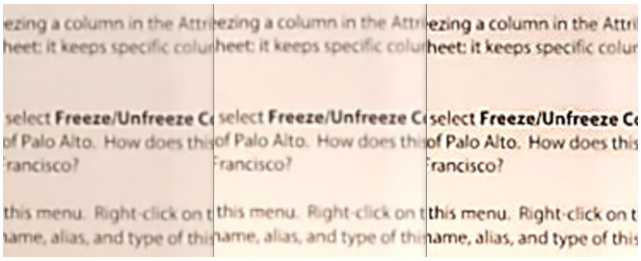

Figure 13: Example Output 1.
Rectified (left), Brightness Adjusted (middle), SR (right)

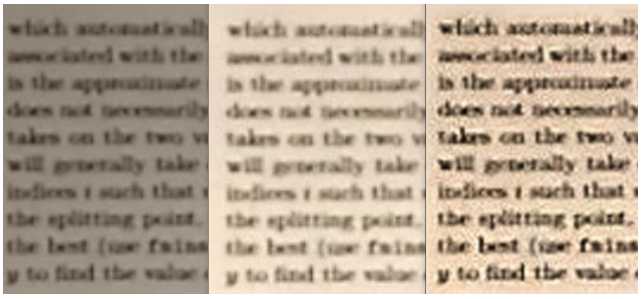

Figure 14: Example Output 2.
Rectified (left), Brightness Adjusted (middle), SR (right)

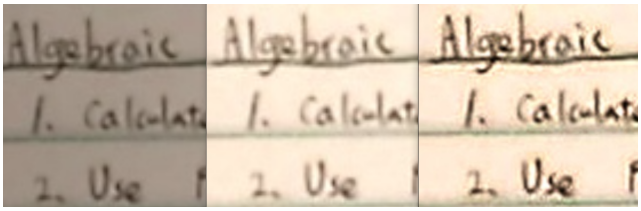

Figure 15: Example Output 3.
Rectified (left), Brightness Adjusted (middle), SR (right)

image enhancements are applied. Because we are using OCR, we decided to limit the scope of this evaluation to images containing printed text, as that is generally easier to discern rather than written text.

Our method is as follows. We took six frames from our videos containing printed text, and found the corresponding pages. We then scanned those pages. Then, we performed super-resolution on each frame. Our evaluation involved performing an OCR on a specified text segment within the scanned page, the frame, and the improved frame. We also ran the test on an image taken from a scanner, which serves as a ground-truth metric. This is needed, as OCR is not perfect, and thus it is quite possible that even if a document is recovered to perfection, OCR could still misread the content.

Document readability is then calculated as the edit distance between the actual text and the text recovered from OCR in each of the three situations. The following table summarizes the result:

	Scanned	Frame	Frame with SR
pg1	0	32	8
pg2	1	54	40
pg3	4	196	29
pg4	48	371	341
pg5	3	86	4
pg6	26	131	113

```
------Text---
The final four weeks of the quarter are devoted to
collaborative research projects enriched through
additional practical and theoretical readings and the
practice of prototyping

------Frame--
The final four week. of the quarter are d..voteil to
collaborative re.earch
projects, enriched liriitigh additional practical and
theoretical reading.,
the practice of prototyptng

------SR-----
The final four weeks of the quarter are devoted to
collaborative research projects enriched through
additional practical and theoretical readings anti the
practice of prototyping,
```

Figure 16: Sample of text comparison demonstrating Super Resolution's improvement

5. Conclusion

The results for image segmentation are positive, and suggest that the methods we have taken are in the right step towards solving this problem. One observation we made was that of the three missing pages, two were the first page of a video. A next step would be to figure some robust way to account for this without hindering false positive rate and hand detection results. For text resolution, the results are promising as well. In all six pages, there was an improvement in OCR capture, suggesting some improvement in readability through Super Resolution. We can also qualitatively observe this readability improvement by looking at the example outputs in Figures 13, 14, and 15. In all three images, a combination of brightness adjustment and SR allows for a better reading experience.

Our pipeline shows that there is much promise in using videos for document scanning. Although the resolution of the output may not be as good as a document that is scanned using traditional methods, some of the low quality constraints can be overcame by using a number of image enhancement techniques such as the ones implemented in this paper. Also, these image enhancement procedures can

be applied in real-time as well. Maybe it is not too far in the future when it becomes a norm to scan documents by using video feeds.

6. Future Works

One of the ways our image enhancement procedure can be made better is to use more than one low resolution image. As the input to the system is a video, we have access to many instances of each page. Therefore, if the program can somehow distinguish which portion of which image contains the clearest subsection of the page, we can use the information to create a new image that is the composite of the clearest portions. Then, this composite image can be used as a new input to the image enhancement procedure for a better result. Another way to improve our work is to create a better method to detect boundaries between letters and the background. The method we devised using a combination of dilation and erosion is only an approximate one, and we noticed many times that the algorithm is not taking advantage of the edge enhanced result as much as it could. Figure 17 shows such an instance. The left picture is the edge enhanced image, and it is noticeably clearer than the SR output. This is due to the fact that the boundary map (rightmost image in the figure) is very dark for this region, meaning that the algorithm decided not to use that much edge enhanced image when blending the original and the edge enhanced images.

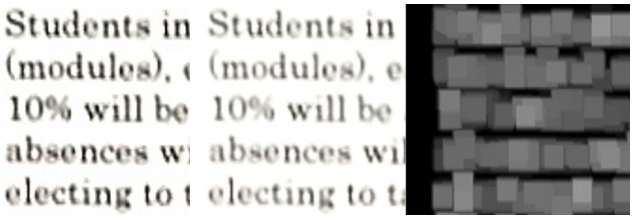

Figure 17: An instance of a weak boundary map
Edge enhanced (left), SR output (middle), boundary map (right). The boundary map is very dim, resulting in a low edge emphasis.

7. Libraries used

OpenCV, Numpy, SciPy, editdistance

8. Code and Data

https://www.dropbox.com/s/ct8qsunhmvvyal1/finalScript.zip?dl=0

9. References

[1] C. Lu and K. Viswanathan, "Exam Digitization for Online Grading." CS231A Class Project, Stanford University, 2014.

[2] Y. Zheng, Xudong Kang, Shutao Li, and Jian Sun, "Real time document image super-resolution by fast matting" IAPR International Workshop on Document Analysis Systems, Loire Valley, 2014.

[3] C. Mancas-Thillou and M. Majid. "Super-resolution text using the teager filter." First International Workshop on Camera-Based Document Analysis and Recognition. pp. 10-16, 2005.

[4] Online OCR onlineocr.net

[5] J. Sun, J. Sun, Z. Xu, and H. Shum. "Image super-resolution using gradient profile prior." In CVPR, 2008.

[6] C. Dong, C. Loy, K. He, and X. Tang. "Image Super-Resolution Using Deep Convolutional Networks." 2015.

[7] M. Muja, d. Lowe. "FLANN - Fast Library for Approximate Nearest Neighbors". 2009