An LCS-based 2D Fragmented Image Reassembly Algorithm

Houman Kamran[1], Kang Zhang[1], Maoqing Li[2], and Xin Li[1*]
[1] School of Electrical Engineering and Computer Science, Louisiana State University, Baton Rouge, USA
[2] Department of Automation, Xiamen University, Xiamen, China
[1] Email: {hkamra1|zhangk4|xinli}@lsu.edu, [2] Email: mqli@xmu.edu.cn

***Abstract*—We propose an algorithm for 2D image fragment reassembly problem based on solving a variation of Longest Common Subsequence (LCS) problem. Our processing pipeline has three steps. First, the boundary of each fragment is extracted automatically from its scanned image, with paper tissue and scanning artifacts removed. Second, inter-fragment boundary matching is computed between each pair of fragments by solving a Longest Common Subsequence problem. The goal is to identify the best possible adjacency relationship among image fragment pairs. Finally, a multi-piece alignment is used to prune incorrect matches and globally compose the final image. We perform experiments on various image fragment datasets and compare our results with existing methods to show the improved efficiency and robustness with respect to images of different resolutions and different levels of noise on boundary pixels.**

Keywords— Fragmented image reassembly; longest common subsequence; curve matching.

I. INTRODUCTION

Fragmented Image reassembly is to reconstruct a 2D image back to its original state, after it has been torn or damaged for various reasons. Being able to accomplish such a task with computers will save us the human labor and time. Analyses show [1] that computers will be able to carry out such a task much faster than humans can do. There are many different types of real world applications that can benefit from solving such a problem. For example, forensic experts can save substantial amount of time and energy if the task of reassembling pieces of an evidence, torn apart as an attempt to destroy it, can be handled automatically. Another application is for archeologists trying to reassemble pieces of artifacts they find through excavation, especially when the pieces are almost flat and smooth, and can be approximated as 2D image fragments. The reassembly process can be faster and more accurate if the task is done automatically by a program compared to when it is done manually.

Existing automatic image reassembly algorithms can be generally categorized into two main groups: color-based approaches and geometry-based approaches. In color-based approaches, the color information of the image fragments are used to guide the reassembly process. This approach can be sensitive to noisy pixel values and also may fail if there exist a few image fragments with similar texture close to the border areas. Geometry-based approaches, on the other hand, use the shapes of image fragments and their borders to find the proper matches between adjacent pieces. These methods could align the fragments more effectively when their contour shapes are irregular. However, reliable partial geometric matching [17, 18] itself could be challenging. Lack of taking advantage of the fragment's color information can be a drawback of this type of approaches.

There is another way of categorizing automatic image reassembly algorithms. Some algorithms operate on the pixel level to guide the reassembly process. These methods are also sensitive to noise. Others reassemble fragments by treating them as 2D geometric regions. Therefore, the time complexity will reduce for the latter approaches compared to the other category. Our idea is to combine the positive aspects of different approaches mentioned above. We propose a three step composition algorithm that takes advantage of both color and geometry information of curve segments on the border of each image fragment, as illustrated in Fig. 1.

A. Artifacts Removal

Each image fragment will be scanned and digitized before it is used as an input. The scanning process often creates undesirable artifacts, especially on and around the border area. A preprocessing step is introduced that ensures a proper execution of all next steps.

B. Pairwise matching

In an effort to find potential matching between each pair of image fragments, we formulate the pairwise matching as solving a variant of Longest Common Subsequence (LCS) problem. Compared with existing pairwise matching techniques, this algorithm improves the time complexity as well as robustness in handling images with possibly noisy values on borders.

C. Global matching

The pairwise matching step of the algorithm runs for each pair of image fragments without global understanding of the layout of the other fragmented pieces. Therefore, alignments suggested by pairwise matching step can be incorrect. We keep many potential alignments between each pair of fragments. Then, a global matching is adopted to select correct alignments offered by the previous step, by optimizing the mutual consistency of alignments of multiple pieces together.

The Main contributions of this paper are as follows:

- We develop a simple but necessary artifact removal preprocessing step to eliminate noisy boundary tissues.
- Effective partial matching is essential to the process. We develop a novel polygonization algorithm to model image fragment's boundary shape feature, and a pairwise matching algorithm based on a revised LCS algorithm. This algorithm effectively utilizes both geometric and color characteristics of image fragments.

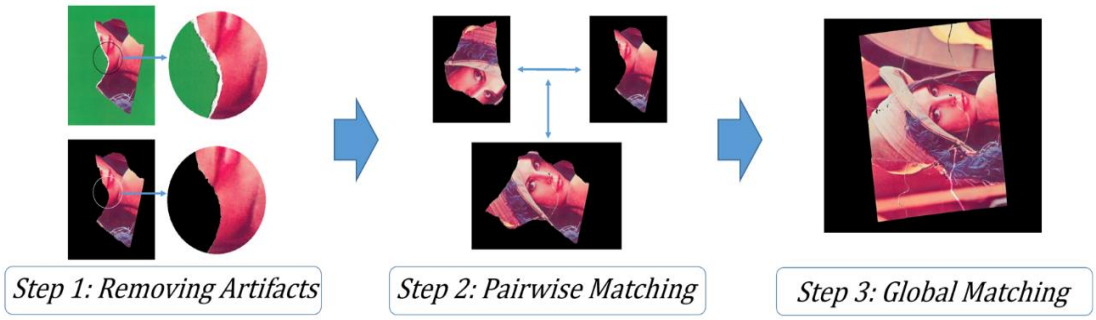

Fig. 1. A demonstration of the 3-step pipeline

II. Related Work

Finding a relative transformation between adjacent image fragments is the essence of most proposed automatic reassembly algorithms. Different authors utilize different measures to investigate a pair of image fragments for any possible matches. The two main approaches are based on color properties of the pixels on and around the border areas and geometry properties of the image fragments such as shape and curvature of the borders.

Color-based Approaches. In color-based approaches, the color information of the image fragments are used to guide the reassembly process. Tsamoura and Pitas [1] form a list of pairs of image fragments that are likely to be adjacent. They use a novel approach to color histograms based on what Cinque et al. proposed in [10], called spatial-chromatic histograms, which not only considers color information but also considers spatial distribution of color. Authors in [1] then present an algorithm based on LCS for identifying matching partial contour curves for each pair of fragments. Their approach to fragment contour matching is based exclusively on pixel color information on the contours. As a result, it loses some efficiency and is sensitive to noisy pixel values on the borders. In [12], the authors focus on the task of matching two image fragments only using the information extracted from the outlines and from the color contents of the fragments. In [15], the texture of a band outside the border of pieces is predicted by inpainting and texture synthesis methods proposed in [16]. An FFT-based registration algorithm is then utilized to find the alignment of the fragment pieces. Color-based approaches can be sensitive to noisy pixel values.

Geometry-based Approaches. In geometry-based approaches, fragment borders are usually modeled as a 2D curve. Some use polygon approximation of the curves and some use curvature information and local shape features of the curves to match image fragments alongside their boundaries. Justino, Oliveira and Freitas in [4] solve the problem of reconstructing shredded documents by first doing a polygonal approximation of the borders to reduce possible complexity of the boundaries. Then relevant features of each polygon, i.e. angle of each vertex with respect to its two neighbors and distances between each vertex and its neighbors, are extracted to lead the matching. In [5], a shape feature, referred to as the turning function, is calculated and used to investigate the matching of fragment pairs. Wolfson in [14] finds the longest curve subsection being shared between polygonised fragment borders through geometric hashing. It does not allow for deletion or mismatch between curve segments while detecting the longest common subsection between two border curves. He does not introduce a global matching step and the focus of the proposed method is on pairwise matching. Zhang and Li in [6] suggest to approximate the border with a polygon. Instead of working on pixel level, the alignment is found alongside the polygon sides. The method used to polygonise the border curve is adapted and improved by our work. Also, in the pairwise matching step, an exhaustive search is used which is affecting the efficiency of the proposed method considering the large search space of the pairwise matching step. They, then, formulate a global matching scheme as solving a maximal compatible edge set from a given graph to reconstruct the original image from calculated local matches. Others, like [1], have proposed global matching steps in the pipeline of their algorithms. However, global matching step is not the focus of this work.The polygonization, in general, could be globally affected by the initial pose of the image fragments and the approximation error bound, and hence, is not completely rotation-invariant and will affect the subsequent matching and reassembly results. Local curvatures of pixels on the contour curve as the fragment's descriptor, on the other hand, will be rotation-invariant but is sensitive to geometric noise on the image boundary, which is often inevitable.

LCS-based Matching algorithms. Based on an algorithm Smith and Waterman propose in [8], some algorithms [3,11] try to find a common subsection between two neigboring border curves. Based on color values or curvature values at pixels they suggest a sequence of pixels as a possible common border subsection. In [3] the boundaries are represented by shape feature strings. A Shape feature representing each pixel in the string is an average of the curvature values of the neighboring pixels. A method based on LCS detects similar subsequences between strings representing two image fragments at progressively increasing scales of resolution. A similar approach is taken in [11] by Kong and Kimia. They try to find the largest common subsequence with the similar color and curvature information and their focus is on pairwise matching. They use dynamic programing to create a coarse alignment on the reduced version of the borders. However, all the LCS-based methods mentioned above operate on pixel level and do not take advantage of polygonization technique to estimate a border curve as a polygon. The algorithms operating on pixel level are sensitive to noisy pixel values on the border. Also their time complexity can be big due to the large number of pixels on each border.

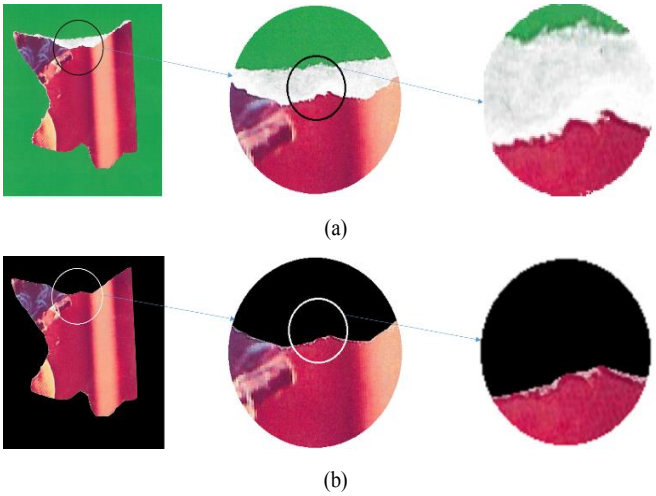

Fig. 2. A demonstration for first step. (a) Artifacts on and around the border area. (b) Paper tissues removed, noise reduced.

III. LSC-Based Image Reassembly

To reassemble 2D image fragments back into their original form, a wide range of approaches are being introduced in existing literature. In this paper, a novel method is devised in three steps that builds on recent research and combines the strengths of previous ideas.

A. Removing Artifacts

Each image fragment has to be digitized before it can be used as an input. A knowledgeable choice for background color during scanning process can enhance future steps to a great extent. The color properties and color histograms of all image fragments are investigated for successful choices. The best possible choice for background color is a color that not only does not exist in the image fragment, but also is furthest from all the existing colors.This will facilitate segmentation and border extraction during next steps.

As a result of the scanning process, there will be scanning artifacts such as shadow around the borderlines of each image fragment. Also, as the foreground of an image is extracted from the background, there will be some undesirable narrow white regions attached to the main body of the image, i.e. paper tissues. The paper tissue artifacts could have happened as the original image was being torn apart. These artifacts will affect both the geometry and color properties of image fragments on and around the border area. Without these artifacts removed, the algorithm will fail. To eliminate these artifacts, the idea is to start with a white-colored pixel on or close to the border and remove that pixel together with all the other connected white components in its neighboring area. The goal is to remove the tissue but keep the main body of the foreground intact. After removing the tissue, some unconnected components might still remain outside of the main body of the image which will be eliminated using some morphological operations [2].

Considering the fact that potential matches between a pair of image fragments are investigated on the border areas, noisy borders can lead to undesired results. After removing the paper tissues and as a result of morphological operations, border curves will have nuances too small that can be considered as noise. Smoothing the curves can help reduce these noise-like nuances. The border for each image fragment is extracted as a curve and each pixel on the curve is pushed toward a position in between its right and left neighboring pixels over several iterations. As a result, the noise on the curve will reduce and the border will become smooth. Failing to smooth the boundary curve can cause these noise-like nuances to be mistaken as feature points. Detecting the correct feature points is vital to our algorithm because correct polygonization, which is a part of next step, depends on it. Fig. 2 depicts a zoomed in version of the result for better illustration, once before paper tissue and noise on the border are removed and once after.

B. Pairwise Matching

The goal of pairwise matching step is to find border subsections on a pair of image fragments that are possibly common between the two of them. This step of the algorithm runs for each pair of image fragments without global understanding of the layout of the other fragmented pieces. Therefore, this step suggests some correct and some incorrect alignments. The global matching (to be discussed in Section C) will detect and eliminate incorrect alignments offered by pairwise matching step.

Some existing pairwise matching algorithms investigate the borderlines on the pixel level while others [6] suggest to approximate the border with a polygon. Therefore, instead of working on pixel level, the alignment is found alongside one of the polygon sides. The border of each image fragment, approximated by a polygon, can be represented as a sequence of sides. The novelty of our idea is based on the observation that if two image fragments are adjacent, their simplified polygons usually share more than only one side. Therefore, a more effective method to describe a common border subsection is by matching a subsequence of sides, rather than considering one side at a time. We can formulate this as finding the longest subsequence of sides shared between the two polygons. Therefore, for two image fragments, finding that possible common border subsection will reduce to the problem of finding the longest common subsequence between two sequences representing the two borders. Smith and Waterman [8] propose a solution to a similar problem based on a dynamic programming approach. The goal for the pairwise matching step can be stated as calculating a 3×3 rigid transformation, $T_{i,j}$, which aligns i_{th} and j_{th} image fragments alongside their common border subsections. Our pairwise matching step is formulated in four steps.

1) To extract border curve for each image fragment and approximate it with a polygon.
2) To calculate a descriptor to represent each segment.
3) To apply a revised LCS approach to find the best sequence of matching segments.
4) To calculate final transformation.

The first step is to extract the border loop on each image fragment. Let us assume C_i and C_j are the boundary curves extracted for the pair of image fragments at hand. Polygonising each contour curve means it is partitioned into a sequence of curve segments, each of which flat enough to be approximated by a line segment. $S_{i,k}$ will be used to denote the k_{th} of such curve segments on C_i for example. The process of decomposing

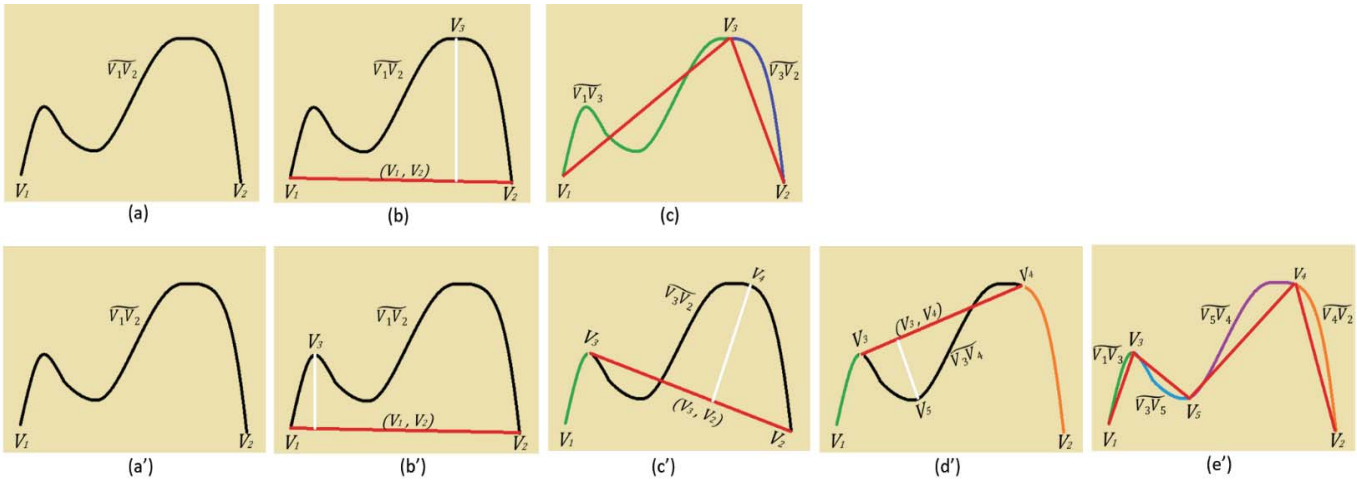

Fig. 3. The importance of incorporating the curvature values into the segmentation process. (a,a') The original curve to be segmented. (b,c) The decision on where to break the curve is made, solely, based on the Euclidean distance each pixels makes with the line segment describing the curve. (b',c',d',e') The Euclidean distance a pixel makes with the approximating line segment is scaled based on the curvature value at that specific pixel. As a result, pixels with high curvature values are more likely to be picked as vertices during segmentation process. For example, the first vertex to be picked in (b') is v_3 due to its high curvature value, while there are points on the curve that have bigger Euclidean distance to the line segment (V_1, V_2).

each boundary loop into small segments is accomplished by employing a revised Douglas-Peucker (DP) algorithm [9]. For a curve segment denoted as $S = \widehat{V_1V_2}$, where V_1 and V_2 are the starting and ending points of the segment, our DP algorithm first uses a line segment connecting V_1 and V_2 to approximate the curve, denoted as (V_1, V_2). The distance from each point V_i on the curve to line segment (V_1, V_2) is calculated and scaled by a value reciprocal to the curvature at V_i. If there exists a point, V_1, on the curve whose scaled distance to (V_1, V_2) is bigger than a predefined threshold, denoted as the DP-threshold, the segment $\widehat{V_1V_2}$ is decomposed into two smaller segments $\widehat{V_1V_3}$ and $\widehat{V_3V_2}$. The DP algorithm then runs recursively on $\widehat{V_1V_3}$ and $\widehat{V_3V_2}$ until no curve segment can be further decomposed. To start the process on a boundary loop C_i, the two points, V_1 and V_2, are chosen such that they are furthest from one another on the boundary. The DP algorithm just described will run on $\widehat{V_1V_2}$ and $\widehat{V_2V_1}$ respectively. As a result, the boundary loop C_i will be partitioned into a sequence of smaller curve segments $S_{i,k}$. Fig. 3 is devised for illustration purposes to emphasize the importance of including curvature value at each point on a curve into its segmentation process. To calculate a reliable curvature value for each point on the curve, a small number of neighboring pixels, before and after the point of interest, are approximated by two line segments. The angle between these lines define the curvature value at the point of interest unambiguously. Picking pixels with higher curvature values as vertices of the polygon, during segmentation process, is closer to the way a human would do such a task compared to the previous methods proposed. Fig. 3 is explaining why using this approach creates better final results compared to other methods through a synthetic example.

Next is to define, for each curve segment $S_{i,k}$ on a border like C_i, a descriptor, $d(S_{i,k})$, based on its color and geometric properties. $d(S_{i,k})$ is chosen to be a 6-tuple value in $\mathbb{R}^6$. The first value is the length of $S_{i,k}$ in pixels. The second, third and fourth values are the average of red, blue and green component values for all the pixels on the curve segment. Finally, the fifth and sixth values are the angles a curve segment makes with its right and left neighboring segments respectively. Matching border curves on two adjacent image fragments are usually composed of more than only one segment. To ensure these matching curves have similar geometric shape, the angles each segment on one curve makes with its neighboring segments must be the same as the angles its counterpart on the other curve makes with its own neighboring segments. For a pair of curve segments $S_{i,k}$ and $S_{j,l}$, on borders C_i and C_j respectively, a $Sim(S_{i,k}, S_{j,l})$ will be calculated based on how similar $d(S_{i,k})$ and $d(S_{j,l})$ are. If they are similar enough, $S_{i,k}$ and $S_{j,l}$ will be considered a match and $Sim(S_{i,k}, S_{j,l})$ will hold a positive value; otherwise, in case of a mismatch, $Sim(S_{i,k}, S_{j,l})$ will be negative.

Considering $d(S_{i,k})$, the descriptor for a curve segment, equivalent to a character, a boundary curve can be represented as a string. The length of the string representing a boundary will be equal to the number of segments on that boundary. Therefore, the problem of finding the longest sequence of curve segments shared between two borders will be equivalent to the problem of finding the longest subsequence shared between two given strings. To solve this problem, we revise the Smith-Waterman algorithm [8], which uses a dynamic programming approach to find the longest common subsequence (LCS) between two given strings. To make matching between characters flexible, jumping over a character in either one of the subsequences and simply leaving it unmatched with any of the characters from the other subsequence is allowed. This act is called a deletion and is justified when, doing so, there will be a prospect of being able to make more promising matches along the path as we go forward. This means dissimilarity between two subsequences at hand. Therefore, a penalty is considered for such a case which is the value represented by W in line 7 of Fig 4.

Fig 4. explains revised Smith-Waterman algorithm in pseudo-code when used to solve our problem. This algorithm builds a matrix called H. The pair of subsequences with maximum similarity is found by first locating the maximum element of H. The other matrix elements leading to this maximum value are then sequentially determined with a trace-back procedure ending with an element of H equal to zero. This procedure identifies the

```
// for a fixed i and a fixed j, i.e. for a specific pair of image fragments
// n is the number of curve segments on the border of image fragment C_i
// m is the number of curve segments on the border of image fragment C_j

1       for all 0 ≤ t ≤ n
2           H_{t,0} = 0

3       for all 0 ≤ s ≤ m
4           H_{0,s} = 0

5       for all 1 ≤ t ≤ n
6           for all 1 ≤ s ≤ m
7               H_{t,s} = max { H_{t-1,s-1} + Sim(d_{i,t} , d_{j,s}) , H_{t-1,s} – W , H_{t,s-1} – W , 0 }

8       produce the corresponding matches by starting from the maximum
        element of H and tracing back to an element of H equal to zero
```

Fig. 4. Algorithm for finding the Longest Common Subsequence between two strings of length n and m respectively.

subsequences as well as produces the corresponding matches between their composing characters which leads, eventually, to a transformation between two image fragments. Fig. 5 shows the most similar pair of curve segment sequences, one from each border, recovered using our version of revised LCS.

C. Global Matching

Pairwise matching step runs for all possible pairs of image fragments without global understanding of the layout of the other fragmented pieces. Therefore, it suggests one or more possible transformations between each pair of image fragments, among which only some can be correct and the others are incorrect. We adopt a global matching step from our previous work in [6] to detect and eliminate incorrect alignments offered in the previous pairwise matching step. Global matching step, in fact, is used to complement the work of pairwise matching. If we develop a simple pairwise matching step, we are simply pushing more work down the pipeline to be accomplished during the global matching step. For example, in [6], due to the relatively simple pairwise matching strategy, there is a need for a powerful global matching scheme. On the other hand, using the pairwise matching method proposed here, the obtained pairwise alignments are more reliable compared to [6]; and consequently, a simpler global matching algorithm should suffice. Nonetheless, the method proposed in [6] is being adopted here for global matching step. The global matching scheme is formulated, in [6], as solving a maximal compatible edge set from a given graph. A greedy algorithm is proposed to iteratively insert an alignment that is consistent with all existing selected alignments and has the highest accumulated matching scores with all the other pieces.

IV. Experimental Results

We perform multiple experiments to evaluate our assembly algorithm. In our experiments, we print out randomly selected digital images on papers. Then, we randomly tear an image into multiple pieces and scan each image fragment. Our reassembly algorithm is then used to recompose the scanned digital image fragments. The results in Fig 6 demonstrate the effectiveness and robustness of our approach for real hand-torn images with

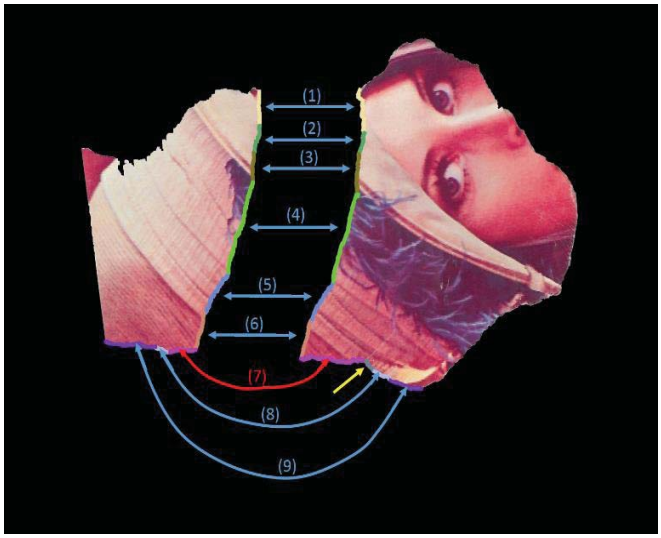

Fig. 5. The result of revised LCS for two image fragments. The matched segments are color coded. All matched segments marked with blue arrows are good matches. The only short curve segment marked with a yellow arrow is showing a case of deletion due to wrong partitioning of the border. Match (7), marked with a red arrow, shows a case of a bad match. These two choices, i.e. deletion and mismatch, are being made for the benefit of matches (8) and (9) that are coming after them.

increasing number of pieces. Comparing the results of this paper with [6] on the same data sets shown in Fig 6, shows the big improvement in running time complexity as a result of reducing the search space in pairwise matching step.

We have selected Tsamoura and Piras [1] method and implemented their proposed algorithm so that we can compare its performance with our method. [1] will fail on cases where the tearing or scanning artifacts on image fragments are not handled properly. Therefore, in order to create a level ground, we created input datasets free of artifacts through simulation. The comparison is performed on cases with 9 and 18 pieces as shown in Fig 7.

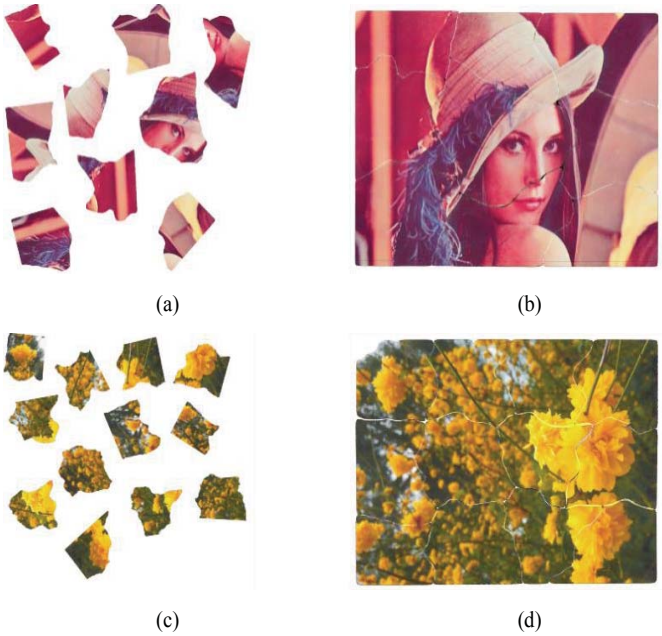

Fig. 6. Demonstrating the effectiveness of our method on real hand-torn images. (a,b) 9-piece input data set, (c,d) 12-piece input data set.

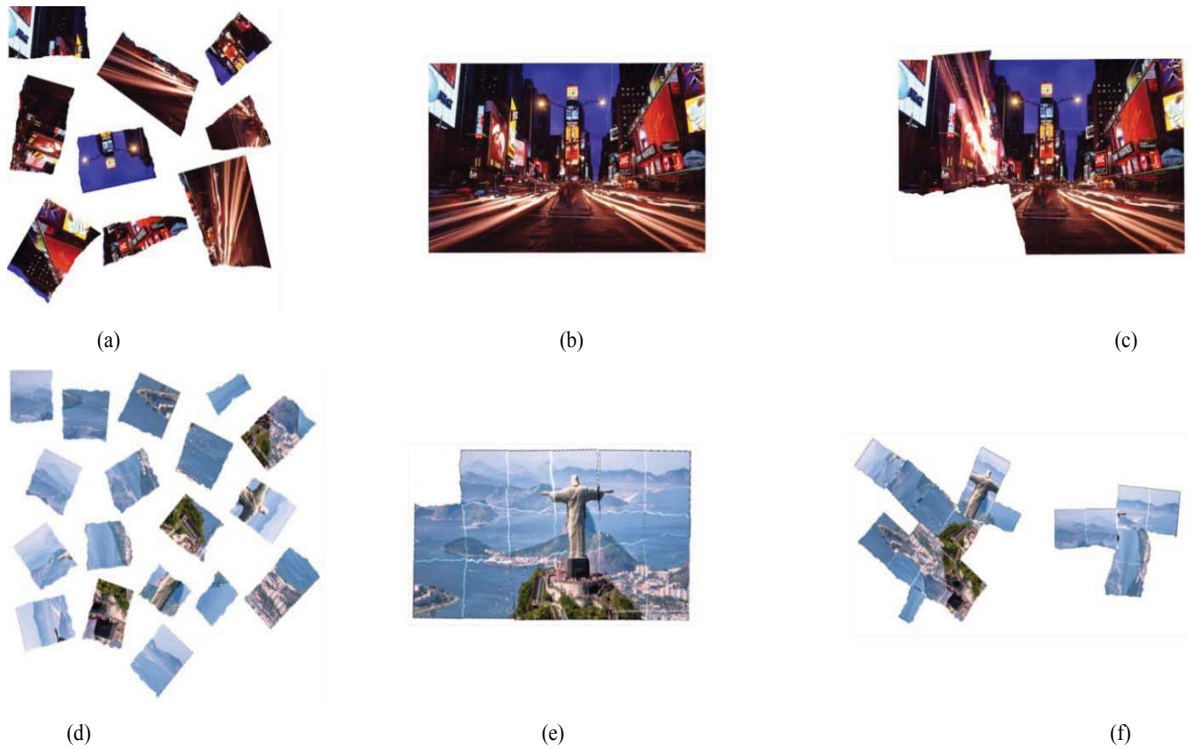

Fig. 7. Comparing our method with the method proposed in [1]. First column is the input data set. Second column shows the result of our method and last column shows the result of [1]. (a,b,c) 9-piece input data set, (d,e,f) 18-piece input data set

V. Conclusion and Future Work

We present a novel computational pipeline for the automatic reassembly of fragmented images. It consists of three main steps: removing artifacts, pairwise matching between two image fragments, and global fragment reassembly. We evaluate our algorithm using various real-world images and demonstrate that it is fast and robust.

On the global matching front, more effective graph searching and backtracking algorithms or some stochastic optimization strategies could be investigated. In addition, adaptive algorithm to pick the proper DP-threshold for different data sets with different image resolutions can help generalize our algorithm even more.

VI. Acknowledgements

This work was partly supported by the National Science Foundation IIS-1320959, and the National Natural Science Foundation of China 61728206.

References

[1] E. Tsamoura and I. Pitas, “Automatic color based reassembly of fragmented images and paintings,” IEEE Transactions on Image Processing, vol. 19, No. 3, pp. 680–690, 2010.

[2] S.H. Oguz, Y.H. Hu, and T.Q. Nguyen, “Image coding ringing artifact reduction using morphological post-filtering,” Proc. Workshop on Multimedia Signal Processing, pp. 628–633, 1998.

[3] H.C. da Gama Leitao and J. Stolfi, “A multiscale method for the reassembly of two-dimensional fragmented objects,” IEEE Trans. Pattern Analysis and Machine Intelligence, 24(9): 1239–1251, 2002.

[4] E. Justino, L.S. Oliveira, and C. Freitas, “Reconstructing shredded documents through feature matching,” Forensic Science International, vol. 160, No. 2, pp. 140–147, 2006.

[5] L. Zhu, Z. Zhou, and D. Hu, “Globally consistent reconstruction of ripped-up documents,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, No. 1, pp. 1–13, 2008.

[6] K. Zhang and X. Li, “A graph-based optimization algorithm for fragmented image reassembly,” Graphical Models, vol. 76, No. 5, pp. 484–495, 2014.

[7] P.J. Besl and N.D. McKay, “A method for registration of 3-D shapes,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, No. 2, pp. 239–256, 1992.

[8] T.F. Smith and M.S. Waterman, “Identification of common molecular subsequences,” J. Molecular Biology, 147(1):195–197, 1981.

[9] D.H. Douglas and T.K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” Cartographica: The International Journal for Geographic Information and Geovisualization, 10(2): 112–122, 1973.

[10] L. Cinque, G. Ciocca, S. Levialdi, A. Pellicano, and R. Schettini, “Color-based image retrieval using spatial-chromatic histograms,” Image and Vision Computing, vol. 19, No. 13, pp. 979–986, 2001.

[11] W. Kong and B.B. Kimia, “On solving 2D and 3D puzzles using curve matching,” Proc. CVPR, vol.2, pp. 583–590, 2001.

[12] F. Amigoni, S. Gazzani, and S. Podico, “A method for reassembling fragments in image reconstruction,” Proc. ICIP, vol. 3, pp. 581–4, 2003.

[13] Y. Chen and G. Medioni, “Object modeling by registration of multiple range images,” Image and Vision Computing, 10(3):145–155, 1992.

[14] H.J. Wolfson, “On curve matching,” IEEE Trans. on Pattern Analysis and. Machine Intelligence, 12(5): 483–489, 1990.

[15] M. Sagiroglu and A. Ercil, “A texture based matching approach for automated assembly of puzzles,” Proc. ICPR, 3:1036–1041, 2006.

[16] A. Criminisi, P. Perez and K. Toyama, “Region filling and object removal by exemplar-based image inpainting,” IEEE Trans. Image Processing, 13(9):1200–1212, 2004.

[17] X. Li and S. Iyengar, "On Computing Mapping of 3D Objects: A Survey," ACM Computing Surveys, 47(2):34:1 – 34:45, 2015.

[18] K. Zhang, M. Manhein, W. Waggenspack, and X. Li, “3D Fragment Reassembly using Integrated Template Guidance and Fracture-Region Matching,” International Conference on Computer Vision (ICCV), 2138-2146, 2015.