Mobile Page Scanner

Iretiayo Akinola*, Zhouyang Li† and Amit Badlani‡
Electrical Engineering, Stanford University, Stanford, California 94305, USA
Email: *iakinola@stanford.edu, †zhouyuan@stanford.edu, ‡abadlani@@stanford.edu

***Abstract*—In this study, we develop an Android application for scanning text documents to produce output with decent resolution and clarity. Image alignment and stitching techniques commonly used in panoramic mosaic applications are adapted and combined with other image processing methods to stitch-together closely captured images of portions of a page. After initial pre-processing, the images are passed into an image-stitching pipeline which extracts and matches the features in the images, performs bundle adjustment and warps the images together. Some post-processing steps used to reduce noticeable seams in joints include exposure-difference compensation and multi-band blending. The application achieved a high level of resolution in the stitched output image. The results were evaluated using OCR analysis to assess the quality of the output compared to a one-time full capture of the entire page and this revealed that the approach is only marginal better than a carefully captured full-image.**

***Keywords*—image features, warping, bundle adjustment, ransac, image-stitching**

I. Introduction

Why wait to get a dedicated scanner when a mobile application can make you scan your document quickly? Specialized scanners are large effective means of converting hard official documents into usable soft formats but there are cases where there is an urgent need to scan documents and a scanner is inaccessible. On the flip side, the camera capabilities of the ubiquitous hand-held devices have increased tremendously in recent years with improving ability to take high quality pictures. Is there a way of leveraging on this optical capability of the mobile phone to readily scan and generate good quality documents?

This project develops a mobile application which will help the user scan pages of documents using smart phones or tablets. The target is to make the output of the application as good, in quality, as that of a specialized scanner. Relevant image processing techniques are applied to sequentially and simultaneously process the captured portions of a page of the document and the portions are combined to form a whole document with decent resolution and clarity. Image processing algorithms are important in handling concerns like blurry capture, perspective correction, uneven lighting conditions, text alignment etc.

II. Related Work

Image alignment/stitching algorithms evolved over the years with applications in areas such as video stabilization, summarization, and the creation of panoramic mosaics [1]. They basically discover the correspondence relationships among images with varying degrees of overlap and stitch them together using smart post-processing techniques to ensure seamless joints. This project draws from smart image stitching concepts usually applied in image panorama [2]. These entail extraction of features from images, matching images with overlapping features, warping images together in a global coordinate system and a few post processing steps to achieve a seamless stitched image. Here, we utilize these and a few other image processing techniques to stitch together closely captured portions of a page of a document. Previous attempts have been made to implement mobile page scanning applications but the resulting apps have been more suited to momentary capturing and tagging of documents which have little usefulness in the formal settings. In addition, Kumar etal worked on a mobile application which uses video to capture images of printed multi-page documents [3]. Since their application takes the full image of page at once, the clarity of the text depends greatly on the quality of the camera. Even with a highly sophisticated camera, there is a limit to the clarity that would be achievable as the camera would have to be at a considerable distance to capture a full-page. The algorithms utilized by their system are well suited to dealing with document images containing significant text content. While our system is not targeted towards handling multiple pages at a rapid rate, it ensures that the quality of a scanned page is clear enough in a way comparable to the output of a dedicated scanner. The techniques we adopted do not need the page to have large amount of text to perform effectively and the resulting output can be put to formal use.

III. System Overview

This mobile application was implemented on the android platform. First, the user captured overlapping portions of the page. Due to imperfection of the user, the blurred-image-rejection stage is necessary to prevent any blurred image from being accepted into the process. The interface shows the user how many images have been captured and there is functionality to view all captured portion before passing it into the stitch pipeline. Section IV below describes the user interface of the application and the menu options. The captured images are passed into stitching pipeline which applies various image processing techniques such as feature detection, feature matching, image warping to rotate, translate, align and merge the capture portions into one single image. Some post processing is then done to ensure seamless joints and uniform intensity level across the document.

Algorithm
Input: n unordered images

- Extract ORB features from all n images
- Match the features between each pair of images and estimate pairwise Homography matrices using RANSAC.

- Use bundle adjustment to find geometrically consistent Homography for each image.
- Warp each image to global coordinate system.
- Compensate for exposure difference to achieve uniform intensity.
- Perform multi-band blending on all the images to smoothen all the edges to get the final stitched image.

Output: Stitched image

IV. Image Stitching Pipeline

Image stitching pipeline consists of the whole process from getting close snapshots of portions of a page from the user to generating a stitched whole page. Brown and Lowe describe this pipeline very clearly in their paper [2]. In our project, we followed his guide and modified several steps according to the requisition of a mobile application. Each stage of image stitching pipeline is described as follows:

A. Feature Detection

The ORB detector (Oriented FAST and Rotated BRIEF) was used to detect the features in each of the images. The ORB detector was used as an efficient alternative to the SIFT/SURF; it has been demonstrated to be at two orders of magnitude faster than SIFT at the same level of feature identification performance [4]. The inherent variations (like hand-motion, camera auto-focus, lighting, exposure changes) associated with capturing different portions of a page lead naturally to scale and rotation invariant feature requirements for detection. Feature detectors/descriptors with such robust capabilities include SIFT, SURF, ORB, etc. Due to the computation limit of a mobile device, we pick ORB (Oriented FAST and Rotated BRIEF) as the feature for our application. It was used as an efficient alternative to the SIFT/SURF and has been demonstrated to be at two orders of magnitude faster than SIFT even at the same level of feature identification performance [4]. ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. Details about ORB detector can be found in paper[4]. We apply ORB feature detection to each captured image after resizing for efficiency.

B. Image Matching and Pairwise Homograph Matrix Computation

Here, common features in overlapping regions of the captured image are matched and the optimal homography matrices between pairs of images are calculated for all possible pairs. These would be needed in subsequent stages for alignment and stitching of the constituent images.

Feature/Image Matching: Once we have the ORB features for all the images, we can perform feature matching on each pair of images. For a particular feature in image A, we can find its nearest and second nearest neighbor in image B. If its nearest neighbor is much closer than its second nearest neighbor, then the feature in image A and its nearest neighbor in image B is a good match. In this way, we can get all pairs of matched features between each pair of two images.

Estimate of Pairwise Homography Matrix: It has been proved that for all points that lie on the same plane in 3D coordinate system, their projected coordinates onto 2 cameras satisfy a simple homography transformation. To be more specific, if a point's coordinate is (x_A, y_A) in image A and (x_B, y_B) in image B. Then:

$$\begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix} \sim H \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} \tag{1}$$

$\sim$ means the two vectors can differ by a scaling factor. H is a 3 by 3 matrix called homography matrix and has 8 degrees of freedom. To estimate the homography matrix we need 4 pairs of matching features, but actually we have much more. Each group of 4 matched features corresponds to a homography matrix and we need to use RANSAC (random sample consensus) algorithm to pick out the best [5]. The best estimated homography matrix after RANSAC contains n_i inliers and $n_f - n_i$. The higher n_i and the lower $n_f - n_i$ is, the more confident we are that the two images contain overlapping region. Each pair of two images has a homography matrix. The homography matrices estimated with higher confidence than a threshold are valid. The homography matrices estimated with lower confidence than a threshold are invalid. More mathematical details can be found in Brown and Lowe's paper[2].

C. Bundle Adjustment

The valid pairwise homography matrices calculated in the previous step if utilized directly would result in a non-uniform misaligned stitched image with accumulated errors. Bundle adjustment aims to attenuate the error by optimally refine the transformation matrices globally [6]. The idea of bundle adjustment is that if we project each image to the other image by homography transformation, the difference in the overlapping area should be minimized. This results in an error function which sums over all the difference. Given an image i and an image j and the k'th matched feature pair between them, the residual is defined as:

$$r_{ij}^k = u_i^k - H_{ij} u_j^k \tag{2}$$

where u_i^k is the homogeneous coordinate of the k'th feature in image i and H_{ij} is the homography matrix. The error function for all the images is defined as:

$$e = \sum_{i=1}^{n} \sum_{j \in L(i)} \sum_{k \in F(i,j)} (r_{ij}^k) \tag{3}$$

where n is the number of images, $j \in L(i)$ if H_{ij} is valid, (i, j) is the set of matched features between image i and image j. $h(x)$ is Huber robust error function[2]. This is a non-linear least squares problem which needs to be solved using Levenberg-Marquardt algorithm. Once this problem is solved, we can get a global transformation matrix for each image which can be later used for warping. Everything up to this step is called *data registration*.

D. Image warping

The image warping step is quite straight-forward and it entails piecing together all the images using the transformation matrices derived from the bundle adjustment step. All the input images are transformed into a global coordinate system using these matrices. Although bundle adjustment alleviates misalignment issue, we can still see very severe boundary effects as illustrated in the image below (figure 1), borrowed from Lowe's paper. This is partly as a result of possible differences in exposure levels for the stitched images.

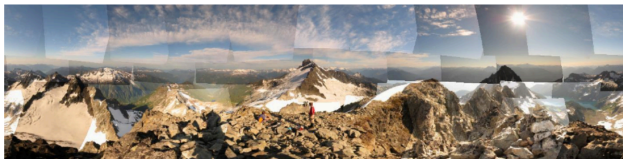


Fig. 1. Bundle Adjusted Image

Figure 2 summarizes the data registration stages and warping. Subsequent stages are classified as post-processing of the now stitched image and is described in the next section of the report.

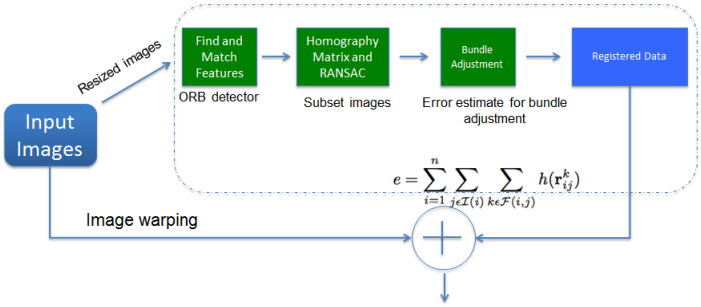


Fig. 2. Image Registration and Warping

V. Post Processing

A. Gain Compensation

Ideally, the overlapping areas between two images should have the same intensity. However, this is not the case in reality. Intensity change that is not observable when capturing can show very obvious boundary effect when aligned together. A very straightforward idea is to adjust the intensity of each image to be rough similar. Mathematically, we are trying to minimize the error:

$$e = \sum_{i=1}^{n} \sum_{j \in L(i)} \sum_{k \in F(i,j)} (g_i I_i(u_i) - g_j I_j(u_j))^2 \qquad (4)$$

where g_i is the intensity gain for image i, and $R(i,j)$ is the overlapping area between image i and image j. To save computation time, it is also good to approximate $I_i(u_i)$ by the average intensity within overlapping area, so we can get rid of the summation over features points. With the averaging version, the optimization problem can be solved in closed form by taking the derivative and set it to zero.

B. Multi-Band Blending

Even after gain compensation, image edges are still visible. Boundary pixels aren't very likely to have the same intensity. In order to smooth the boundaries, we can perform blending of all the images. Blending essentially means assigning weights for different images on the overlapping area. For a particular image, a naive approach of setting weights would be setting 1 to the center pixel and vary the weights to zero at the boundary. Boundary effect is almost eliminated by the approach. However sine blending is somewhat like a low pass filter, high frequency component is also filtered out. To preserve the sharpness of images, we can perform multi-band blending. The idea of multi-band is to blend over a short range for high frequency component, and over a large range for low frequency component. It is not necessary to do a Fourier transform to separate frequency component. The difference of Gaussian images is a good estimation. Applying Gaussian filter recursively, frequency at higher bound is gradually reduced. Mathematically, multi-band blended image is calculated as:

$$I_{k\sigma}^{multi}(x,y) = \frac{\sum_{i=1}^{n} B_{k\sigma}^{i}(x,y) W_{k\sigma}^{i}(x,y)}{\sum_{i=1}^{n} W_{k\sigma}^{i}(x,y)} \qquad (5)$$

where difference of Gaussian $B_{(k+1)\sigma}^{i}(x,y) = I_{k\sigma}(x,y) - I_{(k+1)\sigma}(x,y)$, $I_{(k+1)\sigma}(x,y) = I_{k\sigma}(x,y) * g_{\sigma'}$, $W_{(k+1)\sigma}(x,y) = W_{k\sigma}(x,y) * g_{\sigma'}$. I_1 is the original image, and $W_1$is the nave weights. The weights expand larger and larger at low frequency component. After gain compensation and multi-band blending, image in figure1 looks like this:

VI. Using the Mobile Application

Here we give a short description of how to use the current version of the application implemented on the android platform.

- First, the user clicks on the camera button to activate/turn on the camera of the mobile device.
- Next, the user takes close captures by pointing the camera at portions of the page and clicking the capture button. Each image captured is pushed onto the image stack and the displayed length of the stack is updated.
- There is an optional step of selecting the gallery button to view the captured images and ensure all portions of the document has been captured with minimal noise/blurriness.
- Also, there is an optional clear button which can be used to remove all captured images from the stack to restart the process.
- Finally, the stitch button is what passes the captured images into the image-stitching pipeline.

A feedback on the screen of the phone shows the progress as the images are passed from one stage of processing to another. The resulting stitched image is displayed to the user and saved to a location on the device. Figure 3 shows the user interface and summarizes the steps described while figure 4 shows the feedback screen that shows the progress during stitching.

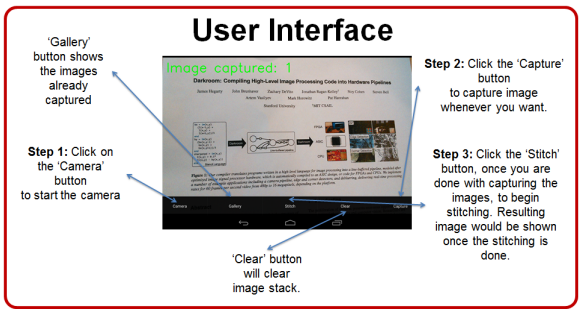


Fig. 3. User Interface for the Android Application

Fig. 4. Progress Feedback User Interface

VII. Experimental Results

A. Observations

The application when tested on the common A-4 sized paged documents typically requires 4-6 close captions of different overlapping portions to cover the entire page (illustrated in figure 8). These images are passed into the image stitching pipeline which usually takes less than 2 minutes to process 4 to 8 images and this meets real time requirement. The stitched result (figure 6) looks quite good when the input images are captured clearly. The resolution and clarity achieved is better than what can be obtained by capturing the image of whole page at once.

A closer look at the specifics of each stage in stitching pipeline reveals the following details:

Image Resize: The size of the captured images vary for different devices/camera settings; some tabs might capture images with larger size than some phones. And the number of features extracted varies with the image size. Hence there is a need to manage the size of the image to minimize the cost

Fig. 5. Multiple Input Images

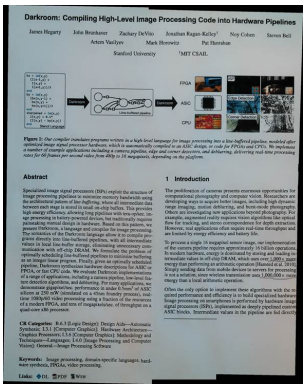


Fig. 6. Final Output Image

of computing large number of features. For our application, we defined the scale as $\frac{min(width * height, 0.6million)}{(width * height)}$ to make sure that we don't calculate too many features.

Feature Extraction: The number of features extracted from each captured portioned varied with the amount of text in the area. This varied from 1100 features in portions with minimal text to 1800 in highly text-dense regions. This is inline with the fact that more text means more edges detected; in other words, more features detected. For the test images shown in figure 8 The number of orb features captured on each image is around 1500(1487, 1478, 503, 1528, 1530, 1516). The two regions with the most text-densities have the highest numbers of features.

Image Matching: As pairs of images are compared, common features in areas of overlap are matched and used to compute the homography matrix. The number of common features would depend on the size of the overlapping area

and the resulting matrix depends on the specified RANSAC parameters. Sometimes, no match is found between a pair of images either because the overlapping area is too small or one of the images is blurred and has poorly identified features or it may be that the RANSAC parameters are too stringent.

Bundle Adjustment: There's always a trade-off between the quality of output of the stitched image and processing time. Some artifacts were noticed in the result stitched image. Surprisingly, these artifacts do not lie along the stitched edges but cuts through the centre of one or two of the constituent captured images. We suspected that this might be as a result of some lowered functionality in the bundle-adjustment pipeline. This artifact can be significantly reduced when some parameters of the bundle adjustment step are tweaked. However, this comes at a speed cost as the stitching process now takes about 50% more time than the initial normal duration.

Comparison of Resolution of Output with Fully captured image. Figure 7 shows a zoomed-in image of the output of the scanning application versus the zoomed-in portion of a single camera capture of the entire page. The difference in resolution performance is amplified and we can see the advantage of having closer capture of portions versus taking the image of the entire page at once. The output of these two methods were passed through an OCR software (ABBYY) to see how well the soft documents perform with respect to character recognition. The results showed that most of the words in both documents were well identified by the OCR. Despite perceptual qualitative assessment suggesting a superior clarity and resolution of the result of the application, both images performed comparatively well with the OCR identifying over 95% of the words on the page. The unidentified words were missed in both documents. More stringent test might be needed to determine the better of the two images as it is very possible that one of the images depend more on the sophisticated intelligence of the OCR software currently used for the analysis.

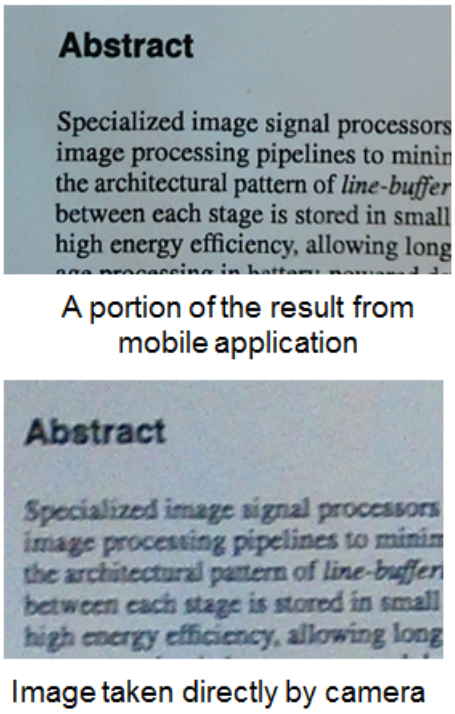
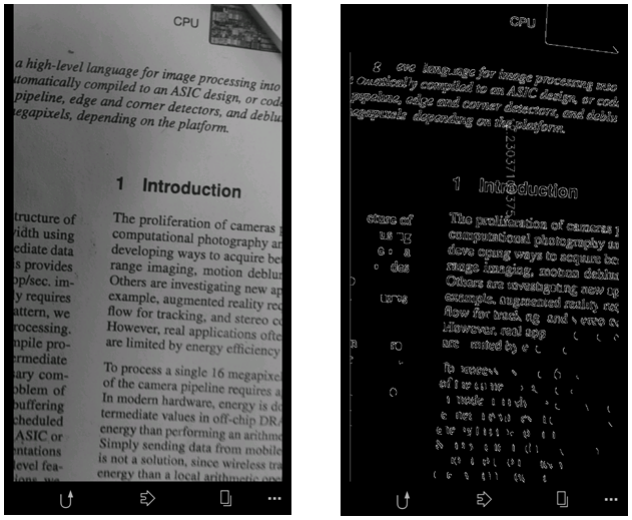


Fig. 7. Stitching Output Image versus Single Full Capture

B. Blurred-Image-Rejection Experiment

In trying to extend the application for video based scanning, we investigated an approach for identifying and rejecting blurred text images. The approach was inspired by the fact that text regions tend to generate a lot of keypoints because of the sharp edges of the letters; however these sharp edges/gradients get flattened when the image is blurred. Edge-detectors (canny/sobel) when applied to an image generate a binary edge-map whose number of white pixels depend on the number of detected edges. The fraction of white pixels in the edge-map of a document could be an indicator of how blurred or otherwise the page is. A threshold can be experimentally picked for the minimum percentage of edge pixels required for an image to be classified as a clear image. A blurred image would have less sharp gradients (edges) detected by the highly selected edge detection parameter used. Tuning the threshold make this quite effective after in a way that no blurred image makes it through the rejection filter. The short coming of this approach is that it might record some false negatives by classifying captured portions with significantly low amount of text/image as blurred. However, experiments revealed that a clear capture of page with a moderate amount of text is sufficient to surpass the set threshold and this is largely the case for the purposes of document scanning. The following figures illustrate how the edge-detector responds to varying degrees of blurry images.



VIII. ACKNOWLEDGEMENT

Many thanks to Sam Tsai (the project mentor) and David Chen for giving very useful and insightful comments/feedback in the course of the project. Also, we appreciate Professor Girod and the entire teaching staff of EE368 for putting up a very exciting and applicable course.

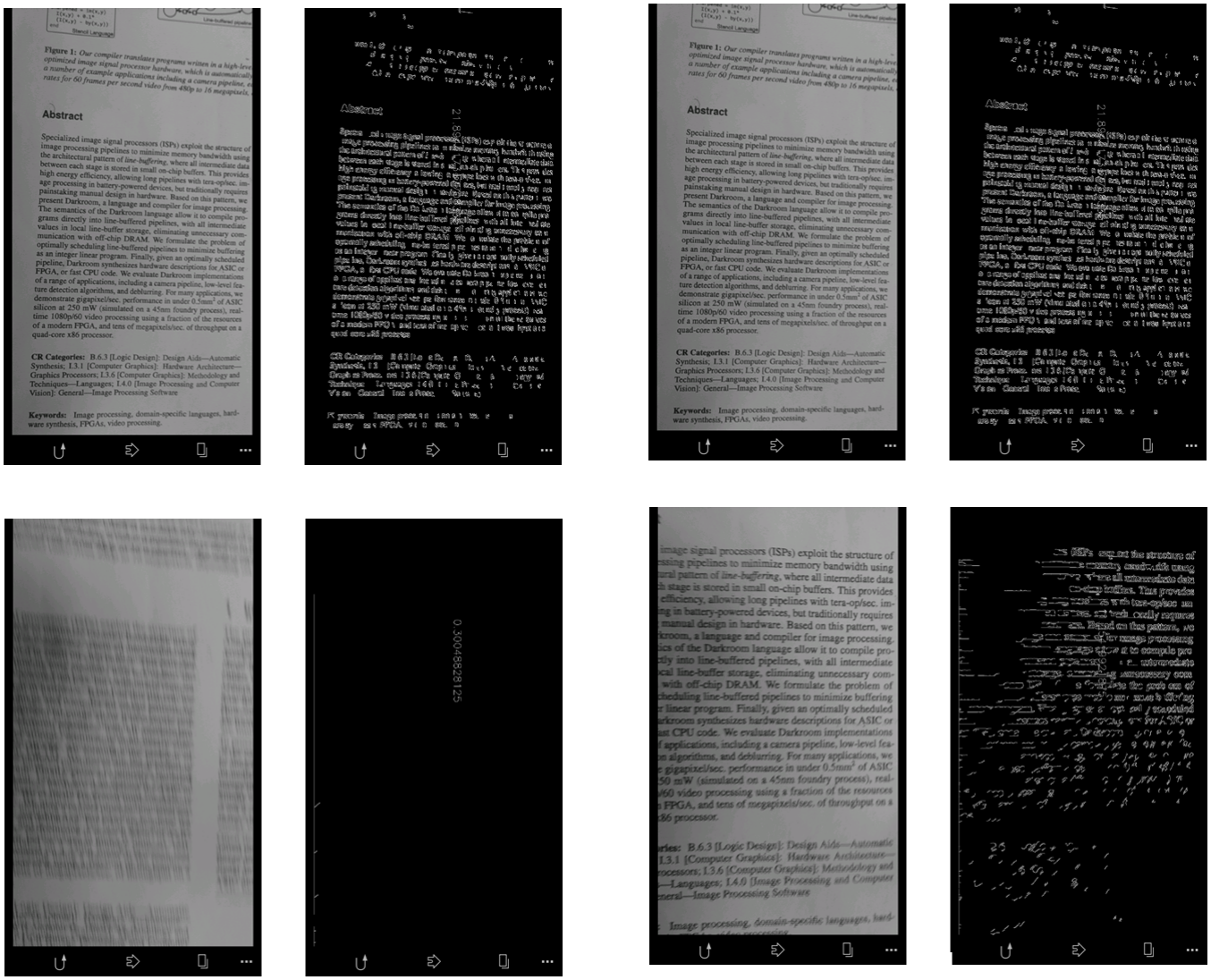


Fig. 8. Edge-Map generated for varying degrees of Blurriness

IX. Future Work

The current implementation of this project is very simple to use but involves a lot of user interaction in the capture of the images. With slight modification to reject blurry image and determine a suitable sampling rate, this application can be made to be a video based page scanning; the camera can automatically/continuously capture images at intervals while the user simply moves the device across the page being scanned. We already looked at a possible blurred-image-rejection technique, described in this paper, which can be used select the non-blurry images that would serve as input into the image stitching pipeline. While the current version of the project is functionally effective, this proposed modification would make it more user-friendly.

X. Conclusions

The project's main aim of using a mobile device to capture high-resolution images of text documents was clearly achieved. Using smart image processing techniques, the stitched output image had much higher resolution compared to the full image of the document captured at once. In fact, the output compares favorably with the output of a dedicated scanner.

References

[1] R. Szeliski, "Image alignment and stitching: A tutorial," *Found. Trends. Comput. Graph. Vis.*, vol. 2, pp. 1–104, Jan. 2006.

[2] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, pp. 59–73, Aug. 2007.

[3] H. D. Jayant Kumar, Raja Bala and P. Emmett, "Mobile video capture of multi-page documents," 2014.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, (Washington, DC, USA), pp. 2564–2571, IEEE Computer Society, 2011.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[6] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, (London, UK, UK), pp. 298–372, Springer-Verlag, 2000.