A Surface Approximation Method for Image and Video Correspondences

Jingwei Huang, Bin Wang, Wenping Wang, and Pradeep Sen, *Senior Member, IEEE*

***Abstract*— Although finding correspondences between similar images is an important problem in image processing, the existing algorithms cannot find accurate and dense correspondences in images with significant changes in lighting/transformation or with the non-rigid objects. This paper proposes a novel method for finding accurate and dense correspondences between images even in these difficult situations. Starting with the non-rigid dense correspondence algorithm [1] to generate an initial correspondence map, we propose a new geometric filter that uses cubic B-Spline surfaces to approximate the correspondence mapping functions for shared objects in both images, thereby eliminating outliers and noise. We then propose an iterative algorithm which enlarges the region containing valid correspondences. Compared with the existing methods, our method is more robust to significant changes in lighting, color, or viewpoint. Furthermore, we demonstrate how to extend our surface approximation method to video editing by first generating a reliable correspondence map between a given source frame and each frame of a video. The user can then edit the source frame, and the changes are automatically propagated through the entire video using the correspondence map. To evaluate our approach, we examine applications of unsupervised image recognition and video texture editing, and show that our algorithm produces better results than those from state-of-the-art approaches.**

***Index Terms*— Dense image correspondence, B-Spline fitting, co-recognition, video texture editing.**

I. Introduction

FINDING dense, reliable correspondences between images that have content in common is a key problem for many interesting applications in image processing and computer vision, including object detection/recognition, video encoding/compression, image and video editing, and 3D scene reconstruction. Given this importance, researchers in the past have proposed a variety of different correspondence algorithms that are able to compute robust correspondences under different conditions. For example, for pairs of images with small motions, optical-flow [2] is fairly robust and can find dense correspondence fields. However, these kinds of methods cannot deal with significant differences in lighting or large changes in the poses of the shared objects in the images.

Other methods such as SIFT [3] utilize sparse features which makes them more robust to geometric and photometric variations. To produce reliable matches, however, these methods are usually followed by a geometric filtering step like RANSAC, which removes outliers based on the assumption that the scene is rigid [3]. New methods like the Non-Rigid Dense Correspondence (NRDC) work of HaCohen et al. [1] have sought to relax the rigidity assumption and enable the generation dense correspondences for non-rigid objects, based on local matching of nearest neighbor patches in the two images. Although this method works quite well for a variety of scenes, the correspondence field produced is still too sparse and noisy for complex scenes to be useful for applications such as image editing.

A possible approach to solve this problem is to apply RANSAC after NRDC. However, because RANSAC uses a linear function to approximate correspondences, it cannot handle non-rigid scenes (e.g., see Fig. 9). In order to improve correspondences for non-rigid objects, we need a more flexible geometric filter and in this work we propose to use a B-Spline [4] for this purpose. Our idea is motivated by the observation that if a shared non-rigid object can be approximated by a smooth surface, its correspondence map should also be smooth. Since B-Spline surfaces are good at approximating smooth surfaces, we hypothesize that they would also be a good approximation for the correspondence map.

To do this, we first generate an initial correspondence map using NRDC [1]. We then use different B-Spline surfaces to approximate the correspondence maps of different shared parts between the two images. Because of the smooth continuity of the B-Spline surface, the outliers and noise of the initial correspondence map are smoothed away. To further increase the size of the regions with valid correspondence, we propose a new correspondence map extension algorithm that leverages the B-Spline approximations to provide global guidance when creating correspondences for these regions.

Compared to previous work, our new method can compute more reliable, continuous, and denser correspondences between images despite significant geometric and photometric

Manuscript received May 31, 2014; revised September 29, 2014, January 26, 2015, and June 15, 2015; accepted July 10, 2015. Date of publication July 29, 2015; date of current version September 29, 2015. This work was supported by the National Science Foundation of China under Grant 61373071. The work of P. Sen was supported by the National Science Foundation under Grant IIS-1342931 and Grant IIS-1321168. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Andrea Cavallaro. *(Corresponding author: Bin Wang.)*

J. Huang and B. Wang are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: hjwdzh@gmail.com; wangbins@tsinghua.edu.cn).

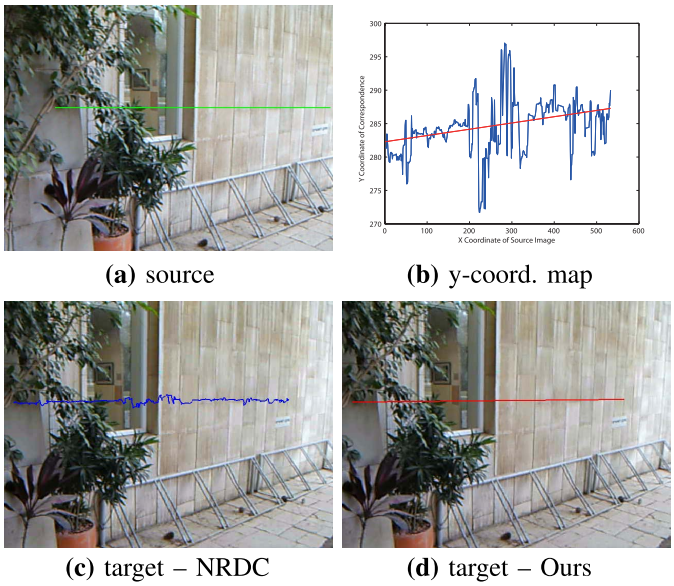
W. Wang is with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: wenping@cs.hku.hk).

P. Sen is with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: psen@ece.ucsb.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes videos for Fig. 14 and video light mapping results (Fig. 15). The total size of the videos is 77.2 MB. Contact wangbins@tsinghua.edu.cn for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2462029



(a) source **(b)** y-coord. map

(c) target – NRDC **(d)** target – Ours

Fig. 1. Initial correspondences and our improved B-Spline approximation. We would like to find a correspondence field from the **(a)** source image to the **(c,d)** target image, so that edits on the source will be mapped to the target. In this case, we examine how pixels on the green line in the source are mapped to the target. **(b)** Plot of the y coordinate of the correspondences from the green line as a function of the x position using both NRDC correspondences (blue) and our B-Spline fit (red). **(c)** The initial correspondences using NRDC map the green line to the blue line in the target, which is quite noisy. **(d)** Our B-Spline fit produces correspondences shown by the red line, which are smoother and less noisy.

variations. For example, Fig. 1 shows how a scanline in the source image (shown by the green line) is mapped to the target image via the calculated correspondences. Because these images differ only by a change in perspective and this part of the scene is relatively planar, the correspondences are given by a homography and so the green line in the source should map to a line in the other. The blue trajectory is computed using the initial correspondences from NRDC, but it is quite noisy and is not a good approximation for a straight line. The red line represents the correspondences approximated by our B-Spline fit, which is smoother and less noisy. This is how our method is able to eliminate errors and find accurate correspondences.

We can also extend our surface approximation idea beyond simply finding correspondences between static images by using a Trivariate-Spline approximation [5] for space-time correspondences. This allows us to ensure the continuity of correspondences not only in space but also in time, which is crucial for video editing applications. Compared to previous methods which use optical-flow as a basic approach to enforce long-term temporal continuity [6], our method can generate more robust correspondences between a source frame and the others, making it more suitable for video editing.

We begin the paper by discussing relevant previous work in Sec. II. We then present the details of our algorithm in Sec. III and perform a thorough evaluation of it in Sec. IV, specifically demonstrating the improved density, precision, and continuity of the correspondence fields generated by our algorithm. In Sec. V, we apply our algorithm to two different applications of interest to the computer graphics and vision communities: unsupervised image recognition and video texture editing. Finally, we conclude in Sec. VI by talking about some limitations of our approach and proposing some potential areas of future exploration.

II. Related Work

A. Finding Correspondences in Images

Algorithms for finding correspondences were first developed for stereo vision applications [2]. Optical flow and stereo reconstruction can be used for adjacent frames with small motions or minor changes in lighting, in which cases they can generate dense correspondence fields with high accuracy. However, when viewpoint and environment change significantly, shared objects will appear very different in both images and these simpler methods would lose their effectiveness.

Algorithms like SIFT [3] perform well on finding reliable correspondences in these harder cases, but their correspondence fields are sparse. Geometric filtering steps like RANSAC are often used to generate more reliable matches and enlarge the correspondence fields, but they assume that objects in the scene are rigid.

Other more powerful methods have been proposed which combine sparse features with dense matching. They are able to deal with large-displacement optical flow [7] and highly different scenes [8], but they do not work well on images with significant change in rotation or scale. Other approaches initially find a few reliable feature matches and "densify" the correspondences [9]. However, these methods are not very accurate and perform poorly on pixelwise correspondences.

Another set of methods use pixel patches (i.e., square blocks of pixels) to compute correspondences between images. For example, the Image Analogies algorithm [10] uses the approximate nearest neighbor (ANN) correspondence [11] from each patch in a target image to a pair of source images in order to mimic a filter that has been applied to the source. Later, Barnes et al. demonstrated that computing ANN correspondences can be significantly accelerated by the randomized PatchMatch algorithm [12], which assumes coherency in the nearest-neighbor correspondence field to approximate it quickly.

Essentially, the PatchMatch algorithm leverages the fact that if patch B in the source is a good match for patch A in the target, then the neighboring patches of A are likely to find good matches in the neighborhood of B. The PatchMatch algorithm has been instrumental in enabling many powerful image editing and image synthesis algorithms based on patch-based synthesis, such as image morphing [13], image melding [14], and patch-based high-dynamic range image reconstruction [15], [16].

Although the original PatchMatch algorithm assumes that corresponding objects do not rotate or scale in the two images, this assumption was later relaxed with the generalized PatchMatch method [17], which increases the search space by taking rotation and scale changes into consideration. There has also been research to address other shortcomings of PatchMatch, such as making it more robust in regions with smooth gradients [14] or near object boundaries [18].

Building upon generalized PatchMatch, HaCohen *et al.* [1] introduced the Non-Rigid Dense Correspondence algorithm (NRDC), a new method for finding correspondences

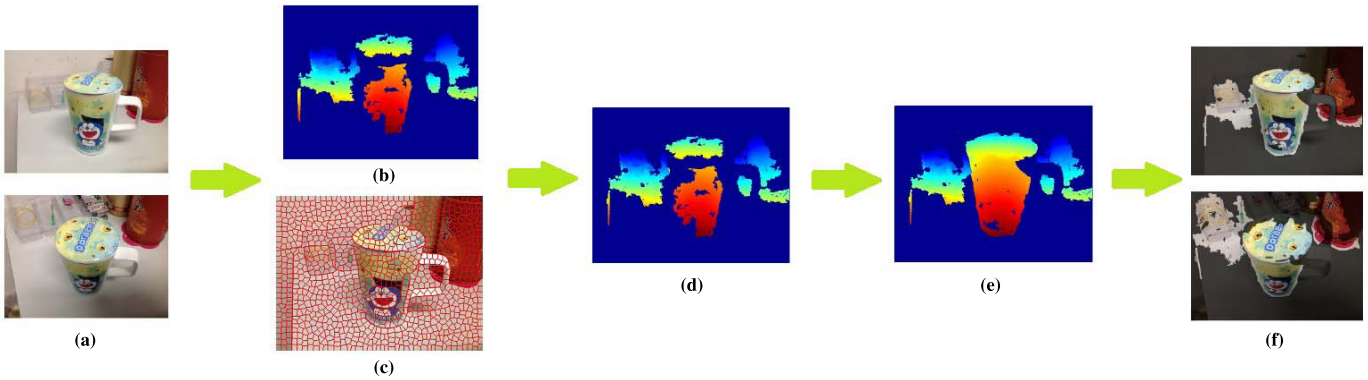


Fig. 2. Overview of our method for finding correspondences between two input images. **(a)** Source and target input images. In this case, we see that the cup has two visible surfaces (the body and lid). **(b)** Initial correspondence map from NRDC (Sec. III-A). **(c)** Result of the automated segmentation step, which breaks up the image into super-pixels (Sec. III-B). **(d)** Intermediate result consisting of five B-Spline surfaces after merging step (Sec. III-B). **(e)** Extension of the five B-Spline correspondence maps (Sec. III-C). **(f)** Final correspondence results. Only regions with high confidence are shown, the others are grayed out.

in images by finding matching patches in both. Generalized PatchMatch method by itself is not effective for computing reliable dense correspondence maps, since it only finds nearest-neighbor matches locally on a single scale and does not try to capture consistent regions or take into account global constraints. For this reason, NRDC [1] uses generalized PatchMatch in an iterative, coarse-to-fine refinement that does several key things: (1) performs a *constrained* search using generalized PatchMatch based on parameters computed in the earlier iteration, (2) aggregates consistent matching regions, (3) computes a color transformation model to map one region to another, and (4) adjusts the search ranges for the next iteration using generalized PatchMatch to improve the correspondence computation.

The NRDC algorithm is able to generate dense, reliable correspondence maps for non-rigid objects. However, when the differences in viewpoint, lighting, or transformation are more extreme, the reliable correspondences from NRDC remain fairly sparse (see, e.g., Fig. 10). Hence, the challenge of finding good, dense correspondences in cases of non-rigid motion and significant changes is still an open problem that we seek to address in this work.

B. Editing Textures in Video Sequences

When editing a video, a user may want to add/edit elements in the scene, such as putting new pictures on the wall, modifying the textures on various surfaces, or changing the lighting in different parts of the scene. It is usually possible for the user to edit one frame of the video, but it is very time-consuming (and unfeasible) to edit all the frames in this manner. Moreover, since our eyes are sensitive to temporal inconsistencies which appear as flickering in the final video, it is not easy to perform this editing manually in a way that will look natural and plausible.

Video texture editing techniques allow a user to edit just one frame and propagate his/her operations to the whole video automatically. This relieves the users from having to do all that tedious work and allows them to generate high-quality, edited videos. Because this is an important problem, there has been some previous work to try to address this.

For example, Rav-Acha *et al.* [19] proposed a method for editing video textures called unwrap mosaics. Their basic idea maps every frame of video to a texture map that is unwrapped in the u, v domain. This texture can be edited manually and then the changes would be automatically propagated to all video frames. Although their algorithm produced remarkable results even in cases of non-rigid objects, complex deformations, or occlusions, unlike our approach their algorithm does not solve explicitly for a correspondence field between frames. This means that it cannot be used directly in applications that require a correspondence field such as stereo reconstruction, shared object recognition, and others.

More recently, Crivelli *et al.* [6] proposed an automatic method for mapping textures between images. They use optical flow as a basic algorithm to find trajectories with space-time continuity over the frames of video. They calculate correspondences between adjacent frames and propose a new strategy to produce trajectories from the correspondences. However, this approach of texture editing still loses accuracy in the long term, because errors will be amplified gradually. Our Trivariate-Spline approximation method, on the other hand, generates more reliable correspondences and ensures the continuity of correspondences in space and time. Because the work of Crivelli *et al.* is considered the state-of-the-art in this field, we compare against it in Sec. V-B.

III. Algorithm

Our goal is to find dense, continuous, and precise correspondences between two images with variations in lighting, viewpoint, and motion. In this section, we present our B-Spline correspondences algorithm (see overview in Alg. 1, as well as an illustration of the different key steps in Fig. 2) that can do this even in complex scenes with multiple shared objects. By combining global and local features, we also introduce a surface extension method to improve density of the resulting correspondence field. Finally, we extend our approach to find space-time correspondences.

We begin by describing how we can use B-Splines to improve correspondences for a single shared surface.

A. Single Surface Correspondence Approximation

A correspondence map is a 2D function $\mathbf{q} = f(\mathbf{p})$, $f : \mathbb{R}^2 \mapsto \mathbb{R}^2$, that maps each 2D-pixel coordinate $\mathbf{p}$

Algorithm 1 B-Spline Correspondences Algorithm

1 **Inputs:** Source image I_S and target image I_T
2 **Preprocessing:** Initialize correspondence map (f_o) and confidence map with NRDC
3 **Segmentation:** Oversegment source image I_S to superpixels $\mathbf{\Omega} = \{\Omega_1, \ldots, \Omega_k\}$
4 **Merging:** $\mathbf{\Omega} \leftarrow$ MergeAndFitSplines$(f_o, \mathbf{\Omega})$ [Alg. 2]
5 **Extension:** $f \leftarrow$ExtendCorrespondences$(\mathbf{\Omega})$ [Alg. 3]
6 **return** final correspondence map f between I_S and I_T

from the source image to its corresponding pixel coordinate $\mathbf{q}$ in the target image. Algorithms using nearest-neighbor patch-matching algorithms [12], [17] can compute reasonably robust, dense correspondence maps in the presence of deformable, non-rigid objects [1]. However, because they only consider features in a local area, their results often still have objectionable errors and noise in cases of extreme variation.

Patch-based correspondence algorithms usually fail because of two reasons. First, because of extreme changes or noise in the images, the patch with the closest distance might not be the correct one geometrically. We treat these correspondences as outliers (matches with large errors). Furthermore, the PatchMatch algorithm (which significantly accelerates the process) does not search through all the candidate patches and might stop before the best match is found. This situation leads to matches with small errors (correspondence noise). Because of these reasons, methods that rely only on patch-based correspondences are not as robust as necessary.

However, these methods could be used to compute a preliminary correspondence map f_o that could be refined later. Therefore, we propose to first compute an initial correspondence map with NRDC, and then apply geometric filtering to discard outliers and eliminate noise in order to produce a better correspondence field closer to the desired f. Our key observation is that the shared objects between two images typically have a smooth correspondence map, so we can approximate the mapping function f with a smooth, B-Spline approximation f_Ω:

$$\forall \mathbf{p} \in \Omega_i,\, f(\mathbf{p}) \approx f_{\Omega_i}(\mathbf{p}), \tag{1}$$

where Ω_i is the region of a single, shared smooth surface in the source image I_S, $f(\mathbf{p})$ is the unknown ground-truth correspondence map, and $f_{\Omega_i}(\mathbf{p})$ is the B-Spline function that approximates $f(\mathbf{p})$ well in region Ω_i. For now, our discussion assumes that there is only one shared surface Ω_i in both images, so it can be approximated with a single B-Spline. We will relax this assumption later in Sec. III-B. To represent f_{Ω_i}, we use the uniform, 2D cubic B-Spline:

$$f_S(x, y) = \sum_{i=0}^{3}\sum_{j=0}^{3} \mathbf{d}(i + \lfloor wx \rfloor,\, j + \lfloor wy \rfloor) N_i^3(\{wx\}) N_j^3(\{wy\}), \tag{2}$$

TABLE I
COMPARISON OF AVERAGE SQUARED ERROR (ASE) BETWEEN POLYNOMIAL AND B-SPLINE APPROXIMATIONS OF DIFFERENT DEGREES FOR THE SCENE IN FIG. 2. GROUND TRUTH CORRESPONDENCES WERE OBTAINED BY MANUALLY MODELING THE SCENE TO FIT THE IMAGES AND COMPUTING THE ACTUAL CORRESPONDENCES BETWEEN THEM

Order	2	3	4
Polynomial	6.10	5.91	5.62
B-Spline	5.32	4.58	4.53

where $f_S(x, y)$ is a 2D vector-valued function, $w = 1/30$ is the inverse distance between adjacent control points of the B-Spline in pixels, $\mathbf{d}(i, j)$ is the 2D position of control point (i, j), $N_i^3(u)$ represents the i-th basis function of order 3 for the uniform cubic B-Spline, and $\{\cdot\}$ is the fractional operator that takes the fraction of a real number (e.g., $\{3.14\} = 0.14$). Note that because the region of Ω_i in the source image may be irregular, we define x and y as the horizontal and vertical offsets of pixel $\mathbf{p}$ from the top-left pixel of Ω_i's bounding box.

The idea is to compute f_{Ω_i} by fitting a smooth B-Spline of the form f_S to the initial correspondence map f_o computed with NRDC. We do this by minimizing sum of square difference between f_S and f_o:

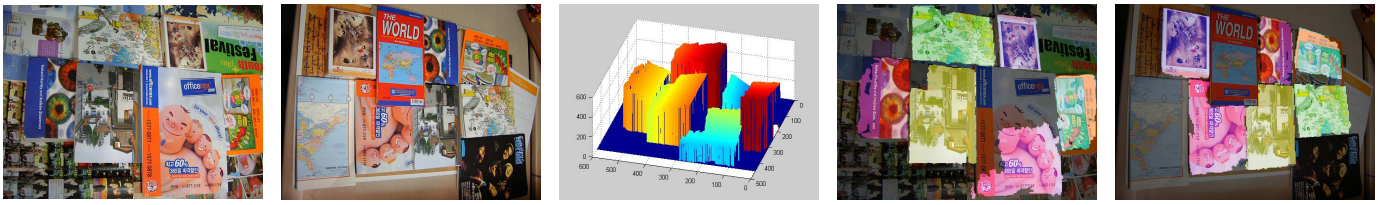
$$f_{\Omega_i} = \arg\min_{f_S} \sum_{(x,y)\in\Omega_i} \|f_S(x, y) - f_o(x, y)\|^2. \tag{3}$$

This linear equation takes $O(|\Omega_i|)$ to solve. Once f_{Ω_i} has been calculated, it acts as our enhanced correspondence map, where outliers and noise have been eliminated because the B-Spline approximation acts as a geometric filtering step. Fig. 1 shows the efficacy of B-Spline approximation in smoothing the noisy correspondences. Here, the green line maps to the blue line using initial correspondence map from NRDC. After our B-Spline approximation process, however, the green line maps to the red one which is smoother and more accurate.

As an aside, we could have considered using a polynomial curve instead of a smooth B-Spline to approximate the correspondence field. We experimented with this initially, but found that it did not work as well. Table I shows the numerical comparison of average squared error (ASE) between our cubic B-Spline and a polynomial approximation of different degrees for the scene in Fig. 2. We see that even a second-degree B-Spline works better than a fourth-degree polynomial. Furthermore, since the cubic B-Spline behaves almost as well as quartic B-Spline, we adopt it as our geometric filter. For reference, we also compared against RANSAC with a homography (the planar assumption), which performs poorly on curved objects like the cup and resulted in a much higher ASE of 795.

B. Unsupervised Segmentation for Multiple Shared Surfaces

Although a B-Spline can successfully approximate the correspondence map for a single shared smooth surface in two images, it cannot do so for two or more different surfaces, something that is common in real scenes. For example,



(a) (b) (c) (d) (e)

Fig. 3. Finding correspondences using our method and its application for object recognition. **(a)** Source image. **(b)** Target image. **(c)** B-Spline approximation for the correspondence field computed by our method (the value of the function plotted is the x coordinate of the pixel in the target image corresponding to a given pixel in the source image). One application of our work is to recognize shared content in both the **(d)** source and **(e)** target images.

in Fig. 2(a) we see that the cup has two surfaces (the body and the lid) which need to be approximated by two different B-Spline surfaces instead of only one.

To handle these situations, we need to automatically split (or segment) the image plane into several regions, each representing the surface of a different shared object. In this case, for example, our algorithm uses two B-Spline surfaces to approximate the cup and three more to approximate the background. Fig. 3 shows another example of a set of books cluttering a desk. In this case, there are six shared surfaces for books, so we need six B-Splines to denote the correspondence field (shown in Fig. 3(c)). Our algorithm automatically segments the images into several regions (see Fig. 3(d)), where each color indicates a different shared surface.

Once the surface has been segmented into regions $\mathbf{\Omega}_i$, we can then compute the surface approximation by solving the minimization in Eq. 3 in each separate region to determine precise correspondence maps $f_{\mathbf{\Omega}_i}$. We now must focus on the algorithm for the automatic segmentation of the image into smooth, coherent surfaces, which is not a trivial problem.

Since image segmentation is an important problem in image processing and computer vision, many approaches have been developed over the years. Some are based on extracting contours in the image [20], but it is often difficult to obtain closed boundaries which leads to incorrect segmentation. Others use thresholding based on image histograms [21], but it is not easy to choose the appropriate threshold. Algorithms based on region-growing methods [22] start from a seed and add points on the boundary that satisfy some criterion of homogeneity. Unfortunately, these methods do not apply to images with complex textures. Chen *et al.* [23] and Chan and Vasconcelos [24] consider segmentation using texture information and generate good results. Their methods need temporal information, however, and are hard to apply to pixelwise correspondences.

We observe that the correspondences in each segment should be smooth, so the segmentation process should somehow try to preserve this smoothness. Although segmentation algorithms based on smoothness have been proposed (e.g., [25]), they have been applied to segment surfaces from point cloud data based on curvature estimation, which does not permit much noise in the input points. In our case, the initial correspondences of the surfaces have considerable noise, so these methods lose their efficacy. Therefore, we need to develop our own algorithm for segmentation.

Our algorithm is based on the observation that if a group of segments belong to a single smooth surface, they can be approximated by a single B-Spline. Based on this observation, we can convert the problem of segmentation into the problem of merging small, pre-defined segments. Therefore, we first segment images into superpixels that are fairly small and then only merge those that are well-approximated by a single B-Spline. We now discuss each of the two steps in turn.

1) Segmentation Step: First, we require an algorithm to break up the source image into superpixels $\mathbf{\Omega}_i$. To do this, we use the method of Ren and Malik [26], results of which are shown in Fig. 2(b). Their content-based method effectively segregates different object surfaces, where each superpixel belongs to only one surface. In our experiments, every super-pixel contains at least 50 pixels and one source image is usually segmented into 1000 non-overlapping superpixels. This ensures that each resulting superpixel contains enough pixels to make each $|\mathbf{\Omega}_i|$ reasonably large for accurate B-Spline approximation.

After applying this method to segment the image, we next approximate each superpixel with a B-Spline surface as described in Sec. III-A. However, superpixels with NRDC confidence value less than 0.8 are deemed unreliable and are discarded. Furthermore, the initial correspondence map may also have outliers which make their corresponding superpixels poorly approximated by B-Spline surfaces. These superpixels are discarded if the distance between the B-Spline approximation and the NRDC correspondence map is greater than some fixed distance, using a distance calculation similar to that in Eq. 4. Once these unreliable superpixels have been discarded, we are ready to begin the merging process.

2) Merging Step: In the next step, we merge superpixels that have similar B-Spline approximations together to form larger coherent regions. In order to accelerate the merging process, we use an iterative greedy algorithm outlined in Alg. 2. At every iteration, we only merge regions that are adjacent to one another if they can be approximated well by a single B-Spline. If no such pair of regions can be found, the merging step terminates. Fig. 4 illustrates the overall merging process by showing selected pairs of regions at different iterations.

In order to decide which pair of regions should be selected to merge at any moment in time, we abstract the image regions and their adjacency relationships as a graph. Here, regions are denoted by vertices and adjacent regions are connected by edges. The merging operation effectively collapses adjacent vertices and their edge into a single vertex.

Algorithm 2 Merge And Fit Splines($f_o, \Omega = \{\Omega_1, \ldots, \Omega_k\}$)

for *every superpixel region* $\Omega_i \in \mathbf{\Omega}$ **do**
  $f_{\Omega_i} \leftarrow$ B-Spline approximation for Ω_i [Eq. 3]
  if B-Spline approximation error too big **or** NRDC confidence not high **then**
    Remove superpixel Ω_i from $\mathbf{\Omega}$
  end
end
Initialize adjacency graph for $\mathbf{\Omega}$ to make every pair of adjacent regions possible candidates for merging
while *adjacent regions in* $\mathbf{\Omega}$ *remain possible candidates for merging* **do**
  for *every pair of regions* $\Omega_u, \Omega_v \in \mathbf{\Omega}$ *that could be merged* **do**
    $\Omega \leftarrow \Omega_u \cup \Omega_v$
    $f_\Omega \leftarrow$ B-Spline Approximation for Ω [Eq. 3]
    if $D(\Omega, \Omega_U, \Omega_V) < d$ [Eq. 4] **then**
      $\Omega_i \leftarrow \Omega$, $f_{\Omega_i} \leftarrow f_\Omega$
    end
    Delete/add regions in $\mathbf{\Omega}$, update adjacency graph
  end
end
return $\mathbf{\Omega} = \{\Omega_1, \ldots, \Omega_n\}$

(a) **(b)** **(c)**

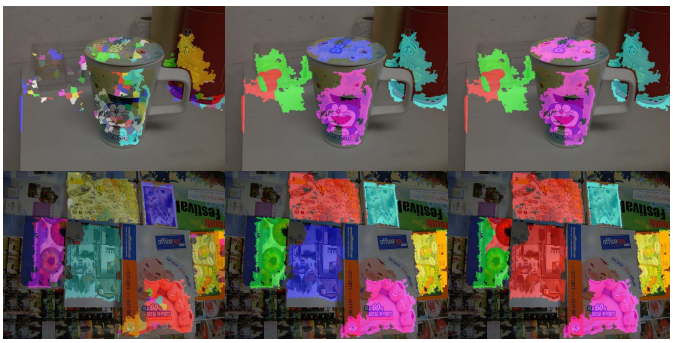
Fig. 4. Merging process through different iterations. The edges of the regions are shown in blue, while the red lines represent edges of the graph. Green lines represent the selected pairs of regions to be merged. **(a)** Iteration 1. **(b)** Iteration 4. **(c)** Iteration 7.

At the beginning, we initialize the graph so that all adjacent superpixels have edges to each other so that they are eligible to be merged together.

In each iteration, we select a pair of adjacent regions (call them Ω_U and Ω_V). Note that two regions are considered adjacent if the minimal distance between pairs of pixels in these regions equals 1. We then calculate the B-Spline approximation for the union of the two by setting $\Omega = \Omega_U \cup \Omega_V$ when minimizing Eq. 3, and measure the difference $D(\Omega, \Omega_U, \Omega_V)$ between the correspondences with this new joint B-Spline and the original correspondences from the individual B-Splines. Formally, we can write the calculation for $D(\Omega, \Omega_U, \Omega_V)$ as:

$$D(\Omega, \Omega_U, \Omega_V) = \frac{1}{|\Omega|}\Big[\sum_{\mathbf{p}\in\Omega_U} \|I_T(f_\Omega(\mathbf{p})) - I_T(f_{\Omega_U}(\mathbf{p}))\|^2 + \sum_{\mathbf{p}\in\Omega_V} \|I_T(f_\Omega(\mathbf{p})) - I_T(f_{\Omega_V}(\mathbf{p}))\|^2\Big], \tag{4}$$

where $I_T(\mathbf{q})$ represents value of pixel $\mathbf{q}$ in the target image. Note that this distance is calculated by comparing the pxiel values (and not the flows directly) because in the end we only care that the pixel values of the correspondences match, not that the actual offset vectors are the same.



(a) **(b)** **(c)**

Fig. 5. Segmentation results with different values for d, the threshold below which we merge two different regions. When $d = 3$ we see that the divisions within a single surface are preserved, while when $d = 30$ we see that multiple surfaces are merged into one, both of which are undesirable effects. In our experiments, we found $d = 10$ gave the best results. **(a)** $d = 3$. **(b)** $d = 10$. **(c)** $d = 30$.

If our distance $D(\Omega, \Omega_U, \Omega_V)$ is less than some constant d, we merge the regions by combining the vertices in the graph and updating the correspondence map. Otherwise, we delete the edge between them so that we do not try to merge them again. We found that setting the threshold $d = 10$ achieved reasonably good segmentation results, as shown in Fig. 5. Note that unconnected pixels in Fig. 5(c) may be merged together, because their respective regions are adjacent.

Note that since the same sets of super-pixels are covered by the final B-Splines, the order in which we merge the regions does not affect the result much. Therefore, we simply merge these superpixels in an arbitrary order based on our graph traversal algorithm. When the edge set is empty, no more regions can be merged and we have fit B-Splines to all regions Ω_i. Note, however, that the regions containing valid B-Spline correspondence approximations do not necessarily cover all pixels in the source image I_S since we have discarded pixels with low confidence values. Since it is important to have correspondences in these regions as well, we extend the surfaces with an algorithm described in the next section.

C. Correspondence Map Extension

We now describe a novel way to extend the merged correspondence field f_Ω into one that is larger but still reliable, as described in Alg. 3. Our idea is that since we have computed B-Spline surfaces in the merging step, we can use them as global information to further extend our correspondence field. Specifically, we observe that a good extension should satisfy three rules:

1) Correspondences in regions of high confidence (where we have fit a good B-Spline approximation f_{Ω_i}) should remain unchanged.
2) Low-confidence pixels immediately outside the high-confidence regions should have a continuous correspondence field with their high-confidence neighbors.

Algorithm 3 Extend Correspondences($\Omega = \{\Omega_1, \ldots, \Omega_n\}$)

forall the $\Omega_i \in \Omega$ **do**
  repeat
    modified $\leftarrow$ false
    Create Ω_i' area within 5 pixels of boundary of Ω_i that is not covered by another region
    forall the $\mathbf{p} \in \Omega_i'$ **do**
      Find nearest pixel $\mathbf{q} \in \Omega_i$
      Compute temp correspondence $\widehat{f_{\Omega_i}}(\mathbf{p})$ [Eq. 5]
      Approximate patch transformation $\mathbf{J_q}$ [Eq. 6]
      Search $\mathbf{M}(\theta, s, \Delta x, \Delta y)$ for correspondence of $\mathbf{p}$ with smallest patch error E [Eq. 7]
      if $E < 0.1$ **then**
        $\Omega_i \leftarrow \Omega_i \cup \mathbf{p}$
        $f_{\Omega_i}(\mathbf{p}) \leftarrow$ searched correspondence of $\mathbf{p}$
        modified $\leftarrow$ true
      end
    end
    if *modified* **then**
      $f_{\Omega_i} \leftarrow$ B-Spline Approximation for Ω_i [Eq. 8]
    end
  until *not modified*
end
$f_\mu \leftarrow f_{\Omega_1} \cup f_{\Omega_2} \cup \ldots \cup f_{\Omega_n}$
return f_μ

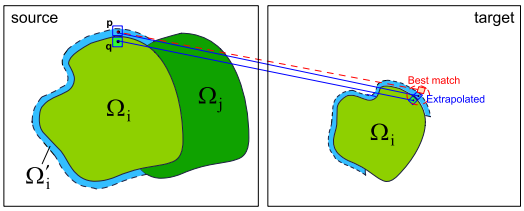


Fig. 6. After merging, several contiguous regions with robust correspondences have been identified, shown here as Ω_i and Ω_j. The purpose of the extension step is to propagate these correspondences into neighboring regions. We first identify the extension region Ω_i', and for every pixel $\mathbf{p} \in \Omega_i'$ we find the nearest pixel $\mathbf{q} \in \Omega$. We then linearly extrapolate the correspondence of $\mathbf{q}$ to $\mathbf{p}$ to compute a preliminary correspondence $\widehat{f_{\Omega_i}}$. Since this correspondence may not be correct, we search patches in a neighborhood around this correspondence to find the closest patch. If a good match is found, $\mathbf{p}$ will be added to Ω_i, else it is discarded. After processing all the pixels in Ω_i in this manner, we fit a new approximate B-spline to the new Ω_i and the algorithm repeats again.

3) Pixels in regions with low confidence should belong to patches that also have similar correspondences.

Our algorithm iteratively adds more pixels to the high-confidence regions, fits new B-Splines to them, and then tries to add more pixels in the next iteration. We begin by describing what happens in the first iteration, referring to Fig. 6 for illustration.

For every surface Ω_i found after the merging step, we identify pixels $\mathbf{p}$ in source image I_S that lie in a small region surrounding Ω_i that are not in any of the other regions $\mathbf{p} \notin \bigcup_{j=1}^{n} \Omega_j$. This region, called Ω_i', is where we want to extend the correspondence map to enlarge it. We then identify the pixel $\mathbf{q}$ that is nearest to $\mathbf{p}$ but still inside the original Ω_i. Note that the region to be extended Ω_i' is small enough that $\|\mathbf{p} - \mathbf{q}\| \leq 5$.

Since $\mathbf{q} \in \Omega_i$, it has a highly-confident correspondence map that is well-approximated by a B-Spline $f_{\Omega_i}(\mathbf{q})$. Because there should be some coherency between the correspondences of $\mathbf{q}$ and $\mathbf{p}$, our idea is to linearly extrapolate the valid correspondence field from $\mathbf{q}$ to $\mathbf{p}$ (leveraging rule #2), but then adjust it to fix problems by doing a local patch-based search (leveraging rule #3). Specifically, we can linearly extrapolate the correspondence field at $\mathbf{p}$ using the B-Spline approximation at $\mathbf{q}$ in the following way:

$$\widehat{f_{\Omega_i}}(\mathbf{p}) = f_{\Omega_i}(\mathbf{q}) + \nabla f_{\Omega_i}(\mathbf{q}) \cdot (\mathbf{p} - \mathbf{q}), \tag{5}$$

where the hat over $\widehat{f_{\Omega_i}}$ indicates that it is the temporary correspondence map because the linear extrapolation is not likely to produce accurate correspondences.

To find a more accurate correspondence, we must search around the region indicated by the linear extrapolation to find the patch in the target that is the most similar to the source patch. To do this, the 7×7 patch of pixels around $\mathbf{p}$ in the source (call it $\mathbf{P}_S(\mathbf{p})$) must be transformed to the potential corresponding patch in the target image I_T (call it $\mathbf{P}_T(\mathbf{p})$), but we need to know more than just the offset from the correspondence map (given by $\mathbf{t} = \widehat{f_{\Omega_i}}(\mathbf{p}) - \mathbf{p}$) since patches could rotate, scale, or deform when they map from I_S to I_T. Fortunately, we can approximate this deformation by computing the Jacobian of f_{Ω_i} at pixel q:

$$\mathbf{J_q} = \begin{bmatrix} \frac{\partial f_{\Omega_{i,1}}(\mathbf{q})}{\partial x} & \frac{\partial f_{\Omega_{i,1}}(\mathbf{q})}{\partial y} \\ \frac{\partial f_{\Omega_{i,2}}(\mathbf{q})}{\partial x} & \frac{\partial f_{\Omega_{i,2}}(\mathbf{q})}{\partial y} \end{bmatrix} \tag{6}$$

where $f_{\Omega_{i,1}}$ and $f_{\Omega_{i,2}}$ refer to the first and second components, respectively, of the vector-valued function f_{Ω_i} which is a B-Spline of the form in Eq. 2. The resulting $\mathbf{J_q}$ is a 2×2 matrix that encodes the rotation, scaling, and other deformations (approximately) induced locally to the patch of pixels around $\mathbf{q}$ by the correspondence map $f_{\Omega_i}(\mathbf{q})$.

Since we assume coherency in the correspondence map from $\mathbf{q}$ to $\mathbf{p}$, we can use the same transformation for the patch $\mathbf{P}_S(\mathbf{p})$ around $\mathbf{p}$ to map it to target I_T. To search the neighborhood of patches around this transformation, we allow the patch transformation to have slight variations in translation, rotation, and scale, similar to what was done for Generalized PatchMatch [17]. Specifically, we compute a new offset vector $\mathbf{t}'(\Delta x, \Delta y) = \mathbf{t} + [\Delta x, \Delta y]^T$, where $\Delta x, \Delta y \in \{-1, 0, 1\}$ which means that the new patch will be within a one-pixel offset of the original linearly-extrapolated correspondence $\widehat{f_{\Omega_i}}$. In a similar way, we compute a new deformation matrix $\mathbf{J}'(\theta, s) = \mathbf{R}(\theta) \cdot \mathbf{S}(s) \cdot \mathbf{J_q}$ in which the original transformation $\mathbf{J_q}$ is first multiplied by 2×2 scaling matrix $\mathbf{S}(s)$ (where $s \in \{0.9, 1.0, 1.1\}$), and then by 2×2 rotation matrix $\mathbf{R}(\theta)$ (where $\theta \in \{-5°, 0, 5°\}$). The new patch translation $\mathbf{t}'$ and deformation $\mathbf{J}'$ can then be combined into

a single 3×3 transformation matrix $\mathbf{M}$:

$$\mathbf{M}(\theta, s, \Delta x, \Delta y) = \begin{bmatrix} \mathbf{J}'(\theta, s) & \mathbf{t}'(\Delta x, \Delta y) \\ \mathbf{0}^T & 1 \end{bmatrix}$$

We can then apply transformation matrix $\mathbf{M}$ to the patch of pixels around $\mathbf{p}$ to get our potential target patch $\mathbf{P}_T(\mathbf{p})$:

$$\mathbf{P}_T(\mathbf{p}, \theta, s, \Delta x, \Delta y) = I_T(\mathbf{M}(\theta, s, \Delta x, \Delta y) \cdot \mathcal{N}(\mathbf{p})),$$

where $\mathcal{N}(\mathbf{p})$ outputs the coordinates of pixels in the 7×7 patch around $\mathbf{p}$ in homogeneous coordinates (a 3×7^2 matrix) which are then multiplied by the 3×3 matrix $\mathbf{M}$ which warps them and adds an offset. The resulting 3×7^2 matrix is used to sample target image I_T with bilinear interpolation to produce the potential corresponding target patch $\mathbf{P}_T(\mathbf{p})$. We must search through all the potential patches to find the one that gives us the minimum error with respect to the source patch, given by $\mathbf{P}_S(\mathbf{p}) = I_S(\mathcal{N}(\mathbf{p}))$, by minimizing:

$$E(\mathbf{P}_S, \mathbf{P}_T) = \min_{\theta, s, \Delta x, \Delta y} \|\mathbf{P}_S^*(\mathbf{p}) - \mathbf{P}_T^*(\mathbf{p}, \theta, s, \Delta x, \Delta y)\|^2 \quad (7)$$

where the * superscripts for the patches indicate that we are using LAB color space when computing differences and standardized them by subtracting the mean of the patch and dividing by its standard deviation, i.e., $\mathbf{P}^* = (\mathbf{P} - \overline{\mathbf{P}})/\sigma(\mathbf{P})$, which helps account for changes in lighting and tone mapping.

If the error $E(\mathbf{P}_S, \mathbf{P}_T)$ in Eq. 7 is greater than a certain threshold (set to 0.1 in our experiments), a correspondence for pixel $\mathbf{p}$ has not been properly found and so $\mathbf{p}$ is removed from the extension region Ω_i'. If the error is below this threshold, then the offset discovered during this search is added to the pixel coordinate and used as the new correspondence map at this pixel:

$$f_{\Omega_i}(\mathbf{p}) = \mathbf{t} + [\Delta x, \Delta y]^T + \mathbf{p},$$

where $\Delta x, \Delta y$ are the offset deltas that resulted in the minimum error in Eq. 7. This process is then repeated for all pixels in the extension region Ω_i'. Once this process is finished, only pixels with good correspondences will remain in Ω_i' since the others have been removed.

Note that although pixels may not be connected with the original region Ω_i, we still accept them without the requirement of pixel adjacency by approximating the entire unconnected region with a single B-Spline surface. In this way, these unconnected pixels will not degenerate the approximation. Finally, to complete the iteration, the original region is combined with its extension $\Omega_i \leftarrow \Omega_i \cup \Omega_i'$ and a B-Spline approximation f_{Ω_i} is computed to fit to the entire new region Ω_i as shown in Eq. 8.

$$f_{\Omega_i} = \arg\min_{f_S} \sum_{(x,y) \in \Omega_i} \|f_S(x, y) - f_{\Omega_i}(x, y)\|^2. \quad (8)$$

Note this is similar to the minimization we did earlier in Eq. 3, except that in this case we fit to the previous B-Spline approximations, not the original correspondences from NRDC. The iterative process then repeats again and a new extension region is identified. This extension process continues until no new pixels are added to any of the current surfaces Ω_i.

Once all n regions have been extended, they are combined together into a single correspondence map $f \leftarrow f_{\Omega_1} \cup f_{\Omega_2} \cup \ldots \cup f_{\Omega_n}$ for output. Here, each f_{Ω_n} is a correspondence map approximated by a single B-Spline for surface Ω_i. This final correspondence map f can be used for recognizing objects in common between the two images, mapping a texture from one image to another, or other applications.

D. Space-Time Correspondence

Our algorithm is not limited to finding correspondences between two static images. We further extend it to compute space-time consecutive correspondences between a source frame I_S and a sequence of target frames I_{T_i}, commonly presented as a video. The principle is similar as before: since motions of objects are often smooth over time, the correspondence map should also be smooth over time and can be well-approximated by a spline function. We first initialize correspondence map $f_o(\mathbf{p}, t)$ and confidence map $c(\mathbf{p}, t)$ between I_S and I_{T_i} using NRDC. We then detect shared surfaces Ω_t in the source image and approximate correspondences by fitting a spline to the initial correspondence map.

To ensure space-time continuity, we use the Trivariate-Spline [5] to approximate correspondences for each detected surface in the source image:

$$f_S(x, y, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} \mathbf{d}_{i+\lfloor wx \rfloor, j+\lfloor wy \rfloor, k+\lfloor lt \rfloor} \cdot N_i^3(\{wx\}) N_j^3(\{wy\}) N_k^3(\{lt\}) \quad (9)$$

where $l = 1/15$ is the inverse temporal distance between adjacent control points in frames, $w = 1/30$ is the inverse spatial distance between control points (as in Eq. 2), and $\mathbf{d}_{i,j,k}$ are the positions of the control points. We then fit the spline by solving a standard least squares problem:

$$f_{\Omega_i}(\mathbf{p}, t) = \arg\min_{f_S} \sum_{(\mathbf{p} \in \Omega_i)} \|f_S(\mathbf{p}, t) - f_o(\mathbf{p}, t)\|^2. \quad (10)$$

Although $f_{\Omega_i}(\mathbf{p}, t)$ is $\mathbb{R}^3 \mapsto \mathbb{R}^3$, note that the initial correspondences $f_o(\mathbf{p}, t)$ are computed frame by frame. Therefore, $f_o(\mathbf{p}, t)$ will preserve the time value and only affect the x and y offsets in $\mathbf{p}$. This ensures that the first two dimensions of $f_{\Omega_i}(\mathbf{p}, t)$, which are the correspondences of $\mathbf{p}$ at frame t, can be solved independently of t. To handle multiple shared surfaces, we use our split-and-merge method and approximate each surface with Trivariate-Spline.

IV. Evaluation

We implemented the algorithm presented in Sec. III using combination of C++ and MATLAB. For a 640×480 pixel image, our implementation takes about 4 to 7 seconds to initialize a correspondence map and an extra 5 seconds to do surface approximation and merging on a 2.9GHz Intel Core i7(2640M) Dell laptop. We test our approach on several datasets and compare our results with previous algorithms.

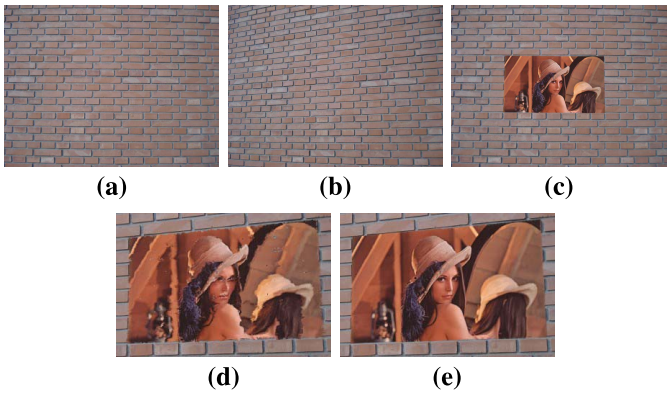


Fig. 7. Mapping a texture from the source to the target using our correspondence field. Given a pair of **(a)** source and **(b)** target images showing a wall from different perspectives, we manually added a texture (in this case the "Lena" image) to the wall in the source and used the correspondence field from different algorithms to resample the source to automatically re-project the texture onto the target image. While the result from **(d)** NRDC has objectionable artifacts because of noise and error in the correspondence field, our B-Spline approximation produces a smoother correspondence field that is free from visible artifacts **(e)**. **(a)** source. **(b)** target. **(c)** source + texture. **(d)** NRDC result. **(e)** Our result.

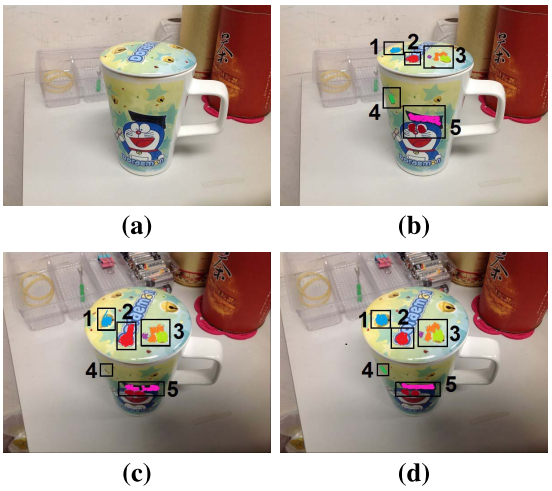


Fig. 8. Mapping user strokes from a source to a target, with the strokes labeled by numbers. Although NRDC maps strokes 1 and 3 fairly reasonably, it does a poorer job for strokes 2 and 5, and loses stroke 4 altogether. On the other hand, our algorithm produces a more robust correspondence that is able to map the user's edits to the target image in a reliable manner. **(a)** source. **(b)** source + strokes. **(c)** NRDC result. **(d)** Our result.

A. Continuity

To test continuity of our correspondence map, we manually added a texture (i.e., a 2D bitmap) to a source image and viewed the surface from another angle in the target image, using our correspondence map to resample the texture from the source to the target. We also tried to do this using NRDC [1] by itself, since it is the most successful previous method for finding dense and reliable correspondences. As we can see in the comparison in Fig. 7, the result obtained with our correspondence map is smooth and undistorted because our B-Spline approximation is able to eliminate noise and outliers, while the result produced by NRDC has visible artifacts.

Our surface approximation can also handle curved surfaces which can be more challenging. In Fig. 8, we map a user's color strokes on a cup from the source image to the target. In this case, NRDC's correspondence map distorts patterns 2 and 5 and is simply not big enough to cover pattern 4 on the cup's body. On the other hand, our algorithm automatically splits the body and cover of the cup into two surfaces (see correspondence map shown in Fig. 2) and performs surface approximation on each of them. This helps us eliminate noise and outliers and map patterns 2 and 5 without distortion. Furthermore, our surface extension algorithm enlarges the region containing valid correspondence fields, so we can map pattern 4 correctly to the cup in the target image.

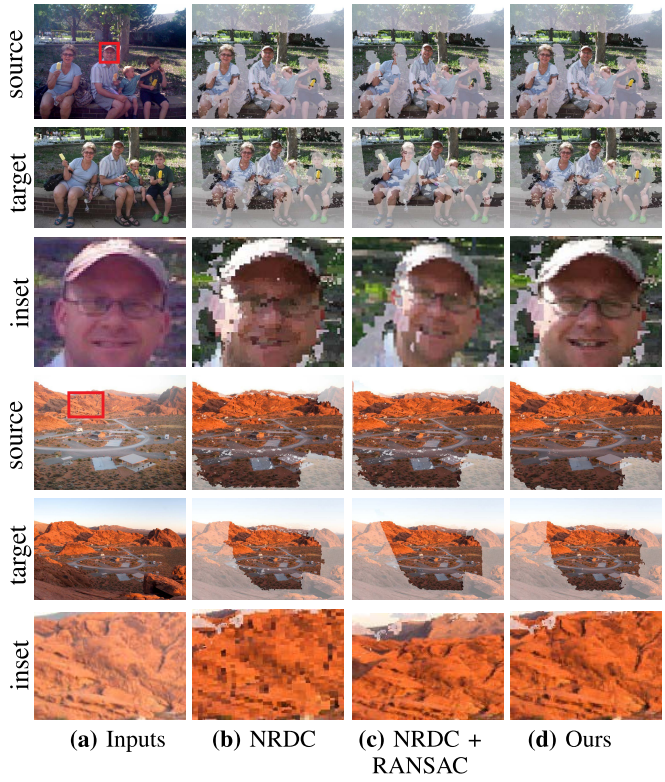


Fig. 9. Performance on real world scenes, taken from HaCohen *et al.* [1]. Here, we compute the correspondences from the source to target using various algorithms (NRDC, NRDC + RANSAC, and our own), and then map pixels from the target back to the source using the correspondences to visualize their quality. The first and fourth rows show the source images remapped with pixels from the target using the correspondence map, with exception of the first column which is simply the original source (the highlighted regions in each image are those with robust correspondence maps). The second and fifth rows show the target images with the matching region shown, while the third and sixth rows show a magnified inset from the remapped source. By comparing the results with the **(a)** original source images, we can tell if the correspondence maps are accurate and smooth. In both cases, we see that **(b)** NRDC's correspondences are not accurate and that the remapped image has lost important texture detail. Furthermore, trying to use **(c)** RANSAC to eliminate NRDC's outliers results in serious distortion of the scene. **(d)** Our method, on the other hand, produces an accurate and continuous correspondence map of a reasonably large region.

B. Real Scenes With Non-Rigid Objects

The proposed algorithm also works well for natural scenes with shared, non-rigid objects. In Fig. 9, we directly map pixels from the source to the target by computing correspondences with NRDC only, with NRDC combined with RANSAC in an attempt to get rid of outliers, and with

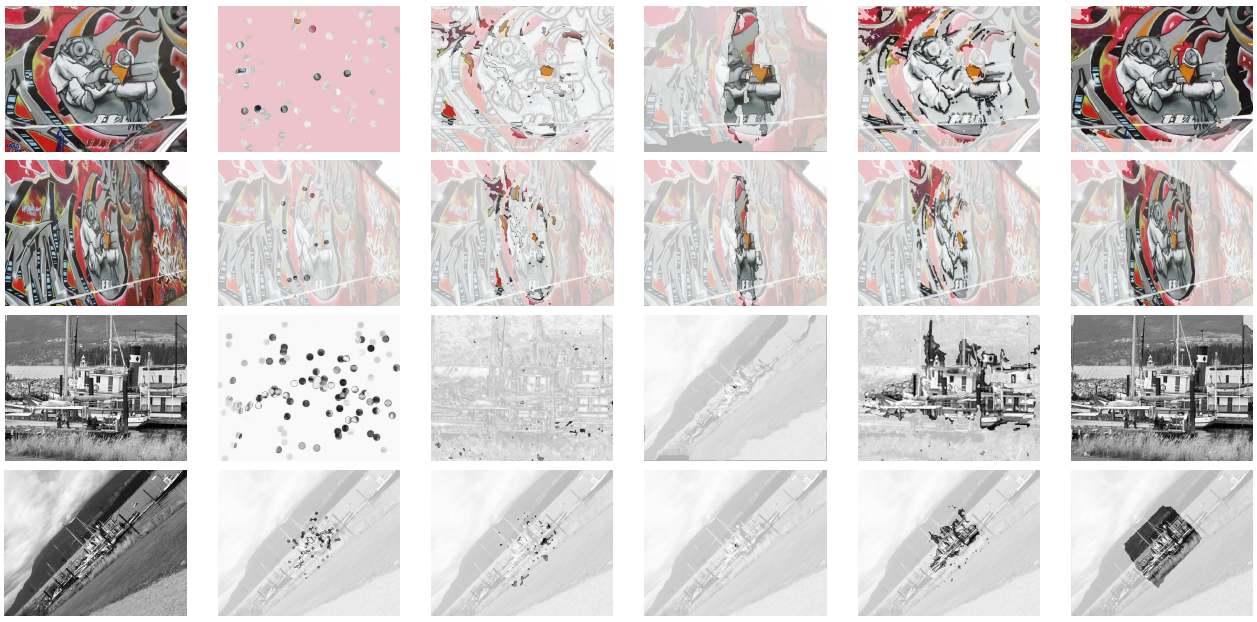


Fig. 10. Two examples from the dataset by Mikolajczyk *et al.* [27] (in each the top row is the source, the bottom is the target). The first example exhibits significant change in viewpoint, while the second features a strong difference in rotation and scale. We compare the correspondence matches found by a variety of algorithms: **(b)** sparse SIFT features, **(c)** Generalized PatchMatch, **(d)** SIFT-Flow, **(e)** Non-Rigid Dense Correspondence, and **(f)** our algorithm. We highlight regions where errors less than or equal to 15 pixels. We can see that our algorithm is able to produce larger, more contiguous regions with high-quality correspondences than the other state-of-the-art approaches. **(a)** Inputs. **(b)** SIFT. **(c)** GPM. **(d)** SIFT-Flow. **(e)** NRDC. **(f)** Ours.

our algorithm. In rows 3 and 6 of Fig. 9(b), NRDC alone produces correspondences that are not entirely accurate resulting in the loss of details. Objects mapped by NRDC+RANSAC are seriously distorted, as shown in rows 3 and 6 of Fig. 9, which are zoomed-in views of the insets in rows 1 and 4, respectively. In row 3 of Fig. 9(c), the head of the man is deformed, while in row 6 of Fig. 9(c), the mountain is lower than it should be. Our method is able to reproduce the target image from the source more faithfully because it finds better correspondences.

C. Density and Accuracy

There are many state-of-the-art methods that can generate reliable correspondences. To demonstrate the density and accuracy of our approach, we compare our results with some of the top dense correspondence methods in the literature: Generalized PatchMatch (GPM) [17], SIFT-Flow [8], and NRDC [1], as well as the original sparse SIFT correspondence algorithm [3] for completeness.

Fig. 10 shows two examples from the dataset by Mikolajczyk *et al.* [27]. The first pair of images are taken with a significant change in viewpoint, and the second pair show big difference in rotation and scale. In these cases, other methods are not able to find dense correspondence maps, because the variations from one image to another are too great. However, our method is able to find good correspondences in a region of reasonable size.

We also adopt the metric proposed by Liu *et al.* [8] to measure accuracy and density of the correspondence field. Here, the correspondence at a pixel is considered correct if its distance in pixels from the ground truth location is

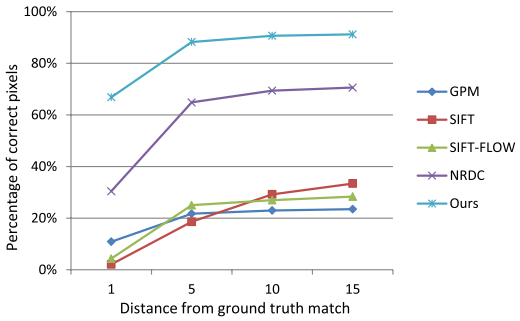


Fig. 11. Comparison between our method and other approaches on the dataset by Mikolajczyk *et al.* [27]. The horizontal axis represents the size of threshold r, the distance from the ground truth correspondence, and the vertical axis represents the percentage of correct correspondences for each r.

less than threshold r. In Fig. 11, we plot the proportion of correct matches with respect to r while comparing state-of-the-art methods against our own for the entire dataset of Mikolajczyk *et al.* [27]. In this case, SIFT's result is too sparse and not effective for finding dense correspondences. SIFT-Flow cannot handle large scale and rotation differences. While GPM is robust to differences in scale and rotation, its nearest neighbor patch feature is not enough to find reliable matches. NRDC generates the best result of the previous methods, but still fails in the case of extreme scale and rotation such as those shown earlier.

On the other hand, our method is able to find a higher percentage of correct correspondences at all values of r. Specifically, our result finds 66.9% correct correspondences on average with error less than 1 pixel, while for the other

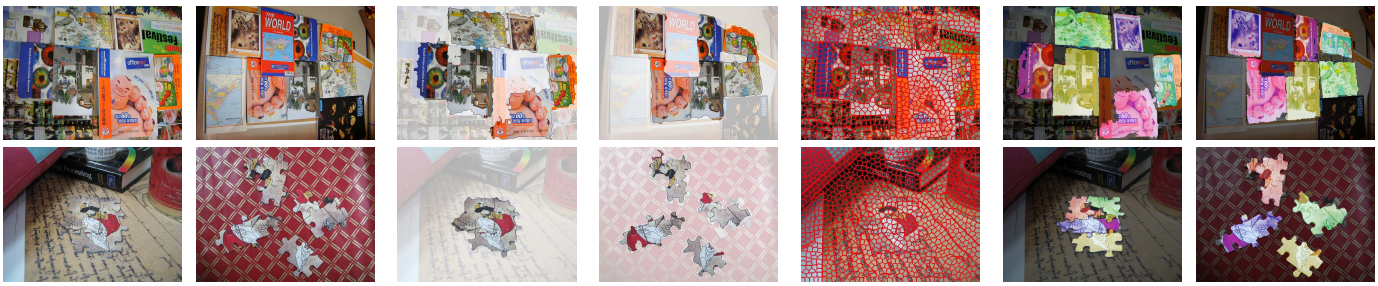


Fig. 12. Results of unsupervised co-recognition using our algorithm. The first line shows the BOOKS example, and the second line shows the JIGSAW example. **(a)** Source images. **(b)** Reference images. **(c)** Source images highlighted with NRDC correspondences. **(d)** Reference images highlighted with NRDC correspondences. **(e)** Source image segmentation. **(f)** Objects detected in source images by our algorithm. **(g)** Objects detected in reference images by our algorithm. Different objects are masked with different colors. Our method recognizes most parts of shared objects accurately.

algorithms the percentage of correspondences found with this degree of accuracy ranges from 2.1% to 30.4%. In terms of density, our method generates larger correspondence field, finding more than 90% correspondences with errors less than 15 pixels, while the other methods find 30.4% to 70.6% correct correspondences with this accuracy.

V. APPLICATIONS

We demonstrate the advantages of our method by applying it to two important problems in image processing and computer vision: (i) unsupervised recognition and segmentation of multiple common objects, and (ii) editing of textures in video sequences. We shall talk about each application in turn.

A. Unsupervised Recognition and Segmentation of Multiple Common Objects

Our method can be used to recognize objects in common between two images. To do this, we first select one image as the source and the other as the target. We then calculate correspondences between the two images and approximate them with B-Spline surfaces as described in Sec. III. Because our algorithm only merges regions that have a smooth correspondence field together, after the algorithm is finished each region covered by a different B-Spline surface can be considered a different shared object.

To test our approach, we use the challenging dataset from Cho *et al.* [9], where the scenes are cluttered and objects are occluded in complex ways. Fig. 12 presents our recognition results. Given source image (a) and reference image (b), we calculate NRDC correspondence map (c, d) and segment the source image (e). We merge the segments and finally detect different objects with dense correspondences (f, g).

These results can be evaluated by the hit-ratio h_r and background ratio b_r metrics[1] defined by Cho *et al.* [9]. Essentially, h_r denotes fraction of the area that is correctly matched for shared objects, so higher h_r means that more correct correspondences are detected. Analogously, b_r denotes fraction of wrong correspondences (i.e., false positives), so lower b_r means a method with higher accuracy.

[1] $h_r = \frac{|\text{GroundTruth} \cap \text{Result}|}{|\text{GroundTruth}|}, \quad b_r = \frac{|\text{Result}| - |\text{Result} \cap \text{GroundTruth}|}{|\text{Result}|}$

TABLE II

NUMERICAL COMPARISON BETWEEN CO-RECOGNITION [9], NRDC [1], AND OUR ALGORITHM. HIGHER h_r AND LOWER b_r INDICATE BETTER RESULTS. WITH THE EXCEPTION OF THE FIRST TWO SCENES (SEEN IN FIG. 12), ALL OTHER SCENES ARE IN THE SUPPLEMENTARY MATERIAL

		Cho	NRDC	Ours
BOOKS	h_r	94.6%	88.9%	94.9%
	b_r	11.8%	3.31%	4.67%
JIGSAW	h_r	80.0%	69.7%	90.1%
	b_r	22.8%	0.93%	3.10%
MINNIES	h_r	83.2%	61.9%	75.1%
	b_r	37.4%	4.28%	3.83%
MICKEYS	h_r	80.7%	48.1%	74.5%
	b_r	20.6%	6.43%	4.25%
TOYS	h_r	83.5%	66.1%	81.4%
	b_r	25.2%	5.14%	4.21%
BULLETINS	h_r	91.2%	52.2%	86.1%
	b_r	16.8%	7.99%	5.01%
Average	h_r	85.5%	64.5%	83.7%
	b_r	22.4%	4.68%	4.19%

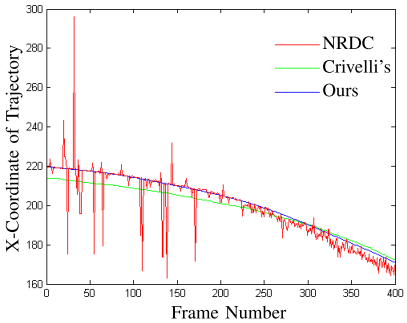


Fig. 13. Trajectory found by Crivelli *et al.*'s algorithm [6], initial correspondences from NRDC [1], and the result of our Trivariate-Spline approximation. Errors in Crivelli *et al.*'s result are large, while the trajectory from NRDC is perhaps more accurate but very noisy (high variance). Our method eliminates these errors and generates a trajectory that is both accurate and precise.

To test our algorithm against other methods, we performed a numerical comparison of our results with those of the Co-recognition work of Cho *et al.* [9] and NRDC [1], shown in Table II. In these cases, Cho *et al.* [9] find large correspondences field, but their method does not scale well to dense pixel-to-pixel correspondences on large images. Therefore, their results have a large number of false positives

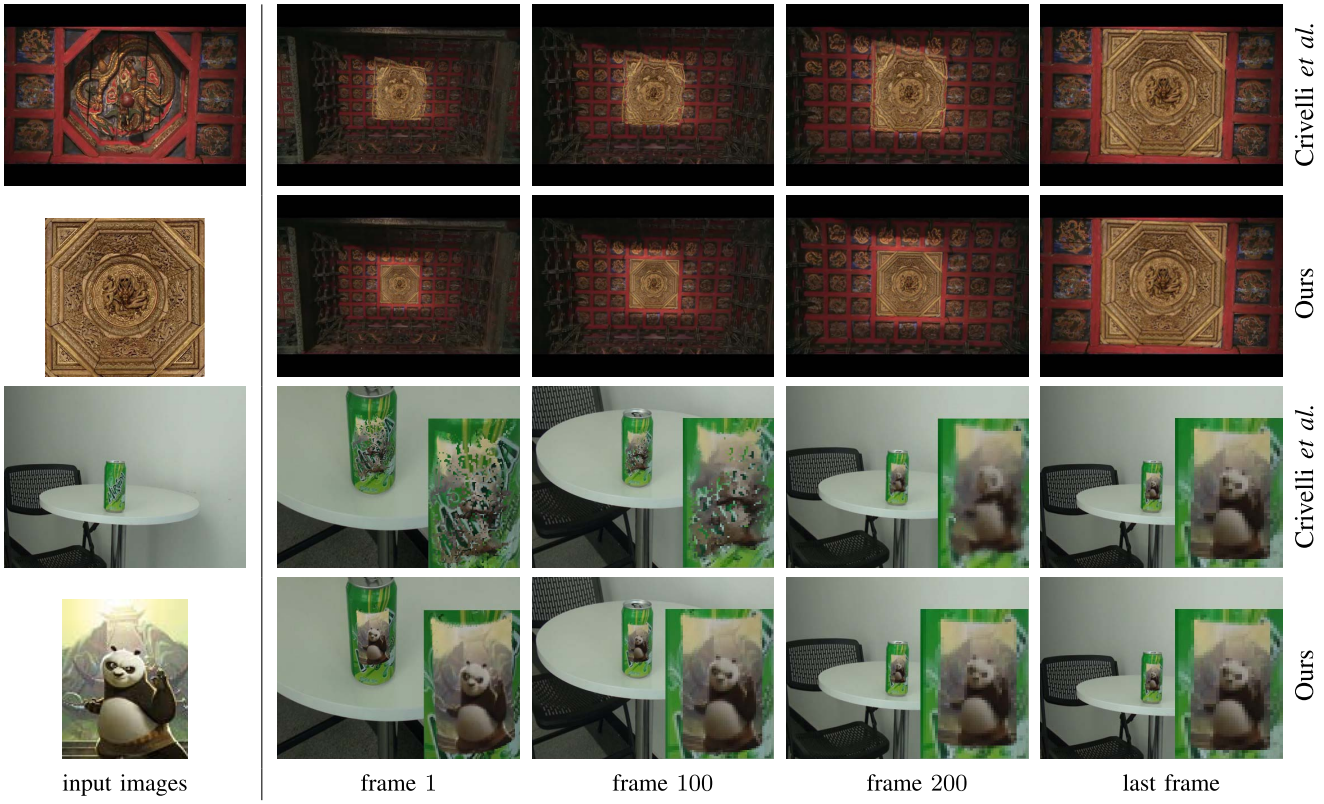


Fig. 14. Comparison between Crivelli *et al.*'s method [6] and our approach for modifying textures in a video sequence. The leftmost column shows (from top to bottom) the original frame of the video, and the bitmap to be added to the surface, respectively for the two scenes. The next four columns show selected frames of the video sequence. Last frames are resulting frames with the bitmaps in place. The first and third rows show the results of Crivelli *et al.*'s method, and the second and fourth rows are generated using our approach. Note how our approach is able to produce plausible mappings whereas the other method breaks down when the change in viewpoint is too great. Please refer to supplemental video.

($b_r = 22.4\%$ on average) and are therefore inaccurate. On the other hand, NRDC is more selective and is able to achieve a high amount of accuracy ($b_r = 4.68\%$ on average). However, its correspondence field is not big enough to cover the entire shared surface ($h_r = 64.5\%$ on average), and it is not able to segment different objects from its correspondence map. Our algorithm generates a large correspondence field with high precision ($h_r = 83.7\%$, $b_r = 4.18\%$).

B. Video Texture Editing

When editing a video, a user may want to add new elements such as pictures on the walls, modify the textures on surfaces, or add lighting effects. Not only is it very time-consuming to edit each individual frame of the video, but it is not easy to produce a natural result in this manner since our eyes are particularly sensitive to temporal inconsistencies between frames. In this section, we discuss the application of our correspondence algorithm for video texture editing, a technique that allows the user to edit just one frame of the video and propagates the modifications to the entire video automatically.

As discussed earlier, there has been limited work on this difficult problem. The state-of-the-art is the approach of Crivelli *et al.* [6], which uses optical-flow to find trajectories with space-time continuity across many frames. They do this by calculating correspondences between adjacent frames and using a novel method to compute motion trajectories from these correspondences. However, this approach of propagating texture edits will lose accuracy in the long term, because correspondence errors will be gradually amplified. Furthermore, this approach has the limitation that it restricts the frame that can be edited (the source frame) to the last frame of the video.

By contrast, our method is able to robustly compute correspondences between the source frame and all the other frames in the video because our Trivariate-Spline approximation (see section III-D) ensures continuity in both space and time. Furthermore, the user can edit any frame of the video as the source frame and still produce dense, robust correspondence maps that accurately propagate the user's edits to the entire video. However, because Crivelli *et al.*'s algorithm is considered state-of-the-art, we implemented it and compared against it in our experiments.

Fig. 13 shows the trajectory of one pixel in the source frame for the first example in Fig. 14, plotted over all the frames of the sequence as generated by the different methods. Since the last frame of the sequence is chosen as the source image, Crivelli *et al.*'s method is more accurate towards the end of the video but less so at the beginning, as can be seen by the divergence of the green line from the others at small frame numbers. Furthermore, the NRDC is too noisy and not usable for texture-mapping. Our Trivariate-Spline approximation can eliminate these errors and generate an accurate trajectory.

Fig. 14 shows two scenes where one of the surfaces has been manually modified by adding a bitmap to it. In the

Fig. 15. Example of our new application of video light mapping. The first row shows selected frames from original video. In this application the user chooses one frame (marked with red rectangle) to adjust the lighting. Here, the illumination gain is adjusted to give the appearance of a spotlight (see visualization in the second row). Only one frame has to be edited, and the new lighting is then automatically propagated by our algorithm to the entire video to produce a new relit result, shown in the third row. Please refer to supplemental video.

first case, Crivelli *et al.*'s method suffers from the change in scale between the source and target images, causing distortion of the bitmap pattern particularly in the early frames of the video sequence. In the second case, their optical-flow suffers from significant change in viewpoint as well as by the change in scale, also resulting in artifacts. By contrast, our algorithm is able to produce reliable correspondences and produce smooth, natural-looking results. Our Trivariate-Spline approximation can also handle curved objects with rotation, zoom, and other complex motions, as shown in Fig. 14 (row 6). Readers are encouraged to look at the supplemental video to see the temporal behavior of the two algorithms more clearly.

In our last application of video editing, we introduce a new kind of editing approach we call *light mapping*, which enables the user to relight the scene by modifying the lighting in the source frame. As shown in Fig. 15, the user can edit the lighting in one frame of the video and we automatically propagate the lighting effects to all other frames. Such an approach could have applications in consumer video editing.

VI. Conclusions and Future Work

We have presented a new algorithm for finding robust, dense correspondences between two images with fairly large differences in viewpoint, lighting, or deformation, as well as between a single video frame and the rest of the video sequence. The robustness of the proposed method comes from the underlying B-Spline approximation that reduces noise and eliminates outliers in the correspondence field. When coupled with an unsupervised segmentation algorithm to detect the different surfaces in the scene, plus a correspondence extension to grow the regions with valid correspondences, our algorithm is able to produce correspondence maps that are better than those from state-of-the-art approaches. Since finding reliable correspondences is critical for many applications in image processing and computer vision, our algorithm could be used to improve a variety of applications. To demonstrate this, we have shown improvement over prior work in applications such as shared object recognition and video texture editing.

Our approach can be further improved in two ways. First, patch-based algorithms have difficulty finding reliable matches in large, smooth regions, such as clear sky [1], [14]. Since we utilize patch-based techniques, we may generate unreliable matches in such cases. This could be improved by either modifying the PatchMatch search to include a smooth gradient term (as in the work of image melding [14]), or by collecting more matching candidates in the extension step and selecting the one that best fits the B-Spline. Second, the B-Spline approximation requires a minimum amount of data for an accurate fit. This means that the correspondences of a small object in the source image might be treated as outliers and discarded by the algorithm. Although this did not turn out to be a serious problem in our experiments, it would be interesting to study techniques to solve the problem.

Finally, our method could be applied to other applications, such as video segmentation [28] or stereo reconstruction, to name a couple. We will release our code upon publication so that others can build upon this approach and use it for other applications that require robust correspondences.

References

[1] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Trans. Graph.*, vol. 30, no. 4, Jul. 2011, Art. ID 70.

[2] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th IJCAI*, 1981, pp. 674–679.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[4] C. de Boor, *A Practical Guide to Splines*, vol. 27. New York, NY, USA: Springer-Verlag, 1978.

[5] G. Awanou and M.-J. Lai, "Trivariate spline approximations of 3D Navier–Stokes equations," *Math. Comput.*, vol. 74, no. 250, pp. 585–601, 2005.

[6] T. Crivelli, P.-H. Conze, P. Robert, and P. Pérez, "From optical flow to dense long term correspondences," in *Proc. 19th IEEE Int. Conf. Image Process. (ICIP)*, Sep./Oct. 2012, pp. 61–64.

[7] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 41–48.

[8] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "SIFT flow: Dense correspondence across different scenes," in *Proc. 10th Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 28–42.

[9] M. Cho, Y. M. Shin, and K. M. Lee, "Co-recognition of image pairs by data-driven Monte Carlo image exploration," in *Proc. 10th Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 144–157.

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proc. 28th ACM SIGGRAPH*, 2001, pp. 327–340.

[11] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, Nov. 1998.

[12] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, Jul. 2009, Art. ID 24.

[13] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz, "Regenerative morphing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San-Francisco, CA, USA, Jun. 2010, pp. 615–622.

[14] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," *ACM Trans. Graph.*, vol. 31, no. 4, Jul. 2012, Art. ID 82.

[15] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman, "Robust patch-based HDR reconstruction of dynamic scenes," *ACM Trans. Graph.*, vol. 31, no. 6, Nov. 2012, Art. ID 203.

[16] N. K. Kalantari, E. Shechtman, C. Barnes, S. Darabi, D. B. Goldman, and P. Sen, "Patch-based high dynamic range video," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013, Art. ID 202.

[17] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized PatchMatch correspondence algorithm," in *Proc. 11th Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 29–43.

[18] N. K. Kalantari, E. Shechtman, S. Darabi, D. B. Goldman, and P. Sen, "Improving patch-based synthesis by learning patch masks," in *Proc. IEEE Int. Conf. Comput. Photogr. (ICCP)*, May 2014, pp. 1–8.

[19] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon, "Unwrap mosaics: A new representation for video editing," *ACM Trans. Graph.*, vol. 27, no. 3, Aug. 2008, Art. ID 17.

[20] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *Int. J. Comput. Vis.*, vol. 43, no. 1, pp. 7–27, 2001.

[21] O. J. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1457–1465, Dec. 2002.

[22] M. Mancas, B. Gosselin, and B. Macq, "Segmentation using a region-growing thresholding," *Proc. SPIE*, vol. 5672, pp. 388–398, Mar. 2005.

[23] J. Chen, G. Zhao, M. Salo, E. Rahtu, and M. Pietikainen, "Automatic dynamic texture segmentation using local descriptors and optical flow," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 326–339, Jan. 2013.

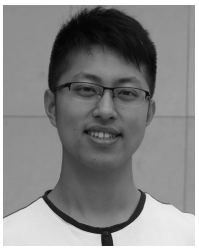
[24] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 909–926, May 2008.

[25] T. Rabbani, F. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *Int. Arch. Photogram., Remote Sens. Spatial Inf. Sci.*, vol. 36, no. 5, pp. 248–253, 2006.

[26] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. 9th IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2003, pp. 10–17.

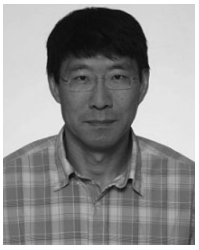
[27] K. Mikolajczyk *et al.*, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, nos. 1–2, pp. 43–72, 2005.

[28] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Proc. 11th Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 282–295.

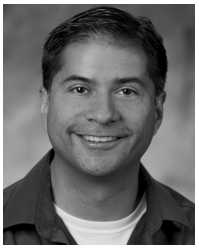


Jingwei Huang received the bachelor's degree from the School of Software, Tsinghua University, Beijing, China. He will pursue the Ph.D. degree with the Department of Computer Science, Stanford University, USA. His research interests include physically based simulation, image processing, and video editing.

Bin Wang received the B.Sc. degree in chemistry in 1999, and the Ph.D. degree in computer science from Tsinghua University, in 2005. He was a Research Assistant with the Department of Computer Science, Hong Kong University, and had post-doctoral research training with the ISA/ALICE Research Group, INRIA-LORIA, France. He is currently an Associate Professor with the School of Software, Tsinghua University, China. His research interests include photorealistic rendering, nonphotorealistic rendering, and image and video processing.



Wenping Wang received the B.Sc. and M.Eng. degrees in computer science from Shandong University, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Alberta, Canada, in 1992. He is currently a Professor of Computer Science with The University of Hong Kong, China. His research interests include computer graphics, geometric computing, and computational geometry.



Pradeep Sen received the B.S. degree from Purdue University, and the M.S. and Ph.D. degrees from Stanford University. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara. He has co-authored over 30 technical publications, including eight *SIG-GRAPH/ACM Transactions on Graphics publications*. His research interests include algorithms for image synthesis, computational image processing, and computational photography. He has received over U.S. $2.2 million in funding, including the 2009 NSF CAREER Award.