

Mobile Scanner: Document Segmentation and Object Removal by Exemplar-based Image Inpainting and Spectral Regularization

Jiafu Wu

Department of Electrical Engineering
Stanford University

jiafuwu@stanford.edu

Wenqi Hou

Department of Electrical Engineering
Stanford University

wenqihou@stanford.edu

Abstract—This report proposed a pipeline for a advanced mobile scanner, which is an application of object segmentation and object removal techniques. We introduced various techniques in the fields of object removal including exemplar-based inpainting as well as spectral regularization, followed by the document segmentation, which is composed of edge detection, contour search, and perspective transform.

I. MOTIVATION

Nowadays, people need to process various images in their daily life, such as photos, receipts, written homeworks, printed documents, and business cards; nevertheless, in this digital age, an important method to process and store those paper documents is to capture them with cameras and store them via digital photos. In order to enhance readability, efficiency, and aesthetics, often people needs to crop the important document part and remove unnecessary and unaesthetic background. Many mobile apps are made to help users automatically do that such as Evernote Scannable, iScanner, Tiny Scanner, etc. However, we can still include some features on top of them like defect/undesirable object removal as well as noise reduction. In this project, we would like to achieve the scanner functionality of automatic document/photo extraction via edge detection and contour, and properly obtain a top-down, bird eye view image via perspective transform.

II. RELATED WORK

There has been significant development of object removal techniques in the past 20 years. Previ-

ously researchers proposed texture synthesis, which produced repetitive two-dimension textual patterns with introduced stochasticity to fill the occluded regions[1]. Out of these techniques, a particular effective one is exemplar-based techniques which introduce sampled and copied color values from the source [3]. However, while they might work well to reproduce consistent repetitive textures, they have trouble to recover real-world scenes which often have non-repetitive textures.

In order to address this image-filling issue for real-world scenes, researchers proposed inpainting techniques to fill holes in the images with isophotes, which are linear structures propagating in the target region via diffusion [4]. These methods work finely as restoration techniques to remove small defects such as scratches and speckles, but blurs would be introduced and become noticeable when filling larger regions.

Criminisi, Perez, and Toyama combined the exemplar-based and inpainting techniques into one single algorithm[2], which is a hybrid design aggregating the advantages of both methods. The linear structures using inpainting were used to determine the pixels values while the exemplar-based texture synthesis was used to determine the fill order of the color.

Another important part of this project is edge detection. By definition, edge detection identifies points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set

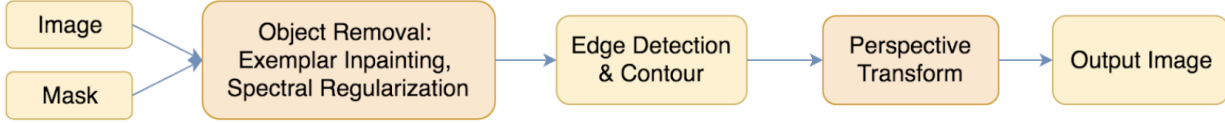


Fig. 1: The pipeline of our object removal and document segmentation.

of curved line segments termed edges. By doing so, we can find a set of connected curves that indicate the boundaries of objects, and further locate the objects.

Most of edge detection methods can be grouped into two categories [5], search-based and zero-crossing based. The search-based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. As a preprocessing step to edge detection, a smoothing stage, typically Gaussian smoothing, is usually applied.

III. INTRODUCTION

The goal for this project is to implement a refined scanner algorithm which removes irrelevant objects or occlusions, recognizes the area of interest in the image, and transforms the area to the center of image. Based on that, we proposed a pipeline shown in figure [1].

As we can observe from the pipeline, first of all, we have the input image and need to manually construct a mask that corresponds to the objects to be removed. We feed the image as well as the mask into the object removal module.

We can apply various object removal methods on it to exclude irrelevant objects in the module. We compared two methods: exemplar-based image inpainting and spectral regularization. We examined and compared their performance in the following

section. The output of the module should be able to remove the designated objects specified by the mask.

Then the output of the object removal module is fed to perform edge detection and the contour. We are going to apply an edge detection method to separate the area of interest from its background. Then based on the detected edges, we would extract the contour of the document and we are going to utilize perspective transform on the area which provides the content of the document.

IV. ALGORITHMS

As we specified in the introduction section, the input image and the mask are fed into the object removal module, and the module can use either the exemplar-based image inpainting or the spectral regularization. Then we have the document segmentation parts which consist of the edge detection, the contour search as well as the perspective transform. In this section, we would discuss details about the exemplar-based image inpainting, the spectral regularization, as well as the document segmentation mechanism.

A. Exemplar-based Image Inpainting

The main task for object removal is to fill the gap after removing the object. In the field of object removal, there are mainly two kinds of algorithms: texture synthesis algorithms and inpainting techniques. Texture synthesis will find and copy a similar patch to the current area, so that to preserve texture continuity. The method, however, usually fails to preserve the boundaries between image regions. On the other hand, the inpainting algorithms work well on preserving edges and boundaries, but they will introduce some blur to the image, especially when filling large areas.

The method that we are using, Exemplar-Based Image Inpainting, combines the two traditional

methods, and gives results that have advantages from both sides. It is also relatively efficient.

The Exemplar-Based Image Inpainting is based on two key ideas.

The first one is that Exemplar-based synthesis is good enough for the task. That is to say if we correctly search for a patch in the known area of the image, we can find a reasonably good patch to fill in the empty space, patch by patch.

The second idea is that the order of filling is a really important factor of the overall result. If we can determine a right order to fill the space, the image result will be much better than a badly selected order. In fig. 2, image *a* is the input image, and *b*, *c*, *d* and *b'*, *c'*, *d'* demonstrate two different orders of filling the empty area. In the first order, we use an onion-peel approach, which leads to significant artifacts. In the second order, we first fill the area which is on a linear structure, and then move on to the edges where we have a lot of known pixels around. We can see that the second approach gives a far better result.

The basic idea of the algorithm is that we want to fill known areas first and we want to encourage linear structures. This is the guideline of what the filling order should be. After selecting the filling order, we go on to find a similar patch in the known area, and copy the content of the patch to the target, and update the values accordingly. The outline of the algorithm is as follows:

- 1) given an empty area, first select a patch *p* on the contour such that: *p* is surrounded by relatively more known pixels, and *p* is on the continuation of a linear structure.
- 2) find a similar patch *q* in the known area, such that the difference between *p* and *q* is minimum.
- 3) Plug *q* to *p*. And update values around *p*.
- 4) Repeat the above steps, until the empty area has all been filled.

Before getting into the details of the algorithm, let's define some notations. In Fig. 3, *I* is the whole image, Φ is the known area, and Ω is the empty area we need to fill. $\delta\Omega$ is the front, which is the border between Φ and Ω . *p* is the target pixel, and Ψ_p is the patch that centered at *p*. n_p is the normal

to the front $\delta\Omega$, and $\Delta I_p^\perp$ is the isophote (direction and intensity) at *p*.

In the algorithm, each pixel also maintains a confidence value *C*, which reflects our confidence in the pixel value, and which is frozen once a pixel has been filled. During the course of the algorithm, patches along the fill front are also given a temporary priority value *P*, which determines the order in which they are filled. Then, our algorithm iterates the following steps until all pixels have been filled:

- 1) Extract the manually selected initial contour $\delta\Omega^0$
Repeat until filled:
- 2) Identify the contour $\delta\Omega^t$
- 3) Compute priorities $P(p) = C(p)D(p)$ for every *p* on the contour $\delta\Omega$, to decide which patch to fill first.
- 4) Find the *p* that has the maximal $P(p)$ as the target pixel.
- 5) Find a patch *q* in the known area Φ that minimizes $d(\Psi_p, \Psi_q)$
- 6) Copy image data from Ψ_q to Ψ_p .
- 7) Update $C(p)$.

The confidence term *C* is given by

$$C(p) = \frac{\sum_{q \in \Phi_p \cap (I - \Omega)} C(q)}{|\Phi_p|}$$

and the data term *D* is given by

$$D(p) = \frac{|\Delta I_p^\perp n_p|}{\alpha}$$

It is easy to see from the confidence term $C(p)$'s definition that it is basically the average of the confidence scores of the pixels surrounding *p*, which is essentially how sure we are about the area. Of course we want to deal with the parts that we have more confidence about first.

As for the data term *D*, it is a function of the strength of linear structure hitting the front $\delta\Omega$ at each iteration. This factor is of fundamental importance in our algorithm because it encourages linear structures to be synthesized first, and, therefore propagate securely into the target region. Broken lines tend to connect.

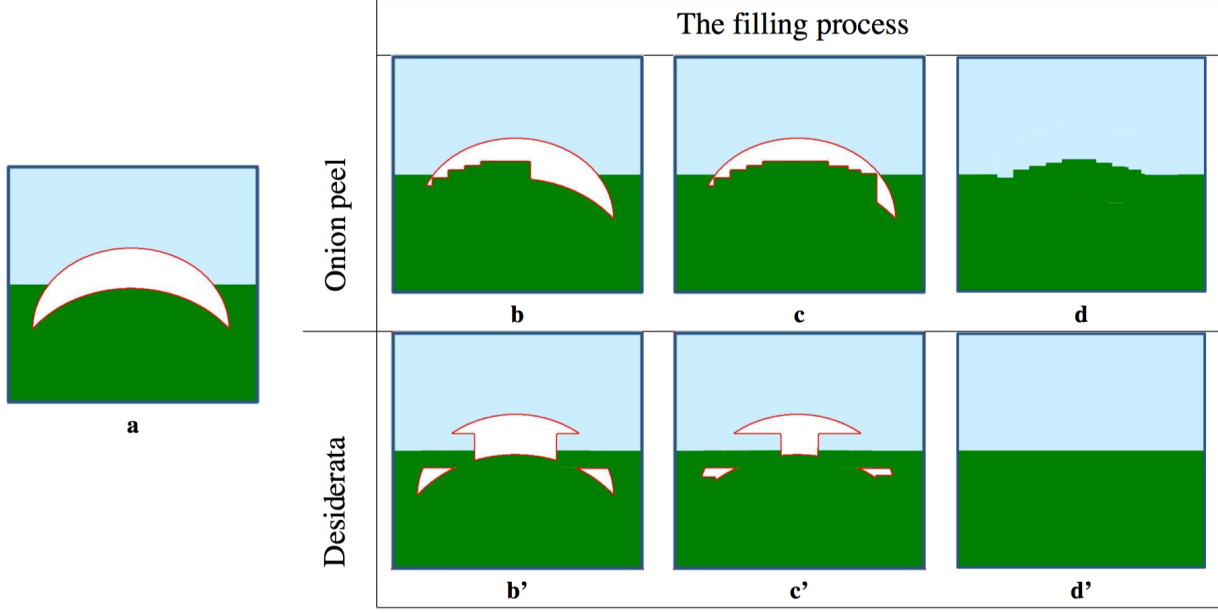


Fig. 2: The importance of filling order

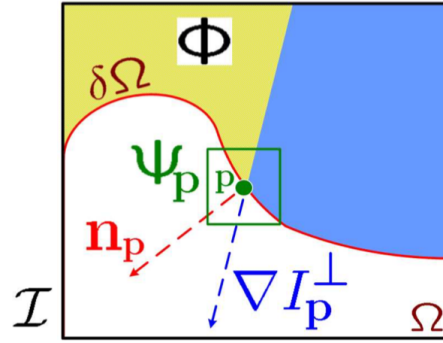


Fig. 3: Notations

These two terms together provide support for the essential ideas we introduced, which is encouraging linear structures and filling known areas first.

B. Spectral Regularization

Spectral regularization is a convex optimization method to solve for the matrix completion problem that minimizes the square reconstruction error between the completed image and the original image, along with a L1 regularization of the norm of the completed image.

In order to solve this convex optimization problem, we use an iterative approach that repetitively

replaces the missing elements with the newly computed values using the Single Value Decomposition.

1) *Terminologies:* Before we introduce the algorithm, we would like to introduce some terminologies to simplify our explanation.

For a matrix X with size m by n , let Ω to be a matrix of the same size and it specifies the indexes of the entries in the observable region. Then we specify the variable $\Omega^\perp$ to be the set of indexes of the entries in the unknown region.

For a given matrix X with unknown indexes, we

can define a matrix notation P_Ω as follow:

$$P_\Omega(X)(i, j) = \begin{cases} Y_{ij} & \text{if } (i, j) \in \Omega \\ 0 & \text{if } (i, j) \notin \Omega \end{cases} \quad (1)$$

That is, for those entries in the region Ω , we pick the original entries in the regions, else, we set them to zero. Based on the notation of $P_\Omega(X)$, we can have $P_\Omega(X) + P_{\Omega^\perp}(X) = X$, which means we can combine the indexes of known and unknown regions and obtain the full matrix X .

2) *Soft-thresholding*: We can propose an optimization problem, with an m -by- n matrix W of rank r :

$$\text{minimize}_Z \frac{1}{2} \|W - Z\|_F^2 + \lambda \|Z\| \quad (2)$$

We can obtain the solution of equation [2] to be $\hat{Z} = S_\lambda(W)$, where the definition of S_λ is a modified version of SVD with regularization:

$$S_\lambda(W) = UD_\lambda V' \quad (3)$$

where SVD of W is $W = UDV'$, $D = \text{diag}[d_1, \dots, d_r]$, and $D_\lambda = \text{diag}[|d_1 - \lambda|, \dots, |d_r - \lambda|]$

This notation S_λ is called soft-thresholding.

3) *Soft-Impute*: Based on the notation and soft-thresholding, we can propose a matrix completion problem as follow:

$$\text{minimize}_Z \frac{1}{2} \|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda \|Z\| \quad (4)$$

Then we can iteratively use soft-thresholding to obtain a more and more refined solution. Such iterative approach to solve for the optimization problem is called soft-impute.

- 1) Initialize target image Z with $Z^{old} = 0$
- 2) For a sequence of regularization coefficients $\lambda_1 > \lambda_2 > \dots > \lambda_k$:
 - (a) Repeat:
 - i. Compute $Z^{new} \leftarrow UD_\lambda V'(X + Z_{\Omega^\perp}^{old})$
 - ii. If $\frac{\|Z^{new} - Z^{old}\|^2}{\|Z^{old}\|^2} < \epsilon$: assign $Z^{old} \leftarrow Z^{new}$
 - (b) Assign $\hat{Z}_{\lambda_k} \leftarrow Z^{new}$
- 3) Output the sequence of solution $\hat{Z}_{\lambda_1}, \dots, \hat{Z}_{\lambda_k}$

C. Segmentation

1) *Edge Detection And Contour*: After the object removal part, we go on to segment the document from the image. We employ the Canny edge detector to perform the edge detection. First, we apply a Gaussian filter to denoise the image. Second, we convolve the image with the derivative filters to find the intensity gradients of the image. Third, in order to remove the unwanted spurious response, we apply non-maximum suppression for edge thinning. Fourth and the last, we introduce thresholds to determine potential edges and apply thresholds again to eliminate all the other relatively weak edges. That is how we obtain the edge of the input image.

Then we need to look for the contour of the document based on the detected edges. Essentially, each continuous edge with the same pixel value is treated as a contour, that is why we would need to perform the previous edge detection in order to obtain the binary image of the edges. And we would store each contour with their outermost boundary points instead of all the boundary points in order to save memory.

For segmenting the document, we take the following heuristic approach: we assume that the document is a quadrilateral which has the largest area among all the other quadrilaterals found in the image. From all the contours found in our previous method, we only select those with four points, which we consider a quadrilateral. Then among all the quadrilaterals, we select the one with the largest area. This would be the area where our document resides.

2) *Perspective Transform*: The final step in our pipeline is the perspective transform. Because our area of interest is the document, and we can discard the other area, we performed a perspective transform that maps the four corners of the document into the four corners of the overall image. In this case, we have our source points X , which is a matrix of four columns with the coordinates of the document corners, as well as the destination points Y , which is a matrix with the coordinates of the image corners. To introduce intercept, we append a row of ones into both matrices, which now became

3 by 4 matrices. Now essentially we are solving for the 3 by 3 matrix A , which is a mapping account for:

$$Y = AX$$

After solving the matrix A , we can apply such transform onto all the points within the boundary of the document. At the end, we can obtain our final image by sampling the values from the perspective transform result.

V. ANALYSIS AND EVALUATION

A. Exemplar-based Image Inpainting

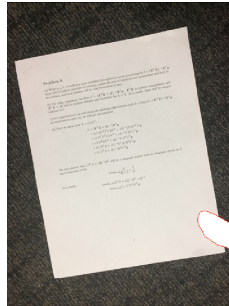
Fig. 4 shows the process of using Exemplar-based Image Inpainting algorithm. We can see in (b) that the algorithm is first trying to deal with the edge of the paper, which is a linear structure. Then in (c) and (d), we notice that once a little sharpness appears, it is immediately dealt with, since the algorithm will give priorities to this kind of pixels that are surrounded by known pixels. Finally, in (e) and (f), the algorithm finishes with the edge of the paper, and moves on to fill the two parts on the sides. The process's filling order is just what we expected.

We have run a variety of experiments, and included some most representative results for the Exemplar-based Image Inpainting method in Fig. 5 side by side.

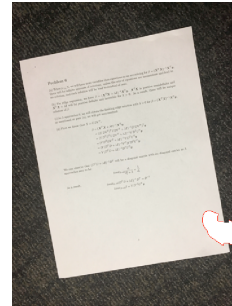
For the first two sets of images in Fig. 5, the Exemplar-based Image Inpainting algorithm works well. The edges of the papers have all been restored, and they look sharp. This fact is really essential for the following steps to find the edges and conduct segmentation and perspective transform. On the other hand, the texture information has also been restored nicely. There is still a little bit of artifact, but it won't matter as we will increase contrast later on.

For the third set of images, the Exemplar-based Image Inpainting algorithm doesn't work so well. This is due to the fact that the algorithm emphasizes more on the linear structures than the curves. So curves like the wire in the image can't be restored as well as the linear structures.

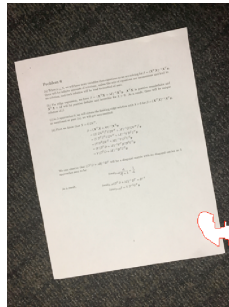
To sum up, the Exemplar-based Image Inpainting algorithm can replace the objects in image with



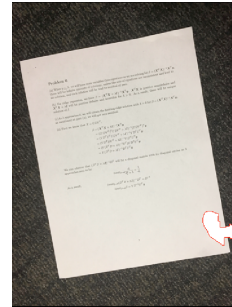
(a) Original masked image



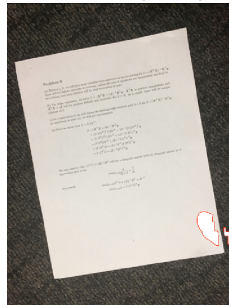
(b) Processing image 1



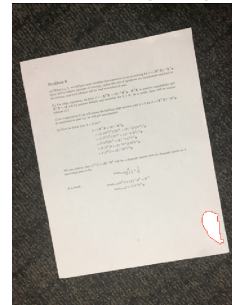
(c) Processing image 2



(d) Processing image 3



(e) Processing image 4



(f) Processing image 5

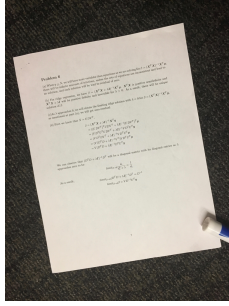
Fig. 4: Exemplar-based Inpainting process

a visually plausible background, and lay a solid foundation for the following segmentation and perspective transform in the pipeline.

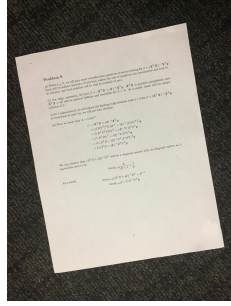
B. Spectral Regularization

In order to examine the performance of the spectral regularization, we have conducted a variety of experiments; which is shown on Fig. 6 side by side with the original images and the results together.

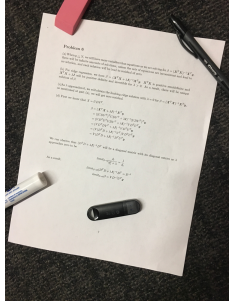
From the original image 1, we can observe that the spectral regularization does help restore the image. There is a sharp edge between the background and the paper but still we can see a



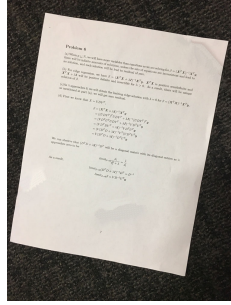
(a) Original image 1



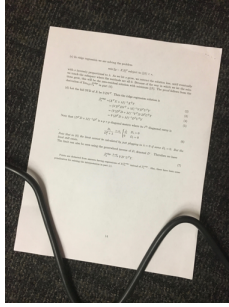
(b) Processed image, removed the marker



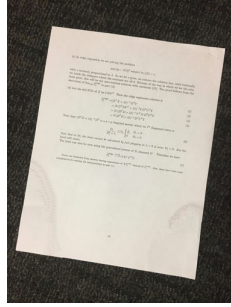
(c) Original image 2



(d) Processed image, removed four items

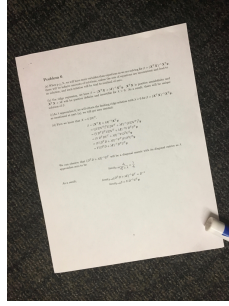


(e) Original image 3

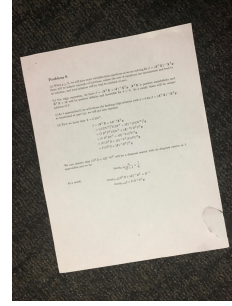


(f) Processed Image, removed the wire

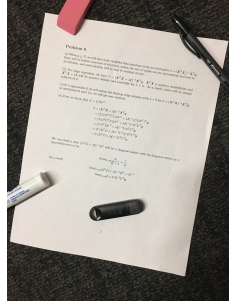
Fig. 5: Exemplar-based Inpainting Results



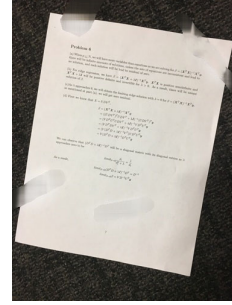
(a) Original image 1



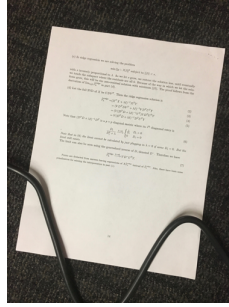
(b) Processed image, removed the marker



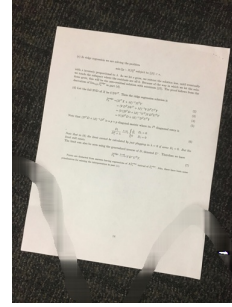
(c) Original image 2



(d) Processed image, removed four items



(e) Original image 3



(f) Processed Image, removed the wire

Fig. 6: Spectral Regularization Results

slightly difference in color between the paper and the restored area.

From the original image 2, which contains four objects to be removed, we can witness the similar behavior where the restored color is similar and close to its neighboring color, but it introduces blended colors in between since there are gray regions between the white and the dark region. This is due to the fact that the spectral regularization is a global method which looks for a solution that minimizes our objective function with regularization; as a result, it focuses less on the local regions.

From the original image 3, there is a relatively large region that is occluded, and the performance is not satisfying as well. In the transitioning region between the white paper and dark background, we can still observe a gradual color change.

Overall, the spectral regularization does help resemble the original color formation but it introduces additional color blending; as a result, the restored edges do not look as sharp as the original. However, in our pipeline, these artifacts can be mitigated later since we would perform Gaussian smoothing and increase the contrast.

VI. RESULTS

The full pipeline is shown on Fig. 7, which shows satisfactory results as we expected. Given the original image and the mask corresponding to the objects to be moved, the object removal method is capable of removing the objects and restoring the occluded areas with either Exemplar-based image inpainting or spectral regularization. The output of the object removal part is fed into the edge detector so that we can look for the contour of the document, which is the largest rectangular region in the image. The perspective transform helps extract the document and reshape into a full rectangle. Overall the pipeline is able to function to act as a document scanner to remove unwanted objects and extract documents as we designed.

VII. DISCUSSION

The Exemplar-based Image Inpainting algorithm combines the advantages of both texture synthesis methods and inpainting methods. It first decides the filling order with the confidence term and data term, which represents how certain we are and how likely there is a linear structure. It then searches for a similar patch similar to the target surroundings, and copies the patch to the target. In this way, the Exemplar-based Image Inpainting algorithm is able to preserve both linear structure and two-dimensional texture into the target region.

The Exemplar-based Image Inpainting algorithm performs well on the restoration of small scratches, and also larger objects are removed. Moreover, the output is very visually plausible. It preserves the sharpness of edges, and it does not need any image segmentation.

Some limitations of the Exemplar-based Image Inpainting algorithm are: *a* if a similar patch does not exist, it cannot produce a plausible result itself; *b* it does not work well on curve structures. These two points are drawbacks originated from our original idea: to encourage linear structure and try to find a similar patch in the image.

In contrast to the Exemplar-based image Inpainting, spectral regularization acts as a convex optimization method to solve for the matrix completion problem. It first initializes the target regions with

zero values, then it solves the problem iteratively by solving the Single Value Decomposition with regularization.

The result of the spectral regularization is also visually plausible when the occluded region is relatively small. It helps preserve the overall color formations; nevertheless, for larger removed region, this method introduces additional color blending which is not visually appealing.

In comparison to the spectral regularization, the Exemplar-based image inpainting has a better performance. However, the drawback of the spectral regularization can be mitigated with the following edge detection method. On the other hand, since the spectral regularization is using Single Value Decomposition, the computation complexity is significantly lower than the Exemplar-based inpainting.

At the end of our pipeline following the object removal part, the segmentation methods, which includes the edge detection, contour search and perspective transform, produce satisfactory results to segment and transform the document part of the image.

REFERENCES

- [1] M. Ashikhmin. Synthesizing natural textures. In Proc. ACM Symposium on Interactive 3D Graphics, pages 217–226, Research Triangle Park, NC, March 2001.
- [2] A. Criminisi, P. Perez, and K. Toyama. "Object removal by exemplar-based inpainting." In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2, pp. II-II. IEEE, 2003.
- [3] A. Efros and W.T. Freeman. Image quilting for texture synthesis and transfer. In Proc. ACM Conf. Comp. Graphics (SIGGRAPH), pages 341–346, Eugene Fiume, August 2001.
- [4] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro. A variational model for filling-in gray level and color images. In Proc. Int. Conf. Computer Vision, pages I: 10–16, Vancouver, Canada, June 2001.
- [5] D. Ziou and S. Tabbone (1998) "Edge detection techniques: An overview", International Journal of Pattern Recognition and Image Analysis, 8(4):537–559, 1998
- [6] J. Canny. "A computational approach to edge detection." In Readings in Computer Vision, pp. 184–203. 1987.
- [7] P. Bao, L. Zhang, and X. Wu. "Canny edge detection enhancement by scale multiplication." IEEE transactions on pattern analysis and machine intelligence 27, no. 9 (2005): 1485–1490.

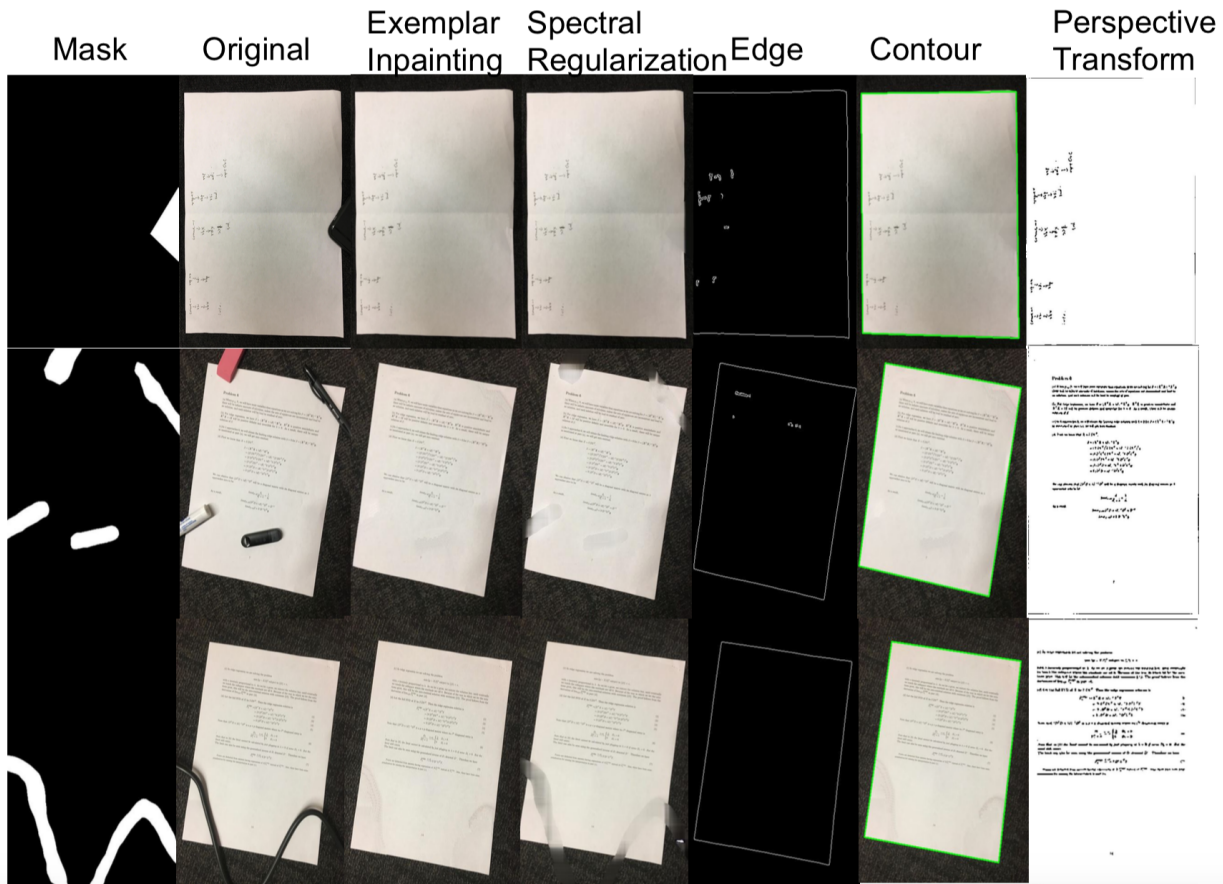


Fig. 7: The pipeline of our object removal and document segmentation.

- [8] Mazumder, Rahul, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices." *Journal of machine learning research* 11.Aug (2010): 2287-2322.